МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри_____ Є. О. Давиденко

підпис

«___» _____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Вебзастосунок онлайн-клініки з використанням телеграм-боту

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ.1 – 408.22120804

Здобувач _____Д. С. Хоменко

підпис

«__» _____ 2024 р.

Керівник канд. техн. наук, доцент _____ Г. В. Горбань

підпис

«__» _____ 2024 р.

Консультант канд. техн. наук, доцент _____ А. О. Алексєєва

підпис

«__» _____ 2024 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ

Зав. кафедри _____ Є. О. Давиденко

«_____» _____ 2024 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи бакалавра

Видано здобувачу групи 408 факультету комп'ютерних наук

_____Хоменко Дмитро Сергійович_____

(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи

Вебзастосунок онлайн-клініки з використанням телеграм-боту

_____

_____

Затверджена наказом по ЧНУ від «22» _____грудня_____ 2023 р. № ___269_____

2. Строк представлення кваліфікаційної роботи «____» _____ 20__ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні

Очікуваним результатом є вебзастосунок онлайн-клініки та телеграм-бот онлайн-клініки_____

_____

4. Перелік питань, що підлягають розробці:

- дослідження предметної області та аналіз існуючих аналогів;
- формування специфікації вимог до програмного забезпечення;
- визначення архітектури для проєктування програмного забезпечення;
- моделювання та проєктування програмного забезпечення;
- розробка програмного забезпечення;
- здійснення тестування роботи програмного забезпечення;

5. Перелік графічних матеріалів

__Презентація_____

6. Завдання до спеціальної частини

__Аналіз охорони праці на робочих місцях фахівців з інформаційних технологій__

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
Алєксєєва А. О.	Кафедра екології	Спеціальна частина з охорони праці

Керівник роботи _____ канд. техн. наук, доцент Горбань Г. В._____
(посада, прізвище, ім’я, по батькові)

_____
(підпис)

Завдання прийнято до виконання

_____Хоменко Дмитро Сергійович_____
(прізвище, ім’я, по батькові)

_____
(підпис)

Дата видачі завдання «____» _____ 2024р.

КАЛЕНДАРНИЙ ПЛАН
виконання бакалаврської кваліфікаційної роботи

Тема: «Вебзастосунок онлайн-клініки з використанням телеграм-боту»

№	Найменування роботи	Початок	Закінчення	Примітки
1	Розробка та затвердження завдання на виконання КРБ	20.02.2024	23.02.2024	Виконано
2	Огляд літератури за темою роботи	24.02.2024	26.02.2024	Виконано
3	Складання календарного плану КРБ	26.02.2024	27.02.2024	Виконано
4	Аналіз предметної області	28.02.2024	01.03.2024	Виконано
5	Розробка проєктних рішень	04.03.2024	10.03.2024	Виконано
6	Моделювання та конструювання ПЗ	22.03.2024	26.03.2024	Виконано
7	Кодування ПЗ, тестування та апробація розробленого ПЗ, аналіз результатів тестування	28.03.2024	22.05.2024	Виконано
8	Розробка спеціальної частини з охорони праці	02.04.2024	22.04.2024	Виконано
9	Оформлення КРБ та презентації	02.05.2024	20.05.2024	Виконано
10	Відгук керівника КРБ	22.05.2024	02.06.2024	Виконано
11	Попередній захист	03.06.2024	05.06.2024	Виконано
12	Рецензування	13.06.2024	14.06.2024	Виконано
13	Захист кваліфікаційної роботи	25.06.2024	26.06.2024	Виконано

Розробив здобувач Хоменко Дмитро Сергійович _____
(прізвище, ім'я, по батькові) *(підпис)*
«24» січня 2024 р.

Керівник роботи канд. техн. наук, доцент Горбань Г. В.
(посада, прізвище, ім'я, по батькові) *(підпис)*
«24»січня 2024 р.

АНОТАЦІЯ

до кваліфікаційної роботи бакалавра

«Вебзастосунок онлайн-клініки з використанням телеграм-боту»

Здобувач 408 гр,: Хоменко Дмитро Сергійович

Керівник: канд. техн. наук, доцент Горбань Г. В.

Актуальність даної роботи зумовлена необхідністю модернізації системи охорони здоров'я та забезпечення швидкого і зручного доступу до медичних послуг у сучасному цифровому світі.

Кваліфікаційна робота присвячена модернізації системи охорони здоров'я та забезпеченням швидкого та зручного доступу до медичних послуг у сучасному цифровому світі.

Об'єктом роботи є процес надання медичних послуг у віртуальному середовищі.

Предметом є засоби розробки та впровадження телеграм-боту в онлайн-клініці для поліпшення доступу до медичних послуг.

Метою даної роботи є розробка вебзастосунку онлайн-клініки для оптимізації процесу надання медичних послуг шляхом використання телеграм-боту в онлайн-клініці.

У першому розділі кваліфікаційної роботи бакалавра представлено аналіз предметної області, аналіз готових рішень, постановка задачі та специфікація вимог. У другому розділі спроєктовано декілька uml діаграм для вебзастосунку онлайн-клініки, а саме діаграму варіантів використання, діаграму класів, діаграму послідовностей та мокап системи. У третьому розділі описані методи розробки вебзастосунку та описано базу даних. У четвертому розділі описані результати розробки вебзастосунку.

КРБ викладена на 57 сторінок, вона містить 4 розділи, 44 ілюстрації, 23 джерел в переліку посилань

Ключові слова: вебсайт, онлайн-клініка, телеграм-бот, Python, PHP, Laravel, база даних, MySQL.

ABSTRACT

to the qualification work of the bachelor
«Online clinic web application using Telegram bot»
Student 408 gr,: Khomenko Dmytro Serhiyovych
Supervisor: candidate technical Sciences, Associate Professor G. V. Horban.

The relevance of this work is determined by the need to modernize the health care system and ensure quick and convenient access to medical services in the modern digital world.

Thesis is dedicated to the modernization of the health care system and ensuring quick and convenient access to medical services in the modern digital world.

The object of work is the process of providing medical services in a virtual environment.

The subject is the means of developing and implementing a Telegram bot in an online clinic to improve access to medical services.

The purpose of this work is to optimize the process of providing medical services by using a Telegram bot in an online clinic.

The first section of the bachelor's qualification work presents the analysis of the subject area, the analysis of ready-made solutions, the statement of the problem and the specification of requirements. In the second chapter, several uml diagrams are designed for the online clinic web application, namely use case diagram, class diagram, sequence diagram and system mockup. The third chapter describes the methods of developing a web application and describes the database. The fourth chapter describes the results of the web application development.

BQW is laid out on 57 pages, it contains 4 sections, 44 illustrations, 23 sources in the list of references

Keywords: website, online clinic, Telegram bot, Python PHP, Laravel, database, MySQL.

ЗМІСТ

ВСТУП ... 3
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ ... 4
1.1 Аналіз предметної області ... 4
1.2 Аналіз готових рішень ... 5
1.2.1 Аналіз «Онлайн клініка» ... 6
1.2.2 Аналіз «Онлайн-консультація «ОН Клиник»» ... 8
1.2.3 Аналіз «Viva клінік» ... 9
1.3 Специфікація вимог до програмного забезпечення ... 11
Висновки до розділу 1 ... 14
2 ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ ОНЛАЙН-КЛІНІКИ ... 15
2.1 Розробка діаграми варіантів використання ... 15
2.2 Розробка діаграми класів ... 16
2.3 Розробка діаграми послідовностей дій системи ... 17
Висновки до розділу 2 ... 18
3 КОНСТРУЮВАННЯ ВЕБЗАСТОСУНКУ ОНЛАЙН-КЛІНІКИ ... 19
3.1 Вибір технологій розробки ... 19
3.2 Моделювання бази даних ... 24
3.3 Розробка мокапу системи ... 30
Висновки до розділу 3 ... 37
4 РЕЗУЛЬТАТИ РОЗРОБКИ ... 38
4.1 Реалізація вебзастосунку ... 38
4.2 Реалізація телеграм-боту ... 51
Висновки до розділу 4 ... 54
ВИСНОВКИ ... 55
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ ... 56
ДОДАТОК А – ЛІСТИНГ КОДУ ПРОГРАМИ ... 58

ВСТУП

Сучасний світ стикається з проблемою доступності медичних послуг та їх ефективного надання. Зростаюча потреба у зручних та оперативних способах отримання консультацій та допомоги від лікарів ставить перед собою виклик – як зробити медичні послуги доступними для кожного, незалежно від географічного розташування та часу?

Виходячи з цього, вибір теми дослідження «Вебзастосунок онлайн-клініки з використанням телеграм-боту» є наочним відображенням потреби суспільства у нових, інноваційних підходах до надання медичних послуг. Подальший розвиток технологій, зокрема використання месенджерів та ботів, відкриває широкі можливості для впровадження зручних та ефективних медичних платформ.

Об'єктом роботи є процес надання медичних послуг у віртуальному середовищі.

Предметом роботи є засоби розробки та впровадження телеграм-боту в онлайн-клініці для поліпшення доступу до медичних послуг.

Метою даної роботи є розробка вебзастосунку онлайн-клініки для оптимізації процесу надання медичних послуг шляхом використання телеграм-боту в онлайн-клініці.

Для досягнення цієї мети передбачено вирішення **наступних завдань**:

1) розробка функціоналу телеграм-боту для онлайн-клініки;
2) інтеграція боту з базою даних медичної системи;
3) тестування та аналіз ефективності роботи онлайн-клініки;
4) вдосконалення функціоналу на основі отриманих результатів тестування;
5) оцінка практичного застосування та можливих переваг для пацієнтів та медичних працівників.

Актуальність даної роботи обумовлена необхідністю модернізації системи охорони здоров'я та забезпеченням швидкого та зручного доступу до медичних послуг у сучасному цифровому світі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз предметної області

У сучасному світі розвиток технологій змінює підходи до надання медичних послуг. Однією з інноваційних тенденцій є використання онлайн-клінік, що надають можливість консультуватися з лікарями та отримувати медичну допомогу віддалено, без виходу з дому. У цьому контексті використання телеграм-ботів стає зручним інструментом комунікації з пацієнтами.

Зокрема, телеграм-боти в медичній сфері можуть забезпечити швидкий доступ до інформації про стан здоров'я, надати рекомендації щодо профілактики захворювань та лікування, а також допомогти в плануванні медичних прийомів. Це особливо актуально для людей, які не мають можливості або бажання відвідувати лікарні через зайнятість, віддаленість від медичних установ або інші обставини.

Перевагами використання телеграм-ботів у медичній сфері є їхня доступність та зручність для користувачів. Люди можуть звертатися до бота у будь-який час, отримуючи необхідну інформацію чи послуги миттєво. Такий підхід може значно зменшити час очікування на консультацію або прийом до лікаря, що особливо важливо в ситуаціях, коли потрібна швидка медична допомога.

Проте, існують і певні недоліки. Наприклад, обмежені можливості взаємодії з пацієнтами у порівнянні з реальними консультаціями. Деякі медичні питання можуть вимагати особистого контакту з лікарем або проведення додаткових обстежень, що в онлайн-форматі може бути ускладненим.

У порівнянні з іншими формами віртуальної медичної допомоги, такими як мобільні застосунки або відеоконференції з лікарем, використання телеграм-ботів має свої особливості. Цей підхід може бути менш функціональним, але більш доступним для користувачів, зокрема для тих, хто вже використовує телеграм для інших цілей та має звичку користуватися цією платформою.

1.2 Аналіз готових рішень

Для розробки та реалізації свого застосунку потрібно визначити переваги та недоліки вже існуючих продуктів на ринку. Було визначено наступні вимоги:

Функціональність:

– забезпечення можливості реєстрації пацієнтів та створення їхнього медичного профілю;

– надання можливості запису на прийом до лікаря в онлайн-режимі;

– організація системи консультування через чат або відеозв'язок;

– можливість ведення медичної документації та історії захворювань пацієнтів;

– реалізація опцій відстеження медичних показників та приписання лікування.

Безпека та конфіденційність:

– забезпечення захищеності медичних даних пацієнтів у відповідності до вимог законодавства про конфіденційність медичної інформації;

– механізми аутентифікації та авторизації для запобігання несанкціонованому доступу до медичної інформації.

Інтеграція та сумісність:

– можливість інтеграції з існуючими інформаційними системами медичних установ (наприклад, Єдиний медичний інформаційний простір);

– сумісність з різними типами пристроїв та операційними системами для максимальної доступності для користувачів.

Зручність користування:

– інтуїтивний та зрозумілий інтерфейс для пацієнтів та медичного персоналу;

– можливість взаємодії з застосунком через різні канали комунікації (чат, відеозв'язок, тощо);

– наявність опційного оформлення та кастомізації для врахування специфіки роботи різних медичних установ.

Технічна підтримка та оновлення:

– наявність технічної підтримки для вирішення технічних проблем та запитань користувачів;

– регулярні оновлення для забезпечення безпеки та функціональності застосунку.

Вартість та ефективність:

– оцінка вартості впровадження та підтримки відповідного вебзастосунку в порівнянні з іншими рішеннями на ринку;

– аналіз витрат та ефективності використання для медичних установ та пацієнтів.

Відповідність з регулятивними вимогами:

– відповідність законодавчим та регулятивним вимогам у сфері медичних послуг та обробки медичної інформації;

– дотримання стандартів якості та безпеки медичних засобів та послуг.

Аналіз готових рішень у відповідності до вищезазначених вимог дозволить визначити найбільш підходящий варіант для розроблення та впровадження онлайн-клініки з використанням телеграм-бота.

Розглянуто наступні вебсервіси:

- онлайн клініка;
- онлайн-консультація «ОН Клиник»;
- Viva.

1.2.1 Аналіз «Онлайн клініка»

«Онлайн клініка» – це вебплатформа або мобільний застосунок, що надає можливість консультування з лікарем та отримання медичної допомоги в онлайн-режимі. Вона дозволяє користувачам здійснювати реєстрацію, запис на прийом до лікаря, спілкуватися з медичним персоналом через чат або відеозв'язок, а також отримувати рекомендації щодо профілактики та лікування захворювань. Онлайн клініка дозволяє отримати медичну допомогу без виходу з дому, зручно для тих, хто має обмежені можливості або потребує швидкої консультації [1].

Головний екран даного готового рішення представлено на рисунку 1.1.

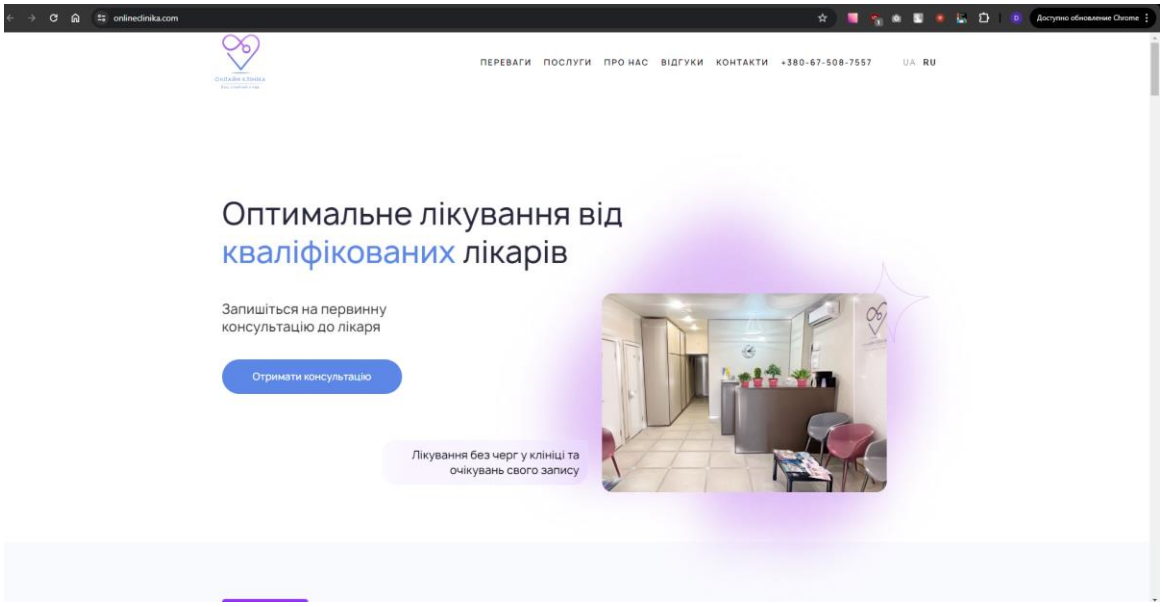

Рисунок 1.1 – Відображення головного екрану «Онлайн клініки»

Функціональність:

– «Онлайн клініка» забезпечує можливість реєстрації пацієнтів, запису на прийом, консультування та отримання медичних рекомендацій в онлайн-режимі. Вона також може включати функції ведення медичної історії та документації про пацієнтів.

Безпека та конфіденційність:

– Система «Онлайн клініка» повинна гарантувати захищеність медичних даних пацієнтів та використовувати механізми аутентифікації та авторизації для запобігання несанкціонованому доступу до них.

Інтеграція та сумісність:

– Важливою властивістю «Онлайн клініки» є можливість інтеграції з існуючими медичними системами, що дозволить обмінюватися даними та забезпечувати єдиний доступ до медичної інформації.

Зручність користування:

– Інтерфейс користувача «Онлайн клініки» повинен бути інтуїтивно зрозумілим та зручним для користувачів будь-якого рівня компетентності.

Технічна підтримка та оновлення:

– Наявність технічної підтримки та регулярних оновлень застосунку

дозволить забезпечити безперервну роботу та вдосконалення функціональності «Онлайн клініки».

Отже, «Онлайн клініка» відповідає багатьом з визначених критеріїв, забезпечуючи функціональність, безпеку, інтеграцію, зручність користування та технічну підтримку. Додаткові дослідження можуть бути проведені для більш детального аналізу цього вебзастосунку.

1.2.2 Аналіз «Онлайн-консультація «ОН Клиник»»

«Онлайн-консультація «ОН Клиник»» – це медичний сервіс, який надає можливість отримати консультацію від лікаря в онлайн-режимі. Цей сервіс може бути доступний через вебсайт або мобільний застосунок. Користувачі можуть зареєструватися, запланувати консультацію з лікарем, обговорити свої медичні питання та отримати рекомендації щодо лікування або подальших дій. Онлайн-консультация дозволяє зручно та швидко отримувати медичну допомогу, не виходячи з дому, що особливо важливо у випадку невеликих захворювань або у ситуаціях, коли важко звернутися до звичайної медичної установи [2].

Головний екран даного готового рішення представлено на рисунку 1.2.

