МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.,
_____Ю. П. Кондратенко
«____» _____ 2022 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

ВЕБЗАСТОСУНОК ДЛЯ ЕЛЕКТРОННОГО ЗАПИСУ ДО ЛІКАРНІ

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 402.21810310

Виконала: студентка 4 курсу, групи 402
_____ А. А. Єськіна
«_21_» __червня__ 2022 р.

Керівник: старший викладач каф.
інженерії
програмного забезпечення
_____ С. В. Дворецька
«_21_» __червня__ 2022 р.

Миколаїв – 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____ Ю. П. Кондратенко
«___»_____ 2021 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

Видано студенту групи 402 факультету комп'ютерних наук
Єськіній Анні Андріївні.
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи
«Вебзастосунок для електронного запису до лікарні».

Керівник роботи Дворецька Світлана Володимирівна ст. викладач каф. інженерії програмного забезпечення.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затв. наказом Ректора ЧНУ ім. Петра Могили від «07» грудня 2021 р. № 318

2. Строк представлення кваліфікаційної роботи студентом «28» червня 2022 р.

3. Вхідні (початкові) дані до роботи: загальна інформація щодо вебзастосунку, його відмінності від вебсайту, існуючі аналоги для усунення недоліків та як приклад для реалізації запису до електронної черги.

Очікуваний результат роботи: реалізація адаптивного вебзастосунку для електронного запису до лікарні, за допомогою якого буде підвищена якість процесу запису до лікаря за рахунок розподілу потоку відвідувачів і завчасного інформування клієнтів про порядок і правила обслуговування.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки)

1. Аналіз актуальності поставленної задачі розробки вебзастосунку.
2. Розгляд загальної теорії.
3. Обґрунтування засобів програмної реалізації.
4. Програмна реалізація вебзастосунку для запису до електронної черги з функціоналом адміністративної/клієнтської частини.
5. Тестування адаптивного інтерфейсу та функціональності вебзастосунку.

5. Перелік графічних матеріалів 3 таблиці, 48 рисунків, презентація.

6. Завдання до спеціальної частини: «Забезпечення вимог охорони праці у приміщенні комп'ютерної лабораторії вищого навчального закладу».

7. Консультанти:

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Охорона праці	Алексєєва А.О.	

Керівник роботи ст.викладач каф. інженерії програмного забезпечення Дворецька С.В.

(наук. ступінь, вчене звання, прізвище та ініціали) _____

(підпис)

Завдання прийнято до виконання __Єськіна А.А.__

(прізвище та ініціали)

_____

(підпис)

Дата видачі завдання «_25_» _листопада_ 2021р.

КАЛЕНДАРНИЙ ПЛАН
виконання бакалаврської кваліфікаційної роботи

Тема: Вебзастосунок для електронного запису до лікарні

№	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження теми та керівників БКР	27.10.2021	27.10.2021	Виконано
2	Отримання завдання на виконання БКР	25.11.2021	25.11.2021	Виконано
3	Складання календарного плану роботи на весь період виконання БКР	08.12.2021	08.12.2021	Виконано
4	Отримання завдання на переддипломну практику	23.05.2022	23.05.2022	Виконано
5	Проходження переддипломної практики, збір та аналіз матеріалів до БКР	23.05.2022	04.06.2022	Виконано
6	Розробка звіту з переддипломної практики	04.06.2022	06.06.2022	Виконано
7	Виконання БКР: аналіз сучасного стану задачі запису до електронної черги, огляд існуючих аналогів та технологій, розробка ПЗ	28.02.2022 та 06.06.2022	27.03.2022 та 19.06.2022	Виконано
8	Попередній захист БКР на засіданні комісії кафедри	30.05.2022	31.05.2022	Виконано
9	Доробка та остаточне оформлення БКР	02.06.2022	20.06.2022	Виконано
10	Подання БКР рецензенту	16.06.2022	18.06.2022	Виконано
11	Подання БКР, її електронної копії та інших документів (відгуку, рецензії) до захисту	20.06.2022	22.06.2022	Виконано
12	Захист БКР перед екзаменаційною комісією (ЕК)	27.06.2022	29.06.2022	Виконано

Розробив студент Єськіна А. А. _____
(прізвище, ім'я, по батькові студента) (*підпис*)

Керівник роботи ст.викл. Дворецька С. В. _____
(*посада, прізвище, ім'я, по батькові)* *(підпис)*

« 12 » грудня 2021 р.

АНОТАЦІЯ

бакалаврської кваліфікаційної роботи студентки групи 402 ЧНУ ім. Петра Могили

Єськіної Анни Андріївни

Тема: «Вебзастосунок для електронного запису до лікарні»

Дана кваліфікаційна робота присвячена розробці вебзастосунку для електронного запису до лікарні, який би забезпечив управління якістю і підвищив лояльність відвідувачів, оптимізував роботу шляхом розподілу потоку відвідувачів і завчасного інформування клієнтів про порядок і правила обслуговування.

Об’єкт дослідження – процес електронного запису користувачів до лікарні.

Предмет дослідження – технології та засоби розробки вебзастосунків для електронного запису до лікарні.

Робота складається з фахового розділу і спеціальної частини з охорони праці. Пояснювальна записка складається зі вступу, чотирьох розділів, висновків та додатків.

Розробка архітектури інтерактивного вебсервісу для електронного запису до лікарні полягає у подальшому використанні фреймворку Yii2, програми якого організовано відповідно до архітектурного патерну MVC.

При розробці даного ПЗ було виконано аналіз предметної області, відомих технічних рішень, вказано їх позитивні сторони та недоліки за допомогою порівняльної таблиці, а також сформульовано функціональні вимоги до розроблюваного програмного забезпечення. Викладено власні міркування з приводу структури бази даних та системи в цілому. Запропоновано варіант власної реалізації проєкту, з використанням проаналізованих та обраних засобів розробки.

Даний застосунок був реалізований у вигляді вебзастосунку, що дозволяє здійснювати авторизацію (реєстрацію) з подальшим записом до електронної черги з будь-якої точки, маючи інтернет.

Бакалаврська кваліфікаційна робота містить 68 сторінок, 48 рисунків, 3 таблиці, 25 використаних джерел та 3 додатка.

Ключові слова: *електронний талон, адміністративна панель, вебсервіс, вебзастосунок, система управління контентом, система управління реляційними базами даних, Yii2.*

ABSTRACT

Bachelor's qualification work of the student of 402 group of Petro Mohyla Black Sea National University

Yeskinoyi Anni Andriyivni

Title: «Web application for electronic hospital records»

This qualification work is dedicated to the development of a web application for electronic hospital records, which would provide quality management and increase visitor loyalty, optimize work by distributing the flow of visitors and informing customers in advance about the order and rules of service.

The object of research is the process of electronic registration of users in the hospital.

The subject of research - technologies and tools for developing web applications for electronic hospital records.

The work consists of a professional section and a special section on labor protection. The explanatory note consists of an introduction, four chapters, conclusions and appendices.

The development of the architecture of the interactive web service for electronic hospital records consists in the further use of the Yii2 framework, the programs of which are organized according to the MVC architectural pattern.

During the development of this software the analysis of the subject area, known technical solutions was performed, their advantages and disadvantages were indicated with the help of a comparative table, as well as the functional requirements for the developed software were formulated. Own considerations about the structure of the database and the system as a whole. The variant of own realization of the project, with use of the analyzed and chosen means of development is offered.

This application was implemented in the form of a web application that allows you to authorize (register) and then write to the electronic queue from anywhere with the Internet.

Bachelor’s qualification work contains 68 pages, 48 figures, 3 tables, 25 sources used and 3 appendices.

Keywords: *electronic coupon, administrative panel, web service, web application, content management system, relational database management system, Yii2.*

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ 11

ВСТУП 13

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 16

1.1 Характеристика предмету управління 16

1.2 Огляд характерних особливостей вебзастосунків 18

1.3 Вебзастосунок і вебсайт – порівняльна характеристика 19

1.4 Аналіз видів вебзастосунків 21

1.5 Типи мобільних вебзастосунків 26

1.6 Огляд та аналіз наявних аналогів 31

Висновки до розділу 1 35

2 ВИБІР ТА ОБҐРУНТУВАННЯ ЗАСОБІВ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ 36

2.1 Мова програмування при реалізації вебзастосунку 36

2.2 Вибір інтегрованого середовища розробки 39

2.3 Вибір системи керування базами даних 40

2.4 Обґрунтування доцільності використання фреймворку розробки 42

Висновки до розділу 2 43

3 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ 44

3.1 Функціональна/процесна модель системи 44

3.2 Розробка архітектури інформаційної системи 49

3.3 Проєктування структури бази даних 57

Висновки до розділу 3 64

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ПЕРЕВІРКА ПРАЦЕЗДАТНОСТІ ВЕБЗАСТОСУНКУ 65
4.1 Старт роботи з Yii2 65
4.2 Керівництво користувача та адміністратора 67
4.3 Тестування програмного забезпечення 79
Висновки до розділу 4 82
ВСТУП 84
5 ОХОРОНА ПРАЦІ 86
ВИСНОВКИ 101
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 103
ДОДАТОК А Код програмного забезпечення 106
ДОДАТОК Б Тест-кейси функціонального тестування 128
ДОДАТОК В Код автоматизованого тестування 132

ПЕРЕЛІК СКОРОЧЕНЬ

ЕМК – електронна медична карта

ОС – операційна система

ПЗ – програмне забезпечення

ЦП – центральний процесор

AJAX – Asynchronous Javascript and XML

CSS – Cascading Style Sheets

ERD – Entity Relationship Diagram

GPS – Global Positioning System

GUI – Graphical User Interface

HTML – HyperText Markup Language

HTTP – HyperText Transfer Protocol

IDE – Integrated development environment

JS – JavaScript

MVC – Model-View-Controller

PHP – Hypertext Preprocessor

PWA – Progressive Web Apps

SEO – Search Engine Optimization

SPA – Single-page application

UI – User Interface

UX – User Experience

Пояснювальна записка

до кваліфікаційної роботи

на тему:

ВЕБЗАСТОСУНОК ДЛЯ ЕЛЕКТРОННОГО ЗАПИСУ ДО ЛІКАРНІ

Спеціальність 122 «Комп’ютерні науки»

122 – БКР – 402.21810310

Виконала: студентка 4 курсу, групи 402
_____ А. А. Єськіна
«_21_» __червня__ 2022 р.

Керівник: старший викладач каф. інженерії
програмного забезпечення
_____ С. В. Дворецька
«_21_» __червня__ 2022 р.

Миколаїв – 2022

ВСТУП

Розробка вебзастосунків у XX столітті є дуже розповсюдженою, популярною та перспективною темою серед багатьох компаній світового рівня та й не тільки, зайнятих у сфері високотехнологічних цифрових та комп'ютерних технологій. Наприклад, для бізнесу більше неможливо досягти піку зростання без належного вебзастосунку. Завдяки покращенню взаємодії користувача з бізнесом або продуктом, вони стали основою для великого бізнесу або брендів і повільно завойовують статичні вебсайти як провідна веб-опція для демонстрації можливостей.

Визначення вебзастосунку є суперечливою темою, бо багато хто вважає вебзастосунки подібними до стандартних вебсайтів. Однак, ми повинні враховувати той фактор, що вебзастосунки створюються за зовсім іншими технологіями в порівнянні зі стандартними вебсайтами. Через це вебзастосунки мають набагато більше функціональних можливостей. Отже, що ж вважається вебзастосунком?

Вебзастосунок – прикладне програмне забезпечення, логіка якого розподілена між клієнтом та сервером, що використовує в якості клієнта веббраузер і працює на стороні вебсервера. При цьому взаємодія між клієнтом і сервером використовується з використанням протоколу HTTP.

Вебзастосунок, реалізований за допомогою сучасних методів розробки, являє собою інформаційний ресурс, завдяки якому можна здійснювати наступне:

- пряму взаємодію з користувачем та його інформаційну підтримку;
- передачу будь-якої інформації про компанію, а також новинний потік для користувачів;
- рекламу, так як вебзастосунок може поєднати у собі відеорекламу та

банери.

З кожним наступним роком все більше користувачів віддають перевагу інтернет сервісам в мережі, взамін звичним стаціонарним точкам, що торкнулося й лікарні, бо надмірні черги та час очікування є однією з основних причин незадоволеності відвідувачів. Інтернет сервіси подібного роду діяльності набули дуже великої популярності через свою зручність та простоту використання.

Електронний запис до лікарні став дуже популярним та актуальним у період пандемії COVID-19. ВООЗ рекомендувало впровадження карантину, самоізоляції та наполегливо не радило залишати житло, щоб мінімізувати контакти між людьми. Для мінімізації контактів та налагодження роботи медпрацівників люди й почали широко використовувати дані програмні продукти.

Тому розробка вебзастосунку, що дозволяє користувачеві записатися на прийом до потрібного йому медичного працівника у відповідний час – це досить актуальне завдання, реалізація якого значно прискорить і спростить процес.

Метою даної роботи є підвищення якості процесу запису до лікаря за рахунок розподілу потоку відвідувачів і завчасного інформування клієнтів про порядок і правила обслуговування шляхом розробки вебзастосунку електронної реєстрації візиту до лікарні.

Для досягнення поставленної мети, необхідно виконати наступні завдання:

- виконати аналіз існуючих аналогів та їх недоліків;
- визначити дизайн та структуру вебзастосунку, що розробляється;
- реалізувати основний функціонал клієнтської та адміністративної частини вебзастосунку для електронного запису до лікарні;
- розробити та виконати тестування адаптивного інтерфейсу;
- провести функціональні тестування вебзастосунку.

Об’єкт дослідження – процес електронного запису користувачів до лікарні.

Предмет дослідження – технології та засоби розробки вебзастосунків для електронного запису до лікарні.

Методологія і методи досліджень. При вирішенні зазначених завдань використовувалися технології програмування, а також візуалізація результатів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Характеристика предмету управління

Реєстратура – це підрозділ амбулаторно-поліклінічної установи, де здійснюється реєстрація та організація документації пацієнтів, забезпечення медичного страхування та оновлення всієї інформації, здійснення запису хворих на прийом до лікаря при первинному або повторному огляді, регулювання черги (потоку). У будь-якій поліклініці є відділ реєстратури. Саме там розпочинається ознайомлення пацієнтів з правилами прийому та чергами, а також можна отримати відповіді на питання: «ЩО, ДЕ і КОЛИ?».

Потік пацієнтів для окремих поліклінічних установ можна регулювати за допомогою системи купонів (талонів), завчасного запису по телефону, в реєстратурі або завдяки журналам самозапису тощо.

При використанні талонної системи час черги в деяких клініках становить годину до відкриття. За відсутності талону ви не можете звернутися до лікаря, якщо в вас відсутній гострий біль чи висока температура.

Щоденні завдання працівника реєстратури включають наступне:

- відповіді на телефонні дзвінки та перенаправлення за необхідності;
- повідомлення графіка роботи лікарів;
- організація невідкладної медичної допомоги хворим з лихоманкою та гострим болем;
- виписка талонів на первинний або повторний огляд (консультацію);
- забезпечення руху та оновлення амбулаторних карт.

Наразі більшість поліклінік реєструють пацієнтів безпосередньо у реєстратурі, де фіксуються особисті дані пацієнта, такі як: ПІБ, адреса та дані прийому, а саме: дата, час, місце прийому, спеціалізація та ПІБ лікаря. Після

цього пацієнт отримує паперовий талон з переліком усіх даних. Тож розберемо переваги та недоліки використання купонної системи у поліклінічних установах (рис 1.1).

Аналіз	Перевага	Недолік
Запис на прийом в реєстратурі поліклініки	Працівник реєстратури може надати інформацію різного роду відповідно до заданого питання	Потрібно стояти у черзі для отримання талону
Завчасний запис	Пацієнти можуть записатися на прийом не виходячи з дому	Працівник реєстратури може неправильно записати дані пацієнта
Запис на прийом у журналі самозапису	Пацієнти можуть записатися на прийом без допомоги працівника реєстратури	Оскільки ведення журналу самозапису відбувається в паперовому вигляді, то можливі дублювання та втрата даних

Рисунок 1.1 – Переваги та недоліки використання купонної системи

У наш час декотрі медичні заклади мають власні вебсайти, куди пацієнти можуть зайти та ознайомитись з потрібною їм інформацією. Крім того, деякі з них підтримують функцію реєстрації на прийом до лікаря. Метою даної роботи є реалізація вебзастосунку для електронного запису до лікарні, який би забезпечив управління якістю і підвищив лояльність відвідувачів, оптимізував роботу шляхом розподілу потоку відвідувачів і завчасного інформування клієнтів про порядок і правила обслуговування.

Розробка даного вебзастосунку та його використання надасть користувачеві наступні переваги:

- не потрібно стояти в черзі для того щоб отримати талон на прийом до лікаря;

- відсутність черг біля кабінету лікаря, оскільки в талоні будуть вказані дата та час прийому;
- наявність електронного розкладу для запису;
- процес реєстрації та електронного запису на прийом проходить набагато швидше, ніж видача талона працівниками реєстратури;
- на відміну від електронних талонів, паперовий талон може втратити якість або навіть загубитися.

1.2 Огляд характерних особливостей вебзастосунків

Зі зростанням використання Інтернету компанії змінюють спосіб роботи та впроваджують все більше вебзастосунків. Розуміння того, що таке вебзастосунок, може допомогти вам усвідомити його важливість у нашому повсякденному житті.

Вебзастосунок – це програма клієнт-сервер. Термін «клієнт» у даному контексті відноситься до програми, яку особа використовує для запуску вебзастосунку. Це частина клієнт-серверного середовища, де комп’ютери обмінюються інформацією. Наприклад, у випадку з базою даних клієнтом є програма за допомогою якої користувач вводить дані. Сервер – це програма, яка зберігає інформацію.

Уся інформація, яку ми бачимо на просторах Інтернету, зберігається на комп’ютерах, які звуться вебсерверами. На цих вебсерверах вставлено спеціальне програмне забезпечення, яке надає можливість користувачам знаходити інформацію відповідно до їх запиту. Більшість такої інформацій з’являється перед користувачами за допомогою вебсайтів, кожен з яких має власне доменне ім’я або адресу в Інтернеті. Позитивною особливістю є те, що користувачу немає необхідності завантажувати на свій пристрій програмне забезпечення. Для

перегляду вебзастосунку з девайсу, у користувача повинна бути встановлена спеціальна програма – веббраузер.

До компонентів вебзастосунків UI/UX відносяться журнали активності, інформаційні панелі, сповіщення, налаштування, статистика тощо. Ці компоненти не мають нічого спільного з функціонуванням архітектури вебзастосунків.

Архітектура вебзастосунків описує взаємодію між програмою, базами даних та проміжними системами в Інтернеті. Команда або веброзробник, що розробляє, вирішує, що робитиме код на сервері щодо коду в браузері. Для написання серверного коду використовуються C#, Java, JavaScript, Python, PHP, Ruby тощо.

Будь-який код, що здатний відповідати на запити HTTP, може виконуватися на сервері. Код на стороні сервера відповідає за створення сторінки, яку запитав користувач, а також за зберігання різних типів даних, включаючи профілі користувачів і введені користувачем дані. Його ніколи не бачить кінцевий користувач.

Для написання клієнтського коду використовується комбінація CSS, HTML і JavaScript. Цей код аналізується веббраузером. На відміну від коду на стороні сервера, код на стороні клієнта можна побачити, а також змінити користувачем. Він реагує на введення користувача. На відміну від серверного коду, він спілкується тільки через HTTP-запити і не може зчитувати файли з сервера безпосередньо.

1.3 Вебзастосунок і вебсайт – порівняльна характеристика

Кожний вебзастосунок вважається вебсайтом, але не кожен вебсайт можна назвати вебзастосунком.

Вебсайт – це сукупність логічно зв'язаної гіпертекстової інформації, оформленої у вигляді окремих сторінок і доступної в мережі Інтернет.

Нижче наведено основні відмінності між вебзастосунком і вебсайтом (таблиця 1.1).

Таблиця 1.1 – Порівняльна характеристика

Критерій	**Вебзастосунок**	**Вебсайт**
Створено для	Вебзастосунок призначений для взаємодії з кінцевим користувачем.	Вебсайт переважно складається зі статичного вмісту. Він є загальнодоступним для всіх відвідувачів.
Взаємодія з користувачем	У вебзастосунку користувач не тільки читає вміст сторінки, але й маніпулює обмеженими даними.	Вебсайт надає візуальний і текстовий вміст, який користувач може переглядати та читати, але не впливає на його функціонування.
Аутентифікація	Веб-програми потребують аутентифікації, оскільки вони пропонують набагато ширший спектр можливостей, ніж вебсайти.	Для інформаційних вебсайтів аутентифікація не є обов’язковою. Користувач може попросити зареєструватися, щоб отримувати регулярні оновлення або отримати доступ до додаткових опцій. Ця функція недоступна для незареєстрованих відвідувачів вебсайту.

Продовження таблиці 1.1

Завдання та складність	Функції вебзастосунків є значно вищими на певний рівень та складними порівняно з вебсайтом.	Вебсайт відображає зібрані дані та інформацію на певній сторінці.
Тип програмного забезпечення	Розробка вебзастосунків є частиною вебсайту. Сам по собі це не повний вебсайт.	Вебсайт являє собою повноцінний продукт, доступ до якого здійснюється за допомогою браузера.
Компіляція	Перед розгортанням сайт має бути попередньо скомпільований.	Сайт не потребує попередньої компіляції.
Розгортання	Усі зміни вимагають повторної компіляції та розгортання всього проєкту.	Невеликі зміни ніколи не потребують повної перекомпіляції та розгортання. Вам просто потрібно оновити HTML-код.

1.4 Аналіз видів вебзастосунків

Якщо є питання щодо типу вебзастосунків, доступних у галузі, то ви потрапили в потрібне місце. У цьому підрозділі буде розглянуто види вебзастосунків та їх переваги.

Загалом вебзастосунки поділяються на дві основні групи: статичні та динамічні [3]. Але, це не кінець. Динамічні вебпрограми поділяються на інші підтипи. Розглянемо кожну групу детальніше.

Статичні вебзастосунки складаються з обмеженого вмісту і не мають гнучкості. Ці програми вважаються сторінками, створеними сервером з дуже малою інтерактивністю або без неї.

Загалом, у цих програмах немає персоналізації, а зміни відбуваються на сторінці після її повного завантаження.

Професійні портфоліо, цифрові резюме, цільові сторінки для маркетингу тощо – найкращі приклади статичних вебзастосунків.

Переваги статичних вебзастосунків:

- простота розміщення;
- швидка реалізація;
- низька вартість розробки;
- легко індексувати в пошукових системах;
- дуже швидка передача даних при повільному інтернет-з’єднанні.

Динамічні вебзастосунки – це один з кращих типів вебзастосунків, оскільки вони витягують дані в режимі реального часу на основі запитів користувачів. Вони володіють підвищеною технічною складністю в порівнянні зі статичними веб-додатками.

Всякий раз, коли вебсервер отримує запит на певну сторінку, запит сторінки надходить в програмне забезпечення, відоме як сервер додатків.

Для вебзастосунків такого типу потрібна база даних для зберігання даних, і її вміст постійно оновлюється для того, щоб користувачі могли отримати до них доступ. Цього можна досягти, використовуючи систему управління контентом, таку як WordPress, яка має вбудовану панель адміністрування.

Для розробки динамічних вебзастосунків можна використовувати різні типи мов програмування та доповнень, такі як Node.js, Ruby on Rails, jQuery, HTML, CSS, PHP, Perl, Python і т. д.

Редагувати або оновлювати вміст у динамічних вебзастосункух дуже просто. Але модифікація бекенда або кодування може бути складним завданням залежно від сервера та інших аспектів.

Динамічні вебзастосунки діляться на інші підтипи, які зазначено нижче:

- Односторінкові вебзастосунки дозволяють користувачам безперешкодно взаємодіяти з вебсторінкою. Запити і відповіді виконуються ефективно завдяки невеликим обсягам даних.

Коротко кажучи, SPA працюють набагато швидше в порівнянні з традиційними веб-додатками, оскільки вони виконують логіку в веббраузері, а не на сервері.

Крім того, ви можете оновити будь-який односторінковий додаток відповідно до вимог в майбутньому. Тим не менш, через універсальні URL-адреси вони не компетентні відповідно до останніх правил SEO.

Переваги односторінковпх вебзастосунків:

- можливість доповнення за необхідності;
- просте налагодження;
- менш складна реалізація;
- краще кешування;
- зручність у використанні.

- Багатосторінкові вебзастосунки функціонують аналогічно традиційним.

Тут вебзастосунок перезавантажується і відображає нову сторінку з сервера в браузері кожен раз, коли користувачі виконують нову дію.

У цих типах вебзастосунків логіка зберігається в серверній частині, а отже, запити від клієнтів прокладають свій шлях на сервер і повертаються.

Процес створення сторінок на сервері, відправки їх клієнту і подання їх в браузері завдає шкоди призначеному для користувача інтерфейсу.

Це можна вирішити, використовуючи технологію AJAX, яка вносить раптові зміни без повного перезавантаження сторінки.

MPA можна створювати за допомогою різних мов, таких як HTML, CSS, JavaScript, AJAX, jQuery тощо. Веб-портали, інтернет-магазини, каталоги, ринки, корпоративні вебзастосунки тощо підпадають під багатосторінкові програми.
Переваги багатосторінкових вебзастосунків:

- більш зручні для SEO;
- можливість додання необмеженої кількість сторінок у наявний додаток.

- Вебпортали – це один із типів веб-програм, у яких різні розділи або категорії доступні на домашній сторінці. Тут ця сторінка складається з різних деталей, таких як чати, електронні листи, форуми, реєстрація користувачів тощо.

Портали найкраще підходять для підприємств та організацій, які хочуть створювати індивідуальні інтерфейси відповідно до вимог своєї цільової аудиторії.

Всякий раз, коли користувач входить в систему, постачальник послуг може перевірити активність користувача. Залежно від виділеного доступу конкретні функції можуть бути обмежені конкретними користувачами.

Урядові, студентські, освітні сайти університетів, пацієнтів і т.д. відносяться до вебпорталів.
Переваги вебпорталів:

- забезпечують покращену взаємодію;
- краща інтеграція;
- багатоканальна присутність.

- Анімовані вебзастосунки тісно пов'язані з технологією FLASH. Створюючи ці типи веб-програм, ви можете представляти вміст за допомогою різних анімованих ефектів.

Тут дизайнери UI/UX мають свободу дій та фантазій для того, щоб стати надзвичайно креативними та інтегрувати речі, які неможливі в різних типах вебзастосунків.

Основним недоліком створення анімованих вебзастосунків є те, що вони не підходять для веб-позиціонування та SEO, оскільки пошукові системи не можуть отримати з них дані.

- Вебпрограми на основі JavaScript. Після доступності провідних фреймворків, таких як Angular.js, React.js, Vue.js, Node.js, логіка вебзастосунків почала рухатися на стороні клієнта, забезпечуючи вищу адаптивність порівняно з AJAX.

Вебзастосунки, створені з використанням вищезгаданих фреймворків JS, пропонують підвищену продуктивність, різні рівні взаємодії з користувачем і зазвичай оптимізовані для SEO. Клієнтські портали, бізнес-орієнтовані веб-програми тощо підпадають під вебпрограми на основі JavaScript.

Переваги вебпрограм на основі JavaScript:

- високоінтерактивні;
- швидкі та чутливі до взаємодій;
- офлайн-підтримка;
- швидка інтеграція.

- Прогресивні вебзастосунки (PWA) – це вебсайти, схожі на мобільні додатки. Користувачі можуть отримати доступ до інформації та всіх функцій веб-програми за допомогою мобільних браузерів.

Різні експерти стверджують, що PWA є модифікованою версією SPA. Незважаючи на те, що це не вірно на основі теорії; однак у реальному житті суть автентична.

Основною метою PWA є не застосування нових правил в архітектурі, а підвищення швидкості та мобільної адаптації вебпрограм. Тут покращено кешування, передачу даних та встановлення головного екрана.

Крім того, PWA дають змогу покращити мобільний Інтернет і надавати свої послуги користувачам, незважаючи на повільне/погане інтернет-з'єднання. Starbucks, Forbes, OLX, MakeMyTrip тощо – одні з найкращих прикладів PWA. Переваги прогресивних вебзастосунків:

- офлайн-підтримка;
- підвищена продуктивність;
- встановлення та оновлення при потребі;
- спеціальні функції платформи доступу.

- Вебзастосунки для електронної комерції. Даний вид вебзастосунку частіше за все використовують для реалізації інтернет-магазину.

