Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: «Розробка методу та програмних засобів для комунікацій в системах телемедицини»

Виконала: студентка IV курсу
групи 4ПІ-20б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Борисова К. О.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ Войтко В. В.
(прізвище та ініціали)

Рецензент: к.т.н., доцент каф. ОТ Тарновський М. Г.
(прізвище та ініціали)

Допущено до захисту

Зав. кафедри ПЗ _____
«_10_» __червня__ 2024 р.

ВНТУ – 2024

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
О. Н. Романюк
«12» березня 2024 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Борисовій Катерині Олексіївні

1. Тема роботи – розробка методу та програмних засобів для комунікацій в системах телемедицини

Керівник роботи: Войтко Вікторія Володимирівна, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від 11 березня 2024 р. №80.

2. Строк подання студентом роботи: 3 червня 2024.

3. Вихідні дані до роботи: середовище розробки – Visual Studio Code, мови розробки – Dart, PHP; фреймворки – Flutter, Laravel; операційна система – Windows 8/10/11; емулятор мобільного пристрою – Pixel 6 з операційною системою Android 14.0.

4. Зміст текстової частини: вступ, аналіз стану розвитку програм для комунікацій в системах телемедицини та постановка задач розробки, розробка методу, моделі та алгоритмів роботи програми, розробка програмного забезпечення, тестування програми, висновки, список використаних джерел, додатки, ілюстративна частина.

5. Перелік ілюстративного матеріалу: титульний слайд – автор, науковий керівник бакалаврської дипломної роботи, тема; актуальність розробки; мета, об'єкт та предмет дослідження; постановка задачі; наукова новизна; практична цінність; огляд та аналіз аналогів; модель роботи системи, загальний алгоритм роботи системи; діаграми станів; схеми інтерфейсу; діаграма обчислення середнього рейтингу лікаря; використані технології; функціонал розробленої системи; апробація та публікація результатів роботи; висновки; фінальний слайд.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-4	Войтко В. В., к.т.н., доцент кафедри ПЗ	13.03.24	03.06.24

7. Дата видачі завдання 13 березня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану розвитку програм для комунікацій в системах телемедицини та постановка задач розробки	14.03.24 – 21.03.24	Виконано
2	Розробка методів, моделі та алгоритмів роботи програми	22.03.24 – 04.04.24	Виконано
3	Розробка програмного забезпечення	05.04.24 – 03.05.24	Виконано
4	Тестування програми	06.05.24 – 17.05.24	Виконано
5	Оформлення матеріалів до захисту БДР	20.05.24 – 30.05.24	Виконано

Студент _____ **Борисова**
(підпис) (ініціали та пр

Керівник бакалаврської дипломної роботи _____ **Войтко В.**
(підпис) (ініціали та пр

АНОТАЦІЯ

Бакалаврська дипломна робота містить 75 сторінок формату А4, на яких є 58 рисунків і 5 таблиць, а список використаних джерел складається з 22 найменування.

У бакалаврській дипломній роботі проведено аналіз стану і розвитку програмних застосунків, що допомагають в організації комунікацій у системах телемедицини. Було проведено аналіз аналогів, визначено їх переваги і недоліки та доведено доцільність розробки власного програмного застосунку.

У роботі обґрунтовано вибір мови програмування, середовища розробки та технологій, що використовувалися під час створення мобільного застосунку. Було розроблено програмний продукт – мобільний застосунок для платформи Android, реалізований за допомогою мови програмування Dart і фреймворку Flutter. Для розробки бекенд-частини затосунку було використано засоби мови програмування PHP та фреймворку Laravel. Як середовище розробки було використано Visual Studio Code.

Отримані результати бакалаврської дипломної роботи можуть бути застосовані для покращення комунікаційних можливостей у телемедичних системах.

Ключові слова: мобільний застосунок, медичні записи, Flutter, Dart, кросплатформенна розробка, інтерфейс користувача.

ABSTRACT

The bachelor's thesis consists of 75 A4 pages, 58 figures and 5 tables, and the list of references includes 22 titles.

The bachelor's thesis analyses the state and development of software applications that help organise communications in telemedicine systems. An analysis of analogues was carried out, their advantages and disadvantages were identified, and the feasibility of developing an in-house software application was proved.

The paper substantiates the choice of programming language, development environment and technologies used to create the mobile application. A software product was developed - a mobile application for the Android platform, implemented using the Dart programming language and the Flutter framework. To develop the backend part of the application, PHP programming language and Laravel framework were used. Visual Studio Code was used as a development environment.

The results of the bachelor's thesis can be used to improve communication capabilities in telemedicine systems.

Keywords: mobile application, medical records, Flutter, Dart, cross-platform development, user interface.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. У сучасному світі, де технології стрімко розвиваються, телемедицина стає невід'ємною частиною системи охорони здоров'я. Вона покращує доступ до медичної допомоги, дозволяючи людям, які живуть далеко від лікарень, або людям з обмеженими можливостями отримувати допомогу, не виходячи з дому. Крім того, телеконсультації та дистанційне спостереження за пацієнтом можуть бути значно дешевшими за відвідування лікаря. Телемедицина також покращує координацію догляду за пацієнтами та прискорює діагностику та лікування захворювань [1]. Це дозволяє виявляти захворювання на ранніх стадіях, коли лікування є більш ефективним, і забезпечує пацієнтам зручність отримання медичної допомоги вдома.

Впровадження нових технологій, таких як мобільні застосунки, хмарні обчислення та штучний інтелект, робить телемедицину ще доступнішою та зручнішою для пацієнтів. Мобільні застосунки спрощують доступ до медичних послуг, хмарні обчислення дозволяють безпечно зберігати та обмінюватися медичними даними, а штучний інтелект використовується для аналізу медичних даних і надання рекомендацій лікарям [2].

Мобільні пристрої є популярними засобами для взаємодії завдяки компактності, автономності та високій функціональності. Інтерфейс операційних систем мобільних пристроїв інтуїтивно зрозумілий і простий у використанні, що не вимагає від користувачів спеціальних навичок.

Ефективне спілкування між лікарями та пацієнтами є запорукою успішного використання телемедицини. У зв'язку з цим особливої актуальності набуває розробка методів і програмних засобів для оптимізації процесів комунікації в телемедицині шляхом створення мобільного застосунку. Удосконалення цих процесів має важливе значення для забезпечення надійності надання медичних

послуг, безпеки обміну медичною інформацією між пацієнтами та лікарями, створення умов для ефективного та безпечного надання медичної допомоги.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою роботи є покращення комунікаційних можливостей у системах телемедицини за рахунок розробки ефективної та зручної автоматизованої системи, яка дозволить пацієнтам та медичним фахівцям легко взаємодіяти між собою та сприятиме підвищенню якості медичного обслуговування.

Відповідно до поставленої мети потрібно вирішити такі завдання:

– провести аналіз інформаційного забезпечення системи;

– розробити метод, модель та алгоритм роботи системи;

– розробити систему для ведення комунікацій в телемедицині, що включає:

1) реєстрацію облікового запису для пацієнта;

2) можливість запису на прийом до лікаря з обранням потрібного спеціаліста та часу;

3) відстеження статусу записів (підтверджено, в очікуванні, відмінено тощо);

4) функціонал додавання медичних фахівців у список улюблених для швидкого доступу до них;

5) можливість переглядати історію записів до лікарів;

6) функціонал додавання медичних фахівців у список улюблених для швидкого доступу до них.

7) функціонал для консультації з лікарем онлайн: текстовий чат та онлайн-прийом з використанням веб-камери пристрою користувачів;

8) розробка дешборду для перегляду лікарем статистики про його роботу.

Об'єктом дослідження виступає процес розробки програмного забезпечення для систем комунікацій у телемедицині.

Предметом дослідження є алгоритми та методи, використовувані для розробки програмного продукту, принципи програмування мови Dart, використання фреймворків Flutter та Laravel та засоби середовища VS Code.

Методи дослідження. У процесі дослідження використовувалися наступні методи:

– методи побудови віджетів за стосунку на базі фреймворку Flutter для реалізації графічного інтерфейсу користувача;

– способи та технології розробки програм для мобільних пристроїв з операційною системою Android;

– методи тестування, що використовуються для перевірки розроблених модулів мобільної системи для ведення комунікацій в телемедицині;

– методи організації баз даних для збереження та обробки медичних даних у локальному сховищі.

Новизна отриманих результатів.

1. Подальшого розвитку отримав метод візуалізації медичних даних, який, на відміну від існуючих, включає інтеграцію з графічним інтерфейсом користувача, що дозволяє лікарям та пацієнтам з легкістю візуалізувати інформацію про лікарів, покращує розуміння та моніторинг стану здоров'я пацієнтів та сприяє ефективному лікуванню та профілактиці захворювань.

2. Подальшого розвитку отримала модель комплексної телемедичної системи, яка, на відміну від існуючих, інтегрує набір необхідних інструментів для ведення переговорів, планування та проведення онлайн-консультацій, а також використовує бази даних для забезпечення безперервного доступу до даних, надає можливість більш гнучкої та широкої організації роботи медичних установ, що

підвищує ефективність роботи медичних закладів і дозволяє надавати послуги телемедицини на високому рівні.

Практичне значення проєкту полягає у можливості використанні клініками, лікарями та пацієнтами розробленого програмного забезпечення для комунікацій в системах телемедицини, що забезпечить усім сторонам цієї взаємодії легкий та зручний доступ до віддалених медичних консультацій. Програмні модулі можуть бути використані розробниками для створення аналогічних систем у різних медичних контекстах, включно з обміном медичною інформацією, діагностикою та лікуванням на відстані. Таке програмне забезпечення може бути адаптоване для використання у приватних клініках, державних лікарнях та інших медичних установах, сприяючи підвищенню ефективності медичних послуг і оптимізації лікувальних процесів.

Особистий внесок здобувача. Усі результати, подані в бакалаврській дипломній роботі, були отримані автором особисто. У науковій праці [3], опублікованій у співавторстві, автору належать такі результати: таблиця порівняння функціональних характеристик аналогів, перелік розроблених функцій.

Апробація та публікація результатів проєкту. Результати бакалаврської дипломної роботи були представлені на конференції «Автоматизація та комп’ютерно-інтегровані технології у виробництві та освіті: стан, досягнення, перспективи розвитку» (2024) та опубліковані у вигляді тез доповідей на цій конференції.

Публікації. Результати дослідження були опубліковані у матеріалах конференції – «Автоматизація та комп’ютерно-інтегровані технології у виробництві та освіті: стан, досягнення, перспективи розвитку» (2024).

Аналіз. У пояснювальній записці до бакалаврської дипломної роботи було розглянуто чотири розділи та було використано 22 літературних джерела.

1 АНАЛІЗ СТАНУ РОЗВИТКУ ПРОГРАМ ДЛЯ КОМУНІКАЦІЙ В СИСТЕМАХ ТЕЛЕМЕДИЦИНИ ТА ПОСТАНОВКА ЗАДАЧ РОЗРОБКИ

1.1 Аналіз предметної області

Зі зростанням ролі цифрових технологій у сфері охорони здоров'я, системи телемедицини стають все більш актуальними і затребуваними. Розробка програмних засобів для комунікацій у системах телемедицини відіграє ключову роль у підвищенні ефективності медичного обслуговування, надаючи пацієнтам зручний доступ до віддалених медичних консультацій. Впровадження ефективних комунікаційних технологій є важливим завданням, що дозволяє медичним установам покращити якість послуг та оптимізувати використання ресурсів [1].

Однією з основних переваг впровадження сучасних комунікаційних систем у телемедицині є можливість забезпечити безперервний зв'язок між пацієнтами та лікарями, зменшуючи час очікування на консультації та підвищуючи загальну доступність медичних послуг. Проте, розробка таких систем вимагає глибокого аналізу потреб користувачів та медичних установ, а також створення надійних і масштабованих програмних рішень [2].

Багато медичних установ звертаються до IT-компаній, які спеціалізуються на розробці телемедичних інформаційних систем, для створення індивідуальних рішень, що задовольняють їхні унікальні потреби. Програмне забезпечення для комунікацій у телемедицині дозволяє інтегрувати різні функції, такі як відеозв'язок та управління медичними записами, спрощуючи процес управління пацієнтською інформацією та підвищуючи задоволеність користувачів [4].

Таким чином, розробка методу і програмних засобів для комунікацій у системах телемедицини є критично важливою для досягнення високих стандартів медичного обслуговування. Інвестиції у розробку високоякісних IT-рішень можуть значно підвищити ефективність роботи медичних установ, забезпечити

високий рівень обслуговування пацієнтів та забезпечити медичним установам конкурентні переваги.

1.2 Аналіз аналогів розроблюваного програмного застосунку

Зі швидкими темпами розвитку технологій, медична галузь переживає значні трансформації, особливо у сфері телемедицини. Подібні застосунки дозволяють медичним закладам більш ефективно розподіляти робочий час, уникати перевантажень і забезпечувати точний та швидкий доступ до медичних послуг для пацієнтів. У зв'язку з цим на ринку з'являється все більше мобільних застосунків, що допомагають ефективно управляти записами на прийоми.

Серед численних аналогів розроблюваної системи варто виділити декілька застосунків: EasyVisit, Docton та Health24, кожен з яких має свої унікальні особливості та переваги.

Наприклад, EasyVisit [5] (рис. 1.1), розроблений компанією Sonic Healthcare Ltd, представляє собою інноваційне рішення в Австралії, яке значно спрощує процедуру запису на медичні прийоми.

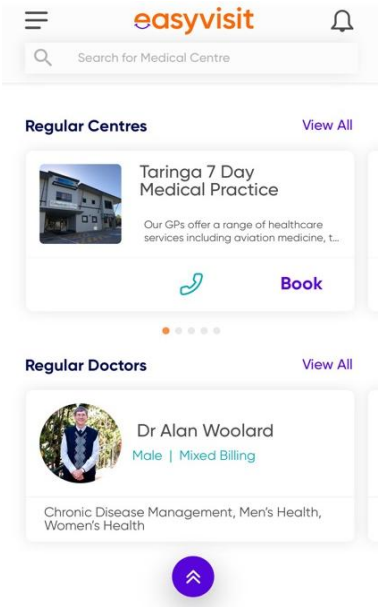


Рисунок 1.1 – Приклад роботи застосунку EasyVisit

Завдяки інтуїтивно зрозумілому інтерфейсу та розширеному асортименту функціональних можливостей, користувачі можуть ефективно вибирати потрібних медичних спеціалістів та встановлювати час прийому (рис. 1.2). Крім того, застосунок дозволяє управляти записами не тільки за себе, але й за інших осіб, що робить його особливо зручним для використання в сімейних умовах. Ці особливості роблять застосунок EasyVisit вельми популярним серед користувачів, які цінують зручність та швидкість при плануванні медичних візитів.

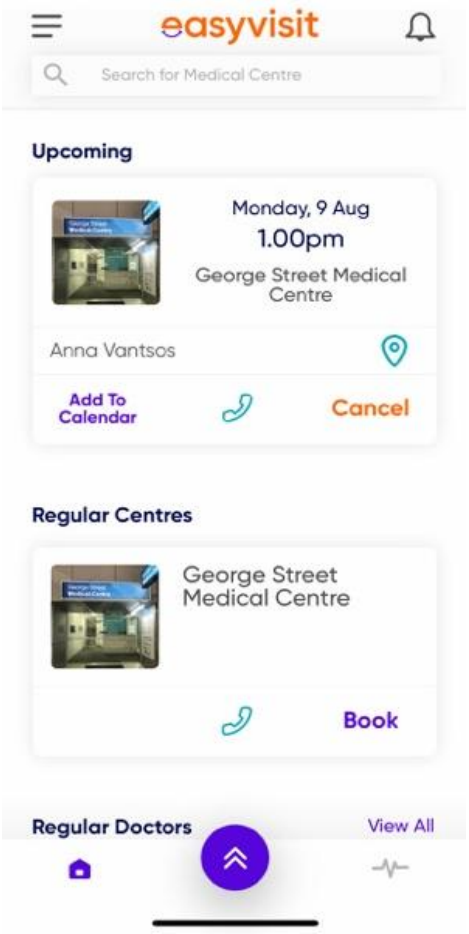


Рисунок 1.2 – Приклад бронювання в застосунку EasyVisit

Docton [6] (рис. 1.3) відзначається тим, що надає користувачам можливість доступу до обширної бази даних лікарів з різними спеціалізаціями, доступних у їхньому регіоні (рис. 1.4).

Рисунок 1.3 – Приклад використання застосунку Docton

Завдяки Docton, користувачі можуть детально ознайомитися з профілями медичних працівників, що включає інформацію про їхні кваліфікації та відгуки інших пацієнтів, забезпечуючи всю необхідну інформацію для обґрунтованого вибору лікаря (рис. 1.5).

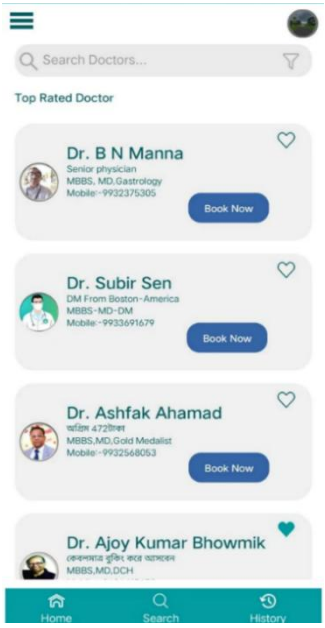


Рисунок 1.4 – Приклад бронювання візиту в застосунку Docton

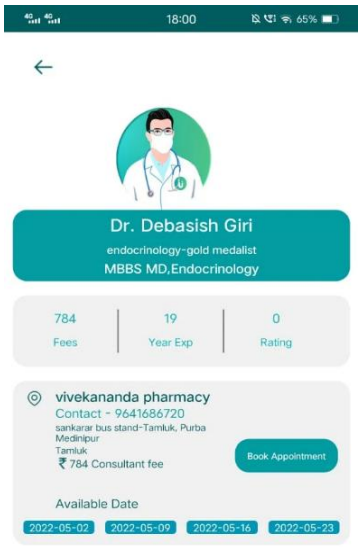


Рисунок 1.5 – Приклад перегляду інформації про лікаря в застосунку Docton

З іншого боку, Health24 [7] (рис. 1.6) є платформою, яка полегшує зв'язок між пацієнтами та медичними закладами по всій Україні, як державними, так і приватними. Цей застосунок надає користувачам можливість легко організувати свої медичні візити та вибрати найзручніший час та лікаря (рис. 1.7). Система категоризації у застосунку допомагає користувачам ефективно навігувати між спеціалізаціями лікарів, спрощуючи процес вибору потрібного спеціаліста.

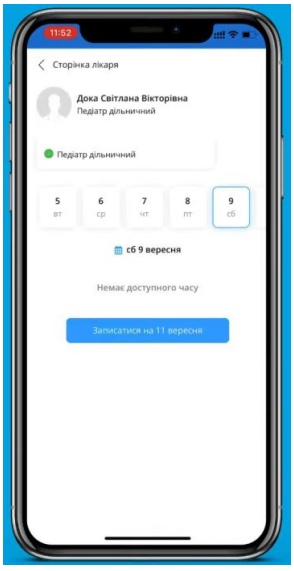


Рисунок 1.6 – Приклад використання застосунку Health24

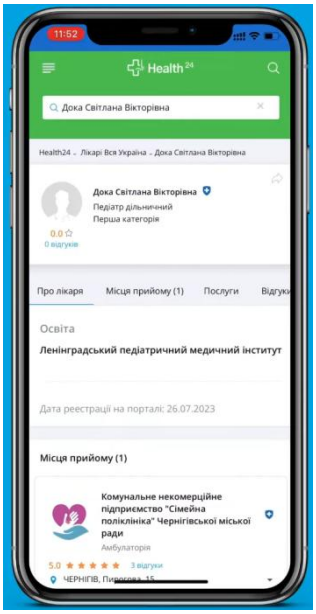


Рисунок 1.7 – Приклад огляду лікаря в застосунку Health24

Ці застосунки ілюструють значимість і потенціал застосування інформаційних технологій у медичній сфері, пропонуючи пацієнтам зручні та ефективні рішення для організації медичних візитів. Розробка таких систем вимагає розуміння потреб користувачів та високого рівня технічної компетенції для створення інтуїтивно зрозумілих, безпечних і ефективних мобільних застосунків, здатних покращити доступ до медичних послуг та загальну якість обслуговування. В результаті аналізу існуючих рішень була складена таблиця (таблиця 1.1), яка дозволяє порівняти функціонал аналогів з власною розробкою.

Таблиця 1.1 – Порівняльні характеристика онлайн платформ

Порівняльна характеристика	EasyVisit	Docton	Health24	Власна розробка
1	2	3	4	5
Обліковий запис пацієнта	1	1	1	1
Функціонал для надання відгуків про медичних фахівців	0	1	1	1

Продовження таблиці 1.1

1	2	3	4	5
Перегляд історії минулих записів до лікарів	1	1	1	1
Створення списку улюблених медичних фахівців	1	1	1	1
Можливість зв'язку зі спеціалістом у чаті в застосунку	0	0	0	1
Сумарний коефіцієнт	3	4	4	5

Аналізуючи дані з таблиці 1.1, можна побачити, що власна розробка показує кращі результати за розглянутими критеріями порівняно з аналогами Docton та Health24 на 20%, а у порівнянні з застосунком EasyVisit – на 40%. Загальний рейтинг власної розробки складає 5 балів з 5 можливих, що є найвищим показником серед усіх аналізованих систем, підкреслюючи її переваги та інноваційний потенціал. Такий підхід не лише забезпечить високий рівень задоволеності користувачів завдяки зручності та ефективності управління медичними записами, але й сприятиме покращенню взаємодії між пацієнтами та медичними спеціалістами завдяки новій функції чату.

1.3 Аналіз методів розв’язання задачі

Розробка програмного забезпечення для комунікацій у системах телемедицини вимагає уважного підходу до вибору технічних рішень, щоб ефективно вирішувати проблеми взаємодії між лікарями та пацієнтами. Існує кілька стратегій, кожна з яких має свої переваги та обмеження, впливаючи на загальну продуктивність і адаптацію системи.

Одна зі стратегій передбачає створення універсальної бібліотеки, яка дозволяє легко інтегрувати комунікаційні модулі в існуючі телемедичні системи. Цей підхід спрощує процес впровадження нових інновацій, однак може призводити до додаткових викликів для розробників, які повинні адаптувати ці модулі під різні медичні системи. Основним недоліком є великий обсяг роботи, необхідний для забезпечення сумісності, що може сповільнити впровадження та збільшити витрати на інтеграцію.

Інший підхід полягає у розробці спеціалізованого сервісу, який може функціонувати як самостійна служба, інтегрована з телемедичними системами. Це знижує необхідність розробки складних систем з нуля та дозволяє зосередитися на поліпшенні специфічних функцій, що відповідають потребам конкретної медичної установи. Такий підхід забезпечує гнучкість у виборі технологій та полегшує масштабування, але може вимагати значних інвестицій у розробку та інфраструктуру.

Створення комплексного мобільного застосунку для комунікацій у телемедицині, який інтегрує всі необхідні модулі та функції, є оптимальним рішенням. Це дозволяє досягти високої індивідуалізації та адаптації системи до потреб кожної медичної установи, уникаючи зовнішніх інтеграцій. Цей підхід також забезпечує повний контроль над безпекою та функціональністю системи, сприяючи інноваціям та оптимізаціям. Проте, такий застосунок вимагає великих ресурсів для розробки та тестування.

Після аналізу переваг та недоліків кожного з підходів було вирішено обрати стратегію створення інтегрованого мобільного застосунку для комунікацій в системах телемедицини. Це забезпечить комплексне та ефективне рішення, яке відповідає вимогам та специфіці медичної галузі, а також забезпечує високий рівень задоволення користувачів, пропонуючи унікальні можливості для ефективної взаємодії пацієнтів з медичними спеціалістами.

1.4 Постановка задач дослідження

Після детального аналізу сильних та слабких сторін існуючих систем комунікації в телемедицині, були визначені основні напрями для розробки програмного рішення:

– розробити ефективний метод комунікації, який дозволить забезпечити надійний зв'язок між пацієнтами та медичними спеціалістами;

– створити інтуїтивно зрозумілий та простий у використанні графічний інтерфейс мобільного застосунку, адаптований до різних типів пристроїв;

– розробити мобільний застосунок, який інтегрує в себе всі необхідні функції для ефективної комунікації, включаючи можливість проведення відео консультацій, обміну повідомленнями, а також надання зворотного зв'язку від пацієнтів;

– провести ретельне тестування розробленої системи, щоб забезпечити її стабільність, ефективність та користувацьку зручність у реальних умовах медичної практики.

Ці завдання спрямовані на створення мобільної системи телемедицини, яка не тільки відповідатиме потребам сучасних пацієнтів у зручному та швидкому доступі до медичної допомоги на відстані, але й надасть медичним закладам потужний інструмент для оптимізації комунікацій та покращення якості обслуговування.

Технічне завдання на розробку наведено в додатку А.

1.5 Висновки

У першому розділі дослідження був здійснений аналіз сучасного стану комунікацій в системах телемедицини», з акцентом на такі платформи як EasyVisit, Docton, та Health24.

Вивчення цих систем показало, що незважаючи на їхню популярність та ефективність у медичній галузі, вони мають певні недоліки, які можна вирішити шляхом розробки власного продукту. Однією з ключових виявлених проблем була відсутність можливості безпосереднього спілкування з медичними фахівцями через систему, а також обмеження у персоналізації послуг. Це підкреслило важливість створення системи автоматизованого запису, яка б інтегрувала існуючі функції та додавала нові можливості, включаючи реальний час спілкування зі спеціалістами та розширені налаштування персоналізації.

На основі цього аналізу було визначено конкретні завдання, які потрібно реалізувати для успішного запуску проєкту. Ці завдання включають розробку зручного та легкого у використанні графічного інтерфейсу мобільного застосунку, який буде адаптований під різні типи пристроїв, створення функцій для легкого запису на прийом, відстеження стану записів. Також, заплановано проведення ретельного тестування системи, щоб гарантувати її стабільність, ефективність та користувацьку зручність у реальних умовах.

Результати аналізу та порівняння з існуючими системами підтверджують, що імплементація власної системи автоматизованого запису до лікарів може значно вдосконалити доступ до медичних послуг, надаючи користувачам більшу зручність, ефективність та персоналізоване взаємодія. Це стане важливим кроком до оптимізації робочих процесів у медичних закладах та збільшення загального рівня задоволеності пацієнтів.

2 РОЗРОБКА МЕТОДУ, МОДЕЛІ ТА АЛГОРИТМУ РОБОТИ СИСТЕМИ

2.1 Аналіз даних

Обробка і аналіз даних є ключовими аспектами роботи сучасних програмних застосунків, зокрема у сфері телемедицини. Обробка даних включає збір, фільтрацію, зберігання та трансформацію інформації з різних джерел у формат, який є зручним для подальшого аналізу.

Програмний застосунок отримує персональні дані від користувача, що дозволяє йому здійснювати записи до лікаря, створювати список улюблених лікарів, вести спілкування в чатах з лікарями, переглядати історію записів і т.д. Дані зберігаються у хмарній базі даних, тому для роботи із застосунком потрібне підключення до Інтернету.

У модулі для авторизації та реєстрації користувачі можуть створювати облікові записи та отримувати доступ до своїх персональних даних і медичних послуг. Вхідними даними є інформація, введена користувачем, така як ім'я, електронна пошта, пароль. Користувачі можуть реєструватися, заповнюючи відповідні поля у формі реєстрації, та авторизуватися, вводячи свій логін та пароль у формі авторизації. Вихідними даними є графічний інтерфейс, що підтверджує успішну реєстрацію або авторизацію, та забезпечує доступ до функцій застосунку, таких як запис на консультацію.

У модулі лікарських консультацій вхідні дані включають інформацію про лікаря, дані пацієнта, а також дату і час запланованої консультації. Для вибору дати використовується спеціальний інтерфейс у вигляді календаря. Користувачі не можуть обирати дати, раніше сьогоднішньої, а також дати, що припадають на вихідні. Для вибору часу надається список доступних годин, з якого користувач може обрати лише один варіант. Після введення даних вони зберігаються в хмарній базі даних. У разі успішного бронювання візиту користувач отримує

повідомлення про успішне завершення операції. Вихідними даними є графічний елемент списку консультацій, який містить інформацію про лікаря, а також заплановані дату і час.

У модулі для перегляду улюблених лікарів користувачі можуть зберігати інформацію про лікарів, з якими вони найчастіше консультуються або до яких планують звертатися в майбутньому. Вхідними даними є інформація про лікаря, вибрана зі списку в системі. Користувачі можуть додавати лікарів до списку улюблених через інтерфейс застосунку, переглядати їхні профілі та швидко записуватися на консультацію. Вихідними даними є графічний інтерфейс зі списком улюблених лікарів, який забезпечує швидкий доступ до важливої медичної інформації та оптимізує процес організації медичних консультацій.

У модулі для здійснення відеодзвінків до лікаря користувачі можуть проводити консультації з медичними фахівцями у режимі реального часу за допомогою відеозв'язку. Вхідними даними є відео- та аудіопотоки, передані від користувача та лікаря під час дзвінка. Користувачі можуть ініціювати відеодзвінок через інтерфейс застосунку, отримуючи можливість безпосередньо спілкуватися з лікарем і обговорювати питання щодо свого здоров'я. Вихідними даними є графічний інтерфейс, що відображає вікно відеодзвінка, забезпечуючи зручний і швидкий доступ до візуальних медичних консультацій. Цей модуль підвищує ефективність взаємодії між пацієнтом і лікарем, дозволяючи отримувати кваліфіковану допомогу без необхідності фізичної присутності.