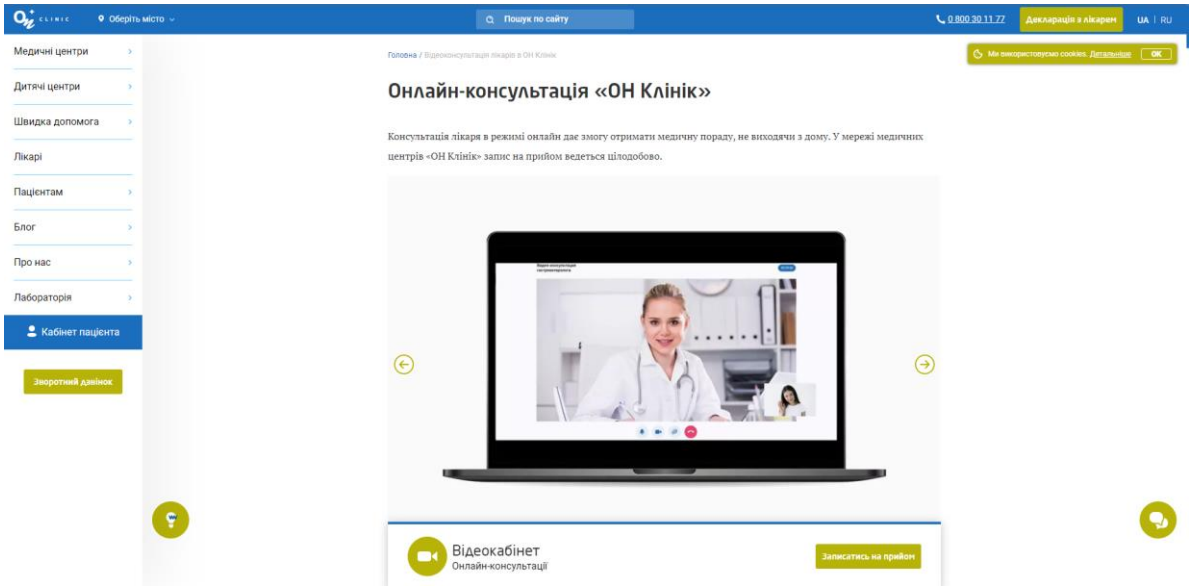

Рисунок 1.2 – Відображення головного екрану «Онлайн клініки»

Функціональність:

– «Онлайн-консультація «ОН Клиник»» надає можливість отримання

консультацій в онлайн-режимі через чат або відеозв'язок. Можливість ведення медичної документації також може бути присутня, але може залежати від умов користування.

Безпека та конфіденційність:

– Важливою аспектом є забезпечення захищеності медичних даних пацієнтів, а також використання механізмів аутентифікації та авторизації для забезпечення конфіденційності.

Інтеграція та сумісність:

– Потрібно вивчити можливість інтеграції «Онлайн-консультації «ОН Клиник»» з іншими медичними системами для обміну даними та забезпечення єдиного доступу до медичної інформації.

Зручність користування:

– Важливо, щоб інтерфейс користувача був зрозумілим та зручним для використання, забезпечуючи простоту та зручність взаємодії з лікарем.

Технічна підтримка та оновлення:

– Наявність технічної підтримки та регулярних оновлень дозволить забезпечити надійну роботу та підтримку функціональності вебзастосунку.

Оцінка «Онлайн-консультація «ОН Клиник»» вказує на те, що вона може відповідати багатьом з визначених критеріїв, забезпечуючи можливість консультації в онлайн-режимі та, можливо, функцію ведення медичної документації. Додаткові дослідження можуть бути проведені для більш детального аналізу цього вебзастосунку.

1.2.3 Аналіз «Viva клінік»

«Viva клінік» – це медичний центр або мережа клінік, що пропонує різноманітні медичні послуги та консультації. Вона може мати фізичні медичні заклади, а також цифрову платформу, що дозволяє отримувати консультації від лікарів в онлайн-форматі. «Viva клінік» може включати в себе широкий спектр медичних напрямків, від загальної медицини та педіатрії до спеціалізованих областей, таких як кардіологія, гінекологія, дерматологія та інші. Крім того, «Viva

клінік» може надавати послуги онлайн-планування прийому до лікаря, електронний обмін медичною інформацією, а також систему відстеження медичних показників та рекомендації щодо здорового способу життя [3].

Головний екран даного готового рішення представлено на рисунку 1.3.

Рисунок 1.3 – Відображення головного екрану «Онлайн клініки»

Функціональність:

– «Viva клінік» пропонує широкий спектр послуг, включаючи онлайн-консультації, планування прийому до лікаря та ведення медичних записів. Функціонал застосунку може також включати можливості реєстрації пацієнтів та створення їхнього медичного профілю.

Безпека та конфіденційність:

– Важливою частиною «Viva клінік» є захист медичних даних пацієнтів та використання механізмів аутентифікації та авторизації для забезпечення конфіденційності.

Інтеграція та сумісність:

– Потрібно дослідити можливість інтеграції «Viva клінік» з іншими медичними системами для обміну даними та забезпечення єдиного доступу до медичної інформації.

Зручність користування:

– Важливо, щоб інтерфейс користувача був зрозумілим та зручним для

використання, забезпечуючи простоту та зручність взаємодії з лікарем та іншими функціями застосунку.

Технічна підтримка та оновлення:

– Наявність технічної підтримки та регулярних оновлень дозволить забезпечити надійну роботу та підтримку функціональності вебзастосунку.

«Viva клінік» може відповідати багатьом з визначених критеріїв, забезпечуючи широкий спектр медичних послуг в онлайн-режимі, забезпечуючи безпеку та конфіденційність даних пацієнтів. Однак, більш детальний аналіз може допомогти з'ясувати всі аспекти функціонування цього вебзастосунку.

1.3 Специфікація вимог до програмного забезпечення

Призначення системи (застосунку), для якої розробляється програмне забезпечення

Призначенням ПЗ, що розробляється, є автоматизація процесу запису на прийом до лікаря.

Погодження, що ухвалені в програмній документації

Погоджень не ухвалено.

Межі проєкту ПЗ

Крайня дата завершення роботи над ПЗ – хх.06.2024р.

Сфера застосування

Дане ПЗ не має обмежень у сферах його застосування, застосунок можна використовувати у повсякденному житті та інших сферах діяльності.

Характеристики користувачів

Основні характеристики користувачів: доступ до мережі Інтернет, наявність ПК або мобільного пристрою.

Загальна структура і склад системи

Основні частини для створення програмного забезпечення: вебзастосунок, сервер та база даних.

Загальні обмеження

Обмеження для роботи з ПЗ – наявність доступу до мережі Інтернет.

Функції вебзастосунку онлайн-клініки:

- реєстрація та авторизація користувачів на вебсайті;
- створення профілів для лікарів та користувачів на вебсайті;
- онлайн запис на прийом до лікаря на вебсайті;
- запис на онлайн консультацію на вебсайті;
- запис на онлайн консультацію через телеграм бота;
- онлайн запис на прийом до лікаря через телеграм бота;
- система відгуків на вебсайті;
- система відгуків в телеграм боті;
- перегляд складу лікарів на веб сайті;
- перегляд складу лікарів в телеграм боті;
- адміністративна панель через телеграм бота;
- адміністративна панель через вебсайт;
- повідомлення та сповіщення про нові події та зміни;
- функція пошуку та фільтрації лікарів за різними критеріями;
- підтримка різних мов;
- можливість відправлення та збереження файлів та документації;
- система сповіщень та нагадувань про важливі події;
- модуль аналітики для відстеження статистики та звітності;
- документування історії записів;
- документування історії онлайн консультацій;
- механізм забезпечення конфіденційності та безпеки даних.

Джерела і зміст вхідної інформації (даних)

В даному ПЗ джерелом вхідної інформації є користувач. Користувач має задавати вручну необхідні дані, а саме дати, локації.

Нормативно-довідкова інформація (класифікатори, довідники тощо)

Вимоги до даного пункту відсутні.

Вимоги до способів організації, збереження та ведення інформації

В якості БД обрано MySQL.

Вимоги до технічного забезпечення

Відсутні жорсткі вимоги до технічного забезпечення. Персональний комп'ютер чи ноутбук з технічними характеристиками, що підтримують будь-який сучасний браузер.

Архітектура програмної системи

Архітектура складається з клієнтської частини, серверної частини та БД.

Системне програмне забезпечення

Застосунок має бути розроблено з використанням фреймворку Laravel. В якості БД для вебзастосунку обрано MySQL.

Мережне програмне забезпечення

Для створення ПЗ використано ОС Windows 11, у якості IDE обрано PhpStorm, для перегляду вебсторінок – браузер Google Chrome.

Програмне забезпечення ведення інформаційної бази

CRUD-операції виконуються через БД MySQL.

Мова і технологія розробки ПЗ

Програмне забезпечення має розроблятися з використанням фреймворку Laravel. Мова розробки – PHP.

Інтерфейс користувача

Інтерфейс користувача повинен задовольняти всі вимоги дизайну, щоб забезпечити користувачеві максимальну зручність та ефективність в роботі з системою, скорочуючи час, необхідний для ознайомлення з її функціоналом.

Апаратний інтерфейс

Апаратним інтерфейсом є пристрій користувача (ПК чи мобільний пристрій), який буде використано для роботи з вебзастосунком.

Програмний інтерфейс

Laravel – фреймворк для розробки вебзастосунків на мові програмування PHP. Він надає зручні засоби для роботи з маршрутизацією, базами даних, автентифікацією, а також включає безліч розширень для спрощення розробки та підтримки коду.

Висновки до розділу 1

В першому розділі кваліфікаційної роботи бакалавра розглянуто вебзастосунок онлайн-клініки, який пропонує зручний підхід до запису на прийом на процедуру чи до спеціаліста.

Описано функціонал вебсайту, що спрямований на полегшення вибору лікаря та оформлення запису.

В аналізі існуючих систем також виявлено ключові аспекти та вдалі рішення, які можна врахувати під час розробки власної платформи. Проаналізовані «Онлайн клініка», «Онлайн-консультація «ОН Клиник»» та «Viva клінік» як потенційні конкуренти на ринку надання медичних послуг.

2 ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ ОНЛАЙН-КЛІНІКИ

2.1 Розробка діаграми варіантів використання

Діаграма варіантів використання (Use Case Diagram) – це графічне зображення функціональних вимог до системи, яке показує, які дії можуть виконувати актори (користувачі або системи) у взаємодії з системою. Основна мета діаграми варіантів використання - це визначення функціональних можливостей системи та її взаємодії з користувачами.

Для створення діаграми варіантів використання, спочатку ідентифікуються всі актори, які взаємодіють з системою, а потім визначаються всі можливі варіанти їх використання. Кожен варіант використання представляє собою конкретну функціональність або дію, яку може здійснити користувач або система.

Діаграма варіантів використання вебзастосунку, представлено на рисунку 2.1.

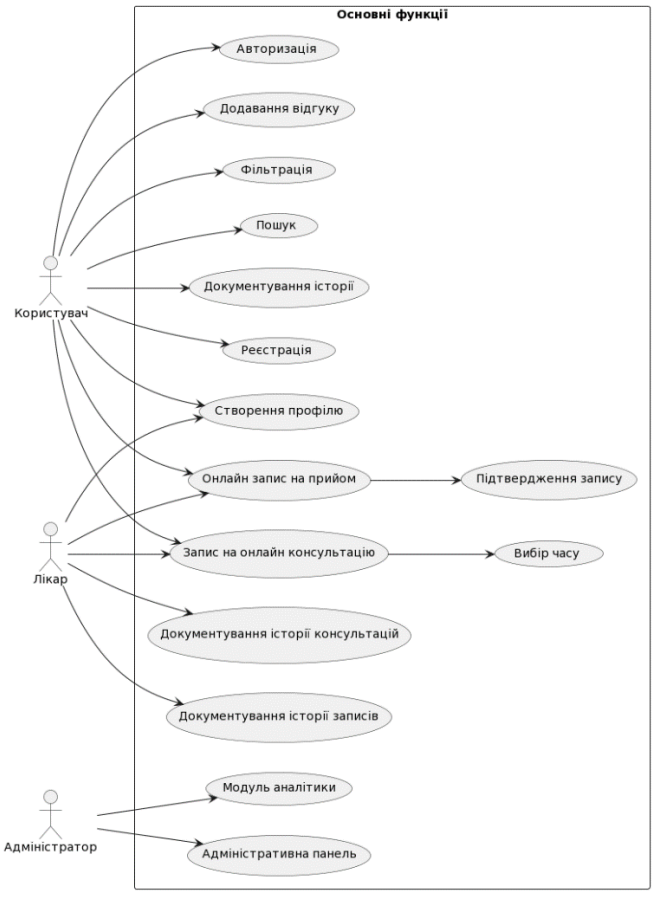

Рисунок 2.1 – Діаграма варіантів використання вебсайту

Дана діаграма варіантів використання відображує 3 актори: адміністратор, лікар та користувач.

2.2 Розробка діаграми класів

Діаграма класів є одним з основних видів діаграм у мові моделювання UML (Unified Modeling Language). Вона використовується для візуалізації структури програмного коду, зокрема класів, їх атрибутів, методів та взаємозв'язків між класами.

Основна мета діаграми класів полягає в тому, щоб показати, як класи системи пов'язані між собою і як вони співпрацюють для виконання функцій системи.

Діаграму класів для вебзастосунку онлайн-клініки, представлено на рисунку 2.2.

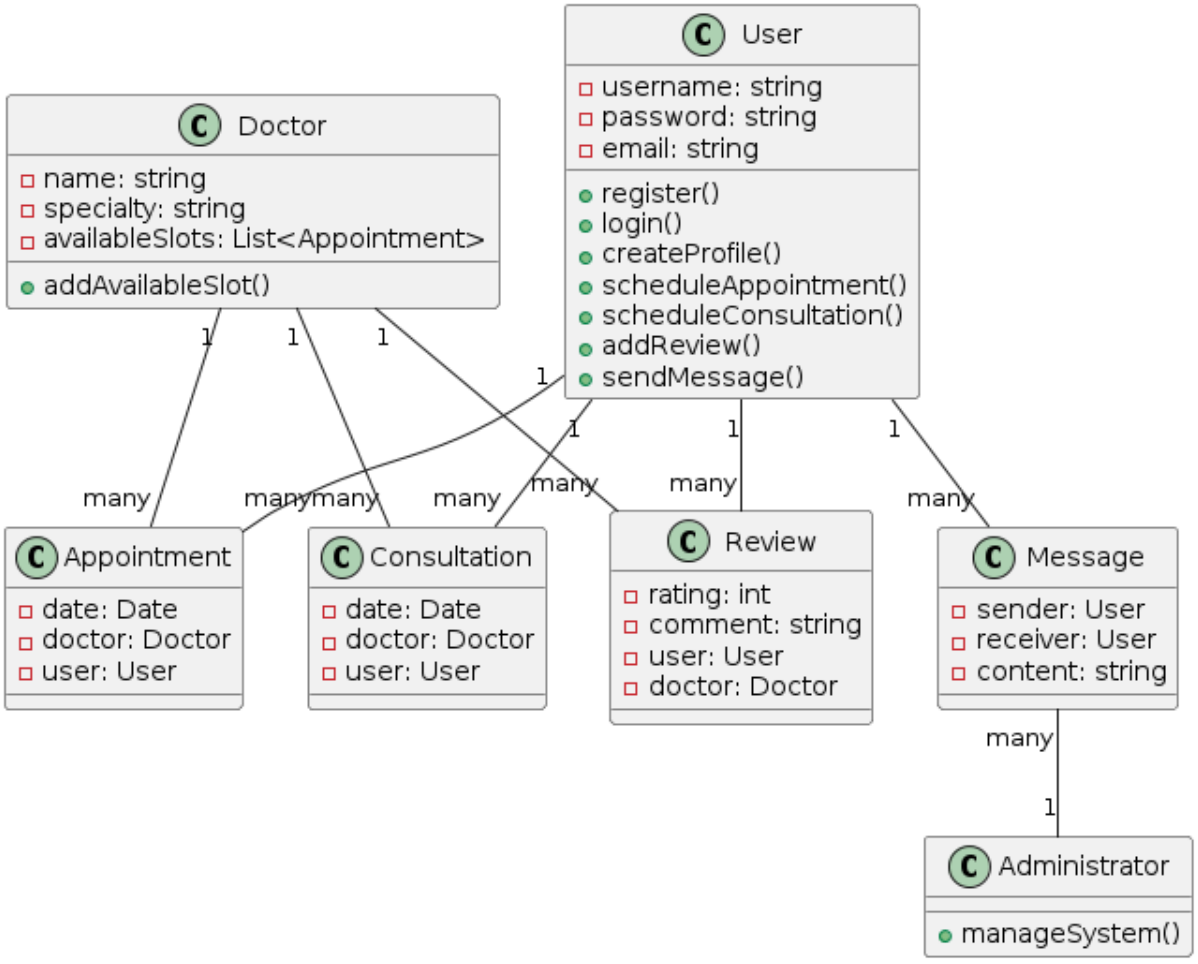

Рисунок 2.2 – Діаграма класів вебсайту

Розглянемо кожен клас окремо:

1) клас Користувача (User): цей клас буде відповідати за реєстрацію, авторизацію та профілі користувачів;

2) клас Лікаря (Doctor): даний клас буде включати дані та функціонал, що стосуються лікарів, такі як інформація про них та записи на прийоми;

3) клас Прийому (Appointment): цей клас буде відповідати за онлайн записи на прийоми;

4) клас Консультації (Consultation): для управління онлайн консультаціями;

5) клас Відгуків (Review): для системи відгуків;

6) клас Адміністратора (Administrator): для управління адміністративною панеллю;

7) клас Повідомлень (Message): для обробки та відправлення повідомлень і сповіщень;

8) клас Аналітики (Analytics): для збору та аналізу даних.

2.3 Розробка діаграми послідовностей дій системи

Діаграма послідовності – це вид діаграми в мові моделювання UML (Unified Modeling Language), який використовується для візуалізації взаємодії між об'єктами в рамках якої-небудь послідовної операції або сценарію. Вона показує порядок виконання повідомлень між об'єктами в часі.

Основна мета діаграми послідовності полягає в тому, щоб зображувати, як об'єкти взаємодіють один з одним у певній послідовності для виконання конкретного сценарію або функції. Вона допомагає уточнити послідовність подій, які відбуваються під час виконання деякої функції або операції, і визначити взаємодію між різними об'єктами в цьому процесі.

Діаграма послідовності системи представлена на рисунку 2.3.

