

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПІДБОРУ
ІНДИВІДУАЛЬНОГО ПЛАНУ ТРЕНУВАНЬ
IMPLEMENTATION OF SOFTWARE FOR SELECTING
AN INDIVIDUAL TRAINING PLAN

Виконав(ла): студент(ка) 4 курсу, групи КНТ-120
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ПОТАПОВ Д.С.

(ПРИЗВИЩЕ та ініціали)

Керівник ШИТІКОВА О. В.

(ПРИЗВИЩЕ та ініціали)

Рецензент ТЕРЕЩЕНКО Е.В.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

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

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2024 року

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

ПОТАПОВА Данила Сергійовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Реалізація програмного забезпечення для підбору
індивідуального плану тренувань.

Implementation of Software for Selecting an Individual Training Plan

керівник проєкту (роботи) к.т.н., ШИТІКОВА Олена Вікторівна,

(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 24 ” квітня 2024 року № 168

2. Строк подання студентом проєкту (роботи) 01 червня 2024 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне
завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно
розробити) 1. Аналіз предметної області та розгляд аналогів.
2. Засоби та середовища розробки. 3 Проєктування застосунку. 4. Опис програми.
5. Експлуатація, тестування та експериментальне дослідження програми.

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

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	ШИТІКОВА О.В., доцент каф. ПЗ		
Нормоконтроль	ДЕЙНЕГА Л.Ю., ст. викладач каф. ПЗ		

7. Дата видачі завдання “24” квітня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

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

Студент(ка)

(підпис)

Данило ПОТАПОВ

(Імя ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис)

Олена ШИТІКОВА

(Імя ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи
бакалавра: 105 с., 26 рис., 4 табл., 3 дод., 11 джерел

ДЕСКТОП ЗАСТОСУНОК, ТРЕНУВАННЯ У ДОМАШНІХ
УМОВАХ, C#, VISUAL STUDIO

Об'єкт дослідження – система тренувань для покращення фізичної форми людини.

Предмет дослідження – програмне забезпечення з підбору тренувань для покращення фізичної форми.

Мета розробки – програмне забезпечення підбору тренувань для різних категорій осіб, які прагнуть покращити своє фізичне здоров'я, з метою оптимізації тренувального процесу і досягнення кращих результатів.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування, мова програмування C#, комп'ютер з операційною системою Windows, IDE Visual Studio, СКБД MySQLWorkbench.

Результати. Створено десктоп-застосунок, який взаємодіє з користувачем та дає необхідний план тренування.

Висновки. Розроблено десктоп-застосунок для правильного та безпечного тренування у домашніх умовах.

Галузь використання – особисте використовувannya для тренування в домашніх умовах.

ABSTRACT

Explanatory note for bachelor's degree qualification: 105 p., 26 figures, 4 tables, 3 appendixes, 11 sources.

DESKTOP APPLICATION, TRAINING AT HOME, C#, VISUAL STUDIO

The object of research is a training system for improving a person's physical form.

The subject of the research is software for selecting training to improve physical fitness.

The purpose of the development is training selection software for various categories of people who want to improve their physical health, with the aim of optimizing the training process and achieving better results.

Materials, methods and technical means: object-oriented programming, C# programming language, computer with Windows operating system, IDE Visual Studio, DBMS MySQLWorkbench.

The results. A desktop application has been created that interacts with the user and provides the necessary training plan.

Conclusions. A desktop application has been developed for proper and safe training at home.

Field of use - personal use for training at home.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ	9
1 Аналіз предметної області та розгляд аналогів	10
1.1 Особливості підбору індивідуальних тренувань	10
1.2 Розгляд аналогів застосунків з підбору індивідуальних тренувань	10
1.3 Постановка завдання	14
1.4 Висновки за розділом 1	15
2 Засоби та середовище розробки	16
2.1 Вибір мови програмування	16
2.2 Вибір СКБД	17
2.3 Вибір середовища розробки	19
2.4 Висновки за розділом 2	21
3 Проєктування застосунку	22
3.1 Функціональні вимоги до програми та вимоги до інтерфейсу	22
3.2 Проєктування інтерфейсу	23
3.3 Структура навігації	25
3.4 Головний алгоритм роботи програми	27
3.5 Діаграма прецедент	29
3.6 Діаграма розгортання	30
3.7 ER діаграма бази даних	30
3.8 Висновки за розділом 3	31
4 Опис програми	32
4.1 Опис класів та функцій	32
4.2 Висновки за розділом 4	37
5 Експлуатація, тестування та експериментальне дослідження програми	38
5.1 Опис роботи програми	38
5.2 Тестування програми	43
5.3 Висновки за розділом 5	49

Висновок	50
Перелік джерел посилання	51
Додаток А Технічне завдання	52
Додаток Б Код програми	61
Додаток В Слайди презентації.....	98

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

ОС – Операційна система;

ПЗ – Програмне забезпечення або застосунок;

СКБД – Система керування базою даних;

IDE – Інтегроване середовище розробки.

ВСТУП

На сьогоднішній день у багатьох країнах світу, зростає проблема зайвої ваги у людей, через неправильне харчування, ведіння сидячого образу життя, малої рухливості, тому попит на якісне програмне забезпечення(ПЗ) по підбору тренувань тільки зростає.

Програмне забезпечення по підбору тренувань на сьогоднішній день повинне вміти підбирати плани тренувань по особистим характеристикам людини, мати легкий та інтуїтивно зрозумілий інтерфейс, вміщувати різноманітну базу програм тренувань.

Об'єктом дослідження є система тренувань для покращення фізичної форми людини.

Предметом дослідження є програмне забезпечення з підбору тренувань для покращення фізичної форми.

Мета розробити програмне забезпечення підбору тренувань для різних категорій осіб, які прагнуть покращити своє фізичне здоров'я, з метою оптимізації тренувального процесу і досягнення кращих результатів.

Завданням є аналіз науково-методичних джерел та рекомендацій з тренувального процесу, розробка структурованої системи тренувань для різних рівнів підготовки, проєктування та розробка програмного забезпечення з підбору тренувань, тестування ефективності отриманого продукту на практиці та корекція за потреби.

У процесі розробки буде використовуватися 2 методи а саме аналітичний, для аналізу наукових джерел та рекомендацій, а також експериментальний, для тестування програми на практиці.

Практичним застосуванням є покращення ефективності тренувань, допомога в позбавленні зайвої ваги

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА РОЗГЛЯД АНАЛОГІВ

1.1 Особливості підбору індивідуальних тренувань

Програмне забезпечення підбору тренувань – це програма, яка допомагає користувачу скинути зайву вагу через підбір (програмою) та виконання (користувачем) плану тренувань. Для коректного підбору програма повинна буде врахувати такі характеристики людини, як вік, вага, зріст, індекс маси тіла (ІМТ). Іноді такі програми робляться під конкретний девайс компанії розробника, який може додатково відслідковувати такі параметри, як рухливість, кількість кисню у крові, пульс, артеріальний тиск. Зазначені параметри допомагають більш точно визначити витрату калорій.

Не включаючи в себе якісь особливі або специфічні можливості, основний базовий функціонал має включати в себе:

- підбір плану тренувань на основі інформації наданої користувачем такої як: вік, вага, зріст, ІМТ;
- відслідковування прогресу виконання плану;
- надавати поради щодо виконання плану;
- підтримувати зацікавленість користувача.

1.2 Розгляд аналогів застосунків з підбору індивідуальних тренувань

1.2.1 FitBit

Fitbit Coach пропонує персоналізовані тренування та плани на основі даних про вашу активність та фітнес-цілі [1]. Головну сторінку інтерфейсу застосунку, що розглядається можна побачити на рисунку 1.1.

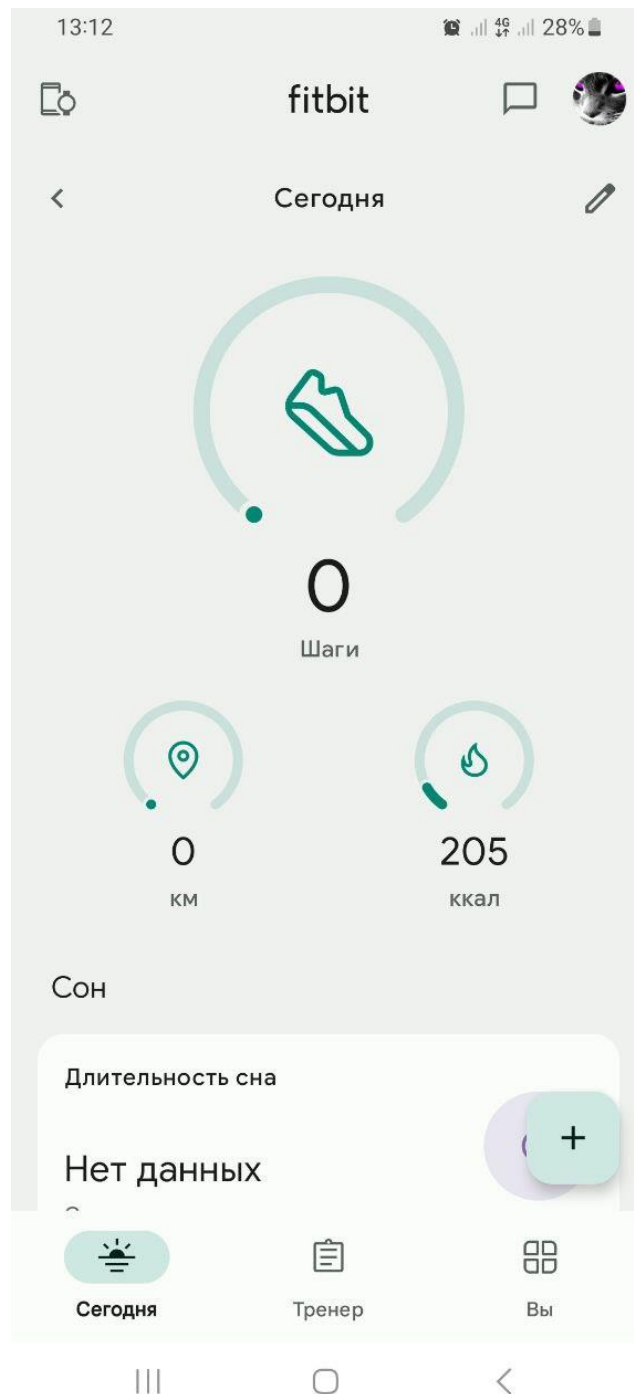


Рисунок 1.1 – Головна сторінка FitBit

Переваги:

- можливість вибрати тренування різної тривалості та складності;
- інтеграція із пристроями Fitbit для відстеження прогресу.

Недоліки:

- платна передплата для доступу до всіх функцій;

- обмежений вибір безкоштовних тренувань.

1.2.2 Fitness Buddy

Fitness Buddy програма, що пропонує різноманітні програми для тренувань, також дозволяє відслідковувати особистий прогрес [2]. Головну сторінку інтерфейсу застосунку Fitness Buddy можна побачити на рисунку 1.2.

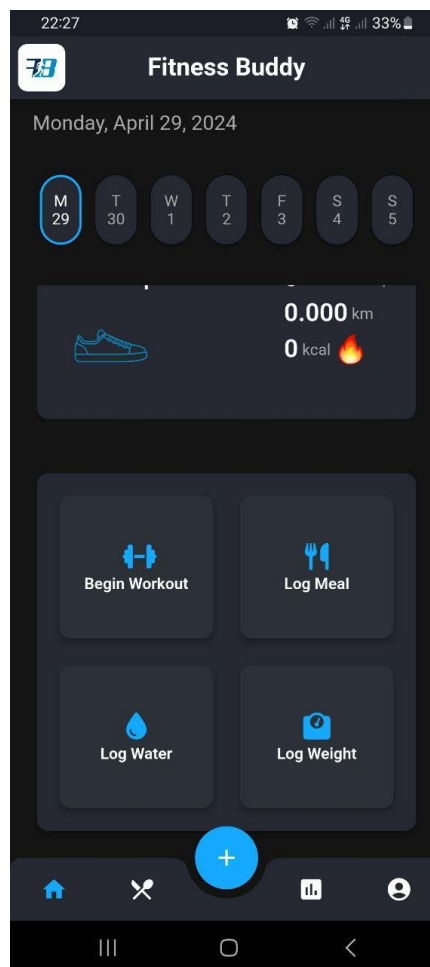


Рисунок 1.2 – Головне меню Fitness Buddy

Переваги:

- велика база даних вправ з докладними інструкціями та відео;
- персоналізовані плани тренувань на основі ваших цілей;
- можливість відслідковувати прогрес та історію тренувань.

Недоліки:

- інтерфейс може здатися не таким інтуїтивно зрозумілим для новачків;
- деякі просунуті функції доступні лише у платній версії.

1.2.38fit

8fit популярна програма тренувань, яка має додаткові матеріали для підбору особливого харчування для вибраного плану тренувань [3]. Головну сторінку інтерфейсу застосунку 8fit можна побачити на рисунку 1.3.

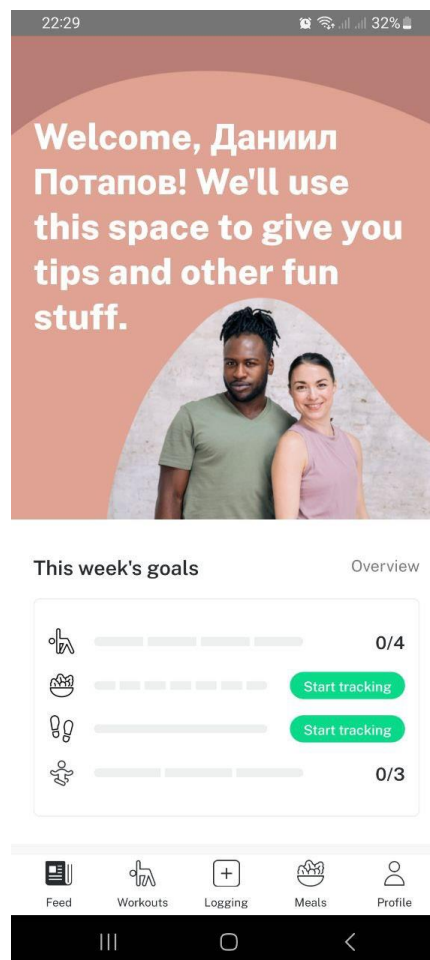


Рисунок 1.3 – Головна сторінка застосунку 8fit

Переваги:

- комбіновані плани тренувань та харчування;
- персоналізовані рекомендації на основі ваших цілей та переваг;
- щоденні плани харчування та рецепти з можливістю налаштування.

Недоліки:

- платна передплата для повного доступу до всіх функцій;
- деякі рецепти та плани харчування можуть бути непрактичними або дорогими.

1.3 Постановка завдання

При розгляді аналогів було визначено який базовий функціонал повинне включати в себе ПЗ, а саме:

- базу програм для різного рівня підготовки для підбору коректного плану для особистих характеристик користувача;
- відслідковування прогресу виконання плану;
- надання порад щодо виконання плану;
- легкий та інтуїтивно зрозумілий інтерфейс.

Також при тестуванні аналогів було виявлено проблему того, що після 2 або 3 днів тренування я втрачав бажання займатися далі. У зв'язку з цим було вирішено додати в проєкт елементи гейміфікації, тобто включити елементи гри, у даному випадку зробити користувача персонажем, який зможе прокачувати свої характеристики тренуваннями, та втрачатиме їх, якщо перестає займатися регулярно.

1.4 Висновки за розділом 1

В першому розділі було проведено аналіз предметної області. Після розгляду та аналізу аналогів, було виявлено їх переваги та недоліки, на основі яких визначено базовий функціонал розроблюваного ПЗ. Вирішено, що програма повинна мати глибку базу програм підготовки для різних рівнів, підбирати програми під кожного користувача, звертаючи увагу на їх персональні характеристики такі, як вік, вага, зріст, рівень активності та ін. Мати легкий інтерфейс, що спростить користувачу почати активніше користуватися застосунком. Також було вирішено додати до розроблюваного ПЗ елементи гейміфікації для того, щоб підтримувати у користувача інтерес до виконання підібраного плану.

2 ЗАСОБИ ТА СЕРЕДОВИЩЕ РОЗРОБКИ

2.1 Вибір мови програмування

На вибір розглядаються мови програмування C++, C#, Python.

C++ широко використовується для розробки програмного забезпечення, будучи однією з найпопулярніших мов програмування. Область його застосування включає створення операційних систем, різноманітних прикладних програм, драйверів пристроїв, додатків для систем, високопродуктивних серверів, а також комп'ютерних ігор [4].

