Міністерство освіти і науки України
Університет митної справи та фінансів

Факультет інноваційних технологій
Кафедра комп'ютерних наук та інженерії програмного забезпечення

Кваліфікаційна робота бакалавра

на тему : «Розроблення вебзастосунку автоматизації роботи поліклініки»

Виконав: студент групи ІПЗ20-1
Спеціальність 121 «Інженерія програмного забезпечення»
Кокойда Валерій Олександрович
(прізвище та ініціали)
Керівник д.е.н., проф. Корнєєв М.В.
(науковий ступінь, вчене звання, прізвище та ініціали)
Рецензент Університет митної справи та фінансів
(місце роботи)
Професор кафедри кібербезпеки та інформаційних технологій
(посада)
д.е.н., професор Паршина О.А.
(науковий ступінь, вчене звання, прізвище та ініціали)

Дніпро – 2024

АНОТАЦІЯ

Кокойда В.О. Розроблення вебзастосунку автоматизації роботи поліклініки.

Кваліфікаційна робота на здобуття освітнього ступеня бакалавр за спеціальністю 121 «Інженерія програмного забезпечення». – Університет митної справи та фінансів, Дніпро, 2024.

Проект розробки вебзастосунку поліклініки має на меті створення зручного та ефективного сервісу для пацієнтів та лікарів. Під час реалізації проекту було проведено аналіз потреб користувачів, обрані оптимальні технології та розроблений функціонал.

У цьому проекті було створено детальний опис бази даних, а також реалізована система з різними ролями користувачів, що дозволяє забезпечити надійний контроль за безпекою та конфіденційністю даних.

Досліджено можливості розроблення вебзастосунку для поліклініки з використанням сучасних технологій програмування та баз даних. В роботі проведено аналіз предметної області та існуючого програмного забезпечення для подальшого визначення мети та завдань проекту. Обґрунтовано актуальність та значення розв'язання задачі автоматизації процесів роботи поліклініки через створення вебзастосунку. Розглянуто вибір програмних засобів, мов програмування та баз даних для реалізації проекту. Запропоновано структуру та функціонал вебзастосунку, що відповідає потребам поліклініки. Проаналізовано технічні аспекти розробки, включаючи опис бази даних, запити до неї, інтерфейсну частину, стиль вебзастосунку та мережевий вебінтерфейс. Зроблено висновки щодо практичного значення отриманих результатів та вирішених задач у цьому проекті.

Ключові слова: вебзастосунок, база даних, інтерфейси, мова програмування C#, ASP.NET, ADO.NET, MSSQLServer.

ABSTRACT

Kokoida V. A. Development of the "Clinic" Web Application.

Qualification of work to obtain a bachelor's degree in specialty 121 «Software Engineering». – University of Customs and Finance, Dnipro, 2024.

The project of developing a web application for the clinic aims to create a convenient and efficient service for patients and doctors. During the project implementation, an analysis of user needs was conducted, optimal technologies were selected, and functionality was developed.

In this project, a detailed description of the database was created, and a system with different user roles was implemented to ensure reliable control over data security and confidentiality.

The possibilities of developing a web application for the clinic using modern programming technologies and databases have been explored. The work analyzes the subject area and existing software to determine the purpose and objectives of the project. The relevance and significance of solving the problem of automating clinic processes through the creation of a web application are substantiated. The choice of software tools, programming languages, and databases for project implementation is considered. The structure and functionality of the web application, which meets the needs of the clinic, are proposed. Technical aspects of development, including a description of the database, queries to it, the interface part, the style of the web application, and the network web interface, are analyzed. Conclusions are drawn regarding the practical significance of the obtained results and the tasks solved in this project.

Keywords: web application, database, interfaces, programming language C#, ASP.NET, ADO.NET, MSSQLServer.

.

ЗМІСТ

ВСТУП 6
РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ. ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ 9
1.1 Опис процесів роботи вебзастосунку «Поліклініка» 9
1.2 Огляд існуючого програмного забезпечення для роботи вебзастосунку поліклініка 10
1.3 Дослідження сучасних інструментів для розроблення вебзастосунку «Поліклініка» 13
1.4 Висновки до першого розділу. Постановка задач дослідження 20
РОЗДІЛ 2 АНАЛІЗ ЗАСОБІВ РОЗРОБЛЕННЯ ВЕБЗАСТОСУНКУ ПОЛІКЛІНІКА 21
2.1 Вибір програмних засобів для розроблення вебзастосунку поліклініка 21
2.2 Мова програмування C# 21
2.3 ASP.NET 28
2.4 ADO.NET 30
2.5 MSSQLServer 32
2.6 Висновок до другого розділу 35
РОЗДІЛ 3 РОЗРОБЛЕННЯ ВЕБЗАСТОСУНКУ ПОЛІКЛІНІКА 36
3.1 Опис бази даних 36
3.2 Запити до бази даних 38
3.3 Інтерфейсна частина вебзастосунку 39
3.4 Стиль вебзастосунку 41
3.5 Мережевий вебінтерфейс 42
3.6 Опис програмних механізмів 44
3.7 Висновки до третього розділу 53
ВИСНОВКИ 54
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 57
ДОДАТОК А 58
ДОДАТОК Б 59

ВСТУП

У сучасному світі технологій та цифрової трансформації медичні заклади все частіше використовують інформаційні системи для покращення якості медичного обслуговування та оптимізації внутрішніх процесів. Одним із ключових інструментів в цьому процесі є вебзастосунки, спрямовані на автоматизацію робочих процесів, підвищення доступності медичних послуг та покращення взаємодії з пацієнтами.

Ця робота присвячена дослідженню та розробці вебзастосунку для поліклініки з метою покращення її ефективності та зручності для пацієнтів та медичного персоналу. Вона включає в себе аналіз сучасних інструментів розроблення вебзастосунків, вибір оптимальних технологій та інструментів, розробку бази даних, створення інтерфейсу користувача та імплементацію програмних механізмів для оптимізації роботи поліклініки.

Подальше дослідження та розробка вебзастосунку поліклініки має великий потенціал у покращенні якості медичного обслуговування, зменшенні бюрократичних процедур та підвищенні задоволеності пацієнтів від медичних послуг.

Мета створення вебзастосунку для поліклініки полягає в поліпшенні доступності та ефективності надання медичних послуг для пацієнтів, а також оптимізації робочих процесів для медичного персоналу.

У рамках цієї роботи буде проведений аналіз засобів розроблення програмного забезпечення для роботи вебзастосунку поліклініки, визначені вимоги до системи та розглянуті основні функціональні можливості. Також буде описана база даних, ролі користувачів, механізми реєстрації та авторизації, а також інші важливі аспекти системи. Результатом цієї роботи буде розроблення функціонального та ефективного вебзастосунку.

Актуальність вебзастосунку «Поліклініка» обґрунтовується факторами, що впливають на сучасну медичну систему.

1) підвищений обсяг медичної інформації: З впровадженням

електронної медичної документації та розвитком технологій збільшується обсяг медичної інформації, що потребує зберігання, обробки та доступу. Вебзастосунок допомагає керувати цим обсягом даних та забезпечує швидкий та зручний доступ до них.

2) зростання попиту на медичні послуги: У зв'язку зі збільшенням кількості населення та покращенням медичної освіти зростає попит на медичні послуги. Вебзастосунок дозволяє оптимізувати робочі процеси та забезпечує ефективніше надання послуг при зростанні обсягу роботи.

3) потреба в покращенні доступності медичної допомоги: За допомогою вебзастосунку пацієнти можуть легко записуватися на прийом до лікаря, отримувати консультації онлайн, переглядати результати обстежень та моніторити свій стан здоров'я, що сприяє покращенню доступності медичної допомоги.

4) необхідність впровадження ефективного управління медичною установою: Вебзастосунок дозволяє автоматизувати ряд процесів управління поліклінікою, таких як облік пацієнтів, призначення медичних процедур, управління рецептами та лікарськими засобами, що допомагає оптимізувати внутрішні ресурси та покращує роботу медичного персоналу.

5) забезпечення безпеки та конфіденційності даних: З огляду на чутливість медичної інформації, особливу увагу потрібно приділити захисту даних та дотриманню вимог щодо конфіденційності. Вебзастосунок повинен забезпечувати високий рівень захисту даних та відповідати вимогам законодавства щодо обробки медичної інформації.

Вебзастосунок «Поліклініка» має за мету забезпечення ефективного та зручного способу ведення медичного обліку та взаємодії між пацієнтами та медичним персоналом. Його завдання включають управління прийомами та записом на них, зберігання та обробку медичної інформації, забезпечення доступу до онлайн консультацій, ведення моніторингу стану здоров'я пацієнтів, управління рецептами та лікарськими засобами, а також забезпечення безпеки та конфіденційності даних. Він також може надавати

звітність та аналітику для вдосконалення роботи поліклініки та підвищення рівня обслуговування пацієнтів.

Поставлене завдання з розробки вебзастосунку для поліклініки надзвичайно актуальне в сучасних умовах швидкої цифрової трансформації в медичній сфері. Активне використання інформаційних технологій в медицині вже стало стандартом для покращення доступності медичних послуг, оптимізації процесів та підвищення якості надання медичної допомоги.

Розв'язання цього завдання має велике значення для предметної області, оскільки дозволяє поліклінікам оптимізувати робочі процеси, автоматизувати реєстрацію пацієнтів та призначення медичних прийомів, забезпечує зручну взаємодію з медичним персоналом та пацієнтами, а також сприяє збереженню та аналізу медичної інформації для підвищення ефективності медичного процесу в цілому. Такий вебзастосунок стає необхідним інструментом для сучасних медичних закладів у досягненні їхніх цілей щодо покращення обслуговування пацієнтів та оптимізації робочих процесів.

Завдання цієї роботи полягає в розробці та впровадженні вебзастосунку для поліклініки з метою покращення якості медичного обслуговування та оптимізації робочих процесів.

Впровадження вебзастосунку для поліклініки дозволить зменшити час очікування пацієнтів на прийом, покращити доступність медичної інформації, спростити процес запису на прийом та сприятиме підвищенню задоволення пацієнтів від медичного обслуговування. Крім того, це також дозволить поліклінікам ефективніше керувати ресурсами, покращити звітність та аналітику та забезпечити більш точне та швидке прийняття рішень.

Кваліфікаційна робота складається зі вступу, 3-х розділів, висновків, використаних джерел з 16 найменувань, 2-х додатків.

Обсяг роботи 77 сторінок, містить 6 таблиць та 12 рисунків.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ. ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Опис процесів роботи вебзастосунку «Поліклініка»

Вебзастосунок «Поліклініка» – це онлайн платформа, розроблена для автоматизації та поліпшення різних аспектів роботи медичної установи. Він може включати в себе різноманітні функції та можливості, спрямовані на полегшення роботи медичного персоналу, зручність для пацієнтів та ефективне управління установою.

Процеси роботи вебзастосунку «Поліклініка» можуть включати такі етапи:

1) користувачі реєструються в системі, надаючи особисті дані, такі як ім'я, адреса електронної пошти, номер телефону тощо;

2) після реєстрації користувачі мають можливість увійти в систему, використовуючи свої облікові дані. Після успішної аутентифікації вони отримують доступ до функціоналу вебзастосунку відповідно до їхніх прав доступу;

3) користувачі можуть переглядати графік роботи лікарів, вибирати підходящий час для прийому та записуватися на прийом до вибраного лікаря;

4) система може надсилати користувачам підтвердження запису на прийом, нагадування про наближення дати прийому, а також інші повідомлення, пов'язані з їхнім здоров'ям та медичними процедурами;

5) користувачі можуть переглядати свою медичну історію, результати обстежень, аналізи, рецепти та іншу важливу медичну інформацію через вебзастосунок;

6) застосунок може надавати можливість проведення дистанційних консультацій з лікарем через відеозв'язок або чат, щоб забезпечити доступ до медичної допомоги навіть на відстані;

7) користувачі можуть оплачувати медичні послуги онлайн через

застосунок, використовуючи різні методи оплати, такі як кредитні картки, електронні гроші тощо;

8) адміністратори поліклініки можуть генерувати звіти та аналітичні дані про роботу закладу, включаючи статистику про записи на прийом, фінансові показники, задоволення клієнтів тощо.

1.2 Огляд існуючого програмного забезпечення для роботи вебзастосунку поліклініка

Приклади деяких відомих вітчизняних та закордонних вебзастосунків:

1) Інтерфейс «BookMyDoctor» зображено на рисунку 1.1.

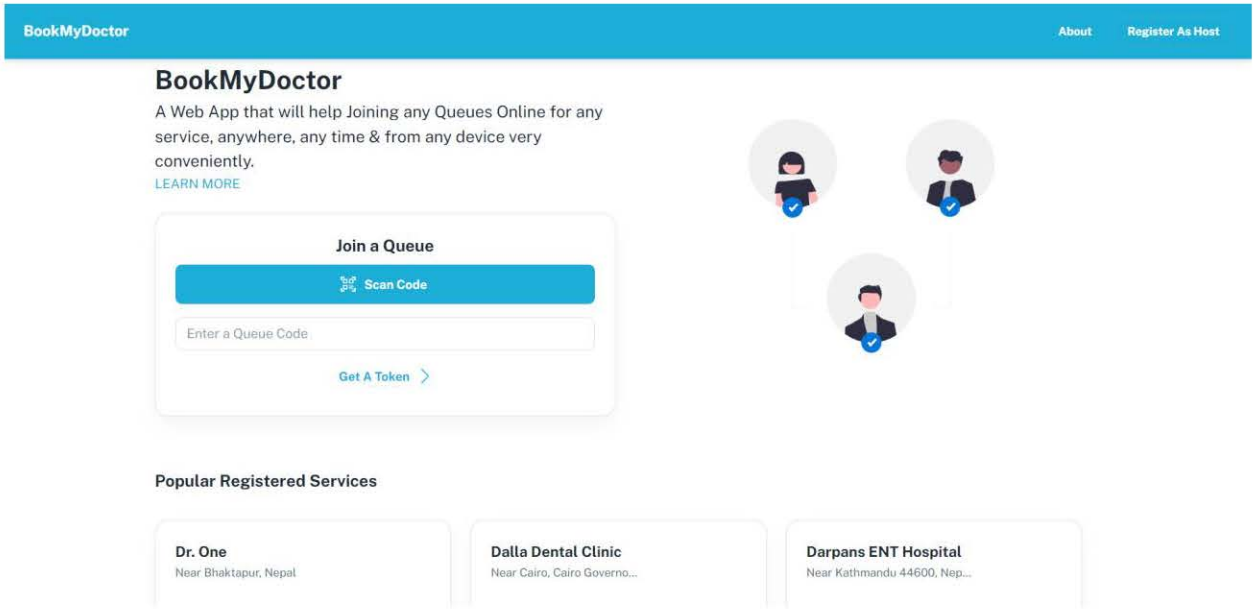

Рисунок 1.1 – Інтерфейс BookMyDoctor

Вебзастосунок BookMyDoctor надає можливість пацієнтам зручно записатися на прийом до лікаря в зручний для них час, шукати лікарів за різними критеріями, проводити онлайн консультації, керувати своїми прийомами, отримувати нагадування про наближені прийоми, переглядати медичну інформацію та історію відвідувань, виставляти рахунки та створювати звіти для страхових компаній, редагувати особистий профіль,

залишати відгуки та оцінювати лікарів.

2) «eZdravie – це вебзастосунок, який надає користувачам можливість швидко і зручно користуватися медичними послугами. Він дозволяє пацієнтам шукати лікарів та клініки, записуватися на прийоми в режимі онлайн, отримувати консультації віддалено, переглядати свою медичну історію та рецепти, вносити оплату за послуги, здійснювати взаємодію зі своїм медичним страхувальником, а також отримувати нагадування про прийоми та попередження щодо необхідних медичних процедур чи аналізів. Крім того, вебзастосунок може надавати корисну інформацію про різні аспекти здоров'я та медицини, таку як статті, поради та новини. (рис. 1.2).