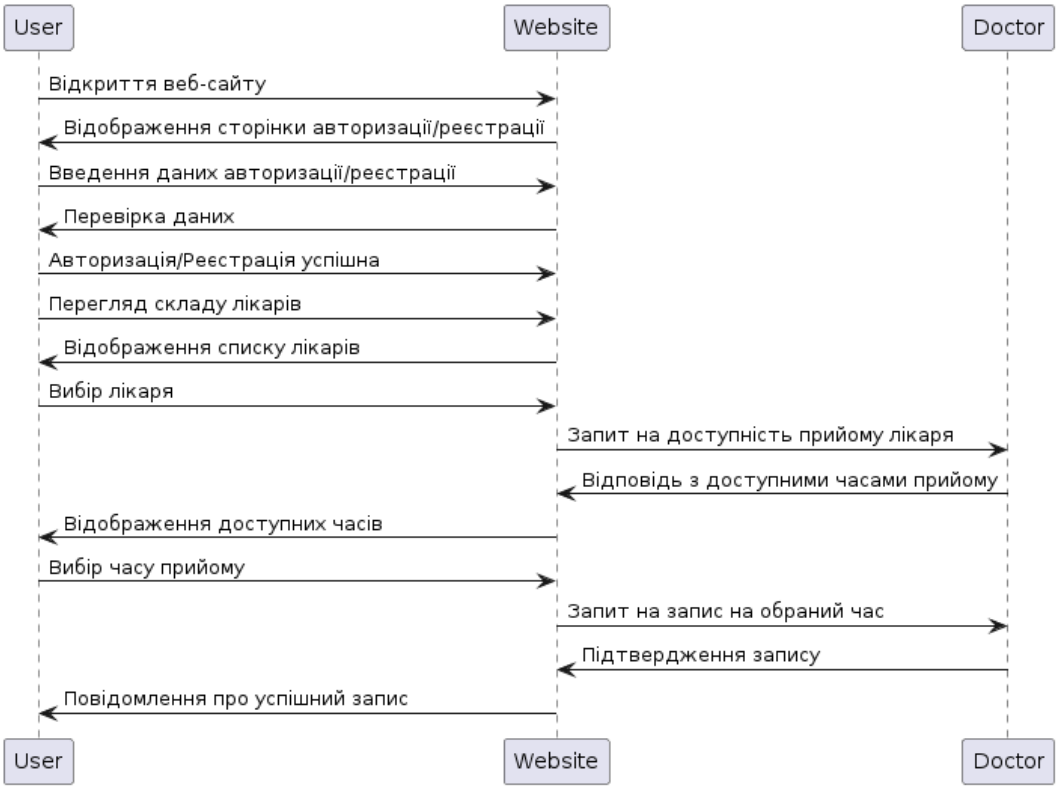

Рисунок 2.3 – Діаграма послідовності системи (див. вище)

Дана діаграма зображує послідовність дій у системі онлайн-запису на прийом до лікаря

Висновки до розділу 2

У другому розділі кваліфікаційної роботи бакалавра описано проєктування вебзастосунку онлайн-клініки. Під цією метою були розроблені та детально розглянуті наступні моделі та діаграми:

Діаграма варіантів використання системи, яка надає загальне уявлення про те, як користувачі будуть взаємодіяти з системою.

Діаграма класів системи, яка визначає основні класи програмної системи та їх взаємозв'язки.

Діаграма послідовностей, яка показує послідовність взаємодій між різними об'єктами або компонентами системи у певному процесі або сценарії.

Ці моделі та діаграми були створені для забезпечення чіткого розуміння архітектури та функціональності розробленого вебзастосунку онлайн-клініки. Вони є ключовими елементами для подальшого розвитку та реалізації проєкту.

3 КОНСТРУЮВАННЯ ВЕБЗАСТОСУНКУ ОНЛАЙН-КЛІНІКИ

3.1 Вибір технологій розробки

PHP, або Hypertext Preprocessor, є однією з найпопулярніших мов програмування, яка використовується для створення динамічних вебсайтів та вебзастосунків. Вона була розроблена Рамусом Лердорфом у 1994 році і спочатку використовувалася як набір скриптів для ведення статистики відвідувань його особистої вебсторінки. З того часу PHP перетворився на потужну, багатофункціональну мову програмування, яка є відкритим кодом і підтримується великою спільнотою розробників [4-6].

PHP забезпечує можливість інтеграції з численними базами даних, такими як MySQL, PostgreSQL, Oracle та Microsoft SQL Server, що дозволяє розробникам створювати динамічні вебзастосунки, які можуть ефективно обробляти і зберігати великі обсяги даних. PHP також підтримує численні протоколи, включаючи HTTP, POP3, IMAP, LDAP та інші, що робить його гнучким інструментом для створення різних видів вебзастосунків.

Однією з головних переваг PHP є його легкість у навчанні та використанні. Синтаксис PHP простий та інтуїтивно зрозумілий, що робить його доступним для новачків, а також ефективним для досвідчених розробників. PHP також надає широкий спектр вбудованих функцій і бібліотек, які спрощують виконання типових завдань, таких як обробка форм, управління файлами, робота з сесіями і куками, генерування зображень та PDF-документів, а також відправка електронної пошти.

PHP широко використовується у великих вебпроєктах завдяки його високій продуктивності та масштабованості. Багато відомих вебсайтів і платформ, такі як Facebook, Wikipedia, WordPress, Joomla і Drupal, побудовані на основі PHP. Це свідчить про надійність і ефективність PHP як мови програмування для веброзробки.

Розробники PHP також мають доступ до великої кількості сторонніх бібліотек і фреймворків, які розширюють функціональність мови і спрощують

процес розробки. Серед найпопулярніших фреймворків можна виділити Laravel, Symfony, CodeIgniter, Zend Framework і Yii. Ці фреймворки надають потужні інструменти для роботи з базами даних, маршрутизації, обробки запитів, автентифікації користувачів, тестування та іншого.

Laravel – це популярний фреймворк для розробки вебзастосунків на мові PHP, який був створений Тейлором Отвеллом у 2011 році. Laravel швидко завоював популярність завдяки своєму елегантному синтаксису, простоті використання і потужному набору функцій, які полегшують розробку складних вебзастосунків [10-15].

Однією з головних переваг Laravel є його модульна архітектура, яка дозволяє розробникам легко додавати і налаштовувати функціональність застосунка за допомогою різних пакетів і модулів. Laravel використовує Composer для управління залежностями, що дозволяє розробникам легко підключати сторонні бібліотеки і пакети до свого проєкту.

Laravel надає потужні інструменти для роботи з базами даних, включаючи ORM (Object-Relational Mapping) систему Eloquent. Eloquent дозволяє розробникам працювати з базами даних, використовуючи об'єктно-орієнтований підхід, що спрощує створення, читання, оновлення і видалення записів у базі даних. Eloquent також підтримує складні зв'язки між таблицями, що дозволяє розробникам ефективно працювати з реляційними базами даних.

Однією з ключових особливостей Laravel є його потужна система маршрутизації, яка дозволяє розробникам легко визначати маршрути для своїх застосунків і прив'язувати їх до відповідних контролерів і методів. Це дозволяє розробникам організувати логіку застосунка у зрозумілий і структурований спосіб.

Laravel також надає потужний механізм обробки запитів і валідації даних, що дозволяє розробникам легко обробляти і перевіряти дані, які надходять від користувачів. Це забезпечує високий рівень безпеки і надійності застосунків, розроблених на основі Laravel.

Однією з унікальних особливостей Laravel є його вбудований шаблонний движок Blade, який дозволяє розробникам легко створювати і управляти шаблонами для своїх застосунків. Blade надає зручний синтаксис для вставки PHP-коду в HTML-шаблони, що спрощує створення динамічних інтерфейсів користувача.

Blade – це вбудований шаблонний движок фреймворку Laravel, який надає розробникам зручний інструмент для створення і управління шаблонами у вебзастосунках. Blade був розроблений для спрощення процесу розробки користувацьких інтерфейсів, забезпечуючи простий і інтуїтивно зрозумілий синтаксис для роботи з HTML-шаблонами і вставки PHP-коду.

Однією з ключових переваг Blade є його легкість у використанні. Синтаксис Blade дозволяє розробникам легко вставляти PHP-код у HTML-шаблони, використовуючи спеціальні директиви, які починаються з символу «@». Наприклад, для відображення змінної у шаблоні Blade використовується директива **{{ $variable }}**, що робить код зрозумілим і читабельним.

Blade також підтримує створення компонентів і макетів, що дозволяє розробникам створювати повторно використовувані частини інтерфейсу і організовувати шаблони у зрозумілий і структурований спосіб. Компоненти Blade дозволяють розробникам інкапсулювати логіку і стиль окремих частин інтерфейсу, що спрощує їх повторне використання у різних частинах застосунка.

Однією з унікальних особливостей Blade є його підтримка інклюзії шаблонів, що дозволяє розробникам включати в один шаблон інші шаблони або частини шаблонів. Це дозволяє створювати складні і гнучкі шаблони, які легко підтримувати і оновлювати.

Blade також підтримує секції і стопки, що дозволяє розробникам визначати і управляти різними частинами шаблонів, які можуть бути заповнені або перевизначені у дочірніх шаблонах. Це дозволяє створювати гнучкі і розширювані шаблони, які легко адаптувати до різних вимог і сценаріїв.

Tailwind CSS – це сучасний CSS-фреймворк, який базується на концепції «utility-first» і дозволяє розробникам швидко створювати стильні і динамічні

користувацькі інтерфейси для вебзастосунків. Tailwind CSS був розроблений Адамом Ватаном і Джонатаном Ренном у 2017 році і швидко здобув популярність завдяки своїй гнучкості і потужності.

Основною ідеєю Tailwind CSS є використання утилітарних класів для стилізації елементів інтерфейсу, що дозволяє розробникам створювати складні дизайни без написання власного CSS-коду. Tailwind CSS надає широкий набір утилітарних класів для різних аспектів стилізації, таких як розмітка, типографіка, кольори, межі, тіні, анімації і багато іншого.

Tailwind CSS дозволяє розробникам створювати повністю адаптивні інтерфейси, використовуючи утилітарні класи для визначення різних стилів для різних розмірів екранів і пристроїв. Це забезпечує високий рівень гнучкості і контролю над зовнішнім виглядом і поведінкою інтерфейсу на різних пристроях.

Однією з ключових переваг Tailwind CSS є його простота у використанні і інтеграції з іншими інструментами і фреймворками. Tailwind CSS легко інтегрується з популярними фреймворками для фронтенд-розробки, такими як React, Vue.js, Angular і інші, що дозволяє розробникам використовувати його разом з улюбленими інструментами і технологіями.

Tailwind CSS також надає потужний інструмент для налаштування і розширення утилітарних класів, що дозволяє розробникам легко адаптувати його до своїх потреб і вимог. Розробники можуть створювати власні утилітарні класи, змінювати існуючі класи і додавати нові стилі, використовуючи конфігураційний файл Tailwind CSS.

Однією з унікальних особливостей Tailwind CSS є його підтримка компонентного підходу до розробки інтерфейсів. Це дозволяє розробникам створювати повторно використовувані компоненти інтерфейсу, які можна легко інтегрувати у різні частини застосунка. Компоненти Tailwind CSS можуть включати утилітарні класи і бути налаштовані для різних сценаріїв використання.

MySQL – це система керування реляційними базами даних (СУБД), яка широко використовується у веброзробці для зберігання і управління даними. MySQL була розроблена компанією MySQL AB у 1995 році і з того часу стала

однією з найпопулярніших СУБД у світі. MySQL є відкритим кодом і підтримується великою спільнотою розробників [18].

MySQL надає потужні інструменти для роботи з базами даних, включаючи мову запитів SQL (Structured Query Language), яка дозволяє розробникам створювати, читати, оновлювати і видаляти записи у базі даних. SQL є стандартною мовою запитів для роботи з реляційними базами даних і широко використовується у веброзробці [19].

Однією з ключових переваг MySQL є його висока продуктивність і масштабованість. MySQL може ефективно обробляти великі обсяги даних і підтримує багатокористувацькі застосунки з високою навантаженістю. MySQL також надає потужні інструменти для оптимізації продуктивності, включаючи індекси, кешування і розділення таблиць.

MySQL підтримує численні типи даних, включаючи числові, рядкові, датові і булеві типи, що дозволяє розробникам зберігати і обробляти різні види даних. MySQL також підтримує складні запити, включаючи об'єднання, вкладені запити і агрегацію, що дозволяє розробникам виконувати складні операції з даними.

MySQL забезпечує високу надійність і безпеку даних. MySQL підтримує транзакції, що дозволяє розробникам виконувати групи операцій як єдине ціле, забезпечуючи цілісність даних. MySQL також надає потужні інструменти для резервного копіювання і відновлення даних, що забезпечує захист даних від втрат і пошкоджень [23].

OpenServer Panel – це програмне забезпечення для локальної розробки вебзастосунків, яке надає зручний інтерфейс для управління сервером і налаштування вебсервера, бази даних та інших компонентів. OpenServer Panel був розроблений для спрощення процесу налаштування і управління локальним середовищем розробки для вебзастосунків.

Однією з ключових переваг OpenServer Panel є його простота у використанні і налаштуванні. OpenServer Panel надає зручний графічний інтерфейс, який дозволяє розробникам легко налаштовувати і управляти різними компонентами локального сервера, такими як Apache, Nginx, MySQL, PHP та інші. Це дозволяє

розробникам швидко налаштувати середовище розробки без необхідності виконання складних команд і налаштувань.

OpenServer Panel також підтримує численні версії PHP, MySQL і інших компонентів, що дозволяє розробникам легко змінювати конфігурацію середовища розробки відповідно до своїх потреб і вимог. Це забезпечує високу гнучкість і адаптивність середовища розробки для різних проєктів і сценаріїв.

OpenServer Panel також надає потужні інструменти для управління базами даних, включаючи інтеграцію з популярними інструментами для управління базами даних, такими як phpMyAdmin. Це дозволяє розробникам легко створювати, налаштовувати і управляти базами даних безпосередньо з інтерфейсу OpenServer Panel.

OpenServer Panel також підтримує інтеграцію з іншими інструментами і сервісами, що дозволяє розробникам легко підключати сторонні бібліотеки і інструменти до свого проєкту. Це забезпечує високу гнучкість і розширюваність середовища розробки, що дозволяє розробникам використовувати улюблені інструменти і технології для розробки вебзастосунків.

OpenServer Panel також надає потужні інструменти для моніторингу і відстеження продуктивності сервера, що дозволяє розробникам виявляти і виправляти проблеми з продуктивністю і оптимізувати роботу застосунка. Це забезпечує високу надійність і продуктивність вебзастосунків, розроблених у локальному середовищі.

Загалом, використання OpenServer Panel спрощує процес розробки вебзастосунків, надаючи розробникам зручне і потужне середовище для налаштування, управління і моніторингу локального сервера і компонентів вебзастосунка.

3.2 Моделювання бази даних

Фізична модель бази даних є найдетальнішим рівнем моделювання бази даних, де визначаються конкретні структури зберігання даних, такі як таблиці, індекси, та інші об'єкти бази даних, які будуть використовуватися у реальній

системі. Вона описує, як дані будуть фізично зберігатися в базі даних, враховуючи апаратні та програмні обмеження. Розглянемо фізичну модель бази даних для онлайн-клініки, яка представлена на рисунку 3.1.

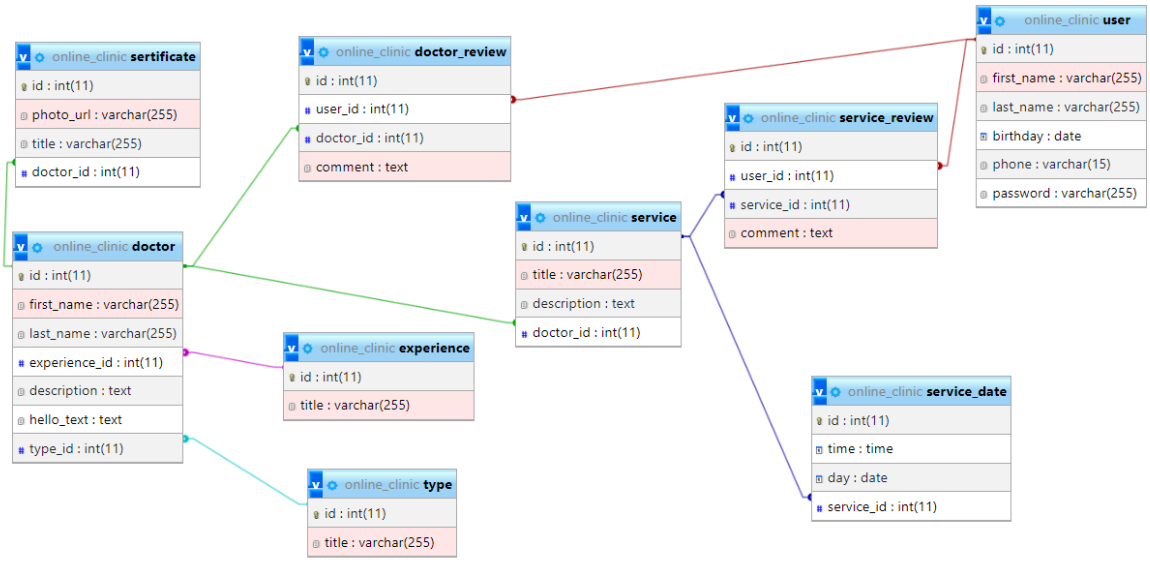

Рисунок 3.1 – Фізична модель бази даних

Структуру таблиці experience представлено на рисунку 3.2

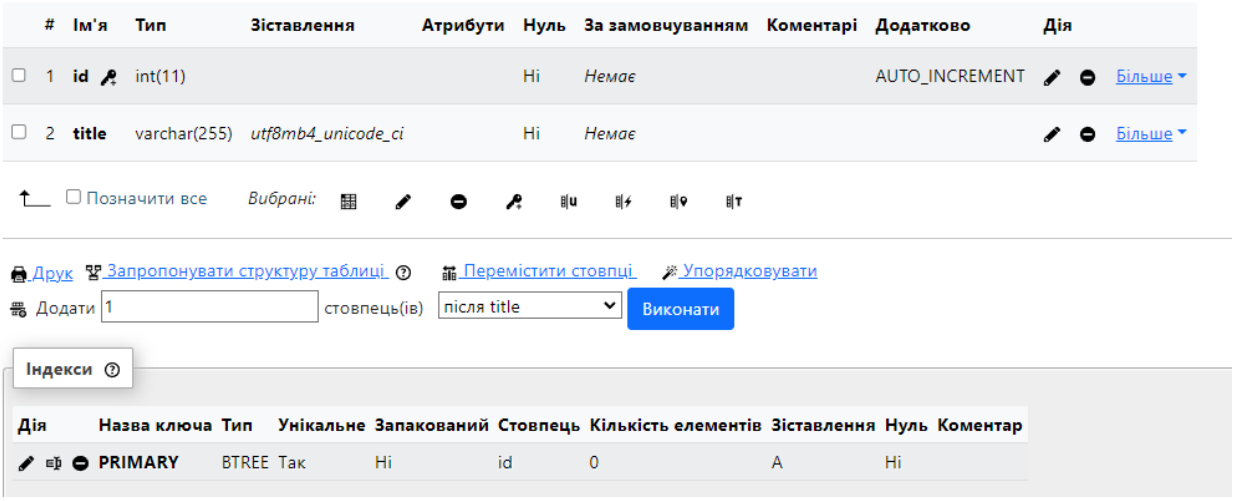

Рисунок 3.2 – Структура таблиці experience

Таблиця experience:

– id (INT, PRIMARY KEY, AUTO_INCREMENT);
– title (VARCHAR(255)).

Структура таблиці type представлена на рисунку 3.3.