C# відноситься до сім'ї мов з C-подібним синтаксисом, їх синтаксис найбільш близький до C++ і Java. Мова має статичну типізацію, підтримує поліморфізм, навантаження операторів (у тому числі операторів явного та неявного приведення типу), делегати, атрибути, події, змінні, властивості, узагальнені типи та методи, ітератори, анонімні функції з підтримкою замикань, LINQ, винятки, коментарі у форматі XML [5].

Python є мультипарадигменною мовою програмування, що підтримує імперативне, процедурне, структурне, об'єктно-орієнтоване програмування, метапрограмування, функціональне програмування та асинхронне програмування. Завдання узагальненого програмування вирішуються рахунок динамічної типізації. Аспектно-орієнтоване програмування частково підтримується через декоратори, повноцінніша підтримка забезпечується додатковими фреймворками. Такі методики як контрактне та логічне програмування можна реалізувати за допомогою бібліотек чи розширень. Основні архітектурні риси - динамічна типізація, автоматичне управління пам'яттю, повна інтроспекція, механізм обробки винятків, підтримка багатопоточних обчислень з глобальним блокуванням інтерпретатора (GIL), високорівневі структури даних. Підтримується розбиття програм на модулі, які можуть об'єднуватися в пакети [6].

Для більш легкого порівняння нижче представлена таблиця 2.1 з коротким описом характеристик кожної мови.

Таблиця 2.1 – Характеристики мов програмування

Характеристика	C++	C#	Python
Тип системи	Статична	Статична	Динамічна
Синтаксис	Складний	Середній	Легкий
Парадигми	Процедурне, об'єктно-орієнтоване, загальне	Об'єктно-орієнтоване, компонентно-орієнтоване	Об'єктно-орієнтоване, компонентно-орієнтоване
Засосування	Системне програмування, ігри, ресурсозатратні додатки	Веброзробка, десктопні додатки, ігри	Веброзробка, наукові обчислення, автоматизація, штучний інтелект
Кроссплатформенність	Ні	Так	Так
Продуктивність	Висока	Висока	Середня
Рівень абстракції	Низький	Високий	Високий
Обробка помилок	Ручна	Винятки	Винятки, твердження

Для програмування застосунку більш за все підходить мова програмування C#, вона відповідає усім потрібним критеріям та має найдоступніший інструментарій для розробки графічного інтерфейсу.

2.2 Вибір СКБД

Microsoft Access – найбільш потужна реляційна повнофункціональна настільна СКБД, за допомогою якої можна не тільки будувати і

супроводжувати БД, але й створювати високоякісний інтерфейс користувача, що переводить БД на більш якісний рівень – рівень застосування. За допомогою СКБД Access зручно створювати клієнтську частину клієнт-серверної інформаційної системи, причому спочатку можна створювати тільки клієнтську частину і вже після її випробування і налагодження розділити застосування на серверну і клієнтську частини [7].

MySQL Workbench – це візуальне проєктування та моделювання баз даних інструмент доступу для MySQL серверна реляційна база даних. Це полегшує створення нових фізичних моделей даних та модифікацію існуючих. MySQL бази даних з функціями зворотного/прямого проєктування та управління змінами. Мета MySQL Workbench має надати інтерфейс для більш простої та структурованої роботи з базами даних [8].

Характеристики представлених вище СКБД наведені у таблиці 2.2.

Таблиця 2.2 – Характеристики СКБД

Характеристика	MS Access	MySQL Workbench
Тип	База даних на основі файлів	Клієнт-серверна база даних
Ліцензія	Пропрієтарна	Відкрита
Операційна система	Windows	Windows, Linux, MacOS
Мова запитів	SQL, Query Design	SQL
Продуктивність	Середня	Висока
Безпека	Низька	Висока
Резервне копіювання	Основне	Розширене

З таблиці можна зрозуміти, що якоїсь великої різниці між представленими СКБД немає, але для проєкту треба клієнт-серверна база даних тому, що користувач буде отримувати по своєму запиту інформацію з бази даних.

2.3 Вибір середовища розробки

Qt Creator — інтегроване середовище розробки, призначене для створення крос-платформових застосунків з використанням бібліотеки Qt. Підтримується розробка як класичних програм мовою C++, так і використання мови QML, для визначення сценаріїв, в якій використовується JavaScript, а структура і параметри елементів інтерфейсу задаються CSS-подібними блоками. Qt Creator може використовувати GCC або Microsoft VC++ як компілятор і GDB як зневаджувач. Для Windows версій бібліотека комплектується компілятором, заголовними і об'єктними файлами MinGW [9].

Інтегроване середовище розробки Visual Studio є творчим стартовим майданчиком, який можна використовувати для редагування, налагодження та складання коду, а також для публікації програми. На додаток до стандартного редактора та відладчика, що надаються більшістю інтегрованих середовищ розробки, Visual Studio включає компілятори, засоби завершення коду, графічні конструктори та багато інших функцій для покращення процесу розробки програмного забезпечення [10].

Visual Studio Code – це спрощений, але потужний редактор вихідного коду, який працює на комп'ютері та доступний для Windows, MacOS та Linux. Має вбудовану підтримку JavaScript, TypeScript та Node.js, а також велику екосистему розширень для інших мов та середовищ виконання (наприклад, C++, C#, Java, Python, PHP, Go, .NET) [11].

Характеристики кожного середовища розробки вказано в таблиці 2.3.

Таблиця 2.3 – Характеристики середовищ розробки

Характеристика	Qt Creator	Visual Studio	VS Code
Тип	IDE	IDE	Редактор коду
Підтримувані мови програмування	C++, QML, Python, JavaScript	C++, C#, JavaScript, Python	Багато мов програмування, зокрема C++, C#, JavaScript, Python
Підтримувані операційні системи	Windows, Linux, macOS	Windows	Windows, Linux, macOS
Інструментарій розробки графічного інтерфейсу	Qt Designer	Visual Designer	Розширені можливості для роботи з UI, але не такі розгалужені як у Qt або VS
Інтеграція з системами керування проєктами	Системи керування версіями Git, Subversion	Системи керування версіями Git, Team Foundation Server	Системи керування версіями Git
Відладка	Вбудована в IDE	Вбудована в IDE	Встановлюється окремим модулем
Ліцензія	Безкоштовна	Платна	Безкоштовна

Для розробки проєкту потрібно, щоб інтегроване середовище розробки підтримувало розробку на мові програмування C#, мало інструментарій для розробки графічного інтрефейсу, містило вбудований компілятор та мало додатковий інструментарій для облегшення підключення бази даних до проєкту. Усім зазначеним вимогам відповідає Visual Studio.

2.4 Висновки за розділом 2

У другому розділі було обрано мову програмування, яка відповідає всім необхідним критеріям, а саме мова програмування C#.

Також було обрано найбільш зручне та багатофункціональне середовище розробки для обраної мови програмування, яким є Visual Studio від Microsoft.

У якості СКБД була вибрана MySQL Workbench, яка має клієнт-серверну архітектуру бази даних.

3 ПРОЄКТУВАННЯ ЗАСТОСУНКУ

3.1 Функціональні вимоги до програми та вимоги до інтерфейсу

Визначемо основні характеристики та функції, які повинно забезпечувати програмне забезпечення для тренувань, щоб забезпечити ефективні та безпечні тренування для користувачів:

- реєстрація – програма має клієнт-серверну базу даних і для того щоб ідентифікувати користувача в програмі він повинен зареєструватися та підтвердити свій обліковий запис шляхом вводу коду відправленого на введenu пошту;
- авторизація – після реєстрації користувач повинен мати змогу увійти у свій обліковий запис шляхом введення логіну та паролю;
- введення та обробка даних – користувач має змогу вводити свої дані такі як: зріст, вага, вік. На основі введених даних програма розраховує характеристики користувача;
- статистика тренувань – користувач має змогу відстежувати прогрес виконання плану шляхом відстеження характеристик;
- надання інформації – застосунок повинен давати всю необхідну інформацію, для безпечного та ефективного тренування, включно із технікою тренування.

Графічний інтерфейс є ключовим елементом, який забезпечує зручну та ефективну роботу користувачів з програмним забезпеченням:

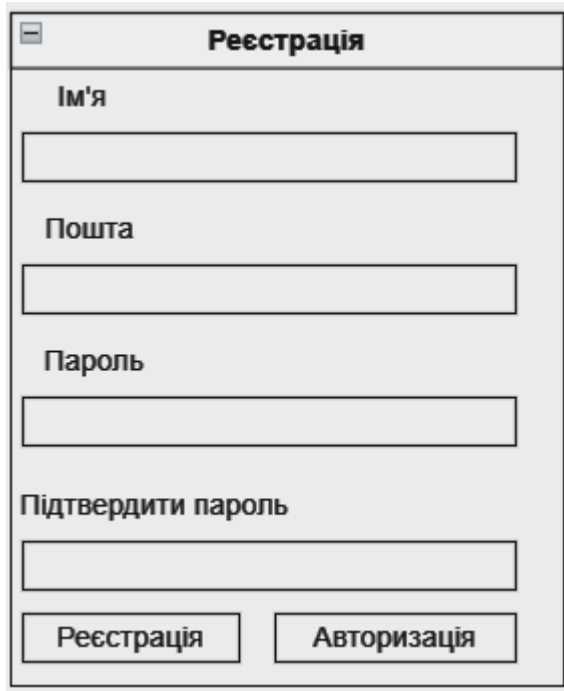
- простота та легкість у використанні – інтерфейс повинен мати низький поріг входу, так у користувача не відпаде бажання почати тренування;
- графічний інтерфейс повинен мати легку навігаційну структуру, щоб користувачу було просто знайти потрібну форму, у розроблювальному застосунку використовується послідовний інтерфейс.

Більш Детальний опис функціональних вимог наведено у Додатку А

3.2 Проектування інтерфейсу

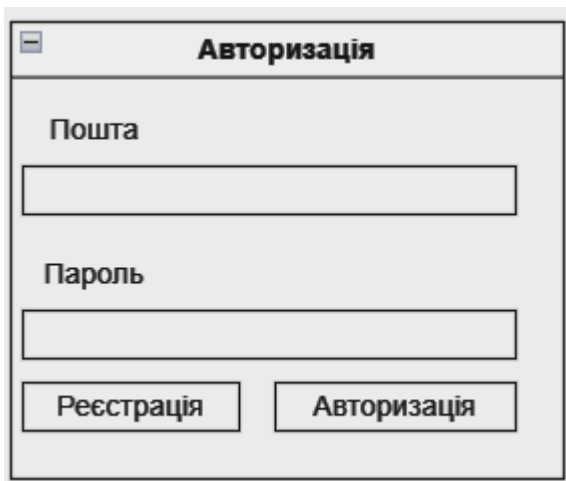
Було розроблено 5 основних форм інтерфейсу.

На рисунках 3.1 та 3.2 можна побачити форми реєстрації та авторизації.



The image shows a window titled "Реєстрація" (Registration). It contains four text input fields labeled "Ім'я" (Name), "Пошта" (Email), "Пароль" (Password), and "Підтвердити пароль" (Confirm password). Below the fields are two buttons: "Реєстрація" (Registration) and "Авторизація" (Authorization).

Рисунок 3.1 – Макет «Реєстрація»



The image shows a window titled "Авторизація" (Authorization). It contains two text input fields labeled "Пошта" (Email) and "Пароль" (Password). Below the fields are two buttons: "Реєстрація" (Registration) and "Авторизація" (Authorization).

Рисунок 3.2 – Макет «Авторизація»

Макет головного меню можна побачити на рисунку 3.3.

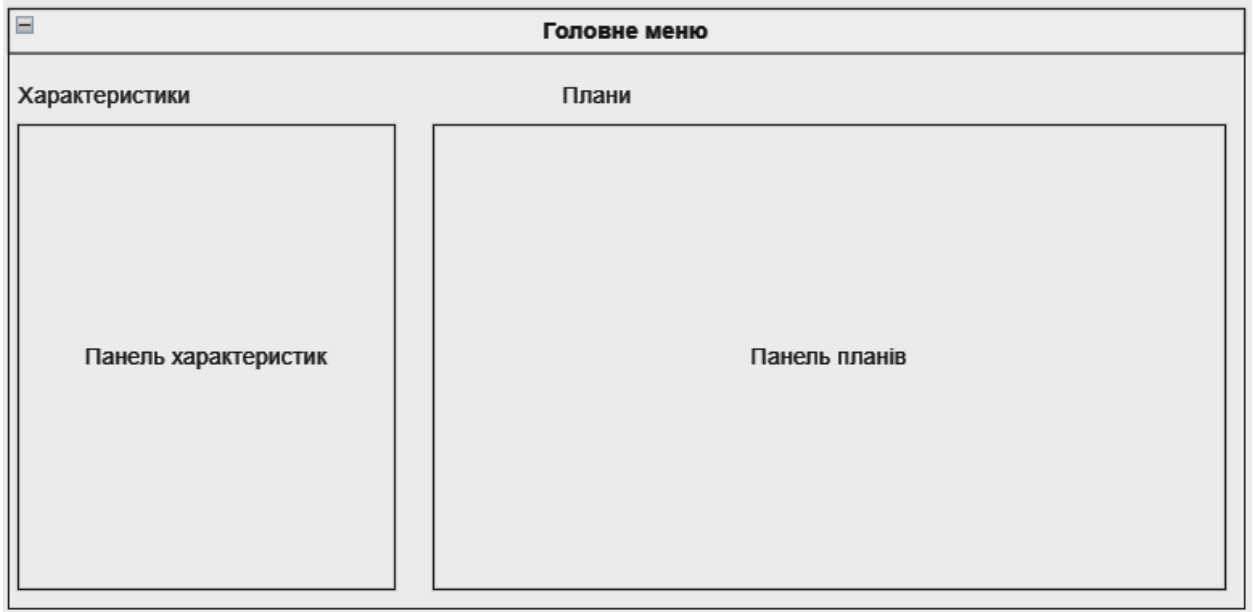


Рисунок 3.3 – Макет «Головного меню»

Макет форми «Опис плану» представлено на рисунку 3.4



Рисунок 3.4 – Макет форми «Опис плану»

Макет форми «Виконання завдання» представлено на рисунку 3.5.

Макет форми «Виконання завдання» має наступну структуру:

- Заголовок форми: **Виконання завдання**
- Поле вводу: **Назва вправи**
- Велика панель: **Панель опису виконання вправи**
- Поле вводу: **Кількість або час виконання вправи**
- Кнопка: **Перехід до наступної вправи**

Рисунок 3.5 – Макет форми «Виконання завдання»

3.3 Структура навігації

Програмне забезпечення має 5 основних форм та 2 допоміжні.

Основні форми:

- form1 – Форма реєстрації де користувач вводить: ім'я, логін та пароль;
- authorization – Форма авторизації де користувач вводить логін та пароль для входу до облікового запису;
- mainForm – головна форма де користувач може отримати характеристики, подивитися плани доступні плани тренувань, почати або продовжити виконувати вже отриманий план;
- planForm – форма опису плану тренувань, користувач може перейти на неї шляхом натиснення на один з планів серед списку доступних у правій частині головної форми;

– activityForm – форма виконання завдання, на неї користувач переходить на початку виконання плану, на цій формі користувач покроково проходить день плану де йому виводиться на екран назву вправи, опис виконання, кількість виконання.

Допоміжні форми:

– confirmForm – форма підтвердження, де користувач може ввести код, який програма відправила на пошту для підтвердження облікового запису.

Ця форма іде допоміжною до форми реєстрації(Form1);

– haracterForm – форма вводу характеристик, де користувач вводить дані для отримання карти характеристик. Ця форма іде допоміжною до головної форми і доступна тільки один раз при відсутності карти характеристик.

Нижче представлена діаграма навігації по інтерфейсу (рис. 3.6).

