

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА РЕАЛІЗАЦІЯ ФУНКЦІОНАЛЬНИХ ВИМОГ ДО
ВЕБЗАСТОСУНКІВ ОНЛАЙН ШКІЛ ЗАДЛЯ ЗАБЕЗПЕЧЕННЯ ІНКЛЮЗИВНИХ
ПІДХОДІВ НАВЧАННЯ

RESEARCH AND IMPLEMENTATION OF THE FUNCTIONAL
REQUIREMENTS TO THE ONLINE SCHOOL WEB APPLICATIONS FOR
INCLUSIVE EDUCATION APPROACHES

Виконав: студент(ка) 2 курсу, групи КНТ-111м
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Джрагацпанян Н.В.

(прізвище та ініціали)

Керівник Табунщик Г.В.

(прізвище та ініціали)

Рецензент Шитікова О.В.

(прізвище та ініціали)

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

Інститут, факультет ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти магістр
 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.
С.О. Субботін
 “ ” 2022 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТКИ

Джрагацпанян Нелі Варданівни
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та реалізація функціональних вимог до вебзастосунків онлайн шкіл задля забезпечення інклюзивних підходів навчання.
Research and Implementation of the Functional Requirements to the Online School
Web Applications for Inclusive Education Approaches.

керівник проєкту (роботи) Табунщик Галина Володимирівна, професор,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “04” листопада 2022 року № 363

2. Строк подання студентом проєкту (роботи) 01 грудня 2022 року

3. Вихідні дані до проєкту (роботи) технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження.
2. Матеріали і методи. 3. Розробка архітектури програми. 4. Розробка програми. 5.
Експлуатація, тестування та експериментальне дослідження програми.

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

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Табунщик Г.В., професор		
Нормоконтролер	Камінська Ж.К., асистент		

7. Дата видачі завдання “01” вересня 2022 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.	2–3 тижні	Розділ 1
3	Розробка та удосконалення методів, моделей й алгоритмів вирішення задачі.	4–5 тижні	Розділ 2
4	Розробка архітектури програми.	6 тиждень	Розділ 3
5	Розробка програми.	7–8 тижні	Розділ 4
6	Тестування та експериментальне дослідження програмного забезпечення.	9 тиждень	Розділ 5
7	Оформлення пояснювальної записки та документів до неї.	10-13 тижні	Додатки
8	Нормоконтроль та рецензування.	14–15 тижні	
9	Захист роботи.	16 тиждень	

Студент(ка)

_____ Джрагацпанян Н.В.
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи)

_____ Табунщик Г.В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра: 129 с., 6 табл., 52 рис., 2 дод., 26 джерел.

ВЕБЗАСТОСУНОК, ОНЛАЙН ШКОЛА, JAVASCRIPT, REACT, TYPESCRIPT.

Об'єкт дослідження – процес організації онлайн навчання.

Предмет дослідження – програмні засоби реалізації вебзастосунку для онлайн шкіл задля забезпечення інклюзивних підходів навчання.

Мета роботи – створення вебзастосунку для онлайн шкіл задля забезпечення інклюзивних підходів навчання.

Матеріали, методи та технічні засоби: вебпрограмування, мова програмування JavaScript, фреймворк React.js, персональний комп'ютер з процесором Intel Core 5 під управлінням операційної системи Microsoft Windows 10.

Результати. Створено вебзастосунок з адміністративною панеллю для онлайн шкіл для комфортного навчання та викладання.

Висновки. Розроблено програмну систему для онлайн шкіл, які хочуть використовувати сучасні методи навчання. Система автоматизує процес навчання та викладання.

Галузь використання – навчальні заклади, онлайн школи.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 129 pages, 52 figures, 6 tables, 2 appendixes, 26 sources.

WEB APPLICATION, ONLINE SCHOOL, JAVASCRIPT, REACT, TYPESCRIPT.

Object of study: the process of organizing online studying.

Subject of study: software tools for implementing a web application for online schools for inclusive education approaches.

The purpose of the work is to create a web application for online schools for inclusive education approaches.

Materials, methods, and tools: web programming, JavaScript programming language, React.js framework, personal computer with Intel Core 5 processor running Microsoft Windows 10 operating system.

Results. A web application with an administrative panel for online schools for comfortable studying and teaching has been created.

Conclusions. A software system for online schools that want to use modern methods of studying has been developed. The system automates the process of studying and teaching.

The field of use: educational institutions, online schools.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	7
Вступ.....	8
1 Аналіз проблеми та постановка завдань дослідження	11
1.1 Аналіз предметної області	11
1.2 Огляд існуючих методів вирішення завдання	12
1.3 Технічне завдання	20
1.4 Висновки за розділом 1	22
2 Матеріали і методи	23
2.1 Вимоги до вебзастосунку.....	23
2.2 Висновки за розділом 2	34
3 Опис програми.....	35
3.1 Логічна структура програми	35
3.2 Вибір інструментарію для розробки програмного забезпечення.....	37
3.3 Висновки за розділом 3.....	49
4 Розробка програми	50
4.1 Опис структури файлів	50
4.2 Основні рішення з реалізації компонентів системи	51
4.3 Опис допоміжних бібліотек, модулів та функцій	56
4.4 Висновки за розділом 4.....	61
5 Експлуатація, тестування та експериментальне дослідження програми	62
5.1 Призначення й умови застосування програми	62
5.2 Тестування та експериментальне дослідження програми	63
5.3 Висновки за розділом 5.....	76
Висновки	77
Перелік джерел посилання	78
Додаток А Текст програми.....	81
Додаток Б Слайди презентації	118

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

- API – Application Programming Interface;
IT – Information Technology;
HTTP – HyperText Transfer Protocol;
SDLC – Software Development Lifecycle;
БД – база даних;
ПЕОМ – персональна електронна обчислювальна машина;
ПЗ – програмне забезпечення;
ПП – програмний продукт;
СУБД – Система Управління Базою Даних.

ВСТУП

Інтеграція інформаційних технологій (ІТ) в сферу освіти автоматизує та оптимізує процеси навчання та викладання. Основне завдання інформаційних технологій у сфері освіти – це формування інтерактивного середовища управління процесом освітньої діяльності та використання інформаційних ресурсів відповідно до потреб користувача. Інформаційні технології в освітньому процесі мають значний вплив на те, яким чином у людей формується сучасна інформаційна картина світу. Існує соціальна потреба підвищення якості освіти, є попит на використання сучасних комп'ютерних програм, тому використання інформаційних технологій в сфері освіти є актуальним.

Застосування інформаційних технологій в сфері освіти сприяє підвищенню ефективності навчання, індивідуалізації та диференціації, підвищенню якості навчання, посилює інтелектуальні можливості учнів в інформаційному суспільстві, сприяє організації нових форм взаємодії в процесі навчання, зміні змісту та характеру діяльності тих, хто навчає та навчається. А в сучасному світі викладачі мають вміти застосовувати нові технічні засоби в сфері освіти задля досягнення успішного функціонування освіти. Завдяки активній інтеграції ІТ в сферу освіти стало можливим онлайн навчання.

Останнім часом дослідження можливостей дистанційного навчання почало цікавити багатьох. Адже прогрес не стоїть на місці, з кожним днем у світі з'являються нові теорії та пристрої для полегшення сучасного життя людини. Причому захоплюються всі сфери – і освіта у тому числі.

Впровадження онлайн навчання значно спрощує освітній процес. Воно допомагає людям економити час на дорогу до навчального закладу, займатись онлайн можна з будь-якої точки світу. Можна стати студентом закордонного ВУЗу не покидаючи рідне місто [1].

А особливо онлайн навчання корисне в умовах нестабільності, наприклад, коли учні та вчителі вимушені сидіти вдома протягом довгого часу через епідемію, або при таких небезпечних ситуаціях, як воєнний стан.

Іншою перевагою онлайн платформ для навчання є гнучкість графіка, бо студенти можуть переглядати лекції та проходити тести саме тоді, коли їм зручно, тож їм не потрібно підлаштовуватися під розклад лекцій. Це особливо важливо для студентів та дорослих людей, щоб не відриватись від повсякденного життя, роботи та сім'ї. Люди можуть вчитися практично у будь-який час, у своєму власному темпі та у будь-якому місці. Це все – дуже великі можливості.

Звісно, не всі люди віддають перевагу дистанційному навчанню, але завдяки постійному розвитку цієї сфери, завдяки використанню нових більш зручних інструментів прихильників онлайн навчання стає більше. Тому з кожним роком традиційними методами навчання користуються все менше людей.

Необхідно розробити вебзастосунок для організації навчання в онлайн школах, учням надати можливість користуватись навчальними матеріалами в особистому кабінеті, викладачу надати можливість роботи з адміністративною панеллю, з базою даних.

Дана задача буде вирішуватися шляхом аналізу предметної області та технологій, які будуть використані для розробки застосунку. Більшість сучасних вебзастосунків розроблені за допомогою таких інструментів, як React.js/Vue.js/Angular.js. При аналізі та порівнянні цих інструментів в процесі роботи над проектом можна буде зробити висновок, який саме інструмент підходить нам для досягнення мети. Вебзастосунок має бути продуктивним та швидким, для сучасних сайтів це один із найважливіших критеріїв. Відомо, що більше половини користувачів закривають сайт, не дочекавшись його повного завантаження, якщо воно тривало більше 5 секунд. Повільні сайти формують у них негативне враження про бренд чи компанію. Тому люди до конкурентів, якщо завантаження веб-сайту зайняло надто багато часу. Для досягнення мети щодо реалізації продуктивного сайту недостатньо лише обрати підходящий інструмент та використати його, а потрібно правильно його застосувати з «best practices», написати логічний код, який в майбутньому можна буде масштабувати іншим членам команди. Наприклад, в React.js рекомендовано використовувати «silly» компоненти. Це компоненти, мета яких – просто відображення, вони не містять в

собі дій. Дії необхідно описувати в контейнерах. Таким чином, завдяки використанню такого підходу відбувається поділ відповідальності (UI краще, якщо писати компоненти таким способом; дизайнер може швидше налаштувати «silly» компоненти, не залазячи в логіку програми), краща реюзабельність (розробник може використовувати один і той же «silly» компонент із абсолютно різними джерелами стану). Також зараз ні один вебзастосунок не реалізується без покриття тестами, тому важливо при аналізі необхідних інструментів для розробки обрати середовище тестування та своєчасно покривати код тестами, щоб мінімізувати кількість багів.

Отже, завдяки правильному підбору технологій та їх раціональному застосуванню можна буде розробити сучасну систему для викладання та навчання. Передбачається, що вебзастосунок, який буде розроблятися, матиме соціальну значимість, в першу чергу для викладачів та студентів онлайн шкіл.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Аналіз предметної області

Через глобальні проблеми, які вразили світ у 2020 році онлайн навчання стало необхідним ресурсом для учнів та шкіл у всіх країнах. Коли всі через пандемію перебралися на дистанційне навчання, онлайн замінив очне навчання та врятував навчальний процес від простоїв. Школярі та студенти змогли займатися через платформи своїх навчальних закладів і завдяки цьому не пропустили заняття і не втратили цінних знань. Онлайн викладання та навчання – це абсолютно новий процес для багатьох закладів освіти, який їм довелося прийняти, але завдяки швидкому прогресу ІТ-індустрії онлайн навчання має більше переваг, ніж недоліків:

- можливість для учня самостійно вибудовувати графік навчання, і навіть тривалість занять;
- доступність – незалежно від географічного положення, стану здоров'я учень має доступ до освітнього ресурсу;
- технологічність – використання в освітньому процесі новітніх досягнень інформаційних та телекомунікаційних технологій;
- зниження витрат на навчання.

Онлайн навчання допомагає розвиватися з використанням сучасних інструментів і дітям, і дорослим. Воно усуває основний бар'єр, який заважає багатьом зайнятим продовжувати освіту, позбавляє необхідності ходити на заняття за встановленим розкладом.

На даний момент існує немало сайтів для користування навчальними матеріалами та проведення онлайн-занять, але їх значно більше, ніж окремих платформ, які містять в собі якомога більше функцій, які б дали можливість менше використовувати сторонні ресурси, які не повністю підлаштовані під потреби певного курсу окремої онлайн школи.

Наприклад, онлайн школи іноземних мов зазвичай проводять заняття в Zoom/Skype/Google Meet, матеріали для вивчення скидують учням в Google

Classroom/Google Drive/Moodle або в Telegram/Skype/Viber, які ще і є месенджерами, тому матеріали зазвичай не відсортовані та виглядає це не зовсім ергономічно. Із перелічених інструментів найзручнішими в використанні є сервіси від Google.

Отже, веб-застосунок для студентів та викладачів онлайн шкіл іноземних мов значно оптимізує процес навчання та викладання наступним чином:

- ергономічність інтерфейсу;
- реєстрація та авторизація з попереднім сортуванням учнів за групами;
- розділ «Відеоуроки», завдяки якому учні можуть повторити та закріпити пройдений матеріал або пройти його окремо, якщо не було можливості бути присутнім на занятті;

- розділ «Домашні завдання», який має містити файли з домашніми завданнями (текстові файли, а також тести, після проходження яких можна одразу подивитись всі свої помилки, легко знайти пояснення кожної відповіді та дізнатись, що саме потрібно повторити для уникнення подібних помилок);

- розділ «Матеріали», в якому будуть відсортовані файли різних форматів (підручники, файли з граматиною та інше);

- розділ «Розклад», в якому учні можуть знайти актуальні данні про заняття.

На сьогоднішній день невеликі онлайн школи або навчальні заклади, які не бажають створювати власні застосунки для викладання та навчання, зазвичай використовують Google Classroom або Moodle.

1.2 Огляд існуючих методів вирішення завдання

1.2.1 Google Classroom

Система Google Classroom розроблена Google в 2014 році, її головна мета – спростити процес поширення файлів між здобувачами освіти та педагогами. Google Classroom містить в собі різні сервіси, щоб допомогти навчальним закладам перейти до безпаперової системи: в Google Docs/Sheets/Slides викладачі

створюють матеріали, в Google Drive обмінюються матеріалами, в Gmail спілкуються, за розкладом слідкують в Google Calendar і так далі.

На рис. 1.1 зображено основну сторінку курсу від онлайн школи в Google Classroom. Видно, що інтерфейс ергономічний, навіть наймолодші користувачі можуть легко користуватись цією платформою.

Сервіси для навчання від Google дійсно дуже зручні та якісні, але якщо онлайн школа хоче урізноманітнити види завдань для учнів (наприклад, більш функціональні тести, завдання в ігровому форматі та інше), тоді сервісів від Google буде недостатньо. Тому не дивно, що онлайн школи з великим досвідом віддають перевагу власним вебзастосункам, щоб легко наповнювати його функціоналом, який необхідний саме для них.

Перевагами Google Classroom є:

- ергономічність;
- ненагромаджений сучасний дизайн;
- велика кількість сучасних інструментів для онлайн занять;
- відсутність багів, завдяки якісно написаному коду всі функції працюють так, як було заплановано;
- декілька інструментів Google в одному.

Із недоліків було виявлено те, що Google Classroom не містить в собі інструментів для створення нетипічних завдань, які часто використовуються на мовних курсах таких як ігри, тести з детальним розбором відповідей та інше.

Користувачам Google Classroom для проведення тестувань/опитувань можна використовувати сервіс Google Forms. Він доволі функціональний, чудово підходить для опитувань, але все ж для різноманітних тестувань від онлайн шкіл функцій іноді не вистачає.

Недоліки Google Classroom:

- в сервісі Google Forms не можна обмежити кількість спроб, не можна запуснути перемішування питань, щоб у кожного учня вони відображалися у випадковому порядку, самі тести більше нагадують опитування;

— недостатньо функціоналу для онлайн шкіл, які використовують нетипічні завдання для розвитку учнів.

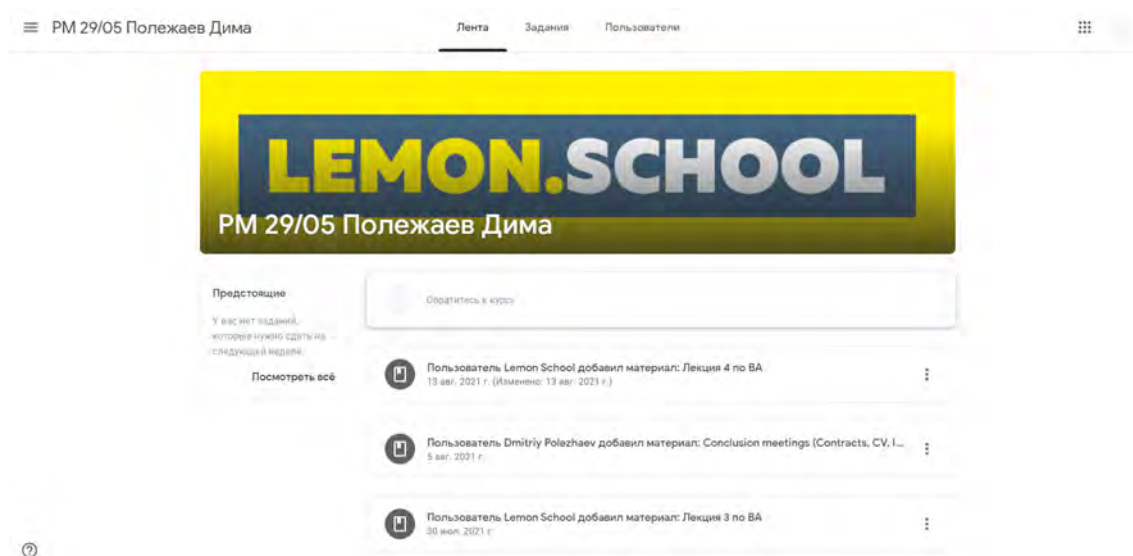


Рисунок 1.1 — Основна сторінка курсу від онлайн школи

1.2.2 Moodle

Moodle (модульне об'єктно-орієнтоване динамічне навчальне середовище) – це веб-система для організації дистанційного навчання та управління ним, яка була запущена у 2001 році.

Потенційні користувачі одразу звертають увагу на основну перевагу Moodle – його безкоштовне використання. Також через те, що Moodle працює за схемою open source (відкритий вихідний код), велика спільнота програмістів часто створює різноманітні модулі та розширення в цьому середовищі. Зараз таких корисних розширень та додатків налічується близько 1500. Наприклад:

- модулі відеоконференцій;
- аудіо- та відеочати;
- плагін для додавання питань, коментарів та зауважень навчальним матеріалам у PDF-документ;
- мотиваційні модулі;
- інтерактивні звіти щодо навчального плану;

- масове розсилання повідомлень;
- електронне портфоліо.

За рахунок безкоштовної версії та гнучкості налаштувань Moodle активно використовують у навчальних закладах, а також його використовують репетитори для створення дистанційних курсів та персонального навчання.

На рис. 1.2 зображена основна сторінка в особистому кабінеті в системі Moodle. Вже при першому відвідуванні сайту помітно, що інтерфейс неергономічний. Та швидкому пошуку необхідних блоків часто заважають непотрібні елементи.

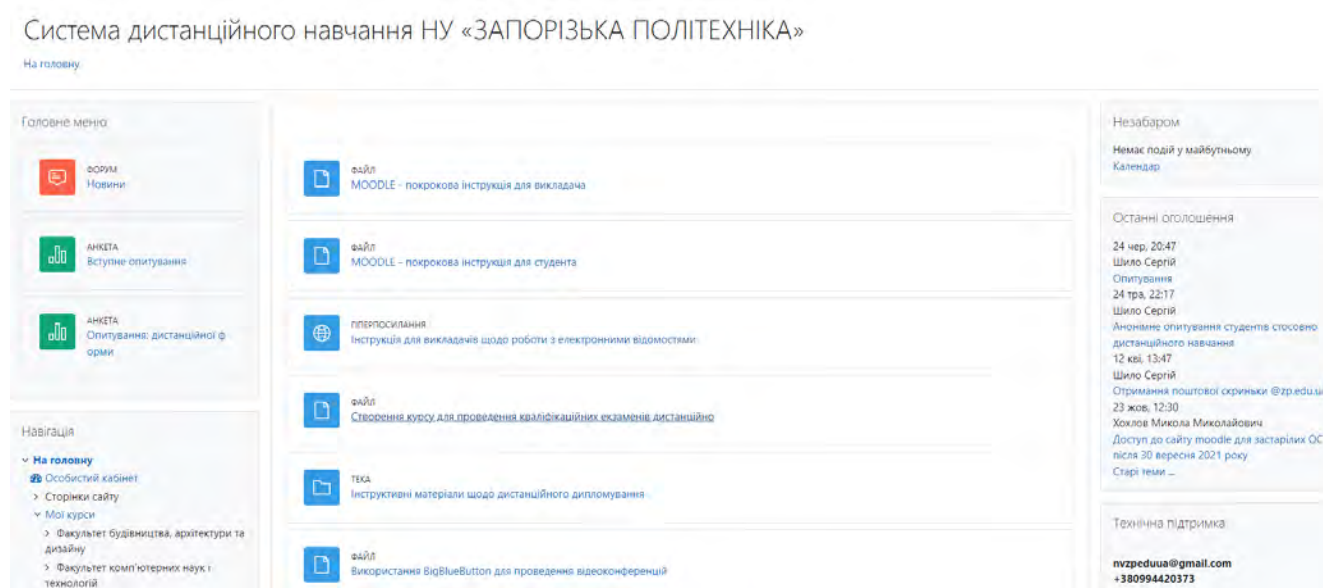


Рисунок 1.2 – Основна сторінка в особистому кабінеті в системі Moodle

Недоліки сайту:

- незручний та заплутаний інтерфейс програми, відсутність інтуїтивно зрозумілого управління;
- щоб організувати навчання та створити повноцінні курси, систему необхідно встановити. Для цього буде потрібно сервер або хостинг, доменне ім'я та інше. Якщо ви не володієте потрібними знаннями та досвідом, доведеться наймати ІТ-фахівця. У результаті налаштування сервісу стає дорогим завданням;

— Moodle – об’ємна система з великою кількістю функцій, частину з яких просто не використовують. А прибрати їх можна лише за допомогою програмістів.

1.2.3 Веб-сайт з тестами ЗНО

Так як планується розробка вебзастосунку для онлайн школи вивчення іноземних мов, доречним буде проаналізувати вебсайт <https://zno.osvita.ua/> (рис. 1.3). Цей сайт не є аналогом Google Classroom або Moodle, але подібний функціонал з тестами планується додати до вебзастосунку, що розробляється.

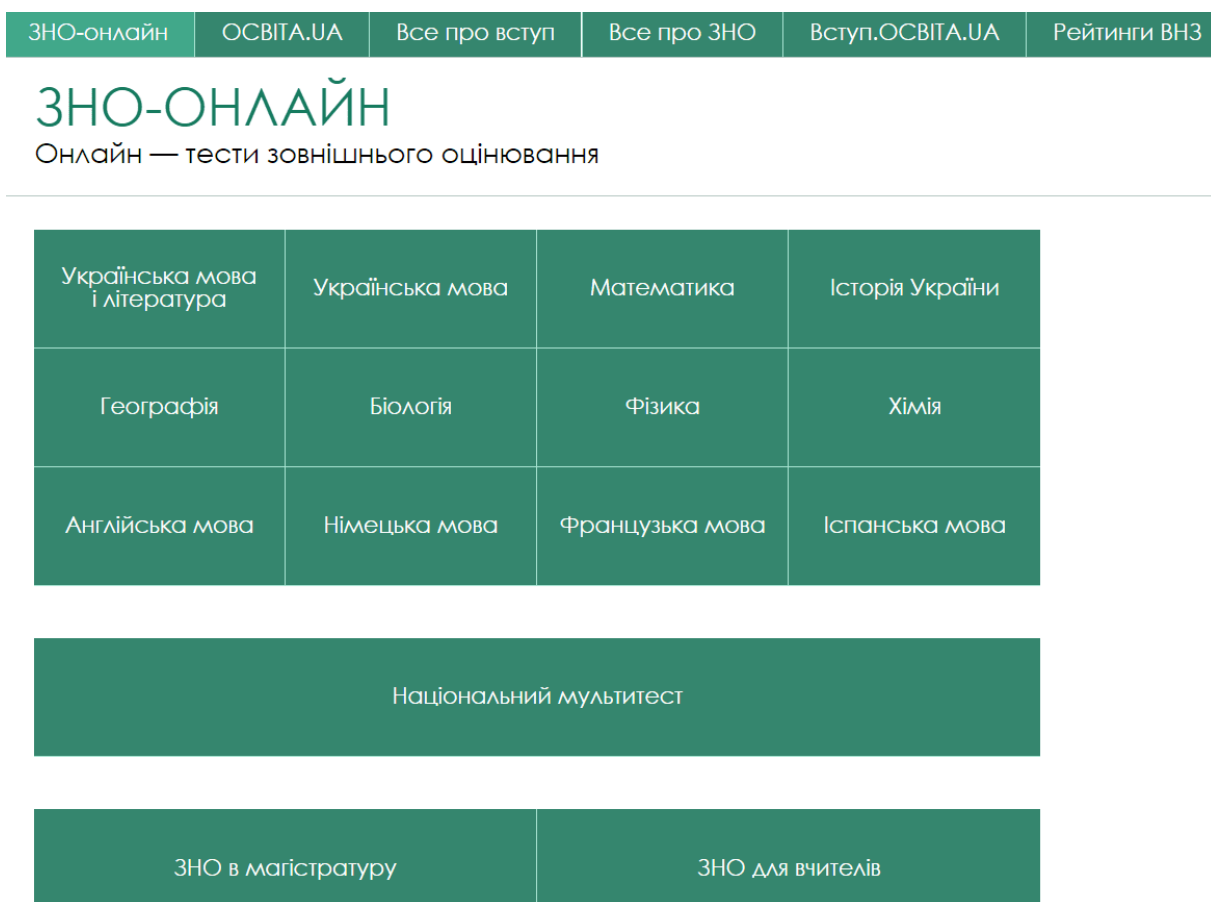


Рисунок 1.3 – Головна сторінка вебсайту

Цей сайт поширений серед українських учнів та студентів, які готуються до державних іспитів. Він містить в собі новини про освіту в Україні та іншу корисну інформацію щодо іспитів. Але найкориснішим, що є на цьому сайті, вважаються тести ЗНО та ЄВІ за всі попередні роки з усіх предметів. Та на цьому сайті можна не просто проходити тести та дізнаватись підсумковий бал, а ще і

використовувати підказки до завдань, в яких детально розписано, чому саме ця відповідь правильна. На онлайн курсах часто учням задають проходити тести, особливо якщо це курс для випускників, тому оптимізація процесу проходження тестів може зекономити немало часу.

На рис. 1.4 можна побачити, як виглядає завдання ЗНО на сайті, що аналізується. Інтерфейс приємний, сучасний, всі важливі елементи виділені. А на рис. 1.5 зображено розгорнутий коментар до завдання, завдяки якому можна не витрачати час на пошуки вирішення проблеми із завданням, а просто розібрати коментар та відкрити те правило, про яке говориться в коментарі.

◀

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

▶

Завдання 33 з 43

One of my **(33)** _____ memorable summers was my trip down under to New Zealand and Australia when I had the opportunity to be a part of the *People to People* Student Exchange program.

Our delegation's twenty-day stay (34) _____ with action-packed thrills from diving on the Great Barrier Reef to nature hikes in the tropics of New Zealand. Our group also travelled to a Maori village where we (35) _____ overnight learning about the native tribal rituals, dances, customs, and beliefs.

The trip also included museum visits, long flights and bus rides. I quickly felt (36) _____ home with the new friends I met from my own delegation. I am still closely involved with many friends from the trip. If you (37) _____ a chance to be involved with the *People to People* summer programs, I highly recommend doing it.

A

 the more

B

 the most

C

 most

D

 much

Позначте відповіді:

A

☐

B

☐

C

☐

D

☐

Рисунок 1.4 – Вигляд завдання з англійської мови

Коментар

ТЕМА: Використання мови. Уміння правильно використовувати граматичні форми частин мови та частини мови відповідно до контексту

У цьому завданні пропонується заповнити пропуски, вибравши необхідне за граматичною формою слово серед чотирьох наданих варіантів.

Швидко прочитайте текст, ігноруючи пропуски, щоб зрозуміти його тему. Уважно прочитайте граматичні форми слова, яке потрібно вставити в текст замість пропуску. Проаналізуйте не тільки зміст речення, а і його структуру. Обов'язково зверніть увагу на речення, що передують реченню з пропуском і розташоване відразу за ним. Це допоможе вибрати правильну відповідь.

У реченні «*One of my (33) _____ memorable summers was my trip down under to New Zealand ...*» (Одним із моїх найбільш пам'ятних літ була подорож на інший кінець світу до Нової Зеландії ...) пропущений прикметник у найвищому ступені порівняння. Серед запропонованих варіантів тільки варіант **C** *most* (найбільший) вписується за змістом та граматикою у пропуск в цьому реченні. Перед прикметником стоїть присвійний займенник, тому прикметник використано без означеного артикля. Отже, правильно буде обрати варіант **C**.

Правильна відповідь – C.

Рисунок 1.5 – Вигляд коментаря до завдання

Перевагами цього вебсайту є:

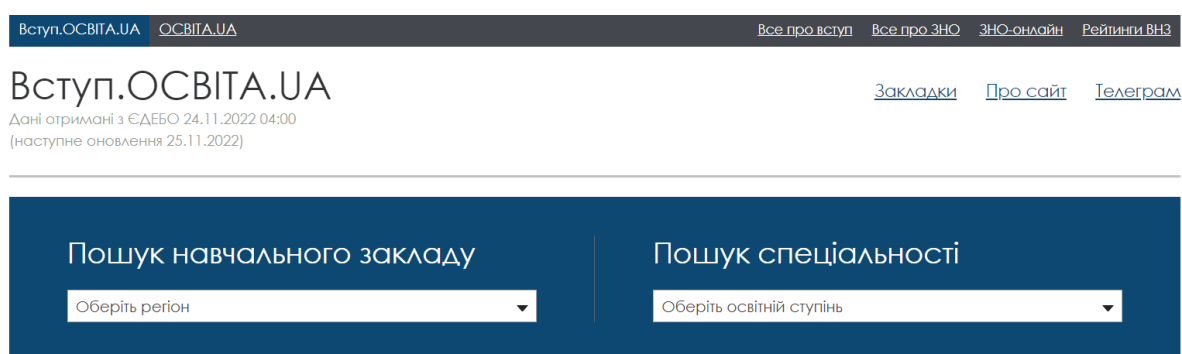
- змістовність;
- наявність найактуальнішої інформації щодо освіти в Україні;
- можливість самостійної роботи над помилками в тестах ЗНО/ЄВІ

завдяки розгорнутим коментарям до кожного завдання.