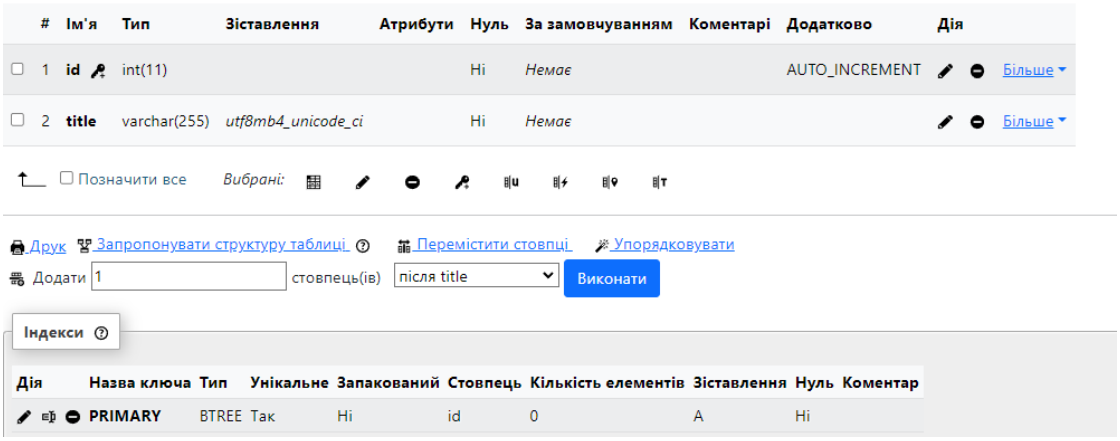

Рисунок 3.3 – Структура таблиці type

Таблиця type:

– id (INT, PRIMARY KEY, AUTO_INCREMENT);
– title (VARCHAR(255)).

Структура таблиці doctor представлена на рисунку 3.4.

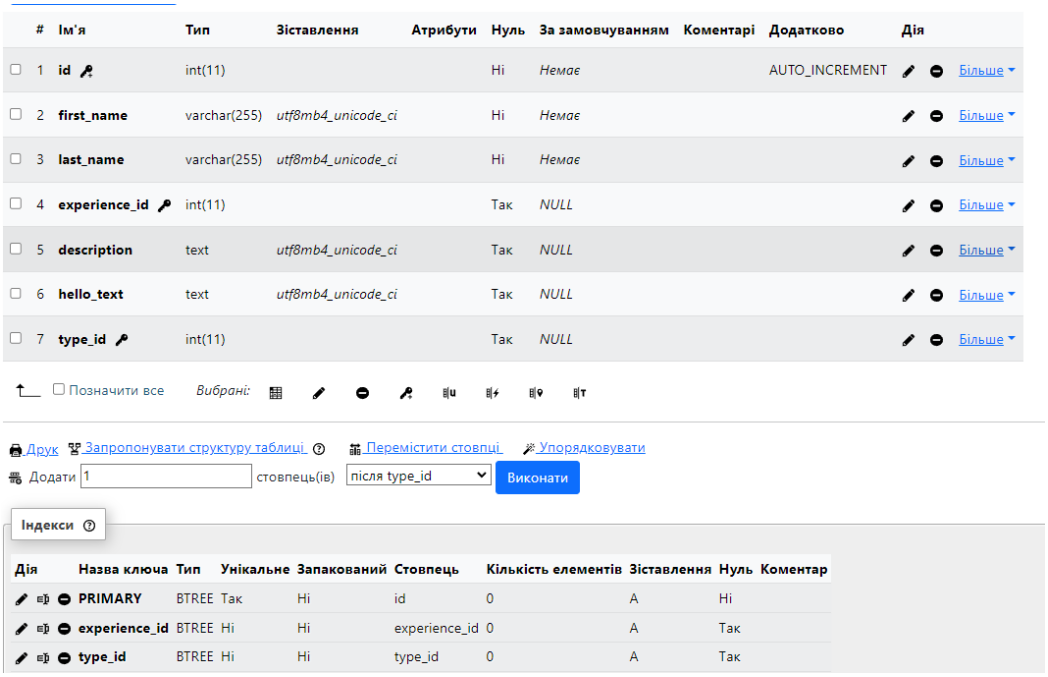

Рисунок 3.4 – Структура таблиці doctor

Таблиця doctor:

– id (INT, PRIMARY KEY, AUTO_INCREMENT);
– first_name (VARCHAR(255));
– last_name (VARCHAR(255));
– experience_id (INT, FOREIGN KEY REFERENCES experience(id));

– description (TEXT);
– hello_text (TEXT);
– type_id (INT, FOREIGN KEY REFERENCES type(id)).

Структура таблиці sertificate представлена на рисунку 3.5.

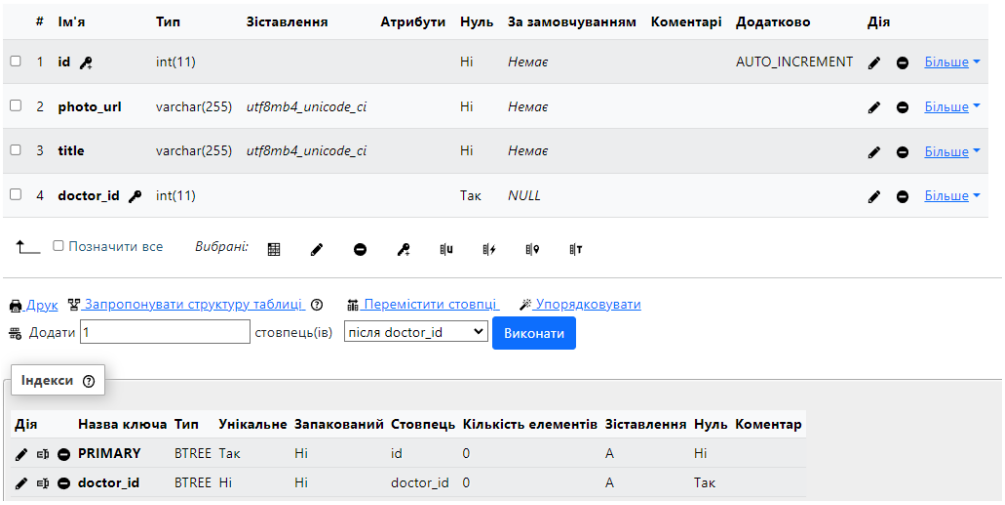

Рисунок 3.5 – Структура таблиці sertificate

Таблиця sertificate:

– id (INT, PRIMARY KEY, AUTO_INCREMENT);
– photo_url (VARCHAR(255));
– title (VARCHAR(255));
– doctor_id (INT, FOREIGN KEY REFERENCES doctor(id)).

Структура таблиці service представлена на рисунку 3.6.

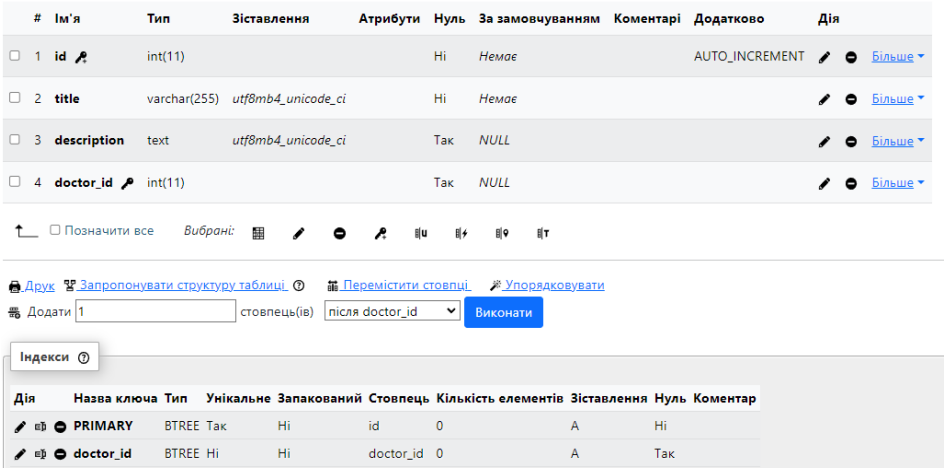

Рисунок 3.6 – Структура таблиці service

Таблиця service:

– id (INT, PRIMARY KEY, AUTO_INCREMENT);
– title (VARCHAR(255));
– description (TEXT);
– doctor_id (INT, FOREIGN KEY REFERENCES doctor(id)).

Структура таблиці service_date представлена на рисунку 3.7.

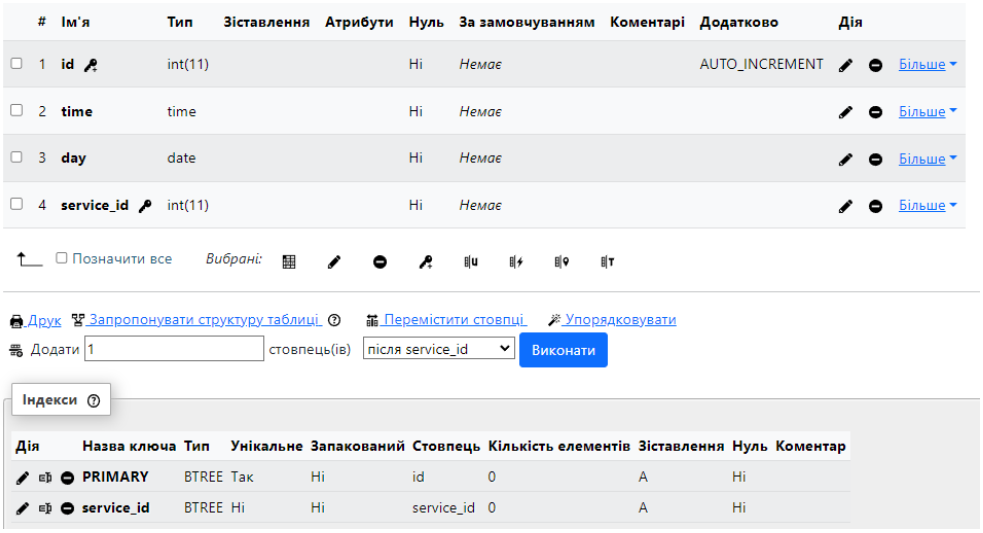

Рисунок 3.7 – Структура таблиці service_date

Таблиця service_date:

– id (INT, PRIMARY KEY, AUTO_INCREMENT);
– time (TIME);
– day (DATE).

Структура таблиці user представлена на рисунку 3.8.

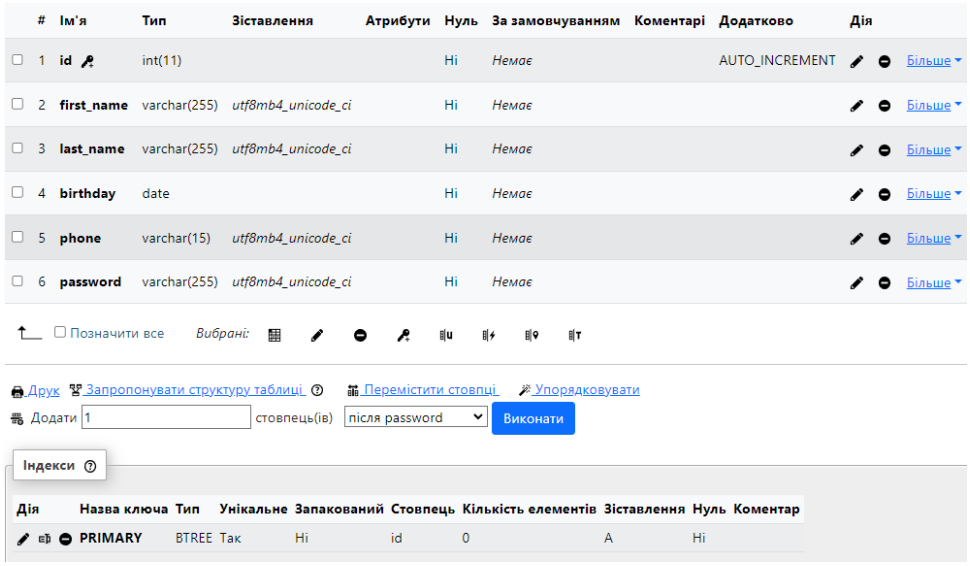

Рисунок 3.8 – Структура таблиці user

Таблиця user:

– id (INT, PRIMARY KEY, AUTO_INCREMENT);
– first_name (VARCHAR(255));
– last_name (VARCHAR(255));
– birthday (DATE);
– phone (VARCHAR(15));
– password (VARCHAR(255)).

Структура таблиці service_review представлена на рисунку 3.9.

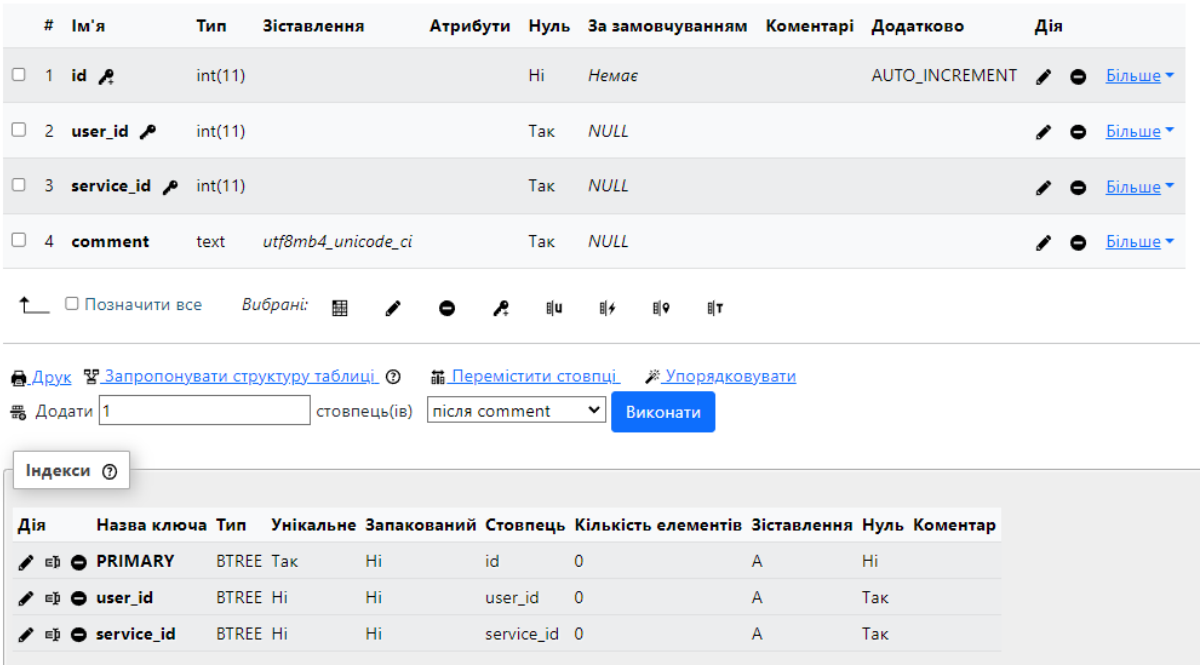

Рисунок 3.9 – Структура таблиці service_review

Таблиця service_review:

– id (INT, PRIMARY KEY, AUTO_INCREMENT);
– user_id (INT, FOREIGN KEY REFERENCES user(id));
– service_id (INT, FOREIGN KEY REFERENCES service(id));
– comment (TEXT).

Структура таблиці doctor_review представлена на рисунку 3.10.

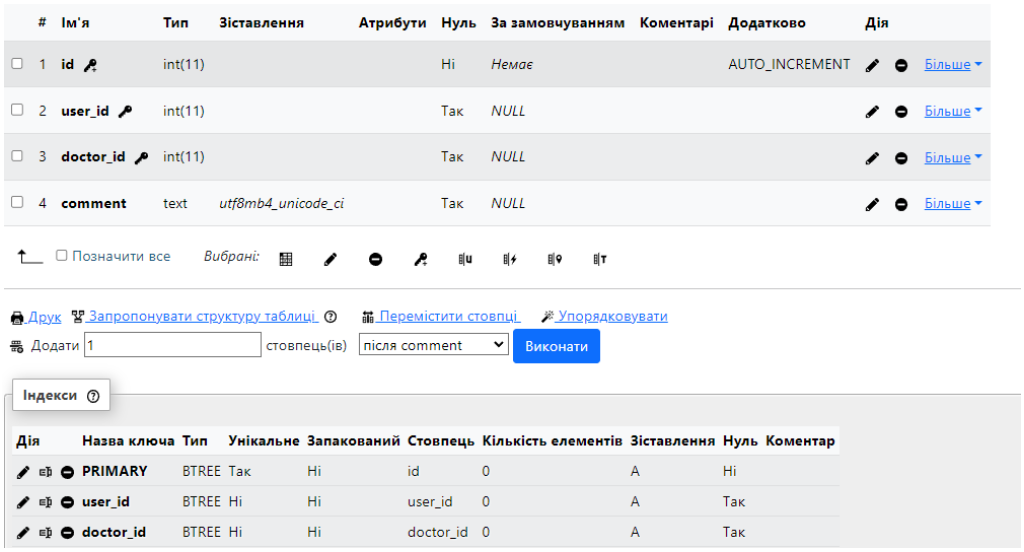

Рисунок 3.10 – Структура таблиці doctor_review

Таблиця doctor_review:

– id (INT, PRIMARY KEY, AUTO_INCREMENT);
– user_id (INT, FOREIGN KEY REFERENCES user(id));
– doctor_id (INT, FOREIGN KEY REFERENCES doctor(id));
– comment (TEXT).

3.3 Розробка мокапу системи

Мокап системи - це імітація вигляду та функціональності програмного забезпечення, що створюється. Він служить інструментом для візуалізації та концептуалізації інтерфейсу користувача, функцій та потоків роботи системи.

Під час проєктування вебзастосунку онлайн-клініки розроблено наступні мокапи сторінок:

– головної сторінки застосунку;
– сторінки про клініку;
– сторінки про послуги;
– сторінки зі списком лікарів;
– сторінки контактів;
– сторінки відгуків;
– сторінки профілю лікаря;

– сторінки профілю пацієнта;

– сторінки запису пацієнта.

Мокап головної сторінки застосунку, представлено на рисунку 3.11.

Рисунок 3.11 – Мокап головної сторінки застосунку

Мокап сторінки про клініку, представлено на рисунку 3.12.

Рисунок 3.12 – Мокап сторінки про клініку

Мокап сторінки про послуги, представлено на рисунку 3.13.

Рисунок 3.13 – Мокап сторінки про послуги

Мокап сторінки зі списком лікарів, представлено на рисунку 3.14.