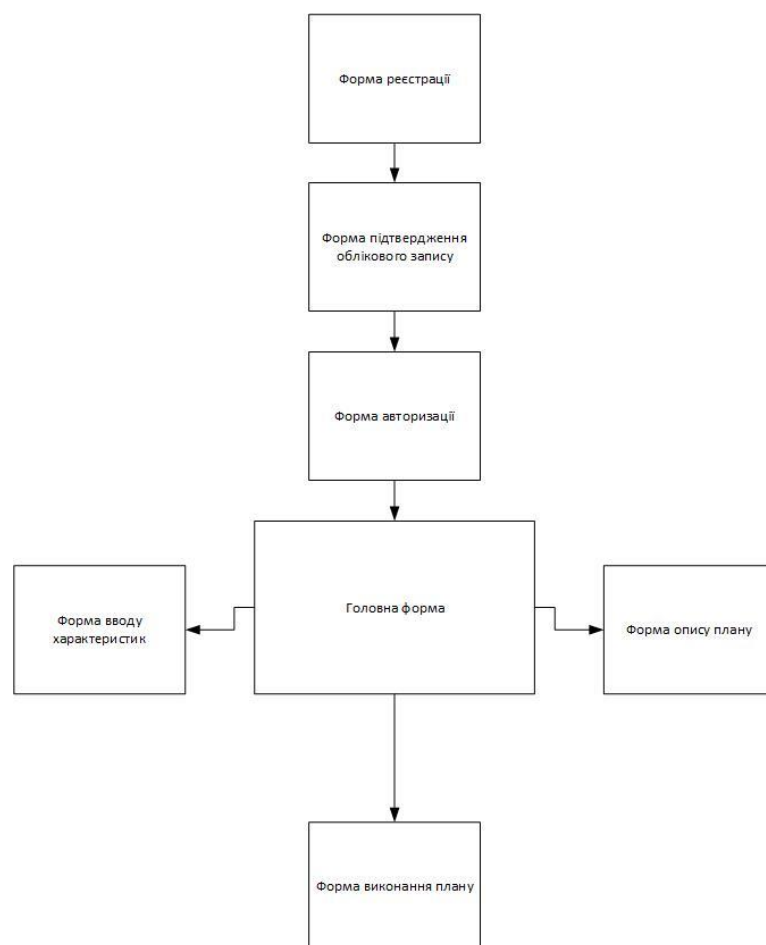


Рисунок 3.6 – Діаграма навігації по інтерфейсу

3.4 Головний алгоритм роботи програми

На рисунку 3.7 представлена функціональна структура програми.

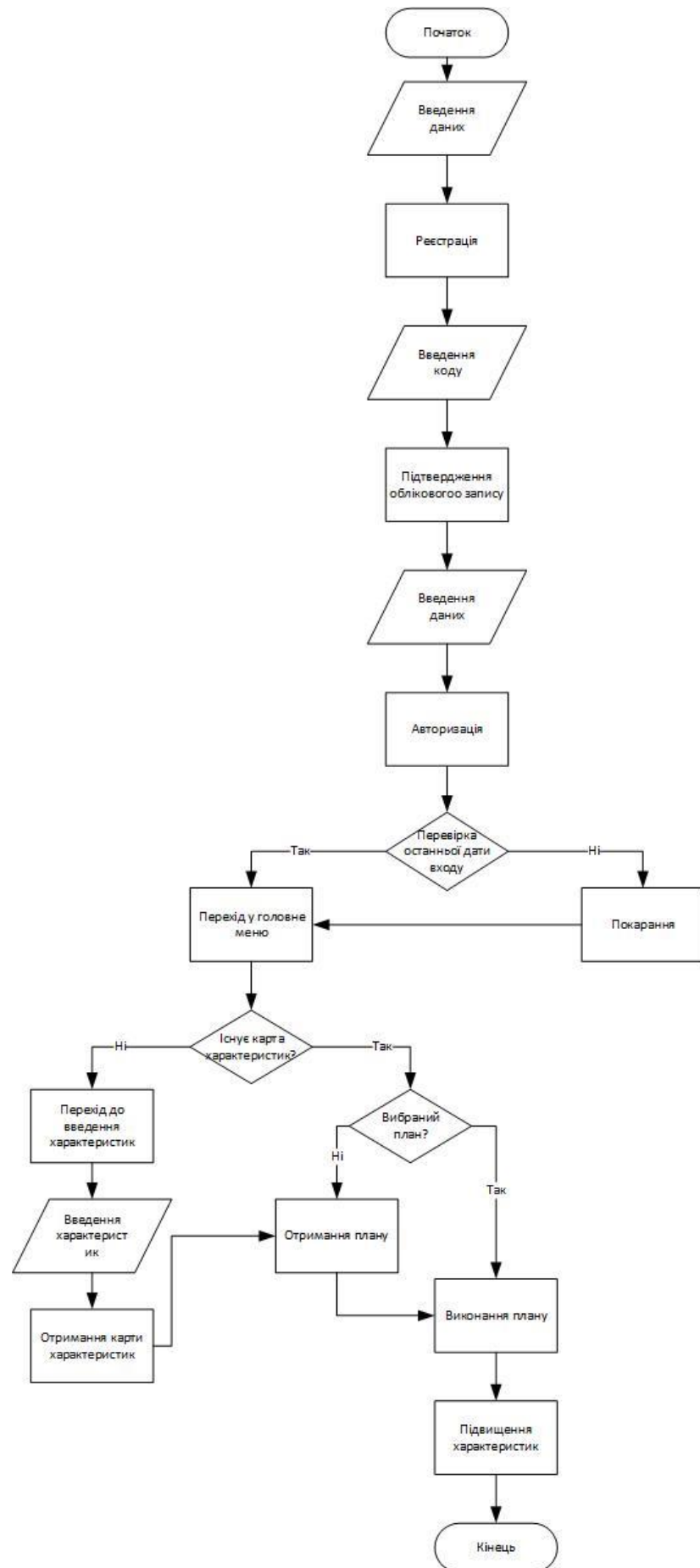


Рисунок 3.7 – Функціональна структура програми

3.5 Діаграма прецедент

На рисунку 3.8 представлена діаграма прецедентів.

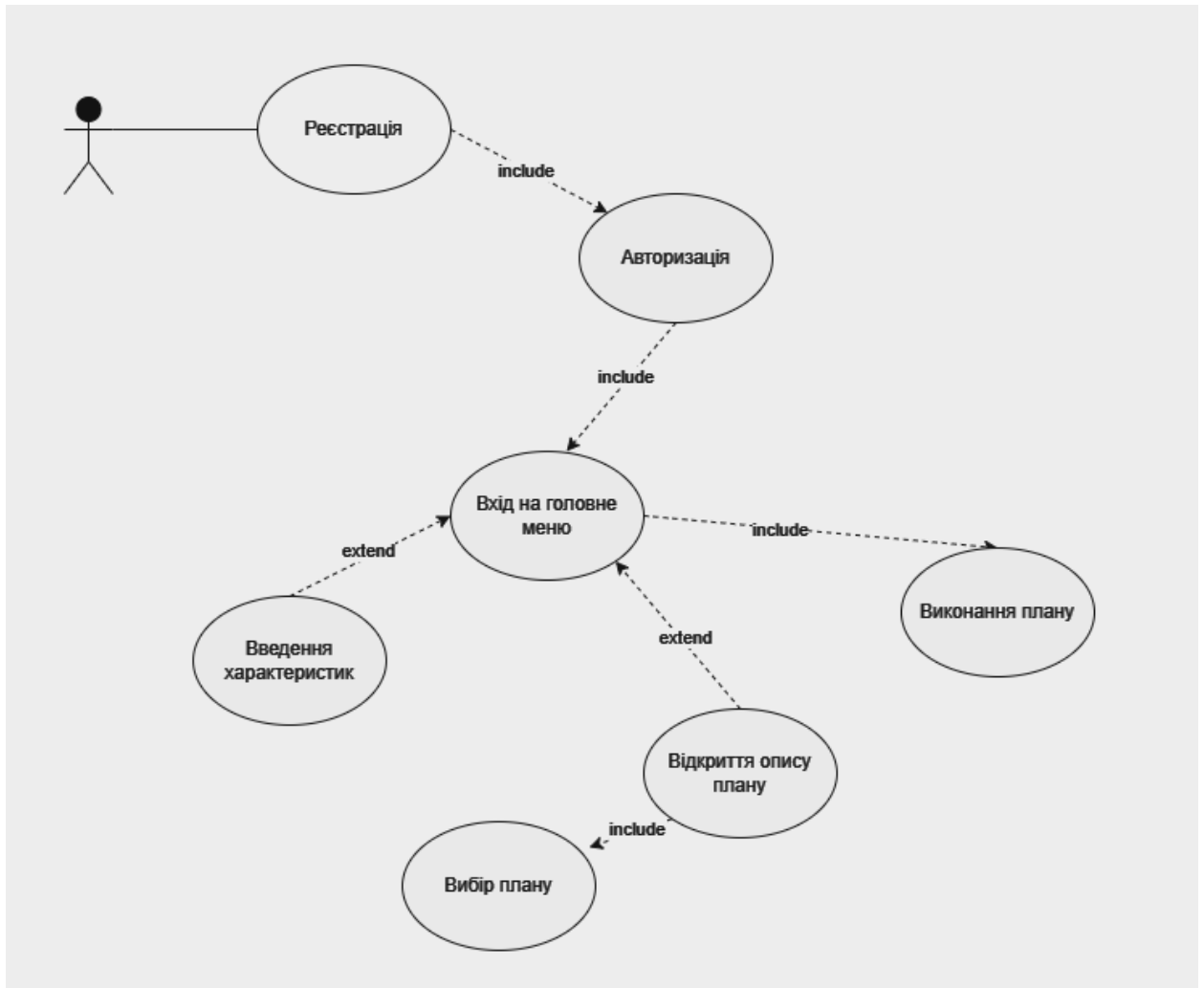


Рисунок 3.8 – Діаграма прецедентів

3.6 Діаграма розгортання

Діаграму розгортання можна побачити на рисунку 3.9.

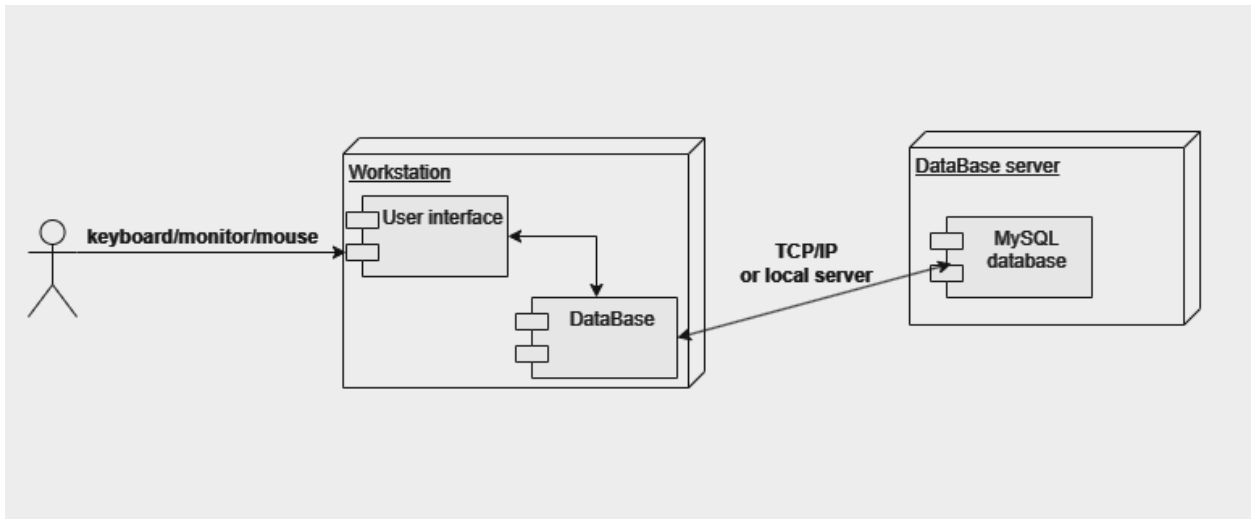


Рисунок 3.9 – Діаграма розгортання

3.7 ER діаграма бази даних

ER модель бази даних представлена на рисунку 3.10.

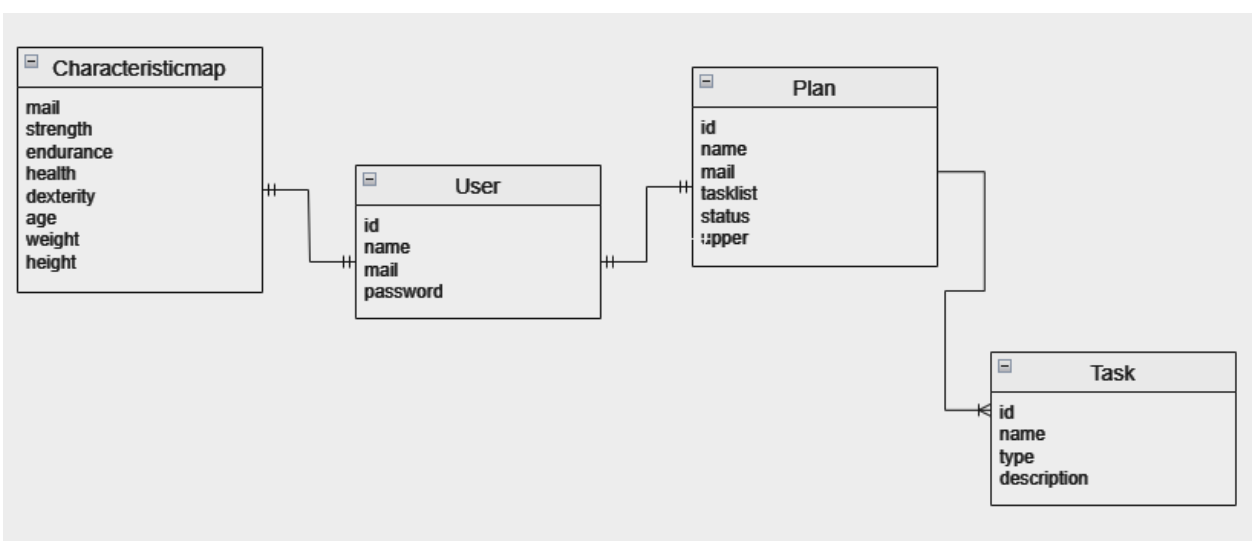


Рисунок 3.10 – ER модель

3.8 Висновки за розділом 3

У даному розділі були визначені функціональні вимоги, спроектовані макети графічного інтерфейсу, описана структура навігації по застосунку. Також були розроблені діаграми: діграма навігації по інтерфейсу, функціональна структура програми, діаграма прецедентів, ER модель бази даних.

4 ОПИС ПРОГРАМИ

4.1 Опис класів та функцій

Було розроблено 9 класів та 5 функцій для управління елементами інтерфейсу та виконання операцій. Більш детальне описання класів їх методів та функцій представлено нижче.

4.1.1 Клас DataBase

Даний клас призначений для взаємодії з базою даних. В основному цей клас та його методи призначені для усіх інших методів. Клас DataBase має такі методи:

- addUser – метод додання облікового запису користувача до бази даних;
- Addharacteristicmap – метод додання карти характеристик користувача до БД;
- DeleteUserById – видалення користувача з БД за його айді;
- findUser – метод відправляє запит до бази даних на отримання даних користувача за його логіном;
- getHaracteristics – метод відправляє запит до бази данх на отримання карти характеристик користувача за його поштою;
- UpdateUserPlan – метод який оновлює вибраний план користувачем у базі даних;
- UpdatePlanStatus – метод який оновлює статус виконання плану користувачем;
- getOneHaracter – допоміжний метод для оновлення характеристик користувача, через запит до БД повертає необхідну характеристику;
- UpdateHaracteristic – метод оновлення характеристики після прогресу виконання плану;

- Punishment – метод покарання, перевіряє дату останнього заходу користувача, якщо користувач пропустив потрібну дату метод зменшує кількість здоров'я користувача і оновлює їх кількість у базі даних;
- getPlan – метод через запит до БД повертає план, який закріплений за користувачем;
- getPlanById – метод через запит отримує план за його айді;
- getTaskById - метод через запит отримує завдання за його айді;
- getPlans – метод через запит отримує з БД усі незайняті плани тренувань.

4.1.2 Клас Auth

Клас для взаємодії з формою авторизації. Використовує клас DataBase для отримання даних з бази даних та авторизовує користувача у системі. Клас має тільки один метод:

- Authorize – метод, який використовуючи введені користувачем логін та пароль отримує дані з бази даних, порівнює порівнює їх і дає доступ до облікового запису або ні.

4.1.3 Клас registration

Клас для взаємодії з формою реєстрації. Використовує клас DataBase для занесення даних до БД. Клас має тільки один метод:

- Register – метод, який після вводу даних користувачем і їх підтвердження, вносить дані до бази даних, що потім дозволяє користувачу авторизуватися у своєму обліковому записі.

4.1.4 Клас Haracteristics

Клас, який містить у собі поля для заповнення параметрів карти характеристик користувача, призначений лише для структурування даних отриманих з бази даних, тому не має методів.

4.1.5 Клас ManageUser

Клас для управління входами користувача у застосунок.

Має такі методи:

- lastDateofEnter – метод, який записує у файл останню дату входу в обліковий запис;
- checkDate – метод, який перевіряє дату останнього заходу користувача та повертає true, якщо користувач вчасно увійшов в обліковий запис, та повертає false, якщо користувач увійшов невчасно. Метод допомагає методу punishment, який знаходиться в класі DataBase, зрозуміти чи треба застосовувати покарання до користувача.

4.1.6 Клас Plan

Клас, який містить у собі поля для заповнення параметрів плану, призначений лише для структурування даних отриманих з бази даних, тому не має методів.

