

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ НАВЧАЛЬНОЇ СИСТЕМИ
З ДИСЦИПЛІН STEM
SOFTWARE FOR TUTORING SYSTEM FOR STEM DISCIPLINES

Виконав(ла): студент(ка) 4 курсу, групи КНТ-129

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ФІЛІППЕНКОВ Д.Ю.

(ПРИЗВИЩЕ та ініціали)

Керівник СЕРДЮК С.М.

(ПРИЗВИЩЕ та ініціали)

Рецензент ПОЛЯКОВ М.О.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Факультет ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 121 Інженерія програмного забезпечення
 (код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
 “ ” 2023 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ФІЛІППЕНКОВА Данііла Юрійовича
 (ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення навчальної системи з дисциплін STEM. Software for Tutoring System for STEM disciplines

керівник проєкту (роботи) к.т.н., доцент, СЕРДЮК Сергій Микитович,
 (науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 04 ” квітня 2022 року № 87

2. Строк подання студентом проєкту (роботи) 01 червня 2022 року
 3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Розробка структури програмного забезпечення. 3. Основні рішення щодо реалізації компонентів системи. 4. Керівництво програміста. 5. Керівництво оператора.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1–5 Основна частина	СЕРДЮК С.М., доцент		
Нормоконтроль	ДЕЙНЕГА Л.Ю., ст.викладач		

7. Дата видачі завдання « 04 » квітня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3–4 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)


(підпис)

Данііл ФІЛІППЕНКОВ
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис)

Сергій СЕРДЮК
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до випускної кваліфікаційної роботи бакалавра: 214 с., 8 табл., 58 рис., 2 дод., 20 джерел.

STEM, НАВЧАННЯ, НАВЧАЛЬНА ПЛАТФОРМА, ГЕЙМІФІКАЦІЯ, ХІМІЯ, UNITY, ASP.NET CORE, RIDER.

Об'єкт дослідження – процес навчання STEM дисциплінам.

Предмет дослідження – методи розробки інтелектуальних навчальних систем у сфері STEM освіти.

Мета роботи – розробка програмного забезпечення навчальної системи з дисциплін STEM.

Матеріали, методи та технічні засоби: мова програмування C#, середовище розробки Rider, середовище розробки UnityEditor, Microsoft SQL Server, персональний комп'ютер з процесором Intel Core i7 на базі операційної системи Windows 10.

Результати. Проведено аналіз предметної області та досліджено існуюче аналогічне програмне забезпечення. Розроблено програмне забезпечення, яке являє собою інтерактивну навчальну платформу для полегшення вивчення STEM дисциплін. Здійснено опис прийнятих рішень, описані реалізовані класи та архітектура отриманого програмного застосунку.

Галузь використання – освіта.

ABSTRACT

Explanatory note to the bachelor's thesis: 214 pages, 8 tables, 58 figures, 2 appendixes, 20 sources.

STEM, LEARNING, TUTORING PLATFORM, GAMIFICATION, CHEMISTRY, UNITY, ASP.NET CORE, RIDER.

The object of the research is the process of teaching STEM disciplines.

The subject of the research is methods of developing intellectual educational systems in the field of STEM education.

The purpose of the work is the development of software for tutoring system for STEM disciplines.

Materials, methods and technical means: C# programming language; the Rider development environment, the UnityEditor development environment, Microsoft SQL Server, a personal computer with an Intel Core i7 processor based on the Windows 10 operating system.

Results. The analysis of the subject area was carried out and the existing similar software was investigated. Software has been developed, which is an interactive educational platform to facilitate the study of STEM disciplines. The adopted decisions are described, the implemented classes and the architecture of the resulting software application are described.

The field of use is education.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ	9
1 Аналіз предметної області	11
1.1 Проблеми STEM освіти	11
1.2 Гейміфікація як спосіб навчання STEM дисциплінам	12
1.3 Аналіз програмних аналогів	14
1.4 Розробка технічного завдання	23
2 Розробка структури програмного забезпечення	29
2.1 Вибір інструментарію розробки програмного забезпечення	29
2.1.1 Огляд особливостей обраних мов програмування і фреймворків	29
2.1.2 Огляд особливостей обраного середовища розробки	31
2.2 Опис модулів системи	34
2.3 Функціональна схема застосунку	36
3 Основні рішення щодо реалізації компонентів системи	40
3.1 Алгоритми функціонування розробленої програми	40
3.2 Вибір патерну проєктування	50
3.3 Опис структури класів системи	51
3.4 Проєктування бази даних	75
3.5 Основні рішення щодо розробки графічного інтерфейсу	79
4 Керівництво програміста	90
4.1 Призначення та умови застосування програми	90
4.2 Характеристики програми	90
4.3 Звертання до програми	94
4.4 Вхідні та вихідні дані	95
4.5 Опис API інтерфейсів	95
4.6 Повідомлення програмісту	97
5 Керівництво оператора	98
5.1 Призначення програми	98

5.2 Умови використання програми	98
5.3 Виконання програми	98
5.4 Повідомлення оператору.....	106
Висновки.....	107
Перелік джерел посилань.....	108
Додаток А Слайди презентації	111
Додаток Б Текст програми.....	120

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API – Application Programming Interface;

Backend – серверна частина додатку;

Frontend – клієнтська частина додатку;

HTTP – HyperText Transfer Protocol;

JSON – JavaScript Object Notation;

JWT – JSON Web Token;

ООП – об'єктно орієнтовне програмування;

ТЗ – технічне завдання.

ВСТУП

Ми живемо у час, коли наука і техніка розвиваються з неймовірною швидкістю. Інформаційні технології, штучний інтелект, робототехніка та біотехнології – це далеко не повний перелік надбань науки, які ми свідомо чи несвідомо використовуємо кожен день.

Одним з головних викликів у сучасному світі є підготовка молоді до життя в умовах цього швидкого розвитку. Знання і вміння у галузі STEM є дуже важливими, оскільки вони надають людям можливості розуміти та працювати з сучасними технологіями. STEM (Science, Technology, Engineering, Mathematics) – це сукупність дисциплін, які вивчають науку та технічні аспекти світу, зокрема фізику, хімію, біологію, інформатику, програмування, електроніку, механіку, інженерію та математику.

Освіта в галузі STEM забезпечує зростання економіки країни, створення нових робочих місць та змінює життя на краще. Однак, в сучасному світі не всі мають рівний доступ до знань STEM. Тому важливо робити все можливе, щоб забезпечити їх доступність.

В Україні в останні декілька років дуже поширена практика впровадження навчальної дисципліни STEM в освітні програми всіх типів шкіл. В нашій державі посилення ролі STEM–освіти є одним із пріоритетів модернізації освіти, складовою частиною політики країни з підвищення рівня конкурентоспроможності національної економіки та розвитку людського капіталу, одним з основних факторів інноваційної діяльності у сфері освіти, що відповідає запитам економіки та потребам суспільства.

В США STEM освіта є однією з найбільш популярних областей навчання. Вона надає студентам підготовку в галузях, пов'язаних з науковими дослідженнями, технологічними інноваціями, розробкою нових матеріалів, інженерією, комп'ютерними науками та багатьма іншими галузями, які вимагають аналітичного мислення та вирішення складних проблем. Багатьом американським студентам дуже складно дається навчання

STEM дисциплінам, тому університет Пенсильванії відкрив грант для створення програмного застосунку, який буде допомагати таким студентам легше засвоювати STEM.

Як видно створення інтерактивного навчального застосунку, який буде допомагати пізнавати світ STEM наук є дуже актуальною і достатньо пріоритетною задачею сьогодні. Тому метою випускної кваліфікаційної роботи бакалавра є створення програмного забезпечення навчальної системи з дисциплін STEM, яке буде повністю відповідати вимогам гранту виданого університетом Penn State, США.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Проблеми STEM освіти

Предметом дослідження даної роботи є методи розробки інтелектуальних навчальних систем у сфері STEM освіти. Метою - є створення інтерактивного навчального застосунку для вивчення STEM дисциплін, який буде допомагати учням шкіл і студентам опановувати ці дисципліни якнайшвидше і ефективніше.

Є ряд дуже важливих проблем, які повинна допомагати розв'язувати інтерактивна навчальна платформа. Недостатнє залучення учнів до навчального процесу та низький рівень мотивації – дуже серйозна перепона освітнього процесу сучасності. Наприклад, у навчальному процесі шкіл це може бути пов'язано зі складним та монотонним форматом уроків, недостатньою зацікавленістю учнів у темі або недостатнім знанням учителів про методи навчання. Всі люди різні і цей факт спричиняє багато складнощів коли справа доходить до освоєння нових знань. У навчальному процесі можуть бути учні з різним рівнем підготовки, але уроки зазвичай розраховані на середній рівень, що може бути складним для слабких або нудним для більш сильних учнів. Також іноді виникає проблема з доступністю до відповідних ресурсів та інформації, або з недостатньою кількістю практичних завдань та вправ, які допомагають закріпити знання та навички.

Сьогодні у сфері освіти активно використовують різноманітні інноваційні інформаційні технології. Широко відомі такі глобальні освітні онлайн платформи як Coursera (113 мільйонів користувачів), Open edX (40 мільйонів користувачів), Udacity (близько 20 мільйонів користувачів). Ці платформи неодноразово і жорстко критикувалися, а відсоток відсіву студентів з подібних навчальних платформ часто сягає 90% [1]. Основною причиною цього є проблеми взаємодії між системою та її користувачами [2], також є відсутність у даних системах елементів гейміфікації, за допомогою яких можна залучити студентів до вивчення складних STEM дисциплін.

Створення інтерактивної навчальної платформи з елементами гейміфікації для вивчення STEM дисциплін може допомогти вирішити ці проблеми, надаючи можливість учням навчатися у захоплюючому та цікавому форматі, індивідуалізувати навчання, забезпечити доступність до відповідних ресурсів інформації, а також забезпечити наявність достатньої кількості практичних завдань для закріплення навичок.

1.2 Гейміфікація як спосіб навчання STEM дисциплінам

STEM дисципліни, які включають науку, технологію, інженерію та математику, є ключовими компонентами сучасної економіки та суспільства. Проте навчання цих дисциплін може бути складним і нудним для багатьох студентів. Один із способів зробити навчання STEM більш захоплюючим та ефективним є гейміфікація.

Гейміфікація – це процес використання елементів гри в неігрових ситуаціях [3]. Це означає, що елементи гри, такі як бали, рівні, досягнення та лідерство, можуть бути використані в навчальних програмах.

Останнім часом гейміфікація стала більш популярною [4] та помітною у повсякденному житті людей [5]. За останні кілька років на ринку з'явилося багато гейміфікованих рішень [6], які були вдало інтегровані у велику кількість областей людського життя і діяльності [7]. Розробка ефективної гейміфікації – це простий прямий процес [8] і віднедавна велика кількість ігрових механік стала інтегруватися в повсякденну діяльність людини через різні типи програмних застосунків. Візьмемо для прикладу таку мобільну програму як FourSquare. FourSquare в сутності являє собою систему винагород, яка спонукає людей відкривати та обмінюватися інформацією про місця, які вони відвідують, отримуючи за це винагороди у вигляді значків [9].

Коли в розробці застосунків елементам гейміфікації приділяється достатня кількість уваги, такі елементи можуть значно покращити і пришвидшити процес навчання [10]. Багато дослідників вважають, що

гейміфікація може мотивувати та активувати цільову поведінку користувачів, одночасно підвищуючи їх лояльність до гейміфікованого досвіду [11]. Більше того такий досвід має дуже великий потенціал, щоб зробити неігрові дії більш захоплюючими, а також мотивувати людей виконувати завдання та підтримувати їх у стані “поток” (концентрації на навчальному процесі) [12].

Переваги гейміфікації зазначаються у низці досліджень. Гейміфікація позитивно впливає на залучення [13], мотивацію [14], участь [15], швидкість і якість навчання [16], оцінки за курс та відвідуваність лекцій, уроків і занять студентами і учнями абсолютно різних категорій [17].

Отже, гейміфікація може бути використана в широкому спектрі STEM дисциплін, а її використання може:

- допомогти залучити студентів до навчання, зробивши процес навчання більш захоплюючим і цікавим. Наприклад, використання балів та досягнень може стимулювати студентів до виконання завдань та досягнення успіху. Використання рівнів може дати студентам почуття прогресу та досягнення мети;
- допомогти студентам розвивати креативність та розв’язувати складні задачі. Граючись, студенти можуть використовувати свої навички та знання, щоб дійти до мети. Це може допомогти розвивати креативність та розв’язувати проблеми, що не мають очевидного рішення;
- допомогти залучити різні типи студентів до STEM дисциплін. Це може бути особливо корисним для тих студентів, які можуть відчувати себе відсталими або недостатньо підготовленими до вивчення STEM дисциплін. Гейміфікація може створити більш різноманітне та доступне середовище для навчання, що може збільшити мотивацію та інтерес студентів до STEM;
- покращити рівень знань студентів і учнів, яких до цього не можна було описати як “примірних” і “голодних до навчання”.

1.3 Аналіз програмних аналогів

Після отримання завдання до дипломного проєкту був досліджений та опрацьований ринок представлених сервісів та програм в даній предметній області. Як результат було виявлено велику кількість аналогів, кожен з яких має свої функції, недоліки та переваги.

Проаналізувавши вже існуючі аналоги, було обрано найбільш розповсюджені, схожі за функціоналом та змістом навчальні платформи: Coursera, KhanAcademy та PhET. Розглянемо їх більш детально.

1.3.1 Навчальна платформа Coursera

Coursera – це онлайн–платформа для навчання, яка пропонує безкоштовні та платні курси від провідних університетів та компаній по всьому світу, включаючи курси пов’язані з вивченням STEM дисциплін. Coursera дозволяє отримати доступ до великої кількості високоякісних навчальних ресурсів від провідних університетів та компаній, що допомагає зробити навчання більш доступним та зручним для широкої аудиторії.

Основні функції навчальної платформа Coursera:

- надання доступу до безкоштовних та платних онлайн–курсів з різних дисциплін;
- можливість взяти участь у курсі повністю віддалено, з використанням відеолекцій, завдань для самостійної роботи та інших ресурсів;
- надання можливості зареєструватися та брати участь у курсі, а також можливість отримати сертифікат про успішне закінчення курсу, який може бути використаний у резюме або під час зарахування на дипломну програму університету;
- надання можливості спілкування з іншими учасниками курсу та викладачами через форуми, чати та інші інструменти;

- можливість проходження курсу у власному темпі та на власний розсуд;
- можливість зберігати та переглядати матеріали курсу після завершення.

Скріншот прикладу роботи навчальної платформи Coursera наведено на рисунку 1.1.

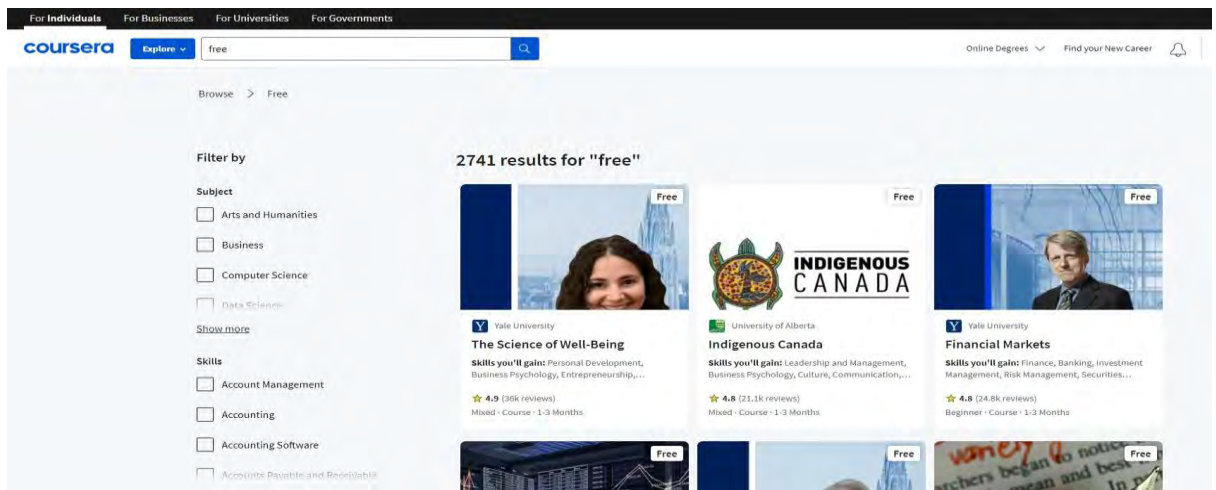


Рисунок 1.1 – Приклад роботи навчальної платформи Coursera [18]

До переваг Coursera слід віднести:

- доступність: Coursera доступна для будь-якого користувача з доступом до Інтернету. Це забезпечує широку доступність для навчання в будь-якому місці та часі;
- широкий вибір курсів: Coursera пропонує широкий вибір курсів з різних тематичних областей, включаючи науку, мистецтво, бізнес, технології та багато іншого;
- якість викладачів: Викладачі на Coursera – експерти у своїх галузях. Їхні курси забезпечуються якісними матеріалами, домашніми завданнями та підтримкою викладача;
- гнучкість: Навчання на Coursera дозволяє учням вивчати на своєму власному графіку та у своєму власному темпі;

- сертифікація: Учасники можуть отримати сертифікати після успішного завершення курсу, що може бути корисним у додатковій освіті та працевлаштуванні;

- функціональний мобільний застосунок.

Недоліками Coursera є:

- висока вартість платних курсів. Курси, які пропонуються на Coursera безкоштовні, проте вартість платних курсів може бути високою;

- відсутність особистого взаємодії. Курси на Coursera не завжди надають можливості прямого спілкування з викладачами або іншими студентами;

- низька якість мобільного застосунку. На офіційній сторінці застосунку у Google Play та AppStore дуже багато негативних відгуків, пов'язаних з нестабільною роботою платформи на мобільних пристроях;

- проблеми з відображенням платформи на мобільних пристроях з малими розмірами дисплеїв;

- перенасиченість інтерфейсу. Типова проблема для великих і багатофункціональних платформ;

- повністю відсутні елементи гейміфікації: отримання знань за допомогою платформи Coursera максимально “сухий” і монотонний процес, який дуже швидко втомляє;

- не всі курси доступні на українській мові;

- дуже велика кількість курсів. Тільки люди, які повністю знають, що їм треба вивчити зможуть самотійно побудувати собі шлях навчання і освоєння нового матеріалу, але у нашому світі не всі такі насправді.

1.3.2 Навчальна платформа KhanAcademy

Khan Academy – це безкоштовна навчальна платформа, яка надає доступ до онлайн–курсів та уроків з різних предметів, таких як математика, науки про природу, історія та інші.

Основна мета Khan Academy полягає у забезпеченні навчання для всіх, хто бажає навчитися чомусь новому, або удосконалити свої знання та навички в різних галузях. На сайті можна знайти більше 10 000 відеоуроків та завдань для самостійної роботи, а також додатки для смартфонів.

Основні функції Khan Academy включають наступне:

- безкоштовний доступ до курсів та уроків з різних предметів;
- навчання в режимі онлайн, з використанням відеоуроків, тестів та завдань для самостійної роботи;
- можливість навчання в будь-який час та в будь-якому місці;
- можливість персоналізації навчального процесу, шляхом вибору тем та складності завдань;
- можливість відстеження свого прогресу та отримання рекомендацій для подальшого навчання;
- наявність розділу для вчителів, який надає можливість створювати навчальні програми та відстежувати прогрес своїх учнів.

Скріншот прикладу роботи навчальної платформи Khan Academy наведено на рисунку 1.2.

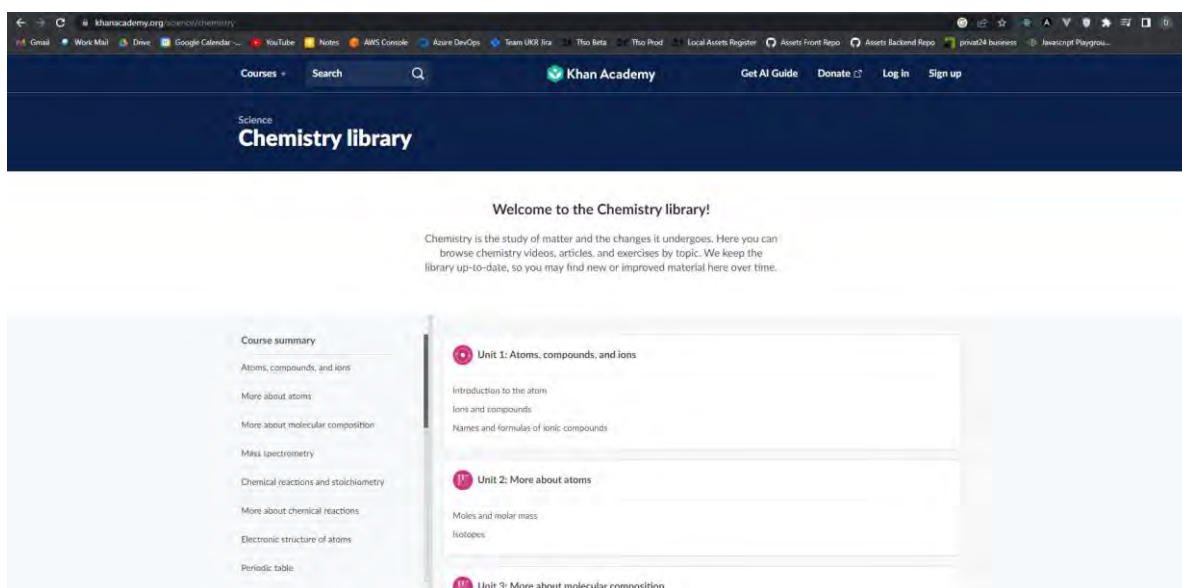


Рисунок 1.2 – Приклад роботи навчальної платформи Khan Academy [19]

Переваги Khan Academy:

- безкоштовний доступ;
- навчання в будь-який час та в будь-якому місці;
- персоналізація навчального процесу. Khan Academy надає можливість персоналізації навчання, шляхом вибору тем та складності завдань, що дозволяє користувачам навчатися відповідно до своїх потреб та інтересів;
- широкий вибір курсів та уроків;
- відстеження прогресу. Платформа дозволяє користувачам відстежувати свій прогрес та отримувати рекомендації для подальшого навчання;
- розділ для вчителів. Khan Academy надає розділ для вчителів, який дозволяє створювати навчальні програми та відстежувати прогрес своїх учнів;
- безпека та конфіденційність. Khan Academy дбає про безпеку та конфіденційність користувачів, забезпечуючи безпечний доступ до платформи та захищаючи персональні дані користувачів;
- інтеграція з Google Classroom;
- представлений функціональний мобільний застосунок навчальної платформи для Android і IOS.

Недоліки Khan Academy:

- відсутність інтерактивності. У порівнянні з іншими навчальними платформами, Khan Academy має менше інтерактивних елементів та можливостей для спілкування користувачів між собою та з викладачами;
- немає навіть натяку на гейміфікацію, або інші практики, які роблять процес навчання легшим і швидшим;
- недостатня глибина матеріалу. Хоча Khan Academy пропонує велику кількість уроків, деякі користувачі можуть вважати, що матеріал не деталізований або не досить глибокий для їхніх потреб;

- немає підтримки української мови;
- перенасиченість користувацького інтерфейсу платформи;
- більшість навчальних матеріалів у вигляді відео, що є проблемою для людей, які мають повільне інтернет з'єднання.

1.3.3 Навчальна платформа PhET Interactive Simulations

PhET Interactive Simulations – це безкоштовна навчальна платформа, що містить інтерактивні симуляції для навчання наукових дисциплін, зокрема фізики, хімії, біології та математики. Платформа пропонує понад 150 симуляцій, які можуть бути використані для демонстрації різних наукових концепцій. Симуляції можуть бути використані як для навчання, так і для відпрацювання умінь та навичок. Кожна симуляція містить відповідні підручники та посібники для вчителів, які містять інструкції щодо використання та навчання.

Основні функції платформи:

- надає доступ до близько двох сотень інтерактивних симуляцій;
- містить низьку корисних матеріалів до навчання учнів використовуючи ту чи іншу симуляцію.

Скріншот прикладу роботи PhET Interactive Simulations наведено на рисунку 1.3.

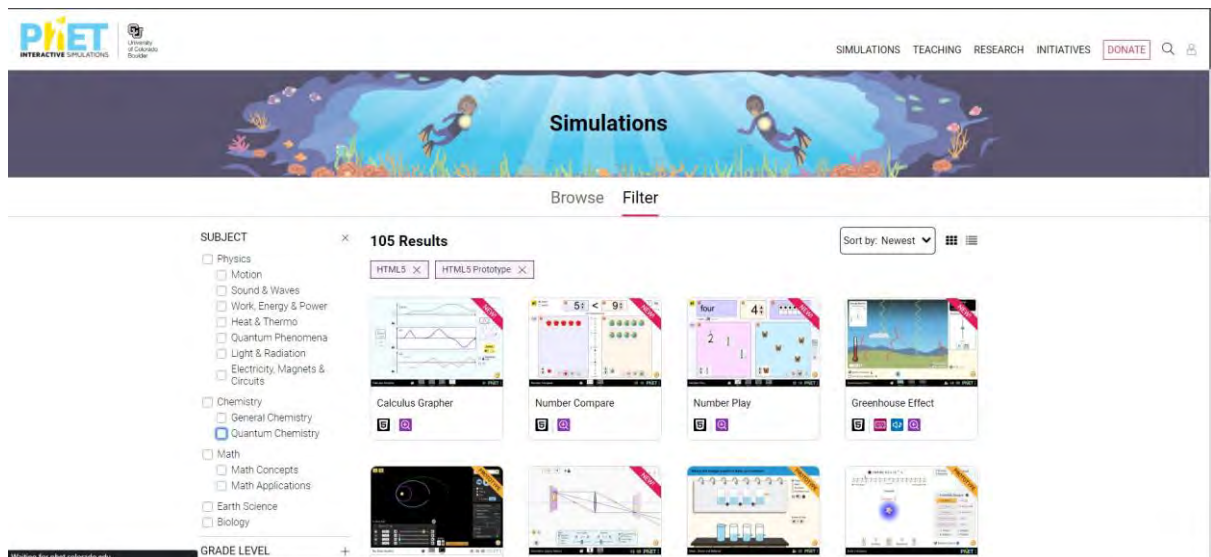


Рисунок 1.3 – Приклад роботи PhET Interactive Simulations [20]

Переваги PhET Interactive Simulations:

- безкоштовний доступ. PhET є безкоштовною платформою, яка доступна для всіх користувачів з Інтернет-підключенням. Немає потреби завантажувати жодного додаткового програмного забезпечення;
- відкритий код. Код PhET є відкритим, тобто кожен користувач може його змінювати та використовувати для своїх потреб;
- навчальні посібники. На сайті PhET доступні навчальні посібники для викладачів та студентів, які містять інформацію про використання симуляцій у навчальному процесі;
- мовна підтримка (у тому числі української мови). PhET доступна у більш ніж 90 мовах, що дозволяє користувачам з різних країн використовувати платформу на своїй рідній мові;
- різні рівні складності. PhET пропонує інтерактивні симуляції на різних рівнях складності, що дає змогу використовувати їх для різних груп користувачів з різними рівнями знань;
- можливість збереження результатів. Користувачі можуть зберігати результати своїх експериментів та спостережень для подальшого аналізу та порівняння;
- PhET працює на різних платформах та пристроях;

- прекрасний мобільний застосунок.

Недоліки PhET Interactive Simulations:

- вимоги до обладнання. Деякі симуляції можуть вимагати досить потужний комп'ютер та високошвидкісний Інтернет-з'єднання для коректної роботи, що може створювати труднощі для користувачів з менш потужним обладнанням;
- відсутність можливості індивідуалізації. PhET не надає можливості створювати індивідуальні завдання або керувати процесом навчання, що може бути недостатньо для деяких користувачів;
- відсутність перевірок знань і засвоєного матеріалу;
- недостатня кількість наукових матеріалів.

1.3.4 Загальне порівняння

Для порівняння всіх розглянутих аналогів було створено таблицю 1.1. В цій таблиці було порівняно: Coursera, Khan Academy та PhET Interactive Simulations за найбільш важливими характеристиками з точки зору зручності використання і освітньої компоненти.

Таблиця 1.1 – Порівняння аналогів

Характеристика	Coursera	Khan Academy	PhET Interactive Simulations
1	2	3	4
Навчальна компонента (наявність освітніх матеріалів)	+	+	+—
Можливість перевірок знань освітян	+	+	—
Безкоштовність	—	+	+
Зручний інтерфейс	—	—	+

Продовження таблиці 1.1

1	2	3	4
Можливість зручно налаштувати інтерфейс	+	–	–
Гейміфікація і інтерактивність	–	–	+
Персоналізація навчального процесу	–	+	–
Платформа у веббраузері	+	+	+
Наявність мобільного застосунку	+	+	+
Наявність desktop застосунку	–	–	–
Підтримка української мови	+–	–	+

Згідно з результатами порівняння в таблиці 1.1 можна побачити, що платформи Coursera та Khan Academy мають дуже великі переваги у кількості і якості навчального матеріалу, але абсолютно не містять і натяку на інтерактивність і “ігрові” елементи (гейміфікацію), що допомагають учням якнайшвидше засвоювати знання. З іншого боку PhET Interactive Simulations дає учням доступ до неперевершеної якості симуляцій, які дозволяють на практиці дізнаватися про деякі природні явища, описані наукою, взаємодіяти з ними і пізнавати їх, але навчальний матеріал дуже “бідний”

Таким чином, даної випускної кваліфікаційної роботи бакалавра стало поєднання позитивних сторін цих навчальних платформ і створення інтерактивного інтелектуального застосунку, який зробить процес навчання STEM дисциплінам максимально простим, цікавим і зручним.

1.4 Розробка технічного завдання

1.4.1 Підстава для розробки

Підставою для розробки служить грант “Teaching Basic Chemistry to Students at Risk” (Frank Ritter, Serhii Serdiuk, and Mary Jo Bojan) державного університету Пенсильванії, США, а також завдання на дипломну кваліфікаційну роботу бакалавра на тему "Програмне забезпечення навчальної системи з дисциплін STEM", затверджене наказом № 87 від 04 квітня 2023 р. по НУ «Запорізька політехніка».

1.4.2 Мета розробки та її призначення

Метою є розробка інтерактивного навчального застосунку для полегшення вивчення STEM дисциплін.

1.4.3 Функціональні вимоги до програмного продукту

Навчальний застосунок повинен бути розроблений з використанням елементи гейміфікації. Процес навчання можна організувати у вигляді гри, де ціль гравця – створити сучасний науковий центр університету.

Користувачі навчальної платформи повинні мати змогу:

- зареєструватися;
- увійти у свій акаунт;
- переглянути доступні для вивчення STEM дисципліни;
- переглянути свій прогрес по конкретній дисципліні;
- переглянути доступні навчальні етапи по дисципліні;
- переглянути свій прогрес по конкретному етапу дисципліни;
- переглянути рівні етапів;
- отримати статистику проходження різних рівнів;

- пройти рівень (передбачає виконання заданих завдань в обмежений проміжок часу);
- доступ до навчальних матеріалів;
- вдосконалити свої знання за допомогою інтерактивного навчального тренажеру.

Навчальна платформа повинна комунікувати з базою знань з хімії, написаній з використанням SWI-Prolog і генерувати питання для рівнів для контролю знань за допомогою даних, що в ній містяться. Повинна бути використана програмна реалізація бази знань з хімії, що створена іншим студентом нашої кафедри.

У навчальній платформі повинні обов'язково бути представлені дисципліна «Хімія» і навчальний етап «Метрики і аналіз розмірностей».

Питання для рівнів навчального етапу «Метрики і аналіз розмірностей» повинні бути згенеровані на основі шаблонів питань, які створила професор з хімії університету Penn State, Пенсильванія, США. Скріншот документу з питаннями наведений на рисунку 1.4.

Metric & dimensional analysis

Definitions: Level 0

What is the standard international unit of length? (m = meters)
can substitute g, s, mol, L, or J, for m (= meters) in items.

Levels 1-4 Basic levels

Level 1

Conversions from units on left to one meter

How many meters are in one kilometer?

Which is larger (smaller), one millimeter or one meter?

Level 2

Conversion from randomly selected units on left and right.

Level 2A

How many nanometers are in one picometer?

Which is smaller, one nanometer or one millimeter?

Level 2B

How many nanometers are in 4.5×10^4 picometer?

Which is smaller, one nanometer or 4.5×10^4 picometer?

Level 3

Conversion from randomly selected units on left and right using a random number of. The quantity

Which is smaller (larger), 1300 nanometers or 2.5 micrometer?

Which is smaller, 1.3×10^4 nanometers or 1.3×10^{-3} micrometer?

Use random number generator & appropriate limits to generate numbers in items.

Level 4

Conversion when units are squared or cubed.

Level 4A

How many mm^2 are in 100 cm^2 ?

How many mm^3 are in 100 cm^3 ?

Level 4B

Which is smaller (larger), 1300 mm^3 or 2.5 micro cm^3 ?

How many mm^2 are in $2.2 \times 10^2 \text{ cm}^2$?

How many mm^3 are in $2.2 \times 10^3 \text{ cm}^3$?

Рисунок 1.4 – Скріншот документу з шаблонами питань для навчального етапу «Метрики і аналіз розмірностей»

1.4.4 Вхідні і вихідні дані програми

Вхідними даними в програмі є персональні дані користувача, які він вказує при реєстрації, його відповіді в тестах рівній, а також інформація з спеціалізованої з певної дисципліни бази знань, що використовується для

генерування питань рівнів, а також навчальної інформації. Також вхідними даними в програмі є відповіді на запити, які надає база знань з хімії.

Вихідними даними є відповідно кількісно представлений прогрес користувача з конкретної дисципліни і досягнення з навчальних етапів, а також згенеровані завдання для навчальних рівнів і корисні матеріали для опанування STEM.

Всі дані, отримані, або згенеровані під час роботи застосунку повинні бути збережені у базі даних.

1.4.5 Вимоги до інтерфейсу користувача

Користувацький інтерфейс дуже важливий компонент програмної системи і його розробці повинна бути приділена достатня кількість уваги. Основні вимоги до користувацького інтерфейсу інтерактивної навчальної платформи для вивчення STEM дисциплін:

- адаптивність під різні розміри дисплеїв. В сучасному світі дисплеї мають різні розміри і форми, тому програмний інтерфейс користувача повинен адаптуватись до будь-якого розміру екрану;
- прямолінійність: інтерфейс повинен бути інтуїтивно зрозумілим та не обтяженим непотрібною інформацією. Користувач повинен швидко зрозуміти, як взаємодіяти з програмою, які є можливості та як їх використовувати. Це досягається за допомогою використання простих та зрозумілих іконок, відповідної назви кнопок, лаконічних повідомлень та т.д.;
- підтримка двох мов: української та англійської. Це дуже важливо, якщо програма розрахована на користувачів з різних країн або регіонів. Також користувач повинен мати можливість вибрати бажану мову в програмному інтерфейсі;
- швидкість та продуктивність: інтерфейс повинен бути швидким та продуктивним. Користувачі не хочуть витратити багато часу на очікування

завантаження сторінки або виконання команд. Тому необхідно мінімізувати час завантаження сторінки та максимально прискорити виконання команд;

- анімації та ефекти: необхідно розумно використовувати анімації та ефекти. Вони можуть допомогти зрозуміти, як працює програма, та зробити взаємодію більш приємною. Однак, вони не повинні стати завадою для користувача або займати багато ресурсів;

- наявність елементів гейміфікації, які зроблять процес вивчення дисциплін більш цікавим і швидким.

1.4.6 Вимоги до технічних засобів, операційної системи і мови програмування

Інтерактивний інтелектуальний навчальний застосунок повинен бути написаним на мові програмування C# у середовищі розробки Rider. Програма повинна бути повністю кросплатформеною, тобто повинна однаково гарно працювати і запускатися на комп'ютері з операційною системою не старше за Windows 10, у сучасному веббраузері і на мобільних пристроях з операційною системою Android не старше 10 версії.

Для забезпечення нормальної роботи ОС Windows 10 згідно мають бути виконані наступні системні вимоги:

- процесор із тактовою частотою 1 ГГц або швидший чи SoC;
- 1 гігабайт оперативної пам'яті (для 32-розрядної версії) або 2 ГБ (для 64-розрядної версії);
- обсяг вільного дискового простору 16 ГБ (для 32-розрядної версії) або 20 ГБ (для 64-розрядної версії);
- графічна плата із підтримкою DirectX 9 або новішої версії з драйвером WDDM 1.0 або новішим;
- дисплей: 800 x 600.

Для забезпечення нормальної роботи у веббраузері він має підтримувати HTML 5 (наприклад, Google Chrome v. 110).

Для забезпечення нормальної роботи мобільного застосунку на мобільному пристрої він повинен мати операційну систему не старше Android 10, а також мати мінімум 2 Гб ОЗУ.

1.4.7 Вимоги до інформаційної та програмної сумісності

Інформаційні структури повинні бути розроблені згідно з визначеними структурами вхідних даних. Отримані вхідні коди повинні бути написані мовою C#. Середовище розробки програмного продукту Rider.

1.4.8 Вимоги до програмної документації

Програмна документація має включати: керівництво програміста, керівництво оператора, опис програми та тексти основних програмних модулів.

1.4.9 Порядок приймання і тестування

Тестування даного застосунку проводиться у вигляді виконання усіх можливих сценаріїв роботи оператора з інтерактивною навчальною системою.

2 РОЗРОБКА СТРУКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Вибір інструментарію розробки програмного забезпечення

Під час вибору інструментів розробки головним критерієм була швидкість розробки, а можливість додавання 3–rd party модулів. Також дуже важливо було обрати новітні інструменти, які широко використовуються в комерційній розробці ПЗ.

Під час планування структури вебсервісу було вирішено все зробити в дусі сучасних вебсервісів: розділити клієнтську частину (Frontend) і серверну (Backend).

Для реалізації клієнтської частини була обрана мова програмування C# і ігровий рушій Unity, а для серверної – мову C# і фреймворк ASP.NET Core. Всі переваги/недоліки і нюанси обраного інструментарію будуть розглянуті у наступному підрозділі.

2.1.1 Огляд особливостей обраних мов програмування і фреймворків

2.1.1.1 Мова програмування C# (C Sharp)

Це об'єктно–орієнтована мова програмування з безпечною системою типізації для платформи .NET. Синтаксис C# близький до C++ і Java. Мова має строгую статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції–члени класів, атрибути, події, властивості – тобто все необхідне для зручної розробки.

Однією з ключових переваг мови програмування C# є те, що вона підтримує багато парадигм програмування, включаючи функціональну, яку я відверто кажучи обожнюю.

Платформа .NET у поєднанні з різноманітністю NuGet пакетів та фреймворків, робить цю мову однією з найкращих, якщо не найкращою для розроблення у тому числі програмного застосунку з серверним компонентом.

Вибір C# як основної мови програмування для розробки інтерактивного навчального застосунку для полегшення вивчення STEM дисциплін дозволив використовувати так сучасні і потужні технології як ASP.NET Core і Unity для розробки вебсерверу і користувацького інтерфейсу відповідно.

2.1.1.2 Фреймворк ASP.NET Core

Цей фреймворк являє собою вільне та відкрите програмне забезпечення для створення серверної частини вебзастосунків з продуктивністю вищою ніж у ASP.NET. Він розроблений корпорацією Microsoft і простими розробниками, які приймають участь у написанні і модернізації open-source проєктів.

Цей фреймворк дуже легкий для розуміння для будь кого, хто зіштовхувався з розробкою серверної частини.

Ключовою перевагою під час вибору цього фреймворку з півсотні інших була висока швидкодія отриманих вебзастосунків, а також те, що код з використанням ASP.NET Core можна писати у тому числі на мові програмування C# (а також F# і Visual Basic)

2.1.1.3 Unity

Unity є платформою для розробки ігор, але вона також має потужні інструменти для розробки користувацьких інтерфейсів. Основними перевагами використання Unity для розробки користувацьких інтерфейсів є:

- кросплатформенність. Unity підтримує розробку для різних платформ, таких як Windows, Mac, Android та iOS. Це означає, що розробники можуть створювати єдиний інтерфейс, який буде працювати на різних платформах без потреби розробляти окремі версії для кожної з них;

- легкість використання. Інтегроване середовище розробки для Unity під назвою Unity Editor має інтуїтивний і зручний інтерфейс, що дозволяє розробникам швидко створювати та налаштовувати елементи інтерфейсу. Бібліотека компонентів Unity також містить широкий набір готових елементів, які можна легко використовувати в проєкті;
- розширюваність. Unity має велику спільноту розробників та багато сторонніх розширень, що дозволяє розробникам розширювати функціональність своїх проєктів;
- підтримка анімації. Unity має потужні інструменти для створення анімацій, що дозволяє розробникам створювати динамічні та привабливі елементи інтерфейсу;
- підтримка 3D-графіки. Unity має потужні можливості для рендерингу 3D-графіки, що дозволяє розробникам створювати складні та деталізовані елементи інтерфейсу.

Отже, використання Unity для розробки користувацького інтерфейсу навчальної платформи є обґрунтованим через його кросплатформеність, легкість використання, розширюваність і підтримку.

2.1.2 Огляд особливостей обраного середовища розробки

2.1.2.1 Unity Editor

Unity Editor – офіційне спеціалізоване інтегроване середовище для розробки з використанням Unity. Саме воно було обране для цієї дипломної роботи. Unity Editor має графічний інтерфейс та дозволяє розробникам створювати, налаштовувати та тестувати свої проєкти в одному місці. Основні можливості Unity Editor включають:

- редактор сцен. Unity Editor має редактор сцен, який дозволяє розробникам створювати та редагувати 3D-сцени, додавати об'єкти, світло та інші елементи;

- редактор компонентів. Unity Editor має редактор компонентів, який дозволяє розробникам додавати та налаштовувати компоненти для об'єктів на сцені. Компоненти дозволяють розробникам додавати функціональність до об'єктів та налаштовувати їх параметри;
- редактор ресурсів. Unity Editor має редактор ресурсів, який дозволяє розробникам додавати та налаштовувати ресурси, такі як моделі, текстури, звуки, анімації та інші;
- редактор скриптів. Unity Editor має редактор скриптів, який дозволяє розробникам писати скрипти для своїх проєктів. Unity підтримує мови програмування C# та JavaScript;
- редактор анімацій. Unity Editor має редактор анімацій, який дозволяє розробникам створювати та редагувати анімації для об'єктів на сцені;
- редактор фізики. Unity Editor має редактор фізики, який дозволяє розробникам налаштовувати параметри фізики для об'єктів на сцені.

2.1.2.2 Rider

Для розробки на мові програмування C# було обране інтегроване середовище розробки Rider.

Воно має значну кількість переваг порівняно зі своїми аналогами по типу Visual Studio:

- швидкодія. Створення коду буде проходити гладко без будь-яких пауз чи збоїв. І як відзначає більшість користувачів, що працювали з Visual Studio і Rider, останній працює набагато стабільніше і швидше ніж VS;
- JetBrains Rider є кросплатформним, він може працювати на платформах Windows, Mac або Linux з однаковою функціональністю та стабільністю. Visual Studio працює переважно на платформі Windows. І якщо є необхідність перейти на Linux або Mac, то необхідно буде придбати додаткові рішення: Visual Studio Code (для Linux) і Visual Studio для Mac.

Головним недоліком є те, що версії Visual Studio для Mac і Linux мають різний функціонал і зовнішній вигляд до якого доведеться звикати. Rider, як зовні так і за своїми функціями, однаковий на всіх платформах, тому якщо користувач вирішить переключитися з Windows на Mac або Linux, він отримає вже звичне середовище розробки і не витрачатиме дорогоцінний час на навчання;

- середовище Rider включає більшість функцій популярного розширення Visual Studio для розробників .NET – ReSharper. У складі Rider є значний набір для рефакторингу, перевірки коду і контекстних дій для всіх мов і технологій, що підтримуються ним. Visual Studio також має набори для рефакторингу і перевірки помилок коду, але набагато більш обмежений, ніж ті, що надані в Rider і ReSharper;

- рішення та проекти, з якими працює JetBrains Rider, повністю сумісні з Visual Studio і не використовують пропрієтарні формати.