Розробка таких веб-програм стає складною, оскільки потрібно обробляти транзакції та інтегрувати різні способи оплати, такі як PayPal, дебетова/кредитна картка тощо.

Деякі основні функції вебпрограми для електронної комерції включають додавання нових продуктів, видалення старих продуктів, обробку платежів, зручний інтерфейс тощо. Щоб виконувати всі ці завдання, адміністратору потрібна ефективна панель керування.

1.5 Типи мобільних вебзастосунків

Найрозповсюдженішими типами мобільних вебзастосунків вважається вебсайт з адаптивним дизайном та мобільний додаток, розглянемо їх детальніше.

Наприкінці 1990-х років вебсайти дійсно були переважно статичними сторінками. Все, що потрібно було для створення вебсайту – це знання мови

гіпертекстової розмітки HTML. Якщо сторінка пропонувала якесь програмне забезпечення, то це були виключно засоби, що міг надати сервер, на якому розміщено вебсайт.

Про зручність та «естетику» вебсайту в ті роки говорити не варто. Час спливає, розвиваються мови програмування, розширюються канали передачі інформації. Зараз Інтернет є самодостатнім сектором економіки, а вебсайти стали повноцінними представництвами компаній в Інтернеті.

З огляду на поширеність і різноманітність мобільних пристроїв, дизайнери, повинні задовольнити варіацію розмірів екранів. Це виклик, з яким зараз стикається кожен вебдизайнер і дизайнер додатків. Від гігантського корпоративного монітора до смарт-годинника існує величезна кількість способів, за допомогою яких користувачі можуть отримати доступ до інформації в Інтернеті на сьогоднішній день. Дизайнери, які прагнуть подолати розрив між пристроями, мають варіант дизайну, який зветься адаптивним.

Адаптивний дизайн відноситься до дизайну графічного інтерфейсу користувача (GUI) [5], який адаптується до різних розмірів екрана. Даний вид дизайну поєднує в собі скупчення різноманітних верств, сіток, зображень та правильного використання CSS. Адаптивний дизайн зазвичай використовує кілька фіксованих розмірів макета – коли система визначає розмір браузера, вона обирає макет, найбільш відповідний для поточного розміру екрана (наприклад, iPad).

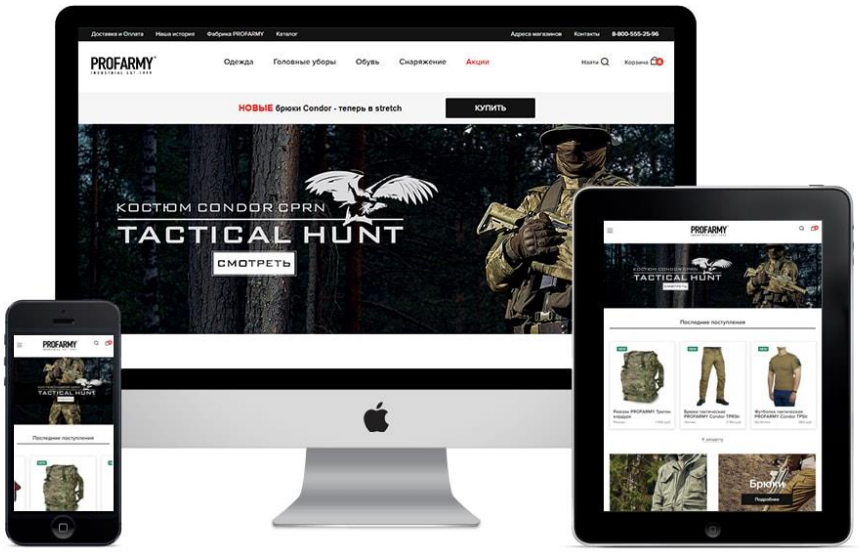

Рисунок 1.2 – Приклад адаптивного дизайну на різних пристроях

Деякі сайти швидко впровадили адаптивний дизайн. Наприклад, Amazon, USA Today, Apple і About.com налаштували себе як вебсайти, оптимізовані для мобільних пристроїв.

У адаптивному дизайні стандартною практикою для дизайнерів є розробка шести дизайнів для шести найбільш поширених розмірів екрана – 320, 480, 760, 960, 1200 і 1600 пікселів.

Перевага адаптивного дизайну полягає в тому, що він дозволяє дизайнеру підбирати рішення, щоб графічний інтерфейс оптимально відображався на екранах різних розмірів. Недоліком є те, що адаптивний дизайн є дорогим, оскільки, по суті, він вимагає від дизайнера створити до шести окремих графічних інтерфейсів або, кажучи в двох словах, еквівалент шести версій однієї вебсторінки, щоб мати найкращу, готову до роботи, засувку з потрібними користувачеві характеристиками екрана. Іншим недоліком є те, що адаптивний дизайн також може залишити користувачів, які не мають екрану стандартного

розміру, без оптимального рішення.

Якісно реалізований адаптивний дизайн забезпечує чіткість та приємність вигляду контента для користувача та вірну функціональність на будь-якому пристрої.

Мобільний додаток – це комп'ютерна програма або програмне забезпечення (ПЗ), призначена для роботи на мобільних пристроях, таких як телефон, планшет або годинник.

Спочатку мобільні додатки були призначені для підвищення продуктивності, таких застосунків як електронна пошта, календар і бази даних контактів, але суспільний попит на програми спричинив швидке розширення в інших сферах, таких як мобільні ігри, автоматизація виробництва, GPS, відстеження замовлень і квитків, перегляд відео, спілкування тощо.

IOS або Android вважаються двома найпопулярнішими операційними системами для мобільних застосунків [4]. Операційна система (ОС) — це програмне забезпечення, яке діє як інтерфейс між компонентами комп'ютерного обладнання та користувачем. Кожна комп'ютерна система повинна мати принаймні одну операційну систему для запуску інших програм.

Рисунок 1.3 – Принцип роботи ОС

Android – це мобільна операційна система на основі модифікованої версії ядра Linux та іншого програмного забезпечення з відкритим вихідним кодом, розроблена в основному для мобільних пристроїв із сенсорним екраном, таких як смартфони та планшети. Найпопулярніша мобільна ОС в світі зі своєю галереєю додатків Google Play (рис. 1.4), який вважається. сервісом цифрового розповсюдження.

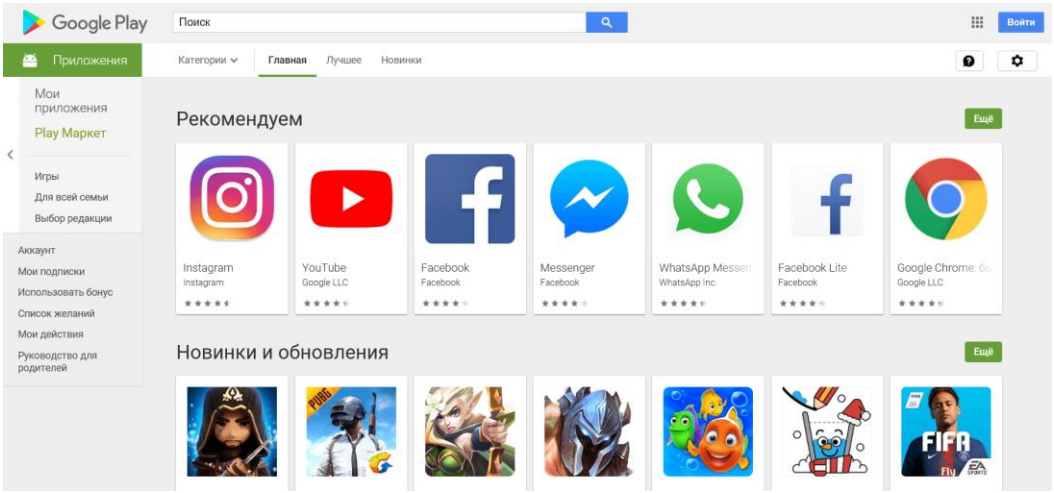

Рисунок 1.4 – Огляд галереї додатків Google Play

Використовується як ОС на безлічі різних мобільних телефонів зручна та легка в використанні, але важка в розробці тому що у мобільних пристроях в рази менші можливості в порівнянні з стаціонарними комп’ютерами.

IOS – це мобільна операційна система, створена та розроблена Apple Inc. виключно для свого апаратного забезпечення. Має власну галерею додатків – Apple App Stope (рис. 1.5).

Рисунок 1.5 – Огляд галереї додатків Apple App Store

На відміну від Android, ця ОС використовується тільки однією компанією та робить життя розробників більш спрощеним, через те, що відсутнє величезне розмаїття пристроїв.

1.6 Огляд та аналіз наявних аналогів

Провівши час на просторах інтернету та переглянувши існуючі сервіси для запису до електронної черги можна виділити декілька з них.

«Поліклініка без черг» – це одна з перших систем контролю та управління потоком пацієнтів в Україні, яка була створена у 2016 році (рис. 1.6). Вона

дозволяє кожному жителю України зареєструватися та записатися на прийом до обраного медичного спеціаліста в лікарню свого міста, за умови, якщо вона додана в систему.

Сервіс передбачає комплексне використання IT-інструментів: введення електронної черги у закладі, онлайн-запис на консультації та прийоми, логістична взаємодія лікарів, спеціалістів, діагностичних центрів та лабораторій між собою задля комфорту пацієнта.

На відміну від інших систем, «Поліклініка без черг» відрізняється простим та зрозумілим інтерфейсом. Велику роль в цьому зіграла активна взаємодія з прогресивними головними лікарями України, які допомагали тестувати систему.

З мінусів можна виділити тільки те, що інколи через велику завантаженість сервісу неможливо здійснити вхід до особистого кабінету і записатися до електронної черги (перевірено на особистому досвіді).

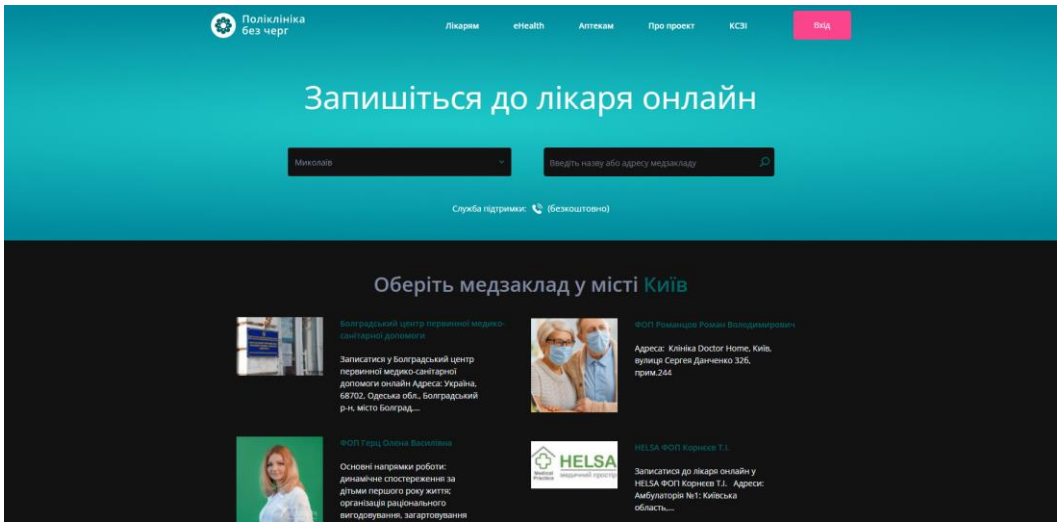

Рисунок 1.6 – Головна сторінка сайту «Поліклініка без черг»

«Helsi» – це електронна медична система, створена для лікарів, пацієнтів, державних та приватних медичних закладів (рис. 1.7). Це не державний продукт, а приватна компанія, яка уклала з державою договір про надання послуг.

Стосовно можливостей даної медичної системи, то вони подібні до сервісу «Поліклініка без черг», але серед них можна виділити облік медичних препаратів, формування звітів та статистики, і доступ пацієнтів до своєї електронної медичної карті (ЕМК), що є однозначним плюсом.

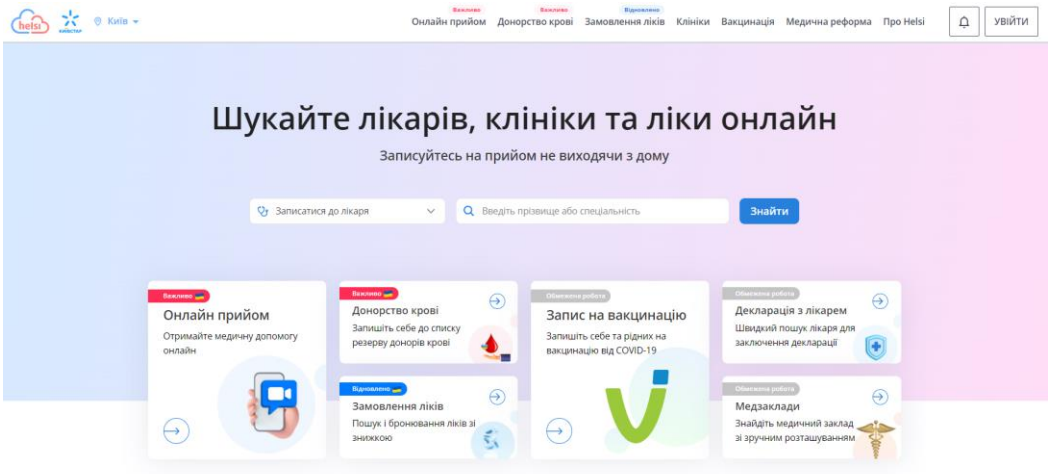

Рисунок 1.7 – Головна сторінка сайту «Helsi»

«Askep.net» – це популярний проєкт DeHealth, котрий забезпечує універсальний доступ до послуг охорони здоров'я для мільйонів людей, покращує якість лікування, економить час та кошти лікарям і їх пацієнтам (рис. 1.8).

Функціонування системи проходить на вищому рівні та надає можливість цілодобового доступа пацієнта до історії усіх його відвідувань медзакладу, результатів аналізів, поставлених діагнозів, призначеного лікування та використання медичного страхування.

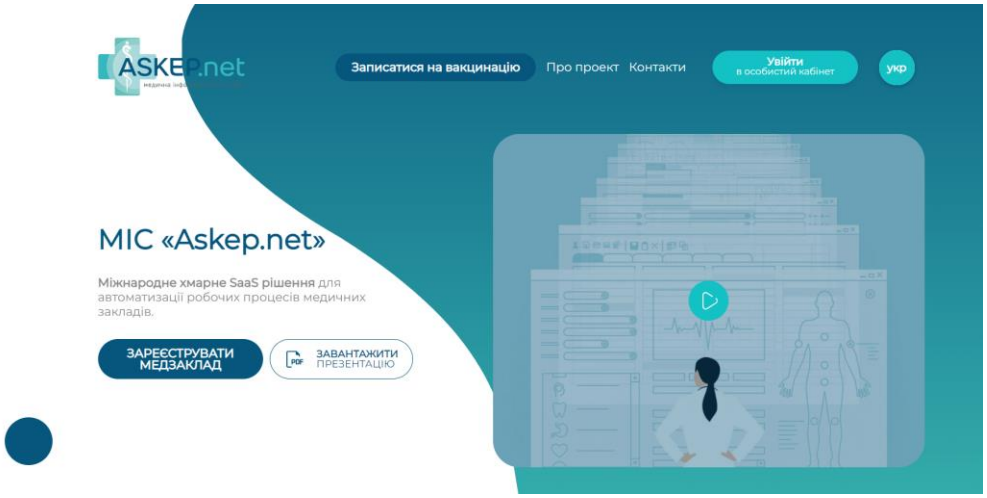

Рисунок 1.8 – Головна сторінка сайту «Askep.net»

Як висновок можна сказати, що бездоганною функцією перелічених програмних продуктів є генерація електронного талону, який за необхідності можна зберегти та роздрукувати.

Для більш наглядного порівняння розглянутих програмних продуктів була створена таблиця 1.2.

Таблиця 1.2 – Порівняльна характеристика програмних продуктів

Назва сервісу	«Поліклініка без черг»	«Helsi»	«Askep.net»
Реєстрація в системі	+	+	+
Інформаційна сторінка	+	+	+
Можливість обрання медичного закладу	+	+	+
Запис на первинний/повторний огляд	+	+	+
Онлайн-консультація	-	+	-
Зворотній зв’язок	+	-	-

Продовження таблиці 1.2

Система нагадування про запис	-	-	+
ЕМК	-	+	+
Мобільний додаток	-	+	+

Висновки до розділу 1

У першому розділі проведено аналіз предметної області, а саме описана характеристика предмету управління (структура та напрями діяльності, бізнес-процеси).

Розглянуто загальну теорію для кращого представлення та розуміння. Встановлено що таке вебзастосунок, чим він відрізняється від вебсайту та за якими відмінностями їх можна визначити.

Визначено, що існують різні види вебзастосунків, які розподіляються в залежності від завдань, які стоять перед ними, та сфери, в якій саме він буде реалізований і запущений.

Проаналізовано найрозповсюдженіші типи мобільних вебзастосунків: вебсайт з адаптивним дизайном та мобільний додаток.

Наприкінці бажано зазначити, що було проведено огляд та аналіз існуючих аналогів, таких як: «Поліклініка без черг», «Helsi», «Askep.net», та виявлено недоліки серед них, що дозволяє зробити висновок про актуальність теми роботи і необхідність реалізації покращеної версії проєкту. В процесі проєктування вебзастосунку необхідно уникати таких недоліків як надмірність інформаційного потоку і додавання зайвого функціоналу зі сторони користувача, що в кінці-кінців не зосереджує, а відволікає від основної задачі сайту.

2 ВИБІР ТА ОБҐРУНТУВАННЯ ЗАСОБІВ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Дле реалізації поставленного завдання необхідно обрати засоби та технології програмної реалізації, а саме: мова програмування, IDE (інтегроване середовище розробки), база даних, фреймворк.

2.1 Мова програмування при реалізації вебзастосунку

Мова програмування - це штучна мова, яка може використовуватися для керування поведінкою машини, зокрема комп'ютера. Мови програмування, як і людські мови, визначаються за допомогою синтаксичних і семантичних правил для визначення структури та значення відповідно. Це набір інструкцій, написаних будь-якою конкретною мовою (C, C++, Java, Python, PHP, JavaScript) для виконання певного завдання.

Вони використовуються для полегшення спілкування щодо завдання організації та маніпулювання інформацією, а також для точного вираження алгоритмів. Деякі автори обмежують термін «мова програмування» тими мовами, які можуть виражати всі можливі алгоритми.

JavaScript (JS) – це мова сценаріїв, яка використовується для створення та керування динамічними та інтерактивними вебелементами [7]. Це дозволяє сторінкам реагувати на події, демонструвати спеціальні ефекти, приймати змінний текст, перевіряти дані, створювати файли cookie, виявляти браузер користувача тощо.

JS вважається як клієнтською, так і серверною мовою. Більшість популярних вебсайтів, таких як Google, Facebook і YouTube, використовують JavaScript для перетворення статичних вебсторінок в інтерактивні, оскільки він вважається об’єткно-орінтовановою мовою комп’ютерного програмування. Все,

що оновлюється, переміщується або змінюється на вебсторінці, не вимагаючи від особи перезавантаження сторінки вручну, використовує JavaScript.

Переваги JavaScript:

1. Популярність. JavaScript існує вже більше 25 років. І це якраз той випадок, коли технологія не застаріла. Натомість, на шляху свого розвитку, мова програмування JavaScript здобула велике співтовариство підтримки, і кількість випадків її використання зростала. Це одна з найпопулярніших мов програмування, і завдяки оновленням популярних браузерів продуктивність JavaScript стає ще кращою з року в рік.
2. Простота. У світі налічується 13,8 мільйонів розробників JS, і їхня чисельність продовжує зростати через попит на послуги розробки JS і легкість освоєння цієї технології.
3. Універсальність. Використання одного лише JavaScript буде достатньо для створення як Front-end, так і Back-end частини програми. Більше того, JS ідеально сумісний з популярними браузерами; він дозволяє створювати адаптивний дизайн і, як правило, не залежить від платформи. Ось чому JavaScript є одним з найпопулярніших варіантів для кросплатформної розробки.
4. Креативність. JavaScript досить гнучкий; у ньому є багато простору для творчості та ефективних ідей інтерфейсу користувача для реалізації. Наприклад, JS ідеально працює для створення приголомшливих слайдерів, функцій перетягування, анімації та інших компонентів інтерфейсу.
5. Величезна кількість інструментів та доповнень. Список переваг JavaScript можна продовжувати нескінченно, але було б неправильно упустити той факт, що JS поставляється з багатьма інструментами та фреймворками, щоб розкрити його найкращі можливості.

Такий широкий спектр інструментів, розширень, фреймворків і бібліотек значно спрощує та прискорює процес розробки за допомогою JavaScript, що є однією з головних переваг.

PHP – це мова сценаріїв на стороні сервера з відкритим вихідним кодом, яку багато розробників використовують для веброзробки [8]. Це також мова загального призначення, яку можна використовувати для створення багатьох проєктів, включаючи графічні інтерфейси користувача (GUI).

PHP залишається популярною мовою програмування протягом майже трьох десятиліть завдяки ряду переваг, які він пропонує користувачам і розробникам. Найбільш значущими з них є:

1. Міжплатформенність. PHP не залежить від платформи. Вам не потрібно мати певну ОС і не доведеться турбуватися про сумісність, щоб використовувати її, оскільки вона працює на кожній платформі, Mac, Windows чи Linux. Це також означає, що ви можете працювати в команді проєкту і не турбуватися про те, що учасники зможуть отримати доступ до коду!
2. Open Source. PHP є відкритим вихідним кодом. Оригінальний код доступний для всіх, хто хоче розробити його. Це одна з причин, чому один з її фреймворків, наприклад Laravel або Yii2, настільки популярний.
3. Блискуче працює з HTML. PHP може допомогти спростити ваші проєкти та безперебійно працює з мовою гіпертекстової розмітки.
4. Легко освоїти. PHP не важко вивчити для абсолютних початківців. Через свою простоту її можна швидше та легше опанувати, ніж деякі інші альтернативи.
5. PHP синхронізується з усіма базами даних. Ви можете легко підключити PHP до всіх баз даних, реляційних і нереляційних. Тому він може швидко

підключитися до MySQL, Postgress, MongoDB або будь-якої іншої бази даних.

6. Існує багато доступних інструментів. Є інструменти, які допоможуть вам практично в будь-якій задачі з PHP, починаючи від інтеграції, підказки коду, підсвічування синтаксису тощо.
7. Швидке завантаження для вебсайтів. Оскільки продуктивність вебсайту все більше і більше залежить від швидкості, швидке завантаження PHP дійсно може допомогти вам досягти успіху.
8. Підтримуюча спільнота. PHP має дуже сприятливу онлайн-спільноту. Офіційна документація містить інструкції щодо використання функцій, і ви можете легко вирішити проблему з її допомогою.

2.2 Вибір інтегрованого середовища розробки

Інтегроване середовище розробки – це програмне забезпечення для створення програм, яке поєднує звичайні інструменти розробника в єдиний графічний інтерфейс користувача (GUI).

IDE зазвичай складається принаймні з редактора вихідного коду, засобів автоматизації збірки та налагоджувача. Деякі IDE, такі як NetBeans і Eclipse, містять необхідний компілятор, інтерпретатор або обидва; інші, такі як SharpDevelop і Lazarus, цього не роблять.

Розглянемо IDE PHPStorm, яке я обрала для подальшої роботи. PhpStorm – це інноваційне інтегроване середовище розробки на основі Java (IDE), розроблене JetBrains для PHP та веб-розробників. Він підтримує PHP різних версій, забезпечує запобігання помилкам на льоту, найкраще автозавершення та рефакторинг коду, що значно пришвидшить роботу, налагодження з нульовою конфігурацією та розширений редактор HTML, CSS та JavaScript. IDE забезпечує

інтелектуальне завершення коду, підсвічування синтаксису, розширену конфігурацію форматування коду, згортання коду, підтримує мовні суміші тощо. Автоматичний рефакторинг дбайливо ставиться до вашого коду, допомагаючи зробити глобальні налаштування проєкту легкими та безпечними.

2.3 Вибір системи керування базами даних

MySQL – це безкоштовна реляційна система керування базами даних (СУБД), розроблена компанією «TcX» для збільшення швидкості обробки великих баз даних. Ця структура управління факторами даних із відкритим шифром була створена як спосіб адаптування комерційних систем [9].

База даних – це структурований набір даних. Це може бути будь-що, від простого списку покупок до картинної галереї або місця для зберігання величезної кількості інформації в корпоративній мережі. Зокрема, реляційна база даних – це цифрове сховище, яке збирає дані та упорядковує їх відповідно до реляційної моделі. У цій моделі таблиці складаються з рядків і стовпців, а зв’язки між елементами даних мають сувору логічну структуру. СУБД — це просто набір програмних засобів, які використовуються для фактичної реалізації, керування та запитів такої бази даних.

MySQL був дуже схожий на MSQL з самого початку, але з часом він продовжував розширюватися, і зараз MySQL являється однією з найбільш відомих систем управління факторіями даний. Загалом він використовується для створення нестатичних вебсторінок, так як має стійку підтримку на базі різних мов програмування.

MySQL є одним із багатьох варіантів програмного забезпечення СУБД. Кілька великих веб-програм, таких як Facebook, Twitter, YouTube, Google і Yahoo! всі використовують MySQL для цілей зберігання даних. Незважаючи на

те, що спочатку він був створений для обмеженого використання, тепер він сумісний з багатьма важливими обчислювальними платформами, такими як Linux, macOS, Microsoft Windows і Ubuntu.

Можливості бази даних MySQL:

1. Лаконічність у встановленні та у використанні.
2. Необмежена кількість користувачів, що одночасно взаємодіють із БД.
3. Кількість рядків у таблицях обмежена, але можна досягати до 50 мільйонів.
4. Висока швидкість виконання завдань.
5. Проста і ефективна системи захисту заздалегідь вбудована у БД.

Усі вебсайти, крім найпростіших, використовують бази даних. Усе, що відбувається на вашому сайті, незалежно від його складності, записується в один. Однак іноді вам може знадобитися отримати доступ і взаємодіяти з вмістом вашої бази даних – а це часто важко без відповідних знань та інструментів. Ось тут на допомогу приходить PhpMyAdmin.

PhpMyAdmin – це інструмент з відкритим вихідним кодом, створений на основі PHP, який дає змогу адмініструвати бази даних MySQL та MariaDB онлайн. Щоб використовувати його, вам потрібно буде встановити програмне забезпечення на сервері під керуванням Windows або одного з кількох дистрибутивів Linux, які воно підтримує.

Ми можемо виконувати запити MySQL, ремонтувати, оптимізувати, перевіряти таблиці, а також виконувати інші команди керування базою даних. PhpMyAdmin також можна використовувати для виконання адміністративних завдань, таких як створення бази даних, виконання запитів. Сама програма проста для початківців, але вона пропонує достатню глибину, щоб вам знадобився деякий час, щоб освоїти все, що вона пропонує.

2.4 Обґрунтування доцільності використання фреймворку розробки

Фреймворк – це платформа для розробки програмних додатків. Він забезпечує основу, на якій розробники програмного забезпечення можуть створювати програми для певної платформи. Наприклад, фреймворк може включати попередньо визначені класи та функції, які можна використовувати для обробки введення, керування апаратними пристроями та взаємодії з системним програмним забезпеченням.

На PHP існує доволі велика кількість зручних фреймворків для реалізації проєктів. Серед них можна виділити: Laravel, SYMFONY 2, CodeIgnite, PHPixie, Yii2.

Поставлена задача буде реалізована за допомогою фреймворку Yii2.

Yii – це високопродуктивний, компонентний фреймворк PHP для швидкого розвитку сучасних вебзастосунків. Ім’я Yii можна вважати абревіатурою від Yes It Is!. Також він є об’єктно орієнтованим, тому що він використовує технологію завантаження за вимогою. Yii2 успадковує основний дух Yii як простий, швидкий і дуже розширюваний фреймворк PHP.

Плюси використання:

1. Yii – це загальна структура вебпрограмування.
2. Yii можна використовувати для розробки всіх видів вебзастосунків на основі PHP.
3. Yii – це архітектура на основі компонентів і складна підтримка кешування.
4. Yii особливо підходить для розробки великомасштабних програм. такі як портали, форуми, системи керування вмістом (CMS), проєкти електронної комерції, вебсервіси RESTful тощо.
5. Yii реалізує шаблон проєктування MVC (Model-View-Controller).

