НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2023 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення інформаційних систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Мобільний застосунок для боротьби з нікотиновою залежністю

Виконав студент IV курсу, групи ІТ-92
(шифр групи)
Василенко Олександр Ігорович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник ас., Храмченко Микола Сергійович _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з графічної документації ст. викл., Вітковська І.І _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент доц., ктн., доц., Сперкач М.О _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ –2023

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення інформаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2023 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Василенку Олександру Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема проєкту Мобільний застосунок для боротьби з нікотиновою залежністю

керівник проєкту Храмченко Микола Сергійович, ас.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «31»травня 2023 р. № 2101-с

2. Термін подання студентом проєкту « 17 » червня 2023 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки
1) Загальні положення: основні визначення та терміни, опис предметної області, огляд ринку програмних продуктів, постановка задачі.
2) Моделювання та конструювання програмного забезпечення: моделювання програмного забезпечення, засоби розробки, опис структури бази даних, опис архітектурного шаблону.
3) Аналіз якості та тестування програмного забезпечення: аналіз якості розробленого програмного забезпечення, опис процесів тестування, опис контрольного прикладу.
4) Розгортання програмного забезпечення: Опис процесів розгортання розробленого програмного забезпечення, опис процесів підтримки розробленого програмного забезпечення.

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань

2) Схема бази даних

3) Креслення вигляду екранних форм

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «10» березня 2023 року

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	10.03.2023	
2	Аналіз існуючих методів розв’язання задачі	17.03.2023	
3	Постановка та формалізація задачі	23.03.2023	
4	Розробка інформаційного забезпечення	30.03.2023	
5	Алгоритмізація задачі	06.04.2023	
6	Обґрунтування вибору використаних технічних засобів	12.04.2023	
7	Розробка програмного забезпечення	20.04.2023	
8	Налагодження програми	30.04.2023	
9	Виконання графічних документів	10.05.2023	
10	Оформлення пояснювальної записки	20.05.2023	
11	Подання ДП на попередній захист	06.06.2023	
12	Подання ДП рецензенту	12.06.2023	
13	Подання ДП на основний захист	19.06.2023	

Студент _____ Олександр ВАСИЛЕНКО
(підпис) (ініціали, прізвище)

Керівник _____ Микола ХРАМЧЕНКО
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з чотирьох розділів, містить 28 таблиць, 22 рисунки та 7 джерел – загалом 62 сторінки.

Дипломний проєкт присвячений розробці мобільного застосунку для боротьби з нікотиновою залежністю.

Метою даного дипломного проєкту є покращення взаємодії користувача з health-care застосунками за допомогою ШІ.

Об'єкт дослідження: процес розробки мобільного застосунку який допоможе людям позбутись нікотинової залежності.

Предмет дослідження: мобільного застосунку який допоможе людям позбутись нікотинової залежності.

У першому розділі було проведено аналіз предметної області, аналіз відомих аналогів та аналіз їх переваг та недоліків. Також було розроблено функціональні та нефункціональні вимоги. Також було надано можливі варіанти використання застосунку та сформульовано задачі.

У другому розділі було змодельовано та детально описано бізнес процеси. Також було розглянуто особливості архітектури MVVM. Окрім цього було проаналізовано безпеку даних користувача, детально описано інструменти які за допомогою яких буде проведено практичну реалізацію застосунку та описано модулі застосунку та їх методи.

У третьому розділі було проведено тестування розробленого застосунку.

У четвертому розділі було описано інструменти для розгортання застосунку, а також сам процес розгортання застосунків для пристроїв, що працюють на операційній системі iOS.

Програмне забезпечення впроваджено на операційній системі iOS

КЛЮЧОВІ СЛОВА: МОБІЛЬНИЙ ЗАСТОСУНОК, IOS, XCODE, GOOGLE FIREBASE, SWIFT, COMBINE, SWIFTUI, UIKIT, COREML.

ABSTRACT

The explanatory note of the diploma project consists of four sections, contains 28 tables, 22 figures and 7 sources - a total of 62 pages.

The diploma project is dedicated to the development of a mobile application to combat nicotine addiction.

The goal of this diploma project is to improve user interaction with health-care applications using AI.

Research object: the process of developing a mobile application that will help people get rid of nicotine addiction.

The subject of the study: a mobile application that will help people get rid of nicotine addiction.

In the first section, an analysis of the subject area, an analysis of known analogues, and an analysis of their advantages and disadvantages was carried out. Functional and non-functional requirements were also developed. Possible options for using the application were also provided and tasks were formulated.

In the second chapter, business processes were modeled and described in detail. The features of the MVVM architecture were also considered. In addition, the security of user data was analyzed, the tools used for the practical implementation of the application were described in detail, and the application modules and their methods were described.

In the third section, the developed application was tested.

In the fourth chapter, the tools for deploying the application were described, as well as the process of deploying applications for devices running on the iOS operating system.

The software is implemented on the iOS operating system

KEYWORDS: MOBILE APP, IOS, XCODE, GOOGLE FIREBASE, SWIFT, COMBINE, SWIFTUI, UIKIT, COREML.

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІТ-9205.045490.01.91	Технічне завдання	11	
3	А4	КПІ.ІТ-9205.045490.02.81	Пояснювальна записка	62	
4	А4	КПІ.ІТ-9205.045490.03.13	Текст програми	22	
5	А4	КПІ.ІТ-9205.045490.04.51	Програма та методика тестування	6	
6	А4	КПІ.ІТ-9205.045490.05.34	Керівництво користувача	12	
7	А3	КПІ.ІТ-9205.045490.06.99	Схема структурна варіантів	1	
			використань		
8	А3	КПІ.ІТ-9205.045490.07.99	Схема бази даних	1	
9	А3	КПІ.ІТ-9205.045490.08.99	Креслення вигляду екранних форм	1	

Змін.	Арк.	№ докум.	Підп.	Дата	КПІ.ІТ-9205.045440.00.90		
Розроб.		Василенко О.І			Відомість дипломного проєкту	Літ.	Аркуш 1 / Аркушів
Перевір.		Храмченко М.С					
Н.контр.		Ліщук К.І.				КПІ ім. Ігоря Сікорського ФІОТ каф. ІПІ гр. ІТ-92	
Затв.		Жаріков Е.В.					

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

Мобільний застосунок для боротьби з нікотиновою залежністю

Технічне завдання

КПІ.ІТ-9205.045490.01.91

“ПОГОДЖЕНО”
Керівник проєкту:
Микола ХРАМЧЕНКО

Нормоконтроль:
Ірина ВІТКОВСЬКА

Виконавець:
Олександр ВАСИЛЕНКО

Київ – 2023

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Мобільний застосунок для боротьби з нікотиновою залежністю.

Галузь застосування:

Наведене технічне завдання поширюється на розробку програмного забезпечення [045490], котра використовується для допомоги користувачам в боротьбі з нікотиновою залежністю та призначена для людей які намагаються позбутись нікотинової залежності.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки Мобільного застосунку для боротьби з нікотиновою залежністю є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для покращення взаємодії користувача з health-care застосунками.

Метою даного дипломного проєкту є покращення взаємодії користувача з health-care застосунками за допомогою ШІ.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1 Для користувача:

– Автентифікація до системи
– Введення персональних даних користувача
– Зміна персональних даних
– Зміна статистичних даних
– Визначення самопочуття
– Перегляд статистичних даних
– Перегляд рекомендацій даних

4.1.2 Додаткові вимоги:

– Підтримка push notifications
– Збереження нових користувачів у базу даних
– Обробка статистичних даних користувача
– Збереження статистичних даних користувачів у базу даних
– Визначення самопочуття користувача

4.2 Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача. Забезпечити цілісність інформації в базі даних. Забезпечити валідацію даних.

4.3 Умови експлуатації

Умови експлуатації згідно ДСанПІН 3.3.2.007-98.

4.3.1 Вид обслуговування

Вимоги до виду обслуговування не висуваються

4.3.2 Обслуговуючий персонал

Вимоги до обслуговуючого персоналу не висуваються

4.4 Вимоги до складу і параметрів технічних засобів

Програмне забезпечення повинно функціонувати на мобільних пристроях з операційною системою iOS.

Мінімальна конфігурація технічних засобів:

– Apple A11 Bionic;
– об‘єм ОЗП: 2 Гб;
– підключення до мережі Інтернет зі швидкістю від 20 мегабіт;
– операційна система iOS версії 16 та вище;

Рекомендована конфігурація технічних засобів :

– Apple A13 Bionic;
– об‘єм ОЗП: 4 Гб;
– підключення до мережі Інтернет зі швидкістю від 100 мегабіт;

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційних систем iOS версії 16.0 та більше

4.5.1 Вимоги до вхідних даних

Вхідні дані повинні бути представлені в наступному форматі: відсутні

4.5.2 Вимоги до вихідних даних

Результати повинні бути представлені в наступному форматі: відсутні.

4.5.3 Вимоги до мови розробки

Розробку виконати на мові програмування Swift

4.5.4 Вимоги до середовища розробки

Розробку виконати за допомогою середовища Xcode

4.5.5 Вимоги до представленню вихідних кодів

Вихідний код програми має бути представлений у вигляді набору файлів з розширенням .swift

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Спеціальні вимоги не висуваються.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

– пояснювальна записка;

– технічне завдання;

– текст програми;

– програма та методика тестування;

– керівництво користувача;

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

– схема структурна варіантів використання;

– схема бази даних;

– креслення вигляду екранних форм;

5.2 Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проекту	21.02	
2.	Розробка технічного завдання	03.03	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	19.04	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	30.04	Схема структурна програмного забезпечення та специфікація компонентів
5.	Програмна реалізація програмного забезпечення	05.05	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	10.05	Тести, результати тестування
7.	Розробка матеріалів текстової частини проекту	14.05	Пояснювальна записка
8.	Розробка матеріалів графічної частини проекту	20.05	Графічний матеріал проекту
9.	Оформлення технічної документації проекту	29.05	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Пояснювальна записка
до дипломного проєкту

на тему: Мобільний застосунок для боротьби з нікотиновою залежністю

КПІ.ІТ-9205.045490.02.81

Київ – 2023

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE	–	Integrated Development Environment – інтегроване середовище розробки.
SDK	–	Software development kit
ER	–	Entity-Relation diagram
ОС	–	Операційна система.
ШІ		Штучний інтелект
БД	–	База даних.

ВСТУП

У сучасному суспільстві куріння залишається поширеною проблемою, яка негативно впливає на здоров'я та благополуччя людини. Відмова від куріння є складним процесом, який часто потребує значної підтримки та керівництва. У цифрову еру технології пропонують багатообіцяючі рішення, які допоможуть людям кинути палити та покращити загальний стан здоров'я. Одним із ефективних методів є розробка додатків для смартфонів, розроблених спеціально для відмови від куріння.

Програми для припинення куріння відіграють вирішальну роль у забезпеченні людей необхідними інструментами, ресурсами та підтримкою для подолання своєї залежності. Ці програми спрямовані на популяризацію здорового способу життя, допомагаючи користувачам на шляху до відмови від куріння. Основною метою даного дипломного проєкту є покращення взаємодії користувача з health-care застосунками за допомогою ШІ.

Однією з основних проблем, з якою стикаються люди, які намагаються кинути палити, є відсутність структурованої системи підтримки. Традиційні методи, такі як нікотинзамісна терапія або консультування, не завжди можуть бути доступними або зручними. Однак мобільний додаток може подолати цю прогалину, пропонуючи персоналізовану та легкодоступну систему підтримки, яку користувачі можуть завжди носити в кишені.

Підсумовуючи, розробка мобільного додатку, який допоможе людям кинути палити, є життєво важливим кроком до вирішення поширеної проблеми залежності від куріння. Пропонуючи персоналізовані плани, освітні ресурси, стратегії подолання та підтримку спільноти, програма має на меті розширити можливості користувачів і збільшити їхні шанси успішно кинути палити. Завдяки використанню технологій цей дипломний проект намагається зробити внесок у здоровіше суспільство, сприяючи відмові від куріння та покращуючи загальний добробут людей.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Куріння є поширеною звичкою серед людей у всьому світі, і її пов’язують із численними ризиками для здоров’я, зокрема раком, хворобами серця та респіраторними захворюваннями. Незважаючи на численні попередження щодо здоров’я та кампанії проти куріння, багато людей продовжують палити, і це стало проблемою для громадського здоров’я. Небезпека куріння добре задокументована, і шкідливі наслідки куріння впливають не лише на людину, але й на оточуючих.

Куріння в громадських місцях є спірним питанням протягом багатьох років. Некурці часто піддаються впливу вторинного куріння, яке так само небезпечно, як і саме куріння, і навіть під час невеликого за тривалістю впливу організм людини може зазнати шкоди[1]. Дим, що виділяється курцями, містить понад 7000 хімічних речовин, багато з яких є токсичними та канцерогенними. Коли некурці вдихають цей дим, вони також вдихають ці шкідливі хімічні речовини, що піддає їх ризику розвитку захворювань, пов’язаних з курінням.

Куріння в громадських місцях, включаючи ресторани та кафе, заборонено в багатьох країнах світу[2]. Проте все ще є місця, де палити дозволено, і це становить загрозу для здоров’я населення. Негативні наслідки куріння в громадських місцях не обмежуються проблемами зі здоров’ям; це також впливає на атмосферу місця. Запах сигаретного диму може відштовхнути некурців і зіпсувати їм обід.

В останні роки в багатьох ресторанах і кафе почали повністю забороняти куріння. Це вітали некурці, які тепер можуть насолоджуватися їжею, не піддаючись пасивному курінню. Проте все ще є заклади, де можна палити у спеціально відведених місцях, і це викликає занепокоєння. Навіть у спеціально відведених місцях для куріння некурці все ще можуть піддаватися впливу диму, що може негативно вплинути на їх здоров’я.

Багато людей, які палять, знають про шкоду куріння, і вони докладають зусиль, щоб кинути палити. Однак це легше сказати, ніж зробити, і кинути палити може бути важким і складним процесом. Багато курців продовжують палити, незважаючи на ризик для свого здоров'я, і це викликає занепокоєння. Залежність від нікотину сильна, і кинути палити вимагає великої сили волі та підтримки.

1.2 Змістовний опис і аналіз предметної області

Важливим аспектом розробки застосунку, який допоможе людям побороти нікотинову залежність, є вибір підходу. Тому для створення застосунку було визначено наступні критерії:

– Підтримка: додаток має створювати сприятливе середовище, яке заохочує та мотивує користувачів кинути палити. Він має пропонувати інструменти та ресурси, які полегшать припинення та зменшать страх.

– Доступність: додаток має бути доступним і простим у використанні для всіх користувачів, щоб забезпечити швидкий і ефективний процес відмови від куріння.

– Персоналізованість: додаток має надавати персоналізовану підтримку користувачам, враховуючи їхні індивідуальні потреби, цілі та вподобання. Він має пропонувати спеціальні ресурси та стратегії, які відповідають особистості та стилю життя користувача. Також програма має використовувати аналітику даних, щоб відстежувати прогрес користувачів

Враховуючи вищесказане, було розроблено додаток, який допоможе людям побороти нікотинову залежність.

1.3 Аналіз існуючих технологій та успішних IT-проєктів

Проаналізуємо відомі на сьогодні технічні рішення, що допоможуть у реалізації мобільного застосунку для боротьби з нікотиновою залежністю Далі будуть розглянуті готові програмні рішення.

1.3.1 Аналіз відомих програмних продуктів

Після пошуку застосунків які допомагають людям кинути палити було обрано 2 застосунки:

– Quit smoking get healthy

– Easy quit

Easy quit – цей застосунок, збирає лише певні дані користувача, та показує обмежену не персоналізовану статистику, а також дає користувачу віртуальні нагороди за те що він не палить протягом певного часу.

Реалізовано інтерфейс користувача у вигляді мобільного додатку, інтерфейс якого зображено на рисунку 1.1

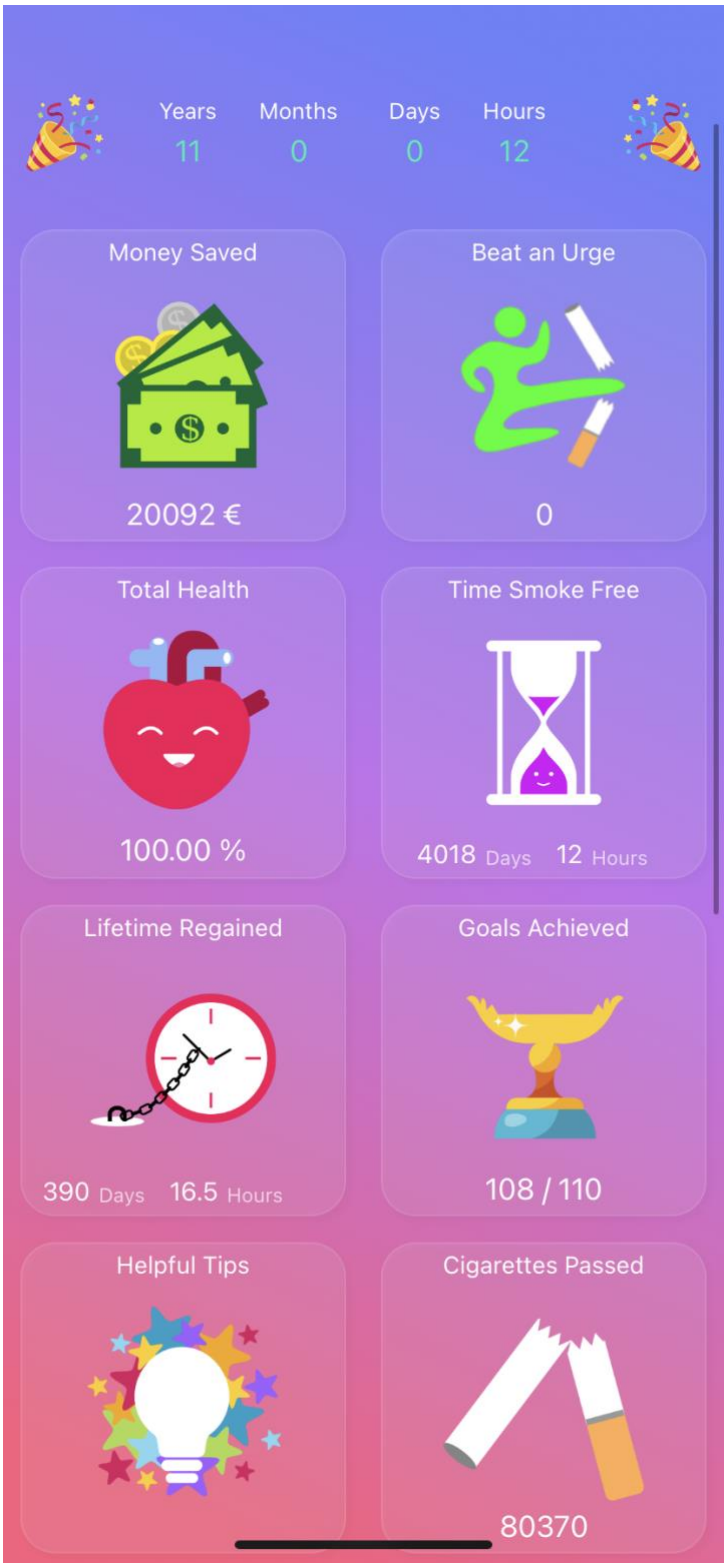


Рисунок 1.1 – Інтерфейс користувача застосунку Easy quit

Переваги:

– Реалізовано систему винагород користувача

Недоліки:

– Відсутня можливість збереження та відновлення даних

– Застосунок надає лише загальну статистику, а не персоналізовану
– Застосунок платний

Quit smoking get healthy – цей застосунок також збирає обмежену кількість даних користувача, та не надає рекомендацій. Застосунок показує кількість часу протягом якої користувач не палить, кількість зекономлених коштів, та загальні статистичні дані про покращення здоров’я користувача.

Реалізовано інтерфейс користувача у вигляді мобільного додатку, інтерфейс якого зображено на рисунку 1.2

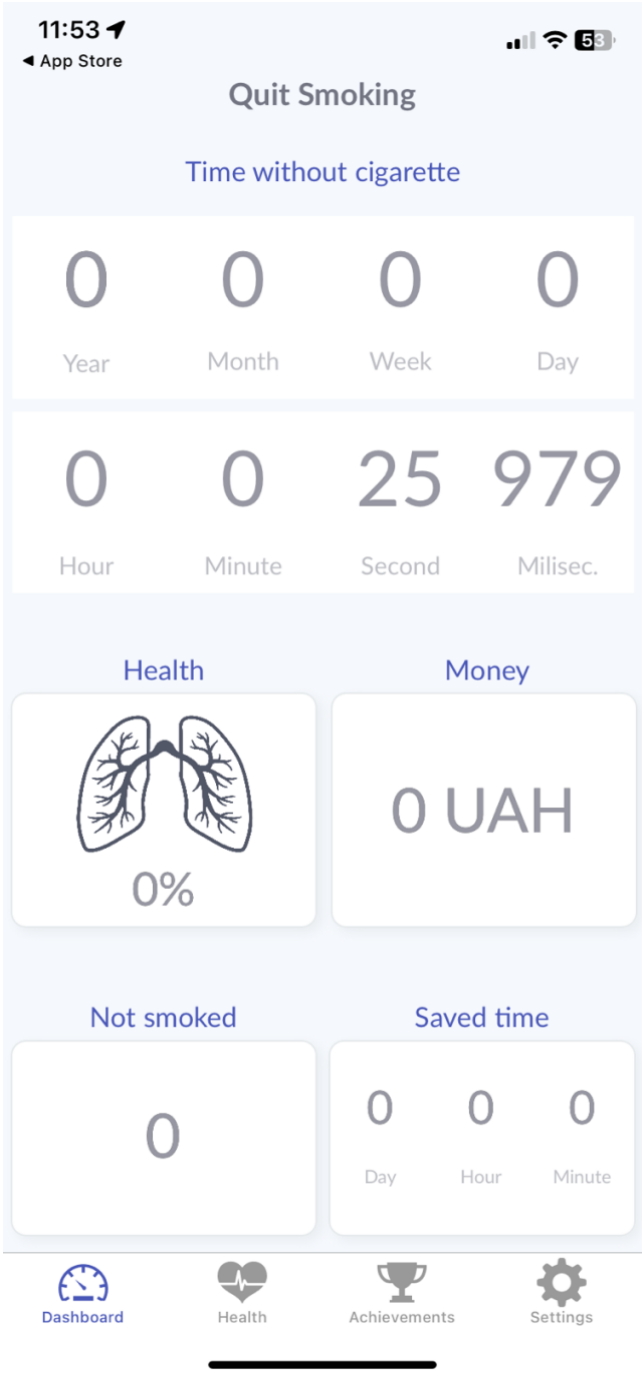


Рисунок 1.2 – Інтерфейс користувача в застосунку Quit smoking get healthy

Переваги:

– Реалізовано систему винагород користувача
– Застосунок надає загальні статистичні дані

Недоліки

– Відсутня можливість збереження та відновлення даних
– Застосунок надає лише загальну статистику, а не персоналізовану

– Застосунок не надає жодних рекомендацій користувачеві

1.4 Аналіз вимог до програмного забезпечення

Головною функцією програмного забезпечення є надання статистики та рекомендацій користувачу, більше функцій можна побачити на рисунку 1.3.