Основним і єдиним недоліком є те, що щоб використовувати Rider необхідно придбати ліцензію.

2.1.3 Огляд особливостей обраної системи управління базами даних

У проєкті використовується база даних MS SQL Server. Вона виконує головну функцію по збереженню та наданню даних у відповідь на запити інших застосунків, які можуть виконуватися як на тому ж самому сервері, так і у мережі.

Microsoft SQL Server – це реляційна система управління базою даних (СКБД). У реляційних базах даних дані зберігаються в таблицях. Взаємопов'язані дані можуть групуватися в таблиці, крім того, можуть бути встановлені також і взаємини між таблицями. Звідси і пішла назва реляційні – від англійського слова relational (споріднений, пов'язаний відносинами, взаємозалежний). Користувачі отримують доступ до даних на сервері через

програми, а адміністратори, виконуючи завдання конфігурування, адміністрування та підтримки бази даних, виробляють безпосередній доступ до сервера. SQL Server є масштабованою базою даних, це означає, що вона може зберігати значні обсяги даних і підтримувати роботу багатьох користувачів, які здійснюють одночасний доступ до бази даних.

Для взаємодії з базою даних був використаний EntityFramework Core. Це вбудований фреймворк, який значно полегшує взаємодію з базою даних, коли ви розробляєте програми з використанням .NET Core (у моєму випадку я використовую ASP.NET Core, який фактично є надбудовою над .NET Core). Даний фреймворк дозволяє взаємодіяти з базою даних так, ніби вона являє собою сукупність масивів, тобто ніяких SQL запитів вам писати самостійно не потрібно – все це за вас зробить Entity Framework.

Встановлюється у проєкт за допомогою пакетного менеджера NuGet.

2.2 Опис модулів системи

Після аналізу технічного завдання була визначена структура майбутнього проєкту.

Було вирішити зробити архітектуру навчальної платформи повністю модульною. Застосунок представляє собою три окремі проєкти: користувацький інтерфейс (UI), вебсервер з бізнес логікою застосунку і базу знань з хімії.

UI відповідає за відображення інформації, отримання вхідних користувацьких даних і представлення вихідних. Користувацький інтерфейс взаємодіє з навчальною платформою за допомогою HTTP запитів.

Вебсервер – ключовий модуль всієї навчальної платформи. Вебсервер фактично в нашому випадку є сполучною ланкою між інтерфейсом користувача, базою даних і системою знань. Вебсервер містить бізнес логіку програми, а саме роботу з користувачами та їх даними, статистикою успішності учнів, а також робить запити в базу даних та систему знань,

агрегує (перетворює) отриману інформацію та повертає її в користувацький інтерфейс, який у свою чергу представляє її у найбільш зручному та зрозумілому вигляді. Вебсервер для зовнішнього світу представлений деяким програмним інтерфейсом REST API, на який можна надсилати HTTP запити та отримувати відповіді у форматі JSON – такий підхід є найбільш прийнятним та популярним у сучасній розробці вебдодатків.

Навчальна платформа проектувалася з розумінням того, що є багато STEM дисциплін, а програмісти зазвичай не є експертами в одразу в усіх точних науках. Тому було вирішено для кожної STEM дисципліни додати підсистему, в яких будуть знаходитись бази знань і налаштований логічний вивід.

У межах цієї кваліфікаційної роботи розглядається лише дисципліна, хімія. База знань з хімії має веб-інтерфейс, у якому налаштовані публічні роути, на які можна відправляти HTTP запити з вебсерверу.

Система знань з хімії реалізована за допомогою SWI Prolog, який дозволяє працювати з собою за допомогою HTTP запитів. Для зручності розробки та розгортання (deploying process) SWI Prolog вебсервер поміщений в Docker контейнер. Це дозволяє встановлювати та запускати його абсолютно на будь-якій операційній системі за допомогою одного кліка. Для запуску не потрібно встановлювати Prolog і розбиратися, як там все побудовано і налаштовано, просто потрібен Docker і програма запускається за допомогою однієї команди `docker run`. Такий рівень абстракції значно допомагає прискорити та полегшити процес розробки та розгортання, і у той же час робить взаємодію вебсерверу і бази знань абсолютно модульною.

Схема взаємодії цих підсистем представлена на рисунку 2.1.

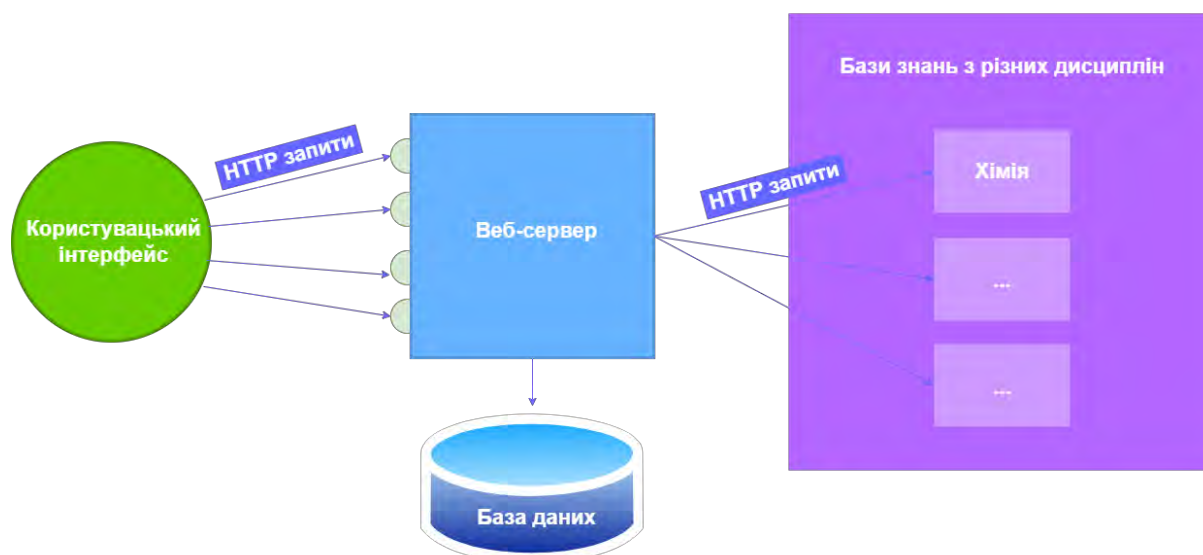


Рисунок 2.1 – Схема взаємодії підсистем навчальної платформи

Така архітектура навчальної платформи надає нам дуже великі простори для розвитку системи і додавання нового матеріалу. Розділення представлення інформації, бізнес логіки, а також баз знань з різних дисциплін дає змогу дизайнерам, програмістам та експертам зі знань робити виключно свою роботу, що дуже пришвидшить розробку застосунку у майбутньому і його якість.

2.3 Функціональна схема застосунку

Функціональна схема застосунку – це схематичне відображення функцій та компонентів, які є частинами програми або системи, та їх взаємодії. Функціональна схема дозволяє детально проаналізувати функціональність програми та визначити способи її реалізації.

Так як за основу була обрана модульна архітектура навчальної платформи, то необхідно розглянути окремо функціональну схему користувачького інтерфейсу і вебсерверу.

Функціональна схема користувачького інтерфейсу навчальної платформи зображена на рисунку 2.2.

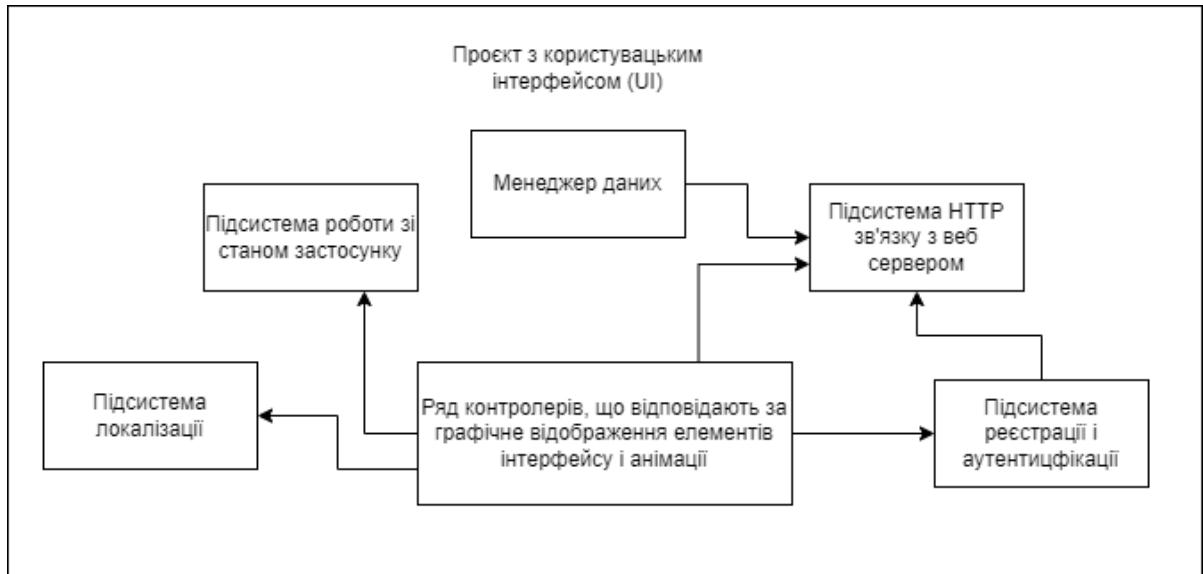


Рисунок 2.2 – Функціональна схема користувацького інтерфейсу

Як видно з рисунку користувацький інтерфейс складається з ряду підсистем, які укупі працюють як повноцінна система. Безпосередньо за зовнішній вигляд платформи, логіку поведінки і анімації відповідають ряд графічних контролерів. Ці контролери у свою чергу взаємодіють з іншими модулями користувацького інтерфейсу. Наприклад, у підсистемі локалізації міститься логіка зміни мови інтерфейсу. Підсистема роботи зі станом застосунку допомагає визначати поточний стан застосунку, змінювати його і залежним контролерам реагувати на його зміну. Менеджер даних представляє деяке сховище даних у пам'яті застосунку (запущеного процесу), які використовуються при відображенні інтерфейсу. Підсистема реєстрації і аутентифікації містить відповідну до назви бізнес логіку. Останньою дуже важливою підсистемою застосунку графічного інтерфейсу є модуль HTTP зв'язку з вебсервером: він містить відповідні налаштування для зв'язку ендпоінтами серверу, допомагає робити запити і перетворювати відповіді у необхідні підтримувані формати. HTTP підсистему напряду використовуються контролери, менеджер даних і модуль реєстрації і аутентифікації.

Функціональна схема вебсерверу навчальної платформи зображена на рисунку 2.3.

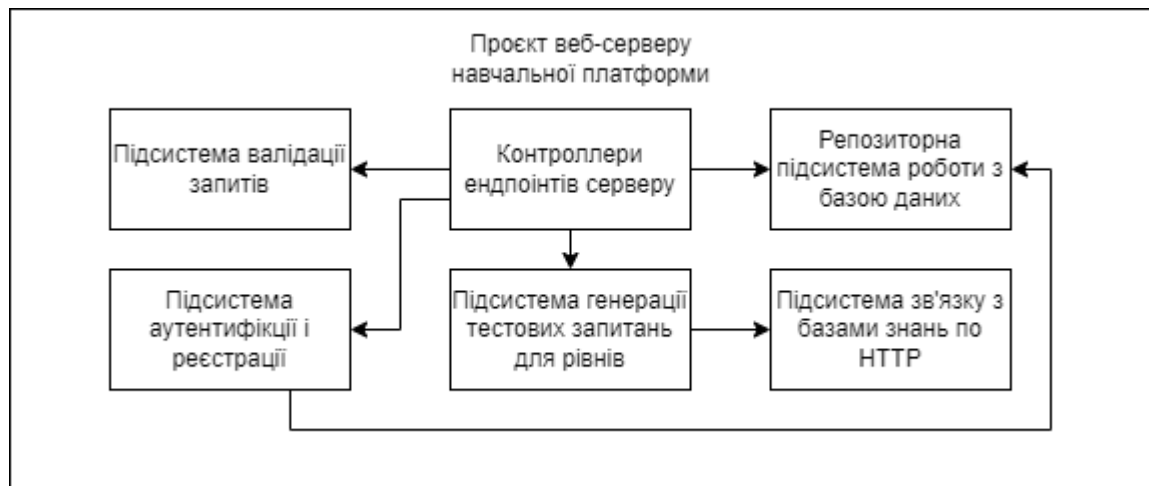


Рисунок 2.3 – Функціональна схема вебсерверу навчальної платформи

Відповідно до функціональної схеми вебсервер складається з 6-и основних підсистем. Підсистема контролерів ендпоінтів відповідає за виклик визначеного обробника для певного шаблону запиту, цей обробник приймає деякий запит, представлений відповідним типом даних, і повертає відповідь у форматі HTTP. Для коректної роботи контролерів ендпоінтів необхідна підсистема валідації запитів: кожен вхідний запит перевіряється на відповідність до формату і визначених правил, якщо запит валідний – по він починає оброблюватись, а якщо невалідний – відхиляється і сервер повертаю HTTP помилку BadRequest. Для зручної роботи з базою даних спроектована репозиторна підсистема, що фактично є обгорткою над програмним API доступу до бази даних. Її використовують контролери ендпоінтів, а також інша дуже важлива підсистема – підсистема аутентифікації і реєстрації, робота якої базується на використанні JWT токенів. Ключовою підсистемою навчальної платформи є підсистема генерації тестових запитань для рівнів. Вона відповідає за псевдо випадкову генерацію питань з дисциплін, а також за валідацію наданих користувачем відповідей. Ця підсистема використовує

підсистему зв'язку з базами знань по HTTP, яка містить логіку відправки запитів і отримання відповідей у відповідних для баз знань форматах.

Отже сукупність підсистем утворюють користувацький інтерфейс і вебсервер, які взаємодіючи представляють інтерактивну навчальну платформу з STEM дисциплін.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

У даному розділі розглянуто основні рішення функціонування бізнес логіки програми, а також структура класів програми.

3.1 Алгоритми функціонування розробленої програми

Як вже було зазначено у розділі 2.3 – вебсервіс використовує дискретні клієнтську і серверну частини (користувацький інтерфейс і вебсервер, Frontend і Backend), які спілкуються між собою за допомогою HTTP запитів.

Далі буде більш детально розглянуто шлях запиту.

Перш за все треба зауважити, що запит створюється на стороні клієнта за допомогою наявного у .NET Core платформі класу HttpClient. Наприклад, користувач натискає на кнопку «Зареєструватися» і на сервер посиляється POST–запит. (рис. 3.1).

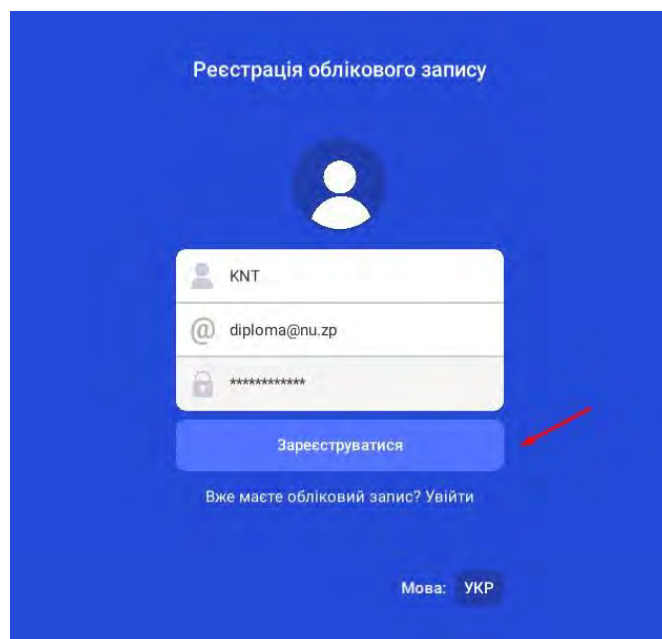


Рисунок 3.1 – Користувач натискає на кнопку «Зареєструватися»

Далі запит потрапляє на сервер проходить через канал запитів Middleware, у якому відбувається логування і перевірка запиту, а потім, якщо запит «валідний» – у відповідний контролер, до якого прив’язана функція–обробник (рис. 3.2).

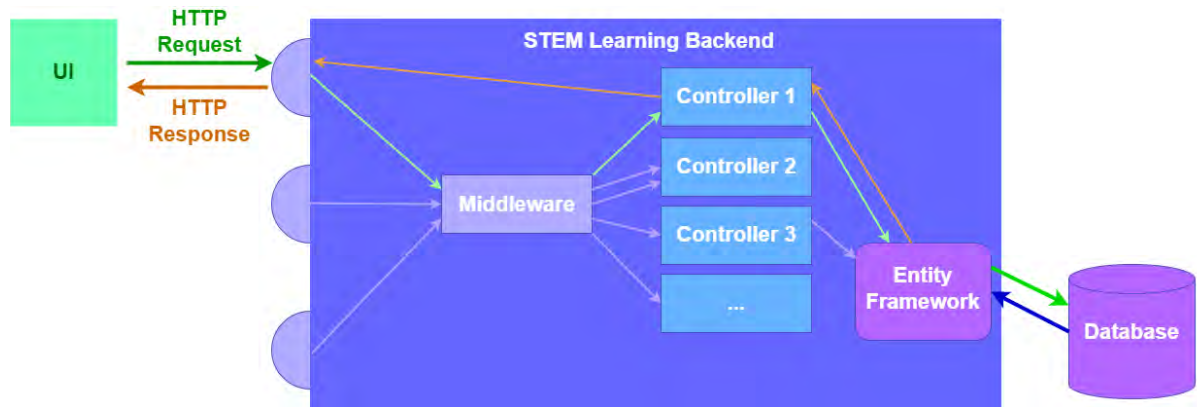


Рисунок 3.2 – Детальна схема роботи серверної частини

Запит, який використовується як приклад (користувач хоче зареєструватися), потрапляє у контролер UserController, в якому функція–обробник звертається до бази даних через посередника, EntityFramework, і робить там відповідні зміни. Відповідь повертається користувачеві і клієнтська частина регенерується відповідно до змін.

Після розуміння деталей взаємодії клієнтської та серверної частини розглянемо основні алгоритми навчальної платформи:

- аутентифікація та реєстрація;
- отримання даних щодо дисциплін;
- генерація рівнів;
- проходження рівнів користувачами (вибір рівня → запит на сервер на генерацію питань → проходження рівня → відображення прогресу і досягнень).

Слід також сказати декілька слів про формат запитів і відповідей вебсерверу. Кожен запит і відповідь представлені окремими C# класами, які знаходяться в спеціалізованому проєкті рішення (solution) вебсерверу. Після

компіляції отримана з цього проєкту динамічна бібліотека (.dll файл) автоматично копіюється в структуру проєкту користувацького інтерфейсу, що дозволяє забезпечити спільний формат запитів і відповідей, який використовує і вебсервер, і користувацький інтерфейс.

3.1.1 Алгоритми аутентифікації та реєстрації

Алгоритми аутентифікації та реєстрації у навчальній платформі у широкому сенсі не відрізняється від загальноприйнятого алгоритмів, які широко використовуються у сучасних системах.

У навчальній платформі, яка була розроблена у рамках цієї дипломної роботи ідентифікація користувача базується на використанні JWT токенів, які забезпечують додаткову безпеку для застосунку.

JWT (JSON Web Token) – це стандарт відкритого формату для передачі обмежених деяких даних між двома сторонами у компактному та самостійному форматі. Він використовується для аутентифікації та авторизації в розподілених системах, зокрема вебдодатках.

JWT складається з трьох частин: заголовка (header), набору даних про користувача (payload) і підпису (signature). Кожна частина розділена крапкою і закодована в форматі Base64URL. Основні компоненти JWT:

- заголовок (header) вказує тип токена (typ) і алгоритм підпису (alg). Зазвичай використовується алгоритм HMAC SHA256 або RSA;
- набір даних про користувача (payload) містить дані, які представляють корисну інформацію про суб'єкта (користувача) та іншу додаткову інформацію. Ці дані закодовані у форматі JSON;
- підпис (signature) використовується для перевірки цілісності та автентичності токена. Підпис обчислюється з використанням заголовка, набору даних про користувача та секретного ключа або приватного ключа.

JWT токен генерується сервером після успішної аутентифікації чи реєстрації користувача і відправляється йому у відповіді. Користувацький

інтерфейс у свою чергу відповідає за отримання такого токenu, його зберігання і відправка у заголовок запиту.

У навчальній платформі вебсервер організований таким чином, що доступ до всіх контролерів дозволений лише за умови наявності валідного JWT токenu.

Зручно показати роботу алгоритмів реєстрації і аутентифікації за допомогою діаграм прецедентів UML. Ці діаграми відображені на рисунках 3.3 і 3.4 відповідно.

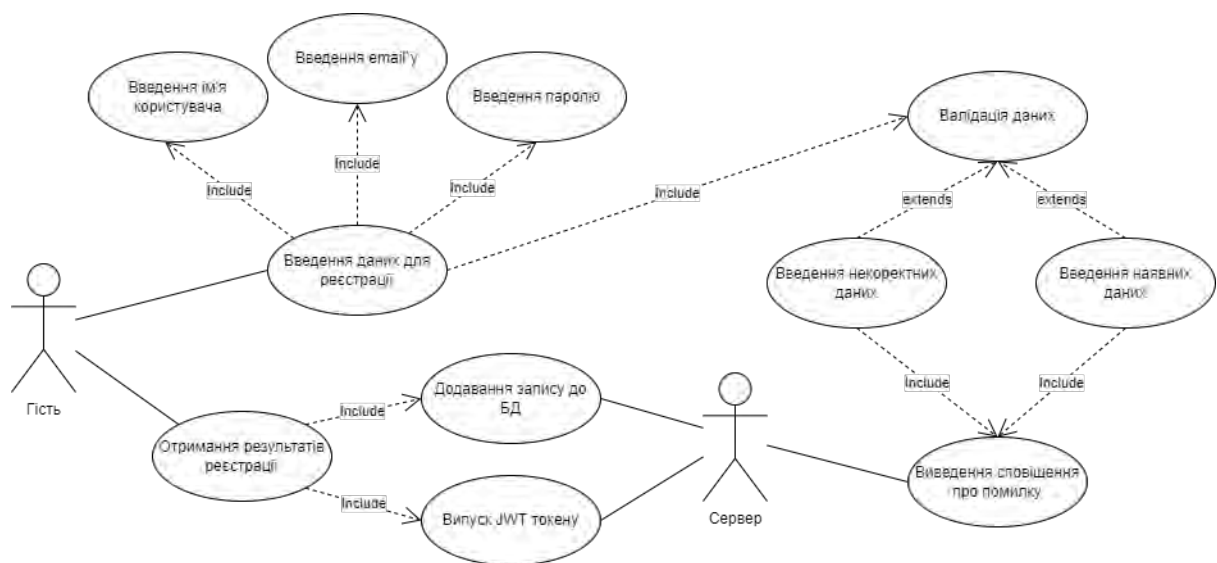


Рисунок 3.3 – Діаграма прецедентів для реєстрації користувача у навчальній платформі

При аутентифікації користувач надає свої дані (ім'я, email та пароль), які відправляються на сервер і там перевіряються на валідність формату (додаткова перевірка на валідність також є у користувацькому інтерфейсі) і звіряються з наявними записами у базі даних. Після успішної аутентифікації, користувач має доступ до свого особистого запису і всіх даних і функцій навчальної платформи. Якщо введені користувачем дані невалідні чи такого користувача не існує, то сервер відповідає помилкою і ця помилка відображається користувачу в інтерфейсі. Як результат вдалої аутентифікації сервер відправляє клієнту щойно згенерований JWT вебтокен.

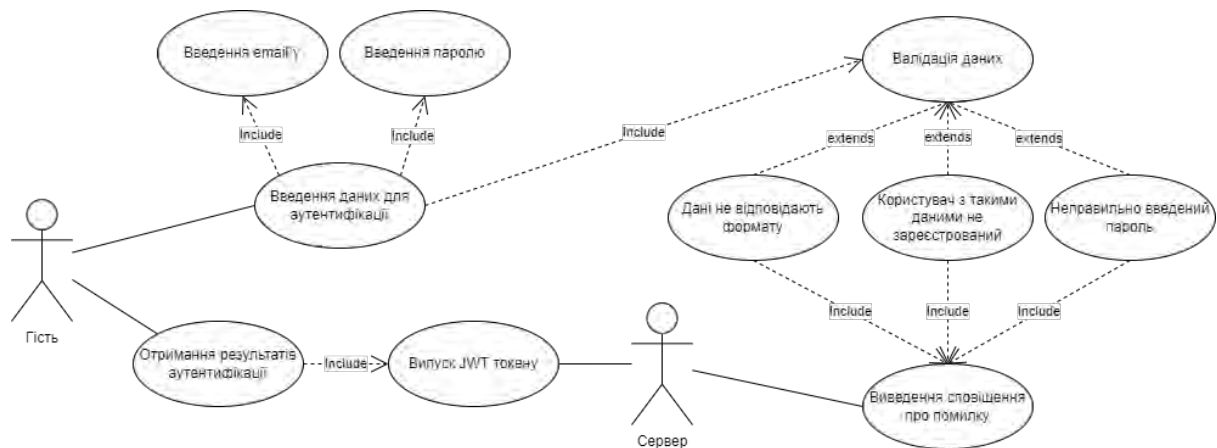


Рисунок 3.4 – Діаграма прецедентів для алгоритму аутентифікації

Логіка алгоритму реєстрації дуже схожа на логіку аутентифікації. Лише деякі аспекти дещо інші. Наприклад, при реєстрації на сервер повинні прийти додаткові особисті дані користувача (у нашому випадку це його ім'я), які потім будуть позначені як його персональна інформація. Також варто зазначити, що при реєстрації система перевіряє чи вже існує користувач з такими даними (у нашому випадку перевірка відбувається по адресі пошти, email'у) і якщо не існує, то відповідний запис з даними про нового користувача додається до бази даних і випускається відповідний JWT токен, а у протилежній ситуації – повертається помилка.

3.1.2 Алгоритм отримання даних щодо дисциплін

Після вдалої аутентифікації/реєстрації користувача спрацьовує спеціальна функція користувацького інтерфейсу, яка ініціює завантаження даних щодо дисциплін навчальної платформи. Відправляється GET запит на адресу “[доменне_ім'я]/api/subjects” і спеціальна функція–обробник починає послідовну вибірку даних з бази даних. Спочатку зі спеціальної таблиці береться інформація щодо дисциплін навчальної платформи. Для тих дисциплін, у яких властивість isAvailable істинна, вибираються їх навчальні

етапи (stages, містяться в окремій таблиці). Для тих етапів, у яких властивість `isAvailable` дорівнює `true`, вибираються рівні пов'язані з цими етапами. Між іншим для кожного виду сутностей вибираються прогрес поточного користувача (прогрес проходження дисциплін, етапів і рівнів). Прогрес представлений числом з плаваючою крапкою для дисциплін і етапів, і булевим значенням для рівнів. Для рівнів також вибирається інформація пов'язана з спробами проходження користувача. Поточний користувач визначається за допомогою JWT токєну (в ньому зберігається емейл користувача, який можна використовувати для знаходження користувача у базі даних). Вибрана з бази даних інформація, щодо дисциплін, етапів і рівнів агрегується в спеціальний об'єкт класу відповіді на поточний запит і відправляється назад у користувацький інтерфейс. Користувацький інтерфейс зберігає ці дані у пам'яті програми і організовує доступ до них за допомогою класу-менеджеру даних (`DataManager`), звідки будь який контролер може отримати до них доступ.

3.1.3 Алгоритм генерації рівнів

Алгоритм генерації рівнів є ключовим у навчальній платформі.

Генерація рівнів – достатньо ресурсоємна процедура навчальною платформою, яка сильно навантажує вебсервер та базу знань з певної дисципліни, тому рівні генеруються тільки у той момент, коли вони необхідні, а не під час планового отримання даних з вебсерверу щодо дисциплін після аутентифікації/реєстрації. Запит на генерацію рівня відправляється, коли користувач в інтерфейсі обирає рівень і натискає кнопку “Грати” (рис. 3.5).

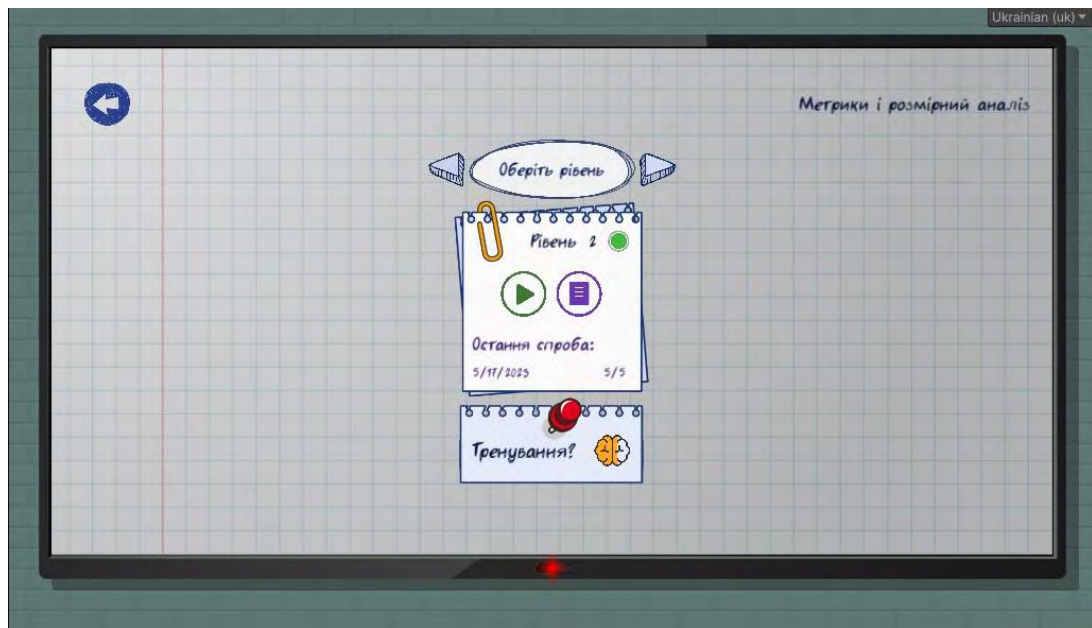


Рисунок 3.5 – Демонстрація екрану інтерфейсу, на якому розташована кнопка “Грати” (зелена кнопка з трикутником), натискання на яку починається генерація рівня

Після натискання на кнопку “Грати” відправляється GET запит на ендпоінт серверу з адресою “[доменне_ім’я]/api/levels/[level_id]?locale=[ua/en]”. Сервер приймає запит, валідує його і спеціальна функція–обробник починає свою роботу. На сервері створений ряд абстракції для простої і модульної генерації рівнів, який буде більш детально розглянутий нижче.

Слід почати з того, що представляє собою рівень. Рівень (у контексті навчальної платформи) – це сукупність тестових питань. Тестове питання являє собою текст питання, варіанти відповідей і правильні/прийнятні відповіді. Рівень представлений окремим класом Question з властивостями Text (string), AcceptableAnswers (List<string>) і AnswersVariants (List<string>). Отже кінцевою ціллю генерації рівня є генерація сукупності питань, які відповідають номеру рівня (різні рівні – різні типи питань і їх складність). Питання повинні бути згенеровані і перемішані псевдо випадковим чином. В залежності від рівня повинна змінюватись і складність питань, що визначена

комплексністю вхідних параметрів (наприклад, питання “Що більше 1 метр чи 1 кілометр?” значно легше ніж питання “Що більше 100 м^2 чи 0.1 км^2 ?”)

У серверній частині навчальної платформи створена класово-інтерфейсна структура, яка полегшує генерацію рівнів. Головне, що варто тут зазначити, це те, що в основі генерації питань лежить патерн проєктування Builder, назва якого в контексті навчальної платформи видозмінена на Generator. Патерн проєктування Builder відноситься до категорії "породжуючих" (creational) патернів. Він використовується для конструювання об'єктів складної структури шляхом поетапного побудови. Основна ідея патерну Builder полягає у тому, що він надає гнучкість у побудові складних об'єктів шляхом розділення самого процесу побудови від його представлення. Патерн Builder є особливо корисним, коли маємо справу з об'єктами, складність яких може змінюватись або коли є багато можливих варіацій структури об'єкта. Він також сприяє покращенню читабельності коду і дозволяє зменшити залежність між клієнтом і складним об'єктом, який створюється.

Генерація у поточному стані проєкту налаштована для 3 типів питань:

- Unit-measure – питання відповідностей для метрики і того, для чого вона використовується (приклад питання: “Що вимірюється величиною ‘моль’?”);
- How-many – питання для визначення кількості однієї метрики в іншій з використанням префіксів (приклад питання: “Скільки метрів в 1 пікнометрі?”);
- Comparison – питання для порівняння різних кількостей сутностей (приклад питання: “Скільки метрів в 1 Гігаметрі?”).

Варто зазначити, що такі типи питань визначені конкретно для навчального етапу “Метрики і аналіз розмірностей” для дисципліни “Хімія” у навчальній платформі.

В навчальній платформі на момент написання цієї кваліфікаційної роботи представлено 9 рівні для навчального етапу “Метрики і аналіз

розмірностей”, які мають наступну нумерацію: 0, 1, 2, 2A, 2B, 3, 4, 4A, 4B. Для кожного з трьох типів питань створений клас шаблон, що відповідає за текстове представлення цих питань, їх локалізацію і місця для підстановок. Для кожного рівня визначений спеціальний клас, який наслідує QuestionDirector. Цей клас відповідає за вибір необхідних типів питань, їх мову перекладу і підстановку значень.

При генерації рівня створюється вибірка всіх можливих питань, а потім з них псевдо випадковим чином обирається необхідна кількість. Останнім етапом є генерація правильних відповідей на питання. Вебсервер в сутності містить тільки бізнес логіку самої програми і керує її даними. Він і гадки не має, що існує, наприклад, така дисципліна як хімія і там є свої закони чи поняття. Тому знання з дисциплін зберігаються у базах знань, до яких можна отримати доступ по HTTP. Отже вебсервер формує спеціальні запити, формат яких визначений базою знань, отримуючи відповіді на задані питання. Для отримання відповіді на одне питання відправляється один HTTP запит. Отримані відповіді агрегуються і додаються по об’єкту питання. Слід також зазначити, що вся логіка перекладів питань і відповідей на різні мови (у нашому випадку на українську і англійську) реалізована на вебсервері за допомогою класів, інтерфейсів і перелічень.

Згенеровані питання з відповідями відправляються у клієнтську частину.

3.1.4 Алгоритм проходження рівнів користувачами

Після того як користувач обрав рівень і натиснув кнопку “Грати” (рис. 3.5) відправляється запит на сервер, відповіддю на який є згенерований набір питань і відповідей. Рівень починається і користувач бачить перед собою екран з питанням і формою для введення/вибору відповіді (рис. 3.6). Є три варіанти завершення проходження рівня: відповіді на всі надані питання, примусовий вихід з рівня і закінчення виділеного часу. Після завершення

рівня на спеціальний ендпоінт вебсерверу з роутом “[доменне_ім’я]/api/levels/[level_id]” вебсервер відправляє POST запит для засвідчення спроби проходження рівня (level try). До тіла запиту додаються дані, щодо кількості правильних відповідей на питання (така кількість визначається в користувацькому інтерфейсі). Після валідації даних спрацьовує спеціальна функція–обробник, яка визначає чи достатня кількість правильних відповідей для проходження рівня. Якщо ні, то до бази даних додається тільки інформація про спробу проходження, а якщо кількість достатня, то також оновлюються дані про прогрес користувача в опануванні дисципліни, навчального етапу (до якого належить рівень) і рівень позначається як пройдений. В якості відповіді від серверу користувацький інтерфейс отримає результат спроби проходження рівня, а також оновлені дані щодо прогресу (якщо вони були оновлені) і перелік рівнів, які стають доступними віднині (якщо рівень пройдений).



Рисунок 3.6 – Екран інтерфейсу, який бачить перед собою користувач під час проходження рівня

3.2 Вибір патерну проєктування

Основним патерном проєктування для серверної частини навчальної платформи був обраний “Медіатор”.

Патерн "Медіатор" є одним з поведінкових патернів проєктування, який дозволяє зменшити залежності між компонентами системи, встановивши зв'язок через посередника. Він сприяє зменшенню прямих зв'язків між об'єктами, забезпечує зв'язок лише через медіатор, що полегшує розширення та підтримку системи.

У контексті ASP.NET Core патерн "Медіатор" може бути використаний для керування взаємодією між різними компонентами системи. ASP.NET Core використовує підхід зворотного виклику для обробки вхідних HTTP-запитів, де обробка розташована у відповідних обробниках запитів (handlers).

За допомогою патерна “Медіатор” можна створити централізований медіатор, який виконує роль посередника між обробниками запитів. Коли HTTP-запит надходить до додатку, він передається в медіатор, який визначає, який обробник запиту повинен бути виконаний. Медіатор передає необхідні дані обробнику запиту і отримує результат його виконання. Це дозволяє знизити залежності між обробниками запитів і спростити систему.

Для ASP.NET Core доступний пакет MediatR, який надає реалізацію патерну "Медіатор" для обробки запитів. Цей пакет дозволяє створювати обробники запитів і реєструвати їх у середовищі ASP.NET Core. Клієнтські запити адресуються медіатору, який визначає, який обробник запиту повинен бути виконаний. Медіатор передає дані обробнику запиту і повертає результат клієнту.

Використання паттерна "Медіатор" у вебсервері навчальної платформи допомагає зробити код більш чистим і організованим, зменшує залежності між компонентами системи і полегшує розширення та підтримку.

Такий підхід робить систему більш модульною і дозволяє змінювати логіку обробки запитів без змінення клієнтського коду.

3.3 Опис структури класів системи

У цьому розділі буде описана структура класів розроблюваної системи.

3.3.1 Користувацький інтерфейс

Хоча Unity і надає програмістам можливість створювати користувацькі інтерфейси декларативним шляхом, переміщуючи компоненти і визначаючи їх властивості – цього недостатньо для створення дійсно високоякісного інтерфейсу. Тому необхідно було звернутися до програмування мовою C#, щоб втілити всі свої задуми.

Користувацький інтерфейс навчальної платформи має більше 10 різноманітних екранів, які містять велику кількість анімацій, логіку поводження багатьох унікальних компонентів, логіку обробки введених користувачем даних і виведення інформації. Ряд вимог до користувацького інтерфейсу став причиною до проєктування складної і великої структури класів, яка дала змогу створити сучасну інтерактивну навчальну платформу, зовнішній вигляд якої значно сприяє виникненню бажання вчитися і пізнавати STEM дисципліни.

В основі структури класів користувацького інтерфейсу лежить AppStateManager. Це клас, створений з використанням патерну проєктування Singleton, який відповідає за стан інтерфейсного застосунку. Стан визначений у межах перелічення AppState і може приймати наступні значення:

- Auth – аутентифікація/реєстрація;
- Menu – режим головного меню застосунку;
- Settings – перехід до налаштувань навчальної платформи;

- PlayLevel – режим проходження рівня;
- Loading – завантаження даних з вебсерверу;
- Training – режим тренування;
- Materials – перегляд матеріалів з дисциплін;
- Quit – вихід з застосунку.

AppStateManager надає можливість змінювати стан застосунку на будь-який обраний за допомогою спеціального методу UpdateState(AppState state) стан. Також в цьому класі визначена “подія” (event, тип даних у C#) на яку всі “бажаючі” класи застосунку можуть підписуватися і “отримувати сповіщення” (а точніше викликати один з своїх методів) після оновлення стану. Така абстракція зі станом застосунку в значній мірі допомогла при розробці користувацького інтерфейсу.

Початковим станом застосунку є Auth: користувач бачить перед собою екран аутентифікації і реєстрації, надає свої дані для входу чи реєстрації і, якщо все добре, то стан застосунку змінюється на Menu і він опиняється у головному меню навчальної платформи.

Іншим дуже важливим класом користувацького інтерфейсу є DataManager. Як видно з назви головною задачею цього класу (до речі теж реалізованого з використанням шаблону проєктування Singleton) робота з даними застосунку. Логіка саме цього класу ініціює завантаження дисциплін, навчальних етапів та рівнів, зберігає всю цю інформацію і надає до неї доступ іншим контролерам. Також в цьому класі реалізований ряд методів, яка роблять можливим відправку запиту на реєстрацію спроби проходження рівня користувачем і оновлення значень прогресу користувача у режимі реального часу.

Ці два класи є основою архітектури застосунку користувацького інтерфейсу, але їх недостатньо для створення повноцінного графічного представлення навчальної платформи. Всю іншу логіку поведінки користувацького інтерфейсу визначають десятки різноманітних класів, які умовно можна розділити на наступні групи:

- базові класи;
- контролери;
- класи для роботи з мережею;
- класи–помічники (helpers);
- моделі;
- класи скриптованих об’єктів (scriptable objects);
- перелічення;
- класи системи локалізації.

У наступних підрозділах будуть детально розглянуті всі ці групи класів.

3.3.1.1 Базові класи

До базових класів можна віднести класи, які містять деяку загальну логіку і можуть бути успадковані іншими класами. Винесення спільного функціоналу в окремі класи сприяє покращенню якості архітектури проєкту.

У навчальній платформі визначений ряд базових класів, які перелічені у таблиці 3.1.

3.3.1.2 Контролери

Контролери – основні класи користувацького інтерфейсу, які містять бізнес логіку поведінки різних елементів інтерфейсу, їх взаємодії, відображення і зміни. Цей тип класів містить найбільше примірників.

Контролери навчального застосунку наведені нижче:

- AuthController – контролер екрану аутентифікації/реєстрації;
- LoadingController – контролер екрану завантаження;
- MenuController – загальний контролер меню;
- SubjectProgressBarController – контролер рядку прогресу з дисципліни;

Таблиця 3.1 – Базові класи користувацького інтерфейсу навчальної платформи

Базовий клас	Призначення класу	Метод	Опис методу
1	2	3	4
Screen Controller	Базовий контролер для екрану застосунку, контролер прив'язаний до певного стану застосунку. Екран вмикається, коли застосунок змінює стан на визначений і вимикається, коли на будь-який інший.	ShowScreen(): void	Метод, що містить логіку включення екрану
		HideScreen(): void	Метод, що містить логіку вимкнення екрану.
		virtual OnEnterScreen(): void	Віртуальний метод, що може бути перевизначеним дочірнім класом, який викликається після включення екрану
		virtual OnExitScreen(): void	Віртуальний метод, що може бути перевизначеним дочірнім класом, який викликається безпосередньо перед вимкненням екрану
Progress Bar Behaviour	Базовий клас для рядку прогресу (progress bar), який містить всю уніфіковану логіку.	Enable(): void	Метод, що містить логіку включення прогрес бару
		Disable(): void	Метод, що містить логіку вимкнення прогрес бару
		SetProgress(float): void	Метод для зміни значення прогрес бару
		GetProgress(): float	Метод для отримання значення контролеру
		Display(): void	Метод, що містить логіку відображення контролеру
		abstract DisplayEnable(): void	Абстрактний метод, який повинен бути реалізований у дочірньому класі і спрацьовує, коли контролер вмикається

Продовження таблиці 3.1

1	2	3	4
		abstract DisplayDisable(): void	Абстрактний метод, який повинен бути реалізований у дочірньому класі і спрацьовує, коли контролер вимикається
		abstract DisplayProgress RateChange(float): void	Абстрактний метод, який повинен бути реалізований у дочірньому класі і спрацьовує, коли змінюється значення прогресу
Slider Behaviour <T>:	Базовий клас для контролеру слайдеру, якому при успадковуванні необхідно передати тип відображуваного об'єкту	Next(): void	Метод, що спрацьовує при зміні об'єкту слайдеру на наступний
		Previous(): void	Метод, що спрацьовує при зміні об'єкту слайдеру на попередній
		abstract Display(T): void	Абстрактний метод, який повинен бути реалізований у дочірньому класі і спрацьовує після зміни об'єкту слайдеру. Метод повинен містити логіку відображення об'єкту

- SubjectSelectorController – контролер слайдеру вибору дисципліни;
- StageProgressBarController – контролер рядку прогресу навчального етапу;
- StageSelectorController – контролер слайдеру вибору навчального етапу;
- MonitorController – контролер монітору, який є частиною меню, його 3-ім екраном;
- MonitorScreenController – контролер екрану монітору (різноманітна логіка зміни екранів на моніторі);

- LevelSelectorController – контролер слайдеру вибору рівня;
- AchievementsController – контролер екрану меню з досягненнями користувача;
- MaterialsController – контролер екрану з матеріалами з дисциплін;
- PopupController – контролер спливаючого вікна з повідомленням користувачу;
- TestingController – контролер процесу тестування (саме тут міститься вся логіка тестування);
- LevelTimerController – контролер таймеру рівня;
- LevelController – загальний контролер рівня;
- AnswersVariantsController – контролер списку вибору варіантів відповідей;
- PassIndicatorController – контролер індикатора проходження рівня;
- ResultScreenController – контролер, який відповідає за відображення результатів проходження рівня (тестування);
- TrainingController – контролер екрану тренування;
- TrainingCalculatorController – контролер блоку з обчисленнями екрану тренування;
- TrainingDemonstrationController – контролер демонстраційного блоку екрану тренування;
- ValueSwitcherController – контролер “змінника” значень (value switcher);
- ValueSwitcherLineController – контролер рядка “змінника” значень.