4.1.7 Клас SendMail

Клас для взаємодії з поштою користувача.

Має один метод

– `Sendletter` – метод, який генерує шестизначний код повертає його при визові методу та відправляє його на пошту введену користувачем.

4.1.8 Клас Task

Клас, який містить у собі поля для заповнення параметрів завдання, призначений лише для структурування даних отриманих з бази даних, тому не має методів.

4.1.9 Клас User

Клас, який містить у собі поля для заповнення параметрів користувача, призначений лише для структурування даних отриманих з бази даних, тому не має методів.

4.1.10 Функція planStep

Функція написана для форми виконання завдання(`ActivityForm`), призначена для оновлення даних елементів інтерфейсу при кожному кроці виконання плану. При кожному переході на наступний крок плану, функція заповнює елементи графічного інтерфейсу такими даними:

- назва завдання;
- опис виконання завдання;
- кількість виконання завдання.

4.1.11 Функція getStatus

Функція на головній формі, яка повертає статус виконання плану який записаний в БД. Функція потрібна для виведення користувачу етапу плану на якому він зупинився.

4.1.12 Функція refreshPlan

Функція на головній формі, яка оновлює панель плану і показує користувачу на якому етапі плану він знаходиться або якщо користувач ще не вибрав план, функція виводить йому усі доступні плани.

4.1.13 Функція refreshPanel

Функція для головної форми, яка оновлює дані про характеристики і виводить на форму для користувача. Виконується кожен раз при оновленні характеристик.

4.1.14 Функція fullPanel

Функція для форми опису плану, яка отримує дані з бази даних і виводить опис плану та його завдань для користувача у виді панелі з структурованими даними.

4.1.15 Діаграма класів

Діаграму класів можна побачити на рисунку 4.1.

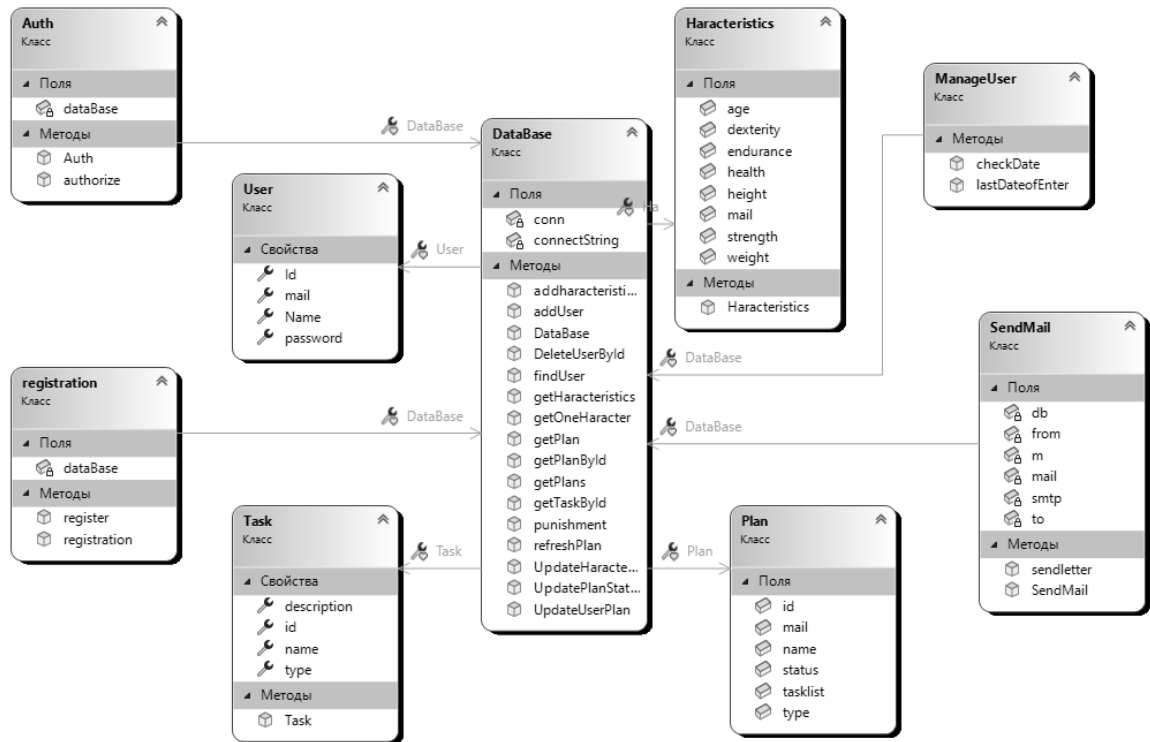


Рисунок 4.1 – Діаграма класів

4.2 Висновки за розділом 4

У цьому розділі були описані класи їх методи та функції необхідні для роботи програми та була розроблена діаграма класів.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРЕМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Опис роботи програми

Для того щоб запустити програму, її треба завантажити та встановити, що показано на рисунку 5.1

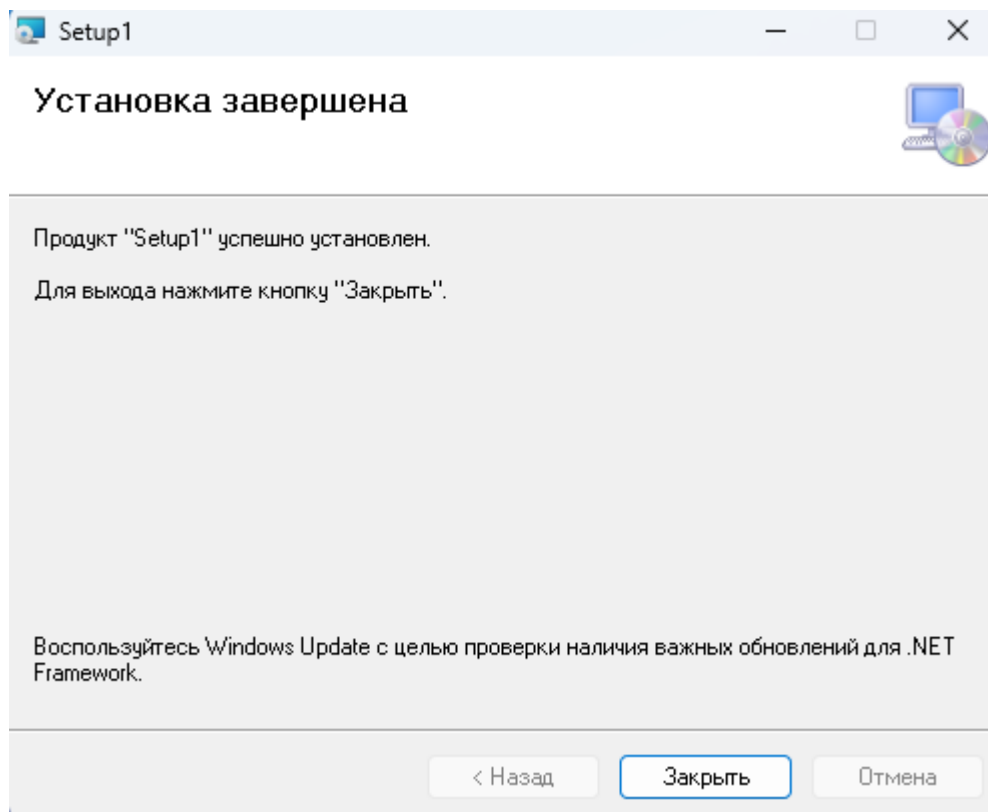


Рисунок 5.1 – Встановлення програми

Після установки ярлик програми з'явиться на робочому столі (рис. 5.2).

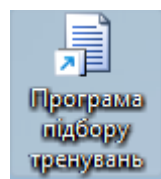


Рисунок 5.2 – Ярлик програми

При першому запуску програми треба пройти реєстрацію (рис. 5.3) та підтвердити пошту (рис. 5.4)(рис. 5.5).

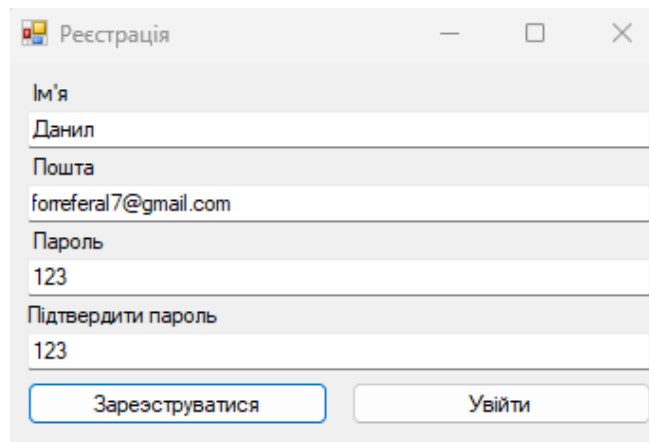


Рисунок 5.3 – Реєстрація користувача



Рисунок 5.4 – Код підтвердження

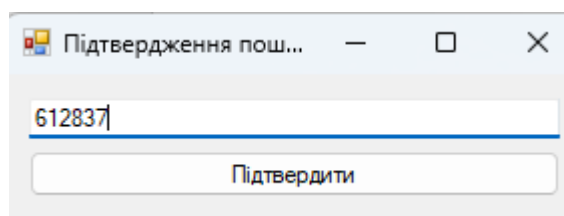


Рисунок 5.5 – Підтвердження пошти

Авторизацію показано на рисунку 5.6.

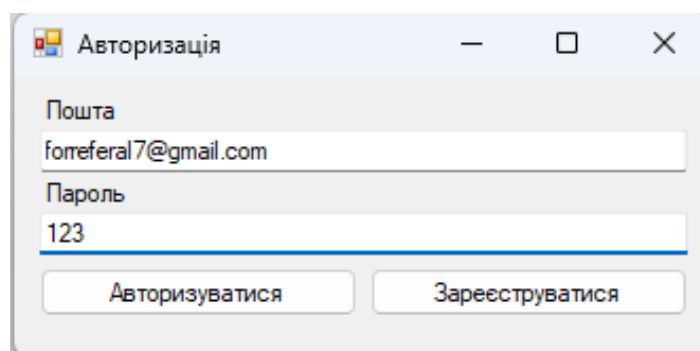


Рисунок 5.6 – Авторизація

Після входу у застосунок треба отримати карту характеристик, для введення своїх даних треба натиснути кнопку «Отримати характеристики» (рис. 5.7).

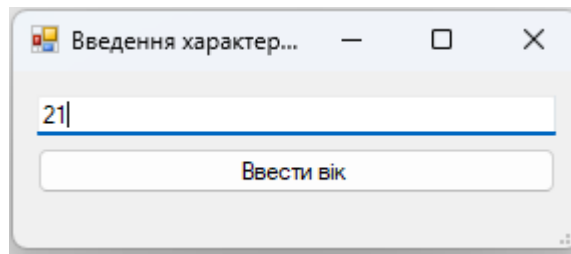


Рисунок 5.7 – Введення характеристик

Після введення даних програма розрахує характеристики користувача і виведе їх на панель для користувача (рис. 5.8).

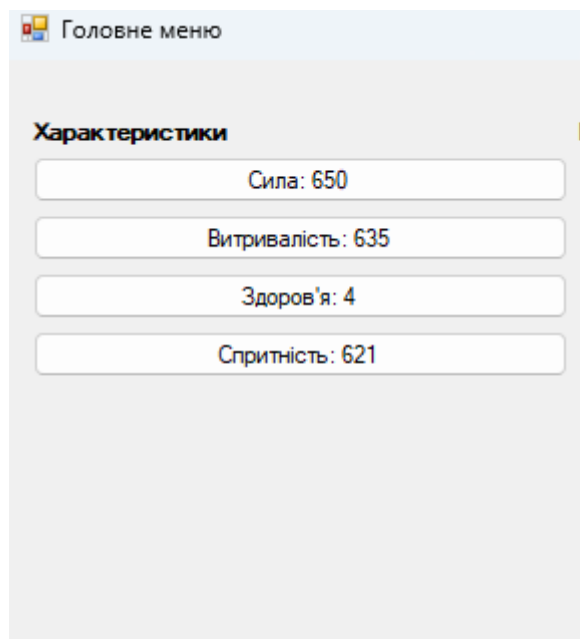
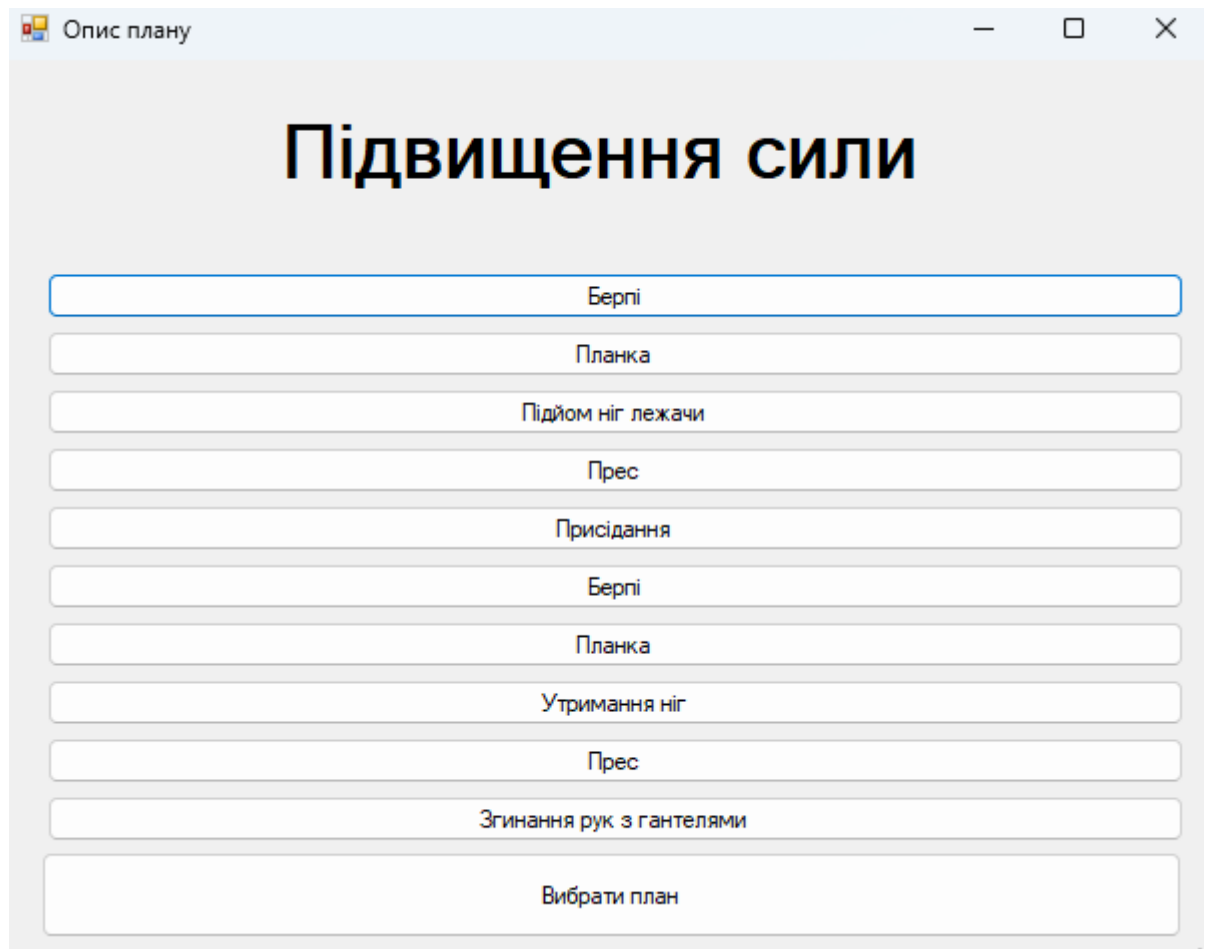


Рисунок 5.8 – Характеристики користувача

Далі можна вибрати план та натиснувши на нього подивитись з чого він складається (рис. 5.9).



The screenshot shows a window titled "Опис плану" (Plan Description) with a light blue header bar. The main content area has a light gray background and a large, bold, black title "Підвищення сили" (Increase strength). Below the title is a vertical list of ten white rectangular buttons with rounded corners and thin gray borders. The buttons are labeled with the following exercises in Ukrainian: "Берпі" (Burpees), "Планка" (Plank), "Підйом ніг лежачи" (Leg lift lying down), "Прес" (Press), "Присідання" (Squats), "Берпі" (Burpees), "Планка" (Plank), "Утримання ніг" (Leg hold), "Прес" (Press), and "Згинання рук з гантелями" (Bicep curls with dumbbells). At the bottom of the list is a larger white button with rounded corners and a thin gray border, labeled "Вибрати план" (Select plan).