У модулі зі статистикою лікарі можуть переглядати зведену інформацію про свою медичну практику. Вхідними даними є записи про пацієнтів, майбутні прийоми, рейтинги та відгуки від пацієнтів. Лікарі можуть отримувати доступ до дешборду через веб-інтерфейс, де відображаються ключові показники їхньої роботи. Ці дані допомагають лікарям аналізувати свою діяльність і приймати обґрунтовані рішення щодо покращення медичних послуг. Вихідними даними є

графічний інтерфейс, що містить зведені дані про діяльність лікаря, а також останні текстові відгуки від пацієнтів. Цей модуль підвищує ефективність роботи лікаря, дозволяючи краще розуміти результати своєї практики.

2.2 Розробка методу побудови вікон застосунку

Для реалізації графічного інтерфейсу користувача в застосунку на Flutter, використовується розроблений метод побудови вікон застосунку:

1. Виконання функції ‘main()’, яка є вхідною точкою в застосунок та з якої відбувається запуск програми.

2. Ініціалізація форматування дати за допомогою функції ‘initializeDateFormatting()’, яка налаштовує локаль на 'uk_UA'.

3. Запуск ‘MyApp’, який будує графічний інтерфейс у Flutter-застосунку.

4. Виконання методу ‘build()’ в класі ‘MyApp’, який задає початкову установку параметрів під час ініціалізації застосунку.

5. Побудова та запуск головного екрана ‘AuthScreen()’, який першим з’являється при запуску застосунку.

6. Використання ‘ChangeNotifierProvider’ для надання моделі аутентифікації ‘AuthModel’ до всіх компонентів застосунку.

7. Налаштування теми застосунку, включаючи параметри для елементів інтерфейсу, таких як поля введення та нижня панель навігації.

8. Реалізація навігації між екранами за допомогою маршрутизації, яка визначена в ‘routes’.

Для побудови вікон застосунку використано Flutter. Flutter – це SDK з відкритим вихідним кодом для розробки застосунків від Google, який дозволяє створювати високоякісні нативні інтерфейси для мобільних, веб та настільних застосунків з єдиного коду.

Flutter пропонує декларативний підхід до розробки користувацьких інтерфейсів. Замість традиційного імперативного підходу, де розробник описує кроки для створення та оновлення інтерфейсу, у Flutter використовується декларативний опис того, як повинен виглядати інтерфейс у будь-який момент часу. Це дозволяє створювати більш простий та зрозумілий код, полегшуючи розробку та підтримку застосунків [8].

Flutter також надає широкий набір віджетів для створення різноманітних елементів інтерфейсу, таких як кнопки, списки, складні макети та анімації. Віджети Flutter забезпечують високий рівень інтерактивності та гнучкості, що робить їх ідеальними для створення сучасних користувацьких інтерфейсів.

Для побудови списків у застосунку використовується такий компонент як ‘ListView’. ‘ListView’ є ключовим компонентом у Flutter, який використовується для створення прокручуваних списків у мобільних за стосунках [9]. Він надає ефективне розміщення елементів у списку, а також автоматичне завантаження та видалення елементів при прокручуванні, що робить його ідеальним для роботи з великими наборами даних.

Навігація між екранами застосунку реалізується за допомогою навігаційного маршруту. У Flutter для цього можна використовувати ‘Navigator’ та ‘MaterialPageRoute’, що забезпечують зручний спосіб визначення та управління навігацією в застосунку.

2.3 Розробка моделі системи

Для кращого розуміння архітектури розроблюваного застосунка було побудовано модель системи у вигляді діаграми класів (рис. 2.1) [10].

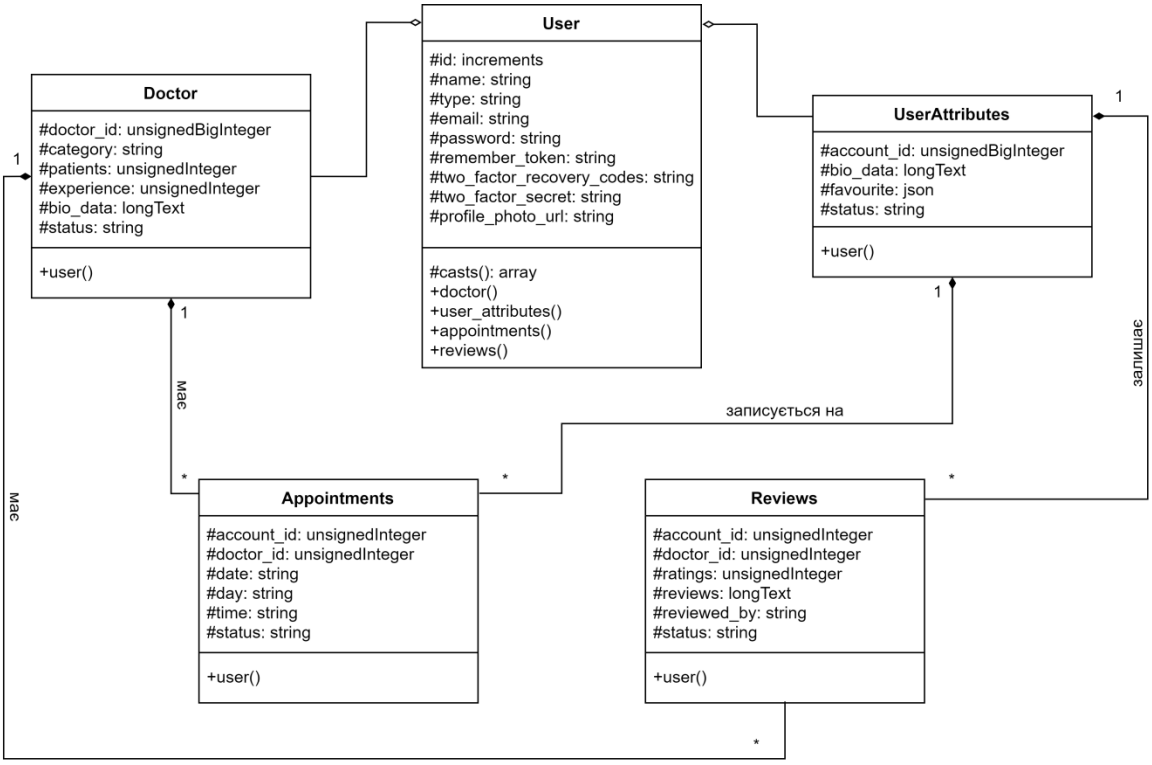


Рисунок 2.1 – Модель роботи системи з використанням діаграми класів

Основним класом у цій системі є ‘User’, який зберігає інформацію про користувачів. Поля цього класу включають ‘id’, що є унікальним ідентифікатором, ‘name’ для зберігання імені, ‘type’, що визначає тип користувача, ‘email’ для електронної пошти, ‘password’ для пароля, ‘remember_token’ для токена запам'ятовування, ‘two_factor_recovery_codes’ і ‘two_factor_secret’ для двофакторної аутентифікації, а також ‘profile_photo_url’ для зберігання URL профільного фото.

Клас ‘Doctor’ має унікальний зв’язок один до одного з класом ‘User’, що означає, що кожен лікар у системі також є користувачем. Поля класу ‘Doctor’ включають ‘doctor_id’, що є унікальним ідентифікатором лікаря, ‘category’, що визначає спеціалізацію лікаря, ‘patients’, що зберігає кількість пацієнтів, ‘experience’ для зберігання років досвіду, ‘bio_data’ для зберігання біографічних даних, та ‘status’, що визначає поточний статус лікаря.

Клас ‘UserAttributes’ також має зв’язок один до одного з класом ‘User’, що дозволяє кожному пацієнту мати окремі атрибути. Поля цього класу включають ‘account_id’, що є унікальним ідентифікатором аккаунта, ‘bio_data’ для зберігання додаткових біографічних даних, ‘favourite’, що зберігає список улюблених лікарів у форматі JSON, та ‘status’, що визначає поточний статус користувача.

Клас ‘Appointments’ відображає записи на прийоми і має зв’язок багато до одного з класом ‘User’. Це означає, що один користувач може мати багато записів на прийоми. Поля класу ‘Appointments’ включають ‘account_id’, що є унікальним ідентифікатором аккаунта користувача, ‘doctor_id’, що вказує на лікаря, до якого записався користувач, ‘date’ для зберігання дати прийому, ‘day’ для зберігання дня тижня, ‘time’ для зберігання часу прийому, та ‘status’, що визначає поточний статус прийому.

Клас ‘Reviews’ зберігає відгуки користувачів про лікарів і також має зв’язок багато до одного з класом ‘User’. Поля класу ‘Reviews’ включають ‘account_id’, що є унікальним ідентифікатором аккаунта користувача, ‘doctor_id’, що вказує на лікаря, якого стосується відгук, ‘ratings’, що зберігає рейтинг, ‘reviews’ для зберігання тексту відгуку, ‘reviewed_by’, що вказує на автора відгуку, та ‘status’, що визначає поточний статус відгуку.

Ця діаграма класів детально ілюструє складні взаємозв’язки між різними компонентами системи, забезпечуючи повне уявлення про те, як користувачі взаємодіють з лікарями, як зберігаються атрибути користувачів, як організовуються записи на прийоми, і як зберігаються відгуки.

2.4 Розробка загального алгоритму роботи системи та діаграм станів

Загальний алгоритм застосунку є основним для побудови програми. Він відображає взаємозв’язок між модулями всієї системи та загальну її поведінку (рис. 2.2).

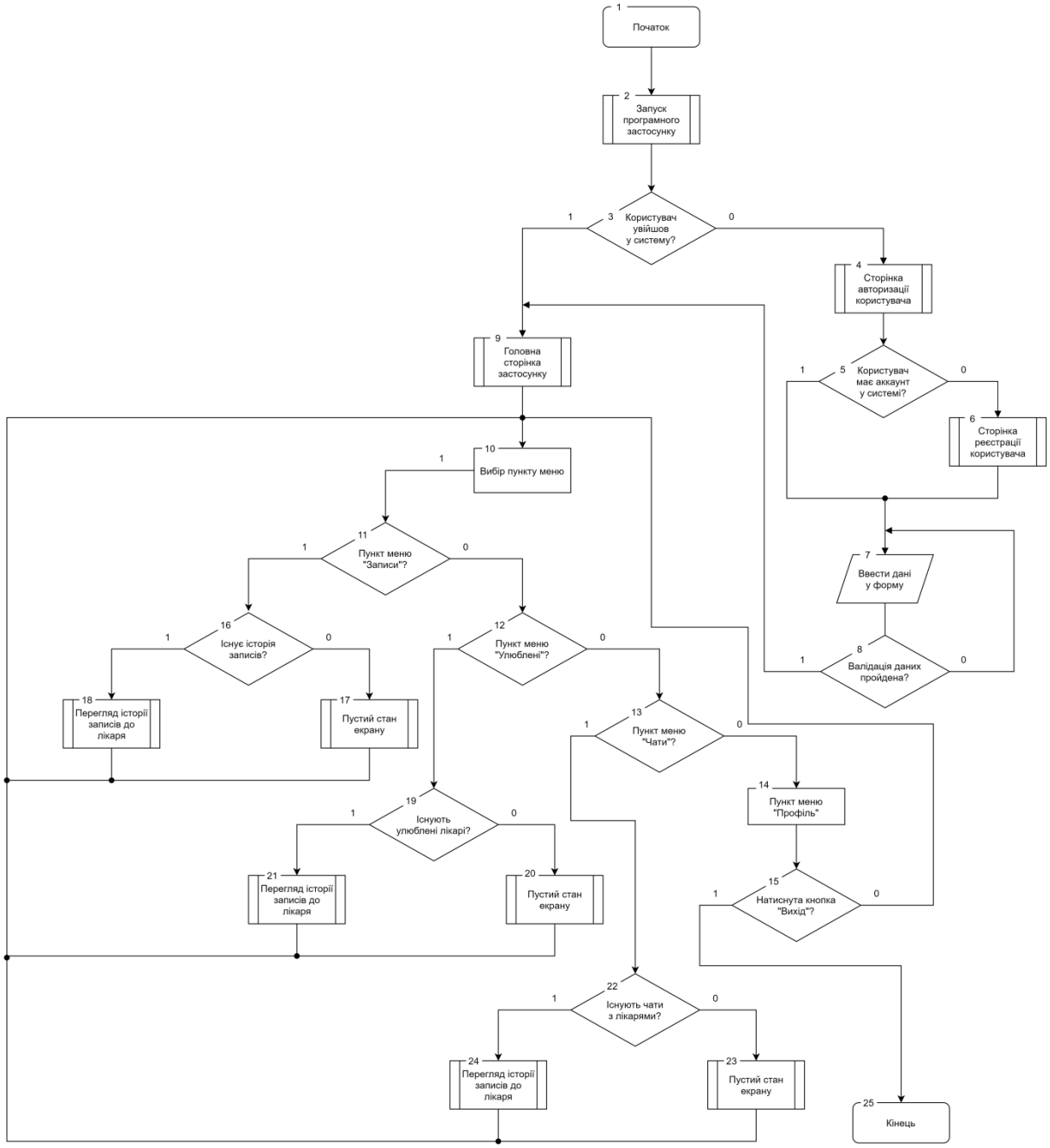


Рисунок 2.2 – Загальний алгоритм роботи системи

На початку користувач перевіряється на наявність входу в систему. Якщо користувач не увійшов, він перенаправляється на сторінку авторизації або реєстрації. У разі наявності облікового запису користувач вводить свої дані у форму, яка потім перевіряється на коректність. Після успішної валідації даних

користувач потрапляє на головну сторінку застосунку. На головній сторінці користувачеві необхідно обрати пункт меню з доступних опцій. Залежно від обраного пункту меню, відбувається перехід до відповідного вікна.

Діаграми станів [11] є інструментом моделювання, який використовується для опису поведінки системи або її частин шляхом визначення різних станів об'єкта та переходів між цими станами. Вони надають графічне представлення станів об'єкта протягом його життєвого циклу і показують, як об'єкт реагує на внутрішні або зовнішні події. Основні компоненти діаграми станів включають стан, подію, дію і перехід. Вони сприяють чіткому визначенню логіки поведінки системи, що особливо корисно при розробці складних застосунків.

Діаграму станів модуля авторизації та реєстрації користувача наведено на рисунку 2.3.

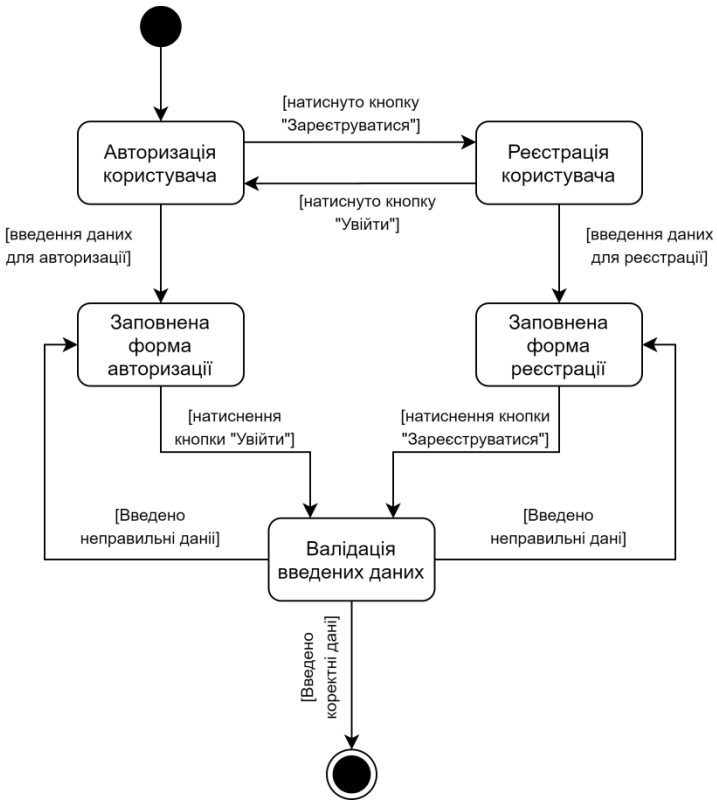


Рисунок 2.3 – Діаграма станів для модуля авторизації та реєстрації користувача

Діаграму станів модуля здійснення запису до лікаря наведено на рисунку 2.4. Вона ілюструє процес запису користувача на прийом до лікаря, починаючи з вибору дати і часу, і закінчуючи успішним бронюванням або обробкою помилок.

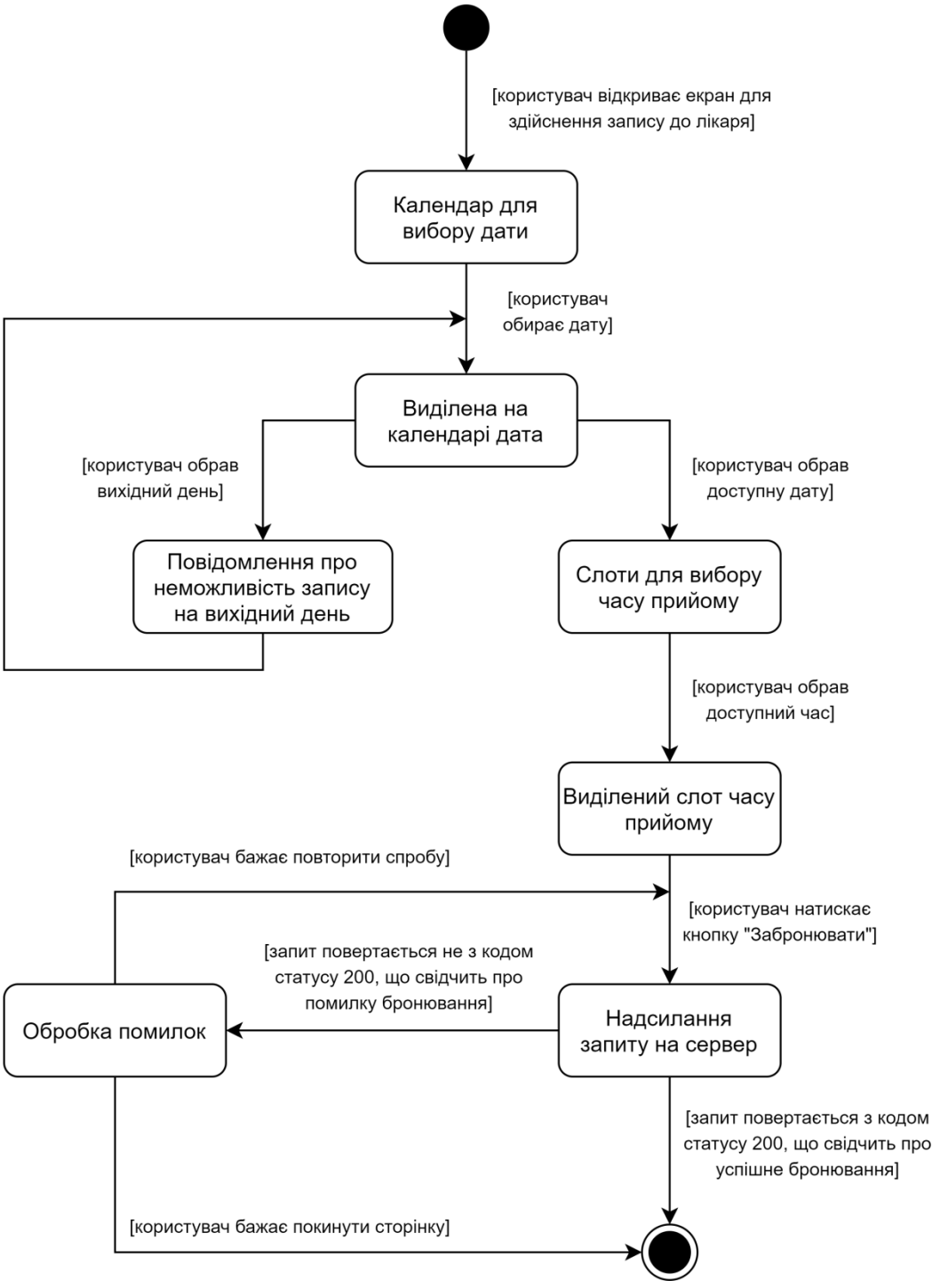


Рисунок 2.4 – Діаграма станів модуля здійснення запису до лікаря

Діаграму станів модуля перегляду інформації про лікаря наведено на рисунку 2.5.

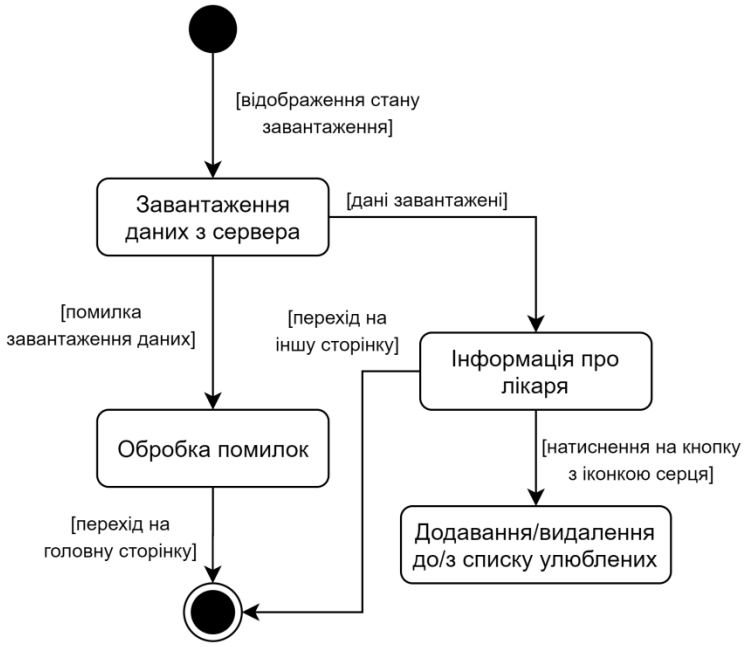


Рисунок 2.5 – Діаграма станів модуля перегляду інформації про лікаря

Діаграму станів модуля дешборду веб-клієнту лікаря наведено на рисунку 2.6.

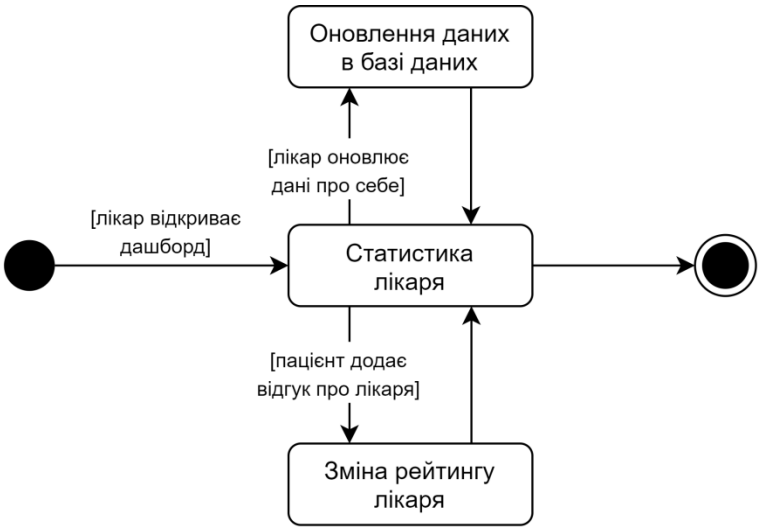


Рисунок 2.6 – Діаграма станів модуля дешборду веб-клієнту лікаря

2.5 Висновки

У другому розділі детально розглянуто основні аспекти розробки застосунку для телемедицини, включаючи обробку та аналіз даних, зберігання інформації в хмарній базі, а також роботу з персональними даними користувачів для запису на консультації. Особливу увагу приділено модулям авторизації, реєстрації, бронювання консультацій, збереження улюблених лікарів, відеодзвінків та перегляду статистики медичної практики.

Метод побудови вікон застосунку на платформі Flutter продемонстровано через створення інтерфейсу користувача за допомогою декларативного підходу та широкого набору віджетів. Це дозволяє розробляти простий і зрозумілий код, полегшуючи підтримку застосунку. ‘ListView’ використовується для створення прокручуваних списків, а ‘Navigator’ і ‘MaterialPageRoute’ забезпечують зручну навігацію між екранами.

Загальний алгоритм роботи системи та діаграми станів чітко показують взаємозв’язки між модулями програми та їх поведінку. Діаграми станів для авторизації, реєстрації, запису до лікаря, перегляду інформації про лікаря та дешборду лікаря ілюструють, як об’єкти реагують на події, що полегшує розробку складних застосунків.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного забезпечення

Dart – це об'єктно-орієнтована мова програмування, розроблена компанією Google. Вона була представлена у 2011 році і з тих пір стала важливим інструментом для розробки веб та мобільних застосунків, особливо з використанням фреймворку Flutter [12]. Dart поєднує в собі простоту синтаксису та потужні можливості, що робить його ідеальним вибором для створення високопродуктивних мобільних застосунків.

Однією з головних особливостей Dart є його інтеграція з Flutter, який дозволяє розробникам створювати нативні мобільні застосунки для iOS та Android з єдиною кодовою базою. Це значно знижує витрати на розробку та обслуговування застосунків, оскільки немає потреби писати окремий код для кожної платформи.

C# – це об’єктно-орієнтована мова програмування, розроблена компанією Microsoft для розробки програмного забезпечення для платформи .NET [13]. Для створення мобільних застосунків на C# використовується платформа Xamarin, яка дозволяє розробникам створювати застосунки для iOS та Android. Хоча Xamarin забезпечує хорошу продуктивність та можливості, вона може бути складнішою у використанні порівняно з Flutter та Dart.

Swift – це мова програмування, розроблена компанією Apple для розробки програмного забезпечення для iOS, macOS, watchOS та tvOS [14]. Swift поєднує в собі простий синтаксис з потужними можливостями, що робить його ідеальним вибором для розробки застосунків для екосистеми Apple. Однак, Swift не підтримує розробку застосунків для Android, що робить його менш гнучким порівняно з Dart.

Python – це інтерпретована мова програмування, яка відома своєю простотою і читабельністю. Вона широко використовується для розробки веб-застосунків, наукових обчислень, машинного навчання та автоматизації [15]. Однак, для розробки мобільних застосунків Python використовується рідко, і для цього зазвичай застосовуються фреймворки, такі як Kivy, які можуть бути менш продуктивними та гнучкими порівняно з Flutter та Dart.

Порівняння функціональних характеристик мов програмування Dart, C#, Swift та Python наведено в таблиці 3.1.

Таблиця 3.1 – Таблиця порівняння функціональних характеристик мов програмування

Характеристика	Dart	C#	Swift	Python
Об’єктно-орієнтованість	1	1	1	1
Функціональність	1	1	1	1
null-безпека	1	0	1	0
Розширюваність	1	1	1	1
Платформонезалежність	1	0	0	1
Лямбда вирази	1	1	1	1
Загальний коефіцієнт	6	4	5	5

Dart, маючи вищий сумарний коефіцієнт, опереджаючи своїх конкурентів в розробці мобільних застосунків і саме тому був обраний для розробки застосунку.

Компанії-розробники програмного забезпечення пропонують широкий вибір сучасних середовищ розробки мобільних застосунків. Розглянемо найпопулярніші з них: Visual Studio Code (VS Code), Eclipse, Android Studio та Visual Studio.

Visual Studio Code (VS Code) – це безкоштовне, легке та потужне інтегроване середовище розробки (IDE), розроблене компанією Microsoft [16]. VS Code

підтримує велику кількість мов програмування та платформ, завдяки чому є універсальним інструментом для розробки програмного забезпечення. Однією з головних переваг VS Code є його розширюваність за допомогою численних плагінів, що дозволяє налаштувати IDE під конкретні потреби розробника.

VS Code забезпечує інтеграцію з різними інструментами та сервісами, такими як Git, Docker, та різними хмарні платформи, що робить його потужним інструментом для сучасної розробки.

Eclipse – це безкоштовне інтегроване середовище розробки (IDE) з відкритим вихідним кодом, яке підтримує велику кількість мов програмування, включаючи Java, C++, Python та інші [17]. Eclipse має потужний набір інструментів для розробки програмного забезпечення, включаючи підтримку налагодження, рефакторингу, контролю версій та багато іншого. Eclipse підтримує розробку мобільних застосунків через різні плагіни, такі як Eclipse Andmore для Android. Однак, порівняно з іншими IDE, Eclipse може бути менш зручним у використанні та налаштуванні для мобільної розробки.

Android Studio – це інтегроване середовище розробки (IDE) для платформи Android, розроблене Google. Це основний інструмент для створення мобільних програм для платформи Android, який надає розробникам усе необхідне для розробки, тестування, налагодження та розгортання програм. Android Studio базується на середовищі розробки IntelliJ IDEA і включає в себе безліч інструментів, таких як вбудований редактор коду, дизайнер інтерфейсу, інструменти для роботи з базами даних, інструменти тестування та інші корисні інструменти [18].

Visual Studio – це потужне інтегроване середовище розробки (IDE), розроблене компанією Microsoft. Воно підтримує велику кількість мов програмування та платформ, включаючи .NET, C++, Python та інші. Visual Studio надає широкий

набір інструментів для розробки програмного забезпечення, включаючи підтримку налагодження, контролю версій, тестування та розгортання [19].

Visual Studio підтримує розробку мобільних застосунків через Xamarin, що дозволяє створювати застосунки для iOS та Android з використанням єдиної кодової бази на C#. Однак, порівняно з VS Code, Visual Studio може бути важчою та менш гнучкою у налаштуванні.