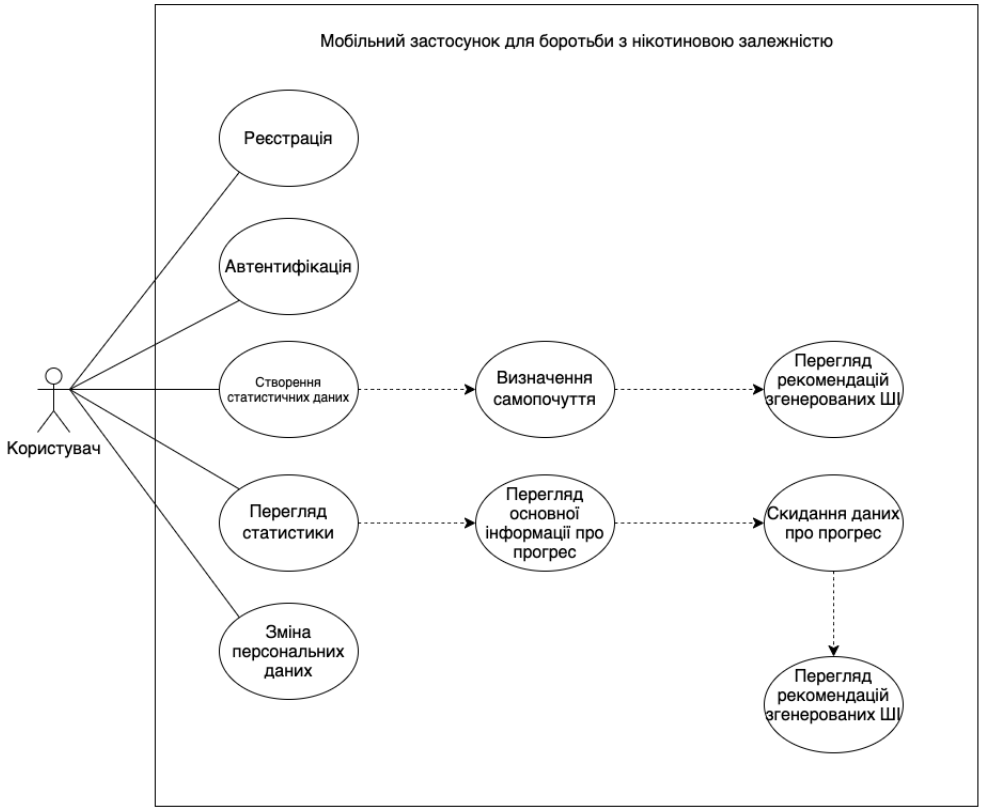


Рисунок 1.3 – Діаграма варіантів використання

В таблицях 1.1 - 1.8 наведені варіанти використання програмного забезпечення.

Таблиця 1.1 - Варіант використання UC-1

Use case name	Реєстрація нового користувача
Use case ID	UC-01
Goals	Реєстрація нового користувача в застосунку
Actors	Незареєстрований користувач
Trigger	Користувач бажає зареєструватися

Pre-conditions	-

Продовження таблиці 1.1

Flow of Events	Користувач відкриває екран реєстрації. В поля для реєстрації вводяться відповідні дані: email користувача, пароль який містить не менше 8 символів та підтвердження підтвердження паролю. Після цього користувач нажимає на кнопку реєстрації. Після цього користувач перенаправляється на екран інформації.
Extension	В випадку введення неправильного email, паролю, або невірного підтвердження паролю, кнопка реєстрації стає неактивною.
Post-Condition	Створення сторінки користувача, перехід на сторінку входу

Таблиця 1.2 - Варіант використання UC-2

Use case name	Автентифікація користувача
Use case ID	UC-02
Goals	Автентифікація користувача в системі
Actors	Зареєстрований користувач
Trigger	Користувач бажає ввійти до системи
Pre-conditions	Користувач зареєстрований
Flow of Events	Користувач запускає застосунок, та вводить логін та пароль за допомогою яких він реєструвався та натискає кнопку входу.
Extension	У випадку спроби входу користувачем за допомогою невірних даних, користувач отримає повідомлення про неможливість входу, та причину, наприклад по тій причині, що він використав електронну пошту яка не була зареєстрована

Post-Condition	Перехід на екран збору додаткової інформації

Таблиця 1.3 - Варіант використання UC-3

Use case name	Збір додаткових даних про користувача
Use case ID	UC-03
Goals	Збір додаткових даних для отримання статистики
Actors	Автентифікований користувач
Trigger	Користувач ввійшов до застосунку вперше
Pre-conditions	Після реєстрації та успішної автентифікації, користувачу необхідно внести більше даних про себе
Flow of Events	Після того як користувач ввійшов до застосунку вперше, йому необхідно ввести дані, а саме: скільки грошей він витрачає на куріння, період протягом якого користувач планує кинути курити, тощо.
Extension	-
Post-Condition	Перехід на головний екран застосунку

Таблиця 1.4 - Варіант використання UC-4

Use case name	Перегляд основної інформації про прогрес користувача
Use case ID	UC-04
Goals	Перегляд статистики
Actors	Автентифікований користувач
Trigger	Користувач бажає переглянути інформацію про свій прогрес
Pre-conditions	Користувач зареєстрований

Flow of Events	Користувач запускає застосунок та потрапляє на головний екран застосунку де може переглянути статистичні дані, наприклад кількість заощаджених коштів, або змінити дані

Продовження таблиці 1.4

Extension	При відсутності доступу до мережі інтернет користувачу відображається помилка та її причина
Post-Condition	-

Таблиця 1.5 - Варіант використання UC-5

Use case name	Визначення самопочуття користувача
Use case ID	UC-05
Goals	Визначити самопочуття користувача
Actors	Автентифікований користувач
Trigger	Користувач бажає додати дані про своє самопочуття до статистики
Pre-conditions	Користувач зареєстрований
Flow of Events	Кожного ранку користувачу на пристрій приходить сповіщення, яке пропонує внести до статистики дані про його самопочуття. Користувач може зробити це або відсканувавши своє обличчя за допомогою фронтальної камери пристрою, або вибравши вручну із запропонованого списку
Extension	Якщо користувач обрав спосіб який визначає самопочуття за допомогою фронтальної камери, але не надав доступу до неї, він отримує повідомлення в якому відображається помилка та її причина.

	Якщо визначене самопочуття, на думку користувача, не відповідає дійсносні, користувач може надати інформації вручну
Post-Condition	Перехід на головний екран застосунку

Таблиця 1.6 - Варіант використання UC-6

Use case name	Перегляд статистики користувача
Use case ID	UC-06
Goals	Перегляд статистики
Actors	Автентифікований користувач
Trigger	Користувач бажає переглянути статистику
Pre-conditions	Користувач зареєстрований
Flow of Events	Користувач може перейти на екран зі статистикою, де йому в вигляді графіків буде показано його зміну його самопочуття за тиждень, та за певний період який він може обрати самостійно. Також відображатимуться певні статистичні дані за вибраний користувачем період
Extension	
Post-Condition	Відображено статистичні дані

Таблиця 1.7 - Варіант використання UC-7

Use case name	Користувач бажає перезапустити таймер
Use case ID	UC-07
Goals	Перезапуск таймеру та статистики
Actors	Автентифікований користувач

Trigger	Користувач знову почав палити
Pre-conditions	Користувач зареєстрований
Flow of Events	Якщо користувач знову почав палити, але все ще бажає кинути, йому необхідно скинути таймер та вказати причину чому саму це сталось

Продовження таблиці 1.7

Extension	-
Post-Condition	Користувачу показується рекомендація щодо того як уникнути такої ситуації наступного разу

Таблиця 1.8 - Варіант використання UC-8

Use case name	Користувач бажає змінити персональні дані
Use case ID	UC-08
Goals	Зміна персональних даних
Actors	Автентифікований користувач
Trigger	Користувач перейшов до екрану зміни персональних даних
Pre-conditions	Користувач зареєстрований
Flow of Events	Користувач переходить до екрану зміни персональних даних, де змінює персональні дані
Extension	В випадку введення не коректних даних, відображається повідомлення про некоректність введених даних
Post-Condition	Повернення до екрану налаштувань

1.4.1 Розроблення функціональних вимог

Також для застосунку для боротьби з нікотиновою залежністю було розроблено функціональні вимоги наведені у таблиці 1.9

Таблиця 1.9 – Функціональна вимога FR-1

FR-1	Автентифікація до системи	Користувач має мати змогу автентифікуватись у системі

Продовження таблиці 1.9

FR-2	Підтримка push notifications	Застосунок має мати змогу взаємодіяти з користувачем за допомогою push notifications
FR-3	Збереження нових користувачів у базу даних	Застосунок має мати змогу зберігати нових користувачів до бази даних
FR-4	Введення персональних даних користувача	Користувач має змогу ввести персональні дані про себе
FR-5	Обробка статистичних даних користувача	Застосунок має оброблювати дані користувача та збирати їх в базі даних
FR-6	Зміна персональних даних	Користувач має змогу змінювати персональні дані в налаштуваннях застосунку
FR-7	Зміна статистичних даних	Користувач має змогу змінювати дані які використовуються для формування статистики
FR-8	Збереження статистичних даних користувачів у базу даних	Застосунок має оброблювати та зберігати статистичні дані користувача у базі даних
FR-9	Визначення	Застосунок має надавати можливіть

	самопочуття користувача	користувачу внести до статистики дані про його самопочуття
FR-10	Перегляд статистичних даних	Користувач має змогу переглянути статистичні дані

Матрицю трасування вимог наведено на рисунку 1.4

	FR-1	FR-2	FR-3	FR-4	FR-5	FR-6	FR-7	FR-8	FR-9	FR-10
UC-1	+		+							
UC-2	+									
UC-3				+						
UC-4										+
UC-5							+		+	
UC-6										+
UC-7							+			
UC-8						+				
UC-9					+					

Рисунок 1.4 – Матриця трасування вимог

Таблиця 1.10 – Модель вимог у загальному вигляді

№	Назва	ID вимоги	Пріоритет	Ризик	Статус

1	Автентифікація до системи	FR-01	Високий	Високий	Підтверджено
2	Підтримка push notifications	FR-02	Низький	Низький	Підтверджено
3	Збереження нових користувачів у базу даних	FR-03	Високий	Високий	Підтверджено

Продовження таблиці 1.10

4	Введення персональних даних користувача	FR-04	Високий	Низький	Підтверджено
5	Обробка статистичних даних користувача	FR-05	Високий	Середній	Підтверджено
6	Зміна персональних даних	FR-06	Високий	Низький	Підтверджено
7	Зміна статистичних даних	FR-07	Середній	Низький	Підтверджено
8	Збереження статистичних даних користувачів у базу даних	FR-08	Низький	Низький	Підтверджено

9	Визначення самопочуття користувача	FR-09	Середній	Низький	Підтверджено
10	Перегляд статистичних даних	FR-10	Низький	Низький	Підтверджено

1.4.2 Розроблення нефункціональних вимог

Для мобільного застосунку для боротьби з нікотиновою залежністю було висунуто наступні нефункціональні вимоги:

– Операційна система: iOS версії 16.0 і вище

– Мова інтерфейсу користувача: Англійська

– Авторизація та автентифікація мають бути виконані з використанням системи авторизації, яка забезпечує надійне зберігання клієнтських даних і паролі

1.5 Постановка задачі

Метою даного дипломного проєкту є покращення взаємодії користувача з health-care застосунками за допомогою ШІ, отже задачами дипломного проєкту будуть:

– Надати користувачеві дані про його прогрес в процесі боротьби з нікотиновою залежністю

– Надати користувачеві статистичні дані

– За допомогою ШІ користувачеві рекомендації базуючись на його самопочутті та історії самопочуття

– Забезпечити надійне зберігання даних користувача

– Надати користувачеві надійні способи автентифікації в застосунку

Висновки до розділу

В даному розділі було проведено аналіз предметної області, аналіз відомих аналогів та аналіз їх переваг та недоліків. Також було надано можливі варіанти використання застосунку та сформульовано задачі. Все це надає нам змогу приступити до подальшого моделювання та розробки програмного забезпечення.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Для опису бізнес процесу програмного забезпечення використовується BPMN модель (рисунки 2.1 – 2.5).

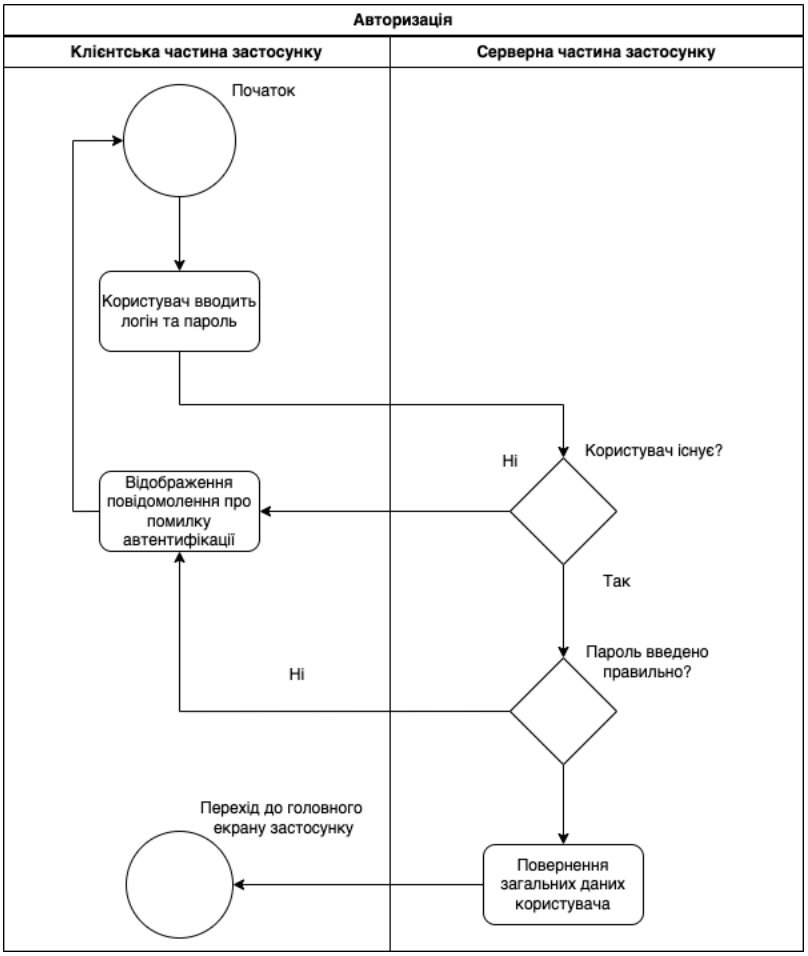


Рисунок 2.1 – Схема бізнес-процесу авторизації користувача

Опис бізнес-процесу авторизації:

– Користувач запускає застосунок
– Користувач вводить логін та пароль
– Логін та пароль надсилаються на сервер та валідуються
– Сервер відправляє загальні дані користувача клієнту
– Клієнт переходить до головного екрану

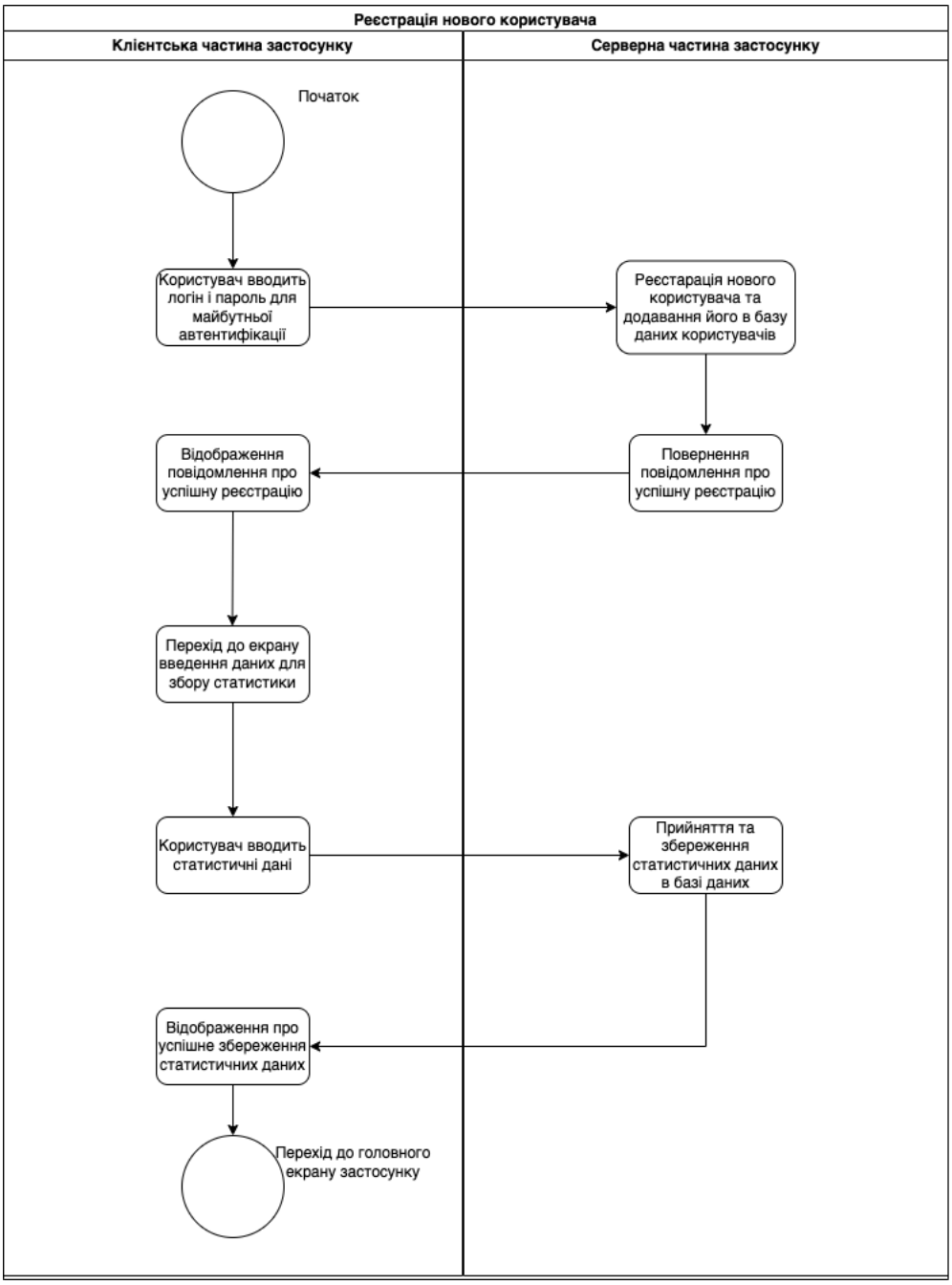


Рисунок 2.2 – Схема бізнес-процесу реєстрації нового користувача

Опис бізнес-процесу реєстрації:

– Користувач запускає застосунок

– Користувач вводить логін та пароль які використовуватиме для майбутньої авторизації у застосунку

– Застосунок відправляє ці дані у шифрованому вигляді до серверу

– Сервер реєструє нового користувача та додає його дані автентифікації до бази даних

– У разі успішної реєстрації сервер повертає повідомлення про успішну реєстрацію користувача

– Клієнт після отримання повідомлення про успішну реєстрацію перенаправляє користувача на екран введення даних для збору статистики та персональних даних

– Користувач заповнює статистичні та персональні дані

– Застосунок перенаправляє користувача до головного екрану застосунку

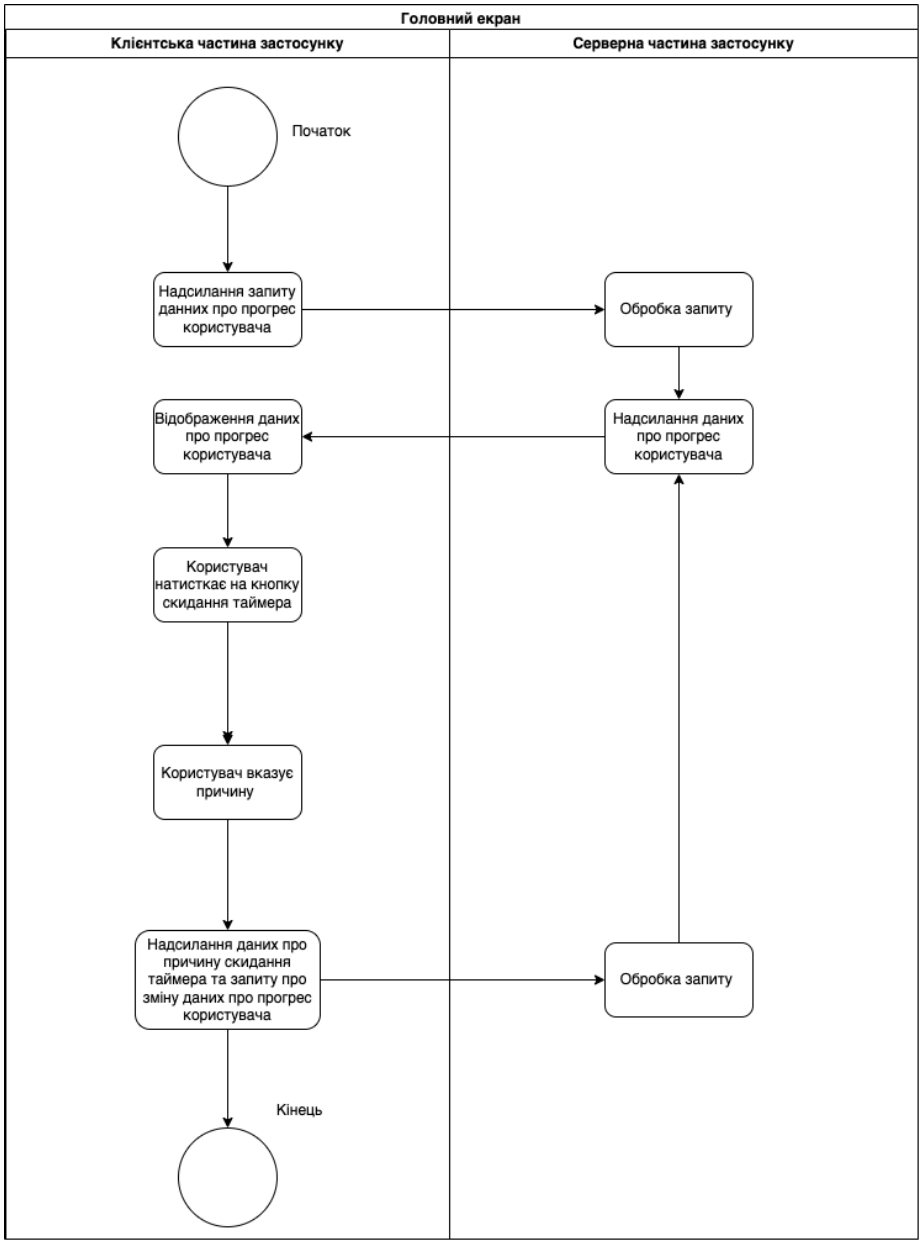


Рисунок 2.3 – Схема бізнес-процесу взаємодії користувача з головним екраном застосунку

Опис бізнес-процесу відображення головного екрану застосунку:

– Користувач відкриває головний екран застосунку

– Клієнт робить запит на сервер про дані користувача

– Сервер приймає, обробляє запит та надсилає клієнту дані про прогрес користувача

– Клієнт відображає отримані дані

– Якщо користувач натиснув кнопку скидування таймеру та підтвердив свої наміри, клієнт надсилає запит до серверу про зміну даних