Висновки до розділу 2

В результаті даного розділу бакалаврської роботи було розглянуто різні технології та засоби програмної реалізації, які будуть взяті за основу даного проєкту. Для цього був обраний набір інструментів, які можуть забезпечити загальну функціональність, зручний інтерфейс та кроссплатформенність. Завдяки, аналізу отриманої інформації, було вирішено забезпечити усі перераховані критерії, реалізувавши локальний сервер, базу даних та вебзастосунок.

Враховуючи весь спектр проблем і вимог, було обрано рішення використати ддя реалізації поставленого завдання мову програмування PHP і JS та базу даних MySQL, а для більш зручної та високоефективної реалізації використаю PHP-фреймворк Yii2.

3 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Функціональна/процесна модель системи

Досліджуючи вже існуючі аналоги програмних продуктів для запису до електронної черги, є можливість виділити рішення, які вважаються більш вдалими на загальному фоні, та уникнути вже існуючих багів, що виникли при розробці даних вебзастосунків («Поліклініка без черг», «Helsi», «Askep.net»).

Для візуалізації функціональності ПЗ приведемо UML діаграму варіацій використання системи (рис. 2.1).

Рисунок 2.1 – UML діаграма варіацій використання системи

Створювана система для електронного запису до лікарні представлена у вигляді інтернет-додатка. Для розробки інтерфейсу (Front-end частини) використовується HTML, CSS та мова програмування JavaScript, а для Back-end – PHP з використанням фреймворку Yii2, як було зазначено у другому розділі.

При першому переході до вебзастосунку користувач потрапляє на головну сторінку, де зібрана основна інформація та надана поетапна інструкція по користуванню данною системою.

У верхній частині сайту знаходиться меню, яке можна використати в якості навігації для головної сторінки сайту. Перейшовши до «Статистики» та натиснувши кнопку «Проконсультуватись», у користувача є можливість за допомогою форми зворотнього зв'язку надіслати запитання або пропозицію до адміністрації сайту, заповниши усю необхідну контактну інформацію.

Усі запити користувачів проходять через БД MySQL, для перевірки на наявність зареєстрованого користувача. Якщо це перший вхід на сайт, то користувачу необхідно зареєструватися; для цього йому потрібно ввести наступні дані: «ПІБ», «Email» та «Пароль».

На рис. 2.2 відображено процес створення нового облікового запису.

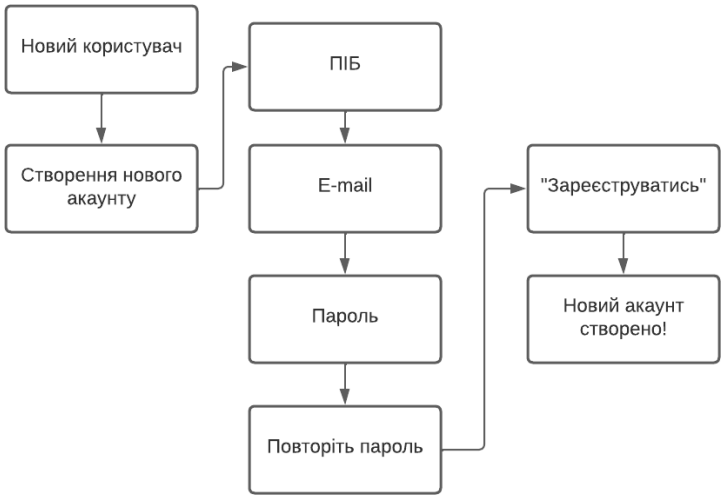

Рисунок 2.2 – Процес створення нового облікового запису

При реєстрації у вебзастосунку створюється новий обліковий запис, який зберігається у СУБД на локальному сервері OpenServer.

Після реєстрації у вебзастосунку необхідно авторизуватися, а саме заповнити два поля: «Email» та «Пароль». Система автоматично переведе користувача до особистого кабінету, де потрібно заповнити усю необхідну інформацію для доступу до онлайн-запису. Після успішного внесення змін, користувач може перейти на сторінку «Запис на прийом» для ознайомлення з інформацією і обрання медичного працівника за спеціалізацією (можливість вибору конкретного лікаря).

Наступний крок та фінальний зі сторони користувача полягає у тому, що після вибору лікаря, потрібно обрати зручний час для огляду/консультації за датою та часом. Виконавши все відповідно до прописаного сценарію, на сторінці «Особисті записи» з’явиться увесь обраний перелік даних, який буде чекати на підтвердження зі сторони адміністратора. Якщо все прошло вдало та адміністратор підтвердив ваш запис, то з’явиться кнопка «Талон», за допомогою якої можна завантажити електронний талон і зберегти, що є дуже зручно.

Також приємним доповненням є те, що для користувача відкриється можливість повноцінного використання функції – «Особистий календар», де можна відслідкувати ваші записи по датам, та й взагалі, переглядати записи упродовж усього часу.

На рис. 2.3 наведено алгоритм поведінки програмного застосунку зі сторони користувацького інтерфейсу та функіональних можливостей.

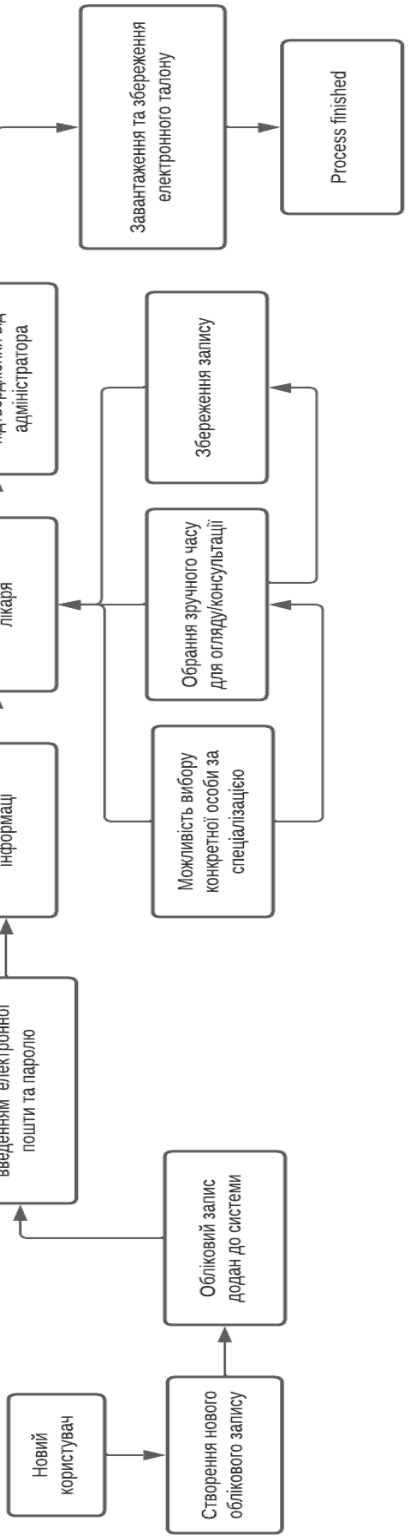

Рисунок 2.3 – Алгоритм поведінки програмного застосунку зі сторони користувача

При спробі увійти в обліковий запис, перевіряється чи існує даний користувач у системі. Якщо так – то його автоматично перенаправляють на сторінку особистого акаунту, а якщо ні – то користувачу пропонується виконати реєстрацію, процес якої був наведений вище у блок–схемі на рис. 2.2.

Система запису до електронної черги наділена спеціальним застосунком для адміністратора, що в свою чергу підтримує такі операції, як:

- управління обліковими записами користувачів. На рис. 2.4 наведено алгоритм поведінки програмного застосунку при управлінні обліковими записами користувачів;
- перегляд списку користувачів/лікарів;
- додавання/видалення медичних працівників (рис. 2.5);
- прийняття рішення щодо підтвердження/відхилення запису;
- відстеження всіх записів у системі;
- додавання/видалення спеціальностей.

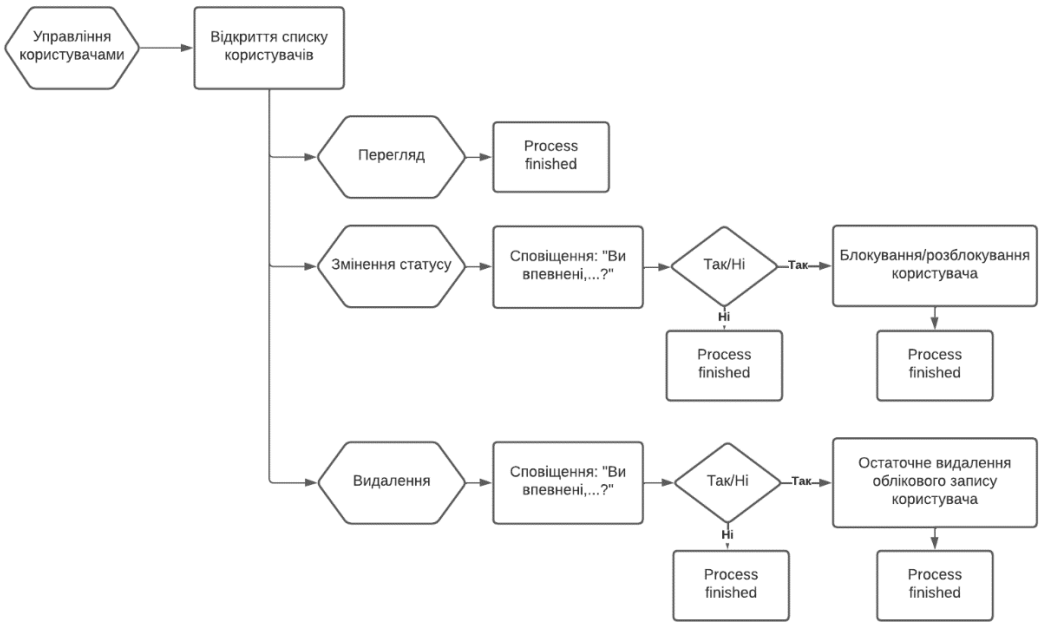

Рисунок 2.4 – Алгоритм поведінки програмного застосунку при управлінні обліковими записами користувачів

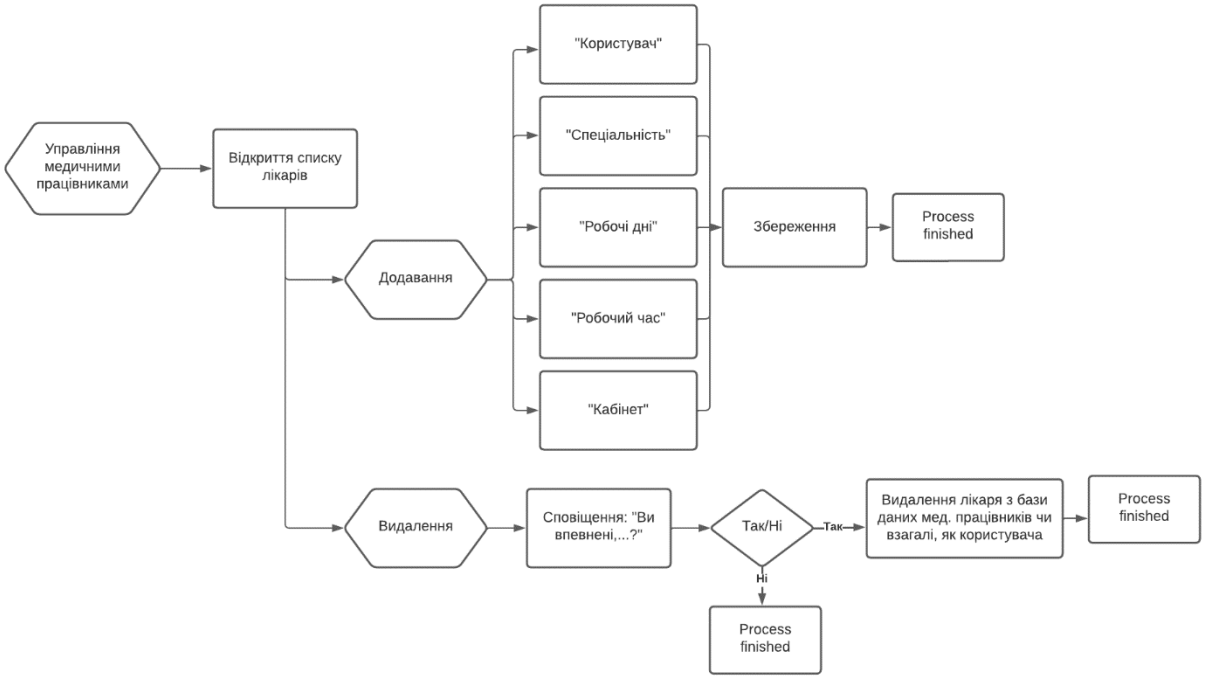

Рисунок 2.5 – Процес додавання/видалення медичних працівників

З блок-схеми зображеної на рис. 2.5 можна прийти до висновку, що таку інформацію як ПІБ, номер телефону та електронну пошту, медичний працівник вводить самостійно під час реєстрації та редагування даних у системі.

3.2 Розробка архітектури інформаційної системи

Як було зазначено, вебзастосунок – прикладне програмне забезпечення, логіка якого розподілена між клієнтом та сервером, що використовує в якості клієнта веббраузер і працює на стороні вебсервера.

Клієнт-серверна архітектура – це обчислювальна модель, в якій сервер розміщує, постачає та керує більшістю ресурсів і послуг, які споживає клієнт [12]. Цей тип архітектури має один або кілька клієнтських комп’ютерів, під’єднаних до центрального сервера через мережу або Інтернет.

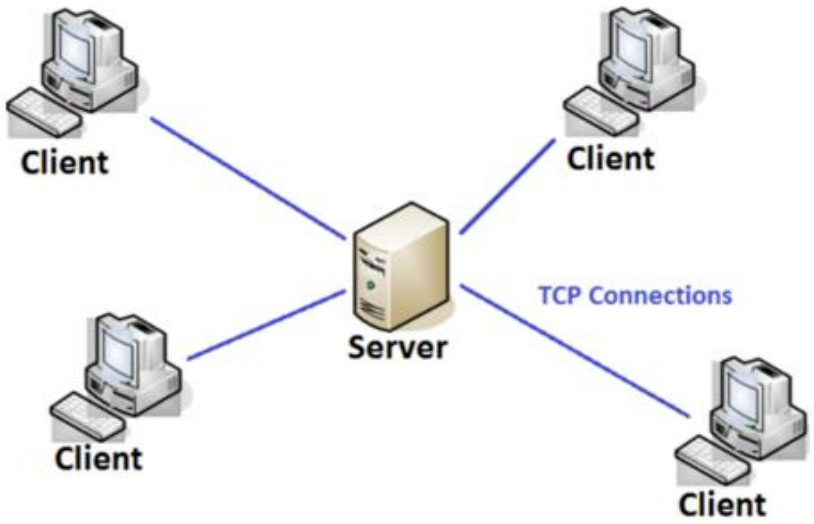

Рисунок 2.6 – Клієнт-серверна архітектура

Клієнт-серверній архітектурі найчастіше притаманно використовуватися в роботі з базами даних та мережі. Також вона забезпечує обмін даними між цими компонентами.

Архітектура клієнт-сервер передбачає наступні три компоненти:

- сервери, що обробляють отримані запити та видають відповідний результат. Сервер одночасно виконує різні типи ролей, наприклад поштовий сервер, сервер баз даних, файловий сервер або контролер домену;
- клієнти, що звертаються до серверів з запитами про дані;
- мережа, що встановлює з’єднання та забезпечує обмін даними між клієнтами і серверами.

На рисунку 2.7 зображено трирівневу схему архітектури вебзастосунку.

Рисунок 2.7 – Схема архітектури клієнт-сервер

Трирівнева схема архітектури вебзастосунку складається з наступного:

- 1-ий рівень – клієнт, який взаємодіє з користувачем. Клієнтом виступає веббраузер.
- 2-ий рівень – сервер додатків, який містить бізнес-логіку програми.
- 3-ий рівень – менеджер ресурсів (СУБД), який зберігає або повертає дані на сервер.

Розробка архітектури інтерактивного вебсервісу для електронного запису до лікарні полягає у подальшому використанні фреймворку Yii2, програми якого організовано відповідно до архітектурного патерну MVC.

MVC (Model-View-Controller) – це архітектура або, можна сказати, шаблон розробки програмного забезпечення. Ідея полягає у тому, щоб розділити програму на три різні компоненти, а саме: модель, представлення і контролер. MVC є корисним шаблоном проєктування для повторного використання коду і скорочує час розробки програми [13].

Нижче наведено схематичне представлення структури MVC (рис. 2.8).

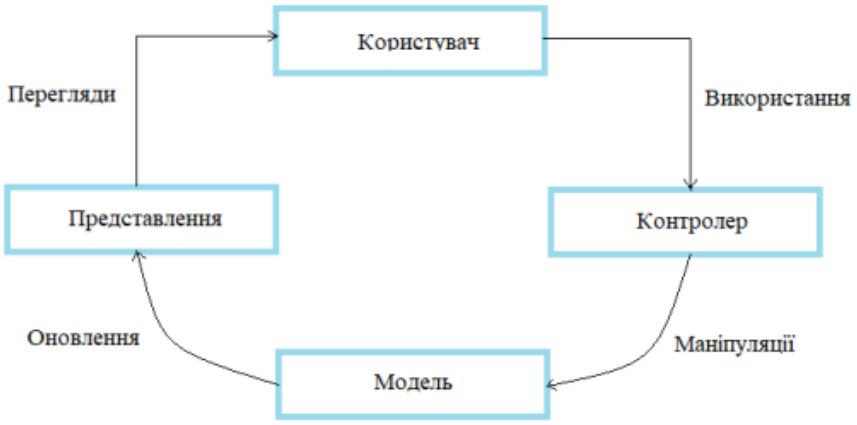

Рисунок 2.8 – Структура MVC

Виходячи з цього, можна прийти до висновку, що папка «models» складається з моделі проєкту, «views» містить представлення, а «controllers» усі контролери реалізованого проєкту.

На наступній схемі (рис. 2.9) зображено типовий робочий процес програми Yii, коли вона обробляє запит користувача:

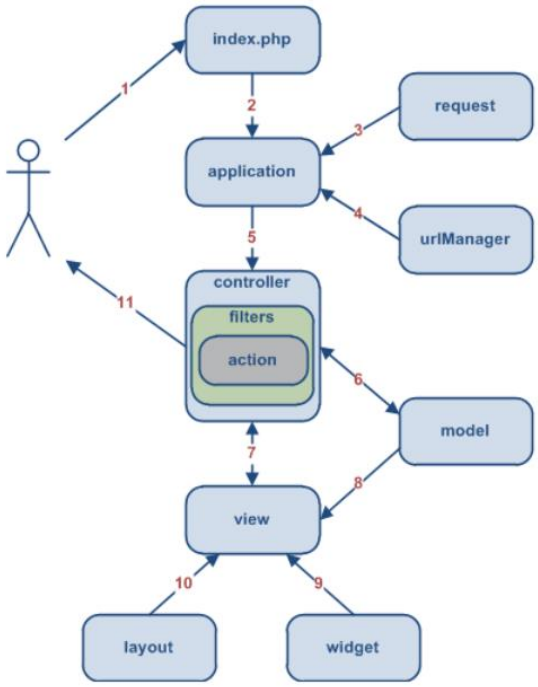

Рисунок 2.9 – Оброблення запиту користувача

1. Користувач надсилає запит за URL-адресою http://www.example.com/index.php, а вебсервер обробляє його, виконуючи сценарій завантаження index.php.
2. Сценарій завантаження створює екземпляр програми та запускає його.
3. Програма отримує детальну інформацію про запит користувача від компонента програми.
4. Програма визначає потрібний контролер і дію за допомогою компонента програми з назвою urlManager..
5. Програма створює екземпляр запитуваного контролера для подальшої обробки запиту користувача. Контролер визначає, що показ дії посилається на метод з назвою actionShow у класі контролера. Потім він створює та виконує фільтри (наприклад, контроль доступу, порівняльний аналіз), пов’язані з цією дією. Дія виконується, якщо це дозволено фільтрами.
6. Дія зчитує модель Post, ідентифікатор якої дорівнює 1 з бази даних.
7. Дія відтворює представлення з назвою show з моделлю Post.
8. Представлення зчитує та відображає атрибути моделі Post.
9. Представлення виконує деякі віджети.
10. Результат візуалізації представлення вбудовується в макет.
11. Дія завершує візуалізацію представлення та відображає результат користувачеві.

На рисунку 2.10 зображена діаграма станів процесу під час проходження реєстрації в системі.

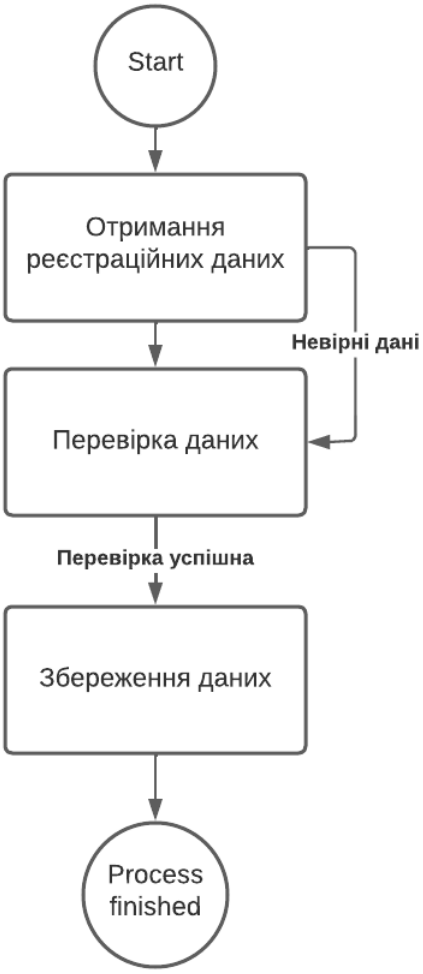

Рисунок 2.10 – Діаграма станів процесу під час реєстрації

Даний вид діаграми використовується для надання абстрактного опису поведінки системи. Ця поведінка є наслідком аналізу та відтворення послідовності подій, які відбуваються в одному або кількох можливих станах системи.

- Стан процесу: відображає поточний статус процесу. Він може бути новим, готовим, запущеним або чекає.
- Лічильник програми: вказує адресу наступної інструкції, яка буде виконана для цього процесу.

- Регістри ЦП: вони включають індексні регістри, покажчик стека та регістри загального призначення. Він використовується для збереження стану процесу, коли виникає переривання, щоб він міг відновитися з цього стану.
- Інформація про планування ЦП: вона включає пріоритет процесу, покажчик на чергу планування.
- Інформація про управління пам'яттю: значення базового та граничного регістрів, таблиці сторінок залежно від системи пам'яті.
- Облікова інформація: містить кількість використовуваного ЦП і реального часу, ліміти часу, номер процесу тощо.
- Інформація про стан вводу/виводу: містить список пристроїв вводу/виводу, призначених для процесу, список відкритих файлів тощо.

Під час реєстрації вебсистема перевіряє на коректність та відповідність введені дані за допомогою валідації (регулярних виразів), а також визначає чи не існує користувача з зазначеною у реєстраційній формі інформацією. Після успішної реєстрації/авторизації дані користувача будуть збережені в системі, та за необхідності користувач зможе змінити пароль у особистому кабінеті.

На рисунку 2.11 наведена діаграма станів процесу під час запису користувачів на прийом.

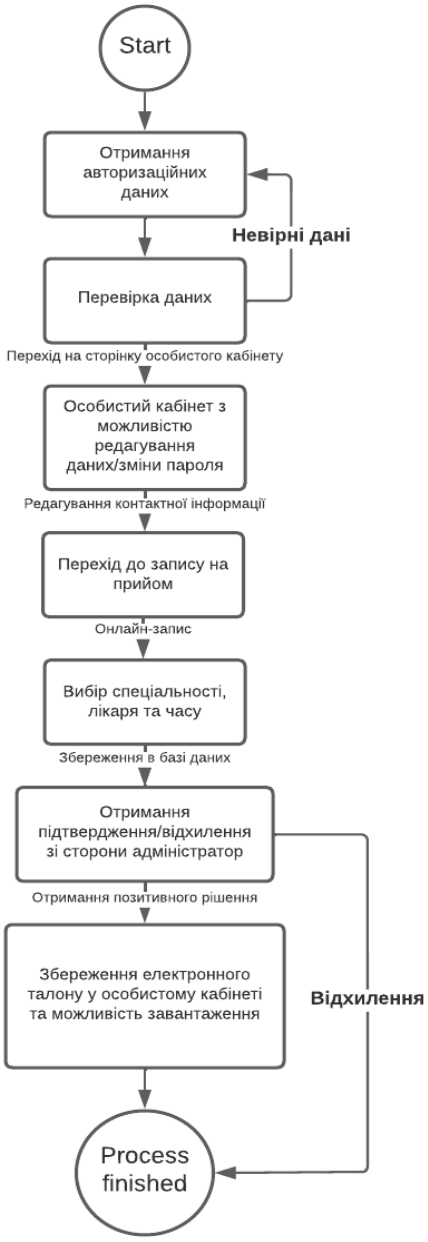

Рисунок 2.11 – Діаграма станів процесу запису на прийом

Процес запису (реєстрації) пацієнтів на прийом до медичного спеціаліста може бути здійснений лише після проходження аутентифікації. Після цього система здійснить автоматичний перехід до особистого кабінету, а слідом до можливості електронного запису. Пацієнт обирає за допомогою фільтру

спеціальність, лікаря та зручний час. Завдяки доволі широкого спектру часу, користувач може обрати той, що є зручним і задовольняє його планування дня.

Що ж до сторони адміністратора, то тут загалом усе просто. Авторизаційні дані, які отримує система, перевіряються на відповідність авторизаційним даним користувача типу «Адміністратор». Після того як перевірка успішно пройдена, відкривається вебсистема з різноманітною функціональною можливістю (описано у підпункті 2.1 «Перелік вимог до програмного продукту»). Усі видозмінення всередині системи проходять поетапну перевірку та зберігаються в базі даних.

3.3 Проєктування структури бази даних

Для відображення взаємозв’язків та внутрішньої «начинки» вебзастосунку було прийнято рішення обрати реляційну модель даних.

Реляційна модель даних є основною моделлю даних, яка широко використовується в усьому світі для зберігання та обробки інформації. Ця модель проста і має всі властивості та можливості, необхідні для ефективної обробки даних.

Під час проєктування структури бази даних було визначено елементи та зв’язки для подальшого створення ER-діаграми [14].

Діаграма «сутність-зв’язок» (ERD) – це тип блок-схеми, яка ілюструє, яким чином «суб’єкти», такі як люди, об’єкти або поняття, пов’язані один з одним у системі. ER-діаграми найчастіше використовуються для проєктування або налагодження реляційних баз даних у сферах програмної інженерії, бізнес-інформаційних систем, освіти та досліджень. Вони використовують певний набір символів, таких як прямокутники, ромби, овали та сполучні лінії, щоб відобразити взаємозв’язок сутностей, відносин та їх атрибутів. Вони

відображають граматичну структуру з сутностями як іменники, а відносини як дієслова. ER-діаграма вебзастосунку для елекронного запису до лікарні наведена на рис. 2.12.