Вправа
Берпі
Планка
Підйом ніг лежачи
Прес
Присідання
Берпі
Планка
Утримання ніг
Прес
Згинання рук з гантелями
Вибрати план

Рисунок 5.9 – Опис вправ плану тренувань

Тепер можна вибрати план і він буде закріплений за користувачем або закрити вікно і активувати вікно іншого плану.

У нашому випадку ми візьмемо план який представлений вище, після переходу у головне меню можна буде почати виконувати план (рис. 5.10).

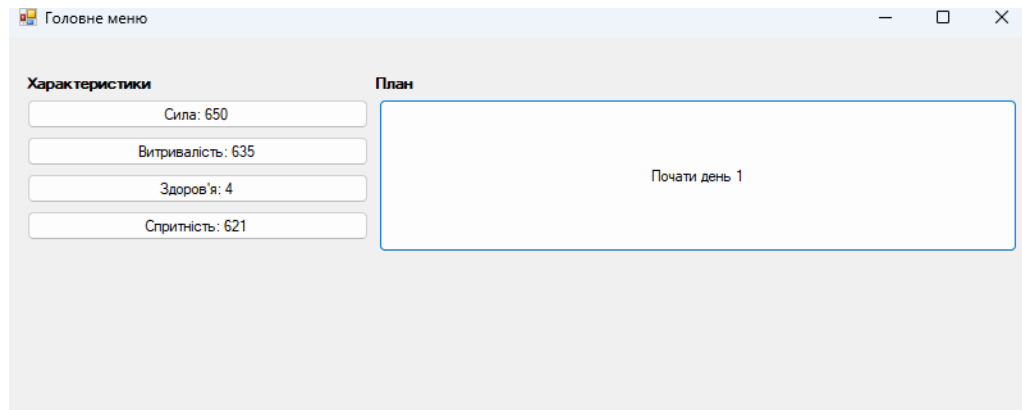


Рисунок 5.10 – Головне меню

Приклад вкладки завдання представлено на рисунку 5.11.

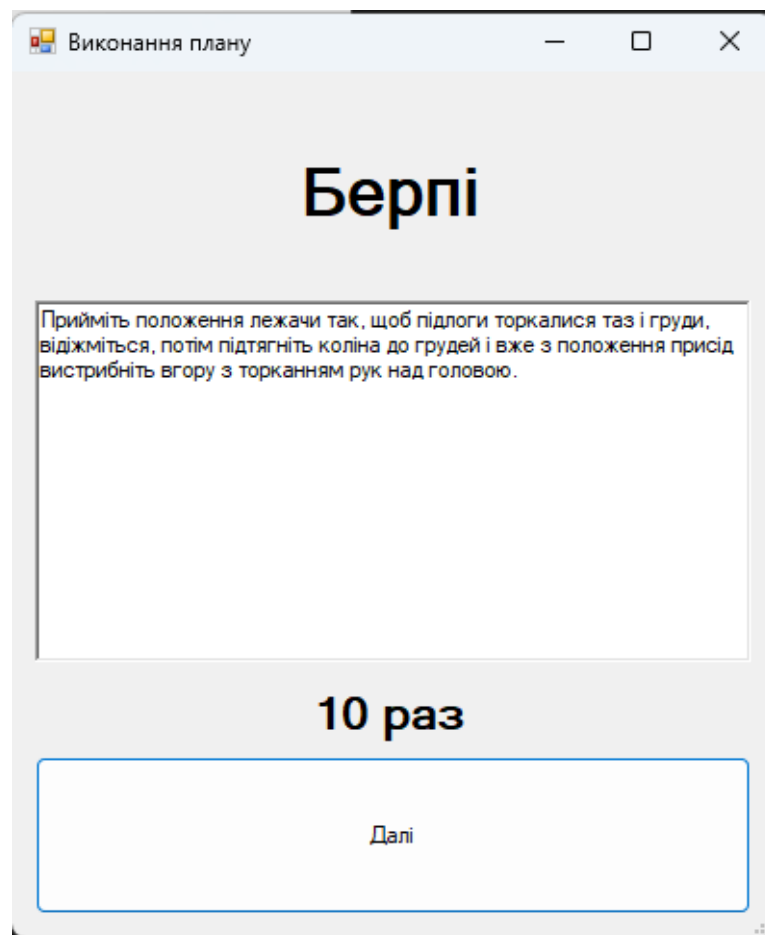


Рисунок 5.11 – Вікно виконання завдання

При виконанні плану оновлюється статус та підвищуються характеристики. На рисунку 5.12 показані характеристики після виконання одного тренування їх можна порівняти з характеристиками на рисунку 5.10.

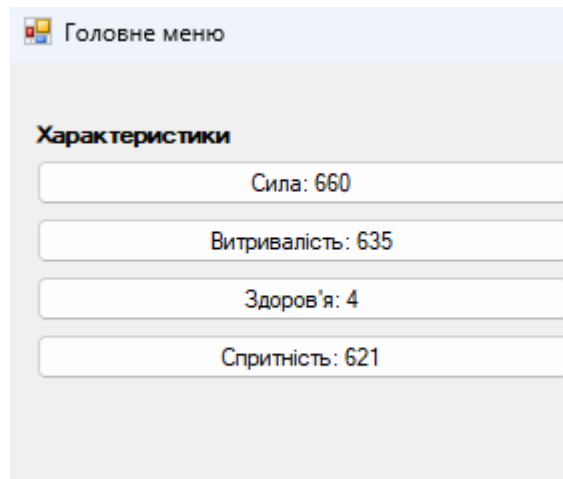


Рисунок 5.12 – Оновлені характеристики

5.2 Тестування програми

Перше що було протестовано це ввід коректної пошти та паролю при реєстрації, як програма реагує показано на рисунках 5.13 та 5.14.

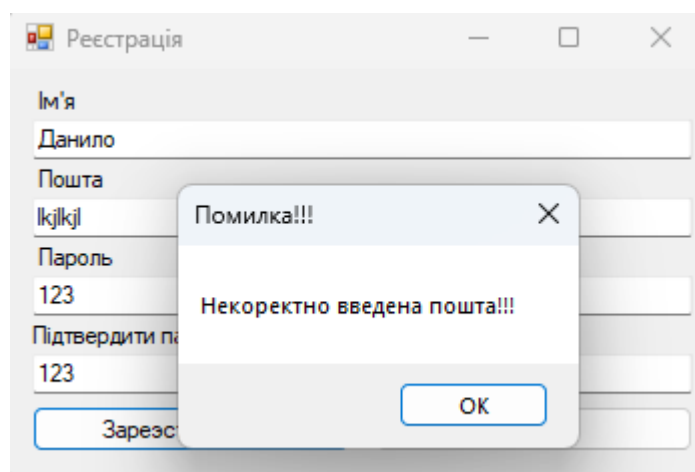


Рисунок 5.13 – Некоректно введена пошта

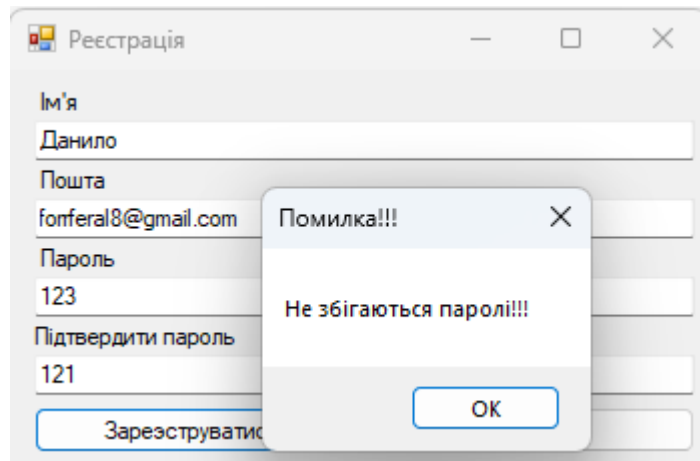


Рисунок 5.14 – Некоректне підтвердження пароля

Тестування програми на введення пошти, під якою вже зареєстрований користувач показано на рисунку 5.15.

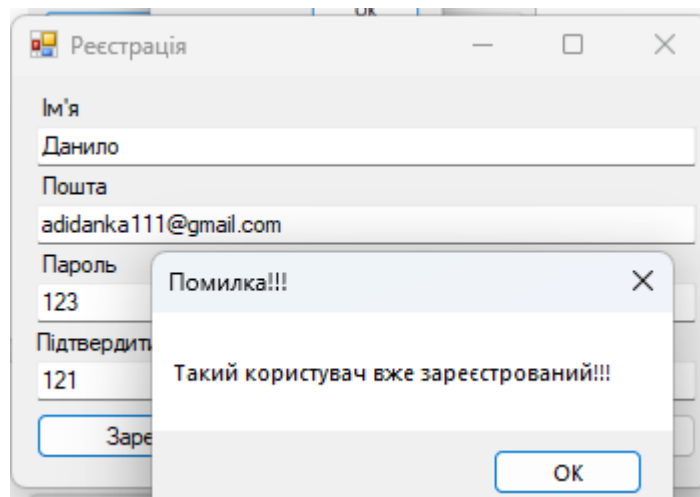


Рисунок 5.15 – Введена вже зареєстрована пошта

Було проведене тестування авторизації на ввід некоректної пошти або неправильного паролю, результати можна побачити на рисунках 5.16 та 5.17.

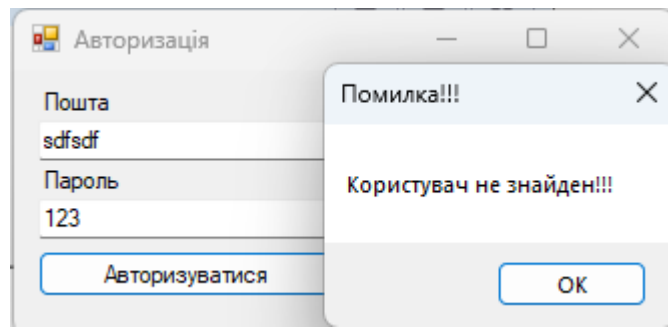


Рисунок 5.16 – Некоректно введена пошта

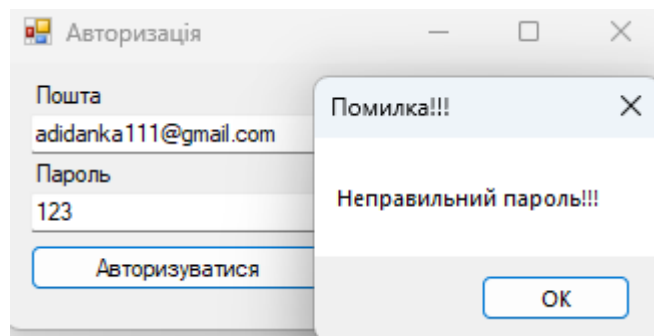


Рисунок 5.17 – Неправильно введений пароль

Тестування на введення коректних характеристик показано на рисунку 5.18.

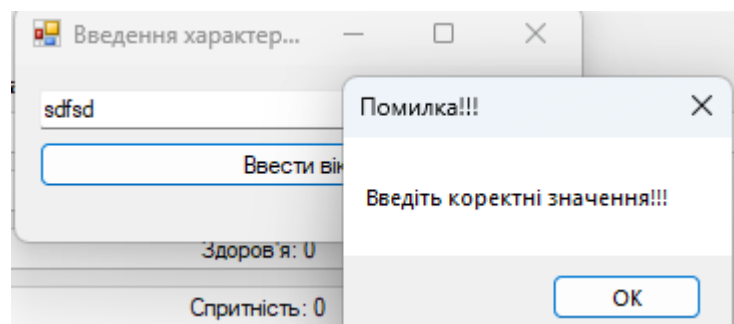


Рисунок 5.18 – Некоректне введення характеристик

Було штучно змінено дату останнього заходу у застосунок, щоб показати покарання у виді віднімання однієї одиниці характеристики «здоров'я» у користувача (рис. 5.19).

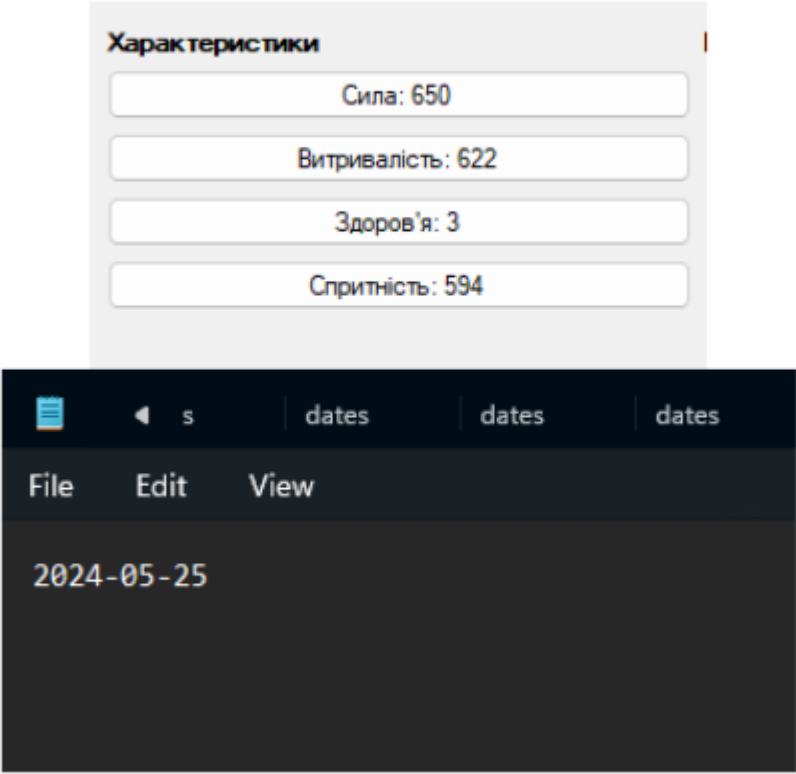


Рисунок 5.19 – Покарання за пропуск плану

Результати тестування програмного забезпечення представлені у таблиці 5.1. Як видно з таблиці всі тести пройдено успішно.

Таблиця 5.1 – Результати тестування ПЗ

№	Назва	Дії	Очікуваний результат	Фактичний результат
1	Реєстрація з валідними даними.	1. Введення необхідних даних: Данило, adidanka111@gmail.com, 123, 123. 2. Натиснути кнопку реєстрації	Вивід форми підтвердження пошти та письмо на пошті.	Вивід форми підтвердження пошти та письмо на пошті.
2	Реєстрація з невалідними даними у полі пошта.	1. Введення необхідних даних: Данило, adidanka111il.com, 123, 123 2. Натиснути кнопку реєстрації	Сповіщення про помилку некоректно введеної пошти.	Сповіщення про помилку некоректно введеної пошти.
3	Реєстрація з невалідними даними у полі пароль.	1. Введення необхідних даних: Данило, adidanka111@gmail.com, 123, 123. 2. Натиснути кнопку реєстрації	Сповіщення про помилку несбігання паролів.	Сповіщення про помилку несбігання паролів.

Продовження таблиці 5.1

4	Реєстрація вже зареєстрованого користувача.	1. Введення необхідних даних: Данило, adidanka111@gmail.com, 123, 123. 2. Натиснути кнопку реєстрації	Сповіщення про помилку, що такий користувач вже є.	Сповіщення про помилку, що такий користувач вже є.
5	Авторизація незареєстрованого користувача.	1. Введення необхідних даних: adidanka111@gmail.com, 123. 2. Натиснути кнопку авторизації	Сповіщення про помилку, що такий користувач не знайдений.	Сповіщення про помилку, що такий користувач не знайдений.
6	Авторизація з неправильним паролем.	1. Введення необхідних даних: adidanka111@gmail.com, 12. 2. Натиснути кнопку	Сповіщення про введення неправильно го пароля.	Сповіщення про введення неправильно го пароля.
7	Вибір плану не завівши карту характеристик	1. Відкриття опису плану	Кнопка підтвердження плану недоступна.	Кнопка підтвердження плану недоступна.

Продовження таблиці 5.1

8	Вибір плану.	1. Відкриття опису плану.	Кнопка підтвердження плану доступна.	Кнопка підтвердження плану доступна..
9	Введення коректних даних для карти характеристик .	1. Натиснути кнопку «Отримати характеристики» 2. Ввести необхідні дані:21,60,177.	На головному меню з'явиться карта характеристик.	На головному меню з'явиться карта характеристик.
10	Введення некоректних даних для карти характеристик .	1. Ввести необхідні дані:фівф. 2. Натиснути кнопку «далі»	Сповіщення про введення некоректних даних.	Сповіщення про введення некоректних даних.

5.3 Висновки за розділом 5

У п'ятому розділі було описано, як працювати з програмою. Було проведено тестування і визначено, як програмний застосунок реагує на відхилення від основного алгоритму дій.