3.3.1.3 Класи для роботи з мережею

Користувачський інтерфейс робить запити на вебсервер навчальної платформи через мережу Інтернет і протокол HTTP. Для роботи з мережею використовуються стандартний клас .NET платформи HttpClient, але для

полегшення роботи з ним і інтеграції додаткової необхідної логіки створений клас `RestHttpClient`.

Клас `RestHttpClient` містить методи для роботи з REST API, а саме методи для GET і POST запитів. Під капотом цей клас використовує стандартний `HttpClient` та серіалізатор `json` з Nuget пакету `Newtonsoft` для серіалізації вхідних даних перед відправкою і десеріалізації відповідей серверу. Також слід зазначити, що саме цей клас має логіку додавання JWT токенів до запитів, якщо вони отримані після реєстрації чи аутентифікації.

3.3.1.4 Класи–помічники

Класи–помічники (`Helpers`) необхідні для полегшення часто зустрічаючихся завдань.

У проєкті користувацького інтерфейсу навчальної платформи визначено два класи–помічники:

`ApiHelper` – цей клас використовується для доступу до API вебсерверу навчальної платформи. Він містить методи для доступу для визначених ендпоінтів і під капотом використовує клас `RestHttpClient`.

`CalculationHelper` – клас, який містить в собі різноманітну логіку для полегшення обчислень в різних модулях застосунку.

3.3.1.5 Моделі

Моделі представлені класами сутностей необхідними для коректної роботи користувацького інтерфейсу навчальної платформи.

Моделі проєкту користувацького інтерфейсу приведені в таблиці 3.2.

Таблиця 3.2 – Моделі проекту користувацького інтерфейсу навчальної платформи

Клас	Призначення класу	Поле	Опис
1	2	3	4
Subject	Модель дисципліни	Id: long	Унікальний ідентифікатор
		Name: string	Назва дисципліни
		IsAvailable: bool	Чи доступна дисципліна
		Stages: List<Stage>	Список навчальних етапів
		Progress: float	Кількісне представлення прогресу
		LocalizedName: LocalizedString	Локалізована назва дисципліни
		Logo: Sprite	Картинка логотипу дисципліни
Stage	Модель навчального етапу	Id: long	Унікальний ідентифікатор
		Name: string	Назва навчального етапу
		IsAvailable: bool	Чи доступний навчальний етап
		Progress: float	Кількісне представлення прогресу
		Levels: List<Level>	Список рівнів
		LocalizedName: LocalizedString	Локалізована назва навчального етапу
		Logo: Sprite	Картинка логотипу навчального етапу
Level	Модель рівня	Id: long	Унікальний ідентифікатор
		Name: string	Назва рівня
		NumberOfQuestions: int	Кількість питань рівня
		IsAvailable: bool	Чи доступен рівень

Продовження таблиці 3.2

1	2	3	4
		IsCompleted: bool	Чи пройдений рівень
		Tries: List<LevelTry>	Список спроб проходження рівня
		TimeLimit: DateTime	Ліміт часу на проходження рівня
LevelTry	Модель спроби проходження рівня	TriedAt: DateTime	Дата і час спроби проходження рівня
		CompletedQuestions: int	Кількість питань, на які була надана правильна кількість відповідей
Question	Модель питання	Text: string	Текст питання
		AcceptableAnswers: List<string>	Список правильних відповідей
		AnswersVariants: List<string>	Список варіантів відповідей

3.3.1.6 Класи скриптованих об'єктів

ScriptableObject – базовий клас Unity, який дозволяє зручно створювати різноманітні об'єкти використовуючи графічний інтерфейс середі розробки UnityEditor, а також використовувати ці об'єкти передаючи їх у якості вхідних параметрів у Unity-компоненти.

У користувацькому інтерфейсі навчальної платформи визначено 4-и класи скриптованих об'єктів, використання яких дозволяє дуже зручно використовувати і ініціалізувати деякі сутності Unity (наприклад, Sprite, який уособлює деяку картинку). Назви цих класів закінчуються на Metadata, тому що головна ціль цих класів у програмному застосунку користувацького інтерфейсу – це надання деяких додаткових даних до моделей.

Скриптовані об'єкти навчальної платформи:

- SubjectMetadata – містить додаткові дані щодо дисципліни, зокрема іконку і локалізовану назву;
- StageMetadata – містить додаткові дані щодо навчального етапу, зокрему іконку і локалізовану назву;
- PrefixFactMetadata – містить локалізовані факти про префікси метрик (наприклад, “1 міліметр – стандартна ціна поділки на більшості лінійок”);
- AchievementMetadata – містить іконки для досягнень, які відкриває користувач по мірі проходження рівнів.

3.3.1.7 Перелічення

Перелічення широко використовуються у проєкті користувацького інтерфейсу навчальної платформи.

У застосунку з графічним інтерфейсом визначено два перелічення:

- Power – є відображенням ступеню числа. Містить значення: First, Second, Third;
- Prefix – є відображенням префіксу виміру. Містить значення: Pico, Nano, Milli, None, Kilo, Mega, Giga.

3.3.1.8 Класи системи локалізації

Класи системи локалізації полегшують роботу з локалізацією у проєкті користувацького інтерфейсу навчальної платформи.

Клас LocaleSelector містить логіку програмної зміни мови інтерфейсу і логіку забезпечення доступу до поточної мови. Мова інтерфейсу визначається використовуючи перелічення Locale, яке містить два значення, що уособлюють підтримувані мови інтерфейсу навчальної платформи: English (англійська), Ukrainian (українська).

3.3.2 Вебсервер

Вебсервер навчальної платформи складається з розгалуженої структури класів, які відповідають за бізнес–логіку застосунку і допомагають писати і підтримувати програмний код.

Основними класами вебсерверу, створеного з використанням ASP.NET Core, є Program і Startup.

Клас Program у ASP.NET Core є точкою входу в додаток і містить метод Main, який є вхідним точкою для виконання додатку. Цей клас виконує конфігурацію сервера, налаштовує служби та запускає сервер для обробки HTTP–запитів.

Клас Startup у ASP.NET Core є важливим класом, що виконує роль точки входу та конфігурації додатку. Він містить два основних методи: ConfigureServices і Configure. Метод ConfigureServices використовується для налаштування служб, які будуть доступні в додатку. Тут реєструються різні служби, такі як бази даних, кешування, аутентифікація, залежності тощо. Метод Configure використовується для налаштування конвеєра обробки HTTP–запитів (middleware pipeline). Сюди треба додавати різні middleware, які оброблюватимуть запити у відповідній послідовності.

Інші класи вебсерверу навчальної платформи можна розділити на групи:

- контролери;
- арі–класи;
- класи конфігурації;
- класи для роботи з базами знань;
- моделі;
- команди і запити;
- класи для роботи з базою даних;
- перелічення;
- класи–фабрики;

- класи системи локалізації;
- класи, що використовуються системою генерації питань.

3.3.2.1 Контролери

Основними функціональними одиницями у серверній частині є контролери, саме вони виконують основні функції, які надає навчальна платформа.

Варто зазначити, що всі контролери містять мінімум логіки відповідно до патерну проєктування медіатор. Єдине що вони роблять: відправляють запити до медіатора і повертають відповіді клієнтам.

У вебсервері навчальної платформи наявні такі контролери:

а) `AuthController` – контролер, який надає функції аутентифікації і реєстрації. Методи:

- 1) `Login` – представляє ендпоїнт аутентифікації;
- 2) `Register` – представляє ендпоїнт реєстрації;

б) `LevelsController` – контролер рівнів. Методи:

- 1) `GetLevels` – представляє ендпоїнт для отримання рівнів по унікальному ідентифікатору навчального етапу;
- 2) `GetLevel` – представляє ендпоїнт доступу до рівня по його унікальному ідентифікатору;
- 3) `TryLevel` – представляє ендпоїнт для реєстрації спроб проходження рівнів;

в) `StagesController` – контролер навчальних етапів. Методи:

- 1) `GetStages` – представляє ендпоїнт для отримання навчальних етапів заданої унікальним ідентифікатором дисципліни

г) `SubjectsController` – контролер дисциплін. Методи:

- 1) `GetSubjects` – представляє ендпоїнт для отримання повного списку дисциплін навчальної платформи.

3.3.2.2 Арі-класи

Арі-класи необхідні для визначення типів запитів і відповідей вебсерверу. Ці класи використовує як вебсервер так і користувацький інтерфейс.

Список арі-класів наведено у таблиці 3.3.

Таблиця 3.3 – Арі-класи навчальної платформи

Клас	Призначення класу	Поле	Опис
1	2	3	4
AuthApi ResponseDto	Арі-відповідь для аутентифікації і реєстрації	Email: string	Електронна адреса
		Name: string	Ім'я користувача
		Token: string	JWT токен
LogInApi RequestDto	Арі-запит для аутентифікації	Email: string	Електронна адреса
		Password: string	Пароль
RegisterRequest Dto	Арі-запит для реєстрації	Email: string	Електронна адреса
		Password: string	Пароль
		Name: string	Ім'я
GetLevelApi ResponseDto	Арі-відповідь для отримання згенерованого рівня	Questions: List<QuestionApi Dto>	Список питань
GetLevelsApi ResponseDto	Арі-відповідь для отримання списку рівнів	Levels: List<LevelApiDto>	Список рівнів
LevelApiDto	Арі-представлення рівня навчальної платформи	Id: long	Унікальний ідентифікатор
		Name: string	Назва рівня
		NumberOfQuestions: int	Кількість питань

Продовження таблиці 3.3

1	2	3	4
		TimeLimit: DateTime	Ліміт часу
		IsCompleted: bool	Чи пройдений рівень
		IsAvailable: bool	Чи доступний рівень
		Tries: List<LevelTryApi Dto>	Список спроб проходження рівня
LevelTryApiDto	Арі– представлення для спроби проходження рівня	CompletedQuestions: int	Кількість правильних відповідей на питання
		TriedAt: DateTime	Дата і час спроби
QuestionApiDto	Арі– представлення питання рівня	Text: string	Текст питання
		AcceptableAnswers: List<string>	Список правильних відповідей
		AnswersVariants: List<string>	Список варіантів відповідей
TryLevelApi Request	Арі–запит для реєстрації спроби проходження рівня	NumberOfCompleted Questions: int	Кількість правильних відповідей на питання рівня
		TriedAt: DateTime	Дата і час спроби
TryLevelApi Response	Арі–відповідь для результатів спроби проходження рівня	IsCompleted: bool	Чи пройдений рівень
		LevelTry: LevelTryApiDto	Об’єкт спроби
		AvailableLevels: List<LevelApiDto>	Список доступних рівнів після спроби
		SubjectProgress: SubjectProgressApi Dto	Прогрес по дисципліні
		StageProgress: StageProgressApiDto	Прогрес по навчальному етапу

Продовження таблиці 3.3

1	2	3	4
GetStagesApiResponseDto	Арі-відповідь для отримання навчальних етапів дисципліни	Stages: List<StageApiDto>	Список навчальних етапів
StageApiDto	Арі-представлення навчального етапу дисципліни	Id: long	Унікальний ідентифікатор
		Name: string	Назва навчального етапу
		IsAvailable: bool	Чи доступний
		Progress: StageProgressApiDto	Об'єкт прогресу навчального етапу
StageProgressApiDto	Арі-представлення прогресу користувача по навчальному етапу	Value: float	Значення прогресу
GetSubjectsApiResponseDto	Арі-відповідь для отримання дисциплін навчальної платформи	Subjects: List<SubjectApiDto>	Список дисциплін навчальної платформи
SubjectApiDto	Арі-представлення дисципліни навчальної платформи	Id: long	Унікальний ідентифікатор
		Name: string	Назва дисципліни
		IsAvailable: bool	Чи доступна дисципліна
		Progress: SubjectProgressApiDto	Об'єкт прогресу дисципліни
SubjectProgressApiDto	Арі-представлення прогресу користувача по дисципліні	Value: float	Значення прогресу

3.3.2.3 Класи конфігурації

Класи конфігурації необхідні для організації зручного зберігання і доступу до конфігурації проєкту. Всі класи конфігурації реалізують інтерфейси з відповідними назвами. Робота з конфігурацією реалізована за допомогою засобів фреймворку ASP.NET Core.

Основним класом конфігурації є `ApplicationConfig` (реалізує `IApplicationContext`). Цей клас надає доступ до інших, більш малих, блоків конфігурації, представлених окремими класами:

а) `DatabaseConfig` (реалізує `IDatabaseConfig`) – цей клас містить поля для конфігурації бази даних, а саме:

- 1) `ConnectionString` – строка підключення до бази даних;
- 2) `Retries` – кількість спроб підключення;
- 3) `Timeout` – допустимий час відповіді бази даних;

б) `JwtConfig` (реалізує `IJwtConfig`) – цей клас містить дані, необхідні для роботи з Jwt токенами, а саме:

- 1) `Issuer` – видавник токenu;
- 2) `Audience`;
- 3) `Authority`;
- 4) `Key` – ключ для генерації.

3.3.2.4 Класи для роботи з базами знань

Цей тип класів є необхідним для організації зв'язку і обміну даними між сервером і базами знань. Так як на даному етапі розробки навчальна платформа надає доступ користувачам тільки до дисципліни “Хімія”, до цього типу класів належить тільки один клас.

Клас ChemistryKnowledgeBase (реалізує інтерфейс IChemistryKnowledgeBase) містить методи для зв'язку з базою знань з хімії. Ці методи:

- GetUnit – використовується для отримання метрики по величині, яку ця метрика вимірює;
- GetMeasure – використовується для отримання величини по вказаній метриці;
- HowMany – використовується для запиту на визначення кількості однієї метрики в іншій з використанням префіксів;
- Compare – використовується для порівняння різних кількостей сутностей;
- GetLarger – використовується для отримання більшої сутності з двох заданих. Під капотом використовує метод Compare;
- GetSmaller – використовується для отримання меншої сутності з двох заданих. Під капотом використовує метод Compare.

3.3.2.5 Моделі

Моделі являють собою класи сутностей, які широко використовуються для різних цілей в інтерактивній навчальній платформі. Класи моделей з детальним описом наведені в таблиці 3.4.

3.3.2.6 Команди і запити

Цей тип класів представлений класами, які реалізують інтерфейси бібліотеки MediatR `IRequest<T>` і `IRequestHandler<IRequest<T>, T>`. Ці класи необхідні для організації бізнес логіки з використанням патерну “Медіатор”. Містять основну логіку програми.

Таблиця 3.4 – Класи моделей

Клас	Призначення класу	Поле	Опис
JwtTokenInfoDto	Модель для передачі даних для генерації JWT токена	UserId: string	Унікальний ідентифікатор користувача
		UserEmail: string	Електронна адреса користувача
		ConfigKey: string	Ключ, для генерації JWT токена (береться з файлу конфігурації)
RegisterDto	Модель для передачі даних для реєстрації	Email: string	Електронна адреса користувача
		Password: string	Пароль від облікового запису користувача
		Name: string	Ім'я користувача
Question	Клас, який являє собою модель питання	Text: string	Текст питання
		AcceptableAnswers: List<string>	Список правильних відповідей
		AnswersVariants: List<string>	Список варіантів відповідей
MetricNumber	Клас, який являє собою модель метричного числа. Широко використовується для генерації питань для навчального етапу “Метрики і аналіз розмірностей”	Value: double	Числове значення
		Prefix: Prefix	Префікс (наприклад, “кіло”, “мега”, “мілі”)

У розробленій навчальній платформі команди – функціональні програмні одиниці, які вносять деякі зміни до бази даних; запити ж відрізняються від команд тим, що вони просто повертають деяку вибрану з бази даних інформацію.

Команди навчальної платформи:

а) `LogInCommand : IRequest<ActionResult>` – команда, обробник якої (`LogInCommandHandler: IRequestHandler<LogInCommand, ActionResult>`) містить логіку аутентифікації користувача. Команда містить такі параметри:

- 1) `Email` – електронна адреса користувача;
- 2) `Password` – пароль облікового запису користувача;

б) `RegisterCommand: IRequest<ActionResult>` – команда, обробник якої (`RegisterCommandHandler: IRequestHandler<RegisterCommand, ActionResult>`) містить логіку реєстрації користувача. Команда містить такі параметри:

- 1) `Email` – електронна адреса користувача;
- 2) `Password` – пароль облікового запису користувача;
- 3) `Name` – ім'я користувача;

в) `TryLevelCommand: IRequest<ActionResult>` – команда, обробник якої (`TryLevelCommandHandler: IRequestHandler<TryLevelCommand, ActionResult>`) містить логіку реєстрації спроби проходження рівня користувачем. Команда містить такі параметри:

- 1) `LevelId` – унікальний ідентифікатор рівня;
- 2) `NumberOfCompletedQuestions` – кількість правильних відповідей на питання рівня;
- 3) `TriedAt` – дата і час спроби.

Запити навчальної платформи:

а) `GetLevelQuery: IRequest<ActionResult>` – запит, обробник якого (`GetLevelQueryHandler: IRequestHandler<GetLevelQuery, ActionResult>`) генерує і повертає рівень. Запит містить такі параметри:

- 1) `LevelId` – унікальний ідентифікатор рівня;

2) Locale – локаль;

б) GetLevelsQuery: IRequest<IActionResult> – запит, обробник якого (GetLevelsQueryHandler: IRequestHandler<GetLevelsQuery, IActionResult>) повертає рівні навчального етапу. Запит містить такі параметри:

1) StageId – унікальний ідентифікатор навчального етапу;

в) GetStagesQuery: IRequest<IActionResult> – запит, обробник якого (GetStagesQueryHandler: IRequestHandler<GetStagesQuery, IActionResult>) повертає навчальні етапи дисципліни. Запит містить такі параметри:

1) SubjectId – унікальний ідентифікатор дисципліни;

г) GetSubjectsQuery: IRequest<IActionResult> – запит, обробник якого (GetSubjectsQueryHandler: IRequestHandler<GetSubjectsQuery, IActionResult>) повертає дисципліни навчальної платформи. Запит не містить параметрів

3.3.2.7 Класи для роботи з базою даних

Цей тип класів призначений для роботи з базою даних. Більш детально про ці класи буде написано у наступному розділі, тут будуть згадані деякі деталі і нюанси.

Для кожної моделі сутності (перелічені у наступному розділі) представлений спеціальний клас–конфігуратор, який виконує функцію налаштування проєкції моделі у базу даних (наприклад, саме тут вказуються зв'язки сутностей і індексовані поля). Список цих класів: LevelConfiguration, LevelProgressConfiguration, LevelTryConfiguration, StageConfiguration, StageProgressConfiguration, SubjectConfiguration, SubjectProgressConfiguration, UserConfiguration. Всі ці класи реалізують інтерфейс IEntityConfiguration<T> (EntityFramework). Логіка співставлення класів з конфігураціями з відповідними моделями визначена у класі EntityConfigurator.

3.3.2.8 Перелічення

Перелічення у мові програмування C# це насправді окремий тип даних. В навчальній платформі створений ряд перелічень, які дозволяють значно сильно прискорити і облегшити розробку застосунку. Перелічення навчальної платформи приведені у таблиці 3.5.

Таблиця 3.5 – Перелічення навчальної платформи

Перелічення	Опис перелічення	Значення	Опис значення
1	2	3	4
Comparison	Перелічення, що використовується для порівнянь кількостей сутностей	Less	Менше
		Bigger	Більше
		Equal	Рівно
Locale	Перелічення, що використовується системою локалізації для задання мови	Ua	Українська мова
		En	Англійська мова
Measure	Перелічення для представлення типу виміру	Length	Довжина
		Mass	Маса
		Time	Час
		Quantity	Кількість речовини
Power	Перелічення, що використовується для представлення математичного ступеню	First	Перший
		Second	Другий
		Third	Третій
Prefix	Перелічення, що використовується для представлення префіксу вимірювання	Pico	Пікно
		Nano	Нано
		Milli	Міллі
		None	—

Продовження таблиці 3.5

1	2	3	4
Unit	Перелічення, що використовується для представлення метрики	Kilo	Кіло
		Giga	Гіга
		Meter	Метер
		Gram	Грам
		Second	Секунда
		Mole	Моль

3.3.2.9 Класи-фабрики

До цього типу класів належать класи, які реалізують патерн проєктування “Фабрика”. Шаблон проєктування "Фабрика" (Factory) відноситься до патернів створення (creational patterns) і дозволяє створювати об'єкти певного типу, не розкриваючи деталей процесу їх створення. Він забезпечує абстракцію між клієнтом і об'єктами, що створюються, дозволяючи легко змінювати тип створюваних об'єктів без впливу на клієнтський код. Класи-фабрики у навчальній платформі: статичні, містять лише один метод Create.

Класи-фабрики застосунку:

- LocaleFactory – фабрика для створення об'єкту перелічення Locale використовуючи звичайний рядок (string);
- UserFactory – фабрика для створення моделі користувача, яка потім буде занесена до бази даних.

3.3.2.10 Класи системи локалізації

Підтримка двох мов інтерфейсу – одна з ключових вимог, вказаний у технічному завданні по створенню навчальної платформи. Вебсервер

повинен генерувати питання для рівнів на двох мовах: українській і англійській. При генерації питань широко використовуються перелічення, які описані трішки вище. Тому система локалізації налаштована саме на переклад перелічень. Основними класами у такій системі локалізації є `EnumTranslationsService` та `EnumAnswersTranslationsService`, які повертають переклади для перелічень і для списків відповідей, утворених з перелічень. Ці сервіси керують класами, які успадковують `EnumTranslator` і `EnumAnswersTranslator` (абстрактні класи) відповідно. Така системна організація дозволяє отримати уніфіковано отримати переклад для перелічення чи списку відповідей, бо можна, наприклад, просто викликати метод цього класу `GetUkrTranslation(Enum enumValue)` (чи `GetEnTranslation`) передавши замість аргументу перелічення інше перелічення будь якого типу.

Класи, які успадковують `EnumTranslator`:

- `ComparisonTranslator` – переклади для перелічення `Comparison`;
- `MeasureTranslator` – переклади для перелічення `Measure`;
- `PrefixTranslator` – переклади для перелічення `Prefix`;
- `UnitTranslator` – переклади для перелічення `Unit`.

Класи, які успадковують `EnumAnswersTranslator`:

- `MeasureAnswersTranslator` – переклади для списків відповідей, утворених з використанням перелічення `Measure`;
- `UnitAnswersTranslator` – переклади для списків відповідей, утворених з використанням перелічення `Unit`.

3.3.2.11 Класи, що використовуються системою генерації питань

Як вже було не раз зазначено: система генерації питань є основною системою навчальної платформи, яка утворена зі складної розгалуженої мережі класів.

Питання рівнів утворені з шаблонів. Шаблони – це окремі класи, які містять переклади текстів питань на підтримувані платформою мови. Тексти

питань організовані таким чином, що вони містять спеціально відведені “комірки” для підстановок параметрів питань. Класи шаблонів питань, наявних в навчальній платформі: `CompareQuestionTemplate`, `GetMeasureQuestionTemplate`, `GetUnitQuestionTemplate`, `HowManyQuestionTemplate`. Всі ці класи реалізують інтерфейс `IQuestionTemplate`. Централізований доступ до шаблонів і їх перекладів організований за допомогою спеціального сервісу `QuestionTemplateService`.

Для генерації питань на основі шаблонів використовуються спеціальні класи–генератори. Загальна реалізація для генератора питання визначена в класі `QuestionGenerator`. Кожна дисципліна містить свої типи питань, і деяка логіка генерації може різнитися між різними дисциплінами, тому `QuestionGenerator` успадковується іншим класом, специфічним до певної дисципліни. Так як на даному етапі розвитку і розробки навчальної платформи представлена тільки дисципліна “Хімія”, то `QuestionGenerator` успадковується класом `ChemistryQuestionGenerator`, який містить додаткову логіку для доступу до бази знань з хімії. Класи генераторів питань навчального етапу, який належить дисципліні “Хімія”, “Метрики і аналіз розмірностей” успадковують `ChemistryQuestionGenerator`. Ці класи: `CompareQuestionGenerator`, `GetMeasureQuestionGenerator`, `GetUnitQuestionGenerator`, `HowManyQuestionGenerator` – містять логіку для генерації окремих типів питань.

Вся загальна “рівнева” логіка винесена в абстрактний клас `QuestionsDirector`. Саме в ньому визначена послідовність отримання всіх можливих генераторів питань для рівня, їх перемішування і безпосередньо виклик генерації і повернення вже готових наборів питань для визначеного рівня. `QuestionsDirector` успадковується класами, які представляються окремі рівні навчальних етапів. Для навчального етапу “Метрики і аналіз розмірностей” цими класами є: `Mda0`, `Mda1`, `Mda2`, `Mda2A`, `Mda2B`, `Mda3`, `Mda4`, `Mda4A`, `Mda4B`. В цих класах міститься безпосередньо специфічна для кожного рівня логіка: які питання генерувати і як їх генерувати.

На вершині ієрархії системи генерації питань стоїть клас `QuestionSetGenerator`, який реалізує інтерфейс `IQuestionSetGenerator`. Задачею цього класу є в залежності від рівня і обраної мови згенерувати набір питань. Саме цей клас містить логіку співставлення класів, які успадковують `QuestionDirector` з відповідними їм рівнями.

3.4 Проєктування бази даних

Як вже було сказано у минулих розділах: вся інформація навчальної платформи зберігається у базі даних, яка знаходиться під керуванням СКБД `MS SQL Server`. Програмний доступ до бази даних організований з використанням `Entity Framework Core 6`.

Під час проєктування був розроблений ряд моделей, які забезпечують організацію даних у базі. Список цих моделей наведений нижче:

а) `Entity` – модель базової сутності у базі даних. Всі сутності платформи використовують її як базову. Містить властивості:

- 1) `Id (long)` – унікальний ідентифікатор;
- 2) `CreatedAt (DateTime)` – дата і час створення;
- 3) `LastUpdatedAt (DateTime)` – дата і час останнього оновлення сутності;

б) `User` – модель користувача навчальної платформи. Містить властивості:

- 1) `Email (string)` – електронна адреса;
- 2) `Name (string)` – ім'я користувача;
- 3) `PasswordHash (string)` – геш паролю;
- 4) `PasswordSalt (string)` – соль для генерування гешу паролю;

в) `Subject` – модель дисципліни навчальної платформи. Містить наступні властивості:

- 1) `Name (string)` – назва дисципліни;

2) `IsAvailable (bool)` – значення, яке відображає доступність дисципліни у навчальній платформі;

г) `SubjectProgress` – модель, яка представляє собою кількісне значення прогресу певного користувача з певної дисципліни. Містить наступні властивості:

- 1) `SubjectId (long)` – унікальний ідентифікатор дисципліни;
- 2) `Subject (Subject)` – об'єкт дисципліни;
- 3) `UserId (long)` – унікальний ідентифікатор користувача;
- 4) `User (User)` – об'єкт користувача;
- 5) `Progress (float)` – кількісне відображення прогресу;

д) `Stage` – модель навчального етапу, пов'язана з моделлю дисципліни зв'язком “багато–до–одного”. Містить властивості:

- 1) `SubjectId (long)` – унікальний ідентифікатор дисципліни;
- 2) `Subject (Subject)` – об'єкт дисципліни;
- 3) `Name (string)` – назва навчального етапу;
- 4) `IsAvailable (bool)` – значення, яке відображає доступність навчального етапу у платформі;

е) `StageProgress` – модель, яка представляє собою кількісне значення прогресу певного користувача з певного навчального етапу. Містить наступні властивості:

- 1) `StageId (long)` – унікальний ідентифікатор навчального етапу;
- 2) `Stage (Stage)` – об'єкт навчального етапу;
- 3) `UserId (long)` – унікальний ідентифікатор користувача;
- 4) `User (User)` – об'єкт користувача;
- 5) `Progress (float)` – кількісне відображення прогресу з навчального етапу;

ж) `Level` – модель, яка представляє собою рівень навчальної платформи, пов'язана з моделлю етапу навчання зв'язком “багато–до–одного”. Містить властивості:

1) StageId (long) – унікальний ідентифікатор навчального етапу;

2) Stage (Stage) – об'єкт навчального етапу;

3) Name (string) – назва навчального етапу;

4) NumberOfQuestions (int) – кількість питань рівня;

з) LevelProgress – модель, яка визначає чи пройшов певний користувач певний рівень. Містить властивості:

1) LevelId (long) – унікальний ідентифікатор рівня;

2) Level (Level) – об'єкт рівня;

3) UserId (long) – унікальний ідентифікатор користувача;

4) User (User) – об'єкт користувача;

5) IsCompleted (bool) – значення, яке є відображенням проходження рівня;

и) LevelTry – модель, яка є представленням спроби проходження певним користувачем певного рівня і містить відповідну щодо цього інформацію. Має такі властивості:

1) LevelId (long) – унікальний ідентифікатор рівня;

2) Level (Level) – об'єкт рівня;

3) UserId (long) – унікальний ідентифікатор користувача;

4) User (User) – об'єкт користувача;

5) CompletedQuestions (int) – кількість питань, на які користувач відповів правильно;

6) TriedAt (DateTime) – дата і час спроби проходження рівня.

Так як для взаємодії з базою даних був використаний EntityFramework, то під час розробки не було необхідності створювати всі таблиці вручну – фреймворк все це робив автоматично за допомогою механізму міграцій. Міграція у широкому сенсі – це керування інкрементними, зворотніми змінами схеми реляційної бази даних. Міграція схеми виконується над базою даних коли необхідно оновити, або повернути схему бази даних до якоїсь більш нової чи старої версії.

Міграції виконуються програмно за допомогою інструменту міграцій. При виклику інструменту міграції з вказанням бажаної версії схеми, інструмент автоматично застосовує або відкачує міграції в правильній послідовності аж поки не приведе базу даних до бажаного стану.

Централізований доступ до бази даних був організований з використанням контексту даних (у навчальній платформі представлений класом `StemLearningDbContext`).

Для полегшення роботи з базою даних був створений ряд класів–репозиторіїв, які містять функції для виконання типових дій з різними сутностями. Було створено 4–и репозиторія:

а) `UsersRepository` – використовується для організованої роботи з сутністю користувача (`User`). Містить наступні методи:

- 1) `CurrentUser` – отримання поточного користувача;
- 2) `User` – отримання користувача за унікальним ідентифікатором;

б) `SubjectsRepository` – використовується для організованої роботи з сутністю дисципліни (`Subject`). Містить наступні методи:

- 1) `Subjects` – отримання всіх дисциплін;
- 2) `Subject` – отримання дисципліни за унікальним ідентифікатором;
- 3) `GetOrCreateSubjectProgress` – отримання чи створення об'єкту прогресу дисципліни для поточного користувача;
- 4) `CreateOrUpdateSubjectProgress` – створення чи оновлення об'єкту прогресу дисципліни для поточного користувача;

в) `StageRepository` – використовується для організованої роботи з сутністю навчального етапу (`Stage`). Містить наступні методи:

- 1) `Stages` – отримання всіх навчальних етапів за унікальним ідентифікатором дисципліни;
- 2) `Stage` – отримання навчального етапу за допомогою його унікального ідентифікатора;

- 3) `GetOrCreateStageProgress` – отримання чи створення об'єкту прогресу навчального етапу для поточного користувача;
- 4) `CreateOrUpdateStageProgress` – створення чи оновлення об'єкту прогресу навчального етапу для поточного користувача;
- г) `LevelRepository` – використовується для організованої роботи з сутністю рівня (`Level`). Містить наступні методи:
 - 1) `Levels` – отримання всіх рівнів навчального етапу за унікальним ідентифікатором навчального етапу;
 - 2) `Level` – отримання об'єкту рівня за його унікальним ідентифікатором;
 - 3) `GetOrCreateLevelProgress` – отримання чи створення прогресу рівня для поточного користувача;
 - 4) `CreateOrUpdateLevelProgress` – створення чи оновлення прогресу рівня для поточного користувача;
 - 5) `LevelTries` – отримання списку спроб для певного рівня за його унікальним ідентифікатором для поточного користувача;
 - 6) `TryLevel` – реєстрація спроби проходження рівня за його унікальним ідентифікатором для поточного користувача. Також цей метод включає логіку оновлення прогресів дисципліни і навчального етапу, до який відноситься цей рівень, при його проходженні для поточного користувача.

3.5 Основні рішення щодо розробки графічного інтерфейсу

Технічне завдання містить ряд вимог до графічного інтерфейсу користувача. У наступних підрозділах будуть розглянуті основні рішення щодо розробки графічного інтерфейсу.

3.5.1 Дизайн

Варто зазначити, що на продумування і створення дизайну користувацького інтерфейсу навчальної платформи було витрачено найбільшу кількість часу. Достатньо складно мислити креативно і вигадувати графічну оболочку, коли ти програміст. Але після багатьох десятків годин спроб, переробок і творчих криз все ж таки вдалось зробити інтерактивний, цікавий і живий інтерфейс, який буде тільки сприяти вдосконаленню навчального процесу кожного користувача платформи.

Основною тематикою дизайну була обрана шкільна атмосфера. Для демонстрації розглянемо головне меню. Воно представлене у вигляді двох крейдових дошок, монітору та шкафу з полицями. Всі інші елементи меню: будь то текст чи картинки – повністю відповідають загальній обстановці. Наприклад, перший екран головного меню (рис. 3.7), який зустрічає користувача після вдалої аутентифікації чи реєстрації являє собою крейдову дошку зеленого кольору, всі написи на якій виконані крейдою, всі картинки підібрані таким чином, що здається, що вони намальовані “від руки”, а для блоку з вибором мови виділено спеціальне місце, фоном якого є “брудна” від білої крейди зона дошки. Всі ці деталі додають цікавості інтерфейсу користувача: його хочеться досліджувати і ним приємно користуватися.

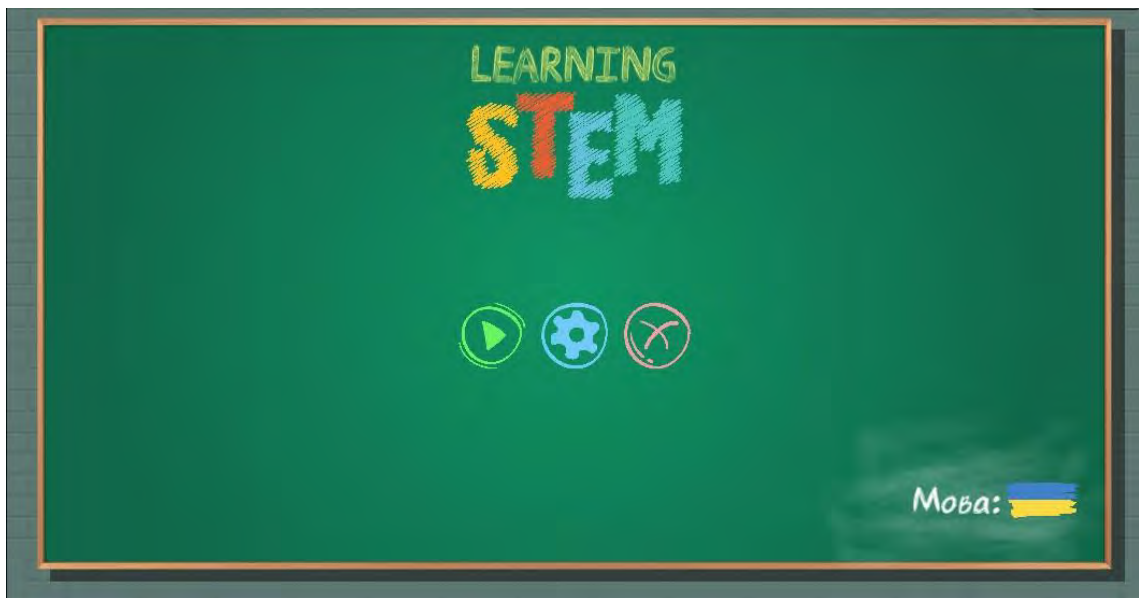


Рисунок 3.7 – Перший екран головного меню навчальної платформи

Всього застосунок користувацького інтерфейсу складається з 12 різноманітних унікальних екранів. Така велика кількість екранів обумовлена необхідністю зробити інтерфейс максимально простим і непереобтяженим надлишковою інформацією, тому кожен екран містить лише специфічну до деякої ситуації інформацію і логіку.

Інтерфейс прямолінійний, всі контролери знаходяться в інтуїтивних місцях, тому користувачі незалежно від кількості досвіду роботи з комп'ютерними пристроями можуть ним користуватися абсолютно без проблем. Кольори інтерфейсу підібрані таким чином, щоб все було природньо і гармонійно поєднувалось між собою.

Користувач отримує необхідну інформацію не тільки за допомогою текстів і картинок. Наприклад, за рахунок інтерактивних індикаторів користувач може зрозуміти чи правильно він відповів на питання (рис. 3.8, рис. 3.9, рис. 3.10), або чи доступен рівень для проходження (рис. 3.11, рис. 3.12, рис. 3.13). Також варто відзначити, що було приділено увагу кольоровим акцентам користувацького інтрефейсу.



Рисунок 3.8 – Вид індикатора відповіді на питання за замовчуванням



Рисунок 3.9 – Вигляд індикатора відповіді на питання, коли дана правильна відповідь на питання



Рисунок 3.10 – Вигляд індикатора відповіді на питання, коли дана неправильна відповідь на питання

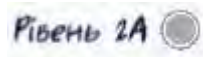


Рисунок 3.11 – Індикатор, який показує, що рівень доступен для проходження

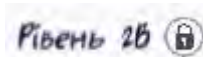


Рисунок 3.12 – Індикатор, який показує, що рівень недоступен для проходження

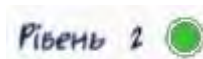


Рисунок 3.13 – Індикатор, який показує, що рівень пройдено

Так, наприклад, недоступні елементи інтерфейсу позначаються сірим кольором (рис. 3.14, рис. 3.15, рис. 3.16), а різний колір тексту в повідомленні, яке бачить користувач після проходження рівня, сприяє

усвідомленню і розумінню результату спроби додатково до напису (рис. 3.17, рис. 3.18).



Рисунок 3.14 – Виділення сірим кольором недоступної дисципліни



Рисунок 3.15 – Виділення сірим кольором недоступного навчального етапу



Рисунок 3.16 – Виділення сірим кольором неотриманих досягнень (предметів)

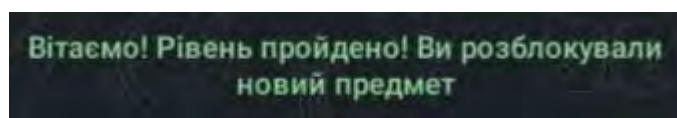


Рисунок 3.17 – Кольоровий акцент тексту, який з'являється після проходження рівня і дає зрозуміти, що рівень пройдено

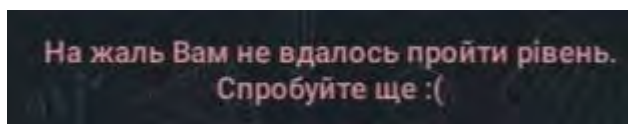


Рисунок 3.18 – Кольоровий акцент тексту, який з'являється після проходження рівня і дає зрозуміти, що рівень не пройдено

Варто також зазначити, що багато уваги приділено гейміфікації користувацького інтерфейсу. Наприклад, реалізована логіка нагородження користувача за успішне проходження рівня: він отримує винагороду у вигляді предмету однієї з STEM дисциплін (наприклад, хімія – колба з реактивом). Отримання нагороди і відображення нагород користувача представлені на рисунках 3.19, 3.20 відповідно.

Зворотній зв'язок користувача і інтерфейсу реалізований за допомогою діалогових вікон. Приклад діалогового вікна навчальної платформи наведений на рисунку 3.21.

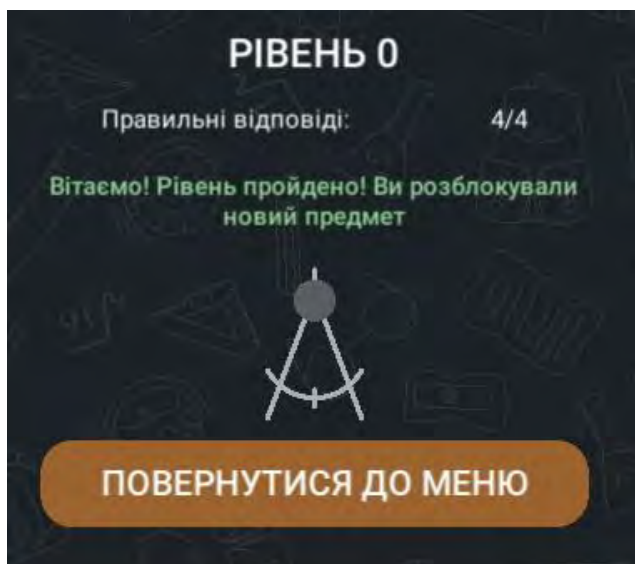


Рисунок 3.19 – Отримання нагороди



Рисунок 3.20 – Відображення отриманої нагороди

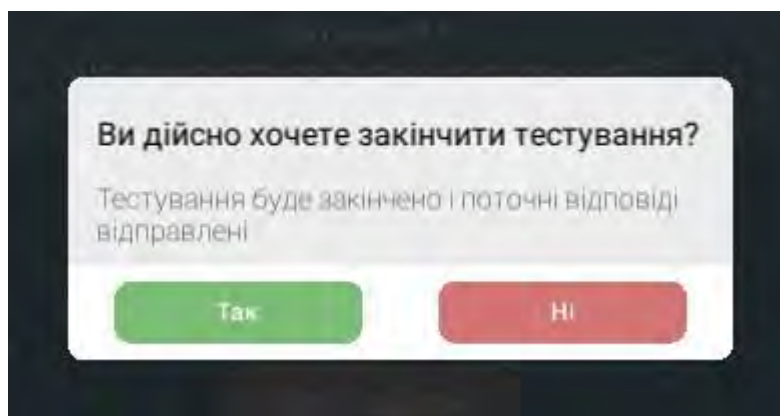


Рисунок 3.21 – Діалогове вікно, яке з'являється при примусовому закінченні спроби проходження рівня

Користувацький інтерфейс містить ряд анімацій і переходів, що роблять його живим, а не нудним і статичним. При переході з одного екрану головного меню до іншого відбувається анімація заміщення предмету, на якому відображена інформація (будь то дошка, монітор, чи шкаф з полицками) наступним. Всі кнопки змінюють розмір і колір при наведенні курсору, а такі компоненти як таймер інтерактивно змінюються кожному секунду.

Всі екрани застосунку і їх функції приведені у розділі 5 (“Керівництво оператора”).