Порівняння функціональних характеристик середовищ наведено в таблиці 3.2.

Таблиця 3.2 – Таблиця порівняння функціональних характеристик середовищ програмування

Характеристика	VS Code	Eclipse	Android Studio	Visual Studio
Підтримка мобільної розробки	1	1	1	1
Вбудована розробка застосунків для Android	1	0	1	0
Можливість створення та розгортання на Android	1	1	1	1
Підтримка різних мов програмування	1	1	0	1
Можливість підключення сторонніх плагінів	1	1	1	1
Підтримка систем контролю версій	1	1	1	1
Підтримка створення та розгортання в хмарних службах	1	1	1	1
Загальний коефіцієнт	7	6	6	6

За таблицею, Visual Studio Code (VS Code) має вищий сумарний коефіцієнт порівняно з іншими середовищами розробки завдяки своїй гнучкості, легкості, розширюваності та підтримці великої кількості мов програмування. Це робить VS Code найкращим середовищем для розробки мобільного застосунку.

3.2 Розробка модуля для авторизації та реєстрації користувача

Призначенням модуля автентифікації є створення простого та зручного інтерфейсу для входу та реєстрації користувачів у застосунку. Користувач спершу бачить екран з можливістю вибору між входом та реєстрацією (рис. 3.1).

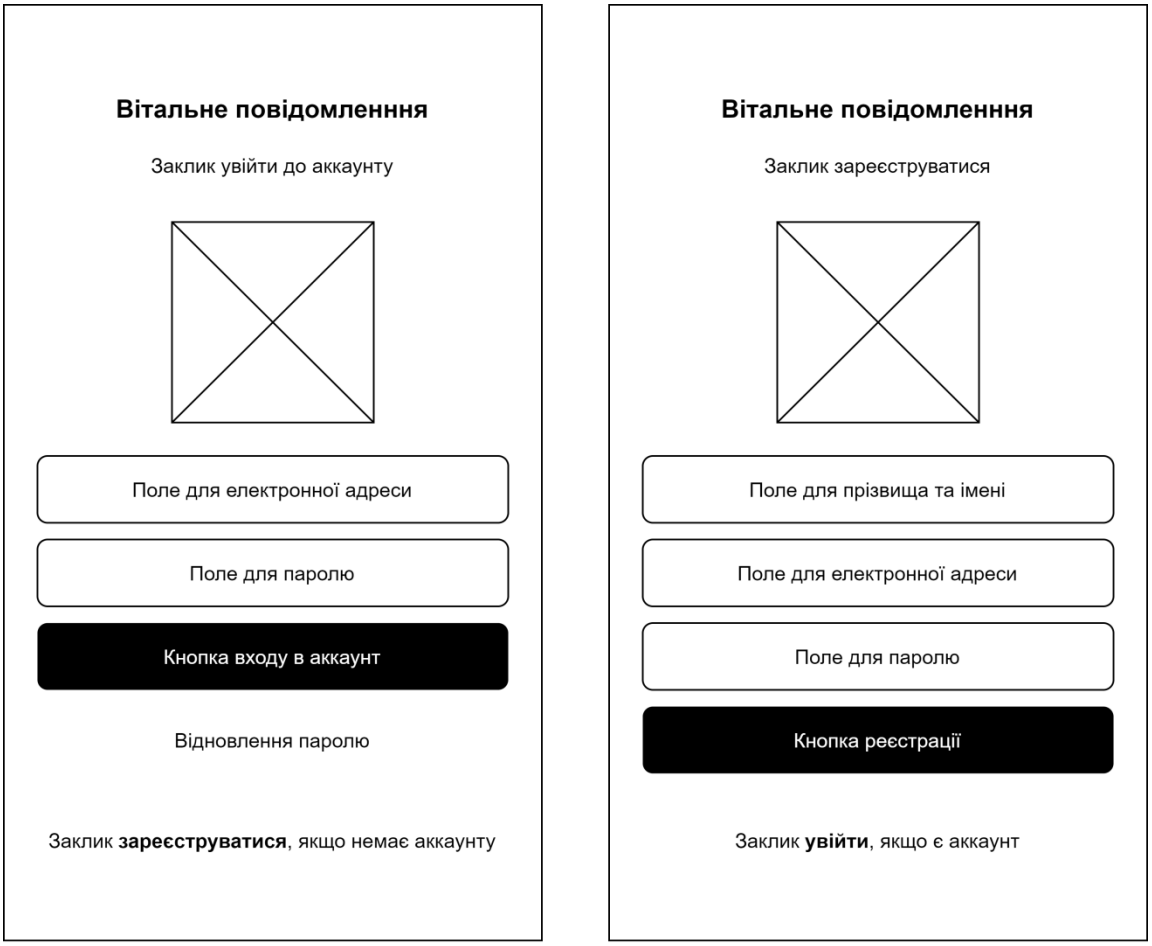


Рисунок 3.1 – Схема інтерфейсу сторінок авторизації та реєстрації

Для створення облікового запису або входу в існуючий, йому потрібно вибрати відповідну опцію, після чого програма відображає форму для введення необхідних даних.

Модуль реалізовано з використанням Flutter і ‘StatefulWidget’. Flutter – це декларативна бібліотека для розробки графічних інтерфейсів мобільних застосунків на платформі Android та iOS. Вона була представлена компанією Google і стала новим підходом до створення користувацького інтерфейсу.

Використання ‘StatefulWidget’ дозволяє віджетам зберігати свій стан, що важливо для динамічних інтерфейсів, які повинні реагувати на дії користувачів. Завдяки цьому, застосунок може легко перемикатися між режимами входу та реєстрації, автоматично оновлюючи інтерфейс відповідно до поточного стану.

Цей підхід забезпечує інтерактивний і зручний інтерфейс для автентифікації користувачів, використовуючи сучасні можливості Flutter для створення адаптивного та стильного інтерфейсу.

Було побудовано діаграму класів для модуля авторизації та реєстрації (рис. 3.2). Ця діаграма класів описує структуру та взаємодію між основними компонентами модуля автентифікації, включаючи віджети, стани та конфігурації.

Клас ‘AuthScreen’ є ‘StatefulWidget’ і представляє екран автентифікації. Він має метод ‘createState()’, який створює стан ‘AuthScreenState’. Цей стан містить логіку та UI для екрану автентифікації. Поле ‘_isSignInMode’ визначає, чи користувач знаходиться в режимі входу або реєстрації. Методи цього класу відповідають за побудову інтерфейсу користувача, ініціалізацію конфігурації, створення тексту привітання, опису, ілюстрації, кнопки для відновлення пароля та кнопки для перемикання між режимами автентифікації. Метод ‘_toggleAuthMode()’ змінює режим.

Клас ‘EasyMedHelpConfiguration’ містить налаштування інтерфейсу застосунку. Він включає інформацію про розміри екрану, кольори теми, тему

тексту, значення для відступів та проміжків, а також стилі рамок для полів введення. Методи цього класу дозволяють ініціалізувати конфігурацію та отримувати розміри екрану.

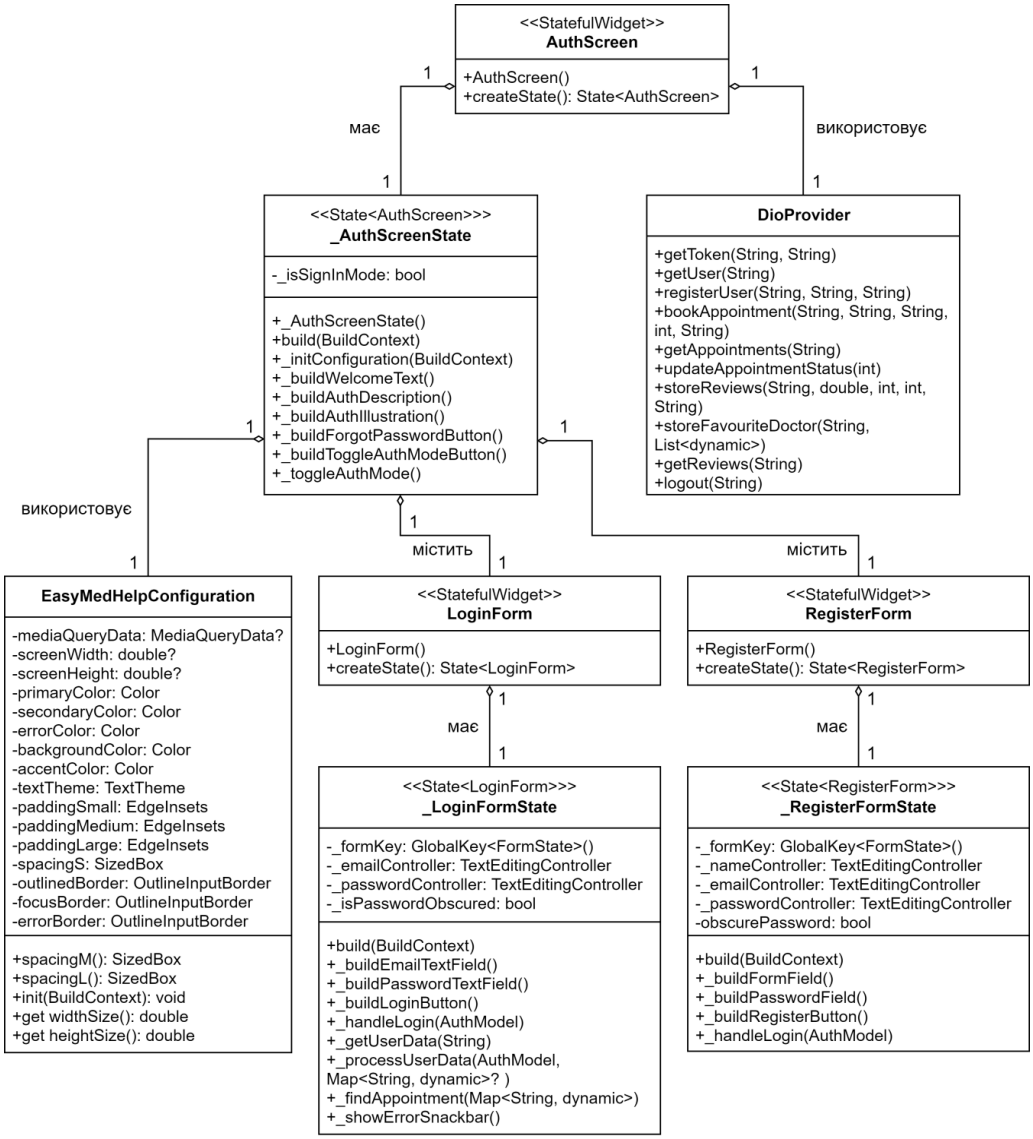


Рисунок 3.2 – Діаграма класів для модуля авторизації та реєстрації

Клас ‘DioProvider’ відповідає за мережеві запити. Він включає методи для отримання токену, інформації про користувача, реєстрації користувача, бронювання записів, отримання списку записів, оновлення статусу запису, збереження відгуків, оновлення списку улюблених лікарів, отримання відгуків та

виходу з системи.

Клас ‘LoginForm’ є ‘StatefulWidget’ і представляє форму для входу. Він має метод ‘createState()’, який створює стан ‘LoginFormState’. Цей стан містить логіку та UI для форми входу. Поля цього класу включають ключ форми, контролери для полів введення та змінну для приховування пароля. Методи відповідають за побудову інтерфейсу користувача, створення текстових полів для вводу, кнопки входу, обробку входу, отримання даних користувача, обробку даних користувача, знаходження записів та показ повідомлення про помилку.

Клас ‘RegisterForm’ є ‘StatefulWidget’ і представляє форму для реєстрації. Він має метод ‘createState()’, який створює стан ‘RegisterFormState’. Цей стан містить логіку та UI для форми реєстрації. Поля цього класу включають ключ форми, контролери для полів введення та змінну для приховування пароля. Методи відповідають за побудову інтерфейсу користувача, створення текстових полів для вводу, кнопки реєстрації та обробку реєстрації.

3.3 Розробка модуля перегляду списку лікарів

Цей модуль відповідає за створення інтерфейсу для відображення списку лікарів у застосунку. Він реалізований з використанням віджета Column та методу List.generate.

Кожен елемент списку представляє картку лікаря з інформацією про нього, а також вказує, чи є цей лікар у списку улюблених спеціалістів користувача. Кожна картка лікаря включає інформацію про відповідного лікаря та візуальний індикатор, який відображає, чи є цей лікар у списку фаворитів. Ця інформація забезпечується шляхом перевірки наявності ідентифікатора лікаря у списку улюблених, що дозволяє користувачу швидко знаходити потрібних спеціалістів. Таке рішення сприяє зручності користування інтерфейсом та підвищує ефективність взаємодії користувача з системою обліку лікарів.

Графічна схема цього модуля наведена на рисунку 3.3.

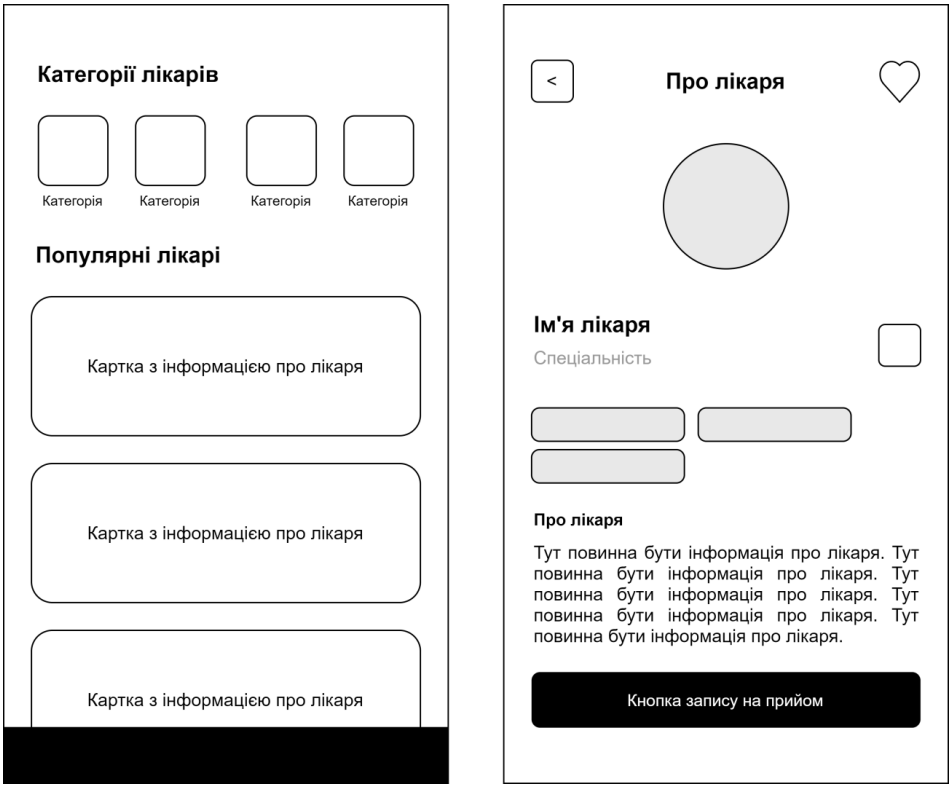


Рисунок 3.3 – Графічна схема модуля перегляду списку лікарів

Функція DoctorCard() відповідає за реалізацію графічного інтерфейсу для елемента, що відображає інформацію про лікаря. Вона містить в собі зображення профілю лікаря, його ім'я, категорію, кнопку для додавання або видалення лікаря зі списку улюблених, а також кнопку для запису на прийом. Графічну схему зображення цього елементу наведено на рисунку 3.4.

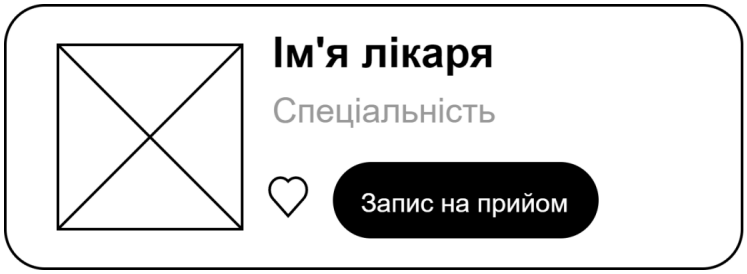


Рисунок 3.4 – Графічна схема елемента функції DoctorCard()

3.4 Розробка модуля для здійснення відеодзвінків до лікаря

Призначення модуля полягає в наданні зручного та функціонального інтерфейсу для проведення відеоконсультацій з лікарем, що забезпечує комфортний досвід для користувачів. Алгоритм описаний у вигляді блок-схеми (рис. 3.5) та відповідає за управління відеодзвінком, включаючи початок і завершення дзвінка, перемикання стану мікрофону та камери, а також надання можливості залишити відгук після завершення дзвінка.

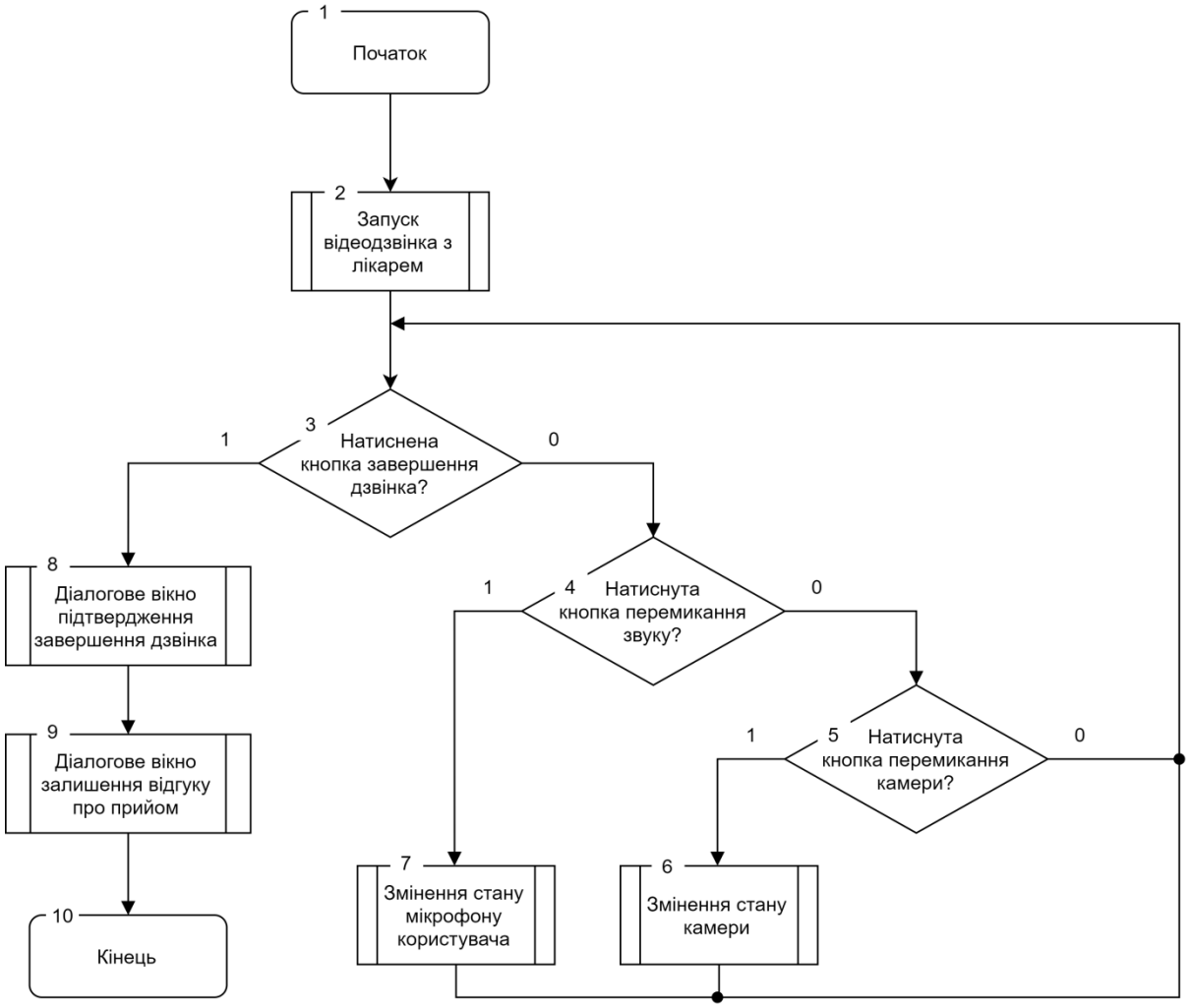


Рисунок 3.5 – Блок-схема алгоритму роботи модуля для здійснення відеодзвінків до лікаря

Алгоритм починається з ініціалізації процесу відеодзвінка. На наступному етапі відбувається запуск відеодзвінка з лікарем, під час якого ініціалізується камера та мікрофон.

Система постійно перевіряє, чи натиснута кнопка завершення дзвінка. Якщо кнопка натиснута, відображається діалогове вікно підтвердження завершення дзвінка. Якщо кнопка завершення дзвінка не натиснута, система перевіряє, чи натиснута кнопка перемикання звуку. У разі натискання цієї кнопки змінюється стан мікрофону користувача.

Якщо кнопка перемикання звуку не натиснута, система перевіряє, чи натиснута кнопка перемикання камери. Якщо натиснута кнопка перемикання камери, змінюється стан камери.

Якщо жодна з кнопок не натиснута, система повертається до перевірки кнопки завершення дзвінка.

При натисканні кнопки завершення дзвінка та підтвердженні дії користувачем, відображається діалогове вікно залишення відгуку про прийом у лікаря. Після завершення залишення відгуку процес відеодзвінка завершується.

Важливим елементом інтерфейсу відеодзвінка є відображення часу, який триває дзвінок. Це дозволяє користувачу легко відстежувати тривалість дзвінка, що може бути корисно для контролю за витраченим часом (рис. 3.6).

```
children: [
  Text(
    '$minutes:${seconds < 10 ? '0$seconds' : seconds}',
    style: TextStyle(fontSize: 18.0),
  ), // Text
],
```

Рисунок 3.6 – Таймер тривалості дзвінка з лікарем

Відображення таймера здійснюється за допомогою компонента Text, який динамічно оновлюється відповідно до поточного значення змінних minutes та

seconds. Форматування часу включає в себе хвилини та секунди, причому для секунд використовується умовний оператор для додавання провідного нуля, якщо значення секунд менше десяти. Це забезпечує зручне і зрозуміле відображення часу у форматі "хвилини:секунди".

3.5 Розробка модуля зі статистикою для лікаря

Модуль статистики для лікаря призначений для надання зручного інтерфейсу, який дозволяє лікарям переглядати важливу інформацію про свою діяльність та взаємодію з пацієнтами. Основні функції модуля включають відображення кількості майбутніх прийомів, загальної кількості пацієнтів, середнього рейтингу лікаря та кількості відгуків, залишених пацієнтами. Такий модуль сприяє покращенню якості медичних послуг, оскільки дозволяє лікарям отримувати зворотний зв'язок і оперативно реагувати на відгуки пацієнтів.

Модуль статистики реалізовано у вигляді інформаційної панелі, яка містить кілька ключових показників. Показники включають кількість майбутніх прийомів, загальну кількість пацієнтів, середній рейтинг лікаря та кількість відгуків. Інтерфейс модуля також містить останні відгуки від пацієнтів, що дозволяє лікарю отримати швидкий доступ до зворотного зв'язку.

Рейтинг лікаря є важливим інструментом для оцінки його професійної діяльності. Пацієнти, залишаючи свої відгуки, допомагають іншим потенційним пацієнтам робити обґрунтований вибір лікаря. Крім того, регулярний аналіз отриманих рейтингів та відгуків дозволяє лікарю виявляти слабкі сторони у своїй практиці та вживати необхідних заходів для їх усунення. Це може включати як покращення професійних навичок, так і підвищення якості спілкування з пацієнтами.

Блок-схема алгоритму обчислення рейтингу лікаря наведена на рисунку 3.7.

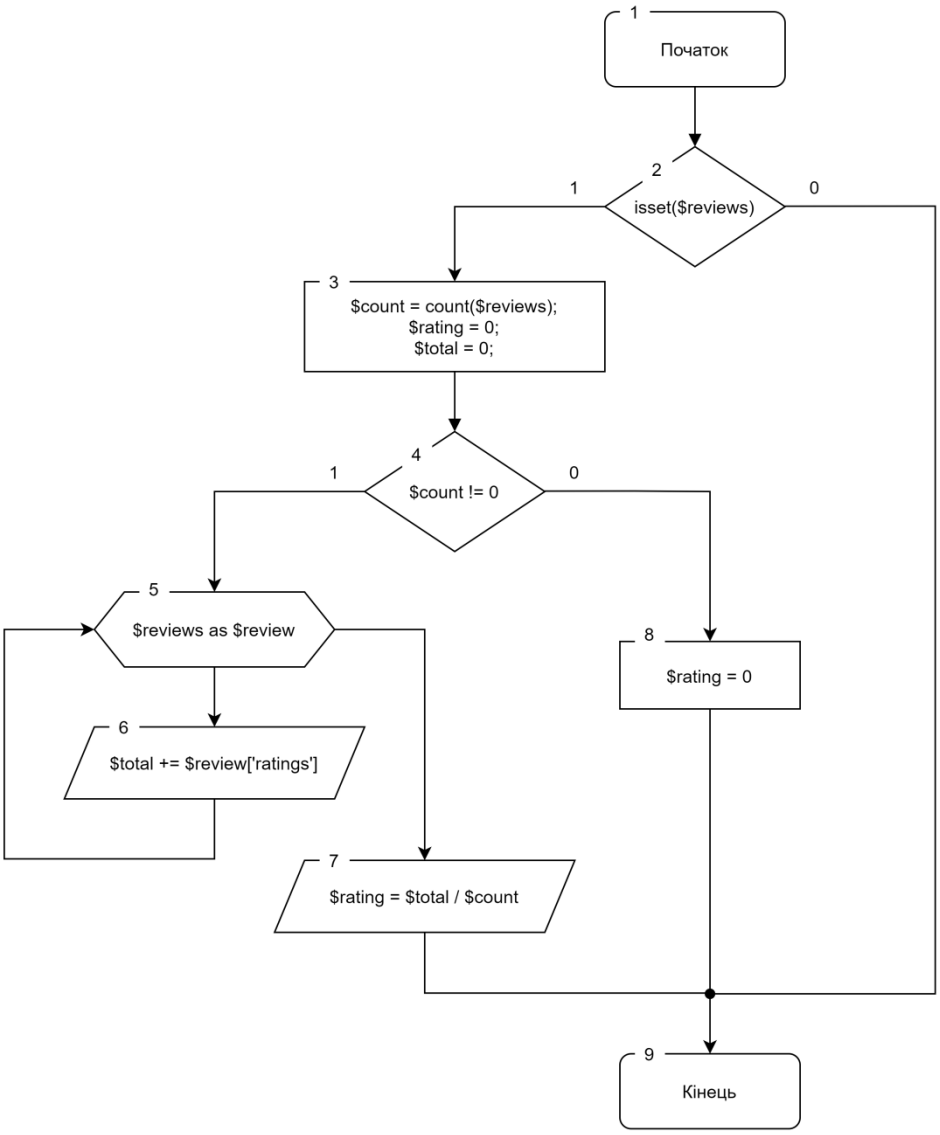


Рисунок 3.7 – Блок-схема алгоритму обчислення рейтингу лікаря

3.6 Розробка структури інтерфейсу програмного застосунку

Оскільки застосунок розробляється для мобільних пристроїв, які мають обмежену робочу область, важливо створити інтерфейс, який складатиметься з невеликих вікон, кожне з яких буде виконувати конкретну задачу. При цьому інтерфейс має забезпечувати комфортне користування програмою.

Таким чином, було прийнято рішення розділити функціонал програми на п’ять окремих вікон. Переміщення між цими вікнами буде здійснюватися за допомогою панелі навігації внизу екрану пристрою (рис. 3.8).

Рисунок 3.8 – Панель навігації застосунку

Головна сторінка призначена для надання користувачам зручного доступу до ключової інформації про їх медичні записи та лікарів. Вона дозволяє пацієнтам легко переглядати заплановані прийоми, управляти ними, а також переглядати категорії та популярних лікарів (рис. 3.9).

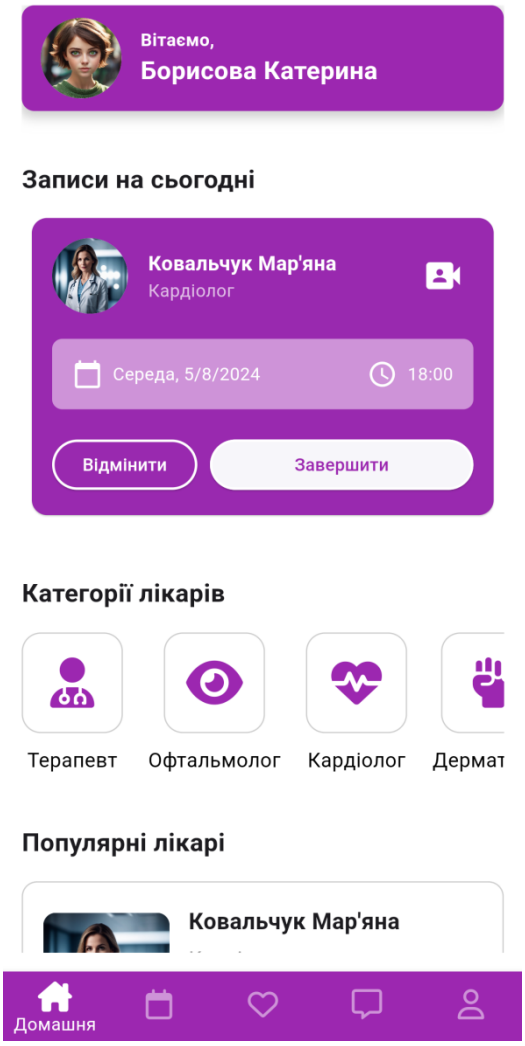


Рисунок 3.9 – Головна сторінка застосунку

Сторінка перегляду історії записів для управління записами до лікарів забезпечує користувачам зручний доступ до інформації про їхні майбутні, завершені та скасовані прийоми. Вона забезпечує можливість перегляду деталей кожного запису, включаючи ім'я лікаря, спеціальність, дату та час прийому, а також дозволяє керувати записами, зокрема, відміняти або переназначати їх (рис. 3.10).