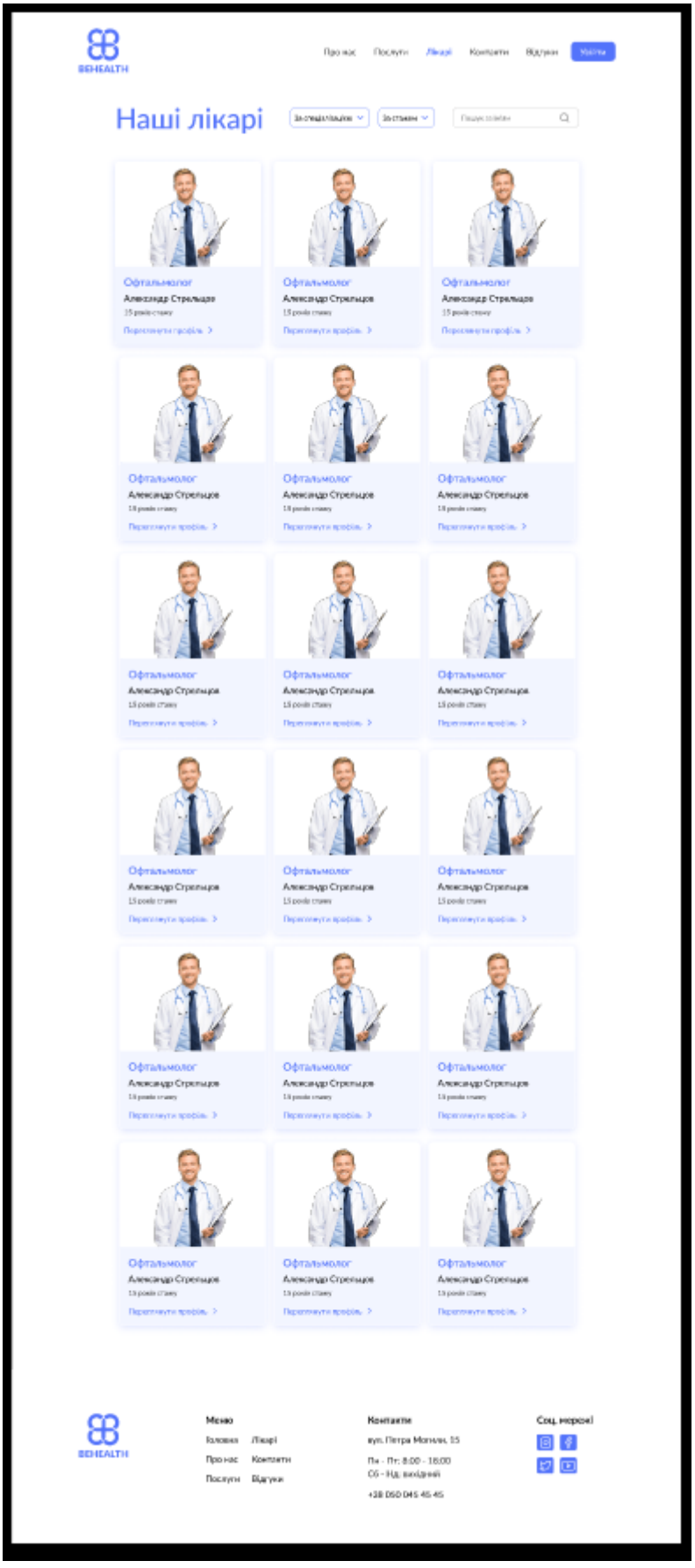

Рисунок 3.14 – Мокап сторінки зі списком лікарів

Мокап сторінки контактів, представлено на рисунку 3.15.

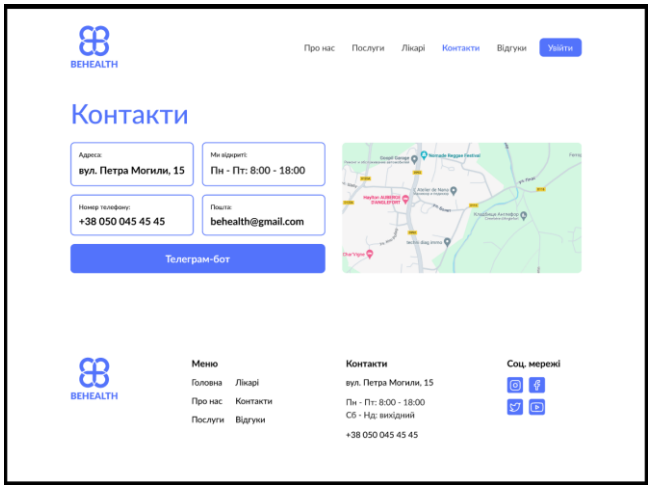

Рисунок 3.15 – Мокап сторінки контактів

Мокап сторінки відгуків, представлено на рисунку 3.16.

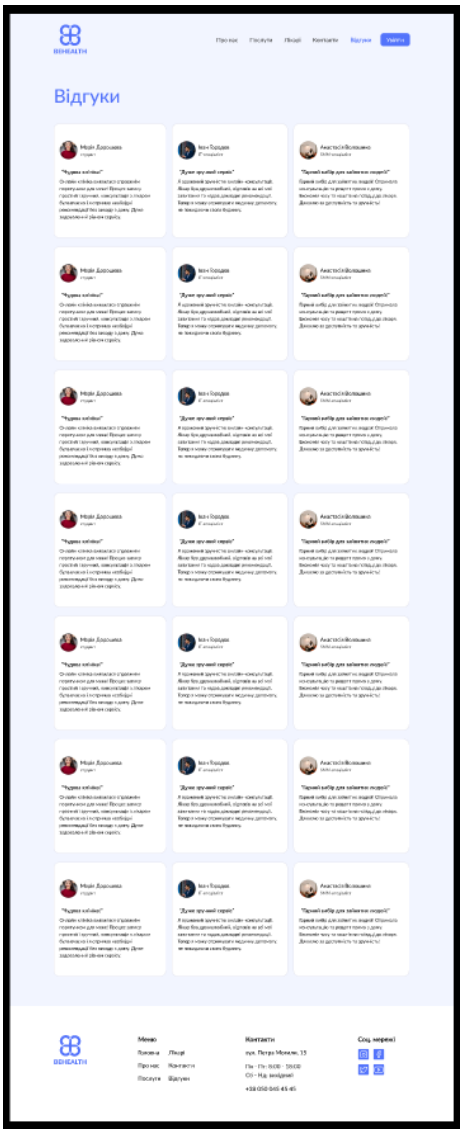

Рисунок 3.16 – Мокап сторінки відгуків

Мокап сторінки профілю лікаря, представлено на рисунку 3.17.

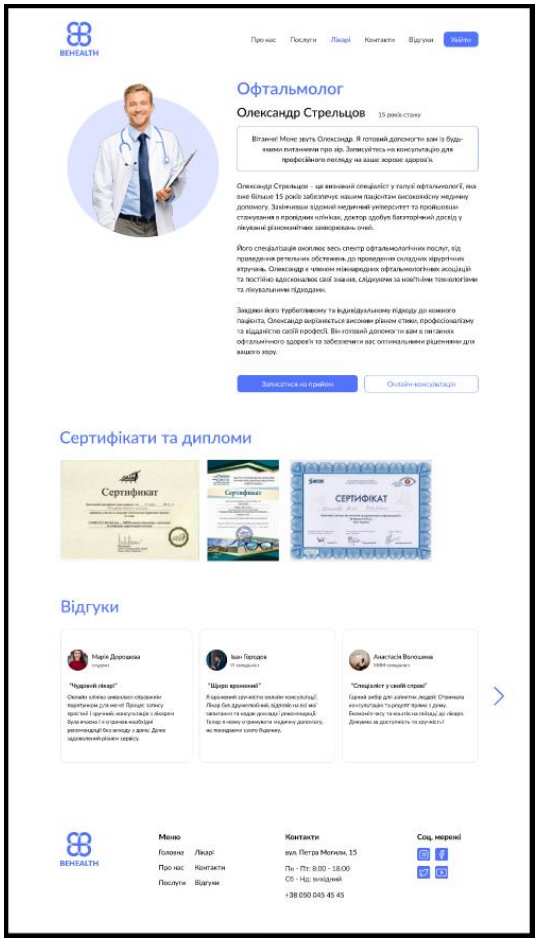

Рисунок 3.17 – Мокап сторінки профілю лікаря

Мокап сторінки профілю пацієнта, представлено на рисунку 3.18.

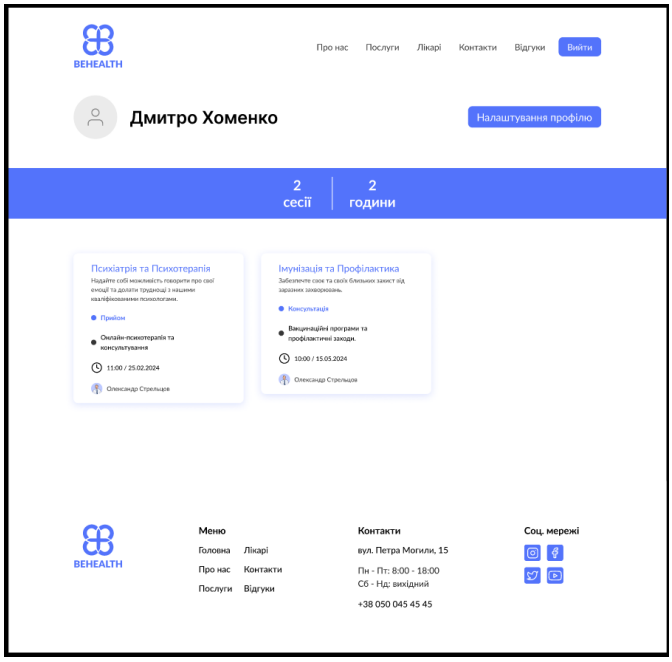

Рисунок 3.18 – Мокап сторінки профілю пацієнта

Мокап сторінки запису пацієнта, представлено на рисунку 3.19.

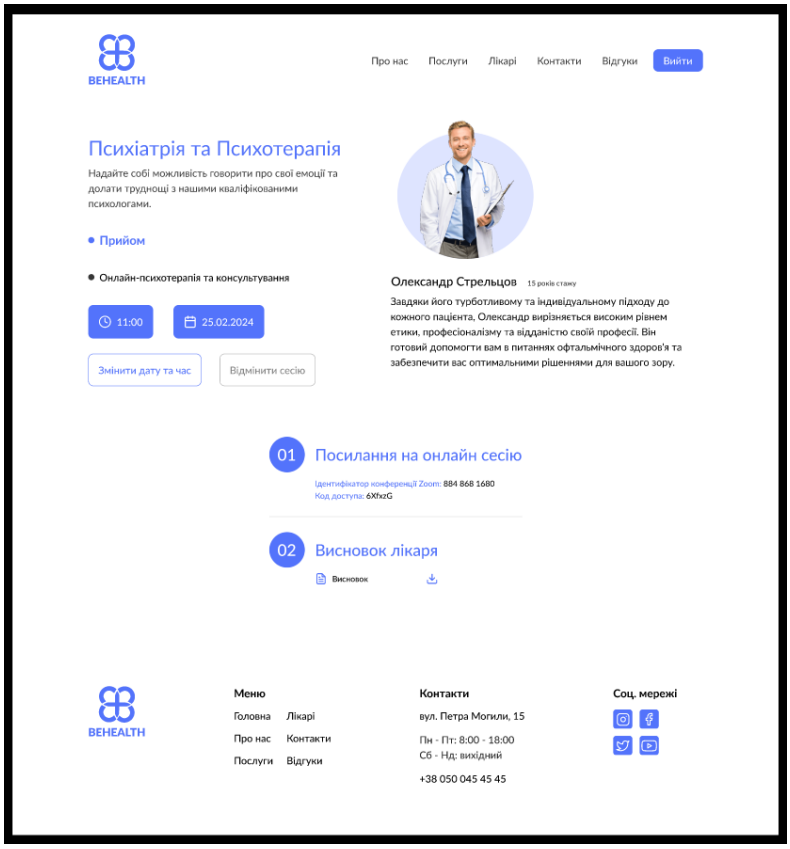

Рисунок 3.19 – Мокап сторінки запису пацієнта

Дані мокапи допоможуть розробити більш структурований та детальний вебзастосунок онлайн-клініки.

Висновки до розділу 3

У третьому розділі представлено опис технологій розробки вебзастосунку онлайн-клініки. Також спроєктовано та представлено модель та структуру бази даних.

Розроблено мокапи сторінок вебсайту, які відображають вигляд та інтерфейс вебсайту онлайн-клініки. Вони надають візуальне уявлення про те, як вебсайт буде виглядати для користувачів та як вони будуть взаємодіяти з різними його частинами.

4 РЕЗУЛЬТАТИ РОЗРОБКИ

4.1 Реалізація вебзастосунку

Під час проєктування вебзастосунку онлайн-клініки розроблено наступні сторінки:

– головна сторінка застосунку;
– сторінка про клініку;
– сторінка про послуги;
– сторінка зі списком лікарів;
– сторінка контактів;
– сторінка відгуків;
– сторінка профілю лікаря;
– сторінка профілю пацієнта;
– сторінка запису пацієнта.

На рисунку 4.1 представлено карту сторінок вебзастосунку.

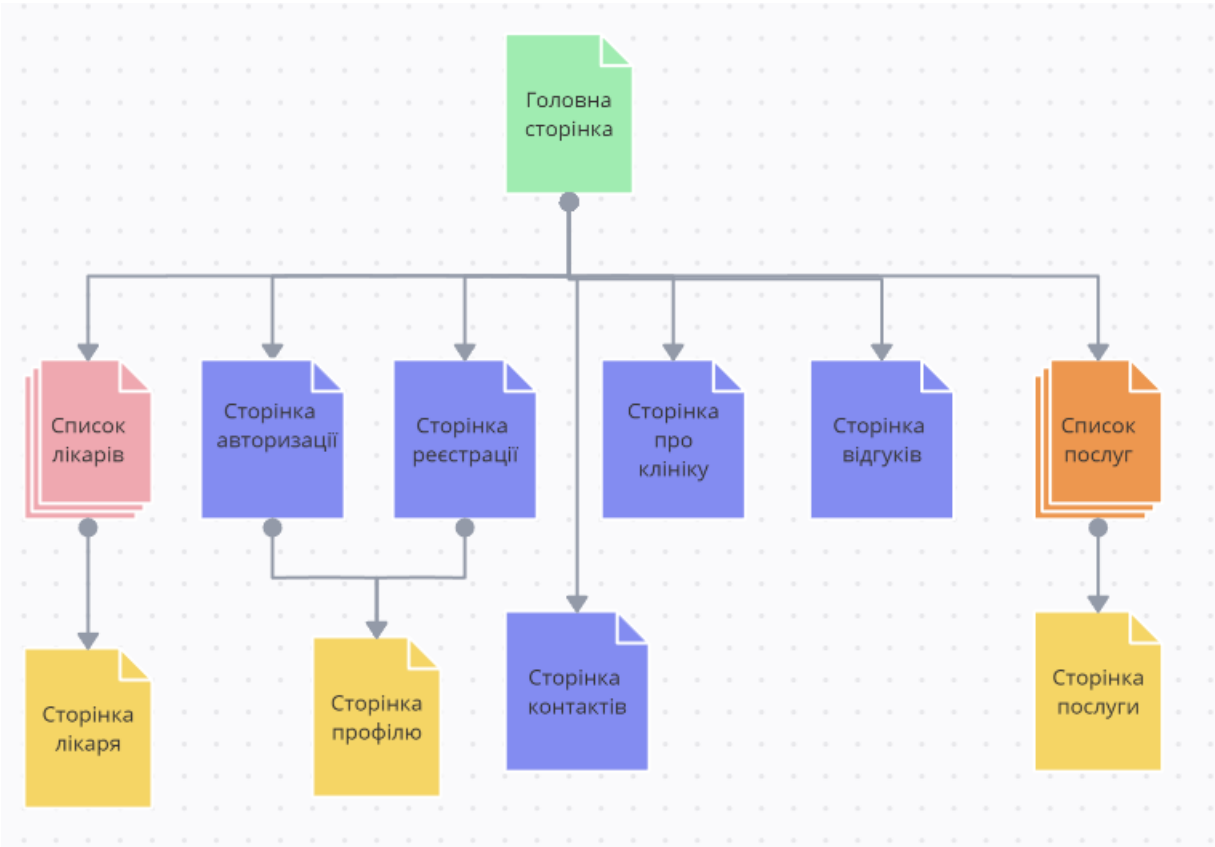

Рисунок 4.1 – Карта сторінок вебзастосунку

На головній сторінці ви можете знайти основні функції нашого застосунку. Тут ви зможете ознайомитися з останніми новинами клініки, отримати доступ до основних розділів, таких як «Послуги», «Лікарі», «Контакти» та інші.

Головну сторінку застосунку, представлено на рисунку 4.2.

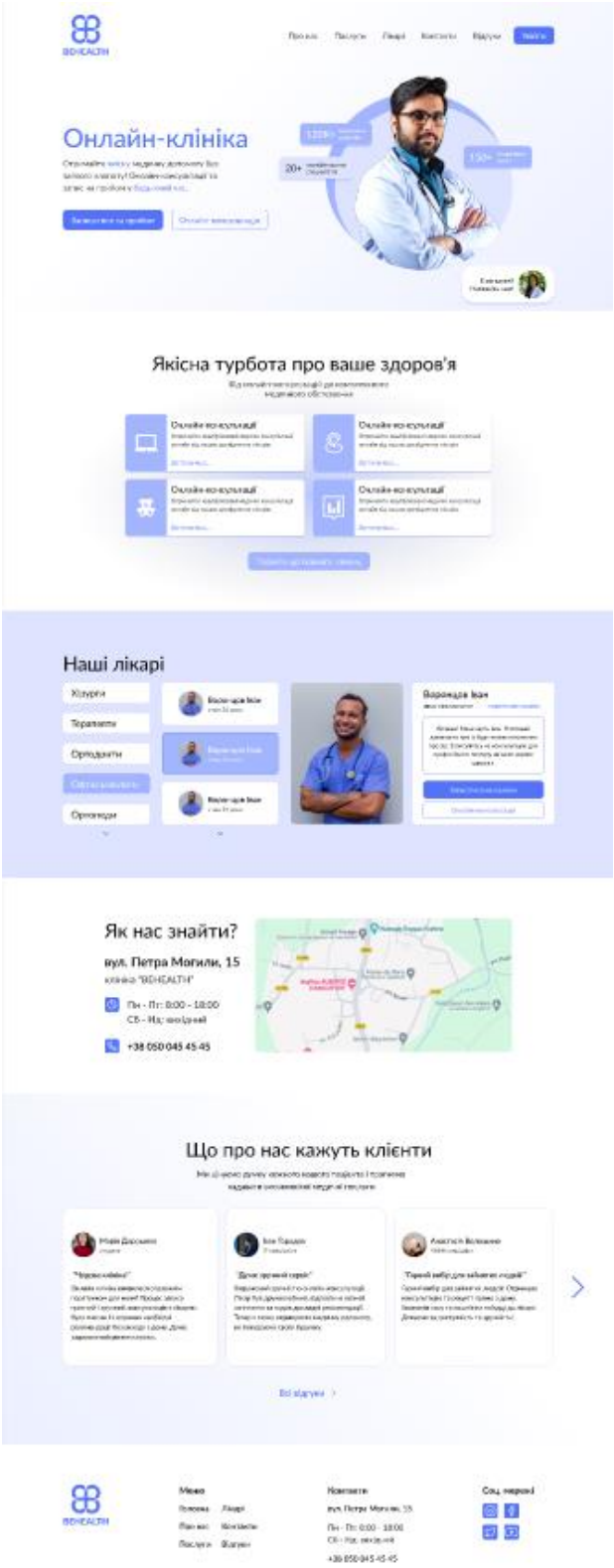

Рисунок 4.2 – Головна сторінка застосунку

На сторінці про клініку представлена докладна інформація про онлайн-клініку. Ви можете дізнатися про історію установи, місію та цінності. Також розміщені фото та відеоматеріали, які допоможуть краще зрозуміти роботу клінки.

Сторінка про клініку, представлена на рисунку 4.3.

Рисунок 4.3 – Сторінка про клініку

На сторінці послуг можна знайти список усіх послуг, що надаються клінікою. Для кожної послуги вказано детальний опис, включаючи ціни та тривалість. Користувач можете записатися на будь-яку послугу, натиснувши відповідну кнопку.

Сторінку про послуги, представлено на рисунку 4.4.