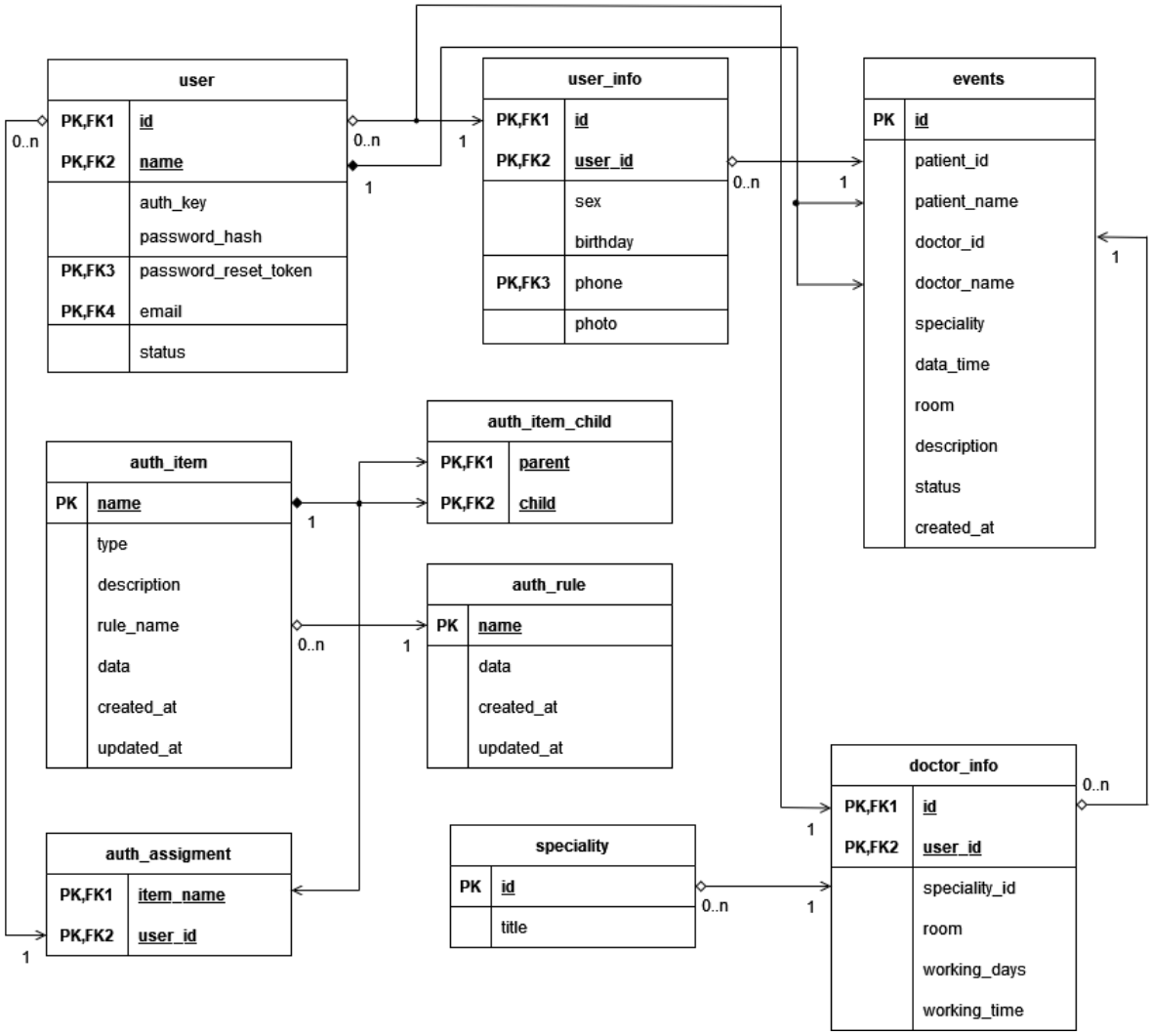

Рисунок 2.12 – Зображення ER-діаграми

На представленій діаграмі можна прослідкувати наступні відношення: auth_assignment, auth_item, auth_item_child, auth_rule, doctor_info, events, speciality, user і user_info.

В основу проєктування структури бази даних закладено створення таблиць у БД, відповідно до ER-діаграми.