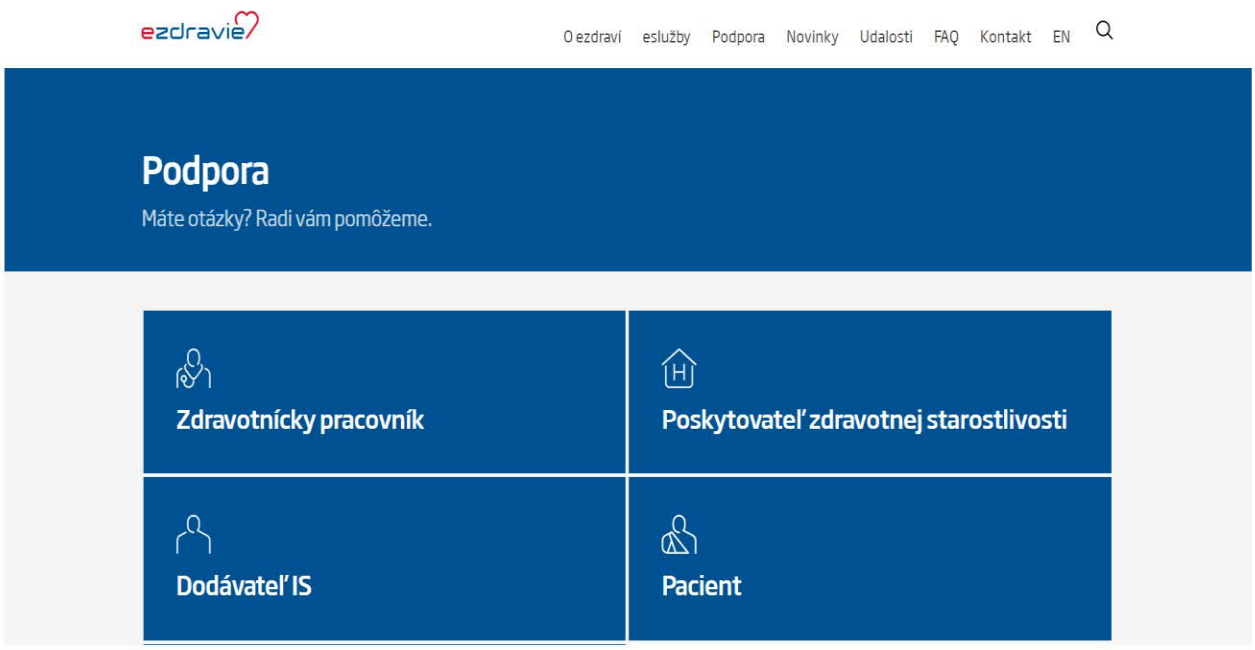

Рисунок 1.2 – Інтерфейс eZdravie

3) MedApp надає користувачам можливість керувати своїм здоров'ям та медичними даними в онлайн-режимі. Користувачі можуть вести свою медичну картку, записувати результати обстежень та аналізів, отримувати рецепти та консультації від лікарів, вносити дані про прийоми ліків та процедур, а також вести щоденник здоров'я, в якому можна фіксувати показники та симптоми. Крім того, застосунок може надавати корисну інформацію про захворювання,

лікування та профілактику різних захворювань, а також нагадування про необхідні вакцинації та медичні процедури. Користувачі також можуть спілкуватися з іншими пацієнтами, обмінюватися досвідом та порадами щодо здоров'я (рис. 1.3).

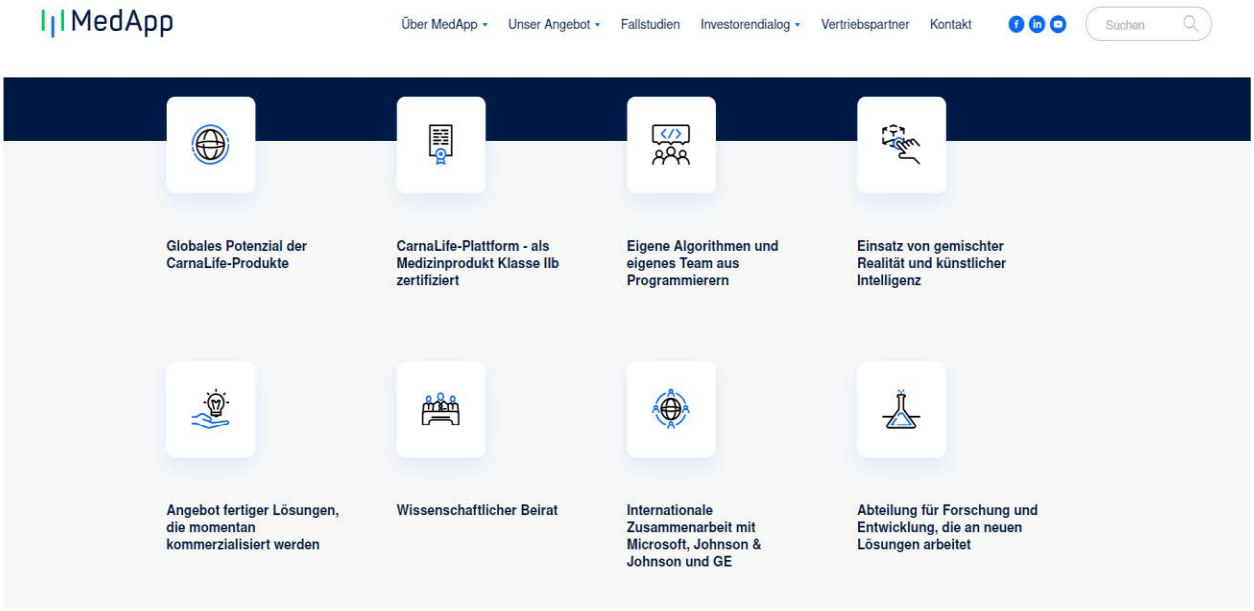

Рисунок 1.3 – Інтерфейс MedApp

4) «eHealth» – це вебзастосунок, який надає широкий спектр послуг у сфері електронного здоров'я. Сервіс дозволяє користувачам зберігати та керувати своїми медичними записами, включаючи результати обстежень, аналізи, лікування та рецепти. Крім того, він може надавати доступ до медичної інформації в реальному часі, що дозволяє швидко реагувати на зміни у стані здоров'я та отримувати консультації від лікарів в онлайн-режимі. Сервіс також може надавати корисну інформацію про захворювання, лікування та профілактику різних захворювань, а також нагадувати про необхідні вакцинації та медичні процедури. Крім того, він може включати можливості моніторингу здоров'я, спілкування з іншими пацієнтами та доступ до онлайн-ресурсів для підтримки здорового способу життя (рис. 1.4).

Рисунок 1.4 – Інтерфейс eHealth

6) Вебзастосунок «Мій лікар» призначений для покращення взаємодії між лікарем та пацієнтом. Сервіс надає користувачам можливість легко знайти лікаря, записатися на прийом, переглянути історію візитів та отримати консультацію в онлайн-режимі. Крім того, Мій лікар може надавати доступ до медичної інформації, включаючи результати обстежень, аналізи та рецепти, що дозволяє пацієнтам бути в курсі свого стану здоров'я. Сервіс також може надавати корисну інформацію про захворювання, лікування та профілактику різних захворювань, а також нагадувати про необхідні вакцинації та медичні процедури. Крім того, Мій лікар може включати можливості моніторингу здоров'я, спілкування з іншими пацієнтами та доступ до онлайн-ресурсів для підтримки здорового способу життя. Інтерфейс зображено на рисунку 1.5.

Кожен з цих вебзастосунків має свої унікальні можливості.

BookMyDoctor – допомагає користувачам швидко знайти лікаря та записатися на прийом.

eZdravie – надає широкий спектр медичних послуг в онлайн-форматі, включаючи консультації, дистанційну діагностику та прийом лікаря.

7)

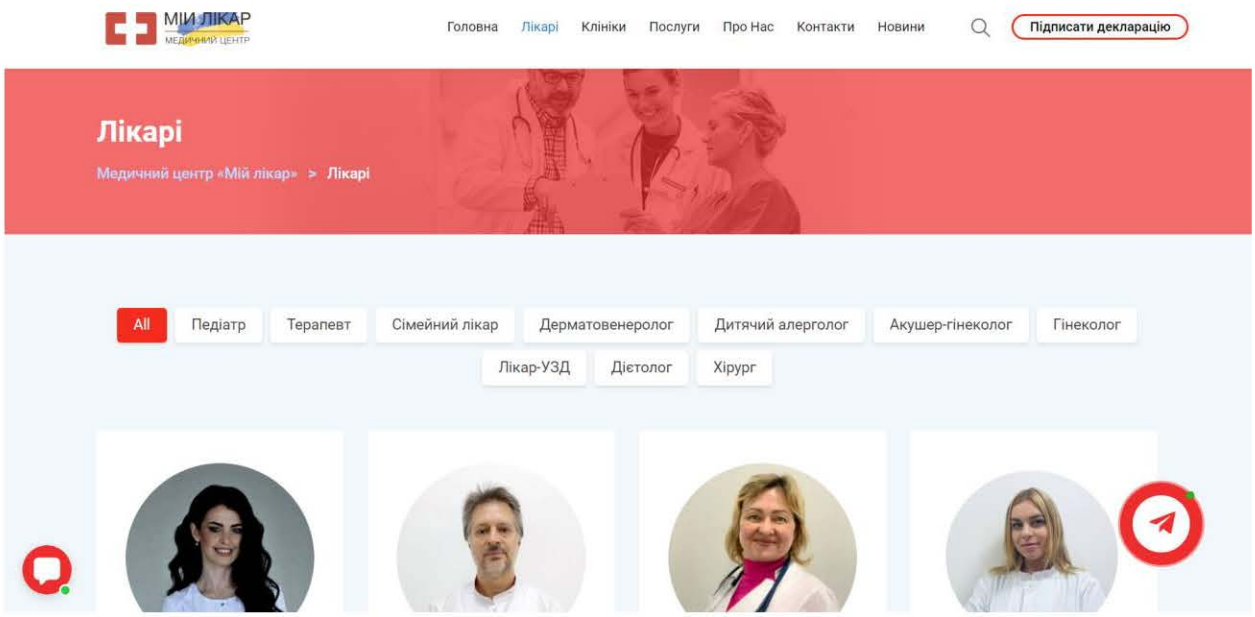

Рисунок 1.5 – Інтерфейс Мій лікар

MedApp – це мобільний додаток, який дозволяє користувачам вести облік медичних показників, отримувати нагадування про прийом лікарів та медикаментів, а також спілкуватися з медичними фахівцями.

eHealth – цей застосунок надає доступ до медичної інформації, включаючи результати обстежень, аналізи та рецепти, а також може надавати корисну інформацію про захворювання та профілактику.

Мій лікар – допомагає пацієнтам знайти лікаря, записатися на прийом, переглянути історію візитів, отримати консультацію в онлайн-режимі, а також надає доступ до медичної інформації та ресурсів для підтримки здорового способу життя.

1.3 Дослідження сучасних інструментів для розроблення вебзастосунку «Поліклініка»

Дослідження сучасних інструментів для розроблення вебзастосунку поліклініки включає аналіз різних мов програмування, які можна використовувати для створення різних частин системи.

C# є популярною мовою програмування для розробки вебзастосунків на платформі .NET. Вона має широкі можливості, включаючи підтримку об'єктно-орієнтованого програмування та інтеграцію з ASP.NET для створення вебсайтів і вебдодатків.

Мова Python використовується для широкого спектру застосувань, включаючи веброзробку. Вона має простий синтаксис, що робить її легкою для вивчення та розробки, і має багато фреймворків, таких як Django та Flask, для створення вебзастосунків.

JavaScript є ключовою мовою для розробки клієнтської частини вебзастосунків. Вона використовується разом з HTML та CSS для створення інтерактивних вебсайтів і вебдодатків. Крім того, JavaScript також використовується на сервері за допомогою платформи Node.js.

Java широко використовується для розробки вебдодатків на сервері за допомогою фреймворка Spring або Java EE. Вона має велику спільноту розробників і робить великий акцент на безпеку та надійність.

PHP також дуже часто застосовується при розробці вебзастосунків на сервері. Вона є основною мовою для багатьох вебфреймворків, таких як Laravel та Symfony, і має велику спільноту розробників.

Ruby є мовою програмування, яка використовується для розробки вебзастосунків, зокрема з використанням фреймворку Ruby on Rails. Вона відома своєю простотою та продуктивністю розробки.

Go є мовою програмування, розробленою Google, яка має швидкий час виконання і підтримує конкурентність. Вона може бути використана для розробки вебдодатків та мікрослужб.

Кожна з цих мов програмування має свої переваги та недоліки, і вибір мови залежить від потреб проекту, досвіду команди розробників та власних вподобань.

Мови програмування для розроблення онлайн поліклініки можуть відрізнятися за різними критеріями, такими як швидкість виконання, зручність у використанні, екосистема інструментів та фреймворків, безпека та інші.

По-перше, швидкість та продуктивність. Деякі мови програмування, такі як Go та Java, відомі своєю швидкістю виконання та продуктивністю. Це важливо для онлайн систем поліклінік, де може бути велика кількість користувачів і потрібна миттєва реакція на їхні запити.

По-друге, безпека. Мови програмування, які мають розширену систему типізації, такі як C# або Java, можуть бути більш безпечними у використанні для розробки медичних систем, де важлива конфіденційність та інтегрітет даних.

По-третє, екосистема та підтримка. Різні мови програмування мають розгалужену екосистему інструментів та фреймворків, що полегшує розробку та підтримку вебдодатків. Наприклад, C# має широку підтримку з боку Microsoft і активну спільноту розробників.

По-четверте, простота використання. Python та JavaScript відомі своєю простотою вивчення та використання, що може бути важливим для команд, які швидко хочуть розпочати роботу над проектом.

По-п'яте, сумісність з вебтехнологіями. Мови програмування, які мають широку підтримку для вебтехнологій, таких як HTML, CSS та JavaScript, можуть бути корисними для розробки вебінтерфейсу онлайн поліклініки.

По-шосте, масштабованість та гнучкість. Всі мови програмування мають свої особливості, які полегшують масштабування та розширення системи, наприклад, підтримку мікросервісної архітектури.

Дослідження сучасних інструментів для розроблення онлайн поліклініки може включати огляд різних технологій та платформ, що можуть бути використані для створення вебзастосунків.

Огляд різних вебфреймворків, таких як ASP.NET Core, Django, Flask, а також фронтендні фреймворки, такі як React, Angular та Vue.js, які можуть бути використані для створення клієнтської частини онлайн поліклініки. Порівняння різних вебфреймворків наведено в таблиці 1.1

Таблиця 1.1

Порівняння вебфреймворків

Критерій	ASP.NET Core	Django	Flask	React	Angular	Vue.js
Мова	C#	Python	Python	JavaScript	TypeScript	JavaScript
Тип	Фреймворк	Фреймворк	Мікрофреймворк	Бібліотека	Фреймворк	JavaScript бібліотека
Архітектура	Модель-Вигляд–Контролер (MVC)	Модель-Вигляд–Контролер (MVC)	Модель-Вигляд-Контролер (MVC)	Компонентно-орієнтована	Компонентно-орієнтована	Компонентно-орієнтована
Швидкість	Висока	Висока	Висока	Висока	Висока	Висока
Простота використання	Висока	Середня	Висока	Висока	Середня	Висока
Підтримка	Microsoft активна спільнота	Django, активна спільнота	Flask, активна спільнота	Facebook, активна спільнота	Google, активна спільнота	Активна спільнота
Розширюваність	Висока	Висока	Висока	Висока	Висока	Висока
Інтеграція з іншими технологіями	Висока	Висока	Висока	Середня	Висока	Висока
Підтримка мобільних платформ	Так	Ні	Ні	Так	Так	Так
Управління станом	Вбудоване	Стандартна	Стандартна	Зовнішній стан	Вбудоване	Зовнішній стан

Дослідження різних систем керування базами даних (СКБД) та їх можливостей для зберігання та обробки медичної інформації.