ВИСНОВОК

Для максимально ефективного підбору плану тренування у домашніх умовах необхідні деякі знання, тому актуальна розробка програмного застосунку з підбору плану тренувань.

Було проаналізовано існуючі застосунки для тренувань, мови програмування, інтегровані середовища розробки, системи керування базами даних. Обрано мову програмування – C#, IDE – Visual Studio, СКБД – MySQL Workbench.

Аналіз існуючих застосунків дозволив визначити функціональні вимоги та вимоги до графічного інтерфейсу програми, вимоги до складу параметрів технічного засобу, де було обрано платформу ОС Windows. Застосунок відповідає меті та темі розробки та усім перерахованим вимогам, які були визначені.

Програмне забезпечення було протестовано не включаючи збійні ситуації. Так як це перша версія застосунку, він буде оновлюватися, отримувати нові функції та модифікуватимуться вже існуючі, наприклад:

- доповнення програм тренувань;
- збільшення видів характеристик;
- більш точний розрахунок характеристик;
- оновлення дизайну графічного інтерфейсу.

Результатом цієї роботи стало розроблений та протестований програмний застосунок для підбору тренувань у домашніх умовах

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Fitbit Mobile [Електрон. ресурс]. – Режим доступу: <https://play.google.com/store/apps/details?id=com.fitbit.FitbitMobile&hl=ru&gl=US>.
2. Fitness Buddy [Електрон. ресурс]. – Режим доступу: <https://play.google.com/store/apps/details?id=fitness.buddy.app&hl=uk>.
3. 8fit [Електрон. ресурс]. – Режим доступу: <https://play.google.com/store/apps/details?id=com.eightfit.app&hl=ru&gl=US>
4. C++ [Електрон. ресурс]. – Режим доступу: <https://ru.wikipedia.org/wiki/C%2B%2B>.
5. C# [Електрон. ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/C_Sharp.
6. Python [Електрон. ресурс]. – Режим доступу: <https://ru.wikipedia.org/wiki/Python>.
7. MS Access [Електрон. ресурс]. – Режим доступу: <https://dspace.uzhnu.edu.ua/jspui/bitstream/lib/24346/1/Microsoft%20A%D1%81%D1%81ess%202016.pdf>.
8. MySQL [Електрон. ресурс]. – Режим доступу: <https://www.guru99.com/ru/introduction-to-mysql-workbench.html>.
9. Qt Creator [Електрон. ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Qt_Creator.
10. Visual Studio [Електрон. ресурс]. – Режим доступу: <https://visualstudio.microsoft.com/ru/>.
11. VS Code [Електрон. ресурс]. – Режим доступу: <https://visualstudio.microsoft.com/ru/>.

ДОДАТОК А
Технічне завдання

A.1 Вступ

A.1.1 Найменування

Найменування – «Програмне забезпечення підбору програми тренувань».

A.1.2 Коротка характеристика області застосування

Програма призначена для використання у сфері спорту для досягнення максимальних результатів при витраті меншої кількості зусиль.

A.2 Підстави для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Реалізація програмного забезпечення для підбору індивідуального плану тренувань», затверджене наказом Національного університету «Запорізька політехніка» № 168 від 24 квітня 2024 р.

A.3 Призначення розробки

A.3.1 Функціональне призначення

Функціональним призначенням програми є надання можливості користувачу:

- реєструватися;
- авторизуватися;
- управляти параметрами свого облікового запису;
- вводити свої дані у програму;
- на основі введених даних підбирати план тренувань;
- відслідковувати прогрес виконання плану;

А.3.2 Експлуатаційне призначення

Програма має експлуатуватися, в особистих інтересах окремо взятого користувача.

А.4 Вимоги до програми чи програмного виробу

А.4.1 Вимоги до функціональних характеристик

А.4.1.1 Групи користувачів

Програмний продукт має використовуватися одним типом користувачів, а саме виконавцем, який може вводити та оновлювати свої дані та на їх основі отримувати план тренувань і відслідковувати їх прогрес виконання плану.

А.4.1.2 Вимоги до складу виконуваних функцій

Програма має задовольняти можливість виконання зазначених нижче функцій.

Для виконавця:

- а) функція введення даних. Потрібно передбачити як введення нових даних, так і оновлювати вже введенних даних;
- б) функція відстеження характеристик користувача. Потрібно передбачити змінення характеристик користувача після виконання кожного етапу виконуваного плану;
- в) функція покарання. Користувач матиме характеристику «здоров'я», яке буде втрачатися при невиконанні плану;
- г) функція реєстрації. Користувач буде вводити свої дані та підтверджувати обліковий запис за допомогою введеної пошти;

д) функція авторизації. Користувач буде входити у свій обліковий запис через пошту;

ж) функція вводу параметрів росту, ваги, віку. На основі введених параметрів користувача буде розраховуватися карта характеристик користувача.

A.4.2 Вимоги до організації вхідних даних

Вхідні дані в контексті програми це ті, які знаходяться у базі даних та початку роботи програми та ті, які вводить користувач при реєстрації на початку користування.

Вхідні дані мають бути організовані як система взаємопов'язаних характеристик користувача.

Структура «користувача» має бути наступною:

- айді;
- ім'я;
- поштова адреса;
- пароль.

Структура «завдання» має бути наступною:

- номер;
- тип;
- назва;
- опис.

Структура плану має бути наступною:

- назва;
- поштова адреса;
- список завдань;
- прогрес виконання.

Структура карти характеристик має бути наступною:

- пошта;
- сила;
- витривалість;
- спритність;
- здоров'я;
- вік;
- вага;
- зріст.

А.4.3 Вимоги до організації вихідних даних

Вихідні дані це та інформація, яку бачить користувач на екрані монітора; яка проаналізована алгоритмами та виведена за допомогою інтерфейсу. До вихідних даних має входити:

- складений план завдань;
- оцінка ефективності виконання плану у виді характеристик.

А.4.4 Вимоги до тимчасових характеристик

До тимчасових характеристик програми треба відносити структуру «План», яка формується на початку створення запиту на створення плану. У процесі виконання плану вона не може змінюватися, однак план може змінюватися після його проходження.

A.5 Вимоги до надійності

A.5.1 Вимоги до забезпечення надійного функціонування програми

Надійне функціонування програми повинно бути забезпечено користувачем у сукупності організаційно-технічних пунктів, які описані нижче:

- безперервне постачання живлення технічних пристроїв на яких буде встановлена дана програма;
- використання ліцензованого програмного забезпечення;
- регулярним виконанням перевірок на наявність вірусів.

A.5.2 Збійні ситуації

Наступні збійні ситуації мають оброблятися програмою:

- занесення до бази даних некоректних даних;
- некоректне зчитування даних з бази даних;
- некоректне формування плану тренувань;
- некоректне зчитування дати;
- некоректне введення характеристик користувача;
- відсутній зв'язок з базою даних;
- відсутній доступ до інтернету.

A.5.3 Умови експлуатації

A.5.3.1 Кліматичні умови експлуатації

Кліматичні умови експлуатації, за яких повинні виконуватися зазначені характеристики, повинні задовольняти умовам, які пред'являються до технічних засобів у частині умов їх експлуатації.

A.5.3.2 Вимоги до персоналу

Кінцевий користувач програми повинен мати навички роботи з графічним інтерфейсом. Його кваліфікація неважлива, через те, що програма призначена для використання типовим користувачем настільного персонального комп'ютера.

Користувач має ознайомитись з керівництвом. Хоч інтерфейс програми має бути інтуїтивно зрозумілим, коректне введення деяких характеристик буде можливе після ознайомлення з їх описом.

A.5.4 Вимоги до складу та параметрів технічних засобів

Для якісної роботи програми рекомендується персональний комп'ютер до складу якого входять наступні компоненти та технічні засоби:

- процесор Intel Core i3-8100 (3.6 ГГц)(Intel UHD Graphics 630) з тактовою частотою, ГГц - 3,6;
- материнська плата Asus Prime H310M-R R2.0 (s1151 v2, intel H310, PCI-Ex16);
- оперативна пам'ять DDR2 з об'ємом 4 Гб або більше;
- блок живлення 450 Вт або вище;
- об'єм логічної пам'яті SSD/HDD – 120 Гб або більше;
- інструменти вводу інформації – клавіатура та миша;
- засоби виводу інформації – монітор.

A.6 Вимоги до програмної документації

Склад програмної документації повинен включати:

- розроблене технічне завдання відповідно до теоретичних відомостей, предметної області, мети та завдань, що ставляться перед системою;
- звіт з виконаної роботи, що включає: опис проєктування системи, опис програми (функціонально-структурна схема програмного забезпечення, опис модулів з визначенням основних функцій та описом їх реалізації), опис проблем;
- керівництво користувача.

A.7 Техніко-економічні показники

Ефективність програмного забезпечення має оцінюватись в часі, який було зекономлено при плануванні. Програма не потребує жодних економічних впливань та є цілком безкоштовною. Орієнтована економічна ефективність не розраховується.

A.8 Стадії та етапи розробки

Розробка ведеться у 8 етапів:

- розробка технічного завдання (1 тиждень);
- проєктування архітектури системи (1-2 тижнів). Потрібно провести аналіз предметної області; виділити основні об'єкти, основні компоненти майбутньої системи; проаналізувати функції, які виконують виділені об'єкти предметної області. В результаті проведеного проєктування має бути отриманий робочий варіант архітектури системи, що має включати:

пакети, модулі, класи, бібліотеки, тощо, з описом призначення та функцій кожного з них;

- реалізація базових функцій системи (1-2 тижнів). Потрібно реалізувати, проаналізувати та протестувати базові функції системи;
- робота з даними (1-2 тижнів). Потрібно реалізувати роботу з даними, за допомогою БД;
- розширення функціональності програмного забезпечення (1 тиждень);
- розробка графічного інтерфейсу програмної системи (2-3 тижні).

A.9 Порядок контролю та приймання

Випробування повинен проводити розробник, у якого має бути повністю перевірений зазначений функціонал програми. Іспити мають бути проведені від особи користувача. Також мають бути перевірені усі можливі збійні ситуації, та їх обробка.

ДОДАТОК Б**Код програми**

Б.1 Код файла ActivityForm.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Numerics;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace DiplomProject
{
    public partial class ActivityForm : Form
    {
        Plan plan;
        DataBase data;
        int count;
        int id;
        int day;
        MainForm mainForm;

        public void planStep(int id)
        {
            Task task= data.getTaskById(id);
            label1.Text = task.name;
            richTextBox1.Text = task.description;
            label2.Text = task.type;
        }
    }
}
```

```

        this.id = id;
    }

    public ActivityForm(string mail,int day, MainForm mainForm)
    {
        InitializeComponent();
        data = new DataBase();
        plan= data.getPlan(mail)[0];
        planStep(plan.tasklist[0]);
        count = 1;
        this.day = day;
        this.mainForm = mainForm;
    }

    private void button1_Click(object sender, EventArgs e)
    {
        if (count < plan.tasklist.Count)
        {
            planStep(plan.tasklist[count]);
            count++;
        } else
        {
            Haracteristics haracteristics = new Haracteristics();
            haracteristics = data.getHaracteristics(plan.mail);
            plan.status[day] = 1;
            string res= string.Join(", ", plan.status);
            data.UpdateHaracteristic(plan.mail, plan.type);
            data.UpdatePlanStatus(plan.id, res);
            mainForm.refreshPlan(plan.mail);
            mainForm.refreshPanel(plan.mail);
            this.Close();
        }
    }

```

```

    }
}
}
}

```

Б.2 Код файла Auth.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace DiplomProject
{
    class Auth
    {
        DataBase dataBase;

        public Auth()
        {
            dataBase = new DataBase();
        }

        internal DataBase DataBase
        {
            get => default;
            set
            {

```

```

    }
}

public bool authorize(string mail, string password)
{
    User user = new User();
    user = DataBase.FindUser(mail);
    if (user.mail == mail)
    {
        if (user.password == password)
        {
            MainForm MainForm = new MainForm(user.mail);
            MainForm.Show();
            return true;
        }
        else
        {
            MessageBox.Show("Неправильний пароль!!!",
"Помилка!!!"); return false; }
        }
    else {
        MessageBox.Show("Користувач не знайдено!!!",
"Помилка!!!"); return false; }
    }
}
}

```

Б.3 Код файлу Authorization.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;

```

```

using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace DiplomProject
{
    public partial class Authorization : Form
    {
        DataBase dataBase;

        public Authorization()
        {
            InitializeComponent();
            dataBase = new DataBase();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Auth auth = new Auth();
            if( auth.authorize(mail.Text, password.Text))
            {
                this.Hide();
            }
        }
    }
}

```

Б.4 Код файла ConfirmForm.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Xml.Linq;

namespace DiplomProject
{
    public partial class ConfirmForm : Form
    {
        string pin;
        string name,mail,password;
        public ConfirmForm(string pin, string name, string mail, string
password)
        {
            InitializeComponent();
            this.pin = pin;
            this.name = name;
            this.mail = mail;
            this.password = password;
        }

        private void button1_Click(object sender, EventArgs e)
```

```

    {
        DataBase dataBase = new DataBase();
        if (pin == textBox1.Text)
            dataBase.addUser(name, mail, password);
        ManageUser manageUser = new ManageUser();
        manageUser.lastDateofEnter();
        this.Close();
    }
}
}

```

Б.5 Код файла DataBase.cs

```

using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DiplomProject
{
    class DataBase
    {
        string connectionString =
"server=127.0.0.1;user=root;database=trainingprogram;password=impossiblepass
wordforyou;port=3306;";
        MySqlConnection conn;
        public DataBase()

```

```
{
    conn = new MySqlConnection(connectString);
}
```

```
internal Task Task
```

```
{
    get => default;
    set
    {
    }
}
```

```
internal Haracteristics Haracteristics
```

```
{
    get => default;
    set
    {
    }
}
```

```
internal Plan Plan
```

```
{
    get => default;
    set
    {
    }
}
```

```
internal User User
```

```
{
```

```

        get => default;
        set
        {
        }
    }

    public void addUser(string name, string mail, string password)
    {
        conn.Open();
        string sql1 = $"INSERT INTO user (name,mail,password)
values(\"{name}\",\"{mail}\",\"{password}\")";
        MySqlCommand command1 = new MySqlCommand(sql1, conn);
        command1.ExecuteNonQuery();
        conn.Close();
    }

    public void addharacteristicmap(string mail, int strength, int endurance,
int health, int dexterity, int age, int weight, int height)
    {
        conn.Open();
        string sql1 = $"INSERT INTO characteristicsmap
(mail,strength,endurance,health,dexterity,age,weight,height)
values(\"{mail}\",\"{strength}\",\"{endurance}\",\"{health}\",\"{dexterity}\",\"{ag
e}\",\"{weight}\",\"{height}\")";
        MySqlCommand command1 = new MySqlCommand(sql1, conn);
        command1.ExecuteNonQuery();
        conn.Close();
    }

    public void DeleteUserById(int id)
    {
        conn.Open();

```

```

        string sql = $"DELETE FROM user WHERE (((user.id)={id}))";
        MySqlCommand command = new MySqlCommand(sql, conn);
        command.ExecuteScalar();
        conn.Close();
    }

    public User findUser(string mail)
    {
        User rez;
        rez = new User();
        conn.Open();
        string sql = $"select * from user WHERE(user.mail=\"{mail}\")";
        MySqlCommand command = new MySqlCommand(sql, conn);
        MySqlDataReader reader = command.ExecuteReader();
        while (reader.Read())
        {
            rez.Id = Convert.ToInt32(reader[0]);
            rez.Name = reader[1].ToString();
            rez.mail = reader[2].ToString();
            rez.password = reader[3].ToString();
        }
        conn.Close();
        return rez;
    }

    public Haracteristics getHaracteristics(string mail)
    {
        conn.Close();
        Haracteristics rez;
        rez = new Haracteristics();
        conn.Open();
    }

```

```

        string sql = $"select * from characteristicsmap
WHERE(characteristicsmap.mail=\"{mail}\");";
        MySqlCommand command = new MySqlCommand(sql, conn);
        MySqlDataReader reader = command.ExecuteReader();
        while (reader.Read())
        {
            rez.mail = reader[0].ToString();
            rez.strength = Convert.ToInt32(reader[1]);
            rez.endurance = Convert.ToInt32(reader[2]);
            rez.health = Convert.ToInt32(reader[3]);
            rez.dexterity = Convert.ToInt32(reader[4]);
            rez.age = Convert.ToInt32(reader[5]);
            rez.weight = Convert.ToInt32(reader[6]);
            rez.height = Convert.ToInt32(reader[7]);
        }
        conn.Close();
        return rez;
    }

    public void UpdateUserPlan(string mail,int id)
    {
        conn.Open();
        string sql = $"UPDATE plan SET plan.mail = \"{mail}\" WHERE
(((plan.id)=\"{id}\"));";
        MySqlCommand command = new MySqlCommand(sql, conn);
        command.ExecuteScalar();
        conn.Close();
    }

    public void UpdatePlanStatus(int id,string status)
    {
        conn.Open();