Рисунок 4.4 – Сторінка про послуги

На сторінці «Лікарі» можна знайти повний список лікарів, які працюють в онлайн-клініці. Для кожного лікаря надана коротка інформація, включаючи ім'я, спеціалізацію та досвід. Натиснувши на ім'я лікаря, користувач перейде до його профілю, де знайде більш детальну інформацію.

Сторінка зі списком лікарів, представлена на рисунку 4.5.

Рисунок 4.5 – Сторінка зі списком лікарів

На сторінці контактів користувач знайде всі необхідні контактні дані клініки, включаючи адресу, номер телефону та електронну пошту.

Також тут розміщена карта, яка допоможе знайти місцерозташування клініки. Якщо у користувача виникнуть запитання або пропозиції, він можете скористатися формою для зворотного зв'язку чи телеграм ботом.

Сторінка контактів, представлена на рисунку 4.6.

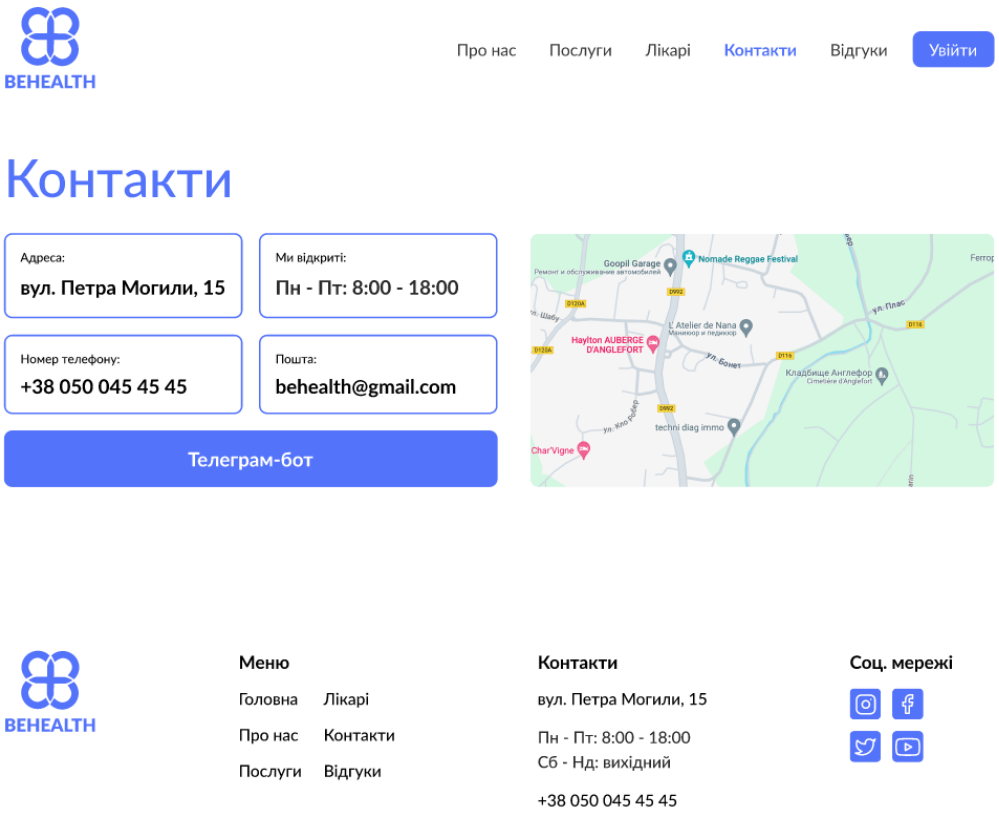

Рисунок 4.6 – Сторінка контактів

На сторінці відгуків пацієнти залишають свої відгуки про клініку та лікарів. Користувач може переглядати відгуки, сортувати їх за різними критеріями та залишати власні.

Сторінка відгуків, представлена на рисунку 4.7.

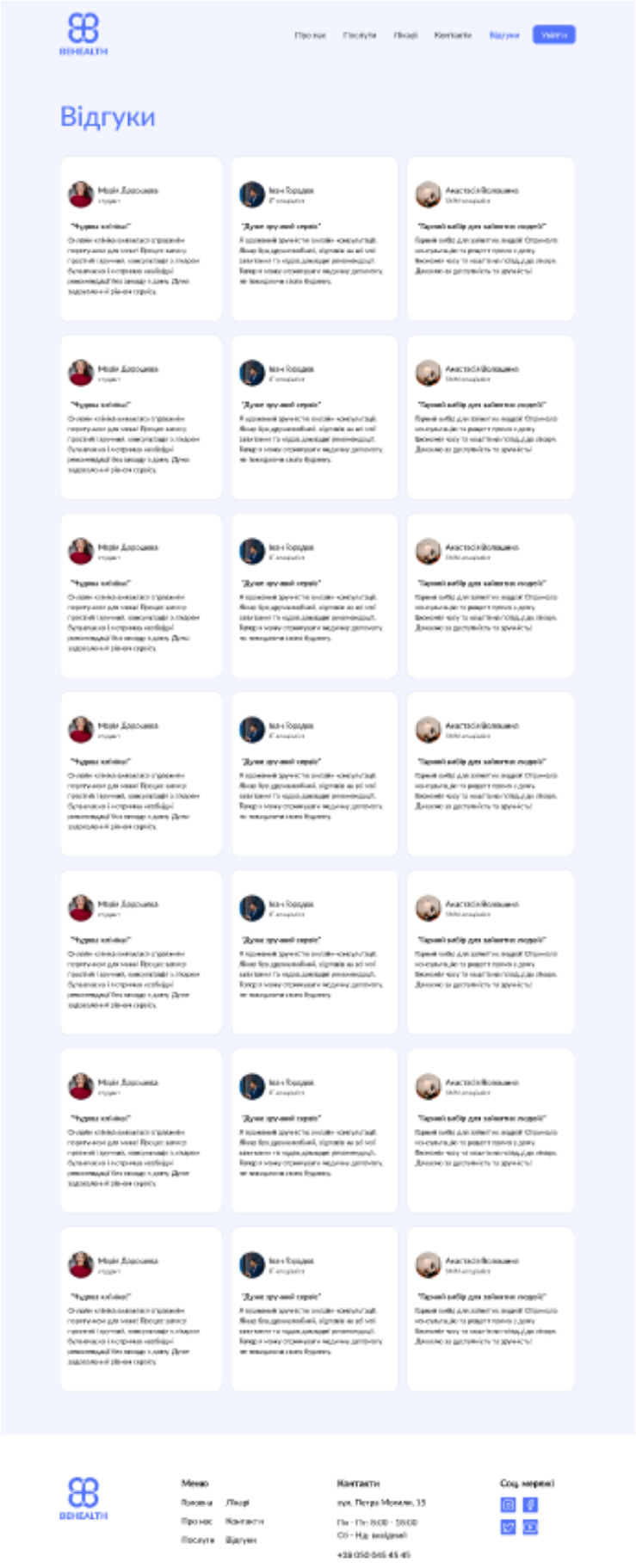

Рисунок 4.7 – Сторінка відгуків

Сторінка профілю лікаря містить детальну інформацію про кожного лікаря, включаючи біографію, освіту та сертифікати. Тут користувач також знайде фото лікаря, графік його прийому та можливість записатися на прийом.

Сторінка профілю лікаря, представлена на рисунку 4.8.

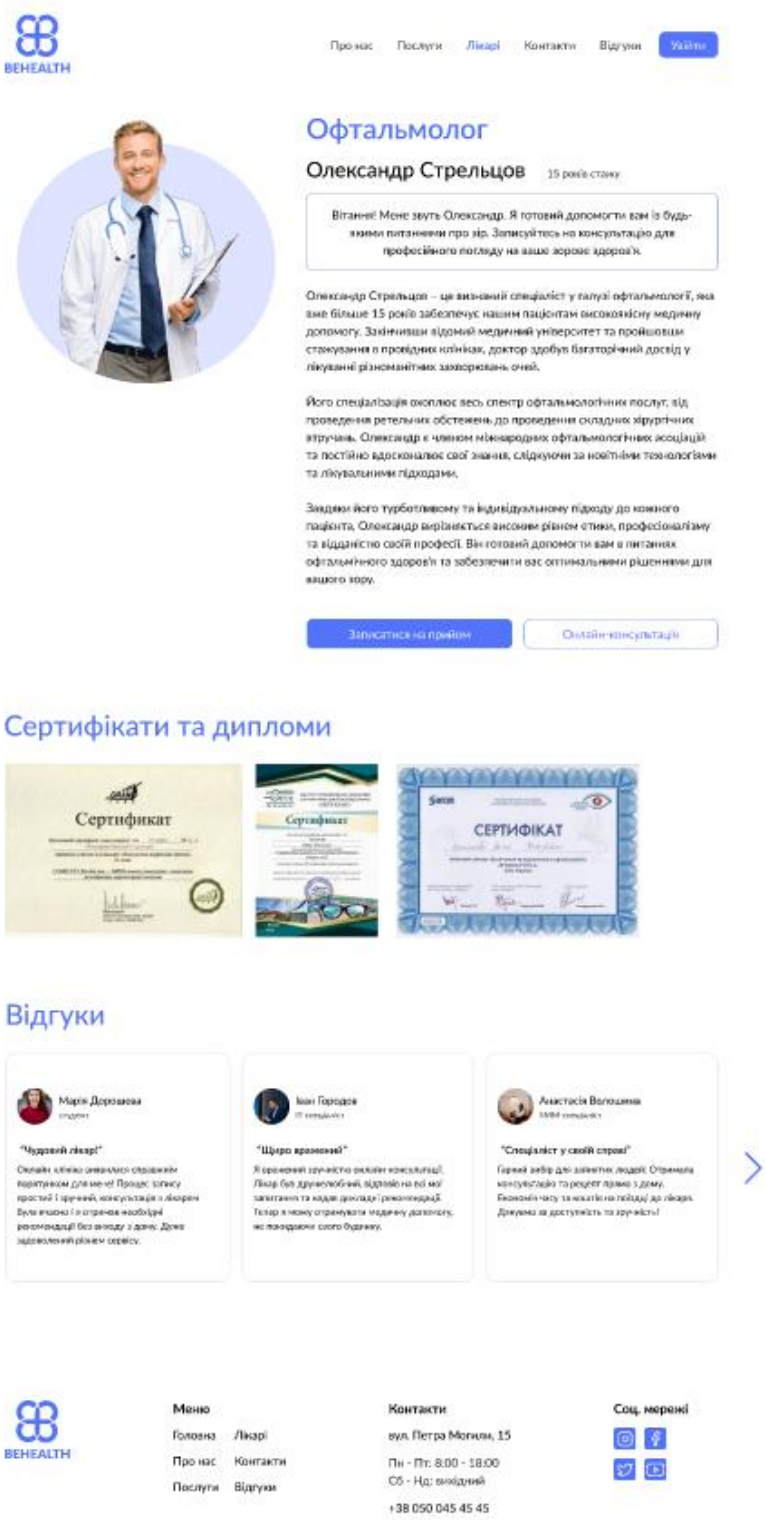

Рисунок 4.8 – Сторінка профілю лікаря

На сторінці профілю пацієнта зберігається його особиста інформація, така як контактні дані та медична історія. Користувач може переглядати свої записи на прийом, історію відвідувань та результати аналізів.

Сторінка профілю пацієнта, представлена на рисунку 4.9.

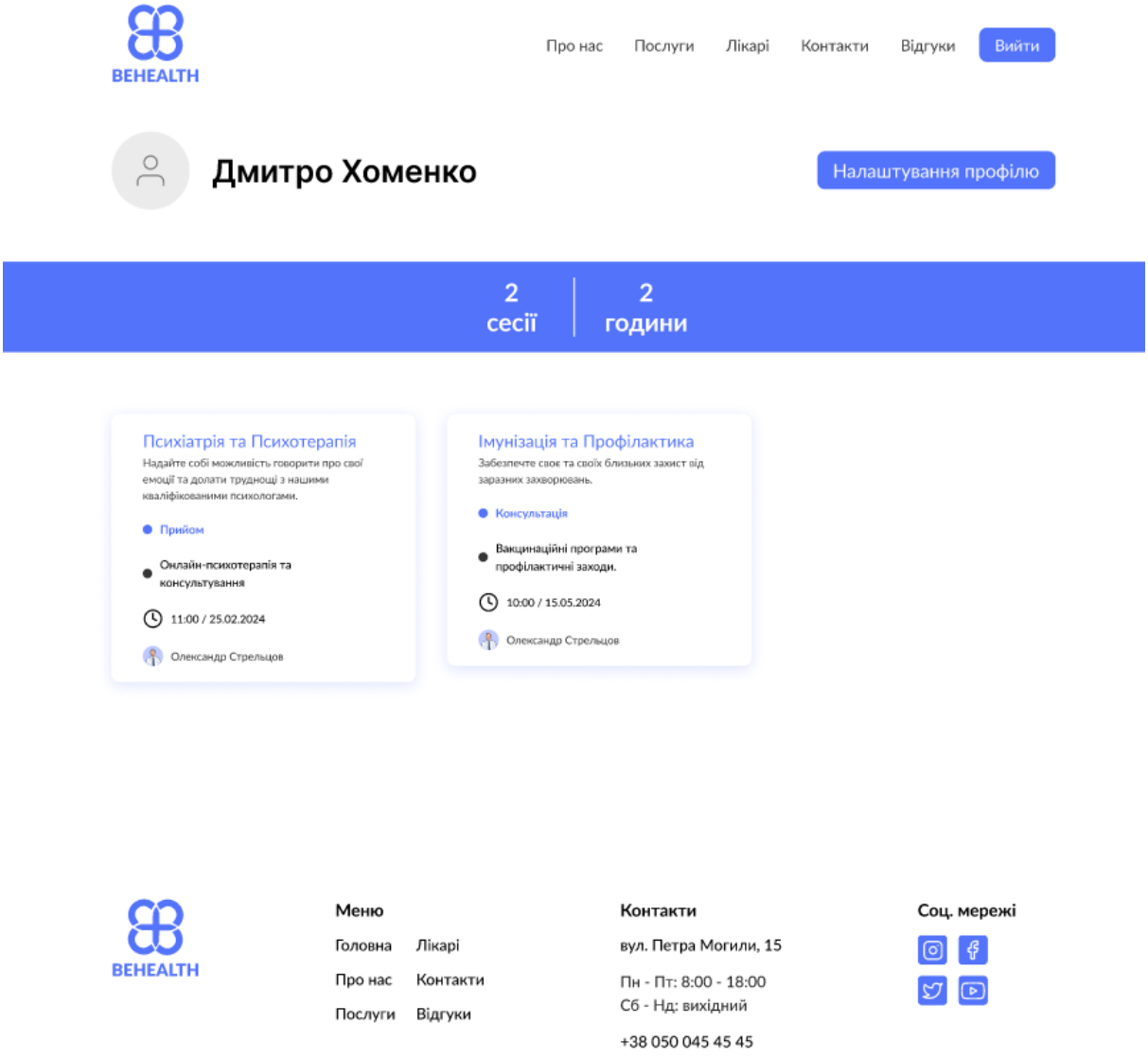

Рисунок 4.9 – Сторінка профілю пацієнта

Сторінка запису дозволяє користувачу записатися на прийом до лікаря. Він може обрати лікаря, дату та час прийому, заповнивши просту форму. Після підтвердження запису користувач отримаєте інструкції для підготовки до візиту.

Сторінка запису пацієнта, представлена на рисунку 4.10.

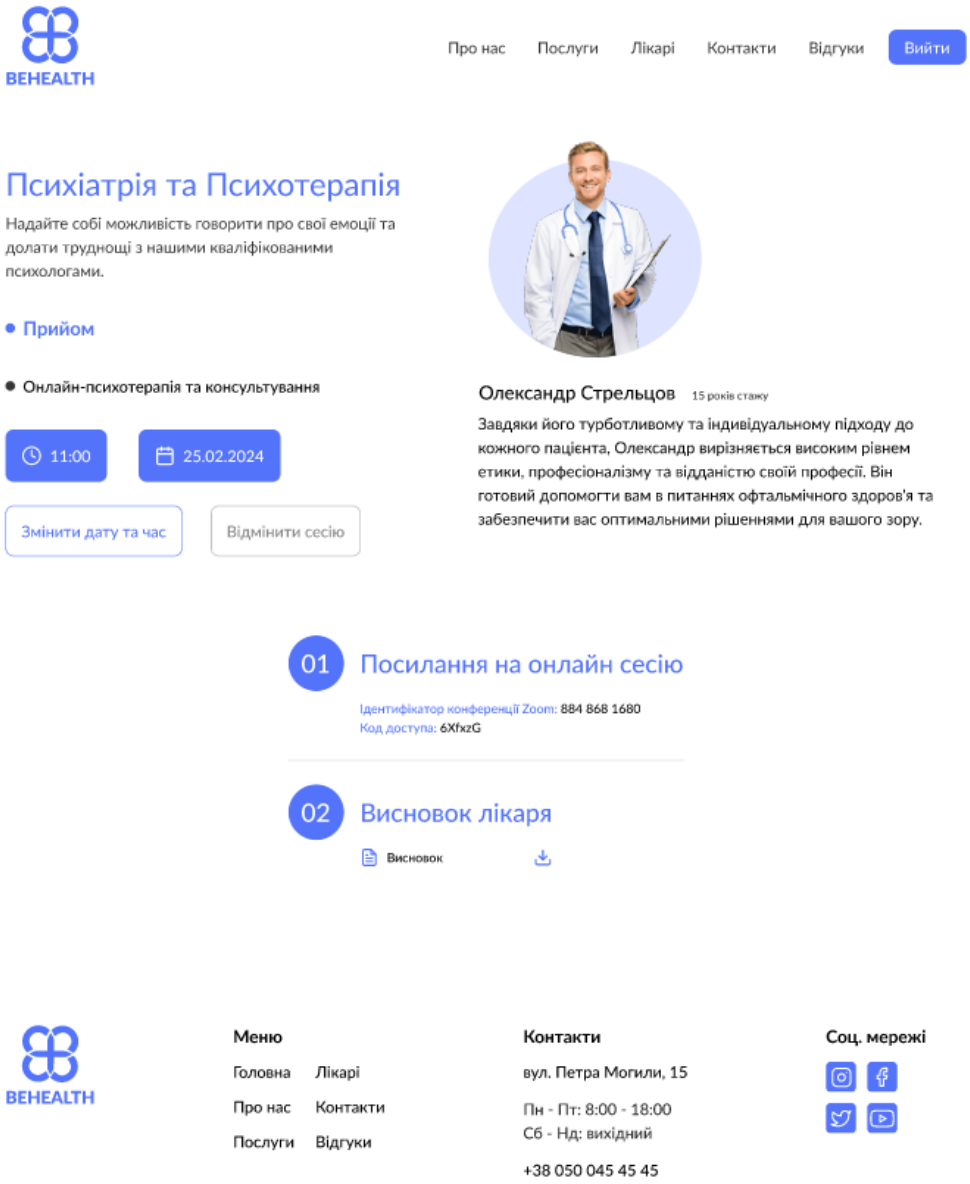

Рисунок 4.10 – Сторінка запису пацієнта

На сторінці авторизації користувачі можуть увійти до свого облікового запису, ввівши свої облікові дані. Форма для входу включає поля для введення електронної пошти або імені користувача та пароля. Також є можливість відновлення паролю у випадку його втрати.