Реляційність даних:

- Microsoft SQL Server – це реляційна база даних, яка спеціалізується на зберіганні структурованих даних у вигляді таблиць.
- PostgreSQL та MySQL також реляційні СКБД, що підтримують структуровані дані та SQL-запити.
- MongoDB, у відміну від Microsoft SQL Server, є нереляційною базою даних, що зберігає дані у форматі документів, а не таблиць.

Масштабованість:

- Microsoft SQL Server зазвичай масштабується вертикально та горизонтально за допомогою різних рішень, таких як реплікація та кластеризація.
- PostgreSQL та MySQL також мають можливості масштабування, але можуть потребувати додаткових налаштувань для горизонтального масштабування.
- MongoDB відомий своєю здатністю горизонтально масштабуватися без значних зусиль завдяки географічній реплікації та розділенню даних на шарди.

Надійність та відмовостійкість:

- Microsoft SQL Server має вбудовані засоби резервного копіювання та відновлення для забезпечення високої надійності.
- PostgreSQL та MySQL також мають засоби для резервного копіювання та відновлення даних, але можливості можуть відрізнятися в залежності від конкретних конфігурацій.
- MongoDB зазвичай більш стійкий до відмов завдяки географічній реплікації та автоматичному відновленню.

Інструменти аналізу:

- Microsoft SQL Server має розвинуті інструменти аналізу даних, включаючи служби OLAP і інтеграцію з BI-системами.
- PostgreSQL та MySQL мають обмежену підтримку аналітики даних порівняно з Microsoft SQL Server.
- MongoDB підтримує базовий рівень агрегації та аналізу даних, але не

має таких розширених можливостей, як Microsoft SQL Server.

Сумісність та розповсюдженість:

- Microsoft SQL Server працює на платформі Windows, але також має версії для Linux. Широко використовується у корпоративному середовищі.
- PostgreSQL та MySQL можна встановлювати на різних операційних системах, включаючи Windows, Linux та macOS. Використовується як у корпоративному, так і у стартап-середовищі.
- MongoDB також доступний для різних операційних систем і використовується часто для стартапів та проектів великих даних.

Це дозволяє отримати загальну картину про можливості та властивості Microsoft SQL Server порівняно з іншими популярними СКБД.

Огляд хмарних платформ, таких як Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP), які можуть забезпечити інфраструктуру для розгортання та масштабування онлайн поліклініки.

Розгляд можливостей розробки мобільних додатків для пацієнтів та медичного персоналу, використовуючи популярні платформи, такі як iOS та Android, а також фреймворки, такі як Flutter або React Native.

Розгляд можливостей інтеграції онлайн поліклініки з іншими медичними системами, такими як системи керування лікарською документацією (СКЛД), лабораторні інформаційні системи (ЛІС) або системи електронної медичної картки (ЕМК).

1.4 Висновки до першого розділу. Постановка задач дослідження

У ході дослідження сучасних інструментів для розроблення вебзастосунку "Поліклініка", огляду існуючого програмного забезпечення для його роботи та опису процесів роботи цього вебзастосунку, стало очевидним, що вебзастосунки стають невід'ємною складовою сучасної медичної системи. Вони відіграють ключову роль у поліпшенні доступу до медичних послуг,

оптимізації процесів прийому пацієнтів, зберіганні та обробці медичної інформації.

Розглянуті інструменти розробки та програмне забезпечення, такі як різноманітні фреймворки та системи керування базами даних, надають широкі можливості для створення потужних та ефективних вебзастосунків для поліклінік. Їхня функціональність дозволяє автоматизувати багато процесів, що спрощує роботу медичного персоналу та підвищує рівень обслуговування пацієнтів.

Опис процесів роботи вебзастосунку "Поліклініка" дозволяє краще зрозуміти, як він функціонує та які конкретно завдання він виконує. Це допомагає розробникам краще налаштувати та оптимізувати вебзастосунок для відповідності потребам конкретної поліклініки та її пацієнтів.

У висновку, вебзастосунок "Поліклініка" є важливим інструментом для сучасної медичної системи, який сприяє покращенню якості та доступності медичних послуг. Його розвиток та впровадження можуть виявитися ключовими для оптимізації роботи поліклінік та задоволення потреб їхніх пацієнтів.

Постановка задачі для подальшого дослідження вебзастосунку поліклініки може включати наступні аспекти:

1) Аналіз потреб поліклінік у вебзастосунках: Визначення основних потреб та вимог медичних установ до функціоналу та можливостей вебзастосунків, оцінка важливості різних функцій та можливостей для оптимізації роботи.

2) Вибір оптимальних інструментів розробки: Дослідження та порівняння різних фреймворків, мов програмування, систем керування базами даних та інших технологій для створення вебзастосунку, з урахуванням вимог до швидкості, масштабованості, безпеки та інші параметри.

3) Розробка функціоналу вебзастосунку: Визначення та розробка ключових функцій, які має включати вебзастосунок для задоволення потреб поліклінік та їхніх клієнтів, включаючи прийом пацієнтів, управління

медичними записами, розклад лікарів та інше.

4) Тестування та впровадження: Проведення тестування вебзастосунку для перевірки його працездатності, безпеки та коректності роботи, а також впровадження його в реальному середовищі поліклініки з належною підготовкою персоналу та користувачів.

5) Моніторинг та підтримка: Забезпечення постійного моніторингу та підтримки вебзастосунку для виявлення та виправлення можливих проблем, а також розширення його функціоналу відповідно до змінних потреб поліклінік та технологічних можливостей.

РОЗДІЛ 2 АНАЛІЗ ЗАСОБІВ РОЗРОБЛЕННЯ ВЕБЗАСТОСУНКУ ПОЛІКЛІНІКА

2.1 Вибір програмних засобів для розроблення вебзастосунку поліклініка

Під час розробки програмного забезпечення можуть бути використані наступні програмні засоби:

1) Мова програмування C#, можливості якої роблять її потужним інструментом для розробки вебзастосунків;

2) ASP.NET, яка надає зручні інструменти для створення різноманітних вебрішень;

3) ADO.NET, що використовується для забезпечення доступу до даних з різних джерел даних у .NET-середовищі;

4) MS SQLServer, що надає широкий спектр можливостей для зберігання, обробки та управління даними у корпоративних середовищах.

2.2 Мова програмування C#

Мова програмування C# – це потужний і високорівневий інструмент програмування, розроблений корпорацією Microsoft. Вона була випущена в 2000 році разом з платформою .NET Framework і з тих пір стала однією з найпопулярніших мов для розробки різноманітних програм, включаючи вебзастосунки, настільні програми, мобільні додатки та ігри.

Із-за підтримки об'єктно-орієнтованого програмування спочатку мова мала назву Cool, що було абревіатурою від «C-like Object Oriented Language». Проте, для запобігання патентним проблемам, назву було змінено.

Назва «C#» була вибрана як еволюційний крок від мов програмування C++, підкреслюючи схожість у синтаксисі та основних принципах мови, а також покращення в порівнянні з попередником.

Семантика оператора #: Символ # (хеш-символ) в програмуванні часто використовується для позначення директив препроцесора або метаданих. Таким чином, додавання символу # в назву мови може вказувати на важливість метаданих та препроцесингу в мові.

Отже, назва «C#» відображає спадщину від мови C++, акцентуючи одночасно схожість і вдосконалення, а також вказує на важливість метаданих та препроцесора в мові.

C# підтримує об'єктно-орієнтоване програмування (ООП), що дозволяє розробникам створювати програми з використанням об'єктів, які взаємодіють один з одним. Ось декілька ключових понять, пов'язаних з ООП, які C# підтримує:

– можна описувати класи, які визначають структуру та поведінку об'єктів, клас є шаблоном для створення об'єктів, які представляють конкретні екземпляри даних;

– інкапсуляція дозволяє об'єднати дані та методи, що працюють з цими даними, в одному класі, і забезпечує контроль доступу до них;

– наслідування дозволяє створювати нові класи на основі існуючих, що успадковують всі їхні властивості та методи і дозволяє уникати дублювання коду, полегшує розширення та підтримку коду;

– поліморфізм означає, що різні об'єкти можуть відповідати на один і той же запит різними способами, що дає можливість створювати загальні методи або інтерфейси, які можна використовувати для роботи з різними типами об'єктів;

– абстракція дозволяє виділяти головні характеристики об'єкта, ігноруючи деталі його реалізації, що спрощує розробку, тестування та зміну коду.

ООП широко використовується у веброзробці, в тому числі і за допомогою мови програмування C# [1–5].

ООП дозволяє створювати програми з окремими модулями або класами, які можна перевикористовувати в різних частинах вебдодатка. Це полегшує

розробку, тестування та підтримку вебдодатків.

Через використання наслідування та поліморфізму, ООП дозволяє легко розширювати функціональність вебдодатків без необхідності модифікувати вже існуючий код.

Багато популярних вебфреймворків, таких як ASP.NET Core, базуються на об'єктно-орієнтованих принципах. Використання мови програмування, яка підтримує ООП, спрощує розробку вебдодатків у таких фреймворках.

ООП дозволяє розбивати складні вебдодатки на менші, керовані класами компоненти, що полегшує розробку та підтримку великих проектів.

Використання класів і об'єктів дозволяє реалізувати спадкування та абстракцію, що робить код більш читабельним і підтримуваним.

Крім ООП, C# також підтримує інші парадигми програмування, такі як структурне програмування, функціональне програмування та асинхронне програмування.

Веброзробка часто включає в себе елементи різних парадигм програмування.

Структурне програмування фокусується на використанні послідовних структур програмування, таких як умови, цикли та функції, для розв'язання завдань. У веброзробці структурне програмування може бути використане для організації логіки бізнес-логіки на серверній стороні або управління різними елементами вебінтерфейсу на клієнтській стороні.

Функціональне програмування зосереджено на використанні функцій як основного будівельного блоку програм. У веброзробці функціональне програмування може виявитися корисним при використанні функцій вищих порядків, замикань та імутабельних структур даних для обробки даних на клієнтській або серверній стороні.

У веброзробці асинхронність грає важливу роль у взаємодії з сервером. Запити до сервера, обробка файлів, взаємодія з базами даних - все це може виконуватися асинхронно для забезпечення кращої продуктивності та реагування на вебдодатки. У мовах програмування, таких як JavaScript на

клієнтській стороні і C# на серверній стороні, існують засоби асинхронного програмування для роботи з асинхронними операціями.

C# є основною мовою розробки для платформи .NET Framework, що відкриває доступ до широкого спектру бібліотек і фреймворків для розробки різноманітних застосунків.

Безпечність типів в мові програмування C# означає, що вона має ретельно визначену систему типів, яка допомагає уникати багатьох типових помилок під час розробки програм.

C# є статично типізованою мовою програмування, що означає, що типи змінних визначаються на етапі компіляції і не можуть змінюватися під час виконання програми. Це допомагає виявляти баги під час розробки ще до запуску програми.

Сильна типізація в C# що означає, що типи даних строго визначені і система не дозволяє неявних приведень типів даних, які можуть призводити до потенційних помилок у програмі.

Система типів C# підтримує поліморфізм та наслідування, що дозволяє створювати ієрархії класів і використовувати специфічні для класів методи та властивості. Це сприяє створенню коду, який є більш структурованим і легким для розуміння, що може підвищити безпеку програми.

Перевірка типів на етапі компіляції і виконання: C# виконує перевірку типів як на етапі компіляції, так і під час виконання програми. Це дозволяє виявляти і вирішувати потенційні проблеми типів ще до того, як вони виникнуть під час роботи програми.

Усі ці аспекти допомагають забезпечити безпеку типів в мові програмування C# і роблять її більш надійною для розробки програм, особливо великих та складних проектів.

Програми, написані на C#, можуть бути запущені на різних операційних системах, таких як Windows, macOS та Linux.

Це стало можливим завдяки платформі .NET, яка є середовищем виконання для мови програмування C#.

Ось декілька ключових аспектів, які дозволяють програмам на C# бути переносимими між різними операційними системами.

.NET Core є крос-платформеною реалізацією платформи .NET, яка підтримує різні операційні системи, включаючи Windows, macOS та Linux. Це дозволяє розробникам створювати програми на C# і запускати їх на будь-якій підтримуваній платформі без змін у вихідному коді.

Сама мова програмування C# є крос-платформеною, що означає, що синтаксис та основні функції мови не залежать від конкретної операційної системи. Це дозволяє писати код, який легко переносити між різними платформами.

При запуску програми на C# на операційній системі, вона виконується на віртуальній машині .NET (Common Language Runtime - CLR). CLR надає інтерфейс між програмою та операційною системою, що дозволяє запускати програми на різних платформах без необхідності перекомпіляції вихідного коду.

Багато бібліотек .NET, які доступні для розробників, також є крос-платформеними, що дозволяє використовувати їх у програмах на C# на будь-якій підтримуваній операційній системі.

Отже, завдяки цим функціям та інструментам програми на C# можуть бути розроблені і запущені на різних операційних системах без значних змін у вихідному коді.

Синтаксис C# схожий на C++ та Java, що робить його зрозумілим для програмістів, які вже володіють цими мовами.

Масштабованість і продуктивність C# в вебзастосунках є ключовими для забезпечення успіху проекту та задоволення потреб користувачів. Ось кілька причин, чому ці характеристики є важливими.

У вебзастосунках часто виникає необхідність обробки великих обсягів даних, таких як інформація користувачів, транзакції, звіти тощо. Масштабованість C# дозволяє ефективно управляти цими даними та забезпечувати швидкий доступ до них.

Вебзастосунки можуть мати велику кількість одночасних користувачів, які взаємодіють з системою одночасно. Хороша масштабованість дозволяє системі ефективно обробляти багатокористувацькі запити і забезпечувати стабільну продуктивність навіть під великим навантаженням.

Висока продуктивність C# дозволяє створювати вебзастосунки, які працюють швидко та миттєво реагують на запити користувачів. Це важливо для забезпечення позитивного досвіду взаємодії з користувачами і збереження їхньої уваги.

C# дозволяє розробникам легко розширювати та модернізувати функціональність вебзастосунків відповідно до зростаючих потреб бізнесу та користувачів. Це дозволяє створювати гнучкі та адаптивні рішення, які можуть змінюватися відповідно до потреб ринку.

Отже, масштабованість і продуктивність C# важливі для створення надійних, швидких та гнучких вебзастосунків, які забезпечують задоволення потреб користувачів та успіх бізнесу.

2.3 ASP.NET

ASP.NET – це технологія, яка використовується для створення динамічних вебсайтів та вебдодатків. Вона базується на платформі .NET Framework (або .NET Core в новіших версіях) і забезпечує зручний спосіб розробки вебдодатків за допомогою мов програмування різних, таких як C#, VB.NET, F# та інших. Це дає можливість розробникам вибирати мову, з якою вони найбільш комфортно відчувають себе і яку найбільш добре знають.

ASP.NET надає високорівневі абстракції та компоненти для швидкої розробки вебдодатків. Це включає в себе вбудовані компоненти для роботи з базами даних, обробки форм, автентифікації та авторизації, кешування, а також можливості створення вебслужб і вебсерверів.

Вебфреймворк ASP.NET має велику кількість вбудованих шаблонів сторінок та майстрів, які дозволяють швидко створювати різноманітні

сторінки і компоненти вебінтерфейсу. Наприклад, є шаблони для створення форм, таблиць, списків, а також можливість створення власних шаблонів [6–9].

Архітектура ASP.NET базується на модульності, що означає, що функціональність вебдодатків розділена на невеликі, незалежні модулі або компоненти. Це дозволяє розробникам використовувати готові компоненти і бібліотеки, що збільшує швидкість розробки, полегшує супровід і сприяє перевикористанню коду.