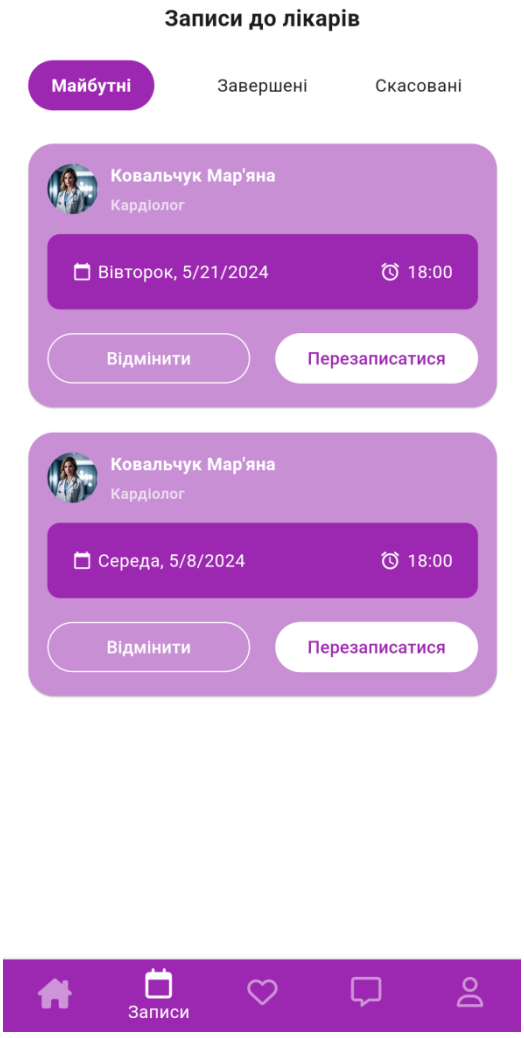


Рисунок 3.10 – Сторінка перегляду історії записів

Сторінка для перегляду списку улюблених лікарів призначена для надання користувачам швидкого доступу до улюблених лікарів, полегшуючи процес

запису на прийом. Тут зберігаються профілі лікарів, з якими користувачі хотіли б підтримувати постійний контакт (рис. 3.11).

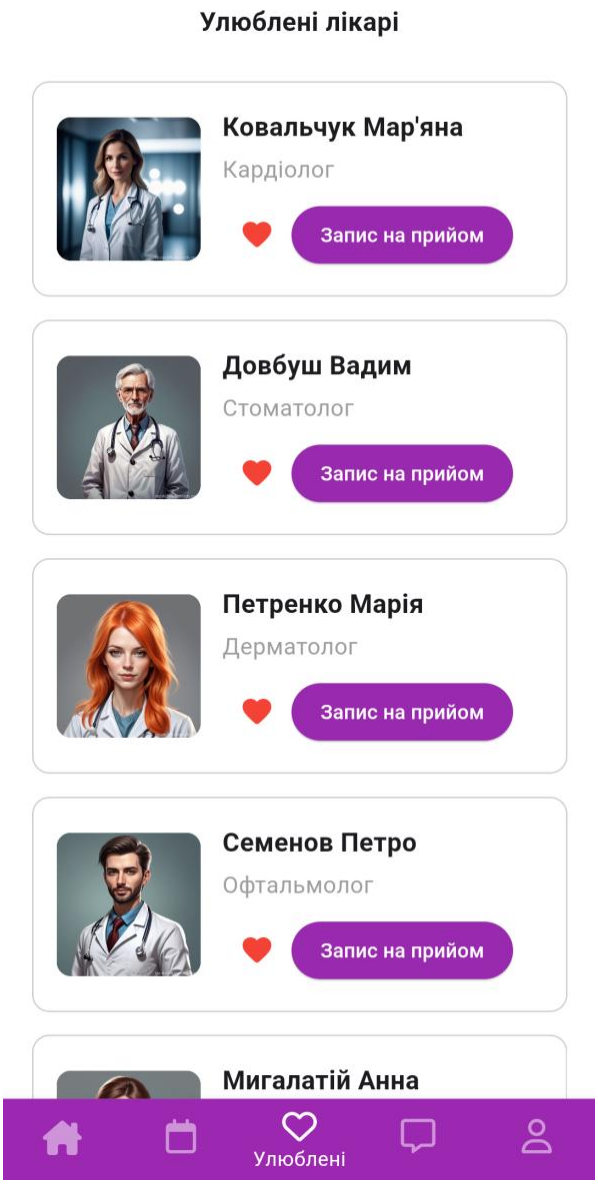


Рисунок 3.11 – Сторінка перегляду списку улюблених лікарів

Сторінка чатів із медичними спеціалістами забезпечує користувачам зручний спосіб отримати доступ до повідомлень та вести бесіди з лікарями. Вона дозволяє швидко переглядати та відповідати на повідомлення від медичних

фахівців, підтримуючи постійний зв'язок між пацієнтами та лікарями для обговорення стану здоров'я, надання рекомендацій та нагадувань (рис 3.12).

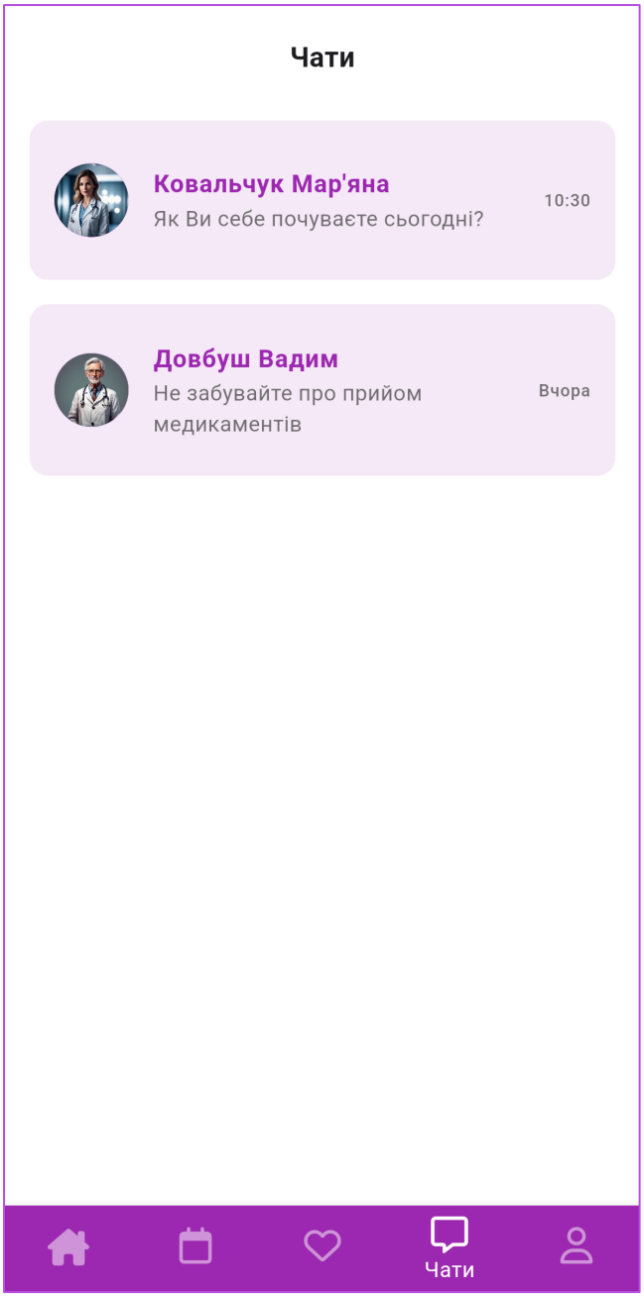


Рисунок 3.12 – Сторінка з чатами з медичними спеціалістами

Сторінка профілю користувача призначена для надання користувачам повного контролю над їх обліковим записом і особистою інформацією в

застосунку. Вона забезпечує доступ до перегляду персональних даних, перегляду історії записів до лікарів та виходу з облікового запису (рис. 3.13).

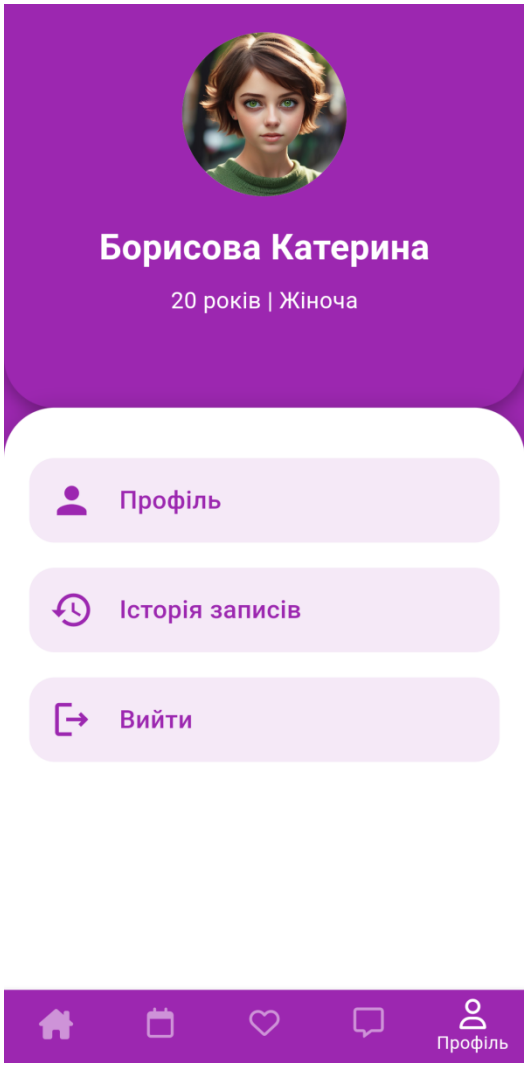


Рисунок 3.13 – Сторінка профілю користувача

3.7 Висновки

У третьому розділі було здійснено аналіз та вибір засобів для реалізації мобільного застосунку, зокрема мови програмування та середовища розробки. Було обґрунтовано вибір мови Dart у поєднанні з фреймворком Flutter для створення застосунку. Visual Studio Code було обрано як інтегроване середовище

розробки завдяки його гнучкості, розширюваності та підтримці великої кількості мов програмування.

Також у цьому розділі було розроблено графічні схеми, блок-схеми та діаграми класів для кількох ключових модулів застосунка, таких як модулі авторизації та реєстрації користувачів, перегляду списку лікарів, здійснення відеодзвінків до лікаря та модуль статистики для лікаря. Кожен модуль було детально описано, включаючи їхні призначення, функціональні можливості та реалізацію. Структура інтерфейсу застосунка, розділеного на п'ять окремих сторінок, забезпечує зручність та ефективність у використанні, дозволяючи користувачам легко керувати своїми медичними записами та підтримувати зв'язок з лікарями.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Аналіз методів тестування програмного забезпечення

Тестування програмного забезпечення є невід'ємною частиною процесу розробки, що забезпечує якість, надійність та відповідність кінцевого продукту вимогам користувачів. Воно включає різні підходи та методи, які допомагають виявляти та виправляти помилки на різних етапах життєвого циклу програмного забезпечення. Основні цілі тестування полягають у забезпеченні коректної роботи програми, підвищенні її продуктивності та зменшенні ризику виникнення критичних помилок у процесі її використання.

Методи тестування програмного забезпечення можна класифікувати за різними критеріями, такими як рівень тестування та підходи до проведення тестування. За рівнями тестування виділяють модульне тестування, яке передбачає тестування окремих модулів або компонентів системи ізольовано один від одного; інтеграційне тестування, яке включає тестування взаємодії між інтегрованими модулями; системне тестування, що передбачає тестування всієї системи як єдиного цілого для перевірки її відповідності вимогам; та приймальне тестування, яке проводиться користувачами або замовниками для підтвердження того, що система відповідає їхнім очікуванням та вимогам [20].

Статичні методи тестування включають огляд коду, що передбачає рецензування коду іншими розробниками для виявлення помилок і недоліків; статичний аналіз, який використовує спеціальні інструменти для аналізу коду без його виконання; та інспекції та перевірки, що є формальними процесами перевірки коду, проектної документації та інших артефактів.

Динамічні методи тестування включають метод "білої скриньки" (white-box testing), яке передбачає тестування внутрішньої структури або роботи програми, коли тестувальник має доступ до вихідного коду; метод "чорної скриньки" (black-

box testing), яке передбачає тестування функціональності програми без знання її внутрішньої структури, і найкраще підходить для оцінки системи з точки зору кінцевого користувача, забезпечуючи перевірку відповідності вимогам та очікуванням; та метод "сірої скриньки" (gray-box testing), яке є комбінацією двох попередніх методів тестування, де тестувальник має деяке знання про внутрішню структуру системи [21].

Тестування програмного забезпечення стикається з багатьма викликами, серед яких виявлення прихованих помилок, які можуть проявитися лише у специфічних умовах та витрати часу та ресурсів, адже тестування може бути дуже трудомістким та вимагати значних ресурсів. Для вирішення цих проблем можуть бути використані стратегії, такі як розробка стратегій тестування, які включають різноманітні методи тестування для забезпечення максимального покриття; використання автоматизованих інструментів, що допомагають зменшити час і витрати, а також підвищити ефективність процесу.

Аналіз методів тестування програмного забезпечення демонструє, що ефективне тестування є критично важливим для забезпечення якості та надійності програмних продуктів. Вибір відповідних методів, таких як метод "чорної скриньки", дозволяє оптимізувати процеси тестування та виявляти більшість помилок до того, як продукт потрапить до кінцевого користувача.

4.2 Тестування розробленого програмного застосунку

Тестування мобільного застосунку на платформі Android включає перевірку його функціональності, безпеки та правильності виконання на пристроях з цією операційною системою.

Процес починається з інсталяції застосунку на мобільний пристрій, що працює на Android. Після цього здійснюється ряд функціональних тестів, які

перевіряють правильність обробки та виведення даних застосунком та його відповідність вимогам бізнес-логіки.

Одним із ключових моментів є тестування екрану реєстрації користувачів (рис 4.1). Якщо під час реєстрації пропускається введення обов'язкових даних, наприклад імені, система має вивести повідомлення про помилку та запросити користувача ввести необхідні дані. Результати такої перевірки продемонстровані на рисунку 4.2.

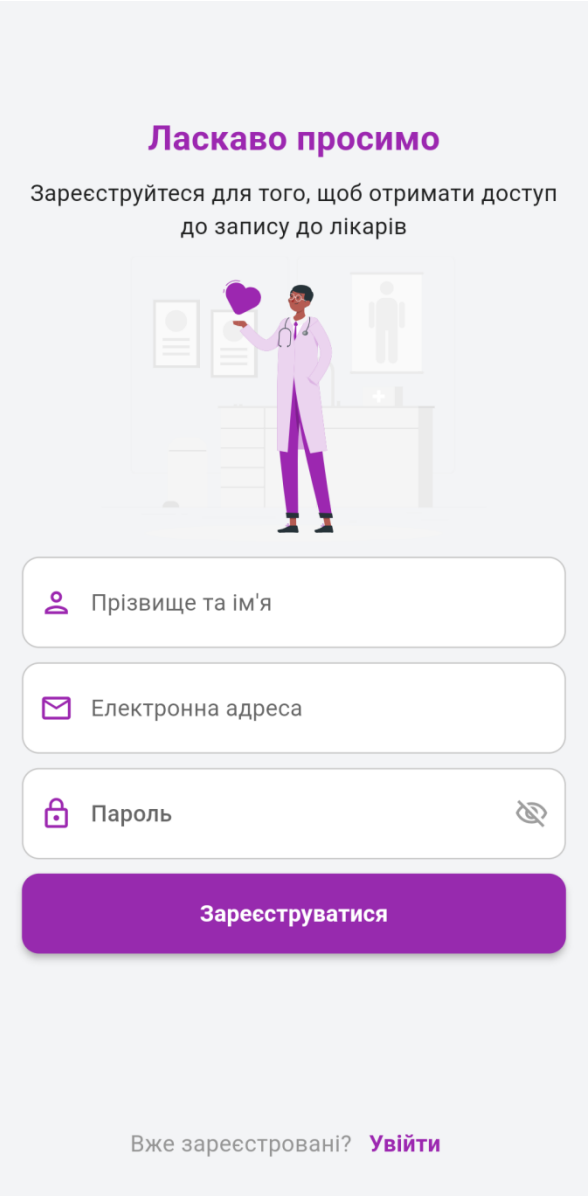


Рисунок 4.1 – Сторінка реєстрації пацієнта

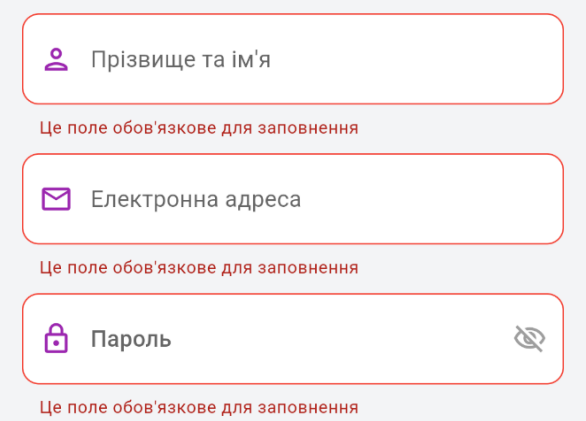


Рисунок 4.2 – Результат пропуску полів для введення даних при реєстрації

Далі можливе проведення тестування на правильність обробки даних, введених у поле пароля. Наприклад, система повинна показувати помилку, якщо формат пароля є неправильним або він не відповідає встановленим критеріям безпеки. Результати цього тесту можуть бути представлені на рисунку 4.3.

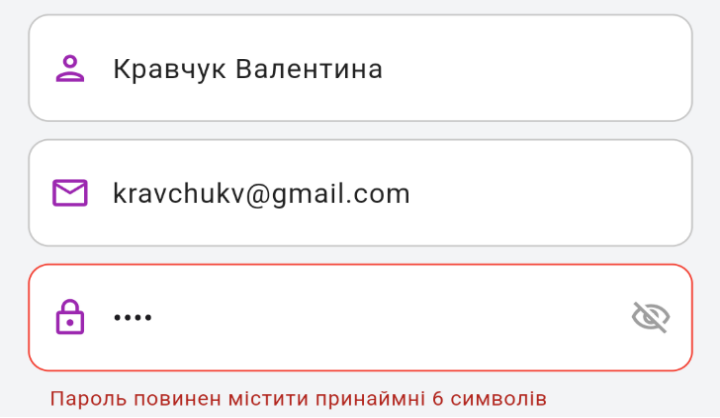
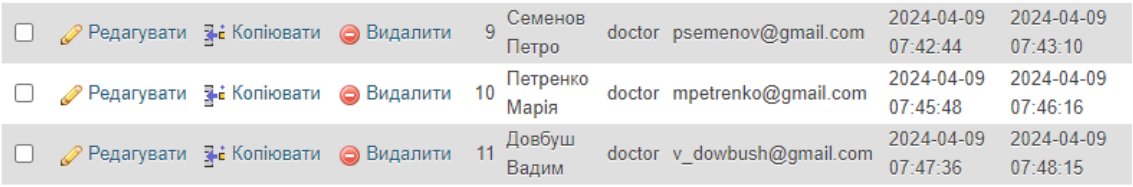


Рисунок 4.3 – Повідомлення про замалу кількість символів у паролі

Для того, щоб перевірити, чи дійсно реєстрація відбувається, потрібно упевнитися, що дані користувача зберігаються правильно. Це включає перевірку коректного збереження таких даних, як ім'я користувача, електронна пошта та пароль, в базі даних. На рисунку 4.4 наведений елемент бази даних, де

зберігаються дані для авторизації користувачів. Крім цього, потрібно впевнитися, що під час реєстрації відповідні записи з'являються в базі даних без затримок.



☐	Редагувати	Копіювати	Видалити	9	Семенов Петро	doctor	psemenov@gmail.com	2024-04-09 07:42:44	2024-04-09 07:43:10
☐	Редагувати	Копіювати	Видалити	10	Петренко Марія	doctor	mpetrenko@gmail.com	2024-04-09 07:45:48	2024-04-09 07:46:16
☐	Редагувати	Копіювати	Видалити	11	Довбуш Вадим	doctor	v_dowbush@gmail.com	2024-04-09 07:47:36	2024-04-09 07:48:15

Рисунок 4.4 – База даних до реєстрації нового користувача

Наступним кроком є введення коректних даних (рис. 4.5) та натискання кнопки «Зареєструватися».

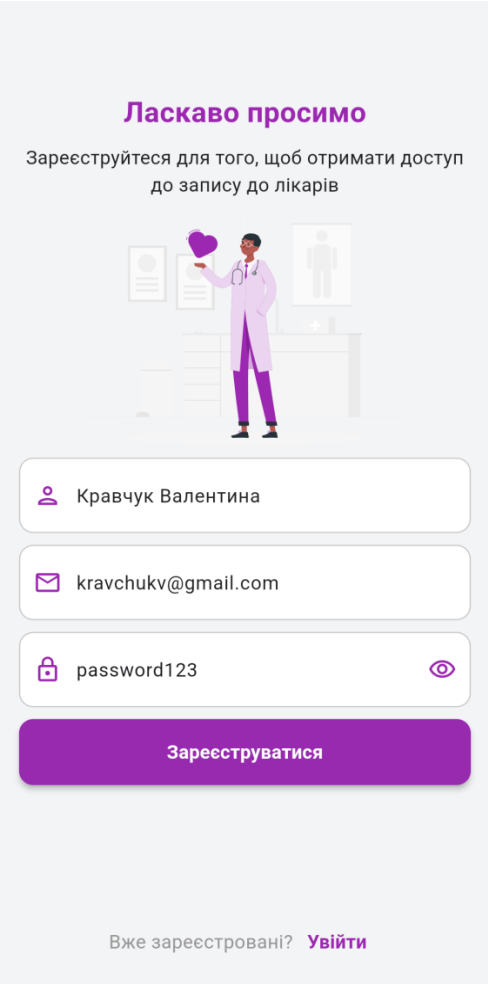


Рисунок 4.5 – Коректно введені дані при реєстрації

У результаті успішної реєстрації, дані користувача записуються до бази даних (рис 4.6), після чого пацієнт автоматично перенаправляється на головну сторінку застосунку (рис 4.7).

☐	Редагувати	Копіювати	Видалити	9	Семенов Петро	doctor	psemenov@gmail.com	2024-04-09 07:42:44	2024-04-09 07:43:10
☐	Редагувати	Копіювати	Видалити	10	Петренко Марія	doctor	mpetrenko@gmail.com	2024-04-09 07:45:48	2024-04-09 07:46:16
☐	Редагувати	Копіювати	Видалити	11	Довбуш Вадим	doctor	v_dowbush@gmail.com	2024-04-09 07:47:36	2024-04-09 07:48:15
☐	Редагувати	Копіювати	Видалити	32	Кравчук Валентина	user	kravchukv@gmail.com	2024-05-08 10:42:03	2024-05-08 10:42:03

Рисунок 4.6 – База даних після реєстрації нового користувача

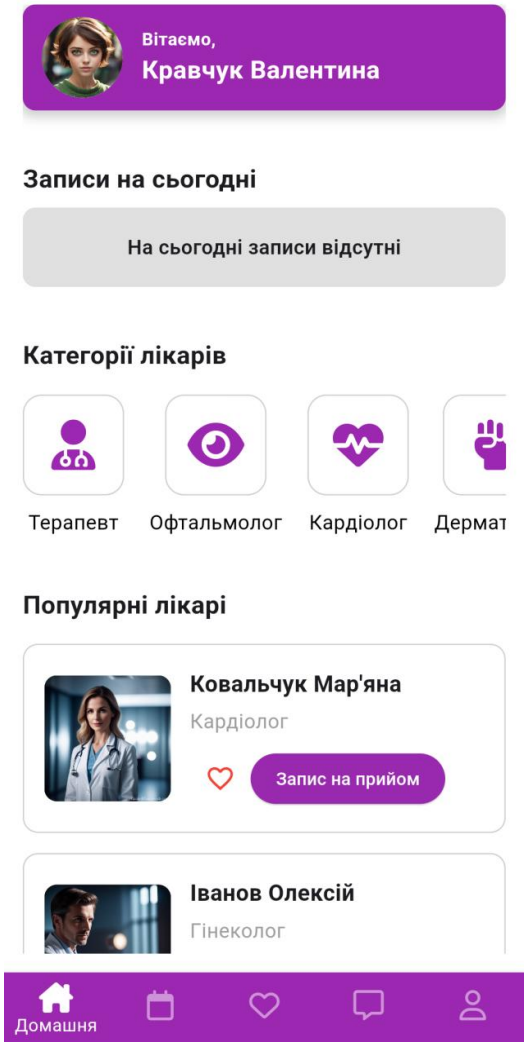


Рисунок 4.7 – Головна сторінка застосунку нового користувача

Також необхідно перевірити коректність роботи процесу авторизації (рис. 4.8). Перш за все, слід переконатися, що всі поля форми авторизації працюють коректно, включаючи введення логіну та пароля. Необхідно протестувати обробку правильних та неправильних даних, щоб переконатися, що система належним чином розпізнає і обробляє кожен випадок. Також варто перевірити, чи коректно працюють повідомлення про помилки. Крім цього, важливо переконатися, що після успішної авторизації користувач перенаправляється на відповідну сторінку застосунку.

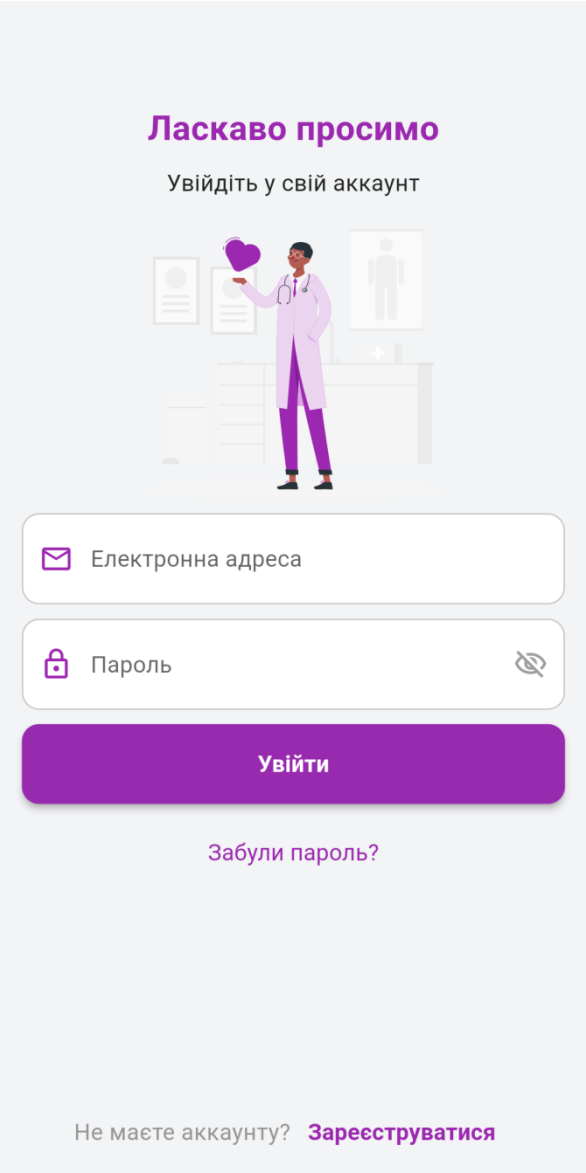


Рисунок 4.8 – Сторінка авторизації користувача

Наступним кроком перевірки буде введення даних у відповідні поля (рис. 4.9). Важливо також перевірити правильність введеного паролю та електронної пошти.

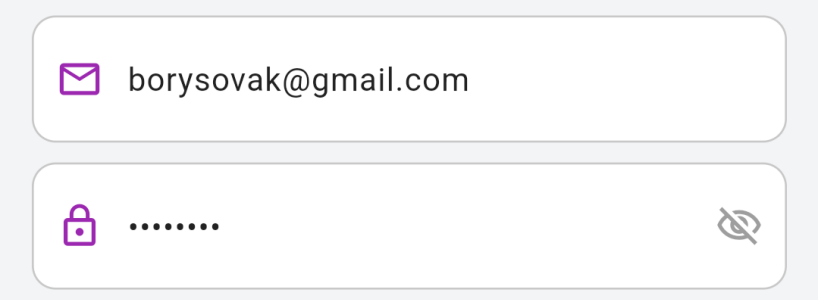


Рисунок 4.9 – Заповнені поля форми

Якщо дані були введені неправильно, при натисненні кнопки «Увійти» з’явиться банер з повідомленням про помилку (рис. 4.10).