3.5.2 Адаптивність

Однією з вимог при проектуванні графічного інтерфейсу була його адаптивність. Тобто користувацький інтерфейс повинен однаково гарно відображатися на пристроях з різними розмірами і форматами дисплеїв.

Адаптивність реалізована за рахунок різних компоновок елементів і їх позиціонування, використовуючи засоби платформи Unity.

Приклади адаптивності навчальної платформи наведено на рис. 3.22 – рис. 3.24.

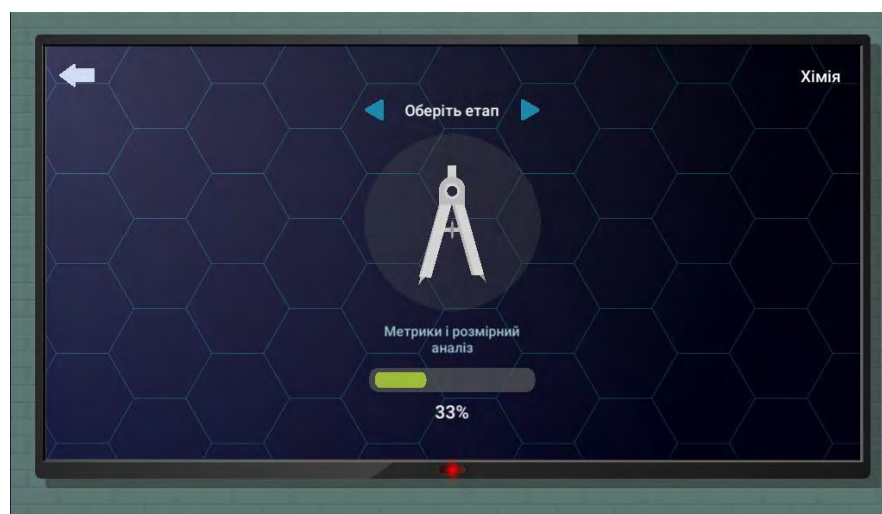


Рисунок 3.22 – Меню вибору навчального етапу. Роздільна здатність екрану 1920x1080 пікселів

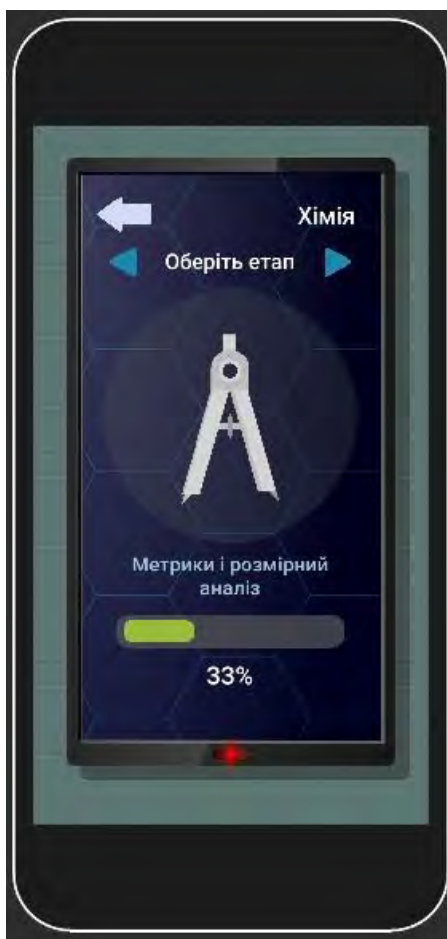


Рисунок 3.23 – Меню вибору навчального етапу. Відображення на мобільному пристрої



Рисунок 3.24 – Екран з досягненнями користувача. Роздільна здатність екрану 1920x1080 пікселів

Також приклад адаптивності навчальної платформи наведено на рис. 3.25.



Рисунок 3.25 – Екран з досягненнями користувача. Відображення на мобільному пристрої

3.5.3 Локалізація

Вибір Unity в якості платформи для розробки користувацького інтерфейсу зробив можливим використання зручної вбудованої системи локалізації для додавання перекладів застосунку інтерфейсу на різні мови. Необхідно було вручну додати переклади для необхідних мов до таблиць локалізації, а потім ці значення прив'язувати до графічних компонентів.

В графічному інтерфейсі користувача передбачені спеціальні кнопки, натискаючи на які користувач може змінити мову інтерфейсу (рис. 3.26, рис. 3.27). Мова інтерфейсу змінюється миттєво, без будь яких затримок.

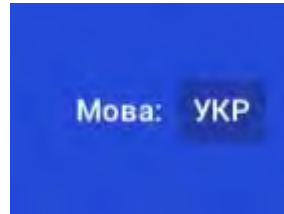


Рисунок 3.26 – Спеціальна кнопка зміни мови застосунку, яка знаходиться на екрані аутентифікації/реєстрації.



Рисунок 3.27 – Спеціальна кнопка зміни мови застосунку, яка знаходиться на екрані головного меню.

3.5.4 Кросплатформеність

Unity дозволяє компілювати отриманий застосунок для різних видів платформ, на яких він буде працювати однаково гарно: Desktop (Windows, Linux, Mac), Mobile (IOS, Android), Web (WebGL).

4 КЕРІВНИЦТВО ПРОГРАМІСТА

У даному розділі розглянуто призначення та умови застосування навчальної платформи, характеристики навчальної платформи, звертання до неї, вхідні та вихідні дані, а також повідомлення.

4.1 Призначення та умови застосування програми

Інтерактивна навчальна платформа була створена для полегшення вивчення STEM дисциплін. Ця платформа може стати корисною для будь-якого бажаного.

4.2 Характеристики програми

У цьому підрозділі окремо розглянемо структуру проєкту користувацького інтерфейсу і вебсерверу. У сутності обидва програмні застосунки є програмним рішеннями (solutions) створеними з використанням мови програмування C#.

4.2.1 Вебсервер

Структура проєкту вебсерверу відображена на рис. 4.1.

Проєкт StemLearning.Shared використовується для класів запитів і відповідей API платформи використовується як самим вебсервером так і користувацьким інтерфейсом.

Основним проєктом вебсерверу є StemLearning.WebApi що містить всю програмну логіку навчальної платформи. Класи, які зберігаються у цьому проєкті детально описані у 3-ому розділі цієї дипломної роботи. Варто відмітити структуру директорій проєкту:

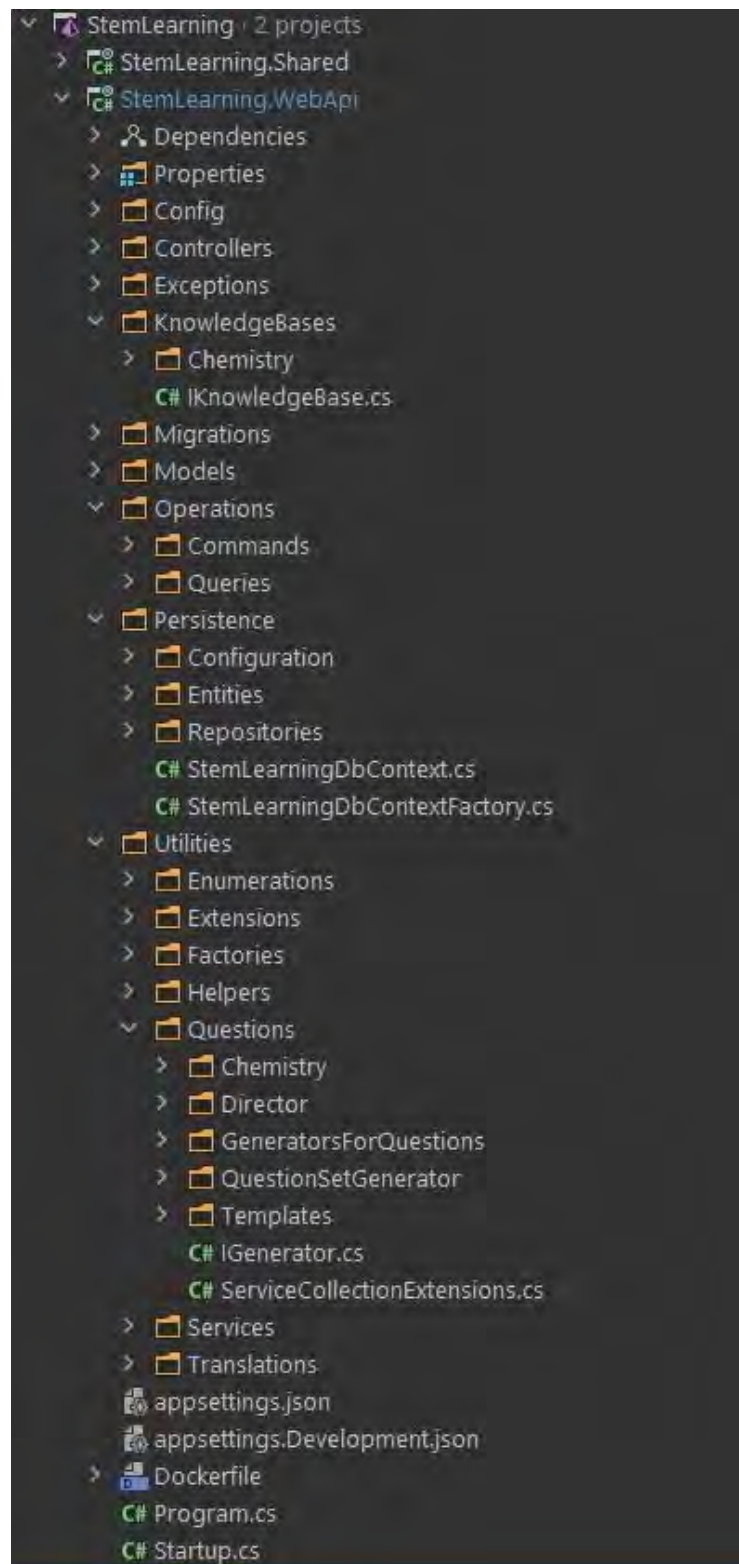


Рисунок 4.1 – Структура проєкту вебсерверу

- Config – директорія, в якій міститься необхідні для конфігурації класи;
- Controllers – папка, що містить контролери вебсерверу;

- Exceptions – класи виключень, які можуть бути “викинуті” застосунком;
- KnowledgeBases – класи необхідні для роботи з базами знань (на момент написання дипломної роботи тут містяться тільки файли необхідні для бази знань дисципліни хімія);
- Migrations – класи міграції бази даних;
- Models – директорія, що містить моделі (класи, які використовуються самим застосунком для різноманітних цілей);
- Operations – папка, в якій містяться класи, що представляють собою запити і команди, а також їх обробники;
- Persistence – класи, для конфігурування і роботи з базою даних. Також тут містяться класи сутностей бази даних;
- Utilities – директорія, в якій міститься ряд інших папок з корисними і необхідними класами. Саме тут зберігаються перелічення, класи-розширення, класи-фабрики, допоміжні класи, сервіси, класи необхідні для роботи системи локалізації і звісно класи для генерації рівнів і роботи з питаннями.

Конфігурація вебсерверу визначається в файлах `appsettings.json` та `appsettings.Development.json`. Також існує можливість визначати конфігураційні змінні використовуючи секрети користувача (`user secrets`) – ця функціональність надається і підтримується фреймворком ASP.NET Core.

4.2.2 Користувацький інтерфейс

Структура проекту користувацького інтерфейсу відображена на рис. 4.2.

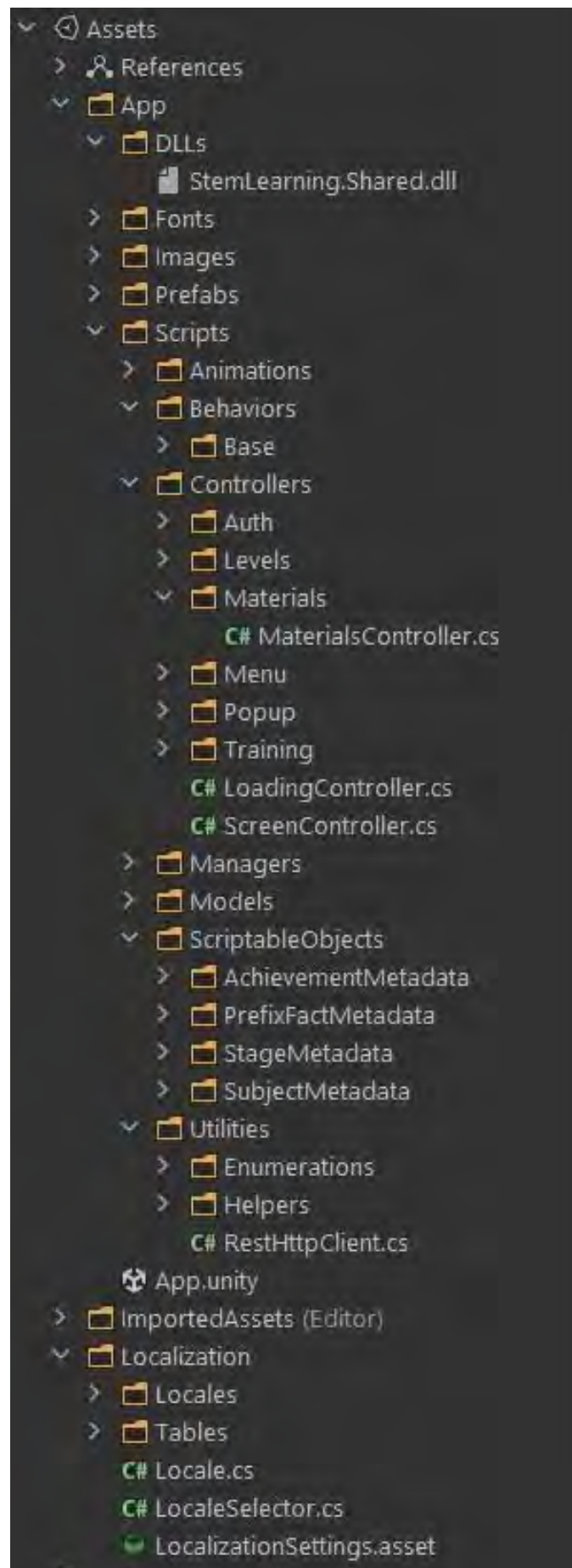


Рисунок 4.2 – Структура проєкту користувацького інтерфейсу

Структура директорій проєкту організована наступним чином:

а) App – ключова директорія проєкту, що містить більшість файлів необхідних для нормального функціонування користувацького інтерфейсу навчальної платформи. Містить папки:

- 1) DLLs – імпортовані бібліотеки для мови програмування C#;
- 2) Fonts – розділ, в якому зберігаються всі шрифти застосунку;
- 3) Images – директорія з картинками, які використовуються користувацьким інтерфейсом;

4) Prefabs – колекція префабів (кастомні компоненти Unity, які створюються програмістами і являють собою збірку компонентів і їх конфігурацій, що можуть бути багаторазово використані);

5) Scripts – саме в цьому розділі міститься вся програмна бізнес логіка користувацького інтерфейсу. Базові класи, контролери, класи анімацій, менеджери, моделі, сервіси, скриптовані об'єкти, перелічення, файли для роботи з мережею і класи-помічники – все це зберігається у директорії Scripts;

б) ImportedAssets – ця папка містить імпортовані класи і файли для користувацького інтерфейсу з магазину Unity під назвою Unity Assets Store;

в) Localization – у цій директорії зберігаються специфічні для системи локалізації файли і класи, включаючи таблиці перекладів.

4.3 Звертання до програми

Для використання програми і звертання до неї користувачами відповідальній людині необхідно встановити серверний застосунок і базу знань на сервер, а користувачеві мати доступ до скомпільованого для необхідної платформи програмного застосунку.

Для того, щоб завантажити вебсервер необхідно використати команду `dotnet run Release`. Для того, щоб запустити базу знань з хімії необхідно: завантажити Docker (програма для віртуалізації), побудувати образ для

проєкту бази знань використовуючи команду `docker build`, потім включити контейнер за допомогою команди `docker start -i [container-name]` і у вже запущеному контейнері застосунку бази знань запустити серверний застосунок, який надасть можливість роботи HTTP запити до бази знань, використовуючи команду `start_server(80, "en", "/app")`.

4.4 Вхідні та вихідні дані

Вхідними даними в програмі є персональні дані користувача, які він вказує при реєстрації, його відповіді в тестах рівній, а також інформація з спеціалізованої з певної дисципліни бази знань, що використовується для генерування питань рівнів, а також навчальних матеріалів. В якості початкових даних виступає інформація, щодо доступних дисциплін, навчальних етапів і рівнів, яка зберігається в базі даних і додана туди вручну.

Вихідними даними є відповідно кількісно представлений прогрес користувача з конкретної дисципліни і досягнення з навчальних етапів, представлених у вигляді тематичних предметів певних STEM дисциплін, а також згенеровані завдання для навчальних рівнів і корисні матеріали для опанування STEM.

4.5 Опис API інтерфейсів

Навчальна платформа на даному етапі її розробки і розвитку містить два API інтерфейси, які необхідно описати у цьому підрозділі.

Перший API інтерфейс являє собою інтерфейс вебсерверу. Його структура приведена в таблиці 4.1.

Таблиця 4.1 – Структура API інтерфейсу вебсерверу

Маршрут	Http метод	Параметри запиту або тіла запиту	Опис
/api/auth/login	POST	тіло запиту повинно містити дані користувача для аутентифікації	Аутентифікація користувача
/api/auth/register	POST	тіло запиту повинно містити дані користувача для реєстрації	Реєстрація користувача
/api/levels	GET	тіло запиту повинно містити унікальний ідентифікатор навчального етапу	Отримання рівнів навчального етапу
/api/levels/{id}	GET	id (параметер запиту) – унікальний ідентифікатор рівня	Генерація питань для рівня
/api/levels/{id}	POST	id (параметер запиту) – унікальний ідентифікатор рівня тіло запиту повинно містити дані щодо спроби проходження рівня	Реєстрація спроби проходження рівня
/api/stages	GET	тіло запиту повинно містити унікальний ідентифікатор дисципліни	Отримання навчальних етапів дисципліни
/api/subjects	GET	–	Отримання списку дисциплін навчальної платформи

Другий API інтерфейс являє собою вебінтерфейс бази знань з хімії. Його структура приведена в таблиці 4.2.

Таблиця 4.2 – Структура API інтерфейсу бази знань з хімії

Маршрут	Http метод	Параметри запиту або тіла запиту	Опис
<code>metr_dim_analysis/unit_quantity?[unit=unit_arg]&[quantity=quantity_arg]</code>	GET	unit_arg (параметер запиту) – значення міри quantity_arg (параметер запиту) – вимірювана величина	Отримання міри по вимірюваній величині, чи навпаки
<code>metr_dim_analysis?goal=how_many&args=[prefix1]&args=[value]&args=[prefix2]&args=[unit_arg]^power_arg]</code>	GET	prefix1 (параметер запиту) – значення першого префіксу value (параметер запиту) – кількісне значення prefix2 (параметер запиту) – значення другого префіксу unit_arg (параметер запиту) – значення міри power_arg (параметер запиту) – значення ступеню	Визначення кількості однієї метрики в іншій з використанням префіксів
<code>metr_dim_analysis?goal=compare_prefixes&args=[value1]&args=[prefix1]&args=[value2]&args=[prefix2]&args=[unit_arg]^power_arg]</code>	GET	prefix1 (параметер запиту) – значення першого префіксу value1 (параметер запиту) – кількісне значення першої сутності prefix2 (параметер запиту) – значення другого префіксу value2 (параметер запиту) – кількісне значення другої сутності unit_arg (параметер запиту) – значення міри power_arg (параметер запиту) – значення ступеню	Порівняння різних кількостей однієї сутностей

4.6 Повідомлення програмісту

Під час виникнення нестандартної, виключної ситуації з'явиться відповідне повідомлення в логах консолі (не залежно це клієнтська частина чи серверна).

5 КЕРІВНИЦТВО ОПЕРАТОРА

У даному розділі буде розглянуто призначення програми, умови використання, виконання та повідомлення для користувача.

5.1 Призначення програми

Розроблювана у рамках цієї випускної кваліфікаційної роботи бакалавра навчальна система створена для покращення і полегшення процесу вивчення STEM дисциплін, і призначена для широкого кола користувачів, основною вимогою до яких є “бажання навчатися і пізнавати нове”.

5.2 Умови використання програми

Ряд апаратно–технічних засобів, необхідних для нормальної роботи навчальної платформи наведений у п. 1.3.6.

Для комфортної роботи з інтерактивним навчальним застосунком користувачеві необхідно:

- мати базові навички користування комп’ютером або мобільним пристроєм (в залежності від того для якої платформи скомпільована розроблювана платформа);
- володіння українською або англійською мовою, бо інтерфейс застосунку підтримує саме ці дві мови наразі;
- користувач повинен бути ознайомленим з керівництвом оператора платформи.

5.3 Виконання програми

Щоб запустити навчальну платформу на своєму пристрої необхідно виконати одну з трьох дій, в залежності від того у якій формі

розповсюджується навчальна платформа: запустити виконуваний файл програми на комп'ютері, відкрити вебсайт на якому знаходиться застосунок або завантажити застосунок з мобільного пристрою після встановлення.

Як вже раніше було зазначено – користувацький інтерфейс інтерактивної навчальної платформи представлений 12-ю різноманітними екранами. Нижче будуть всі ці екрани і їх функції будуть детально розглянуті.

Після запуску навчальної платформи користувача зустрічає екран аутентифікації/реєстрації. Користувач повинен надати дані для входу в обліковий запис або для реєстрації і відправити ці дані для перевірки на вебсервер шляхом натискання кнопки, яка в залежності від обраного режиму може мати назву “Увійти” або “Зареєструватися”. На цьому екрані користувач може змінити тип форми (режим аутентифікації → на режим реєстрації, і в зворотньому порядку), натиснувши на кнопку яка знаходиться під полями введення особистих облікових даних користувачем. Також існує можливість зміни мови інтерфейсу: необхідно натиснути на спеціальну кнопку у нижньому правому кутку. Головний екран застосунку представлений на рисунку 5.1.

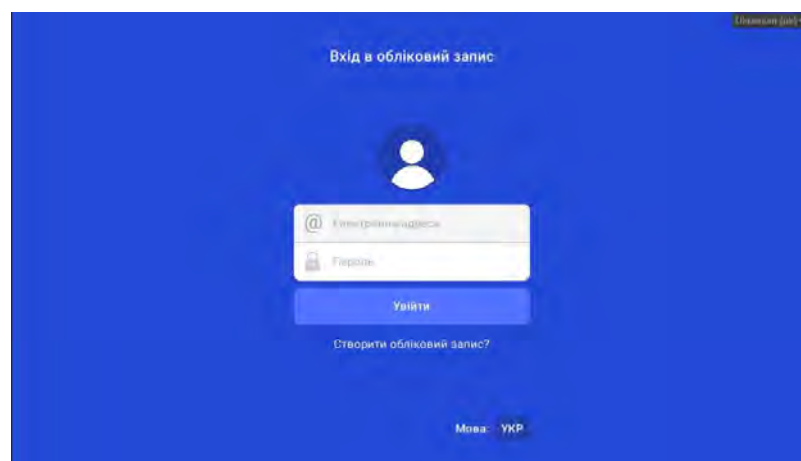


Рисунок 5.1 – Екран аутентифікації/реєстрації

Після вдалої аутентифікації/реєстрації користувач буде направлений у головне меню навчальної платформи. Перший екран меню застосунку

представлений у вигляді крейдової дошки (рис. 3.7). На даному етапі користувач може змінити мову інтерфейсу, обравши необхідну за допомогою зручної кнопки у вигляді прапора у правому нижньому куті дошки. Користувач може перейти далі натиснувши зелену кнопку у вигляді трикутника, що відображена у блоці кнопок посередині екрану. Також користувач може перейти до налаштувань платформи або закінчити виконання застосунку натиснувши синю кнопку з шестернею або червону кнопку з хрестиком відповідно.

Якщо користувач вирішить захоче рухатись далі і натисне зелену кнопку на першому екрані головного меню, то екран буде змінено на крейдову дошку чорного кольору, що являє собою другий екран головного меню. Цей екран насправді дуже простий і містить лише дві можливі дії: повернутися назад (необхідно натиснути на стрілку назад у верхньому лівому куті) або обрати дисципліну, до вивчення якої він хоче перейти. Вибір дисципліни відбувається за допомогою вибору значення зі спеціального слайдери, який знаходиться по середини екрану. Для обрання іншої дисципліни необхідно користуватися навігаційними кнопками, які знаходяться зліва і справа “об’єкту” дисципліни. Для кожної дисципліни відображена її назва, іконка, а також прогрес користувача у вигляді рядку прогресу (progress bar). Екран представлений на рис. 5.2.

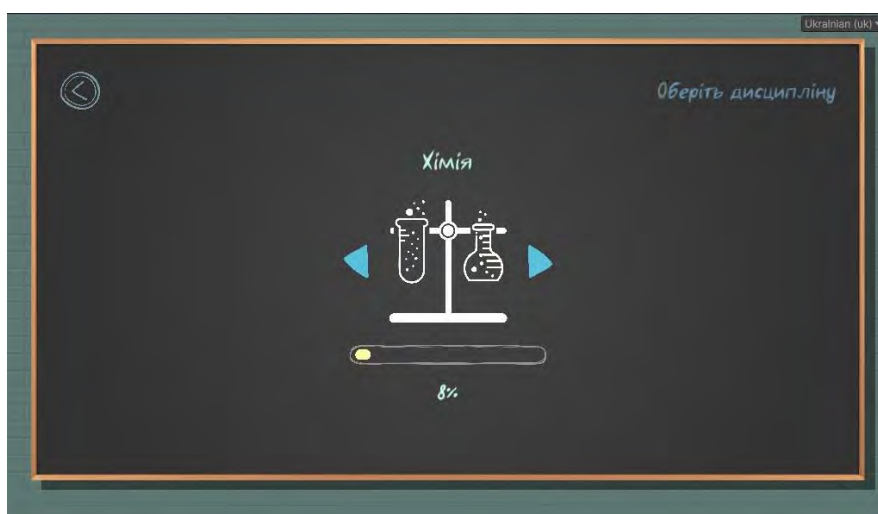


Рисунок 5.2 – Другий екран головного меню

Після вибору дисципліни екран навчальної платформи змінюється на наступний. Користувач бачить перед собою монітор, який автоматично і інтерактивно вмикається і містить групу контролерів для вибору навчального етапу дисципліни. Як і на минулому екрані головного меню тут є кнопка, щоб повернутися на попередній екран, а також слайдер, який пропонує обрати навчальний етап. Під слайдером розташований рядок прогресу, який показує прогрес користувача у проходженні навчального етапу. Недоступні навчальні етапи відображені сірим кольором і містять напис “Недоступно”. Цей екран головного меню представлений на рис. 3.22.

Після вибору необхідного навчального етапу екран монітору змінюється на наступний (рис. 5.3). Цей екран умовно називається “екраном вибору подальшої активності”. На ньому користувачеві вже по традиції доступна спеціальна кнопка, щоб повернутися до попереднього етапу меню, а також три кнопки, які визначають “наступний перехід” користувача. Ці кнопки: “Матеріали” (перехід до екрану з навчальними матеріалами), “Тестування” (перехід до екрану вибору рівня для контролю знань з навчального етапу), “Досягнення” (для перегляду особистих досягнень користувача, отриманих предметів).

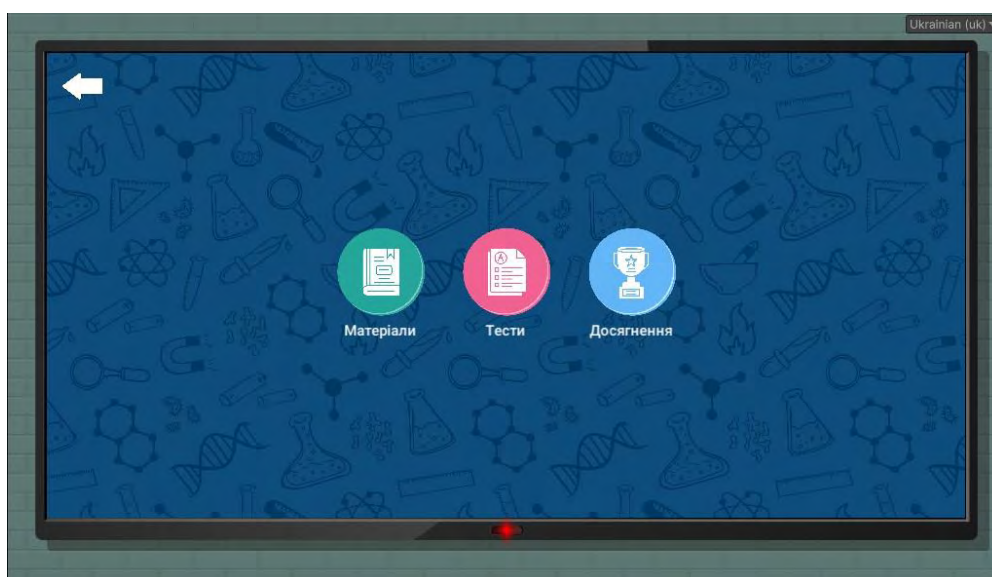


Рисунок 5.3 – Екран вибору активності, який з’являється після вибору навчального етапу на минулому екрані

Якщо на екрані активностей користувач натискає на кнопку “Матеріалів”, то екран застосунку змінюється на екран з навчальними матеріалами (рис. 5.4). На даному етапі розвитку і розробки навчальної платформи цей екран має достатньо просту структуру компонентів, але у майбутньому планується додати сюда додаткову функціональність, яка допоможе користувачам більш швидко отримувати доступ до необхідних матеріалів. Екран містить кнопку повернення у головне меню, розташовану у лівому верхньому кутку, а також блок з матеріалами з дуже зручною полозою прокрутки тексту.

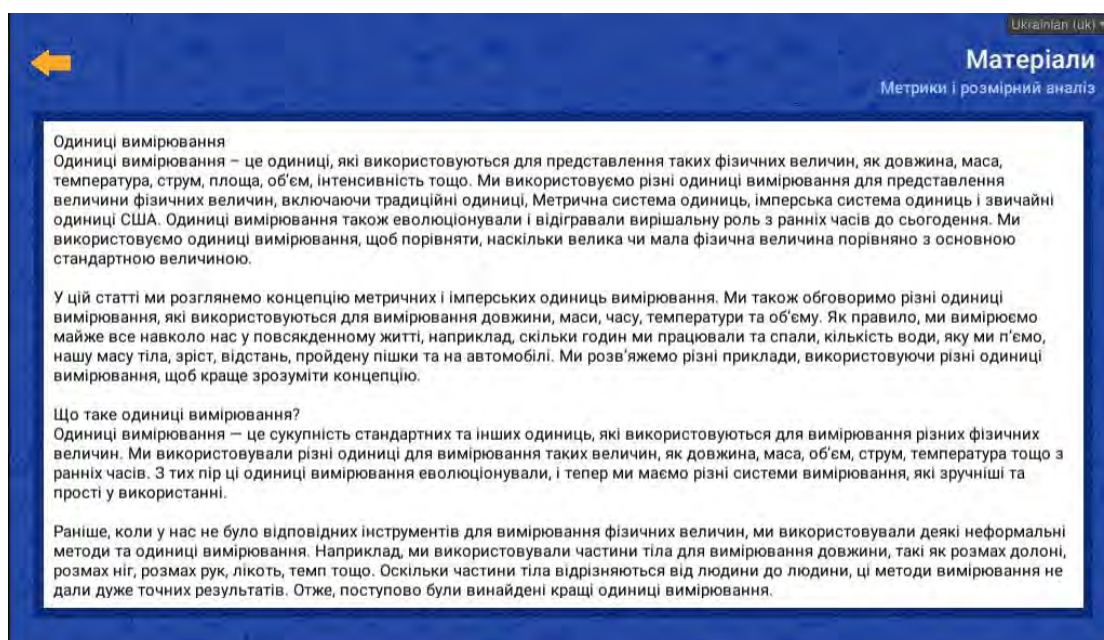


Рисунок 5.4 – Екран з навчальними матеріалами

Якщо на екрані з вибором активностей користувач натисне кнопку “Тестування”, то екран монітору буде змінено на екран вибору рівня. На цьому екрані звісно міститься кнопка повернення назад, слайдер, який дає можливість вибрати бажаний рівень для перевірки знань і блок, який пропонує перейти до екрану з інтерактивним тренажером для додаткового розуміння і засвоєння знань навчального етапу. Селектор рівня надає інформацію про поточний рівень: відображена назва рівня, індикатор статусу рівня (рівень недоступний, рівень доступний, рівень пройдений), а також

інформація щодо останньої спроби проходження рівня. Щоб почати рівень необхідно натиснути зелену трикутну кнопку, яка знаходиться у блоці з вибором рівня. Варто відмітити, що недоступні рівні виділені сірим кольором. Екран вибору рівня представлений на рис. 3.5.

Після вибору рівня перед користувачем з'являється екран, який містить більш детальну інформацію про рівень, а саме кількість питань і кількість виділеного на проходження часу (рис. 5.5). Після ознайомлення з цією інформацією користувач може почати проходження рівня натиснувши на зелену кнопку “Почати”. У лівому кутку екрану знаходиться кнопка для повернення на минулий екран.

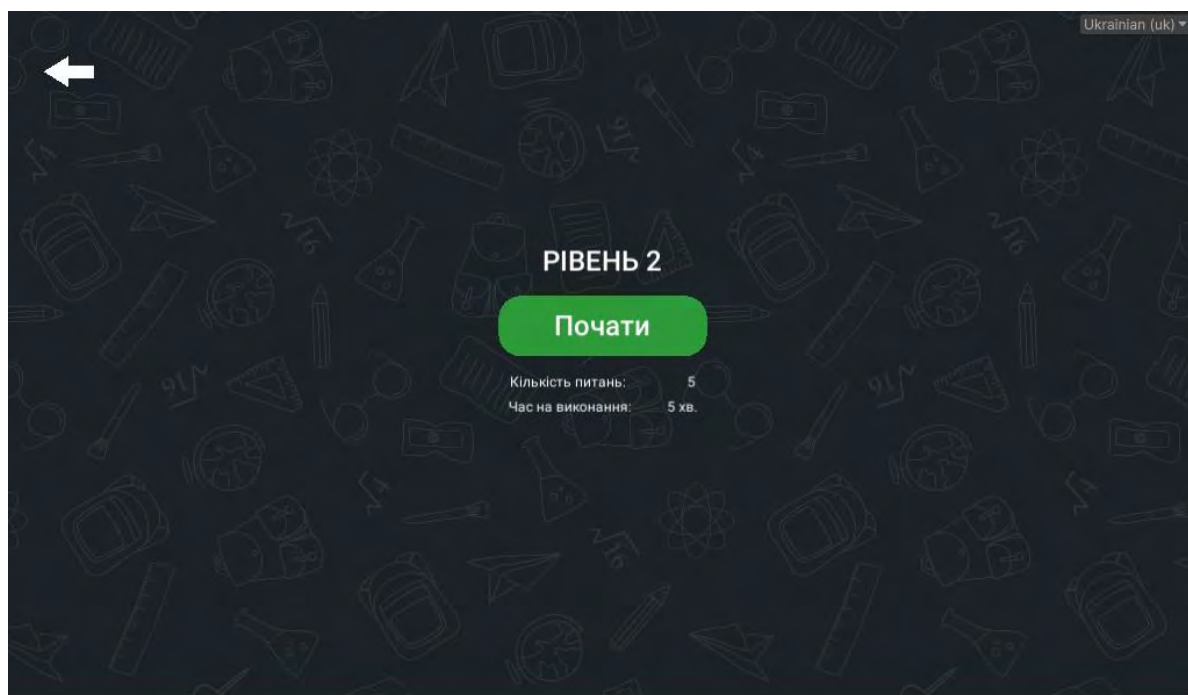


Рисунок 5.5 – Екран, що містить більш детальну інформацію щодо обраного рівня

Після того як користувач натисне зелену кнопку “Почати” – почнеться спроба проходження рівня. Екран зміниться на екран з тестовим питанням, який містить такі компоненти як: кнопка примусового закінчення спроби (правий верхній кут), текст питання в окремому блоці, варіанти відповідей або поле для введення відповіді, кнопка для відправки відповіді на питання,

індикатор правильної відповіді (червоний – відповідь неправильна, зелений – відповідь правильна). У правому нижньому кутку екрану розміщений анімований таймер, який показує кількість часу до завершення рівня. Екран відповідей на питання рівня представлений на рис. 3.6.

Після закінчення спроби користувач потрапляє на екран з результатами тестування (рис. 5.6). На цьому екрані відображена інформація, щодо кількості правильних відповідей на питання рівня, а також результат проходження. Результат проходження представлений у вигляді текстового повідомлення з кольоровим акцентом і картинкою щойно відкритого предмету, якщо рівень успішно пройдений. Екран також містить кнопку, натискання на яку повертає його до меню вибору рівня.

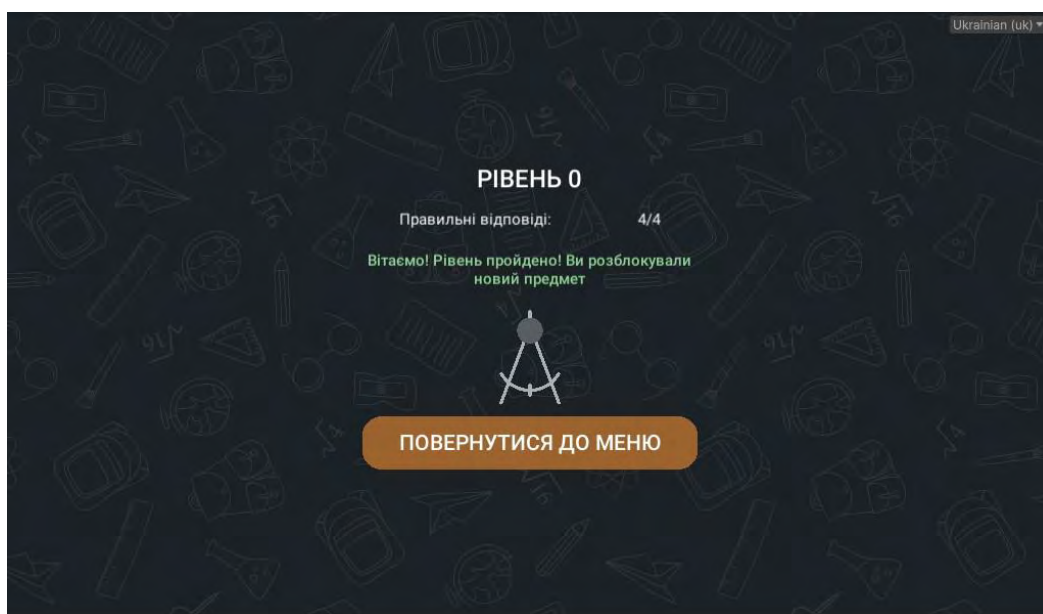


Рисунок 5.6 – Екран з результатами проходження рівня

Як вже було сказано: після спроби проходження рівня користувач повертається до меню вибору рівня. Тепер розглянемо, що станеться, якщо користувач натисне на кнопку з вибором інтерактивного тренажеру, яка має вигляд іконки з “мізками”. Після натискання на цю кнопку користувач буде направлений на інший екран навчальної платформи – екран тренажеру. Варто зазначити, що для кожного навчального етапу екран тренажеру буде різним,

бо навички, які треба відточити – різні. На даному етапі розвитку і розробки навчальної платформи реалізований тільки один тренувальний екран – для навчального етапу “Метрики і аналіз розмірностей”. Інтерактивний тренажер для навчального етапу “Метрика і аналіз розмірностей” представлений у вигляді 3-х інтерактивних блоків: блок переведення співставлення міри і величини вимірювання і навпаки, блок калькулятора, який містить логіку конвертації величини з визначеним префіксом і ступенем в іншу величину з іншим префіксом, і демонстраційний блок, що містить цікаву інформацію про величини з різними префіксами розмірностей. Екран з інтерактивним тренажером відображений на рис. 5.7.

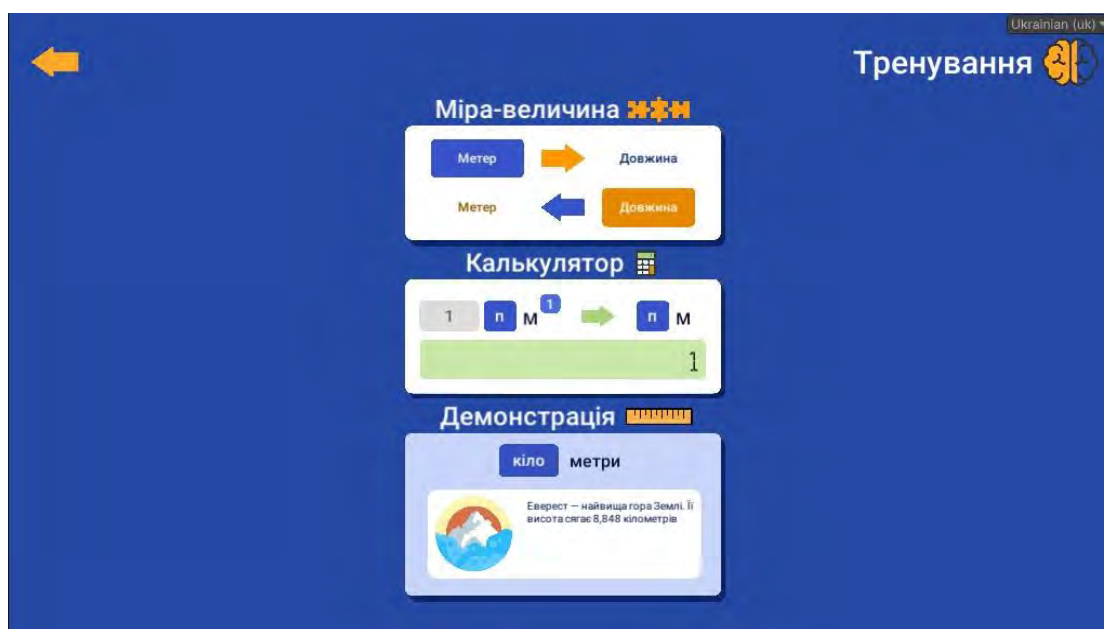


Рисунок 5.7 – Екран інтерактивного тренажеру з навчального етапу “Метрики і аналіз розмірностей”

Варто відмітити ще один екран навчальної платформи, який я забув згадати. На екрані вибору активностей є кнопка “Досягнення”, натискання на яку спричиняє перехід до екрану з досягненнями, який представлений у вигляді шафи з предметами. Відкриті предмети, отримані в результаті успішного проходження рівнів відображені як звичайні картинки з текстом білого кольору. Закриті предмети відображені як сірі контури предметів з

написами такого ж кольору. Екран з досягненнями представлений на рис. 3.24.

5.4 Повідомлення оператору

Повідомлення оператору в інтерактивній навчальній платформі для полегшення вивчення STEM дисциплін представлені у вигляді спливаючих вікон з повідомленнями. Ці вікна можуть бути інформативними, тобто просто доносити необхідну інформацію або діалоговими, тобто таким що чекають деякого вводу користувача. Прикладом інформативного спливаючого вікна є вікно, текст повідомлення якого доносить інформацію про помилку на вебсервері вебплатформи (рис. 5.8). Прикладом діалогового вікна може виступати вікно, яке з'являється, коли користувач натискає на кнопку примусового завершення рівня на екрані проходження певного рівня (рис. 3.21). Це діалогове вікно має на меті визначити дійсні наміри користувача: чи дійсно він хоче закінчити спробу проходження рівня.