До основних переваг модульної архітектури ASP.NET можна віднести перевикористання коду. Розробники можуть створювати компоненти, які можна використовувати у різних проектах, що полегшує розробку і зменшує кількість дублювання коду.

Готові модулі дозволяють швидко створювати функціональність, не потрібно розробляти її з нуля. Це особливо корисно для стандартних завдань, таких як аутентифікація користувачів, керування сесіями тощо.

Модульна архітектура дозволяє легко впроваджувати зміни в окремі компоненти без впливу на інші частини системи. Це полегшує супровід і розширення вебдодатків.

Завдяки модульності можна легко додавати новий функціонал або змінювати існуючий, не ламаючи всю систему.

Модульна архітектура сприяє проведенню модульних тестів, оскільки окремі компоненти можна тестувати незалежно від інших частин додатку.

Платформа розробки вебдодатків ASP.NET ASP.NET має вбудовані механізми та інструменти, що дозволяють забезпечити високий рівень безпеки для вебдодатків.

Фреймворк надає гнучкі засоби для аутентифікації користувачів і контролю доступу до різних ресурсів. Це включає вбудовані засоби для роботи з різними методами аутентифікації (такими як форма, Windows, ідентифікація на основі ролей тощо) і можливість налаштування дозволів на рівні контролів або сторінок.

Технологія веброзробки ASP.NET має вбудовані захисти від різних типів атак, таких як SQL-ін'єкції, міжсайтовий скриптінг (XSS), міжсайтова поділка даних (XSRF), перехоплення сесій тощо. Наприклад, ASP.NET за замовчуванням використовує параметризовані запити для запобігання SQL-ін'єкціям та автоматично обробляє вхідні дані для захисту від XSS.

ASP.NET надає можливості для валідації вхідних даних користувачів перед їх обробкою. Це включає вбудовані засоби для валідації форми на клієнтському та серверному боці, а також можливість створення власних правил валідації.

Вебфреймворк підтримує різні методи шифрування для захисту конфіденційних даних, таких як паролі, сесійні ключі, куки тощо. Це допомагає уникнути витоку конфіденційної інформації під час передачі по мережі.

Моніторинг та журналювання: ASP.NET надає можливості для моніторингу та журналювання подій вебдодатків. Це дозволяє виявляти аномалії в роботі додатку, відслідковувати дії користувачів та швидко реагувати на потенційні проблеми безпеки.

2.4 ADO.NET

ADO.NET (ActiveX Data Objects for .NET) – це набір технологій, що використовується в платформі .NET для доступу до даних з різних джерел даних, таких як бази даних, XML-файли, об'єктні моделі даних тощо. ADO.NET надає програмістам можливість створювати, зчитувати, оновлювати та видаляти дані з різних джерел даних у .NET-середовищі.

До основних компонентів ADO.NET можна віднести наступні.

Провайдери даних (Data Providers). ADO.NET надає різні провайдери даних для взаємодії з різними джерелами даних. Наприклад, для роботи з Microsoft SQL Server використовується провайдер System.Data.SqlClient, для роботи з Oracle – System.Data.OracleClient, для роботи з MySQL –

MySQL.Data.MySqlClient тощо [10,11].

DataSet і DataTable. DataSet є набором даних, які можуть бути зчитані з бази даних і збережені у вигляді таблиць та взаємозв'язків між ними. DataTable представляє собою одну таблицю даних у пам'яті. DataSet може містити одну або більше таблиць DataTable.

DataAdapter. Додатковий клас, який використовується для зчитування даних з бази даних у DataSet або DataTable та збереження змін у джерелі даних. DataAdapter виконує операції взаємодії з базою даних, такі як вибірка (select), вставка (insert), оновлення (update) та видалення (delete).

Command. Об'єкт, який представляє SQL-запит до бази даних. Він використовується для виконання команд SQL, таких як SELECT, INSERT, UPDATE та DELETE.

Connection. Об'єкт, який представляє з'єднання з джерелом даних, таким як база даних. Він використовується для встановлення з'єднання з базою даних перед виконанням команд SQL.

ADO.NET використовується для забезпечення доступу до даних з різних джерел даних у .NET-середовищі, таких як реляційні бази даних, XML-файли, об'єктні моделі даних та інші. Він дозволяє програмістам створювати, зчитувати, оновлювати та видаляти дані з різних джерел даних за допомогою різноманітних компонентів, таких як провайдери даних, DataSet, DataTable, DataAdapter і т. д.

До основних цілей використання ADO.NET включають:

– забезпечення можливості отримання доступу до даних з різних джерел, щоб програми могли взаємодіяти з реляційними базами даних, XML-файлами та іншими джерелами даних;

– дозвіл на зчитування, запис, оновлення та видалення даних з баз даних або інших джерел даних з використанням SQL-запитів або інших методів доступу;

– забезпечення можливості встановлення та управління з'єднаннями з базами даних для виконання операцій з даними;

– надання можливості обробки даних в пам'яті за допомогою DataSet і DataTable, що дозволяє працювати з даними у вигляді наборів таблиць та взаємозв'язків між ними;

– забезпечення можливості забезпечення безпеки даних шляхом використання параметризованих запитів, валідації даних та інших заходів безпеки.

У загальному, ADO.NET дозволяє програмістам легко взаємодіяти з різними джерелами даних та ефективно управляти даними у вебдодатках.

2.5 MSSQLServer

Microsoft SQL Server (MSSQLServer) – це система управління базами даних (СУБД), розроблена компанією Microsoft. Вона надає широкий спектр можливостей для зберігання, обробки та управління даними у корпоративних середовищах. MSSQLServer дозволяє створювати, зберігати, запитувати, оновлювати та видаляти дані, а також забезпечує різноманітні функції для забезпечення безпеки, резервного копіювання та відновлення даних.

MSSQLServer підтримує роботу з великими обсягами даних і високими навантаженнями, що робить його ідеальним вибором для підприємств з різноманітними потребами у зберіганні та обробці даних.

SQLServer дозволяє налаштовувати різні методи автентифікації користувачів, такі як Windows аутентифікація або аутентифікація SQL Server. Це дозволяє контролювати, як користувачі отримують доступ до бази даних [12–16].

Сервер баз даних надає можливість встановлювати деталізовані права доступу до об'єктів бази даних, таких як таблиці, представлення, процедури та інші об'єкти. Це дозволяє обмежувати доступ користувачів до конкретних даних і функцій системи.

SQL Server підтримує різні методи шифрування даних, включаючи рівень бази даних, рівень стовпця та рівень додатку. Це забезпечує безпеку

конфіденційної інформації, зберіганої в базі даних, і захищає її від несанкціонованого доступу.

SQL сервер від Microsoft дозволяє ведення журналу аудиту, за допомогою якого можна відстежувати та аналізувати дії користувачів в базі даних. Це дозволяє виявляти потенційні загрози безпеці та вживати відповідних заходів для їх запобігання.

Набір функцій SQL Server може бути досить широким і включати:

1) мову запитів Transact-SQL (T-SQL): SQL Server надає мову програмування T-SQL для взаємодії з базою даних. T-SQL містить різноманітні оператори для роботи з даними, такі як SELECT, INSERT, UPDATE, DELETE, а також управління структурою бази даних, таке як CREATE, ALTER, DROP;

2) функції для управління даними, включаючи можливості індексації, транзакцій, реплікації, а також засоби резервного копіювання та відновлення даних;.

3) набір засобів безпеки для захисту даних, включаючи рівні автентифікації, авторизації, шифрування даних, аудиту та моніторингу;

4) можливості для виконання аналітичних запитів та операцій, таких як OLAP (Online Analytical Processing) та інтеграція з аналітичними інструментами, наприклад, засобами Business Intelligence;

5) інтеграцію з різними технологіями та іншими продуктами Microsoft, такими як Visual Studio, .NET Framework, Azure, Excel і багато іншого, що розширює можливості розробки та використання баз даних.

Це лише деякі з основних функцій, які надає SQL Server. В залежності від версії і конкретних потреб можуть бути доступні додаткові функції та можливості.

SQL Server надає розширений набір інструментів для аналізу даних та створення звітів, що дозволяє організаціям отримувати цінні інсайти і приймати обґрунтовані рішення.

Сервер баз даних підтримує OLAP-технології (Online Analytical

Processing), які дозволяють проводити аналіз даних у режимі реального часу та виконувати складні аналітичні операції, такі як куби даних, зрізи, динамічні таблиці та інші.

SQL Server інтегрується з інструментами BI, такими як Microsoft Power BI, що дозволяє створювати динамічні та інтерактивні звіти, візуалізації даних, аналітику та інші ресурси для аналізу та вивчення даних.

Система управління базами даних Microsoft SQL має вбудовані інструменти для створення звітів, такі як SQL Server Reporting Services (SSRS). SSRS надає можливості для створення різноманітних типів звітів, включаючи табличні звіти, графіки, діаграми, карти та інші.

Сервер баз даних може інтегруватися з Microsoft Excel, що дозволяє ефективно використовувати дані з бази даних для створення різноманітних звітів, діаграм, підсумкових таблиць та іншого аналітичного змісту.

SQL Server має різноманітні аналітичні функції, які дозволяють виконувати складний аналіз даних, включаючи прогнозування, кластеризацію, регресійний аналіз та інші.

Ці засоби дозволяють користувачам створювати різноманітні звіти та аналітичні дашборди для ефективного аналізу та використання даних у бізнес-середовищі.

Як видно, SQL Server може інтегруватися з різними технологіями та іншими продуктами Microsoft, такими як Visual Studio, .NET Framework, Azure, Excel і багато іншого, що розширює можливості розробки та використання баз даних.

2.6 Висновок до другого розділу

У цьому розділі було розглянуто вибір програмних засобів для розроблення вебзастосунку поліклініка, а також можливості мови програмування C#, фреймворку ASP.NET, технологій ADO.NET та MSSQLServer для розробки бази даних.

Вибір правильних програмних засобів є критичним етапом у процесі розробки будь-якого вебзастосунку, зокрема для поліклініки, де потрібна висока надійність, швидкість та безпека обробки медичної інформації. Вибір мови програмування C# та фреймворку ASP.NET забезпечує широкі можливості для розробки потужних та масштабованих вебзастосунків, а технології ADO.NET та MSSQLServer забезпечують ефективне управління та обробку бази даних з високою продуктивністю та безпекою.

Ці програмні засоби дозволяють розробникам створювати високоякісний вебзастосунок поліклініки, який відповідає всім потребам управління медичними даними та надає зручний і безпечний інтерфейс для користувачів. Вибір цих засобів є ключовим для успішної розробки та експлуатації вебзастосунку поліклініки.

РОЗДІЛ 3 РОЗРОБЛЕННЯ ВЕБЗАСТОСУНКУ ПОЛІКЛІНІКА

3.1 Опис бази даних

Розробка бази даних включає створення логічної та фізичної моделей. Логічна модель описує сутності предметної області в узагальненому табличному форматі з урахуванням нормалізації даних і не пов'язана з конкретними програмними засобами. Цю модель можна представити різними способами, найбільш інформативним є повна атрибутивна модель. Фізична модель, у свою чергу, це схема бази даних, адаптована до обраної системи управління базами даних. Так як вихідна інформація системи автоматизованої інформаційної системи формується на основі запитів користувача до бази даних, то сама інформація не зберігається в базі даних.

Побудова повної атрибутивної моделі бази даних означає відображення таблиць бази даних у вигляді табличної структури, де кожний рядок містить інформацію про атрибути конкретної сутності. Ця інформація включає природні назви атрибутів, їх синтетичні аналоги, а також тип даних та розмір атрибута. У таблиці також виділяються первинні та зовнішні ключі, і кожна таблиця отримує природну назву, а в дужках записується назва таблиці в базі даних. Наприклад, база даних поліклініки складається з трьох таблиць, які представлені у таблицях 3.1 – 3.3.

Таблиця 3.1

Логічна структура таблиці БД «Лікарі» (Doc)

Код лікаря	Прізвище	Ім’я	По батькові	Спеціальність	Категорія
Doc_id	Surname	Name	Middle_name	Specialty	Category
I4	C20	C20	C20	C20	C5

Таблиця 3.2

Логічна структура таблиці БД «Пацієнти» (Patients)

Код пацієнта	Прізвище	Ім'я	По батькові	Стать	Рік народження
Patient_id	Surname	Name	Middle_name	Gender	Year_of_birth
I4	C20	C20	C20	C9	I4

Таблиця 3.3

Логічна структура таблиці даних «Звернення» (Referral)

Код звернення	Код лікаря	Код пацієнта	Дата звернення	Діагноз	Вартість лікування
Referral_id	Doc_id	Patient_id	Date_of_referral	Diagnosis	Cost_of_ treatment
I4	I4	I4	D8	C30	F15

Фізична модель даних включає в себе опис об'єктів предметної області в контексті конкретних можливостей обраної системи управління базами даних (СУБД). В цій моделі описується всю інформацію про фізичні об'єкти – таблиці, колонки, індекси та збережені процедури. Фізична структура бази даних, розроблена в рамках СУБД MSSQLServer, наведена на рисунку 3.1 у вигляді діаграми бази даних.

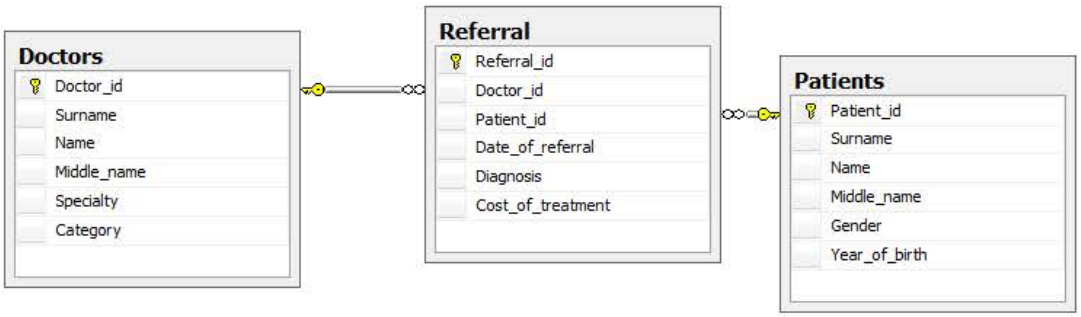

Рисунок 3.1 – Діаграма бази даних

3.2 Запити до бази даних

Запити SQL використовуються для формування звітної та вихідної інформації у вебдодатку. Завдяки SQL-запитам як лікар, так і пацієнт можуть переглядати інформацію, враховуючи вхідні параметри запитів. SQL-запити для відображення інформації:

1. SQL-запит для відбору усіх записів зі списку звернень:

   SELECT Referral_id, Doc_id, Patient_id, Date_of_referral, Diagnosis, Cost_of_treatment FROM Referral

   Звернення можна шукати:

   – за кодом

   SELECT Referral_id, Doc_id, Patient_id, Date_of_referral, Diagnosis, Cost_of_treatment FROM Referral WHERE Referral_id = @Referral_id

   – за датою звернення

   SELECT Referral_id, Doc_id, Patient_id, Date_of_referral, Diagnosis, Cost_of_treatment FROM Referral WHERE Date_of_referral = @Date_of_referral

   – за діагнозом

   SELECT Referral_id, Doc_id, Patient_id, Date_of_referral, Diagnosis, Cost_of_treatment FROM Referral WHERE Diagnosis = @Diagnosis

2. SQL-запит для відбору усіх записів зі списку лікарів:

   SELECT Doc_id, Surname, Name, Middle_name, Specialty, Category FROM Doc

   Лікарів можна шукати:

– за кодом

   SELECT Doc_id, Surname, Name, Middle_name, Specialty, Category FROM Doc WHERE Doc_id = @Doc_id

– за прізвищем

   SELECT Doc_id, Surname, Name, Middle_name, Specialty, Category FROM Doc WHERE Surname = @Surname

– за спеціальністю

SELECT Doc_id, Surname, Name, Middle_name, Specialty, Category FROM Doc WHERE Specialty = @Specialty

3. SQL-запит для відбору усіх записів зі списку пацієнтів:

SELECT Patient_id, Surname, Name, Middle_name, Gender, Year_of_birth FROM Patients

Пацієнтів можна шукати:

– за кодом

SELECT Patient_id, Surname, Name, Middle_name, Gender, Year_of_birth FROM Patients WHERE Patient_id = @Patient_id

– за прізвищем

SELECT Patient_id, Surname, Name, Middle_name, Gender, Year_of_birth FROM Patients WHERE Surname = @Surname

3.3 Інтерфейсна частина вебзастосунку

Інтерфейс користувача (UI – user interface) – це сукупність інструментів, які дозволяють користувачу взаємодіяти з системою. Він включає всі елементи та компоненти програми, які можуть впливати на спосіб взаємодії користувача з програмним забезпеченням.