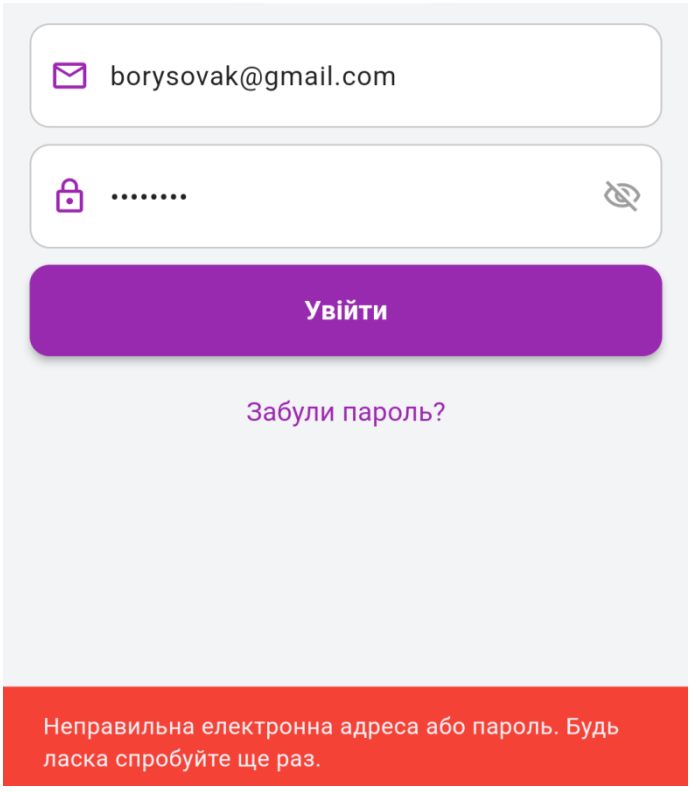


Рисунок 4.10 – Банер з повідомленням про помилку

У разі виправлення користувачем помилок і повторного натискання кнопки «Увійти», завантажиться головна сторінка застосунку (рис. 3.9).

На головному екрані мобільного застосунку користувача вітає персоналізований вітальний банер, де відображається його ім’я. Нижче розташована інформація про найближчі записи, що забезпечує користувача зручним доступом до актуальних даних про прийоми.

Категорії лікарів представлені через іконки, що дозволяє швидко переглянути різні спеціальності. Розділ “Популярні лікарі” містить карточки спеціалістів із зазначенням їхньої спеціальності та фотографією, що робить пошук особистого лікаря інтуїтивно зрозумілим та зручним. Кнопка “Запис на прийом” в кожній карточці лікаря спрощує процедуру бронювання консультації.

Функціонал перегляду записів до лікарів в застосунку (рис. 3.10) дозволяє переглянути список запланованих зустрічей з різними медичними спеціалістами. Цей екран мобільного застосунку для запису на прийом до лікарів показує майбутні візити користувача. Зверху екрану розміщені категорії для сортування візитів: "Майбутні", "Завершені" та "Скасовані". Ці категорії дозволяють користувачам легко знаходити потрібну інформацію про свої медичні прийоми, що сприяє зручності та ефективності використання застосунку.

Кожен запис містить ім'я лікаря, його спеціальність, дату і час прийому, а також дії, які може виконати користувач, зокрема, можливість скасувати запис ("Відмінити") або перенести зустріч ("Перезаписатися").

В розділі "Завершені" видно записи, які були успішно завершені в минулому. Користувач може запланувати новий прийом, натиснувши кнопку "Перезаписатися" (рис. 4.11). Завершені записи зберігаються в історії, що дозволяє користувачам відстежувати свої медичні консультації та за необхідності звертатися до попередніх візитів.

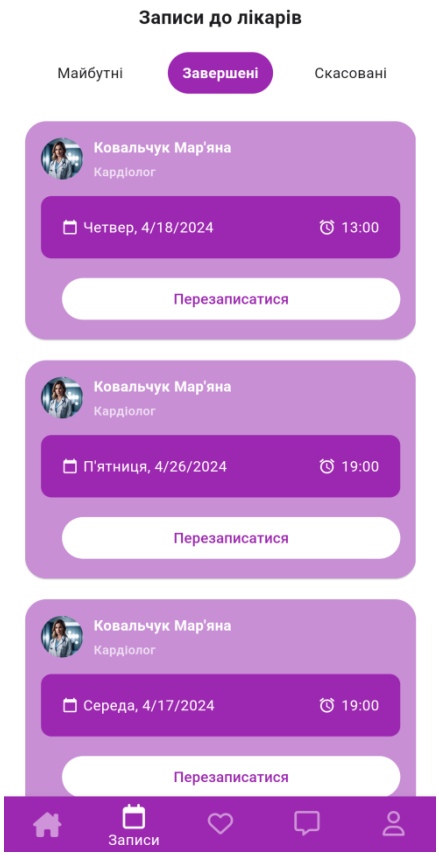


Рисунок 4.11 – Сторінка записів (вкладка «Завершені»)

В розділі «Скасовані» відображаються скасовані записи до лікарів, або їх відсутність (рис. 4.12).

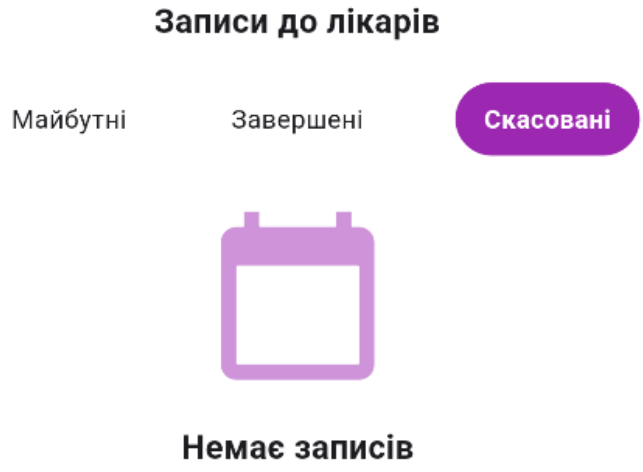


Рисунок 4.12 – Сторінка записів (вкладка «Скасовані»)

На сторінці «Улюблені» (рис. 3.11) розташовуються картки лікарів з фотографією, ім'ям, спеціальністю та можливістю записатися на прийом. Кожна картка містить кнопку для вибору лікаря як улюбленого, що підкреслюється сердечком, та кнопку для запису на прийом.

На наступній сторінці (рис. 3.12) зображено інтерфейс сторінки "Чати". У верхній частині екрана розташований заголовок "Чати". Під заголовком відображено список чатів з лікарями, кожен з яких містить фотографію лікаря, його ім'я, коротке повідомлення та час або дату останнього повідомлення.

Сторінка чату демонструє особистий чат із лікарем (рис. 4.13). У верхній частині екрану розташований заголовок з ім'ям лікаря. Ліворуч від заголовка знаходиться кнопка назад, яка позначена стрілкою. У нижній частині екрану знаходиться поле для введення тексту та кнопка для надсилання повідомлення.

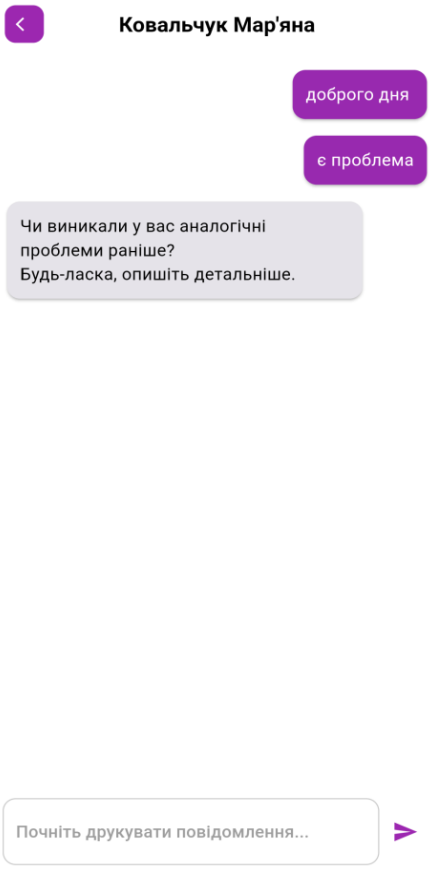


Рисунок 4.13 – Екран особистого чату з лікарем

Наступний екран демонструє зручну та інтуїтивно зрозумілу можливість комунікації з медичними фахівцями (рис. 4.14).

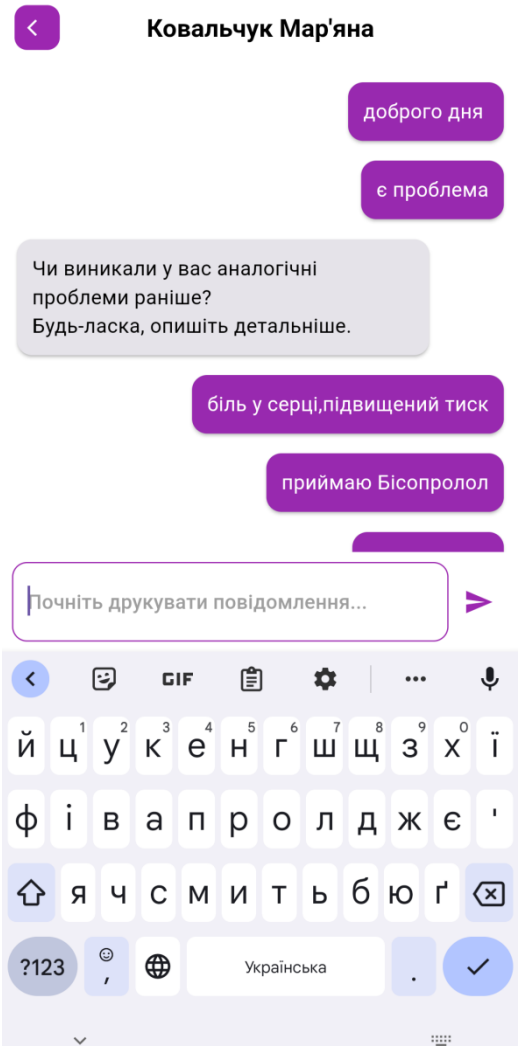


Рисунок 4.14 – Спілкування з лікарем у чаті

На екрані (рис. 4.15) зображено інтерфейс розділу "Про лікаря" мобільного застосунку. У верхній частині екрану розташований заголовок з назвою розділу. Ліворуч від заголовка знаходиться кнопка назад, позначена стрілкою, а праворуч - іконка у вигляді серця для додавання лікаря до улюблених.

Під заголовком розміщено фотографію лікаря. Нижче фотографії міститься інформація про лікаря в кількох фіолетових блоках, які містять дані про досвід,

кількість пацієнтів та оцінку лікаря. Далі розташований текстовий блок, де наведено опис лікаря та його кваліфікації. У нижній частині екрану знаходиться велика фіолетова кнопка, яка дозволяє користувачам записатися на прийом до лікаря.

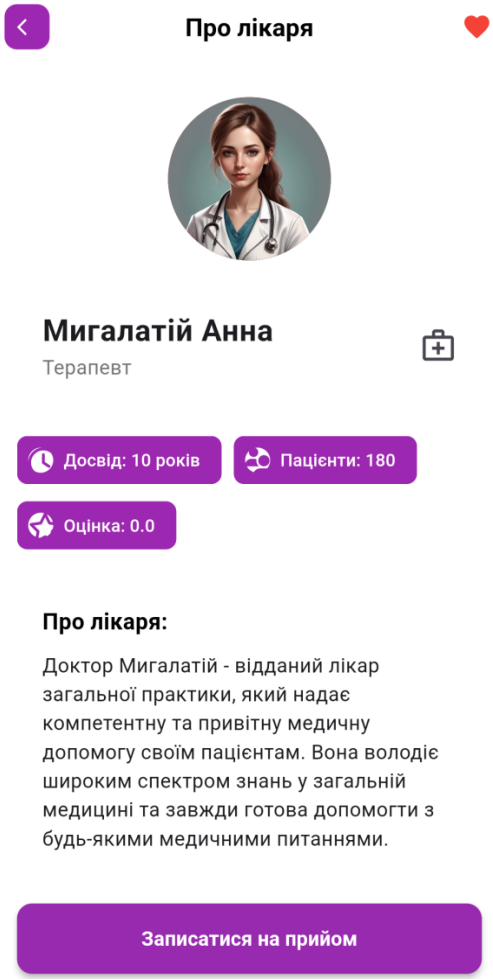


Рисунок 4.15 – Сторінка перегляду інформації про лікаря

Наступний екран (рис. 4.16) демонструє інтерфейс сторінки "Запис на прийом". У верхній частині екрану розташований заголовок, ліворуч від заголовка знаходиться кнопка назад, позначена стрілкою.

Під заголовком розташований календар, де користувач може вибрати дату для прийому. Вибрана дата підсвічена фіолетовим кольором. Поруч з календарем є стрілки для перемикання між місяцями.

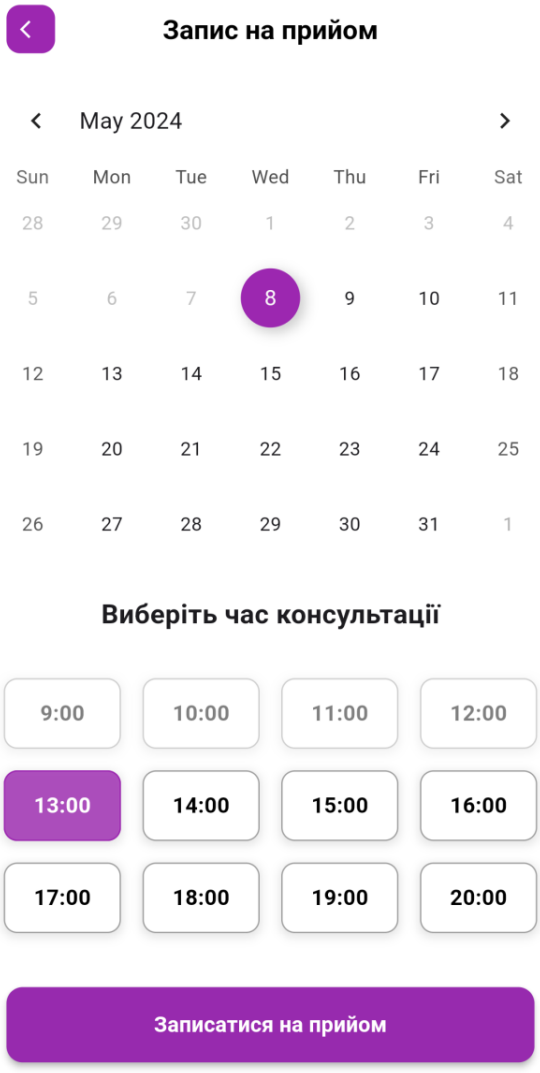


Рисунок 4.16 – Запис до лікаря

Нижче календаря знаходиться розділ з вибором часу консультації. Представлено декілька варіантів часу, які відображаються у вигляді кнопок, що дозволяють користувачу вибрати зручний час для прийому. Недоступні для

прийому години відображаються за допомогою світло-сірого кольору і не можуть бути обрані.

У нижній частині екрану знаходиться велика фіолетова кнопка "Записатися на прийом", яка дозволяє користувачам підтвердити вибір дати та часу для запису на прийом до лікаря.

Якщо обрана дата недоступна для запису, з'являється повідомлення про недоступність запису на цю дату та пропозиція вибрати іншу дату (рис. 4.17). У такому випадку кнопка для підтвердження запису неактивна.

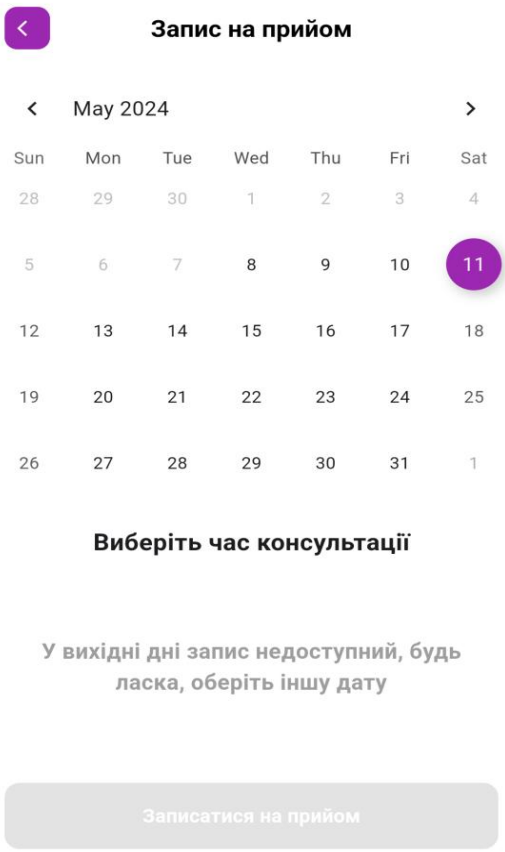


Рисунок 4.17 – Повідомлення про те, що запис на цю дату недоступний

Екран підтвердження запису (рис. 4.18) підтверджує, що запис на прийом відбувся успішно. Великий знак галочки символізує завершення дії, а напис "Запис успішно здійснено!" повідомляє про вдале здійснення запису.

Рисунок 4.18 – Повідомлення про успішне здійснення запису

Програмний застосунок підтримує функцію відеодзвінка до лікаря. Щоб здійснити відеодзвінок, необхідно перейти на головну сторінку та натиснути кнопку у вигляді відеокамери на картці з найближчим прийомом (рис. 4.19).

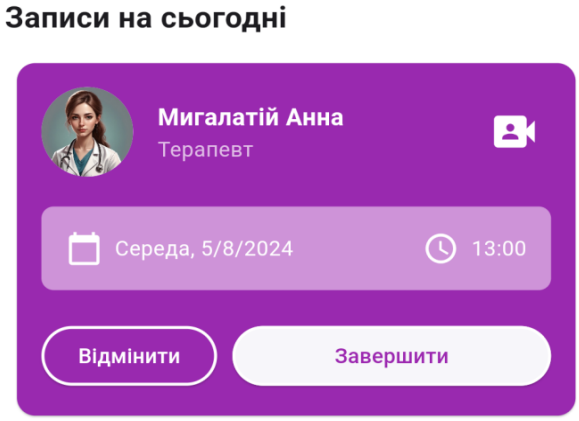


Рисунок 4.19 – Картка з найближчим прийомом

Після натискання цієї кнопки відкривається екран відеодзвінка (рис. 4.20).

Рисунок 4.20 – Екран відеодзвінку з лікарем

Користувач може увімкнути та вимкнути мікрофон та камеру (рис. 4.21), а також покинути дзвінок. У верхньому лівому кутку відображається ім’я лікаря, а у правому – тривалість дзвінка.

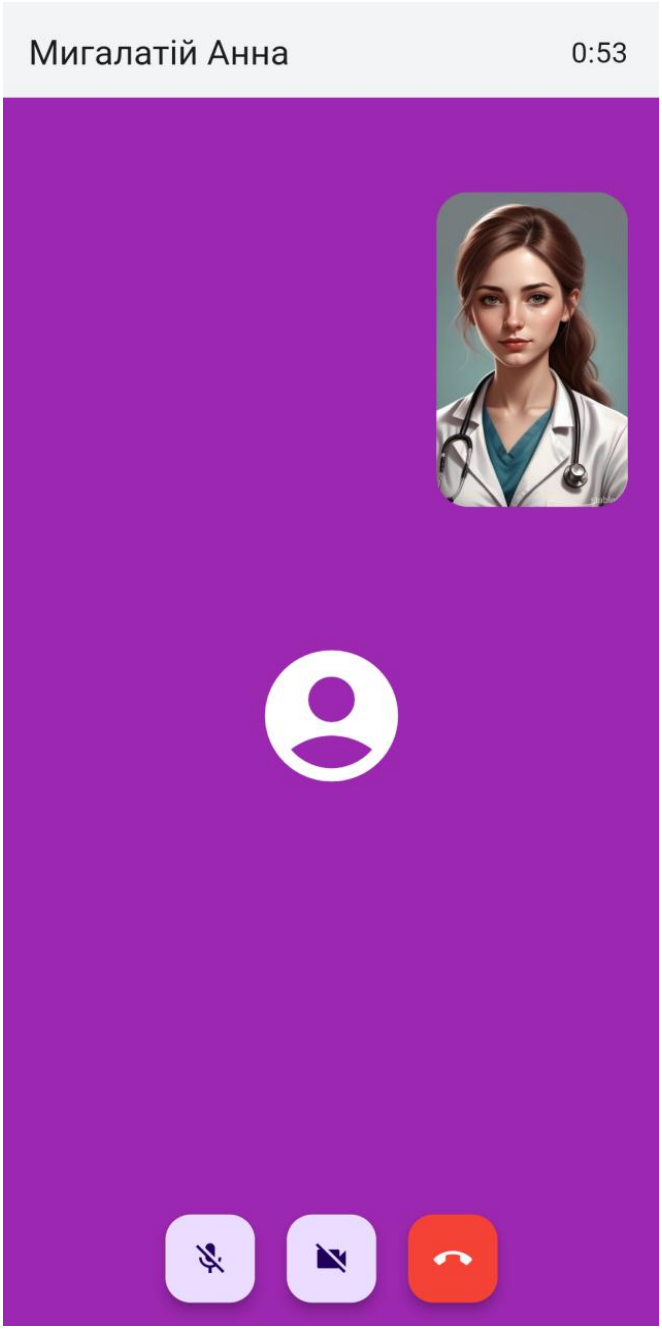


Рисунок 4.21 – Вимкнені камера та мікрофон у користувача

Якщо користувач бажає завершити дзвінок, він може натиснути червону кнопку, що виконує цю функцію. Після натискання кнопки з'явиться вікно з підтвердженням, де користувач повинен підтвердити своє бажання покинути дзвінок (рис. 4.22).

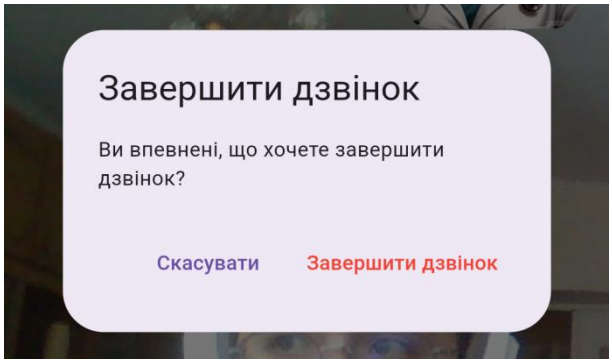


Рисунок 4.22 – Діалогове вікно підтвердження завершення дзвінка

У разі завершення дзвінка наступне діалогове вікно запропонує залишити відгук про прийом, оцінивши його від 1 до 5 та додавши коментар (рис. 4.23). Ця функція дозволяє збирати зворотний зв'язок, що допомагає покращити якість послуг. Після відправлення відгуку користувач автоматично повертається на головну сторінку застосунку.

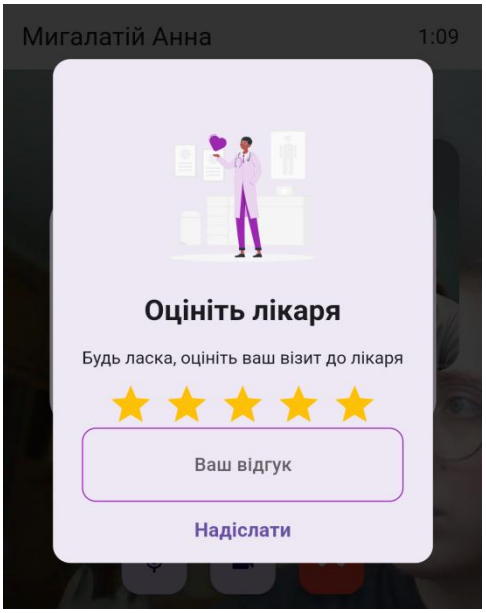
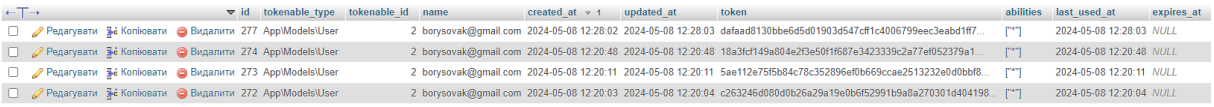


Рисунок 4.23 – Діалогове вікно залишення відгуку про лікаря

Екран профілю користувача відображає персональний профіль користувача у медичному застосунку (рис 3.13). Вище на екрані розташований аватар та

особисті дані користувача. Нижче знаходяться три кнопки: "Профіль" для доступу до особистих налаштувань, "Історія записів" для перегляду минулих та майбутніх візитів, та "Вийти" для виходу з аккаунта.

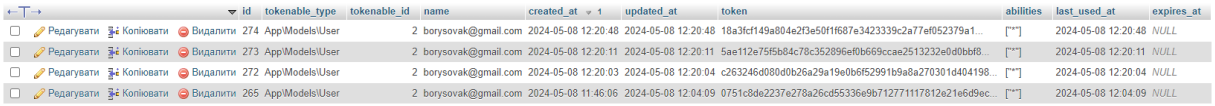
Кожного разу під час входу в систему у таблиці бази даних створюється запис про розпочату сесію. Цей запис містить інформацію про користувача, час входу та інші важливі дані (рис. 4.24).



	id	tokenable_type	tokenable_id	name	created_at	updated_at	token	abilities	last_used_at	expires_at
Редагувати Копіювати Видалити	277	App\Models\User	2	borysovak@gmail.com	2024-05-08 12:28:02	2024-05-08 12:28:03	dafaad8130bbe6d5d01903d547cff1c4006799eec3eabd1ff7...	["*"]	2024-05-08 12:28:03	NULL
Редагувати Копіювати Видалити	274	App\Models\User	2	borysovak@gmail.com	2024-05-08 12:20:48	2024-05-08 12:20:48	18a3fcf149a804e2f3e50f1f687e3423339c2a77ef052379a1...	["*"]	2024-05-08 12:20:48	NULL
Редагувати Копіювати Видалити	273	App\Models\User	2	borysovak@gmail.com	2024-05-08 12:20:11	2024-05-08 12:20:11	5ae112e75f5b84c78c352896ef0b669ccae2513232e0d0bbf8...	["*"]	2024-05-08 12:20:11	NULL
Редагувати Копіювати Видалити	272	App\Models\User	2	borysovak@gmail.com	2024-05-08 12:20:03	2024-05-08 12:20:04	c263246d080d0b26a29a19e0b6f52991b9a8a270301d404198...	["*"]	2024-05-08 12:20:04	NULL

Рисунок 4.24 – Таблиця з сесіями до виходу з профілю

При виході з профілю (шляхом натиснення кнопки «Вийти» на сторінці профілю користувача) запис про сесію видаляється з таблиці, щоб забезпечити безпеку та актуальність інформації (рис. 4.25).



	id	tokenable_type	tokenable_id	name	created_at	updated_at	token	abilities	last_used_at	expires_at
Редагувати Копіювати Видалити	274	App\Models\User	2	borysovak@gmail.com	2024-05-08 12:20:48	2024-05-08 12:20:48	18a3fcf149a804e2f3e50f1f687e3423339c2a77ef052379a1...	["*"]	2024-05-08 12:20:48	NULL
Редагувати Копіювати Видалити	273	App\Models\User	2	borysovak@gmail.com	2024-05-08 12:20:11	2024-05-08 12:20:11	5ae112e75f5b84c78c352896ef0b669ccae2513232e0d0bbf8...	["*"]	2024-05-08 12:20:11	NULL
Редагувати Копіювати Видалити	272	App\Models\User	2	borysovak@gmail.com	2024-05-08 12:20:03	2024-05-08 12:20:04	c263246d080d0b26a29a19e0b6f52991b9a8a270301d404198...	["*"]	2024-05-08 12:20:04	NULL
Редагувати Копіювати Видалити	265	App\Models\User	2	borysovak@gmail.com	2024-05-08 11:46:06	2024-05-08 12:04:09	0751c8de2237e278a26cd55336e9b712771117812e21e6d9ec...	["*"]	2024-05-08 12:04:09	NULL

Рисунок 4.25 – Таблиця з сесіями після виходу з профілю

Після виходу користувач автоматично перенаправляється на сторінку авторизації. Такий механізм управління сесіями допомагає підтримувати безпеку користувачів і знижує ризики несанкціонованого доступу.

4.3 Тестування веб-клієнта для лікарів

Окрім мобільного застосунку для пацієнтів, також було розроблено веб-клієнт для лікарів. Цей веб-клієнт надає лікарям можливість реєструватися у системі, щоб розпочати прийом пацієнтів. Після реєстрації лікарі можуть керувати

своїм профілем, додавати інформацію про спеціалізацію та досвід. Крім того, веб-клієнт дозволяє лікарям переглядати відгуки пацієнтів про надані послуги. Лікарі можуть отримувати зворотний зв'язок, оцінки та коментарі, що допомагає їм покращувати якість обслуговування.

Такий інструмент робить взаємодію з пацієнтами зручнішою та ефективнішою, сприяючи підвищенню рівня довіри та задоволеності пацієнтів.

У даному веб-клієнті лікарі мають можливість створити новий профіль (рис. 4.26) або увійти до вже існуючого (рис. 4.27).