Сторінка авторизації представлена на рисунку 4.11.

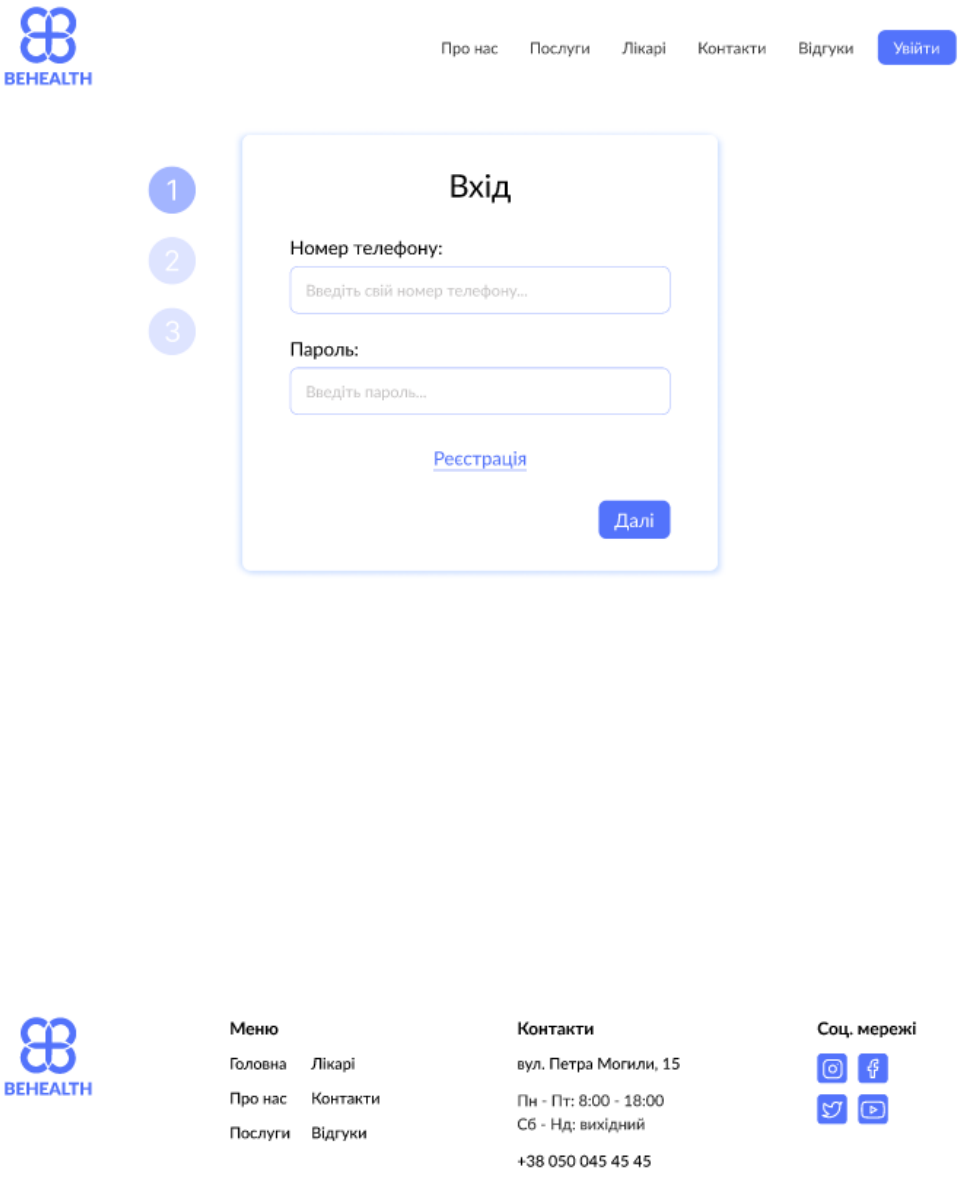

Рисунок 4.11 – Сторінка авторизації

Сторінка запису на послугу дозволяє користувачам записатися на конкретну послугу. Користувачі можуть вибрати послугу, дату та час для її надання. Після вибору послуги та часу користувачі підтверджують запис. Інструкції для підготовки до послуги можуть бути надані після підтвердження.

Сторінка запису на послугу у лікаря представлена на рисунку 4.12.

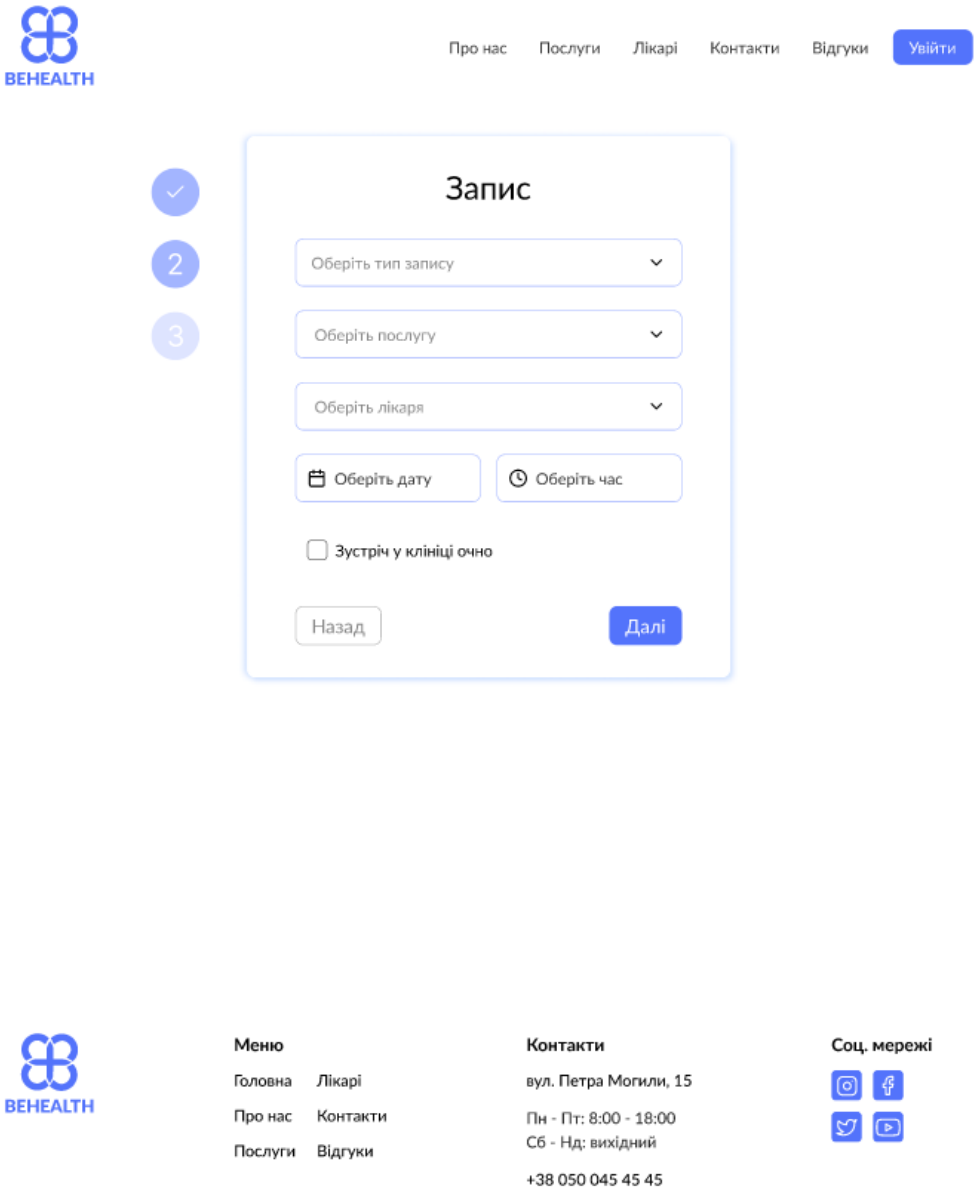

Рисунок 4.12 – Сторінка запису на послугу до лікаря

На сторінці оплати користувачі можуть здійснити оплату за обрані послуги. Вони обирають спосіб оплати, наприклад, кредитною карткою або через платіжну систему. Форма оплати включає поля для введення платіжних даних. Після введення даних та підтвердження платежу користувачі отримують підтвердження оплати.

Сторінка оплати представлена на рисунку 4.13.

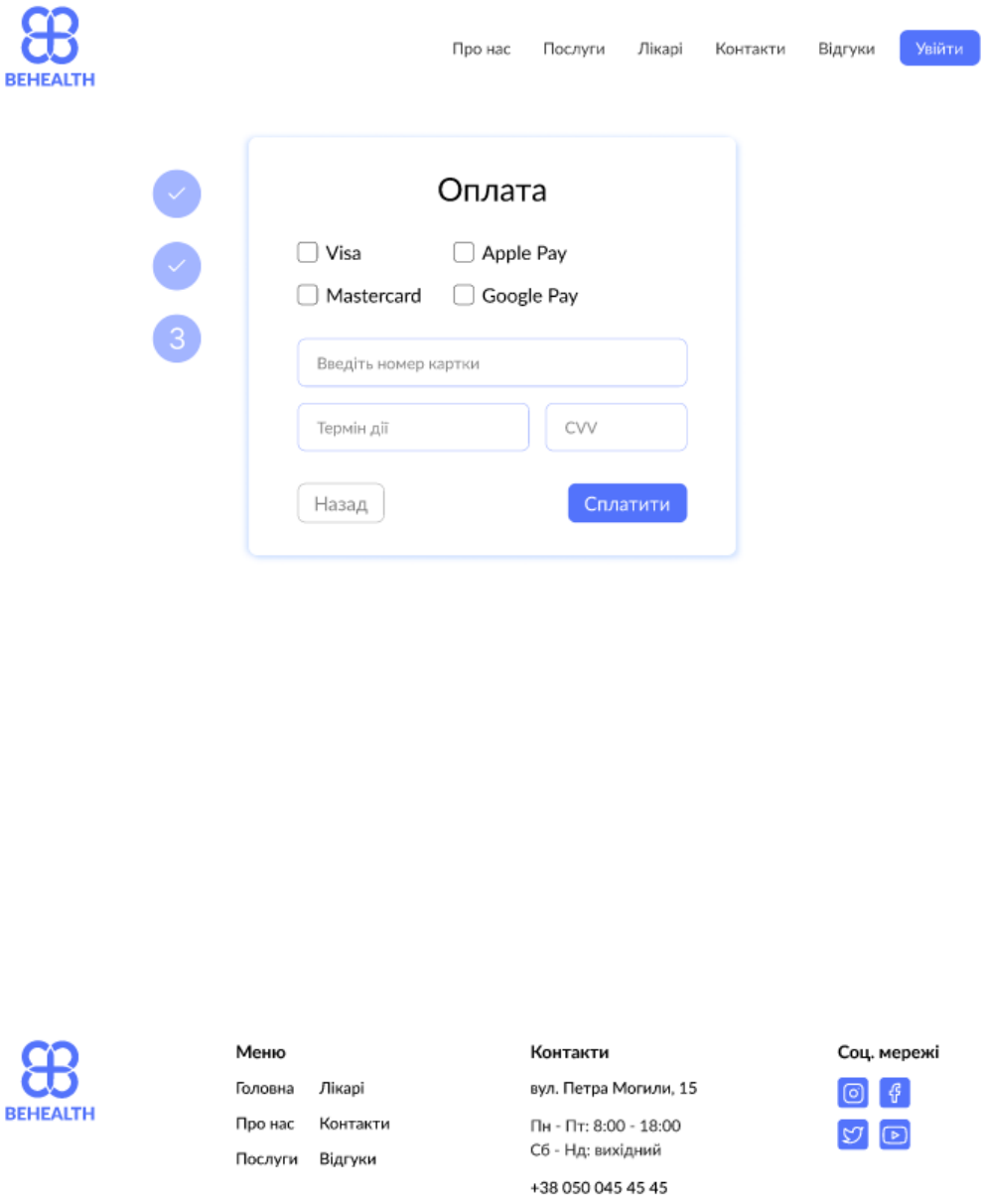

Рисунок 4.13 – Сторінка оплати

Сторінка реєстрації дозволяє новим користувачам створити обліковий запис. Користувачі вводять особисту інформацію, таку як ім'я, прізвище, електронну пошту, номер телефону та обирають пароль. Після реєстрації користувачі можуть увійти в систему, використовуючи свої нові облікові дані.

Сторінка реєстрації представлена на рисунку 4.14.

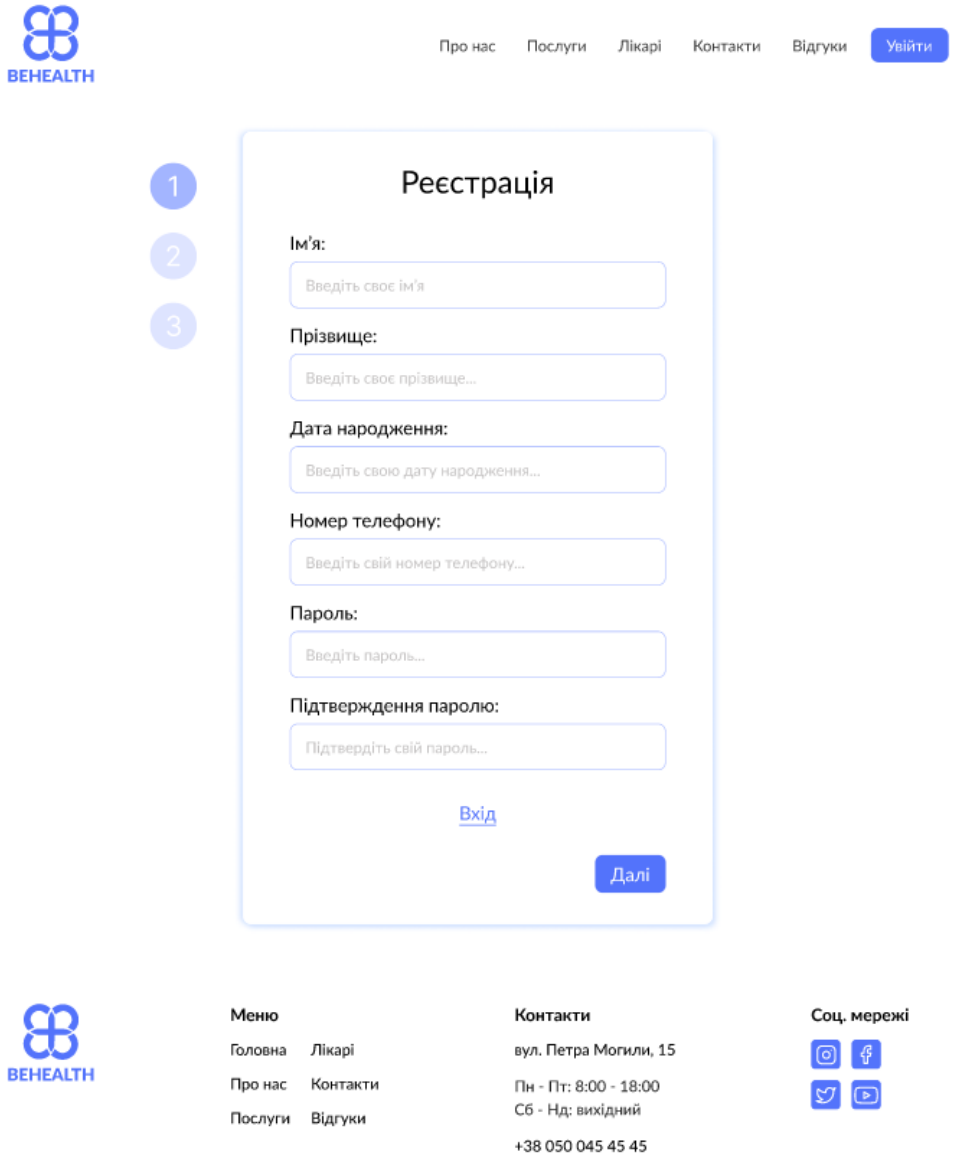

Рисунок 4.14 – Сторінка реєстрації

Ці описи допоможуть користувачам зрозуміти функціональність кожної сторінки та полегшать навігацію по вебзастосунку онлайн-клініки.

4.2 Реалізація телеграм-боту

При створення телеграм боту реалізована наступний функціонал:

– реєстрація нових користувачів;
– авторизація існуючих користувачів;
– перегляд списку доступних лікарів;
– перегляд профілю конкретного лікаря;

– перегляд досвіду лікаря;
– перегляд сертифікатів лікаря.

Початкове повідомлення бота представлено на рисунку 4.15.

Рисунок 4.15 – Початкове повідомлення бота

Реєстрацію нового користувача представлено на рисунку 4.16.

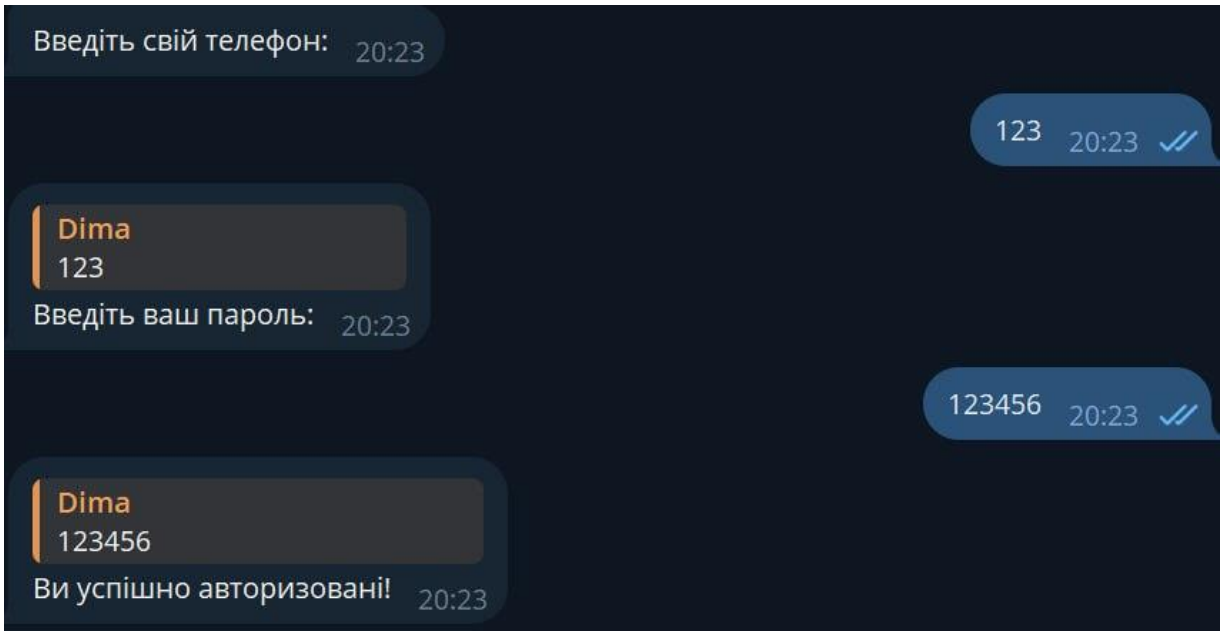

Рисунок 4.16 – Реєстрація у боті

Перелік доступних лікарів представлено на рисунку 4.17.