Приклад лістингу DDL коду для створення таблиці в БД наведено нижче.

```
CREATE TABLE `auth_assignment` (
  `item_name` varchar(64) COLLATE utf8_unicode_ci NOT NULL,
  `user_id` varchar(64) COLLATE utf8_unicode_ci NOT NULL,
  `created_at` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

Структура таблиці «auth_assignment» зображена на рисунку 2.13.

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
1	item_name	varchar(64)	utf8_unicode_ci		Ні	*Немає*		
2	user_id	varchar(64)	utf8_unicode_ci		Ні	*Немає*		
3	created_at	int(11)			Так	*NULL*		

Рисунок 2.13 – Структура таблиці «auth_assignment»

Таблиця «auth_assignment» має наступний вміст:

1. item_name – роль користувача. Тип даних varchar(64);
2. user_id – ідентифікатор користувача. Тип даних varchar(64);
3. created_at – дата створення. Тип даних ineteger.

Структура таблиці «auth_item» зображена на рисунку 2.14.

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
1	name	varchar(64)	utf8_unicode_ci		Ні	*Немає*		
2	type	smallint(6)			Ні	*Немає*		
3	description	text	utf8_unicode_ci		Так	*NULL*		
4	rule_name	varchar(64)	utf8_unicode_ci		Так	*NULL*		
5	data	blob			Так	*NULL*		
6	created_at	int(11)			Так	*NULL*		
7	updated_at	int(11)			Так	*NULL*		

Рисунок 2.14 – Структура таблиці «auth_item»

Таблиця «auth_item» має наступний вміст:

1. name – назва ролі. Тип даних varchar;
2. type – тип. Тип даних smallint;

3. description – опис. Тип даних text;
4. rule_name – дозволи для кожного типу ролей. Тип даних varchar(64);
5. data – дані. Тип даних blob;
6. created_at – дата створення. Тип даних integer;
7. updated_at – дата редагування. Тип даних integer.

Структура таблиці «auth_item_child» зображена на рисунку 2.15.

	#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
☐	1	**parent** 🔑	varchar(64)	utf8_unicode_ci		Ні	*Немає*		
☐	2	**child** 🔑 🔗	varchar(64)	utf8_unicode_ci		Ні	*Немає*		

Рисунок 2.15 – Структура таблиці «auth_item_child»

Таблиця «auth_item_child» має наступний вміст:

1. parent – батьківський елемент відношення. Тип даних varchar(64);
2. child – дочірній елемент відношення. Тип даних varchar(64).

Коротко кажучи, ця таблиця складається з успадкування ролей та дозволів один від одного.

Структура таблиці «auth_rule» зображена на рисунку 2.16.

	#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
☐	1	**name** 🔑	varchar(64)	utf8_unicode_ci		Ні	*Немає*		
☐	2	**data**	blob			Так	*NULL*		
☐	3	**created_at**	int(11)			Так	*NULL*		
☐	4	**updated_at**	int(11)			Так	*NULL*		

Рисунок 2.16 – Структура таблиці «auth_rule»

Треба зазначити спираючись на структуру бази даних, що функціональні можливості вебзастосунку залежать від зазначеного типу користувача, а саме адміністратора, лікаря чи користувача.

Таблиця «auth_rule» має наступний вміст:

1. name – назва. Тип даних varchar(64);

2. data – дані. Тип даних blob.
3. created_at – дата створення. Тип даних integer;
4. updated_at – дата редагування. Тип даних integer.

Структура таблиці «doctor_info» зображена на рисунку 2.17.

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
1	id	int(11)			Ні	*Немає*	id	AUTO_INCREMENT
2	user_id	int(11)			Ні	*Немає*	Користувач	
3	speciality_id	int(11)			Ні	*Немає*	Спеціальність	
4	room	varchar(50)	utf8_general_ci		Ні	*Немає*	Кабінет	
5	working_days	text	utf8_general_ci		Ні	*Немає*	Робочі дні	
6	working_time	varchar(15)	utf8_general_ci		Ні	*Немає*	Робочий час	

Рисунок 2.17 – Структура таблиці «doctor_info»

Таблиця «doctor_info» має наступний вміст:

1. id – унікальний ідентифікатор. Тип даних integer;
2. user_id – ідентифікатор лікаря. Тип даних integer;
3. speciality_id – ідентифікатор спеціальності. Тип даних integer;
4. room – номер кабінету. Тип даних varchar(50);
5. working_days – графік роботи (робочі дні). Тип даних text.
6. working_time – графік роботи (робочий час). Тип даних varchar(15).

Структура таблиці «events» зображена на рисунку 2.18.

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
1	id	int(11)			Ні	*Немає*	ID	AUTO_INCREMENT
2	patient_id	int(11)			Ні	*Немає*	ID пацієнта	
3	patient_name	varchar(255)	utf8_general_ci		Ні	*Немає*	ПІБ пацієнта	
4	doctor_id	int(11)			Ні	*Немає*	ID лікаря	
5	doctor_name	varchar(255)	utf8_general_ci		Ні	*Немає*	ПІБ лікаря	
6	speciality	varchar(255)	utf8_general_ci		Ні	*Немає*	Спеціальність	
7	date_time	datetime			Ні	*Немає*	Дата та час	
8	room	varchar(50)	utf8_general_ci		Ні	*Немає*	Кабінет	
9	description	text	utf8_general_ci		Ні	*Немає*	Примітка	
10	status	enum('Новий запис', 'Підтверджено', 'Відхилено')	utf8_general_ci		Ні	Новий запис	Статус	
11	created_at	timestamp			Ні	CURRENT_TIMESTAMP	Створено	

Рисунок 2.18 – Структура таблиці «events»

Таблиця «events» має наступний вміст:

1. id – унікальний ідентифікатор. Тип даних integer;
2. patient_id – ідентифікатор пацієнта. Тип даних integer;
3. patient_name – ПІБ пацієнта. Тип даних varchar(256);
4. doctor_id – ідентифікатор лікаря. Тип даних integer;
5. doctor_name – ПІБ лікаря. Тип даних varchar(256);
6. speciality – спеціальність. Тип даних varchar(256);
7. date_time – дата та час запису. Тип даних datetime;
8. room – кабінет. Тип даних varchar(50);
9. description – примітка. Тип даних text.
10. status – статус запису на прийом. Тип даних enum(«Новий запис», «Підтверджено», «Відхилено»);
11. created_at – дата реєстрації запису. Тип даних timestamp.

Структура таблиці «speciality» зображена на рисунку 2.19.

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
1	id	int(11)			Ні	*Немає*	ID	AUTO_INCREMENT
2	title	varchar(255)	utf8_general_ci		Ні	*Немає*	Назва	

Рисунок 2.19 – Структура таблиці «speciality»

Таблиця «speciality» має наступний вміст:

1. id – унікальний ідентифікатор. Тип даних integer;
2. title – назва спеціальності. Тип даних varchar(256).

Структура таблиці «user» зображена на рисунку 2.20.

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
1	id	int(11)			Ні	*Немає*	ID	AUTO_INCREMENT
2	name	varchar(255)	utf8_unicode_ci		Ні	*Немає*	Логін	
3	auth_key	varchar(32)	utf8_unicode_ci		Ні	*Немає*	Ключ авторизації	
4	password_hash	varchar(255)	utf8_unicode_ci		Ні	*Немає*	Пароль	
5	password_reset_token	varchar(255)	utf8_unicode_ci		Так	*NULL*	Токен скидування пароля	
6	email	varchar(255)	utf8_unicode_ci		Ні	*Немає*	E-mail	
7	status	smallint(6)			Ні	10	Статус	

Рисунок 2.20 – Структура таблиці «user»

Таблиця «user» має наступний вміст:

1. id – унікальний ідентифікатор. Тип даних integer;
2. name – логін користувача. Тип даних varchar(255).
3. auth_key – ключ авторизації. Тип даних varchar(32).
4. password_hash – пароль. Тип даних varchar(255).
5. password_reset_token – токен скидування пароля. Тип даних varchar(255).
6. email – електронна пошла (E-mail). Тип даних varchar(255).
7. status – статус користувача. Тип даних smallint.

Структура таблиці «user_info» зображена на рисунку 2.21.

	#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
☐	1	id	int(11)			Ні	*Немає*	ID	AUTO_INCREMENT
☐	2	user_id	int(11)			Ні	*Немає*	Користувач	
☐	3	sex	enum('Чоловіча', 'Жіноча')	utf8_general_ci		Ні	*Немає*	Стать	
☐	4	birthday	varchar(10)	utf8_general_ci		Ні	*Немає*	День народження	
☐	5	phone	varchar(20)	utf8_general_ci		Ні	*Немає*	Телефон	
☐	6	photo	varchar(50)	utf8_general_ci		Так	*NULL*	Фото	

Рисунок 2.21 – Структура таблиці «user_info»

Таблиця «user_info» має наступний вміст:

1. id – унікальний ідентифікатор. Тип даних integer;
2. user_id – ідентифікатор користувача. Тип даних integer;
3. sex – стать. Тип даних enum(«Чоловіча», «Жіноча»);
4. birthday – день народження. Тип даних varchar(10);
5. phone – номер телефону. Тип даних varchar(10);
6. photo – фото в особистому кабінеті. Тип даних varchar(50).

Висновки до розділу 3

Отже, у даному розділі була описана функціональна модель програми та її реалізація як з боку користувача, так і з точки зору адміністратора, котрий буде займатися налаштуванням системи та її працездатністю.

Після аналізу вимог, що були створені під час розгляду існуючих аналогів в першому розділі, було сформовано ряд функціональних вимог до розроблюваної системи. Вони поєднали в собі як потреби користувачів у візуально зрозумілому інтерфейсі, так і в необхідних функціональних можливостях даного вебзастосунку.

Було описано структуру додатку, розглянуто архітектурний патерн MVC (Model-View-Controller) та описано поведінку програми при обробці запиту користувача.

Внаслідок формування чітких вимог була розпочата безпосередня розробка підсистем що підтримують працездатність вебзастосунку. Проведена розробка структури бази даних, а також процес підключення до даного проєкту. Фінальний варіант розробленого вебзастосунку повністю відповідає завданням, котрі були поставлені на початку реалізації та роботи над проєктом.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ПЕРЕВІРКА ПРАЦЕЗДАТНОСТІ ВЕБЗАСТОСУНКУ

4.1 Старт роботи з Yii2

Найпростіший спосіб розпочати роботу з Yii2 – це використовувати базовий шаблон програми, наданий командою Yii2. Цей шаблон доступний для завантаження, використовуючи утиліту Composer або завантаживши архів.

Composer – це інструмент для управління залежностями в PHP. Він дозволяє оголошувати бібліотеки, від яких залежить ваш проєкт, і керуватиме (встановлювати/оновлювати) ними за вас [15].

При розгляді цих двох способів було прийнято рішення віддати перевагу першому, оскільки Composer за замовчуванням не встановлює нічого глобально. Таким чином, це менеджер залежностей. Однак, він надає змогу оновлення застосунку або додання нових розширень використовуючи одну глобальну команду.

Першим етапом є встановлення утиліти Composer. Ця процедура є простою і під час її виконання не повинно виникнути питань.

Пісдя цього виконується завантаження самого шаблону за допомогою командного рядка, і допоможе нам в цьому наступна команда:
composer create-project --prefer-distyiisoft/yii2- app-"назва директорії" "назва директорії" (рис. 4.1).

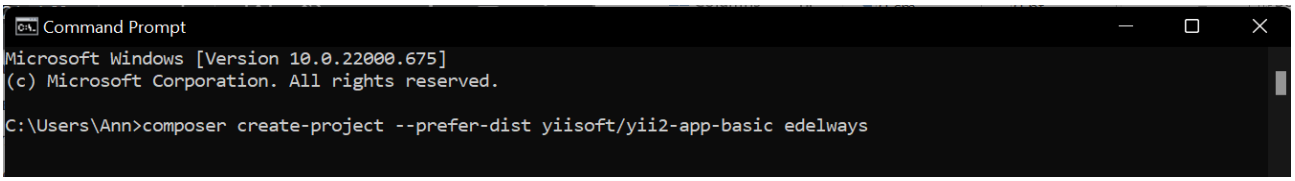

Рисунок 4.1 – Завантаження шаблону Yii2

Якщо все пройшло вдало, то перейшовши за адресою http://localhost/(назва директорії)/web/index у своєму браузері, можна побачити сторінку привітання (рис. 4.2).

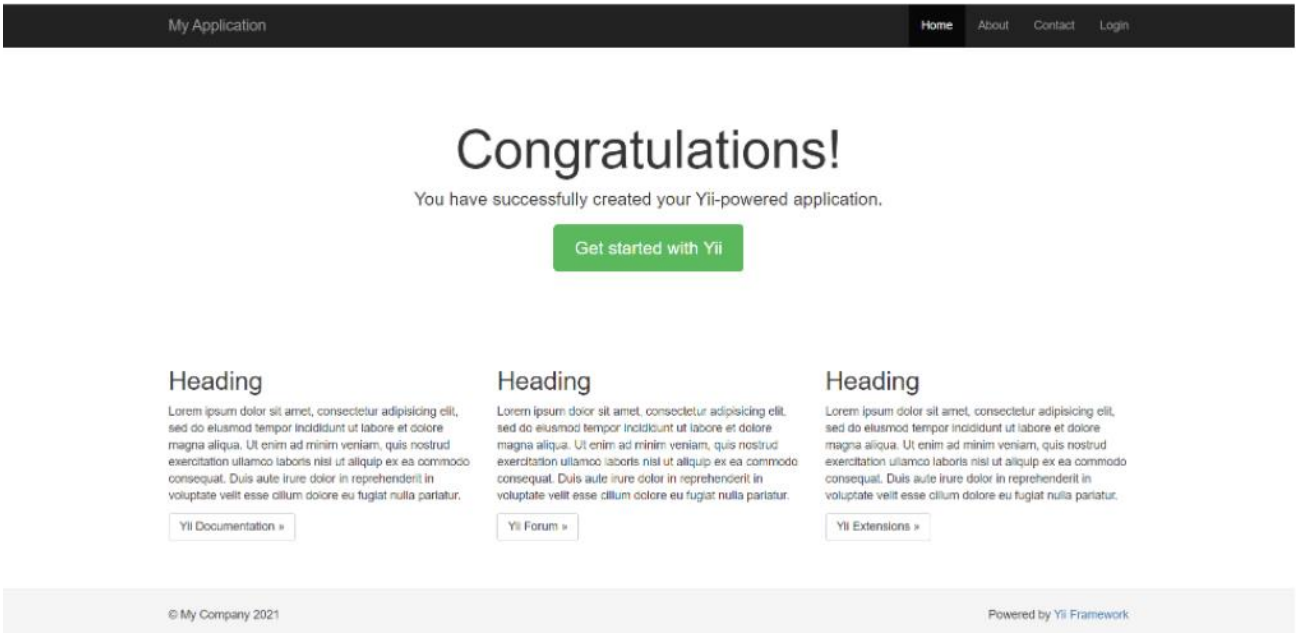

Рисунок 4.2 – Сторінка привітання Yii2

Структура додатку Yii2 є доволі простою і зрозумілою, та складається з наступних папок:

- assets – містить усі файли .js і .css, на які посилаються на вебсторінці;
- commands – консольні команди;
- config – містить файли конфігурації (console.php, db.php, params.php, web.php);
- controllers – містить класи контролерів;
- mail – ця папка містить макет електронної пошти;
- models – містить моделі, які використовуються в програмі;
- runtime – папка призначена для зберігання файлів, створених під час роботи ;
- tests – містить усі тести (прийомні, одиничні, функціональні);

- vendor – ця папка містить усі пакети сторонніх розробників, якими керує Composer;
- views – представлення додатку;
- web – основна папка вебзастосунку.

У загальній базі коду є лише одна папка, яка є загальнодоступною для вебсервера. Це саме вебкаталог (web). Інші папки за межами кореневого вебкаталогу не доступні для вебсервера, так як складаються зі службової інформації.

Усі залежності проєкту розташовані у файлі composer.json. Yii2 має кілька важливих пакетів, які вже включені у ваш проєкт від Composer:

1. Gii – інструмент для створення коду.
2. Консоль налагодження (модуль Debug).
3. Структура тестування Codeception.
4. Бібліотека SwiftMailer.
5. Бібліотека інтерфейсу користувача Twitter Bootstrap.

Перші три пакети корисні лише в середовищі розробки.

4.2 Керівництво користувача та адміністратора

У системі присутні два класи користувачів: адміністратор та користувачі [16]. Функціонал відрізняється різними рівнями доступу та можливостей у системі. Для початку розглянемо користувацьку частину вебзастосунку.

На рис. 4.3, 4.4 наведено інтерфейс головної сторінки розробленої системи, а саме вебзастосунку для електронного запису до лікарні. Для доступу до усього переліку функціональних можливостей, перш за все користувачу необхідно зареєструватись на сайті. Кнопка «Приєднатись» відповідає за процес реєстрації (рпс. 4.3).

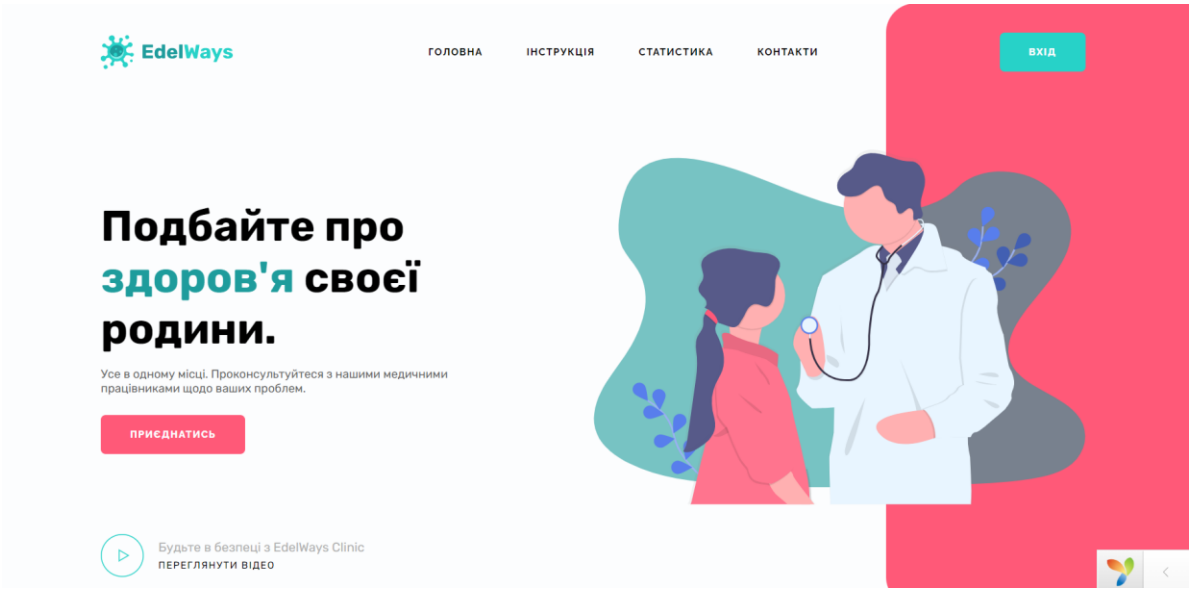

Рисунок 4.3 – Головна сторінка вебзастосунку

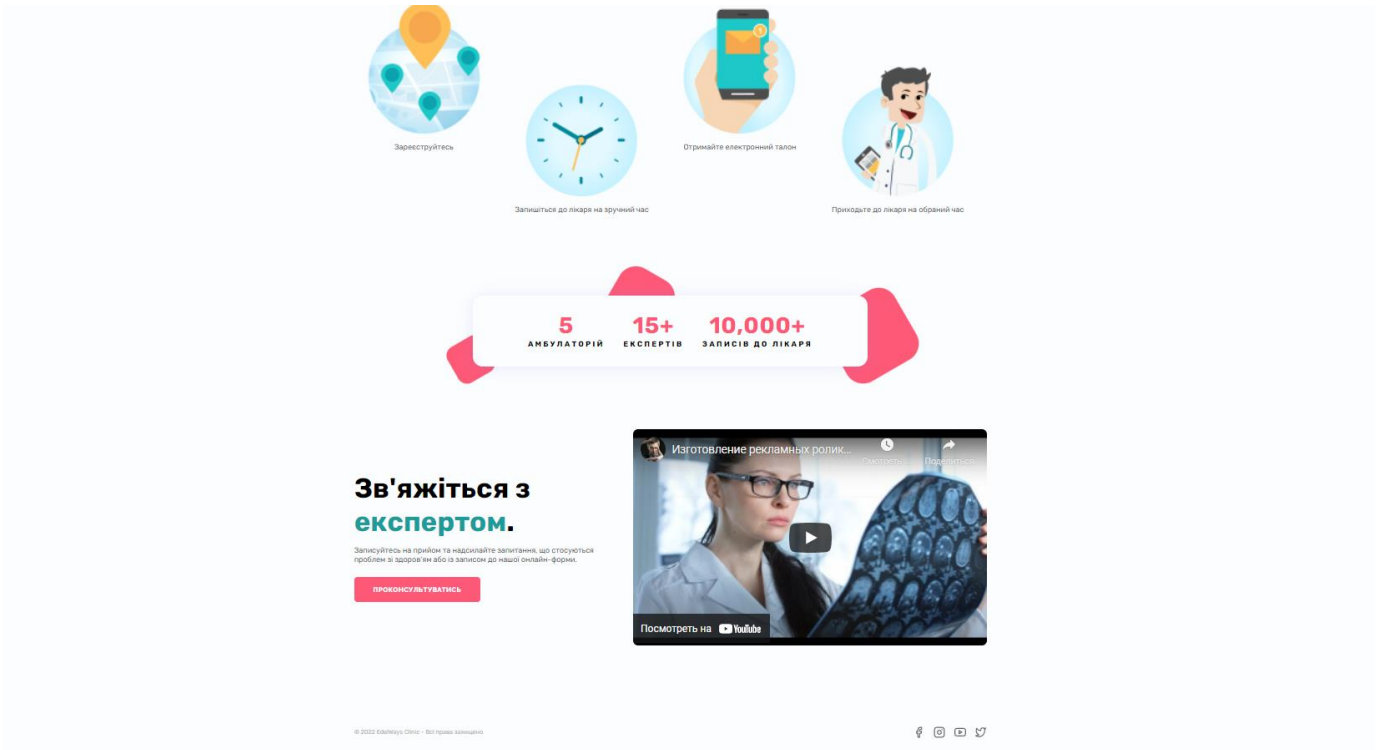

Рисунок 4.4 – Продовження головної сторінки вебзастосунку

Реєстрація у системі займає декілька хвилин та передбачає заповнення наступних даних:

- ПІБ;

- E-mail;
- пароль.

Слід зазначити, що при некоректному заповненні інформації або присутності незаповнених полів, буде виведено помилки, які призведуть до невдалого завершення процесу реєстрації.

Сторінка реєстрації нового акаунту зображена на рис. 4.5.

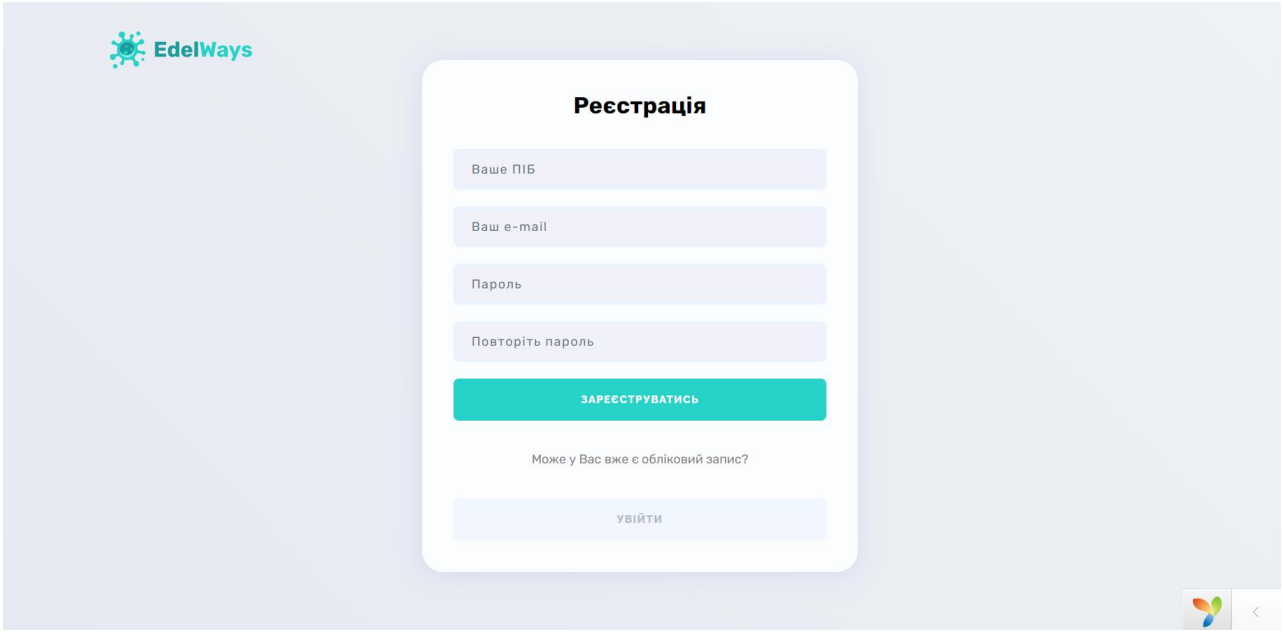

Рисунок 4.5 – Створення нового акаунту

Після вдалого створення облікового запису або якщо його було створено раніше, необхідно авторизуватись у системі, натиснувши кнопку «Вхід», та заповнити всі зазначені поля (рис. 4.6).

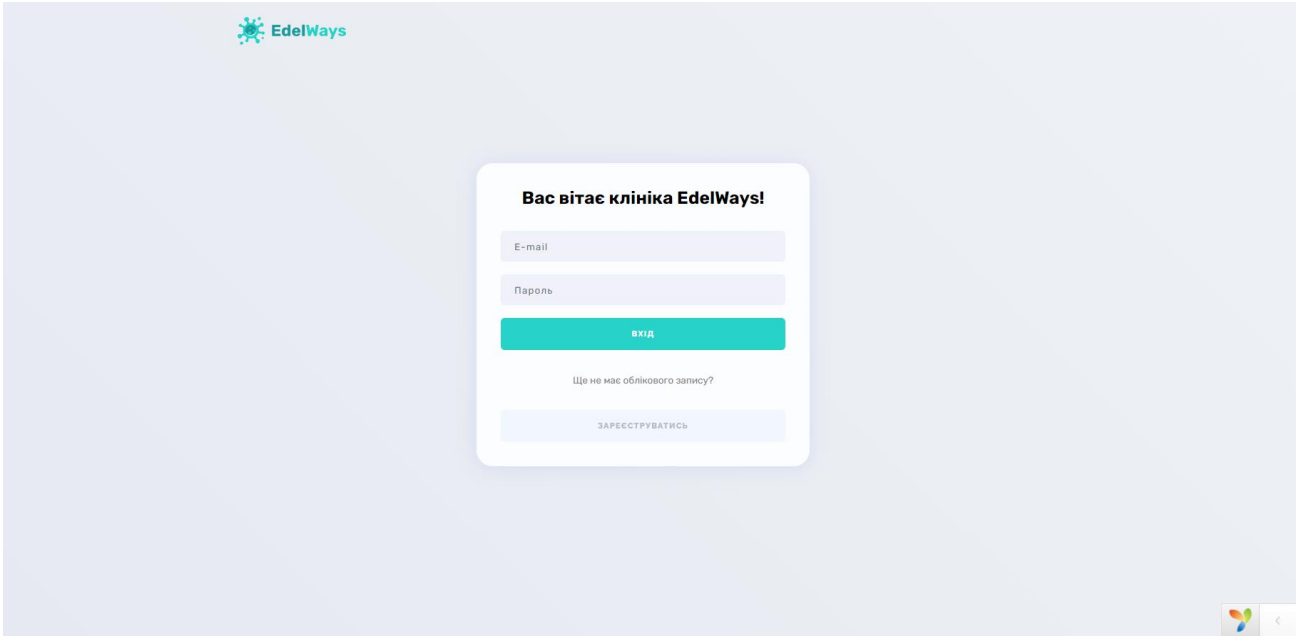

Рисунок 4.6 – Вхід в акаунт

За умови, якщо користувач ввів усе правильно та система перевірила його наявність в базі даних, буде здійснена переадресація до особистого кабінету користувача (рис 4.7), звідкіля й починається ознайомлення з вебзастосунком та його переліком функціональних можливостей.

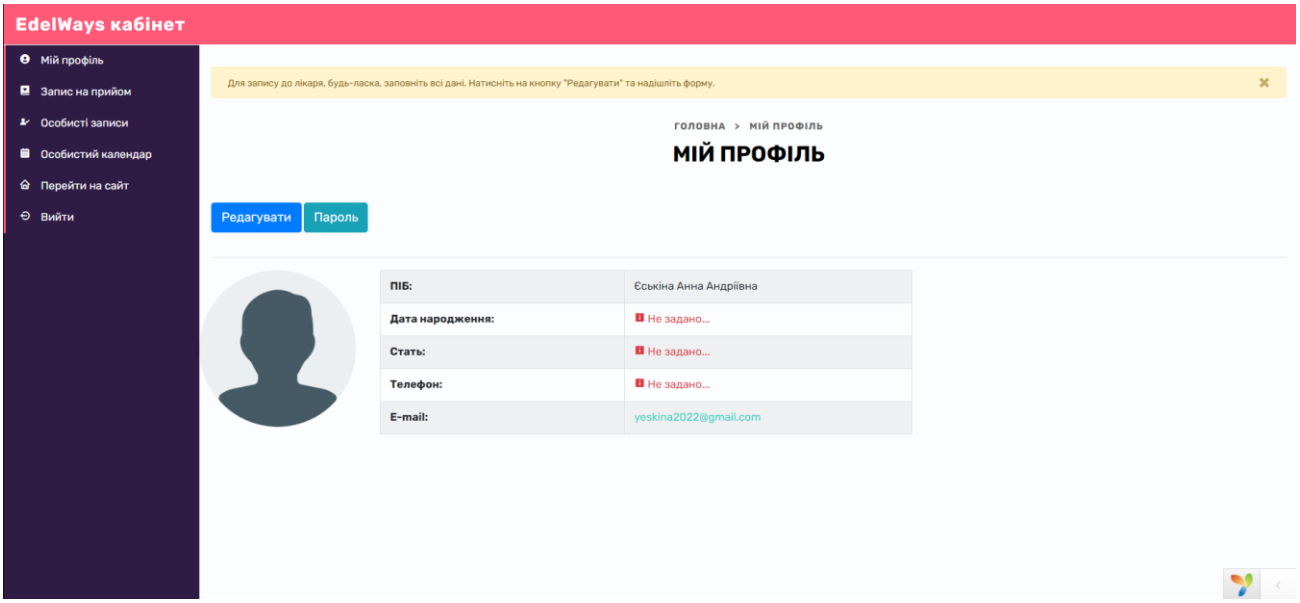

Рисунок 4.7 – Особистий кабінет користувача

Увійшовши до системи, можна побачити повідомлення, яке наголошує на тому, що для запису до лікаря, потрібно заповнити дані, а саме: «Дата народження», «Стать», «Телефон» та за власним бажанням додати фотографію профіля. Для цього необхідно натиснути кнопку «Редагувати», заповнити усю необхідну інформацію та зберегти форму (рис. 4.8).

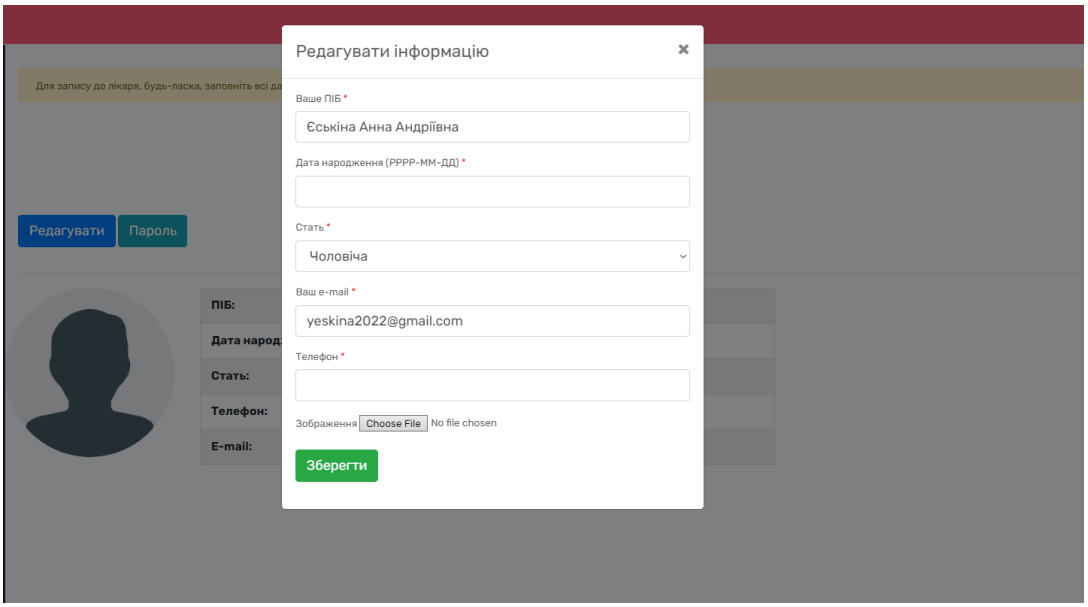

Рисунок 4.8 – Заповнення контактної форми

Натиснувши кнопку «Зберегти», буде видано інформацію стосовно того, що профіль успішно відредаговано, та відкриється можливість запису на прийом. Також я не можу упустити момент і не повідомити, що у особистому кабінеті є функціональна можливість щодо зміни паролю.

При переході до сторінки «Запис на прийом» (рис. 4.9) перед користувачем відображається стрічка, де можна побачити форму реєстрації запису: «Спеціальність», «Лікар», «Дата та час», «Кабінет» (усі пункти є обов’язковими для заповнення, окрім кабінету – автоматичне заповнення). «Примітка» не є обов’язковою, але за потреби користувач може вказати мету запису, симптоматику тощо.

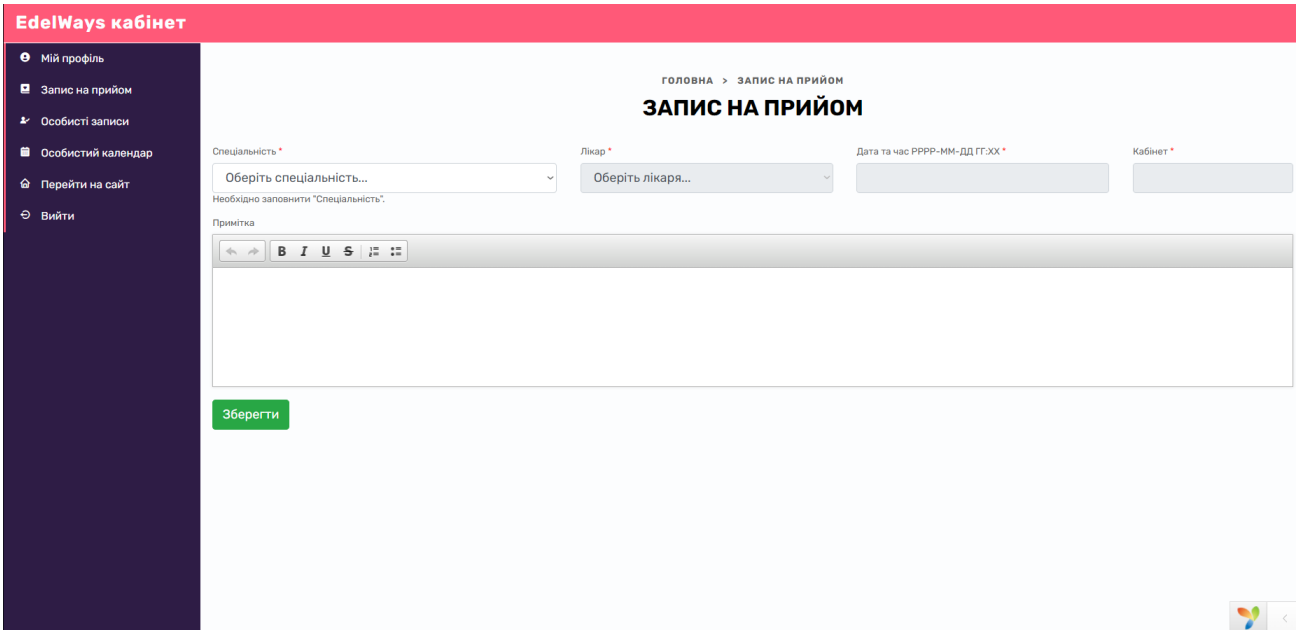

Рисунок 4.9 – Сторінка «Запис на прийом»

Обравши спеціальність та медичного працівника з випадаючого списку, користувач зможе переглянути графік роботи спеціаліста (рис. 4.10) і відштовхуючись від цього, визначити дату та час свого візиту.

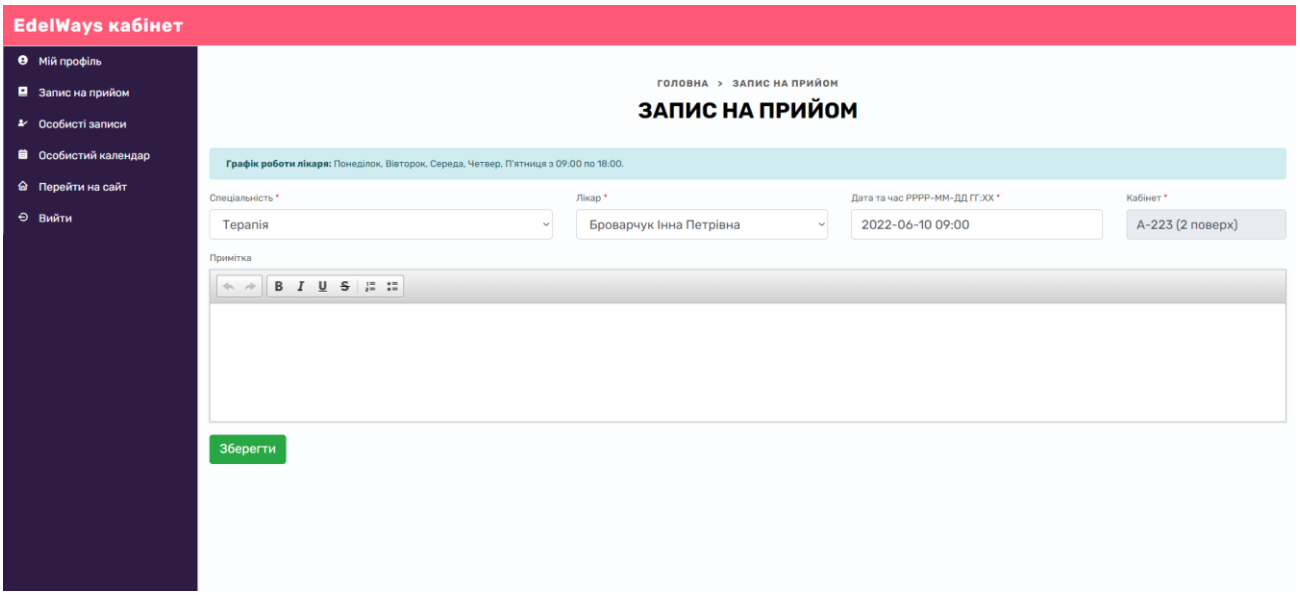

Рисунок 4.10 – Ознайомлення з графіком роботи медичного працівника

Після вдалого збереження обраних даних, користувач може перейти до

сторінки «Особисті записи» та побачити створений запис на прийом, який чекає підтвердження зі сторони адміністратора (рис. 4.11). У випадку, якщо обрані дата та час не підходять та плани змінились, то запис можна відредагувати.

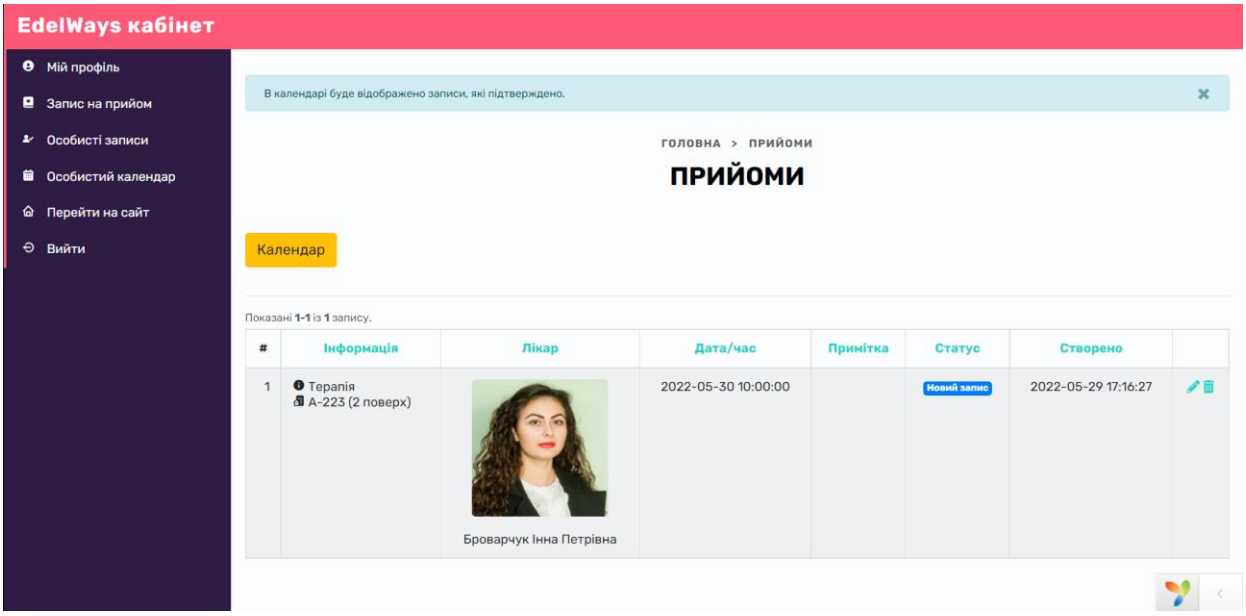

Рисунок 4.11 – Сторінка «Особисті записи»

Коли адміністратор розгляне даний запис, переконається у відповідності до загальних умов та підтвердить його, то статус зміниться з «Новий запис» на «Підтверджено». Внаслідок цього, буде автоматично згенеровано електронний талон, який можна зберегти на будь-який електронний пристрій (рис. 4.13). Для завантаження електронного талону необхідно натиснути на кнопку «Талон», яка з’явиться у відділі «Дата/час» (рис. 4.12)

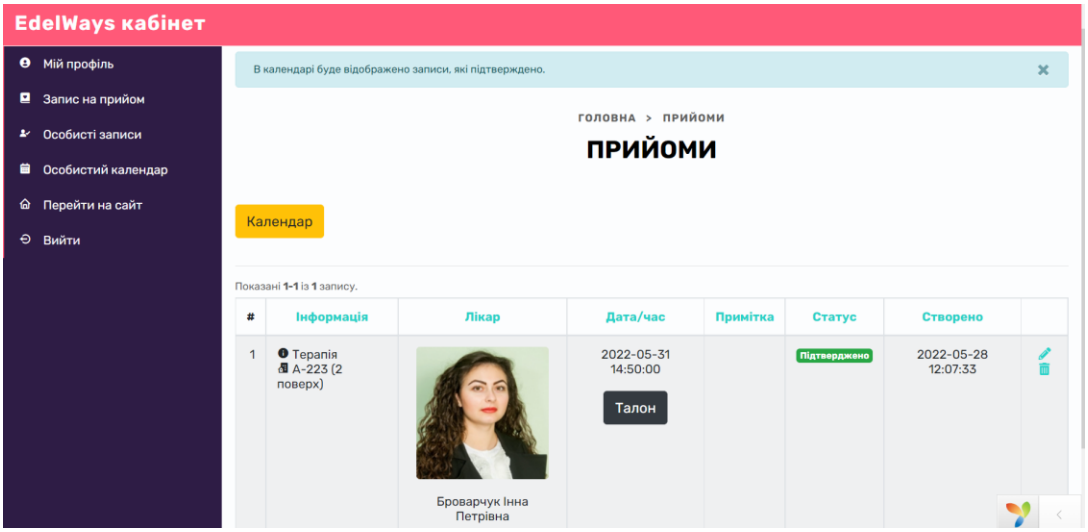

Рисунок 4.12 – Результат підтвердження запису

Рисунок 4.13 – Приклад згенерованого електронного талону

Крім цього, хочу зазначити, що реєстраційні дані, дані щодо запису як зі сторони пацієнта, так і зі сторони лікаря, будуть надіслані в вигляді макету електронної пошти (папка runtime/mail).

Переглядати історії запису та деталі можна двома шляхами: сторінка «Особисті записи» та «Особистий календар». Перейшовши на сторінку «Особистий календар» (рис. 4.14), користувач може ознайомитись з плануванням записів, попередніми відвідуваннями тощо. Для більш зручного та швидкого пошуку календар записів був розділений за допомогою фільтру на місяць, тиждень, день.

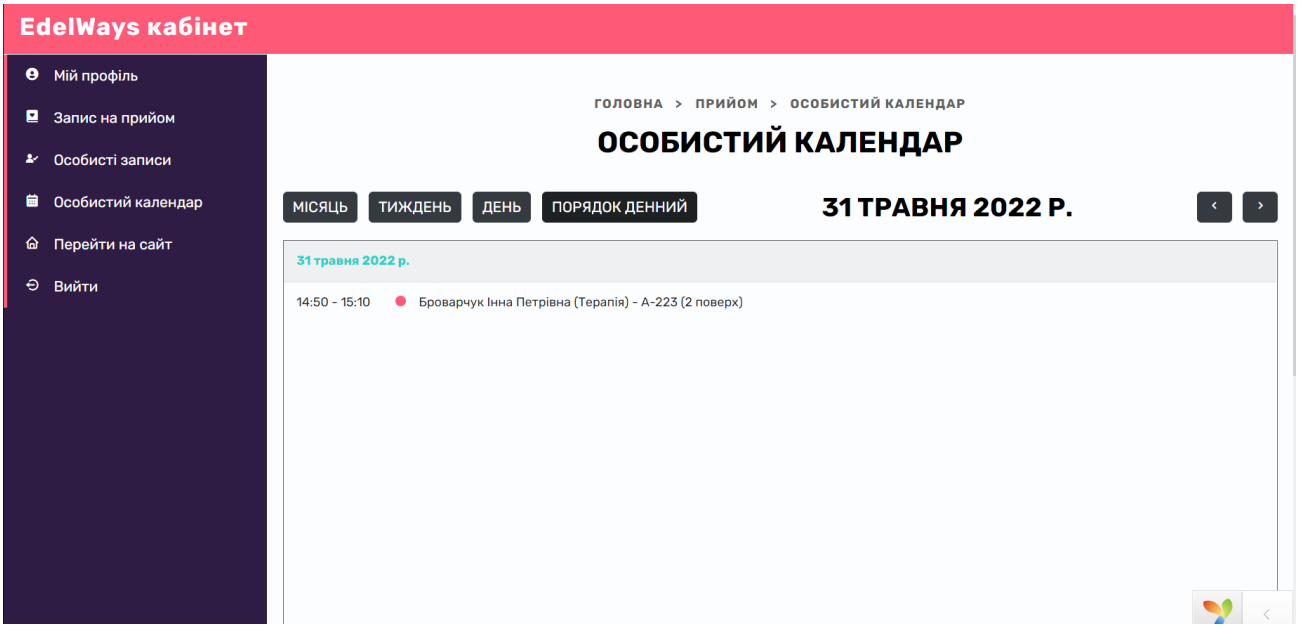

Рисунок 4.14 – Сторінка «Особистий календар»

Натиснувши пункт меню «Перейти на сайт», користувач знову ж таки потрапляє на головну сторінку, де при невеличкій прокрутці потрапляє в блок ститистики. Можна побачити, що там наявна кнопка «Проконсультуватись», перейшовши по ній, ми потрапляємо до форми зворотнього зв’язку (рис. 4.15).

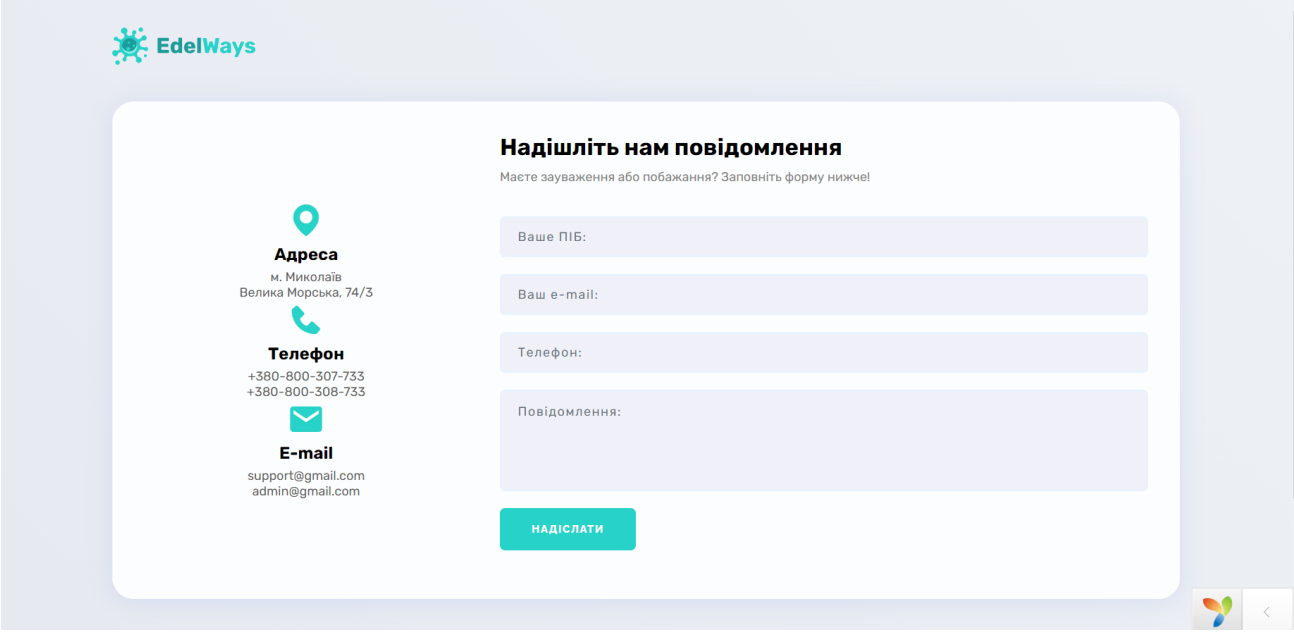

Рисунок 4.15 – Форма зворотнього зв’язку

Форма зворотнього зв’язку – корисна фукнція (виступає доповненням), яка дає змогу навіть не зареєстрованому користувачу, задати питання, які цікавлять, або ж надати зауваження чи побажання до адміністративної частини сайту.

Закінчивши розгляд користувацької частини, прийшов час до огляду функціональних можливостей та прав доступу Адміністратора вебзастосунку.

Ввійшовши до системи як адміністратор, ми бачимо спочатку те ж саме що й звичайний користувач – особистий кабінет. У адміністратора є така ж можливість запису на прийом, перегляду особистого календаря, але на цьому схожість прав доступу завершується.

Перейшовши до сторінки «Всі записи» (рис. 4.16) , адміністратор має можливість перегляду електронних записів користувачів зареєстрованих в системі, та приймати рішення щодо відхилення або підтвердження запису. Також можна за необхідності або по бажанню видаляти старі записи.

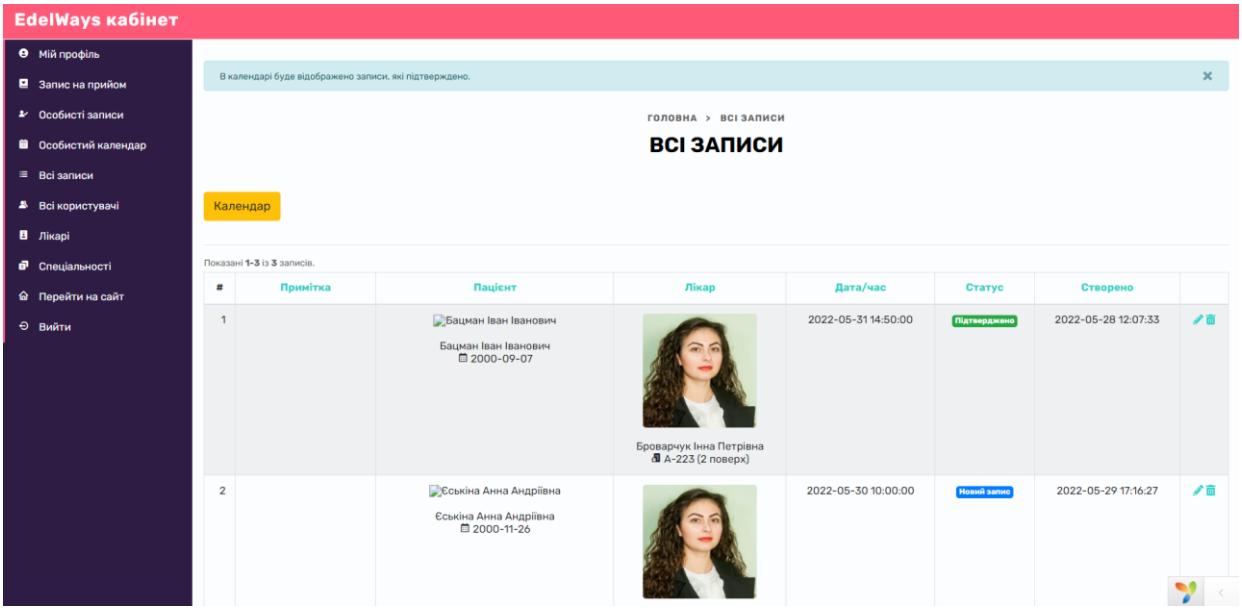

Рисунок 4.16 – Сторінка «Всі записи»

Далі відкривається можливість переходу до сторінок «Всі користувачі» (4.17), «Лікарі» (рис. 4.18) та «Спеціальності».

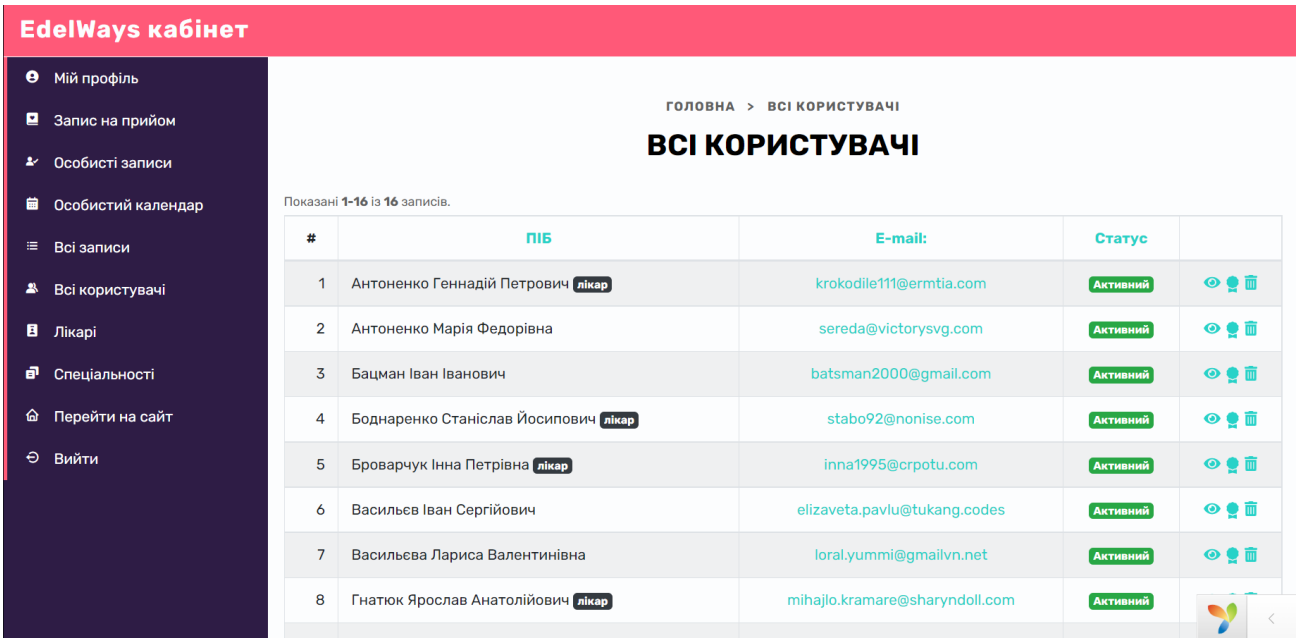

Рисунок 4.17 – Сторінка «Всі користувачі»

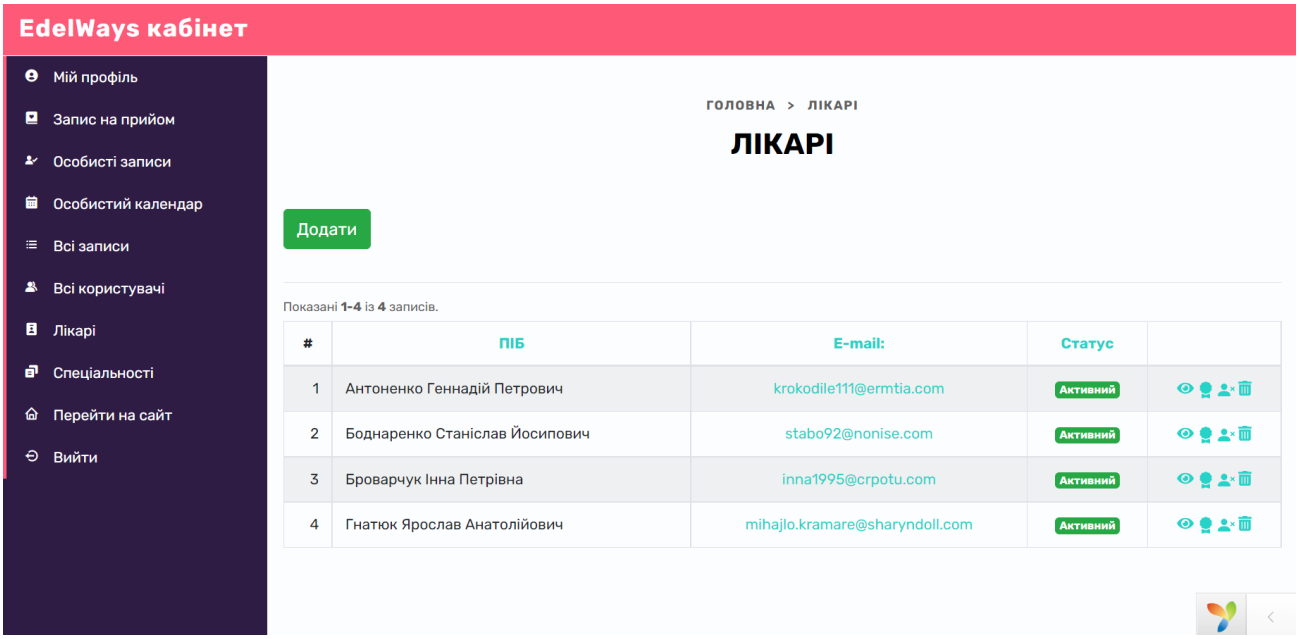

Рисунок 4.18 – Сторінка «Лікарі»

При розгляді перших двох, можна провести паралельну лінію та знайти схожість функціональних можливостей, а саме: перегляд особистих даних користувачів та лікарів, змінення статусу (блокування і розблокування) та остаточне видалення користувача/лікаря з системи.

Що ж до відмінностей, то перейшовши до сторінки «Лікарі», крім перегляду особистих даних, є можливість редагування інформації щодо графіка роботи і спеціалізації, та видалення з системи саме як медичного працівника. Крім цього, можна додавати медичних спеціалістів за допомгою кнопки «Додати», обравши потрібного користувача зі списку і заповнивши форму з усією необхідною інформацією (графік роботи).

Сторінка «Спеціальності» (рис. 4.19) має схожу характеристику. За необхідності адміністратор може відредагувати, видалити або додати спеціальність.

Уся перелічена інформація стосовно особистих даних, спеціальностей і т.д., зберігається в базі даних, структура якої була описана у 3 розділі.

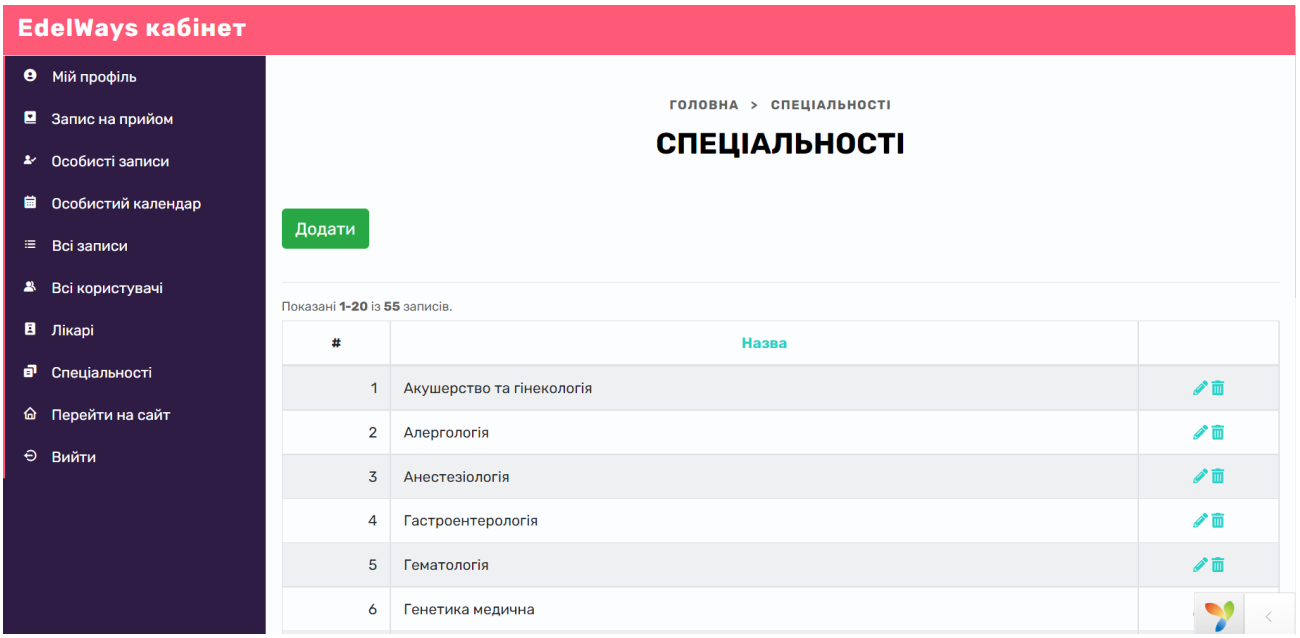

Рисунок 4.19 – Сторінка «Спеціальності»

4.3 Тестування програмного забезпечення

Тестування ПЗ – це процес оцінки та перевірки того, що програмний продукт або додаток відповідає всім поставленим вимогам. Переваги тестування включають запобігання помилкам, зниження витрат на розробку та підвищення продуктивності. Це один із важливих етапів розробки програмного забезпечення, який іноді недооцінюють.

Було обрано та проведено наступні види тестування: GUI та функціональне тестування.

Функціональне тестування – це своєрідне тестування методом «чорного ящика», яке проводиться для підтвердження того, що функціональність програми чи системи поводиться належним чином та відповідає вимогам [17, 19].

Огляд функціонального тесту включає наступні кроки:

1. Надання вхідних даних.

2. Виконання тестового кейсу.
3. Порівняння фактичного і очікуваного результату.

Рисунок 4.20 – Етапи функціонального тестування

Для проведення функціонального тестування було прийнято рішення створити тест-кейси у вигляді таблиці, що містить наступні дані:

- id – унікальний ідентифікатор;
- test case description – опис тестового випадку;
- precondition – передумови тест-кейсу;
- test steps – етапи тестування;
- expected result – очікуваний результат;
- pass/fail – провалено/пройдено.

При перевірці поведінки програмного продукту було розроблено 16 тест-кейсів. За результатом тестування встановлено, що все відповідає прописаним функціональним вимогам та працює безперебійно. Тобто 16 із 16 тестових випадків пройшли успішно. Функціональне тестування користувацької частини вебзастосунку наведено в додатку Б.

GUI тестування – це процес тестування програмного забезпечення, а саме графічного інтерфейсу користувача, щоб переконатися, що він відповідає його специфікаціям [18]. Метою тестування графічного інтерфейсу користувача (GUI) є забезпечення функціональності програмного додатка відповідно до специфікацій шляхом перевірки екранів і елементів керування, таких як меню,

кнопки, значки тощо. Процес тестування здійснювався за допомогою використання програмного продукту Selenium.

Selenium – це безкоштовна (з відкритим кодом) автоматизована платформа тестування, яка використовується для перевірки вебзастосунків у різних браузерах і платформах.

Результат проведення тестування користувацького вебінтерфейсу зображено на рисунку 4.21.

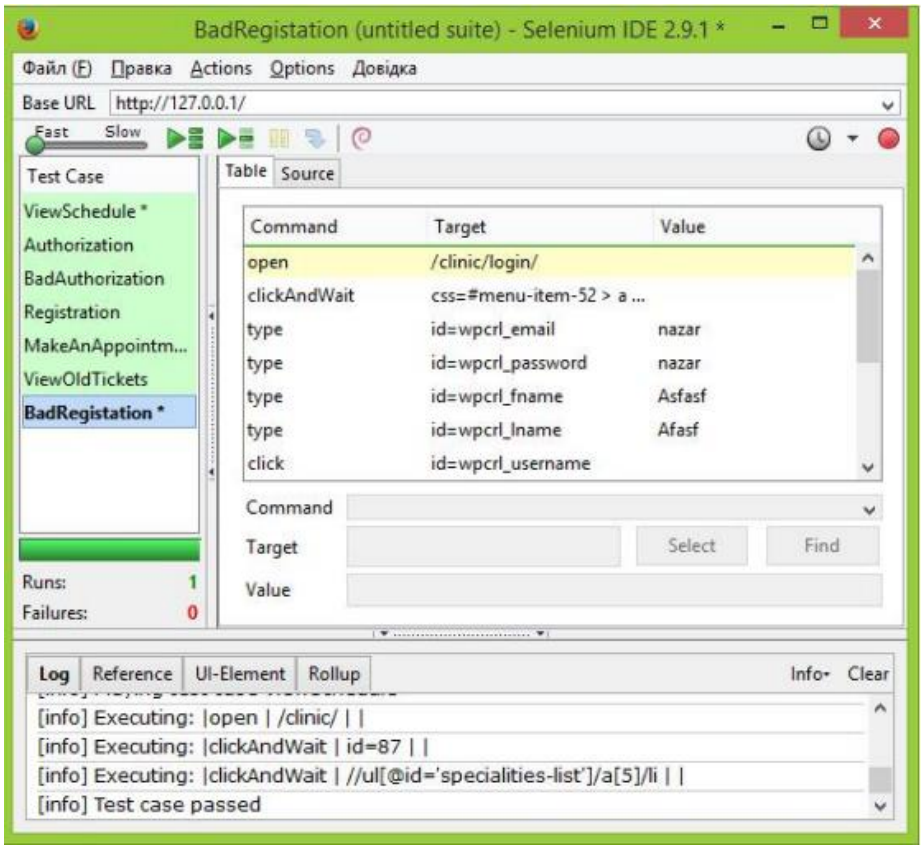

Рисунок 4.21 – GUI-тестування

В результаті тестування, можна встановити, що всі тест-кейси прошли успішну перевірку та не підлягають доробці, а отже реалізація користувацького вебінтерфейсу пройшла на вищому рівні. Код тестів зображених на рисунку 4.21, наведений у додатку В.

Висновки до розділу 4

Розглядаючи фінальну частину бакалаврської кваліфікаційної роботи, було розглянуто поетапну процедуру розгортання проєкту за допомогою фреймворку Yii2, а якщо бути точніше то структуру папок та їх призначення.

Було проведено загальний аналіз розробленого вебзастосунку, а саме: розглянуто різні рівні доступу (адміністративна і користувацька частина) та функціональні можливості у системі за допомогою поетапного зображення і опису.

Також реалізовано функціональне та графічне тестування користувацького інтерфейсу. Як висновок з’ясовано, що по завершенню усіх видів тестування, не було знайдено жодного багу, а отже вебзастосунок розроблено вдало та відповідно до усіх зазначених вимог.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Спеціальний розділ

ОХОРОНА ПРАЦІ

до кваліфікаційної роботи

на тему:

ВЕБЗАСТОСУНОК ДЛЯ ЕЛЕКТРОННОГО ЗАПИСУ ДО ЛІКАРНІ

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 402.21810310

Виконала студентка 4-го курсу, групи 402

_____*А.А.Єськіна*
(підпис, ініціали та прізвище)

«_21_» __червня__ 2022 р.

Консультант к.т.н., доцент

_____*А.О.Алексєєва*
(підпис, ініціали та прізвище)

«_21_» __червня__ 2022 р.

Миколаїв – 2022

ВСТУП

Сучасний розвиток технічного та технологічного стану виробництва передбачає постійну автоматизацію та оптимізацію виробничих процесів. Сьогодні, напевно, важко уявити компанію, господарська діяльність в якій здійснювалась би без використання комп’ютерної техніки. Через масовий характер робіт, що виконуються працівниками за допомогою комп’ютера, законодавством України чітко врегульовано норми та вимоги до використання комп’ютерної техніки на підприємстві, безпосередньо й охорона праці на підприємстві при роботі за комп’ютером.

Охорона здоров'я - один з пріоритетних напрямів державної діяльності та національної політики. Держава формує політику охорони здоров'я та забезпечує її реалізацію, використовуючи світовий досвід роботи з поліпшення умов і безпеки праці.

Гарантування безпечних умов праці, ліквідація професійних захворювань і виробничого травматизму, усунення шкідливих факторів є однією з головних принципів державної політики України в сфері охорони праці.

Метою роботи є аналіз умов праці у приміщенні комп'ютерної лабораторії вищого навчального закладу.

Відповідно до мети виділені наступні **завдання:**

1. Встановити загальні умови до приміщень з використанням комп’ютерної техніки.
2. Описати вимоги щодо організації та обладнання робочих місць.
3. Виконати опис комп'ютерної лабораторії, робочого місця адміністратора та виробничого обладнання.
4. Оцінити умови праці у лабораторії.

5. Сформулювати рекомендації щодо поліпшення умов праці на робочому місці адміністратора.