```

```

        string sql = $"UPDATE plan SET plan.status = \"{status}\" WHERE
        (((plan.id)=\"{id}\"));";
        MySqlCommand command = new MySqlCommand(sql, conn);
        command.ExecuteScalar();
        conn.Close();
    }
    public int getOneHaracter(string mail, string haracter)
    {
        Haracteristics ha = new Haracteristics();
        ha = getHaracteristics(mail);
        conn.Open();
        Random rnd = new Random();
        rnd.Next(10, 30);
        string sql = $"select characteristicsmap.{haracter} from
        characteristicsmap WHERE (((characteristicsmap.mail)=\"{mail}\"));";
        MySqlCommand command1 = new MySqlCommand(sql, conn);
        MySqlDataReader reader = command1.ExecuteReader();
        int result = 0;
        while (reader.Read())
        {
            result = Convert.ToInt32(reader[0]);
        }
        result += rnd.Next(10, 30);
        conn.Close();
        return result;
    }
    public void UpdateHaracteristic(string mail, string haracter)
    {
        int result=getOneHaracter(mail, haracter);
        conn.Open();

```

```

        string sql = $"UPDATE characteristicsmap SET
characteristicsmap.{haracter} = \"{result}\" WHERE
(((characteristicsmap.mail)= \"{mail}\"));";

        MySqlCommand command = new MySqlCommand(sql, conn);
        command.ExecuteScalar();
        conn.Close();
    }

    public void refreshPlan(string mail)
    {
        string result = "unknown";
        conn.Open();
        string sql = $"UPDATE plan SET plan.mail = \"{result}\" WHERE
(((plan.mail)= \"{mail}\"));";

        MySqlCommand command = new MySqlCommand(sql, conn);
        command.ExecuteScalar();
        conn.Close();
    }

    public void punishment(string mail)
    {
        conn.Open();
        string sql = $"select health from characteristicsmap where
(characteristicsmap.mail= \"{mail}\"));";

        MySqlCommand command = new MySqlCommand(sql, conn);
        MySqlDataReader reader = command.ExecuteReader();
        int result=0;
        while (reader.Read())
        {
            result = Convert.ToInt32(reader[0]);
        }
    }

```

```

        result--;
        conn.Close();
        conn.Open();
        string sql1 = $"UPDATE characteristicsmap SET
characteristicsmap.mail = \"{mail}\" WHERE
(((characteristicsmap.health)=\"{result}\"));";
        MySqlCommand command1 = new MySqlCommand(sql1, conn);
        command1.ExecuteNonQuery();
        conn.Close();
    }
    public List<Plan> getPlan(string mail)
    {
        conn.Open();
        List<Plan> plan = new List<Plan>();
        string sql = $"select * from plan where (plan.mail= \"{mail}\"));";
        MySqlCommand command = new MySqlCommand(sql, conn);
        MySqlDataReader reader = command.ExecuteReader();
        //Console.WriteLine(name);
        while (reader.Read())
        {
            Plan tmp = new Plan();

            tmp.id = Convert.ToInt32(reader[0]);

            tmp.name = reader[1].ToString();

            tmp.mail = reader[2].ToString();
            tmp.tasklist
reader[3].ToString().Split(',').Select(int.Parse).ToList();

```

```

        tmp.status
reader[4].ToString().Split(',').Select(int.Parse).ToList();
        tmp.type = reader[5].ToString();
        plan.Add(tmp);
    }
    conn.Close();
    return plan;
}
public List<Plan> getPlanById(int id)
{
    conn.Open();
    List<Plan> plan = new List<Plan>();
    string sql = $"select * from plan where (plan.id=\"{id}\")";
    MySqlCommand command = new MySqlCommand(sql, conn);
    MySqlDataReader reader = command.ExecuteReader();
    //Console.WriteLine(name);
    while (reader.Read())
    {
        Plan tmp = new Plan();

        tmp.id = Convert.ToInt32(reader[0]);

        tmp.name = reader[1].ToString();

        tmp.mail = reader[2].ToString();
        tmp.tasklist
reader[3].ToString().Split(',').Select(int.Parse).ToList();
        tmp.status
reader[4].ToString().Split(',').Select(int.Parse).ToList();
        plan.Add(tmp);

```

```

    }
    conn.Close();
    return plan;
}

public Task getTaskById(int id)
{
    conn.Open();
    List<Task> task = new List<Task>();
    string sql = $"select * from task where (task.id='{id}')";
    MySqlCommand command = new MySqlCommand(sql, conn);
    MySqlDataReader reader = command.ExecuteReader();
    //Console.WriteLine(name);
    while (reader.Read())
    {
        Task tmp = new Task();
        tmp.id = Convert.ToInt32(reader[0]);
        tmp.type = reader[1].ToString();
        tmp.name = reader[2].ToString();
        tmp.description = reader[3].ToString();
        task.Add(tmp);
    }
    conn.Close();
    return task[0];
}

public List<Plan> getPlans()
{
    conn.Open();
    List<Plan> plan = new List<Plan>();
    string sql = "select * from plan;";
    MySqlCommand command = new MySqlCommand(sql, conn);

```

```

        MySqlDataReader reader = command.ExecuteReader();
        //Console.WriteLine(name);
        while (reader.Read())
        {
            Plan tmp = new Plan();

            tmp.id = Convert.ToInt32(reader[0]);

            tmp.name = reader[1].ToString();

            tmp.mail = reader[2].ToString();
            tmp.tasklist =
reader[3].ToString().Split(',').Select(int.Parse).ToList();
            tmp.status =
reader[4].ToString().Split(',').Select(int.Parse).ToList();
            plan.Add(tmp);
        }
        conn.Close();
        return plan;
    }
}

```

Б.6 Код файла Form1.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;

```

```

using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using                                     static
System.Windows.Forms.VisualStyleElement.Window;

namespace DiplomProject
{
    public partial class Form1 : Form
    {
        DataBase dataBase;
        public Form1()
        {
            InitializeComponent();
            dataBase = new DataBase();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            registration reg = new registration();
            reg.register(name.Text, mail.Text, password.Text, password1.Text);
        }

        private void button2_Click(object sender, EventArgs e)
        {
            Authorization authorization = new Authorization();
            authorization.Show();
            this.Hide();
        }
    }
}

```

```

    }
}
}

```

Б.7 Код файла haracterForm.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace DiplomProject
{
    public partial class haracterForm : Form
    {
        int weight, height, age, strength, endurance, dexterity, health;
        int count;
        string mail;
        MainForm mainForm;

        public haracterForm(string mail, MainForm mainForm)
        {
            InitializeComponent();
            this.mail = mail;
            count = 0;

```

```
button1.Text = "Ввести вік";  
this.mainForm = mainForm;  
  
}  
private void button1_Click(object sender, EventArgs e)  
{  
    switch (count)  
    {  
        case 0:  
        {  
            age = Convert.ToInt32(textBox1.Text);  
            count++;  
            button1.Text = "Ввести вагу";  
            break;  
        }  
        case 1:  
        {  
            weight = Convert.ToInt32(textBox1.Text);  
            strength = weight * 10;  
            count++;  
            button1.Text = "Ввести зріст";  
            break;  
        }  
        case 2:  
        {  
            height = Convert.ToInt32(textBox1.Text);  
            count++;  
            health = 4;  
            float tmp = ((float)weight / (height * height)) * 100000;  
            dexterity = Convert.ToInt32(tmp);  
        }  
    }  
}
```

```

        dexterity *= 3;
        endurance = (dexterity + strength) / 2;
        DataBase data = new DataBase();
        data.addharacteristicmap(mail, strength, endurance, health,
dexterity, age, weight, height);
        mainForm.refreshPanel(mail);
        this.Close();
        break;
    }
}
}
}
}

```

Б.8 Код файла Haracteristics.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DiplomProject
{
    class Haracteristics
    {
        public string mail;
        public int strength;
        public int endurance;
        public int health;
    }
}

```

```

    public int dexterity;
    public int age;
    public int weight;
    public int height;
    public Haracteristics()
    {

    }
}
}

```

Б.9 Код файлу MainForm.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Net.NetworkInformation;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace DiplomProject
{
    public partial class MainForm : Form
    {
        string mail;
    }
}

```

```

DataBase dataBase;

int day;

public MainForm(string mail)
{
    InitializeComponent();
    this.mail = mail;
    ManageUser manageUser = new ManageUser();
    dataBase=new DataBase();
    refreshPanel(mail);
    refreshPlan(mail);
    manageUser.lastDateofEnter();
}

public int getStatus(string mail)
{
    Plan plan=dataBase.getPlan(mail)[0];
    for (int i = 0; i < plan.status.Count; i++)
        if (plan.status[i] == 0)
            return i;
    return 100;
}

public void refreshPlan(string mail)
{
    loop:
    flowLayoutPanel1.Controls.Clear();
    List<Plan> plan = dataBase.getPlan(mail);
    if (plan.Count == 0)
    {
        List<Plan> plans = dataBase.getPlans();
        for (int i = 0; i < plans.Count; i++)

```

```

    {
    if (plans[i].mail == "unknown")
    {
        Button button = new Button();
        button.Text = plans[i].id + " " + plans[i].name;
        button.Width = flowLayoutPanel1.Width - 2;
        button.Click += button2_Click;
        flowLayoutPanel1.Controls.Add(button);
    }
    }
}
else
{
    int res = getStatus(mail);
    if (res == 100)
    {
        Plan pl = dataBase.getPlan(mail)[0];
        for(int i = 0; i < pl.status.Count; i++)
            pl.status[i] = 0;
        string res1 = string.Join(",", pl.status);
        dataBase.UpdatePlanStatus(pl.id, res1);
        dataBase.refreshPlan(mail);
        goto loop;
    }
    Button button = new Button();
    day = getStatus(mail);
    button.Text = "Почти день " + (res+1);
    button.Width = flowLayoutPanel1.Width - 2;
    button.Height = flowLayoutPanel1.Height/2;
    button.Click += button3_Click;

```

```

        flowLayoutPanel1.Controls.Add(button);
    }
}

public void refreshPanel(string mail)
{
    ManageUser manageUser = new ManageUser();
    if(manageUser.checkDate());
    dataBase.punishment(mail);
    flowLayoutPanel2.Controls.Clear();
    flowLayoutPanel2.BringToFront();
    Haracteristics haracteristics = new Haracteristics();
    haracteristics=dataBase.getHaracteristics(mail);

    List<string> list = new List<string>();
    list.Add(haracteristics.strength.ToString());
    list.Add(haracteristics.endurance.ToString());
    list.Add(haracteristics.health.ToString());
    list.Add(haracteristics.dexterity.ToString());
    list.Add("Сила:");
    list.Add("Витривалість:");
    list.Add("Здоров'я:");
    list.Add("Спритність:");
    for (int i = 0; i<4; i++)
    {
        Button button = new Button();
        button.Text =list[i+4] + " " + list[i];
        button.Width = flowLayoutPanel2.Width - 2;
        flowLayoutPanel2.Controls.Add(button);
    }
    if (Convert.ToInt32(list[0])==0)

```

```

    {
        Button button = new Button();
        button.Text = "Отримати характеристики";
        button.Width = flowLayoutPanel2.Width - 2;
        button.Click += button1_Click;
        flowLayoutPanel2.Controls.Add(button);
    }
}

private void button1_Click(object sender, EventArgs e)
{
    haracterForm haracterForm = new haracterForm(mail,this);
    haracterForm.ShowDialog();
}

private void button2_Click(object sender, EventArgs e)
{
    Button btn = (Button)sender;
    PlanForm planForm = new PlanForm(btn.Text, mail);
    planForm.ShowDialog();
}

private void button3_Click(object sender, EventArgs e)
{
    ActivityForm activityForm = new ActivityForm(mail,day,this);
    activityForm.ShowDialog();
}
}
}

```

Б.10 Код файла ManageUser.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DiplomProject
{
    class ManageUser
    {
        internal DataBase DataBase
        {
            get => default;
            set
            {
            }
        }

        public void lastDateofEnter()
        {
            using (StreamWriter file = new StreamWriter("dates.txt"))
                file.WriteLine(DateTime.Now.ToString("yyyy-MM-dd"));
        }

        public bool checkDate()
        {

```

```

List<string> dates = new List<string>();
using (StreamReader sr = new StreamReader("dates.txt"))
{
    string line;
    while ((line = sr.ReadLine()) != null)
        dates.Add(line);
}
DateTime result = Convert.ToDateTime(dates[0]).AddDays(1);
lastDateofEnter();
if (Convert.ToDateTime(DateTime.Now.ToString("yyyy-MM-
dd")) == result || Convert.ToDateTime(dates[0]) ==
Convert.ToDateTime(DateTime.Now.ToString("yyyy-MM-dd")))
    return true;
else
    return false;
}
}
}

```

Б.11 Код файла Plan.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DiplomProject
{
    class Plan

```

```

{
    public int id;
    public string name;
    public string mail;
    public List<int> tasklist;
    public List<int> status;
    public string type;
}
}

```

Б.12 Код файла PlanForm.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace DiplomProject
{
    public partial class PlanForm : Form
    {
        Plan plan;
        DataBase db;
        string mail;
    }
}

```

```

public void fullPanel()
{
    for (int i = 0; i < plan.tasklist.Count; i++)
    {

        Button button = new Button();
        button.Text = db.getTaskById(plan.tasklist[i]).name;
        button.Width = flowLayoutPanel1.Width - 2;
        flowLayoutPanel1.Controls.Add(button);
    }
}

public PlanForm(string text, string mail)
{
    InitializeComponent();
    db = new DataBase();
    Match match = Regex.Match(text, @"^\d+");
    plan = db.getPlanById(Convert.ToInt32(match.Value))[0];
    label1.Text = plan.name;
    this.mail = mail;
    fullPanel();

}

private void button1_Click(object sender, EventArgs e)
{
    db.UpdateUserPlan(mail, plan.id);
}
}
}

```

Б.13 Код файла registration.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace DiplomProject
{
    class registration
    {
        DataBase dataBase;

        public registration()
        {
            this.dataBase = new DataBase();
        }

        internal DataBase DataBase
        {
            get => default;
            set
            {
            }
        }

        public void register(string name, string mail, string password, string
password1)
```

```

{
    if (mail.Contains("@"))
    {
        User user = dataBase.findUser(mail);
        if (user.mail == mail)
            MessageBox.Show("Такий користувач вже зареєстрований!!!", "Помилка!!!");
        else
        {
            if (password == password1)
            {
                SendMail sendMail = new SendMail(mail);
                bool res = new bool();
                ConfirmForm confirmForm = new ConfirmForm(sendMail.sendletter(), name, mail, password);
                confirmForm.ShowDialog();
            }
            else
                MessageBox.Show("Не збігаються паролі!!!", "Помилка!!!");
        }
    }
    else
        MessageBox.Show("Некоректно введена пошта!!!", "Помилка!!!");
}
}
}

```

Б.14 Код файлу SendMail.cs

```
using System;
```

```

using System.Collections.Generic;
using System.Linq;
using System.Net.Mail;
using System.Net;
using System.Text;
using System.Threading.Tasks;

namespace DiplomProject
{
    class SendMail
    {
        string mail;
        MailAddress from;
        MailAddress to;
        MailMessage m;
        SmtpClient smtp;
        DataBase db;
        public SendMail(string userlogin)
        {
            db = new DataBase();
            mail = userlogin;
            from = new MailAddress("adidanka994@gmail.com", "Програма
підбору тренувань");
            to = new MailAddress(mail);
            m = new MailMessage(from, to);
            smtp = new SmtpClient("smtp.gmail.com", 587);
            smtp.Credentials = new
NetworkCredential("adidanka994@gmail.com", "rcslavykkqlapobh");
            smtp.EnableSsl = true;
        }
    }

```

```

internal DataBase DataBase
{
    get => default;
    set
    {
    }
}

public string sendletter()
{
    List<Task> tasks = new List<Task>();
    List<string> ints = new List<string>();
    Random random = new Random();
    for (int i = 0; i < 6; i++) ints.Add(random.Next(1,10).ToString());
    m.Subject = "Сповідання системи безпеки!!!";
    m.Body = string.Concat(ints);
    m.IsBodyHtml = true;
    smtp.Send(m);
    return string.Concat(ints);
}
}
}

```

Б.15 Код файлу Task.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

```

```

using System.Threading.Tasks;

namespace DiplomProject
{
    class Task
    {
        public Task() { }
        public int id { get; set; }
        public string name { get; set; }
        public string description { get; set; }
        public string type { get; set; }

    }
}

```

Б.16 Код файла User.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DiplomProject
{
    class User
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string mail { get; set; }
    }
}

```

```
        public string password { get; set; }  
    }  
}
```

ДОДАТОК В
Слайди презентації

Національний університет «Запорізька політехніка»

1

Кафедра програмних засобів

Дипломна кваліфікаційна робота

Реалізація програмного забезпечення для підбору індивідуального плану тренувань

ВИКОНАВ

СТ. ГР. КНТ-120

КЕРІВНИК

ДАНИЛО ПОТАПОВ

ОЛЕНА ШИТКОВА

Рисунок В.1 – Слайд 1

Постановка завдання

2

- ▶ Об'єкт дослідження – система тренувань для покращення фізичної форми людини.
- ▶ Предмет дослідження – програмне забезпечення з підбору тренувань для покращення фізичної форми.
- ▶ Мета розробки – програмне забезпечення підбору тренувань для різних категорій осіб, які прагнуть покращити своє фізичне здоров'я, з метою оптимізації тренувального процесу і досягнення кращих результатів.
- ▶ Завданням є аналіз науково-методичних джерел та рекомендацій з тренувального процесу, розробка структурованої системи тренувань для різних рівнів підготовки, проскривання та розробка програмного забезпечення з підбору тренувань, тестування ефективності отриманого продукту на практиці та корекція за потреби.