– Сервер оброблює запит та надсилає оновлені дані клієнту

– Клієнт отримує оновлені дані

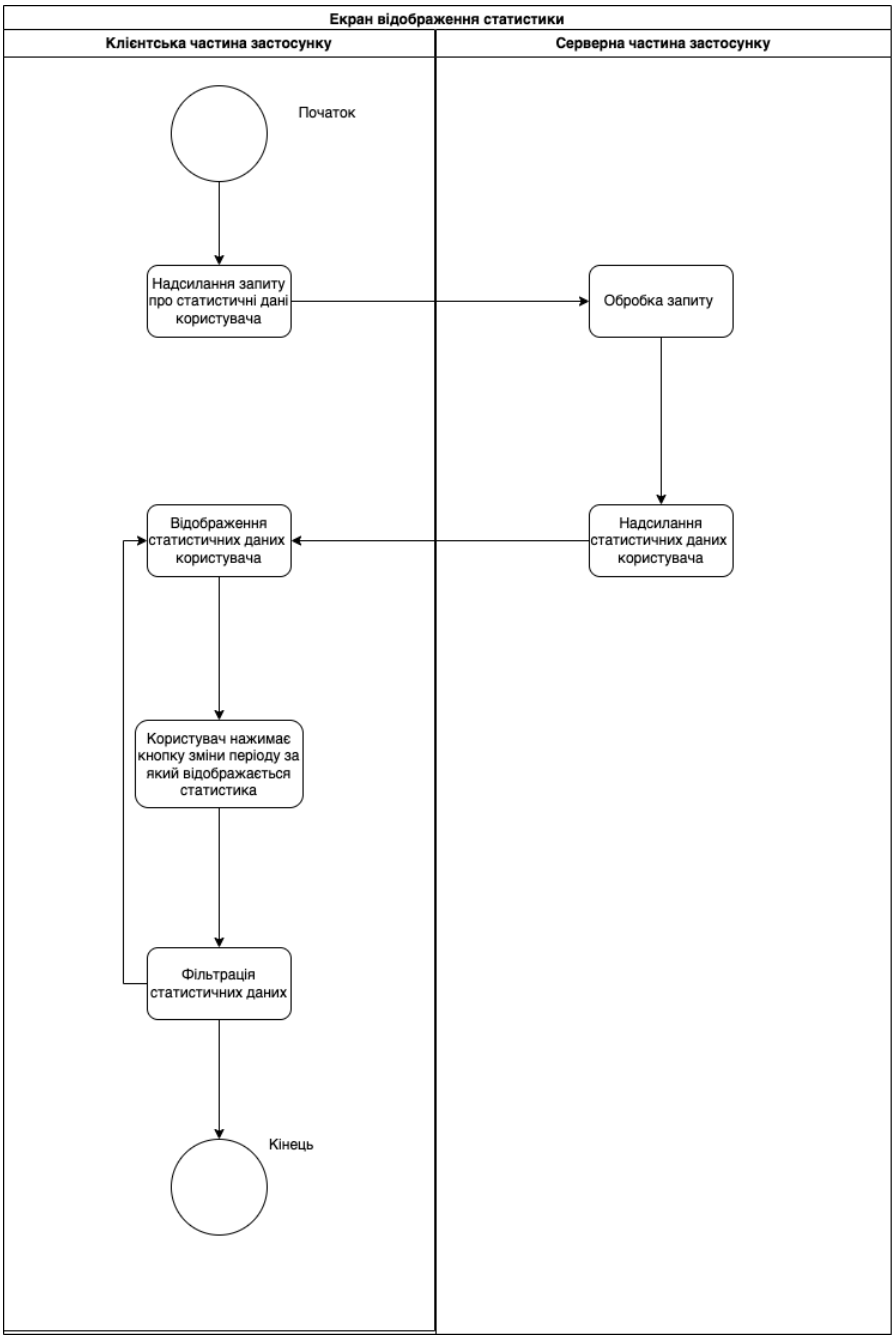


Рисунок 2.4 – Схема бізнес-процесу взаємодії користувача з екраном відображення статистичних даних

Опис бізнес-процесу відображення статистики:

– Користувач відкриває екран відображення статистики

– Клієнт робить запит на сервер про статистичні дані користувача

– Сервер приймає, обробляє запит та надсилає клієнту дані про статистичні дані користувача

– Клієнт відображає отримані статистичні дані

– Якщо користувач натиснув кнопку фільтрації даних то клієнт фільтрує дані за заданим параметром проміжку часу

– Клієнт відображає оновлені дані

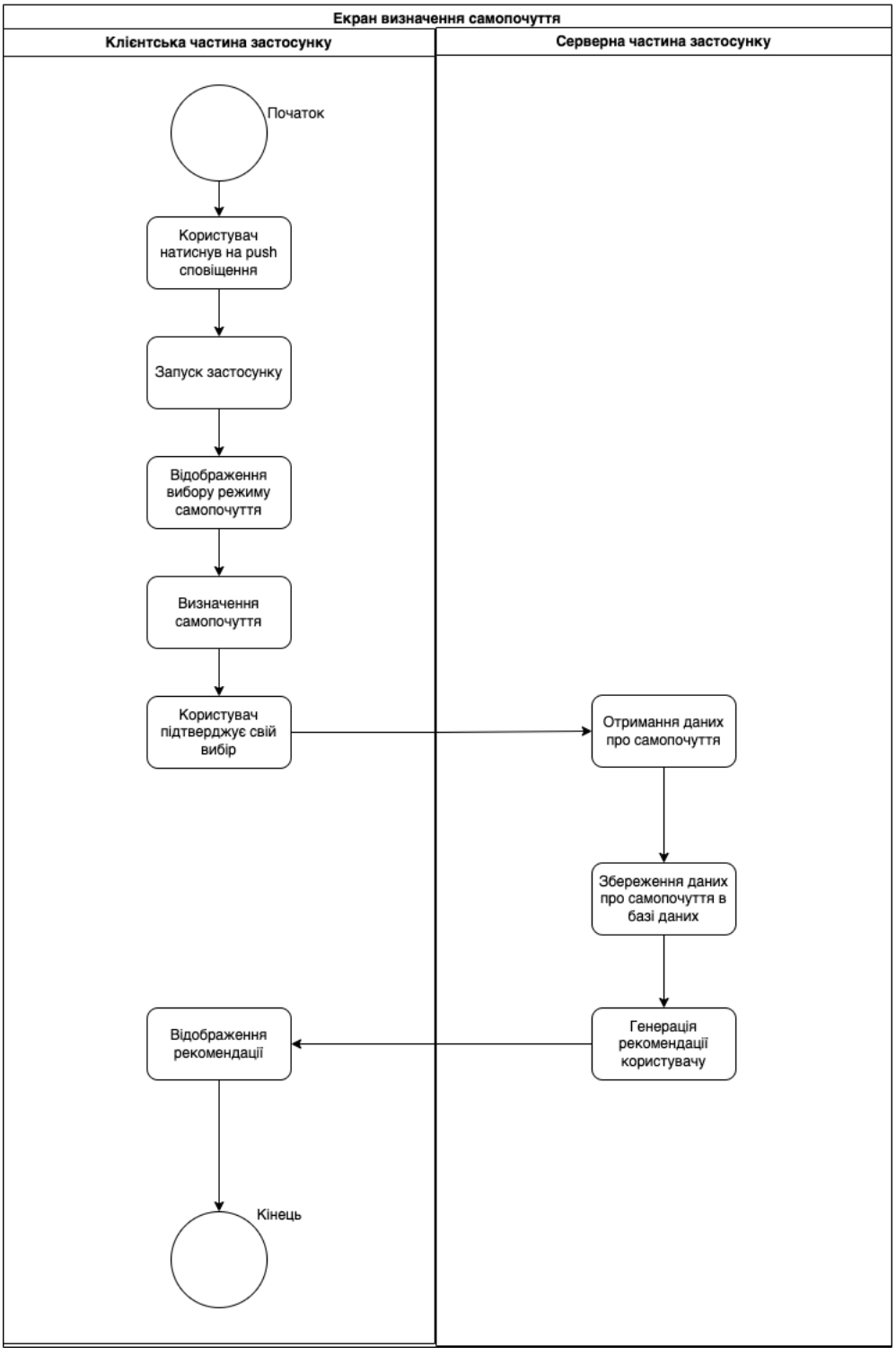


Рисунок 2.5 – Схема бізнес-процесу взаємодії користувача з екраном визначення самопочуття користувача

Опис бізнес-процесу визначення самопочуття користувача:

– Користувачу кожного ранку надсилається push сповіщення з пропозицією додати дані про своє самопочуття

– Користувач натискає на push сповіщення та запускає застосунок

– Користувач вибирає режим визначення самопочуття та переходить на екран визначення самопочуття

– Користувач визначає самопочуття

– Клієнт відправляє отримані дані на сервер

– Сервер генерує рекомендацію базуючись на попередніх даних про самопочуття

– Застосунок перенаправляє користувача на екран з рекомендацією

– Користувач закриває екран з рекомендацією

– Застосунок перенаправляє користувача до головного екрану

2.2 Архітектура програмного забезпечення

Мобільний застосунок було створено з використанням клієнт-серверної архітектури. Клієнт-серверна архітектура — це архітектура, при використанні якої клієнт і сервер спілкуються один з одним через мережу інтернет, тобто коли клієнтська частина робить запити до серверної частини, а вона відповідає даними[3].

В даній роботі клієнтською стороною виступає мобільний застосунок, який взаємодіє з користувачем і відображає інформацію. Сервер, з іншого боку, надає серверні послуги зберігання даних, необхідних клієнту.

Клієнтська сторона застосунку буде побудована з використанням архітектури Model-ViewModel-View(MVVM), який складається з наступних шарів:

Model: цей шар відповідає за роботу за даними, та бізнес логіку.

ViewModel: відповідає за надання даних з моделі для відображення їх в View.

View: відповідає за відображення даних, та інформування шару ViewModel про зміни внесені користувачем.

Візуальна репрезентація архітектури MVVM зображена на рисунку 2.6

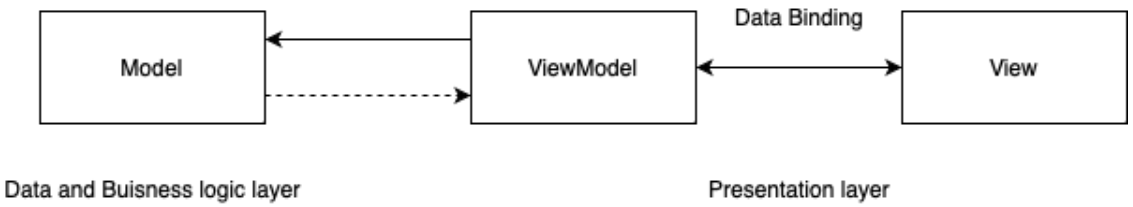


Рис. 2.6 - Візуальна репрезентація архітектури MVVM

В розробленому мобільному застосунку кожен модуль міститиме ViewModel, задачами якого будуть:

– Завантаження даних з серверної частини застосунку
– Відображення даних
– Обробка дій користувача

2.3 Конструювання програмного забезпечення

Клієнтську частину застосунку буде розділено на 6 окремих модулів:

– AuthentificationModule
– RegistrationModule
– MainScreenModule
– MoodClassifierModule
– ProgressChartsModule
– ReasonsToStopModule

Нижче, в таблицях 2.1–2.6 наведено короткий опис кожного модуля, його призначення та короткий опис методів

AuthentificationModule надає користувачу доступ до автентифікації та отримання даних з бази даних

Таблиця 2.1 – Опис методів модулю “AuthentificationModule”

didSelectLoginWithEmailLogin	Відповідає за автентифікацію користувача за допомогою електронної пошти та паролю

RegistrationModule – це модуль який надає користувачу змогу зареєструватись в застосунку використовуючи електронну пошту та пароль

Таблиця 2.2 – Опис методів модулю “RegistrationModule”

didSelecRegisterWithEmailLogin	Відповідає за реєстрацію користувача за допомогою електронної пошти та паролю

MainScreenModule – це модуль який надає користувачу загальну інформацію про його прогрес

Таблиця 2.3 – Опис методів модулю “MainScreenModule”

getUserModel	Відповідає за викочування даних з бази даних для подальшої обробки
setupPiplines	Відповідає за виклик методів які займаються обробкою даних користувача
getMoneySavedOnSigarets	Відповідає за розрахунок та відображення коштів які користувач зекономив після того як перестав палити
getPercentsToFinish	Відповідає за розрахунок та відображення прогресу користувача в відсотках
getTodayDate	Відповідає за відображення сьогоднішньої дати користувачу

MoodClassifierModule – це модуль який відповідає за визначення самопочуття користувача. Користувач може вибрати, як він себе почуває з запропонованого списку, або обрати режим визначення самопочуття за допомогою штучного інтелекту.

Таблиця 2.4 – Опис методів модулю “MoodClassifierModule”

handleMoodSelection	Відповідає за обробку вибраного користувачем самопочуття зі списку
didTapOnDoneButton	Відповідає за збереження оброблених даних в базі даних
validateImage	Відповідає за валідацію зображення наданого користувачем для визначення самопочуття за допомогою штучного інтелекту
detectMood	Відповідає за визначення самопочуття за допомогою штучного інтелекту

ProgressChartsModule – це модуль який за допомогою графіків відображає повну статистику користувача за вибраний ним період часу.

Таблиця 2.5 – Опис методів модулю “ProgressChartsModule”

getChartsData	Відповідає за отримання даних з бази даних
filterChartsData	Відповідає за фільтрацію та відображення даних про прогрес користувача за певний період
getStatistics	Відповідає за обробку даних отриманих бази даних

ReasonsToStopModule – це модуль який відповідає за уточнення обставин за яких користувач знову почав палити

Таблиця 2.6 – Опис методів модулю “ ReasonsToStopModule ”

showArrayOfReasons	Відповідає за надання користувачеві вибору можливих причин чому він знову почав курити
didTapDoneButton	Відповідає за згоду користувача з вибраним пунктом, на подальшою його обробкою

2.3.1 Вибір інструментів для написання клієнтської частини застосунку

Оскільки для написання клієнтської частини застосунку було обрано мову програмування Swift для створення користувацького інтерфейсу застосунку у розробників є вибір з двух фреймворків:

– UIKit

– SwiftUI

UIKit — це потужний фреймворк, який надає інструменти для створення користувацького інтерфейсу з використанням підходу імперативного програмування, тобто розробники визначають кроки, які програма має виконати для досягнення певної мети. UIKit надає розробникам повний контроль над створенням інтерфейсу користувача, але оскільки він підтримує всі версії iOS починаючи з версії 2.0, він містить в собі певні обмеження, які не завжди є доречними при створенні сучасних користувацьких інтерфейсів[4]. До того ж при використанні UIKit застосовується шаблон MVC архітектури замість MVVM.

SwiftUI — це новіший за UIKit фреймворк для створення користувацьких інтерфейсів. Цей фреймворк використовує декларативний підхід для створення інтерфейсу користувача, тобто розробники описують бажаний результат, а фреймворк обробляє деталі реалізації[5], а отже пропонує простіший синтаксис, ніж UIKit, до того ж при використанні

SwiftUI використовується саме шаблон MVVM. Але він має ряд обмежень в вигляді більших вимог до пристроїв.

Отже ключовими відмінностями між ними є:

– Використання різних підходів програмування
– Вимоги до пристрою користувача
– Використання різних архітектурних рішень

Ці фреймворки мають свої переваги та недоліки, але завдяки UIViewControllerRepresentable та UIHostingController ми маємо змогу використовувати елементи обох фреймворків водночас, а отже дозволяє використовувати архітектуру MVVM незалежно від обраного фреймворку. Тому для створення інтерфейсу користувача було вирішено використовувати UIKit та SwiftUI водночас.

Також задля забезпечення обробки асинхронних запитів було вирішено використовувати фреймворк Combine, який входить в стандартний набір SDK Swift. Фреймворк Combine — це декларативний фреймворк Swift, який надає уніфікований API для обробки значень у часі[6]. Це дозволяє розробляти застосунки за допомогою реактивного, функціонального та декларативного підходів. за допомогою паблішерів і підписників, які обробляють асинхронні події.

Combine дозволяє розробникам писати менше коду необхідного для обробки асинхронних подій, що значно спрощує розробку. Також Combine надає розробникам набір інструментів для обробки асинхронних подій у різних типах джерел даних, таких як отримання та надсилання даних до серверної частини застосунку та обробки подій, які відбуваються у інтерфейсі користувача. Це надає змогу перевикористання коду, а також полегшує тестування асинхронних подій.

Ще одна суттєва перевага Combine полягає в тому, що він дозволяє легко складати паблішерів і підписників. Видавців можна об’єднати, щоб створити більш складну поведінку, наприклад об’єднати кілька запитів API або об’єднати дані з різних джерел. Крім того, фреймворк дозволяє

створювати власних паблішерів і підписників, що забезпечує розробникам гнучкість у створенні більш конкретної поведінки.

Таким чином, платформа Combine спрощує код і зменшує ймовірність помилок під час обробки асинхронних подій. Він надає уніфікований API для обробки асинхронних подій у різних джерелах даних і дозволяє легко комбінувати паблішерів і підписників.

2.3.2 Вибір інструментів для написання серверної частини застосунку

Для написання серверної частини застосунку, яка відповідає за автентифікацію, збереження даних, та їх обробку було вибрано інструменти Firebase. Даний фреймворк надає ряд інструментів, які полегшують розробникам створення та розгортання програм без необхідності керувати складною інфраструктурою[7].

Деякі з ключових функцій Firebase включають:

– База даних яка працює у реальному часі: база даних NoSQL, дозволяє розробникам зберігати та синхронізувати дані в режимі реального часу, що полегшує створення додатків, які потребують оновлення даних в реальному часі.

– Автентифікація: проста та безпечна система автентифікації, яка підтримує вхід за допомогою електронної адреси та паролю, вхід із соціальних мереж та інші популярні методи автентифікації.

– Хмарне сховище: масштабоване та безпечне хмарне сховище, яке дозволяє розробникам зберігати та обслуговувати створений користувачами вміст, наприклад зображення та відео.

2.3.3 База даних

Основна частина даних буде зібрана в хмарному сховиші Firebase Realtime Database. Firebase Realtime Database – це хмарна база даних NoSQL, яка має вбудовані засоби для роботи з мобільними застосунками, та має ряд переваг, а саме:

– Синхронізація в реальному часі: Firebase Realtime Database забезпечує синхронізацію даних у реальному часі між усіма підключеними до неї клієнтами. А отже будь-які зміни, внесені до бази даних, негайно поширюються між всіма клієнтами

– Масштабованість: Оскільки Firebase Realtime Database це хмарна база даних вона легко масштабується, що дозволяє оброблювати великі обсяги даних і оброблювати запити великої кількості користувачів без зниження швидкості.

Отже для збереження даних користувача було обрано NoSQL базу даних Firebase

Схему бази даних зображено на рисунку 2.7:

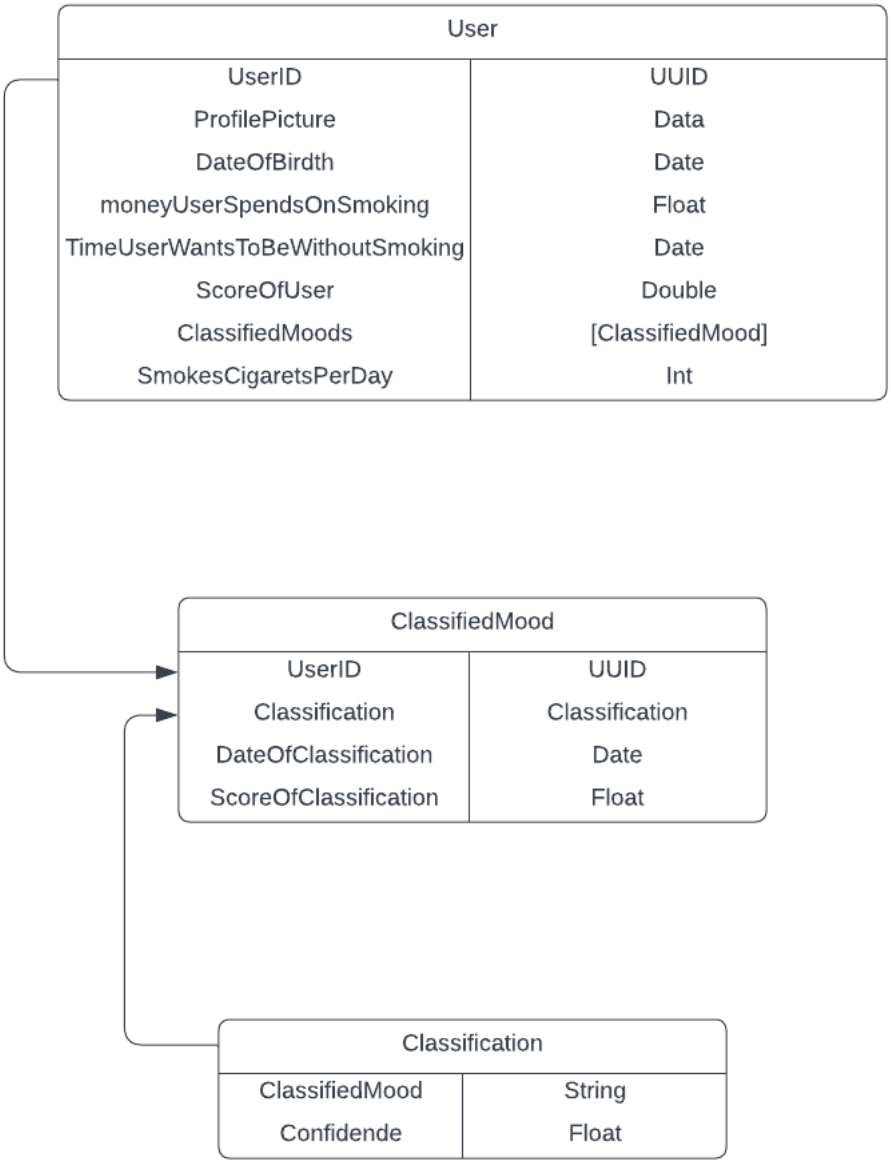


Рисунок 2.7 – Схема бази даних застосунку

Детальний опис користувацьких сутностей бази даних наведено у таблицях 2.7–2.10

Таблиця 2.7– Опис таблиці “User”

Назва таблиці	User
Призначення	Відповідає за відображення даних про користувача
Опис атрибутів таблиці	

Продовження таблиці 2.8

UserID	Відповідає за збереження унікального ідентифікатора користувача
ProfilePicture	Відповідає за збереження фотографії профілю користувача
DateOfBirdth	Дата народження користувача
MoneyUserSpendsOnSmoking	Кількість грошей яку користувач зазвичай витрачає на покупку сигарет чи інших засобів куріння в день
TimeUserWantsToBeWithoutSmoking	Кількість часу протягом якої користувач планує кинути курити
ScoreOfUser	Загальний бал користувача
ClassifiedMoods	Масив данних, що містить в собі данні самопочуття користувача за весь період використання застосунку
SmokesCigaretsPerDay	Кількість цигарок яку викурює користувач за день

Таблиця 2.9 – Опис таблиці “ClassifiedMood”

Назва таблиці	ClassifiedMood
Призначення	Відповідає за опис визначеного самопочуття користувача
Опис атрибутів таблиці	
UserID	Відповідає за збереження унікального ідентифікатора користувача
Classification	Самопочуття користувача

Продовження таблиці 2.9

DateOfClassification	Дата коли було визначено самопочуття користувача
ScoreOfClassification	Бал який користувач отримав за визначене самопочуття

Таблиця 2.10 – Опис таблиці “Classification”

Назва таблиці	Classification
Призначення	Відповідає за збереження даних про класифікацію самопочуття користувача
Опис атрибутів таблиці	
ClassifiedMood	Визначене самопочуття користувача
Confidence	Впевненість моделі ШІ в визначеному результаті

2.4 Аналіз безпеки даних

Використання системи автентифікації Firebase Auth гарантує, що конфіденційні дані користувача, такі як паролі, не зберігатиметься в базі даних клієнтської сторони застосунку. Натомість Firebase Auth повертає клієнтській стороні JSON Web Token (JWT), який надсилається на сервер разом із запитами та перевіряється за допомогою інструментів Firebase. Цей підхід гарантує безпечну автентифікацію користувача та запобігає несанкціонованому доступу до конфіденційних даних.

Розроблений застосунок також уникає зберігання будь-яких особистих даних користувачів, за винятком інформації, пов'язаної з курінням. Адреса електронної пошти, надана користувачем під час створення замовлення, використовується лише для відправки електронного листа з підтвердженням і не зберігається в базі даних.

Також для забезпечення безпеки даних у розробленому застосунку дані перевіряються як на стороні клієнта, так і на стороні сервера. Перевірка на стороні клієнта використовує спеціальні правила перевірки для перевірки даних, наприклад адрес електронної пошти

Використання Firebase для автентифікації та зберігання бази даних у поєднанні з обмеженим збором даних і перевіркою введених даних допомагає мінімізувати ризик витоку даних або несанкціонованого доступу до конфіденційної інформації.

Окрім уже згаданих заходів безпеки, застосунок також використовує безпечні протоколи автентифікації під час підключення з Firebase. Додаток використовує автентифікацію Firebase, яка підтримує кілька методів автентифікації, включаючи електронну адресу/пароль, номер телефону та сторонніх постачальників, таких як Google і Facebook.

Коли користувач реєструється або входить, його облікові дані безпечно передаються за допомогою HTTPS, безпечного протоколу зв'язку, який шифрує дані під час передачі між програмою та серверами Firebase. Це гарантує, що облікові дані користувача не будуть перехоплені або скомпрометовані під час передачі.

Крім того, програма використовує базу даних Firebase у реальному часі, яка є хмарною базою даних NoSQL, яка підтримує безпечний зв'язок за допомогою HTTPS. Програма отримує та надсилає дані до бази даних за допомогою протоколу HTTPS, який шифрує дані під час передачі, гарантуючи, що дані не можуть бути перехоплені чи змінені під час передачі.

Таким чином, програма для припинення куріння використовує безпечні протоколи автентифікації під час реєстрації користувача та входу в систему, а весь зв'язок із базою даних у реальному часі Firebase шифрується за допомогою HTTPS. Ці заходи гарантують безпеку даних користувача та облікових даних для входу, а також мінімізують ризик перехоплення чи модифікації даних.