Будь-який вебзастосунок складається з набору взаємопов'язаних сторінок. Якісно розроблений вебзастосунок має не лише привабливий та зручний дизайн, але й ефективну систему навігації, що дозволяє легко переміщатися між різними розділами та виконувати інші завдання. ASP.NET має широкі можливості для реалізації складних систем навігації.

Карти сайту використовуються для структуризації великої кількості сторінок вебзастосунку. Вони надають зручний механізм для визначення структури та її відображення за допомогою декількох елементів управління. Ці елементи управління доступні в розділі "Панель інструментів" під назвою "Навігація".

У роботі було використано елемент управління TreeView. Він відображає дані у вигляді дерева, дозволяючи розгортати та згортати вузли. При цьому можна обробляти відповідні події.

При оголошенні елемента управління TreeView на сторінці, вузли описуються за допомогою тегів TreeNode. Вони можуть мати будь-який рівень вкладеності один в одного. Вузли можна також редагувати візуально. Програмний код для створення навігаційного елемента вебзастосунку зображено на рисунку 3.2.

```
<asp:TreeView ID="TreeView1" runat="server" OnSelectedNodeChanged="TreeView1_Sel
<Nodes>
<asp:TreeNode Text="Переглянути" Value="Головна сторінка">
<asp:TreeNode Text="Таблиця лікарі" Value="Таблиця лікарі"></asp:TreeNode>
<asp:TreeNode Text="Таблиця пацієнти" Value="Таблиця пацієнти"></asp:TreeNode>
<asp:TreeNode Text="Таблиця звернення" Value="Таблиця звернення"></asp:TreeNode>
</asp:TreeNode>
<asp:TreeNode Text="Редагувати" Value="Редагувати"></asp:TreeNode>
</Nodes>
</asp:TreeView>
```

Рисунок 3.2 – Створення навігаційного елемента

Результат використання елемента TreeView для створення карти вебзастосунку зображено на рисунку 3.3.

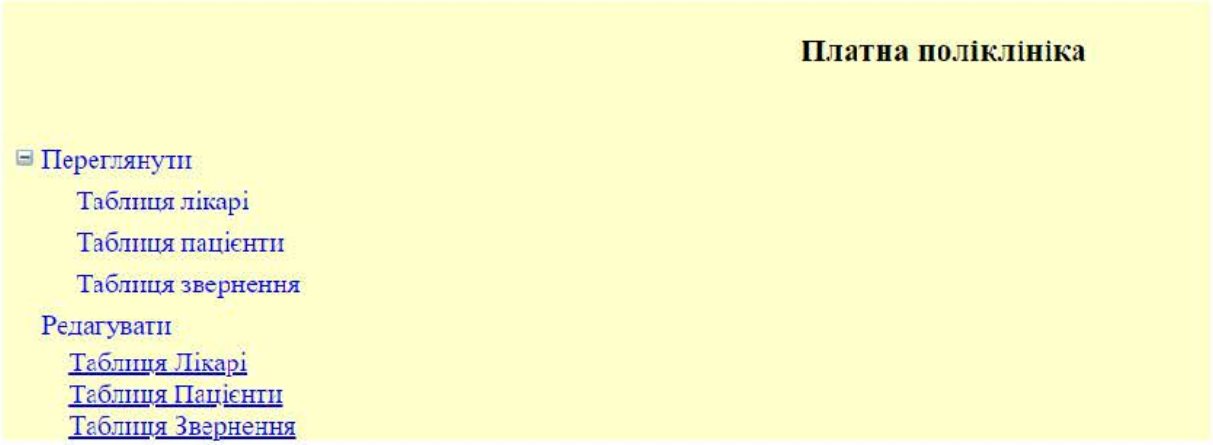

Рисунок 3.3 – Карта вебзастосунку

3.4 Стиль вебзастосунку

Для створення стилю вебзастосунку використовувалася каскадна таблиця стилів CSS. Каскадні таблиці стилів є основою для візуального оформлення вебзастосунків. Вони визначають зовнішній вигляд і форматування елементів HTML, таких як текст, фон, розміщення, колір і розмір, що дозволяє створювати стильовані та привабливі вебінтерфейси.

Основні принципи CSS:

- відділяє візуальне форматування вебсторінок від їх структури, що дозволяє змінювати вигляд сторінки, не змінюючи її структуру і вміст;
- дозволяє визначати стилі для конкретних елементів, а також унаслідувати стилі від батьківських елементів, що дозволяє створювати складні структури стилів та забезпечує можливість перевизначення стилів за необхідності;
- використовує селектори для вибору елементів, які потребують стилізації. Селектори можуть бути простими (наприклад, назви елементів або класи) або складними (наприклад, комбінація селекторів);
- стилі можна застосовувати до елементів HTML через внутрішній стиль (inline style), внутрішній CSS (embedded style) або зовнішній CSS файл (external style sheet);
- дозволяє налаштовувати стилі для різних мультимедійних пристроїв і розділів, таких як екран, принтер, або екран з високим роздільним здатністю (HD).

Використання каскадних таблиць стилів дозволяє розробникам створювати стильовані, зручні та привабливі вебінтерфейси, що поліпшують користувацький досвід і сприяють успіху вебзастосунків.

3.5 Мережевий вебінтерфейс

Вебформи призначені для спрощення процесу введення даних користувачем у базу даних. Вони повинні бути легко доступними та інтуїтивно зрозумілими, оскільки в іншому випадку втрачається їхня корисність. У таблиці 3.4 наведена структура інтерфейсної частини системи.

Таблиця 3.4

Перелік вебсторінок

Програмна назва сторінки	Опис функціонального призначення сторінки
Default.aspx	Головна сторінка. Перегляд списку основних можливостей.
Doc.aspx	Сторінка для редагування таблиці «Лікарі» у базі даних «Платна поліклініка»
Patients.aspx	Сторінка для редагування таблиці «Пацієнти» у базі даних «Платна поліклініка»
Referral.aspx	Сторінка для редагування таблиці «Звернення» у базі даних «Платна поліклініка»

Вебекрани програми показані на рисунках 3.4 – 3.7. Сторінка Default.aspx відображена на рисунку 3.4. На цій сторінці користувач може переглядати інформацію про всіх лікарів, пацієнтів та звернення, а також переходити на сторінки редагування таблиць.

Вебсторінка для редагування лікарів зображена на рисунку 3.5. Тут можна знайти лікаря за його кодом, прізвищем або спеціальністю. Також можна додати нового лікаря, або видалити чи відредагувати обраного.

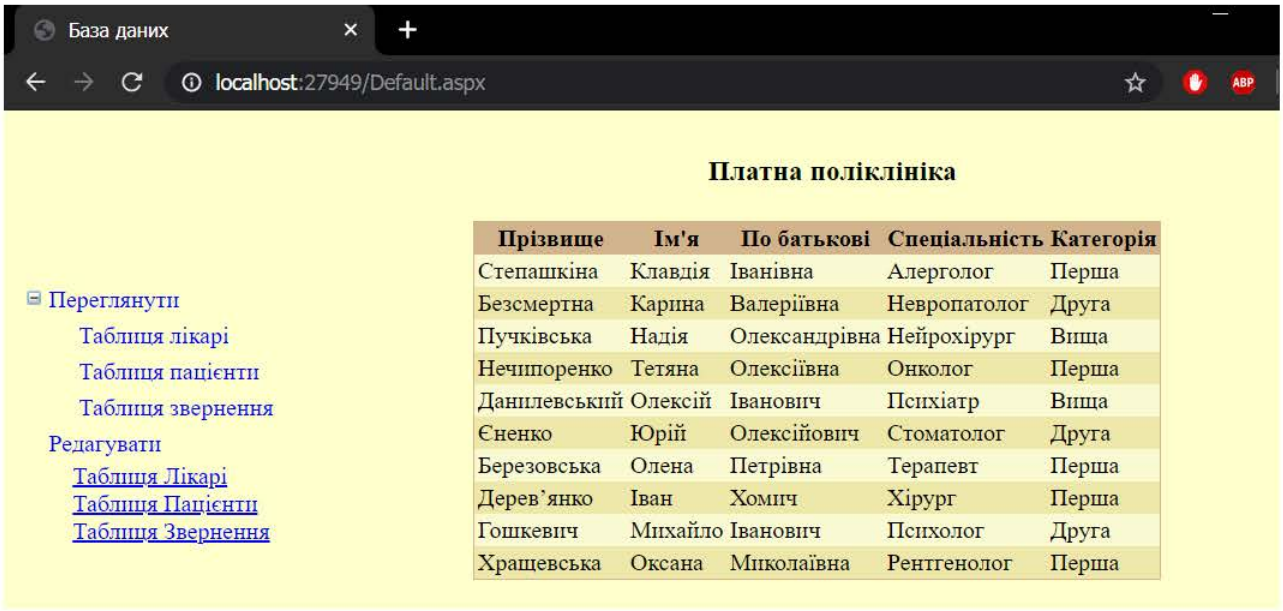

Рисунок 3.4 – Головна вебсторінка

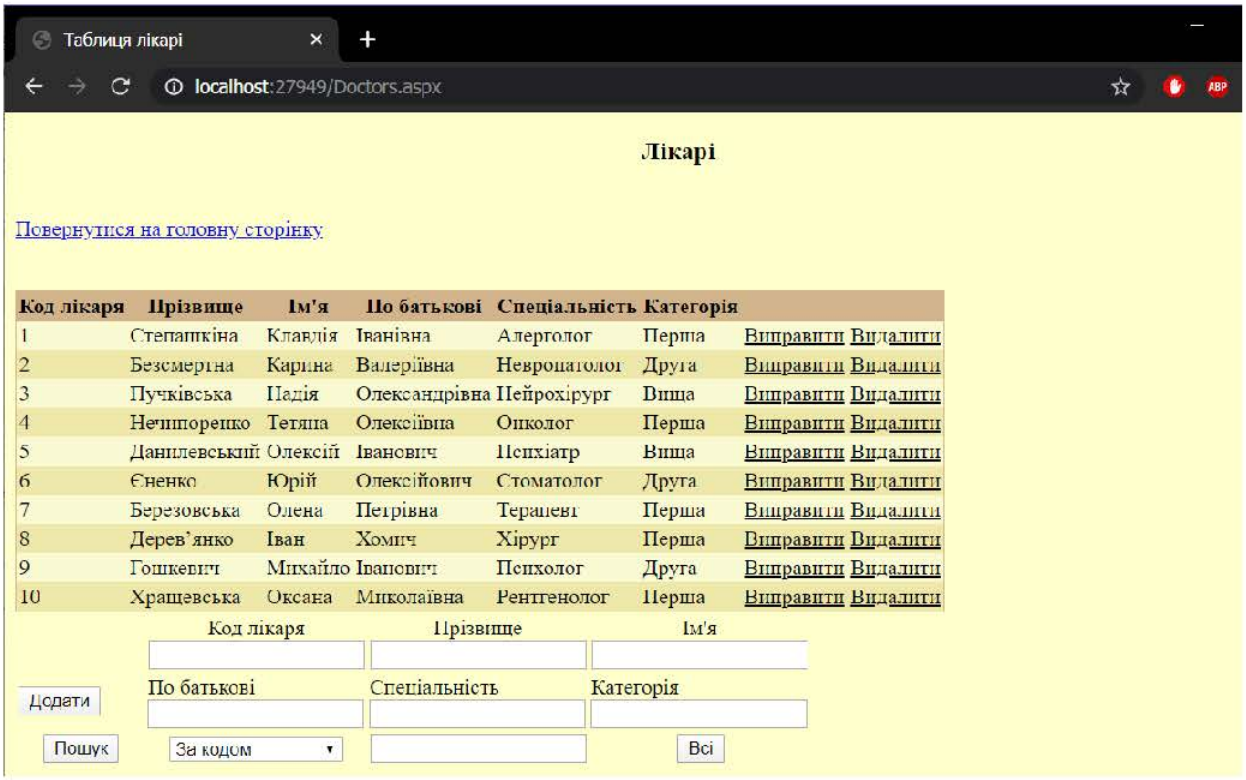

Рисунок 3.5 – Сторінка для редагування лікарів

Вебсторінка для редагування пацієнтів представлена на рисунку 3.6. Тут можна знайти пацієнта за його кодом або прізвищем. Також можна додати нового пацієнта, або видалити чи відредагувати обраного.

Вебсторінка для редагування звернень представлена на рисунку 3.7. Тут

можна знайти звернення за його кодом, діагнозом або по даті. Також можна додати нове звернення, або видалити чи відредагувати обране.

3.6 Опис програмних механізмів

У межах цього вебзастосунку було створено головну вебсторінку та сторінки для редагування таблиць "Лікарі", "Пацієнти" та "Звернення". Код головної сторінки наведений у додатку Б, у розділах: лістинг файлу Default.aspx та Default.aspx.cs. Код сторінки редагування лікарів подано у додатку Б, у розділах: лістинг файлу Doc.aspx та Doc.aspx.cs. Код сторінки редагування пацієнтів подано у додатку Б, у розділах: лістинг файлу Patients.aspx та Patients.aspx.cs. Код сторінки редагування звернень подано у додатку Б, у розділах: лістинг файлу Referral.aspx та Referral.aspx.cs.

Для перегляду, пошуку, видалення, запису та редагування даних використовувався SQLDataSource. Механізми сторінок вебзастосунку представлені в табл. 3.5.