Рисунок В.2 – Слайд 2

Функціональні вимоги

3

- Реєстрація – програма має клієнт-серверну базу даних і для того щоб ідентифікувати користувача в програмі він повинен зареєструватися та підтвердити свій обліковий запис шляхом вводу коду відправленого на введenu пошту.
 - Авторизація – після реєстрації користувач повинен мати змогу увійти у свій обліковий запис шляхом введення логіну та паролю.
 - Введення та обробка даних – користувач має змогу вводити свої дані такі як: зріст, вага, вік. На основі введених даних програма розраховує характеристики користувача.
 - Статистика тренувань – користувач має змогу відстежувати прогрес виконання плану шляхом відстеження характеристик.
 - Надання інформації – застосунок повинен давати всю необхідну інформацію, для безпечного та ефективного тренування, включно із технікою тренування.
- Простота та легкість у використанні – інтерфейс повинен мати низький поріг входу, так у користувача не відпаде бажання почати тренування.
- Графічний інтерфейс повинен мати легку навігаційну структуру, щоб користувачу було просто знайти потрібну форму, у розроблювальному застосунку використовується послідовний інтерфейс.

Рисунок В.3 – Слайд 3

Вибір мови програмування

4

Характеристика	C++	C#	Python
Тип системи	Статична	Статична	Динамічна
Синтаксис	Складний	Середній	Легкий
Парадигми	Процедурне, об'єктно-орієнтоване, загальне	Об'єктно-орієнтоване, компонентно-орієнтоване	Об'єктно-орієнтоване, компонентно-орієнтоване
Застосування	Системне програмування, ігри, ресурсозатратні додатки	Веброзробка, десктопні додатки, ігри	Веброзробка, наукові обчислення, автоматизація, штучний інтелект
Кроссплатформеність	Ні	Так	Так
Продуктивність	Висока	Висока	Середня
Рівень абстракції	Низький	Високий	Високий
Обробка помилок	Ручна	Винятки	Винятки, твердження

Рисунок В.4 – Слайд 4

Вибір системи керування базами даних

5

Характеристика	MS Access	MySQL Workbench
Тип	База даних на основі файлів	Клієнт-серверна база даних
Ліцензія	Пропріетарна	Відкрита
Операційна система	Windows	Windows, Linux, MacOS
Мова запитів	SQL, Query Design	SQL
Продуктивність	Середня	Висока
Безпека	Низька	Висока
Резервне копіювання	Основне	Розширене

Рисунок В.5 – Слайд 5

Вибір інтегрованого середовища розробки

6

Характеристика	Qt Creator	Visual Studio	VS Code
Тип	IDE	IDE	Редактор коду
Підтримувані мови програмування	C++, QML, Python, JavaScript	C++, C#, JavaScript, Python	Багато мов програмування, зокрема C++, C#, JavaScript, Python
Підтримувані операційні системи	Windows, Linux, macOS	Windows	Windows, Linux, macOS
Інструментарій розробки графічного інтерфейсу	Qt Designer	Visual Designer	Розширені можливості для роботи з UI, але не такі розгалужені як у Qt або VS
Інтеграція з системами керування проєктами	Системи керування версіями Git, Subversion	Системи керування версіями Git, Team Foundation Server	Системи керування версіями Git
Відеадка	Вбудована в IDE	Вбудована в IDE	Встановлюється окремим модулем
Ліцензія	Безкоштовна	Платна	Безкоштовна

Рисунок В.6 – Слайд 6

Схема навігації по програмі

7

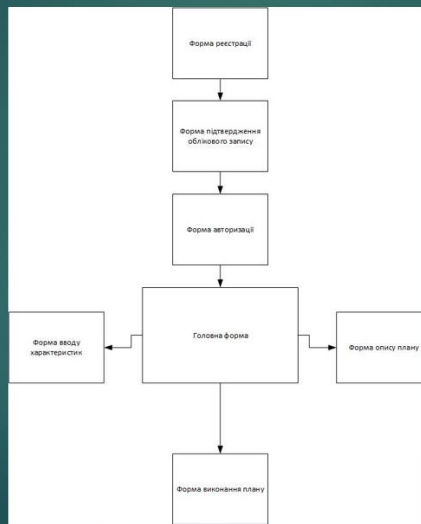


Рисунок В.7 – Слайд 7

ER модель бази даних

8

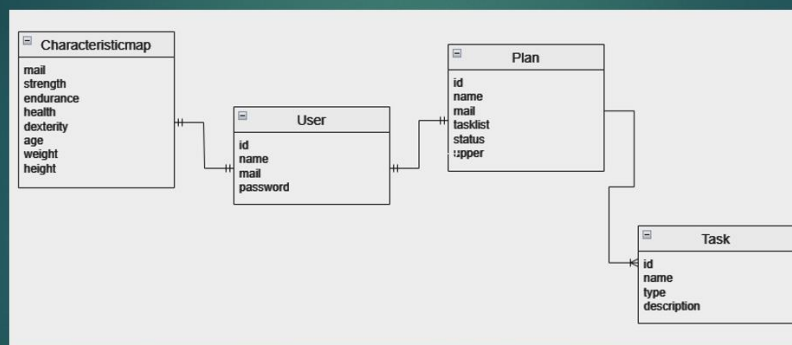


Рисунок В.8 – Слайд 8

Діаграма класів

9

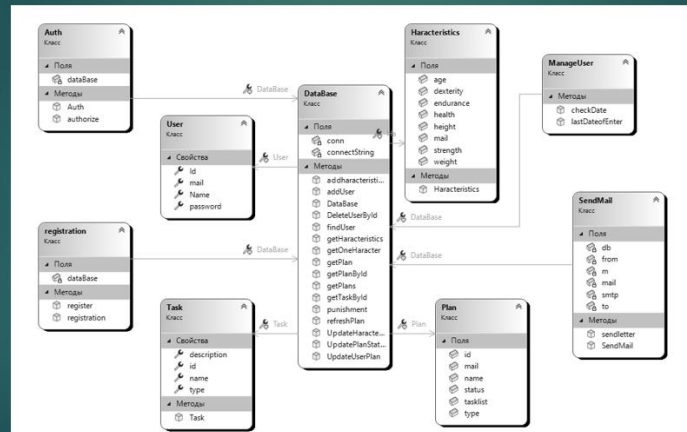


Рисунок В.9 – Слайд 9

Діаграма розгортання

10

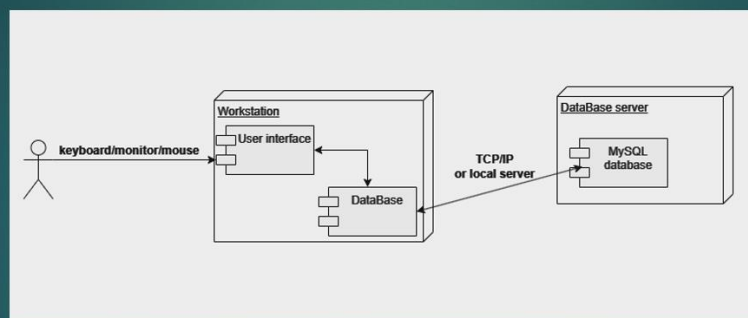


Рисунок В.10 – Слайд 10

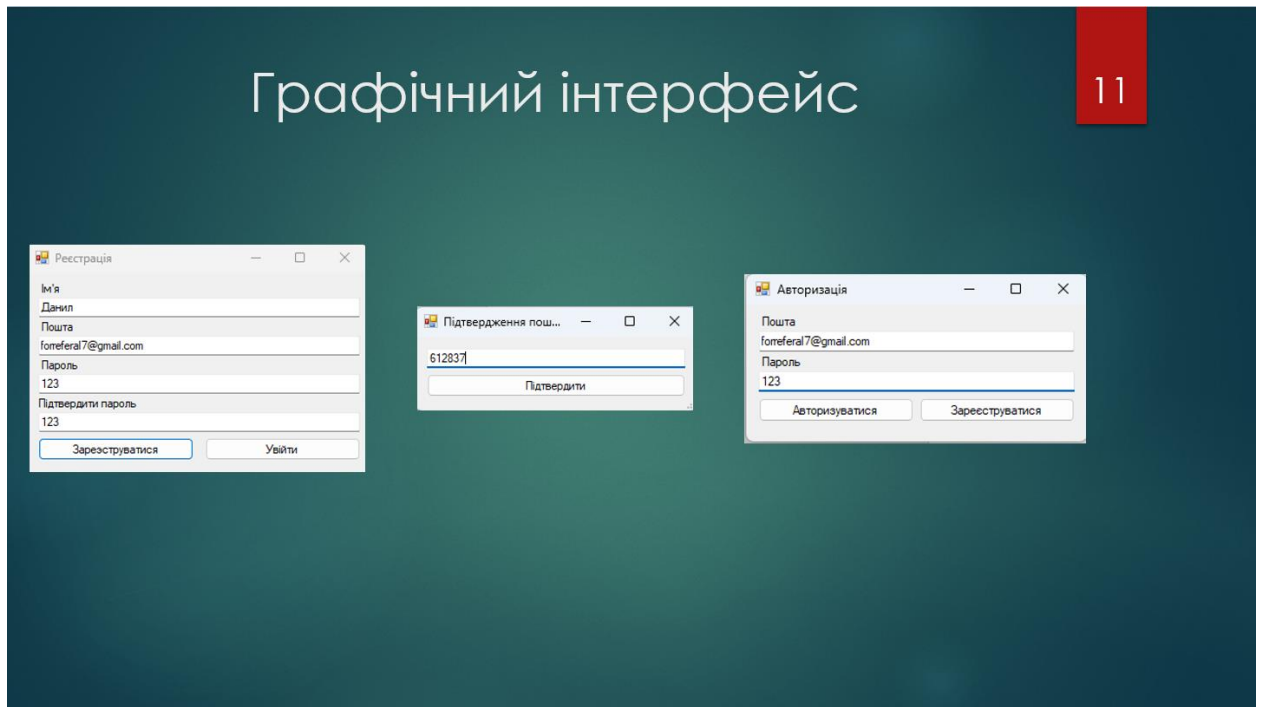


Рисунок В.11 – Слайд 11

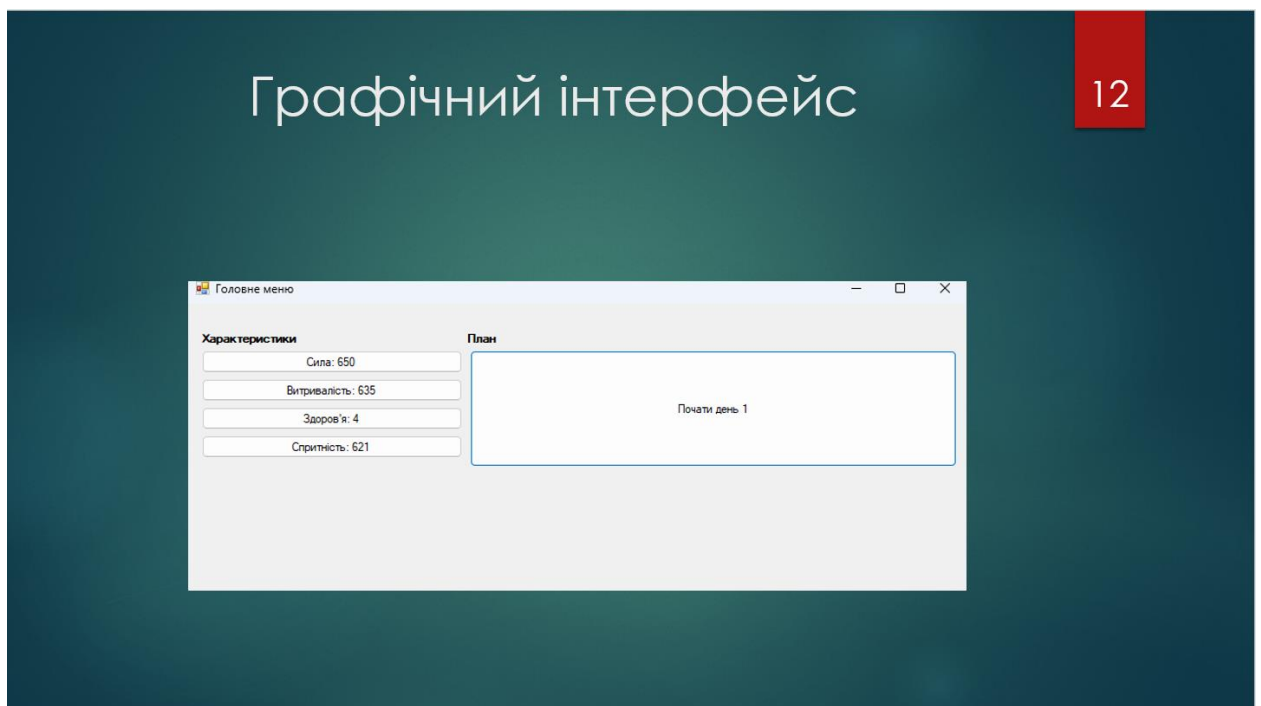


Рисунок В.12 – Слайд 12

Графічний інтерфейс

13

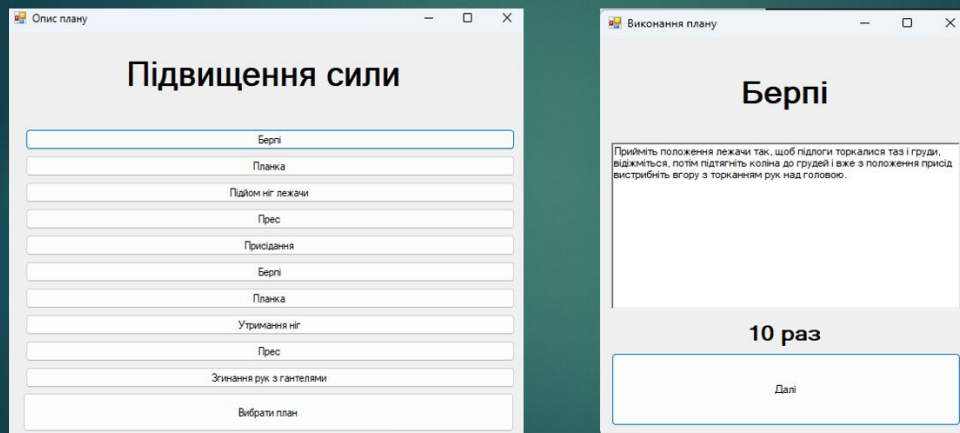


Рисунок В.13 – Слайд 13

Висновки

14

- Для максимально ефективного підбору плану тренування у домашніх умовах необхідні деякі знання, тому актуальна розробка програмного застосунку з підбору плану тренувань.
- Було проаналізовано існуючі застосунки для тренувань, мови програмування, інтегровані середовища розробки, системи керування базами даних. Обрано мову програмування – C#, IDE – Visual Studio, СКБД – MySQL Workbench.
- Аналіз існуючих застосунків дозволив визначити функціональні вимоги та вимоги до графічного інтерфейсу програми, вимоги до складу параметрів технічного засобу, де було обрано платформу ОС Windows. Застосунок відповідає меті та темі розробки та усім перерахованим вимогам, які були визначені.
- Результатом цієї роботи стало розроблений та протестований програмний застосунок для підбору тренувань у домашніх умовах.

Рисунок В.14 – Слайд 14