Висновки до розділу

В даному розділі було змодельовано та детально описано бізнес процеси. Також було розглянуто особливості архітектури MVVM. Окрім цього було проаналізовано безпеку даних користувача, детально описано інструменти які за допомогою яких буде проведено практичну реалізацію застосунку та описано модулі застосунку та їх методи. Моделювання та аналіз дозволяє провести реалізацію застосунку та його тестування.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Перед тим як фіналізувати розробку програмного забезпечення, необхідно оцінити його якість. Зробити це можна проаналізувавши код і протестувавши роботу застосуноку

Для того щоб провести аналіз коду в даній роботі було використано застосунок embold. Результати аналізу наведені на рисунку 3.1

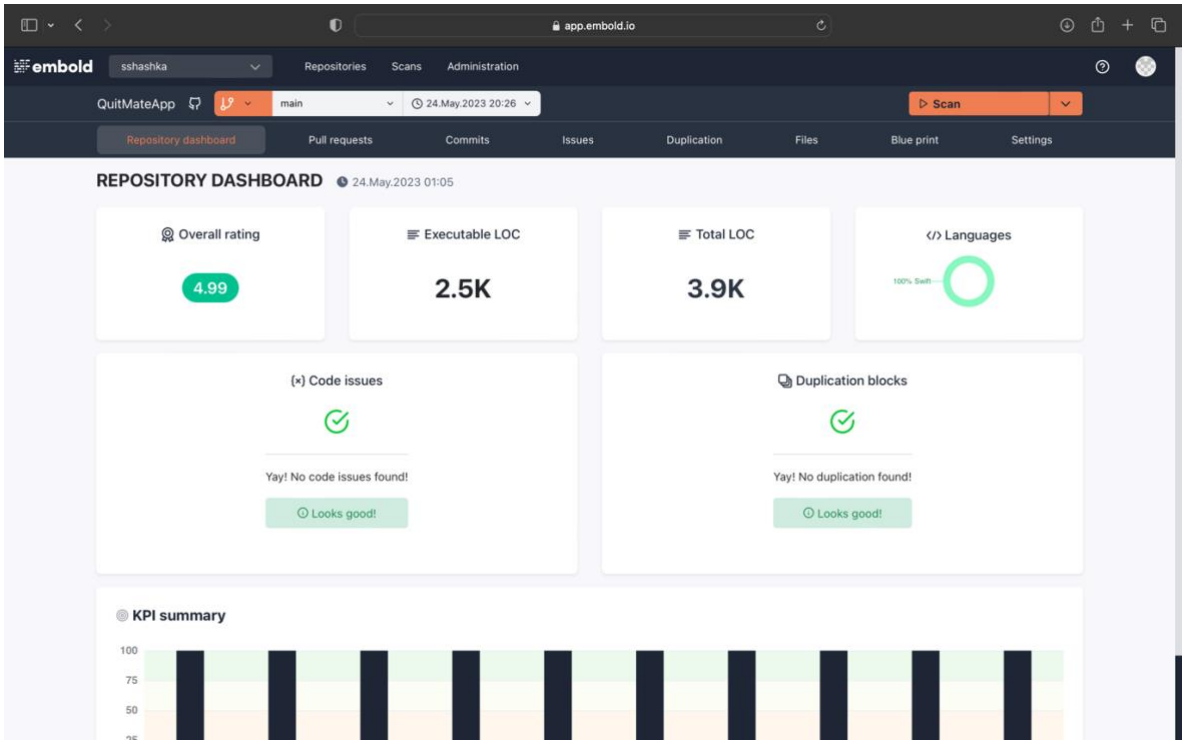


Рисунок 3.1 – Результати аналізу коду за допомогою сервісу embold.io

Даний застосунок проводить аналіз коду за наступними параметрами:

– Пошук проблеми з кодом та вразливих місць
– Повторювання коду
– Використання антипаттернів

Для тестування роботи застосунку було обрано мануальне тестування

3.2 Опис процесів тестування

В даній роботі було виконане мануальне тестування функціональної частини програмного забезпечення. Опис відповідних тестів наведено у таблицях 3.1 – 3.9

Таблиця 3.1 – Тест 1.1

Тест	Реєстрація користувача
Модуль	Реєстрація користувача
Номер тесту	1.1
Початковий стан системи	Відкрито екран реєстрації
Вхідні дані	email, пароль, підтвердження паролю
Опис проведення тесту	У відповідні поля вводяться: правильний та існуючий email, який до цього не був вже використаний для реєстрації та пароль більше 8 символів. Після цього натискається кнопка підтвердження реєстрації.
Очікуваний результат	Реєстрація успішна, дані користувача додаються до БД і він перенаправляється на екран збору персональних даних користувача.
Фактичний результат	Реєстрація успішна, дані користувача додаються до БД і він перенаправляється на екран збору персональних даних користувача.

Таблиця 3.2 – Тест 1.2

Тест	Відображення помилки реєстрації користувача
Модуль	Реєстрації користувача
Номер тесту	1.2
Початковий стан системи	Користувач знаходиться на екрані реєстрації
Вхідні дані	email, пароль, підтвердження паролю

Продовження таблиці 3.2

Опис проведення тесту	У відповідні поля вводяться: некоректний email, пароль. Після цього натискається кнопка підтвердження реєстрації.
Очікуваний результат	Реєстрація не проходить, користувач не додається у систему і користувачу відображається повідомлення про використання неіснуючої адреси.
Фактичний результат	Реєстрація не проходить, користувач не додається у систему і користувачу відображається повідомлення про використання неіснуючої адреси.

Таблиця 3.3 – Тест 1.3

Тест	Підсвічування поля з некоректними даними червоним
Модуль	Реєстрація користувача
Номер тесту	1.3
Початковий стан системи	Користувач знаходиться на сторінці реєстрації
Вхідні дані	email, пароль, підтвердження паролю
Опис проведення тесту	У відповідні поля вводяться: некоректний email або пароль менше 8 символів, або неправильне підтвердження паролю.
Очікуваний результат	Кнопка реєстрації стає недоступною, а поле з неправильними даними підсвічується червоним кольором
Фактичний результат	Кнопка реєстрації стає недоступною, а поле з неправильними даними підсвічується червоним кольором

Таблиця 3.4 – Тест 1.4

Тест	Авторизація користувача
Модуль	Авторизація користувача
Номер тесту	1.4
Початковий стан системи	Користувач знаходиться на сторінці авторизації
Вхідні дані	Електронна пошта, пароль
Опис проведення тесту	У відповідні поля вводяться: коректна електронна пошта та пароль не менше 8 символів
Очікуваний результат	Успішна авторизація користувача та перехід до головного екрану застосунку
Фактичний результат	Успішна авторизація користувача та перехід до головного екрану застосунку

Таблиця 3.5 – Тест 1.5

Тест	Введення неправильних даних для авторизації
Модуль	Авторизація користувача
Номер тесту	1.5
Початковий стан системи	Користувач знаходиться на сторінці авторизації
Вхідні дані	неправильні електронна пошта та пароль
Опис проведення тесту	У відповідні поля вводяться: некоректна електронна пошта та пароль не менше 8 символів
Очікуваний результат	Відмова в авторизації та відображення користувачеві повідомлення в якому вказано причину відмови
Фактичний результат	Відмова в авторизації та відображення користувачеві повідомлення в якому вказано причину відмови

Таблиця 3.6– Тест 1.6

Тест	Визначення самопочуття користувача вручну
Модуль	Визначення самопочуття користувача
Номер тесту	1.6
Початковий стан системи	Користувач знаходиться на сторінці ручного визначення самопочуття
Вхідні дані	Дані про самопочуття
Опис проведення тесту	Користувачу на вибір надається список з варіантами самопочуття, коли колистувач обрав щось з цього списку, кнопка підтвердження стає активною
Очікуваний результат	Кнопка підтвердження вибору стає активною
Фактичний результат	Кнопка підтвердження вибору стає активною

Таблиця 3.7 – Тест 1.7

Тест	Визначення самопочуття користувача автоматично
Модуль	Визначення самопочуття користувача
Номер тесту	1.7
Початковий стан системи	Користувач знаходиться на сторінці автоматичного визначення самопочуття
Вхідні дані	Фото обличчя користувача
Опис проведення тесту	Користувач відкриває екран з автоматичним визначенням свого самопочуття, та робить знімок свого обличчя
Очікуваний результат	Застосунок за допомогою ШІ визначає самопочуття користувача та відображає йому результат
Фактичний результат	Застосунок за допомогою ШІ визначає самопочуття користувача та відображає йому результат

Таблиця 3.8 – Тест 1.8

Тест	Перевірка наявності обличчя в кадрі при визначенні самопочуття користувача автоматично
Модуль	Визначення самопочуття користувача
Номер тесту	1.8
Початковий стан системи	Користувач знаходиться на сторінці автоматичного визначення самопочуття
Вхідні дані	Фото обличчя користувача
Опис проведення тесту	Користувач відкриває екран з автоматичним визначенням свого самопочуття, та робить знімок свого обличчя, але обличчя не попадає в кадр
Очікуваний результат	Застосунок перевіряє чи є обличчя на фото, та повертає повідомлення користувачеві про відсутність обличчя
Фактичний результат	Застосунок перевіряє чи є обличчя на фото, та повертає повідомлення користувачеві про відсутність обличчя

Таблиця 3.9 – Тест 1.9

Тест	Фільтрація статистичних даних користувача за певний період часу
Модуль	Відображення статистичних даних
Номер тесту	1.9
Початковий стан системи	Користувач знаходиться на екрані відображення статистичних даних
Вхідні дані	Період часу за який користувач хоче переглянути дані
Опис проведення тесту	Користувач відкриває екран з який відображає статистичні дані, натискає кнопку зміни періоду, та вибирає за який проміжок часу він хоче переглянути дані
Очікуваний результат	Застосунок фільтрує дані за вибраним користувачем фільтром та відображає їх

Продовження таблиці 3.9

Фактичний результат	Застосунок фільтрує дані за вибраним користувачем фільтром та відображає їх

3.3 Опис контрольного прикладу

Далі протестуємо контрольний приклад. В цьому розділі будуть описані кроки виконання тестування, та ілюстрації до них

Запускаємо застосунок, та потрапляємо до екрану авторизації в застосунку. Зовнішній вигляд екрану авторизації наведено на рисунку 3.2

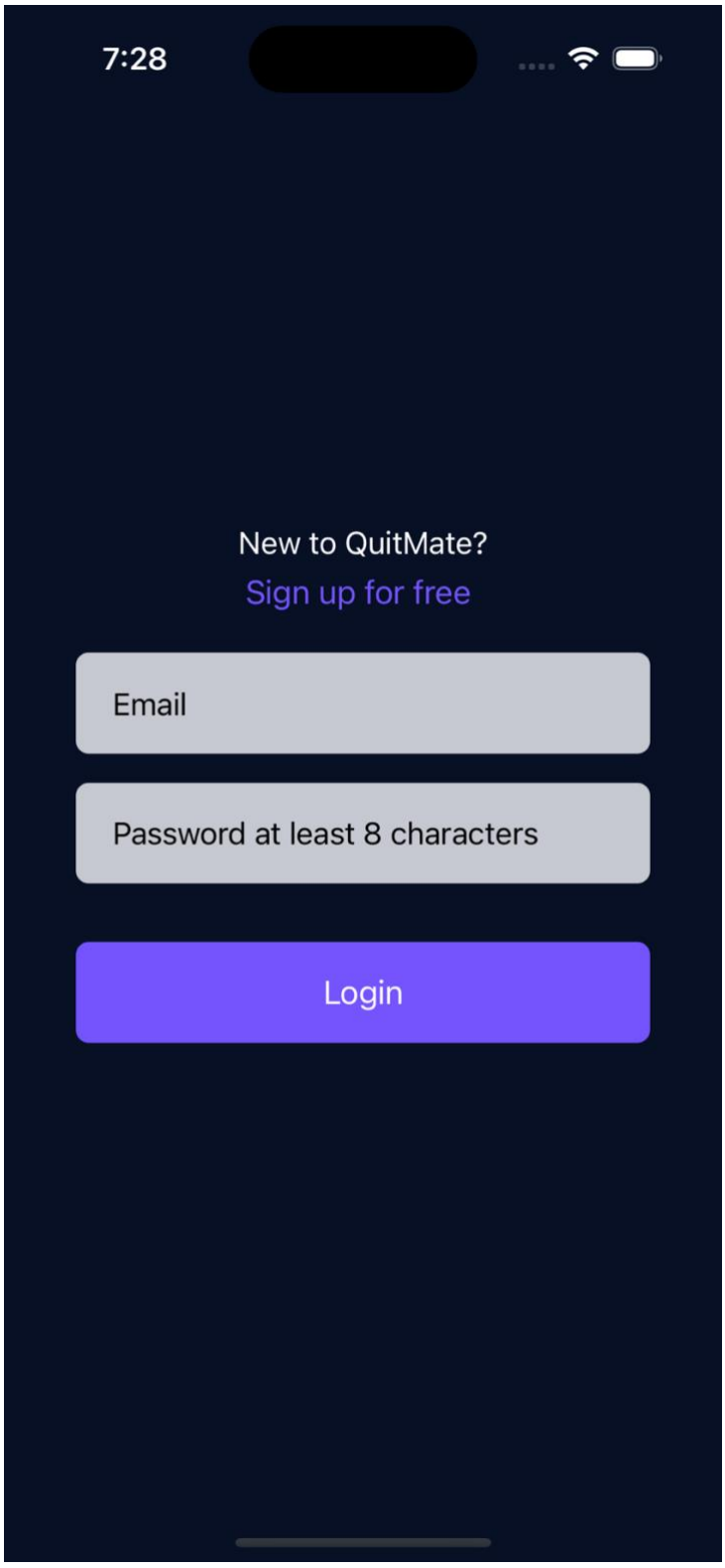


Рис. 3.2 – Зовнішній вигляд екрану авторизації

Для початку введемо неправильні дані, а саме неправильну електронну пошту користувача та натиснемо кнопку авторизіції. Як можна побачити на рисунку 3.3 застосунок відображає повідомлення про те, що користувач ввів невірний пароль

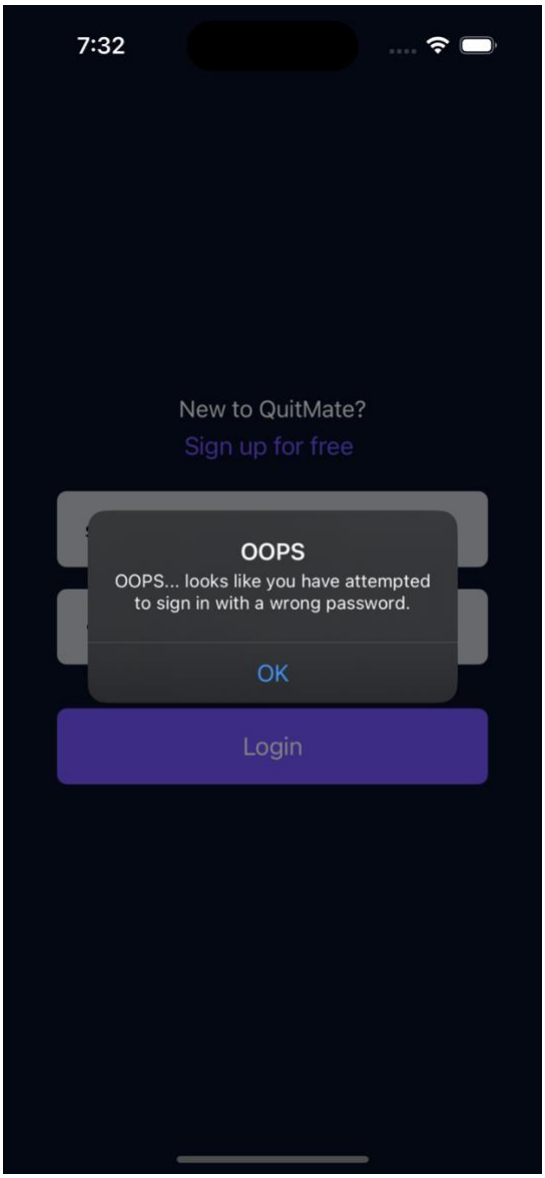


Рисунок 3.3 – Відображення повідомлення про введення неправильного паролю для авторизації

Тепер введемо неправильну пошту. Як можна побачити на рисунку 3.x4 застосунок робить кнопку авторизації неактивною, якщо користувач ввів неправильну електронну пошту

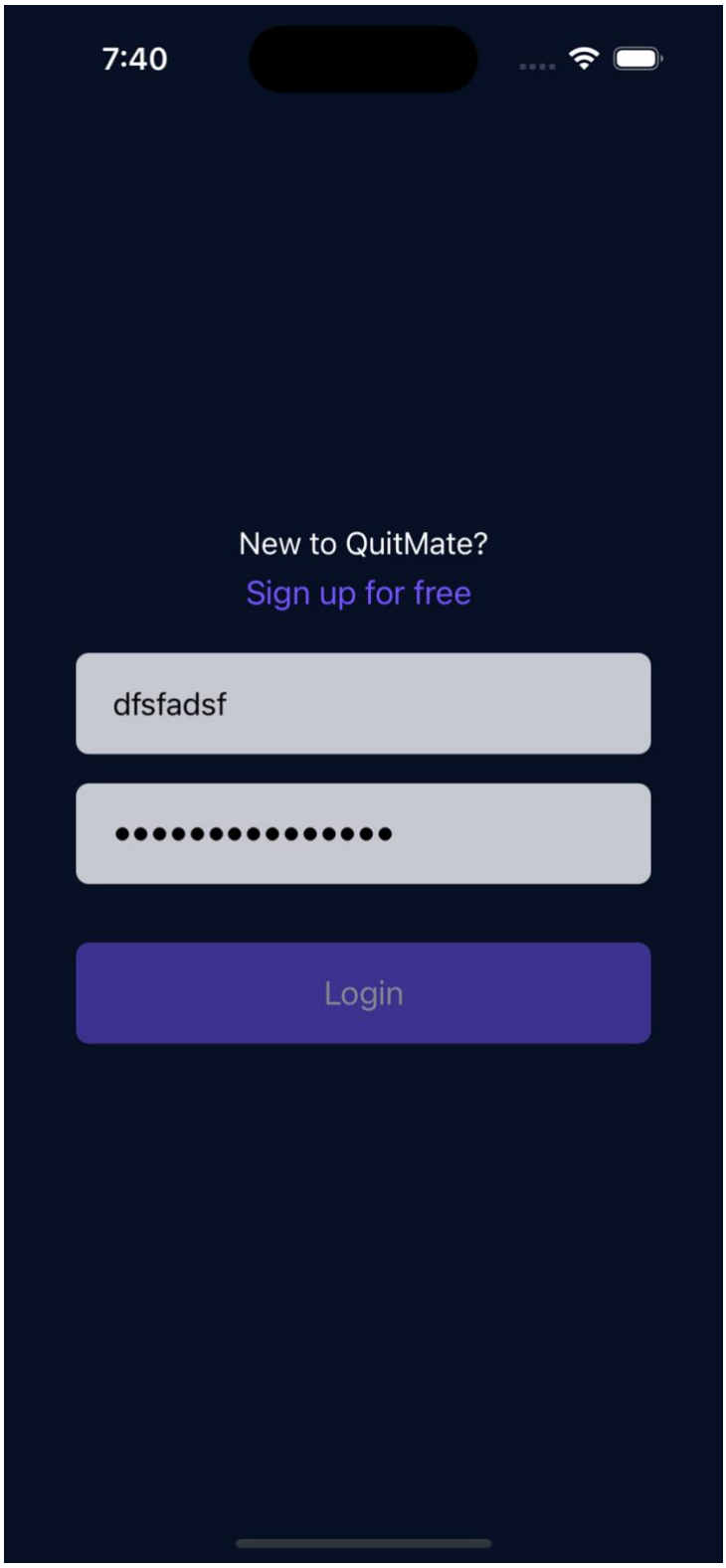


Рисунок 3.4 – Відображення повідомлення про введення неправильного паролю для авторизації

Перейдемо до екрану реєстрації. В ньому користувач має ввести наступні дані:

– Електронну пошту

– Пароль

– Підтвердження паролю

Якщо користувач ввів неправильну електронну пошту застосунок має зробити кнопку реєстрації неактивною та підсвітити поле вводу пошти червоним. На рисунку 3.5 зображено вигляд інтерфейсу застосунку в цьому випадку

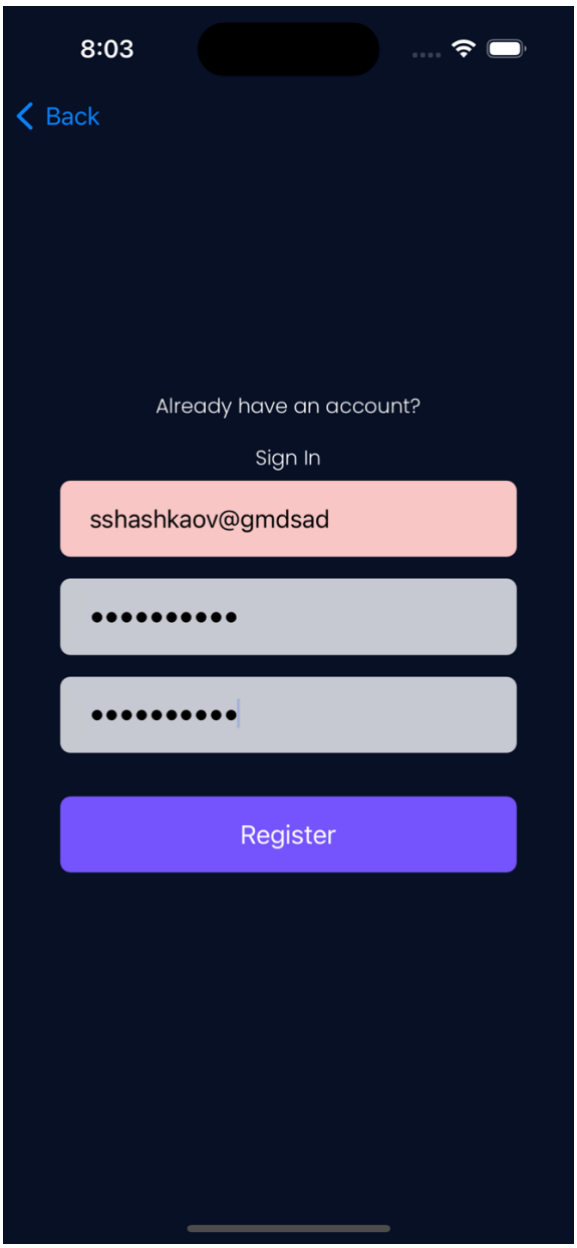


Рисунок 3.5 – Відображення повідомлення про введення неправильного паролю для авторизації

Перейдемо до головного екрану застосунку (рисунок 3.6)