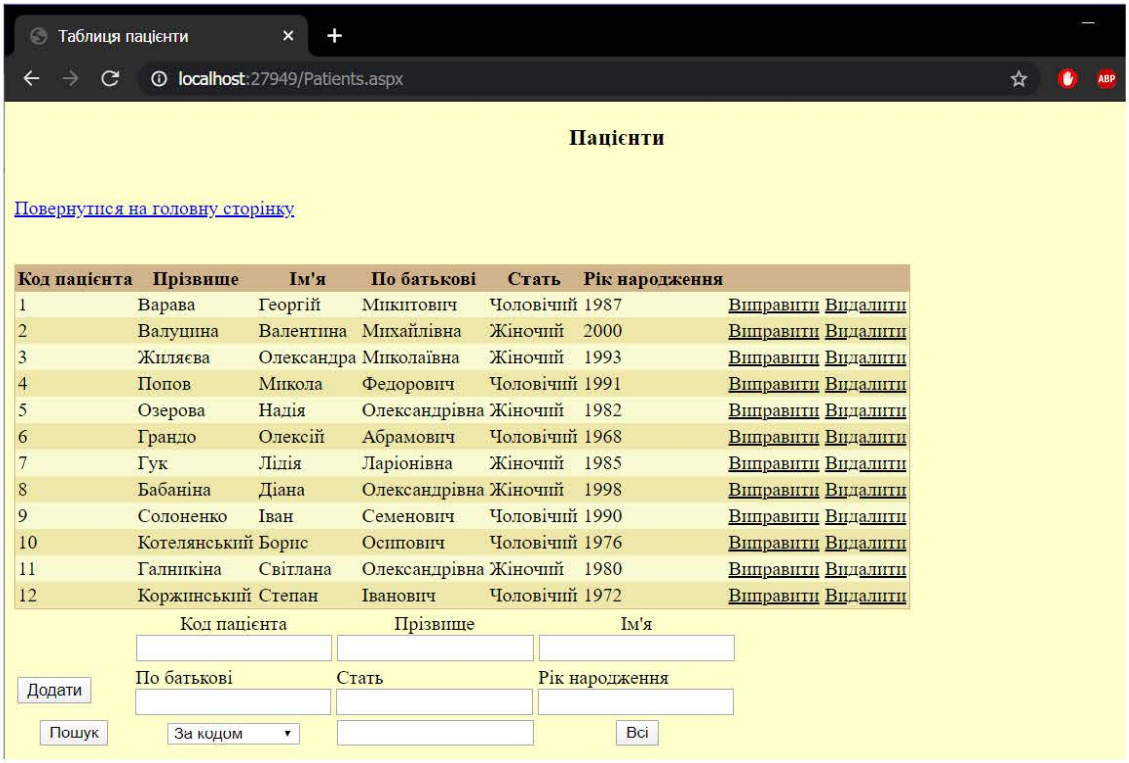

Рисунок 3.6 – Сторінка для редагування пацієнтів

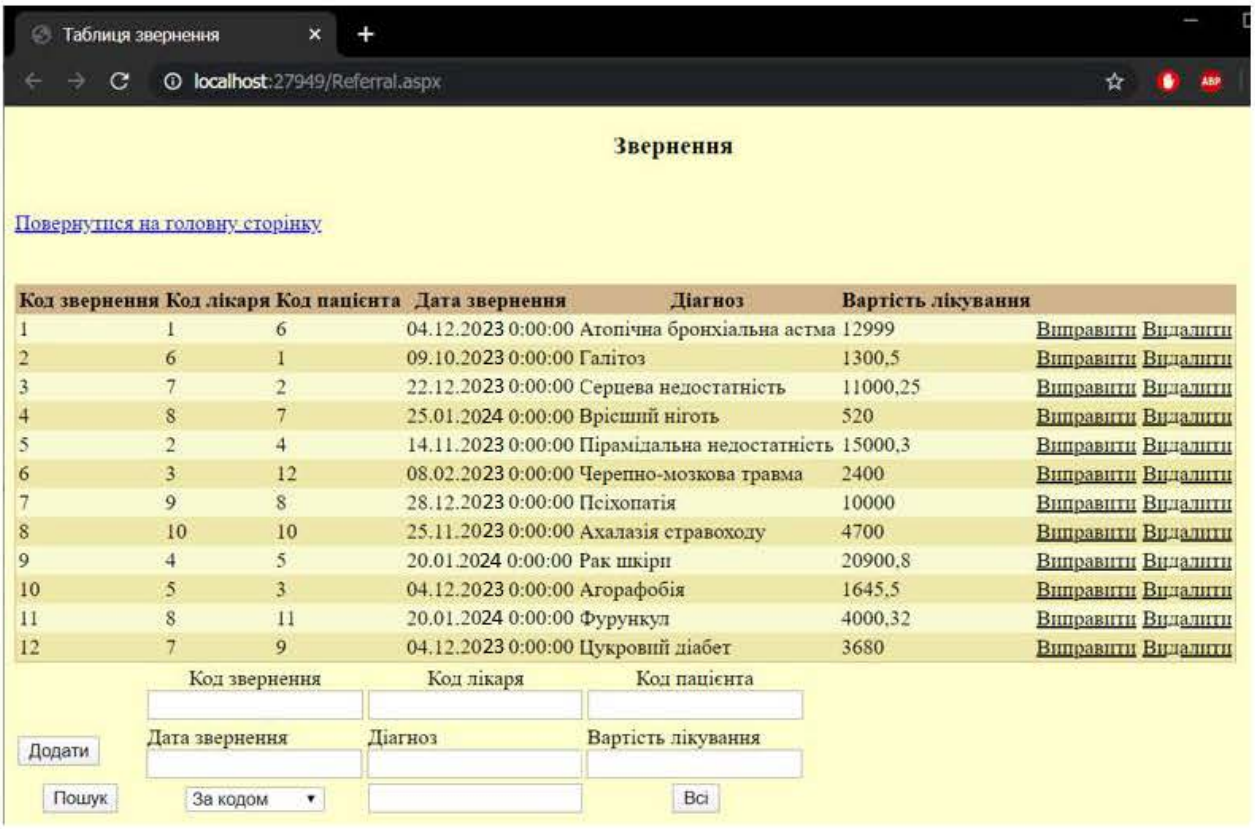

Рисунок 3.7 – Сторінка для редагування звернень

Таблиця 3.5

Програмні механізми сторінок вебзастосунку

Default.aspx.cs	Перевірка вибору з TreeView для показу таблиць.
	protected void TreeView1_SelectedNodeChanged(object sender, EventArgs e)
	{
	switch(TreeView1.SelectedNode.Text){
	case "Таблиця лікарі":{
	GridView1.Visible = true;
	GridView2.Visible = false;
	GridView3.Visible = false;
	break;
	}
	case "Таблиця пацієнти":{
	GridView1.Visible = false;
	GridView2.Visible = true;
	GridView3.Visible = false;
	break;

	}
	case "Таблиця звернення":{
	GridView1.Visible = false;
	GridView2.Visible = false;
	GridView3.Visible = true;
	break;
	}
	}

Doc.aspx.cs	Редагування таблиці «Лікарі». Інформація з бази даних відображається на вебформі за допомогою запитів елемента SqlDataSource. Створено обробники подій:
	1) клік на кнопку «Додати»:
	protected void Button1_Click(object sender, EventArgs e)
	{
	try{
	SqlDataSource1.InsertCommand = "INSERT INTO Doc VALUES (" + Entry_field1.Text + ",'" + Entry_field2.Text + "','" + Entry_field3.Text + "', '" + Entry_field4.Text + "', '" + Entry_field6.Text + "','" + Entry_field7.Text + "')";
	SqlDataSource1.Insert();
	catch(Exception ex){
	Response.Write("Помилка"+ex.Message);
	}}
	2) клік на кнопку «Пошук»:
	protected void Button2_Click(object sender, EventArgs e)
	{
	try{
	switch (DropDownList2.SelectedIndex){

	case 0:{
	SqlDataSource1.SelectCommand = "SELECT * FROM Doc WHERE Doc_id='" + Entry_field5.Text + "'";
	SqlDataSource1.DataBind();
	GridView1.DataBind();
	break; }
	case 1: {
	SqlDataSource1.SelectCommand = "SELECT * FROM Doc WHERE Surname='" + Entry_field5.Text + "'";
	SqlDataSource1.DataBind();
	GridView1.DataBind();
	break; }
	case 2:{
	SqlDataSource1.SelectCommand = "SELECT * FROM Doc WHERE Specialty='" + Entry_field5.Text + "'";

	SqlDataSource1.DataBind();
	GridView1.DataBind();
	break;}}}
	catch (Exception ex){
	Response.Write("Помилка" + ex.Message);
	}}
	3) клік на кнопку «Всі»:
	protected void Button3_Click(object sender, EventArgs e)
	{
	SqlDataSource1.SelectCommand = "SELECT * FROM Doc";
	SqlDataSource1.DataBind();
	GridView1.DataBind();
	}

Patients.aspx.cs	Редагування таблиці «Пацієнти». Інформація з бази даних відображається на вебформі за допомогою запитів елемента SqlDataSource. Створено обробники подій:
	1) клік на кнопку «Додати»:
	protected void Button1_Click(object sender, EventArgs e)
	{
	try{
	SqlDataSource2.InsertCommand = "INSERT INTO Patients VALUES (" + Entry_field1.Text + ",'" + Entry_field2.Text + "','" + Entry_field3.Text + "', '" + Entry_field4.Text + "', '" + Entry_field6.Text + "','" + Entry_field7.Text + "')";
	SqlDataSource2.Insert();
	}
	catch (Exception ex) {
	Response.Write("Помилка" + ex.Message);
	}
	}
	2) клік на кнопку «Пошук»:
	protected void Button2_Click(object sender, EventArgs e)
	{
	try {
	switch (DropDownList2.SelectedIndex){
	case 0: {
	SqlDataSource2.SelectCommand = "SELECT * FROM Patients WHERE Patient_id='" + Entry_field5.Text + "'";
	SqlDataSource2.DataBind();
	GridView2.DataBind();
	break;
	}

	case 1:{
	SqlDataSource2.SelectCommand = "SELECT * FROM Patients WHERE Surname='" + Entry_field5.Text + "'";
	SqlDataSource2.DataBind();
	GridView2.DataBind();
	break;
	}
	}
	}
	catch (Exception ex){
	Response.Write("Помилка" + ex.Message);
	}
	}
	3) клік на кнопку «Всі»:
	protected void Button3_Click(object sender, EventArgs e)
	{
	SqlDataSource2.SelectCommand = "SELECT * FROM Patients";
	SqlDataSource2.DataBind();
	GridView2.DataBind();
	}

Referral.aspx.cs	Редагування таблиці «Звернення». Інформація з бази даних відображається на вебформі за допомогою запитів елемента SqlDataSource. Створено обробники подій:
	1) клік на кнопку «Додати»:
	protected void Button1_Click(object sender, EventArgs e)
	{
	try{
	SqlDataSource3.InsertCommand = "INSERT INTO Referral VALUES (" + Entry_field1.Text + ",'" + Entry_field2.Text + "','" + Entry_field3.Text + "', '" + Entry_field4.Text + "', '" + Entry_field6.Text + "','" + Entry_field7.Text + "')";
	SqlDataSource3.Insert();
	}
	catch (Exception ex){
	Response.Write("Помилка" + ex.Message);
	}
	}
	2) клік на кнопку «Пошук»:
	protected void Button2_Click(object sender, EventArgs e) {
	try{
	switch (DropDownList2.SelectedIndex){
	case 0: {

```
        SqlDataSource3.SelectCommand = "SELECT * FROM
        Referral WHERE Referral_id='" + Entry_field5.Text +
        "'";
         SqlDataSource3.DataBind();
        GridView3.DataBind();
        break; }
        case 1: {
        SqlDataSource3.SelectCommand = "SELECT * FROM
        Referral WHERE Date_of_referral='" +
        Entry_field5.Text + "'";
         SqlDataSource3.DataBind();
        GridView3.DataBind();
        break;}
        case 2: {
        SqlDataSource3.SelectCommand = "SELECT * FROM
        Referral WHERE Diagnosis='" + Entry_field5.Text + "'";
        SqlDataSource3.DataBind();
        GridView3.DataBind();
        break; }
        }
        }
        catch (Exception ex) {
        Response.Write("Помилка" + ex.Message);
         }
}
```

3) клік на кнопку «Всі»:

```
protected void Button3_Click(object sender, EventArgs e)
      {
         SqlDataSource3.SelectCommand = "SELECT * FROM
Referral";
         SqlDataSource3.DataBind();
         GridView3.DataBind();
      }
```

3.7 Висновки до третього розділу

У цьому розділі було розглянуто ключові аспекти розробки вебзастосунку для поліклініки, включаючи опис бази даних, запити до бази даних, інтерфейсну частину вебзастосунку, стиль та мережевий вебінтерфейс, а також програмні механізми.

Опис бази даних зумовлює структуру та організацію зберігання

медичної інформації, що є важливою складовою вебзастосунку для поліклініки. Запити до бази даних дозволяють отримувати, оновлювати та обробляти інформацію, що зберігається в базі даних, що є необхідним для функціонування вебзастосунку.

Інтерфейсна частина вебзастосунку визначає спосіб взаємодії користувача з системою та представлення медичної інформації. Стиль вебзастосунку визначає його зовнішній вигляд та спосіб відображення даних, що є важливим для забезпечення зручного та ефективного використання користувачами.

Мережевий вебінтерфейс забезпечує доступ до вебзастосунку через Інтернет, що робить його доступним для користувачів з будь-якої точки зв'язку. Опис програмних механізмів включає в себе розгляд різних аспектів розробки програмного забезпечення, таких як безпека, масштабованість та продуктивність.

У цьому розділі було розглянуто всі необхідні елементи для успішної розробки та експлуатації вебзастосунку для поліклініки, які забезпечують зручну та ефективну роботу з медичною інформацією.

ВИСНОВКИ

Мета створення вебзастосунку поліклініки полягає в поліпшенні доступності та ефективності надання медичних послуг для пацієнтів, а також оптимізації робочих процесів для медичного персоналу.

Основні цілі включають:

1) забезпечення можливості пацієнтам легко та швидко записуватися на прийом до лікаря, переглядати свою медичну історію та результати обстежень, а також спілкуватися з медичним персоналом онлайн;

2) автоматизація ряду процесів в поліклініці, таких як запис на прийом, управління рецептами та медичними документами, облік фінансових операцій тощо, що дозволяє медичному персоналу більш ефективно виконувати свої обов'язки;

3) покращення якості медичних послуг, забезпечуючи доступ до актуальної медичної інформації, швидку реакцію на запити пацієнтів, а також можливість проведення дистанційних консультацій та віддаленого моніторингу стану пацієнтів;

4) зручність та доступність для пацієнтів, що сприяє покращенню їхнього досвіду користування медичними послугами та підвищенню рівня задоволення від взаємодії з поліклінікою;

5) гарантування захисту медичної інформації пацієнтів, дотримання норм безпеки та конфіденційності даних відповідно до вимог законодавства.

У першому розділі було проведено дослідження сучасних інструментів для розроблення онлайн поліклініки. Виявлено, що для створення вебзастосунків поліклініки широко використовуються такі інструменти, як фреймворки ASP.NET Core, Django, Flask, React, Angular та Vue.js. Кожен з цих інструментів має свої переваги і недоліки, які варто врахувати при виборі для конкретного проекту.

У другому розділі було проведено аналіз існуючого програмного забезпечення для роботи вебзастосунків поліклініки. Досліджені програми

BookMyDoctor, eZdravie, MedApp, eHealth та Мій лікар надають різноманітний функціонал для реалізації потреб поліклінічної системи. Кожен з цих вебзастосунків має свої особливості та можливості, які можуть використовуватися в різних ситуаціях.

У третьому розділі було проведено дослідження сучасних інструментів для розроблення вебзастосунку поліклініки, зокрема вибір програмних засобів та мов програмування. Було обрано набір технологій, що включає мову програмування C#, фреймворк ASP.NET Core, а також використання ADO.NET для роботи з базою даних MSSQLServer. Ці інструменти відомі своєю надійністю, продуктивністю та широким спектром можливостей, що робить їх ідеальним вибором для розробки вебзастосунків поліклініки.

Загалом, проведений аналіз дозволяє визначити оптимальні технології та підходи для створення вебзастосунку поліклініки, що відповідає сучасним стандартам якості та забезпечує зручну та ефективну роботу з медичною інформацією.

Об'єктом дослідження цієї роботи був вебзастосунок «Поліклініка».

Предметом дослідження цієї роботи був процес розробки та реалізації онлайн платформи для автоматизації та оптимізації процесу звернень пацієнтів до поліклініки.

У цьому проекті були вирішені наступні задачі:

1) вибір програмних засобів для розроблення вебзастосунку поліклініки, що включав аналіз і порівняння різних технологій та інструментів розробки;
2) розробка імплементація бази даних для зберігання медичної інформації, включаючи опис структури бази даних та запити до неї;
3) створення інтерфейсної частини вебзастосунку, яка включає в себе розробку користувацького інтерфейсу для реєстрації пацієнтів, запису на прийоми, перегляду медичних карток тощо;
4) розробка стилю вебзастосунку для створення зручного та привабливого інтерфейсу для користувачів;

5) реалізація мережевого вебінтерфейсу для забезпечення доступу до вебзастосунку через Інтернет;
6) опис програмних механізмів, які використовуються в розробленому вебзастосунку, таких як мова програмування C#, фреймворк ASP.NET, технологія ADO.NET для роботи з базою даних MSSQLServer.