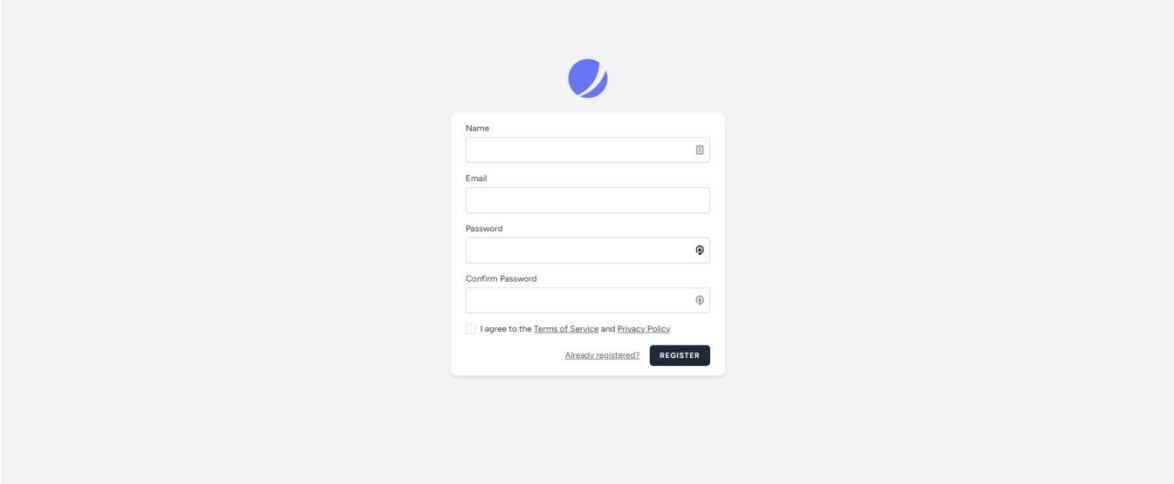


Рисунок 4.26 – Форма реєстрації профілю лікаря

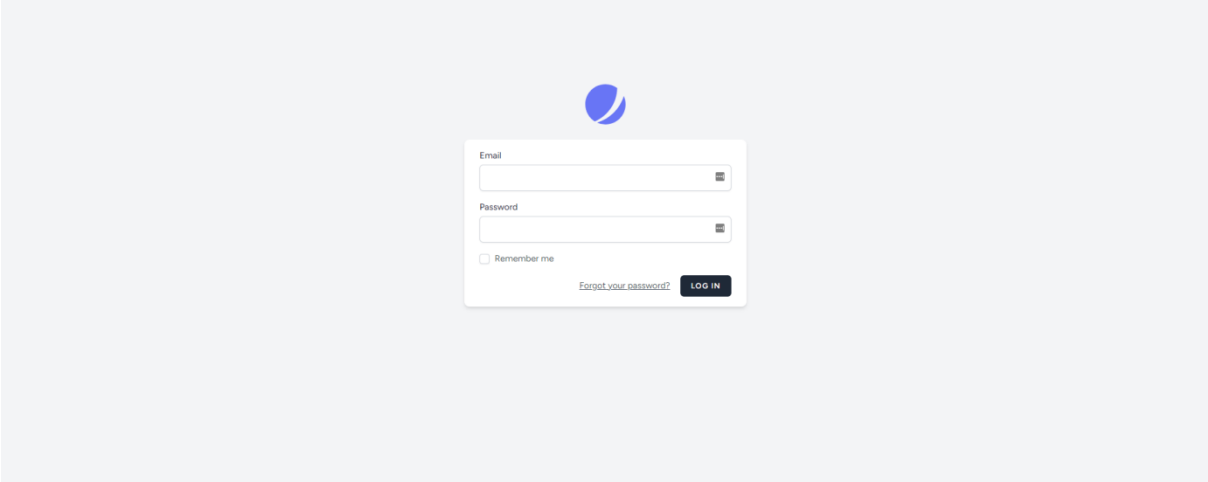


Рисунок 4.27 – Форма логіну у профіль лікаря

Для реєстрації лікар повинен ввести свої ім’я та прізвище, електронну адресу та пароль, який потрібно підтвердити (рис. 4.28). Також для реєстрації необхідно погодитись з правилами та умовами використання сервісу.

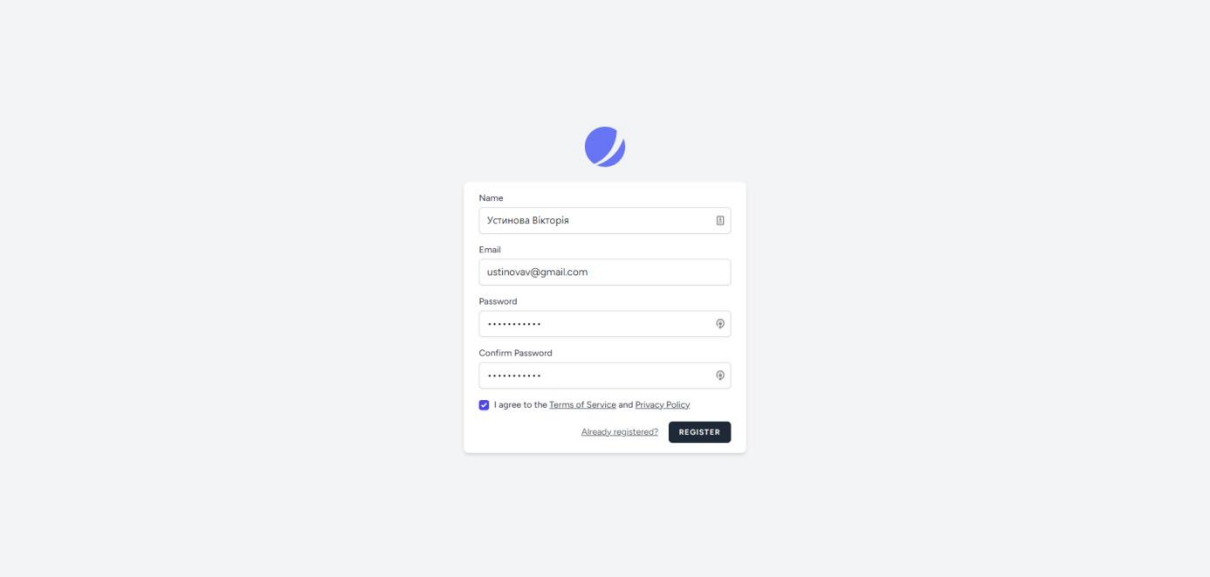


Рисунок 4.28 – Заповнена форма реєстрації лікаря

Після реєстрації користувач переходить на сторінку з панеллю зі статистикою про себе (рис. 4.29). У верхній частині екрана розташоване меню з логотипом та вкладкою "Dashboard". Під заголовком "Панель керування" відображаються чотири основні інформаційні блоки, значення яких у щойно зареєстрованого лікаря дорівнюватимуть нулю. Перший блок показує кількість запланованих прийомів, другий блок відображає кількість прийнятих пацієнтів, третій блок інформує про рейтинг лікаря, четвертий блок показує кількість відгуків від пацієнтів.

Нижче розташований блок "Останні відгуки", де відображаються найновіші відгуки пацієнтів. У даний момент відгуків немає, про що свідчить повідомлення "Ще немає відгуків!". Фон екрану має світло-фіолетовий колір, а інформаційні блоки виділені фіолетовим кольором для кращої видимості.

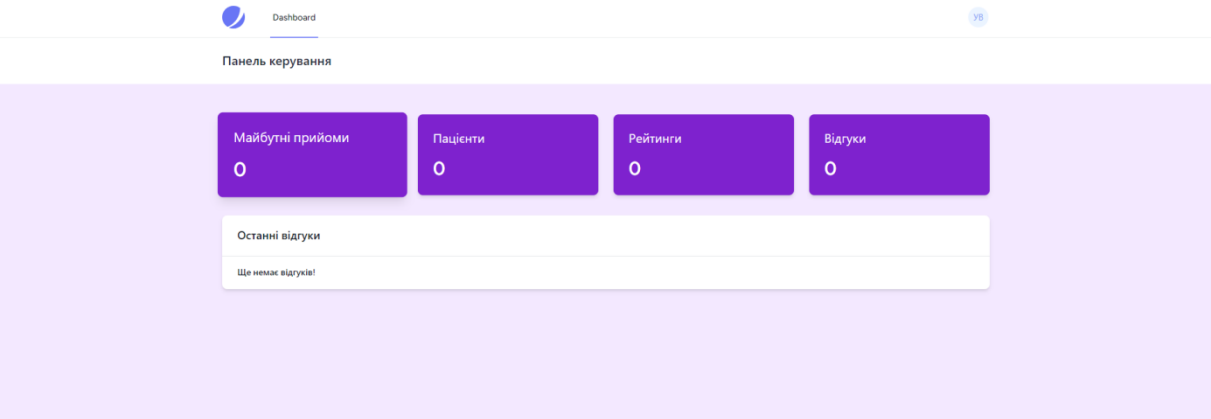


Рисунок 4.29 – Панель керування лікаря

Лікар також може відредагувати свій профіль (рис. 4.30), перейшовши до налаштувань шляхом натиснення на іконку з зображенням фото користувача.

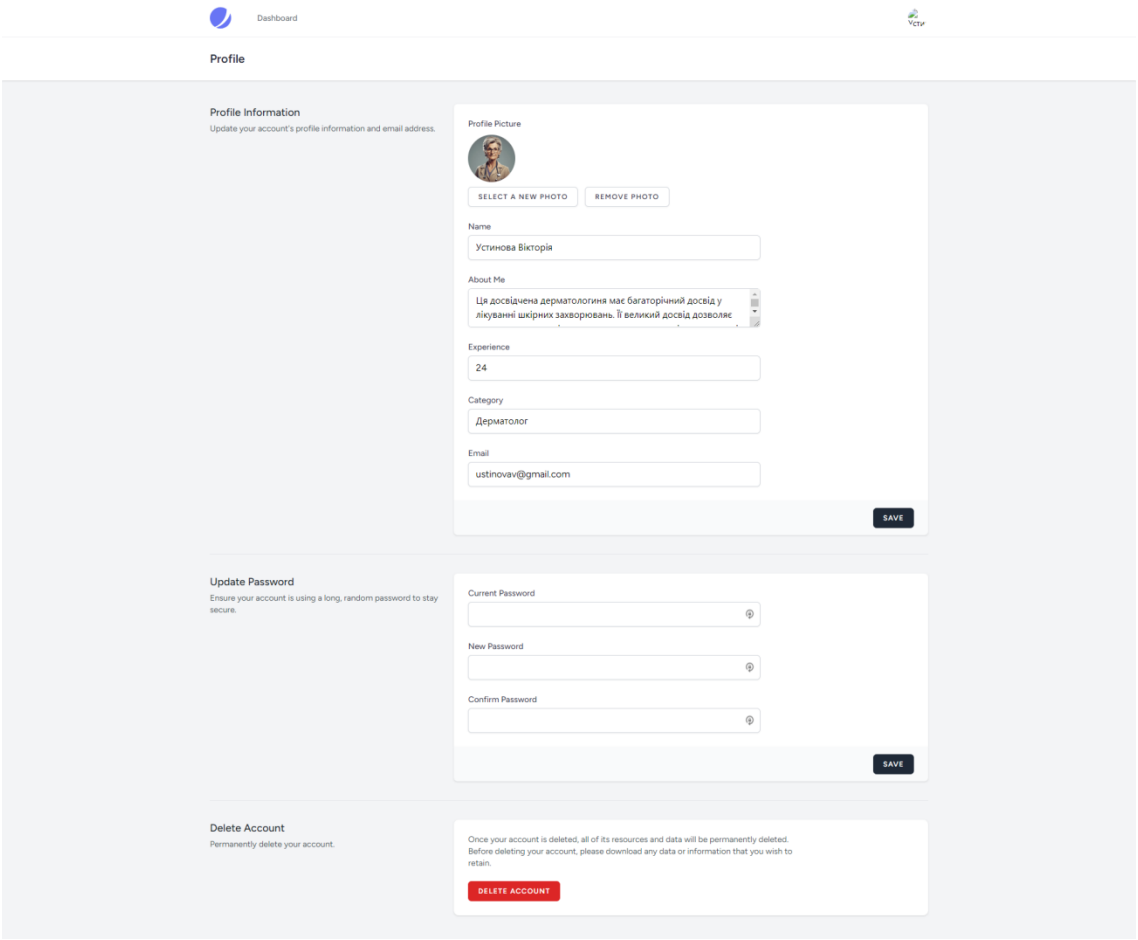


Рисунок 4.30 – Відредагований профіль лікаря

Усі зміни, які лікар вносить до свого профілю, автоматично відображаються в застосунку для пацієнтів (рис. 4.31).

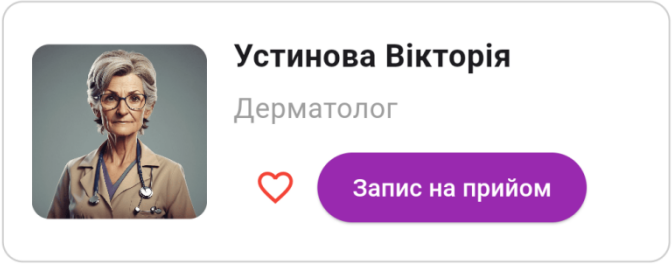


Рисунок 4.31 – Новий доданий лікар у застосунку

Це включає оновлення особистої інформації, додавання нових спеціалізацій або послуг (рис. 4.32).

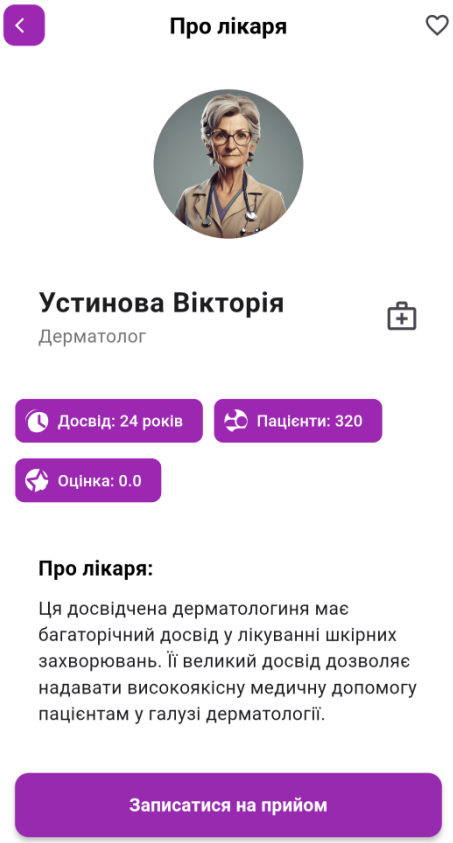


Рисунок 4.32 – Оновлена інформація про лікаря

Такий механізм забезпечує актуальність інформації, що надається пацієнтам, і сприяє зручності користування застосунком.

4.4 Розробка інструкції користувача мобільного застосунку

Для встановлення застосунку необхідно завантажити .apk файл, який є стандартним форматом для інсталяції програмного забезпечення на Android. Користувачі можуть знайти такі файли в офіційних магазинах застосунків, на сторонніх веб-сайтах або безпосередньо на веб-сайті розробника.

Перш ніж почати інсталяцію, важливо перевірити, чи відповідає пристрій системним вимогам застосунку, щоб уникнути проблем під час його використання. Інформація про мінімальні та рекомендовані системні вимоги для коректної роботи застосунку наведена в таблицях 4.1 та 4.2.

Після успішної установки застосунок буде готовий до використання.

Таблиця 4.1 – Мінімальна конфігурація:

Тип процесора	MediaTek Helio P90
Об’єм оперативної пам’яті	2 ГБ для 64-разрядної системи
Місце на жорсткому диску	1 ГБ
Операційна система	Android 11

Таблиця 4.2 – Рекомендована конфігурація:

Тип процесора	.4GHz Octa-core Qualcomm Snapdragon 888
Об’єм оперативної пам’яті	4 ГБ для 64-разрядної системи
Місце на жорсткому диску	4 ГБ
Операційна система	Android 14

Перед доступом до функціоналу система перевіряє, чи користувач авторизований. Якщо авторизація не пройдена, користувачу пропонується увійти в систему. Без входу в обліковий запис користувач не може записатися на прийом або залишити відгук про лікаря. Після авторизації користувач має можливість записатися на прийом, переглянути графік роботи лікаря та вибрати доступний час для візиту.

Успішна авторизація направляє користувача на головний екран, де він може переглянути перелік лікарів і їх спеціальності, а також вибрати лікаря за інтересом. На сторінці обраного лікаря користувач може ознайомитися з його профілем і відгуками інших пацієнтів.

Зареєстровані користувачі також можуть оцінювати якість послуг, залишаючи свої відгуки в застосунку. Це створює ефективну систему зворотного зв'язку, що допомагає іншим зробити обґрунтований вибір медичного спеціаліста.

4.4 Висновки

У четвертому розділі дипломної роботи було здійснено детальне тестування розробленого методу та програмних засобів для забезпечення комунікацій в системах телемедицини.

Крім того, розроблено детальну інструкцію для користувачів, яка описує процеси встановлення та налаштування програмного забезпечення на мобільних пристроях. Інструкція також включає кроки реєстрації в системі, методи взаємодії з медичними фахівцями через застосунок.

Результати тестування підтвердили, що розроблений метод та програмні засоби відповідають встановленим критеріям ефективності та безпеки для систем телемедицини. Таким чином, проєкт демонструє значний потенціал для покращення якості медичного обслуговування та забезпечення зручності для пацієнтів і лікарів через використання сучасних технологій комунікації.

ВИСНОВКИ

У результаті виконання бакалаврської дипломної роботи було розроблено мобільну систему для комунікацій у сфері телемедицини, яка включає такі функціональні модулі:

– реєстрація облікового запису для пацієнта, можливість запису на прийом до лікаря з обранням потрібного спеціаліста та часу, відстеження статусу записів, можливість переглядати історію записів до лікарів, додавання медичних фахівців у список улюблених для швидкого доступу до них;

– функціонал для консультації з лікарем онлайн: текстовий чат та онлайн-прийом з використанням веб-камери пристрою користувачів;

– розробка дешборду для перегляду лікарем статистики про його роботу.

Для реалізації цієї системи було розроблено метод, модель, алгоритми її роботи. Використано мову програмування Dart у поєднанні з фреймворком Flutter; для бекенд-частини використано PHP та Laravel, а також середовище розробки Visual Studio Code. Робота оформлена з урахуванням рекомендацій [22].

У першому розділі було проведено аналіз сучасних мобільних систем для комунікацій у сфері телемедицини. Було виявлено їх недоліки, що стало основою для розробки власного застосунку. Другий розділ охоплював аналіз даних, розробку методу і моделі роботи системи, а також побудову алгоритмів роботи застосунку та його окремих модулів. У третьому розділі було обґрунтовано вибір засобів для реалізації програмного забезпечення, проведено варіантний аналіз та розроблено програмні модулі й графічний інтерфейс користувача. Четвертий розділ описує тестування розробленої системи, яке підтвердило її працездатність і відповідність очікуванням. Крім того, визначено мінімальні та рекомендовані вимоги до пристроїв для використання застосунку, а також розроблено інструкцію зі встановлення програмного забезпечення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Цифрова трансформація в охороні здоров'я: аналіз технологічних трендів [Електронний ресурс] – Режим доступу до ресурсу: https://yur-gazeta.com/publications/practice/informaciyne-pravo-telekomunikaciyi/cifrova-transformaciya-v-ohoroni-zdorovya-analiz-tehnologichnih-trendiv.html

2. The five big trends in healthcare technology for 2024 [Електронний ресурс] – Режим доступу до ресурсу: https://tediselmedical.com/en/the-five-big-trends-in-healthcare-technology-for-2024/

3. Войтко В.В. Розробка автоматизованої мобільної системи запису відвідувачів до лікаря / В.В. Войтко, К. О. Борисова / Автоматизація та комп’ютерно-інтегровані технології у виробництві та освіті: стан, досягнення, перспективи розвитку: матеріали Всеукраїнської науково-практичної Internet-конференції. – Черкаси, 2024. - 384 с. – С. 67-69

4. Цифровізація охорони здоров’я у 2024 році – які сервіси будуть впроваджені [Електронний ресурс] – Режим доступу до ресурсу: https://www.kmu.gov.ua/news/tsyfrovizatsiia-okhorony-zdorovia-u-2024-rotsi-iaki-servisy-budut-vprovadzheni

5. EasyVisit [Електронний ресурс] – Режим доступу до ресурсу: https://easyvisit.com.au/

6. Docton [Електронний ресурс] – Режим доступу до ресурсу: https://www.docton.in/

7. Health24 [Електронний ресурс] – Режим доступу до ресурсу: https://h24.ua/

8. Windmill, E. (2020). Flutter in Action. Сполучені Штати Америки: Manning.

9. ListView class. [Електронний ресурс] – Режим доступу до ресурсу: https://api.flutter.dev/flutter/widgets/ListView-class.html

10. Що таке діаграма класів [Електронний ресурс] – Режим доступу до ресурсу: https://www.mindonmap.com/uk/blog/what-is-uml-class-diagram/

11. All You Need to Know about State Diagrams [Електронний ресурс] – Режим доступу до ресурсу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/about-state-diagrams/

12. Sande, J. (2022). Dart Apprentice: Fundamentals (First Edition): Modern Cross-Platform Programming with Dart (n.p.): Kodeco Incorporated.

13. Mueller, J. P. (2022). C# 10.0 All-in-One For Dummies. Велика Британія: Wiley.

14. Swift.org - Welcome to Swift.org [Електронний ресурс] – Режим доступу до ресурсу: https://www.swift.org/

15. What Is Python Used For? A Beginner’s Guide [Електронний ресурс] – Режим доступу до ресурсу: https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python

16. Visual Studio Code - Code Editing. Redefined Guide [Електронний ресурс] – Режим доступу до ресурсу: https://code.visualstudio.com/

17. Eclipse [Електронний ресурс] – Режим доступу до ресурсу: https://www.eclipse.org/

18. Android Studio: переваги та особливості [Електронний ресурс] – Режим доступу до ресурсу: https://qagroup.com.ua/publications/android-studio-perevagy-ta-osoblyvosti/

19. What is Visual Studio? [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/uk-ua/visualstudio/get-started/visual-studio-ide

20. Рівні тестування [Електронний ресурс] – Режим доступу до ресурсу: https://qalearning.com.ua/theory/lectures/material/testing-levels/

21. Learn Software Testing in 24 Hours: Definitive Guide to Learn Software Testing for Beginners. (2020). (n.p.): Guru99.

22. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів студентів спеціальності 121 "Інженерія програмного забезпечення" / Уклад. О. Н. Романюк, Г. О. Черноволик – Вінниця: ВНТУ, 2022. – 52с.

Додатки

79

Додаток А. Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
«_12_» __березня__ 2024 р.

Технічне завдання
на бакалаврську дипломну роботу
«Розробка методу та програмних засобів для комунікацій
в системах телемедицини»
за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:
к.т.н., доц. Войтко В. В.
«_12_» __березня__ 2024 р.

Виконала:
студентка гр. 4ПІ-20б Борисова К. О.
«_12_» __березня__ 2024 р.

Вінниця – 2024 року

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка методу та програмних засобів для комунікацій в системах телемедицини».

Галузь застосування – сфера охорони здоров’я: медичні заклади.

2. Підстава для розробки.

Підставою для виконання бакалаврської дипломної роботи (БДР) є індивідуальне завдання на БДР та наказ №80 від 11 березня 2024 року ректора по ВНТУ про закріплення тем БДР.

3. Мета та призначення розробки.

Метою дослідження є покращення комунікаційних можливостей у системах телемедицини за рахунок розробки ефективної системи, яка сприятиме підвищенню якості медичного обслуговування.

Призначення роботи – розробка мобільного застосунку для комунікацій в системах телемедицини.

4. Вихідні дані для проведення НДР

1. The five big trends in healthcare technology for 2024 [Електронний ресурс] – Режим доступу до ресурсу: https://tediselmedical.com/en/the-five-big-trends-in-healthcare-technology-for-2024/

2. Sande, J. (2022). Dart Apprentice: Fundamentals (First Edition): Modern Cross-Platform Programming with Dart (n.p.): Kodeco Incorporated.

3. Learn Software Testing in 24 Hours: Definitive Guide to Learn Software Testing for Beginners. (2020). (n.p.): Guru99.

5. Технічні вимоги

Комп’ютер з 64-розрядним (x64) процесором та тактовою частотою 2 ГГц. Об’єм оперативної пам’яті 8 ГБ, розмір жорсткого диску 256 ГБ. Операційна система Windows 10. З клієнтського боку: мобільний пристрій з ОС Android; ПК з будь-якою операційною системою та будь-який браузер.

6. Конструктивні вимоги.

Мобільний застосунок та веб-клієнт повинні відповідати ергономічним та технічним вимогам. Текстова та графічна документація повинна відповідати стандартам України.

7. Перелік технічної документації, що пред’являється по закінченню робіт:

- пояснювальна записка до бакалаврської дипломної роботи;
- технічне завдання;
- лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи
1	Аналіз стану розвитку програм для комунікацій в системах телемедицини та постановка задач розробки	14.03.24 – 21.03.24
2	Розробка методів, моделі та алгоритмів роботи програми	22.03.24 – 04.04.24
3	Розробка програмного забезпечення	05.04.24 – 03.05.24
4	Тестування програми	06.05.24 – 17.05.24
5	Оформлення матеріалів до захисту БДР	20.05.24 – 30.05.24

10. Порядок контролю та прийняття.

Всі етапи бакалаврської дипломної роботи контролюються науковим керівником згідно плану по виконанню роботи. Прийняття бакалаврської дипломної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком захисту.

83

Додаток Б. Протокол перевірки бакалаврської дипломної роботи на наявність текстових запозичень

ПРОТОКОЛ
ПЕРЕВІРКИ БАКАЛАВРСЬКОЇ ДИПЛОМНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: «Розробка методу та програмних засобів для комунікацій в системах телемедицини»

Тип роботи: БДР

Підрозділ: кафедра програмного забезпечення, ФІТКІ

Науковий керівник: Войтко В. В.

Оригінальність	97,9%
Схожість	2,1%

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Черноволик Г. О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unichek

Автор роботи _____ Борисова К. О.

Керівник роботи _____ Войтко В. В.