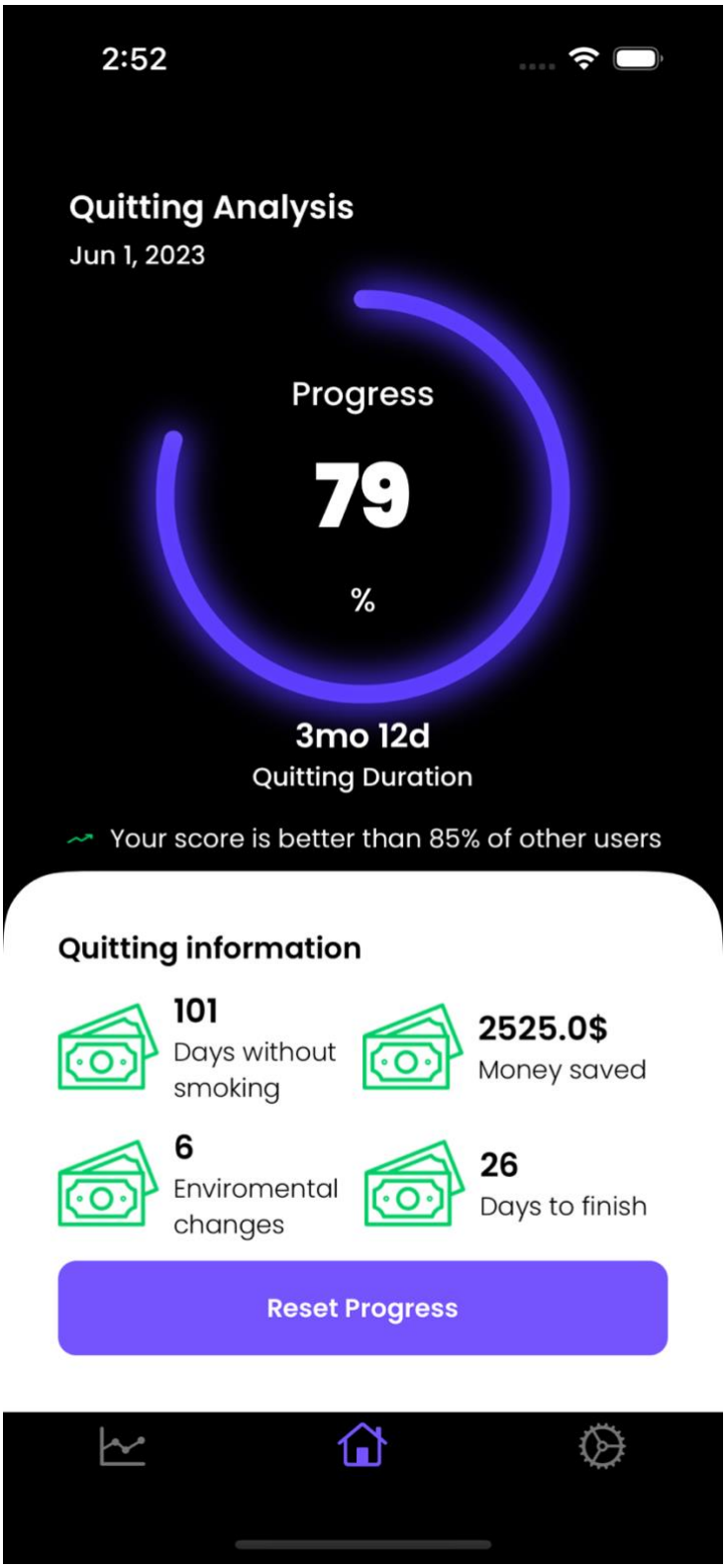


Рисунок 3.6 – Зовнішній вигляд головного екрану застосунку

На головному екрані відображаються статистичні дані користувача. На головному екрані є кнопка яка відповідає за скидання встановлених цілей та статистичних даних користувача. Коли користувач натискає на цю кнопку йому має висвітитись повідомлення з попередженням про те, що ця дія

незворотня та приведе до скидання встановлених цілей та статистичних даних користувача. На рисунку 3.7 зображено вигляд інтерфейсу застосунку в цьому випадку

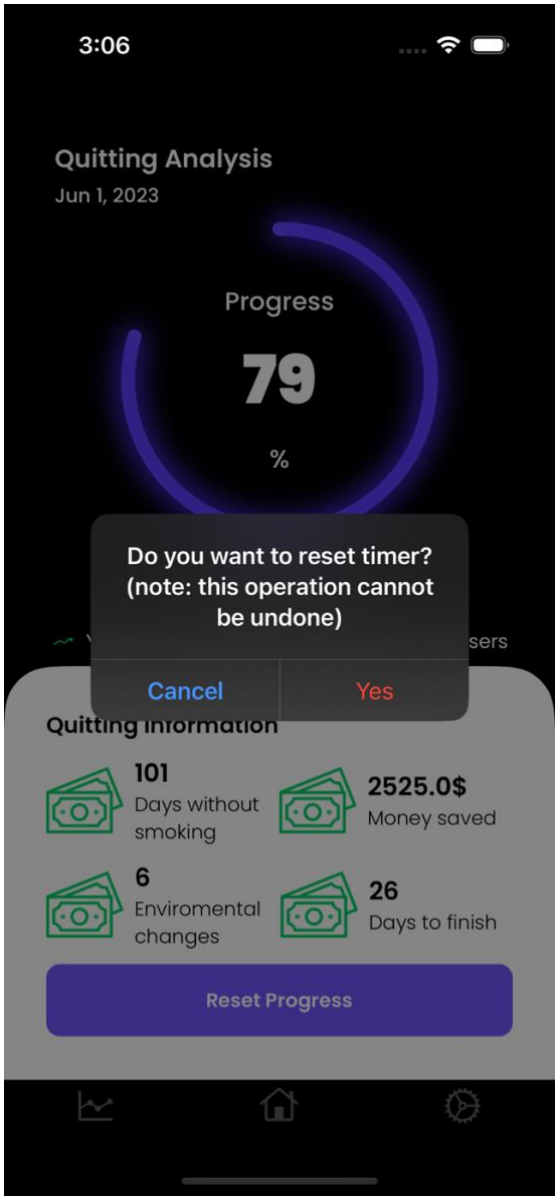


Рисунок 3.7 – Зовнішній вигляд головного екрану застосунку при натисканні на кнопку скидання прогресу

Якщо користувач підтверджує свій вибір, застосунок відображає користувачеві екран, де йому необхідно вибрати, із запропонованого списку, причини чому саме він бажає скинути встановлені цілі та статистичні дані. Зовнішній вигляд екрану наведено на рисунку 3.8

Рисунок 3.8 – Зовнішній вигляд екрану вказання причини скидання прогрегсу

Після того як користувач вказав причини чому він вирішив скинути прогрес застосунок надає йому рекомендації згенеровані ШІ на основі його самопочуття та причини чому він вирішив скинути прогрес (рисунок 3.9).

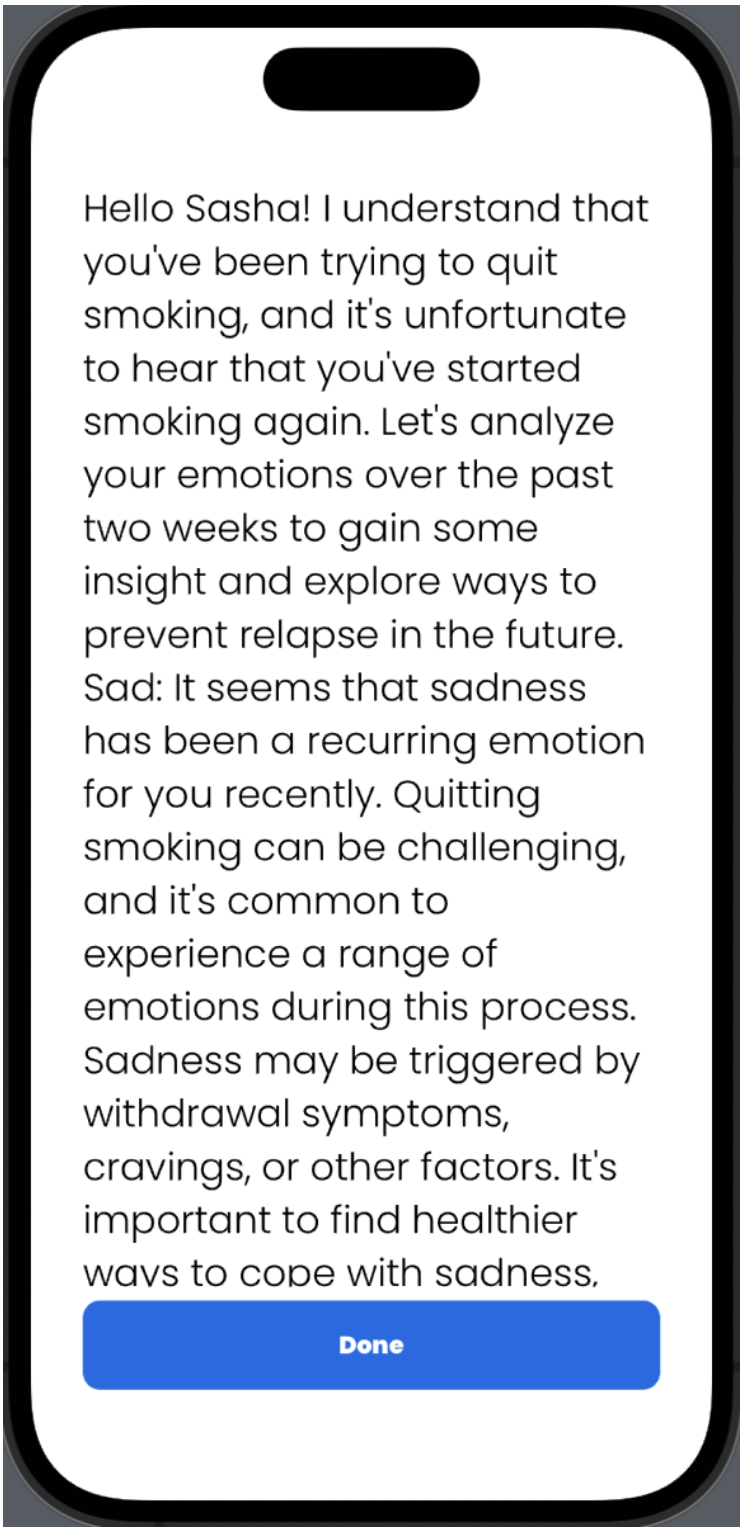


Рисунок 3.9 – Зовнішній вигляд екрану рекомендацій згенерованих ШІ

Також користувач має змогу переглянути зміну свого самопочуття у вигляді графіків. На рисунку 3.10 зображено зовнішній вигляд екрану перегляду даних щодо зміни самопочуття користувача

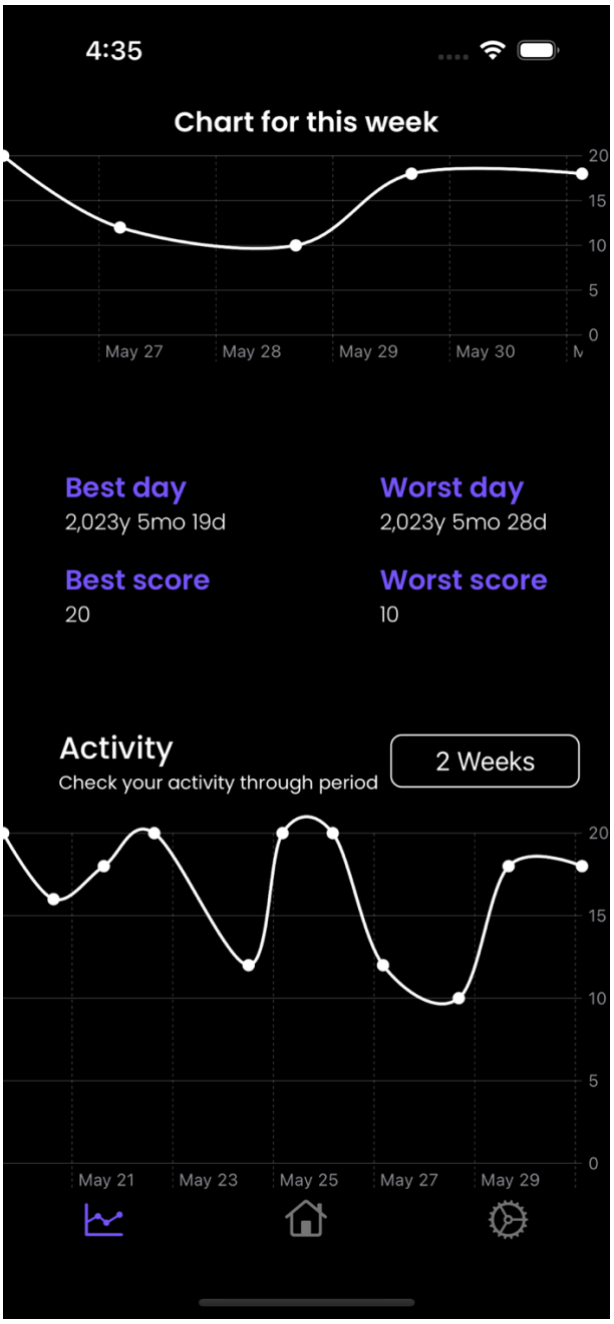


Рисунок 3.10 – Зовнішній вигляд екрану перегляду даних щодо зміни самопочуття користувача

Користувач також може фільтрувати дані за певний період, наприклад: 2 тижні, 1 місяць та 3 місяці. Зовнішній вигляд меню зміни періоду відображено на рисунку 3.11

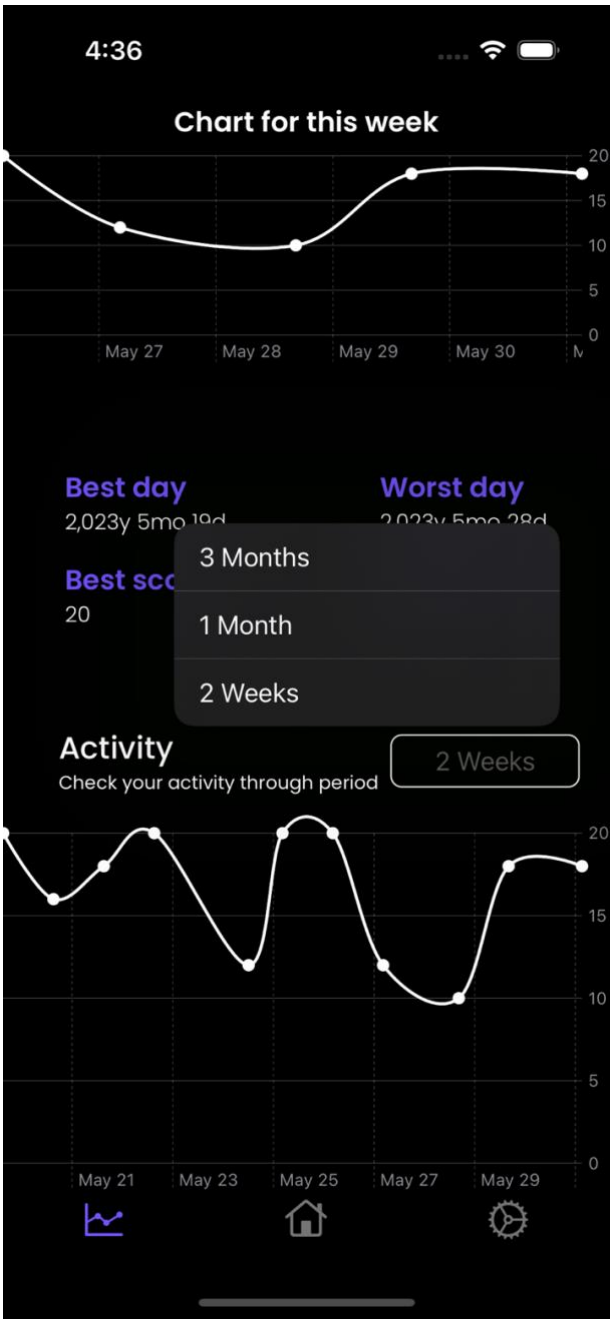


Рисунок 3.11 – Зовнішній вигляд меню зміни періоду

Висновки до розділу

В даному розділі було проведено тестування розробленого застосунку. Тестування показало, що розроблений застосунок було реалізовано в повній мірі. Також тестування показало, що функціональні вимоги було задоволено в повній мірі. Результати тестування є задовільними і це дозволяє перейти до опису процесу розгортання програмного забезпечення.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Розроблений застосунок буде розгорнуто за допомогою App Store Connect. App Store Connect служить центральною платформою для розробників, щоб керувати всім життєвим циклом своїх програм в App Store. Він дає розробникам інструменти для публікації, розповсюдження, моніторингу та оновлення своїх програм, що зрештою допомагає їм охопити глобальну аудиторію.

Ось деякі ключові аспекти та функції App Store Connect:

– Керування програмами: App Store Connect дозволяє розробникам створювати списки своїх програм і керувати ними в App Store. Ви можете надати метадані програми, зокрема назву програми, опис, знімки екрана, ключові слова, категорії та ціни. Він також дає змогу налаштовувати покупки в програмі, підписки та пропонує такі функції, як попередні замовлення для майбутніх програм.

– Надсилання додатків і перевірка: коли додаток буде готовий до надсилання, ми використовуємо App Store Connect, щоб завантажити двійковий файл додатка разом із необхідними ресурсами додатка, такими як значки та знімки екрана. Після надсилання додаток проходить процес перевірки командою Apple App Review, щоб переконатися, що він відповідає вимогам App Store.

– Розповсюдження та випуск: App Store Connect надає параметри для керування доступністю та випуском застосунку. В ньому можна вибрати країни чи регіони, де буде доступний застосунок, установити рівні цін і запланувати дати випуску. Він також підтримує поетапні випуски, що дозволяє поступово розгортати додаток для різних сегментів користувачів.

– Аналітика додатків. Важливо розуміти, наскільки добре працює застосунок в реальному житті і App Store Connect надає комплексні

інструменти аналітики. Ми можемо отримати доступ до детальної статистики про завантаження вашого додатка, дохід, залучення користувачів і показники утримання. Ці аналітичні дані допомагають вам приймати зважені рішення щодо оновлень програм, маркетингових стратегій і оптимізації монетизації.

– Відгуки та оцінки користувачів: App Store Connect дозволяє відстежувати відгуки та оцінки користувачів і відповідати на них. Ми можемо взаємодіяти з користувачами, відповідаючи на їхні відгуки, вирішуючи проблеми та демонструючи свою відданість вдосконаленню програми. Активне спілкування з користувачами може допомогти створити позитивну репутацію та підвищити задоволеність користувачів.

Розгортання застосунку в AppStore відбувається наступним чином:

– Реєстрація в програмі розробників Apple: необхідно зареєструватись в програмі розробників Apple,. Це вимагає щорічної плати в 99 доларів США та 99 центів

– Підготовка застосунку: необхідно створіти ідентифікатор програми, та згенерувати профіль надання розповсюдження в App Store. Також необхідно налаштувати необхідні метадані програми, такі як назва програми, опис, знімки екрана та піктограми програми.

– Надсилання програми: необхідно надіслати застосунок на перевірку через App Store Connect,. Також необхідно надайти всю необхідну інформацію, зокрема ціни, категорії та будь-які конкретні вказівки щодо вмісту чи функцій застосунку.

– Перевірка програми: компанія Apple перевірить застосунок на відповідність їхнім інструкціям

– Випуск: після схвалення програми необхідно встановити дату випуску або вручну випустити її в App Store. Після цього користувачі зможуть завантажити та встановити застосунок.

4.2 Підтримка програмного забезпечення

Підтримка застосунку відбуватиметься через AppStore Connect. Процес розгортання оновлень схожий з процесом першого розгортання застосунку в AppStore, але не потребує реєстрації в програмі розробників та повторного процесу підготовки застосунку.

Висновки до розділу

В даному розділі було описано інструменти для розгортання застосунку, а також сам процес розгортання застосунків для пристроїв, що працюють на операційній системі iOS.

ВИСНОВКИ

В даному дипломному проєкті було проведено аналіз предметної області. Також було розроблено функціональні та нефункціональні вимоги, проведено наведено варіанти використання застосунку та проведено аналіз переваг та недоліків аналогів, що дало змогу приступити до етапу моделювання та розробки програмного забезпечення.

На етапі програмного моделювання та аналізу бізнес-процеси були описані за допомогою моделей BPMN. Ці моделі забезпечили чітке розуміння послідовності дій і взаємодії користувача в застосунку. Також під час моделювання було вирішено використовувати архітектуру MVVM. Архітектурний паттерні MVVM, дозволив чітко розділити відповідальність між компонентами моделі, представлення та контролера. Цей поділ полегшив організацію коду, зручність обслуговування та масштабованість програми.

На етапі аналізу та тестування якості програмного забезпечення було підкреслено важливість оцінки якості програмного забезпечення та опис проведення процесів оцінки якості коду та мануального тестування. Процеси тестування та контрольний приклад були описані для забезпечення функціональності, надійності та зручності використання застосунку користувачем.

Також було розглянуто процес впровадження та підтримки програмного забезпечення, зосереджуючись на розгортанні програмного забезпечення та постійній підтримці для забезпечення безперебійної роботи програми.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Health Problems Caused by Secondhand Smoke [Електронний ресурс] // Centres for Disease Control and Prevention. – 2022. – Режим доступу до ресурсу: https://www.cdc.gov/tobacco/secondhand-smoke/health.html.

2) Smoke-free environments [Електронний ресурс] // European Comission. – 2021. – Режим доступу до ресурсу: https://health.ec.europa.eu/tobacco/smoke-free-environments_en

3) What is Client Server Architecture? [Електронний ресурс] // Intellipaat. – 2019. – Режим доступу до ресурсу: https://intellipaat.com/blog/what-is-client-server-architecture/?US

4) UIKit documentation [Електронний ресурс] // Apple – Режим доступу до ресурсу: https://developer.apple.com/documentation/uikit

5) SwiftUI Documentation [Електронний ресурс] // Apple – Режим доступу до ресурсу: https://developer.apple.com/documentation/SwiftUI.

6) Combine Documentation [Електронний ресурс] // Apple – Режим доступу до ресурсу: https://developer.apple.com/documentation/Combine.

7) Firebase Docs [Електронний ресурс] // Google – Режим доступу до ресурсу: https://firebase.google.com/docs.