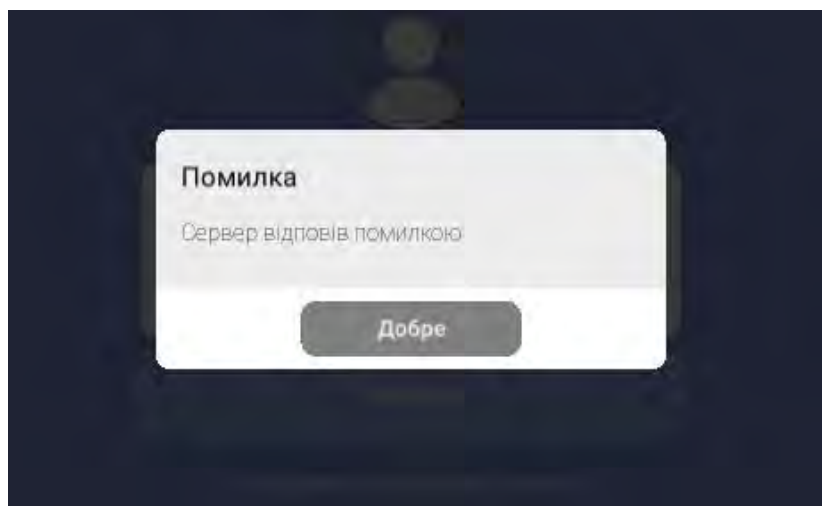


Рисунок 5.8 – Приклад інформаційного спливаючого вікна

ВИСНОВКИ

Під час виконання випускної кваліфікаційної роботи бакалавра вирішено актуальну задачу розробки програмного забезпечення навчальної системи з дисциплін STEM.

В ході виконання роботи було проведено аналіз технологій створення інтерактивної навчальної платформи на основі фреймворка ASP.NET Core і ігрового рушія Unity, проведено аналіз предметної області, а також вже наявних програм–аналогів, створено технічне завдання з переліком вимог до проєкту.

Після етапу проєктування, виконаного з урахуванням усіх вимог, висунутих у технічному завданні, було проведено реалізацію навчальної платформи для полегшення вивчення STEM дисциплін.

Розроблений програмний застосунок може бути використаний у навчальному процесі шкіл і університетів для вдосконалення процесу вивчення дисциплін STEM всіма категоріями учнів і студентів.

Готовий програмний продукт відповідає всім вимогам, поставленим у технічному завданні.

Програмна документація включає: технічне завдання; керівництво програміста; керівництво оператора та тексти основних програмних модулів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Rieber, L. Participation patterns in a massive open online course (MOOC) about statistics / Lloyd P Rieber // British Journal of Educational Technology. - 2017. - Vol. 48 №6. - P. 1295–1304.
2. Hone, K. Exploring the factors affecting MOOC retention: A survey study / Hone, K., & Said, G. // Computers & Education. - 2016. - Vol. 98. - P. 157–168.
3. Gamification: Toward a definition / [Deterding, S., Khaled, R., Nacke, L., & Dixon, D.]. // CHI Conference on Human Factors in Computing Systems : 11th international conference, Vancouver BC, Canada, 7 - 12 May, 2011. - 4 p.
4. Gamification of education: A review of literature / [Nah, F. F., Zeng, Q., Telaprolu, V. R., Ayyappa, A. P., & Eschenbrenner, B.]. // HCI in Business, Government and Organizations : Seventh International Conference, Copenhagen, 19-24 July, 2020. - P. 401-409.
5. “Everybody is playing the game, but nobody's rules are the same”: Towards adaptation of gamification Computers in Human Behavior gamification based on users' characteristics / Klock, A. C., Gasparini, I., Pimenta, M. S., & de Oliveira, J. P. M. // Bulletin of the IEEE Technical Committee on Learning Technology. - 2015. - Vol. 17, №4. - P. 22-25.
6. Gamify: How gamification motivates people to do extraordinary things / Burke, B. - Brookline, MA: Bibliomotion, 2014. - 194 p.
7. Does gamification work? - A literature review of empirical studies on gamification / [Hamari, J., Koivisto, J., & Sarsa, H.]. // In System sciences (HICSS) : 47th Hawaii international conference, Hawaii, January, 2014. - P. 3025-3034.
8. Fitz-Walter, Z. Driven to drive? Investigating the effect of gamification on learner driver behavior, perceived motivation and user experience / Fitz-Walter, Z., Johnson, D., Wyeth, P., Tjondronegoro, D., & Scott-Parker, B. // Computers in Human Behavior. - June, 2017. - Vol. 71 - P. 586-595.

9. Framework for designing and evaluating game achievements / [Hamari, J., & Eranti, V.]. // DiGRA International Conference, Hilversum, 14-17 September, 2011. - 20 p.
10. Dicheva, D. Gamification in education: A systematic mapping study / Dicheva, D., Dichev, C., Agre, G., & Angelova, G. // Educational Technology & Society. - 2015. - Vol. 18 №3. - P. 75-88.
11. Reality is broken: Why games make us better and how they can change the world / McGonigal, J. - London: Penguin, 2011. - 27 p.
12. Gamification by design: Implementing game mechanics in web and mobile apps / Zichermann, G., & Cunningham, C. - Sebastopol, CA: O'Reilly Media, 2011. - 182 p.
13. Engaging engineering students with gamification / [Barata, G., Gama, S., Jorge, J., & Gonçalves, D.]. // Games and virtual worlds for serious applications (VSGAMES) : 5th international conference, Poole, 13 September, 2013. - P. 1-8.
14. Abramovich, S. Are badges useful in education?: it depends upon the type of badge and expertise of learner / Abramovich, S., Schunn, C., & Higashi, R. M. // Educational Technology Research and Development. - 2013. - Vol. 217 №2. - 16 p.
15. Cronk, M. Using gamification to increase student engagement and participation in class discussion / [eds.: T. Amiel, B. Wilson]. // World conference on educational media and technology : Denver, Colorado, 26-29 June, 2012. - P. 311-315.
16. Alcivar, I. Design and evaluation of a gamified system for ERP training / Alcivar, I., Abad, A. G. // Computers in Human Behavior. - 2016. - Vol. 58. - P. 109-118.
17. Lister, M. Gamification: The effect on student motivation and performance at the post-secondary level / Lister, M. C. // Computers in Human Behavior. - 2015. - Vol. 3 №2. - 22 p.
18. Coursera [Electronic resource]. - Access mode: <https://www.coursera.org/>.

19. Khan Academy [Electronic resource]. - Access mode:
<https://www.khanacademy.org/>.

20. Phet Interactive Simulations [Electronic resource]. - Access mode:
<https://phet.colorado.edu/>.

ДОДАТОК А
Слайди презентації

ДИПЛОМНА РОБОТА

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ НАВЧАЛЬНОЇ СИСТЕМИ З ДИСЦИПЛІН STEM

Розробив: ст. групи КНТ-129

Керівник: доцент, к.т.н.

Даніїл ФІЛІППЕНКОВ

Сергій СЕРДЮК

Рисунок А.1 – Слайд 1

МЕТА РОБОТИ

Розробка програмного забезпечення навчальної системи з дисциплін STEM.

Об'єкт дослідження – процес навчання STEM дисциплінам.

Предмет дослідження – методи розробки інтелектуальних навчальних систем у сфері STEM освіти.



НУ "ЗП", Кафедра програмних засобів

2

Рисунок А.2 – Слайд 2

АНАЛІЗ ПРОГРАМНИХ АНАЛОГІВ

Характеристика	Coursera	Khan Academy	PhET Interactive Simulations
Навчальна компонента (наявність освітніх матеріалів)	+	+	+—
Можливість перевірок знань освітян	+	+	—
Безкоштовність	—	+	+
Зручний інтерфейс	—	—	+
Гейміфікація і інтерактивність	—	—	+
Персоналізація навчального процесу	—	+	—
Платформа у веббраузері	+	+	+
Наявність мобільного застосунку	+	+	+
Наявність desktop застосунку	—	—	—
Підтримка української мови	+—	—	+



НУ “ЗП”, Кафедра програмних засобів

3

Рисунок А.3 – Слайд 3

ОСНОВНІ ВИМОГИ ДО ПРОЄКТНОГО РІШЕННЯ

- Аутентифікація, реєстрація і відстежування прогресу для користувачів;
- Зв'язок з базою знань з хімії, написаній іншим студентом кафедри;
- Генерація питань для рівнів, використовуючи шаблони питань надані професором хімії з університету Penn State, США
- Організувати доступ до навчальних матеріалів з хімії для користувачів;
- Інтерактивний тренажер для навчального етапу «Метрики і аналіз розмірностей»;
- Наявність елементів гейміфікації, зокрема додання досягнень за проходження рівнів.

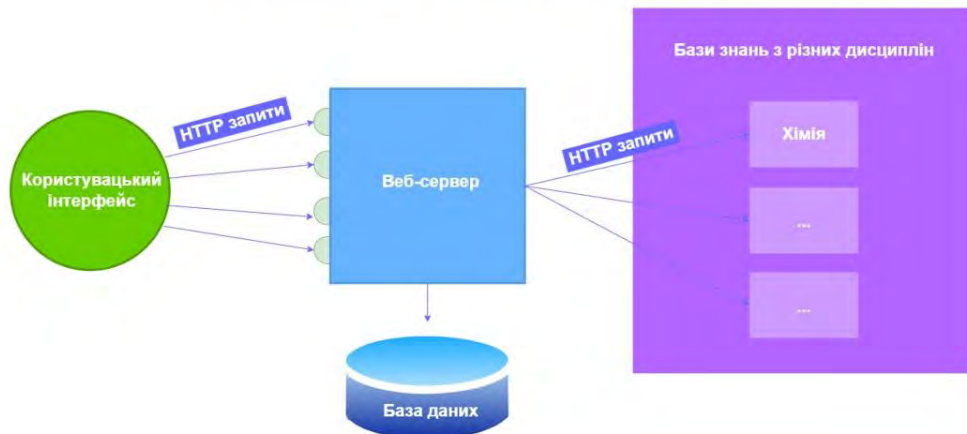


НУ “ЗП”, Кафедра програмних засобів

4

Рисунок А.4 – Слайд 4

СТРУКТУРА ПРОЄКТУ

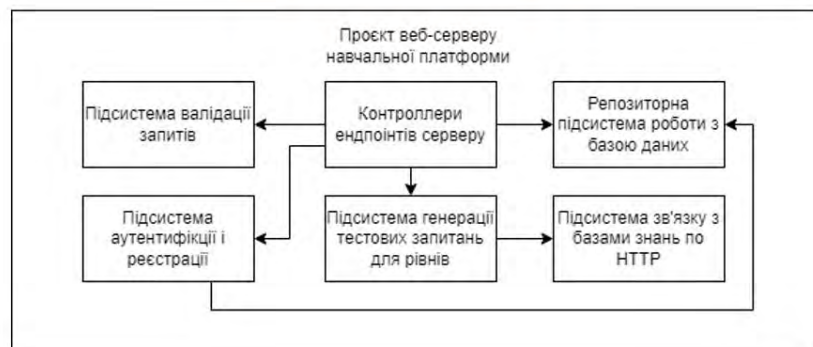


НУ "ЗП", Кафедра програмних засобів

5

Рисунок А.5 – Слайд 5

ФУНКЦІОНАЛЬНА СТРУКТУРА ВЕБСЕРВЕРУ

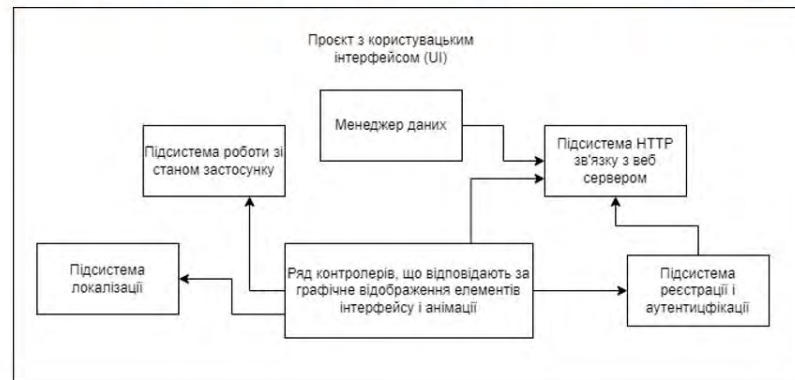


НУ "ЗП", Кафедра програмних засобів

6

Рисунок А.6 – Слайд 6

ФУНКЦІОНАЛЬНА СТРУКТУРА КОРИСТУВАЦЬКОГО ІНТЕРФЕЙСУ



НУ "ЗП", Кафедра програмних засобів

7

Рисунок А.7 – Слайд 7

API ВЕБСЕРВЕРУ

Маршрут	Http метод	Параметри запиту або тіла запиту	Опис
/api/auth/login	POST	тіло запиту повинно містити дані користувача для аутентифікації	Аутентифікація користувача
/api/auth/register	POST	тіло запиту повинно містити дані користувача для реєстрації	Реєстрація користувача
/api/levels	GET	тіло запиту повинно містити унікальний ідентифікатор навчального етапу	Отримання рівнів навчального етапу
/api/levels/{id}	GET	id (параметер запиту) – унікальний ідентифікатор рівня	Генерація питань для рівня
/api/levels/{id}	POST	id (параметер запиту) – унікальний ідентифікатор рівня тіло запиту повинно містити дані щодо спроби проходження рівня	Реєстрація спроби проходження рівня
/api/stages	GET	тіло запиту повинно містити унікальний ідентифікатор дисципліни	Отримання навчальних етапів дисципліни
/api/subjects	GET	–	Отримання списку дисциплін навчальної платформи



НУ "ЗП", Кафедра програмних засобів

8

Рисунок А.8 – Слайд 8

АРІ БАЗИ ЗНАНЬ З ХІМІЇ

Маршрут	Http метод	Параметри запиту або тіла запиту	Опис
metr_dim_analysis/unit_quantity?[unit=unit_arg]&[quantity=quantity_arg]	GET	unit_arg (параметер запиту) – значення міри quantity_arg (параметер запиту) – вимірювана величина	Отримання міри по вимірюваній величині, чи навпаки
metr_dim_analysis?goal=how_many&args=[prefix1]&args=[value]&args=[prefix2]&args=[unit_arg]^&args=[power_arg]	GET	prefix1 (параметер запиту) – значення першого префіксу value (параметер запиту) – кількісне значення prefix2 (параметер запиту) – значення другого префіксу unit_arg (параметер запиту) – значення міри power_arg (параметер запиту) – значення ступеню	Визначення кількості однієї метрики в іншій з використанням префіксів
metr_dim_analysis?goal=compare_prefixes&args=[value1]&args=[prefix1]&args=[value2]&args=[prefix2]&args=[unit_arg]^&args=[power_arg]	GET	prefix1 (параметер запиту) – значення першого префіксу value1 (параметер запиту) – кількісне значення першої сутності prefix2 (параметер запиту) – значення другого префіксу value2 (параметер запиту) – кількісне значення другої сутності unit_arg (параметер запиту) – значення міри power_arg (параметер запиту) – значення ступеню	Порівняння різних кількостей однієї сутностей

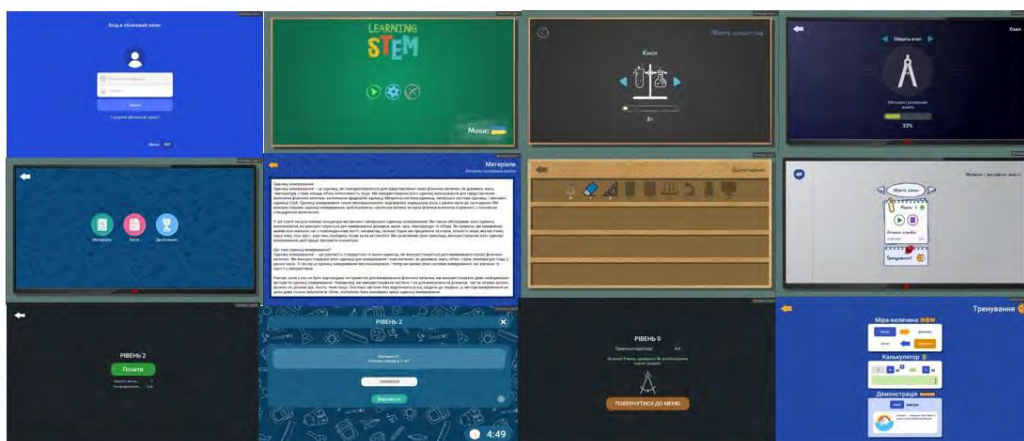


НУ "ЗП", Кафедра програмних засобів

9

Рисунок А.9 – Слайд 9

СУЧАСНИЙ ДИЗАЙН

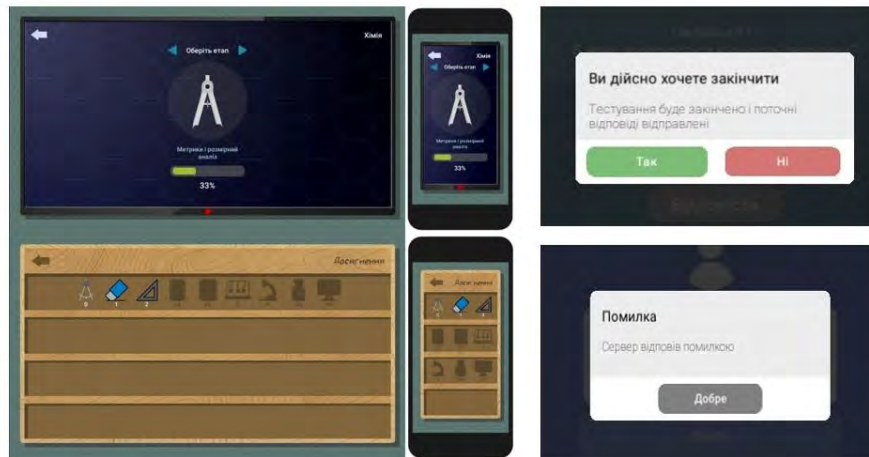


НУ "ЗП", Кафедра програмних засобів

10

Рисунок А.10 – Слайд 10

АДАПТИВНІСТЬ ТА ІНТЕРАКТИВНІСТЬ



НУ "ЗП", Кафедра програмних засобів

11

Рисунок А.11 – Слайд 11

ПІДТРИМКА УКРАЇНСЬКОЇ І АНГЛІЙСЬКОЇ МОВ



НУ "ЗП", Кафедра програмних засобів

12

Рисунок А.12 – Слайд 12

АПРОБАЦІЯ РОБОТИ

Сердюк, С.М. Розробка інтерактивного навчального застосунку «STEM LEARNING» / С.М. Сердюк, Д.Ю. Філіппенков // Тиждень науки-2023. Тези доповідей науково-практичної конференції, Запоріжжя, 24-28 квітня 2023 р. [Електронний ресурс] / Редкол.: В.А. Шаломєєв (відпов. ред.) Електрон. дані. – Запоріжжя : НУ «Запорізька політехніка», 2023. У друці.



НУ “ЗП”, Кафедра програмних засобів

13

Рисунок А.13 – Слайд 13

ВИСНОВКИ

- Проведено аналіз предметної області;
- Проведено огляд програмних аналогів;
- Розроблено технічне завдання;
- Розроблено програмне забезпечення навчальної платформи з дисциплін STEM
- Розроблено необхідну програмну документацію



НУ “ЗП”, Кафедра програмних засобів

14

Рисунок А.14 – Слайд 14

ДОКЛАД ЗАКІНЧЕНО
ДЯКУЮ ЗА УВАГУ!



НУ "ЗП", Кафедра програмних засобів

15

Рисунок А.15 – Слайд 15

ДОДАТОК Б
Текст програми

Проект користувацького інтерфейсу.

Б.1 Код файлу AppStateManager.cs

```
using System;
using UnityEngine;

namespace App.Scripts.Managers
{
    public class AppStateManager : MonoBehaviour
    {
        public static AppStateManager Instance;

        public static event Action<AppState> OnAppStateExit;
        public static event Action<AppState> OnAppStateChange;

        private AppState _state;

        private readonly AppState _firstAppState = AppState.Auth;

        public AppStateManager()
        {
            _state = _firstAppState;
        }

        void Awake ()
        {
            Instance = this;
        }

        void Start ()
        {
            UpdateState(_firstAppState);
        }

        public void UpdateState(AppState newState)
        {
            OnAppStateExit?.Invoke(_state);
            _state = newState;
            OnAppStateChange?.Invoke(newState);

            switch (_state)
            {
                case AppState.Quit:
                    HandleQuit();
                    break;
            };
        }

        private void HandleQuit()
        {
            Application.Quit();
        }
    }

    public enum AppState
    {
        Auth,
        Menu,
        Settings,
    }
}
```

```

        PlayLevel,
        Loading,
        Training,
        Materials,
        Quit
    }
}

```

Б.2 Код файлу DataManager.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using App.Scripts.Models;
using App.Scripts.ScriptableObjects.AchievementMetadata;
using App.Scripts.ScriptableObjects.StageMetadata;
using App.Scripts.ScriptableObjects.SubjectMetadata;
using App.Scripts.Utilities.Helpers;
using UnityEngine;

namespace App.Scripts.Managers
{
    public class DataManager : MonoBehaviour
    {
        [SerializeField] private List<SubjectMetadata> subjectsMetadataList;
        [SerializeField] private List<StageMetadata> stagesMetadataList;
        [SerializeField] private List<AchievementMetadata>
achievementMetadataList;
        public List<Subject> Subjects { get; private set; } = new
List<Subject>();

        public event Action OnUpdateSubjectsData;

        public static DataManager Instance;

        void Awake()
        {
            Instance = this;
        }

        public async Task InitLoading()
        {
            Subjects = await LoadSubjects();
            Debug.Log("Finished loading");
        }

        public void UpdateSubjectProgress(long subjectId, float
progressValue)
        {
            Subjects.First(x => x.Id == subjectId).Progress = progressValue;
            OnUpdateSubjectsData?.Invoke();
        }

        public void UpdateStageProgress(long subjectId, long stageId, float
progressValue)
        {
            Subjects.First(x => x.Id == subjectId).Stages.First(x => x.Id ==
stageId).Progress = progressValue;
            OnUpdateSubjectsData?.Invoke();
        }

        public void UpdateLevelProgress(long subjectId, long stageId, long

```

```

levelId, bool isCompleted, LevelTry levelTry)
{
    var stages = Subjects.First(x => x.Id == subjectId).Stages;
    var levels = stages.First(x => x.Id == stageId).Levels;
    var level = levels.First(x => x.Id == levelId);

    level.IsCompleted = isCompleted;
    level.Tries.Add(levelTry);

    if (isCompleted)
    {
        var indx = levels.IndexOf(level);
        if (indx < levels.Count - 1)
        {
            levels[indx + 1].IsAvailable = true;
        }
    }
    OnUpdateSubjectsData?.Invoke();
}

private async Task<List<Subject>> LoadSubjects()
{
    var subjectsApiResponseDto = await ApiHelper.GetSubjects();
    var subjects = new List<Subject>();
    foreach (var subject in subjectsApiResponseDto.Subjects)
    {
        var stages = new List<Stage>();
        if (subject.IsAvailable)
        {
            stages = await LoadStages(subject.Id);
        }

        var subjectMetadata = subjectsMetadataList.First(x => x.id ==
subject.Id);

        subjects.Add(new Subject()
        {
            Id = subject.Id,
            Name = subject.Name,
            IsAvailable = subject.IsAvailable,
            Stages = stages,
            Progress = subject.Progress.Value,
            LocalizedName = subjectMetadata.localizedName,
            Logo = subjectMetadata.logo
        });
    }

    return subjects;
}

private async Task<List<Stage>> LoadStages(long subjectId)
{
    var stagesResponse = await ApiHelper.GetStages(subjectId);

    var stages = new List<Stage>();
    foreach (var stage in stagesResponse.Stages)
    {
        var levels = new List<Level>();
        if (stage.IsAvailable)
        {
            levels = await LoadLevels(stage.Id);
        }

        var stageMetadata = stagesMetadataList.First(m => m.id ==

```

```

stage.Id);

        stages.Add(new Stage()
        {
            Id = stage.Id,
            Name = stage.Name,
            IsAvailable = stage.IsAvailable,
            Progress = stage.Progress.Value,
            LocalizedName = stageMetadata.localizedName,
            Logo = stageMetadata.logo,
            Levels = levels
        });
    }

    return stages;
}

private async Task<List<Level>> LoadLevels(long stageId)
{
    var levelsResponse = await ApiHelper.GetLevels(stageId);

    return levelsResponse.Levels.Select(x => new Level()
    {
        Id = x.Id,
        Name = x.Name,
        IsAvailable = x.IsAvailable,
        IsCompleted = x.IsCompleted,
        NumberOfQuestions = x.NumberOfQuestions,
        TimeLimit = x.TimeLimit,
        Tries = x.Tries.Select(t => new LevelTry()
        {
            CompletedQuestions = t.CompletedQuestions,
            TriedAt = t.TriedAt
        }).ToList()
    }).ToList();
}

public void ClearCache()
{
    Subjects = new List<Subject>();
}

public AchievementMetadata GetAchievementMetadata(string levelName)
{
    return achievementMetadataList.First(x => x.levelName ==
levelName);
}

}
}

```

Б.3 Код файла HoverAnimation.cs

```

using System.Collections;
using UnityEngine;

namespace App.Scripts.Animations
{
    public class HoverAnimation : MonoBehaviour
    {
        [SerializeField] private float scaleAmount = 1.1f;
        [SerializeField] private float lerpSpeed = 10f;

        private Vector3 _originalScale;
    }
}

```

```

    void Start()
    {
        _originalScale = transform.localScale;
    }

    public void OnPointerEnter()
    {
        StopAllCoroutines();
        StartCoroutine(LerpScale(_originalScale, _originalScale *
scaleAmount));
    }

    public void OnPointerExit()
    {
        StopAllCoroutines();
        StartCoroutine(LerpScale(transform.localScale, _originalScale));
    }

    private IEnumerator LerpScale(Vector3 startScale, Vector3 endScale)
    {
        float t = 0f;
        while (t < 1f)
        {
            t += Time.deltaTime * lerpSpeed;
            transform.localScale = Vector3.Lerp(startScale, endScale, t);
            yield return null;
        }
    }
}

```

Б.4 Код файла ProgressBarBehaviour.cs

```

using System;
using UnityEngine;

namespace App.Scripts.Behaviors.Base
{
    public abstract class ProgressBarBehaviour: MonoBehaviour
    {
        public bool IsEnabled { get; private set; }
        private float _progress;

        public ProgressBarBehaviour()
        {
            _progress = 0f;
        }

        protected virtual void Start() { }

        public void Enable()
        {
            IsEnabled = true;
            Display();
        }

        public void Disable()
        {
            IsEnabled = false;
            Display();
        }
    }
}

```

```

public void SetProgress(float rate)
{
    _progress = rate;
    DisplayProgressRateChange(_progress);
}

public float GetProgress()
{
    if (!IsEnabled)
    {
        throw new Exception("Unable to get value of disabled progress
bar");
    }

    return _progress;
}

private void Display()
{
    if (IsEnabled)
    {
        DisplayEnable();
    }
    else
    {
        DisplayDisable();
    }
}

protected abstract void DisplayEnable();

protected abstract void DisplayDisable();

protected abstract void DisplayProgressRateChange(float rate);
}

```

Б.5 Код файла SliderBehavior.cs

```

using System.Collections.Generic;
using UnityEngine;

namespace App.Scripts.Behaviors.Base
{
    public abstract class SliderBehavior<T> : MonoBehaviour where T: class
    {
        protected List<T> Objects;

        public T Current { get; private set; }

        private int _currentObject = 0;

        protected virtual void Start()
        {
            Current = Objects[_currentObject];
            Display(Objects[_currentObject]);
        }

        public void Next()
        {
            if (_currentObject == Objects.Count - 1)
            {
                _currentObject = 0;
            }
        }
    }
}

```

```

    }
    else
    {
        _currentObject++;
    }

    Current = Objects[_currentObject];
    Display(Objects[_currentObject]);
}

public void Previous()
{
    if (_currentObject == 0)
    {
        _currentObject = Objects.Count - 1;
    }
    else
    {
        _currentObject--;
    }

    Current = Objects[_currentObject];
    Display(Objects[_currentObject]);
}

protected abstract void Display(T stage);
}
}

```

Б.6 Код файла AuthController.cs

```

using System;
using App.Scripts.Controllers.Popup;
using App.Scripts.Managers;
using App.Scripts.Utilities.Helpers;
using StemLearning.Shared.Dto.Authorization;
using UnityEngine;
using UnityEngine.Localization;
using UnityEngine.Localization.Components;
using UnityEngine.UI;

namespace App.Scripts.Controllers.Auth
{
    public class AuthController : ScreenController
    {
        [SerializeField] private RectTransform bodyLayout;
        [SerializeField] private RectTransform controlsLayout;
        [SerializeField] private RectTransform inputsLayout;

        [SerializeField] private GameObject nameInput;

        [SerializeField] private Text nameInputTextField;
        [SerializeField] private Text emailInputTextField;
        [SerializeField] private Text passwordInputTextField;

        [SerializeField] private LocalizeStringEvent headerLabel;
        [SerializeField] private LocalizedString logInHeaderLocalizedString;
        [SerializeField] private LocalizedString
registerHeaderLocalizedString;

        [SerializeField] private LocalizeStringEvent modeChangerLabel;
        [SerializeField] private LocalizedString
logInModeChangerLocalizedString;

```

```

        [SerializeField] private LocalizedString
registerModeChangerLocalizedString;

        [SerializeField] private LocalizeStringEvent submitButtonLabel;
        [SerializeField] private LocalizedString
loginSubmitButtonLocalizedString;
        [SerializeField] private LocalizedString
registerSubmitButtonLocalizedString;

        private readonly float _inputsLayoutHeightWithTwoFields = 320;
        private readonly float _inputsLayoutHeightWithThreeFields = 485;

        private readonly float _controlsLayoutHeightWithTwoFields = 665;
        private readonly float _controlsLayoutHeightWithThreeFields = 820;

        private readonly float _bodyLayoutHeightWithTwoFields = 1025;
        private readonly float _bodyLayoutHeightWithThreeFields = 1180;

        private AuthMode _mode;

        protected override void OnEnterScreen()
        {
            SetMode(AuthMode.LogIn);
        }

        public void ChangeMode()
        {
            if (_mode == AuthMode.LogIn)
            {
                SetMode(AuthMode.Register);
            }
            else if (_mode == AuthMode.Register)
            {
                SetMode(AuthMode.LogIn);
            }
        }

        private void SetMode(AuthMode mode)
        {
            _mode = mode;
            OnModeChange();
        }

        private void OnModeChange()
        {
            switch (_mode)
            {
                case AuthMode.LogIn:
                    OnLogInMode();
                    break;
                case AuthMode.Register:
                    OnRegisterMode();
                    break;
            }
        }

        private void OnLogInMode()
        {
            headerLabel.StringReference = loginHeaderLocalizedString;
            modeChangerLabel.StringReference =
loginModeChangerLocalizedString;
            submitButtonLabel.StringReference =
loginSubmitButtonLocalizedString;
            nameInput.SetActive(false);

```

```

        inputsLayout.sizeDelta = new Vector2(inputsLayout.sizeDelta.x,
        _inputsLayoutHeightWithTwoFields);
        controlsLayout.sizeDelta = new
        Vector2(controlsLayout.sizeDelta.x, _controlsLayoutHeightWithTwoFields);
        bodyLayout.sizeDelta = new Vector2(bodyLayout.sizeDelta.x,
        _bodyLayoutHeightWithTwoFields);

    }

    private void OnRegisterMode()
    {
        headerLabel.StringReference = registerHeaderLocalizedString;
        modeChangerLabel.StringReference =
        registerModeChangerLocalizedString;
        submitButtonLabel.StringReference =
        registerSubmitButtonLocalizedString;
        nameInput.SetActive(true);
        inputsLayout.sizeDelta = new Vector2(inputsLayout.sizeDelta.x,
        _inputsLayoutHeightWithThreeFields);
        controlsLayout.sizeDelta = new
        Vector2(controlsLayout.sizeDelta.x, _controlsLayoutHeightWithThreeFields);
        bodyLayout.sizeDelta = new Vector2(bodyLayout.sizeDelta.x,
        _bodyLayoutHeightWithThreeFields);

    }

    public async void ClickSubmit()
    {
        AppStateManager.Instance.UpdateState(AppState.Loading);

        try
        {
            if (_mode == AuthMode.Register)
            {
                var response = await ApiHelper.Register(new
                RegisterRequestDto()
                {
                    Email = emailInputTextField.text,
                    Password = passwordInputTextField.text,
                    Name = nameInputTextField.text
                });
                Debug.Log($"Successful registration for
                {response.Email}");
            }
            else if (_mode == AuthMode.LogIn)
            {
                var response = await ApiHelper.Login(new
                LogInApiRequestDto()
                {
                    Email = emailInputTextField.text,
                    Password = passwordInputTextField.text
                });
                Debug.Log($"Successful authentication for
                {response.Email}");
            }

            await DataManager.Instance.InitLoading();

            AppStateManager.Instance.UpdateState(AppState.Menu);
        }
        catch (Exception e)
        {
            Debug.Log(e.Message);
            AppStateManager.Instance.UpdateState(AppState.Auth);
        }
    }

```

```

        PopupController.Instance.ShowUp(PopupType.Error);
    }
}

private enum AuthMode
{
    LogIn,
    Register
}
}
}

```

Б.7 Код файла AnswersVariantsController.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using UnityEngine;
using UnityEngine.UI;

namespace App.Scripts.Controllers.Levels
{
    public class AnswersVariantsController : MonoBehaviour
    {
        [SerializeField] private List<GameObject> answersVariantsObjects;

        [SerializeField] private Color selectedColor;

        private List<Image> _answersVariantsImages;

        private List<Text> _answersVariantsLabels;
        private Color NotSelectedColor { get; set; }

        public string SelectedVariant { get; private set; }

        private int _selectedVariantIndex;

        public event Action<string> OnSelectedVariantChanged;
        public static AnswersVariantsController Instance { get; private set; }
    }

    private void Awake()
    {
        Instance = this;
        _answersVariantsImages = answersVariantsObjects.Select(x =>
x.GetComponent<Image>()).ToList();
        _answersVariantsLabels = answersVariantsObjects.Select(x =>
x.GetComponentInChildren<Text>()).ToList();
        NotSelectedColor = _answersVariantsImages[0].color;
        _selectedVariantIndex = -1;
    }

    void Start()
    {
        RemoveSelection();
    }

    public void SelectVariant(int variantNumber)
    {
        if (variantNumber == _selectedVariantIndex)
        {

```

```

        variantNumber = -1;
    }

    if (variantNumber == -1)
    {
        SelectedVariant = "";
        foreach (var image in _answersVariantsImages)
        {
            image.color = NotSelectedColor;
        }
    }
    else
    {
        SelectedVariant = _answersVariantsLabels[variantNumber].text;
        _answersVariantsImages[variantNumber].color = selectedColor;
        foreach (var image in _answersVariantsImages)
        {
            if (image != _answersVariantsImages[variantNumber])
            {
                image.color = NotSelectedColor;
            }
        }
    }

    _selectedVariantIndex = variantNumber;
    OnSelectedVariantChanged?.Invoke(SelectedVariant);
}

public void SetActive(bool isActive)
{
    foreach (var o in answersVariantsObjects)
    {
        o.SetActive(isActive);
    }
}

public void PopulateWithAnswersVariants(List<string> answersVariants)
{
    foreach (var label in _answersVariantsLabels)
    {
        label.text =
answersVariants[_answersVariantsLabels.IndexOf(label)];
    }
}

public void RemoveSelection()
{
    SelectVariant(-1);
}
}
}

```

Б.8 Код файла LevelController.cs

```

using System;
using System.Collections;
using System.Linq;
using App.Scripts.Controllers.Menu.Menu2;
using App.Scripts.Controllers.Menu.Menu3;
using App.Scripts.Controllers.Popup;
using App.Scripts.Managers;
using App.Scripts.Models;
using App.Scripts.Utilities.Helpers;

```

```

using Localization;
using UnityEngine;
using UnityEngine.UI;

namespace App.Scripts.Controllers.Levels
{
    public class LevelController : ScreenController
    {
        [SerializeField] private GameObject startScreen;
        [SerializeField] private GameObject testScreen;
        [SerializeField] private GameObject resultsScreen;

        [SerializeField] private Text levelLabel;
        [SerializeField] private Text numberOfQuestionsLabel;
        [SerializeField] private Text timeLabel;

        [SerializeField] private Text questionBlockLabel;
        [SerializeField] private Text headerLevelText;

        [SerializeField] private GameObject inputVariantField;
        [SerializeField] private RectTransform questionBlock;

        [SerializeField] private Image submitButton;
        [SerializeField] private GameObject submitButtonDisableFilter;

        private QuestionType CurrentQuestionType { get; set; }

        public string MinLabelText { get; set; }
        public string LevelLabelText { get; set; }

        private string _localizedQuestionWord = "Question";

        private int _questionNumber;

        private Color _submitButtonDefaultColor;

        private InputField _inputField;

        protected override void Awake ()
        {
            base.Awake ();
            _inputField = inputVariantField.GetComponent<InputField>();
            _inputField.onValueChanged.AddListener(InputAnswerChanged);
            TestingManager.Instance.OnTestingStarted += OnTestingStarted;
            TestingManager.Instance.OnQuestionChanged += OnQuestionChanged;
            TestingManager.Instance.OnTestingEnded += OnTestingEnded;
            TestingManager.Instance.OnQuestionAnsweredRight +=
OnQuestionAnsweredRight;
            TestingManager.Instance.OnQuestionAnsweredWrong +=
OnQuestionAnsweredWrong;
            LocaleSelector.Instance.OnLocaleChanged += OnLocaleChanged;
            AnswersVariantsController.Instance.OnSelectedVariantChanged +=
OnSelectedVariantChanged;
        }

        private void OnSelectedVariantChanged(string answer)
        {
            if (answer == "")
            {
                SetActiveSubmitButton(false);
            }
            else
            {

```

```

        SetActiveSubmitButton(true);
    }

    private void OnTestingStarted()
    {
        _questionNumber = 0;
    }

    private void OnLocaleChanged(Locale locale)
    {
        if (locale == Locale.English)
        {
            _localizedQuestionWord = "Question";
        }
        else if (locale == Locale.Ukrainian)
        {
            _localizedQuestionWord = "Питання";
        }
    }

    protected override void OnDestroy()
    {
        base.OnDestroy();

        _inputField.GetComponent<InputField>().onValueChanged.RemoveListener(InputAnswerChanged);
        TestingManager.Instance.OnTestingStarted -= OnTestingStarted;
        TestingManager.Instance.OnQuestionChanged -= OnQuestionChanged;
        TestingManager.Instance.OnTestingEnded -= OnTestingEnded;
        TestingManager.Instance.OnQuestionAnsweredRight -= OnQuestionAnsweredRight;
        TestingManager.Instance.OnQuestionAnsweredWrong -= OnQuestionAnsweredWrong;
        LocaleSelector.Instance.OnLocaleChanged -= OnLocaleChanged;
        AnswersVariantsController.Instance.OnSelectedVariantChanged -= OnSelectedVariantChanged;
    }

    private void Start()
    {
        _localizedQuestionWord = questionBlockLabel.text;
        _submitButtonDefaultColor = submitButton.color;
    }

    private void OnQuestionAnsweredWrong(Question question, string answer)
    {
        PassIndicatorController.Instance.NotPassed();
        SetActiveSubmitButton(false);
    }

    private void OnQuestionAnsweredRight(Question question, string answer)
    {
        PassIndicatorController.Instance.Passed();
        SetActiveSubmitButton(false);
    }

    private async void OnTestingEnded(Level level, int rightAnswers)
    {
        try
        {
            var response = await ApiHelper.TryLevel(level.Id,

```

```

rightAnswers, DateTime.Now);

DataManager.Instance.UpdateSubjectProgress (SubjectSelectorController.Instance
.Current.Id,
        response.SubjectProgress.Value);

DataManager.Instance.UpdateStageProgress (SubjectSelectorController.Instance.C
urrent.Id,
        StageSelectorController.Instance.Current.Id,
        response.StageProgress.Value);

DataManager.Instance.UpdateLevelProgress (SubjectSelectorController.Instance.C
urrent.Id,
        StageSelectorController.Instance.Current.Id,
        LevelSelectorController.Instance.Current.Id,
        response.IsCompleted,
        new LevelTry ()
        {
            CompletedQuestions =
response.LevelTry.CompletedQuestions,
            TriedAt = response.LevelTry.TriedAt
        });

        testScreen.SetActive (false);
        resultsScreen.SetActive (true);
        ResultScreenController.Instance.DisplayResults (level,
rightAnswers);
    }
    catch (Exception e)
    {
        AppStateManager.Instance.UpdateState (AppState.Menu);
        PopupController.Instance.ShowUp (PopupType.Error);
    }

    _questionNumber = 0;
}

private void OnQuestionChanged (Question question)
{
    _questionNumber++;

    PassIndicatorController.Instance.Clear ();
    AnswersVariantsController.Instance.RemoveSelection ();
    _inputField.text = "";

    if (question.AnswersVariants.Any ())
    {
        CurrentQuestionType = QuestionType.WithAnswersVariants;
    }
    else
    {
        CurrentQuestionType = QuestionType.WithInput;
    }

    if (CurrentQuestionType == QuestionType.WithAnswersVariants)
    {
        inputVariantField.SetActive (false);
        AnswersVariantsController.Instance.SetActive (true);

        AnswersVariantsController.Instance.PopulateWithAnswersVariants (question.Answ
ersVariants);
    }
}

```

```

        var sizeDelta = questionBlock.sizeDelta;
        sizeDelta.y = 950;
        questionBlock.sizeDelta = sizeDelta;
    }
    else if (CurrentQuestionType == QuestionType.WithInput)
    {
        inputVariantField.SetActive(true);
        AnswersVariantsController.Instance.SetActive(false);
        var sizeDelta = questionBlock.sizeDelta;
        sizeDelta.y = 850;
        questionBlock.sizeDelta = sizeDelta;
    }

    questionBlockLabel.text = $"({_localizedQuestionWord}
#{_questionNumber}):\n{question.Text}";
}

protected override void OnEnterScreen()
{
    var level = LevelSelectorController.Instance.Current;
    startScreen.SetActive(true);
    levelLabel.text = LevelLabelText.ToUpper() + $" {level.Name}";
    numberOfQuestionsLabel.text = level.NumberOfQuestions.ToString();
    timeLabel.text = $"{level.TimeLimit.Minute.ToString()} " +
MinLabelText;

    testScreen.SetActive(false);
    resultsScreen.SetActive(false);
}

public void ClickStartLevel()
{
    startScreen.SetActive(false);
    testScreen.SetActive(true);

    var level = LevelSelectorController.Instance.Current;
    headerLevelText.text = LevelLabelText.ToUpper() + $"
{level.Name}";

    TestingManager.Instance.StartTesting();
}

public void ClickBack()
{
    IEnumerator BackToMenu()
    {
        yield return new WaitForSeconds(1f);
        AppStateManager.Instance.UpdateState(AppState.Menu);
    }

    AppStateManager.Instance.UpdateState(AppState.Loading);
    StartCoroutine(BackToMenu());
}

public void InputAnswerChanged(string text)
{
    if (text == "")
    {
        SetActiveSubmitButton(false);
    }
    else
    {
        SetActiveSubmitButton(true);
    }
}

```

```

    }

    private void SetActiveSubmitButton(bool isActive)
    {
        if (isActive)
        {
            submitButtonDisableFilter.SetActive(false);
            submitButton.color = _submitButtonDefaultColor;
        }
        else
        {
            submitButtonDisableFilter.SetActive(true);
            submitButton.color = Color.grey;
        }
    }

    public void SubmitAnswer()
    {
        string answer;
        if (CurrentQuestionType == QuestionType.WithInput)
        {
            answer = _inputField.text.Trim().ToLower();
        }
        else
        {
            answer = AnswersVariantsController.Instance.SelectedVariant;
        }
        TestingManager.Instance.AnswerToQuestion(answer);
    }

    public void ClickEndTesting()
    {
        PopupController.Instance.ShowUp(PopupType.EndTesting, (res) =>
        {
            if (res == PopupResult.Yes)
            {
                TestingManager.Instance.EndTesting();
            }
        });
    }

    private enum QuestionType
    {
        WithInput, WithAnswersVariants
    }
}