5 ОХОРОНА ПРАЦІ

5.1 Вимоги до приміщень з використанням комп'ютерної техніки

Приміщення, в яких планується установка та подальша робота з комп’ютером, повинні відповідати проєктній документації будинку, погодженій з уповноваженими державними органами. Крім того, роботодавець повинен враховувати санітарні нормативи освітлення, вимоги до параметрів мікроклімату (температура, відносна вологість), ступеня і сили вібрації, звукового шуму і вогнестійкості приміщення, а також характеристики електромагнітного, ультрафіолетового та інфрачервоного полів [20]. Конкретні показники зазначених санітарних норм див. в Державних санітарних правилах і нормах роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПІН 3.3.2.007-98, затверджених Постановою Головного державного санітарного лікаря України №7 від 10 грудня 1998 року. Правила поширюються на умови й організацію праці при роботі з візуальними дисплейними терміналами (ВДТ) усіх типів вітчизняного та зарубіжного виробництва на основі електронно-променевих трубок (ЕПТ), що використовуються в електронно-обчислювальних машинах (ЕОМ) колективного використання та персональних ЕОМ (ПЕОМ). Так, наприклад, роботодавцю заборонено установлювати комп’ютери в приміщеннях, розташованих у підвалах будинків. Для уникнення можливих аварій та замикань, поряд з приміщеннями, де вестиметься робота з комп’ютером (над чи під ними), також не дозволяється проведення робіт, що потребують здійснення надмірно вологих технологічних процесів. Відповідне приміщення повинно бути укомплектоване системами центрального або індивідуального опалення, кондиціонування чи вентиляції повітря. Але при установці зазначених систем, необхідно переконатись, що батареї опалення, водопровідні труби,

вентиляційні кабелі тощо, надійно сховані під захисними щитками, які перешкоджатимуть можливому потраплянню робітника під напругу [21].

Екран монітора не єдине джерело шкідливих електромагнітних випромінювань. Розробники моніторів досить давно і успішно займаються їх подоланням. Менше уваги приділяється шкідливим побічним випромінюванням, що виникають з боку бічних і задньої стінок обладнання.

В сучасних комп'ютерних системах ці зони найбільш небезпечні. Монітор комп'ютера слід розташовувати так, щоб задньою стінкою він був звернений не до людей, а до стіни приміщення. У комп'ютерних класах, що мають кілька комп'ютерів, робочі місця повинні розташовуватися по периферії приміщення, залишаючи вільним центр. При цьому додатково необхідно перевірити кожне з робочих місць на відсутність прямого відображення зовнішніх джерел освітлення. Як правило, домогтися цього для всіх робочих місць одночасно досить важко. Можливе рішення полягає у використанні штор на вікнах і продуманому розміщенні штучних джерел загального і місцевого освітлення.

Вигляд комп'ютерних класів відповідно до загальноприйнятих вимог представлено на рис. 5.1.

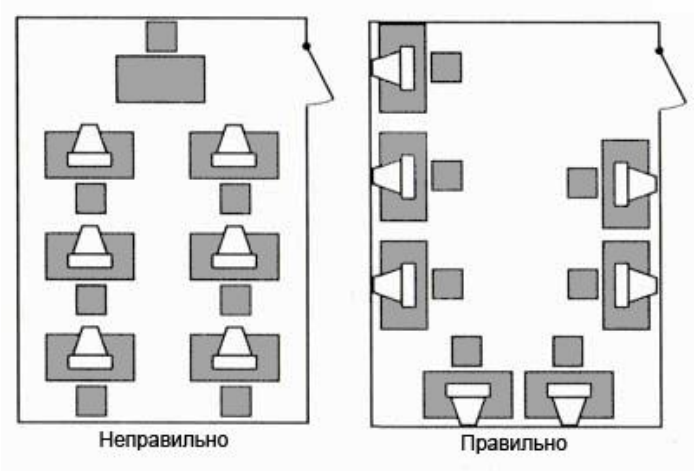

Рисунок 5.1 – Вигляд комп'ютерних класів

Сильними джерелами електромагнітних випромінювань є пристрої безперебійного живлення. Розташовувати їх слід якомога далі від місць користувачів.

У кожній кімнаті, де обладнуватимуться робочі місця співробітників, що працюватимуть на комп’ютері, повинні бути наявні елементи природного та штучного освітлення. При цьому, на вікнах слід встановити легко регульовані жалюзі чи штори, які дозволять працівникам коригувати рівень освітлення в приміщенні. Бажано розмістити комп’ютери в кімнаті таким чином, щоб світло потрапляло на екрани моніторів з півдня чи північного сходу. З метою досягнення максимального рівня безпечності і охорони праці при роботі з комп’ютером, виробничі приміщення необхідно обладнати аптечками першої медичної допомоги, системами автоматичної пожежної сигналізації і вогнегасниками. В приміщенні, в якому разом працюють 5 або більше комп’ютерів, на видимомумісці установлюється службовий вимикач, який у разі потреби дозволить повністю відключити електричне живлення кімнати.

Рисунок 5.2 – Організація та обладнання робочого місця

5.2 Вимоги щодо організації та обладнання робочих місць

Площа, відведена на одне робоче місце має становити не менше 6 кв.м., а об’єм – не менше 20 куб.м.. Конструкція робочого місця повинна забезпечувати підтримання оптимальної робочої пози, тобто такої, яка дозволяє працівникові виконувати роботу з мінімальним напруженням тіла, і яка дозволяє уникнути перевтоми в ході і після закінчення робочого процесу. Раціональна робоча поза має важливе значення для збереження здоров’я працівника, оскільки тривале перебування його в незручній і напруженій позі може призвести до таких захворювань, як сколіоз (викривлення хребта), варикозне розширення вен, плоскостопість тощо. Установлено, що робота в зігнутому положенні збільшує затрати енергії на 20%, а при значному нахиленні – на 45% порівняно з прямим положенням корпуса [22].

Загальні вимоги до організації робочого місця представлено на рис. 5.3.

Рисунок 5.3 – Загальні вимоги до організації робочого місця

За потреби особливої концентрації уваги під час виконання робіт суміжні робочі місця операторів необхідно відділяти одне від одного перегородками висотою 1,5 – 2 м.

Робочі місця слід розташовувати відносно джерела природного світла (вікон) таким чином, щоб світло падало збоку, переважно зліва. Також робоче місце має відповідати сучасним вимогам ергономіки [23]:

- стіл повинен мати висоту поверхні 680 – 720 мм, ширину 600 – 1600 мм і глибину 800 – 1000 мм. Такі параметри забезпечують можливість виконання операцій в зоні досяжності працівника;
- робочий стілець має бути підйомно – поворотним, з можливістю регулювання висоти, бажано зі стаціонарними або змінними підлікотниками і напівм’якою нековзкою поверхнею сидіння, що легко чиститься і не електризується;
- екран комп’ютера має розташовуватися на оптимальній відстані від користувача, що становить 600 – 700 мм, але не менше 600 мм з урахуванням літерно – цифрових знаків і символів;
- відстань між бічними поверхнями персональних комп’ютерів повинна бути не менше 1,2 метри;
- відстань від тильної поверхні одного персонального комп’ютера до екрана іншого – 2,5 метри;
- персональний комп’ютер та його комплектуючі (монітор та інші периферійні пристрої) не повинні потрапляти під прямі промені сонячного світла та під дію інших джерел тепла (батареї опалення та інші прилади для обігріву приміщень).

5.3 Опис комп'ютерної лабораторії, робочого місця адміністратора та виробничого обладнання

Приміщення комп'ютерної лабораторії закладу ресторанного типу розташовано на другому поверсі пятиповерхової будівлі. Розміри приміщення

складають $a \times b \times H = 6,0 \times 4,0 \times 3,5$м. У приміщенні влаштовано два дерев'яних вікна (з подвійними склопакетами) розмірами $c \times d = 1,8 \times 2,2$ м.

Конструкція стін приміщення герметична, при цьому стіни і двері володіють вогнестійкістю не менше 45 хвилин, а міжповерхові перекриття крім цього мають гідроізоляцію. Ширина дверей 515(910 ббуло) мм, высота – 2000 мм. Двері відкриваються всередину на 90˚ (180), дверна коробка має невеликий поріг. В конструкцію дверей вмонтована ущільнювальна прокладка.

Приміщення має звичайний інтер’єр який притаманний більшій кількості лабораторій ЧНУ ім. Петра Могили. Стеля виконана у вигляді підвісної конструкції із синтетичного матеріалу світло-сірого кольору. Стіни мають гладку поверхню світло-зеленого кольору. Підлога має покриття із лінолеуму. Вікна обладнані світлозахисними пристроями у вигляді вертикальних регульованих жалюзі.

Для підтримки температури в діапазоні від 18 до 24 градусів і відносній вологості від 30 до 55% встановлена система кондиціонування, що складається з одного кондиціонера. Потужність кондиціонера на даний момент перевищує сумарне тепловиділення всього устаткування і систем, розташованих в лабораторії.

Система вентиляції не забезпечує в приміщенні надлишковий тиск – об'єм повітря, що поступає на 20% менше, ніж обсяг відведеного. Потужність системи не змінює повітря кожну годину. При цьому на повітропроводах припливної та притяжної вентиляції передбачаються захисні клапани, керовані автоматикою установки газового пожежогасіння. Системи кондиціювання та вентиляції відключаються за сигналом пожежної сигналізації.

У приміщенні розташовано одне робоче місце, обладнане сучасним персональним комп’ютером з необхідними периферійними пристроями. Центр приміщення займають шафи з серверним обладнання. Для зберігання необхідного інструменту передбачена шафа. Перелік обладнання наведено в табл. 5.1.

Таблиця 5.1 – Найменування предметів у компьютерній лабораторії

№	Назва	Кількість
1	Шафа для обладнання	1
2	Стіл	23
3	Стілець	38
4	Персональний комп’ютер	12
5	Кондиціонер	1

Напруга джерела живлення електро споживної техніки – 380 В. Електромережа виконана у вигляді трипровідної з дотриманням усіх вимог нормативних документів. За небезпекою ураження електричним струмом приміщення відноситься до приміщень без підвищеної небезпеки ураження електричним струмом.

Мікрокліматичні умови у літній період (частково у перехідний) забезпечується спліт-системою кондиціонування, потужності якої вистачає для забезпечення комфортних умов праці. У зимовий період опалення здійснюється центральною системою, яка забезпечує необхідний тепловий режим.

Завдяки двом великим вікнам немає дискомфорту з природнім освітленням. Електроживлення освітлення комп'ютерної лабораторії та електроживлення телекомунікаційного обладнання, встановленого в комп'ютерній лабораторії,

подається від різних розподільних електричних щитів. Світильники розміщуються на стелі.

Для управління освітленням користуватися одним або декількома вимикачами і розташовувати їх поряд з дверима на висоті 1,5 м від рівня підлоги.

Пожежна безпека в обраному виробничому приміщенні забезпечується дотриманням вимог НПАОП 0.00-1.28-10 [24].

У табл. 5.2. приведено фактори умов праці на робочому місці лабораторії.

Таблиця 5.2 – Фактори умов праці на робочому місці

№ з/п	Фактор умов праці на робочому місці	Значення показника	Тривалість дії фактора, хв.
1	Температура повітря на робочому місці (РМ) у виробничому приміщенні, ℃: - теплий період; - холодний період	22 -	420 -
2	Відносна вологість повітря на РМ, %	70	420
3	Швидкість руху повітря на РМ, м/с	0,25	480
4	Освітленість на РМ, лк	150	240
5	Мінімальний розмір об’єкта розпізнавання, мм	0,5	240
6	Виробничий шум, дБА	62	480
7	Інтенсивність теплового випромінювання, Вт/м²	180	420
8	Токсична речовина, озон, кратність перевищення ГДК	5	480
9	Виробничий пил (паперовий та ін.), кратність перевищення ГДК	1	480

Продовження таблиці 5.2

10	Робоче місце (РМ), поза та переміщення у просторі	Робоче місце стаціонарне, маса переміщення вантажу до 5 кг	480
11	Кількість важливих об’єктів спостереження	6	420
12	Тривалість зосередженого спостереження, % часу зміни	75	360

5.4 Оцінка умов праці в комп'ютерній лабораторії

Для оцінки умов праці в комп'ютерній лабораторії слід скористатися даними табл. 5.2 та здійснити оцінку питомої ваги кожного із представлених там факторів виробничого середовища та трудового процесу [25].

У табл. 5.3 представлені параметри, що необхідні для оцінки умов праці:

$x_{н_i}$ – нормативне значення i – того фактору умов праці (прийняті значення відповідають оптимальному (допустимому) класу умов праці згідно з Гігієнічною класифікацією);

$x_{аб_i}$ – дійсне значення i – того фактору умов праці;

x_{x_i} – оцінка i – того фактору умов праці, балів;

t_i – тривалість дії i – того фактору умов праці, хв.;

$t_{пит_i}$ – відносна тривалість дії i – того фактору умов праці (за прийнятої тривалості робочої зміни t_p = 480 хв.), хв., тобто:

$$t_{пит_i} = \frac{t_i}{t_p} = \frac{t_i}{480};$$

$x_{ф_i}$ – фактична оцінка питомої ваги i – того фактору умов праці, балів, а саме:

$$x_{\phi_i} = x_{x_i} t_{num_i} = x_{x_i} \frac{t_i}{480}.$$

За даними табл. 5.3 визначаємо елемент умов праці, який одержав у балах найбільшу оцінку $x_{\max}$. Принципово таких елементів може бути декілька [25]

Таблиця 5.3 – Параметри, що необхідні для розрахунку оцінки умов праці

№ з/п	Фактор умов праці на робочому місці	Нормоване значення фактора $x_{\text{н}_i}$	Оцінка фактора		Тривалість дії фактора		Фактична оцінка питомої ваги фактора $x_{\text{ф}i}$
			Абсолютна $x_{\text{абі}}$	У балах $x_{\text{хі}}$	Хвилин $t_{\text{і}}$	У долях робочої зміни $t_{\text{пит і}}$	
1	Температура повітря на робочому місці℃ - теплий період	23…25	22	1	420	0,875	0,875
2	Відносна вологість повітря на РМ, %	40..60	70	3	420	0,875	2,625
3	Швидкість руху повітря на РМ, м/с	<0,2	0,25	2	480	1	2
4	Освітленість на РМ, лк	200	150	4	240	0,5	2
5	Мінімальний розмір об’єкта розпізнавання, мм	>1	0,5	2	240	0,5	1
6	Виробничий шум, дБА	50	62	3	480	1	3
7	Інтенсивність теплового випромінювання, Вт/м²	≤140	180	2	420	0,875	1,75
8	Токсична речовина, озон, поліхромовані біфеніли, протипожежні бромовані компоненти, кратність перевищення ГДК	≤1	5	5	480	1	5
9	Виробничий пил (паперовий), кратність перевищення ГДК	≤1	1	2	480	1	2
10	Робоче місце (РМ), поза та переміщення у просторі	РМ стаціонарне,маса переміщення до 5 кг	РМ стаціонарне,маса переміщення до 5 кг	1	480	1	1
11	Кількість важливих об’єктів спостереження	<5	6	2	420	0,875	1,75
12	Тривалість зосередженого спостереження, % часу зміни	<25	75	3	360	0,75	2,25

Таким елементом являється елемент x_8, який пов'язаний з тривалістю роботи за добу, тобто $x_{max} = x_8 = 5$. Даний елемент вважається визначаючим.

Далі розраховується:

1. Середній бал усіх елементів крім визначаючого $\overline{x}$, балів:

$$\overline{x} = \frac{\sum_{i=1}^{n-1} x_{\phi_i}}{n-1},$$

де n – фактична кількість врахованих елементів умов праці (у даному випадку $n = 12$).

Тоді:

$$\overline{x} = \frac{\sum_{i=1}^{n-1} x_{\phi_i}}{n-1} = \frac{25,25}{12-1} = 2,3.$$

2. Оцінка умов праці на робочому місці в комп'ютерній лабораторії I_n, балів.

$$I_n = 10 \cdot (x_{max} + \overline{x}\frac{6 - x_{max}}{6}) = 10 \cdot (5 + 2,1\frac{6-5}{6}) = 53,5.$$

Умови праці на визначеному робочому місці відносяться до IV категорії, коли спостерігається робота у несприятливих умовах праці [26].

Аналіз причин низької оцінки важкості праці на зазначеному робочому місці дозволяє зробити висновок про суттєвий негативний вплив токсичних речовин, який значно перевищує гігієнічні нормативи.

В наступному підрозділі представлено матеріали щодо визначення параметрів системи вентиляції, призначеної для забезпечення необхідної чистоти повітряного середовища в комп'ютерній лабораторії.

Рівень токсичної речовини значно перевищує норми. Необхідно розрахувати систему вентиляцію для вирішення цієї проблеми.

5.5 Оцінка умов праці в комп'ютерній лабораторії

Розрахунок припливно-витяжної вентиляції виконується в залежності від розв'язуваних завдань для конкретного приміщення. При цьому основними параметрами, які регулюються вентиляційними агрегатами, є вологість, температура. При монтажі витяжки забезпечується одноразовий повітрообмін, додавання припливної вентиляції обмін робить дворазовим. Схему припливно-витяжної системи зображено на рис. 5.4.

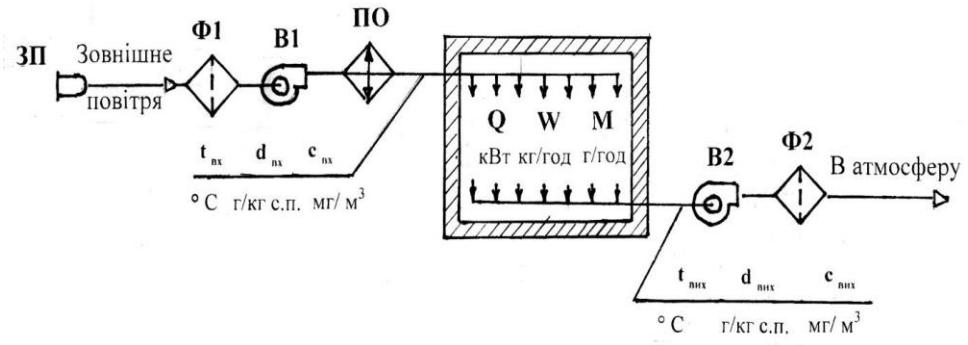

Рисунок 5.4 – Принципова схема припливно-витяжної системи вентиляції: *ЗП – забірний пристрій; Ф1, Ф2 – фільтри, В1, В2 – вентилятори; ПО – повітроохолодник; Q, W, M – відповідно теплоприпливи, вологонадлишки та надходження шкідливих речовин у виробниче приміщення.*

Згідно з вимірами, які проводилися на початку весни, встановлено, що концентрація шкідливих компонентів у повітрі на вході у виробниче приміщення дорівнює $c_{вх}$ = 0,001 мг/м^3 (озон, полібромовані біфеніли, протипожежні бромовані компоненти пластмас). Приймається, що допустимою концентрацію шкідливих компонентів у повітрі обраного приміщення є гранично допустима концентрація основного шкідливого компонента – озону, тобто $c_{вих}$ = 0,1 мг/м^3. Надходження (приведене) шкідливих компонентів у повітряне середовище виробничого приміщення за даними може бути оціненим величною M = 0,25 г/год [21].

Повітропродуктивність системи вентиляції, що необхідна для компенсації надходжень шкідливих компонентів повітряного середовища виробничого приміщення.

$V = 1000 \cdot M/(c_{вих} – c_{вх}) = 1000 \cdot 0,25 / (0,1 – 0,001) = 2500$ м^3/год.

Повітропродуктивність системи вентиляції, що необхідна для компенсації надходжень шкідливих компонентів повітряного середовища виробничого приміщення, оцінюється величиною 2500 м^3/год.

Виконані розрахунки показують, що наявна система вентиляції не виконує поставлене завдання. Для забезпечення необхідного повітрообміну слід встановити припливно-витяжну систему вентиляції, що дасть можливість знизити рівень токсичної величини до мінімального рівня.

Згідно з розрахунками оптимальним варіантом є вентилятор Вентс ОВ 4Д 350 осьовий з продуктивністю 2520 м^3/год.

Висновки до розділу 5

В цьому розділі розглянуто важливість забезпечення вимог охорони праці в приміщенні комп'ютерної лабораторії вищого навчального закладу.

Перевірено забезпечення вимог охорони праці , а саме санітарні нормативи освітлення, вимоги до параметрів мікроклімату (температура, відносна вологість), ступеня і сили вібрації, звукового шуму і вогнестійкості приміщення, а також характеристики електромагнітного, ультрафіолетового та інфрачервоного полів. Встановлено, що завдяки двом великим вікнам немає дискомфорту з природнім освітленням. Мікрокліматичні умови у літній період (частково у перехідний) забезпечується спліт-системою кондиціонування, потужності якої вистачає для забезпечення комфортних умов праці. У зимовий

період опалення здійснюється центральною системою, яка забезпечує необхідний тепловий режим.

За даними табл. 5.3 визначили елемент умов праці, який одержав у балах найбільшу оцінку – токсична речовина, озон, поліхромовані біфеніли, протипожежні бромовані компоненти, кратність перевищення ГДК, що дорівнює $x_{max} = x_8 = 5$. Внаслідок цього, було розраховано середній бал усіх елементів крім визначаючого $\overline{x} = 2,3.$

Наступні розрахунки стосувались оцінки умов праці на робочому місці, що дорівнює $I_n = 53,5$. Виявлено, що оцінка умов праці на робочому місці відноситься до IV категорії, коли спостерігається робота у несприятливих умовах праці. З метою їх покращення розраховано припливно-витяжну вентиляцію повітропродуктивності для компенсації надходжень шкідливих компонентів $V \geq 2500$ м^3/год. Як висновок, встановлено, що система вентиляції не виконує поставлене завдання та підібрано вентилятор з необхідною витратно-напірною характеристикою (Вентс ОВ 4Д 350 осьовий з продуктивністю 2520 м^3/год).

ВИСНОВКИ

В результаті виконання бакалаврської кваліфікаційної роботи було реалізовано вебзастосунок для електронного запису до лікарні, який би підвищив якість процесу запису до лікаря за рахунок розподілу потоку відвідувачів і завчасного інформування клієнтів про порядок і правила обслуговування.

Зазначену мету досягнуто завдяки виконання наступних завдань:

- проаналізовано актуальність поставленої задачі розробки вебзастосунку;
- проведено аналіз існуючих аналогів та внаслідок цього визначено їх недоліки;
- розглянуто загальну теорію та поняття;
- обрано та обгрунтовано засоби програмної реалізації;
- реалізовано вебзастосунок для запису до електронної черги з функціоналом адміністративної/клієнтської частини;
- проведено тестування адаптивного інтерфейсу та функціональності вебзастосунку.

Для реалізації поставленого завдання було використано мову програмування PHP, а якщо бути точніше то PHP-фреймворк Yii2, і JS, інтегроване середовище розробки PHPStorm та локальний вебсервер OpenServer, де за допомогою PHPMyAdmin, було створено та підключено базу даних MySQL до проєкту.

Що ж до переваг розробленого вебінтерфейсу, можна виділити наступне:

- можливість розширення функціоналу та оновлення вебзастосунку;
- простий та зрозумілий інтерфейс користувацької/адміністративної частини;
- адаптивність;

- відсутня надмірна кількість інформаційного потоку, що зосереджує користувача на основній задачі програмного продукту.

Для того щоб усунути несправність в працездатності вебзастосунку, було вирішено провести наступні види тестування: GUI та функціональне. За результатами проведення встановлено, що система працює безперебійно, відповідаючи на всі запити користувача та повертаючи відповідь.

Кваліфікаційна робота складається з вступу, чотирьох розділів, висновку, переліку джерел посилання, 3 додатків та спеціальної частини з охорони праці. Основна частина роботи викладена на 68 сторінках тексту (без додатків), містить 48 рисунків та 25 джерел посилання.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Щодо рекомендацій з навчально-методичного забезпечення : лист Міністерства освіти і науки України від 09.07.2018 № 1/9-434. URL: http://ru.osvita.ua/legislation/Vishya_osvita/61422/ (дата звернення: 21.04.2022).

2. Юр'єв В.К., Куценко Г.І. Громадське здоров'я та охорона здоров'я. СП, 2000. 240-283 с.

3. Static vs Dynamic Website: What is the Difference? URL: https://wpamelia.com/static-vs-dynamic-website/_ (дата звернення: 25.04.2022).

4. Android vs IOS – Difference and Comparison. URL: https://www.diffen.com/difference/Android_vs_iOS (дата звернення: 26.04.2022).

5. Adaptive design. URL: https://www.invisionapp.com/defined/adaptive-design (дата звернення: 28.04.2022).

6. Welch B., Welch B. Managing performance on SCO OpenServer 1996. URL: https://www.researchgate.net/publication/262318094_ Managing_performance_on_SCO_OpenServer (дата звернення: 01.05.2022).

7. Advantages and Disadvantages of JavaScript. URL: https://www.geeksforgeeks.org/advantages-and-disadvantages-of-javascript/ (дата звернення: 04.05.2022).

8. Zandstra M. PHP and Objects 2021. DOI:10.1007/978-1-4842-6791-2_2. URL: https://www.researchgate.net/publication/350595325_PHP_and_ Objects (дата звернення: 04.05.2022).

9. Christudas B. MySQL 2019. DOI:10.1007/978-1-4842-4501-9_27. URL: https://www.researchgate.net/publication/334012533_MySQL (дата звернення: 05.05.2022).

10. What is phpMyAdmin – Handy Backup. URL: https://www.handybackup.net/backup_terms/phpmyadmin-definition.shtml (дата звернення: 07.05.2022).

11. Сафронов М. Разработка веб-приложений в Yii 2/ М. Сафронов - М. : ДМК Пресс - ISBN 978-5-97060-252-2, 2015. 392 с.

12. What is a client-server model? A guide to client-server architecture. URL: https://www.serverwatch.com/guides/client-server-model/ (дата звернення: 16.05.2022).

13. Сивцев Н. С. Шаблон проєктирования MVC – Model Viev Control/Н. С. Сивцев, Р. Р. Мусабаев, К. Д. Бондаренко; ТУСУР, РТФ. –Томск, 2014. 3 с.

14. ER Diagram: Entity Relationship Diagram Model | DBMS Example. URL: https://www.guru99.com/er-diagram-tutorial-dbms.html (дата звернення: 18.05.2022).

15. Installing Yii2 with Composer. URL: https://subscription.packtpub.com/book/webdevelopment/9781785287411/1/ch01lvl1sec09/installing-yii2-with-composer (дата звернення: 02.06.2022).

16. RBAC, роли и пользователи в Yii2 URL: RBAC, роли и пользователи в Yii2 - Developer.uz Developer.uz (дата звернення: 03.06.2022).

17. What is Software Testing and How Does It Work? URL: https://www.ibm.com/topics/software-testing (дата звернення: 05.06.2022).

18. GUI Testing Tutorial: A Complete User Interface (UI) Testing Guide. URL: https://www.softwaretestinghelp.com/gui-testing/_ (дата звернення: 05.06.2022).

19. «Тестування програмного забезпечення»[Онлайновий]. URL: https://uk.wikipedia.org/wiki/Тестування_програмного_забезпечення (дата звернення: 06.06.2022).

20. Жидецький В.Ц. Охорона праці користувачів комп’ютерів. – Львів: Афіша, 2000. 176 с.

21. Жидецький В.Ц. Основи охорони праці. Підручник. – Львів: Афіша, 2002. 320 с.

22. Охорона праці, при роботі з персональним комп’ютером. URL: https://studwood.net/570840/informatika/ohorona_pratsi (дата звернення: 20.05.2022).

23. Виробниче середовище та його вплив на людину. URL: http://dspace.wunu.edu.ua/jspui/bitstream/316497/9184/1/опорний%20конспект%20лекцій.pdf (дата звернення: 20.05.2022).

24. Про затвердження Правил охорони праці під час експлуатації електронно-обчислювальних машин (НПАОП 0.00-1.31-99). URL: https://zakon.rada.gov.ua/laws/show/z047214#:~:text=4%20клас%20(небезпечні%20умови%20праці,1.2 (дата звернення: 25.05.2022).

25. Щербак Ю.Г., Макарова О.В. Методичні рекомендації до практичних занять із дисципліни «Основи охорони праці». – Миколаїв: Вид-во ЧДУ імені Петра Могили – (Методична серія; Вип.215), 2014. 68 с.

ДОДАТОК А

Код програмного забезпечення

index.php