Ці задачі спрямовані на створення функціонального та ефективного вебзастосунку для поліклініки, який забезпечить покращення обслуговування пацієнтів та оптимізацію робочих процесів медичного закладу.

Практичне значення отриманих результатів полягає в можливості ефективної розробки та впровадження вебзастосунку для поліклініки з використанням оптимальних програмних технологій та інструментів. Це дозволить поліклініці покращити обслуговування пацієнтів, оптимізувати робочі процеси, забезпечити зручний та доступний інтерфейс для користувачів та сприяти підвищенню ефективності роботи медичного закладу в цілому.

Результатами роботи є вебзастосунок «Поліклініка».

Розроблений вебзастосунок поліклініки має велику актуальність у сучасному світі медичної індустрії. Завдяки цьому застосунку можна значно поліпшити доступ пацієнтів до медичних послуг, забезпечити ефективнішу організацію роботи поліклініки, скоротити час очікування на прийом та спростити процеси обслуговування. Такий вебзастосунок дозволить поліклінікам забезпечити високий рівень сервісу, зробити роботу з пацієнтами більш зручною та ефективною, а також покращити обмін даними між медичним персоналом. З урахуванням зростання популярності та важливості цифровізації у сфері охорони здоров'я, розроблений вебзастосунок стає актуальним і необхідним інструментом для сучасних медичних установ.

Кваліфікаційна робота виконана у відмовідності до стандарту спеціальності 121 «Інженерія програмного забезпечення» і демонструє володіння такими компетентностями як:

– здатність ідентифікувати, класифікувати та формулювати вимоги до програмного забезпечення;

– здатність брати участь у проектуванні програмного забезпечення, включаючи проведення моделювання (формальний опис) його структури, поведінки та процесів функціонування;

– здатність розробляти архітектури, модулі та компоненти програмних систем;

– володіння знаннями про інформаційні моделі даних, здатність створювати програмне забезпечення для зберігання, видобування та опрацювання даних;

– здатність застосовувати фундаментальні і міждисциплінарні знання для успішного розв'язання завдань інженерії програмного забезпечення;

– здатність обґрунтовано обирати та освоювати інструментарій з розробки та супроводження програмного забезпечення;

– здатність до алгоритмічного та логічного мислення тощо.

Серед програмних результатів, визначених стандартом, кваліфікаційна робота реалізовує наступні:

– аналізувати, цілеспрямовано шукати і вибирати необхідні для вирішення професійних завдань інформаційно-довідникові ресурси і знання з урахуванням сучасних досягнень науки і техніки;

– сміти розробляти людино-машинний інтерфейс;

– знати та вміти використовувати методи та засоби збору, формулювання та аналізу вимог до програмного забезпечення;

– проводити передпроектне обстеження предметної області, системний аналіз об'єкта проектування;

– вибирати вихідні дані для проектування, керуючись формальними методами опису вимог та моделювання;

– мотивовано обирати мови програмування та технології розробки для розв'язання завдань створення і супроводження програмного забезпечення;

– знати та вміти застосовувати інформаційні технології обробки, зберігання та передачі даних;

– вміти документувати та презентувати результати розробки програмного забезпечення тощо.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Andrew Troelsen, Philip Japikse. C# 6.0 and the .NET 4.6 Framework (English Edition), 2015.

2. C# Documentation. URL: https://learn.microsoft.com/en-us/dotnet/csharp/

3. Sarcar V. Interactive C #, 2018.

4. Bill Wagner. Effective C# (Covers C# 6.0): 50 Specific Ways to Improve Your C# (Effective Software Development Series), 2017.

5. Christian Nagel. Professional C# 7 and .net Core 2.0, 2018.

6. Dirk Strauss. Creating ASP.NET Core Web Applications, 2021.

7. Andreas Helland, Vincent Maverick Durano. ASP.NET Core 5 for Beginners: Kick-start your ASP.NET web development journey with the help of step-by-step tutorials and examples, 2020.

8. Tamir Dresher, Amir Zuker. Hands-On Full-Stack Web Development with ASP.NET Core: Learn end-to-end web development with leading frontend frameworks, such as Angular, React, and Vue, 2018.

9. ASP.NET Documentation. URL: https://learn.microsoft.com/en-us/aspnet/core

10. Mark J. Price. C# 12 and .NET 8 – Modern Cross-Platform Development Fundamentals: Start building websites and services with ASP.NET Core 8, Blazor, and EF Core 8, Eighth Edition, 2023.

11. ADO.NET Documentation. URL: https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/

12. Mitchell, R. SQL: The Ultimate Beginner's Guide to Learn SQL Programming Step by Step. Independently published, 2019.

13. G. V. Reilly. SQL Queries for Mere Mortals: A Hands-On Guide to Data Manipulation in SQL. Addison-Wesley, 2018.

14. W3School. The SQL LIKE Operator – [Електронний ресурс] – Режим доступу: http://www.w3schools.com/sql/sql_like.asp.

15. Kellyn Gorman, Allan Hirt. Introducing Microsoft SQL Server 2019: Reliability, scalability, and security both on premises and in the cloud, 2020.

16. Microsoft SQL Server. Documentation. URL: https://learn.microsoft.com/en-us/sql/sql-server/

ДОДАТОК А

Лістинг створення БД

```
CREATE DATABASE Paid_polyclinic
```

Створення таблиць БД

```
USE Paid_polyclinic
CREATE TABLE Doc
(Doc_id INT PRIMARY KEY,
Surname CHAR(20),
Name CHAR(20),
Middle_name CHAR(20),
Specialty CHAR(20),
Category CHAR(5))
CREATE TABLE Patients
(Patient_id INT PRIMARY KEY,
Surname CHAR(20),
Name CHAR(20),
Middle_name CHAR(20),
Gender CHAR(9),
Year_of_birth INT)
CREATE TABLE Referral
(Referral_id INT PRIMARY KEY,
Doc_id INT FOREIGN KEY REFERENCES Doc (Doc_id),
Patient_id INT FOREIGN KEY REFERENCES Patients(Patient_id),
Date_of_referral DATETIME,
Diagnosis CHAR(30),
Cost_of_treatment FLOAT)
```

Заповнення таблиць БД

```
USE Paid_polyclinic
INSERT INTO Doc VALUES
(1,'Степашкіна','Клавдія','Іванівна','Алерголог','Перша'),
(2,'Безсмертна','Карина','Валеріївна','Невропатолог','Друга'),
(3,'Пучківська','Надія','Олександрівна','Нейрохірург','Вища'),
(4,'Нечипоренко','Тетяна','Олексіївна','Онколог','Перша'),
(5,'Данилевський','Олексій','Іванович','Психіатр','Вища'),
(6,'Єненко','Юрій','Олексійович','Стоматолог','Друга'),
(7,'Березовська','Олена','Петрівна','Терапевт','Перша'),
(8,'Дерев'янко','Іван','Хомич','Хірург','Перша'),
(9,'Гошкевич','Михайло','Іванович','Психолог','Друга'),
```

(10,'Хращевська','Оксана','Миколаївна','Рентгенолог','Перша')
INSERT INTO Patients VALUES
(1,'Варава','Георгій','Микитович','Чоловічий',1987),
(2,'Валуцина','Валентина','Михайлівна','Жіночий',2000),
(3,'Жиляєва','Олександра','Миколаївна','Жіночій',1993),
(4,'Попов','Микола','Федорович','Чоловічий',1991),
(5,'Озерова','Надія','Олександрівна','Жіночий',1982),
(6,'Трандо','Олексій','Абрамович','Чоловічий',1968),
(7,'Тук','Лідія','Ларіонівна','Жіночий',1985),
(8,'Бабаніна','Діана','Олександрівна','Жіночий',1998),
(9,'Солоненко','Іван','Семенович','Чоловічий',1990),
(10,'Котелянський','Борис','Осипович','Чоловічий',1976),
(11,'Галникіна','Світлана','Олександрівна','Жіночий',1980),
(12,'Коржинський','Степан','Іванович','Чоловічий',1972)
INSERT INTO Referral VALUES
(1,1,6,'20171204','Атопічна бронхіальна астма',12999.90),
(2,6,1,'20171009','Галітоз',1300.50),
(3,7,2,'20171222','Серцева недостатність',11000.25),
(4,8,7,'20180125','Врісший ніготь',520.00),
(5,2,4,'20171114','Піраміdальна недостатність',15000.30),
(6,3,12,'20180208','Черепно-мозкова травма',2400.00),
(7,9,8,'20171228','Психопатія',10000.00),
(8,10,10,'20171125','Ахалазія стравоходу',4700.40),
(9,4,5,'20180120','Рак шкіри',20900.80),
(10,5,3,'20181204','Агорафобія',1645.50),
(11,8,11,'20180120','Фурункул',4000.32),
(12,7,9,'20181204','Цукровий діабет',3680.60)

Додаток Б

Лістинг створення вебсторінок: вміст файлів .aspx та .cs

Default.aspx

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="Default.aspx.cs" Inherits="WebApplication1.WebForm1" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
   <title>База данних</title>
   <style type="text/css">
      .auto-style1_1 {
         width: 588px;
      }
      .auto-style1_2 {
         height: 52px;
      }
      .auto-style1_3 {
         width: 1535px;
      }
      </style>
</head>
<body style="background-color: #FFFFCC">
   <form id="form1" runat="server">
      <asp:SqlDataSource ID="SqlDataSource1" runat="server"
ConnectionString="<%$ ConnectionStrings:Paid_polyclinicConnectionString
%>"
         SelectCommand="SELECT * FROM [Doc]"
         DeleteCommand="DELETE FROM Doc WHERE (Doc_id = @Doc_id)"
         InsertCommand="INSERT INTO Doc (Doc_id, Surname, Name,
Middle_name, Specialty, Category) VALUES (@Doc_id, @Surname, @Name,
@Middle_name, @Specialty, @Category)"
         UpdateCommand="UPDATE Doc SET Doc_id = @Doc_id, Surname =
@Surname, Name = @Name, Middle_name = @Middle_name, Specialty =
@Specialty, Category = @Category">
      </asp:SqlDataSource>
      <asp:SqlDataSource ID="SqlDataSource2" runat="server"
ConnectionString="<%$ ConnectionStrings:Paid_polyclinicConnectionString
%>" SelectCommand="SELECT * FROM [Patients]"
DeleteCommand="DELETE FROM Patients WHERE (Patient_id =
@Patient_id)" InsertCommand="INSERT INTO Patients(Patient_id, Surname,
Name, Middle_name, Gender, Year_of_birth) VALUES (@Patient_id, @Surname,
```

```
@Name, @Middle_name, @Gender, @Year_of_birth)"
UpdateCommand="UPDATE Patients SET Patient_id = @Patient_id, Surname =
@Surname, Name = @Name, Middle_name = @Middle_name, Gender =
@Gender, Year_of_birth = @Year_of_birth"></asp:SqlDataSource>
    <asp:SqlDataSource ID="SqlDataSource3" runat="server"
ConnectionString="<%$ ConnectionStrings:Paid_polyclinicConnectionString
%>" SelectCommand="SELECT * FROM [Referral]"
DeleteCommand="DELETE FROM Referral WHERE (Referral_id =
@Referral_id)" InsertCommand="INSERT INTO Referral(Referral_id, Doc_id,
Patient_id, Date_of_referral, Diagnosis, Cost_of_treatment) VALUES
(@Referral_id, @Doc_id, @Patient_id, @Date_of_referral, @Diagnosis,
@Cost_of_treatment)" UpdateCommand="UPDATE Referral SET Referral_id =
@Referral_id, Doc_id = @Doc_id, Patient_id = @Patient_id, Date_of_referral =
@Date_of_referral, Diagnosis = @Diagnosis, Cost_of_treatment =
@Cost_of_treatment"></asp:SqlDataSource>
    <table>
      <tr>
          <td colspan="2" style="text-align:center;" class="auto-style2"><h3
style="width: 1085px">Платна поліклініка</h3></td>
        </tr>
      <tr>
        <td class="auto-style1">
          <br />
          <asp:TreeView ID="TreeView1" runat="server"
OnSelectedNodeChanged="TreeView1_SelectedNodeChanged">
            <Nodes>
              <asp:TreeNode Text="Переглянути" Value="Головна
сторінка">
                <asp:TreeNode Text="Таблиця лікарі" Value="Таблиця
лікарі"></asp:TreeNode>
                <asp:TreeNode Text="Таблиця пацієнти" Value="Таблиця
пацієнти"></asp:TreeNode>
                <asp:TreeNode Text="Таблиця звернення" Value="Таблиця
звернення"></asp:TreeNode>
              </asp:TreeNode>
              <asp:TreeNode Text="Редагувати"
Value="Редагувати"></asp:TreeNode>
            </Nodes>
          </asp:TreeView>
            
    <asp:LinkButton ID="LinkButton1" runat="server"
PostBackUrl="~/Doc.aspx">Таблиця Лікарі</asp:LinkButton>
          <br />
                  
```

```
                <asp:LinkButton ID="LinkButton2" runat="server"
PostBackUrl="~/Patients.aspx">Таблиця Пацієнти</asp:LinkButton>
                <br />
           
   <asp:LinkButton ID="LinkButton3" runat="server"
PostBackUrl="~/Referral.aspx">Таблиця Звернення</asp:LinkButton>
                <br />
            </td>
            <td class="auto-style3">
        <asp:GridView ID="GridView1" runat="server"
AutoGenerateColumns="False" DataKeyNames="Doc_id"
DataSourceID="SqlDataSource1" style="margin-top: 0px" Visible="False"
BackColor="LightGoldenrodYellow" BorderColor="Tan" BorderWidth="1px"
CellPadding="2" ForeColor="Black" GridLines="None">
            <AlternatingRowStyle BackColor="PaleGoldenrod" />
            <Columns>
                <asp:BoundField DataField="Surname" HeaderText="Прізвище"
SortExpression="Surname" />
                <asp:BoundField DataField="Name" HeaderText="Ім'я"
SortExpression="Name" />
                <asp:BoundField DataField="Middle_name" HeaderText="По
батькові" SortExpression="Middle_name" />
                <asp:BoundField DataField="Specialty" HeaderText="Спеціальність"
SortExpression="Specialty" />
                <asp:BoundField DataField="Category" HeaderText="Категорія"
SortExpression="Category" />
            </Columns>
            <FooterStyle BackColor="Tan" />
            <HeaderStyle BackColor="Tan" Font-Bold="True" />
            <PagerStyle BackColor="PaleGoldenrod" ForeColor="DarkSlateBlue"
HorizontalAlign="Center" />
            <SelectedRowStyle BackColor="DarkSlateBlue"
ForeColor="GhostWhite" />
            <SortedAscendingCellStyle BackColor="#FAFAE7" />
            <SortedAscendingHeaderStyle BackColor="#DAC09E" />
            <SortedDescendingCellStyle BackColor="#E1DB9C" />
            <SortedDescendingHeaderStyle BackColor="#C2A47B" />
        </asp:GridView>
                <asp:GridView ID="GridView2" runat="server"
AutoGenerateColumns="False" DataKeyNames="Patient_id"
DataSourceID="SqlDataSource2" Visible="False"
BackColor="LightGoldenrodYellow" BorderColor="Tan" BorderWidth="1px"
CellPadding="2" ForeColor="Black" GridLines="None">
                    <AlternatingRowStyle BackColor="PaleGoldenrod" />
```