```

Б.9 Код файла LevelTimer.cs

```

using System;
using UnityEngine;
using UnityEngine.UI;

namespace App.Scripts.Controllers.Levels
{
    public class LevelTimer : MonoBehaviour
    {
        [SerializeField] private Text timerTextLabel;

        public static LevelTimer Instance;

        private float _timeRemaining;
    }
}

```

```

private bool _isTimerEnabled;

public event Action TimerEnded;

private void Awake()
{
    Instance = this;
}

private void Start()
{
    _isTimerEnabled = false;
}

public void StartTimer(float seconds)
{
    _timeRemaining = seconds;
    _isTimerEnabled = true;
}

public void StopTimer()
{
    _timeRemaining = 0;
}

public bool IsEnabled()
{
    return _isTimerEnabled;
}

private void Update()
{
    if (_timeRemaining > 0)
    {
        _timeRemaining -= Time.deltaTime;
        UpdateTimerText();
    }
    else
    {
        if (_isTimerEnabled)
        {
            _timeRemaining = 0;
            OnTimerEnded();
            _isTimerEnabled = false;
        }
    }
}

private void UpdateTimerText()
{
    var minutes = Mathf.FloorToInt(_timeRemaining / 60.0f);
    var seconds = Mathf.FloorToInt(_timeRemaining % 60.0f);
    timerTextLabel.text = $"{minutes:0}:{seconds:00}";
}

private void OnTimerEnded()
{
    TimerEnded?.Invoke();
}
}

```

Б.10 Код файла PassIndicatorController.cs

```
using System;
using UnityEngine;
using UnityEngine.UI;

namespace App.Scripts.Controllers.Levels
{
    public class PassIndicatorController : MonoBehaviour
    {
        public static PassIndicatorController Instance;

        [SerializeField] private Image background;
        [SerializeField] private Image indicatorCircle;

        [SerializeField] private Color passedColor;
        [SerializeField] private Color passedBackgroundColor;

        [SerializeField] private Color notPassedColor;
        [SerializeField] private Color notPassedBackgroundColor;

        private Color _defaultColor;
        private void Awake()
        {
            Instance = this;
            _defaultColor = background.color;
        }

        public void Clear()
        {
            background.color = _defaultColor;
            indicatorCircle.color = _defaultColor;
        }

        public void Passed()
        {
            background.color = passedBackgroundColor;
            indicatorCircle.color = passedColor;
        }

        public void NotPassed()
        {
            background.color = notPassedBackgroundColor;
            indicatorCircle.color = notPassedColor;
        }
    }
}
```

Б.11 Код файла ResultsScreenController.cs

```
using App.Scripts.Managers;
using App.Scripts.Models;
using UnityEngine;
using UnityEngine.UI;

namespace App.Scripts.Controllers.Levels
{
    public class ResultScreenController : MonoBehaviour
    {
        [SerializeField] private Text levelTitleLabel;
        [SerializeField] private Text passedTestsLabel;
        [SerializeField] private GameObject levelPassedLabel;
    }
}
```

```

[SerializeField] private GameObject levelNotPassedLabel;
[SerializeField] private RectTransform layout;
[SerializeField] private GameObject achievement;

private readonly float _layoutHeightWithoutIcon = 720;
private readonly float _layoutHeightWithIcon = 1020;

public string LevelLabelText { get; set; }
public static ResultScreenController Instance { get; private set; }

private void Awake()
{
    Instance = this;
    levelPassedLabel.SetActive(false);
    levelNotPassedLabel.SetActive(false);
}

public void DisplayResults(Level level, int
numberOfCompletedQuestions)
{
    levelTitleLabel.text = LevelLabelText.ToUpper() + $"
{level.Name}";
    passedTestsLabel.text =
$"{{numberOfCompletedQuestions}}/{{level.NumberOfQuestions}}";
    var sizeDelta = layout.sizeDelta;

    if (numberOfCompletedQuestions == level.NumberOfQuestions)
    {
        levelPassedLabel.SetActive(true);
        levelNotPassedLabel.SetActive(false);
        sizeDelta.y = _layoutHeightWithIcon;
        achievement.SetActive(true);
        var icon =
DataManager.Instance.GetAchievementMetadata(level.Name).icon;
        achievement.GetComponent<Image>().sprite = icon;
    }
    else
    {
        levelPassedLabel.SetActive(false);
        levelNotPassedLabel.SetActive(true);
        sizeDelta.y = _layoutHeightWithoutIcon;
        achievement.SetActive(false);
    }
    layout.sizeDelta = sizeDelta;
}
}
}

```

Б.12 Код файла TestingManager.cs

```

using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using App.Scripts.Models;
using App.Scripts.Utilities.Helpers;
using Localization;
using UnityEngine;

namespace App.Scripts.Controllers.Levels
{
    public class TestingManager : MonoBehaviour

```

```

{
    public static TestingManager Instance { get; private set; }

    private List<Question> Questions { get; set; }

    public event Action OnTestingStarted;

    public event Action<Question> OnQuestionChanged;

    public event Action<Question, string> OnQuestionAnsweredRight;

    public event Action<Question, string> OnQuestionAnsweredWrong;

    public event Action<Level, int> OnTestingEnded;

    private int _currentQuestionIndex;

    private Level _currentLevel;

    private bool _isTestingEnded;

    private int _rightAnswers;

    private void Awake()
    {
        Instance = this;
        LevelTimer.Instance.TimerEnded += EndTesting;
    }

    public async Task LoadLevelQuestions(Level level)
    {
        _currentLevel = level;

        var responseDto = await ApiHelper.GetLevel(level.Id,
LocaleSelector.Instance.CurrentLocale);

        if (!responseDto.Questions.Any())
        {
            throw new Exception("No questions for this level were
retrieved");
        }

        Questions = new List<Question>();

        foreach (var question in responseDto.Questions)
        {
            Questions.Add(new Question()
            {
                Text = question.Text,
                AcceptableAnswers = question.AcceptableAnswers,
                AnswersVariants = question.AnswersVariants
            });
        }

    }

    public void StartTesting()
    {
        _isTestingEnded = false;
        _currentQuestionIndex = 0;
        _rightAnswers = 0;
        OnTestingStarted?.Invoke();
        OnQuestionChanged?.Invoke(Questions[_currentQuestionIndex]);
        LevelTimer.Instance.StartTimer(_currentLevel.TimeLimit.Minute *
60f);
    }
}

```

```

    }

    public void AnswerToQuestion(string answer)
    {
        IEnumerator WithDelay()
        {
            yield return new WaitForSeconds(1f);
            if (_currentQuestionIndex == Questions.Count - 1)
            {
                EndTesting();
            }
            else
            {
                NextQuestion();
            }
        }

        var question = Questions[_currentQuestionIndex];
        var questionAnsweredRight = question.AcceptableAnswers.Any(x => x
== answer);

        if (questionAnsweredRight)
        {
            _rightAnswers++;
            OnQuestionAnsweredRight?.Invoke(question, answer);
        }
        else
        {
            OnQuestionAnsweredWrong?.Invoke(question, answer);
        }

        StartCoroutine(WithDelay());
    }

    public void EndTesting()
    {
        if (!_isTestingEnded)
        {
            if (LevelTimer.Instance.IsEnabled())
            {
                LevelTimer.Instance.StopTimer();
            }

            OnTestingEnded?.Invoke(_currentLevel, _rightAnswers);
            _isTestingEnded = true;
            _currentQuestionIndex = 0;
        }
    }

    private void NextQuestion()
    {
        _currentQuestionIndex++;
        OnQuestionChanged?.Invoke(Questions[_currentQuestionIndex]);
    }

    private void OnDestroy()
    {
        LevelTimer.Instance.TimerEnded -= EndTesting;
    }
}

```

Б.13 Код файла MaterialsController.cs

```
using System.Collections;
using App.Scripts.Managers;
using UnityEngine;

namespace App.Scripts.Controllers.Materials
{
    public class MaterialsController : ScreenController
    {
        public void BackToMenu()
        {
            IEnumerator Delayed()
            {
                yield return new WaitForSeconds(1f);
                AppStateManager.Instance.UpdateState(AppState.Menu);
            }

            AppStateManager.Instance.UpdateState(AppState.Loading);
            StartCoroutine(Delayed());
        }
    }
}
```

Б.14 Код файла SubjectProgressBarController.cs

```
using App.Scripts.Behaviors.Base;
using UnityEngine;
using UnityEngine.UI;

namespace App.Scripts.Controllers.Menu.Menu2
{
    public class SubjectProgressBarController : ProgressBarBehaviour
    {
        [SerializeField] private GameObject rateLabel;
        [SerializeField] private RectTransform innerPartTransform;
        [SerializeField] private Image innerPartImage;

        private Text _rateLabelText;
        private Vector3 _innerPartTransformSize;
        private float _innerPartImagePixelsPerUnitMultiplier;

        protected override void Start()
        {
            base.Start();
            _rateLabelText = rateLabel.GetComponent<Text>();
            _innerPartTransformSize = innerPartTransform.localScale;
            _innerPartImagePixelsPerUnitMultiplier =
innerPartImage.pixelPerUnitMultiplier;
        }

        protected override void DisplayEnable()
        {
            rateLabel.SetActive(true);
            innerPartImage.color = new Color(0.99f, 0.98f, 0.68f);
        }

        protected override void DisplayDisable()
        {
            rateLabel.SetActive(false);
            innerPartImage.color = Color.grey;
        }
    }
}
```

```

protected override void DisplayProgressRateChange(float rate)
{
    _rateLabelText.text = $"{Mathf.RoundToInt(rate * 100)}%";

    var scale = innerPartTransform.localScale;
    scale.x = _innerPartTransformSize.x * rate;
    innerPartTransform.localScale = scale;

    innerPartImage.pixelsPerUnitMultiplier =
(float) (_innerPartImagePixelsPerUnitMultiplier / 0.8) * rate;
}
}
}

```

Б.15 Код файла SubjectSelectorController.cs

```

using System;
using System.Linq;
using App.Scripts.Behaviors.Base;
using App.Scripts.Managers;
using App.Scripts.Models;
using App.Scripts.ScriptableObjectObjects.SubjectMetadata;
using UnityEngine;
using UnityEngine.Localization.Components;
using UnityEngine.UI;

namespace App.Scripts.Controllers.Menu.Menu2
{
    public class SubjectSelectorController : SliderBehavior<Subject>
    {
        [SerializeField] private Image logo;
        [SerializeField] private LocalizeStringEvent
titleLocalizeStringEvent;
        [SerializeField] private Text title;
        [SerializeField] private GameObject unavailableLabel;
        [SerializeField] private GameObject buttonDisableFilter;
        [SerializeField] private SubjectProgressBarController
subjectProgressBarController;

        [NonSerialized] public static SubjectSelectorController Instance;
        private void Awake()
        {
            Instance = this;
        }

        private void OnDestroy()
        {
            DataManager.Instance.OnUpdateSubjectsData -=
OnUpdateSubjectsData;
        }

        private void OnUpdateSubjectsData()
        {
            var currentSubjectId = Current.Id;
            Objects = DataManager.Instance.Subjects;
            Display(Objects.First(x => x.Id == currentSubjectId));
        }

        protected override void Start()
        {
            DataManager.Instance.OnUpdateSubjectsData +=
OnUpdateSubjectsData;
        }
    }
}

```

```

        Objects = DataManager.Instance.Subjects;
        base.Start();
    }

    protected override void Display(Subject subject)
    {
        logo.sprite = subject.Logo;
        titleLocalizeStringEvent.StringReference = subject.LocalizedName;

        if (subject.IsAvailable)
        {
            logo.color = Color.white;
            title.color = new Color(0.76f, 1f, 0.89f);
            unavailableLabel.SetActive(false);
            buttonDisableFilter.SetActive(false);
            subjectProgressBarController.SetProgress(subject.Progress);
            subjectProgressBarController.Enable();
        }
        else
        {
            logo.color = Color.grey;
            title.color = new Color(0.73f, 0.73f, 0.73f);
            unavailableLabel.SetActive(true);
            buttonDisableFilter.SetActive(true);
            subjectProgressBarController.SetProgress(1f);
            subjectProgressBarController.Disable();
        }
    }
}
}
}

```

Б.16 Код файла LevelSelectorController.cs

```

using System;
using System.Linq;
using App.Scripts.Behaviors.Base;
using App.Scripts.Controllers.Menu.Menu2;
using App.Scripts.Managers;
using App.Scripts.Models;
using UnityEngine;
using UnityEngine.UI;

namespace App.Scripts.Controllers.Menu.Menu3
{
    public class LevelSelectorController : SliderBehavior<Level>
    {
        [SerializeField] private Image playButton;
        [SerializeField] private Image materialsButton;

        [SerializeField] private Text levelName;
        [SerializeField] private Text lastTryDate;
        [SerializeField] private Text lastTryResult;

        [SerializeField] private Image levelStatusHolder;

        [SerializeField] private GameObject openLevelStatus;
        [SerializeField] private GameObject lockedLevelStatus;

        [SerializeField] private Color lockedColor;
        [SerializeField] private Color completedLevelColor;
        [SerializeField] private Color notCompletedLevelColor;

        [SerializeField] private GameObject buttonsDisableFilter;
    }
}

```

```

private Color _openLevelStatusColor;
private Image _openLevelStatusImage;

private Color _playButtonColor;
private Color _materialsButtonColor;

[NonSerialized] public static LevelSelectorController Instance;

private void Awake()
{
    Instance = this;
}

private void OnDestroy()
{
    DataManager.Instance.OnUpdateSubjectsData -=
OnUpdateSubjectsData;
}

private void OnUpdateSubjectsData()
{
    var currentLevelId = Current.Id;
    Objects = DataManager.Instance.Subjects
        .First(x => x.Id ==
SubjectSelectorController.Instance.Current.Id).Stages
        .First(x => x.Id ==
StageSelectorController.Instance.Current.Id).Levels;
    Display(Objects.First(x => x.Id == currentLevelId));
}

protected override void Start()
{
    DataManager.Instance.OnUpdateSubjectsData +=
OnUpdateSubjectsData;
    Objects = DataManager.Instance.Subjects
        .First(x => x.Id ==
SubjectSelectorController.Instance.Current.Id).Stages
        .First(x => x.Id ==
StageSelectorController.Instance.Current.Id).Levels;

    _openLevelStatusImage = openLevelStatus.GetComponent<Image>();
    _openLevelStatusColor = levelStatusHolder.color;

    _playButtonColor = playButton.color;
    _materialsButtonColor = materialsButton.color;

    base.Start();
}

protected override void Display(Level stage)
{
    levelName.text = stage.Name;
    if (stage.Tries is { Count: > 0 })
    {
        lastTryDate.text =
stage.Tries.Last().TriedAt.ToShortDateString();
        lastTryResult.text =
$"{{stage.Tries.Last().CompletedQuestions}}/{{stage.NumberOfQuestions}}";
    }
    else
    {
        lastTryDate.text = "-";
        lastTryResult.text = "-";
    }
}

```

```

    }

    if (stage.IsAvailable)
    {
        buttonsDisableFilter.SetActive(false);

        levelStatusHolder.color = _openLevelStatusColor;

        openLevelStatus.SetActive(true);
        lockedLevelStatus.SetActive(false);

        if (stage.IsCompleted)
        {
            _openLevelStatusImage.color = completedLevelColor;
        }
        else
        {
            _openLevelStatusImage.color = notCompletedLevelColor;
        }

        playButton.color = _playButtonColor;
        materialsButton.color = _materialsButtonColor;
    }
    else
    {
        playButton.color = lockedColor;
        materialsButton.color = lockedColor;

        buttonsDisableFilter.SetActive(true);

        levelStatusHolder.color = lockedColor;

        openLevelStatus.SetActive(false);
        lockedLevelStatus.SetActive(true);
    }
}
}
}
}

```

Б.17 Код файла MonitorController.cs

```

using System;
using System.Collections;
using UnityEngine;

namespace App.Scripts.Controllers.Menu.Menu3
{
    public class MonitorController : MonoBehaviour
    {
        [NonSerialized] public static MonitorController Instance;
        [SerializeField] private GameObject monitorLight;
        [SerializeField] private GameObject monitorScreen;
        private readonly float _timeOffset = 0.5f;

        private void Awake()
        {
            Instance = this;
        }

        private void Start()
        {
            DisableMonitor();
        }
    }
}

```

```

public void EnableMonitor()
{
    IEnumerator EnableMonitorScreen()
    {
        yield return new WaitForSeconds(_timeOffset);
        monitorScreen.SetActive(true);
    }

    monitorLight.SetActive(true);
    StartCoroutine(EnableMonitorScreen());
}

public void DisableMonitor()
{
    IEnumerator DisableMonitorLight()
    {
        yield return new WaitForSeconds(_timeOffset);
        monitorLight.SetActive(false);
    }

    monitorScreen.SetActive(false);
    StartCoroutine(DisableMonitorLight());
}
}
}

```

Б.18 Код файла MonitorScreenController.cs

```

using UnityEngine;

namespace App.Scripts.Controllers.Menu.Menu3
{
    public class MonitorScreenController : MonoBehaviour
    {
        [SerializeField] private GameObject stageSelectorScreen;

        [SerializeField] private GameObject levelSelectorScreen;

        [SerializeField] private GameObject activitySelectorScreen;

        private GameObject[] _screens;

        void Start()
        {
            _screens = new[]
            {
                stageSelectorScreen,
                levelSelectorScreen,
                activitySelectorScreen
            };
            EnableScreen(stageSelectorScreen);
        }

        public void GoToLevelSelectorScreen()
        {
            EnableScreen(levelSelectorScreen);
        }

        public void GoToStageSelectorScreen()
        {
            EnableScreen(stageSelectorScreen);
        }
    }
}

```

```

public void GoToActivitySelectorScreen()
{
    EnableScreen(activitySelectorScreen);
}

private void EnableScreen(GameObject screen)
{
    foreach (var s in _screens)
    {
        if (s == screen)
        {
            s.SetActive(true);
        }
        else
        {
            s.SetActive(false);
        }
    }
}
}

```

Б.19 Код файла StageProgressBarController.cs

```

using App.Scripts.Behaviors.Base;
using UnityEngine;
using UnityEngine.UI;

namespace App.Scripts.Controllers.Menu.Menu3
{
    public class StageProgressBarController : ProgressBarBehaviour
    {
        [SerializeField] private GameObject rateLabel;
        [SerializeField] private RectTransform innerPartTransform;
        [SerializeField] private Image innerPartImage;

        private Text _rateLabelText;
        private Color _innerPartEnabledColor;
        private Vector3 _innerPartTransformSize;
        private float _innerPartImagePixelsPerUnitMultiplier;

        protected override void Start()
        {
            base.Start();
            _innerPartEnabledColor = innerPartImage.color;
            _rateLabelText = rateLabel.GetComponent<Text>();
            _innerPartTransformSize = innerPartTransform.localScale;
            _innerPartImagePixelsPerUnitMultiplier =
innerPartImage.pixelsPerUnitMultiplier;
        }

        protected override void DisplayEnable()
        {
            innerPartImage.color = _innerPartEnabledColor;
            rateLabel.SetActive(true);
        }

        protected override void DisplayDisable()
        {
            innerPartImage.color = Color.grey;
            rateLabel.SetActive(false);
        }
    }
}

```

```

protected override void DisplayProgressRateChange(float rate)
{
    _rateLabelText.text = $"{Mathf.RoundToInt(rate * 100)}%";

    var scale = innerPartTransform.localScale;
    scale.x = _innerPartTransformSize.x * rate;
    innerPartTransform.localScale = scale;

    innerPartImage.pixelsPerUnitMultiplier =
(float) (_innerPartImagePixelsPerUnitMultiplier / 0.8) * rate;
}
}
}

```

Б.20 Код файла StageSelectorController.cs

```

using System;
using System.Linq;
using App.Scripts.Behaviors.Base;
using App.Scripts.Controllers.Menu.Menu2;
using App.Scripts.Managers;
using App.Scripts.Models;
using UnityEngine;
using UnityEngine.Localization.Components;
using UnityEngine.UI;

namespace App.Scripts.Controllers.Menu.Menu3
{
    public class StageSelectorController : SliderBehavior<Stage>
    {
        [SerializeField] private Image logo;
        [SerializeField] private Text title;
        [SerializeField] private LocalizeStringEvent
titleLocalizeStringEvent;
        [SerializeField] private GameObject unavailableLabel;
        [SerializeField] private GameObject buttonDisableFilter;
        [SerializeField] private StageProgressBarController
stageProgressBarController;

        [NonSerialized] public static StageSelectorController Instance;

        private Color _titleBaseColor;

        private void Awake()
        {
            Instance = this;
        }

        private void OnDestroy()
        {
            DataManager.Instance.OnUpdateSubjectsData -=
OnUpdateSubjectsData;
        }

        private void OnUpdateSubjectsData()
        {
            var currentStageId = Current.Id;
            Objects = DataManager.Instance.Subjects.First(x => x.Id ==
SubjectSelectorController.Instance.Current.Id).Stages;
            Display(Objects.First(x => x.Id == currentStageId));
        }
    }
}

```

```

        protected override void Start()
        {
            DataManager.Instance.OnUpdateSubjectsData +=
OnUpdateSubjectsData;
            _titleBaseColor = title.color;
            Objects = DataManager.Instance.Subjects.First(x => x.Id ==
SubjectSelectorController.Instance.Current.Id).Stages;
            base.Start();
        }

        protected override void Display(Stage stage)
        {
            logo.sprite = stage.Logo;
            titleLocalizeStringEvent.StringReference = stage.LocalizedName;

            if (stage.IsAvailable)
            {
                title.color = _titleBaseColor;
                unavailableLabel.SetActive(false);
                buttonDisableFilter.SetActive(false);
                stageProgressBarController.SetProgress(stage.Progress);
                stageProgressBarController.Enable();
            }
            else
            {
                title.color = Color.grey;
                unavailableLabel.SetActive(true);
                buttonDisableFilter.SetActive(true);
                stageProgressBarController.SetProgress(1f);
                stageProgressBarController.Disable();
            }
        }
    }
}

```

Б.21 Код файлу AchievementController.cs

```

using System.Collections.Generic;
using App.Scripts.Managers;
using UnityEngine;
using UnityEngine.UI;

namespace App.Scripts.Controllers.Menu.Menu4
{
    public class AchievementsController : MonoBehaviour
    {
        [SerializeField] private GridLayoutGroup gridLayoutGroup;
        [SerializeField] private GameObject prefab;
        [SerializeField] private Color disableAchievementColor;

        private Dictionary<string, GameObject> _achievementsObjects;

        private void OnDestroy()
        {
            DataManager.Instance.OnUpdateSubjectsData -=
OnUpdateSubjectsData;
        }

        private void OnUpdateSubjectsData()
        {
            var levels = DataManager.Instance.Subjects[0].Stages[0].Levels;
            foreach (var level in levels)

```

```

        {
            SetActive(level.Name, level.IsCompleted);
        }
    }

    private void Start()
    {
        DataManager.Instance.OnUpdateSubjectsData +=
OnUpdateSubjectsData;

        var levels = DataManager.Instance.Subjects[0].Stages[0].Levels;
        _achievementsObjects = new Dictionary<string, GameObject>();

        foreach (var level in levels)
        {
            var obj = Instantiate(prefab, gridLayoutGroup.transform);
            var achievement =
DataManager.Instance.GetAchievementMetadata(level.Name);
            _achievementsObjects.Add(level.Name, obj);
            var objText = obj.GetComponentInChildren<Text>();
            var objImage = obj.GetComponentInChildren<Image>();
            objText.text = achievement.levelName;
            objImage.sprite = achievement.icon;

            SetActive(level.Name, level.IsCompleted);
        }
    }

    private void SetActive(string levelName, bool isActive)
    {
        var achievement = _achievementsObjects[levelName];
        var foregroundImage =
achievement.GetComponentInChildren<Image>()[1];
        var achievementText = achievement.GetComponentInChildren<Text>();

        var foregroundColor = foregroundImage.color;

        if (isActive)
        {
            foregroundColor.a = 0;
            achievementText.color = Color.white;
        }
        else
        {
            foregroundColor.a = 1;
            achievementText.color = disableAchievementColor;
        }

        foregroundImage.color = foregroundColor;
    }
}

```

Б.22 Код файлу MenuController.cs

```

using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using App.Scripts.Controllers.Levels;
using App.Scripts.Controllers.Menu.Menu2;
using App.Scripts.Controllers.Menu.Menu3;
using App.Scripts.Controllers.Popup;

```

```

using App.Scripts.Managers;
using UnityEngine;

namespace App.Scripts.Controllers.Menu
{
    public class MenuController : ScreenController
    {
        [SerializeField] private List<GameObject> menuScreens;

        [SerializeField] private float animationTime = 1.4f;

        private readonly Vector2 _centerScreen = Vector2.zero;

        private readonly float _xOffset;

        private int _currentMenuScreenIndex;

        public MenuController()
        {
            _currentMenuScreenIndex = -1;
            _xOffset = 10000;
        }

        protected override void OnEnterScreen()
        {
            if (_currentMenuScreenIndex == -1)
            {
                MoveMenuCarouselLeft();
            }
        }

        protected override void OnExitScreen() { }

        void Start()
        {
            PutAwayAllObjects();
        }

        public void ClickPlay()
        {
            MoveMenuCarouselLeft();
        }

        public void ClickBack()
        {
            MoveMenuCarouselRight();
        }

        public void ClickSettings()
        {
        }

        public void ClickQuit()
        {
            AppStateManager.Instance.UpdateState(AppState.Quit);
        }

        public void ClickSelectSubject()
        {
            IEnumerator EnableMonitor()
            {
                yield return new WaitForSeconds(1f);
                MonitorController.Instance.EnableMonitor();
            }
        }
    }
}

```

```

        if (SubjectSelectorController.Instance.Current.IsAvailable)
        {
            MoveMenuCarouselLeft();
            StartCoroutine(EnableMonitor());
        }
    }

    public void ClickReturnFromMonitor()
    {
        IEnumerator SwitchMenu()
        {
            yield return new WaitForSeconds(1f);
            MoveMenuCarouselRight();
        }

        MonitorController.Instance.DisableMonitor();
        StartCoroutine(SwitchMenu());
    }

    public void ClickReturnFromResults()
    {
        MoveMenuCarouselRight();
    }

    public void OpenMaterials()
    {
        IEnumerator Delayed()
        {
            yield return new WaitForSeconds(1f);
            AppStateManager.Instance.UpdateState(AppState.Materials);
        }

        AppStateManager.Instance.UpdateState(AppState.Loading);
        StartCoroutine(Delayed());
    }

    public async void ClickPlayLevel()
    {
        AppStateManager.Instance.UpdateState(AppState.Loading);
        try
        {
            await
TestingManager.Instance.LoadLevelQuestions(LevelSelectorController.Instance.C
urrent);
            AppStateManager.Instance.UpdateState(AppState.PlayLevel);
        }
        catch (Exception e)
        {
            AppStateManager.Instance.UpdateState(AppState.Menu);
            PopupController.Instance.ShowUp(PopupType.Error);
        }
    }

    public void GoToTraining()
    {
        IEnumerator Delayed()
        {
            yield return new WaitForSeconds(1f);
            AppStateManager.Instance.UpdateState(AppState.Training);
        }

        AppStateManager.Instance.UpdateState(AppState.Loading);
        StartCoroutine(Delayed());
    }

```

```

    }

    public void ClickResults()
    {
        MoveMenuCarouselLeft();
    }

    private void MoveMenuCarouselLeft()
    {
        if (_currentMenuScreenIndex + 1 >= menuScreens.Count)
        {
            return;
        }

        _currentMenuScreenIndex++;

        foreach (var (menuScreen, index) in menuScreens
            .Select((item, index) => (item, index)))
        {
            if (index == _currentMenuScreenIndex - 1)
            {
                MoveMenuScreenOutTheCenter(menuScreen, -_xOffset);
            }
            else if (index == _currentMenuScreenIndex)
            {
                MenuScreenToCenter(menuScreen);
            }
        }
    }

    private void MoveMenuCarouselRight()
    {
        if (_currentMenuScreenIndex - 1 < 0)
        {
            return;
        }

        _currentMenuScreenIndex--;

        foreach (var (menuScreen, index) in menuScreens
            .Select((item, index) => (item, index))
            .Reverse())
        {
            if (index == _currentMenuScreenIndex + 1)
            {
                MoveMenuScreenOutTheCenter(menuScreen, _xOffset);
            }
            else if (index == _currentMenuScreenIndex)
            {
                MenuScreenToCenter(menuScreen);
            }
        }
    }

    private void PutAwayAllObjects()
    {
        foreach (var o in menuScreens)
        {
            PlaceObjectOutsideOfTheScreen(o);
        }
    }

    private void PlaceObjectOutsideOfTheScreen(GameObject obj)
    {

```

```

        var outsideScreenOffset = 30;
        var newPosition = obj.transform.position;
        newPosition.x = outsideScreenOffset;
        obj.transform.position = newPosition;
    }

    private void MoveObject(GameObject obj, Vector2 destination)
    {
        obj.LeanMoveLocal(destination, animationTime).setEaseInOutBack();
    }

    private void MenuScreenToCenter(GameObject menuScreen)
    {
        MoveObject(menuScreen, _centerScreen);
    }

    private void MoveMenuScreenOutTheCenter(GameObject menuScreen, float
xOffset)
    {
        var oldPos = _centerScreen;

        var newPos = new Vector2(xOffset , oldPos.y);

        MoveObject(menuScreen, newPos);
    }
}

```

Б.23 Код файлу PopupController.cs

```

using System;
using UnityEngine;
using UnityEngine.Localization;
using UnityEngine.Localization.Components;
using UnityEngine.UI;

namespace App.Scripts.Controllers.Popup
{
    public class PopupController : MonoBehaviour
    {
        [SerializeField] private GameObject popupScreen;

        [SerializeField] private LocalizedString ok;
        [SerializeField] private LocalizedString yes;
        [SerializeField] private LocalizedString no;

        [SerializeField] private LocalizedString errorHeader;
        [SerializeField] private LocalizedString endTestingHeader;

        [SerializeField] private LocalizedString endTestingBody;
        [SerializeField] private LocalizedString errorBody;

        [SerializeField] private LocalizeStringEvent
button1LocalizedStringEvent;
        [SerializeField] private LocalizeStringEvent
button2LocalizedStringEvent;
        [SerializeField] private LocalizeStringEvent
headerLocalizedStringEvent;
        [SerializeField] private LocalizeStringEvent
bodyLocalizedStringEvent;

        [SerializeField] private GameObject button2;
        [SerializeField] private Image button1;
    }
}

```

```

private PopupType _currentPopupType;
private Color _button1Color;
private Action<PopupResult> _callback;
private PopupResult Result { get; set; } = PopupResult.None;

public static PopupController Instance;
private void Awake()
{
    Instance = this;
    _button1Color = button1.color;
    DisablePopup();
}

public void Button1Clicked()
{
    if (_currentPopupType == PopupType.Error)
    {
        Result = PopupResult.Ok;
    }
    else if (_currentPopupType == PopupType.EndTesting)
    {
        Result = PopupResult.Yes;
    }
    DisablePopup();
}

public void Button2Clicked()
{
    if (_currentPopupType == PopupType.EndTesting)
    {
        Result = PopupResult.No;
    }
    DisablePopup();
}

public void ShowUp(PopupType popupType, Action<PopupResult> callback
= null)
{
    _callback = callback;
    _currentPopupType = popupType;
    popupScreen.SetActive(true);
    if (_currentPopupType == PopupType.Error)
    {
        button2.SetActive(false);
        button1.color = Color.grey;
        headerLocalizedStringEvent.StringReference = errorHeader;
        bodyLocalizedStringEvent.StringReference = errorBody;
        button1LocalizedStringEvent.StringReference = ok;
    }
    else if (_currentPopupType == PopupType.EndTesting)
    {
        button2.SetActive(true);
        button1.color = _button1Color;
        headerLocalizedStringEvent.StringReference =
endTestingHeader;
        bodyLocalizedStringEvent.StringReference = endTestingBody;
        button1LocalizedStringEvent.StringReference = yes;
        button2LocalizedStringEvent.StringReference = no;
    }
}

private void DisablePopup()
{

```

```

        popupScreen.SetActive(false);
        _callback?.Invoke(Result);
    }
}

public enum PopupType
{
    Error, EndTesting
}

public enum PopupResult
{
    None, Yes, No, Ok
}
}

```

Б.24 Код файла TrainingCalculatorController.cs

```

using System;
using System.Collections.Generic;
using System.Globalization;
using App.Scripts.Utilities.Enumerations;
using App.Scripts.Utilities.Helpers;
using UnityEngine;
using UnityEngine.Localization;
using UnityEngine.Localization.Components;
using UnityEngine.UI;

namespace App.Scripts.Controllers.Training
{
    public class TrainingCalculatorController : MonoBehaviour
    {
        [SerializeField] private List<LocalizedString> prefixesTranslations;

        [SerializeField] private LocalizeStringEvent prefixPlaceholder1;
        [SerializeField] private LocalizeStringEvent prefixPlaceholder2;
        [SerializeField] private Text powerPlaceholder;
        [SerializeField] private Text readonlyPowerPlaceholder;

        [SerializeField] private Text resultLabel;
        [SerializeField] private InputField numberInput;

        [SerializeField] private ValueSwitcherController
prefix1ValueSwitcher;
        [SerializeField] private ValueSwitcherController
prefix2ValueSwitcher;
        [SerializeField] private ValueSwitcherController powerValueSwitcher;

        private readonly List<Prefix> _prefixes = new()
        {
            Prefix.Pico,
            Prefix.Nano,
            Prefix.Milli,
            Prefix.None,
            Prefix.Kilo,
            Prefix.Mega,
            Prefix.Giga
        };

        private void Awake()
        {
            prefix1ValueSwitcher.OnSwitchValue += OnPrefix1ValueSwitch;
            prefix2ValueSwitcher.OnSwitchValue += OnPrefix2ValueSwitch;

```

```

        powerValueSwitcher.OnSwitchValue += OnPowerValueSwitch;

numberInput.OnValueChanged.AddListener(OnNumberInputValueChanged);
    }

    private void OnDestroy()
    {
        prefix1ValueSwitcher.OnSwitchValue -= OnPrefix1ValueSwitch;
        prefix2ValueSwitcher.OnSwitchValue -= OnPrefix2ValueSwitch;
        powerValueSwitcher.OnSwitchValue -= OnPowerValueSwitch;

numberInput.OnValueChanged.RemoveListener(OnNumberInputValueChanged);
    }

    private void OnPrefix1ValueSwitch(int value)
    {
        Calculate();
    }

    private void OnPrefix2ValueSwitch(int value)
    {
        Calculate();
    }

    private void OnPowerValueSwitch(int value)
    {
        Calculate();
    }

    private void OnNumberInputValueChanged(string value)
    {
        Calculate();
    }

    private void Calculate()
    {
        var powerValue = powerValueSwitcher.GetValue() + 1;
        var prefix1Value = prefix1ValueSwitcher.GetValue();
        var prefix2Value = prefix2ValueSwitcher.GetValue();

        var prefix1 = _prefixes[prefix1Value];
        var prefix2 = _prefixes[prefix2Value];
        var power = (Power) powerValue;

        var prefix1Text = prefixesTranslations[prefix1Value];
        var prefix2Text = prefixesTranslations[prefix2Value];

        prefixPlaceholder1.StringReference = prefix1Text;
        prefixPlaceholder2.StringReference = prefix2Text;
        powerPlaceholder.text = powerValue.ToString();
        readonlyPowerPlaceholder.text = powerValue == 1 ? "" :
powerValue.ToString();

        try
        {
            var number = double.Parse(numberInput.text);
            var result = CalculationHelper.ConvertPrefix(number, prefix1,
prefix2, power);
            resultLabel.text =
result.ToString(CultureInfo.InvariantCulture);
        }
        catch (Exception e)
        {
            resultLabel.text = "-";
        }
    }

```

```

    }
}

void Start()
{
    if (prefixesTranslations.Count != _prefixes.Count)
    {
        throw new Exception("TrainingCalculatorController
initialization error");
    }
    Calculate();
}
}
}

```

Б.25 Код файла TrainingController.cs

```

using System.Collections;
using App.Scripts.Managers;
using UnityEngine;

namespace App.Scripts.Controllers.Training
{
    public class TrainingController : ScreenController
    {
        public static TrainingController Instance { get; private set; }

        protected override void Awake()
        {
            base.Awake();
            Instance = this;
        }

        public void ClickBack()
        {
            IEnumerator Delayed()
            {
                yield return new WaitForSeconds(1f);
                AppStateManager.Instance.UpdateState(AppState.Menu);
            }

            AppStateManager.Instance.UpdateState(AppState.Loading);
            StartCoroutine(Delayed());
        }
    }
}

```

Б.26 Код файла ValueSwitcherController.cs

```

using System;
using UnityEngine;

namespace App.Scripts.Controllers.Training
{
    public class ValueSwitcherController : MonoBehaviour
    {
        [SerializeField] private int numberOfValues;

        public event Action<int> OnSwitchValue;
    }
}

```

```

private int _currentIndex = -1;

public void Start()
{
    Switch();
}

public void Switch()
{
    if (_currentIndex == numberOfValues - 1)
    {
        _currentIndex = 0;
    }
    else
    {
        _currentIndex++;
    }
    OnSwitchValue?.Invoke(_currentIndex);
}

public int GetNumberOfValues()
{
    return numberOfValues;
}

public int GetValue()
{
    return _currentIndex == -1 ? 0 : _currentIndex;
}
}

```

Б.27 Код файла ValueSwitcherLineController.cs

```

using System;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Localization;
using UnityEngine.Localization.Components;

namespace App.Scripts.Controllers.Training
{
    public class ValueSwitcherLineController : MonoBehaviour
    {
        [SerializeField] private ValueSwitcherController
valueSwitcherController;

        [SerializeField] private List<LocalizedString> valuesList1;
        [SerializeField] private List<LocalizedString> valuesList2;

        [SerializeField] private LocalizeStringEvent stringPlaceholder1;
        [SerializeField] private LocalizeStringEvent stringPlaceholder2;

        private void Awake()
        {
            valueSwitcherController.OnSwitchValue += OnSwitchValue;
        }

        private void OnDestroy()
        {
            valueSwitcherController.OnSwitchValue -= OnSwitchValue;
        }
    }
}

```

```

        private void Start()
        {
            var valuesNumberSupported =
valueSwitcherController.GetNumberOfValues();
            if (valuesList1.Count != valuesNumberSupported ||
valuesList2.Count != valuesNumberSupported)
            {
                throw new Exception("ValueSwitcherLine controller
initialization error");
            }
        }

        private void OnSwitchValue(int index)
        {
            stringPlaceholder1.StringReference = valuesList1[index];
            stringPlaceholder2.StringReference = valuesList2[index];
        }
    }
}

```

Б.28 Код файлу LoadingController.cs

```

using System.Collections;
using UnityEngine;

namespace App.Scripts.Controllers
{
    public class LoadingController : ScreenController
    {
        [SerializeField] private RectTransform loadingCircle;
        private float _rotateSpeed = -30f;

        IEnumerator Rotate()
        {
            while (true)
            {
                loadingCircle.Rotate(0f, 0f, _rotateSpeed);
                yield return new WaitForSeconds(0.1f);
            }
        }

        protected override void OnEnterScreen()
        {
            StopCoroutine(Rotate());
            StartCoroutine(Rotate());
        }

        protected override void OnExitScreen() { }
    }
}

```

Б.29 Код файлу ScreenController.cs

```

using App.Scripts.Managers;
using UnityEngine;

namespace App.Scripts.Controllers
{
    public abstract class ScreenController : MonoBehaviour
    {
        [SerializeField] private AppState stateOnWhichDepends;
    }
}

```

```

[SerializeField] private GameObject screen;
protected virtual void Awake ()
{
    HideScreen ();

    AppStateManager.OnAppStateChange += OnAppStateChange;
    AppStateManager.OnAppStateExit += OnAppStateExit;
}

protected virtual void OnDestroy ()
{
    AppStateManager.OnAppStateChange -= OnAppStateChange;
    AppStateManager.OnAppStateExit -= OnAppStateExit;
}

private void OnAppStateChange (AppState state)
{
    if (state == stateOnWhichDepends)
    {
        ShowScreen ();
        OnEnterScreen ();
    }
}

private void OnAppStateExit (AppState state)
{
    if (state == stateOnWhichDepends)
    {
        OnExitScreen ();
        HideScreen ();
    }
}

private void ShowScreen ()
{
    screen.SetActive (true);
}
private void HideScreen ()
{
    screen.SetActive (false);
}

protected virtual void OnEnterScreen () { }

protected virtual void OnExitScreen () { }
}

```

Б.30 Код файла Level.cs

```

using System;
using System.Collections.Generic;

namespace App.Scripts.Models
{
    public class Level
    {
        public long Id { get; set; }
        public string Name { get; set; }

        public int NumberOfQuestions { get; set; }

        public bool IsAvailable { get; set; } = false;
    }
}

```

```

        public bool IsCompleted { get; set; } = false;

        public List<LevelTry> Tries { get; set; }

        public DateTime TimeLimit { get; set; }
    }
}

```

Б.31 Код файла LevelTry.cs

```

using System;

namespace App.Scripts.Models
{
    public class LevelTry
    {
        public DateTime TriedAt { get; set; }

        public int CompletedQuestions { get; set; }
    }
}

```

Б.32 Код файла Question.cs

```

using System.Collections.Generic;

namespace App.Scripts.Models
{
    public class Question
    {
        public string Text { get; set; }

        public List<string> AcceptableAnswers { get; set; }

        public List<string> AnswersVariants { get; set; }
    }
}

```

Б.33 Код файла Stage.cs

```

using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Localization;

namespace App.Scripts.Models
{
    public class Stage
    {
        public long Id { get; set; }
        public string Name { get; set; } = null!;

        public bool IsAvailable { get; set; }

        public float Progress { get; set; }

        public List<Level> Levels { get; set; }

        public LocalizedString LocalizedName { get; set; }
    }
}

```

```

        public Sprite Logo { get; set; }
    }
}

```

Б.34 Код файла Subject.cs

```

using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Localization;

namespace App.Scripts.Models
{
    public class Subject
    {
        public long Id { get; set; }
        public string Name { get; set; } = null!;

        public bool IsAvailable { get; set; }

        public List<Stage> Stages { get; set; }

        public float Progress { get; set; }

        public LocalizedString LocalizedName { get; set; }

        public Sprite Logo { get; set; }
    }
}

```

Б.35 Код файла Power.cs

```

namespace App.Scripts.Utilities.Enumerations
{
    public enum Power
    {
        One = 1,
        Two = 2,
        Three = 3
    }
}

```

Б.36 Код файла Prefix.cs

```

namespace App.Scripts.Utilities.Enumerations
{
    public enum Prefix
    {
        Pico = -12,
        Nano = -9,
        Milli = -3,
        None = 0,
        Kilo = 3,
        Mega = 6,
        Giga = 9
    }
}

```

Б.37 Код файла ApiHelper.cs

```

using System;
using System.Threading.Tasks;
using Localization;
using StemLearning.Shared.Dto.Authorization;
using StemLearning.Shared.Dto.Levels;
using StemLearning.Shared.Dto.Stages;
using StemLearning.Shared.Dto.Subjects;
using UnityEngine;

namespace App.Scripts.Utilities.Helpers
{
    public static class ApiHelper
    {
        private static readonly RestHttpClient HttpClient = new
RestHttpClient("https://localhost:5000/");
        public static async Task<AuthApiResponseDto>
Register(RegisterRequestDto registerRequestDto)
        {
            var responseDto = await
HttpClient.Post<AuthApiResponseDto>("api/auth/register", registerRequestDto);
            PlayerPrefs.SetString("Token", responseDto.Token);
            return responseDto;
        }

        public static async Task<AuthApiResponseDto> Login(LogInApiRequestDto
logInRequestDto)
        {
            var responseDto = await
HttpClient.Post<AuthApiResponseDto>("api/auth/login", logInRequestDto);
            PlayerPrefs.SetString("Token", responseDto.Token);
            return responseDto;
        }

        public static async Task<GetSubjectsApiResponseDto> GetSubjects()
        {
            return await
HttpClient.Get<GetSubjectsApiResponseDto>("api/subjects");
        }

        public static async Task<GetStagesApiResponseDto> GetStages(long
subjectId)
        {
            return await
HttpClient.Get<GetStagesApiResponseDto>($"api/stages?subjectId={subjectId}");
        }

        public static async Task<GetLevelsApiResponseDto> GetLevels(long
stageId)
        {
            return await
HttpClient.Get<GetLevelsApiResponseDto>($"api/levels?stageId={stageId}");
        }

        public static async Task<GetLevelApiResponse> GetLevel(long levelId,
Locale locale)
        {
            var localeString = locale == Locale.Ukrainian ? "ua" : "en";
            return await
HttpClient.Get<GetLevelApiResponse>($"api/levels/{levelId}?locale={localeStri
ng}");
        }

        public static async Task<TryLevelApiResponse> TryLevel(long levelId,
int numberOfCompletedQuestions,

```

```

        DateTime triedAt)
    {
        var body = new TryLevelApiRequest()
        {
            NumberOfCompletedQuestions = numberOfCompletedQuestions,
            TriedAt = triedAt
        };
        return await
HttpClient.Post<TryLevelApiResponse>($"api/levels/{levelId}", body);
    }
}

```

Б.38 Код файла CalculationHelper.cs

```

using System;
using App.Scripts.Utilities.Enumerations;

namespace App.Scripts.Utilities.Helpers
{
    public static class CalculationHelper
    {
        public static double ConvertPrefix(double value, Prefix
currentPrefix, Prefix targetPrefix, Power power)
        {
            var powerValue = (int) power;

            var prefixIndependentValue1 = Math.Pow(value * Math.Pow(10, (int)
currentPrefix), powerValue);
            var prefixIndependentValue2 = Math.Pow(Math.Pow(10, (int)
targetPrefix), powerValue);

            return prefixIndependentValue1 / prefixIndependentValue2;
        }
    }
}

```

Б.39 Код файла RestHttpClient.cs

```

using System;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Threading.Tasks;
using Newtonsoft.Json;
using UnityEngine;

namespace App.Scripts.Utilities
{
    public class RestHttpClient : IDisposable
    {
        private readonly HttpClient _httpClient;
        private readonly JsonSerializerSettings _jsonSerializerSettings;
        private readonly string _url;

        public RestHttpClient(string url)
        {
            _httpClient = new HttpClient();
            _jsonSerializerSettings = new JsonSerializerSettings
            {
                NullValueHandling = NullValueHandling.Ignore
            };
        }
    }
}

```

```

        _url = url;
    }

    public async Task<T> Get<T>(string path)
    {
        PopulateWithToken();
        HttpResponseMessage response = await _httpClient.GetAsync(_url +
path);

        if (response.IsSuccessStatusCode)
        {
            string responseContent = await
response.Content.ReadAsStringAsync();
            T result = JsonConvert.DeserializeObject<T>(responseContent,
_jsonSerializerSettings);
            return result;
        }
        else
        {
            throw new HttpRequestException("GET request failed with
status code: " + response.StatusCode);
        }
    }

    public async Task<T> Post<T>(string path, object data = null)
    {
        PopulateWithToken();
        var json = JsonConvert.SerializeObject(data,
_jsonSerializerSettings);
        var content = new StringContent(json, System.Text.Encoding.UTF8,
"application/json");
        HttpResponseMessage response = await _httpClient.PostAsync(_url +
path, content);

        if (response.IsSuccessStatusCode)
        {
            string responseContent = await
response.Content.ReadAsStringAsync();
            T result = JsonConvert.DeserializeObject<T>(responseContent,
_jsonSerializerSettings);
            return result;
        }
        else
        {
            throw new HttpRequestException("POST request failed with
status code: " + response.StatusCode);
        }
    }

    private void PopulateWithToken()
    {
        var jwtToken = PlayerPrefs.GetString("Token");
        if (jwtToken != string.Empty)
        {
            _httpClient.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", jwtToken);
        }
    }

    public void Dispose()
    {
        _httpClient.Dispose();
    }

```

```
}
}
```

Проект вебсерверу.