Єдиним недоліком, який трохи заважає спокійному користуванню сайтом, є відсутність єдиного дизайну в різних розділах сайту. На рис. 1.3, 1.6 та 1.7 зображені різні розділи одного сайту. Було б краще, якби хоча б навігаційна панель була в єдиному дизайні.



Рисунок 1.6 – Один із розділів сайту



Реклама

Кількість заяв вступників за регіонами

Регіон	Бакалавр та мед.магістр	Магістр	Фаховий молодший бакалавр	Молодший бакалавр	Всього
Київ	206965	78110	25215	110	333027
Вінницька область	23843	10682	10183	111	48665
Волинська область	15718	7952	7537	0	34114
Дніпропетровська область	44391	19416	25303	70	96687

Рисунок 1.7 – Один із розділів сайту

1.2.4 Порівняльний аналіз існуючих рішень

Отже, було проаналізовано такі системи, як Moodle, zno.osvita.ua та сервіси від Google, результати аналізу було додано у табл. 1.1. Видно, що кожна система

має свої переваги та недоліки. В дипломній роботі планується створити власний вебзастосунок із удосконаленням переваг та вилученням недоліків існуючих рішень.

Таблиця 1.1 – Порівняння існуючих рішень

Критерії	Існуючі застосунки		
	Сервіси від Google	Moodle	zno.osvita.ua
Ергономічність	Ненагромаджений сучасний дизайн	Незручний та заплутаний інтерфейс програми, відсутність інтуїтивно зрозумілого управління	В цілому, сайт user-friendly, але помічена відсутність єдиного дизайну в різних розділах сайту
Функціонал	Багато корисних сервісів: Google Docs/Sheets/Slides, Google Drive, Gmail Google Calendar, Google Meet	Дуже багато функцій, частину з яких просто не використовують. А прибрати їх можна лише за допомогою програмістів.	Є можливість самостійної роботи над помилками в тестах ЗНО/ЄВІ завдяки розгорнутим коментарям до кожного завдання.
Налаштування під свої потреби	Легко можна самостійно налаштувати середовище для роботи онлайн школи	Щоб організувати навчання та створити повноцінні курси, систему необхідно встановити, зазвичай для цього наймають ІТ-фахівця	Так як функціонал не такий широкий (тільки тести), нічого налаштовувати не потрібно
Вартість	Майже весь функціонал безкоштовний, оплата необхідна тільки для деяких функцій, як збільшення пам'яті в Google Drive, створення корпоративної пошти і т.д.	Використання безкоштовне	Базові функції безкоштовні, а для отримання преміум-доступу необхідно заплатити

1.3 Технічне завдання

Програма матиме свою архітектуру та клієнт-сервер, тобто буде можливий віддалений доступ до бази даних (БД).

Функції веб-застосунку:

— реєстрація та авторизація користувача;

- можливість відновлення пароля;
- адміністративна панель для маніпуляції усіма даними сайту: створення, видалення та редагування даних учнів, матеріалів, розподіл всіх матеріалів для учнів в залежності від групи та інше;
- сортування адміністратором учнів за групами;
- розміщення адміністратором відеозаписів пройдених уроків в розділі «Відеоуроки» та перегляд учнями цього розділу;
- розміщення адміністратором файлів з домашнім завданням (файли в різних форматах та тести) в розділі «Домашні завдання» та перегляд учнями цього розділу;
- розміщення учнями викнаного домашнього завдання в особистому кабінеті в розділі «Домашні завдання»;
- тести в розділі «Домашні завдання» мають працювати наступним чином: тести можуть бути з обмеженням часу та без; має бути можливість для адміністратора перемішувати питання в рамках одного тесту, щоб в особистих кабінетах учнів питання були в різному порядку; підрахунок балів за тест; має бути можливість додати в тести коментарі за бажанням адміністратора, щоб після проходження тестів виділялась не тільки правильна відповідь, а і відображались розгорнуті коментарі до кожної відповіді з посиланнями на файли в розділі «Матеріали»;
- розміщення адміністратором додаткових матеріалів для навчання (підручники, файли з граматиною в різних форматах та інше,) в розділі «Матеріали» та перегляд учнями цього розділу;
- пошук за назвою в кожному розділі;
- створення та редагування інформації (розклад, новини про перенесення уроків тощо) адміністратором в залежності від групи учня в розділі «Розклад» та перегляд учнями цього розділу.

1.4 Висновки за розділом 1

Проаналізувавши методи та програмні засоби реалізації систем для онлайн шкіл можна сказати, що за останні роки онлайн навчання сильно прогресувало, було створено багато актуальних застосунків для автоматизації роботи онлайн шкіл, але через великий попит онлайн шкіл на користування власними вебзастосунками, які б повністю задовольняли усім вимогам, що висуваються до такого типу програмного забезпечення, для оптимізації роботи було вирішено розробити вебзастосунок для приватної онлайн школи іноземних мов.

Програмний продукт дипломної роботи має поєднати в собі функціонал, який є у існуючих альтернативах та стати новим, повноцінним інструментом для організації роботи та навчання в онлайн школах.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вимоги до вебзастосунку

Усі програми починаються з ідеї, яка є потенційним рішенням визначеної проблеми. Веб-додатки не є винятком. Після визначення ідеї для веб-програми важливо формально визначити вимоги до програми для зацікавлених сторін.

Вимога — це будь-яка умова, якій повинна відповідати система або програмний засіб, що розробляється. Вимогою може бути можливість, яку система має мати, і обмеження, якому система повинна задовольняти. Відповідно до загальноприйнятого міжнародного стандартного глосарія – Глосарія термінів програмної інженерії IEEE, вимога це:

- можливості чи умови, які мають бути у системи чи системних компонентах, щоб виконати контракт чи задовольняти специфікаціям, стандартам, чи іншим формальним документам;
- можливості чи умови, які потрібні користувачеві для досягнення цілей чи вирішення проблем;
- документоване подання можливостей чи умов для пунктів 1 та 2.

Управління вимогами, вироблення вимог та визначення вимог — складові успіху будь-якого ІТ-проекту.

За даними дослідження, проведеного IBM в галузі ІТ, 60% витрат часу організації-розробники програмного забезпечення несуть внаслідок неефективного підходу до управління вимогами. В організаціях, які не мають достатніх можливостей бізнес-аналізу, проекти втричі частіше закінчуються невдачею, ніж успіхом. Виявлено, що можна знизити перевитрати за проектом на 20%, якщо скоротити кількість неповних та неточних вимог, тобто якщо правильно визначити вимоги та управляти ними [2].

Процес формування вимог (тобто потреб, завдань і цілей програми) відомий як збір вимог. Процес збору вимог є першим і вирішальним етапом процесу Software Development Lifecycle (SDLC), життєвого циклу розробки програмного забезпечення, оскільки він закладає основу для наступних етапів.

Саме на цьому етапі конкретизується оцінка бізнес-цілей або завдань, яких має досягти вебзастосунок.

У розробці програмного забезпечення збір вимог визначає функціональні та нефункціональні вимоги до веб-додатку [3].

Функціональні вимоги (functional requirements) – це потреби, пов’язані з технічною функціональністю системи. Вони визначають функціональність програмного забезпечення, яку мають побудувати розробники, щоб задовольнити потреби користувачів у рамках бізнес-вимог. Їх іноді називають behavioral requirements (вимогами поведінки), вони містять положення з традиційним «повинен» або «має»: «Система повинна надсилати користувачам електронною поштою підтвердження про реєстрацію». Як правило, функціональні вимоги лягатимуть в основу технічного завдання на розробку ПЗ.

Нефункціональні вимоги дозволяють визначити критерії, за якими можна оцінити роботу конкретної системи, тобто визначити умови, а не конкретну поведінку. У той час як функціональні вимоги визначають, що повинна робити система, нефункціональні вимоги визначають, якою має бути система. Нefункціональні вимоги також називаються «атрибутами якості» системи. Прикладами нефункціональних вимог є легкість та простота використання, зручність експлуатації та технічного обслуговування, надійність та стійкість до збоїв, розширюваність, вимоги до інтерфейсів користувача.

Зараз існує багато методів для визначення вимог до застосунків, зазвичай організації, які здійснюють бізнес-діяльність, використовують опитування, анкетування, «фотографування» робочого дня, спостереження за виробничою діяльністю, аналіз конкурентних продуктів, аналіз статистики використання попередніх версій системи та інше.

На етапі визначення вимог до вебзастосунку для онлайн навчання було використано метод аналізу конкурентних продуктів, проаналізовано переваги та недоліки інших вебзастосунків, які використовуються для онлайн навчання, що дало можливість визначити функціонал, яким має бути наповнений вебзастосунок. Але іноді таких методів недостатньо.

В сучасному світі навчання стало популярним у будь-якому віці, саморозвитку зараз приділяється особлива увага. Також користувачі вебзастосунків пред'являють високі вимоги до платформ, які використовують. У випадку, якщо застосунок не задовольняє користувача, йому не важко буде вибрати іншу платформу з подібним, але більш якісним функціоналом. Тому організаціям, які здійснюють бізнес-діяльність у сфері освіти, необхідно і важливо моментально орієнтуватися в нестійкому та швидко мінливому зовнішньому середовищі, підлаштовуватися під його зміни. Керівникам проектів необхідно вносити корективи в початковий план, які допоможуть організації, а в нашому випадку – онлайн школі, адаптуватися в середовищі, що динамічно змінюється. І це не завжди вдається зробити своєчасно. Саме тому в даній роботі пропонується розглянути додаткові методи аналізу вимог до проекту. Для дослідження було обрано QFD аналіз та сценарний підхід.

2.1.1 Дослідження QFD аналізу

В даний час споживач має великий вибір різних послуг і товарів, які мають мінімальні відмінності між собою. Споживачі роблять свій вибір, ґрунтуючись на якості товару чи його цінності, бажають отримати як можна більше за витрачені ними гроші. Щоб компанія/організація залишалась конкурентоспроможною, щоб споживач завжди обирав компанію, необхідно зрозуміти, що саме споживачі очікують отримати від продукту, що цінують в продукті, які критерії використовуються при виборі між подібними продуктами-пропозиціями.

Є багато інструментів, які сприяють виявленню цієї інформації. Після проведеного аналізу було обрано для дослідження метод Quality Function Deployment (QFD-матриця), розгортання функцій якості. Головний принцип QFD – зіставлення переваг клієнтів з характеристиками продукту, це дає можливість сформулювати найефективнішу стратегію його розвитку. Завдяки такому аналізу можна з'ясувати ставлення споживачів до продукту, виявити, що їм найбільше в ньому подобається і не подобається. Аналіз з використанням матриці QFD

допомагає з'ясувати, чим керується цільова аудиторія при виборі послуги або товару.

Завдяки QFD аналізу простіше зрозуміти клієнтів, їхні бажання та проблеми, які вони намагаються вирішити за допомогою певного продукту. QFD аналіз можна використовувати протягом усього життєвого циклу продукту, тоді він буде максимально ефективним, з його допомогою буде забезпечена постійна увага до бажань вашого покупця. QFD аналіз більше підходить вже існуючим продуктам, бо при виведенні на ринок нового товару, який був невідомим раніше, потенційним клієнтам важко буде сформулювати свої бажання [4].

Переваги QFD матриці:

- може бути використаної на будь-якому етапі життєвого циклу продукту;
- не вимагає значних витрат;
- покращення продукту відповідно до переваг клієнтів;
- виявлення ключових переваг клієнтів;
- отримання інформації безпосередньо від споживача.

Після аналізу QFD матриці, було виявлено, що так як вона допомагає виявити вподобання користувачів, результати QFD аналізу також можуть бути використані для виявлення актуальних вимог до продукту.

Враховуючи переваги QFD аналізу, пропонується побудувати QFD матрицю для проекту онлайн школи. Вона допоможе виявити необхідність додання нового функціоналу, окрім того, який було вирішено додати до вебзастосунку після аналізу вебзастосунків – аналогів.

Першим кроком для побудови матриці є збір даних від потенційних користувачів. Було вирішено провести анкетування двадцяти людей різного віку, з досвідом навчання в онлайн школах та без. Потенційні користувачі мали відповісти на 8 питань про важливість обраних критеріїв та поставити оцінки кожному критерію вибору. Критерії обирались різні: ті, які стосуються процесу навчання в цілому, та ті, які стосуються саме функціоналу вебзастосунку. Питання мали просту структуру, наприклад: «Наскільки для вас важлива

наявність в школі можливості навчатися за гнучким розкладом?». Потенційний користувач має оцінити важливість кожного пункту та присудити їм певну кількість балів від 1 до 10. Потім необхідно скласти між собою отримані бали з кожної категорії. Кількість балів, отриманих по кожному критерію, та результати всіх наступних кроків наведені на рис. 2.1.

Критерії вибору	Бал	Вага	Front								Back						Переваги школи							
			UI/UX		Швидкість роботи		Перевикористання компонентів		Адаптивність		Швидкість обробки даних		Безпечність		Логічність структури БД		Доступність викладачів		Кваліфікація викладачів		Комунікативні здібності викладачів		Лояльність	
1.Гнучкість розкладу	169	0,13	1	0,13	1	0,13	1	0,13	1	0,13	1	0,13	1	0,13	9	1,17	9	1,17	1	0,13	1	0,13	3	0,39
2.Якість викладання, зручність навчання	200	0,16	9	1,44	9	1,44	1	0,16	9	1,44	9	1,44	1	0,16	3	0,48	9	1,44	9	1,44	9	1,44	1	0,16
3.Різноманітність курсів	144	0,11	1	0,11	1	0,11	1	0,11	1	0,11	1	0,11	1	0,11	1	0,11	3	0,33	9	0,99	3	0,33	1	0,11
4.Лояльна система відвідування	159	0,13	1	0,13	1	0,13	1	0,13	1	0,13	1	0,13	1	0,13	1	0,13	1	0,13	1	0,13	1	0,13	9	1,17
5.Новизна	82	0,06	3	0,18	1	0,06	1	0,06	1	0,06	1	0,06	1	0,06	1	0,06	1	0,06	3	0,18	3	0,18	3	0,18
6.Доступність матеріалів	167	0,13	3	0,39	1	0,13	1	0,13	3	0,39	3	0,39	1	0,13	9	1,17	1	0,13	9	1,17	3	0,39	3	0,39
7.Продуктивність системи (стабільність роботи, швидкість)	186	0,15	3	0,45	9	1,35	9	1,35	3	0,45	9	1,35	3	0,45	9	1,35	1	0,15	1	0,15	1	0,15	1	0,15
8.Дизайн	159	0,13	9	1,17	3	0,39	9	1,17	9	1,17	1	0,13	1	0,13	1	0,13	1	0,13	1	0,13	1	0,13	1	0,13
Результат	1266	1	4		3,74		3,24		3,88		3,74		1,3		4,6		3,54		4,32		2,88		2,68	

Рисунок 2.1 – QFD матриця проекту онлайн школи

Після підсумку отриманих балів з питань необхідно обчислити вагу від їх загальної кількості. Для цього кількість балів за кожним із критеріїв розділимо на загальну суму балів за всіма критеріями вибору. В нашому випадку загальна сума критеріїв вибору становить 1266, а за критерієм «гнучкість розкладу» 169. Таким чином, 169 слід розділити на 1266. Для інших критеріїв застосуємо той же алгоритм обчислення ваги. Отримана вага повинна бути в діапазоні від 0 до 1, а його сума повинна становити 1.

Після цього необхідно з розробниками продукту визначити основні характеристики продукту, які є конкурентними перевагами або відмінними якостями. Їх необхідно розмістити в другій частині таблиці по горизонталі.

Наступний етап – аналіз даних. Для цього на перетинах рядків і стовпців слід помножити значення ваги на одне з трьох значень: 1, 3 або 9. За правилами QFD аналізу ці значення позначають ступінь взаємозв'язку між перевагами споживачів і основними характеристиками. Якщо взаємозв'язок слабкий – слід помножити на 1, середній – на 3, якщо взаємозв'язок сильний – на 9 [5].

Як було з'ясовано, UI/UX є однією з основних характеристик. Далі нам потрібно з'ясувати її вплив на вибір користувачів. Таким чином, ми запитуємо себе: наскільки UI/UX взаємопов'язаний, наприклад, з першим критерієм – гнучкість розкладу? Не взаємопов'язаний, тому ми множимо на 1. Але UI/UX має вплив на другий критерій (якість викладання, зручність навчання), тому множимо на 9. Завдяки user – friendly інтерфейсу вся інформація в вебзастосунку буде структурованою та зручною в використанні.

Наведемо ще приклади. Завдяки логічному UI/UX можна задовольнити потреби користувача в доступності матеріалів за тією ж причиною – так як на сайті все буде ергономічно структуровано, буде легко та зручно користуватись матеріалами для навчання, але все ж UI/UX не єдине, що відповідає за цей критерій, тому коефіцієнт 3. А логічність структури бази даних та наявність доступний викладачів впливають на критерій «гнучкість розкладу», бо правильно налаштована база дозволить легко адаптувати розклад занять під різні групи та викладачів.

Після цього ми підсумовуємо бали за кожною з характеристик. В результаті можна визначити найбільш значущі властивості продукту і зрозуміти, на що слід насамперед звернути увагу для вдосконалення продукту. QFD аналіз також може допомогти виявити зайві функції, які з часом втратили свою актуальність.

2.1.2 Застосування сценарного підходу в реалізації функціональних вимог

Для того, щоб забезпечити гнучкість та ефективність розвитку організації, необхідні такі інструменти, які дозволяють враховувати різні варіанти подій розвитку. Іншою методологією, яку пропонується використати для виявлення вимог до проекту, є сценарний підхід. Сценарний підхід – це підхід, у якому проводять теоретичний аналіз системи, що має на меті виявити можливі варіанти розвитку подій та визначити їх наслідки. Цей універсальний метод застосовують в сферах інформатики, економіки, соціології для прогнозування, а також в прикладних сферах, таких, як керування підприємством для вибору оптимальної стратегії розвитку. Передумови формування та основні принципи та галузі практичного застосування сценарного підходу розглядали такі вчені, як Дж. К. Джонс та Дж. А. Огілві.

Слово сценарій бере початок від слова *scaena* (латинська мова), що означає "сцена". У XX столітті Герман Кан, провідний аналітик корпорації RAND, адаптував цей термін для використання, коли йдеться про написання можливих історій розвитку подій у майбутньому. Існує безліч різних визначень поняття сценарій. Наприклад, якась статична картинка майбутнього або послідовність подій у часі; також виділяються відмінності і за іншими характеристиками сценарію. Філіп Ван Ноттен, голландський вчений, виділяє загальні характеристики сценаріїв, які відзначають більшість фахівців у цій сфері, це: опис альтернативних варіантів майбутнього; гіпотетична природа сценарію; причинно-наслідковий зв'язок та наявність внутрішньої узгодженості; опис майбутнього як кінцевого результату або як ланцюжки подій; можливість використовувати сценарій як базис для дій; достовірність; описова природа; пояснювальна природа; з'єднання минулого, сьогодення та майбутнього. В результаті узагальнення цих характеристик приймемо як найбільш загальне визначення сценарію таке: сценарій - це послідовний опис альтернативних гіпотетично можливих варіантів розвитку подій у майбутньому, що відображає різні точки

зору на минуле, сьогодення та майбутнє, а також яке може служити базисом для планування дій [6].

За останні десятиліття сценарний підхід набув особливої популярності через об'єктивну необхідність системного дослідження економічних об'єктів та прискорення процесів у світовій економіці, а це ускладнює прогнозування подальшого розвитку організацій. Для цього доцільно використовувати сценарний підхід завдяки різноплановому аналізу інформації, що надходить ззовні, та середовища, також він допомагає вирішувати управлінські завдання. Сценарний підхід допомагає приймати оптимальні рішення в певних обставинах та допомагає спрогнозувати наслідки обраних дій [7].

Сценарний підхід іноді є витратним, але у поєднанні з традиційними інструментами дослідження він сприяє зниженню ризиків, дозволяє сформулювати ефективний захист від потенційних загроз, створює умови для серйозного та стійкого базису при прийнятті стратегічних рішень, забезпечує якість прогнозів.

Раніше цю методологію застосовували тільки великі організації, але зараз її починають використовувати і середні організації при плануванні та аналізі бізнес – діяльності.

Теоретичні та практичні підходи до розвитку ІТ-продуктів активно розробляються протягом останніх років, та ще багато років будуть актуальні та затребувані спеціальні наукові дослідження, спрямовані на виявлення шляхів удосконалення управління проектом, а в нашому випадку – це визначення вимог до продукту. Аналіз наукових публікацій дозволив встановити, що здійснено цілу низку досліджень, спрямованих на виявлення теоретичних та методичних підходів сценарного підходу в різних галузях: у політичному аналізі та прогнозуванні, у плануванні виробничої діяльності машинобудівних підприємств, у стратегічному плануванні, у сільськогосподарській галузі, при аналізі інвестиційних проектів і т.д. Це пов'язано з тим, що у діяльності підприємств тієї чи іншої галузі є свої особливості, як, власне, і при проектуванні ІТ-продукту. Сучасні напрямки розвитку ІТ-сфери звертають особливу увагу на скорочення непродуктивних витрат ІТ-проектів. Одним із способів скорочення подібних

витрат є максимально точне виявлення змісту ІТ-проекту в ході його ініціації та планування.

В сучасних умовах керівникам ІТ-продуктів доводиться керувати дуже складними системами і безперервно здійснювати вибір однієї з численних альтернатив при вирішенні проблем, що постійно виникають, враховуючи, що кожне управлінське рішення породжує безліч наслідків, які також доводиться враховувати. Ефективно долати вищезгадані тенденції можливо із застосуванням відповідної системи аналізу виникаючих проблем та моделювання відповідних сценаріїв їх вирішення.

Поведінковий сценарій – усереднений варіант поведінки групи людей, в якому поєднуються природні бажання людини з умовностями та нормами зовнішнього світу. Тобто це передбачуваний чи можливий перебіг подій у тій чи іншій галузі, що стосується діяльності організації. При проектуванні об'ємних вебзастосунків сценарний підхід допомагає проаналізувати можливі сценарії використання окремих сторінок та всього застосунку в цілому, допомагає визначити «слабкі» місця. Не так багато власників ІТ-продукту безпосередньо вказують сценарний підхід як складову частину методів розвитку свого проекту, однак окремі його принципи простежуються в інших підходах, які використовуються для оптимізації функціонального наповнення додатків.

Метою сценарного підходу є виявлення наслідків сьогодишньої поведінки організації та позначення прийнятих припущень, концентруючи увагу на правильно поставлених проблемах. В даному проекті сценарний підхід буде використано для реалізації функціональних вимог вебзастосунку. Для цього необхідно розглянути шлях експлуатації вебзастосунку з позиції користувача [8].

З погляду діяльності в сфері ІТ, сценарій - це оповідання, за допомогою якого визначається, як буде кінцевий споживач використовувати виріб, це дозволяє розробникам та іншим членам команди визначити місце виробу в повсякденному житті людини. Сценарний аналіз допомагає визначити можливі сценарії поведінки споживачів різних типів під час використання ними продукту.

Сценарний аналіз можна подавати в різному вигляді: від словесного (контекстного) опису до візуалізації у форматі фотографій, схем, малюнків і т.д. Зазвичай форма подання сценарного аналізу безпосередньо залежить від галузі, яка досліджується, специфіки дослідження та обсягу інформації. Так як продукт, який розробляється, на даний момент не можна віднести до великих проектів, достатньо буде використати контекстний опис або схеми. При плануванні ІТ-продукту сценарний метод може допомогти визначити функціональні вимоги.

Наведемо приклади. В проекті, який розробляється, передбачається, що у кожного учня буде особистий кабінет. Зазвичай для цього просто потрібно пройти реєстрацію на сайті. Але в нашому випадку, якщо зробити типічну реєстрацію на сайті, то потенційний учень отримає доступ до неоплачених приватних даних, тобто доступ до матеріалів особистого кабінету (рис 2.2).

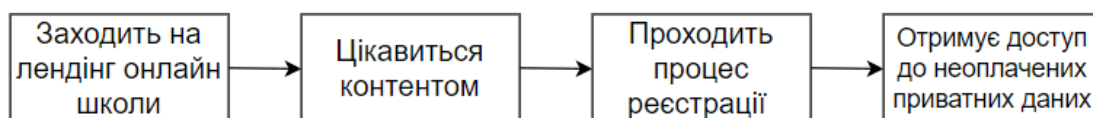


Рисунок 2.2 – Сценарій використання вебзастосунку потенційним учнем

В великих онлайн школах зазвичай процес оплати та розподілу в групу автоматизований та присутній на сайті, але для нової невеликої онлайн школи такий функціонал може бути зайвим. Тому більш доречним є реєстрація учня через адміністративну панель (рис. 2.3) вже після проведення заходів, необхідних для того, щоб оформити учня у школу.

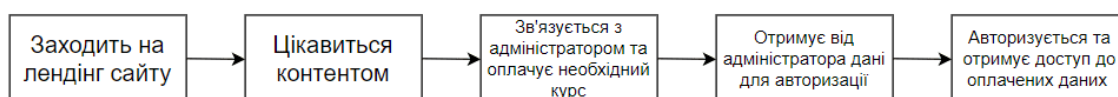


Рисунок 2.3 – Сценарій використання вебзастосунку потенційним учнем

Також можна розглянути випадок, коли учень онлайн школи залишився незадоволеним заняттям та бажає залишити відгук. При визначенні технічного

завдання не було передбачено реалізацію функції підтримки в особистих кабінетах студентів. Тому учень буде вимушений прикласти зусиль, щоб відправити анонімний відгук (анонімні відгуки для деяких людей більш привабливі, ніж неанонімні) для привернення уваги на ситуацію (рис. 2.4).

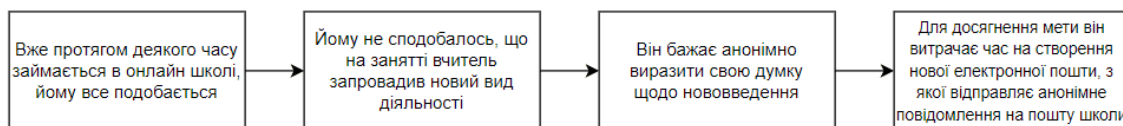


Рисунок 2.4 – Сценарій для досягнення мети залишити відгук без вбудованої можливості

При аналізі, що саме може піти не так на онлайн уроках та якою може бути реакція учня/вчителя чи іншого учасника, було виявлено, що доречним буде додати до вебзастосунку розділ підтримки (рис. 2.5), який буде корисним навіть на початкових стадіях роботи школи, а при її розширенні стане незамінною функцією.

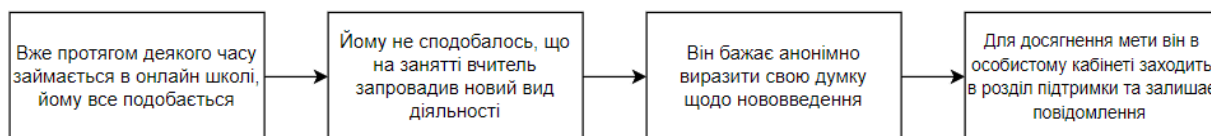


Рисунок 2.5 – Сценарій для досягнення мети залишити відгук із вбудованою можливістю

Також в контексті сценарного аналізу варто виділити такі корисні інструменти, як Jobs to Be Done (JtBD) та Customer Journey Map (CJM). Застосування концепції JtBD дозволяє моделювати шлях клієнта до продукту і робити його привабливішим. Для цього потрібно уявляти собі шлях клієнта, коли він приймає рішення, виявляти його справжні цілі та диференціювати послуги та продукти на основі JtBD. Ця концепція допомагає творчо обмежити уявлення про продукт та дозволяє зосередитися на створенні речей, які люди справді хочуть [9].

Концепція CJM – це практичний інструмент для реалізації концепції Jobs to Be Done. Це карта, яку використовують для аналізу мотивів та очікувань споживача. Така мапа дає уявлення про почуття, дії, та настрої користувача, включаючи позитивні, негативні та нейтральні моменти, які бувають при тривалій взаємодії з продуктом або послугою. Інформацію в CJM зазвичай групують наступним чином: попередньо проводять інтерв'ювання/анкетування, потім на основі цього обирають найважливіші твердження та висловлювання респондентів, що відображають враження та емоції; далі стосовно об'єкта, який досліджується, виявляється конкретна проблема, до якої у наступному пункті пропонуються будь-які пошукові концепти та рішення. CJM відображає серію подій і дій користувача, таким чином допомагає розробникам розширити бачення і зосередитися не тільки на оперативній системноорієнтованій стороні питання, але і на контексті, в якому продукт або послуга, що розробляється, використовуватиметься на практиці [10].

Таким чином, застосування сценарного методу при розробці продукту допомагає розширити список актуальних вимог до продукту.

2.2 Висновки за розділом 2

Було розглянуто типи вимог до ІТ-продуктів та проаналізовано застосування QFD аналізу та сценарного підходу в реалізації вимог. Завдяки принципам розглянутих підходів біло роширено список вимок до вебзастосунку, який розроблюється.

3 ОПИС ПРОГРАМИ

3.1 Логічна структура програми

Для ознайомлення з системою було спроектовано блок – схему, яка відображає основні дії користувача (рис. 3.1 – 3.3). Після того, як користувач ознайомився з курсами та програмами навчання, яка пропонує школа, та записався на навчання за допомогою адміністратора, він може авторизуватись (рис. 3.1). Коли учень забуде свій пароль, він зможе легко відновити його, якщо при авторизації натисне на «Забули пароль» та отримає на пошту дані для відновлення пароля. Після авторизації учень потрапляє в особистий кабінет, де може обрати необхідну вкладку: Профіль, Домашні завдання, Матеріали, Відеоуроки, Розклад (рис. 3.2). На сторінці профілю користувач може відредагувати свої дані, також може залишити відгук. На сторінці з домашніми завданнями учень може обрати конкретну дату уроку та перейти на сторінку із завданнями. Там він може знайти інформацію про завдання, які має зробити, файли із завданнями, які може скачати на свій пристрій, відеозапис пройденого уроку, тест, який необхідно пройти, а також саме тут учень може завантажити файли із виконаним домашнім завданням, щоб вчитель зміг перевірити.