ДОДАТОК А ЗВІТ ПОДІБНОСТІ

UNICHECK

Ім'я користувача:
Лісовиченко Олег Іванович

ID перевірки:
1015406140

Дата перевірки:
03.06.2023 14:26:48 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
03.06.2023 14:30:36 EEST

ID користувача:
76913

Назва документа: IT-92_Василенко_ПЗ

Кількість сторінок: 66 **Кількість слів:** 6302 **Кількість символів:** 53113 **Розмір файлу:** 5.65 MB **ID файлу:** 1015069748

170 слів позначені як "вилучені" та не враховуються у підрахунку слів

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

13.3% Схожість

Найбільша схожість: 5.22% з джерелом з Бібліотеки (ID файлу: 1015062492)

3% Джерела з Інтернету	76	Сторінка 68
13.1% Джерела з Бібліотеки	233	Сторінка 69

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0.02% Вилучень

Деякі джерела вилучено автоматично (фільтри вилучення: кількість знайдених слів є меншою за 8 слів та 0%)

0% Вилучення з Інтернету	3	Сторінка 70
0.02% Вилученого тексту з Бібліотеки	2	Сторінка 70

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування	11 сторінок

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

Мобільний застосунок для боротьби з нікотиновою залежністю

Текст програми

КПІ.ІТ-9205.045490.03.12

“ПОГОДЖЕНО”
Керівник проєкту:
Микола ХРАМЧЕНКО

Нормоконтроль:
Ірина ВІТКОВСЬКА

Виконавець:
Олександр ВАСИЛЕНКО

Київ – 2023

Файл MoodClassifierCoordinator.swift

```
import UIKit
import SwiftUI

protocol MoodClassifierCoordinatorProtocol: Coordinator {
    func showMoodClassifierChoiceViewController()
    func showCoreMLClassifierViewController()
}

final class MoodClassifierCoordinator: MoodClassifierCoordinatorProtocol, CoordinatorFinishDelegate {
    func coordinatorDidFinish(childCoordinator: Coordinator) {
        switch childCoordinator {
        default:
            self.finish()
        }
    }

    func showMoodClassifierChoiceViewController() {
        let viewModel = MoodClassifierSelectionViewModel()
        let view = MoodClassifierSelectionView(viewModel: viewModel)
        let vc = UIHostingController(rootView: view)
        viewModel.didSendEventClosure = { [weak self] event in
            switch event {
            case .automatic:
                self?.showCoreMLClassifierViewController()
            case .manual:
                self?.showManualClassifierViewController()
            }
        }
        navigationController.pushViewController(vc, animated: true)
    }

    func showCoreMLClassifierViewController() {
        let coordinator = CoreMLClassifierCoordinator(navigationController)
        coordinator.start()
        childCoordinators.append(coordinator)
    }

    func showManualClassifierViewController() {
        let coordinator = ManualMoodCoordinator(navigationController)
        coordinator.start()
```

```
        coordinator.finishDelegate = self
        childCoordinators.append(coordinator)
    }

    weak var finishDelegate: CoordinatorFinishDelegate?

    var navigationController: UINavigationController

    var childCoordinators: [Coordinator] = []

    var type: CoordinatorType {.moodClassifier}

    func start() {
        showMoodClassifierChoiceViewController()
    }

    init(_ navigationController: UINavigationController) {
        self.navigationController = navigationController
    }

    deinit {
        print("\(self) deinited")
    }
}
```

Файл ManualMoodCoordinator.swift

```
protocol ManualMoodCoordinatorProtocol: Coordinator {
    func showManualSelectionView()
}

final class ManualMoodCoordinator: ManualMoodCoordinatorProtocol {
    var didSendEventClosure: ((CoordinatorType) -> Void)?
    func showManualSelectionView() {
        let viewModel = ManualMoodClassifierModuleViewModel()
        let view = MoodClassifierMainScreenView(viewModel: viewModel)
        let vc = UIHostingController(rootView: view)
        viewModel.didSendEndEventClosure = { [weak self] event in
            self?.finish()
        }
        navigationController.pushViewController(vc, animated: true)
    }
```

```
    weak var finishDelegate: CoordinatorFinishDelegate?

    var navigationController: UINavigationController

    var childCoordinators: [Coordinator] = []

    var type: CoordinatorType {.moodClassifier}

    func start() {
      showManualSelectionView()
    }

    init(_ navigationController: UINavigationController) {
      self.navigationController = navigationController
    }

    deinit {
      print("\(self) deinited")
    }

}
```

Файл CoreMLClassifierCoordinator.swift

```
protocol CoreMLClassifierCoordinatorProtocol: Coordinator {
  func showCoreMLClassifier()
}

class CoreMLClassifierCoordinator: CoreMLClassifierCoordinatorProtocol {
  func showCoreMLClassifier() {
    let vc = MoodClassifierViewController.module
    navigationController.pushViewController(vc, animated: true)
  }

  var finishDelegate: CoordinatorFinishDelegate?

  var navigationController: UINavigationController

  var childCoordinators: [Coordinator] = []
```

```
    var type: CoordinatorType {.moodClassifier}

    func start() {
        showCoreMLClassifier()
    }

    required init(_ navigationController: UINavigationController) {
        self.navigationController = navigationController
    }
```

Файл HeaderViewViewModel.swift

```
class HeaderViewViewModel: ObservableObject {
    @Published var image: Data?
    @Published var name: String = ""
    @Published var email: String = ""

    init(user: User) {
        updateWith(user: user)
    }

    func updateWith(user: User) {
        self.image = user.profileImage
        self.name = user.name
        self.email = user.email
    }

}
```

Файл SettingsViewController.swift

```
import UIKit
import SwiftUI
import Combine

class SettingsViewController: UIViewController {
    private let settingsLabels: [String] = ["Change password", "Terms and conditions", "About app"]
    private let gradientLayer = CAGradientLayer()
```

```
    private let viewModel = SettingsViewModel()
    private var disposeBag = Set<AnyCancellable>()
    private let settingsTableView: UITableView = {
        let tv = UITableView()
        tv.translatesAutoresizingMaskIntoConstraints = false
        tv.register(SettingsTableViewCell.self, forCellReuseIdentifier: SettingsTableViewCell.identifier)
        return tv
    }()

    private lazy var rootStackView: UIStackView = {
        let stackView = UIStackView(arrangedSubviews: [settingsTableView])
        stackView.axis = .vertical
        stackView.spacing = Spacings.spacing15
        stackView.translatesAutoresizingMaskIntoConstraints = false
        return stackView
    }()

    override func viewDidLoad() {
        super.viewDidLoad()
        view.addSubview(rootStackView)
        view.backgroundColor = .systemBackground
        settingsTableView.delegate = self
        settingsTableView.dataSource = self
//        setupHeaderProfileView()
        setupEditButton()
        setupConstraints()
        setupHeaderProfileView(viewModel: viewModel.headerViewModel)
    }
}

private extension SettingsViewController {
    func setupHeaderProfileView(viewModel: HeaderViewViewModel) {

        print("")
        let profileHeaderView = UIHostingController(rootView: SettingsViewProfileHeaderView(viewModel:
viewModel))
        addChild(profileHeaderView)
```

```
        profileHeaderView.didMove(toParent: self)
        profileHeaderView.view.translatesAutoresizingMaskIntoConstraints = false
        rootStackView.insertArrangedSubview(profileHeaderView.view, at: 0)
        NSLayoutConstraint.activate([
            profileHeaderView.view.heightAnchor.constraint(equalToConstant: view.frame.height / 3)
        ])
    }

    func setupEditButton() {
        let button = UIBarButtonItem(title: "Edit", style: .plain, target: self, action:
#selector(editButtonDidTap))
        self.navigationItem.rightBarButtonItems = [button]
    }

    func setupConstraints() {
        NSLayoutConstraint.activate([
            rootStackView.topAnchor.constraint(equalTo: view.topAnchor),
            rootStackView.leadingAnchor.constraint(equalTo: view.leadingAnchor, constant:
Spacings.spacing20),
            rootStackView.bottomAnchor.constraint(equalTo: view.safeAreaLayoutGuide.bottomAnchor),
            rootStackView.trailingAnchor.constraint(equalTo: view.trailingAnchor, constant: -
Spacings.spacing20),

        ])
    }


    func didSelectPasswordReset() {
        let alert = UIAlertController(title: "Do you want to reset your password?", message: "Note: ehis action
cannot be undone ", preferredStyle: .alert)
        let action = UIAlertAction(title: "Sure", style: .destructive)
        let cancelAction = UIAlertAction(title: "Cancel", style: .cancel)

        alert.addAction(action)
        alert.addAction(cancelAction)

        present(alert, animated: true)
```

```
    }

    @objc func editButtonDidTap() {

    }
}

extension SettingsViewController: UITableViewDelegate, UITableViewDataSource {
    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return settingsLabels.count
    }

    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
        let cell = tableView.dequeueReusableCell(withIdentifier: SettingsTableViewCell.identifier, for:
indexPath) as? SettingsTableViewCell
        guard let cell = cell else { return UITableViewCell() }
        cell.setText(text: settingsLabels[indexPath.row])
        return cell
    }

    func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
        switch indexPath.row {
        case 0:
            didSelectPasswordReset()
        case 1:

        case 2:

        default:

        }
    }

    func tableView(_ tableView: UITableView, viewForFooterInSection section: Int) -> UIView? {
        return SettingsTableViewFooterView(frame: CGRect(x: 0, y: 0, width: tableView.frame.width, height:
40))
    }
```

```
    func tableView(_ tableView: UITableView, heightForFooterInSection section: Int) -> CGFloat {
        return 40
    }
}

}
```

Файл SettingsTableViewCell.swift

```
class SettingsTableViewCell: UITableViewCell {

    static let identifier = "SettingsTableViewCell"
    private let settingsTextLabel: UILabel = {
        let label = UILabel()
        label.font = UIFont(name: FontsEnum.poppinsLight.rawValue, size: 14)
        return label
    }()

    override init(style: UITableViewCell.CellStyle, reuseIdentifier: String?) {
        super.init(style: style, reuseIdentifier: reuseIdentifier)
        contentView.addSubview(settingsTextLabel)
        self.backgroundColor = .clear
        self.accessoryType = .disclosureIndicator
        settingsTextLabel.frame = contentView.frame
    }

    func setText(text: String) {
        settingsTextLabel.text = text
    }

    required init?(coder: NSCoder) {
        fatalError("init(coder:) has not been implemented")
    }
}

}
```

Файл SettingsViewProfileHeaderView.swift

```
struct SettingsViewProfileHeaderView: View {
    @StateObject var viewModel: HeaderViewViewModel
    var body: some View {
        VStack(spacing: Spacings.spacing10) {
            createImage()
                .resizable()
                .clipShape(Circle())
                .frame(width: 100, height: 100)
            TextView(text: viewModel.name, font: .poppinsMedium, size: 16)
            TextView(text: viewModel.email, font: .poppinsMedium, size: 14)
                .foregroundColor(.gray)
        }
        .padding()
    }

    private func createImage() -> Image {
        let data = viewModel.image
        guard let data = data else { return Image("3")}
        let UIKitImage = UIImage(data: data)
        return Image(uiImage: UIKitImage!)
    }
}

}
```

Файл ManualMoodClassifierModuleViewModel.swift

```
class ManualMoodClassifierModuleViewModel: ObservableObject {
    @Published var moodsArray: [ClassifiedMood] = []
    @Published var selectedMood: ClassifiedMood? = nil
    var didSendEndEventClosure: ((ManualMoodClassifierModuleViewModel.EndEvent) -> Void)?

    init() {
        let moods = ClassifiedMood.allCases
        moodsArray = moods
```

```
    }

    func handleMoodSelection(selectedMood: ClassifiedMood?) {
        self.selectedMood = selectedMood
    }

    func didTapOnDoneButton() {
        guard let selectedMood = selectedMood else { return }
        FirebaseStorageService().uploadNewUserMood(mood: selectedMood)
        self.didSendEndEventClosure?(.moodClassifier)
    }
}

extension ManualMoodClassifierModuleViewModel {
    enum EndEvent { case moodClassifier }
}

}
```

Файл MoodClassifierSelectionViewModel.swift

```
class MoodClassifierSelectionViewModel: ObservableObject {
    var didSendEventClosure: ((MoodClassifierSelectionViewModel.EventType) -> Void)?

    func didChooseClassifierType(selectedMethod: EventType) {
        self.didSendEventClosure?(selectedMethod)
    }

    enum EventType {
        case automatic
        case manual
    }
}

}
```

Файл ClassifiedMood.swift

```
enum ClassifiedMood: String, Codable, CaseIterable {
    case angry = "Angry"
    case disgust = "Disgust"
    case fear = "Fear"
    case happy = "Happy"
    case neutral = "Neutral"
    case sad = "Sad"
    case surprise = "Surprised"

    var classifiedMood: ClassifiedMood? {
        return ClassifiedMood(rawValue: self.rawValue)
    }
}
```

} **Файл ClassificationResultModel.swift**

```
struct ClassificationResultModel {
    let moodString: String
    let moodConfidence: Double

    var classifiedMood: ClassifiedMood? {
        return ClassifiedMood(rawValue: moodString)
    }

    func toDict() -> [String: Any] {
        [
            "MoodString": moodString,
            "MoodConfidence": moodConfidence
        ]
    }
}
struct MoodModel: Codable {
//    let dateOfClassification = Date.now
    let id: String
    let classification: ClassifiedMood
    var totalScore: Int {
        switch classification {
```

```
        case .angry:
            return 1
        case .disgust:
            return 2
        case .fear:
            return 3
        case .happy:
            return 4
        case .neutral:
            return 5
        case .sad:
            return 6
        case .surprise:
            return 7
        }
    }
}

enum ClassificationResults {
    case succes, failure
}

protocol UserMoodClassifierServiceProtocol: AnyObject {
    func classifyImage(image: Data)
    func checkIfFacePresent(image: Data, completion: @escaping(ClassificationResults) -> Void)
    func cropImage(image: Data) -> UIImage?
}

}
```

Файл UserMoodClassifierService.swift

```
final class UserMoodClassifierService: UserMoodClassifierServiceProtocol {
    var classificationPublisher = PassthroughSubject<String, Never>()
    private lazy var classificationRequest: VNCoreMLRequest = {
        do {
            let model = try VNCoreMLModel(for: CNNEmotions().model)
```

```
        let request = VNCoreMLRequest(model: model) { request, _ in
            if let classifications = request.results as? [VNClassificationObservation] {
                print("Classification results: \(classifications)")
                let topClassification = classifications.first.map {
                    (confidence: $0.confidence, identifier: $0.identifier)
                }
                print("Top classifications: \(topClassification)")
                if let topClassification {
                    self.classificationPublisher.send(topClassification.identifier)
                }
            }
        }
        request.imageCropAndScaleOption = .centerCrop
        return request
    } catch {
        fatalError("Not ")
    }
}()

func classifyImage(image: Data) {
    guard let imageFromData = UIImage(data: image) else { return }
    guard let orientation = CGImagePropertyOrientation(
        rawValue: UInt32(imageFromData.imageOrientation.rawValue)) else {
        return
    }
    let croppedImage = cropImage(image: image)
    guard let ciImage = CIImage(image: croppedImage!) else {
        fatalError("Unable to create \(CIImage.self) from \(image).")
    }
    DispatchQueue.global(qos: .userInitiated).async {
        let handler =
        VNImageRequestHandler(ciImage: ciImage, orientation: orientation)
        do {
            let classificationRequest = self.classificationRequest
            try handler.perform([classificationRequest])
        } catch {
            print("Failed to perform classification.\n\(error.localizedDescription)")
```

```
        }
      }
    }

    func classifierResultsHandler(_ request: VNRequest, error: Error?) {


    }

    func checkIfFacePresent(image: Data, completion: @escaping(ClassificationResults) -> Void) {
      let image = UIImage(data: image)!
      let imageCG = image.cgImage!
      let request = VNDetectFaceRectanglesRequest { (req, err) in
        if let err = err {
          completion(.failure)
          return
        }
        req.results?.forEach({ (res) in
          guard let faceObservation = res as? VNFaceObservation else {return}
          completion(.succes)
        })
      }
#if targetEnvironment(simulator)
      request.usesCPUOnly = true
#endif
      let handler = VNImageRequestHandler(cgImage: imageCG, options: [:])
      do {
        try handler.perform([request])
      } catch let reqErr {
        print("Error", reqErr)
      }
    }

    func cropImage(image: Data) -> UIImage? {
      let image = UIImage(data: image)
      guard let imageUnwrapped = image else { return nil }
      let cgImage = imageUnwrapped.cgImage
      guard let cgImageUnwrapped = cgImage else { return nil }
```

```
        // Resize the cropped image to 224x224 pixels
        let renderer = UIGraphicsImageRenderer(size: CGSize(width: 224, height: 224))
        let resizedImage = renderer.image { context in
            UIImage(cgImage: cgImageUnwrapped).draw(in: CGRect(origin: .zero, size: CGSize(width: 224,
height: 224)))
        }

        return resizedImage
    }


} struct MoodClassifierSelectionView: View {
    @ObservedObject var viewModel: MoodClassifierSelectionViewModel
    var body: some View {
        VStack {
            Group {
                TextView(text: "Please specify your mood", font: .poppinsBold, size: 24)
                TextView(text: "I want to do this...", font: .poppinsMedium, size: 18)
            }.multilineTextAlignment(.leading)
            Group {
                Button {
                    viewModel.didChooseClassifierType(selectedMethod: .automatic)
                } label: {
                    TextView(text: "Automaticly", font: .poppinsSemiBold, size: 14)
                }
                Button {
                    viewModel.didChooseClassifierType(selectedMethod: .manual)
                } label: {
                    TextView(text: "By myself", font: .poppinsSemiBold, size: 14)
                }
            }.buttonStyle(StandartButtonStyle())
        }.padding(Spacings.spacing25)
    }
}

struct MoodClassifierSelectionView_Previews: PreviewProvider {
    static var previews: some View {
```

```
        @StateObject var vm = MoodClassifierSelectionViewModel()
        MoodClassifierSelectionView(viewModel: vm)
    }
}


struct MoodClassifierButton: View {
    @State var buttonPressed: Bool = false
    @Binding var labelText: ClassifiedMood
    @Binding var selectedMood: ClassifiedMood?

    var body: some View {
        Button {
            buttonPressed.toggle()
            selectedMood = buttonPressed ? labelText : nil
        } label: {
            Group {
                HStack {
                    TextView(text: labelText.rawValue, font: .poppinsSemiBold, size: 14)
                    Image(systemName: buttonPressed ? "checkmark" : "")
                }
            }
        }
        .buttonStyle(ButtonWithCheckMarkButtonStyle())
    }
}
struct MoodSelectionView: View {
    @Binding var moods: [ClassifiedMood]
    @Binding var selectedMood: ClassifiedMood?

    var body: some View {
        ScrollView {
            VStack {
                ForEach($moods, id: \.self) { $moodLabel in
                    MoodClassifierButton(labelText: $moodLabel, selectedMood: $selectedMood)
                }
            }
```

```
        }
    }
}

}
```

Файл ReasonsToStopModule.swift

```
final class ReasonsToStopModulePresenter: ReasonsToStopPresenterProtocol {
    let array = ["Nicotine addiction", "Stress", "Social situations", "Habits and routines", "Weight gain", "Boredom", "Lack of support", "Alcohol consumption", "Advertising", "Low mood", "Peer pressure", "Mental health conditions", "Lack of information", "Feeling overwhelmed", "Lack of alternatives", "The belief that smoking is enjoyable", "Access to cigarettes", "Lack of commitment", "Lack of self-efficacy", "Fear of failure"]
    weak var view: ReasonsToStopViewProtocol?

    init(view: ReasonsToStopViewProtocol) {
        self.view = view

    }

    func showArrayOfReasons() {
        view?.showReasons(reasons: array.sorted(by: <))
    }

    func didTapDoneButton(reasons: [String]) {
        // save reasons and so
    }
}
class ReasonsToStopCollectionViewCell: UICollectionViewCell {
    static let identifier = "ReasonsToStopCollectionViewCell"


    private let label: UILabel = {
        let label = UILabel()
        label.setPoppinsFont(size: 18, style: .bold)
        label.textColor = .black
        label.translatesAutoresizingMaskIntoConstraints = false
```

```
        return label
    }()

    override init(frame: CGRect) {
        super.init(frame: frame)
        setupContentView()
        setupConstraints()
    }

    required init?(coder: NSCoder) {
        fatalError("init(coder:) has not been implemented")
    }

    func setText(text: String) {
        label.text = text
    }

    func handleCellSelectedState() {
        contentView.backgroundColor = .systemGreen
    }

    func handleCellDeselectedState() {
        contentView.backgroundColor = .systemGray
    }

//    override func layoutSubviews() {
//        super.layoutSubviews()
//
//        layer.shadowPath = UIBezierPath(
//            roundedRect: bounds,
//            cornerRadius: 20
//        ).cgPath
//    }

}

private extension ReasonsToStopCollectionViewCell {
```

```
    func setupConstraints() {
        NSLayoutConstraint.activate([
            label.topAnchor.constraint(equalTo: contentView.topAnchor, constant: 8),
            label.leadingAnchor.constraint(equalTo: contentView.leadingAnchor, constant: 8),
            label.trailingAnchor.constraint(equalTo: contentView.trailingAnchor, constant: -8),
            label.bottomAnchor.constraint(equalTo: contentView.bottomAnchor, constant: -8)
        ])
    }

    func setupContentView() {
        contentView.addSubview(label)
        contentView.makeGlassEffectOnView(style: .extraLight)
        contentView.layer.masksToBounds = true
        contentView.layer.cornerRadius = 7
//        contentView.layer.borderWidth = 3
//        contentView.layer.borderColor = UIColor.white.cgColor
    }


}
final class ReasonsToStopViewController: UIViewController {
    private var reasonsToStopCollectionView: UICollectionView!
    private var gradientLayer = CAGradientLayer()
    var presenter: ReasonsToStopPresenterProtocol?
    private var doneButton: UIButton = {
        let button = UIButton()
        button.setTitle("Done", for: .normal)
        button.titleLabel?.font = UIFont(name: "Poppins-SemiBold", size: 18)
        button.translatesAutoresizingMaskIntoConstraints = false
        button.backgroundColor = UIColor(named: ColorConstants.purpleColor)
        button.isEnabled = false
        button.layer.cornerRadius = 10
        return button
    }()

    private lazy var rootStackView: UIStackView = {
        let stackView = UIStackView(arrangedSubviews: [reasonsToStopCollectionView, doneButton])
```

```
        stackView.translatesAutoresizingMaskIntoConstraints = false
        stackView.axis = .vertical
        return stackView
    }()

    private var selectedItems = Set<Int>()
    private var reasonsToStopArray: [String] = []



    override func viewDidLoad() {
        super.viewDidLoad()
        view.setBackgroundColor()
        setupCollectionView()
        view.addSubview(rootStackView)
        setupConstraints()
    }
}

private extension ReasonsToStopViewController {

//    func setupNav

    func setupLayout() -> UICollectionViewLayout {
        let itemSize = NSCollectionLayoutSize(widthDimension: .estimated(100), heightDimension:
.fractionalHeight(0.8))
        let item = NSCollectionLayoutItem(layoutSize: itemSize)

        let groupSize = NSCollectionLayoutSize(widthDimension: .fractionalWidth(1), heightDimension:
.absolute(50))
        let group = NSCollectionLayoutGroup.horizontal(layoutSize: groupSize, subitems: [item])
        group.interItemSpacing = NSCollectionLayoutSpacing.fixed(10)

        let section = NSCollectionLayoutSection(group: group)
        section.contentInsets = NSDirectionalEdgeInsets(top: LayoutConstants.spacing16, leading: 0,
bottom: 0, trailing: 0)
        let layout = UICollectionViewCompositionalLayout(section: section)
```

```
        return layout
    }


    func setupCollectionView() {
        reasonsToStopCollectionView = UICollectionView(frame: .zero, collectionViewLayout:
setupLayout())
        reasonsToStopCollectionView.translatesAutoresizingMaskIntoConstraints = false
        reasonsToStopCollectionView.allowsMultipleSelection = true
        reasonsToStopCollectionView.register(ReasonsToStopCollectionViewCell.self,
forCellWithReuseIdentifier: ReasonsToStopCollectionViewCell.identifier)
        reasonsToStopCollectionView.dataSource = self
        reasonsToStopCollectionView.backgroundColor = .clear
        reasonsToStopCollectionView.allowsMultipleSelection = true
        reasonsToStopCollectionView.delegate = self

        presenter?.showArrayOfReasons()
//        view.addSubview(reasonsToStopCollectionView)
    }

    func setupConstraints() {
        NSLayoutConstraint.activate([
            rootStackView.topAnchor.constraint(equalTo: view.safeAreaLayoutGuide.topAnchor),
            rootStackView.leadingAnchor.constraint(equalTo: view.safeAreaLayoutGuide.leadingAnchor,
constant: LayoutConstants.spacing16),
            rootStackView.trailingAnchor.constraint(equalTo: view.safeAreaLayoutGuide.trailingAnchor,
constant: -LayoutConstants.spacing16),
            rootStackView.bottomAnchor.constraint(equalTo: view.safeAreaLayoutGuide.bottomAnchor,
constant: -LayoutConstants.spacing16)
        ])
    }
}

extension ReasonsToStopViewController: ReasonsToStopViewProtocol {
    func showReasons(reasons: [String]) {
        self.reasonsToStopArray = reasons
        reasonsToStopCollectionView.reloadData()
```

```
    }
}

extension ReasonsToStopViewController: UICollectionViewDataSource, UICollectionViewDelegate {
    func collectionView(_ collectionView: UICollectionView, cellForItemAt indexPath: IndexPath) ->
UICollectionViewCell {
        let cell = collectionView.dequeueReusableCell(withReuseIdentifier:
ReasonsToStopCollectionViewCell.identifier, for: indexPath) as? ReasonsToStopCollectionViewCell
        guard let cell = cell else { return UICollectionViewCell() }
        cell.setText(text: reasonsToStopArray[indexPath.row])
        return cell
    }

    func collectionView(_ collectionView: UICollectionView, numberOfItemsInSection section: Int) -> Int {
        return reasonsToStopArray.count
    }

    func collectionView(_ collectionView: UICollectionView, shouldSelectItemAt indexPath: IndexPath) ->
Bool {
        if selectedItems.count >= 10 {
            return false
        }
        return true
    }

    func collectionView(_ collectionView: UICollectionView, didSelectItemAt indexPath: IndexPath) {
        selectedItems.insert(indexPath.item)
        let cell = collectionView.cellForItem(at: indexPath) as? ReasonsToStopCollectionViewCell
        guard let cell = cell else { return }
        cell.handleCellSelectedState()
    }

    func collectionView(_ collectionView: UICollectionView, didDeselectItemAt indexPath: IndexPath) {
        selectedItems.remove(indexPath.item)
        let cell = collectionView.cellForItem(at: indexPath) as? ReasonsToStopCollectionViewCell
        guard let cell = cell else { return }
        cell.handleCellDeselectedState()
```

```
    }
}
extension ReasonsToStopViewController {
    static var module: ReasonsToStopViewController {
        let view = ReasonsToStopViewController()
        let presenter = ReasonsToStopModulePresenter(view: view)
        view.presenter = presenter
        return view
    }
}
```