Додаток В. Лістинг програми

```
import 'package:flutter/material.dart';
import 'package:flutter_svg/svg.dart';
import 'package:easymedhelp_app/components/login_form.dart';
import 'package:easymedhelp_app/components/register_form.dart';
import 'package:easymedhelp_app/utils/configuration.dart';

class AuthScreen extends StatefulWidget {
  const AuthScreen({super.key});

  @override
  State<AuthScreen> createState() => _AuthScreenState();
}

class _AuthScreenState extends State<AuthScreen> {
  bool _isSignInMode = true;

  @override
  Widget build(BuildContext context) {
    _initConfiguration(context);

    return Scaffold(
      resizeToAvoidBottomInset: false,
      backgroundColor: EasyMedHelpConfiguration.backgroundColor,
      body: Padding(
        padding: EasyMedHelpConfiguration.paddingMedium,
        child: SafeArea(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            crossAxisAlignment: CrossAxisAlignment.center,
            children: <Widget>[
```

```
            const SizedBox(height: 64),
            _buildWelcomeText(),
            EasyMedHelpConfiguration.spacingS,
            _buildAuthDescription(),
            EasyMedHelpConfiguration.spacingS,
            _buildAuthIllustration(),
            EasyMedHelpConfiguration.spacingS,
            _isSignInMode ? const LoginForm() : const RegisterForm(),
            EasyMedHelpConfiguration.spacingS,
            _isSignInMode ? _buildForgotPasswordButton() : Container(),
            const Spacer(),
            _buildToggleAuthModeButton(),
          ],
        ),
      ),
    ),
  );
}

void _initConfiguration(BuildContext context) {
  EasyMedHelpConfiguration.init(context);
}

Widget _buildWelcomeText() {
  return Text(
    'Ласкаво просимо',
    style: EasyMedHelpConfiguration.textTheme.displayLarge,
  );
}

Widget _buildAuthDescription() {
  return Text(
```

```
    _isSignInMode
        ? 'Увійдіть у свій аккаунт'
        : 'Зареєструйтеся для того, щоб отримати доступ до запису до лікарів',
    style: EasyMedHelpConfiguration.textTheme.bodyLarge,
    textAlign: TextAlign.center,
  );
}

Widget _buildAuthIllustration() {
  return SvgPicture.asset(
    'assets/doctor_auth.svg',
    height: 200.0,
    fit: BoxFit.contain,
  );
}

Widget _buildForgotPasswordButton() {
  return Center(
    child: TextButton(
      onPressed: () {},
      child: Text(
        'Забули пароль?',
        style: EasyMedHelpConfiguration.textTheme.bodyLarge!.copyWith(
          color: EasyMedHelpConfiguration.primaryColor,
        ),
      ),
    ),
  );
}

Widget _buildToggleAuthModeButton() {
  return Row(
```

```
      mainAxisAlignment: MainAxisAlignment.center,
      children: <Widget>[
        Text(
          _isSignInMode ? 'Не маєте аккаунту?' : 'Вже зареєстровані?',
          style: EasyMedHelpConfiguration.textTheme.bodyLarge
              ?.copyWith(color: const Color.fromARGB(255, 150, 150, 150)),
        ),
        TextButton(
          onPressed: _toggleAuthMode,
          child: Text(
            _isSignInMode ? 'Зареєструватися' : 'Увійти',
            style: EasyMedHelpConfiguration.textTheme.bodyLarge!.copyWith(
              color: EasyMedHelpConfiguration.primaryColor,
              fontWeight: FontWeight.bold,
            ),
          ),
        )
      ],
    );
  }

  void _toggleAuthMode() {
    setState(() {
      _isSignInMode = !_isSignInMode;
    });
  }
}

import 'dart:convert';
import 'package:easymedhelp_app/components/button.dart';
import 'package:easymedhelp_app/main.dart';
import 'package:easymedhelp_app/models/auth_model.dart';
```

```
import 'package:easymedhelp_app/providers/dio_provider.dart';
import 'package:easymedhelp_app/utils/configuration.dart';
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'package:shared_preferences/shared_preferences.dart';

class LoginForm extends StatefulWidget {
  const LoginForm({super.key});

  @override
  State<LoginForm> createState() => _LoginFormState();
}

class _LoginFormState extends State<LoginForm> {
  final _formKey = GlobalKey<FormState>();
  final TextEditingController _emailController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();
  bool _isPasswordObscured = true; // Чи приховувати пароль
  @override
  Widget build(BuildContext context) {
    // Ініціалізуємо конфігурацію за допомогою контексту
    EasyMedHelpConfiguration.init(context);
    return Form(
      key: _formKey,
      child: Column(
        mainAxisAlignment: MainAxisAlignment.start,
        children: <Widget>[
          // Віджет для введення адреси електронної пошти
          _buildEmailTextField(),
          EasyMedHelpConfiguration.spacingS,
          // Віджет для введення пароля
          _buildPasswordTextField(),
```

```
        EasyMedHelpConfiguration.spacingS,
        // Кнопка для входу
        _buildLoginButton(),
      ],
    ),
  );
}
// Метод для створення поля вводу електронної пошти
Widget _buildEmailTextField() {
  return TextFormField(
    controller: _emailController,
    keyboardType: TextInputType.emailAddress,
    cursorColor: EasyMedHelpConfiguration.primaryColor,
    decoration: const InputDecoration(
      filled: true,
      fillColor: Colors.white,
      border: EasyMedHelpConfiguration.outlinedBorder,
      hintText: 'Електронна адреса',
      prefixIcon: Icon(Icons.email_outlined,
          color: EasyMedHelpConfiguration.primaryColor),
    ),
    style: EasyMedHelpConfiguration.textTheme.bodyLarge,
    validator: (value) {
      if (value == null || value.isEmpty) {
        return 'Будь ласка, введіть адресу електронної пошти';
      }
      if (!RegExp(r"^[a-zA-Z0-9.]+@[a-zA-Z0-9]+\.[a-zA-Z]+")
          .hasMatch(value)) {
        return 'Будь ласка, введіть дійсну адресу електронної пошти';
      }
      return null;
    },
```

```
  );
}
// Метод для створення поля введення паролю
Widget _buildPasswordTextField() {
  return TextFormField(
    controller: _passwordController,
    keyboardType: TextInputType.visiblePassword,
    cursorColor: EasyMedHelpConfiguration.primaryColor,
    obscureText: _isPasswordObscured,
    decoration: InputDecoration(
      filled: true,
      fillColor: Colors.white,
      border: EasyMedHelpConfiguration.outlinedBorder,
      hintText: 'Пароль',
      prefixIcon: const Icon(Icons.lock_outline,
          color: EasyMedHelpConfiguration.primaryColor),
      suffixIcon: IconButton(
        onPressed: () {
          setState(() {
            _isPasswordObscured = !_isPasswordObscured;
          });
        },
        icon: _isPasswordObscured
            ? const Icon(Icons.visibility_off_outlined, color: Colors.black38)
            : const Icon(Icons.visibility_outlined,
                color: EasyMedHelpConfiguration.primaryColor),
      ),
    ),
    style: EasyMedHelpConfiguration.textTheme.bodyLarge,
    validator: (value) {
      if (value == null || value.isEmpty) {
        return 'Будь ласка, введіть свій пароль';
```

```
      }
      if (value.length < 6) {
        return 'Пароль має бути не менше 6 символів';
      }
      return null;
    },
  );
}
// Метод для створення кнопки входу
Widget _buildLoginButton() {
  return Consumer<AuthModel>(
    builder: (context, auth, child) {
      return Button(
        width: double.infinity,
        title: 'Увійти',
        onPressed: () async {
          if (_formKey.currentState!.validate()) {
            final bool token = await DioProvider().getToken(
              _emailController.text,
              _passwordController.text,
            );
            if (token) {
              _handleLogin(auth);
            } else {
              _showErrorSnackbar();
            }
          }
        },
        disabled: false,
      );
    },
  );
```

```
  }
  // Метод для обробки входу користувача
  void _handleLogin(AuthModel auth) async {
    // Отримуємо доступ до локальних налаштувань для отримання токена
    final SharedPreferences prefs = await SharedPreferences.getInstance();
    final String? token = prefs.getString('token');
    // Перевіряємо, чи токен не є порожнім або нульовим
    if (token != null && token.isNotEmpty) {
      // Отримуємо дані користувача за допомогою токена
      final getUserResult = await _getUserData(token);
      // Якщо дані користувача успішно отримано
      if (getUserResult != null) {
        // Обробляємо дані користувача
        _processUserData(auth, getUserResult);
      }
    }
  }
// Метод для отримання даних користувача за допомогою токена
  Future<Map<String, dynamic>?> _getUserData(String token) async {
    // Отримуємо дані користувача за допомогою DioProvider
    final getUserResult = await DioProvider().getUser(token);
    // Перевіряємо, чи отримано дані користувача
    if (getUserResult != null) {
      // Повертаємо дані у форматі Map<String, dynamic>
      return json.decode(getUserResult);
    } else {
      // Якщо дані не отримано, повертаємо null
      return null;
    }
  }
// Метод для обробки отриманих даних користувача
  void _processUserData(AuthModel auth, Map<String, dynamic>? userData) {
```

```
    // Якщо дані користувача не порожні
    if (userData != null) {
      // Шукаємо записи до лікаря, якщо вони є
      final appointment = _findAppointment(userData);
      // Передаємо дані користувача та записи моделі AuthModel
      auth.loginSuccess(userData, appointment);
      // Переходимо на головний екран додатка
      MyApp.navigatorKey.currentState!.pushNamed('main');
    }
  }
// Метод для пошуку призначення лікаря у отриманих даних користувача
  Map<String, dynamic> _findAppointment(Map<String, dynamic> userData) {
    Map<String, dynamic> appointment = {};
    // Шукаємо записи до лікаря у списку лікарів користувача
    for (var doctorData in userData['doctor']) {
      if (doctorData['appointments'] != null) {
        appointment = doctorData;
        break;
      }
    }
    // Повертаємо знайдене призначення
    return appointment;
  }

  // Метод для відображення сповіщення про помилку
  void _showErrorSnackbar() {
    ScaffoldMessenger.of(context).showSnackBar(const SnackBar(
      content: Text(
          "Неправильна електронна адреса або пароль. Будь ласка спробуйте ще раз."),
      backgroundColor: EasyMedHelpConfiguration.errorColor,
    ));
  }
```

```
}

import 'dart:convert';

import 'package:easymedhelp_app/components/button.dart';
import 'package:easymedhelp_app/models/auth_model.dart';
import 'package:easymedhelp_app/providers/dio_provider.dart';
import 'package:easymedhelp_app/utils/configuration.dart';
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'package:easymedhelp_app/main.dart';
import 'package:shared_preferences/shared_preferences.dart';

class RegisterForm extends StatefulWidget {
  const RegisterForm({super.key});
  @override
  State<RegisterForm> createState() => _RegisterFormState();
}

class _RegisterFormState extends State<RegisterForm> {
  final _formKey = GlobalKey<FormState>();
  final _nameController = TextEditingController();
  final _emailController = TextEditingController();
  final _passwordController = TextEditingController();
  bool obscurePassword = true;

  @override
  Widget build(BuildContext context) {
    return SingleChildScrollView(
      child: Form(
        key: _formKey,
        child: Column(
```

```
        mainAxisAlignment: MainAxisAlignment.start,
        children: <Widget>[
          _buildFormField(_nameController, TextInputType.text,
              'Прізвище та ім\'я', Icons.person_outlined),
          EasyMedHelpConfiguration.spacingS,
          _buildFormField(_emailController, TextInputType.emailAddress,
              'Електронна адреса', Icons.email_outlined),
          EasyMedHelpConfiguration.spacingS,
          _buildPasswordField(),
          EasyMedHelpConfiguration.spacingS,
          _buildRegisterButton(),
          const SizedBox(height: 16), // Add some space at the bottom
        ],
      ),
    ),
  );
}
Widget _buildFormField(TextEditingController controller,
    TextInputType keyboardType, String hintText, IconData icon) {
  return TextFormField(
    controller: controller,
    keyboardType: keyboardType,
    cursorColor: EasyMedHelpConfiguration.primaryColor,
    decoration: InputDecoration(
      filled: true,
      fillColor: Colors.white,
      border: EasyMedHelpConfiguration.outlinedBorder,
      hintText: hintText,
      prefixIcon: Icon(icon),
      prefixIconColor: EasyMedHelpConfiguration.primaryColor,
    ),
    style: EasyMedHelpConfiguration.textTheme.bodyLarge,
```

```
    validator: (value) {
      if (value == null || value.isEmpty) {
        return 'Це поле обов\'язкове для заповнення';
      }
      return null;
    },
  );
}
Widget _buildPasswordField() {
  return TextFormField(
    controller: _passwordController,
    keyboardType: TextInputType.visiblePassword,
    cursorColor: EasyMedHelpConfiguration.primaryColor,
    decoration: InputDecoration(
      filled: true,
      fillColor: Colors.white,
      border: EasyMedHelpConfiguration.outlinedBorder,
      hintText: 'Пароль',
      suffixIcon: IconButton(
        onPressed: () {
          setState(() {
            obscurePassword = !obscurePassword;
          });
        },
        icon: obscurePassword
            ? const Icon(
                Icons.visibility_off_outlined,
                color: Colors.black38,
              )
            : const Icon(
                Icons.visibility_outlined,
                color: EasyMedHelpConfiguration.primaryColor,
```

```
            ),
        ),
        prefixIcon: const Icon(Icons.lock_outline),
        prefixIconColor: EasyMedHelpConfiguration.primaryColor,
      ),
      obscureText: obscurePassword,
      validator: (value) {
        if (value == null || value.isEmpty) {
          return 'Це поле обов\'язкове для заповнення';
        }
        if (value.length < 6) {
          return 'Пароль повинен містити принаймні 6 символів';
        }
        return null;
      },
    );
  }
  Widget _buildRegisterButton() {
    return Consumer<AuthModel>(builder: (context, auth, child) {
      return Button(
        width: double.infinity,
        title: 'Зареєструватися',
        onPressed: () async {
          if (_formKey.currentState!.validate()) {
            final registrationResult = await DioProvider().registerUser(
                _nameController.text,
                _emailController.text,
                _passwordController.text);
            if (registrationResult != null) {
              final token = await DioProvider()
                  .getToken(_emailController.text, _passwordController.text);
              if (token != null) {
```

```
            final prefs = await SharedPreferences.getInstance();
            final tokenValue = prefs.getString('token');
            if (tokenValue != null && tokenValue.isNotEmpty) {
              final response = await DioProvider().getUser(tokenValue);
              if (response != null) {
                _handleLogin(response, auth);
              }
            }
          }
        }
      },
      disabled: false,
    );
  });
 }

 Future<void> _handleLogin(String response, AuthModel auth) async {
  setState(() {
    Map<String, dynamic> appointment = {};
    final user = json.decode(response);

    for (var doctorData in user['doctor']) {
      if (doctorData['appointments'] != null) {
        appointment = doctorData;
      }
    }
    auth.loginSuccess(user, appointment);
    MyApp.navigatorKey.currentState!.pushNamed('main');
  });
 }
}
```

```
import 'package:easymedhelp_app/main.dart';
import 'package:easymedhelp_app/models/auth_model.dart';
import 'package:easymedhelp_app/providers/dio_provider.dart';
import 'package:easymedhelp_app/utils/configuration.dart';
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'package:shared_preferences/shared_preferences.dart';

class AccountScreen extends StatefulWidget {
  const AccountScreen({super.key});

  @override
  State<AccountScreen> createState() => _AccountScreenState();
}

class _AccountScreenState extends State<AccountScreen> {
  Map<String, dynamic> user = {};

  @override
  Widget build(BuildContext context) {
    EasyMedHelpConfiguration.init(context);
    user = Provider.of<AuthModel>(context, listen: false).getUser;

    return Scaffold(
      backgroundColor: EasyMedHelpConfiguration
          .primaryColor, // Change to your desired background color
      body: Column(
        crossAxisAlignment: CrossAxisAlignment.stretch,
        children: [
          Expanded(
            flex: 4,
```

```
child: Container(
  decoration: BoxDecoration(
    color: EasyMedHelpConfiguration.primaryColor,
    borderRadius: const BorderRadius.only(
      bottomLeft: Radius.circular(40),
      bottomRight: Radius.circular(40),
    ),
    boxShadow: [
      BoxShadow(
        color: Colors.black.withOpacity(0.3),
        blurRadius: 10,
        spreadRadius: 1,
        offset: const Offset(0, 3),
      ),
    ],
  ),
  padding: const EdgeInsets.symmetric(horizontal: 20, vertical: 40),
  child: Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: <Widget>[
      const CircleAvatar(
        radius: 65.0,
        backgroundImage: AssetImage('assets/profile_image.jpg'),
      ),
      const SizedBox(height: 20),
      Text(
        user['name'],
        style: const TextStyle(
          color: Colors.white,
          fontSize: 28,
          fontWeight: FontWeight.bold,
        ),
```

```
        ),
        const SizedBox(height: 10),
        const Text(
          '20 років | Жіноча',
          style: TextStyle(
            color: Colors.white,
            fontSize: 18,
          ),
        ),
      ],
    ),
  ),
),
Expanded(
  flex: 5,
  child: Container(
    decoration: BoxDecoration(
      color: Colors.white,
      borderRadius: const BorderRadius.only(
        topLeft: Radius.circular(40),
        topRight: Radius.circular(40),
      ),
      boxShadow: [
        BoxShadow(
          color: Colors.black.withOpacity(0.1),
          blurRadius: 10,
          spreadRadius: 1,
          offset: const Offset(0, -3),
        ),
      ],
    ),
    padding: const EdgeInsets.symmetric(horizontal: 20, vertical: 40),
```

```
child: Column(
  crossAxisAlignment: CrossAxisAlignment.stretch,
  children: [
    _buildMenuItem(Icons.person, "Профіль", () {}),
    const SizedBox(height: 20),
    _buildMenuItem(Icons.history, "Історія записів", () {}),
    const SizedBox(height: 20),
    _buildMenuItem(
      Icons.logout_outlined,
      "Вийти",
      () async {
        final SharedPreferences prefs =
            await SharedPreferences.getInstance();
        final token = prefs.getString('token') ?? '';

        if (token.isNotEmpty && token != '') {
          final response = await DioProvider().logout(token);

          if (response == 200) {
            await prefs.remove('token');
            MyApp.navigatorKey.currentState!
                .pushReplacementNamed('/');
          } else {
            // Handle error if logout failed
            showDialog(
              context: context,
              builder: (ctx) => AlertDialog(
                title: const Text('Logout Failed'),
                content: const Text(
                    'Unable to logout. Please try again later.'),
                actions: <Widget>[
                  TextButton(
```

```
                        onPressed: () {
                          Navigator.of(ctx).pop();
                        },
                        child: const Text('OK'),
                      ),
                    ],
                  ),
                );
              }
            }
          },
        ),
      ],
    ),
  ),
 );
}

Widget _buildMenuItem(IconData icon, String label, VoidCallback onTap) {
  return GestureDetector(
    onTap: onTap, // Use the passed onTap callback
    child: Container(
      padding: const EdgeInsets.symmetric(horizontal: 16, vertical: 16),
      decoration: BoxDecoration(
        color: EasyMedHelpConfiguration.primaryColor.withOpacity(0.1),
        borderRadius: BorderRadius.circular(20),
      ),
      child: Row(
        children: [
```

```
            Icon(
              icon,
              color: EasyMedHelpConfiguration.primaryColor,
              size: 35,
            ),
            const SizedBox(width: 20),
            Text(
              label,
              style: const TextStyle(
                color: EasyMedHelpConfiguration.primaryColor,
                fontSize: 20,
                fontWeight: FontWeight.w500,
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

```
import 'package:easymedhelp_app/main.dart';
import 'package:easymedhelp_app/providers/dio_provider.dart';
import 'package:easymedhelp_app/utils/configuration.dart';
import 'package:flutter/material.dart';
import 'package:camera/camera.dart';
import 'package:flutter_svg/svg.dart';
import 'dart:async';

import 'package:rating_dialog/rating_dialog.dart';
```

```
import 'package:shared_preferences/shared_preferences.dart';

class VideoCallScreen extends StatefulWidget {
  final NetworkImage doctorImage;
  final String doctorName;

  final Map<String, dynamic> doctor;

  const VideoCallScreen({
    Key? key,
    required this.doctorImage,
    required this.doctorName,
    required this.doctor,
  }) : super(key: key);

  @override
  State<VideoCallScreen> createState() => _VideoCallScreenState();
}

class _VideoCallScreenState extends State<VideoCallScreen> {
  late CameraController controller;
  late bool isCameraOn = true;
  late bool isMicrophoneOn = true;
  late int minutes = 0;
  late int seconds = 0;
  late Timer timer;
```

```
@override
void initState() {
  initializeCamera();
  super.initState();
  isCameraOn = true;
  isMicrophoneOn = true;
  startTimer();
}

@override
void dispose() {
  controller.dispose();
  timer.cancel();
  super.dispose();
}

Future<void> initializeCamera() async {
  final cameras = await availableCameras();
  controller = CameraController(cameras[0], ResolutionPreset.max);
  await controller.initialize();
  if (mounted) {
    setState(() {});
  }
}

void toggleCamera() {
  setState(() {
```

```
    isCameraOn = !isCameraOn;
  });
}

void toggleMicrophone() {
  setState(() {
    isMicrophoneOn = !isMicrophoneOn;
  });
}

void hangUpCall() {
  showDialog(
    context: context,
    builder: (BuildContext context) {
      return AlertDialog(
        title: Text("Завершити дзвінок"),
        content: Text("Ви впевнені, що хочете завершити дзвінок?"),
        actions: [
          TextButton(
            onPressed: () {
              Navigator.of(context).pop(); // Закрити діалогове вікно
            },
            child: Text("Скасувати"),
          ),
          TextButton(
            onPressed: () {
              showDialog(
```

```
context: context,
builder: (context) {
  return RatingDialog(
    initialRating: 1.0,
    title: const Text(
      'Оцініть лікаря',
      textAlign: TextAlign.center,
      style: TextStyle(
        fontSize: 25,
        fontWeight: FontWeight.bold,
      ),
    ),
    message: const Text(
      'Будь ласка, оцініть ваш візит до лікаря',
      textAlign: TextAlign.center,
      style: TextStyle(
        fontSize: 15,
      ),
    ),
    image: SvgPicture.asset(
      'assets/doctor_auth.svg',
      height: 120.0,
      fit: BoxFit.contain,
    ),
    submitButtonText: 'Надіслати',
    commentHint: 'Ваш відгук',
    onSubmitted: (response) async {
```

```
                final SharedPreferences prefs =
                    await SharedPreferences.getInstance();
                final token = prefs.getString('token') ?? '';

                final rating = await DioProvider().storeReviews(
                  response.comment,
                  response.rating,
                  widget.doctor['appointments']['id'],
                  widget.doctor['doctor_id'],
                  token,
                );

                if (rating == 200 && rating != '') {
                  MyApp.navigatorKey.currentState!.pushNamed('main');
                }
              },
            );
          },
        );
      },
      child: Text("Завершити дзвінок",
          style: TextStyle(color: Colors.red)),
    ),
  ],
 );
},
);
```

```
}

void startTimer() {
  timer = Timer.periodic(Duration(seconds: 1), (Timer t) {
    setState(() {
      if (seconds < 59) {
        seconds++;
      } else {
        seconds = 0;
        minutes++;
      }
    });
  });
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      backgroundColor: EasyMedHelpConfiguration.backgroundColor,
      title: Align(
        alignment: Alignment.centerLeft,
        child: Text(
          widget.doctorName,
          textAlign: TextAlign.left,
        ),
      ),
```

```
  automaticallyImplyLeading: false, // Hides the back button
  actions: [
   // Timer Display in the AppBar
   Padding(
    padding: EdgeInsets.only(right: 20.0),
    child: Row(
     children: [
      Text(
       '$minutes:${seconds < 10 ? '0$seconds' : seconds}',
       style: TextStyle(fontSize: 18.0),
      ),
     ],
    ),
   ),
  ],
 ),
 body: Stack(
  children: [
   // Camera Preview or Placeholder
   if (isCameraOn && controller.value.isInitialized)
    Positioned.fill(
     child: ClipRect(
      child: OverflowBox(
       alignment: Alignment.center,
       child: FittedBox(
        fit: BoxFit.cover,
        child: SizedBox(
```

```
            width: controller.value.previewSize!.height,
            height: controller.value.previewSize!.width,
            child: CameraPreview(controller),
          ),
        ),
      ),
    ),
  ),
if (!isCameraOn || !controller.value.isInitialized)
  Container(
    color: Colors.purple,
    child: Center(
      child: Icon(
        Icons.account_circle,
        size: 100,
        color: Colors.white,
      ),
    ),
  ),
// Doctor Image
Positioned(
  top: 60.0,
  right: 20.0,
  child: Container(
    width: 120,
    height: 200,
    decoration: BoxDecoration(
```

```
      image: DecorationImage(
        image: widget.doctorImage,
        fit: BoxFit.cover,
      ),
      borderRadius: BorderRadius.circular(20.0),
    ),
  ),
),
// Control Buttons
Align(
  alignment: Alignment.bottomCenter,
  child: Padding(
    padding: const EdgeInsets.only(bottom: 20.0),
    child: Row(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        FloatingActionButton(
          onPressed: toggleMicrophone,
          child: Icon(isMicrophoneOn ? Icons.mic : Icons.mic_off),
        ),
        SizedBox(width: 20.0),
        FloatingActionButton(
          onPressed: toggleCamera,
          child:
              Icon(isCameraOn ? Icons.videocam : Icons.videocam_off),
        ),
        SizedBox(width: 20.0),
```