В розділі «Матеріали» учень може скористуватись пошуком по сторінці, переглядати необхідні файли та скачувати їх (рис. 3.3). В розділі «Відеоуроки» теж можна скористатись пошуком, щоб швидше знайти відео на номером/назвою уроку та подивитись його. В розділі «Розклад» можна перевірити актуальний розклад занять.

А адміністратор може виконувати наступні функції через адміністративну панель: створення, видалення та редагування всіх даних, які студенти бачать в особистих кабінетах, а також адміністратор може обробляти особисті дані студентів, розподіляти по групах і т.п.

```

graph LR
    Start((1)) --> Profile[Сторінка профілю]
    Profile --> EditData{Відредагувати особисті дані}
    EditData -- Так --> EditDataBox[Редагуємо особисті дані]
    EditData -- Ні --> LeaveFeedback{Залишити відгук}
    LeaveFeedback -- Так --> LeaveFeedbackBox[Залишаємо відгук]
    LeaveFeedback -- Ні --> Profile
    
    Profile --> HomeTasks[Сторінка з домашніми завданнями]
    HomeTasks --> SearchLesson{Виконати пошук по сторінці}
    SearchLesson -- Так --> SearchLessonBox[Виконуємо пошук]
    SearchLesson -- Ні --> OpenLesson{Відкрити ДЗ з певного заняття}
    OpenLesson -- Так --> WatchVideo[Подивитись відеоурок]
    WatchVideo --> WatchVideoBox[Відкриваємо та дивимось відео]
    WatchVideoBox -- Ні --> OpenFiles{Відкрити файли з ДЗ}
    OpenFiles -- Так --> OpenFilesBox[Відкриваємо та переглядаємо файли з ДЗ]
    OpenFilesBox -- Ні --> TakeTest{Пройти тест}
    TakeTest -- Так --> TestResults[Проходимо тестування]
    TestResults -- Так --> AnalyzeResults{Проаналізувати результати}
    AnalyzeResults -- Ні --> Profile
    AnalyzeResults -- Так --> UseExplanations[Користуючись поясненням до відповідей, працюємо над помилками]
    UseExplanations --> UnderstandRule{При користуванні поясненнями перейти за посиланням до правила}
    UnderstandRule -- Ні --> Profile
    UnderstandRule -- Так --> UnderstandRuleBox[Розібрати правило]
    UnderstandRuleBox --> Profile
    
    Profile --> Materials[Сторінка з матеріалами]
    Materials --> Videos[Сторінка з відеоуроками]
    Videos --> VideosBox[Відеоуроки]
    VideosBox --> VideosBox2[Відеоуроки]
    VideosBox2 --> VideosBox3[Відеоуроки]
    VideosBox3 --> VideosBox4[Відеоуроки]
    VideosBox4 --> VideosBox5[Відеоуроки]
    VideosBox5 --> VideosBox6[Відеоуроки]
    VideosBox6 --> VideosBox7[Відеоуроки]
    VideosBox7 --> VideosBox8[Відеоуроки]
    VideosBox8 --> VideosBox9[Відеоуроки]
    VideosBox9 --> VideosBox10[Відеоуроки]
    VideosBox10 --> VideosBox11[Відеоуроки]
    VideosBox11 --> VideosBox12[Відеоуроки]
    VideosBox12 --> VideosBox13[Відеоуроки]
    VideosBox13 --> VideosBox14[Відеоуроки]
    VideosBox14 --> VideosBox15[Відеоуроки]
    VideosBox15 --> VideosBox16[Відеоуроки]
    VideosBox16 --> VideosBox17[Відеоуроки]
    VideosBox17 --> VideosBox18[Відеоуроки]
    VideosBox18 --> VideosBox19[Відеоуроки]
    VideosBox19 --> VideosBox20[Відеоуроки]
    VideosBox20 --> VideosBox21[Відеоуроки]
    VideosBox21 --> VideosBox22[Відеоуроки]
    VideosBox22 --> VideosBox23[Відеоуроки]
    VideosBox23 --> VideosBox24[Відеоуроки]
    VideosBox24 --> VideosBox25[Відеоуроки]
    VideosBox25 --> VideosBox26[Відеоуроки]
    VideosBox26 --> VideosBox27[Відеоуроки]
    VideosBox27 --> VideosBox28[Відеоуроки]
    VideosBox28 --> VideosBox29[Відеоуроки]
    VideosBox29 --> VideosBox30[Відеоуроки]
    VideosBox30 --> VideosBox31[Відеоуроки]
    VideosBox31 --> VideosBox32[Відеоуроки]
    VideosBox32 --> VideosBox33[Відеоуроки]
    VideosBox33 --> VideosBox34[Відеоуроки]
    VideosBox34 --> VideosBox35[Відеоуроки]
    VideosBox35 --> VideosBox36[Відеоуроки]
    VideosBox36 --> VideosBox37[Відеоуроки]
    VideosBox37 --> VideosBox38[Відеоуроки]
    VideosBox38 --> VideosBox39[Відеоуроки]
    VideosBox39 --> VideosBox40[Відеоуроки]
    VideosBox40 --> VideosBox41[Відеоуроки]
    VideosBox41 --> VideosBox42[Відеоуроки]
    VideosBox42 --> VideosBox43[Відеоуроки]
    VideosBox43 --> VideosBox44[Відеоуроки]
    VideosBox44 --> VideosBox45[Відеоуроки]
    VideosBox45 --> VideosBox46[Відеоуроки]
    VideosBox46 --> VideosBox47[Відеоуроки]
    VideosBox47 --> VideosBox48[Відеоуроки]
    VideosBox48 --> VideosBox49[Відеоуроки]
    VideosBox49 --> VideosBox50[Відеоуроки]
    VideosBox50 --> VideosBox51[Відеоуроки]
    VideosBox51 --> VideosBox52[Відеоуроки]
    VideosBox52 --> VideosBox53[Відеоуроки]
    VideosBox53 --> VideosBox54[Відеоуроки]
    VideosBox54 --> VideosBox55[Відеоуроки]
    VideosBox55 --> VideosBox56[Відеоуроки]
    VideosBox56 --> VideosBox57[Відеоуроки]
    VideosBox57 --> VideosBox58[Відеоуроки]
    VideosBox58 --> VideosBox59[Відеоуроки]
    VideosBox59 --> VideosBox60[Відеоуроки]
    VideosBox60 --> VideosBox61[Відеоуроки]
    VideosBox61 --> VideosBox62[Відеоуроки]
    VideosBox62 --> VideosBox63[Відеоуроки]
    VideosBox63 --> VideosBox64[Відеоуроки]
    VideosBox64 --> VideosBox65[Відеоуроки]
    VideosBox65 --> VideosBox66[Відеоуроки]
    VideosBox66 --> VideosBox67[Відеоуроки]
    VideosBox67 --> VideosBox68[Відеоуроки]
    VideosBox68 --> VideosBox69[Відеоуроки]
    VideosBox69 --> VideosBox70[Відеоуроки]
    VideosBox70 --> VideosBox71[Відеоуроки]
    VideosBox71 --> VideosBox72[Відеоуроки]
    VideosBox72 --> VideosBox73[Відеоуроки]
    VideosBox73 --> VideosBox74[Відеоуроки]
    VideosBox74 --> VideosBox75[Відеоуроки]
    VideosBox75 --> VideosBox76[Відеоуроки]
    VideosBox76 --> VideosBox77[Відеоуроки]
    VideosBox77 --> VideosBox78[Відеоуроки]
    VideosBox78 --> VideosBox79[Відеоуроки]
    VideosBox79 --> VideosBox80[Відеоуроки]
    VideosBox80 --> VideosBox81[Відеоуроки]
    VideosBox81 --> VideosBox82[Відеоуроки]
    VideosBox82 --> VideosBox83[Відеоуроки]
    VideosBox83 --> VideosBox84[Відеоуроки]
    VideosBox84 --> VideosBox85[Відеоуроки]
    VideosBox85 --> VideosBox86[Відеоуроки]
    VideosBox86 --> VideosBox87[Відеоуроки]
    VideosBox87 --> VideosBox88[Відеоуроки]
    VideosBox88 --> VideosBox89[Відеоуроки]
    VideosBox89 --> VideosBox90[Відеоуроки]
    VideosBox90 --> VideosBox91[Відеоуроки]
    VideosBox91 --> VideosBox92[Відеоуроки]
    VideosBox92 --> VideosBox93[Відеоуроки]
    VideosBox93 --> VideosBox94[Відеоуроки]
    VideosBox94 --> VideosBox95[Відеоуроки]
    VideosBox95 --> VideosBox96[Відеоуроки]
    VideosBox96 --> VideosBox97[Відеоуроки]
    VideosBox97 --> VideosBox98[Відеоуроки]
    VideosBox98 --> VideosBox99[Відеоуроки]
    VideosBox99 --> VideosBox100[Відеоуроки]
    VideosBox100 --> VideosBox101[Відеоуроки]
    VideosBox101 --> VideosBox102[Відеоуроки]
    VideosBox102 --> VideosBox103[Відеоуроки]
    VideosBox103 --> VideosBox104[Відеоуроки]
    VideosBox104 --> VideosBox105[Відеоуроки]
    VideosBox105 --> VideosBox106[Відеоуроки]
    VideosBox106 --> VideosBox107[Відеоуроки]
    VideosBox107 --> VideosBox108[Відеоуроки]
    VideosBox108 --> VideosBox109[Відеоуроки]
    VideosBox109 --> VideosBox110[Відеоуроки]
    VideosBox110 --> VideosBox111[Відеоуроки]
    VideosBox111 --> VideosBox112[Відеоуроки]
    VideosBox112 --> VideosBox113[Відеоуроки]
    VideosBox113 --> VideosBox114[Відеоуроки]
    VideosBox114 --> VideosBox115[Відеоуроки]
    VideosBox115 --> VideosBox116[Відеоуроки]
    VideosBox116 --> VideosBox117[Відеоуроки]
    VideosBox117 --> VideosBox118[Відеоуроки]
    VideosBox118 --> VideosBox119[Відеоуроки]
    VideosBox119 --> VideosBox120[Відеоуроки]
    VideosBox120 --> VideosBox121[Відеоуроки]
    VideosBox121 --> VideosBox122[Відеоуроки]
    VideosBox122 --> VideosBox123[Відеоуроки]
    VideosBox123 --> VideosBox124[Відеоуроки]
    VideosBox124 --> VideosBox125[Відеоуроки]
    VideosBox125 --> VideosBox126[Відеоуроки]
    VideosBox126 --> VideosBox127[Відеоуроки]
    VideosBox127 --> VideosBox128[Відеоуроки]
    VideosBox128 --> VideosBox129[Відеоуроки]
    VideosBox129 --> VideosBox130[Відеоуроки]
    VideosBox130 --> VideosBox131[Відеоуроки]
    VideosBox131 --> VideosBox132[Відеоуроки]
    VideosBox132 --> VideosBox133[Відеоуроки]
    VideosBox133 --> VideosBox134[Відеоуроки]
    VideosBox134 --> VideosBox135[Відеоуроки]
    VideosBox135 --> VideosBox136[Відеоуроки]
    VideosBox136 --> VideosBox137[Відеоуроки]
    VideosBox137 --> VideosBox138[Відеоуроки]
    VideosBox138 --> VideosBox139[Відеоуроки]
    VideosBox139 --> VideosBox140[Відеоуроки]
    VideosBox140 --> VideosBox141[Відеоуроки]
    VideosBox141 --> VideosBox142[Відеоуроки]
    VideosBox142 --> VideosBox143[Відеоуроки]
    VideosBox143 --> VideosBox144[Відеоуроки]
    VideosBox144 --> VideosBox145[Відеоуроки]
    VideosBox145 --> VideosBox146[Відеоуроки]
    VideosBox146 --> VideosBox147[Відеоуроки]
    VideosBox147 --> VideosBox148[Відеоуроки]
    VideosBox148 --> VideosBox149[Відеоуроки]
    VideosBox149 --> VideosBox150[Відеоуроки]
    VideosBox150 --> VideosBox151[Відеоуроки]
    VideosBox151 --> VideosBox152[Відеоуроки]
    VideosBox152 --> VideosBox153[Відеоуроки]
    VideosBox153 --> VideosBox154[Відеоуроки]
    VideosBox154 --> VideosBox155[Відеоуроки]
    VideosBox155 --> VideosBox156[Відео
```

Рисунок 3.2 – Блок-схема програми, частина 2

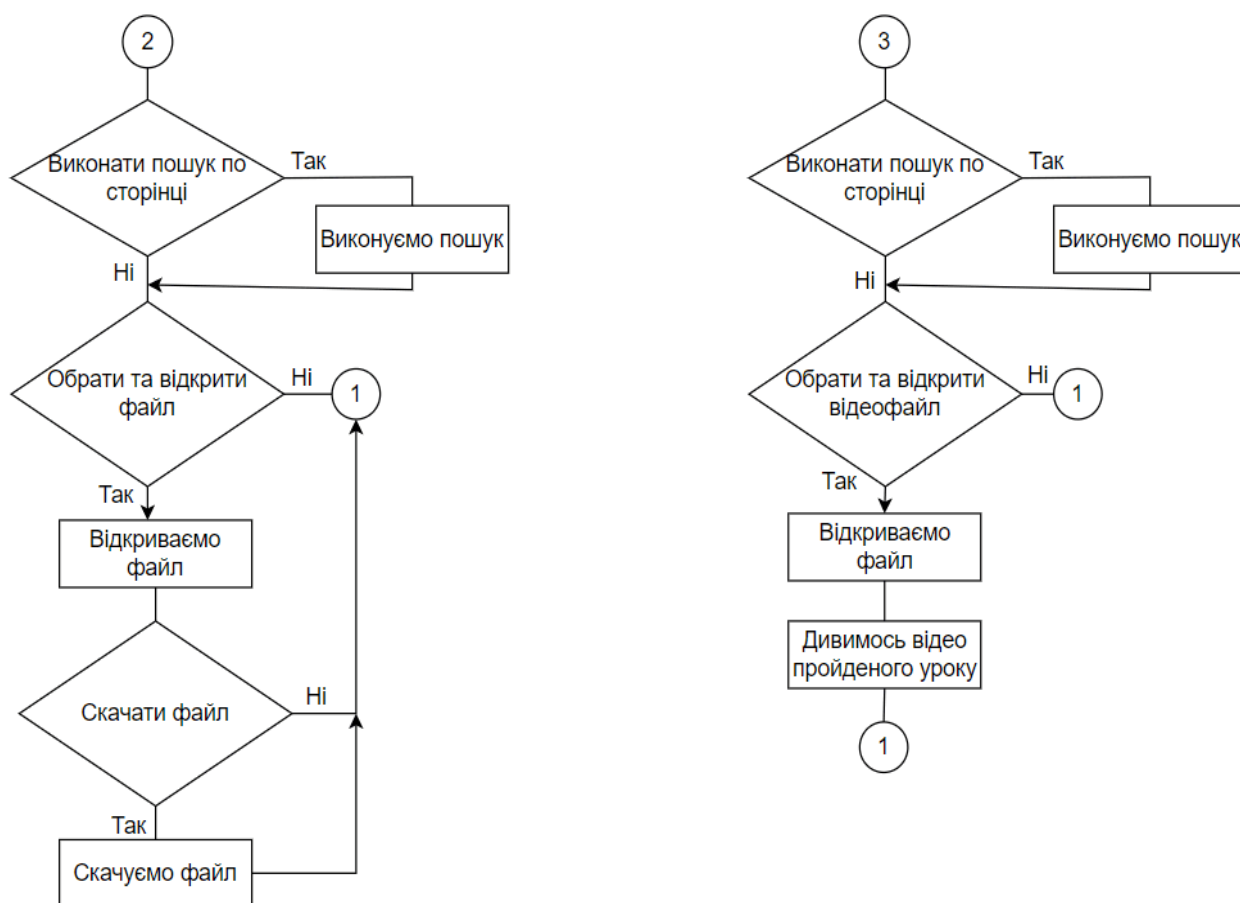


Рисунок 3.3 – Блок-схема програми, частина 3

3.2 Вибір інструментарію для розробки програмного забезпечення

3.2.1 Вибір мови програмування, фреймворку та інших технологій розробки

Для розробки веб-застосунку було обрано React.js – веб-фреймворк для розробки призначених для користувача інтерфейсів на мові програмування JavaScript. JavaScript мультипарадигмова мова програмування, підтримує об'єктно-орієнтований, імперативний та функціональний стилі, але для створення сучасних вебзастосунків потрібно використовувати фреймворки та бібліотеки.

Javascript існує вже чверть століття, значний термін для подібних технологій. За цей час він пройшов велику дорогу, став потужним інструментом ІТ-індустрії. За рівнем популярності JS у групі лідерів.

Було обрано саме фреймворк від JavaScript, бо JS ідеально підходить для веб-розробки:

- незамінність для веб-розробки. Усі популярні браузері підтримують скрипти, наявність повної інтеграції з версткою сторінок (HTML+CSS) та серверною частиною (backend);

- швидкість роботи та продуктивність: часткова обробка веб-сторінок на комп'ютерах користувачів без запитів на сервер. Завдяки цьому заощаджується час і трафік, знижується навантаження на сервер;

- зручність інтерфейсів. Активація кнопок, реагування на наведення/кліки миші, заповнення форм, перевірки введення дають високий рівень юзабіліті;

- простота та раціональність застосування. Для вирішення простого завдання достатньо 5 хвилин, не треба виконувати зайву роботу. Для складіших завдань є різні варіанти вирішення, можна обрати найкраще, адаптувати;

- потужна інфраструктура (екосистема). Перші 10 років цього не було, але потім кількість готових рішень у відкритому доступі так зросла, що працювати з Javascript та його фреймворками стало дуже зручно [11].

В таблиці 2.1 наведено порівняння JS з іншими мовами програмування.

Із всіх бібліотек та фреймворків JS було обрано саме React.js завдяки його багатофункціональності. React.js – потужний інструмент для створення user interface, інтерфейсів користувача (UI). Завдяки бібліотеці React.js можна більше уваги приділяти UI та компонентам програми, UI частина одна із найважливіших для користувача, бо користувач бачить саме UI та взаємодіє з ним. Ця бібліотека допомагає створювати програми швидше, декларативний стиль програмування значно спрощує налагодження компонентів та проекту загалом. Розглянемо основні особливості React.js, які роблять цю бібліотеку такою зручною:

- віртуальна об'єктна модель документа (DOM): React взаємодіє з віртуальною DOM, полегшеною копією реальної об'єктної моделі документа, з якою незручно взаємодіяти. Саме тому спочатку відбувається взаємодія з

віртуальною DOM, а вже потім оновлюється реальна DOM, це забезпечує високу продуктивність та ефективність;

- східний потік даних ("від батька до дитини" / "згори донизу"). Завдяки цьому мінімізується кількість помилок та полегшується налагодження;

- повторне застосування компонентів: при розробці на ReactJS створюються багаторазові компоненти, щоб перевикористовувати їх в інших частинах програми або навіть в інших проектах практично без змін. Більш того, розробники React мають доступ до бібліотек готових компонентів з відкритим вихідним кодом;

- браузерні інструменти React-розробника: розробники React можуть користуватись React Developer Tools (це безкоштовне розширення для Chrome і Firefox, що надає цілий набір віджетів перевірки). Цей інструмент спрощує налагодження, дозволяючи розробникам не лише шукати за списком усіх компонентів, але й переглядати глибоко вкладені компоненти прямо у браузері;

- величезна спільнота: компанія Facebook зробила React потужним інструментом для поліпшення інтерфейсу користувача будь-яких веб-додатків, також активно заохочувала розробників покращувати бібліотеку, тому React став таким завдяки спільним зусиллям людей з усього світу. Спільнота React ділиться великою кількістю важливих бібліотек, що полегшують процес розробки, для розробників відкриті нові способи та можливості покращити якість коду програм [12].

Ядро React містить тільки необхідний функціонал для взаємодії з інтерфейсом. Завдяки цьому воно компактне та є можливість його легкої інтеграції з іншими технологіями.

Отже, було вирішено, що React.js буде кращим варіантом. У табл. 2.2 наведено порівняльну характеристику JS-інструментів для створення додатків у відповідності до різних критеріїв.

Таблиця 2.1 – Порівняння мов програмування

Можливість	Мова					
	JavaScript	Python	PHP	C#	Java	Go
Імперативність	+	+	+	+	+	+
Об'єктно-орієнтованість	+	+	+	+	+	+/-
Функціональність	+/-	+	+/-	+/-	+/-	+/-
First class functions	+	+	+	+	-	-
Анонімні функції	+	+	+	+	+	-
Лексичні замикання	+	+	+	+	+	-
Часткове застосування	+	+	-	-	-	-
Карирування	+	+	+	+	-	-
Пітримка Unicode в ідентифікаторах	+	+	+/-	+	+	+
Значення параметрів за замовчуванням	+	+	+	+	-	+
Наявність бібліотек для роботи з графікою і мультимедіа	+	+	+/-	+	+	-

Таблиця 2.2 – JS-фреймворки

Критерії	Інструмент		
	React.js	Vue.js	Angular.js
Тип	Бібліотека для створення користувацького інтерфейсу	Фреймворк	Фреймворк
Типи додатків	Односторінкові (SPA) і мобільні додатки	Модернізовані односторінкові додатки і підтримка нативних додатків	Нативні, гібридні і веб-додатки
Модель	Віртуальна DOM	Віртуальна DOM	MVC
Переважаюча мова розробки	Рекомендовано використовувати JSX – JavaScript XML	HTML-шаблони та JavaScript.	Рекомендовано використовувати TypeScript
Ідеально підходить для	Веб-додатків і нативних додатків для iOS і Android	Веб-додатків і односторінкових (SPA) додатків	Великомасштабних додатків з багатим функціоналом

Продовження таблиці 2.2

Критерії	Інструмент		
	React.js	Vue.js	Angular.js
Спрямованість і класи залежностей	Одностороння передача даних	Одностороння передача даних	Взаємодії об'єктів відбувається в службах
Зворотня сумісність	Повна сумісність між версіями	Повна сумісність між версіями	Абсолютна залежність від попередніх версій
Розмір бібліотек	100 KB	80 KB	500+ KB

React.JS реалізує всі сучасні підходи до розробки Web-UI, це гнучкий фреймворк, який високо інтегрований зі сторонніми технологіями. Для розробки проекту було використано не лише React.js, а і допоміжні інструменти: Redux, react router, Typescript, antd, mui, socket-io.

React router одна з найпотрібніших функцій React, бо у разі використання веб-сайту користувачам важливо мати можливість безперешкодно переглядати різні сторінки або переходити з однієї сторінки на іншу. React router – бібліотека, яка робить це можливим. React створює односторінкову програму, а react router відіграє важливу роль для відображення декількох уявлень без необхідності перезавантажувати браузер в односторінковому додатку. Варто зазначити, що без react Router практично неможливо відобразити кілька уявлень в додатках React.

Redux – це менеджер станів. За допомогою React теж можна керувати станами, але цей метод має недоліки, наприклад, погане масштабування. Для простих додатків переміщення стану вгору по дереву працює непогано, але у складніших архітектурах зміна стану здійснюється через властивості (props). А краще використовувати для цього зовнішнє глобальне сховище.

Завдяки бібліотеці Redux можна ефективно керувати станом програми. Він ідеально підходить для середніх та великих додатків, в простих програмах Redux зазвичай не потрібен. Redux варто використовувати тоді, коли стандартного менеджера станів React недостатньо. Основні концепції бібліотеки Redux:

— єдине джерело стану: всі дані про state зберігаються в одному місці. State tree – глобальний стан, організований як дерево об'єктів. Завдяки єдиному джерелу робота стає простішою, бо дані знаходяться в одному місці, також воно потрібне для налагодження програми та централізації;

— зміни – лише через редуктори. Вони беруть об'єкт стану компонента, що змінився, та дію, потім на їх основі генерують новий об'єкт стану. Новий об'єкт стану з актуальними даними – це результат роботи редукторів;

— доступ до стану – лише для читання. Глобальний стан заблоковано для запису, компоненти програми можуть лише читати з нього. Завдяки цьому не буду непередбачуваних змін [13].

Отже, завдяки використанню React.js та додаткових інструментів можна створити якісний вебзастосунок для онлайн школи.

3.2.2 Вибір другої мови програмування

TypeScript – це мова програмування, створена для того, щоб уникнути обмежень JavaScript у великих проектах. Наприклад, особливістю TypeScript є статична типізація. В JavaScript використовується динамічна типізація. Це може стати проблемою, якщо проект великий, і над ним працюють багато розробників. Наприклад, може бути таке, що під час виконання коду якась змінна стане NaN. Щоб вирішити проблему, доведеться ретельно вивчити весь код, щоб знайти помилку, через яку змінився тип змінної. А завдяки статичній типізації в TypeScript тип змінних не може змінитися під час виконання коду, тому розробники можуть бути впевнені у статусі змінних. Іншою особливістю статичної типізації, яка полегшує роботу, є те, що перевірка відбувається прямо в редакторі, тобто виконання коду для цього не потрібно. А в JavaScript через динамічну типізацію знайти помилки можна тільки після остаточного написання коду. Це набагато важче, бо якщо код об'ємний, помилки краще відловлювати на початку. Це основні переваги TypeScript, завдяки яким можна впевнитись, що його використання в проекті, що розробляється, є раціональним [14].

3.2.3 Вибір бібліотек для оформлення інтерфейсу

React підтримує інтерфейси користувача (UI) майже 10 мільйонів веб-сайтів по всьому світу. Базова бібліотека React доволі об'ємна, існує багато бібліотек компонентів, наповнених цінними елементами дизайну для програми React або проекту веб-розробки. Але для розширення можливостей розробники часто використовують додаткові бібліотеки. Найпопулярнішими бібліотеками для оформлення інтерфейсу є React Bootstrap, Grommet, MUI, Ant Design. Порівняння даних бібліотек зображено в табл. 2.3.

Таблиця 2.3 – Порівняння бібліотек для оформлення інтерфейсу

Інструменти	Критерії		
	Кастомізація компонентів	Простота та швидкість розробки	Різноманітність елементів
React Bootstrap	Можливостей для кастомізації не так багато, тому більшість додатків на Bootstrap дуже схожі	Висока швидкість розробки завдяки коду, який можна перевикористовувати	Не приділяє достатньо уваги динамічності елементів. Використовує мінімалізм у дизайні
MUI	Багато можливостей для кастомізації, можна розробити унікальний дизайн	Швидкість розробки не дуже висока, але можна використовувати шаблони для більшої простоти	Динамічні та привабливі елементи: анімація, різні повзунки, спливаючі вікна [15]
Grommet	Не призначений для широкого налаштування. Є багато властивостей елементів, які взагалі не можна змінити	Ця гнучка бібліотека дозволяє легко реалізовувати компоненти	Великий набір компонентів для візуалізації даних (від стовпчастих діаграм до map) [16]
Ant Design	Дуже проста кастомізація елементів	Розробка навіть кастомізованих компонентів доволі проста	Кількість елементів не дуже велика, але є багато можливостей для оформлення таблиць та форм [17]

Після проведеного аналізу було вирішено обрати 2 бібліотеки для розробки інтерфейсу. Однією з найкращих бібліотек є MUI (стара назва – Material UI). MUI – це проста і настроювана бібліотека компонентів React, заснована на Material Design від Google, побудована на 63,9% JavaScript та 36,1% TypeScript.

Вона використовується більш ніж 745000 проектів на GitHub, це не просто бібліотека компонентів, а ціла дизайн-система. MUI включає цілісну систему керівництв, принципів дизайну і кращих практик дизайну інтерфейсу користувача.

Оскільки система дизайну Material-UI, на якій засновано MUI, була створена Google, вона також використовується на власній платформі Google. Таким чином, компоненти MUI можуть мати чітко схожий на зовнішній вигляд Google.

Завдяки великому набору компонентів MUI можна легко створювати інтерфейс користувача в додатках React. Можна використовувати існуючі компоненти Material Design, налаштовувати їх, або можна створити власну систему дизайну. MUI має багато готових компонентів та детальну документацію, це дає можливість швидко створювати чисті та красиві програми.

Тому було вирішено для оформлення сторінок вебзастосунку використовувати MUI. А для оформлення сторінок адміністративної панелі було обрано бібліотеку Ant Design. Ant Design має дві важливі переваги, які вигідно виділяють його серед аналогічних бібліотек: таблиці та форми. Так як в адміністративній панелі майже кожна сторінка складається з таблиць, доречно використовувати для їх оформлення саме Ant Design [18].

3.2.4 Вибір засобу для обміну даними між клієнтом та сервером

Socket.IO — бібліотека JavaScript, яка заснована на веб-сокетах та інших технологіях. Вона використовує веб-сокети, коли вони доступні, або такі технології, як AJAX Long Polling, Flash Socket, AJAX Multipart Stream, коли веб-сокети недоступні. Аналогіями для порівняння можуть бути Fetch API та Axios.

Головні переваги Socket.IO:

— Socket.IO дозволяє надсилати повідомлення всім підключеним клієнтам, на відміну від веб-сокетів. Наприклад, якщо ви використовуєте Socket.IO для чату та хочете додати функцію повідомлення всіх користувачів про

підключення нового користувача, це можна реалізувати за допомогою однієї операції. При використанні веб-сокетів, для досягнення мети розробникам знадобиться список підключених клієнтів та надсилання повідомлень по одному;

- у веб-сокетах складно використовувати проксування та балансувальники навантаження. Socket.IO підтримує ці технології з коробки;

- Socket.IO підтримує автоматичне перепідключення під час розриву з'єднання [19].

Отже, як засіб для обміну даними між клієнтом та сервером було обрано Socket.IO.

3.2.5 Вибір середовища розробки

В якості середовища розробки обрано середовище WebStorm. WebStorm – середовище розробки для JavaScript, його можна використовувати як для frontend'a, так і для створення додатків на Node.js. Цей інструмент розроблений компанією JetBrains.

Його головною перевагою є зручний і розумний редактор JavaScript, HTML і CSS, який підтримує також інші мови і фреймворки.

WebStorm робить розробку проекту простою та зручною, забезпечує автодоповнення та підсвічування коду, аналізує код в процесі редагування, є швидка навігація та рефакторинг. Він має потужні інструменти налагодження та інтеграції із системами керування версіями, розуміє структуру проекту та код, відстежує помилки за допомогою систем ESLint, JSHint, JSLint, TSLint, Stylelint та пропонує їх рішення. WebStorm має вбудовані інструменти для тестування та роботи з проектом, які допомагають у розробці та роблять її зручнішою та продуктивнішою [20].