Б.40 Код файлу Program.cs

```
using Serilog;
using Serilog.Events;
using Serilog.Sinks.SystemConsole.Themes;

namespace StemLearning.WebApi;

public static class Program
{
    public static Task Main(string[] args)
    {
        try
        {
            return CreateHostBuilder(args).Build().RunAsync();
        }
        catch (Exception e)
        {
            Console.WriteLine("App exited with error {0}", e);
            throw;
        }
    }

    private static IHostBuilder CreateHostBuilder(string[] args)
    {
        return Host
            .CreateDefaultBuilder(args)
            .ConfigureAppConfiguration((ctx, builder) =>
            {
                builder.AddUserSecrets<Startup>(true);
                builder.AddEnvironmentVariables();
            })
            .UseSerilog((context, services, loggerConfiguration) =>
            {
                var logOutputTemplate = "[{Timestamp:HH:mm:ss.fff}] "
                    + "[{RequestId}] "
                    + "[{SourceContext:l}] "
                    + "[{Level:u3}] "
                    + "{Message:l}{NewLine}{Exception}";

                loggerConfiguration
                    .ReadFrom.Configuration(context.Configuration)
                    .Enrich.FromLogContext()
                    .Enrich.WithProperty("Environment",
context.HostingEnvironment.EnvironmentName)
                    .Enrich.WithThreadId();

                loggerConfiguration.WriteTo.Console(
                    outputTemplate: logOutputTemplate,
                    theme: AnsiConsoleTheme.Literate,
                    restrictedToMinimumLevel: LogEventLevel.Debug);
            })
            .ConfigureWebHostDefaults(webBuilder => {
webBuilder.UseStartup<Startup>(); });
```

```
}
}
```

Б.41 Код файла Startup.cs

```
using System.Text;
using FluentValidation;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Internal;
using Microsoft.IdentityModel.Tokens;
using Newtonsoft.Json;
using Newtonsoft.Json.Converters;
using Newtonsoft.Json.Serialization;
using StemLearning.WebApi.Config;
using StemLearning.WebApi.Config.Interfaces;
using StemLearning.WebApi.KnowledgeBases;
using StemLearning.WebApi.Persistence;
using StemLearning.WebApi.Persistence.Configuration;
using StemLearning.WebApi.Persistence.Repositories;
using StemLearning.WebApi.Persistence.Repositories.Interfaces;
using StemLearning.WebApi.Utilities.Extensions;
using StemLearning.WebApi.Utilities.Questions;

namespace StemLearning.WebApi;

public class Startup
{
    private readonly bool _isDevelopment;

    public Startup(IConfiguration configurationRoot, IWebHostEnvironment
hostEnvironment)
    {
        _isDevelopment = hostEnvironment.IsDevelopment();
        ConfigurationRoot = configurationRoot;
        Configuration = new ApplicationConfig();
        ConfigurationRoot.Bind(Configuration);
    }

    private IConfiguration ConfigurationRoot { get; }

    private IApplicationConfig Configuration { get; }

    public void ConfigureServices(IServiceCollection services)
    {
        services.RegisterConfig(Configuration);

        services.AddLogging();

        services.AddMediatR(x =>
x.RegisterServicesFromAssemblyContaining<Startup>());
        services.AddValidatorsFromAssemblyContaining<Startup>();

        services.AddHttpContextAccessor();
        services.AddControllers()
            .AddNewtonsoftJson(ConfigureJson);

        services.AddSwaggerGen();

        Configuration.Database.Use(config =>
        {
```

```

services.AddDbContext<StemLearningDbContext>(dbContextOptionsBuilder =>
{
    if (_isDevelopment)
dbContextOptionsBuilder.EnableSensitiveDataLogging();

    dbContextOptionsBuilder
        .UseSqlServer(config.ConnectionString, builder =>
builder
            .EnableRetryOnFailure(config.Retries)
            .CommandTimeout(config.Timeout)
            );
    }, ServiceLifetime.Transient
);
});
services.AddEntityConfigurationsFromAssemblyWith<Startup>();
services.AddRepositories();

services.AddGenerators();

services.AddAuthentication(options =>
{
    options.DefaultAuthenticateScheme =
JwtBearerDefaults.AuthenticationScheme;
    options.DefaultScheme =
JwtBearerDefaults.AuthenticationScheme;
    options.DefaultChallengeScheme =
JwtBearerDefaults.AuthenticationScheme;
})
.AddJwtBearer(options =>
{
    var key = Encoding.UTF8.GetBytes(Configuration.Jwt.Key);

    options.SaveToken = true;
    options.TokenValidationParameters = new
TokenValidationParameters()
    {
        ValidateIssuerSigningKey = true,
        IssuerSigningKey = new SymmetricSecurityKey(key),
        ValidateIssuer = false, // for dev
        ValidateAudience = false, // for dev
        RequireExpirationTime = false, // for dev -- needs to be
updated when refresh token is added
        ValidateLifetime = true
    };
});

services.AddSingleton<ISystemClock, SystemClock>();
}

public void Configure(IApplicationBuilder app,
    StemLearningDbContext dbContext)
{
    dbContext.Database.Migrate();

    app.UseSwagger();
    app.UseSwaggerUI();

    if (_isDevelopment) app.UseDeveloperExceptionPage();

    app.UseHttpsRedirection();

    app.UseRouting();
    app.UseAuthentication();
    app.UseAuthorization();

```

```

        app.UseEndpoints(endpoints => { endpoints.MapControllers(); });
    }

    private static void ConfigureJson(MvcNewtonsoftJsonOptions x)
    {
        var settings = x.SerializerSettings;
        settings.ContractResolver = new
        CamelCasePropertyNamesContractResolver();
        settings.Converters.Add(new StringEnumConverter { NamingStrategy =
        new CamelCaseNamingStrategy() });
        settings.NullValueHandling = NullValueHandling.Include;
    }
}

```

Б.42 Код файла ApplicationConfig.cs

```

using StemLearning.WebApi.Config.Interfaces;

namespace StemLearning.WebApi.Config;

public class ApplicationConfig : IApplicationConfig
{
    public IDatabaseConfig Database { get; } = new DatabaseConfig();
    public IJwtConfig Jwt { get; } = new JwtConfig();
}

```

Б.43 Код файла DatabaseConfig.cs

```

using StemLearning.WebApi.Config.Interfaces;
namespace StemLearning.WebApi.Config;

public class DatabaseConfig : IDatabaseConfig
{
    public string ConnectionString { get; set; } = null!;
    public int Retries { get; set; }
    public int Timeout { get; set; }
}

```

Б.44 Код файла JwtConfig.cs

```

using StemLearning.WebApi.Config.Interfaces;

namespace StemLearning.WebApi.Config;

public class JwtConfig : IJwtConfig
{
    public string Issuer { get; set; } = null!;
    public string Audience { get; set; } = null!;
    public string Authority { get; set; } = null!;
    public string Key { get; set; } = null!;
}

```

Б.45 Код файла ChemistryKnowledgeBase.cs

```

using System.Globalization;
using System.Text.RegularExpressions;
using StemLearning.WebApi.Models;
using StemLearning.WebApi.Utilities.Enumerations;

```

```

using StemLearning.WebApi.Utilities.Helpers;

namespace StemLearning.WebApi.KnowledgeBases.Chemistry;

public class ChemistryKnowledgeBase : IChemistryKnowledgeBase
{
    private readonly HttpClient _httpClient;
    private readonly string _serverUrl = "http://localhost:8080/";

    private readonly string _unitQuantityUrlPath;
    private readonly string _howManyUrlPath;
    private readonly string _prefixesComparisonUrlPath;

    public ChemistryKnowledgeBase()
    {
        _httpClient = new HttpClient();

        _unitQuantityUrlPath = _serverUrl +
"metr_dim_analysis/unit_quantity?";
        _howManyUrlPath = _serverUrl + "metr_dim_analysis?goal=how_many&";
        _prefixesComparisonUrlPath = _serverUrl +
"metr_dim_analysis?goal=compare_prefixes&";
    }

    public async Task<Unit> GetUnit(Measure measure)
    {
        var convertedQuantity = EnumHelper.ToString(measure);
        if (measure == Measure.Quantity)
        {
            convertedQuantity = "amount of substance, quantity";
        }

        var response = await _httpClient.GetAsync(_unitQuantityUrlPath +
$"quantity={convertedQuantity}");
        var responseString = await response.Content.ReadAsStringAsync();
        var trimmedString = responseString.Trim(' ', '\n');

        Unit result;
        switch (trimmedString)
        {
            case "m":
                result = Unit.Meter;
                break;
            case "kg":
                result = Unit.Gram;
                break;
            case "s":
                result = Unit.Second;
                break;
            case "mol":
                result = Unit.Mole;
                break;
            default:
                result = (Unit)Enum.Parse(typeof(Unit), trimmedString);
                break;
        }

        return result;
    }

    public async Task<Measure> GetMeasure(Unit unit)
    {
        var convertedUnit = EnumHelper.ToString(unit);
        var response = await _httpClient.GetAsync(_unitQuantityUrlPath +
$"unit={convertedUnit}");
    }
}

```

```

var responseString = await response.Content.ReadAsStringAsync();
var trimmedString = responseString.Trim(' ', '\');

Measure result;
if (trimmedString == "amount of substance, quantity")
{
    result = Measure.Quantity;
}
else
{
    TextInfo textInfo = CultureInfo.CurrentCulture.TextInfo;
    string capitalizedStr = textInfo.ToTitleCase(trimmedString);

    result = (Measure)Enum.Parse(typeof(Measure), capitalizedStr);
}

return result;
}

public async Task<string> HowMany(Prefix prefix1, Prefix prefix2, double
value = 1, Unit unit = Unit.Meter, Power power = Power.First)
{
    var unitArg = EnumHelper.ToString(unit);
    var prefix1Arg = EnumHelper.ToString(prefix1);
    var prefix2Arg = EnumHelper.ToString(prefix2);
    var powerArg = (int) power;
    var response = await _httpClient.GetAsync(_howManyUrlPath +
$"args={prefix1Arg}&args={value}&args={prefix2Arg}&args={unitArg}^{powerArg}"
);
    var responseString = await response.Content.ReadAsStringAsync();

    string numberPattern = @"^\d+\.\d+$";
    if (Regex.IsMatch(responseString, numberPattern))
    {
        var dec = decimal.Parse(responseString);
        if (dec > 1)
        {
            responseString = Math.Round(dec).ToString();
        }
    }
    return responseString;
}

public async Task<Comparison> Compare(MetricNumber metricNumber1,
MetricNumber metricNumber2, Unit unit = Unit.Meter,
Power power = Power.First)
{
    var unitArg = EnumHelper.ToString(unit);
    var prefix1Arg = EnumHelper.ToString(metricNumber1.Prefix);
    var prefix2Arg = EnumHelper.ToString(metricNumber2.Prefix);
    var powerArg = (int) power;

    var response = await _httpClient.GetAsync(_prefixesComparisonUrlPath
+
$"args={metricNumber1.Value}&args={prefix1Arg}&args={metricNumber2.Value}&arg
s={prefix2Arg}&args={unitArg}^{powerArg}");
    var responseString = await response.Content.ReadAsStringAsync();
    var trimmedString = responseString.Trim(' ', '\');

    switch (trimmedString)
    {
        case ">":
            return Comparison.Bigger;
        case "<":

```

```

        return Comparison.Less;
    case "=":
        return Comparison.Equal;
    default:
        throw new Exception("Unexpected server response");
    }
}

public async Task<MetricNumber> GetLarger(MetricNumber metricNumber1,
MetricNumber metricNumber2, Unit unit = Unit.Meter,
Power power = Power.First)
{
    var comparisonResult = await Compare(metricNumber1, metricNumber2,
unit, power);
    if (comparisonResult == Comparison.Bigger) return metricNumber1;
    if (comparisonResult == Comparison.Less) return metricNumber2;
    throw new Exception("Operands are equal");
}

public async Task<MetricNumber> GetSmaller(MetricNumber metricNumber1,
MetricNumber metricNumber2, Unit unit = Unit.Meter,
Power power = Power.First)
{
    var comparisonResult = await Compare(metricNumber1, metricNumber2,
unit, power);
    if (comparisonResult == Comparison.Bigger) return metricNumber2;
    if (comparisonResult == Comparison.Less) return metricNumber1;
    throw new Exception("Operands are equal");
}
}

```

Б.46 Код файла JwtTokenInfoDto.cs

```

namespace StemLearning.WebApi.Models.Dto;

public class JwtTokenInfoDto
{
    public string UserId { get; set; } = null!;

    public string UserEmail { get; set; } = null!;

    public string ConfigKey { get; set; } = null!;
}

```

Б.47 Код файла RegisterDto.cs

```

namespace StemLearning.WebApi.Models.Dto;

public class RegisterDto
{
    public string Email { get; set; } = null!;

    public string Password { get; set; } = null!;

    public string Name { get; set; } = null!;
}

```

Б.48 Код файла Question.cs

```
namespace StemLearning.WebApi.Models.Question;

public class Question
{
    public string Text { get; set; } = null!;

    public List<string> AcceptableAnswers { get; set; } = null!;

    public List<string> AnswersVariants { get; set; } = null!;
}
```

Б.49 Код файла MetricNumber.cs

```
using StemLearning.WebApi.Utilities.Enumerations;

namespace StemLearning.WebApi.Models;

public class MetricNumber
{
    public double Value { get; set; }

    public Prefix Prefix { get; set; }
}
```

Б.50 Код файла Login.cs

```
using MediatR;
using Microsoft.AspNetCore.Mvc;
using StemLearning.Shared.Dto.Authorization;
using StemLearning.WebApi.Config.Interfaces;
using StemLearning.WebApi.Models.Dto;
using StemLearning.WebApi.Persistance;
using StemLearning.WebApi.Utilities.Helpers;

namespace StemLearning.WebApi.Operations.Commands;

public class LogInCommand : IRequest<IActionResult>
{
    public string Email { get; set; } = null!;

    public string Password { get; set; } = null!;
}

public class LogInCommandHandler : IRequestHandler<LogInCommand, IActionResult>
{
    private readonly StemLearningDbContext _dbContext;
    private readonly IJwtConfig _jwtConfig;

    public LogInCommandHandler(StemLearningDbContext dbContext, IJwtConfig jwtConfig)
    {
        _dbContext = dbContext;
        _jwtConfig = jwtConfig;
    }

    public Task<IActionResult> Handle(LogInCommand request, CancellationToken cancellationToken)
    {
        try
```

```

        {
            var user = _dbContext.Users.First(u => u.Email == request.Email);
            if (PasswordHelper.VerifyPasswordHash(request.Password,
user.PasswordHash, user.PasswordSalt))
            {
                return Task.FromResult<IActionResult>(new OkObjectResult(new
AuthApiResponseDto()
                {
                    Token = JwtHelper.GenerateToken(new JwtTokenInfoDto()
                    {
                        UserId = user.Id.ToString(),
                        userEmail = user.Email,
                        ConfigKey = _jwtConfig.Key
                    }),
                    Email = user.Email,
                    Name = user.Name
                }));
            }
            else
            {
                return Task.FromResult<IActionResult>(new
UnauthorizedResult());
            }
        }
        catch (Exception e)
        {
            return Task.FromResult<IActionResult>(new UnauthorizedResult());
        }
    }
}

```

Б.51 Код файла Register.cs

```

using MediatR;
using Microsoft.AspNetCore.Mvc;
using StemLearning.Shared.Dto.Authorization;
using StemLearning.WebApi.Config.Interfaces;
using StemLearning.WebApi.Models.Dto;
using StemLearning.WebApi.Persistence;
using StemLearning.WebApi.Utilities.Factories;
using StemLearning.WebApi.Utilities.Helpers;

namespace StemLearning.WebApi.Operations.Commands;

public class RegisterCommand: IRequest<IActionResult>
{
    public string Email { get; set; } = null!;

    public string Password { get; set; } = null!;

    public string Name { get; set; } = null!;
}

public class RegisterCommandHandler: IRequestHandler<RegisterCommand,
IActionResult>
{
    private readonly StemLearningDbContext _dbContext;
    private readonly IJwtConfig _jwtConfig;

    public RegisterCommandHandler(StemLearningDbContext dbContext, IJwtConfig
jwtConfig)
    {
        _dbContext = dbContext;
    }
}

```

```

        _jwtConfig = jwtConfig;
    }
    public async Task<IActionResult> Handle(RegisterCommand request,
        CancellationToken cancellationToken)
    {
        try
        {
            var userFactory = new UserFactory();
            var user = userFactory.Create(new RegisterDto()
            {
                Email = request.Email,
                Password = request.Password,
                Name = request.Name
            });

            _dbContext.Users.Add(user);
            await _dbContext.SaveChangesAsync(cancellationToken);

            return new OkObjectResult(new AuthApiResponseDto()
            {
                Token = JwtHelper.GenerateToken(new JwtTokenInfoDto()
                {
                    UserId = user.Id.ToString(),
                    userEmail = user.Email,
                    ConfigKey = _jwtConfig.Key
                },
                Name = request.Name,
                Email = request.Email
            ));
        }
        catch (Exception e)
        {
            return new BadRequestResult();
        }
    }
}

```

Б.52 Код файлу TryLevel.cs

```

using MediatR;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using StemLearning.Shared.Dto.Levels;
using StemLearning.Shared.Dto.Stages;
using StemLearning.Shared.Dto.Subjects;
using StemLearning.WebApi.Persistence.Repositories.Interfaces;

namespace StemLearning.WebApi.Operations.Commands;

public class TryLevelCommand : IRequest<IActionResult>
{
    public long LevelId { get; set; }

    public int NumberOfCompletedQuestions { get; set; }

    public DateTime TriedAt { get; set; }
}

public class TryLevelCommandHandler : IRequestHandler<TryLevelCommand,
    IActionResult>
{
    private readonly ILevelsRepository _levelsRepository;
    private readonly IStagesRepository _stagesRepository;
}

```

```

private readonly ISubjectsRepository _subjectsRepository;

public TryLevelCommandHandler(ILevelsRepository levelsRepository,
IStagesRepository stagesRepository, ISubjectsRepository subjectsRepository)
{
    _levelsRepository = levelsRepository;
    _stagesRepository = stagesRepository;
    _subjectsRepository = subjectsRepository;
}

public async Task<IActionResult> Handle(TryLevelCommand request,
CancellationTokens cancellationTokens)
{
    var level = _levelsRepository.Level(request.LevelId);
    if (level.NumberOfQuestions < request.NumberOfCompletedQuestions)
    {
        throw new Exception("Invalid number of completed questions");
    }

    var levelTry = await _levelsRepository.TryLevel(request.LevelId,
request.NumberOfCompletedQuestions, request.TriedAt,
cancellationTokens);

    var levelProgressAfterTry = await
_levelsRepository.GetOrCreateLevelProgress(request.LevelId,
cancellationTokens);

    var availableLevels = new List<LevelApiDto>();

    if (levelProgressAfterTry.IsCompleted)
    {
        var userLevels = await
_levelsRepository.Levels(level.StageId).ToListAsync(cancellationTokens);
        var levelIndex = userLevels.IndexOf(level);
        if (levelIndex != userLevels.Count() - 1)
        {
            var newAvailableLevel = userLevels[levelIndex + 1];
            availableLevels.Add(new LevelApiDto()
            {
                Id = newAvailableLevel.Id
            });
        }
    }

    var stage = _stagesRepository.Stage(level.StageId);
    var stageProgressAfterTry = await
_stagesRepository.GetOrCreateStageProgress(stage.Id, cancellationTokens);

    var subject = _subjectsRepository.Subject(stage.SubjectId);
    var subjectProgressAfterTry = await
_subjecsRepository.GetOrCreateSubjectProgress(subject.Id,
cancellationTokens);

    return new OkObjectResult(new TryLevelApiResponse()
    {
        AvailableLevels = availableLevels,
        IsCompleted = levelProgressAfterTry.IsCompleted,
        LevelTry = new LevelTryApiDto()
        {
            CompletedQuestions = levelTry.CompletedQuestions,
            TriedAt = levelTry.TriedAt
        },
        SubjectProgress = new SubjectProgressApiDto()
        {

```

```

        Value = subjectProgressAfterTry.Progress
    },
    StageProgress = new StageProgressApiDto()
    {
        Value = stageProgressAfterTry.Progress
    }
});
}
}

```

Б.53 Код файла GetLevel.cs

```

using MediatR;
using Microsoft.AspNetCore.Mvc;
using StemLearning.Shared.Dto.Levels;
using StemLearning.WebApi.Persistence.Entities;
using StemLearning.WebApi.Persistence.Repositories.Interfaces;
using StemLearning.WebApi.Utilities.Factories;
using StemLearning.WebApi.Utilities.Questions;
using StemLearning.WebApi.Utilities.Questions.QuestionSetGenerator;

namespace StemLearning.WebApi.Operations.Queries;

public class GetLevelQuery: IRequest<IActionResult>
{
    public long LevelId { get; set; }

    public string Locale { get; set; } = null!;
}

public class GetLevelQueryHandler : IRequestHandler<GetLevelQuery,
IActionResult>
{
    private readonly IQuestionsSetGenerator _questionsSetGenerator;
    private readonly ILevelsRepository _levelsRepository;

    public GetLevelQueryHandler(IQuestionsSetGenerator questionsSetGenerator,
ILevelsRepository levelsRepository)
    {
        _questionsSetGenerator = questionsSetGenerator;
        _levelsRepository = levelsRepository;
    }

    public async Task<IActionResult> Handle(GetLevelQuery request,
Cancellation token cancellation token)
    {
        Level level;
        try
        {
            level = _levelsRepository.Level(request.LevelId);
        }
        catch (Exception e)
        {
            throw new Exception("Level is not exist");
        }

        var localeFactory = new LocaleFactory();
        var locale = localeFactory.Create(request.Locale);

        var questions = await _questionsSetGenerator
            .Level(level)
            .Locale(locale)

```

```

        .Generate();

var questionsDtos = questions.Select(x => new QuestionApiDto()
{
    Text = x.Text,
    AcceptableAnswers = x.AcceptableAnswers,
    AnswersVariants = x.AnswersVariants
}).ToList();

return new OkObjectResult(new GetLevelApiResponse()
{
    Questions = questionsDtos
});
}
}

```

Б.54 Код файла GetLevels.cs

```

using MediatR;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using StemLearning.Shared.Dto.Levels;
using StemLearning.Shared.Dto.Stages;
using StemLearning.WebApi.Persistence.Repositories.Interfaces;

namespace StemLearning.WebApi.Operations.Queries;

public class GetLevelsQuery : IRequest<IActionResult>
{
    public long StageId { get; set; }
}

public class GetLevelsQueryHandler : IRequestHandler<GetLevelsQuery,
IActionResult>
{
    private readonly ILevelsRepository _levelsRepository;

    public GetLevelsQueryHandler(ILevelsRepository levelsRepository)
    {
        _levelsRepository = levelsRepository;
    }

    public async Task<IActionResult> Handle(GetLevelsQuery request,
Cancellation token cancellationToken)
    {
        var levels = await
        _levelsRepository.Levels(request.StageId).ToListAsync(cancellationToken);
        var levelsDtos = levels.Select(x => new LevelApiDto()
        {
            Id = x.Id,
            Name = x.Name,
            NumberOfQuestions = x.NumberOfQuestions,
            TimeLimit = new DateTime(1, 1, 1, 1, 5, 0),
        }).ToList();

        var lastLevelIsPassed = true;
        foreach (var levelDto in levelsDtos)
        {
            var levelProgressProgress = await
            _levelsRepository.GetOrCreateLevelProgress(levelDto.Id, cancellationToken);
            levelDto.IsCompleted = levelProgressProgress.IsCompleted;

            var levelTries = await
            _levelsRepository.LevelTries(levelDto.Id).ToListAsync(cancellationToken);

```

```

        levelDto.Tries = levelTries.Select(x => new LevelTryApiDto()
        {
            CompletedQuestions = x.CompletedQuestions,
            TriedAt = x.TriedAt
        }).ToList();

        if (lastLevelIsPassed)
        {
            levelDto.IsAvailable = true;
            var successfulTries = levelDto.Tries.Where(x =>
x.CompletedQuestions == levelDto.NumberOfQuestions);
            if (!successfulTries.Any())
            {
                lastLevelIsPassed = false;
            }
        }
    }

    var response = new GetLevelsApiResponseDto()
    {
        Levels = levelsDtos
    };
    return new OkObjectResult(response);
}
}

```

Б.55 Код файла GetStages.cs

```

using MediatR;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using StemLearning.Shared.Dto.Stages;
using StemLearning.WebApi.Persistence.Repositories;
using StemLearning.WebApi.Persistence.Repositories.Interfaces;

namespace StemLearning.WebApi.Operations.Queries;

public class GetStagesQuery : IRequest<IActionResult>
{
    public long SubjectId { get; set; }
}

public class GetStagesQueryHandler : IRequestHandler<GetStagesQuery,
IActionResult>
{
    private readonly IStagesRepository _stagesRepository;

    public GetStagesQueryHandler(IStagesRepository stagesRepository)
    {
        _stagesRepository = stagesRepository;
    }

    public async Task<IActionResult> Handle(GetStagesQuery request,
Cancellation token cancellationToken)
    {
        var stages = await
        _stagesRepository.Stages(request.SubjectId).ToListAsync(cancellationToken);
        var stagesDtos = stages.Select(x => new StageApiDto()
        {
            Id = x.Id,
            Name = x.Name,
            IsAvailable = x.IsAvailable
        }).ToList();
    }
}

```

```

        foreach (var stageDto in stagesDtos)
        {
            var stageProgress = await
            _stagesRepository.GetOrCreateStageProgress(stageDto.Id, cancellationTokens);
            stageDto.Progress = new StageProgressApiDto()
            {
                Value = stageProgress.Progress
            };
        }

        var response = new GetStagesApiResponseDto()
        {
            Stages = stagesDtos
        };
        return new OkObjectResult(response);
    }
}

```

Б.56 Код файлу GetSubjects.cs

```

using MediatR;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using StemLearning.Shared.Dto.Subjects;
using StemLearning.WebApi.Persistence.Repositories;
using StemLearning.WebApi.Persistence.Repositories.Interfaces;

namespace StemLearning.WebApi.Operations.Queries;

public class GetSubjectsQuery : IRequest<IActionResult> { }

public class GetSubjectsQueryHandler : IRequestHandler<GetSubjectsQuery,
IActionResult>
{
    private readonly ISubjectsRepository _subjectsRepository;

    public GetSubjectsQueryHandler(ISubjectsRepository subjectsRepository)
    {
        _subjectsRepository = subjectsRepository;
    }

    public async Task<IActionResult> Handle(GetSubjectsQuery request,
CancellationTokens cancellationTokens)
    {
        var subjects = await
        _subjectsRepository.Subjects().ToListAsync(cancellationTokens:
cancellationTokens);
        var subjectsDtos = subjects.Select(x => new SubjectApiDto()
        {
            Id = x.Id,
            Name = x.Name,
            IsAvailable = x.IsAvailable
        }).ToList();

        foreach (var subjectDto in subjectsDtos)
        {
            var subjectProgress = await
            _subjectsRepository.GetOrCreateSubjectProgress(subjectDto.Id,
cancellationTokens);
            subjectDto.Progress = new SubjectProgressApiDto()
            {
                Value = subjectProgress.Progress
            };
        }
    }
}

```

```

    }

    var response = new GetSubjectsApiResponseDto()
    {
        Subjects = subjectsDtos
    };
    return new OkObjectResult(response);
}
}

```

Б.57 Код файла EntityConfigurator.cs

```

using Microsoft.EntityFrameworkCore;

namespace StemLearning.WebApi.Persistence.Configuration;

public class EntityConfigurator : IEntityConfigurator
{
    private readonly IEnumerable<IEntityTypeConfigurationTag>
    _configurations;
    private readonly ILogger<EntityConfigurator> _logger;

    public EntityConfigurator(IEnumerable<IEntityTypeConfigurationTag>
    configurations,
        ILogger<EntityConfigurator> logger)
    {
        _configurations = configurations;
        _logger = logger;
    }

    public void Configure(ModelBuilder modelBuilder)
    {
        _logger.LogDebug(
            "Configuring entities, {ConfigurationCount} to configure",
            _configurations.Count().ToString());

        foreach (var configuration in _configurations)
        {
            var configurationType = configuration.GetType();

            _logger.LogDebug("Configuring {ConfigurationType}",
            configurationType.Name);

            var configurationGenericInterface = configurationType
                .GetInterface(typeof(IEntityTypeConfiguration<>).Name);
            if (configurationGenericInterface is null)
                throw new InvalidOperationException(string.Format(
                    "Type {0} marked with {1} does not implement {2}",
                    configurationType,
                    nameof(IEntityTypeConfigurationTag),
                    typeof(IEntityTypeConfiguration<>).Name));

            var configuredType =
            configurationGenericInterface.GetGenericArguments()[0];
            var entityTypeBuilder = modelBuilder
                .GetType()
                .GetMethods()
                .First(x => x.Name == nameof(ModelBuilder.Entity) &&
                x.IsGenericMethodDefinition)
                .MakeGenericMethod(configuredType)
                .Invoke(modelBuilder, null);

            var configureMethod =

```

```

configurationType.GetMethod(nameof(IEntityTypeConfiguration<object>.Configure
));
    if (configureMethod is null)
        throw new InvalidOperationException(string.Format(
            "Type {0} doesn't contain Configure method from {1}
interface",
            configurationType,
            configurationGenericInterface.Name));

    configureMethod.Invoke(configuration, new[] {entityTypeBuilder});
}
}
}

```

Б.58 Код файла LevelConfiguration.cs

```

using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata.Builders;
using StemLearning.WebApi.Persistence.Entities;

namespace StemLearning.WebApi.Persistence.Configuration.Specific;

public class LevelConfiguration : IEntityTypeConfiguration<Level>,
IEntityTypeConfigurationTag
{
    public void Configure(EntityTypeBuilder<Level> builder)
    {
        builder.HasOne(x => x.Stage);

        builder.HasIndex(x => new {x.StageId, x.Name});
    }
}

```

Б.59 Код файла LevelProgressConfiguration.cs

```

using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata.Builders;
using StemLearning.WebApi.Persistence.Entities;

namespace StemLearning.WebApi.Persistence.Configuration.Specific;

public class LevelProgressConfiguration :
IEntityTypeConfiguration<LevelProgress>, IEntityTypeConfigurationTag
{
    public void Configure(EntityTypeBuilder<LevelProgress> builder)
    {
        builder.HasOne(x => x.Level);
        builder.HasOne(x => x.User);

        builder.HasIndex(x => new {x.LevelId, x.UserId});
    }
}

```

Б.60 Код файла LevelTryConfiguration.cs

```

using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata.Builders;
using StemLearning.WebApi.Persistence.Entities;

namespace StemLearning.WebApi.Persistence.Configuration.Specific;

```

```
public class LevelTryConfiguration : IEntityTypeConfiguration<LevelTry>,
IEntityTypeConfigurationTag
{
    public void Configure(EntityTypeBuilder<LevelTry> builder)
    {
        builder.HasOne(x => x.Level);
        builder.HasOne(x => x.User);
    }
}
```

Б.61 Код файла StageConfiguration.cs

```
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata.Builders;
using StemLearning.WebApi.Persistence.Entities;

namespace StemLearning.WebApi.Persistence.Configuration.Specific;

public class StageConfiguration : IEntityTypeConfiguration<Stage>,
IEntityTypeConfigurationTag
{
    public void Configure(EntityTypeBuilder<Stage> builder)
    {
        builder.HasIndex(x => new {x.Name, x.SubjectId}).IsUnique();
        builder.HasOne(x => x.Subject);
    }
}
```

Б.62 Код файла StageProgressConfiguration.cs

```
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata.Builders;
using StemLearning.WebApi.Persistence.Entities;

namespace StemLearning.WebApi.Persistence.Configuration.Specific;

public class StageProgressConfiguration :
IEntityTypeConfiguration<StageProgress>, IEntityTypeConfigurationTag
{
    public void Configure(EntityTypeBuilder<StageProgress> builder)
    {
        builder.HasOne(x => x.Stage);
        builder.HasOne(x => x.User);

        builder.HasIndex(x => new {x.StageId, x.UserId});
    }
}
```

Б.63 Код файла SubjectConfiguration.cs

```
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata.Builders;
using StemLearning.WebApi.Persistence.Entities;

namespace StemLearning.WebApi.Persistence.Configuration.Specific;

public class SubjectConfiguration : IEntityTypeConfiguration<Subject>,
IEntityTypeConfigurationTag
{
}
```

```

public void Configure(EntityTypeBuilder<Subject> builder)
{
    builder.HasIndex(x => x.Name).IsUnique();
}
}

```

Б.64 Код файла SubjectProgressConfiguration.cs

```

using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata.Builders;
using StemLearning.WebApi.Persistence.Entities;

namespace StemLearning.WebApi.Persistence.Configuration.Specific;

public class SubjectProgressConfiguration :
    IEntityTypeConfiguration<SubjectProgress>, IEntityTypeConfigurationTag
{
    public void Configure(EntityTypeBuilder<SubjectProgress> builder)
    {
        builder.HasOne(x => x.Subject);
        builder.HasOne(x => x.User);

        builder.HasIndex(x => new {x.SubjectId, x.UserId});
    }
}

```

Б.65 Код файла UserConfiguration.cs

```

using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata.Builders;
using StemLearning.WebApi.Persistence.Entities;

namespace StemLearning.WebApi.Persistence.Configuration.Specific;

public class UserConfiguration : IEntityTypeConfiguration<User>,
    IEntityTypeConfigurationTag
{
    public void Configure(EntityTypeBuilder<User> builder)
    {
        builder.HasIndex(u => u.Email).IsUnique();
    }
}

```

Б.66 Код файла Level.cs

```

using System.ComponentModel.DataAnnotations.Schema;
using StemLearning.WebApi.Persistence.Entities.Base;

namespace StemLearning.WebApi.Persistence.Entities;

public class Level : Entity
{
    [ForeignKey(nameof(Stage))] public long StageId { get; set; }
    public Stage Stage { get; set; } = null!;

    public string Name { get; set; } = null!;

    public int NumberOfQuestions { get; set; }
}

```

Б.67 Код файла LevelProgress.cs

```
using System.ComponentModel.DataAnnotations.Schema;
using StemLearning.WebApi.Persistence.Entities.Base;

namespace StemLearning.WebApi.Persistence.Entities;

public class LevelProgress : Entity
{
    [ForeignKey(nameof(Level))] public long LevelId { get; set; }

    public Level Level { get; set; } = null!;

    [ForeignKey(nameof(User))] public long UserId { get; set; }

    public User User { get; set; } = null!;

    public bool IsCompleted { get; set; }
}
```

Б.68 Код файла LevelTry.cs

```
using System.ComponentModel.DataAnnotations.Schema;
using StemLearning.WebApi.Persistence.Entities.Base;

namespace StemLearning.WebApi.Persistence.Entities;

public class LevelTry : Entity
{
    [ForeignKey(nameof(Level))] public long LevelId { get; set; }

    public Level Level { get; set; } = null!;

    [ForeignKey(nameof(User))] public long UserId { get; set; }

    public User User { get; set; } = null!;

    public int CompletedQuestions { get; set; }

    public DateTime TriedAt { get; set; }
}
```

Б.69 Код файла Stage.cs

```
using System.ComponentModel.DataAnnotations.Schema;
using StemLearning.WebApi.Persistence.Entities.Base;

namespace StemLearning.WebApi.Persistence.Entities;

public class Stage : Entity
{
    [ForeignKey(nameof(Subject))] public long SubjectId { get; set; }

    public Subject Subject { get; set; } = null!;

    public string Name { get; set; } = null!;

    public bool IsAvailable { get; set; }
}
```

Б.70 Код файла StageProgress.cs

```
using System.ComponentModel.DataAnnotations.Schema;
using StemLearning.WebApi.Persistence.Entities.Base;

namespace StemLearning.WebApi.Persistence.Entities;

public class StageProgress : Entity
{
    [ForeignKey(nameof(Stage))] public long StageId { get; set; }

    public Stage Stage { get; set; } = null!;

    [ForeignKey(nameof(User))] public long UserId { get; set; }

    public User User { get; set; } = null!;

    public float Progress { get; set; }
}
```

Б.71 Код файла Subject.cs

```
using StemLearning.WebApi.Persistence.Entities.Base;

namespace StemLearning.WebApi.Persistence.Entities;

public class Subject : Entity
{
    public string Name { get; set; } = null!;

    public bool IsAvailable { get; set; }
}
```

Б.72 Код файла SubjectProgress.cs

```
using System.ComponentModel.DataAnnotations.Schema;
using StemLearning.WebApi.Persistence.Entities.Base;

namespace StemLearning.WebApi.Persistence.Entities;

public class SubjectProgress : Entity
{
    [ForeignKey(nameof(Subject))] public long SubjectId { get; set; }

    public Subject Subject { get; set; } = null!;

    [ForeignKey(nameof(User))] public long UserId { get; set; }

    public User User { get; set; } = null!;

    public float Progress { get; set; }
}
```

Б.73 Код файла User.cs

```
using StemLearning.WebApi.Persistence.Entities.Base;

namespace StemLearning.WebApi.Persistence.Entities;

public class User : Entity
```

```

{
    public string Email { get; set; } = null!;

    public string Name { get; set; } = null!;

    public string PasswordHash { get; set; } = null!;

    public string PasswordSalt { get; set; } = null!;
}

```

Б.74 Код файла StemLearningDbContext.cs

```

using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Internal;
using StemLearning.WebApi.Persistence.Configuration;
using StemLearning.WebApi.Persistence.Entities;
using StemLearning.WebApi.Persistence.Entities.Base;

namespace StemLearning.WebApi.Persistence;

public class StemLearningDbContext : DbContext
{
    private readonly ISystemClock _clock;
    private readonly IEntityConfigurator _entityConfigurator;

    public StemLearningDbContext(DbContextOptions options,
        IEntityConfigurator entityConfigurator,
        ISystemClock clock)
        : base(options)
    {
        _entityConfigurator = entityConfigurator;
        _clock = clock;
    }

    public DbSet<User> Users { get; set; } = null!;

    public DbSet<Subject> Subjects { get; set; } = null!;

    public DbSet<SubjectProgress> SubjectsProgresses { get; set; } = null!;

    public DbSet<Stage> Stages { get; set; } = null!;

    public DbSet<StageProgress> StagesProgresses { get; set; } = null!;

    public DbSet<Level> Levels { get; set; } = null!;

    public DbSet<LevelProgress> LevelsProgresses { get; set; } = null!;

    public DbSet<LevelTry> LevelsTries { get; set; } = null!;

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        base.OnModelCreating(modelBuilder);
        _entityConfigurator.Configure(modelBuilder);
    }

    public override int SaveChanges(bool acceptAllChangesOnSuccess)
    {
        throw new NotSupportedException();
    }

    public override Task<int> SaveChangesAsync(bool
acceptAllChangesOnSuccess,

```

```

        CancellationToken cancellationToken = default)
    {
        OnBeforeSaving();
        return base.SaveChangesAsync(acceptAllChangesOnSuccess,
            cancellationToken);
    }

    private void OnBeforeSaving()
    {
        var entries = ChangeTracker.Entries();

        foreach (var entry in entries)
        {
            if (entry.Entity is IAuditableEntity entity)
            {
                var now = _clock.UtcNow.DateTime;
                switch (entry.State)
                {
                    case EntityState.Modified:
                        entity.LastUpdatedAt = now;
                        break;
                    case EntityState.Added:
                        entity.CreatedAt = now;
                        entity.LastUpdatedAt = now;
                        break;
                }
            }
        }
    }
}

```

Б.75 Код файла LocaleFactory.cs

```

using StemLearning.WebApi.Utilities.Enumerations;
using StemLearning.WebApi.Utilities.Helpers;

namespace StemLearning.WebApi.Utilities.Factories;

public class LocaleFactory
{
    public Locale Create(string localeString)
    {
        var locale = localeString.ToLower();
        var enLocale = EnumHelper.ToString(Locale.En);
        var uaLocale = EnumHelper.ToString(Locale.Ua);

        if (locale == enLocale)
        {
            return Locale.En;
        }
        if (locale == uaLocale)
        {
            return Locale.Ua;
        }

        throw new Exception("Invalid locale specified");
    }
}