```
                FloatingActionButton(
                  onPressed: hangUpCall,
                  backgroundColor: Colors.red,
                  foregroundColor: Colors.white,
                  child: Icon(Icons.call_end),
                ),
              ],
            ),
          ),
        ),
      ],
    ),
  );
 }
}
```

Додаток Г. Ілюстративна частина

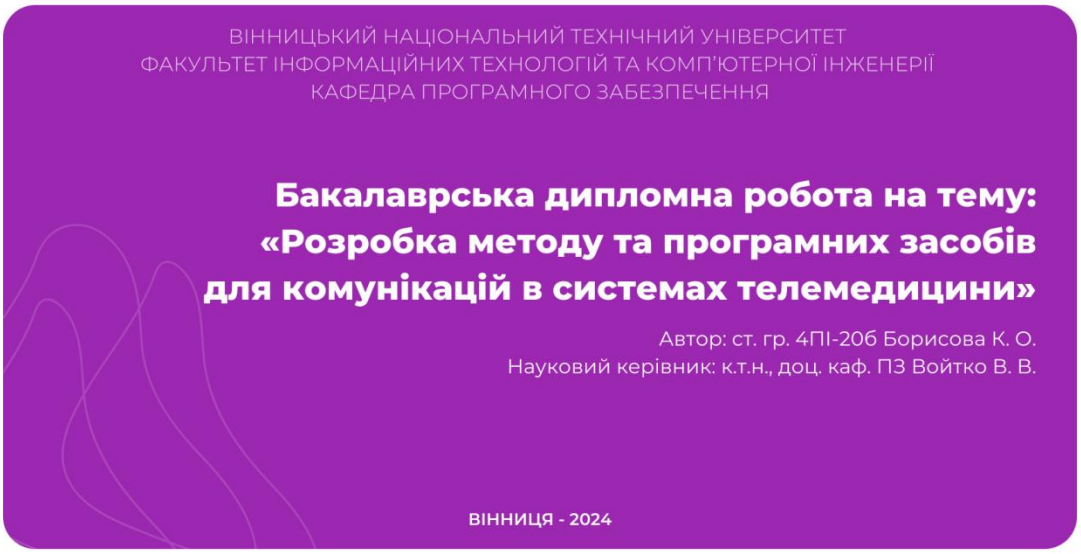


Рисунок Г.1 – Титульний слайд

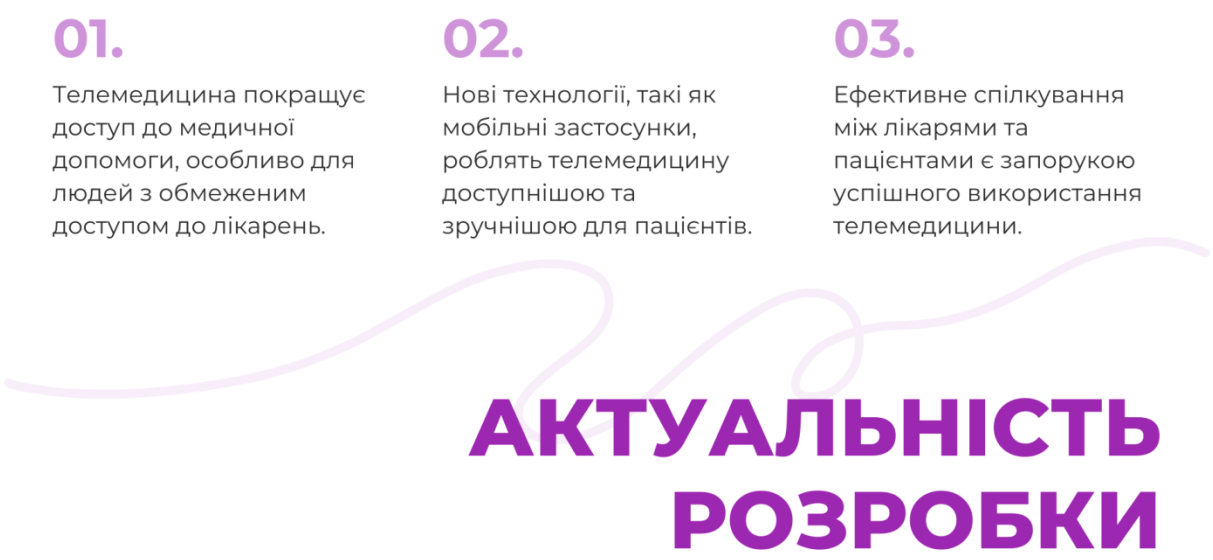


Рисунок Г.2 – Актуальність розробки

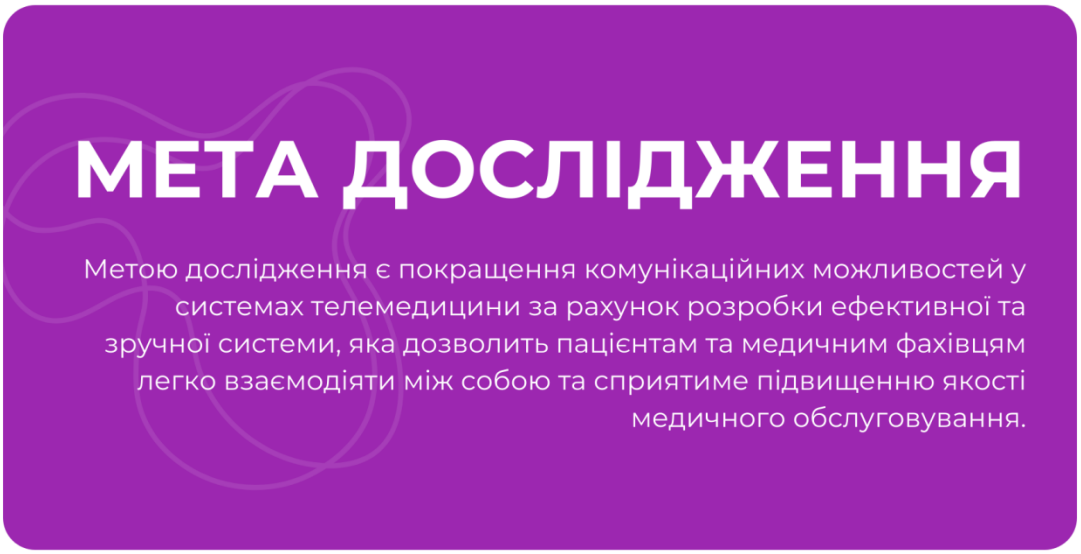


Рисунок Г.3 – Мета дослідження

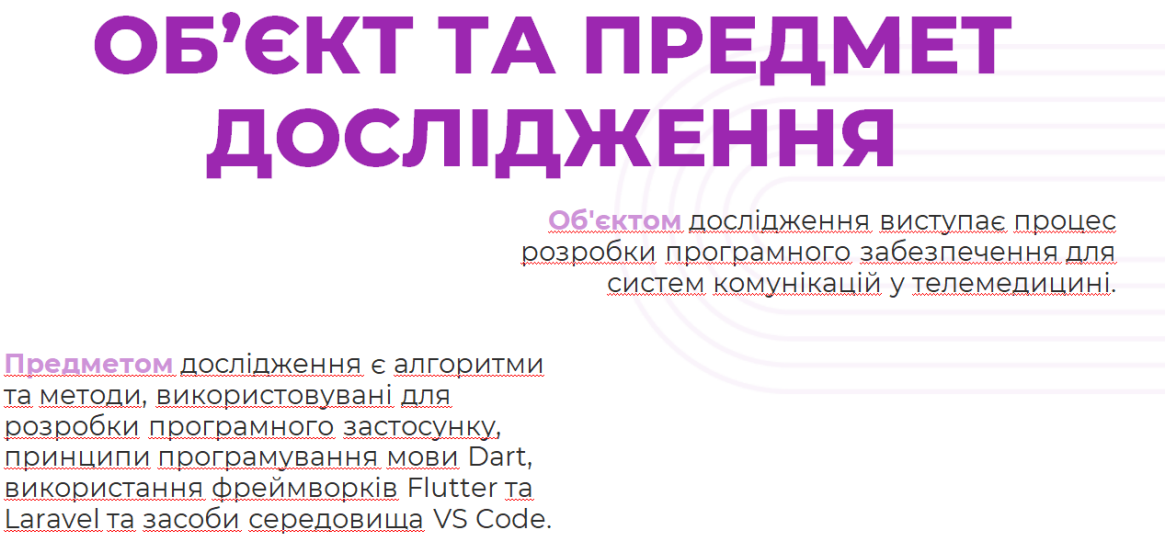


Рисунок Г.4 – Об’єкт та предмет дослідження

ПОСТАНОВКА ЗАДАЧІ

Після детального аналізу сильних та слабких сторін існуючих систем комунікації в телемедицині, були визначені основні напрями для розробки програмного рішення:

01.

Створити інтуїтивно зрозумілий та простий у використанні графічний інтерфейс мобільного застосунку, адаптований до різних типів пристроїв.

02.

Розробити мобільний застосунок, який інтегрує в себе всі необхідні функції для ефективної комунікації, включаючи можливість проведення відео консультацій, обміну повідомленнями, а також надання зворотного зв'язку від пацієнтів

03.

Провести ретельне тестування розробленої системи, щоб забезпечити її стабільність, ефективність та користувацьку зручність.

Рисунок Г.5 – Постановка задачі

НОВИЗНА

01.

Подальшого розвитку отримав метод візуалізації медичних даних, який, на відміну від існуючих, включає інтеграцію з графічним інтерфейсом користувача, що дозволяє лікарям та пацієнтам з легкістю візуалізувати інформацію про лікарів, покращує розуміння та моніторинг стану здоров'я пацієнтів та сприяє ефективному лікуванню та профілактиці захворювань.

02.

Подальшого розвитку отримала модель комплексної телемедичної системи, яка, на відміну від існуючих, інтегрує набір необхідних інструментів для ведення переговорів, планування та проведення онлайн-консультацій, а також використовує бази даних для забезпечення безперервного доступу до даних, надає можливість більш гнучкої та широкої організації роботи медичних установ.

Рисунок Г.6 – Новизна розробленого застосунку

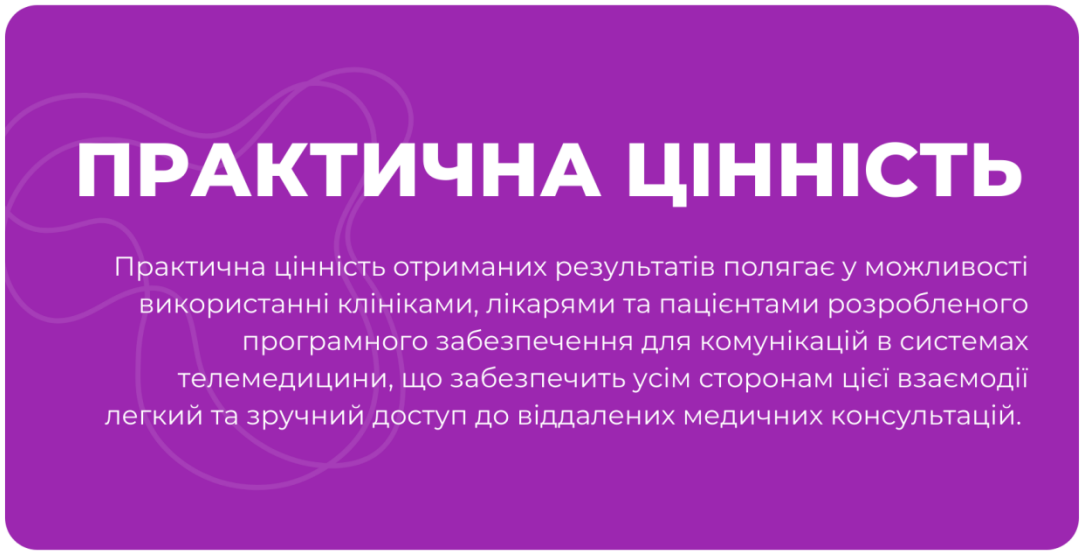
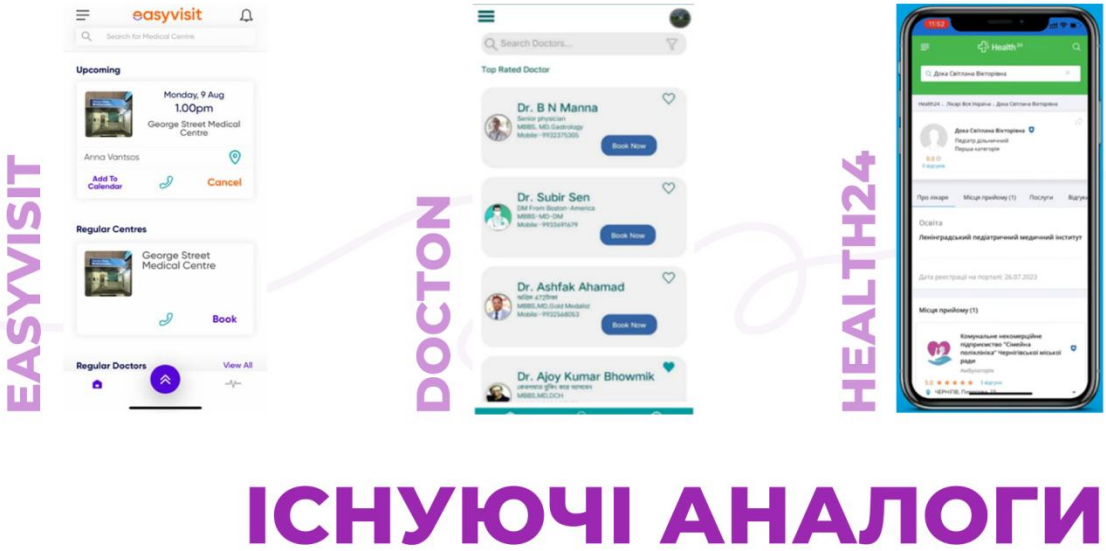


Рисунок Г.7 – Практична цінність



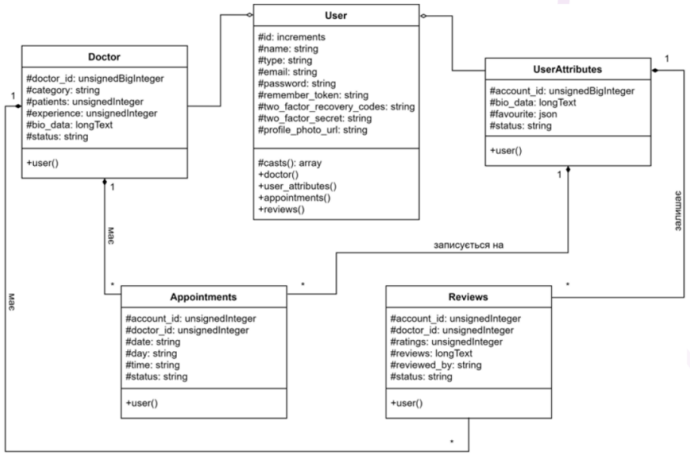
ІСНУЮЧІ АНАЛОГИ

Рисунок Г.8 – Існуючі аналоги

АНАЛІЗ АНАЛОГІВ

Порівняльна характеристика	EasyVisit	Docton	Health24	Власна розробка
Обліковий запис пацієнта	1	1	1	1
Надання відгуків про медичних фахівців	0	1	1	1
Перегляд історії минулих записів до лікарів	1	1	1	1
Список улюблених медичних фахівців	1	1	1	1
Можливість зв'язку зі спеціалістом у чаті в застосунку	0	0	0	1
Сумарний коефіцієнт	**3**	**4**	**4**	**5**

Рисунок Г.9 – Аналіз аналогів



МОДЕЛЬ РОБОТИ СИСТЕМИ

Рисунок Г.10 – Модель роботи системи

ЗАГАЛЬНИЙ АЛГОРИТМ РОБОТИ СИСТЕМИ

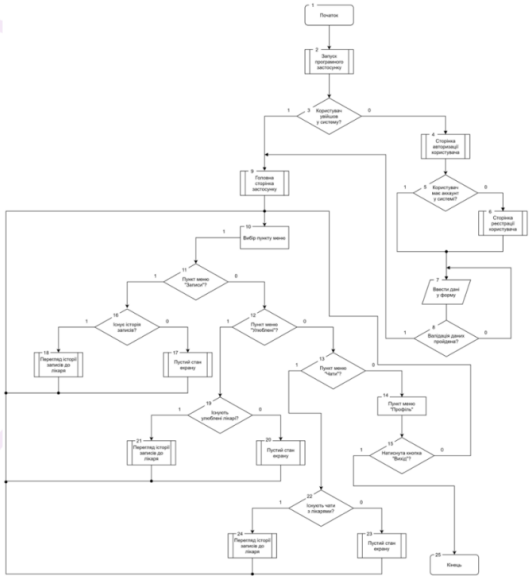


Рисунок Г.11 – Загальний алгоритм роботи системи

ДІАГРАМИ СТАНІВ

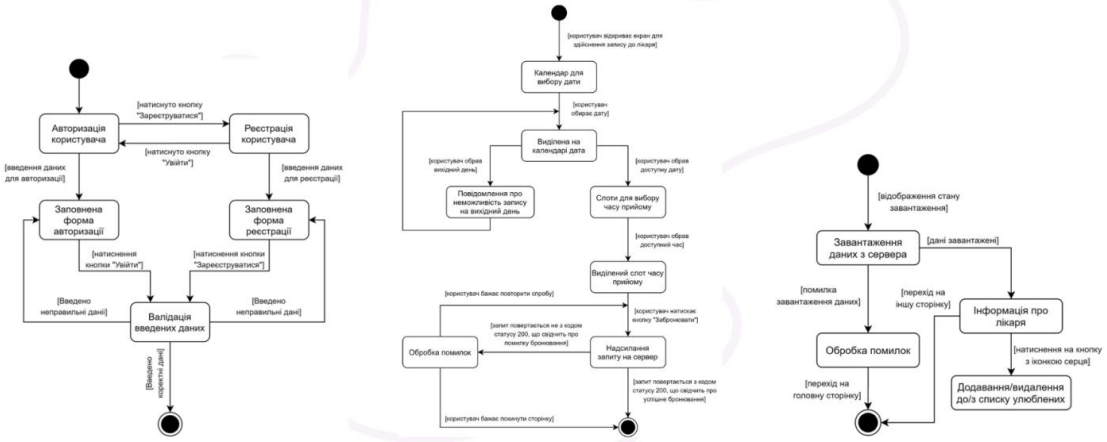


Рисунок Г.12 – Діаграми станів

Рисунок Г.13 – Схеми інтерфейсу

Рисунок Г.14 – Алгоритм обчислення рейтингу лікаря

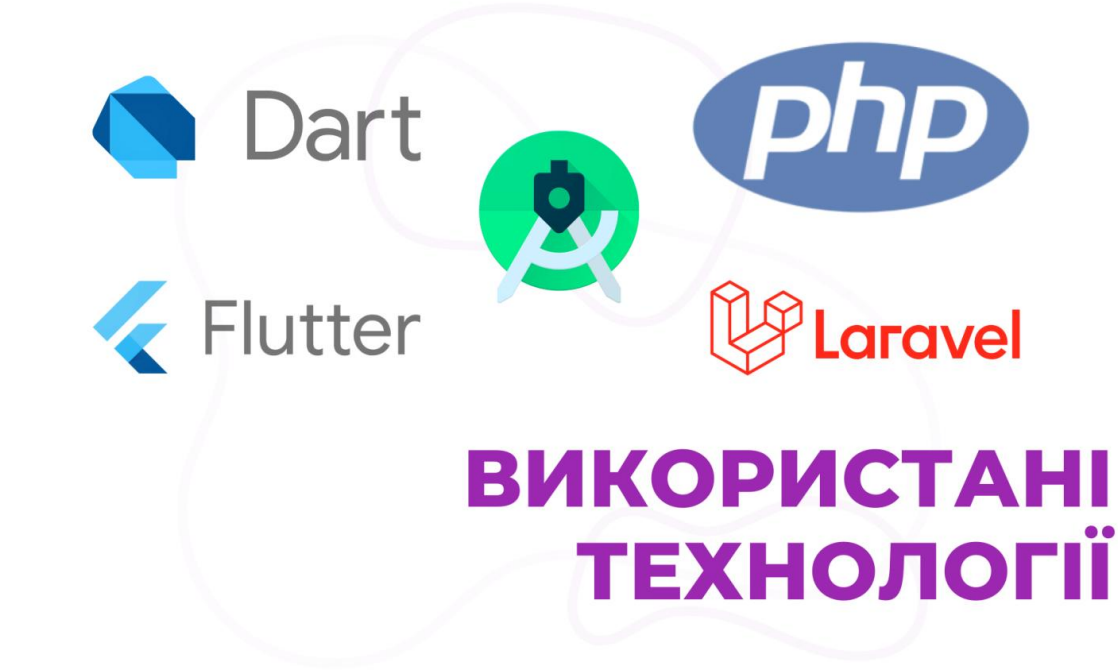


Рисунок Г.15 – Використані технології

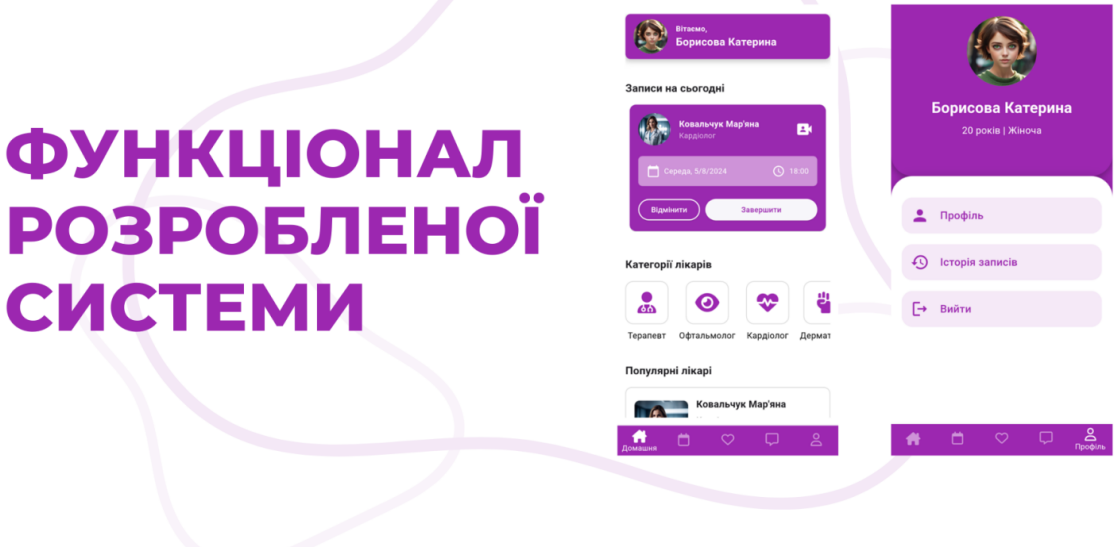


Рисунок Г.16 – Функціонал розробленої системи

(головна сторінка та профіль користувача)

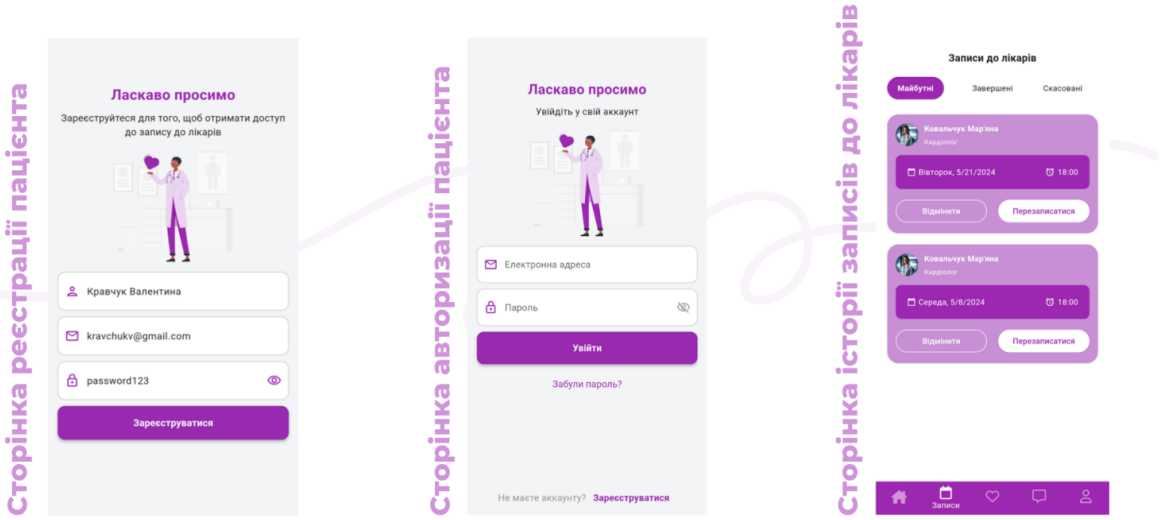


Рисунок Г.17 – Функціонал розробленої системи
(реєстрація, авторизація та історія записів)

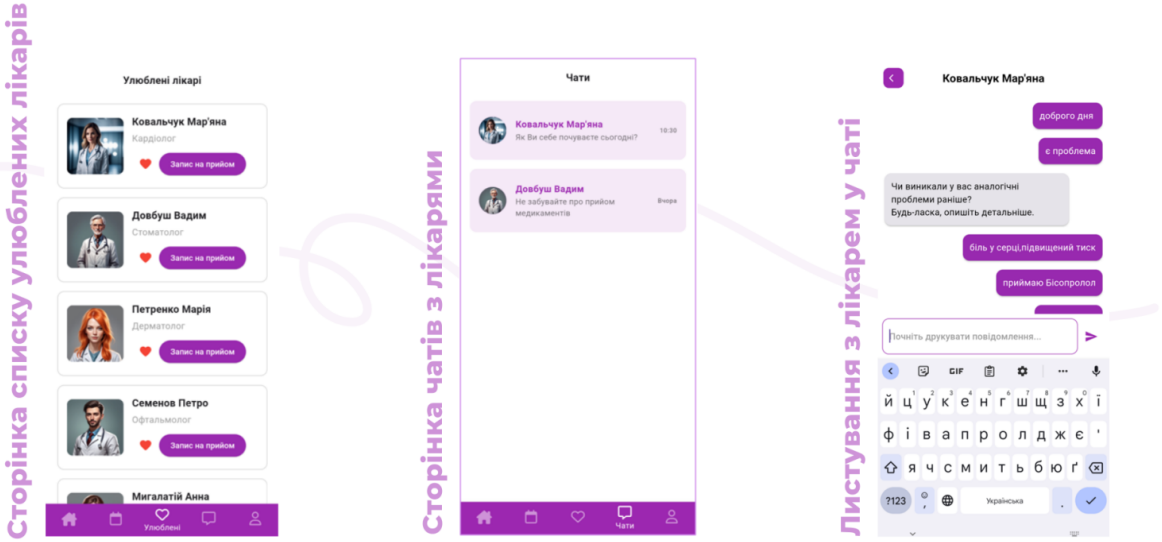


Рисунок Г.18 – Функціонал розробленої системи
(улюблені лікарі та функціонал чатів)

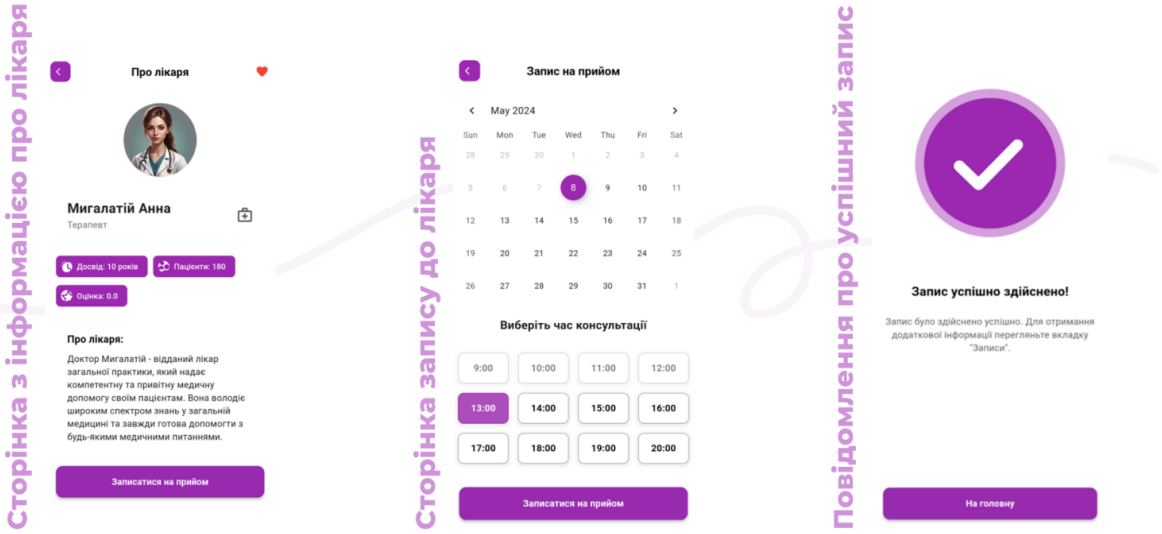


Рисунок Г.19 – Функціонал розробленої системи
(запис на прийом до лікаря)

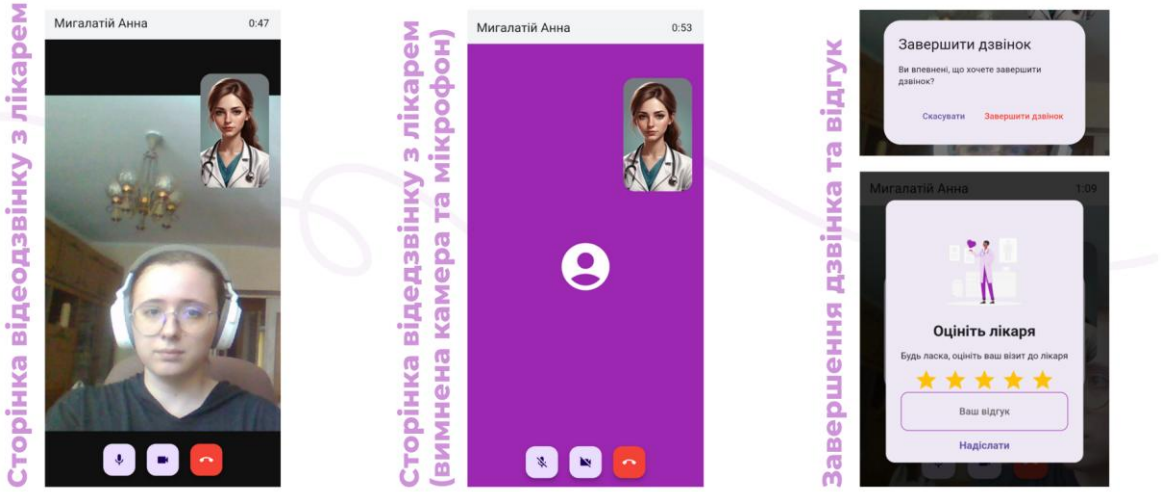


Рисунок Г.20 – Функціонал розробленої системи
(відео дзвінок та залишення відгуку)

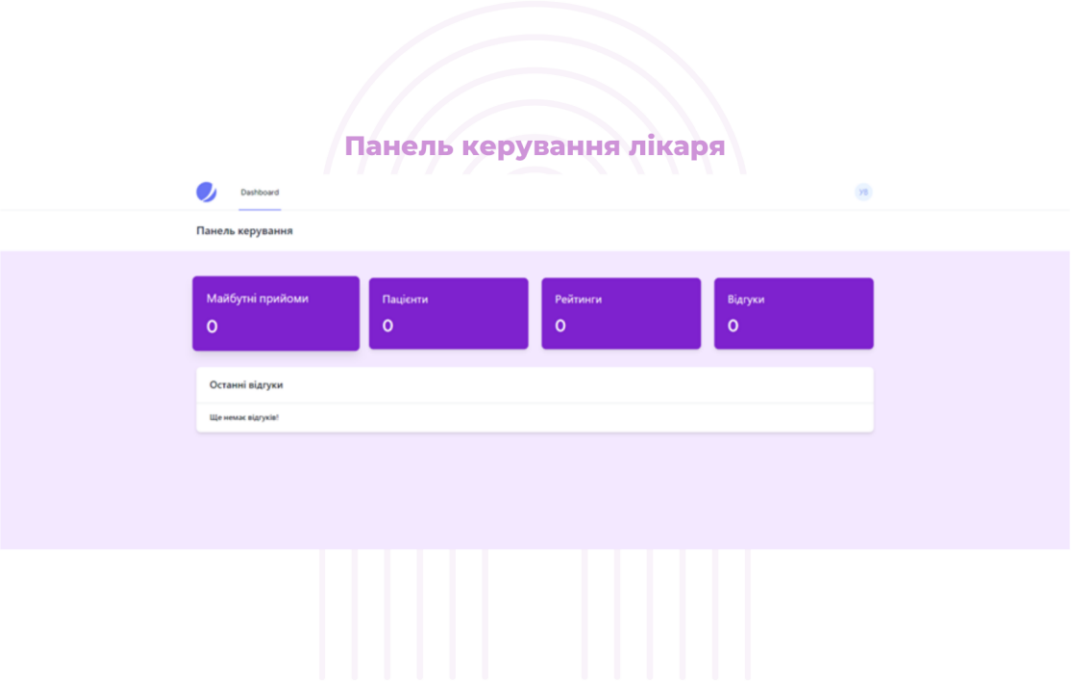


Рисунок Г.21 – Панель керування лікаря

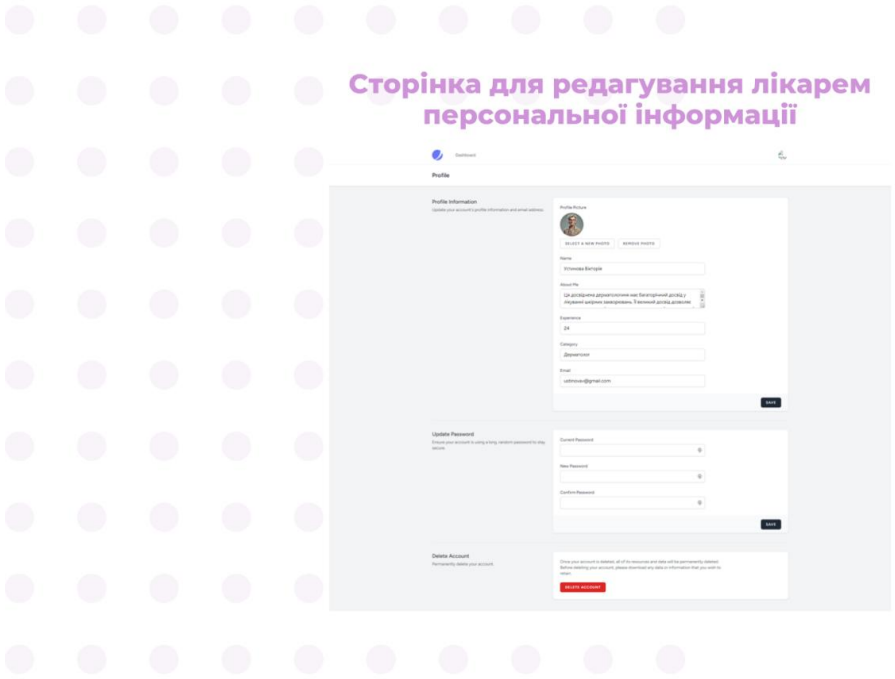


Рисунок Г.22 – Сторінка для редагування лікарем персональної інформації

АПРОБАЦІЯ ТА ПУБЛІКАЦІЯ РЕЗУЛЬТАТІВ РОБОТИ

Матеріали бакалаврської дипломної роботи доповідалися на:

Автоматизація та комп'ютерно-інтегровані технології у виробництві та освіті: стан, досягнення, перспективи розвитку» (2024)

За тематикою дослідження опубліковано 1 наукову працю у збірниках матеріалів конференції.


ЧЕРКАСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
імені Богдана Хмельницького

СЕРТИФІКАТ

засвідчує, що

Катерина Борисова

приймав (ла) участь у Всеукраїнській науково-практичній Інтернет-конференції
"Автоматизація та комп'ютерно-інтегровані технології у виробництві та освіті: стан, досягнення, перспективи розвитку"
11-17 березня 2024 року
м. Черкаси, Україна

РЕКТОР ОЛЕКСАНДР ЧЕРЕВКО

Рисунок Г.23 – Апробація та публікація результатів роботи

ВИСНОВКИ

У результаті виконання бакалаврської дипломної роботи було розроблено мобільну систему для комунікацій у сфері телемедицини, яка включає такі функціональні модулі:

- реєстрація облікового запису для пацієнта, можливість запису на прийом до лікаря з обранням потрібного спеціаліста та часу, відстеження статусу записів, можливість переглядати історію записів до лікарів, додавання медичних фахівців у список улюблених для швидкого доступу до них;
- функціонал для консультації з лікарем онлайн: текстовий чат та онлайн-прийом з використанням веб-камери пристрою користувачів.

Крім цього, було розроблено веб-дешборд для перегляду лікарем статистики про його роботу.

Рисунок Г.24 – Висновки

Рисунок Г.25 – Фінальний слайд