3.2.6 Вибір інструменту для тестування API

Для тестування API було обрано платформу Postman, потужний інструмент з простим і інтуїтивно зрозумілим інтерфейсом. За допомогою графічного інтерфейсу обраної платформи можна зберігати запити для легкого запуску в майбутньому, легко вводити інформації в правильному форматі, переглядати відповіді в форматі JSON або в необробленому вигляді.

Саме завдяки наявності графічного інтерфейсу набагато зручніше використовувати Postman, ніж консоль. В Postman зручно налаштовувати передані Cookie, HTTP заголовки, а також параметри форм, що дозволяє легко перевірити роботу будь-якого API.

3.2.7 Вибір системи управління базами даних

Існує багато систем управління базами даних (СУБД). СУБД – це комплекс програмно-язичних засобів, які дозволяють створити бази даних і керувати даними. Тобто СУБД — це набір програм, що дозволяють організувати, контролювати та адмініструвати бази даних. На даний момент існує багато видів СУБД (PostgreSQL, MySQL, MongoDB та інші), тому необхідно розібратись із перевагами та недоліками основних варіантів, щоб зробити правильний вибір – яку саме СУБД використати в програмі, яка реалізовується. Порівняння основних СУБД наведено в табл. 2.4.

Таблиця 2.4 – Порівняння основних СУБД

СУБД	Критерії			
	Структура	Масштабованість	Функціональність	Особливості
PostgreSQL	Реляційна	Чудова масштабованість. Вертикальна масштабованість є відмінною рисою PostgreSQL	Багатий функціонал. Транзакції, тригери, підзапити, зовнішні ключі, уявлення, обмеження цілісності, різні блокування та інше	Надійність, продуктивність, розширюваність, SQL підтримка, підтримка численних типів даних

Продовження таблиці 2.4

СУБД	Критерії			
	Структура	Масштабованість	Функціональність	Особливості
MySQL	Реляційна	Бувають проблеми з масштабованістю: можна масштабувати MySQL, але для цього знадобиться більше інженерних зусиль порівняно з будь-якою базою даних NoSQL.	Пропонує багато функцій, навіть у безкоштовній версії, але у безкоштовній версії найбільший упор робиться на швидкість і надійність	Дуже швидкий, багатопоточний, розрахований на багато користувачів і підтримує SQL
MongoDB	Нереляційна	Чудова масштабованість. Крім вертикального масштабування для розміщення більшої кількості даних MongoDB також може масштабуватися по горизонталі.	Широкий функціонал. Має досить гнучку мову для формування запитів, підтримує зберігання документів у JSON-подібному форматі, ефективно забезпечує зберігання великих бінарних об'єктів, може створювати індекси для різних атрибутів, що зберігаються.	Підходить для проєктів з різними даними, які важко піддаються класифікації [21]

На основі порівняння різних СУБД було вирішено використовувати в роботі PostgreSQL. Вона строго структурована, тим самим при розробці можна бути впевненими в тому, що збережені дані будуть коректними. Також мова запитів SQL переважно швидше працює, ніж в нереляційних базах даних.

Отже, PostgreSQL — це об'єктно-реляційна СУБД, найбільш розвинена з відкритих СУБД у світі. Має відкритий вихідний код і є альтернативною комерційною базою даних. Переваги PostgreSQL:

— об'єктно-реляційна модель: тобто PostgreSQL підтримує і об'єктний, і реляційний підхід. В реляційній моделі СУБД дані, які в них зберігаються, представлені у вигляді записів, що пов'язані один з одним відносинами. Так ми отримуємо різні списки, які можуть мати між собою ті чи інші відносини, таким чином і утворюється таблиця. В об'єктній моделі дані представлені у вигляді об'єктів, їх атрибутів, методів і класів;

— підтримка безлічі типів даних: не тільки стандартних цілочисленних значень, чисел з плаваючою точкою, строк та булевих значень, але й грошових, геометричних, бінарних та інших типів. PostgreSQL «з коробки» підтримує бітові строки та мережеві адреси, масиви даних, у тому числі багатомірні, композитні типи та інші складні структури. У ній є підтримка XML, JSON і NoSQL-бази. При необхідності до СУБД можна підключити підтримку типів даних, які потрібні в конкретному проекті. У PostgreSQL є кілька внутрішніх форматів, які використовуються тільки в ній;

— робота з великими об'ємами: в PostgreSQL немає обмежень за обсягом баз і кількістю записів у них, як у багатьох СУБД, які розраховані на середні та невеликі проекти. В PostgreSQL є обмеження тільки щодо конкретних записів. Одна таблиця може зайняти не більше 32 Тб, а один запис — 1,6 Тб. В одному полі запису може бути не більше 1 Гб даних, а максимальна кількість полів залежить від типу і становить від 250 до 1600 штук. Максимальних значень достатньо, щоб зберегти в БД будь-які дані;

— підтримка складних запитів: PostgreSQL справляється із завданнями розбору та виконання трудоміжких операцій, які включають в себе і читання, і запис, і валідацію одночасно. PostgreSQL повільніше аналогів тільки при читанні, але в інших аспектах переважає конкурентів;

— написання функцій на кількох мовах: PostgreSQL підтримує більше мов, ніж аналоги. Крім стандартного SQL, в PostgreSQL можна писати на C і C++, Java, Python, PHP, Lua, Ruby та інші рідкісні мови. А при необхідності можна розширити систему під інші мови програмування;

— одночасна модифікація бази, тобто є можливість одночасного доступу до бази з декількох пристроїв. У PostgreSQL для цього використовується технологія Multiversion Concurrency Control. Кожен користувач отримує "знімок" бази, в якій вносяться зміни. Тільки після фіксації транзакції вони надходять у вихідну базу даних. Поки людина вносить зміни, їх не видно іншим користувачам. Конфліктів немає, як і необхідності блокувати читання чи запис.

Для керування PostgreSQL було обрано крос-платформний інструмент DataGrip. DataGrip надає консоль запитів, навігацію за схемами, плани пояснень, інтелектуальне завершення коду, аналіз у реальному часі та швидкі виправлення, рефакторинг, інтеграцію керування версіями та інші функції [22].

3.3 Висновки за розділом 3

Для створення вебзастосунку для оптимізації роботи онлайн школи було обрано ефективний та сучасний набір інструментів. Фреймворком було обрано React.js, середовищем розробки WebStorm. Додаткові інструменти: Redux, react router, Typescript, antd, mui, socket-io. Також було описано обрану систему управління базами даних – PostgreSQL.

4 РОЗРОБКА ПРОГРАМИ

4.1 Опис структури файлів

Проект включає в себе клієнтську частину та адміністративну панель. Клієнтська частина складається з наступних частин:

Папка `public` містить в собі файл `index.html` – головний файл для відображення додатку, та інші. Папка `src` містить наступні папки: `components`, `configs`, `containers`, `pages`, `router`, `services`, `store`, `tests`.

Папка `components` містить наступні папки: `Header` (компонент шапки сторінки), `PreviewDocument` (відображає документи `.pdf`, `.docx` для читання), `Test` (компонент, який використовується для тестування студентів), `VideoPlayer` (компонент для відображення відеоматеріалів), в кожній із них є файли:

- `index.tsx` – в цьому файлі реалізується логіка відображення компоненту;

- `style.scss` – цей файл містить в собі стилі компоненту.

Папка `configs` містить в собі `index.ts`, в якому зберігаються конфігураційні значення (підключення до бази даних, API KEY для поштового сервісу та інше). Папка `containers` складається з папок `App` (головний контейнер сайту) та `ProfileTabs` (тут міститься реалізація вкладок в особистому профілі користувача), які містять в собі `index.tsx` та `style.scss`. Папка `pages` складається з папок `Home` (тут реалізовано лендінг), `Homeworks` (відображає домашні завдання), `Login` (сторінка авторизації), `Materials` (відображає список навчальних матеріалів для учнів), `Profile` (основна сторінка в особистому кабінеті), `TimeTable` (відображає розклад учнів), які містять файли `index.tsx` та `style.scss`.

Папка `router` містить в собі файли:

- `index.tsx` – зберігає в собі URL-шляхи до сторінок;
- `paths.ts` – константні значення шляхів, які зручно перевикористовувати; та інші.

Папка `services` містить в собі папки `Api` (сервіс для запитів до API) та `Sendgrid` (сервіс, який відповідає за відправку поштових листів), які містять в собі

index.tsx. Папка store складається з папки auth (зберігає в собі дані про авторизацію користувача для зручного перевикористання в інших компонентах) та файлу index.tsx (містить в собі store, actions, getters). Папка tests містить в собі папку Homeworks, в якій є файли answerTest.spec.ts та sendHomework.spec.ts (для перевірки всіх кейсів, які можуть отримати користувачі). Також в головній папці src є файли global.scss (глобальні стилі), index.tsx (головний компонент для відображення сайту), react-app-env.d.ts.

Також в проекті розміщені файли .env (для зберігання головних конфігурацій, наприклад, доступ до бази даних, API KEY до Sendgrid), .gitignore, .craco.config.js (конфігураційний файл для запуску проекту), package.json, README.md, tsconfig.json, yarn.lock.

А файли, пов'язані з адміністративною панеллю, описані нижче. Так як в них є багато спільного з клієнтською частиною, будуть описані лише основні.

Папка src містить в собі папку components, configs, containers (складається з папок App та ProfileTabs), pages (складається з папок Groups, Home, Homeworks, Login, Materials, Tests, Users; в кожному page реалізовано CRUD для маніпулювання даними), router (папки index.ts, paths.ts), services, store.

Папка components містить в собі папки Field, Header, Modal (складається з папок Document та Group), Material, User, Video lessons, в кожній з яких є файли index.tsx та style.scss та майже в усіх є папки create, delete, edit, які реалізують можливість створення, редагування, видалення відповідних даних. Також папка components містить в собі папки MultiSelect, PreviewDocument, Select, Tables (складається з папок Groups, Homeworks, Materials, Tests, Users).

4.2 Основні рішення з реалізації компонентів системи

4.2.1 Основні рішення щодо організації даних

Для маніпулювання даними використовується сторонній API, в якому реалізовані всі необхідні ендпоінти для роботи з даними. Ендпоінт (Endpoint - кінцева точка) - це саме звернення до маршруту окремим HTTP методом.

Ендпоінти виконують конкретне завдання, приймають параметри і повертають дані клієнту.

Для маніпуляції даними виконувались наступні методи HyperText Transfer Protocol (HTTP) запиту:

- GET – запитує уявлення ресурсу. При використанні цього методу можна тільки отримувати дані. Цей метод не має тіла;
- POST – відправляє сутності до певного ресурсу. Цей метод має тіло;
- PUT – використовується для заміни всіх поточних уявлень ресурсу даними запиту;
- DELETE – видаляє вказаний ресурс;
- PATCH – використовується для часткової зміни ресурсу.

Більш детальна інформація про запити до API описана в таблиці 4.1.

Таблиця 4.1 — Запити до API

Тип запиту	Шлях запиту	Опис запиту – що він приймає	Опис відповіді – що повертається
post	/admin/auth/login	Дані для авторизації	Access token
get	/admin/users	Дані для пошуку та фільтрації	Список користувачів
get	/admin/user/:userId	Ідентифікатор користувача	Дані користувача
post	/admin/user	Дані для створення користувача	Дані створеного користувача
put	/admin/user/:userId	Ідентифікатор користувача та дані, які треба змінити	Повідомлення про успішне оновлення користувача
delete	/admin/user/:userId	Ідентифікатор користувача	Повідомлення про успішне видалення користувача
get	/admin/groups	Дані для пошуку та фільтрації	Список груп
get	/admin/group/:groupId	Ідентифікатор групи	Дані групи
post	/admin/group	Дані для створення групи	Дані створеної групи

Продовження таблиці 4.1

Тип запити	Шлях запити	Опис запити – що він приймає	Опис відповіді – що повертається
put	/admin/group/:groupId	Ідентифікатор групи та дані, які треба змінити	Повідомлення про успішне оновлення групи
delete	/admin/group/:groupId	Ідентифікатор групи	Повідомлення про успішне видалення групи
get	/admin/materials	Дані для пошуку матеріалів	Список матеріалів
get	/admin/material/:materialId	Ідентифікатор матеріалу	Дані матеріалу
post	/admin/material	Дані для створення матеріалу	Дані створеного матеріалу
put	/admin/material/:materialId	Ідентифікатор матеріалу та дані, які треба змінити	Повідомлення про успішне оновлення матеріалу
delete	/admin/material/:materialId	Ідентифікатор матеріалу	Повідомлення про успішне видалення матеріалу
get	/admin/timetable	Діапазон часу	Дані розкладу за обраний діапазон часу
post	/admin/timetable	Дані для створення розкладу	Дані про створений розклад
delete	/admin/timetable/:timetableId	Ідентифікатор розкладу	Повідомлення про успішне видалення розкладу
get	/admin/video-lessons	Дані для пошуку для фільтрації відеоуроків	Список відеоуроків
get	/admin/video-lesson/:videoLessonId	Ідентифікатор відеоуроку	Дані про відеоурок
post	/admin/video-lesson	Дані для завантаження відеоуроку	Дані про завантажений відеоурок

Продовження таблиці 4.1

Тип запити	Шлях запити	Опис запити – що він приймає	Опис відповіді – що повертається
put	/admin/video-lesson/:videoLessonId	Ідентифікатор відеоуроку та дані, які треба змінити	Повідомлення про успішне оновлення відеоуроку
delete	/admin/video-lesson/:videoLessonId	Ідентифікатор відеоуроку	Повідомлення про успішне видалення відеоуроку
get	/admin/homework/tests	Дані для пошуку для фільтрації тестів	Список тестів
get	/admin/homework/test/:testId	Ідентифікатор тесту	Дані про тест
post	/admin/homework/test	Дані для створення тесту	Дані про створений тест
put	/admin/homework/test/:testId	Ідентифікатор та дані, які треба змінити	Повідомлення про успішне оновлення тесту
delete	/admin/homework/test/:testId	Ідентифікатор тесту	Повідомлення про успішне видалення тесту
get	/admin/homework	Дані для пошуку та фільтрації домашніх завдань	Список домашніх завдань
get	/admin/homework/:homeworkId	Ідентифікатор домашнього завдання	Дані про домашнє завдання
post	/admin/homework	Дані для створення домашнього завдання	Дані про створене домашнє завдання
put	/admin/homework/:homeworkId	Ідентифікатор та дані, які треба змінити	Повідомлення про успішне оновлення домашнього завдання
delete	/admin/homework/:homeworkId	Ідентифікатор домашнього завдання	Повідомлення про успішне видалення домашнього завдання

Продовження таблиці 4.1

Тип запити	Шлях запити	Опис запити – що він приймає	Опис відповіді – що повертається
post	/auth/login	Дані для авторизації користувача	Access token
get	/me	(тільки Access token)	Дані про користувача
get	/homeworks	Дані для пошуку та фільтрації домашніх завдань	Список домашніх завдань користувача
get	/homework/:homeworkId	Ідентифікатор домашнього завдання	Дані про домашнє завдання
post	/homework/:homeworkId/answer	Ідентифікатор домашнього завдання та виконане домашнє завдання	Повідомлення про успішну відправку домашнього завдання
get	/test/:testId	Ідентифікатор тесту	Дані про тест
post	/test/:testId/answer	Ідентифікатор тесту та відповідь на тест	Повідомлення про успішну відправку
get	/materials	Дані для пошуку та фільтрації матеріалів	Список матеріалів
get	/material/:materialId	Ідентифікатор матеріалу	Дані про матеріал
get	/video-lessons	Дані для пошуку та фільтрації відеоуроків	Список відеоуроків
get	/video-lesson/:videoLessonId	Ідентифікатор відеоуроку	Дані про відеоурок
get	/timetable	Діапазон часу	Дані розкладу

4.2.2 Введення та виведення даних

Введення даних виконується за допомогою загальних елементів інтерфейсу, комп'ютерної миші та клавіатури. Звісно, розробленим сайтом можна користуватись через будь-який пристрій з доступом до Інтернету. Виведення даних виконується за допомогою монітору комп'ютера/екрана телефона.

4.2.3 Користувацький інтерфейс GUI

Архітектура користувацького інтерфейсу є компонентною. Компоненти React - це блоки коду, які підходять для багаторазового використання. Кожний компонент оформлюється завдяки ant design та mui. Для зрозумілого відображення програми використовуються загальні елементи інтерфейсу, наприклад: Form inputs and controls (Form, Checkbox, Textarea, File input, , Input, Slider, Text field, Radio button), Tooltip, Table, Dialog, Footer та інші.

4.2.4 Захищення даних

Для захищення даних користувач матиме свій логін і пароль. Пароль має бути не менше 6 символів та містити як цифри, так і літери. Усі паролі зберігаються у базі даних, до якої має доступ тільки адміністратор системи.

Неzareєстрований користувач має обмежений доступ до системи.

4.2.5 Адаптивність до змін та оновлень додатку

Оскільки програма WebStorm дуже гнучка, то доповнювати код дуже легко, та цей процес ніяк не відобразиться на роботоздатності старого коду. Програміст може легко додавати необхідні пакети для оптимізації процесу розробки. Це дозволяє максимально швидко та якісно розробляти програмний продукт.

4.3 Опис допоміжних бібліотек, модулів та функцій

Окрім вже перелічених основних фреймворків та бібліотек, для розробки були використані додаткові інструменти, такі як Axios, dotenv, lodash, sass, validator, pdfjs-dist, jest та інше.

Для спрощення роботи з запитами до API краще використовувати Axios – бібліотеку, яка дозволяє робити запити на REST API. Axios – це HTTP-клієнт, заснований на Promise для node.js та браузера. Він ізоморфний (може працювати у браузері та node.js з тією ж базою кодів). На стороні сервера Axios використовує нативний node.js http-модуль, а на стороні клієнта (браузер) він використовує XMLHttpRequests. Цей HTTP – клієнт заснований на промісах, завдяки цьому можна використовувати можливості JavaScript async і await для отримання зручнішого асинхронного коду. За допомогою Axios можна перехоплювати та скасовувати запити, також у клієнта є вбудований захист від підробки запитів між сайтами. Використання Axios дозволяє уникнути написання великих обсягів шаблонного коду і зробити код чистішим і зрозумілішим [23].

Наведемо приклад коду з проекту, де ми ініціалізуємо об'єкт Axios, який приймає URL до бекенду, також створюємо хук для того, щоб додати access token до запиту:

```
import axios from 'axios'
const instance = axios.create({
  baseURL: BACKEND.URL
})
instance.interceptors.request.use(
  (config) => {
    if (!config.headers) {
      config.headers = {}
    }
    config.headers.authorization = `Bearer ${localStorage.getItem('token')}`
    return config
  },
)
```

Наступним компонентом, який було використано для оптимізації коду, є Dotenv. Пакет Dotenv аналізує файли .env, щоб змінні оточення, що зберігаються в них, були доступні через getenv(), \$_ENV або \$_SERVER. Тобто Dotenv надає доступ до прихованого файлу .env, в якому зберігаються важливі дані.

Наведемо приклад коду з проекту, де ми підключаємо в головний компонент бібліотеку dotenv, тим самим маємо доступ до конфігураційних даних у всьому проекті:

```
import React from 'react';
...
import 'dotenv/config'
...
root.render(
  <Provider store={store}>
```

```

    <RouterProvider router={router} />
  </Provider>
);

```

Для оптимізації коду також було використано **Lodash**, бібліотеку, яка надає готові рішення для роботи з об'єктами, масивами, колекціями і т.п. **Lodash** застосовується в основному для функціонального програмування, успадковує основні ідеї **Underscore.js** і включає наступні частини:

- роботу з масивами, наприклад, методи видалення елементів за певним правилом, отримання певного елемента в масиві, створення копії без дублікатів та багато іншого;
- для роботи з об'єктами бібліотека має методи перевірки значення об'єктів на належність тому чи іншому типу та функції клонування об'єктів;
- крім колекцій, які являють собою суперсети масивів, є ще секвенції, з яких зручно вибудовувати послідовні обробники колекцій чи списків;
- для роботи з функціями бібліотека має багато методів для роботи з контекстом, часом виконанням/виклику функції, наявна можливість визначення композиції функцій;
- є функції роботи з рядками - додавання великої літери, видалення зайвих прогалін та інше [24].

Наведемо приклад коду з проекту, де ми використовуємо **Lodash** для перевірки на наявність даних, інакше показуємо повідомлення, що дані не знайдені:

```

import React, { useEffect, useState } from 'react'
import isEmpty from 'lodash/isEmpty'
...
const Materials = () => {
  ...
  return <ProfileTabs>
    <Box sx={{
      display: 'flex',
      flexWrap: 'wrap',
    }}>
      {
        isEmpty(materials)
        ? (<Empty />)
        : {materials.map(item => <PreviewDocument key={item.id} title={item.title}
file={item.url} />)}
      }
    </Box>
  </ProfileTabs>
}
export default Materials

```

Sass – це метамова, яка спрощує та прискорює написання CSS-коду. Метамова – це мова для опису іншої мови. Sass вважається препроцесором CSS, бо Sass має свій синтаксис, з якого генерує CSS-код, зрозумілий будь-якому браузеру.

Препроцесор Sass допомагає зробити CSS-код простішим і зрозумілішим. Його легше масштабувати, оновлювати та підтримувати. Використання Sass розширює функціональність, бо він дозволяє використовувати CSS-константи, вбудовані функції, вкладені правила, домішки (змішані стилі), успадкування тощо. Sass допомагає уникнути багаторазового повторення однакових фрагментів коду. Це заощаджує час розробника, зменшує обсяг файлів стилів та прискорює обробку сторінок [25].

Отже, при розробці вебзастосунку препроцесор Sass допоміг писати стилі з використанням змінних та міксінів.

Наведемо приклад коду з проекту, де ми ініціалізуємо міксін для дефолтного скролу для подальшого перевикористання в інших компонентах:

```
@mixin default-scrollbar() {
  &::-webkit-scrollbar {
    background-color: transparent;
  }
  &::-webkit-scrollbar-thumb {
    border-radius: 6px;
    background-color: transparent;
  }
  &::-webkit-scrollbar {
    width: $scrollbarWidth;
    height: 8px;
  }
  &:hover {
    &::-webkit-scrollbar {
      background-color: $scrollbar;
    }
    &::-webkit-scrollbar-thumb {
      background-color: $thumb;
    }
  }
}
```

Практично кожен веб-сайт включає збір введення користувача за допомогою HTML-форм. Щоразу, коли ми збираємо дані користувача, ми повинні їх перевіряти. У комп'ютерних науках перевірка даних - це процес перевірки того, чи є введені дані розумними та обґрунтованими. З метою валідації даних було

обрано сучасний пакет `validator.js`, який має безліч попередньо встановлених методів для перевірки дійсності різних входних даних [26].

Наведемо приклад коду з проекту, де ми використовуємо `Validator` для валідації введених кристувачем даних:

```
...
import validator from 'validator'
...
const checkValid = (value: string) => {
  if (!validation) {
    return ''
  }
  const { require, email, maxLength, minLength } = validation
  if (require && validator.isEmpty(value)) {
    return 'The field is required'
  }
  if (email && !validator.isEmail(value)) {
    return 'Email is incorrect'
  }
  if (maxLength && !validator.isLength(value, { max: maxLength })) {
    return `Length must not exceed ${maxLength} characters`
  }
  if (minLength && !validator.isLength(value, { min: minLength })) {
    return `Length must not be less than ${minLength} characters`
  }
  return ''
}
...
```

`Pdfjs-dist` – це бібліотека, яка надає можливість працювати з `.pdf` файлами. Так як в вебзастосунку для онлайн школи є багато навчальних матеріалів в різних форматах, необхідно зробити роботу з кожним типом комфортною, з форматом `.pdf` корисним виявився саме пакет `pdfjs-dist`.

Наведемо приклад коду з проекту, де реалізовано коректне відображення документів для читання.

```
...
import { Document, Page } from "react-pdf/dist/esm/entry.webpack";
<Document file={file}>
  <Page className='page' pageNumber={1} height={height} />
</Document>
```

Ефективна стратегія автоматизованого тестування коду є надзвичайно важливою для забезпечення швидкої та якісної роботи вебзастосунків. Для цього в проекті було використано `Jest`.

`Jest` - це середовище тестування JavaScript, яке дозволяє розробникам запускати тести на код JavaScript і TypeScript і добре інтегрується з React. `Jest` використовує бібліотеку `jsdom` і може емулювати всі дії з DOM в `nodejs`. Вся логіка з перезапуску, збору статистики додана до самого фреймворку. У

результаті вся взаємодія з тестами відбувається в консолі, браузер не відкривається-закривається. Великою перевагою Jest є багатопоточність, тобто можна виконувати тести паралельно, а отже, швидше. Jest може запускати паралельно кожен файл з тестами. Кількість потоків за умовчанням дорівнює кількості ядер машини.

Наведемо приклад коду з проекту, де ми перевіряємо кейси, які може отримати користувач при відправленні відповідей на тест:

```
describe('Tests/answerTests', () => {
  test('should return success if tests answers is correct', () => {
    const payload = {
      '1': 'a',
      '2': 'c',
      '3': 'b',
    }
    const result = answerTests(payload)
    expect(result).toBe(true)
  })
  test('should return fail if tests answers is correct', () => {
    const payload = {
      '1': 'a',
      '2': 'a',
      '3': 'a',
    }
    const result = answerTests(payload)
    expect(result).toBe(false)
  })
})
```

4.4 Висновки за розділом 4

Було описано структуру файлів, допоміжні бібліотеки, модулі та функції, які сприятимуть стабільній роботі вебзастосунку. Також було прийнято основні рішення з реалізації компонентів системи.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Призначення й умови застосування програми

Розроблена система призначена для забезпечення гнучкого та простого механізму викладання та навчання в онлайн школі.

За допомогою розробленої адміністративної панелі можна проводити маніпуляції (створення, видалення та редагування) з даними студентів, навчальних курсів, матеріалів курсів.

Вебзастосунок надає користувачу наступні можливості:

- доступ до лендінгу онлайн школи;
- авторизація (реєстрація відбувається зі сторони адміністратора);
- відновлення пароля;
- доступ до розділу «Профіль» з особистими даними в особистому кабінеті;
- доступ до розділу «Домашні завдання», в якому можна отримати інформацію від вчителя в різноманітному вигляді (текстовий формат, відео, малюнки, файли .docx, .pdf) та виконувати завдання (проходити тести, а після цього отримувати пояснення щодо логіки проходження тестів з усіма деталями), в яких можуть бути обмеження в часі, в перемішеному порядку;
 - завантажувати виконане домашнє завдання зі свого пристрою в розділ «Домашні завдання»;
 - доступ до розділу «Відеоуроки», в якому можна переглядати пройдених на заняттях матеріал;
 - доступ до розділу «Матеріали», в якому можна легко знайти основні матеріали для навчання (підручники, файли з граматиною в різних форматах та інше);
 - доступ до розділу «Розклад», в якому розміщено новини про перенесення уроків, сам розклад, тощо;
 - пошук за назвою в кожному розділі.

Для користування розробленим веб-сайтом учню треба лише мати доступ до інтернету та мати зареєстрований адміністратором акаунт. Щоб запустити веб-сайт, потрібно просто перейти за посиланням на сайт. Користувачеві одразу відкриється головна сторінка сайту.

А для програміста до складу технічних засобів повинна входити персональна електронна обчислювальна машина (ПЕОМ), що виконує роль сервера, що включає в себе:

- процесор Intel Core i5 3.4Hz, не менше;
- оперативну пам'ять об'ємом 16 Гігабайт, не менше;
- HDD 100 Гігабайт, не менше;
- операційну систему Ubuntu 20.04;
- Node 14.x;
- Npm 6.14.x;
- платформу для розгортання проекту Docker 19.3.x;
- базу даних PostgreSQL 14.x.

5.2 Тестування та експериментальне дослідження програми

Для запуску програми користувачу необхідно перейти на сайт за посиланням. При переході на веб-сайт відкривається головна сторінка, яка містить базову інформацію про навчання в онлайн школі. Для авторизації користувач повинен в хедері перейти в розділ «Log in».

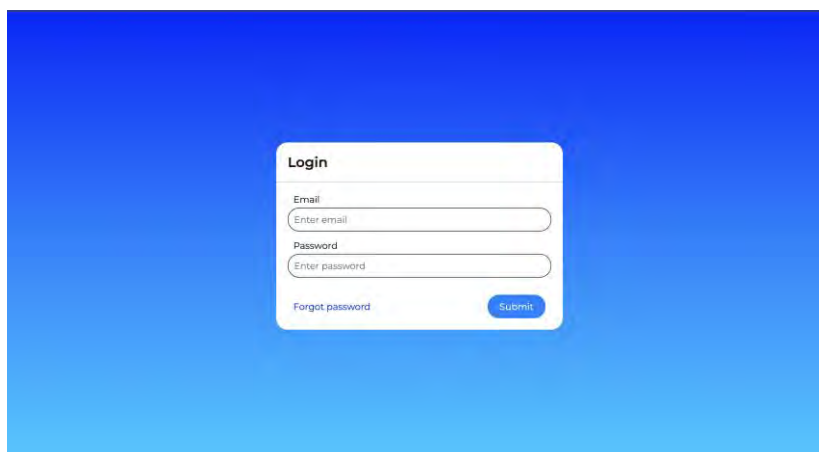


Рисунок 5.1 – Авторизація

Якщо учень забув пароль, він може відновити доступ до акаунту, для цього потрібно натиснути на «Forgot password» в розділі авторизації, а потім ввести email (рис. 5.2), куди прийде повідомлення для відновлення пароля.