```
<?php

/** @var yii\web\View $this */

use yii\bootstrap4\Html;
use yii\helpers\Url;

$title = 'Головна';
$keywords = 'EdelWays Clinic, здоров\'я, запишіться до лікаря';
$description = 'Подбайте про здоров\'я своєї родини. Усе в одному місці.
Проконсультуйтеся з нашими медичними працівниками щодо ваших проблем.';

$this->title = $title;

$this->registerMetaTag([
    'name' => 'description',
    'content' => $description,
]);
$this->registerMetaTag([
    'name' => 'keywords',
    'content' => $keywords,
]);

Yii::$app->seo->putOpenGraphTags(
    [
        'og:site_name' => Yii::$app->name,
        'og:title' => $title,
        'og:description' => $description,
        'og:image' => Url::to('@web/img/banner.jpg', true),
        'og:url' => Url::canonical(),
    ]
);

Yii::$app->seo->putGooglePlusMetaTags(
    [
        'name' => $title,
        'description' => $description,
        'image'  => Url::to('@web/img/banner.jpg', true),
```

```
    ]
);
?>
<section class="home container" id="home">
    <div class="home__content">
        <div class="home__info">
            <h2 class="home__title">Подбайте про <span>здоров'я</span> своєї
родини.</h2>
            <p class="home__description">Усе в одному місці. Проконсультуйтеся з
нашими медичними працівниками щодо ваших проблем.</p>
            <a class="home__btn btn btn__main btn__main--flower" href="<?=
Yii::$app->urlManager->createUrl(['/site/sign-in']) ?>">Приєднатись</a>
        </div>
        <?= Html::img('@web/img/get-started/image.svg', ['class' => 'home__img img-
fluid','alt' => 'Лікар з дитиною']) ?>
    </div>
    <a class="video-btn" href="https://www.youtube.com/watch?v=q0F843jV8dk"
target="_blank">
        <?= Html::img('@web/img/get-started/video.svg', ['class' => 'video-
btn__icon', 'alt' => 'Запустити відео', 'width' => 44, 'height' => 44]) ?>
        <span class="video-btn__info">
            <span class="video-btn__slogan">Будьте в безпеці з <?= Yii::$app->name
?></span>
            <span class="video-btn__title">Переглянути відео</span>
        </span>
    </a>
</section>

<section class="instruction container" id="instruction">
    <div class="instruction__item">
        <?= Html::img('@web/img/instruction/1.png', ['class' => 'instruction__img',
'width' => 210, 'height' => 250, 'alt' => 'Зареєструйтесь']) ?>
        <p class="instruction__text">Зареєструйтесь</p>
    </div>

    <div class="instruction__item">
        <?= Html::img('@web/img/instruction/2.png', ['class' => 'instruction__img',
'width' => 210, 'height' => 250,  'alt' => 'Запишіться до лікаря на зручний час'])
?>
        <p class="instruction__text">Запишіться до лікаря на зручний час</p>
    </div>

    <div class="instruction__item">
```

```
        <?= Html::img('@web/img/instruction/3.png', ['class' => 'instruction__img',
'width' => 210, 'height' => 250,  'alt' => 'Отримайте електронний талон у
особистому кабінеті']) ?>
        <p class="instruction__text">Отримайте електронний талон</p>
    </div>

    <div class="instruction__item">
        <?= Html::img('@web/img/instruction/4.png', ['class' => 'instruction__img',
'width' => 210, 'height' => 250,  'alt' => 'Приходьте до лікаря на обраний час'])
?>
        <p class="instruction__text">Приходьте до лікаря на обраний час</p>
    </div>
</section>

<section class="statistics" id="statistics">
    <div class="statistics__wrap">
        <div class="statistics__item">
            <span class="statistics__value">5</span>
            <p class="statistics__title">Амбулаторій</p>
        </div>

        <div class="statistics__item">
            <span class="statistics__value">15+</span>
            <p class="statistics__title">Експертів</p>
        </div>

        <div class="statistics__item">
            <span class="statistics__value">10,000+</span>
            <p class="statistics__title">Записів до лікаря</p>
        </div>
    </div>

    <div class="statistics__decor statistics__decor--first"></div>
    <div class="statistics__decor statistics__decor--second"></div>
    <div class="statistics__decor statistics__decor--third"></div>
</section>

<section class="contact container" id="contact">
    <div class="contact__content">
        <h2 class="contact__title">Зв'яжіться з <span>експертом</span>.</h2>
        <p class="contact__text">Записуйтесь на прийом та надсилайте запитання, що
стосуються проблем зі здоров’ям або із записом до нашої онлайн-форми.</p>
        <a class="btn btn__main btn__main--flower" href="<?= Yii::$app->urlManager-
>createUrl(['/site/contact']) ?>">Проконсультуватись</a>
```

```
    </div>
    <div class="contact__video">
        <iframe width="560" height="315"
src="https://www.youtube.com/embed/AXYeWgS3UGU" title="YouTube video player"
frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media;
gyroscope; picture-in-picture" allowfullscreen></iframe></iframe>
    </div>
</section>
```

dashboard.php

```
<?php

/* @var $this \yii\web\View */
/* @var $content string */

use app\assets\AppAsset;
use app\widgets\Alert;
use yii\bootstrap4\Html;

AppAsset::register($this);
?>
<?php $this->beginPage() ?>
<!DOCTYPE html>
<html lang="<?= Yii::$app->language ?>" class="h-100">
<head>
    <meta charset="<?= Yii::$app->charset ?>">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
fit=no">
    <meta name="msapplication-TileColor" content="#f3ecb8">
    <meta name="theme-color" content="#f3ecb8">
    <?php $this->registerCsrfMetaTags() ?>
    <?= $this->registerCssFile("@web/css/dashboard.css", [
        'depends' => [\yii\bootstrap4\BootstrapAsset::class],
    ], 'dashboard');
    ?>

    <title><?= Yii::$app->name .' - '.Html::encode($this->title) ?></title>

    <?php $this->head() ?>
</head>
<body itemscope itemtype="http://schema.org/WebPage" class="d-flex flex-column h-
100">
<?php $this->beginBody() ?>
```

```
<header class="dashboard__header">
    <?php
        $words = explode(' ', Yii::$app->name);
        echo $words[0] . ' кабінет';
    ?>
</header>
<div id="dashboard-hamburger" class="dashboard__hamburger" title="Меню">
    <span class="ham top"></span>
    <span class="ham middle"></span>
    <span class="ham bottom"></span>
</div>
<nav id="dashboard-menu" class="sidebar-container">
    <ul class="sidebar-navigation">
        <li>
            <a href="<?=Yii::$app->urlManager->createUrl(['/user/profile'])?>">
                <i class="bx bxs-user-circle"></i>
                <span>Мій профіль</span>
            </a>
        </li>
        <li>
            <a href="<?=Yii::$app->urlManager->createUrl(['/events/booking'])?>">
                <i class="bx bxs-book-heart"></i>
                <span>Запис на прийом</span>
            </a>
        </li>
        <li>
            <a href="<?=Yii::$app->urlManager->createUrl(['/events/private'])?>">
                <i class="bx bxs-user-check"></i>
                <span>Особисті записи</span>
            </a>
        </li>
        <li>
            <a href="<?=Yii::$app->urlManager->createUrl(['/events/my-
calendar'])?>">
                <i class="bx bxs-calendar"></i>
                <span>Особистий календар</span>
            </a>
        </li>
        <?php if(Yii::$app->user->can('doctor')): ?>
            <li>
                <a href="<?=Yii::$app->urlManager-
>createUrl(['/events/patients'])?>">
                    <i class="bx bxs-user-detail"></i>
                    <span>Записи пацієнтів</span>
```

```
                </a>
            </li>
            <li>
                <a href="<?=Yii::$app->urlManager->createUrl(['/events/patients-
calendar'])?>">
                    <i class="bx bxs-calendar-star"></i>
                    <span>Робочий календар</span>
                </a>
            </li>
        <?php endif; ?>
        <?php if(Yii::$app->user->can('admin')): ?>
            <li>
                <a href="<?=Yii::$app->urlManager->createUrl(['/events/index'])?>">
                    <i class="bx bx-list-ul"></i>
                    <span>Всі записи</span>
                </a>
            </li>
            <li>
                <a href="<?=Yii::$app->urlManager->createUrl(['/user/all'])?>">
                    <i class="bx bxs-group"></i>
                    <span>Всі користувачі</span>
                </a>
            </li>
            <li>
                <a href="<?=Yii::$app->urlManager->createUrl(['/user/doctors'])?>">
                    <i class="bx bxs-user-badge"></i>
                    <span>Лікарі</span>
                </a>
            </li>
            <li>
                <a href="<?=Yii::$app->urlManager-
>createUrl(['/speciality/index'])?>">
                    <i class="bx bxs-copy-alt"></i>
                    <span>Спеціальності</span>
                </a>
            </li>
        <?php endif; ?>
        <li>
            <a href="<?=Yii::$app->urlManager->createUrl(['/site/index'])?>">
                <i class="bx bx-home"></i>
                <span>Перейти на сайт</span>
            </a>
        </li>
        <li>
```

```
            <a href="<?=Yii::$app->urlManager->createUrl(['/site/logout'])?>">
                <i class="bx bx-log-out-circle"></i>
                <span>Вийти</span>
            </a>
        </li>
    </ul>
</nav>

<div class="content-container">

    <div class="container-fluid">

        <?= Alert::widget() ?>
        <?= \app\widgets\BreadcrumbsWidget::widget([
            'options' => [
                'class' => 'breadcrumb',
            ],
            'homeLink' => [
                'label' => 'Головна',
                'url' => ['/'],
                'class' => 'home',
                'template' => '<li>{link}</li>',
            ],
            'links' => isset($this->params['breadcrumbs']) ? $this-
>params['breadcrumbs'] : [],
            'itemTemplate' => '<li>{link}</li>',
            'activeItemTemplate' => '<li class="active">{link}</li>',
            'tag' => 'ul',
            'encodeLabels' => false
        ]);
        ?>
        <?= $content ?>

    </div>
</div>

<?=
    $this->registerJsFile(
        '@web/js/dashboard.js',
        ['depends' => [\yii\web\JqueryAsset::class]]
    );
?>

<?php $this->endBody() ?>
```

```
</body>
</html>
<?php $this->endPage() ?>
```

SpecialityController.php

```
<?php

namespace app\controllers;

use Yii;
use app\models\Speciality;
use yii\data\ActiveDataProvider;
use yii\filters\AccessControl;
use yii\web\Controller;
use yii\web\NotFoundHttpException;
use yii\filters\VerbFilter;

/**
 * SpecialityController implements the CRUD actions for Speciality model.
 */
class SpecialityController extends Controller
{
    public $layout = 'dashboard.php';

    /**
     * @inheritDoc
     */
    public function behaviors()
    {
        return array_merge(
            parent::behaviors(),
            [
                'access' => [
                    'class' => AccessControl::className(),
                    'only' => ['index', 'update', 'delete'],
                    'rules' => [
                        [
                            'actions' => ['index', 'update', 'delete'],
                            'allow' => true,
                            'roles' => ['admin'],
                        ],
                    ],
                ],
```

```
                'verbs' => [
                    'class' => VerbFilter::className(),
                    'actions' => [
                        'delete' => ['POST'],
                    ],
                ],
            ]
        );
    }

    public function actionIndex()
    {
        $model = new Speciality();

        if ($this->request->isPost) {
            if ($model->load($this->request->post()) && $model->save()) {

                Yii::$app->session->setFlash('success', 'Спеціальність успішно
додано.');

                return $this->redirect(['index']);
            }
        } else {
            $model->loadDefaultValues();
        }

        $dataProvider = new ActiveDataProvider([
            'query' => Speciality::find(),
            'sort' => [
                'defaultOrder' => [
                    'title' => SORT_ASC,
                ]
            ],
        ]);

        return $this->render('index', [
            'dataProvider' => $dataProvider,
            'model' => $model,
        ]);
    }

    public function actionUpdate($id)
    {
        $model = $this->findModel($id);
```

```
        if ($this->request->isPost && $model->load($this->request->post()) &&
$model->save()) {

            Yii::$app->session->setFlash('success', 'Спеціальність успішно
відредаговано.');

            return $this->redirect(['index']);
        }

        return $this->render('update', [
            'model' => $model,
        ]);
    }

    public function actionDelete($id)
    {
        $this->findModel($id)->delete();

        return $this->redirect(['index']);
    }

    protected function findModel($id)
    {
        if (($model = Speciality::findOne(['id' => $id])) !== null) {
            return $model;
        }

        throw new NotFoundHttpException('Запитуваної сторінки не існує.');
    }
}
```

UserController.php