Файл MoodChartModule.swift

```
struct MoodChartView: View {
    @Binding var dataForCharts: [ChartModel]

    var body: some View {
        VStack {
            Chart(dataForCharts) { day in
                LineMark(x: .value("Day", day.dateOfClassification), y: .value("Score", day.scoreForUser))
                    .foregroundStyle(Color(ColorConstants.labelColor))
                    .interpolationMethod(.cardinal)
                PointMark(x: .value("Day", day.dateOfClassification), y: .value("Score", day.scoreForUser))
                    .foregroundStyle(Color(ColorConstants.labelColor))
                    .interpolationMethod(.cardinal)
//                AreaMark(x: .value("Day", day.dateOfClassification), y: .value("Score", day.scoreForUser))
//                    .foregroundStyle(Gradient(colors: [Color(ColorConstants.labelColor), Color(uiColor:
.systemBackground)]))
//                    .interpolationMethod(.cardinal)
            }
//            .chartXScale(domain: [0, dataForCharts.endIndex])
            .animation(.linear, value: dataForCharts.count)
            .scaleEffect(CGSize(width: 1, height: calculateScale()))
        }
    }
```

```
    private func calculateScale() -> CGFloat {
        guard let firstDate = dataForCharts.first?.dateOfClassification,
              let lastDate = dataForCharts.last?.dateOfClassification else {
            return 1.0
        }

        let daysBetween = Calendar.current.dateComponents([.day], from: firstDate, to: lastDate).day ?? 0

        // Set a minimum scale of 1.0 to avoid the chart becoming too small
        return max(1.0, CGFloat(daysBetween) / 100.0)
    }
}

struct MoodChartView_Previews: PreviewProvider {
    static var previews: some View {
        @State var mockData = [ChartModel(id: "mock", dateOfClassification: Date.now, classification:
.happy), ChartModel(id: "mock", dateOfClassification: Date.now + 8, classification: .sad), ChartModel(id:
"mock", dateOfClassification: Date.now + 9, classification: .disgust), ChartModel(id: "mock",
dateOfClassification: Date.now + 25, classification: .happy), ChartModel(id: "mock", dateOfClassification:
Date.now + 26, classification: .angry), ChartModel(id: "mock", dateOfClassification: Date.now + 31,
classification: .surprise), ChartModel(id: "mock", dateOfClassification: Date.now + 124, classification:
.fear), ChartModel(id: "mock", dateOfClassification: Date.now + 512, classification: .neutral)]
        MoodChartView(dataForCharts: $mockData)
    }
}

struct FilterMenuView: View {
    @Binding var selectedFilteringMethod: ProgressChartsPeriods
    var body: some View {
        HStack {
            VStack(alignment: .leading) {
                TextView(text: TextStringConstants.activity, font: .poppinsMedium, size: 20)
                TextView(text: TextStringConstants.checkYourActivityThroughPeriod, font: .poppinsRegular,
size: 12)
            }

            .fixedSize()
```

```
            Spacer()
            RoundedRectangle(cornerRadius: LayoutConstants.cornerRadius)
                .stroke()
                .overlay {
                    Menu {
                        Button("2 Weeks", action: {
                            selectedFilteringMethod = .twoWeeks
                        })
                        Button("1 Month", action: {
                            selectedFilteringMethod = .oneMonth
                        })
                        Button("3 Months") {
                            selectedFilteringMethod = .threeMonth
                        }
                    } label: {
                        Text(selectedFilteringMethod.rawValue)
                    }
                }
        }
        .foregroundColor(Color(ColorConstants.labelColor))
        .padding(.leading)
    }
}
//
struct FilterMenuView_Previews: PreviewProvider {
    static var previews: some View {
        FilterMenuView(selectedFilteringMethod: .constant(.oneMonth))
    }
}
struct ProgressChartsModuleMainScreen: View {
    @StateObject var viewModel = ProgressChartsViewModel()

    var body: some View {

        GeometryReader { frame in
            VStack(spacing: Spacings.spacing20) {
                MoodChartView(dataForCharts: $viewModel.weekDataForCharts)
```

```
                .frame(height: frame.size.height * 2/10)
            Group {
                ProgressStatsViewContainer(bestDay: $viewModel.bestDay, worstDay:
$viewModel.worstDay, bestScore: $viewModel.bestDayScore, worstScore: $viewModel.worstDayScore)
                    .frame(height: frame.size.height * 3/10)
                FilterMenuView(selectedFilteringMethod: $viewModel.selectedSotringMethod)
                    .frame(height: frame.size.height * 1/20)
            }.padding(.horizontal, Spacings.spacing20)
            MoodChartView(dataForCharts: $viewModel.dataForCharts)
                .padding(.top, Spacings.spacing10)
            //                    .frame(height: frame.size.height * 4/10)
        }
      }
   }
}
//
//struct ProgressChartsModuleMainScreen_Previews: PreviewProvider {
//    static var previews: some View {
//        ProgressChartsModuleMainScreen()
//    }
//}

struct ProgressStatsViewContainer: View {
    @Binding var bestDay: String
    @Binding var worstDay: String
    @Binding var bestScore: String
    @Binding var worstScore: String
    var body: some View {
        HStack (alignment: .top) {
            VStack(alignment: .leading, spacing: Spacings.spacing15) {
                ProgressStatsView(titleText: "Best day", secondaryText: bestDay)
                ProgressStatsView(titleText: "Best score", secondaryText: bestScore)
//                ProgressStatsView(titleText: "Wiii", secondaryText: "Character #1")
            }
            Spacer()
            VStack(alignment: .leading, spacing: Spacings.spacing15) {
                ProgressStatsView(titleText: "Worst day", secondaryText: worstDay)
```

```
                ProgressStatsView(titleText: "Worst score", secondaryText: worstScore)
            }
        }
        .padding(Spacings.spacing20)

    }
}

struct ProgressStatsViewContainer_Previews: PreviewProvider {
    static var previews: some View {

        ProgressStatsViewContainer(bestDay: .constant("54"), worstDay: .constant("R3e"), bestScore:
.constant("333"), worstScore: .constant("e332423"))
    }
}
struct ProgressStatsView: View {
    let titleText: String
    let secondaryText: String
    var body: some View {
        VStack(alignment: .leading) {
            TextView(text: titleText, font: .poppinsSemiBold, size: 18)
                .foregroundColor(Color(ColorConstants.purpleColor))
            TextView(text: secondaryText, font: .poppinsLight, size: 14)
                .lineLimit(2)
        }
    }
}

struct ProgressStatsView_Previews: PreviewProvider {
    static var previews: some View {
        ProgressStatsView(titleText: "", secondaryText: "Says")
    }
}
enum ProgressChartsPeriods: String {
    case oneWeek = "1 Week"
    case twoWeeks = "2 Weeks"
    case oneMonth = "1 Month"
```

```
    case threeMonth = "3 Month"
    case sixMonth = "6 Month"

}

final class ProgressChartsViewModel: ObservableObject {
    private var cancellables = Set<AnyCancellable>()

    private var chartModelData: [ChartModel] = [] {
        didSet {
            self.dataForCharts = chartModelData
            filterChartsData(for: selectedSotringMethod)
            filterChartsData(for: .oneWeek)
            print("Latest day: \(chartModelData.last?.dateOfClassification)")
        }
    }

    @Published var selectedSotringMethod: ProgressChartsPeriods = .twoWeeks {
        didSet {
            filterChartsData(for: selectedSotringMethod)
        }
    }



    @Published var dataForCharts: [ChartModel] = [] {
        didSet {
            getStatistics()
        }
    }

    @Published var weekDataForCharts: [ChartModel] = []

    @Published var bestDay: String = ""
    @Published var worstDay: String = ""
    @Published var bestDayScore = ""
    @Published var worstDayScore = ""
    @Published var savedOnSigs = ""
```

```
    init() {
        getChartsData()
    }
}

extension ProgressChartsViewModel {
    private func getChartsData() {
        FirebaseStorageService().getChartsData()
            .sink { finish in
                print(finish)
            } receiveValue: { [weak self] data in
                self?.chartModelData = data
                    .sorted(by: {
                        $0.dateOfClassification < $1.dateOfClassification
                    })
            }.store(in: &cancellables)
    }


    private func filterChartsData(for period: ProgressChartsPeriods) {
        let currentDate = Date.now
        let calendar = Calendar.current
        let oneWeekAgo = calendar.date(byAdding: .weekday, value: -7, to: currentDate)!
        let twoWeeksAgo = calendar.date(byAdding: .weekday, value: -14,to: currentDate)!
        let oneMonthAgo = calendar.date(byAdding: .month, value: -1, to: currentDate)!
        let threeMonthAgo = calendar.date(byAdding: .month, value: -3, to: currentDate)!
        let sixMonthAgo = calendar.date(byAdding: .month, value: -6, to: currentDate)!
        switch period {
        case .oneMonth:
            dataForCharts = chartModelData.filter {
                $0.dateOfClassification > oneMonthAgo && $0.dateOfClassification < currentDate
            }
        case .threeMonth:
            dataForCharts = chartModelData.filter {
                $0.dateOfClassification > threeMonthAgo && $0.dateOfClassification < currentDate
            }
```

```
        case .sixMonth:
            dataForCharts = chartModelData.filter {
                $0.dateOfClassification > sixMonthAgo && $0.dateOfClassification < currentDate
            }
        case .oneWeek:
            weekDataForCharts = chartModelData.filter({
                $0.dateOfClassification > oneWeekAgo && $0.dateOfClassification < currentDate
            })

        case .twoWeeks:
            dataForCharts = chartModelData.filter({
                $0.dateOfClassification > twoWeeksAgo && $0.dateOfClassification < currentDate
            })
        }
    }

    private func getStatistics() {
        let maxScoreDay = dataForCharts.max(by: {
            $0.scoreForUser < $1.scoreForUser
        })
        let minScoreDay = dataForCharts.min {
            $0.scoreForUser < $1.scoreForUser
        }
        guard let maxScoreDay = maxScoreDay, let minScoreDay = minScoreDay else { bestDay = "No data"; worstDay = "No data"; return }

        bestDay = maxScoreDay.dateOfClassification.toDateComponents(neededComponents: [.year, .month, .day]).toSting()
        worstDay = minScoreDay.dateOfClassification.toDateComponents(neededComponents: [.year, .month, .day]).toSting()

        bestDayScore = "\(maxScoreDay.scoreForUser)"
        worstDayScore = "\(minScoreDay.scoreForUser)"
    }
}

struct ChartModel: Codable, Identifiable {
```

```
    let id: String
    var dateOfClassification = Date()
    let classification: ClassifiedMood
    var classificationString: String {
        self.classification.rawValue
    }
//    var animate = false

    var scoreForUser: Int {
        switch classification {
        case .angry:
            return 11
        case .disgust:
            return 12
        case .fear:
            return 10
        case .happy:
            return 20
        case .neutral:
            return 18
        case .sad:
            return 16
        case .surprise:
            return 22
        }
    }
} class ProgressChartsModuleHostingViewController: UIViewController {

    private let mainScreenView = UIHostingController(rootView: ProgressChartsModuleMainScreen())
    private let qutittingInfoLabel: UILabel = {
        let label = UILabel()
        label.text = "Chart for this week"
        label.font = UIFont(name: "Poppins-SemiBold", size: 18)
        label.textColor = UIColor(named: ColorConstants.labelColor)
        label.textAlignment = .left
        return label
    }()
```

```
    override func viewDidLoad() {
        super.viewDidLoad()
//        navigationController?.navigationBar.addSubview(qutittingInfoLabel)
        navigationItem.titleView = qutittingInfoLabel
        view.addSubview(mainScreenView.view)
        addChild(mainScreenView)
        mainScreenView.view.translatesAutoresizingMaskIntoConstraints = false
        mainScreenView.didMove(toParent: self)
        setupConstraints()
    }

    private func setupConstraints() {
        NSLayoutConstraint.activate([
            mainScreenView.view.topAnchor.constraint(equalTo: view.topAnchor),
            mainScreenView.view.leadingAnchor.constraint(equalTo: view.leadingAnchor),
            mainScreenView.view.bottomAnchor.constraint(equalTo:
view.safeAreaLayoutGuide.bottomAnchor),
            mainScreenView.view.trailingAnchor.constraint(equalTo: view.trailingAnchor)

        ])
    }

}
```

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

Мобільний застосунок для боротьби з нікотиновою залежністю

Програма та методика тестування

КПІ.ІТ-9205.045490.04.51

“ПОГОДЖЕНО”
Керівник проєкту:
Микола ХРАМЧЕНКО

Нормоконтроль:
Ірина ВІТКОВСЬКА

Виконавець:
Олександр ВАСИЛЕНКО

Київ – 2023

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробування є Мобільний застосунок для боротьби з нікотиновою залежністю, створений для ОС iOS із використанням технологій Swift, SwiftUI та UIKit.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

– перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;

– перевірка збереження даних;

– перевірка сумісності веб-додатку з останніми версіями операційних систем iOS

– знаходження проблем, помилок і недоліків з метою їх усунення;

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

– статичне тестування – перевіряється програма разом з усією документацією, яка аналізується на предмет дотримання стандартів програмування;

– динамічне тестування – застосовується в процесі виконання програми. Коректність програмного засобу перевіряється на певній кількості тестів. При прогоні кожного з них збираються та аналізуються дані про проблеми та помилки в роботі програми;

– функціональне тестування – полягає у перевірці відповідності реальної поведінки програмного забезпечення очікуваній;

– системне тестування – перевіряється усе програмне забезпечення в цілому;

– мануальне тестування – тестування без використання автоматизації, тест-кейси пише особа, що тестує програмне забезпечення;

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується мануально з метою знаходження помилок та недоліків як у функціональній частині програмного забезпечення так і в зручності користування. Для того, щоб перевірити працездатність та відмовостійкість застосунку, необхідно провести наступні тестування:

– динамічне тестування на відповідність функціональним вимогам;

– тестування на мобільних пристроях з різною роздільною здатністю екрану;

– тестування на мобільних пристроях з різною версією операційної системи;

– тестування на виведення повідомлень про помилку, коли це необхідно;

– тестування інтерфейсу користувача;

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

Мобільний застосунок для боротьби з нікотиновою залежністю

Керівництво користувача

КПІ.ІТ-9205.045490.05.34

“ПОГОДЖЕНО”
Керівник проєкту:
Микола ХРАМЧЕНКО

Нормоконтроль:
Ірина ВІТКОВСЬКА

Виконавець:
Олександр ВАСИЛЕНКО

Київ – 2023

ЗМІСТ

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

«QuitMate» - це мобільний застосунок для боротьби з нікотиновою залежністю. QuitMate використовує потужність штучного інтелекту, щоб надавати індивідуальні рекомендації, відстежувати прогрес і пропонувати статистику в реальному часі.

Завдяки інтуїтивно зрозумілому інтерфейсу та зручним функціям QuitMate створює спільноту людей, які прагнуть до вільного від нікотину життя. QuitMate служить віртуальним компаньйоном, надаючи користувачам змогу подолати тягу та виробити здоровіші звички.

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Для успішної роботи даного застосунку необхідне виконання наступних вимог:

– наявність мобільного пристрою з операційною системою iOS з версією 16.0 або вище;

– наявність доступу до Інтернету;

– для встановлення додатку на мобільному пристрої повинно бути не менше 70 МБ вільної пам’яті.

2.2 Завантаження застосунку

Завантаження застосунку виконується через, вбудований в операційну систему iOS, магазин застосунків AppStore.

2.3 Перевірка коректної роботи

По завершенню встановлення додатка на робочому столі мобільного пристрою повинна відобразитись іконка даного застосунку. У разі, якщо дана іконка не з’явилась, то встановлення відбулось не успішно. Інакше користувач має змогу запустити додаток, клацнувши на його іконку. Після натискання повинна відобразитись сторінка автентифікації в застосунку.

3 ВИКОНАННЯ ПРОГРАМИ

При запуску програмного застосунку користувачу буде відображено два поля для вводу пошти та паролю, а також кнопка яка переводить користувача на сторінку реєстрації (рисунок 3.1).

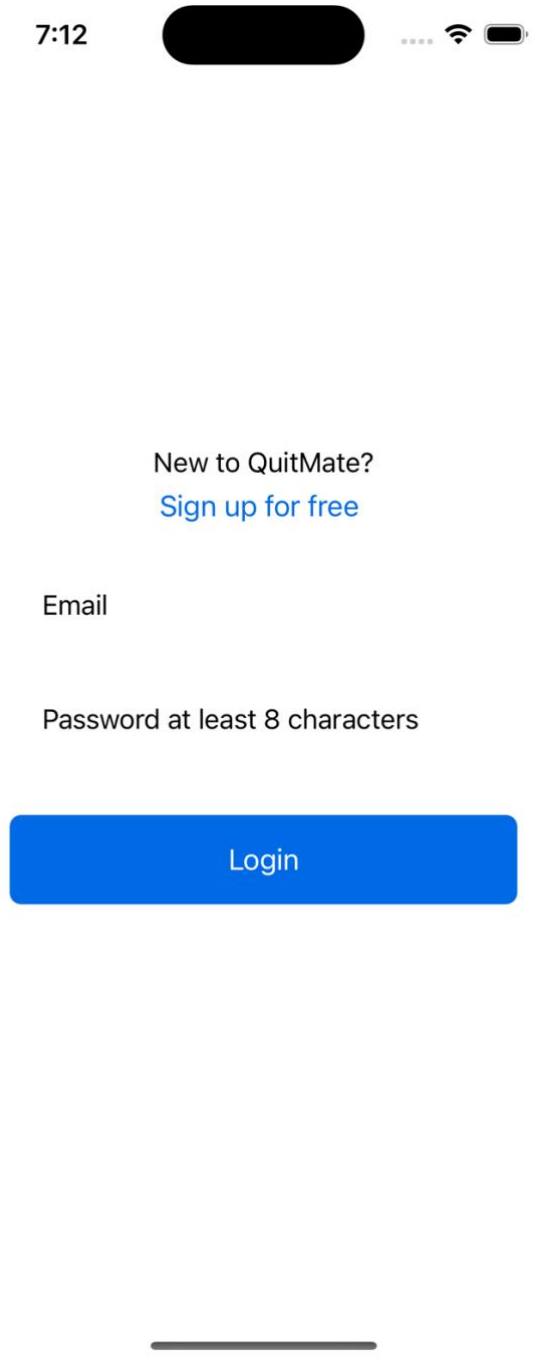


Рисунок 3.1 – Початкова сторінка додатку

Користувач має змогу ввести адресу email та пароль після чого натиснути кнопку входу в застосунок.

Перейдемо до екрану реєстрації. В ньому користувач має ввести наступні дані:

– Електронну пошту
– Пароль
– Підтвердження паролю

Якщо користувач ввів неправильну електронну пошту застосунок має зробити кнопку реєстрації неактивною та підсвітити поле вводу пошти червоним. На рисунку 3.2 зображено вигляд інтерфейсу застосунку в цьому випадку

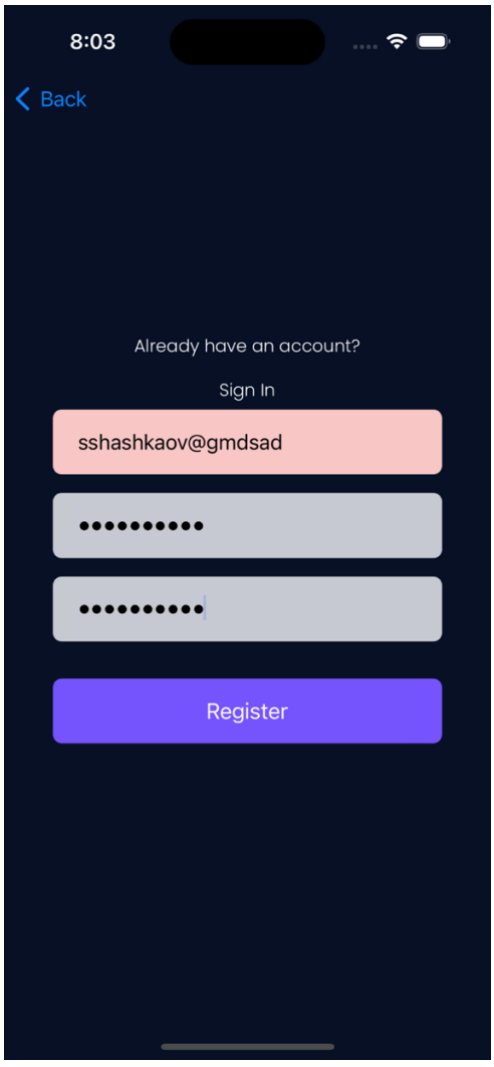


Рисунок 3.2 – Відображення повідомлення про введення неправильного паролю для авторизації

Перейдемо до головного екрану застосунку (рисунок 3.3)

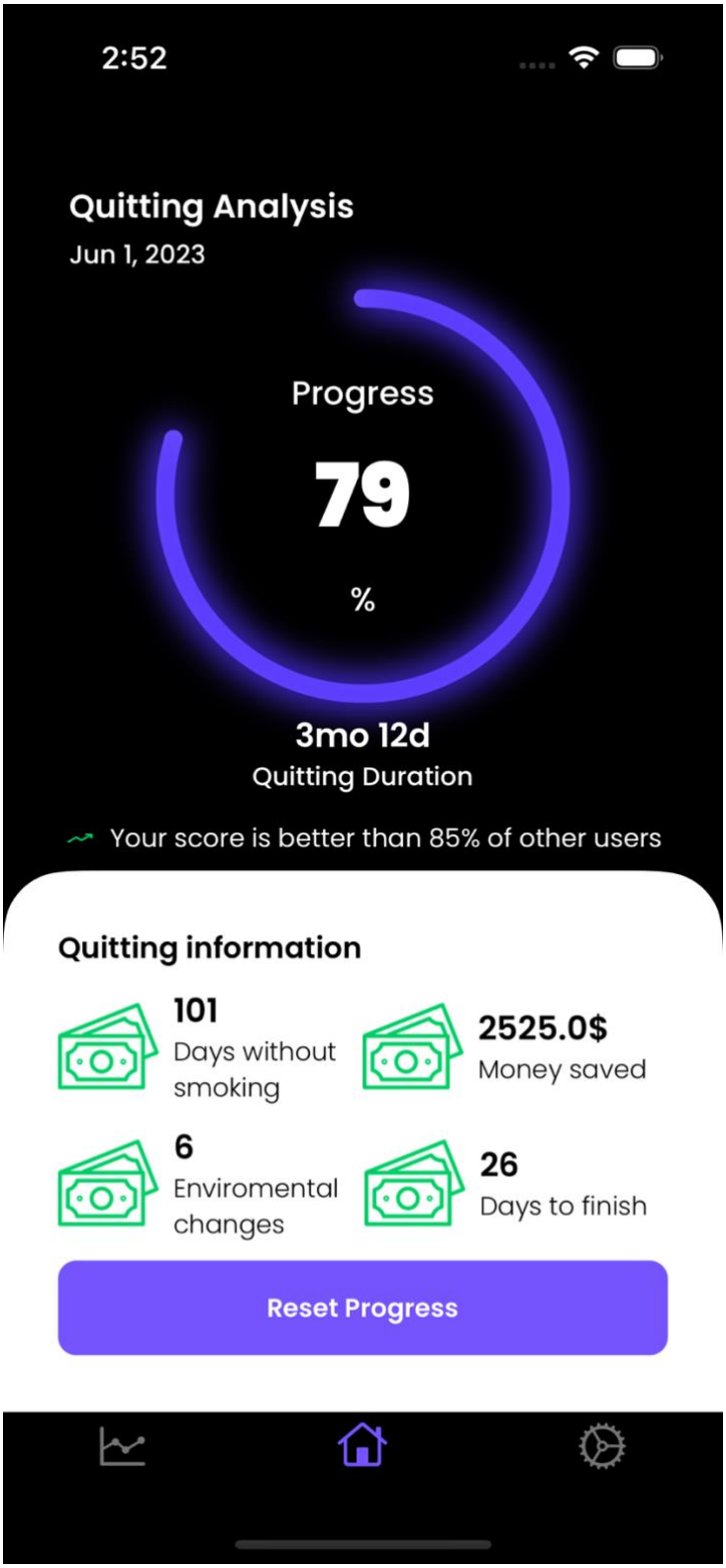


Рисунок 3.3 – Зовнішній вигляд головного екрану застосунку

На головному екрані відображаються статистичні дані користувача. На головному екрані є кнопка яка відповідає за скидання встановлених цілей та статистичних даних користувача. Коли користувач натискає на цю кнопку йому має висвітитись повідомлення з попередженням про те, що ця дія

незворотня та приведе до скидання встановлених цілей та статистичних даних користувача. На рисунку 3.4 зображено вигляд інтерфейсу застосунку в цьому випадку

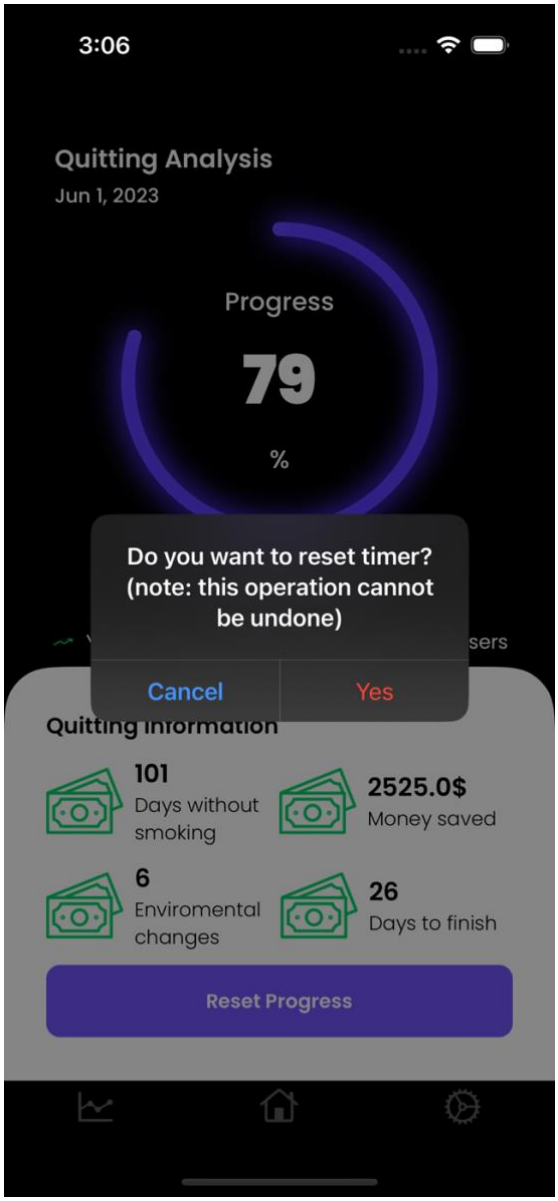


Рисунок 3.4 – Зовнішній вигляд головного екрану застосунку при натисканні на кнопку скидання прогресу

Якщо користувач підтверджує свій вибір, застосунок відображає користувачеві екран, де йому необхідно вибрати, із запропонованого списку, причини чому саме він бажає скинути встановлені цілі та статистичні дані. Зовнішній вигляд екрану наведено на рисунку 3.5

Рисунок 3.5 – Зовнішній вигляд екрану вказання причини скидання прогрегсу

Після того як користувач вказав причини чому він вирішив скинути прогрес застосунок надає йому рекомендації згенеровані ШІ на основі його самопочуття та причини чому він вирішив скинути прогрес (рисунок 3.6).