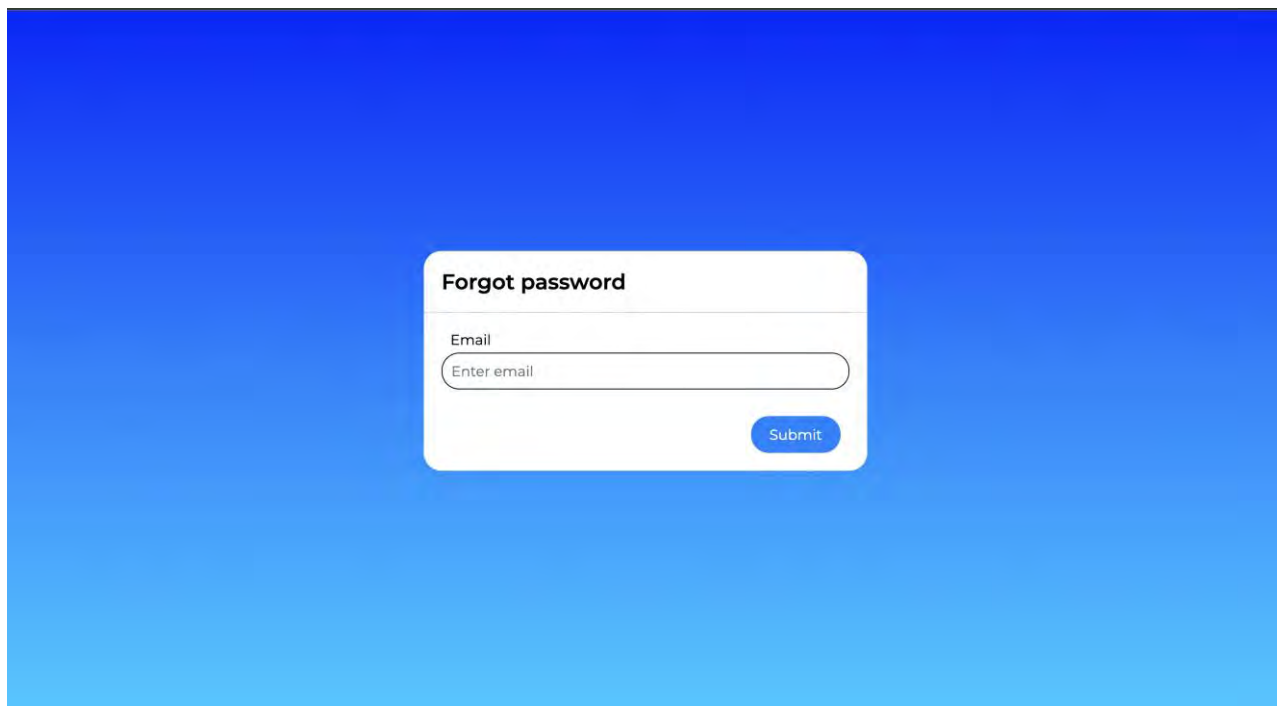


Рисунок 5.2 – Відновлення пароля

В особистому кабінеті учень може користуватись розділами «Профіль», «Домашні завдання», «Матеріали», «Відеоуроки», «Розклад». В розділі з домашніми завданнями учень обирає дату уроку, домашнє завдання до якого хоче подивитись, та потрапляє на сторінку, де розміщено домашнє завдання в текстовому форматі з поясненням, відеозапис уроку (який прикріплюється з розділу «Відеоуроки»), прикріплені файли, які потрібно використати для виконання домашнього завдання, онлайн тест, який відкривається при натисненні та який теж потрібно пройти в рамках домашнього завдання (рис. 5.3). Також на цій сторінці розміщена секція, за в яку учень повинен завантажити виконане домашнє завдання.

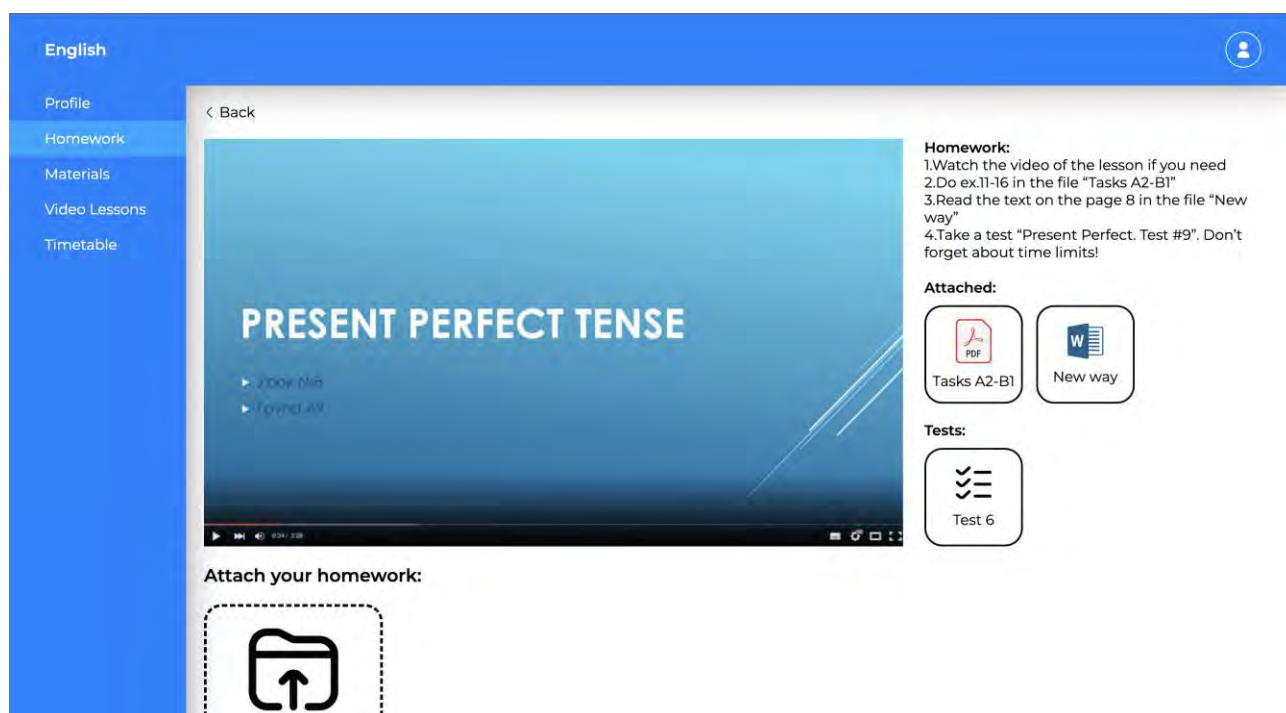


Рисунок 5.3 – Сторінка з домашнім завданням

Після проходження тестування в рамках домашнього завдання учень може самостійно зробити роботу над помилками завдяки детальним поясненням до кожного питання в тесті (рис. 5.4). Також кожне пояснення містить посилання до файлів з важливою для тесту інформацією (наприклад, граматика), які знаходяться в розділі «Матеріали».

English

Profile

Homework

Materials

Video Lessons

Timetable

Test 6

Lost time: 40 min 53 sec of 45 min

Score: 18 of 30

☒ Show all descriptions

1. In an open _____ over breakfast, Microsoft executives Shane Kim and Chris Satchell revealed that two of the home video game projects which The Lord of the Rings director Peter Jackson was working on with Bungie, an American video game developer, would take the form of episodic series.

Фрагмент речення з пропуском означає: У відкритому обговоренні під час сніданку, керівники Microsoft Шейн Кім і Кріс Сетчелл з'ясували... Тобто в речення потрібно вставити іменник. Тому з-поміж поданих варіантів відповідей discuss, discussion, discussing, discussed граматично підходить лише discussion. Для повного розуміння різниці між варіантами відповідей рекомендується повторити правила **'Частини мови'**, **'Форми дієслова'**, **'Герундій'** та **'Past Participle'** в розділі 'Матеріали'.

☒ Show descriptions

☐ discuss

Цей варіант не підходить, бо це не іменник, а дієслово. Приклад використання: They discuss it every morning. Повторити **'Частини мови'**

☒ discussion

Цей варіант правильний, discussion - іменник. Повторити **'Частини мови'**

☐ discussing

Цей варіант не підходить, бо це не іменник, а форма дієслова - герундій. Приклад використання: We enjoy discussing our future. Повторити **'Герундій'**

Рисунок 5.4 – Сторінка з результатами тесту

В розділі «Розклад» користувач може знайти свій актуальний розклад занять (рис. 5.5) в залежності від кількості курсів, які він проходить.

English

Profile

Homework

Materials

Video Lessons

Timetable

	Su	Mo	Tu	We	Th	Fr	Sa
8:00							
9:00							
10:00							
11:00							
12:00							
13:00							
14:00							
15:00							
16:00				Lesson			
17:00							
18:00		Lesson				Lesson	
19:00							
20:00							
21:00							
22:00							

Рисунок 5.5 – Сторінка «Розклад»

Зручний та простий інтерфейс має не тільки клієнтська частина, а і адміністративна панель. На рис. 5.6 зображено сторінку з користувачами системи в адміністративній панелі, а на рис. 5.7 зображено результат фільтрації учнів за групами.

Admin Panel Log out

Users Groups Materials Timetable Homework Video Lessons

Q Search Create User

Nº	Full Name	Email	Group	Phone	Date	Actions
1	Tammy Wisoky	Torrance71@gmail.com	A1	+38(099)-006-43-53	Sun Jul 17 2022	  
2	Vanessa Trantow	August39@gmail.com	B2	+38(099)-613-53-26	Sat Sep 17 2022	  
3	Kent Considine	Nolan_Goyette86@hotmail.com	C2	+38(099)-180-86-66	Mon Sep 19 2022	  
4	Claude Koch Sr.	Darrell_Lemkel6@yahoo.com	B2	+38(099)-599-16-03	Fri Mar 04 2022	  
5	Erica Buckridge	Freddie.Hand@hotmail.com	A1	+38(099)-620-33-82	Thu Aug 25 2022	  
6	Sylvia Thiel	Missouri.Gulgowski89@hotmail.com	B2	+38(099)-314-63-11	Wed Feb 16 2022	  
7	Alberta Schulist	Earline.Ziemann51@gmail.com	C2	+38(099)-320-53-58	Thu Aug 18 2022	  
8	Javier Kling	Deanna_Turcotte93@yahoo.com	A2	+38(099)-611-84-66	Mon Jun 27 2022	  
9	Krista Walker	Kellie72@hotmail.com	B1	+38(099)-497-53-30	Thu Dec 23 2021	  
10	Stephanie Homenick	Herminio_Wuckert68@yahoo.com	A2	+38(099)-925-79-51	Wed Oct 19 2022	  

< > 1-10 from 44 users Show on page 10 rows

Рисунок 5.6 – Сторінка з користувачами в адміністративній панелі

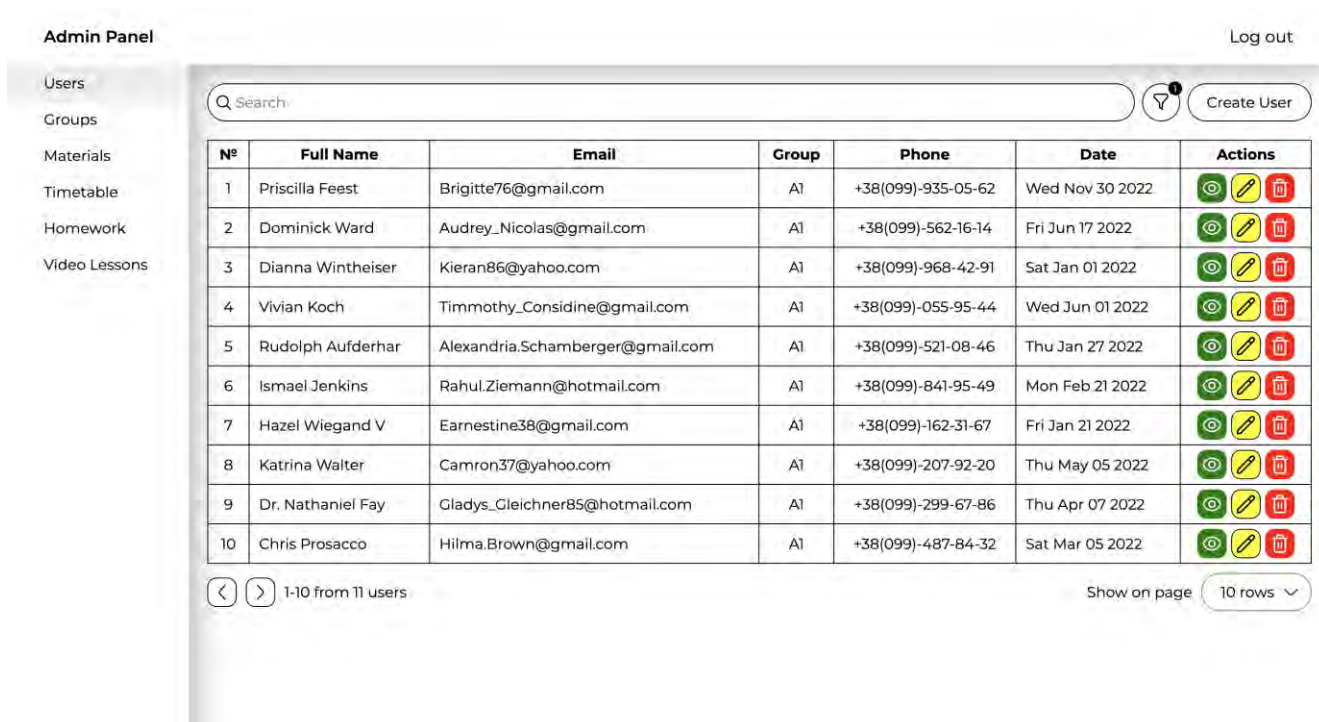


Рисунок 5.7 – Результат фільтрації

Для створення нового користувача адміністратор має натиснути на «Create user», а потім ввести дані в модальному вікні (рис. 5.8). На рис. 5.9 зображено розділ з відеоуроками та модальне вікно, через яке проходить завантаження нового відеоуроку до системи.

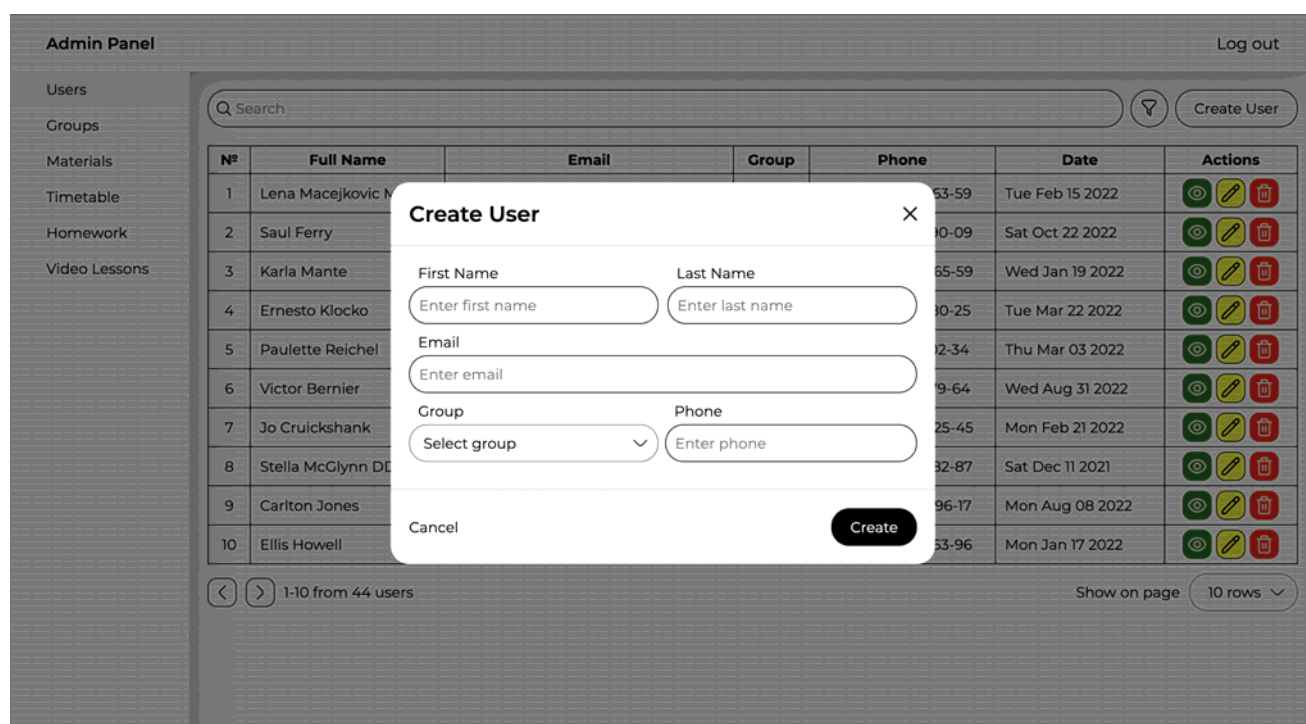


Рисунок 5.8 – Сторінка створення користувача

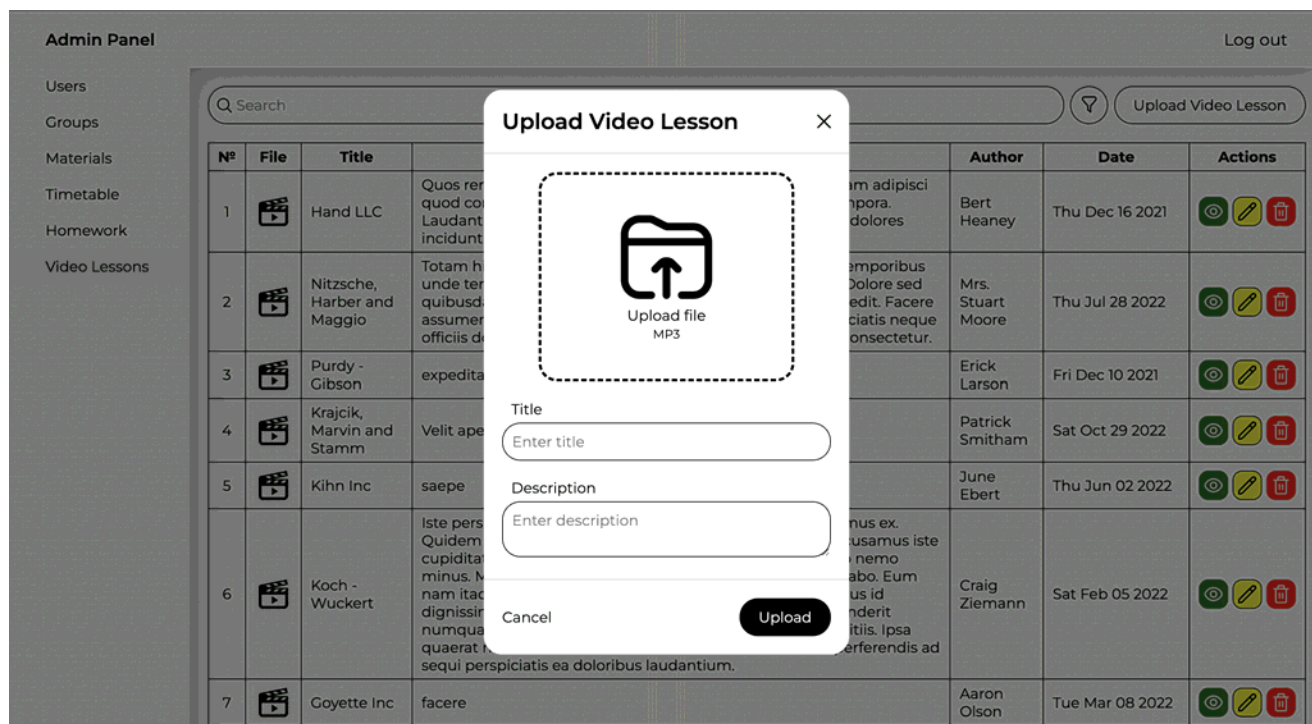


Рисунок 5.9 – Завантаження відеоуроку

На рис. 5.10 зображено розділ з навчальними матеріалами та модальне вікно, через яке відбувається завантаження нових матеріалів до бази. На рис. 5.11 зображено сторінку створення домашнього завдання, де можна легко прикріпити відеоуроки/матеріали/тести, які вже були створені в відповідних розділах, або завантажити нові файли із відеоуроками/матеріалами, а в текстовому полі можна ввести інструкції до домашнього завдання.

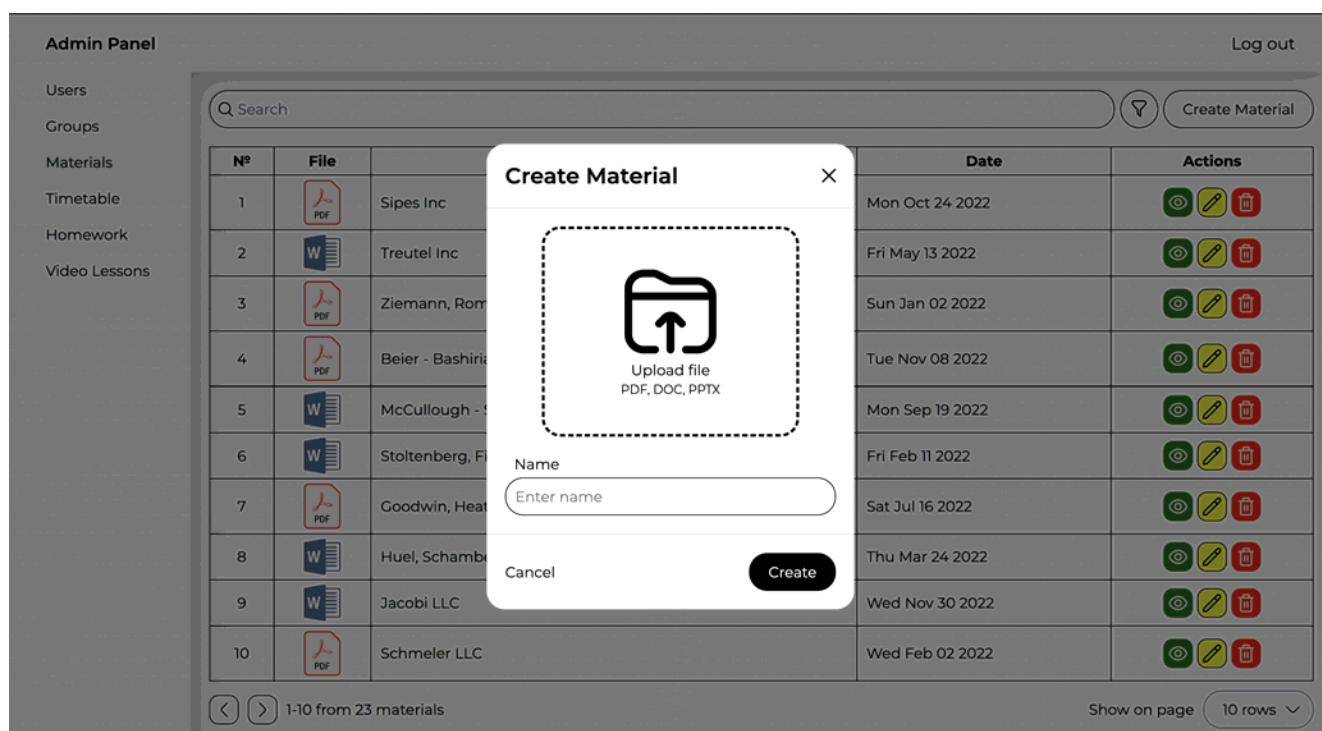


Рисунок 5.10 – Завантаження навчальних матеріалів

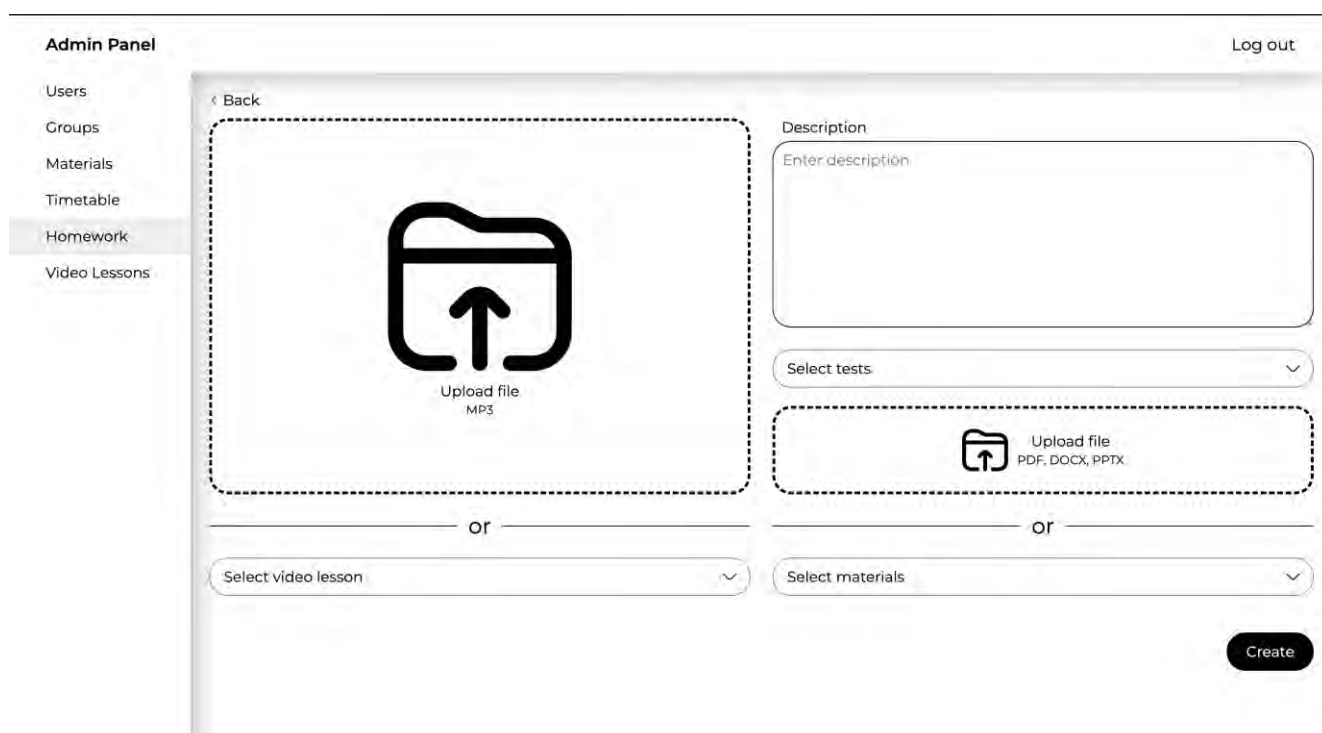


Рисунок 5.11 – Додання нового домашнього завдання

Отже, тестування програмного забезпечення для онлайн школи було здійснено на різних операційних системах з використанням різних веббраузерів. Результати тестування програмного забезпечення для онлайн навчання та

викладання при роботі під керуванням різних операційних систем, з використанням різних веббраузерів та на різних пристроях показали, що програма функціонує коректно та відповідає вимогам технічного завдання. Але, звісно, функціонального тестування недостатньо, щоб впевнитись в коректній роботі розробленої програми. Тому також було проведено модульне та інтеграційне тестування ПЗ. В якості інструменту для тестування було використано Jest.

Модульне тестування (юніт тестування, Unit Testing) – тип тестування ПЗ, при якому тестуються окремі модулі або компоненти ПЗ. Мета юніт тестування – впевнитись, що кожна одиниця програмного коду працює коректно. Під одиницею мається на увазі окрема функція, метод, процедура, модуль, або об'єкт. Наведемо принцип реалізації модульного тесту для кейсів відображення, наприклад, компоненту Field, тобто поля для введення даних, який перевикористовується в різних компонентах. Були розглянуті різні кейси, наприклад, відображення компоненту, якщо були введені невалідні значення. Завдяки створеним тестам можна бути впевненими, що наступні зміни в коді не вплинуть на роботу компоненту. Отже, розроблені модульні тести працюють наступним чином: за допомогою бібліотеки 'react-test-renderer' ми емуємо рендер компоненту (`const component = create()`), потім парсимо компонент (`let tree = component.toTree()`), перетворюючи його в сутність JSON, та автоматично створюється snapshot компоненту (`expect(tree).toMatchSnapshot()`):

```
import { create } from 'react-test-renderer'
import Field from '../components/Field'
describe('Components/Field', () => {
  test('should render simple component', () => {
    const component = create(
      <Field
        title="Email"
        placeholder="Enter email"
        value=""
        onChange={() => {}}
      />
    );
    let tree = component.toTree();
    expect(tree).toMatchSnapshot();
  })
  test('should render component with valid validation', () => {
    const component = create(
      <Field
        title="Email"
        placeholder="Enter email"
        value='test@example.com'
      />
    );
    let tree = component.toTree();
    expect(tree).toMatchSnapshot();
  })
})
```

```

        onChange={() => {}}
        validation={{
          require: true,
          email: true,
        }}
      />
    )
    let tree = component.toTree()
    expect(tree).toMatchSnapshot();
  })
  test('should render component with invalid validation (incorrect email)', async
  () => {
    const component = create(
      <Field
        title="Email"
        placeholder="Enter email"
        value='not valid value'
        onChange={() => {}}
        validation={{
          email: true,
        }}
      />
    )
    let tree = component.toTree()
    expect(tree).toMatchSnapshot();
  })
  test('should render component with invalid validation (required value)', async
  () => {
    const component = create(
      <Field
        title="Email"
        placeholder="Enter email"
        value=''
        onChange={() => {}}
        validation={{
          require: true,
        }}
      />
    )
    let tree = component.toTree()
    expect(tree).toMatchSnapshot();
  })
  test('should render component with invalid validation (min length)', async ()
=> {
    const component = create(
      <Field
        title="Email"
        placeholder="Enter email"
        value='123'
        onChange={() => {}}
        validation={{
          minLength: 4,
        }}
      />
    )
    let tree = component.toTree()
    expect(tree).toMatchSnapshot();
  })
  test('should render component with invalid validation (max length)', async ()
=> {
    const component = create(
      <Field
        title="Email"
        placeholder="Enter email"

```

```

    value='123456'
    onChange={() => {}}
    validation={{
      maxLength: 5,
    }}
  />
)
let tree = component.toTree()
expect(tree).toMatchSnapshot();
})
})

```

На рис. 5.12 зображено частину снєпшоту, який являє собою компонент у вигляді json розмітки. Цей снєпшот було згенеровано бібліотекою Jest.