```

Б.76 Код файла UserFactory.cs

```

using StemLearning.WebApi.Models.Dto;
using StemLearning.WebApi.Persistence.Entities;

```

```

using StemLearning.WebApi.Utilities.Helpers;

namespace StemLearning.WebApi.Utilities.Factories;

public class UserFactory
{
    public User Create(RegisterDto registerDto)
    {
        var user = new User
        {
            Email = registerDto.Email,
            Name = registerDto.Name
        };

        PasswordHelper.CreatePasswordHash(registerDto.Password, out var
passwordHash, out var passwordSalt);

        user.PasswordHash = passwordHash;
        user.PasswordSalt = passwordSalt;

        return user;
    }
}

```

Б.77 Код файла Mda0.cs

```

using StemLearning.WebApi.Utilities.Enumerations;
using StemLearning.WebApi.Utilities.Helpers;
using StemLearning.WebApi.Utilities.Questions.Director;
using StemLearning.WebApi.Utilities.Questions.GeneratorsForQuestions;
using StemLearning.WebApi.Utilities.Questions.Templates;

namespace
StemLearning.WebApi.Utilities.Questions.Chemistry.MetricAndDimensionalAnalysi
s;

public class Mda0QuestionsDirector : QuestionsDirector
{
    public Mda0QuestionsDirector(Locale locale) : base(locale) { }

    protected override List<QuestionGenerator>
GetAllPossibleQuestionGenerators()
    {
        var questionTypes = new List<QuestionType>()
        {
            QuestionType.GetUnit, QuestionType.GetMeasure
        };

        var questionGenerators = new List<QuestionGenerator>();

        foreach (var type in questionTypes)
        {
            if (type == QuestionType.GetMeasure)
            {
                var qgs = GetMeasureQuestionsGenerators();
                questionGenerators.AddRange(qgs);
            }
            else if (type == QuestionType.GetUnit)
            {
                var qgs = GetUnitQuestionsGenerators();
                questionGenerators.AddRange(qgs);
            }
        }
    }
}

```

```

    }
    }
    return questionGenerators;
}

private List<QuestionGenerator> GetMeasureQuestionsGenerators()
{
    var generators = new List<QuestionGenerator>();
    var units = EnumHelper.DecomposeToList<Unit>();

    foreach (var unit in units)
    {
        var questionGenerator = new GetMeasureQuestionGenerator(Locale)
            .WithUnit(unit);

        var unitArg = Locale == Locale.Ua
            ? EnumTranslationsService.GetTranslation(unit)
            : EnumTranslationsService.GetPluralTranslation(unit);

        var template =
            QuestionTemplateService.GetTemplate(QuestionType.GetMeasure);

        questionGenerator
            .FromTemplate(template)
            .TemplateArg(unitArg);

        generators.Add(questionGenerator);
    }

    return generators;
}

private List<QuestionGenerator> GetUnitQuestionsGenerators()
{
    var generators = new List<QuestionGenerator>();

    var measures = EnumHelper.DecomposeToList<Measure>();

    foreach (var measure in measures)
    {
        var questionGenerator = new GetUnitQuestionGenerator(Locale)
            .WithMeasure(measure);

        var measureArg = EnumTranslationsService.GetTranslation(measure);

        var template =
            QuestionTemplateService.GetTemplate(QuestionType.GetUnit);

        questionGenerator
            .FromTemplate(template)
            .TemplateArg(measureArg);

        generators.Add(questionGenerator);
    }

    return generators;
}
}

```

Б.78 Код файла Mda1.cs

```

using StemLearning.WebApi.Models;
using StemLearning.WebApi.Utilities.Enumerations;

```

```

using StemLearning.WebApi.Utilities.Helpers;
using StemLearning.WebApi.Utilities.Questions.Director;
using StemLearning.WebApi.Utilities.Questions.GeneratorsForQuestions;
using StemLearning.WebApi.Utilities.Questions.Templates;

namespace
StemLearning.WebApi.Utilities.Questions.Chemistry.MetricAndDimensionalAnalysis;

public class MdalQuestionsDirector : QuestionsDirector
{
    public MdalQuestionsDirector(Locale locale) : base(locale)
    {
    }

    protected override List<QuestionGenerator>
    GetAllPossibleQuestionGenerators()
    {
        var questionTypes = new List<QuestionType>()
        {
            QuestionType.HowMany, QuestionType.Compare
        };

        var questionGenerators = new List<QuestionGenerator>();

        foreach (var type in questionTypes)
        {
            if (type == QuestionType.HowMany)
            {
                var qgs = GetHowManyQuestionsGenerators();
                questionGenerators.AddRange(qgs);
            }
            else if (type == QuestionType.Compare)
            {
                var qgs = GetCompareQuestionsGenerators();
                questionGenerators.AddRange(qgs);
            }
        }
        return questionGenerators;
    }

    private List<QuestionGenerator> GetHowManyQuestionsGenerators()
    {
        var generators = new List<QuestionGenerator>();

        var prefixes = EnumHelper.DecomposeToList<Prefix>();
        foreach (var prefix in prefixes)
        {
            if (prefix == Prefix.None)
            {
                continue;
            }

            var questionGenerator = new HowManyQuestionGenerator(Locale)
                .WithPrefix1(Prefix.None)
                .WithPrefix2(prefix)
                .WithValue(1)
                .WithPower(Power.First)
                .WithUnit(Unit.Meter);

            var template =
                QuestionTemplateService.GetTemplate(QuestionType.HowMany);

```

```

        questionGenerator
            .FromTemplate(template)
            .TemplateArg("")

        .TemplateArg(EnumTranslationsService.GetPluralTranslation(Unit.Meter))

        .TemplateArg(EnumTranslationsService.GetShortTranslation(Unit.Meter))
            .TemplateArg("1 ")

        .TemplateArg(EnumTranslationsService.GetShortTranslation(prefix));

        generators.Add(questionGenerator);
    }

    return generators;
}

private List<QuestionGenerator> GetCompareQuestionsGenerators()
{
    var generators = new List<QuestionGenerator>();
    var prefixes = EnumHelper.DecomposeToList<Prefix>();

    foreach (var prefix in prefixes)
    {
        if (prefix == Prefix.None)
        {
            continue;
        }

        var questionGenerator1 = new CompareQuestionGenerator(Locale)
            .WithMetricNumber1(new MetricNumber()
            {
                Value = 1,
                Prefix = prefix
            })
            .WithMetricNumber2(new MetricNumber()
            {
                Value = 1,
                Prefix = Prefix.None
            })
            .WithUnit(Unit.Meter)
            .WithPower(Power.First);

        var questionGenerator2 = new CompareQuestionGenerator(Locale)
            .WithSmallerNeedsToBeFound()
            .WithMetricNumber1(new MetricNumber()
            {
                Value = 1,
                Prefix = prefix
            })
            .WithMetricNumber2(new MetricNumber()
            {
                Value = 1,
                Prefix = Prefix.None
            })
            .WithUnit(Unit.Meter)
            .WithPower(Power.First);

        var prefix1Arg = EnumTranslationsService.GetTranslation(prefix);
        var prefix2Arg =
EnumTranslationsService.GetTranslation(Prefix.None);
        var unitArg = EnumTranslationsService.GetTranslation(Unit.Meter);
        var largerText =
EnumTranslationsService.GetTranslation(Comparison.Bigger);

```

```

        var smallerText =
EnumTranslationsService.GetTranslation(Comparison.Less);

        var template =
QuestionTemplateService.GetTemplate(QuestionType.Compare);

        var answerVariant1 = $"1 {prefix1Arg}{unitArg}";
        var answerVariant2 = $"1 {prefix2Arg}{unitArg}";

        questionGenerator1
            .WithAnswerVariant(answerVariant1)
            .WithAnswerVariant(answerVariant2)
            .FromTemplate(template)
            .TemplateArg(largerText)
            .TemplateArg("1")
            .TemplateArg("1")
            .TemplateArg(prefix1Arg)
            .TemplateArg(unitArg)
            .TemplateArg(prefix2Arg)
            .TemplateArg(unitArg);

        questionGenerator2
            .WithAnswerVariant(answerVariant1)
            .WithAnswerVariant(answerVariant2)
            .FromTemplate(template)
            .TemplateArg(smallerText)
            .TemplateArg("1")
            .TemplateArg("1")
            .TemplateArg(prefix1Arg)
            .TemplateArg(unitArg)
            .TemplateArg(prefix2Arg)
            .TemplateArg(unitArg);

        generators.Add(questionGenerator1);
        generators.Add(questionGenerator2);
    }

    return generators;
}
}

```

Б.79 Код файла Mda2.cs

```

using StemLearning.WebApi.Utilities.Enumerations;
using StemLearning.WebApi.Utilities.Helpers;
using StemLearning.WebApi.Utilities.Questions.Director;
using StemLearning.WebApi.Utilities.Questions.GeneratorsForQuestions;
using StemLearning.WebApi.Utilities.Questions.Templates;

namespace
StemLearning.WebApi.Utilities.Questions.Chemistry.MetricAndDimensionalAnalysi
s;

public class Mda2QuestionsDirector : QuestionsDirector
{
    public Mda2QuestionsDirector(Locale locale) : base(locale)
    {
    }

    protected override List<QuestionGenerator>
GetAllPossibleQuestionGenerators()
    {
        return GetHowManyQuestionsGenerators();
    }
}

```

```

    }
    private List<QuestionGenerator> GetHowManyQuestionsGenerators()
    {
        var generators = new List<QuestionGenerator>();

        var units = EnumHelper.DecomposeToList<Unit>();
        var prefixes = EnumHelper.DecomposeToList<Prefix>();

        foreach (var unit in units)
        {
            if (unit == Unit.Mole) continue;
            foreach (var prefix in prefixes)
            {
                if (prefix == Prefix.None)
                {
                    continue;
                }

                var questionGenerator = new HowManyQuestionGenerator(Locale)
                    .WithPrefix1(Prefix.None)
                    .WithPrefix2(prefix)
                    .WithValue(1)
                    .WithPower(Power.First)
                    .WithUnit(unit);

                var template =
                    QuestionTemplateService.GetTemplate(QuestionType.HowMany);

                questionGenerator
                    .FromTemplate(template)
                    .TemplateArg("")

                    .TemplateArg(EnumTranslationsService.GetPluralTranslation(unit))

                    .TemplateArg(EnumTranslationsService.GetShortTranslation(unit))
                    .TemplateArg("1 ")

                    .TemplateArg(EnumTranslationsService.GetShortTranslation(prefix));

                generators.Add(questionGenerator);
            }
        }

        return generators;
    }
}

```

Б.80 Код файла Mda2A.cs

```

using StemLearning.WebApi.Models;
using StemLearning.WebApi.Utilities.Enumerations;
using StemLearning.WebApi.Utilities.Helpers;
using StemLearning.WebApi.Utilities.Questions.Director;
using StemLearning.WebApi.Utilities.Questions.GeneratorsForQuestions;
using StemLearning.WebApi.Utilities.Questions.Templates;

namespace
StemLearning.WebApi.Utilities.Questions.Chemistry.MetricAndDimensionalAnalysis;

public class Mda2AQuestionsDirector : QuestionsDirector
{

```

```

public Mda2AQuestionsDirector(Locale locale) : base(locale)
{
}

protected override List<QuestionGenerator>
GetAllPossibleQuestionGenerators()
{
    var questionTypes = new List<QuestionType>()
    {
        QuestionType.HowMany, QuestionType.Compare
    };

    var questionGenerators = new List<QuestionGenerator>();

    foreach (var type in questionTypes)
    {
        if (type == QuestionType.HowMany)
        {
            var qgs = GetHowManyQuestionsGenerators();
            questionGenerators.AddRange(qgs);
        }
        else if (type == QuestionType.Compare)
        {
            var qgs = GetCompareQuestionsGenerators();
            questionGenerators.AddRange(qgs);
        }
    }
    return questionGenerators;
}

private List<QuestionGenerator> GetHowManyQuestionsGenerators()
{
    var generators = new List<QuestionGenerator>();

    var prefixesCombinations =
ListHelper.GetCombinations(EnumHelper.DecomposeToList<Prefix>());
    foreach (var prefixesCombination in prefixesCombinations)
    {
        var prefix1 = prefixesCombination.Item1;
        var prefix2 = prefixesCombination.Item2;

        if (prefix1 == Prefix.None || prefix2 == Prefix.None)
        {
            continue;
        }

        if ((prefix1 == Prefix.Nano && prefix2 == Prefix.Giga)
            || (prefix2 == Prefix.Nano && prefix1 == Prefix.Giga))
        {
            continue;
        }

        var questionGenerator = new HowManyQuestionGenerator(Locale)
            .WithPrefix1(prefix1)
            .WithPrefix2(prefix2)
            .WithValue(1)
            .WithPower(Power.First)
            .WithUnit(Unit.Meter);

        var template =
QuestionTemplateService.GetTemplate(QuestionType.HowMany);

        questionGenerator

```

```

        .FromTemplate(template)
        .TemplateArg(EnumTranslationsService.GetTranslation(prefix1))

    .TemplateArg(EnumTranslationsService.GetPluralTranslation(Unit.Meter))

    .TemplateArg(EnumTranslationsService.GetShortTranslation(Unit.Meter))
        .TemplateArg("1 ")

    .TemplateArg(EnumTranslationsService.GetShortTranslation(prefix2));

    generators.Add(questionGenerator);
}

return generators;
}

private List<QuestionGenerator> GetCompareQuestionsGenerators()
{
    var generators = new List<QuestionGenerator>();
    var prefixesCombinations =
ListHelper.GetCombinations(EnumHelper.DecomposeToList<Prefix>());

    foreach (var prefixesCombination in prefixesCombinations)
    {
        var questionGenerator1 = new CompareQuestionGenerator(Locale)
            .WithMetricNumber1(new MetricNumber()
            {
                Value = 1,
                Prefix = prefixesCombination.Item1
            })
            .WithMetricNumber2(new MetricNumber()
            {
                Value = 1,
                Prefix = prefixesCombination.Item2
            })
            .WithUnit(Unit.Meter)
            .WithPower(Power.First);

        var questionGenerator2 = new CompareQuestionGenerator(Locale)
            .WithSmallerNeedsToBeFound()
            .WithMetricNumber1(new MetricNumber()
            {
                Value = 1,
                Prefix = prefixesCombination.Item1
            })
            .WithMetricNumber2(new MetricNumber()
            {
                Value = 1,
                Prefix = prefixesCombination.Item2
            })
            .WithUnit(Unit.Meter)
            .WithPower(Power.First);

        var prefix1Arg =
EnumTranslationsService.GetTranslation(prefixesCombination.Item1);
        var prefix2Arg =
EnumTranslationsService.GetTranslation(prefixesCombination.Item2);
        var unitArg = EnumTranslationsService.GetTranslation(Unit.Meter);
        var largerText =
EnumTranslationsService.GetTranslation(Comparison.Bigger);
        var smallerText =
EnumTranslationsService.GetTranslation(Comparison.Less);

        var template =

```

```

QuestionTemplateService.GetTemplate(QuestionType.Compare);

var answerVariant1 = $"1 {prefix1Arg}{unitArg}";
var answerVariant2 = $"1 {prefix2Arg}{unitArg}";

questionGenerator1
    .WithAnswerVariant(answerVariant1)
    .WithAnswerVariant(answerVariant2)
    .FromTemplate(template)
    .TemplateArg(largerText)
    .TemplateArg("1")
    .TemplateArg("1")
    .TemplateArg(prefix1Arg)
    .TemplateArg(unitArg)
    .TemplateArg(prefix2Arg)
    .TemplateArg(unitArg);

questionGenerator2
    .WithAnswerVariant(answerVariant1)
    .WithAnswerVariant(answerVariant2)
    .FromTemplate(template)
    .TemplateArg(smallerText)
    .TemplateArg("1")
    .TemplateArg("1")
    .TemplateArg(prefix1Arg)
    .TemplateArg(unitArg)
    .TemplateArg(prefix2Arg)
    .TemplateArg(unitArg);

generators.Add(questionGenerator1);
generators.Add(questionGenerator2);
}

return generators;
}
}

```

Б.81 Код файлу Mda2B.cs

```

using StemLearning.WebApi.Models;
using StemLearning.WebApi.Utilities.Enumerations;
using StemLearning.WebApi.Utilities.Helpers;
using StemLearning.WebApi.Utilities.Questions.Director;
using StemLearning.WebApi.Utilities.Questions.GeneratorsForQuestions;
using StemLearning.WebApi.Utilities.Questions.Templates;

namespace
StemLearning.WebApi.Utilities.Questions.Chemistry.MetricAndDimensionalAnalysis;

public class Mda2BQuestionsDirector : QuestionsDirector
{
    private List<double> Values { get; } = new ()
    {
        4500, 5500, 1230, 0.053, 2300
    };

    public Mda2BQuestionsDirector(Locale locale) : base(locale)
    {
    }

    protected override List<QuestionGenerator>
    GetAllPossibleQuestionGenerators()

```

```

{
    var questionTypes = new List<QuestionType>()
    {
        QuestionType.HowMany, QuestionType.Compare
    };

    var questionGenerators = new List<QuestionGenerator>();

    foreach (var type in questionTypes)
    {
        if (type == QuestionType.HowMany)
        {
            var qgs = GetHowManyQuestionsGenerators();
            questionGenerators.AddRange(qgs);
        }
        else if (type == QuestionType.Compare)
        {
            var qgs = GetCompareQuestionsGenerators();
            questionGenerators.AddRange(qgs);
        }
    }
    return questionGenerators;
}

private List<QuestionGenerator> GetHowManyQuestionsGenerators()
{
    var generators = new List<QuestionGenerator>();

    var prefixesCombinations =
ListHelper.GetCombinations(EnumHelper.DecomposeToList<Prefix>());
    foreach (var prefixesCombination in prefixesCombinations)
    {
        var prefix1 = prefixesCombination.Item1;
        var prefix2 = prefixesCombination.Item2;

        if (prefix1 == Prefix.None || prefix2 == Prefix.None)
        {
            continue;
        }

        if ((prefix1 == Prefix.Nano && prefix2 == Prefix.Giga)
            || (prefix2 == Prefix.Nano && prefix1 == Prefix.Giga))
        {
            continue;
        }

        foreach (var value in Values)
        {
            var questionGenerator = new HowManyQuestionGenerator(Locale)
                .WithPrefix1(prefix1)
                .WithPrefix2(prefix2)
                .WithValue(value)
                .WithPower(Power.First)
                .WithUnit(Unit.Meter);

            var template =
QuestionTemplateService.GetTemplate(QuestionType.HowMany);

            questionGenerator
                .FromTemplate(template)
                .TemplateArg(EnumTranslationsService.GetTranslation(prefix1))

```

```

.TemplateArg(EnumTranslationsService.GetPluralTranslation(Unit.Meter))
.TemplateArg(EnumTranslationsService.GetShortTranslation(Unit.Meter))
.TemplateArg(StringHelper.ToScientificNotation(value.ToString()) + " ")
.TemplateArg(EnumTranslationsService.GetShortTranslation(prefix2));

        generators.Add(questionGenerator);
    }
}

return generators;
}

private List<QuestionGenerator> GetCompareQuestionsGenerators()
{
    var generators = new List<QuestionGenerator>();
    var prefixesCombinations =
ListHelper.GetCombinations(EnumHelper.DecomposeToList<Prefix>());

    foreach (var prefixesCombination in prefixesCombinations)
    {
        foreach (var value in Values)
        {
            var prefix1 = prefixesCombination.Item1;
            var prefix2 = prefixesCombination.Item2;

            if ((int) prefix1 < (int) prefix2 && value > 1)
            {
                (prefix1, prefix2) = (prefix2, prefix1);
            }

            var questionGenerator1 = new CompareQuestionGenerator(Locale)
                .WithMetricNumber1(new MetricNumber())
            {
                Value = 1,
                Prefix = prefix1
            }
            .WithMetricNumber2(new MetricNumber())
            {
                Value = value,
                Prefix = prefix2
            }
            .WithUnit(Unit.Meter)
            .WithPower(Power.First);

            var questionGenerator2 = new CompareQuestionGenerator(Locale)
                .WithSmallerNeedsToBeFound()
                .WithMetricNumber1(new MetricNumber())
            {
                Value = 1,
                Prefix = prefix1
            }
            .WithMetricNumber2(new MetricNumber())
            {
                Value = value,
                Prefix = prefix2
            }
            .WithUnit(Unit.Meter)
            .WithPower(Power.First);

            var prefix1Arg =
EnumTranslationsService.GetTranslation(prefix1);

```

```

        var prefix2Arg =
EnumTranslationsService.GetTranslation(prefix2);
        var unit1Arg =
EnumTranslationsService.GetTranslation(Unit.Meter);
        var unit2Arg =
EnumTranslationsService.GetPluralTranslation(Unit.Meter);
        var largerText =
EnumTranslationsService.GetTranslation(Comparison.Bigger);
        var smallerText =
EnumTranslationsService.GetTranslation(Comparison.Less);
        var value2Text =
StringHelper.ToScientificNotation(value.ToString());

        var template =
QuestionTemplateService.GetTemplate(QuestionType.Compare);

        var answerVariant1 = $"1 {prefix1Arg}{unit1Arg}";
        var answerVariant2 = $"{value2Text} {prefix2Arg}{unit2Arg}";

        questionGenerator1
            .WithAnswerVariant(answerVariant1)
            .WithAnswerVariant(answerVariant2)
            .FromTemplate(template)
            .TemplateArg(largerText)
            .TemplateArg("1")
            .TemplateArg(value2Text)
            .TemplateArg(prefix1Arg)
            .TemplateArg(unit1Arg)
            .TemplateArg(prefix2Arg)
            .TemplateArg(unit2Arg);

        questionGenerator2
            .WithAnswerVariant(answerVariant1)
            .WithAnswerVariant(answerVariant2)
            .FromTemplate(template)
            .TemplateArg(smallerText)
            .TemplateArg("1")
            .TemplateArg(value2Text)
            .TemplateArg(prefix1Arg)
            .TemplateArg(unit1Arg)
            .TemplateArg(prefix2Arg)
            .TemplateArg(unit2Arg);

        generators.Add(questionGenerator1);
        generators.Add(questionGenerator2);

    }

    }

    return generators;
}

```

Б.82 Код файлу Mda3.cs

```

using StemLearning.WebApi.Models;
using StemLearning.WebApi.Utilities.Enumerations;
using StemLearning.WebApi.Utilities.Helpers;
using StemLearning.WebApi.Utilities.Questions.Director;
using StemLearning.WebApi.Utilities.Questions.GeneratorsForQuestions;
using StemLearning.WebApi.Utilities.Questions.Templates;

namespace
StemLearning.WebApi.Utilities.Questions.Chemistry.MetricAndDimensionalAnalysis;

public class Mda3QuestionsDirector : QuestionsDirector
{
    private List<(double, double)> ValuesSet { get; } = new ()
    {
        (2.5, 1300),
        (0.0132, 13000),
        (0.15, 23000)
    };

    public Mda3QuestionsDirector(Locale locale) : base(locale)
    {
    }

    protected override List<QuestionGenerator>
    GetAllPossibleQuestionGenerators()
    {
        return GetCompareQuestionsGenerators();
    }

    private List<QuestionGenerator> GetCompareQuestionsGenerators()
    {
        var generators = new List<QuestionGenerator>();
        var prefixesCombinations =
ListHelper.GetCombinations(EnumHelper.DecomposeToList<Prefix>());

        foreach (var prefixesCombination in prefixesCombinations)
        {
            foreach (var values in ValuesSet)
            {
                var prefix1 = prefixesCombination.Item1;
                var prefix2 = prefixesCombination.Item2;

                if ((int) prefix1 < (int) prefix2)
                {
                    (prefix1, prefix2) = (prefix2, prefix1);
                }

                var questionGenerator1 = new CompareQuestionGenerator(Locale)
                    .WithMetricNumber1(new MetricNumber()
                    {
                        Value = values.Item1,
                        Prefix = prefix1
                    })
                    .WithMetricNumber2(new MetricNumber()
                    {
                        Value = values.Item2,
                        Prefix = prefix2
                    })
                    .WithUnit(Unit.Meter)
                    .WithPower(Power.First);

                var questionGenerator2 = new CompareQuestionGenerator(Locale)
                    .WithSmallerNeedsToBeFound()

```

```

        .WithMetricNumber1(new MetricNumber()
        {
            Value = values.Item1,
            Prefix = prefix1
        })
        .WithMetricNumber2(new MetricNumber()
        {
            Value = values.Item2,
            Prefix = prefix2
        })
        .WithUnit(Unit.Meter)
        .WithPower(Power.First);

        var prefix1Arg =
EnumTranslationsService.GetTranslation(prefix1);
        var prefix2Arg =
EnumTranslationsService.GetTranslation(prefix2);
        var unit1Arg =
EnumTranslationsService.GetPluralTranslation(Unit.Meter);
        var unit2Arg =
EnumTranslationsService.GetPluralTranslation(Unit.Meter);
        var largerText =
EnumTranslationsService.GetTranslation(Comparison.Bigger);
        var smallerText =
EnumTranslationsService.GetTranslation(Comparison.Less);
        var value1Text =
StringHelper.ToScientificNotation(values.Item1.ToString());
        var value2Text =
StringHelper.ToScientificNotation(values.Item2.ToString());

        var template =
QuestionTemplateService.GetTemplate(QuestionType.Compare);

        var answerVariant1 = $"{value1Text} {prefix1Arg}{unit1Arg}";
        var answerVariant2 = $"{value2Text} {prefix2Arg}{unit2Arg}";

        questionGenerator1
            .WithAnswerVariant(answerVariant1)
            .WithAnswerVariant(answerVariant2)
            .FromTemplate(template)
            .TemplateArg(largerText)
            .TemplateArg(value1Text)
            .TemplateArg(value2Text)
            .TemplateArg(prefix1Arg)
            .TemplateArg(unit1Arg)
            .TemplateArg(prefix2Arg)
            .TemplateArg(unit2Arg);

        questionGenerator2
            .WithAnswerVariant(answerVariant1)
            .WithAnswerVariant(answerVariant2)
            .FromTemplate(template)
            .TemplateArg(smallerText)
            .TemplateArg(value1Text)
            .TemplateArg(value2Text)
            .TemplateArg(prefix1Arg)
            .TemplateArg(unit1Arg)
            .TemplateArg(prefix2Arg)
            .TemplateArg(unit2Arg);

        generators.Add(questionGenerator1);
        generators.Add(questionGenerator2);
    }
}

```

```

        return generators;
    }
}

```

Б.83 Код файлу Mda4.cs

```

using StemLearning.WebApi.Utilities.Enumerations;
using StemLearning.WebApi.Utilities.Helpers;
using StemLearning.WebApi.Utilities.Questions.Director;
using StemLearning.WebApi.Utilities.Questions.GeneratorsForQuestions;
using StemLearning.WebApi.Utilities.Questions.Templates;

namespace
StemLearning.WebApi.Utilities.Questions.Chemistry.MetricAndDimensionalAnalysis;

public class Mda4QuestionsDirector : QuestionsDirector
{
    public Mda4QuestionsDirector(Locale locale) : base(locale)
    {
    }

    protected override List<QuestionGenerator>
    GetAllPossibleQuestionGenerators()
    {
        return GetHowManyQuestionsGenerators();
    }

    private List<QuestionGenerator> GetHowManyQuestionsGenerators()
    {
        var generators = new List<QuestionGenerator>();

        var prefixesCombinations =
ListHelper.GetCombinations(EnumHelper.DecomposeToList<Prefix>());
        var powers = EnumHelper.DecomposeToList<Power>();

        foreach (var prefixesCombination in prefixesCombinations)
        {
            foreach (var power in powers)
            {
                if (power == Power.First)
                {
                    continue;
                }

                var prefix1 = prefixesCombination.Item1;
                var prefix2 = prefixesCombination.Item2;

                if ((prefix1 == Prefix.Nano && prefix2 == Prefix.Giga)
                    || (prefix2 == Prefix.Nano && prefix1 == Prefix.Giga))
                {
                    continue;
                }

                var questionGenerator = new HowManyQuestionGenerator(Locale)
                    .WithPrefix1(prefix1)
                    .WithPrefix2(prefix2)
                    .WithValue(1)
                    .WithPower(power)
                    .WithUnit(Unit.Meter);

                var template =

```

```

QuestionTemplateService.GetTemplate(QuestionType.HowMany);

        var unitArg =
$"{EnumTranslationsService.GetShortTranslation(Unit.Meter)}^{(int) power}";

        questionGenerator
            .FromTemplate(template)

.TemplateArg(EnumTranslationsService.GetShortTranslation(prefix1))
            .TemplateArg(unitArg)
            .TemplateArg(unitArg)
            .TemplateArg("1 ")

.TemplateArg(EnumTranslationsService.GetShortTranslation(prefix2));

        generators.Add(questionGenerator);
    }

}

return generators;
}
}

```

Б.84 Код файла Mda4A.cs

```

using StemLearning.WebApi.Utilities.Enumerations;
using StemLearning.WebApi.Utilities.Helpers;
using StemLearning.WebApi.Utilities.Questions.Director;
using StemLearning.WebApi.Utilities.Questions.GeneratorsForQuestions;
using StemLearning.WebApi.Utilities.Questions.Templates;

namespace
StemLearning.WebApi.Utilities.Questions.Chemistry.MetricAndDimensionalAnalysi
s;

public class Mda4AQuestionsDirector : QuestionsDirector
{
    private List<double> Values { get; } = new()
    {
        100, 200, 54, 0.3, 0.12, 0.089
    };

    public Mda4AQuestionsDirector(Locale locale) : base(locale)
    {
    }

    protected override List<QuestionGenerator>
GetAllPossibleQuestionGenerators()
    {
        return GetHowManyQuestionsGenerators();
    }

    private List<QuestionGenerator> GetHowManyQuestionsGenerators()
    {
        var generators = new List<QuestionGenerator>();

        var prefixesCombinations =
ListHelper.GetCombinations(EnumHelper.DecomposeToList<Prefix>());
        var powers = EnumHelper.DecomposeToList<Power>();

        foreach (var prefixesCombination in prefixesCombinations)
        {

```

```

        foreach (var power in powers)
        {
            if (power == Power.First)
            {
                continue;
            }

            foreach (var value in Values)
            {
                var prefix1 = prefixesCombination.Item1;
                var prefix2 = prefixesCombination.Item2;

                if ((prefix1 == Prefix.Nano && prefix2 == Prefix.Giga)
                    || (prefix2 == Prefix.Nano && prefix1 ==
Prefix.Giga))
                {
                    continue;
                }

                var questionGenerator = new
HowManyQuestionGenerator(Locale)
                    .WithPrefix1(prefix1)
                    .WithPrefix2(prefix2)
                    .WithValue(value)
                    .WithPower(power)
                    .WithUnit(Unit.Meter);

                var template =
QuestionTemplateService.GetTemplate(QuestionType.HowMany);

                var unitArg =
$" {EnumTranslationsService.GetShortTranslation(Unit.Meter)} ^ {(int) power}";

                questionGenerator
                    .FromTemplate(template)

                    .TemplateArg(EnumTranslationsService.GetShortTranslation(prefix1))
                    .TemplateArg(unitArg)
                    .TemplateArg(unitArg)
                    .TemplateArg($" {value} ");

                .TemplateArg(EnumTranslationsService.GetShortTranslation(prefix2));

                generators.Add(questionGenerator);
            }
        }

        return generators;
    }
}

```

Б.85 Код файлу Mda4B.cs

```

using StemLearning.WebApi.Models;
using StemLearning.WebApi.Utilities.Enumerations;
using StemLearning.WebApi.Utilities.Helpers;
using StemLearning.WebApi.Utilities.Questions.Director;
using StemLearning.WebApi.Utilities.Questions.GeneratorsForQuestions;
using StemLearning.WebApi.Utilities.Questions.Templates;

namespace
StemLearning.WebApi.Utilities.Questions.Chemistry.MetricAndDimensionalAnalysi

```

```

s;

public class Mda4BQuestionsDirector : QuestionsDirector
{
    private List<double> HowManyValuesSet { get; } = new ()
    {
        220, 165, 4500, 980, 0.095
    };

    private List<(double, double)> CompareValuesSet { get; } = new ()
    {
        (2.5, 1300),
        (0.0182, 1800),
        (0.15, 19500)
    };

    public Mda4BQuestionsDirector(Locale locale) : base(locale)
    {
    }

    protected override List<QuestionGenerator>
    GetAllPossibleQuestionGenerators()
    {
        var questionTypes = new List<QuestionType>()
        {
            QuestionType.HowMany, QuestionType.Compare
        };

        var questionGenerators = new List<QuestionGenerator>();

        foreach (var type in questionTypes)
        {
            if (type == QuestionType.HowMany)
            {
                var qgs = GetHowManyQuestionsGenerators();
                questionGenerators.AddRange(qgs);
            }
            else if (type == QuestionType.Compare)
            {
                var qgs = GetCompareQuestionsGenerators();
                questionGenerators.AddRange(qgs);
            }
        }
        return questionGenerators;
    }

    private List<QuestionGenerator> GetHowManyQuestionsGenerators()
    {
        var generators = new List<QuestionGenerator>();

        var prefixesCombinations =
ListHelper.GetCombinations(EnumHelper.DecomposeToList<Prefix>());
        var powers = EnumHelper.DecomposeToList<Power>();

        foreach (var prefixesCombination in prefixesCombinations)
        {
            foreach (var power in powers)
            {
                if (power == Power.First)
                {
                    continue;
                }
            }
        }
    }
}

```

```

        foreach (var value in HowManyValuesSet)
        {
            var prefix1 = prefixesCombination.Item1;
            var prefix2 = prefixesCombination.Item2;

            if ((prefix1 == Prefix.Nano && prefix2 == Prefix.Giga)
                || (prefix2 == Prefix.Nano && prefix1 ==
Prefix.Giga))
            {
                continue;
            }

            var questionGenerator = new
HowManyQuestionGenerator(Locale)
                .WithPrefix1(prefix1)
                .WithPrefix2(prefix2)
                .WithValue(value)
                .WithPower(power)
                .WithUnit(Unit.Meter);

            var template =
QuestionTemplateService.GetTemplate(QuestionType.HowMany);

            var unitArg =
 $"{EnumTranslationsService.GetShortTranslation(Unit.Meter)}^{(int) power}";
            var valueArg =
StringHelper.ToScientificNotation(value.ToString());

            questionGenerator
                .FromTemplate(template)

.TemplateArg(EnumTranslationsService.GetShortTranslation(prefix1))
                .TemplateArg(unitArg)
                .TemplateArg(unitArg)
                .TemplateArg($"{valueArg} ");

.TemplateArg(EnumTranslationsService.GetShortTranslation(prefix2));

            generators.Add(questionGenerator);
        }
    }

    return generators;
}

private List<QuestionGenerator> GetCompareQuestionsGenerators()
{
    var generators = new List<QuestionGenerator>();
    var prefixesCombinations =
ListHelper.GetCombinations(EnumHelper.DecomposeToList<Prefix>());
    var powers = EnumHelper.DecomposeToList<Power>();

    foreach (var prefixesCombination in prefixesCombinations)
    {
        foreach (var values in CompareValuesSet)
        {
            foreach (var power in powers)
            {
                if (power == Power.First)
                {
                    continue;
                }
            }
        }
    }
}

```

```

        var prefix1 = prefixesCombination.Item1;
        var prefix2 = prefixesCombination.Item2;

        if ((int) prefix1 < (int) prefix2)
        {
            (prefix1, prefix2) = (prefix2, prefix1);
        }

        var questionGenerator1 = new
CompareQuestionGenerator(Locale)
        .WithMetricNumber1(new MetricNumber()
        {
            Value = values.Item1,
            Prefix = prefix1
        })
        .WithMetricNumber2(new MetricNumber()
        {
            Value = values.Item2,
            Prefix = prefix2
        })
        .WithUnit(Unit.Meter)
        .WithPower(power);

        var questionGenerator2 = new
CompareQuestionGenerator(Locale)
        .WithSmallerNeedsToBeFound()
        .WithMetricNumber1(new MetricNumber()
        {
            Value = values.Item1,
            Prefix = prefix1
        })
        .WithMetricNumber2(new MetricNumber()
        {
            Value = values.Item2,
            Prefix = prefix2
        })
        .WithUnit(Unit.Meter)
        .WithPower(power);

        var prefix1Arg =
EnumTranslationsService.GetShortTranslation(prefix1);
        var prefix2Arg =
EnumTranslationsService.GetShortTranslation(prefix2);
        var unit1Arg =
$"{{EnumTranslationsService.GetShortTranslation(Unit.Meter)}}^{{(int) power}}";
        var unit2Arg =
$"{{EnumTranslationsService.GetShortTranslation(Unit.Meter)}}^{{(int) power}}";
        var largerText =
EnumTranslationsService.GetTranslation(Comparison.Bigger);
        var smallerText =
EnumTranslationsService.GetTranslation(Comparison.Less);
        var value1Text =
StringHelper.ToScientificNotation(values.Item1.ToString());
        var value2Text =
StringHelper.ToScientificNotation(values.Item2.ToString());

        var template =
QuestionTemplateService.GetTemplate(QuestionType.Compare);

        var answerVariant1 = $"{{value1Text}}
{{prefix1Arg}}{{unit1Arg}}";
        var answerVariant2 = $"{{value2Text}}
{{prefix2Arg}}{{unit2Arg}}";

```

```

        questionGenerator1
            .WithAnswerVariant(answerVariant1)
            .WithAnswerVariant(answerVariant2)
            .FromTemplate(template)
            .TemplateArg(largerText)
            .TemplateArg(value1Text)
            .TemplateArg(value2Text)
            .TemplateArg(prefix1Arg)
            .TemplateArg(unit1Arg)
            .TemplateArg(prefix2Arg)
            .TemplateArg(unit2Arg);

        questionGenerator2
            .WithAnswerVariant(answerVariant1)
            .WithAnswerVariant(answerVariant2)
            .FromTemplate(template)
            .TemplateArg(smallerText)
            .TemplateArg(value1Text)
            .TemplateArg(value2Text)
            .TemplateArg(prefix1Arg)
            .TemplateArg(unit1Arg)
            .TemplateArg(prefix2Arg)
            .TemplateArg(unit2Arg);

        generators.Add(questionGenerator1);
        generators.Add(questionGenerator2);
    }
}

return generators;
}
}

```

Б.86 Код файла QuestionSetGenerator.cs

```

using StemLearning.WebApi.Models.Question;
using StemLearning.WebApi.Persistence.Entities;
using StemLearning.WebApi.Persistence.Repositories.Interfaces;
using StemLearning.WebApi.Utilities.Enumerations;
using
StemLearning.WebApi.Utilities.Questions.Chemistry.MetricAndDimensionalAnalysis;
using StemLearning.WebApi.Utilities.Questions.Director;

namespace StemLearning.WebApi.Utilities.Questions.QuestionSetGenerator;

public class QuestionsSetGenerator : IQuestionsSetGenerator
{
    private readonly ILevelsRepository _levelsRepository;

    private Level _level = null!;
    private Locale _locale;

    public QuestionsSetGenerator(ILevelsRepository levelsRepository)
    {
        _levelsRepository = levelsRepository;
        _locale = Enumerations.Locale.En;
    }

    public Task<List<Question>> Generate()
    {
        if (_level == null)
        {

```

```

        throw new Exception("Level is not specified");
    }

    QuestionsDirector questionDirector;
    switch (_level.Name)
    {
        case "0":
            questionDirector = new Mda0QuestionsDirector(_locale);
            break;
        case "1":
            questionDirector = new Mda1QuestionsDirector(_locale);
            break;
        case "2":
            questionDirector = new Mda2QuestionsDirector(_locale);
            break;
        case "2A":
            questionDirector = new Mda2AQuestionsDirector(_locale);
            break;
        case "2B":
            questionDirector = new Mda2BQuestionsDirector(_locale);
            break;
        case "3":
            questionDirector = new Mda3QuestionsDirector(_locale);
            break;
        case "4":
            questionDirector = new Mda4QuestionsDirector(_locale);
            break;
        case "4A":
            questionDirector = new Mda4AQuestionsDirector(_locale);
            break;
        case "4B":
            questionDirector = new Mda4BQuestionsDirector(_locale);
            break;
        default:
            throw new Exception("No questions could be generated for this
level");
    }

    return questionDirector.GetQuestions(_level.NumberOfQuestions);
}

public IQuestionsSetGenerator Locale(Locale locale)
{
    _locale = locale;
    return this;
}

public IQuestionsSetGenerator Level(Level level)
{
    _level = level;
    return this;
}
}

```

Б.87 Код файлу EnumAnswersTranslationService.cs

```

using StemLearning.WebApi.Utilities.Enumerations;
using StemLearning.WebApi.Utilities.Translations;
using StemLearning.WebApi.Utilities.Translations.Base;

namespace StemLearning.WebApi.Utilities.Services;

public class EnumAnswersTranslationsService

```

```

{
    private readonly Locale _locale;
    public EnumAnswersTranslationsService(Locale locale)
    {
        _locale = locale;
    }

    public List<string> GetTranslations<T>(T enumValue) where T : Enum
    {
        var stringEnum = enumValue.ToString("G");
        IEnumAnswersTranslator translator;
        switch (typeof(T))
        {
            case { } t when t == typeof(Unit):
                translator = new UnitAnswersTranslator();
                break;
            case { } t when t == typeof(Measure):
                translator = new MeasureAnswersTranslator();
                break;
            default:
                throw new Exception("Enum type translator is not provided");
        }

        if (_locale == Locale.Ua) return
translator.GetUkrAnswersTranslations(stringEnum);
        if (_locale == Locale.En) return
translator.GetEnAnswersTranslations(stringEnum);
        throw new Exception("Unknown locale");
    }
}

```

Б.88 Код файлу EnumTranslationService.cs

```

using StemLearning.WebApi.Utilities.Enumerations;
using StemLearning.WebApi.Utilities.Translations;
using StemLearning.WebApi.Utilities.Translations.Base;

namespace StemLearning.WebApi.Utilities.Services;

public class EnumTranslationsService
{
    private readonly Locale _locale;
    public EnumTranslationsService(Locale locale)
    {
        _locale = locale;
    }

    public string GetTranslation<T>(T enumValue) where T : Enum
    {
        var stringEnum = enumValue.ToString("G");
        IEnumTranslator translator = GetTranslator<T>();

        if (_locale == Locale.Ua) return
translator.GetUkrTranslation(stringEnum);
        if (_locale == Locale.En) return
translator.GetEnTranslation(stringEnum);
        throw new Exception("Unknown locale");
    }

    public string GetPluralTranslation<T>(T enumValue) where T : Enum
    {
        var stringEnum = enumValue.ToString("G");
        IEnumTranslator translator = GetTranslator<T>();
    }
}

```

```

        if (_locale == Locale.Ua) return
translator.GetPluralUkrTranslation(stringEnum);
        if (_locale == Locale.En) return
translator.GetPluralEnTranslation(stringEnum);
        throw new Exception("Unknown locale");
    }

    public string GetShortTranslation<T>(T enumValue) where T : Enum
    {
        var stringEnum = enumValue.ToString("G");
        IEnumTranslator translator = GetTranslator<T>();

        if (_locale == Locale.Ua) return
translator.GetShortUkrTranslation(stringEnum);
        if (_locale == Locale.En) return
translator.GetShortEnTranslation(stringEnum);
        throw new Exception("Unknown locale");
    }

    private IEnumTranslator GetTranslator<T>() where T : Enum
    {
        IEnumTranslator translator;
        switch (typeof(T))
        {
            case { } t when t == typeof(Unit):
                translator = new UnitTranslator();
                break;
            case { } t when t == typeof(Prefix):
                translator = new PrefixTranslator();
                break;
            case { } t when t == typeof(Measure):
                translator = new MeasureTranslator();
                break;
            case { } t when t == typeof(Comparison):
                translator = new ComparisonTranslator();
                break;
            default:
                throw new Exception("Enum type translator is not provided");
        }

        return translator;
    }
}

```