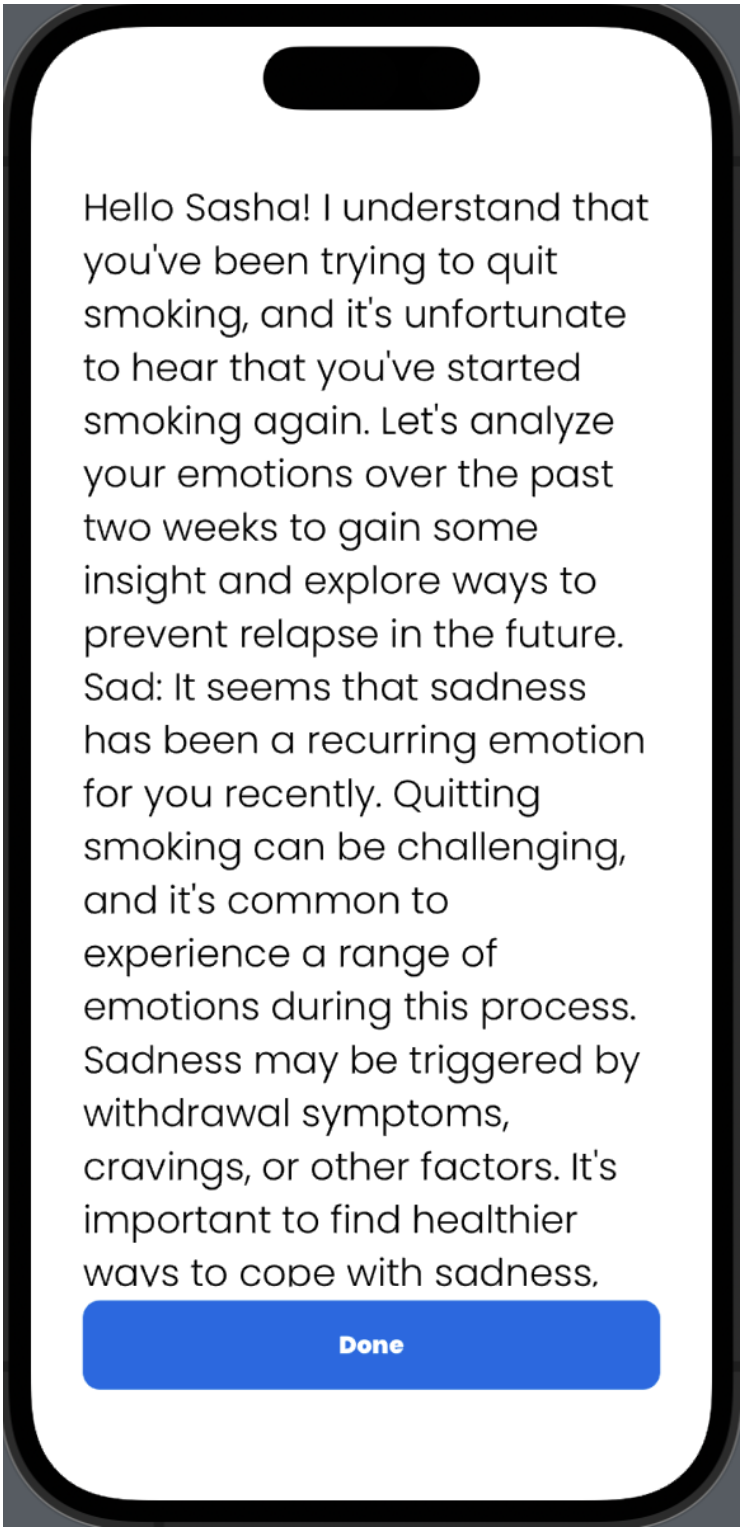


Рисунок 3.6 – Зовнішній вигляд екрану рекомендацій згенерованих ШІ

Також користувач має змогу переглянути зміну свого самопочуття у вигляді графіків. На рисунку 3.7 зображено зовнішній вигляд екрану перегляду даних щодо зміни самопочуття користувача

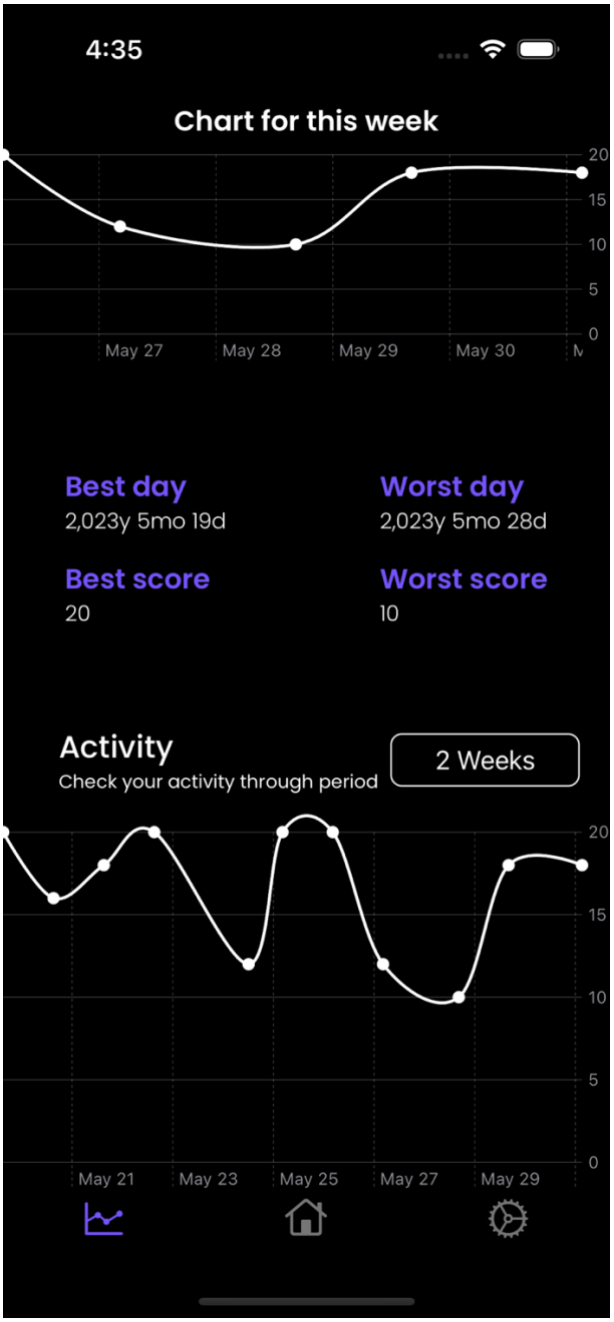


Рисунок 3.7 – Зовнішній вигляд екрану перегляду даних щодо зміни самопочуття користувача

Користувач також може фільтрувати дані за певний період, наприклад: 2 тижні, 1 місяць та 3 місяці. Зовнішній вигляд меню зміни періоду відображено на рисунку 3.8

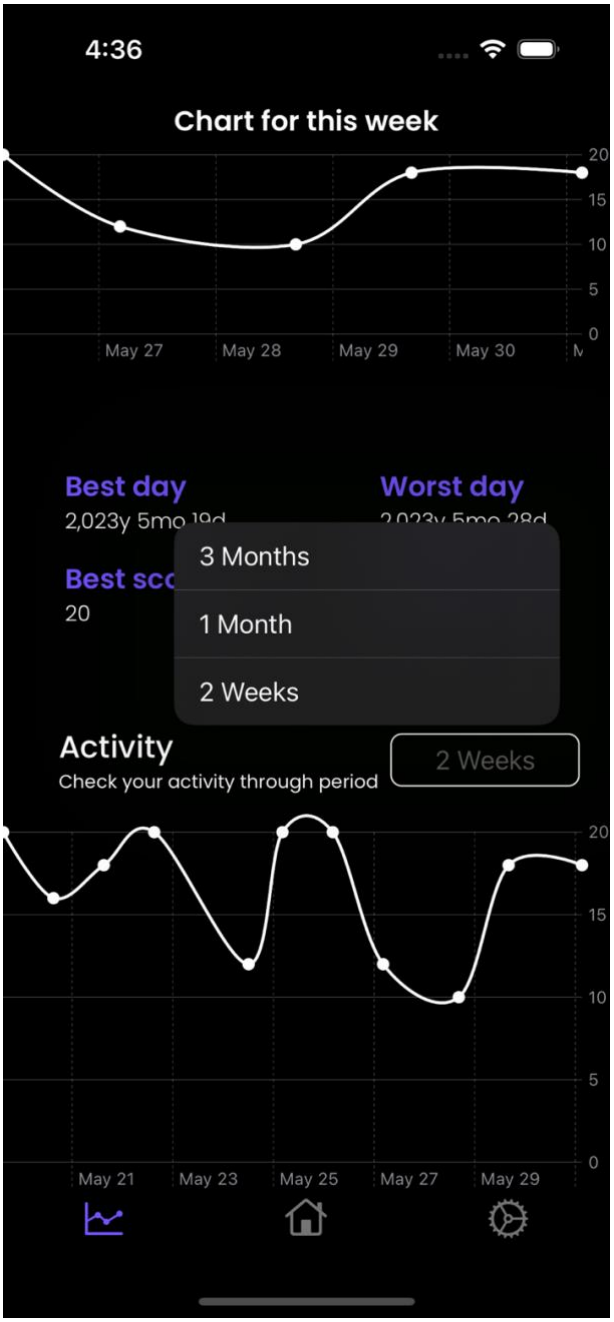


Рисунок 3.8 – Зовнішній вигляд меню зміни періоду

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

Мобільний застосунок для боротьби з нікотиновою залежністю

Графічний матеріал

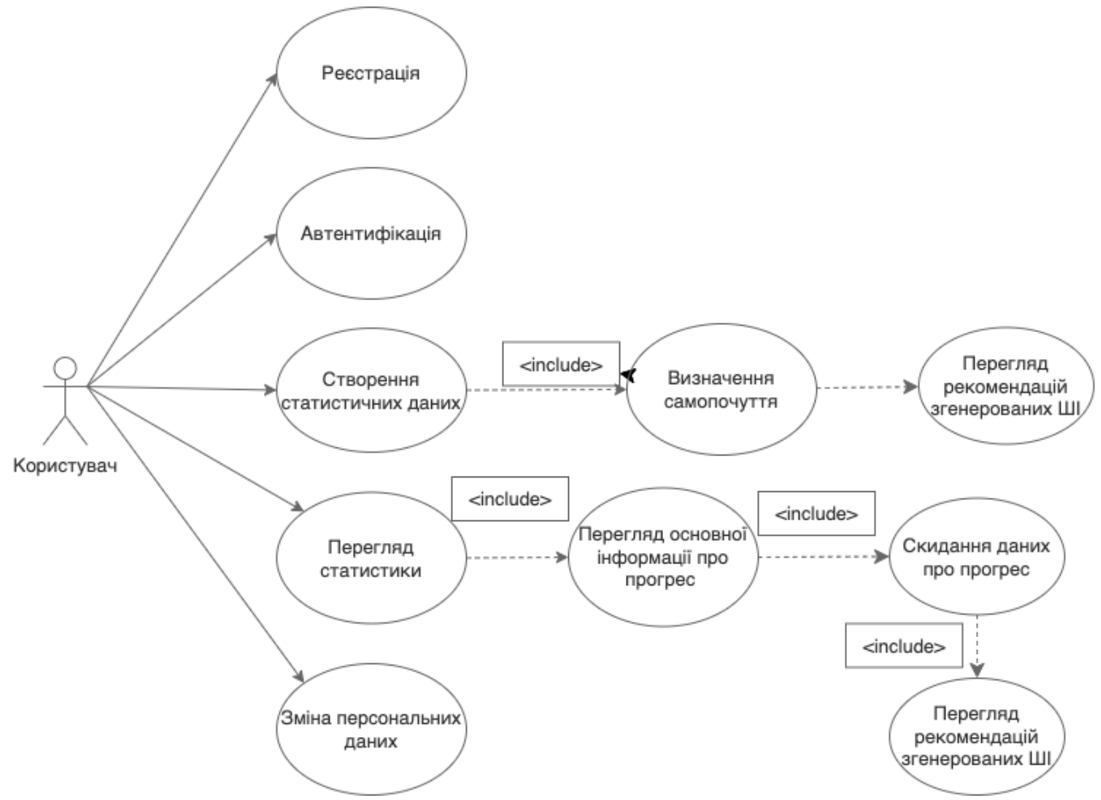
КПІ.ІТ-9205.045490.06.99

“ПОГОДЖЕНО”
Керівник проєкту:
Микола ХРАМЧЕНКО

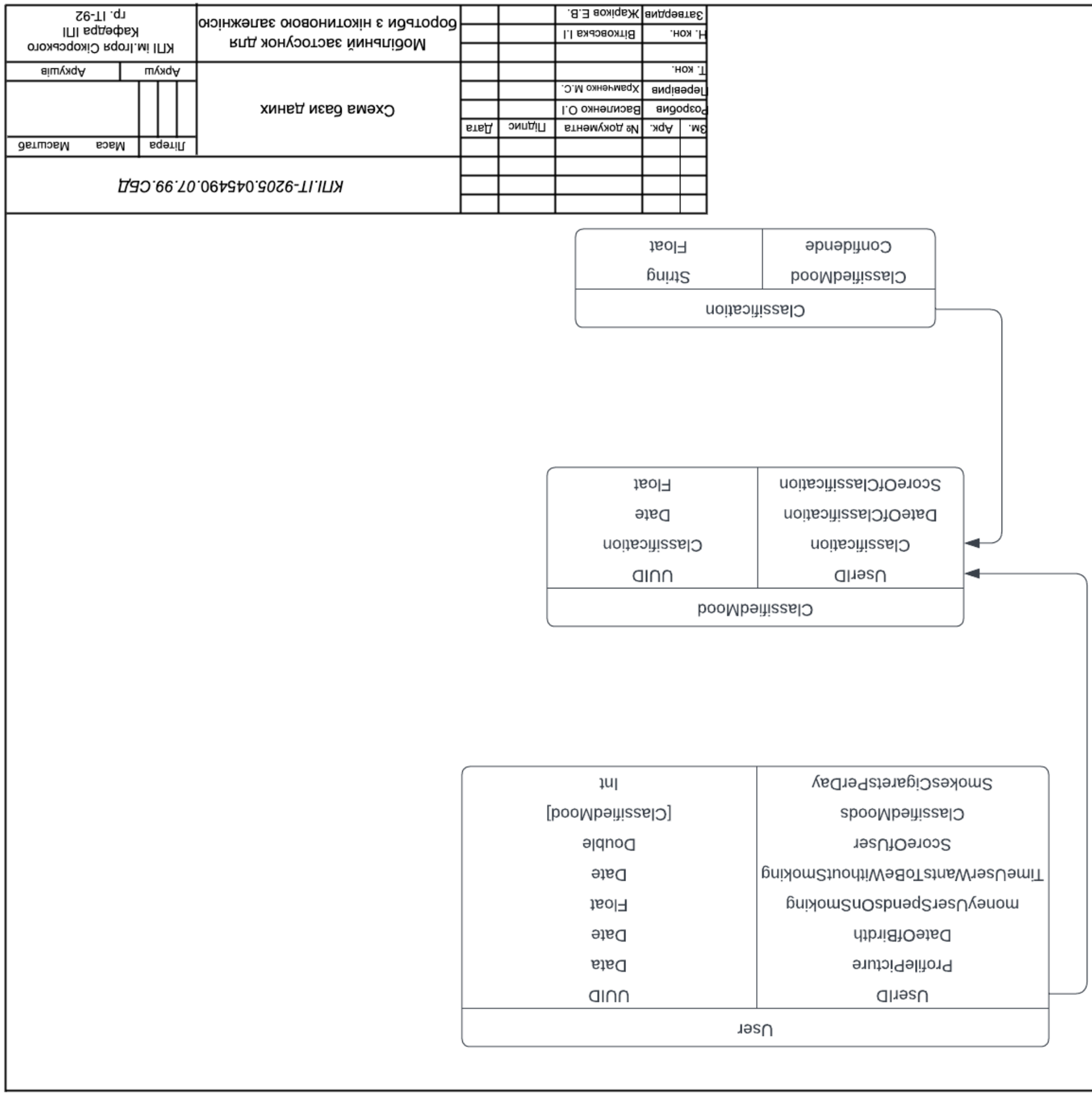
Нормоконтроль:
Ірина ВІТКОВСЬКА

Виконавець:
Олександр ВАСИЛЕНКО

Київ – 2023



					*КПІ.ІТ-9205.*045490.*06.99.ССВ*	
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна варіантів використань	Літера / Маса / Масштаб
Розробив		Василенко О.І				
Перевірив		Храмченко М.С.				Аркуш / Аркушів
Т. кон.						
Н. кон.		Вітковська І.І			Мобільний застосунок для боротьби з нікотиновою залежністю	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-92
Затвердив		Жаріков Е.В.				



Зм.	Арк.	№ документа	Підпис	Дата
Розробив		Васильєнко О.І		
Перевірив		Храмченко М.С.		
Т. контр.				
Н. контр.		Бітковська І.І		
Затвердив		Жарікова Е.В.		

КПІ.ІТ-9205.045490.07.99.СБД

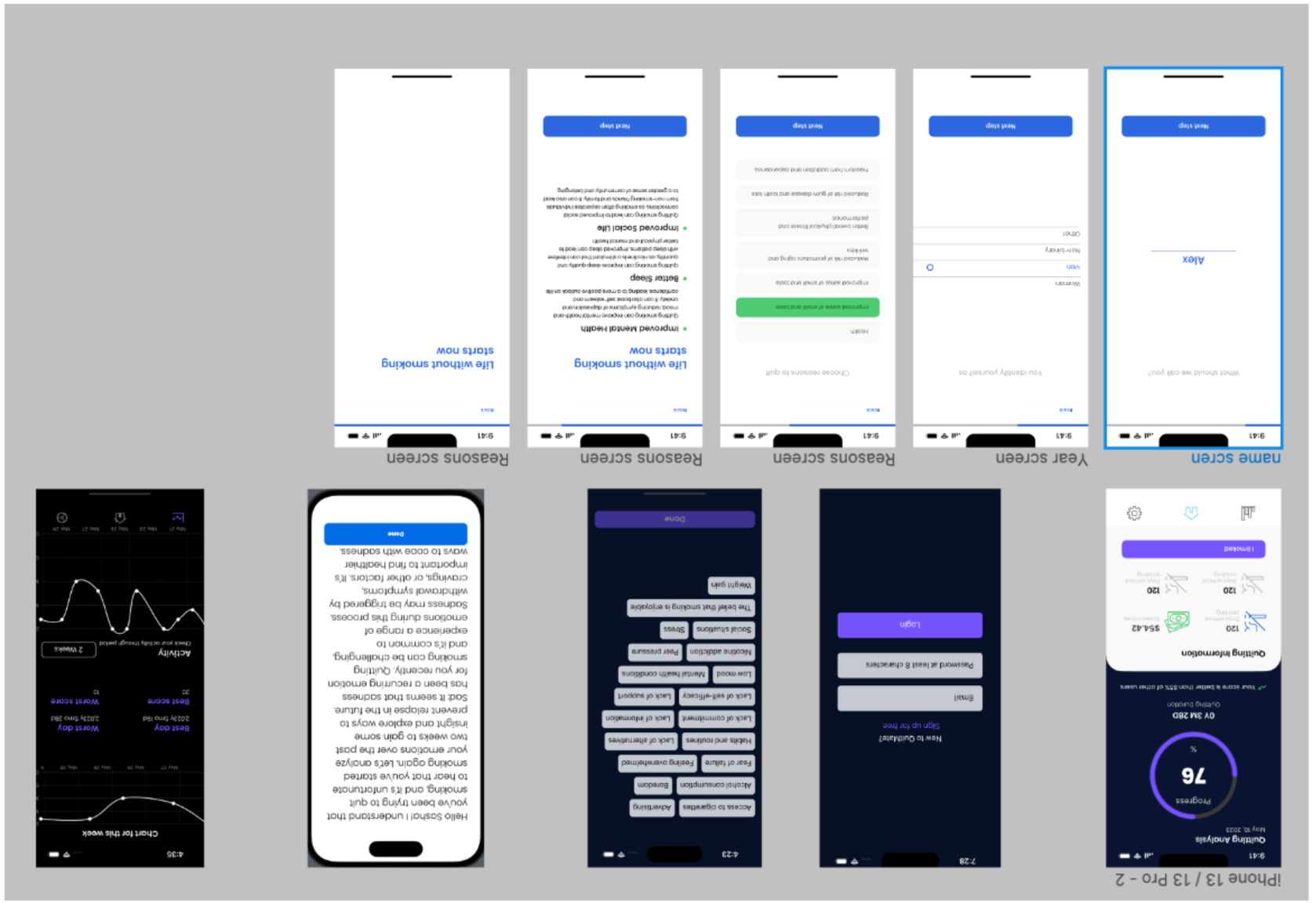
Схема бази даних

Літера	Маса	Масштаб

Аркуш	Аркушів

Мобільний застосунок для боротьби з нікотиновою залежністю

КПІ ім.Ігоря Сікорського
Кафедра ІПІ
гр. ІТ-92



КПІ.ІТ-9205.045490.08.99.КЕ

Зм.	Арк.	№ докумeнта	Підпис	Дата			
Розробив		Васильєнко О.І			Крєслення вигляду екранних форм		
Перевірив		Храмєнко М.С.					
Т. кон.							
Н. кон.		Бітковська І.І					
Затвердив		Жарікова Е.В.					

Літера	Маса	Масштаб
Аркуш		Аркушів

КПІ ім.Ігоря Сікорського
Кафедра ІПІ
гр. ІТ-92

Мобільний застосунок для боротьби з нікотиновою залежністю