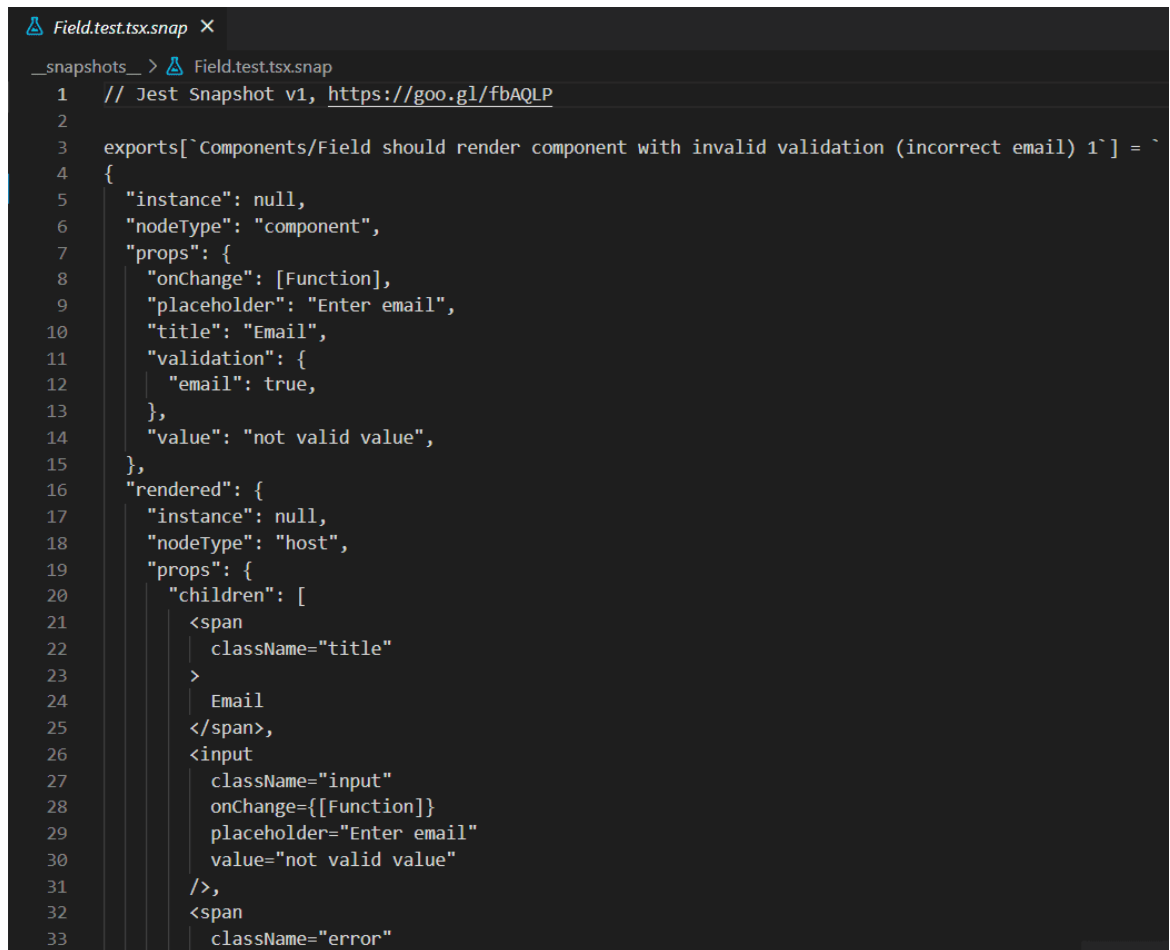


Рисунок 5.12 – Снєпшот модульного тесту

Результати тестування, яке було пройдено успішно, зображено на рис. 5.13, помилок при модульному тестуванні компоненту Field не було виявлено.

```

PASS src/test/Components/Field.test.tsx
Components/Field
  ✓ should render simple component (6 ms)
  ✓ should render component with valid validation (1 ms)
  ✓ should render component with invalid validation (incorrect email)
  ✓ should render component with invalid validation (required value)
  ✓ should render component with invalid validation (min length) (1 ms)
  ✓ should render component with invalid validation (max length)

  > 6 snapshots written.
Snapshot Summary
  > 6 snapshots written from 1 test suite.

Test Suites: 1 passed, 1 total
Tests:       6 passed, 6 total
Snapshots:   6 written, 6 total
Time:        0.46 s, estimated 1 s
Ran all test suites.

```

Рисунок 5.13 – Результати модульного тестування

Далі розглянемо один із розроблених інтеграційних тестів. При інтеграційному тестуванні програмні модулі поєднуються логічно та тестуються групою. Наприклад, у випадку зі сторінкою авторизації поєднуються такі модулі, як поля для введення логіна та пароля. Реалізація інтеграційного тесту для перевірки процесу роботи авторизації:

```

import { create, act } from 'react-test-renderer'
import Login from '../pages/Login'
describe('Pages/Login', () => {
  test('should render login page', () => {
    const page = create(<Login />)
    const tree = page.toTree()
    expect(tree).toMatchSnapshot()
  })
  test('should render login page with valid values', () => {
    const page = create(<Login />)
    const instance = page.root
    act(() => {
      const inputs = instance.findAllByType('input')
      inputs[0].props.onChange({ target: { value: 'test@example.com' } })
      inputs[1].props.onChange({ target: { value: 'secret' } })
    })
    const tree = page.toTree()
    expect(tree).toMatchSnapshot()
  })
  test('should render login page with incorrect email value', () => {
    const page = create(<Login />)
    const instance = page.root
    act(() => {
      const inputs = instance.findAllByType('input')

```

```

    inputs[0].props.onChange({ target: { value: 'not valid email' } })
  })
  const tree = page.toTree()
  expect(tree).toMatchSnapshot();
})
test('should render login page with incorrect password value', () => {
  const page = create(<Login />)
  const instance = page.root
  act(() => {
    const inputs = instance.findAllByType('input')
    inputs[1].props.onChange({ target: { value: '12' } })
  })
  const tree = page.toTree()
  expect(tree).toMatchSnapshot();
})
})
})

```

На рис. 5.14 зображено частину снєпшоту, який було автоматично створено після запуску інтеграційного тесту.

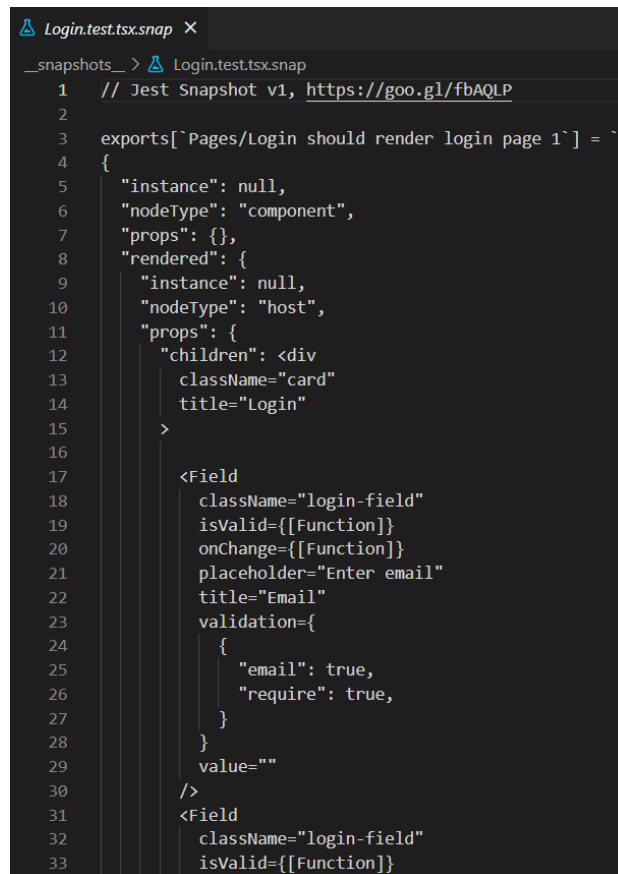


Рисунок 5.14 – Снєпшот інтеграційного тесту

Результати інтеграційного тестування процесу авторизації зображені на рис. 5.15, помилок не виявлено.

```
PASS src/test/Pages/Login.test.tsx
Pages/Login
  ✓ should render login page (7 ms)
  ✓ should render login page with valid values (4 ms)
  ✓ should render login page with incorrect email value (1 ms)
  ✓ should render login page with incorrect password value (1 ms)

  > 4 snapshots written.
Snapshot Summary
  > 4 snapshots written from 1 test suite.

Test Suites: 1 passed, 1 total
Tests:       4 passed, 4 total
Snapshots:   4 written, 4 total
Time:        0.398 s, estimated 1 s
Ran all test suites.
```

Рисунок 5.15 – Результати інтеграційного тестування

5.3 Висновки за розділом 5

Було протестовано розроблене програмне забезпечення, для тестування було використано систему Jest, з використанням якого було проведено модульне та інтеграційне тестування.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи магістра, було проведено огляд та аналіз існуючих систем для онлайн шкіл. Проведений аналіз показав, що для онлайн школи, яка активно розвивається, не завжди достатньо функціоналу таких систем, як Moodle та Google Classroom. Було використано методології для визначення вимог та розроблено вебзастосунок, яким вже може користуватись приватна онлайн школа.

Складено технічне завдання, в якому зазначена мета розробки та її призначення, основні вимоги до програми, порядок її функціонування.

Розроблена система дозволяє виконувати реєстрацію та авторизацію користувача, є можливість відновлення пароля, є адміністративна панель для маніпуляції усіма даними сайту (створення, видалення та редагування даних учнів, матеріалів, розподіл всіх матеріалів для учнів в залежності від групи та інше), сортування адміністратором учнів за групами, розміщення адміністратором відеозаписів пройдених уроків в розділі «Відеоуроки» та перегляд учнями цього розділу, розміщення адміністратором файлів з домашнім завданням в розділі «Домашні завдання» та перегляд учнями цього розділу, розміщення учнями викнаного домашнього завдання в особистому кабінеті в розділі «Домашні завдання», є тести в розділі «Домашні завдання» з різноманітним функціоналом, можливе створення та редагування інформації (розклад, новини про перенесення уроків тощо) адміністратором в залежності від групи учня в розділі «Розклад» та перегляд учнями цього розділу.

Програма була розроблена з використанням сучасних технологій, має гарний і зрозумілий інтерфейс, завдяки якому користуватись системою легко та комфортно.

Система може бути задіяна в приватній онлайн школі. Вебзастосунок повністю дороблено та готове до змін у будь-який час, зручне у використанні та буде зрозумілим для кожного користувача, навіть якщо у нього немає спеціальних навичок. Програма захищена авторськими правами.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Information Technology in Education [Electronic resource]. – Access mode:
<https://www.sciencedirect.com/science/article/pii/S1877050910004370>
2. Керування вимогами як об'єкт автоматизації [Електрон. ресурс]. – Режим доступу: <https://posibniki.com.ua/post-avtomatizovane-keruvannya-vimogami>
3. Requirements in Web Applications: Examples and Challenges [Electronic resource]. – Access mode:
<https://study.com/academy/lesson/requirements-in-web-applications-examples-challenges.html>
4. Quality Function Deployment [Electronic resource]. – Access mode:
<https://www.sciencedirect.com/topics/engineering/quality-function-deployment>
5. Introduction to Quality Function Deployment [Electronic resource]. – Access mode: <https://quality-one.com/qfd/>
<https://quality-one.com/qfd/>
6. Van Notten Ph. Writing on the wall: scenario development in times of discontinuity. – Florida: Boca Raton, 2005. – 209 p.
7. Breslin M., Buchanan R. On the case study method of research and teaching in design. – Cambridge: The MIT Press, 2008.
8. Yin R.K. Case study research. – New York: Sage, 2009.
9. Traynor D. Intercom on jobs to be done. – San Francisco: Intercom Inc., 2017.
10. Costa T. Design personas and customer journey maps. – Cambridge: Forrester Research, 2015.
11. Мови програмування [Електрон. ресурс]. – Режим доступу:
<https://web-academy.ua/blog/junior/kakie-iazyki-programmirovania-izuchat-v-2022>
12. Pros and Cons of React [Electronic resource]. – Access mode:
<https://www.techpocket.org/pros-and-cons-of-react-virtual-dom-jsx-syntax-and-other-factors/>

13. Redux [Electronic resource]. – Access mode:
<https://blog.logrocket.com/understanding-redux-tutorial-examples>
14. TypeScript [Electronic resource]. – Access mode:
<https://www.altexsoft.com/blog/typescript-pros-and-cons/>
15. Bootstrap vs MUI [Electronic resource]. – Access mode:
<https://flatlogic.com/blog/bootstrap-vs-material-ui-which-one-to-use-for-the-next-web-app/>
16. Grommet [Electronic resource]. – Access mode:
<https://retool.com/blog/react-component-libraries/#grommet>
17. Ant Design [Electronic resource]. – Access mode:
<https://pixso.net/reviews/ant-design-vs-material-ui/>
18. The most popular React UI Component Libraries [Electronic resource]. – Access mode: <https://www.sitepoint.com/popular-react-ui-component-libraries/>
19. Socket.IO [Electronic resource]. – Access mode:
<https://itnext.io/differences-between-websockets-and-socket-io-a9e5fa29d3dc>
20. WebStorm [Електрон. ресурс]. – Режим доступу:
<https://axoft.ua/webstorm-2/>
21. Comparing Database Management Systems [Electronic resource]. – Access mode:
<https://www.altexsoft.com/blog/business/comparing-database-management-systems-mysql-postgresql-mssql-server-mongodb-elasticsearch-and-others/>
22. Що таке PostgreSQL [Електрон. ресурс]. – Режим доступу:
<https://uk.css-code.org/8225113-what-is-postgresql-introduction-advantages-and-disadvantages>
23. Axios [Електрон. ресурс]. – Режим доступу:
<https://www.digitalocean.com/community/tutorials/react-axios-react>
24. Lodash [Electronic resource]. – Access mode:
<https://www.geeksforgeeks.org/lodash/>
25. Sass [Електрон. ресурс]. – Режим доступу:
<https://ritorika.com.ua/informacija/7/scss-kak-podkljuchit-k-css>

26. Validator [Electronic resource]. – Access mode:
<https://www.npmjs.com/package/validator>

ДОДАТОК А
Текст програми

A.1 Файл /src/components/PreviewDocument/index.tsx

```
import React, { useState } from 'react'
import { Document, Page } from "react-pdf/dist/esm/entry.webpack";

import Box from '@mui/material/Box';
import Card from '@mui/material/Card';
import Modal from '@mui/material/Modal';
import Typography from '@mui/material/Typography'
import CardContent from '@mui/material/CardContent'

import './style.scss'

const PreviewDocument = ({
  title,
  file,
  height = 300
}: {
  title: string,
  file: string
  height?: number
}) => {
  const [numPages, setNumPages] = useState(null);
  const [open, setOpen] = useState(false);
  const handleOpen = () => setOpen(true);
  const handleClose = () => setOpen(false);

  function onDocumentLoadSuccess({ numPages }: { numPages: any }) {
    setNumPages(numPages);
  }

  return <>
    <Card onClick={handleOpen} className='preview_document' sx={{ width: '280px',
marginRight: '20px', marginBottom: '20px' }}>
      <CardContent>
        <Document file={file}>
          <Page className='page' pageNumber={1} height={height} />
        </Document>
        <Typography variant="h6">{title}</Typography>
      </CardContent>
    </Card>
    <Modal open={open} onClose={handleClose}>
      <Box className="preview_document__show">
        <Document
          file={file}
          onLoadSuccess={onDocumentLoadSuccess}
        >
          {Array.from(new Array(numPages), (el, index) => (
            <Page key={`page_${index + 1}`} pageNumber={index + 1} />
          ))}
        </Document>
      </Box>
    </Modal>
  </>
}

export default PreviewDocument
```

A.2 Файл /src/components/Header/index.tsx

```
import React from 'react'
import { Link } from 'react-router-dom'
import { useSelector, useDispatch } from 'react-redux';

import AppBar from '@mui/material/AppBar';
import Toolbar from '@mui/material/Toolbar';
import Typography from '@mui/material/Typography';
import Button from '@mui/material/Button';

import { setIsAuth, selectIsAuth } from '../../store/auth'

const Header = () => {
  const isAuth = useSelector(selectIsAuth)
  const dispatch = useDispatch();

  return <AppBar position='sticky'>
    <Toolbar>
      <Typography variant="h6" component="div" sx={{ flexGrow: 1 }}>
        <Link to="/">
          English
        </Link>
      </Typography>
      { isAuth ?
        <Button color="inherit" onClick={() => {
          localStorage.setItem('token', '')
          dispatch(setIsAuth(false))
        }}>Log Out</Button> :
        <Button color="inherit">
          <Link to="/login">
            Login
          </Link>
        </Button>
      }
    </Toolbar>
  </AppBar>
}

export default Header
```

A.3 Файл /src/containers/App/index.tsx

```
import React from 'react'
import { Outlet } from 'react-router-dom'

import Header from '../../components/Header'

const App = () => {
  return <>
    <Header />
    <Outlet />
  </>
}

export default App
```

A.4 Файл /src/containers/ProfileTabs/index.tsx

```
import React, { FC, PropsWithChildren } from 'react';
import { Link } from 'react-router-dom';
import Box from '@mui/material/Box';
import Stack from '@mui/material/Stack';
import Button from '@mui/material/Button';

import { PROFILE, HOMEWORKS, MATERIALS } from '../../router/paths'

const ProfileTabs: FC<PropsWithChildren> = ({ children }) => {
  return <Box sx={{ display: 'flex', height: '100%' }}>
    <Box sx={{
      borderRight: '3px solid #1976d2',
      padding: '20px',
      boxShadow: '0 0 15px 0 rgba(0,0,0,0.75)'
    }}>
      <Stack direction="column">
        <Button sx={{ marginBottom: '10px' }} variant="text">
          <Link to={PROFILE}>
            Profile
          </Link>
        </Button>
        <Button sx={{ marginBottom: '10px' }} variant="text">
          <Link to={HOMEWORKS}>
            Homeworks
          </Link>
        </Button>
        <Button sx={{ marginBottom: '10px' }} variant="text">
          <Link to={MATERIALS}>
            Materials
          </Link>
        </Button>
      </Stack>
    </Box>
    <Box sx={{ padding: '25px', width: '100%', overflow: 'scroll' }}>
      {children}
    </Box>
  </Box>
}

export default ProfileTabs
```

A.5 Файл /src/pages/Login/index.tsx

```
import React from 'react'
import validator from 'validator'
import { useDispatch } from 'react-redux'

import Container from "@mui/material/Container";
import Alert from '@mui/material/Alert';
import Box from '@mui/material/Box';
import TextField from '@mui/material/TextField';
import AccountCircle from '@mui/icons-material/AccountCircle';
import Card from '@mui/material/Card';
import CardActions from '@mui/material/CardActions';
import CardContent from '@mui/material/CardContent';
import Button from '@mui/material/Button';
import Typography from '@mui/material/Typography';
import LockIcon from '@mui/icons-material/Lock';
```

```

import api from '../services/Api'

import { setIsAuth } from '../store/auth'

const Login = () => {
  const dispatch = useDispatch();

  const [isValidEmail, setIsValidEmail] = React.useState<boolean>(false)
  const [isValidPassword, setIsValidPassword] = React.useState<boolean>(false)

  const [email, setEmail] = React.useState<string>('')
  const [password, setPassword] = React.useState<string>('')

  const [errorMessage, setErrorMessage] = React.useState<string>('')

  const updateEmail = (email: string) => {
    setIsValidEmail(validator.isEmail(email))

    setEmail(email)
  }

  const updatePassword = (password: string) => {
    setIsValidPassword(validator.minLength(password, { min: 6 }))

    setPassword(password)
  }

  const login = React.useCallback(async () => {
    const data = await api.post('/login', { email, password })

    if (data.result) {
      localStorage.setItem('token', data.result.accessToken)

      dispatch(setIsAuth(true))

      return
    }

    setErrorMessage(data.error.message)
  }, [email, password])

  return<Container sx={{ height: '100%' }}>
    <Box sx={{ mt: 10 }}>
      <Card sx={{ p: 3 }}>
        <CardContent>
          <Typography variant="h5" component="div" align="center">
            Login
          </Typography>
          { errorMessage && <Alert severity="error">{errorMessage}</Alert> }
          <Box sx={{ display: 'flex', alignItems: 'flex-end' }}>
            <AccountCircle sx={{ color: 'action.active', mr: 1, my: 0.5 }} />
            <TextField
              required
              fullWidth
              type="email"
              label="Email"
              variant="standard"
              onChange={({ target }) => updateEmail(target.value)}
            />
          </Box>
          <Box sx={{ display: 'flex', alignItems: 'flex-end' }}>
            <LockIcon sx={{ color: 'action.active', mr: 1, my: 0.5 }} />
            <TextField

```

```

        required
        fullWidth
        type="password"
        label="Password"
        variant="standard"
        onChange={({ target }) => updatePassword(target.value)}
      />
    </Box>
  </CardContent>
  <CardActions sx={{ display: 'flex', justifyContent: 'center' }}>
    <Button disabled={!isValidEmail || !isValidPassword} onClick={login}
variant="contained">Login</Button>
  </CardActions>
</Card>
</Box>
</Container>
}

export default Login

```

A.6 Файл /src/pages/Materials/index.tsx

```

import React, { useEffect, useState } from 'react'
import Box from '@mui/material/Box'
import PreviewDocument from "../../components/PreviewDocument/index";
import ProfileTabs from '../../containers/ProfileTabs'
import api from '../../services/Api'

const Materials = () => {
  const [materials, setMaterials] = useState<any[]>([])

  useEffect(() => {
    api.get('/materials').then((data) => setMaterials(data.result))
  }, [])

  return <ProfileTabs>
    <Box sx={{
      display: 'flex',
      flexWrap: 'wrap',
    }}>
      {materials.map(item => <PreviewDocument key={item.id} title={item.title}
file={item.url} />)}
    </Box>
  </ProfileTabs>
}

export default Materials

```

A.7 Файл /src/pages/Profile/index.tsx

```

import React, { useState, useEffect } from 'react'
import Box from '@mui/material/Box';
import Typography from '@mui/material/Typography'

```

```

import ProfileTabs from '../containers/ProfileTabs'

import api from '../services/Api'

const Profile = () => {
  const [group, setGroup] = useState<string>>('')
  const [email, setEmail] = useState<string>>('')
  const [phone, setPhone] = useState<string>>('')
  const [lastName, setLastName] = useState<string>>('')
  const [firstName, setFirstName] = useState<string>>('')

  useEffect(() => {
    api.get('/me')
      .then((data) => {
        setPhone(data.result.user.profile.phone)
        setGroup(data.result.user.profile.group.name)
        setLastName(data.result.user.profile.lastName)
        setEmail(data.result.user.emailCredential.email)
        setFirstName(data.result.user.profile.firstName)
      })
  })

  return <ProfileTabs>
    <Box>
      <Typography variant="h4" textAlign="center">Profile</Typography>
      <table>
        <tbody>
          <tr>
            <td><Typography sx={{ fontWeight: 'bold' }}>Full
Name</Typography></td>
            <td><Typography>{firstName + ' ' + lastName}</Typography></td>
          </tr>
          <tr>
            <td><Typography sx={{ fontWeight: 'bold' }}>Group
Name</Typography></td>
            <td><Typography>{group}</Typography></td>
          </tr>
          <tr>
            <td><Typography sx={{ fontWeight: 'bold' }}>Email</Typography></td>
            <td><Typography>{email}</Typography></td>
          </tr>
          <tr>
            <td><Typography sx={{ fontWeight: 'bold', paddingRight: '15px'
}}>Phone number</Typography></td>
            <td><Typography>{phone}</Typography></td>
          </tr>
        </tbody>
      </table>
    </Box>
  </ProfileTabs>
}

export default Profile

```

A.8 Файл /src/router/Private/index.tsx

```

import React, { useEffect, FC, PropsWithChildren } from 'react'
import { useNavigate } from 'react-router-dom'
import { useSelector } from 'react-redux'

import * as path from './paths'

```

```
import { selectIsAuth } from '../store/auth'

const Private: FC<PropsWithChildren> = ({ children }) => {
  const navigate = useNavigate();

  const isAuth = useSelector(selectIsAuth)

  useEffect(() => {
    if (!isAuth) {
      navigate(path.HOME)
    }
  }, [isAuth])

  return <>{children}</>
}

export default Private
```

A.9 Файл /src/router/Public/index.tsx

```
import React, { useEffect, FC, PropsWithChildren } from 'react'
import { useNavigate } from 'react-router-dom'
import { useSelector } from 'react-redux'

import * as path from './paths'

import { selectIsAuth } from '../store/auth'

const Public: FC<PropsWithChildren> = ({ children }) => {
  const navigate = useNavigate();

  const isAuth = useSelector(selectIsAuth)

  useEffect(() => {
    if (isAuth) {
      navigate(path.PROFILE)
    }
  }, [isAuth])

  return <>{children}</>
}

export default Public
```

A.10 Файл /src/services/Api/index.ts

```
import axios from 'axios'
import { BACKEND } from '../../../configs'

const instance = axios.create({
  baseURL: BACKEND.URL
})

instance.interceptors.request.use(
  (config) => {
    if (!config.headers) {
      config.headers = {}
    }
  }
)
```

```

    config.headers.authorization = `Bearer ${localStorage.getItem('token')}`

    return config
  },
)

export default {
  get: async (url: string) => {
    try {
      const response = await instance.get(url)
      return response.data
    } catch (error) {
      // @ts-ignore
      return error.response.data
    }
  },
  put: async (url: string, data: any) => {
    try {
      const response = await instance.put(url, { payload: data })
      return response.data
    } catch (error) {
      // @ts-ignore
      return error.response.data
    }
  },
  post: async (url: string, data: any) => {
    try {
      const response = await instance.post(url, { payload: data })
      return response.data
    } catch (error) {
      // @ts-ignore
      return error.response.data
    }
  },
  delete: async (url: string) => {
    try {
      const response = await instance.delete(url)
      return response.data
    } catch (error) {
      // @ts-ignore
      return error.response.data
    }
  },
}

```

A.11 Файл /src/store/auth/index.ts

```

import { createSlice } from '@reduxjs/toolkit'

export const authSlice = createSlice({
  name: 'auth',
  initialState: {
    isAuth: (localStorage.getItem('is_auth') &&
JSON.parse(localStorage.getItem('is_auth') || '')) || false,
  },

```

```

    reducers: {
      setIsAuth: (state, action) => {
        localStorage.setItem('is_auth', action.payload)
        state.isAuth = action.payload
      }
    }
  })

export const { setIsAuth } = authSlice.actions

export const selectIsAuth = (state: any) => state.auth.isAuth

export default authSlice.reducer

```

A.12 Файл /src/index.tsx

```

import React from 'react';
import ReactDOM from 'react-dom/client';
import { Provider } from 'react-redux'
import { RouterProvider } from 'react-router-dom'
import store from './store'
import router from './router'
import './global.scss'

const root = ReactDOM.createRoot(
  document.getElementById('root') as HTMLElement
);
root.render(
  <Provider store={store}>
    <RouterProvider router={router} />
  </Provider>
);

```

A.13 Файл /src/components/Field/index.tsx

```

import React, { ChangeEvent, useState } from "react";
import { Input, Typography } from 'antd'
import validator from 'validator'

import './style.scss'

interface FieldProps {
  type?: 'password'
  value: string
  title?: string
  className?: string
  placeholder?: string
  onChange: (value: string) => void
  isValid?: (value: boolean) => void
  validation?: {
    require?: boolean,
    email?: boolean,
    maxLength?: number
    minLength?: number
  }
}

```

```

const Field = ({
  isValid,
  value,
  type,
  title,
  onChange,
  className,
  placeholder,
  validation,
}: FieldProps) => {
  const [errorMessage, setErrorMessage] = useState<string>('')

  const checkValid = (value: string) => {
    if (!validation) {
      return ''
    }

    const { require, email, maxLength, minLength } = validation

    if (require && validator.isEmpty(value)) {
      return 'The field is required'
    }

    if (email && !validator.isEmail(value)) {
      return 'Email is incorrect'
    }

    if (maxLength && !validator.isLength(value, { max: maxLength })) {
      return `Length must not exceed ${maxLength} characters`
    }

    if (minLength && !validator.isLength(value, { min: minLength })) {
      return `Length must not be less than ${minLength} characters`
    }

    return ''
  }

  const handlerUpdate = (e: ChangeEvent<HTMLInputElement>) => {
    onChange(e.target.value)

    const validResult = checkValid(e.target.value)

    setErrorMessage(validResult)

    if (isValid) isValid(!validResult)
  }

  return <div className={`field ${className}`}>
    <Typography.Text className="title">{title}</Typography.Text>
    <Input
      type={type}
      value={value}
      status={errorMessage ? 'error' : ''}
      className="input"
    />
  </div>

```

```

        placeholder={placeholder}
        onChange={handlerUpdate}
      />
      <Typography.Text className='error'{errorMessage}</Typography.Text>
    </div>
  }

export default Field

```

A.14 Файл /src/components/Header/index.tsx

```

import React from 'react'
import { Link } from 'react-router-dom'
import { useDispatch } from "react-redux";

import { Layout, Typography, Button } from 'antd'

import { USERS } from '../../../router/paths'
import { setIsAuth } from "../../../store/auth";

import './style.scss'

const Header = () => {
  const dispatch = useDispatch();

  return <Layout.Header className='header'>
    <Link to={USERS}>
      <Typography.Title level={4} className='title'>Learn English - Admin
panel</Typography.Title>
    </Link>
    <Button className='logout-button' onClick={() => {
      localStorage.setItem('token', '')
      dispatch(setIsAuth(false))
    }}>
      Logout
    </Button>
  </Layout.Header>
}

export default Header

```

A.15 Файл /src/components/Modal/Group/Create/index.tsx

```

import React, { useState, useEffect, useCallback } from 'react'

import { Alert, Modal } from 'antd'

import Field from '../../../Field'
import MultiSelect from '../../../MultiSelect'

import api from '../../../services/Api'

import './style.scss'

const CreateGroup = ({
  isOpen,

```

```

    close,
  }: {
    isOpen: boolean
    close: () => void
  }) => {
    const [materials, setMaterials] = useState([])

    const [errorMessage, setErrorMessage] = useState<string>('')

    const [name, setName] = useState<string>('')
    const [materialIds, setMaterialIds] = useState<string[]>([])

    useEffect(() => {
      api.get('/materials')
        .then(data => {
          setMaterials(data.result)
        })
    }, [])

    const submit = useCallback(async () => {
      setErrorMessage('')

      const data = await api.post('/groups', { name, materialIds })

      if (data.result) {
        return close()
      }

      setErrorMessage(data.error.message)
    }, [name, materialIds])

    return <Modal
      className='create-group-modal'
      open={isOpen}
      okText='Create'
      cancelText='Cancel'
      title='Create Group'
      onOk={submit}
      onCancel={close}
    >
      { errorMessage && <Alert className='alert' message={errorMessage}
type='error' /> }
      <Field
        title="Title*"
        value={name}
        onChange={setName}
        placeholder="Enter title"
        validation={{
          require: true
        }}
      />
      <MultiSelect
        value={materialIds}
        title='Material*'
        onChange={setMaterialIds}

```

```

        options={materials.map((material: any) => ({ value: material.id, label:
material.title })))}
        placeholder={'Enter groups'}
        validation={{
            require: true,
        }}
    />
</Modal>
}

export default CreateGroup