```
<?php

namespace app\controllers;

use app\models\AuthAssignment;
use app\models\DoctorInfo;
use app\models\Speciality;
use app\models\UpdatePasswordForm;
use app\models\UpdateUserInfoForm;
use app\models\User;
```

```
use app\models\UserInfo;
use Yii;
use yii\data\ActiveDataProvider;
use yii\filters\AccessControl;
use yii\web\Controller;
use yii\web\NotFoundHttpException;
use yii\web\UploadedFile;

class UserController extends Controller
{
    public $layout = 'dashboard.php';

    /**
     * @inheritDoc
     */
    public function behaviors()
    {
        return array_merge(
            parent::behaviors(),
            [
                'access' => [
                    'class' => AccessControl::className(),
                    'only' => ['all', 'doctors', 'delete', 'profile', 'release',
'status', 'view'],
                    'rules' => [
                        [
                            'actions' => ['all', 'doctors', 'delete', 'release',
'status', 'view'],
                            'allow' => true,
                            'roles' => ['admin'],
                        ],
                        [
                            'actions' => ['profile'],
                            'allow' => true,
                            'roles' => ['patient'],
                        ],
                    ],
                ],
            ]
        );
    }

    public function actionAll()
    {
```

```
        $dataProvider = new ActiveDataProvider([
            'query' => User::find()->andWhere(['not', ['id' => 1]]),
            'sort' => [
                'defaultOrder' => [
                    'name' => SORT_ASC,
                ]
            ],
        ]);

        return $this->render('all', [
            'dataProvider' => $dataProvider,
        ]);
    }

    public function actionDelete($id)
    {
        $model = $this->findModel($id);
        $model->delete();

        if(($user_info = UserInfo::findOne(['user_id' => $id])) !== null) {
            $photo = $user_info->photo;
            $user_info->delete();
            if($photo == $id . '-user.jpg') {
                unlink('img/users/' . $model->photo);
            }
        }

        if(($doctor_info = DoctorInfo::findOne(['user_id' => $id])) !== null) {
            $doctor_info->delete();
        }

        $aas = AuthAssignment::findOne(['user_id' => $id]);
        $aas->delete();

        Yii::$app->session->setFlash('success', 'Користувача успішно видалено.');

        return $this->redirect(['all']);
    }



    public function actionDoctors()
    {
        $users_id = [];
```

```
        $model = AuthAssignment::find()->where(['item_name' => 'doctor'])->all();

        foreach ($model as $item) {
            array_push($users_id, $item->user_id);
        }

        $dataProvider = new ActiveDataProvider([
            'query' => User::find()->where(['id' => $users_id])->andWhere(['not',
['id' => 1]]),
            'sort' => [
                'defaultOrder' => [
                    'name' => SORT_ASC,
                ]
            ],
        ]);


        $doctor_info = new DoctorInfo();
        $speciality = Speciality::find()->orderBy(['title' => SORT_ASC])->all();
        $users = User::find()->andWhere(['not', ['id' => $users_id]])-
>andWhere(['not', ['id' => 1]])->orderBy(['name' => SORT_ASC])->all();

        if ($this->request->isPost) {
            if ($doctor_info->load($this->request->post())) {
                $doctor_info->user_id = (int) $doctor_info->user_id;
                $doctor_info->speciality_id = (int) $doctor_info->speciality_id;
                $doctor_id = $doctor_info->user_id;
                $doctor_info->working_days = implode(';', $doctor_info-
>working_days);

                $doctor_info->body = '<p>Тепер ви лікар на сайті ' . Yii::$app-
>name . '</p>' .
                                     '<p>Ви можете увійти в ваш кабінет, щоб
переглянути записи від пацієнтів.</p>';

                if($doctor_info->save()  && $doctor_info-
>contact(User::findOne(['id' => 1])->email)) {
                    $aas = AuthAssignment::findOne(['user_id' => $doctor_id]);
                    $aas->item_name = 'doctor';
                    $aas->save();

                    Yii::$app->session->setFlash('success', 'Лікаря успішно додано
в базу.');
```

```
                return $this->refresh();
            }
        }
    }

    return $this->render('doctors', [
        'dataProvider' => $dataProvider,
        'doctor_info' => $doctor_info,
        'speciality' => $speciality,
        'users' => $users,
    ]);
}

public function actionGetDoctorRoom()
{
    if (Yii::$app->request->post('doctor')) {
        $doctor = Yii::$app->request->post('doctor');
        $room = '';

        if(($doctor = DoctorInfo::findOne(['user_id' => $doctor])) !== null) {
            $room = $doctor->room;
        }

        echo $room;
    }
}

public function actionGetDoctors()
{
    if (Yii::$app->request->post('speciality')) {
        $speciality = Yii::$app->request->post('speciality');
        $list = '<option value="">Оберіть лікаря...</option>';

        $all_doctors_arr = [];

        $doctor_info = DoctorInfo::find()->Where(['speciality_id' =>
$speciality])->all();

        $doctors_id = [];

        foreach ($doctor_info as $item) {
            array_push($doctors_id, $item->user_id);
        }
```

```
            $doctors = User::find()->where(['id' => $doctors_id, 'status' => 10])-
>orderBy(['name' => SORT_ASC])->all();
            foreach ($doctors as $item) {
                $all_doctors_arr[$item->id] = $item->name;
            }

            if (count($all_doctors_arr) == 0) {
                echo $list;
            } else {
                foreach ($all_doctors_arr as $key => $value) {
                    $list .= '<option value="'. $key .'">'. $value .'</option>';
                }
                echo $list;
            }
        }
    }

    public function actionGetDateTimeInfo()
    {
        if (Yii::$app->request->post('doctor')) {
            $doctor = Yii::$app->request->post('doctor');
            $working_time = '<div class="alert alert-danger" role="alert">Помилка
при визначенні графіка роботи лікаря. Обновіть сторінку і спробуйте ще раз.</div>';

            if(($doctor = DoctorInfo::findOne(['user_id' => $doctor])) !== null) {
                $week = ['Понеділок', 'Вівторок', 'Середа', 'Четвер', 'П\'ятниця',
'Субота', 'Неділя'];
                $days = explode(';', $doctor->working_days);
                $days_str = '';

                foreach ($days as $item) {
                    $days_str .= $week[$item - 1] . ', ';
                }

                $days_str = mb_substr($days_str, 0, -2);

                $time = explode('-', $doctor->working_time);
                $time_start = $time[0];
                $time_finish = $time[1];
                echo '<div class="alert alert-info" role="alert"><strong>Графік
роботи лікаря:</strong> ' . $days_str . ' з ' . $time_start . ' по ' . $time_finish
. '.</div>';
            } else {
                echo $working_time;
```

```
            }
        }
    }

    public function actionProfile()
    {

        $user_id = Yii::$app->user->identity->id;

        $user_info_form = new UpdateUserInfoForm();
        $user_password_form = new UpdatePasswordForm();

        if ($this->request->isPost) {
            if ($user_info_form->load($this->request->post())) {
                if($user_info_form->checkUser($user_id)) {
                    $user_info_form->imageFile =
UploadedFile::getInstance($user_info_form, 'imageFile');

                    if (($user = User::findOne(['id' => $user_id])) !== null) {
                        $user->name = $user_info_form->name;
                        $user->email = $user_info_form->email;

                        if ((UserInfo::findOne(['user_id' => $user_id])) !== null)
{
                            $user_info = UserInfo::findOne(['user_id' =>
$user_id]);
                        } else {
                            $user_info = new UserInfo();
                        }

                        $user_info->user_id = $user_id;
                        $user_info->birthday = $user_info_form->birthday;
                        $user_info->sex = $user_info_form->sex;
                        $user_info->phone = $user_info_form->phone;
                    } else {
                        Yii::$app->session->setFlash('warning', 'Виникла помилка
при збереженні даних.');

                        return $this->redirect(['profile']);
                    }

                    if ($user_info_form->imageFile) {
                        $user_info_form->image = $user_id . '-user.' .
$user_info_form->imageFile->extension;
```

```
                    $user_info_form->upload();
                    $user_info->photo = $user_info_form->image;
                }

                if ($user->save() && $user_info->save()) {
                    Yii::$app->session->setFlash('success', 'Профіль успішно
відредаговано.');
                } else {
                    Yii::$app->session->setFlash('warning', 'Виникла помилка
при збереженні даних.');

                    return $this->redirect(['profile']);
                }
            } else {
                $errorMsg= 'Користувач з таким e-mail вже існує.';
                $user_info_form->addError('email', $errorMsg);
            }
        }

        if ($user_password_form->load($this->request->post())) {
            if ($user_password_form->coincidence()) {
                $user = User::findOne(['id' => $user_id]);
                $user->setPassword($user_password_form->password);
                if ($user->save()) {
                    Yii::$app->session->setFlash('success', 'Пароль успішно
змінено.');

                    return $this->refresh();
                }
            } else {
                Yii::$app->session->setFlash('danger', 'Помилка введених даних,
паролі не співпадають. Відкрийте форму ще раз, введіть дані повторно та
збережіть.');
                $errorMsg= 'Паролі не співпадають.';
                $user_password_form->addError('repeat', $errorMsg);
            }
        }
    }

    if (($user = User::findOne(['id' => $user_id])) !== null) {
        $user_info_form->name = $user->name;
        $user_info_form->email = $user->email;
```

```
            if (($user_info = UserInfo::findOne(['user_id' => $user_id])) !== null)
{
                $user_info_form->birthday = $user_info->birthday;
                $user_info_form->sex = $user_info->sex;
                $user_info_form->phone = $user_info->phone;

                if(($doctor_info = DoctorInfo::findOne(['user_id' => $user_id]))
!== null) {

                    $doctor_info->working_days = explode(';', $doctor_info-
>working_days);

                    return $this->render('profile', [
                        'doctor_info' => $doctor_info,
                        'user' => $user,
                        'user_info' => $user_info,
                        'user_info_form' => $user_info_form,
                        'user_password_form' => $user_password_form,
                    ]);
                } else {
                    return $this->render('profile', [
                        'user' => $user,
                        'user_info' => $user_info,
                        'user_info_form' => $user_info_form,
                        'user_password_form' => $user_password_form,
                    ]);
                }
            } else {
                Yii::$app->session->setFlash('warning', 'Для запису до лікаря,
будь-ласка, заповніть всі дані. Натисніть на кнопку "Редагувати" та надішліть
форму.');

                if(($doctor_info = DoctorInfo::findOne(['user_id' => $user_id]))
!== null) {

                    $doctor_info->working_days = explode(';', $doctor_info-
>working_days);

                    return $this->render('profile', [
                        'doctor_info' => $doctor_info,
                        'user' => $user,
                        'user_info_form' => $user_info_form,
                        'user_password_form' => $user_password_form,
                    ]);
```

```
                } else {
                    return $this->render('profile', [
                        'user' => $user,
                        'user_info_form' => $user_info_form,
                        'user_password_form' => $user_password_form,
                    ]);
                }
            }


        }

        throw new NotFoundHttpException('Запитуваної сторінки не існує.');
    }

    public function actionRelease($id)
    {
        if(($model = AuthAssignment::findOne(['user_id' => $id])) !== null) {
            $model->item_name = 'patient' ;

            if($model->save()) {
                $doctor_info = DoctorInfo::findOne(['user_id' => $id]);
                $doctor_info->delete();

                Yii::$app->session->setFlash('success', 'Лікаря успішно видалено зі
списку лікарів.');

                return $this->redirect(['doctors']);
            }
        }

        throw new NotFoundHttpException('Запитуваної сторінки не існує.');
    }

    public function actionStatus($id)
    {
        $model = $this->findModel($id);
        $model->status = $model->status == 10 ? 0 : 10;
        $model->save();


        return $this->redirect(['all']);
    }
```

```
    public function actionView($id)
    {

        if (($user_info = UserInfo::findOne(['user_id' => $id])) !== null) {

            if(($doctor_info = DoctorInfo::findOne(['user_id' => $id])) !== null) {

                $doctor_info->working_days = explode(';', $doctor_info-
>working_days);

                $speciality = Speciality::find()->orderBy(['title' => SORT_ASC])-
>all();
                $users = User::find()->andWhere(['not', ['id' => $id]])-
>andWhere(['not', ['id' => 1]])->orderBy(['name' => SORT_ASC])->all();

                if ($this->request->isPost) {
                    if ($doctor_info->load($this->request->post())) {
                        $doctor_info->user_id = $id;
                        $doctor_info->speciality_id = (int) $doctor_info-
>speciality_id;
                        $doctor_id = $doctor_info->user_id;
                        $doctor_info->working_days = implode(';', $doctor_info-
>working_days);

                        if($doctor_info->save()) {
                            $aas = AuthAssignment::findOne(['user_id' =>
$doctor_id]);
                            $aas->item_name = 'doctor';
                            $aas->save();

                            Yii::$app->session->setFlash('success', 'Інформацію про
лікаря успішно змінено.');

                            return $this->refresh();
                        }
                    }
                }

                return $this->render('view', [
                    'doctor_info' => $doctor_info,
                    'speciality' => $speciality,
                    'users' => $users,
                    'user' => $this->findModel($id),
                    'user_info' => $user_info,
```

```
                ]);
            } else {
                return $this->render('view', [
                    'user' => $this->findModel($id),
                    'user_info' => $user_info,
                ]);
            }
        } else {
            Yii::$app->session->setFlash('warning', 'Користувач не заповнив всі
дані про себе.');

            if(($doctor_info = DoctorInfo::findOne(['user_id' => $id])) !== null) {

                $doctor_info->working_days = explode(';', $doctor_info-
>working_days);

                $speciality = Speciality::find()->orderBy(['title' => SORT_ASC])-
>all();
                $users = User::find()->andWhere(['not', ['id' => $id]])-
>andWhere(['not', ['id' => 1]])->orderBy(['name' => SORT_ASC])->all();

                if ($this->request->isPost) {
                    if ($doctor_info->load($this->request->post())) {
                        $doctor_info->user_id = $id;
                        $doctor_info->speciality_id = (int) $doctor_info-
>speciality_id;
                        $doctor_id = $doctor_info->user_id;
                        $doctor_info->working_days = implode(';', $doctor_info-
>working_days);

                        if($doctor_info->save()) {
                            $aas = AuthAssignment::findOne(['user_id' =>
$doctor_id]);
                            $aas->item_name = 'doctor';
                            $aas->save();

                            Yii::$app->session->setFlash('success', 'Інформацію про
лікаря успішно змінено.');

                            return $this->refresh();
                        }
                    }
                }
```

```
                return $this->render('view', [
                    'doctor_info' => $doctor_info,
                    'speciality' => $speciality,
                    'users' => $users,
                    'user' => $this->findModel($id),
                ]);
            } else {
                return $this->render('view', [
                    'user' => $this->findModel($id),
                ]);
            }
        }
    }

    /**
     * Finds the User model based on its primary key value.
     * If the model is not found, a 404 HTTP exception will be thrown.
     * @param int $id ID
     * @return User the loaded model
     * @throws NotFoundHttpException if the model cannot be found
     */
    protected function findModel($id)
    {
        if (($model = User::findOne(['id' => $id])) !== null) {
            return $model;
        }

        throw new NotFoundHttpException('Запитуваної сторінки не існує.');
    }
}
```

ДОДАТОК Б

Тест-кейси функціонального тестування

Таблиця Б.1 – Тест-кейси функціонального тестування

Id	Test Case Description	Precondition	Test Steps	Expected result	Passed/Failed
1	Перевірити можливість авторизації користувача з введенням валідних даних	1. Користувач знаходиться на сторінці входу. 2. Користувач має обліковий запис.	1.Введіть валідний Email. 2.Введіть валідний пароль. 3.Натисніть кнопку « Вхід».	1. Користувач успішно здійснив вхід в акаунт. 2. Система перенаправляє користувача в особистий кабінет.	Passed
2	Перевірити можливість користувача здійснити вхід з введенням валідного Email, але невалідного пароля	1. Користувач знаходиться на сторінці входу. 2. Користувач має обліковий запис.	1.Введіть валідний Email. 2.Введіть невалідний пароль. 3.Натисніть кнопку « Вхід».	З’явиться повідомлення: «Неправильне ім’я користувача або пароль»	Passed
3	Перевірити можливість користувача зареєструватися на сайті з введенням валідних даних	Користувач знаходиться на сторінці реєстрації	1. Натисніть на кнопку «Приєднатись». 2. Введіть ПІБ. 3. Введіть Email. 4. Введіть пароль. 5. Повторіть введення пароля. 6. Натісніть кнопку «Зареєструватись».	З’явиться повідомлення: «Реєстрація прошла успішно. Тепер Ви можете перейти в свій персональний кабінет»	Passed
4	Перевірити можливість користувача зареєструватись на сайті залишивши порожнє поле «Ваше ПІБ»	Користувач знаходиться на сторінці реєстрації	1. Натисність на кнопку «Приєднатись», 2. Залиште порожнім поле «Ваше ПІБ». 3. Введіть Email. 4. Введіть пароль.	З’явиться повідомлення: «Необхідно заповнити ваше ПІБ»	Passed

Продовження таблиці Б.1

			5. Повторіть введення пароля. 6. Натисніть кнопку «Зареєструватись».		
5	Перевірити можливість користувача зареєструватися на сайті ввівши некоректно Email	Користувач знаходиться на сторінці реєстрації	1. Натисність на кнопку «Приєднатись». 2. Введіть ПІБ. 3. Введіть некоректно Email. 4. Введіть пароль. 5. Повторіть введення пароля. 6. Натисніть кнопку «Зареєструватись».	З’явиться повідомлення: «Значення “Ваш Email” не є правильною Email адресою»	Passed
6	Перевірити можливість реєстрації існуючого користувача	1. Користувач знаходиться на сторінці реєстрації. 2. Користувач з такою електронною поштою зареєстрований в системі.	1. Натисність на кнопку «Приєднатись», 2. Введіть ПІБ. 3. Введіть Email. 4. Введіть пароль. 5. Повторіть введення пароля. 6. Натисніть кнопку «Зареєструватись».	З’явиться повідомлення: «Користувач вже зареєстрований»	Passed
7	Перевірити здатність користувача до редагування особистих даних	1. Користувач авторизувався. 2. Користувач знаходиться на сторінці «Мій профіль».	1. Натисніть на кнопку «Редагувати». 2. Змініть за необхідності ПІБ. 3. Заповніть Дату народження. 4. Оберіть стать. 5. Змініть за необхідності Email. 6. Заповніть поле з телефоном. 7. За бажанням додайте фотографію. 8. Натисніть кнопку «Зберегти»,	З’явиться повідомлення: «Профіль успішно відредаговано»	Passed

Продовження таблиці Б.1

8	Перевірити можливість користувача зберегти особисті дані залишивши порожнє поле	1. Користувач авторизувався. 2. Користувач знаходиться на сторінці «Мій профіль».	1. Натисніть на кнопку «Редагувати». 2. Змініть за необхідності ПІБ. 3. Заповніть Дату народження. 4. Оберіть стать. 5. Змініть за необхідності Email. 6. Залиште порожнім поле «...». 7. За бажанням додайте фотографію. 8. Натисніть кнопку «Зберегти»,	З’явиться повідомлення: «Необхідно заповнити “…”»	Passed
9	Перевірити можливість змі-ни пароля	1. Користувач авторизувався. 2. Користувач знаходиться на сторінці «Мій профіль».	1. Натисніть на кнопку «Пароль». 2. Введіть пароль. 3. Повторіть пароль.	З’явиться повідомлення: «Пароль успішно змінено»	Passed
10	Перевірити можливість змі-ни пароля при невідповідності заповнення форми	1. Користувач авторизувався. 2. Користувач знаходиться на сторінці «Мій профіль».	1. Натисніть на кнопку «Пароль». 2. Введіть пароль. 3. Введіть повторно пароль з помилкою.	З’явиться повідомлення: «Помилка введення даних, паролі не співпадають»	Passed
11	Перевірити можливість користувача запису на прийом	1. Користувач авторизувався. 2. Користувач знаходиться на сторінці «Запис на прийом».	1. Оберіть спеціальність. 2. Оберіть лікаря. 3. Відповідно до графіка роботи лікаря, оберіть дату та час. 4. Заповніть за необхідності примітку. 5. Натисніть кнопку «Зберегти».	З’явиться повідомлення: «Ви успішно подали заявку на прийом до лікаря. Переглянути статус можна на сторінці Прийоми»	Passed
12	Перевірити можливість користувача редагувати запис	1. Користувач авторизувався. 2. Користувач знаходиться на сторінці «Особисті записи».	1. Натисніть кнопку «Оновити». 2. Змініть дату та час. 3. Внесіть за необхідності зміни у примітці.	З’явиться повідомлення: «Запис успішно відредаговано»	Passed

Продовження таблиці Б.1

			4. Натисніть кнопку «Зберегти».		
13	Перевірити можливість користувача видалити запис	1. Користувач авторизувався. 2. Користувач знаходиться на сторінці «Особисті записи».	1. Натисніть кнопку «Видалити запис». 2. Підтвердіть випливаюче повідомлення, натиснувши «ОК».	З’явиться повідомлення: «Запис успішно видалено»	Passed
14	Перевірити здатність користувача до перегляду попередніх записів після підтвердження адміністратора	1. Користувач авторизувався. 2. Користувач знаходиться на сторінці «Особисті записи».	1. Натисніть кнопку «Календар». 2. Оберіть можливість перегляду натиснувши кнопку «Місяць», «Тиждень» або «День.	1. Відбувається перехід на сторінку «Особистий календар». 2. Система виконує оновлення сторінки після вибору часового проміжку для перегляду.	Passed
15	(2) Перевірити здатність користувача до перегляду попередніх записів	1. Користувач авторизувався. 2. Користувач знаходиться на сторінці «Особисті записи».	-	Коректне відображення персональних записів на прийом	Passed
16	Перевірити здатність користувача до збереження електронного талону	1. Користувач авторизувався. 2. Користувач знаходиться на сторінці «Особисті записи».	Натисніть кнопку «Талон»	Електронний талон успішно збережено на електронний пристрій	Passed

ДОДАТОК В

Код автоматизованого тестування

```
      <?xml version="1.0" encoding="UTF-8"?>
      <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
      <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

      <head profile="http://selenium-ide.openqa.org/profiles/test-case">
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
      <link rel="selenium.base" href="http://127.0.0.1/" />
      <title>BadRegistation</title>
      </head>

      <body>
      <table cellpadding="1" cellspacing="1" border="1">
      <thead>
      <tr><td rowspan="1" colspan="3">BadRegistation</td></tr>
      </thead><tbody>
      <tr>
            <td>open</td>
            <td>/clinic/login/</td>
            <td></td>
      </tr>
      <tr>
            <td>clickAndWait</td>
            <td>css=#menu-item-52 &gt; a &gt; span</td>
            <td></td>
      </tr>
      <tr>
            <td>type</td>
            <td>id=wpcrl_email</td>
            <td>nazar</td>
      </tr>
      <tr>
            <td>type</td>
            <td>id=wpcrl_password</td>
            <td>nazar</td>
      </tr>
      <tr>
            <td>type</td>
            <td>id=wpcrl_fname</td>
            <td>Asfasf</td>
      </tr>
      <tr>
            <td>type</td>
            <td>id=wpcrl_lname</td>
```

```
            <td>Afasf</td>
      </tr>
      <tr>
            <td>click</td>
            <td>id=wpcrl_username</td>
            <td></td>
      </tr>
      <tr>
            <td>type</td>
            <td>id=wpcrl_username</td>
            <td>afsafsa</td>
      </tr>
      <tr>
            <td>type</td>
            <td>id=wpcrl_email</td>
            <td>asfsafa</td>
      </tr>
      <tr>
            <td>type</td>
            <td>id=wpcrl_password</td>
            <td>asfasf</td>
      </tr>
      <tr>
            <td>click</td>
            <td>id=wpcrl_password2</td>
            <td></td>
      </tr>
      <tr>
            <td>type</td>
            <td>id=wpcrl_password2</td>
            <td>agsgga</td>
      </tr>
      <tr>
            <td>type</td>
            <td>name=wpcrl_captcha</td>
            <td>sfgs</td>
      </tr>

      </tbody></table>
      </body>
      </html>

      <?xml version="1.0" encoding="UTF-8"?>
      <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
      <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
      <head profile="http://selenium-ide.openqa.org/profiles/test-case">
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
      <link rel="selenium.base" href="http://127.0.0.1/" />
```

```
	<title>ViewSchedule</title>
	</head>
	<body>
	<table cellpadding="1" cellspacing="1" border="1">
	<thead>
	<tr><td rowspan="1" colspan="3">ViewSchedule</td></tr>
	</thead><tbody>
	<tr>
		<td>open</td>
		<td>/clinic/</td>
		<td></td>
	</tr>
	<tr>
		<td>clickAndWait</td>
		<td>id=87</td>
		<td></td>
	</tr>
	<tr>
		<td>clickAndWait</td>
		<td>//ul[@id='specialities-list']/a[5]/li</td>
		<td></td>
	</tr>

	</tbody></table>
	</body>
	</html>

	<?xml version="1.0" encoding="UTF-8"?>
	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
	<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
	<head profile="http://selenium-ide.openqa.org/profiles/test-case">
	<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
	<link rel="selenium.base" href="http://127.0.0.1/" />
	<title>New Test</title>
	</head>
	<body>
	<table cellpadding="1" cellspacing="1" border="1">
	<thead>
	<tr><td rowspan="1" colspan="3">New Test</td></tr>
	</thead><tbody>
	<tr>
		<td>open</td>
		<td>/clinic/login/</td>
		<td></td>
	</tr>
	<tr>
		<td>clickAndWait</td>
		<td>css=#menu-item-52 &gt; a &gt; span</td>
```

```
      <td></td>
</tr>
<tr>
      <td>type</td>
      <td>id=wpcrl_email</td>
      <td>nazar</td>
</tr>
<tr>
      <td>type</td>
      <td>id=wpcrl_password</td>
      <td>nazar</td>
</tr>
<tr>
      <td>type</td>
      <td>id=wpcrl_fname</td>
      <td>Asfasf</td>
</tr>
<tr>
      <td>type</td>
      <td>id=wpcrl_lname</td>
      <td>Afasf</td>
</tr>
<tr>
      <td>click</td>
      <td>id=wpcrl_username</td>
      <td></td>
</tr>
<tr>
      <td>type</td>
      <td>id=wpcrl_username</td>
      <td>afsafsa</td>
</tr>
<tr>
      <td>type</td>
      <td>id=wpcrl_email</td>
      <td>asfsafa</td>
</tr>
<tr>
      <td>type</td>
      <td>id=wpcrl_password</td>
      <td>asfasf</td>
</tr>
<tr>
      <td>click</td>
      <td>id=wpcrl_password2</td>
      <td></td>
</tr>
<tr>
```

```
            <td>type</td>
            <td>id=wpcrl_password2</td>
            <td>agsgga</td>
      </tr>
      <tr>
            <td>type</td>
            <td>name=wpcrl_captcha</td>
            <td>sfgs</td>
      </tr>

      </tbody></table>
      </body>
      </html>

      <?xml version="1.0" encoding="UTF-8"?>
      <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
      <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
      <head profile="http://selenium-ide.openqa.org/profiles/test-case">
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
      <link rel="selenium.base" href="http://127.0.0.1/" />
      <title>New Test</title>
      </head>
      <body>
      <table cellpadding="1" cellspacing="1" border="1">
      <thead>
      <tr><td rowspan="1" colspan="3">New Test</td></tr>
      </thead><tbody>
      <tr>
            <td>open</td>
            <td>/clinic/doctors/gregory_house/?clinic=87&amp;speciality=%D0%A2%D0%B
            5%D1%80
%D0%B0%D0%BF%D0%B5%D0%B2%D1%82&amp;room=409</td>
            <td></td>
      </tr>
      <tr>
            <td>clickAndWait</td>
            <td>link=авторизуватися!</td>
            <td></td>
      </tr>
      <tr>
            <td>type</td>
            <td>id=wpcrl_username</td>
            <td>nazar</td>
      </tr>
      <tr>
            <td>type</td>
            <td>id=wpcrl_password</td>
            <td>nazar</td>
```

```
</tr>
<tr>
	<td>click</td>
	<td>css=button.btn.btn-primary</td>
	<td></td>
</tr>
<tr>
	<td>clickAndWait</td>
	<td>css=#menu-item-263 &gt; a &gt; span</td>
	<td></td>
</tr>

</tbody></table>
</body>
</html>
```