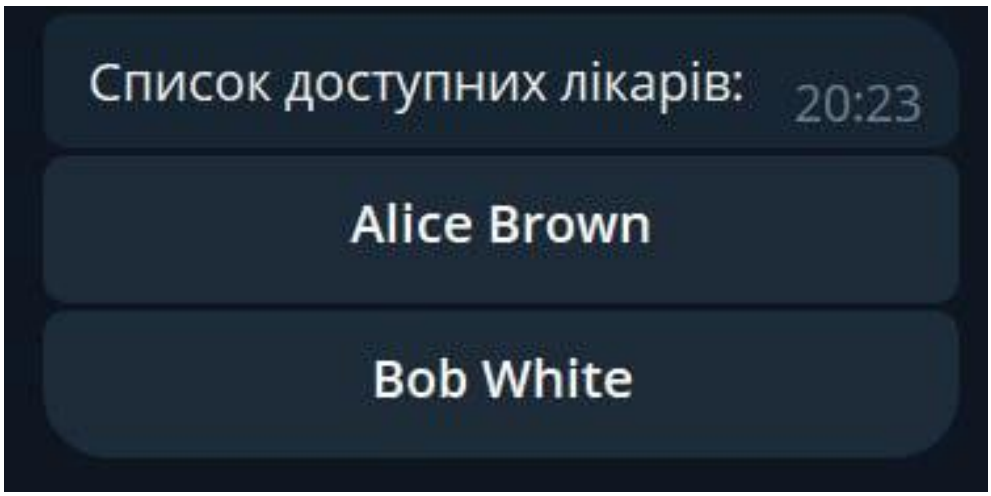

Рисунок 4.17 – Список лікарів

Перегляд детальної інформації про лікаря представлено на рисунку 4.18.

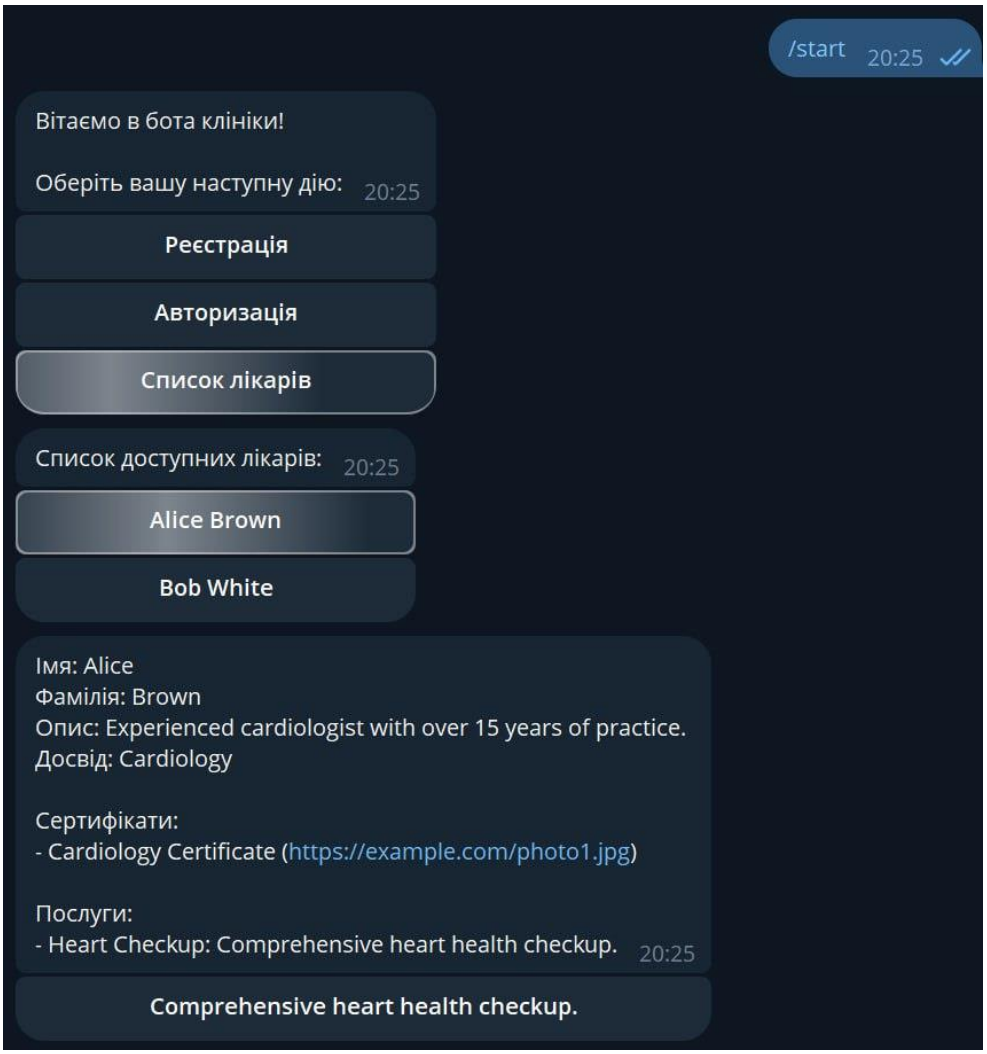

Рисунок 4.18 – Повна інформація про лікаря

Пошук лікаря по імені, фамілії чи спеціалізації представлено на рисунку 4.19.

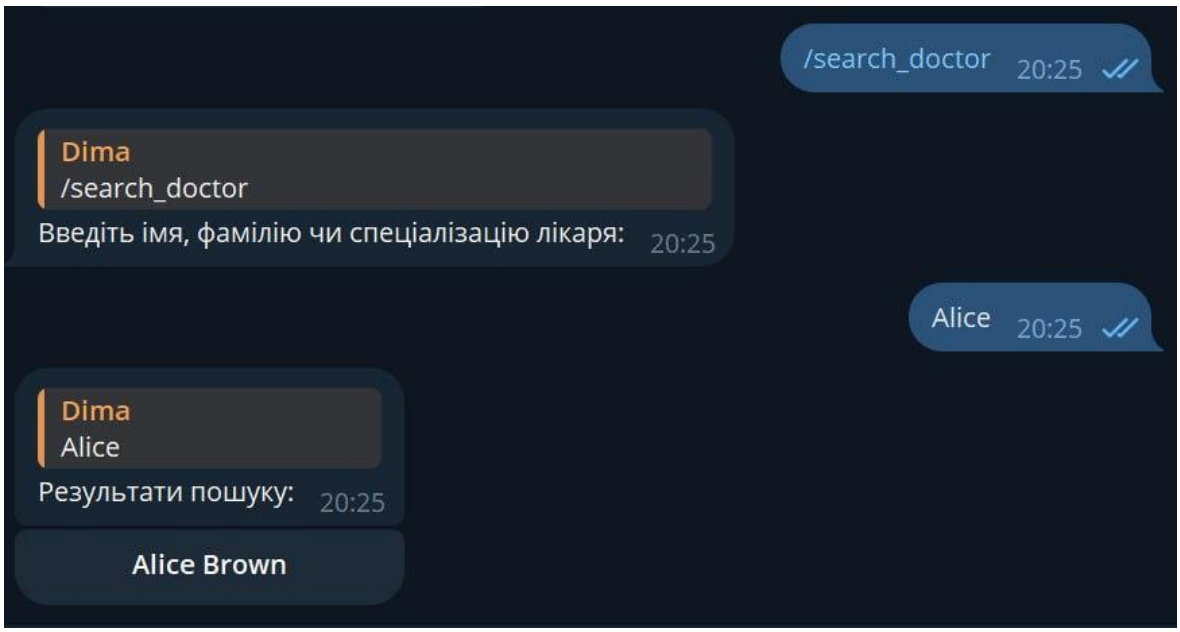

Рисунок 4.19 – Пошук лікаря за ім’ям

Даний функціонал дозволить зберегти час при виборі лікаря, не відкриваючи вебсайт.

Висновки до розділу 4

У четвертому розділі кваліфікаційної роботи представлено результати розробки вебзастосунку онлайн-клініки, відображено усі розроблені сторінки.

Також представлено розроблений функціонал у телеграм-боті лікарні.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи успішно досягнуто поставлених завдань щодо розробки вебзастосунку онлайн-клініки.

Створено функціонал телеграм-боту, який дозволяє пацієнтам зручно записуватись на прийом до лікаря та отримувати консультації. Здійснено інтеграцію боту з базою даних медичної системи, що забезпечує безперебійний обмін інформацією між користувачами та медичними працівниками.

У процесі роботи проведено тестування розробленого вебзастосунку, що дозволило оцінити його ефективність та виявити можливі недоліки.

На основі результатів тестування вдосконалено функціонал системи, що підвищило її зручність та надійність. Завдяки цьому вдалося оцінити практичне застосування вебзастосунку та визначити його переваги для пацієнтів та медичних працівників, такі як зменшення часу на організацію прийому та покращення комунікації між учасниками медичного процесу.

Таким чином, у ході виконання кваліфікаційної роботи розроблено повнофункціональний вебзастосунок для онлайн-клініки, який відповідає поставленим завданням і вимогам.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Онлайн клініка: вебсайт. URL: https://www.php.net/docs.php (Last accessed: 03.04.2024)

2. Он-Клінік: вебсайт. URL: https://www.php.net/docs.php (Last accessed: 03.04.2024)

3. Viva: вебсайт. URL: https://www.php.net/docs.php (Last accessed: 03.04.2024)

4. Мова програмування Php: підручник . Welling L., Thomson L. PHP and MySQL Web development. Sams publishing, 2003. 143 с.

5. Мова програмування Php: підручник . Nixon R. Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5. " O'Reilly Media, Inc.", 2014. 452 с.

6. Мова програмування Php: підручник . Nixon R. Learning PHP, MySQL, JavaScript, and CSS: A step-by-step guide to creating dynamic websites. " O'Reilly Media, Inc.", 2012. 230 с.

7. Ullman L. PHP and MySQL for dynamic Web sites: visual quickpro guide. Peachpit Press, 2011. 172 с.

8. Yuliano T. Pengenalan Php //IlmuKomputer. com. 2007. С. 176-254.

9. Mitchell W. H. PHP. 2008. 32 с.

10. Stauffer M. Laravel: Up & Running. " O'Reilly Media, Inc.", 2023. 456 с.

11. Bean M. Laravel 5 essentials. Packt Publishing Ltd, 2015. 765 с.

12. Фреймворк Laravel: підручник. Armel J. Web application development with Laravel PHP Framework version 4. 2014. 472 с.

13. Фреймворк Laravel: підручник. McCool S. Laravel starter. Packt Publishing, 2012. 32 с

14. Фреймворк Laravel: підручник. Sinha S., Dave H. J. Beginning Laravel. Packt Publishing, 2017. 543 с.

15. Фреймворк Laravel: підручник. He R. Y. Design and implementation of web based on Laravel framework //2014 International Conference on Computer Science and Electronic Technology (ICCSET 2014). Atlantis Press, 2015. С. 301-304.

16. Фреймворк Laravel: підручник. Yadav N., Rajpoot D. S., Dhakad S. K. LARAVEL: a PHP framework for e-commerce website //2019 Fifth International Conference on Image Information Processing (ICIIP). IEEE, 2019. С. 503-508.

17. Фреймворк Laravel: підручник. Dangar H. Learning laravel 4 application development. Packt Publishing, 2013. 834 с.

18. Мова програмування Sql: підручник . DuBois P. MySQL. Addison-Wesley, 2013. 634 с.

19. Мова програмування Sql: підручник . Гольцман В. MySQL 5.0. 2018. 696 с.

20. Мова програмування Php: підручник . Welling L., Thomson L. PHP and MySQL Web development. Sams publishing, 2003. 7с.

21. Php docs: вебсайт. URL: https://www.php.net/docs.php (Last accessed: 03.04.2024)

22. Laravel docs: вебсайт. URL: https://laravel.com/docs/11.x/readme (Last accessed: 03.04.2024)

23. MySQL docs: вебсайт. URL: https://dev.mysql.com/doc/ (Last accessed: 03.04.2024)

ДОДАТОК А – ЛІСТИНГ КОДУ ПРОГРАМИ

```
from aiogram import executor

from my_bot.bot_instance import dp
import my_bot.handlers as handlers


from aiogram import types
from aiogram.dispatcher import FSMContext

from my_bot.bot_instance import dp, bot
from my_bot.states import BookAppointment, CancelAppointment,
ChangeAppointment
from my_bot.storage.mysql_storage import cursor, conn
from my_bot.utils.utils import create_service_buttons,
create_date_buttons, create_time_buttons


@dp.callback_query_handler(lambda c: c.data.startswith('book_'))
async def process_book_appointment(callback_query:
types.CallbackQuery):
    doctor_id = callback_query.data.split('_')[1]
    cursor.execute("SELECT id, title FROM service WHERE doctor_id =
%s", (doctor_id,))
    services = cursor.fetchall()
    markup = create_service_buttons(services)
    await bot.send_message(callback_query.from_user.id, "Оберіть
послугу:", reply_markup=markup)
    await BookAppointment.select_service.set()


@dp.callback_query_handler(lambda c:
c.data.startswith('select_service_'),
state=BookAppointment.select_service)
```

```
async def process_select_service(callback_query: types.CallbackQuery, state: FSMContext):
    service_id = callback_query.data.split('_')[2]
    async with state.proxy() as data:
        data['service_id'] = service_id
    cursor.execute("SELECT id, day FROM service_date WHERE service_id = %s GROUP BY day", (service_id,))
    available_dates = cursor.fetchall()
    markup = create_date_buttons(available_dates)
    await bot.send_message(callback_query.from_user.id, "Оберіть дату:", reply_markup=markup)
    await BookAppointment.select_date.set()


@dp.callback_query_handler(lambda c: c.data.startswith('select_date_'), state=BookAppointment.select_date)
async def process_select_date(callback_query: types.CallbackQuery, state: FSMContext):
    date_id = callback_query.data.split('_')[2]
    async with state.proxy() as data:
        data['date_id'] = date_id
    cursor.execute(
        "SELECT id, time FROM service_date WHERE day = (SELECT day FROM service_date WHERE id = %s) AND service_id = %s",
        (date_id, data['service_id']))
    available_times = cursor.fetchall()
    markup = create_time_buttons(available_times)
    await bot.send_message(callback_query.from_user.id, "Оберіть час:", reply_markup=markup)
    await BookAppointment.select_time.set()


@dp.callback_query_handler(lambda c: c.data.startswith('select_time_'), state=BookAppointment.select_time)
```

```
async def process_select_time(callback_query: types.CallbackQuery,
state: FSMContext):
    service_time_id = callback_query.data.split('_')[2]
    async with state.proxy() as data:
        date_id = data['date_id']
        service_id = data['service_id']
        cursor.execute("INSERT INTO appointment (user_id,
service_date_id, service_id) VALUES (%s, %s, %s)",
                       (callback_query.from_user.id,
service_time_id, service_id))
        conn.commit()
    await state.finish()
    await bot.send_message(callback_query.from_user.id, "Ви успішно
записані на прийом!")


@dp.callback_query_handler(lambda c:
c.data.startswith('confirm_appointment_'),
state=BookAppointment.select_date)
async def process_confirm_appointment(callback_query:
types.CallbackQuery, state: FSMContext):
    service_date_id = callback_query.data.split('_')[2]
    async with state.proxy() as data:
        service_id = data['service_id']
        cursor.execute("INSERT INTO appointment (user_id,
service_date_id, service_id) VALUES (%s, %s, %s)",
                       (callback_query.from_user.id,
service_date_id, service_id))
        conn.commit()
    await state.finish()
    await bot.send_message(callback_query.from_user.id, "Ви спішно
записані на прийом!")


@dp.message_handler(commands=['upcoming_appointments'])
async def upcoming_appointments(message: types.Message):
```

```
    cursor.execute(
        "SELECT service_date.day, service_date.time,
doctor.first_name, doctor.last_name FROM appointment "
        "INNER JOIN service_date ON appointment.service_date_id =
service_date.id "
        "INNER JOIN doctor ON appointment.doctor_id = doctor.id WHERE
appointment.user_id = %s",
        (message.from_user.id,))
    appointments = cursor.fetchall()
    appointment_text = ""
    for appt in appointments:
        appointment_text += f"Дата: {appt[0]} Время: {appt[1]}\nВрач:
{appt[2]} {appt[3]}\n\n"
    await message.reply(appointment_text)


@dp.message_handler(commands=['cancel_appointment'])
async def cancel_appointment(message: types.Message):
    await CancelAppointment.appointment_id.set()
    await message.reply("Введите ID записи, которую хотите
отменить:")


@dp.message_handler(state=CancelAppointment.appointment_id)
async def process_cancel_appointment(message: types.Message, state:
FSMContext):
    appointment_id = message.text
    cursor.execute("DELETE FROM appointment WHERE id = %s AND user_id
= %s", (appointment_id, message.from_user.id))
    conn.commit()
    await state.finish()
    await message.reply("Запись успешно отменена!")


@dp.message_handler(commands=['change_appointment'])
async def change_appointment(message: types.Message):
```

```
    await ChangeAppointment.appointment_id.set()
    await message.reply("Введите ID записи, которую хотите изменить:")


@dp.message_handler(state=ChangeAppointment.appointment_id)
async def process_change_appointment_id(message: types.Message, state: FSMContext):
    async with state.proxy() as data:
        data['appointment_id'] = message.text
    await ChangeAppointment.next()
    await message.reply("Введите новый ID даты и времени:")


@dp.message_handler(state=ChangeAppointment.new_service_date_id)
async def process_change_appointment_date(message: types.Message, state: FSMContext):
    async with state.proxy() as data:
        appointment_id = data['appointment_id']
        new_service_date_id = message.text
        cursor.execute("UPDATE appointment SET service_date_id = %s WHERE id = %s AND user_id = %s",
                       (new_service_date_id, appointment_id, message.from_user.id))
        conn.commit()
    await state.finish()
    await message.reply("Запись успешно изменена!")

 from aiogram import types
from aiogram.dispatcher import FSMContext
from aiogram.types import InlineKeyboardMarkup, InlineKeyboardButton

from my_bot.bot_instance import dp
from my_bot.states import SearchDoctor
from my_bot.storage.mysql_storage import cursor
```

```
@dp.message_handler(commands=['search_doctor'])
async def search_doctor(message: types.Message):
    await SearchDoctor.query.set()
    await message.reply("Введіть ім'я, фамілію чи спеціалізацію лікаря:")


@dp.message_handler(state=SearchDoctor.query)
async def process_search_doctor(message: types.Message, state: FSMContext):
    search_query = message.text
    cursor.execute(
        "SELECT id, first_name, last_name FROM doctor "
        "WHERE first_name LIKE %s OR last_name LIKE %s OR type_id IN (SELECT id FROM type WHERE title LIKE %s)",
        (f'%{search_query}%', f'%{search_query}%', f'%{search_query}%'))
    doctors = cursor.fetchall()
    markup = InlineKeyboardMarkup()
    for doc in doctors:
        markup.add(InlineKeyboardButton(f"{doc[1]} {doc[2]}", callback_data=f"doctor_{doc[0]}"))
    await message.reply("Результати пошуку:", reply_markup=markup)
    await state.finish()

 if __name__ == '__main__':
    handlers.init_handlers()

    executor.start_polling(dp, skip_updates=True)
```