```

A.16 Файл /src/components/Modal/Group/Edit/index.tsx

```

import React, {useCallback, useEffect, useState} from "react";

import { Alert, Modal } from 'antd'

import Field from '../../../Field'
import MultiSelect from '../../../MultiSelect'

import api from "../../../../services/Api";

import './style.scss'

const EditGroup = ({
    groupId,
    isOpen,
    close,
}: {
    groupId: string
    isOpen: boolean
    close: () => void
}) => {
    const [materials, setMaterials] = useState([])

    const [errorMessage, setErrorMessage] = useState<string>('')

    const [name, setName] = useState<string>('')
    const [materialIds, setMaterialIds] = useState<string[]>([])

    useEffect(() => {
        api.get('/materials')
            .then(data => {
                setMaterials(data.result)
            })
    }, [])

    useEffect(() => {
        if (groupId) {
            api.get(`/groups/${groupId}`)
                .then(data => {
                    setName(data?.result?.name)
                    setMaterialIds(data?.result?.groupMaterial?.map((groupMaterial: any) =>
groupMaterial.materialId))
                })
        }
    }, [groupId])

```

```

const submit = useCallback(async () => {
  setErrorMessage('')

  const data = await api.put(`/groups/${groupId}`, { name, materialIds })

  if (data.result) {
    return close()
  }

  setErrorMessage(data.error.message)
}, [groupId, name, materialIds])

return <Modal
  className='edit-group-modal'
  open={isOpen}
  okText='Edit'
  cancelText='Cancel'
  title='Edit Group'
  onOk={submit}
  onCancel={close}
>
  { errorMessage && <Alert className='alert' message={errorMessage}
type='error' /> }
  <Field
    title="Title*"
    value={name}
    onChange={setName}
    placeholder="Enter title"
    validation={{
      require: true
    }}
  />
  <MultiSelect
    value={materialIds}
    title='Material*'
    onChange={setMaterialIds}
    options={materials.map((material: any) => ({ value: material.id, label:
material.title })))}
    placeholder={'Enter groups'}
    validation={{
      require: true,
    }}
  />
</Modal>
}

export default EditGroup

```

A.17 Файл /src/components/Modal/Material/Create/index.tsx

```

import React, { useState, useEffect, useCallback } from 'react'
import { UploadOutlined } from '@ant-design/icons';

import { Alert, Modal, Upload, Button } from 'antd'

import { BACKEND } from '../../../../configs'

import Field from '../../../../Field'
import MultiSelect from '../../../../MultiSelect'

```

```

import api from '../../../services/Api'

import './style.scss'

const CreateMaterial = ({
  isOpen,
  close,
}): {
  isOpen: boolean
  close: () => void
} => {
  const [groups, setGroups] = useState([])

  const [errorMessage, setErrorMessage] = useState<string>('')

  const [title, setTitle] = useState<string>('')
  const [url, setUrl] = useState<string>('')
  const [groupIds, setGroupIds] = useState<string[]>([])

  useEffect(() => {
    api.get('/groups')
      .then(data => {
        setGroups(data.result)
      })
  }, [])

  const submit = useCallback(async () => {
    setErrorMessage('')

    const data = await api.post('/materials', { title, url, groupIds })

    if (data.result) {
      return close()
    }

    setErrorMessage(data.error.message)
  }, [title, url, groupIds])

  return <Modal
    className='create-material-model'
    open={isOpen}
    okText='Create'
    cancelText='Cancel'
    title='Create Material'
    onOk={submit}
    onCancel={close}
  >
    { errorMessage && <Alert className='alert' message={errorMessage}
type='error' /> }
    <Field
      title="Title*"
      value={title}
      onChange={setTitle}
      placeholder="Enter title"
      validation={{
        require: true

```

```

    }}
  />
  <Upload
    name='file'
    action={`${BACKEND.URL}/upload-file`}
    headers={{
      authorization: `Bearer ${localStorage.getItem('token')}`
    }}
    onChange={({ file, fileList }) => {
      if (file.response) setUrl(file.response.result)
      if (!fileList.length) setUrl('')
    }}
  >
    <Button icon={<UploadOutlined />}>Click to Upload</Button>
  </Upload>
  <MultiSelect
    value={groupIds}
    title='Groups*'
    onChange={setGroupIds}
    options={groups.map((group: any) => ({ value: group.id, label: group.name
  })))}
    placeholder='Enter groups'
    validation={{
      require: true,
    }}
  />
</Modal>
}

export default CreateMaterial

```

A.18 Файл /src/components/Modal/Material/Edit/index.tsx

```

import React, { useEffect, useState, useCallback } from 'react'
import { Alert, Modal } from 'antd'

import Field from '../../../Field'
import MultiSelect from '../../../MultiSelect'

import api from '../../../services/Api'

import './style.scss'

const EditMaterial = ({
  materialId,
  isOpen,
  close,
}: {
  materialId: string
  isOpen: boolean
  close: () => void
}) => {
  const [groups, setGroups] = useState([])

  const [errorMessage, setErrorMessage] = useState<string>('')

  const [title, setTitle] = useState<string>('')

```

```

const [groupIds, setGroupIds] = useState<string[]>([])

useEffect(() => {
  api.get('/groups')
    .then(data => {
      setGroups(data.result)
    })
}, [])

useEffect(() => {
  if (materialId) {
    api.get(`/materials/${materialId}`)
      .then(data => {
        setTitle(data?.result?.title)
        setGroupIds(data?.result?.groupMaterial?.map((groupMaterial: any) =>
groupMaterial.groupId))
      })
  }
}, [materialId])

const submit = useCallback(async () => {
  setErrorMessage('')

  const data = await api.put(`/materials/${materialId}`, { title, groupIds })

  if (data.result) {
    return close()
  }

  setErrorMessage(data.error.message)
}, [materialId, title, groupIds])

return <Modal
  className='edit-material-model'
  open={isOpen}
  okText='Edit'
  cancelText='Cancel'
  title='Edit Material'
  onOk={submit}
  onCancel={close}
>
  { errorMessage && <Alert className='alert' message={errorMessage}
type='error' /> }
  <Field
    title="Title*"
    value={title}
    onChange={setTitle}
    placeholder="Enter title"
    validation={{
      require: true
    }}
  />
  <MultiSelect
    value={groupIds}
    title='Groups*'
    onChange={setGroupIds}
    options={groups.map((group: any) => ({ value: group.id, label: group.name
})))}
    placeholder={'Enter groups'}
    validation={{
      require: true,
    }}
  />
</Modal>

```

```

}

export default EditMaterial

```

A.19 Файл /src/components/Modal/Material/Delete/index.tsx

```

import React, {useCallback, useState} from 'react'

import {Alert, Modal} from 'antd'

import api from '../../../services/Api'

const DeleteMaterial = ({
  materialId,
  isOpen,
  close,
}: {
  materialId: string
  isOpen: boolean
  close: () => void
}) => {
  const [errorMessage, setErrorMessage] = useState<string>('')

  const submit = useCallback(async () => {
    setErrorMessage('')

    const data = await api.delete(`/materials/${materialId}`)

    if (data.result) {
      return close()
    }

    setErrorMessage(data.error.message)
  }, [materialId])

  return <Modal
    open={isOpen}
    okText='Yes'
    cancelText='No'
    title='Material User'
    onOk={submit}
    onCancel={close}
  >
    { errorMessage && <Alert className='alert' message={errorMessage}
type='error' style={{ marginBottom: '8px' }} /> }
    Are you sure you want to delete?
  </Modal>
}

export default DeleteMaterial

```

A.20 Файл /src/components/Modal/User/Create/index.tsx

```

import React, { useState, useEffect, useCallback } from 'react'

import { Alert, Modal } from 'antd'

import Field from '../../../Field'
import Select from '../../../Select'

import api from '../../../services/Api'

import './style.scss'

const CreateUser = ({
  isOpen,
  close,
}: {
  isOpen: boolean
  close: () => void
}) => {
  const [groups, setGroups] = useState([])

  const [errorMessage, setErrorMessage] = useState<string>('')

  const [email, setEmail] = useState<string>('')
  const [password, setPassword] = useState<string>('')
  const [firstName, setFirstName] = useState<string>('')
  const [lastName, setLastName] = useState<string>('')
  const [phone, setPhone] = useState<string>('')
  const [groupId, setGroupId] = useState<string>('')

  useEffect(() => {
    api.get('/groups')
      .then(data => {
        setGroups(data.result)
      })
  }, [])

  const submit = useCallback(async () => {
    setErrorMessage('')

    const data = await api.post('/users', {
      email,
      password,
      firstName,
      lastName,
      phone,
      groupId,
    })

    if (data.result) {
      return close()
    }

    setErrorMessage(data.error.message)
  }, [email, password, firstName, lastName, phone, groupId])

  return <Modal
    className='create-user-model'
    open={isOpen}
    okText='Create'
    cancelText='Cancel'
    title='Create User'
  >

```

```

    onOk={submit}
    onCancel={close}
  >
  {
    errorMessage    &&    <Alert    className='alert'    message={errorMessage}
type={'error'} /> }
    <Field
      title="Email*"
      value={email}
      onChange={setEmail}
      placeholder={'Enter email'}
      validation={{
        require: true,
        email: true,
      }}
    />
    <Field
      title="Password*"
      value={password}
      onChange={setPassword}
      placeholder={'Enter password'}
      type="password"
      validation={{
        require: true,
        minLength: 6
      }}
    />
    <Field
      title="First Name*"
      value={firstName}
      onChange={setFirstName}
      placeholder={'Enter first name'}
      validation={{
        require: true,
      }}
    />
    <Field
      title="Last Name*"
      value={lastName}
      onChange={setLastName}
      placeholder={'Enter last name'}
      validation={{
        require: true,
      }}
    />
    <Field
      title="Phone Number*"
      value={phone}
      onChange={setPhone}
      placeholder={'Enter phone number'}
      validation={{
        require: true,
      }}
    />
    <Select
      value={groupId}
      title="Group*"
      onChange={setGroupId}
      options={groups.map((group: any) => ({ value: group.id, label: group.name
    })))}
      placeholder={'Enter group'}
      validation={{
        require: true,
      }}
    />

```

```

    </Modal>
  }

  export default CreateUser

```

A.21 Файл /src/components/Modal/User/Edit/index.tsx

```

import React, { useEffect, useState, useCallback } from 'react'
import { Alert, Modal } from 'antd'

import api from '../../../../services/Api'
import Field from "../../../../Field";
import Select from "../../../../Select";

import './style.scss'

const EditUser = ({
  userId,
  isOpen,
  close,
}: {
  userId: string
  isOpen: boolean
  close: () => void
}) => {
  const [groups, setGroups] = useState([])

  const [errorMessage, setErrorMessage] = useState<string>('')

  const [email, setEmail] = useState<string>('')
  const [firstName, setFirstName] = useState<string>('')
  const [lastName, setLastName] = useState<string>('')
  const [role, setRole] = useState<string>('')
  const [phone, setPhone] = useState<string>('')
  const [groupId, setGroupId] = useState<string>('')

  useEffect(() => {
    api.get('/groups')
      .then(data => {
        setGroups(data.result)
      })
  }, [])

  useEffect(() => {
    api.get(`/users/${userId}`)
      .then(data => {
        setEmail(data?.result?.emailCredential?.email)
        setFirstName(data?.result?.profile?.firstName)
        setLastName(data?.result?.profile?.lastName)
        setRole(data?.result?.role)
        setPhone(data?.result?.profile?.phone)
        setGroupId(data?.result?.profile?.groupId)
      })
  }, [userId])

  const submit = useCallback(async () => {
    setErrorMessage('')

    const data = await api.put(`/users/${userId}`, {
      email,
      firstName,
      lastName,

```

```

        role,
        phone,
        groupId,
    })

    if (data.result) {
        return close()
    }

    setErrorMessage(data.error.message)
}, [userId, email, firstName, lastName, role, phone, groupId])

return <Modal
  className='edit-user-model'
  open={isOpen}
  okText='Edit'
  cancelText='Cancel'
  title='Edit User'
  onOk={submit}
  onCancel={close}
>
  { errorMessage    &&    <Alert    className='alert'    message={errorMessage}
type={'error'} /> }
  <Field
    title="Email*"
    value={email}
    onChange={setEmail}
    placeholder={'Enter email'}
    validation={{
      require: true,
      email: true,
    }}
  />
  <Field
    title="First Name*"
    value={firstName}
    onChange={setFirstName}
    placeholder={'Enter first name'}
    validation={{
      require: true,
    }}
  />
  <Field
    title="Last Name*"
    value={lastName}
    onChange={setLastName}
    placeholder={'Enter last name'}
    validation={{
      require: true,
    }}
  />
  <Select
    title="Role*"
    value={role}
    options={[
      { value: 'admin', label: 'Admin' },
      { value: 'user', label: 'User' },
    ]}
    onChange={setRole}
    placeholder={'Enter role'}
    validation={{
      require: true,
    }}
  />

```

```

    <Field
      title="Phone Number*"
      value={phone}
      onChange={setPhone}
      placeholder={'Enter phone number'}
      validation={{
        require: true,
      }}
    />
    <Select
      value={groupId}
      title="Group*"
      onChange={setGroupId}
      options={groups.map((group: any) => ({ value: group.id, label: group.name
    })))}
      placeholder={'Enter group'}
      validation={{
        require: true,
      }}
    />
  </Modal>
}

export default EditUser

```

A.22 Файл /src/components/Modal/User/Delete/index.tsx

```

import React, {useCallback, useState} from 'react'
import {Alert, Modal} from 'antd'
import api from '../../../services/Api'

const DeleteUser = ({
  userId,
  isOpen,
  close,
}: {
  userId: string
  isOpen: boolean
  close: () => void
}) => {
  const [errorMessage, setErrorMessage] = useState<string>('')

  const submit = useCallback(async () => {
    setErrorMessage('')

    const data = await api.delete(`/users/${userId}`)

    if (data.result) {
      return close()
    }

    setErrorMessage(data.error.message)
  }, [userId])

  return <Modal
    open={isOpen}
    okText='Yes'
    cancelText='No'
    title='Delete User'

```

```

    onOk={submit}
    onCancel={close}
  >
    { errorMessage && <Alert className='alert' message={errorMessage}
type='error' style={{ marginBottom: '8px' }} /> }
    Are you sure you want to delete?
  </Modal>
}

export default DeleteUser

```

A.23 Файл /src/components/Modal/Document/index.tsx

```

import React, { useState } from 'react'

import { Document, Page } from "react-pdf/dist/esm/entry.webpack";

import { Modal } from 'antd'

const DocumentModal = ({
  file,
  isOpen,
  close,
}: {
  file: string,
  isOpen: boolean
  close: () => void
}) => {
  const [numPages, setNumPages] = useState(null);

  function onDocumentLoadSuccess({ numPages }: { numPages: any }) {
    setNumPages(numPages);
  }

  return <Modal
    open={isOpen}
    footer={null}
    onCancel={close}
    width={1030}
  >
    <Document
      file={file}
      onLoadSuccess={onDocumentLoadSuccess}
    >
      {Array.from(new Array(numPages), (el, index) => (
        <Page key={`page_${index + 1}`} pageNumber={index + 1} />
      ))}
    </Document>
  </Modal>
}

export default DocumentModal

```

A.24 Файл /src/components/MultiSelect/index.tsx

```

import React, { useState } from 'react'
import { Select, Typography } from 'antd'

```

```

import './style.scss'

interface MultiSelectProps {
  value: string[]
  options: { value: string, label: string }[]
  title?: string
  className?: string
  placeholder?: string
  onChange: (value: string[]) => void
  isValid?: (value: boolean) => void
  validation?: {
    require?: boolean,
  }
}

const MultiSelect = ({
  value,
  title,
  options,
  isValid,
  onChange,
  className,
  validation,
  placeholder,
}: MultiSelectProps) => {
  const [errorMessage, setErrorMessage] = useState<string>('')

  const checkValid = (value: string[]) => {
    if (!validation) {
      return ''
    }

    const { require } = validation

    if (require && !value.length) {
      return 'The field is required'
    }

    return ''
  }

  const handlerUpdate = (result: string[]) => {
    onChange(result)

    const validResult = checkValid(result)

    setErrorMessage(validResult)

    if (isValid) isValid(!validResult)
  }

  console.log(value)

  return <div className={`multiselect-field ${className}`}>
    <Typography.Text className='title'>{title}</Typography.Text>
    <Select
      mode='multiple'
      value={value}
      options={options}
      status={errorMessage ? 'error' : ''}
      className="select"
      placeholder={placeholder}
      onChange={handlerUpdate}
    />
  </div>

```

```

    <Typography.Text className='error'>{errorMessage}</Typography.Text>
  </div>
}

export default MultiSelect

```

A.25 Файл /src/components/Select/index.tsx

```

import React, { useState } from "react";
import { Select, Typography } from 'antd'
import validator from "validator";

import './style.scss'

interface CustomSelectProps {
  value: string
  options: { value: string, label: string }[]
  title?: string
  className?: string
  placeholder?: string
  onChange: (value: string) => void
  isValid?: (value: boolean) => void
  validation?: {
    require?: boolean,
  }
}

const CustomSelect = ({
  value,
  title,
  options,
  isValid,
  onChange,
  className,
  validation,
  placeholder,
}: CustomSelectProps) => {
  const [errorMessage, setErrorMessage] = useState<string>('')

  const checkValid = (value: string) => {
    if (!validation) {
      return ''
    }

    const { require } = validation

    if (require && validator.isEmpty(value)) {
      return 'The field is required'
    }

    return ''
  }

  const handlerUpdate = (result: string) => {
    onChange(result)

    const validResult = checkValid(result)

    setErrorMessage(validResult)

    if (isValid) isValid(!validResult)
  }

```

```

}

return <div className={`select-field ${className}`}>
  <Typography.Text className='title'>{title}</Typography.Text>
  <Select
    value={value}
    options={options}
    status={errorMessage ? 'error' : ''}
    className="select"
    placeholder={placeholder}
    onChange={handlerUpdate}
  />
  <Typography.Text className='error'>{errorMessage}</Typography.Text>
</div>
}

export default CustomSelect

```

A.26 Файл /src/components/Table/Group/index.tsx

```

import React, {useEffect, useState} from 'react'

import { Space, Table, Button, Tag } from 'antd'

import CreateGroup from '../../../Modal/CreateGroup'
import EditGroup from '../../../Modal/EditGroup'

import api from '../../../services/Api'

import './style.scss'

const GroupsTable = () => {
  const [groups, setGroups] = useState([])

  const [groupId, setGroupId] = useState<string>('')

  const [isOpenEditModal, setIsOpenEditModal] = useState<boolean>(false)
  const [isOpenCreateModal, setIsOpenCreateModal] = useState<boolean>(false)

  const columns = [
    {
      title: 'ID',
      key: 'id',
      dataIndex: 'id',
    },
    {
      title: 'Name',
      key: 'name',
      dataIndex: 'name',
    },
    {
      title: 'Materials',
      key: 'materials',
      render: (_, any, group: any) => (
        <>{group?.groupMaterial?.map((materialGroup: any) =>
          <Tag>{materialGroup.material.title}</Tag>)}</>
        )
      },
    {
      title: 'Actions',
      key: 'actions',

```

```

    render: (_: any, group: any) => (
      <Space size="middle">
        <Button
          className="btn-edit"
          onClick={() => {
            setGroupId(group.id)
            setIsOpenEditModal(true)
          }}
        >
          Edit
        </Button>
        <Button className="btn-delete">Delete</Button>
      </Space>
    )
  }
]

useEffect(() => {
  api.get('/groups')
    .then(data => {
      setGroups(data.result)
    })
}, [])

return <div className="group-table">
  <div
    style={{
      display: 'flex',
      justifyContent: 'flex-end',
      margin: '16px'
    }}
  >
    <Button onClick={() => setIsOpenCreateModal(true)} className="btn-
create">Create</Button>
    <CreateGroup
      isOpen={isOpenCreateModal}
      close={() =>
setIsOpenCreateModal(false)} />
  </div>
  <Table columns={columns} dataSource={groups} />
  <EditGroup groupId={groupId} isOpen={isOpenEditModal} close={() => {
    setGroupId('')
    setIsOpenEditModal(false)
  }} />
</div>
}

export default GroupsTable

```

A.27 Файл /src/components/Table/Material/index.tsx

```

import React, { useState, useEffect } from 'react'
import { Document, Page } from "react-pdf/dist/esm/entry.webpack";

import { Tag, Space, Table, Button } from 'antd'

import DocumentModal from '../../../Modal/Document'
import EditMaterial from '../../../Modal/EditMaterial'
import CreateMaterial from '../../../Modal/CreateMaterial'
import DeleteMaterial from '../../../Modal/DeleteMaterial'

import api from '../../../services/Api'

```

```

import './style.scss'

const MaterialsTable = () => {
  const [materials, setMaterials] = useState([])

  const [materialId, setMaterialId] = useState<string>('')

  const [isOpenEditModal, setIsOpenEditModal] = useState<boolean>(false)
  const [isOpenCreateModal, setIsOpenCreateModal] = useState<boolean>(false)
  const [isOpenDeleteModal, setIsOpenDeleteModal] = useState<boolean>(false)
  const [isOpenDocumentModal, setIsOpenDocumentModal] = useState<boolean>(false)

  const [file, setFile] = useState<string>('')

  const columns = [
    {
      title: 'Title',
      key: 'title',
      dataIndex: 'title'
    },
    {
      title: 'Document',
      key: 'document',
      render: (_, any, material: any) => (
        <div
          onClick={() => {
            setFile(material.url)
            setIsOpenDocumentModal(true)
          }}
          style={{ display: 'flex', justifyContent: 'center', cursor: 'pointer' }}
        >
          <Document file={material.url}>
            <Page pageNumber={1} height={150} />
          </Document>
        </div>
      )
    },
    {
      title: 'Groups',
      key: 'groups',
      render: (_, any, material: any) => (
        <>{material?.groupMaterial?.map((materialGroup: any) =>
          <Tag>{materialGroup.group.name}</Tag>)}</>
      )
    },
    {
      title: 'Actions',
      key: 'actions',
      render: (_, any, material: any) => (
        <Space size='middle'>
          <Button
            className='btn-edit'
            onClick={() => {
              setMaterialId(material.id)
              setIsOpenEditModal(true)
            }}
          >
            Edit
          </Button>
          <Button
            className='btn-delete'
            onClick={() => {
              setMaterialId(material.id)
              setIsOpenDeleteModal(true)
            }}
          >
            Delete
          </Button>
        </Space>
      )
    }
  ]
}

```

```

        }}
      >
        Delete
      </Button>
    </Space>
  ),
}
]

useEffect(() => {
  api.get('/materials')
    .then(data => {
      setMaterials(data.result)
    })
}, [])

return <div className="material-table">
  <div
    style={{
      display: 'flex',
      justifyContent: 'flex-end',
      margin: '16px'
    }}
  >
    <Button onClick={() => setIsOpenCreateModal(true)} className="btn-
create">Create</Button>
    <CreateMaterial isOpen={isOpenCreateModal} close={() =>
setIsOpenCreateModal(false)} />
  </div>
  <Table columns={columns} dataSource={materials} />

  <DocumentModal
    file={file}
    close={() => {
      setFile('')
      setIsOpenDocumentModal(false)
    }}
    isOpen={isOpenDocumentModal}
  />

  <EditMaterial
    materialId={materialId}
    isOpen={isOpenEditModal}
    close={() => {
      setMaterialId('')
      setIsOpenEditModal(false)
    }}
  />

  <DeleteMaterial
    materialId={materialId}
    isOpen={isOpenDeleteModal}
    close={() => {
      setMaterialId('')
      setIsOpenDeleteModal(false)
    }}
  />
</div>
}

export default MaterialsTable

```

A.28 Файл /src/components/Table/User/index.tsx

```

import React, { useState, useEffect } from 'react'

import { Space, Table, Button } from 'antd'

import CreateUser from '../../../Modal/CreateUser'
import EditUser from '../../../Modal/EditUser';
import DeleteUser from '../../../Modal/DeleteUser'

import api from '../../../../services/Api'

import './style.scss'

const UsersTable = () => {
  const [userId, setUserId] = useState<string>('')
  const [isOpenEditModal, setIsOpenEditModal] = useState<boolean>(false)
  const [isOpenCreateModal, setIsOpenCreateModal] = useState<boolean>(false)
  const [isOpenDeleteModal, setIsOpenDeleteModal] = useState<boolean>(false)
  const [users, setUsers] = useState([])

  const columns = [
    {
      title: 'First Name',
      key: 'firstName',
      render: (_, user: any) => <>{user.profile.firstName}</>
    },
    {
      title: 'Last Name',
      key: 'lastName',
      render: (_, user: any) => <>{user.profile.lastName}</>
    },
    {
      title: 'Role',
      dataIndex: 'role',
      key: 'role',
    },
    {
      title: 'Email',
      key: 'email',
      render: (_, user: any) => <>{user.emailCredential.email}</>
    },
    {
      title: 'Phone',
      key: 'phone',
      render: (_, user: any) => <>{user.profile.phone}</>
    },
    {
      title: 'Group name',
      key: 'group',
      render: (_, user: any) => <>{user.profile.group.name}</>
    },
    {
      title: 'Actions',

```

```

    key: 'actions',
    render: (_, user: any) => (
      <Space size='middle'>
        <Button
          className="btn-edit"
          onClick={() => {
            setUserId(user.id)
            setIsOpenEditModal(true)
          }}
        >
          Edit
        </Button>
        <Button
          className="btn-delete"
          onClick={() => {
            setUserId(user.id)
            setIsOpenDeleteModal(true)
          }}
        >
          Delete
        </Button>
      </Space>
    )
  }
]

useEffect(() => {
  api.get('/users')
    .then(data => {
      setUsers(data.result)
    })
}, [])

return <div className="user-table">
  <div
    style={{
      display: 'flex',
      justifyContent: 'flex-end',
      margin: '16px'
    }}
  >
    <Button onClick={() => setIsOpenCreateModal(true)} className="btn-
create">Create</Button>
    <CreateUser
      isOpen={isOpenCreateModal}
      close={() =>
setIsOpenCreateModal(false)} />
  </div>
  <Table columns={columns} dataSource={users} />

  <EditUser
    userId={userId}
    isOpen={isOpenEditModal}
    close={() => {
      setUserId('')
      setIsOpenEditModal(false)
    }}
  />

```

```

    <DeleteUser
      userId={userId}
      isOpen={isOpenDeleteModal}
      close={() => {
        setUserId('')
        setIsOpenDeleteModal(false)
      }}
    />
  </div>
}

export default UsersTable

```

A.29 Файл /src/tests/Pages/Login.test.tsx

```

import { create, act } from 'react-test-renderer'
import Login from '../pages/Login'

describe('Pages/Login', () => {
  test('should render login page', () => {
    const page = create(<Login />)

    const tree = page.toTree()

    expect(tree).toMatchSnapshot();
  })

  test('should render login page with valid values', () => {
    const page = create(<Login />)

    const instance = page.root

    act(() => {
      const inputs = instance.findAllByType('input')

      inputs[0].props.onChange({ target: { value: 'test@example.com' } })
      inputs[1].props.onChange({ target: { value: 'secret' } })
    })

    const tree = page.toTree()

    expect(tree).toMatchSnapshot();
  })

  test('should render login page with incorrect email value', () => {
    const page = create(<Login />)

    const instance = page.root

    act(() => {
      const inputs = instance.findAllByType('input')

```

```

    inputs[0].props.onChange({ target: { value: 'not valid email' } })
  })

  const tree = page.toTree()

  expect(tree).toMatchSnapshot();
})

test('should render login page with incorrect password value', () => {
  const page = create(<Login />)

  const instance = page.root

  act(() => {
    const inputs = instance.findAllByType('input')

    inputs[1].props.onChange({ target: { value: '12' } })
  })

  const tree = page.toTree()

  expect(tree).toMatchSnapshot();
})
})

```

A.30 Файл /src/tests/Components/Field.test.tsx

```

import React, { useState, useEffect } from 'react'
import { create } from 'react-test-renderer'
import Field from '../components/Field'

describe('Components/Field', () => {
  test('should render simple component', () => {
    const component = create(
      <Field
        title="Email"
        placeholder="Enter email"
        value=""
        onChange={() => {}}
      />
    );

    let tree = component.toTree()

    expect(tree).toMatchSnapshot();
  })

  test('should render component with valid validation', () => {
    const component = create(
      <Field
        title="Email"
        placeholder="Enter email"
        value='test@example.com'
        onChange={() => {}}
      />
    );
  })
})

```

```

        validation={{
          require: true,
          email: true,
        }}
      />
    )

    let tree = component.toTree()

    expect(tree).toMatchSnapshot();
  })

  test('should render component with invalid validation (incorrect email)', async
    () => {
    const component = create(
      <Field
        title="Email"
        placeholder="Enter email"
        value='not valid value'
        onChange={() => {}}
        validation={{
          email: true,
        }}
      />
    )

    let tree = component.toTree()

    expect(tree).toMatchSnapshot();
  })

  test('should render component with invalid validation (required value)', async
    () => {
    const component = create(
      <Field
        title="Email"
        placeholder="Enter email"
        value=''
        onChange={() => {}}
        validation={{
          require: true,
        }}
      />
    )

    let tree = component.toTree()

    expect(tree).toMatchSnapshot();
  })

  test('should render component with invalid validation (min length)', async ()
    => {
    const component = create(
      <Field
        title="Email"
        placeholder="Enter email"

```

```

        value='123'
        onChange={() => {}}
        validation={{
            minLength: 4,
        }}
    />
)

let tree = component.toTree()

expect(tree).toMatchSnapshot();
})

test('should render component with invalid validation (max length)', async ()
=> {
    const component = create(
        <Field
            title="Email"
            placeholder="Enter email"
            value='123456'
            onChange={() => {}}
            validation={{
                maxLength: 5,
            }}
        />
    )

    let tree = component.toTree()

    expect(tree).toMatchSnapshot();
})
})

```

ДОДАТОК Б
Слайди презентації

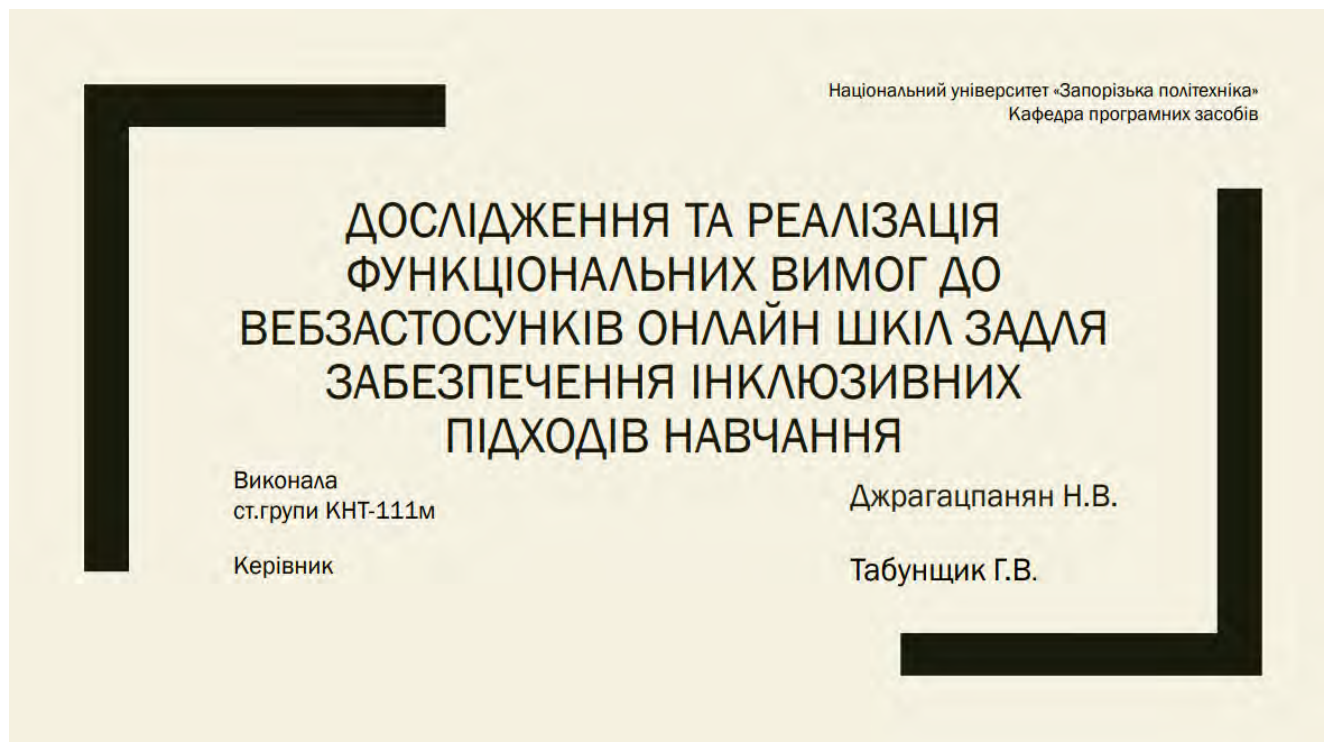


Рисунок Б.1 – Слайд 1

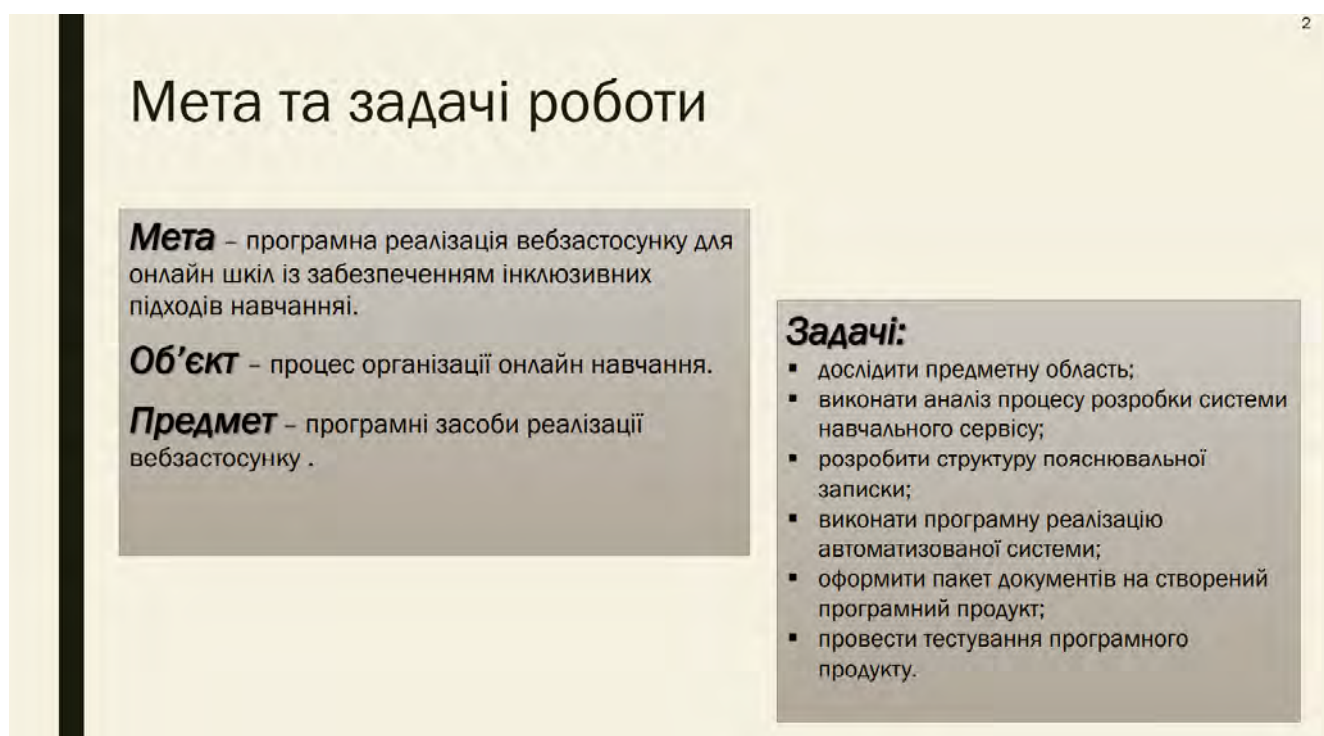


Рисунок Б.2 – Слайд 2

Переваги онлайн навчання

За допомогою онлайн шкіл можна:



Рисунок Б.3 – Слайд 3

Порівняння існуючих рішень

Критерії	Існуючі застосунки		
	Сервіси від Google	Moodle	zno.osvita.ua
Ергономічність	Ненагромаджений сучасний дизайн	Незручний та заплутаний інтерфейс програми, відсутність інтуїтивно зрозумілого управління	В цілому, сайт user-friendly, але помічена відсутність єдиного дизайну в різних розділах сайту
Функціонал	Багато корисних сервісів: Google Docs/Sheets/Slides, Google Drive, Gmail, Google Calendar, Google Meet	Дуже багато функцій, частину з яких просто не використовують. А прибрати їх можна лише за допомогою програмістів.	Є можливість самостійної роботи над помилками в тестах ЗНО/ЄВІ завдяки розгорнутим коментарям до кожного завдання.
Налаштування під свої потреби	Легко можна самостійно налаштувати середовище для роботи онлайн школи	Щоб організувати навчання та створити повноцінні курси, систему необхідно встановити, зазвичай для цього наймають IT-фахівця	Так як функціонал не такий широкий (тільки тести), нічого налаштовувати не потрібно
Вартість	Майже весь функціонал безкоштовний, оплата необхідна тільки для деяких функцій, як збільшення пам'яті в Google Drive, створення корпоративної пошти і т.д.	Використання безкоштовне	Базові функції безкоштовні, а для отримання преміум-доступу необхідно заплатити

Рисунок Б.4 – Слайд 4

Порівняння JS-фреймворків

Критерії	Інструмент		
	React.js	Vue.js	Angular.js
Тип	Бібліотека для створення користувацького інтерфейсу	Фреймворк	Фреймворк
Типи додатків	Односторінкові (SPA) і мобільні додатки	Модернізовані односторінкові додатки і підтримка нативних додатків	Нативні, гібридні і веб-додатки
Модель	Віртуальна DOM	Віртуальна DOM	MVC
Переважаюча мова розробки	Рекомендовано використовувати JSX – JavaScript XML	HTML-шаблони та JavaScript	Рекомендовано використовувати TypeScript
Ідеально підходить для	Веб-додатків і нативних додатків для iOS і Android	Веб-додатків і односторінкових (SPA) додатків	Великомасштабних додатків з багатим функціоналом
Спрямованість і класи залежностей	Одностороння передача даних	Одностороння передача даних	Взаємолії об'єктів відбувається в службах
Зворотня сумісність	Повна сумісність між версіями	Повна сумісність між версіями	Абсолютна залежність від попередніх версій
Розмір бібліотек	100 KB	80 KB	500+ KB

Рисунок Б.7 – Слайд 7

Порівняння СУБД

СУБД	Критерії			
	Структура	Масштабованість	Функціональність	Особливості
PostgreSQL	Реляційна	Чудова масштабованість. Вертикальна масштабованість є відмінною рисою PostgreSQL	Багатий функціонал. Транзакції, тригери, підзапити, зовнішні ключі, уявлення, обмеження цілісності, різні блокування та інше	Надійність, продуктивність, розширюваність, SQL підтримка, підтримка численних типів даних
MySQL	Реляційна	Бувають проблеми з масштабованістю: можна масштабувати MySQL, але для цього знадобиться більше інженерних зусиль порівняно з будь-якою базою даних NoSQL	Пропонує багато функцій, навіть у безкоштовній версії, але у безкоштовній версії найбільший упор робиться на швидкість і надійність	Дуже швидкий, багатопоточний, розрахований на багато користувачів і підтримує SQL
MongoDB	Нереляційна	Чудова масштабованість. Крім вертикального масштабування для розміщення більшої кількості даних MongoDB також може масштабуватися по горизонталі.	Широкий функціонал. Має досить гнучку мову для формування запитів, підтримує зберігання документів у JSON-подібному форматі, ефективно забезпечує зберігання великих бінарних об'єктів, може створювати індекси для різних атрибутів, що зберігаються.	Підходить для проектів з різними даними, які важко піддаються класифікації

Рисунок Б.8 – Слайд 8

Порівняння інструментів для UI

Інструменти	Критерії		
	Кастомізація компонентів	Простота та швидкість розробки	Різноманітність елементів
React Bootstrap	Можливостей для кастомізації не так багато, тому більшість додатків на Bootstrap дуже схожі	Висока швидкість розробки завдяки коду, який можна перевикористовувати	Не приділяє достатньо уваги динамічності елементів. Використовує мінімалізм у дизайні
Grommet	Не призначений для широкого налаштування. Є багато властивостей елементів, які взагалі не можна змінити	Ця гнучка бібліотека дозволяє легко реалізовувати компоненти	Великий набір компонентів для візуалізації даних (від стовпчастих діаграм до map)
MUI	Багато можливостей для кастомізації, можна розробити унікальний дизайн	Швидкість розробки не дуже висока, але можна використовувати шаблони для більшої простоти	Динамічні та привабливі елементи: анімація, різні повзунки, спливаючі вікна
Ant Design	Дуже проста кастомізація елементів	Розробка навіть кастомізованих компонентів доволі проста	Кількість елементів не дуже велика, але є багато можливостей для оформлення таблиць та форм

Рисунок Б.9 – Слайд 9

ЕНДПОІНТИ	Тип запити	Шлях запити	Опис запити (що він приймає)	Опис відповіді (що повертається)
	get	/admin/materials	Дані для пошуку матеріалів	Список матеріалів
	get	/admin/material/:materialId	Ідентифікатор матеріалу	Дані матеріалу
	post	/admin/material	Дані для створення матеріалу	Дані створеного матеріалу
	put	/admin/material/:materialId	Ідентифікатор матеріалу та дані, які треба змінити	Повідомлення про успішне оновлення матеріалу
	delete	/admin/material/:materialId	Ідентифікатор матеріалу	Повідомлення про успішне видалення матеріалу
	get	/materials	Дані для пошуку та фільтрації матеріалів	Список матеріалів
	get	/material/:materialId	Ідентифікатор матеріалу	Дані про матеріал

Рисунок Б.10 – Слайд 10

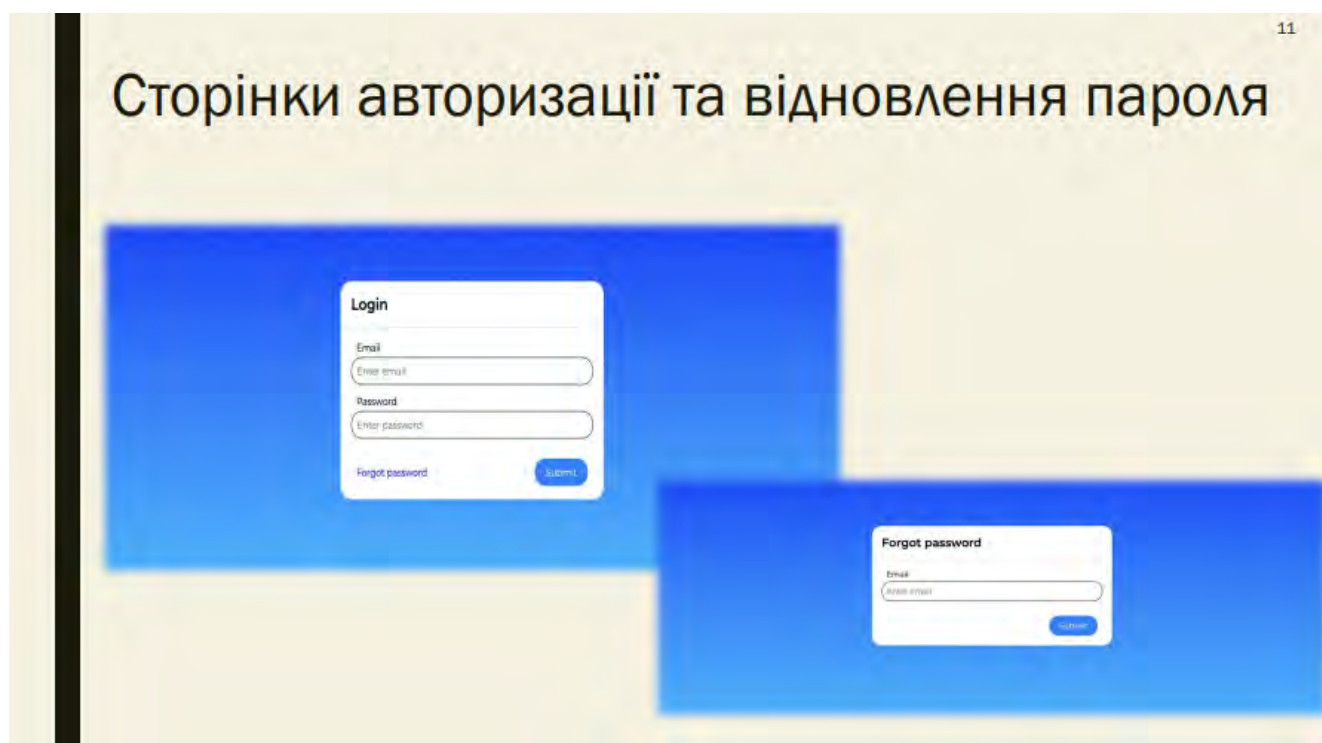


Рисунок Б.11 – Слайд 11

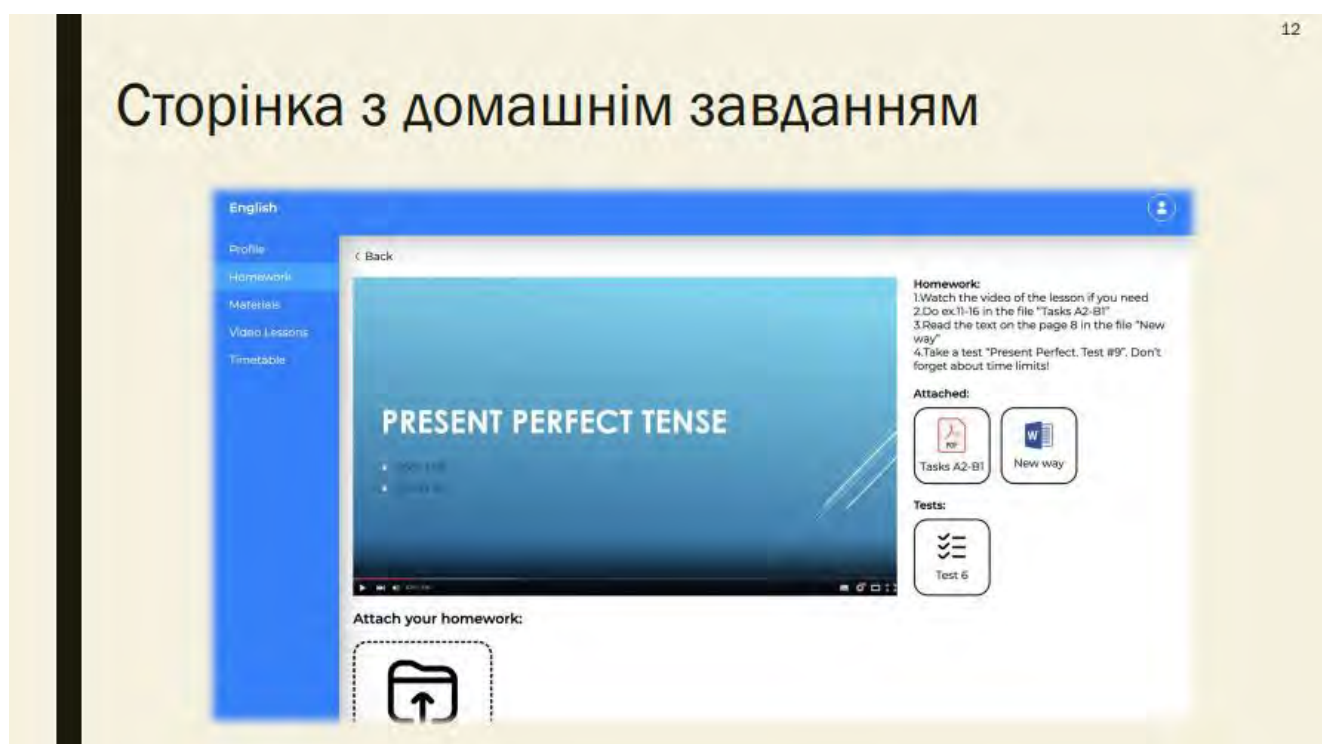


Рисунок Б.12 – Слайд 12

Сторінка з результатами тесту

English

Profile

Homework

Materials

Video Lessons

Timetable

Test 6
 Lost time: 40 min 53 sec of 45 min
 Score: 18 of 30
☒ Show all descriptions

1. In an open _____ over breakfast, Microsoft executives Shane Kim and Chris Satchell revealed that two of the home video game projects which The Lord of the Rings director Peter Jackson was working on with Bungie, an American video game developer, would take the form of episodic series.

Фрагмент речення з пропуском означає: У відкритому обговоренні під час сніданку, керівники Microsoft Шейн Кім і Кріс Сетчелл з'ясували... Тобто в реченні потрібно вставити іменник. Тому з-поміж поданих варіантів відповідей discuss, discussion, discussing, discussed граматично підходить лише discussion. Для повного розуміння різниці між варіантами відповідей рекомендується повторити правила **"Частини мови"**, **"Форми дієслова"**, **"Герундія"** та **"Past Participle"** в розділі "Матеріали".

☒ Show descriptions

☒ discuss

Цей варіант не підходить, бо це не іменник, а дієслово. Приклад використання: They discuss it every morning. Повторити **"Частини мови"**.

☒ discussion

Цей варіант правильний, discussion – іменник. Повторити **"Частини мови"**.

☐ discussing

Цей варіант не підходить, бо це не іменник, а форма дієслова - герундія. Приклад використання: We enjoy discussing our future. Повторити **"Герундія"**.

Рисунок Б.13 – Слайд 13

Сторінка з розкладом

English

Profile

Homework

Materials

Video Lessons

Timetable

	Su	Mo	Tu	We	Th	Fr	Sa
8:00							
9:00							
10:00							
11:00							
12:00							
13:00							
14:00							
15:00							
16:00				Lesson			
17:00							
18:00		Lesson				Lesson	
19:00							
20:00							
21:00							
22:00							

Рисунок Б.14 – Слайд 14

15

Адміністративна панель. Сторінка з користувачами

Admin Panel

Log out

Users

Groups

Materials

Timetable

Homework

Video Lessons

Q Search

Create User

#	Full Name	Email	Group	Phone	Date	Actions
1	Tammy Wisoky	Torrance71@gmail.com	A1	+38(099)-006-43-53	Sun Jul 17 2022	 
2	Vanessa Trentow	August33@gmail.com	B2	+38(099)-613-53-26	Sat Sep 17 2022	 
3	Kent Considine	Nolan_Goyette8@gmail.com	C2	+38(099)-180-86-66	Mon Sep 19 2022	 
4	Claude Koch Sr.	Darrell_Lemke16@yahoo.com	B2	+38(099)-599-16-03	Fri Mar 04 2022	 
5	Erica Buckridge	Freddie.Hand@hotmail.com	A1	+38(099)-620-33-82	Thu Aug 25 2022	 
6	Sylvia Thiel	Missouri.Culgowski889@hotmail.com	B2	+38(099)-314-63-11	Wed Feb 16 2022	 
7	Alberta Schulist	Earline_Ziemann51@gmail.com	C2	+38(099)-320-53-58	Thu Aug 18 2022	 
8	Javier Kling	Deanna_Turcotte93@yahoo.com	A2	+38(099)-611-84-66	Mon Jun 27 2022	 
9	Krista Walker	Kellie72@hotmail.com	B1	+38(099)-497-53-30	Thu Dec 23 2021	 
10	Stephanie Homenick	Herminio_Wuckert68@yahoo.com	A2	+38(099)-925-79-51	Wed Oct 19 2022	 

1-10 from 44 users

Show on page 10 rows

Рисунок Б.15 – Слайд 15

16

Адміністративна панель. Сторінка з користувачами

Admin Panel

Log out

Users

Groups

Materials

Timetable

Homework

Video Lessons

Q Search

Create User

#	Full Name	Email	Group	Phone	Date	Actions
1	Priscilla Feest	Brigitte76@gmail.com	A1	+38(099)-935-05-62	Wed Nov 30 2022	 
2	Dominick Ward	Audrey_Nicolas@gmail.com	A1	+38(099)-562-16-14	Fri Jun 17 2022	 
3	Dianna Winthelser	Kieran86@yahoo.com	A1	+38(099)-968-42-91	Sat Jan 01 2022	 
4	Vivian Koch	Timothy_Considine@gmail.com	A1	+38(099)-055-95-44	Wed Jun 01 2022	 
5	Rudolph Aufderhar	Alexandria.Schamberger@gmail.com	A1	+38(099)-521-08-46	Thu Jan 27 2022	 
6	Imael Jenkins	Rahul.Ziemann@hotmail.com	A1	+38(099)-841-95-49	Mon Feb 21 2022	 
7	Hazel Wiegand V	Earnestine38@gmail.com	A1	+38(099)-162-31-67	Fri Jan 21 2022	 
8	Katrina Walter	Camron37@yahoo.com	A1	+38(099)-207-92-20	Thu May 05 2022	 
9	Dr. Nathaniel Fay	Gladys_Gleichner85@hotmail.com	A1	+38(099)-299-67-86	Thu Apr 07 2022	 
10	Chris Prosacco	Hilma.Brown@gmail.com	A1	+38(099)-487-64-32	Sat Mar 05 2022	 

1-10 from 11 users

Show on page 10 rows

Рисунок Б.16 – Слайд 16

Адміністративна панель. Сторінка створення користувача

Admin Panel

Log out

Users

Groups

Materials

Timetable

Homework

Video Lessons

Search

Create user

ID	Full Name	Email	Group	Phone	Date	Actions
1	Lena Macgypic			55-59	Tue Feb 15 2022	
2	Saul Ferry			60-69	Sat Oct 22 2022	
3	Kana Maite			55-59	Wed Jan 19 2022	
4	Elnasto Kiookp			60-69	Tue Mar 22 2022	
5	Dialette Reiche			2-34	Thu Mar 03 2022	
6	Victor Benier			9-45	Wed Aug 31 2022	
7	Jo Cruchshank			25-45	Mon Feb 21 2022	
8	Stella McZynn D			32-87	Sat Dec 11 2021	
9	Canton Jones			96-17	Mon Aug 08 2022	
10	Ellis Howell			53-96	Mon Jan 17 2022	

1-10 from 44 Users

Show on page 10 rows

Close

Create User

First Name

Enter first name

Last Name

Enter last name

Email

Enter email

Group

Select group

Phone

Enter phone

Cancel

Create

Рисунок Б.17 – Слайд 17

Адміністративна панель

Admin Panel

Log out

Users

Groups

Materials

Timetable

Homework

Video Lessons

Search

Upload Video Lesson

ID	File	Title	Author	Date	Actions
1	Hand LLC	Quod re quid ca Laudant	Bert Heaney	Thu Dec 16 2021	
2	McGraw, Harper and Morgan	Tidem B unde fer nibus assum	Mrs. Stuart Moore	Thu Jul 28 2022	
3	Purdy-Gibson	varia	Erick Larson	Fri Dec 10 2021	
4	Kovach, Marvin and Stamm	Vel, ap	Patrick Smitam	Sat Oct 29 2022	
5	Kihn Inc.	sapre	June Ebert	Fri Jun 02 2022	
6	Koch-Wuckert	ise pers Quidem cupidat	Craig Ziemann	Sat Feb 05 2022	
7	Coyette Inc.	facere	Nelson Cronin	Thu Mar 09 2022	

Close

Upload Video Lesson

Upload File

MP3

Title

Enter title

Description

Enter description

Cancel

Upload

Завантаження відеоуроку

Рисунок Б.18 – Слайд 18

Адміністративна панель

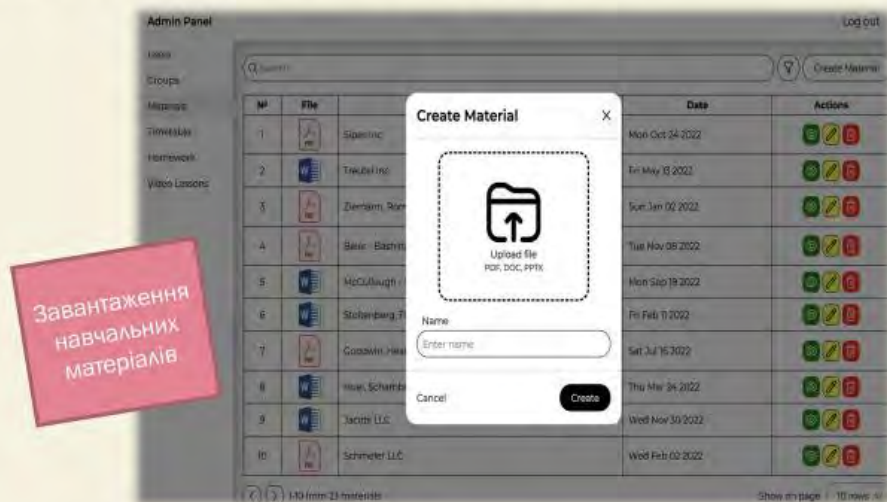


Рисунок Б.19 – Слайд 19

Адміністративна панель

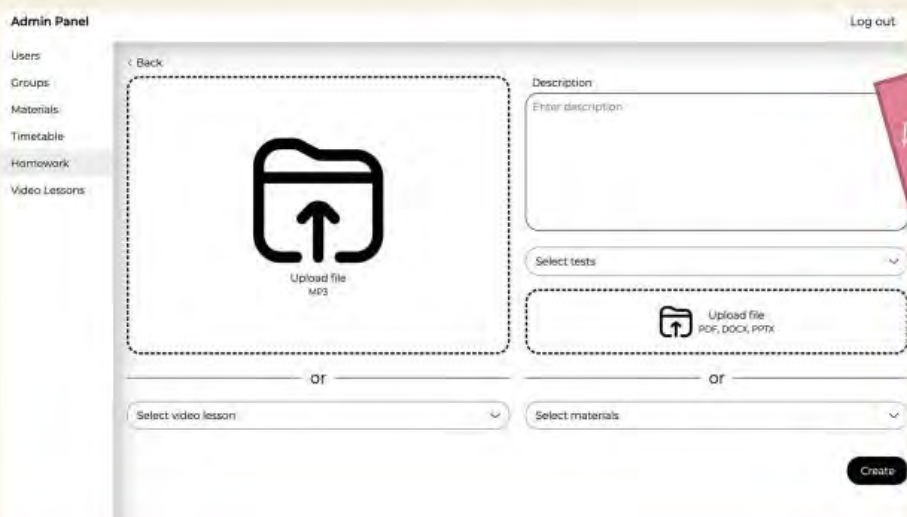


Рисунок Б.20 – Слайд 20

21

Результати тестування

```

src/test/Components/Field.test.tsx
Components/Field
  ✓ should render simple component (8 ms)
  ✓ should render component with valid validation (1 ms)
  ✓ should render component with invalid validation (incorrect email)
  ✓ should render component with invalid validation (required value)
  ✓ should render component with invalid validation (min length) (1 ms)
  ✓ should render component with invalid validation (max length)

  4 snapshots written.
Snapshot Summary
  4 snapshots written from 1 test suite.

Test Suites: 1 passed, 1 total
Tests: 6 passed, 6 total
Snapshots: 4 written, 4 total
Time: 0.46 s, estimated 1 s
Ran 10 test suites

```

Модульне
тестування

```

src/test/Pages/Login.test.tsx
Pages/Login
  ✓ should render login page (7 ms)
  ✓ should render login page with valid values (4 ms)
  ✓ should render login page with incorrect email value (1 ms)
  ✓ should render login page with incorrect password value (1 ms)

  4 snapshots written.
Snapshot Summary
  4 snapshots written from 1 test suite.

Test Suites: 1 passed, 1 total
Tests: 4 passed, 4 total
Snapshots: 4 written, 4 total
Time: 0.398 s, estimated 1 s
Ran all test suites.

```

Інтеграційне
тестування

Рисунок Б.21 – Слайд 21

22

Висновки

- В ході виконання кваліфікаційної роботи магістра, було проведено огляд та аналіз існуючих систем для навчання та викладання онлайн. Було розроблено вебзастосунок для онлайн шкіл, з урахуванням переваг та недоліків існуючих систем.
- Програма була розроблена з використанням сучасних технологій, має гарний і зрозумілий інтерфейс, завдяки якому користуватись системою легко та комфортно.
 - Система може вже зараз бути задіяна в онлайн школі.

Рисунок Б.22 – Слайд 22