```
        <Columns>
            <asp:BoundField DataField="Surname" HeaderText="Прізвище"
SortExpression="Surname" />
            <asp:BoundField DataField="Name" HeaderText="Ім'я"
SortExpression="Name" />
            <asp:BoundField DataField="Middle_name" HeaderText="По
батькові" SortExpression="Middle_name" />
            <asp:BoundField DataField="Gender" HeaderText="Стать"
SortExpression="Gender" />
            <asp:BoundField DataField="Year_of_birth" HeaderText="Рік
народження" SortExpression="Year_of_birth" />
        </Columns>
                <FooterStyle BackColor="Tan" />
                <HeaderStyle BackColor="Tan" Font-Bold="True" />
                <PagerStyle BackColor="PaleGoldenrod"
ForeColor="DarkSlateBlue" HorizontalAlign="Center" />
                <SelectedRowStyle BackColor="DarkSlateBlue"
ForeColor="GhostWhite" />
                <SortedAscendingCellStyle BackColor="#FAFAE7" />
                <SortedAscendingHeaderStyle BackColor="#DAC09E" />
                <SortedDescendingCellStyle BackColor="#E1DB9C" />
                <SortedDescendingHeaderStyle BackColor="#C2A47B" />
    </asp:GridView>
            <asp:GridView ID="GridView3" runat="server"
AutoGenerateColumns="False" DataKeyNames="Referral_id"
DataSourceID="SqlDataSource3" Visible="False"
BackColor="LightGoldenrodYellow" BorderColor="Tan" BorderWidth="1px"
CellPadding="2" ForeColor="Black" GridLines="None">
                <AlternatingRowStyle BackColor="PaleGoldenrod" />
        <Columns>
            <asp:BoundField DataField="Doc_id" HeaderText="Код лікаря"
SortExpression="Doc_id" />
            <asp:BoundField DataField="Patient_id" HeaderText="Код
пацієнта" SortExpression="Patient_id" />
            <asp:BoundField DataField="Date_of_referral" HeaderText="Дата
звернення" SortExpression="Date_of_referral" />
            <asp:BoundField DataField="Diagnosis" HeaderText="Діагноз"
SortExpression="Diagnosis" />
            <asp:BoundField DataField="Cost_of_treatment"
HeaderText="Вартість лікування" SortExpression="Cost_of_treatment" />
        </Columns>
                <FooterStyle BackColor="Tan" />
                <HeaderStyle BackColor="Tan" Font-Bold="True" />
                <PagerStyle BackColor="PaleGoldenrod"
```

```
ForeColor="DarkSlateBlue" HorizontalAlign="Center" />
                <SelectedRowStyle BackColor="DarkSlateBlue"
ForeColor="GhostWhite" />
                <SortedAscendingCellStyle BackColor="#FAFAE7" />
                <SortedAscendingHeaderStyle BackColor="#DAC09E" />
                <SortedDescendingCellStyle BackColor="#E1DB9C" />
                <SortedDescendingHeaderStyle BackColor="#C2A47B" />
        </asp:GridView>
          </td>
    </tr>
        </table>
    </form>
</body>
</html>
```

Default.aspx.cs

```
using System;
using System.Linq;
using System.Collections.Generic;
using System.Web;
using System.Web.UI.WebControls;
using System.Web.UI;
namespace WebApplication1
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
        }
        protected void TreeView1_SelectedNodeChanged(object sender, EventArgs e)
        {
            switch(TreeView1.SelectedNode.Text)
            {
                case "Переглянути":
                    {
                        GridView1.Visible = false;
                        GridView2.Visible = false;
                        GridView3.Visible = false;
                        break;
                    }
                case "Таблиця лікарі":
                    {
                        GridView1.Visible = true;
                        GridView2.Visible = false;
                        GridView3.Visible = false;
```

```
                break;
            }
        case "Таблиця пацієнти":
            {
                GridView1.Visible = false;
                GridView2.Visible = true;
                GridView3.Visible = false;
                break;
            }
        case "Таблиця звернення":
            {
                GridView1.Visible = false;
                GridView2.Visible = false;
                GridView3.Visible = true;
                break;
            }
        case "Редагувати":
            {
                GridView1.Visible = false;
                GridView2.Visible = false;
                GridView3.Visible = false;
                break;
            }
    }
  }
}
```

Doc.aspx

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="Doc.aspx.cs" Inherits="WebApplication1.WebForm2" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
  <title>Таблиця лікарі</title>
</head>
<body style="background-color: #FFFFCC">
  <form id="form1" runat="server">
    <h3 style="text-align:center">Лікарі</h3>
    <asp:SqlDataSource ID="SqlDataSource1" runat="server"
ConnectionString="<%$ ConnectionStrings:Paid_polyclinicConnectionString
%>"
      SelectCommand="SELECT * FROM [Doc]"
      DeleteCommand="DELETE FROM Doc WHERE (Doc_id = @Doc_id)"
```

```
        InsertCommand="INSERT INTO Doc (Doc_id, Surname, Name,
Middle_name, Specialty, Category) VALUES (@Doc_id, @Surname, @Name,
@Middle_name, @Specialty, @Category)"
        UpdateCommand="UPDATE Doc SET Doc_id = @Doc_id, Surname =
@Surname, Name = @Name, Middle_name = @Middle_name, Specialty =
@Specialty, Category = @Category">
    </asp:SqlDataSource>
    <br />
    <asp:LinkButton ID="LinkButton1" runat="server"
PostBackUrl="~/Default.aspx">Повернутися на головну
сторінку</asp:LinkButton>
    <br /><br /><br />
    <asp:GridView ID="GridView1" runat="server"
AutoGenerateColumns="False" DataKeyNames="Doc_id"
DataSourceID="SqlDataSource1" style="margin-top: 0px"
BackColor="LightGoldenrodYellow" BorderColor="Tan" BorderWidth="1px"
CellPadding="2" ForeColor="Black" GridLines="None">
        <AlternatingRowStyle BackColor="PaleGoldenrod" />
        <Columns>
            <asp:BoundField DataField="Doc_id" HeaderText="Код лікаря"
ReadOnly="True" SortExpression="Doc_id" />
            <asp:BoundField DataField="Surname" HeaderText="Прізвище"
SortExpression="Surname" />
            <asp:BoundField DataField="Name" HeaderText="Ім'я"
SortExpression="Name" />
            <asp:BoundField DataField="Middle_name" HeaderText="По
батькові" SortExpression="Middle_name" />
            <asp:BoundField DataField="Specialty" HeaderText="Спеціальність"
SortExpression="Specialty" />
            <asp:BoundField DataField="Category" HeaderText="Категорія"
SortExpression="Category" />
            <asp:CommandField ShowEditButton="True"
CancelText="Відмінити" DeleteText="Видалити" EditText="Виправити"
InsertText="Вставити" NewText="Створити" SelectText="Вибір"
UpdateText="Оновити" />
            <asp:CommandField ShowDeleteButton="True"
CancelText="Відмінити" DeleteText="Видалити" EditText="Виправити"
InsertText="Вставити" NewText="Створити" SelectText="Вибір"
UpdateText="Оновити" />
        </Columns>
        <FooterStyle BackColor="Tan" />
        <HeaderStyle BackColor="Tan" Font-Bold="True" />
        <PagerStyle BackColor="PaleGoldenrod" ForeColor="DarkSlateBlue"
HorizontalAlign="Center" />
```

```
        <SelectedRowStyle BackColor="DarkSlateBlue"
ForeColor="GhostWhite" />
        <SortedAscendingCellStyle BackColor="#FAFAE7" />
        <SortedAscendingHeaderStyle BackColor="#DAC09E" />
        <SortedDescendingCellStyle BackColor="#E1DB9C" />
        <SortedDescendingHeaderStyle BackColor="#C2A47B" />
      </asp:GridView>
      <table style="width: 594px" >
      <tr style="text-align:center;">
        <td style ="width: 115px">
        <span class="field-validation-error" data-valmsg-for="
Entry_field1" data-valmsg-replace="true" >Код лікаря не може бути
пустим</span></td>
        <td>
          <asp:Label ID="Label1" Font-Size="16px" runat="server" Text="Код
лікаря"></asp:Label> <br/>
          <asp:TextBox ID ="Entry_field1" runat="server"></asp:TextBox>
        </td>
        <td>
          <asp:Label ID="Label2" Font-Size="16px" runat="server"
Text="Прізвище"></asp:Label> <br/>
          <asp:TextBox ID ="Entry_field2" runat="server"></asp:TextBox>
        </td>
        <td>
          <asp:Label ID="Label3" Font-Size="16px" runat="server"
Text="Ім'я"></asp:Label> <br/>
          <asp:TextBox ID ="Entry_field3" runat="server"></asp:TextBox>
        </td>
        </tr>
        <tr>
          <td><asp:Button ID="Button1" runat="server" Text="Додати"
OnClick="Button1_Click" /></td>
        <td style="width: 108px">
          <asp:Label ID="Label4" Font-Size="16px" runat="server" Text="По
батькові"></asp:Label> <br/>
          <asp:TextBox ID ="Entry_field4" runat="server"></asp:TextBox>
        </td>
        <td style="width: 108px">
          <asp:Label ID="Label5" Font-Size="16px" runat="server"
Text="Спеціальність"></asp:Label> <br/>
          <asp:TextBox ID ="Entry_field6" runat="server"></asp:TextBox>
        </td>
        <td style="width: 108px">
          <asp:Label ID="Label6" Font-Size="16px" runat="server"
```

```
Text="Категорія"></asp:Label> <br/>
            <asp:TextBox ID ="Entry_field7" runat="server"></asp:TextBox>
          </td>
        </tr>
        <tr style="text-align:center;">
          <td><asp:Button ID="Button2" runat="server" Text="Пошук"
OnClick="Button2_Click" /></td>
          <td style="text-align:center;">
            <asp:DropDownList ID="DropDownList2" runat="server">
            <asp:ListItem>За кодом</asp:ListItem>
            <asp:ListItem>За прізвищем</asp:ListItem>
            <asp:ListItem>За спеціальністю</asp:ListItem>
            </asp:DropDownList>
        </td>
          <td><asp:TextBox ID="Entry_field5"
runat="server"></asp:TextBox></td>
          <td colspan="2"><asp:Button ID="Button3" runat="server" Text="Всі"
OnClick="Button3_Click" /></td></tr>
    </table>
        </form>
</body>
</html>
```

Doc.aspx.cs

```
using System;
using System.Linq;
using System.Collections.Generic;
using System.Web;
using System.Web.UI.WebControls;
using System.Web.UI;
namespace WebApplication1
{
    public partial class WebForm2 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
        }
        protected void Button1_Click(object sender, EventArgs e)
        {
            try
            {
                SqlDataSource1.InsertCommand = "INSERT INTO Doc VALUES (" +
Entry_field1.Text + ",'" + Entry_field2.Text + "','" + Entry_field3.Text + "', '" +
Entry_field4.Text + "', '" + Entry_field6.Text + "','" + Entry_field7.Text + "')";
                SqlDataSource1.Insert();
```

```
        }
        catch(Exception ex)
        {
            Response.Write("Помилка"+ex.Message);
        }
        Entry_field1.Text = "";
        Entry_field2.Text = "";
        Entry_field3.Text = "";
        Entry_field4.Text = "";
        Entry_field6.Text = "";
        Entry_field7.Text = "";
    }
    protected void Button2_Click(object sender, EventArgs e)
    {
        try
        {
            switch (DropDownList2.SelectedIndex)
            {
                case 0:
                    {

                        SqlDataSource1.SelectCommand = "SELECT * FROM Doc
WHERE Doc_id='" + Entry_field5.Text + "'";
                        SqlDataSource1.DataBind();
                        GridView1.DataBind();
                        break;
                    }
                case 1:
                    {
                        SqlDataSource1.SelectCommand = "SELECT * FROM Doc
WHERE Surname='" + Entry_field5.Text + "'";
                        SqlDataSource1.DataBind();
                        GridView1.DataBind();
                        break;
                    }
                case 2:
                    {
                        SqlDataSource1.SelectCommand = "SELECT * FROM Doc
WHERE Specialty='" + Entry_field5.Text + "'";
                        SqlDataSource1.DataBind();
                        GridView1.DataBind();
                        break;
                    }
            }
```

```
        }
        catch (Exception ex)
        {
            Response.Write("Помилка" + ex.Message);
        }
        Entry_field5.Text = "";
    }
    protected void Button3_Click(object sender, EventArgs e)
    {
        SqlDataSource1.SelectCommand = "SELECT * FROM Doc";
        SqlDataSource1.DataBind();
        GridView1.DataBind();
    }
  }
}
```

Patients.aspx

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="Patients.aspx.cs" Inherits="WebApplication1.WebForm3" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
  <title>Таблиця пацієнти</title>
</head>
<body style="background-color: #FFFFCC">
  <form id="form1" runat="server">
    <h3 style="text-align:center">Пацієнти</h3>
    <asp:SqlDataSource ID="SqlDataSource2" runat="server"
ConnectionString="<%$ ConnectionStrings:Paid_polyclinicConnectionString
%>"
      SelectCommand="SELECT * FROM [Patients]"
      DeleteCommand="DELETE FROM Patients WHERE (Patient_id =
@Patient_id)"
      InsertCommand="INSERT INTO Patients(Patient_id, Surname, Name,
Middle_name, Gender, Year_of_birth) VALUES (@Patient_id, @Surname,
@Name, @Middle_name, @Gender, @Year_of_birth)"
      UpdateCommand="UPDATE Patients SET Patient_id = @Patient_id,
Surname = @Surname, Name = @Name, Middle_name = @Middle_name,
Gender = @Gender, Year_of_birth = @Year_of_birth"></asp:SqlDataSource>
    <br />
    <asp:LinkButton ID="LinkButton1" runat="server"
PostBackUrl="~/Default.aspx">Повернутися на головну
сторінку</asp:LinkButton>
      <br /><br /><br />
```

```
<asp:GridView ID="GridView2" runat="server"
AutoGenerateColumns="False" DataKeyNames="Patient_id"
DataSourceID="SqlDataSource2" BackColor="LightGoldenrodYellow"
BorderColor="Tan" BorderWidth="1px" CellPadding="2" ForeColor="Black"
GridLines="None">
    <AlternatingRowStyle BackColor="PaleGoldenrod" />
    <Columns>
        <asp:BoundField DataField="Patient_id" HeaderText="Код
пацієнта" ReadOnly="True" SortExpression="Patient_id" />
        <asp:BoundField DataField="Surname" HeaderText="Прізвище"
SortExpression="Surname" />
        <asp:BoundField DataField="Name" HeaderText="Ім'я"
SortExpression="Name" />
        <asp:BoundField DataField="Middle_name" HeaderText="По
батькові" SortExpression="Middle_name" />
        <asp:BoundField DataField="Gender" HeaderText="Стать"
SortExpression="Gender" />
        <asp:BoundField DataField="Year_of_birth" HeaderText="Рік
народження" SortExpression="Year_of_birth" />
        <asp:CommandField ShowEditButton="True"
CancelText="Відмінити" DeleteText="Видалити" EditText="Виправити"
InsertText="Вставити" NewText="Створити" SelectText="Вибір"
UpdateText="Оновити" />
        <asp:CommandField ShowDeleteButton="True"
CancelText="Відмінити" DeleteText="Видалити" EditText="Виправити"
InsertText="Вставити" NewText="Створити" SelectText="Вибір"
UpdateText="Оновити" />
    </Columns>
    <FooterStyle BackColor="Tan" />
    <HeaderStyle BackColor="Tan" Font-Bold="True" />
    <PagerStyle BackColor="PaleGoldenrod" ForeColor="DarkSlateBlue"
HorizontalAlign="Center" />
    <SelectedRowStyle BackColor="DarkSlateBlue"
ForeColor="GhostWhite" />
    <SortedAscendingCellStyle BackColor="#FAFAE7" />
    <SortedAscendingHeaderStyle BackColor="#DAC09E" />
    <SortedDescendingCellStyle BackColor="#E1DB9C" />
    <SortedDescendingHeaderStyle BackColor="#C2A47B" />
</asp:GridView>
<table style="width: 594px" >
<tr style="text-align:center;">
    <td style ="width: 115px">
    </td>
    <td>
```

```
            <asp:Label ID="Label1" Font-Size="16px" runat="server" Text="Код
пацієнта"></asp:Label> <br/>
            <asp:TextBox ID ="Entry_field1" runat="server"></asp:TextBox>
          </td>
          <td>
            <asp:Label ID="Label2" Font-Size="16px" runat="server"
Text="Прізвище"></asp:Label> <br/>
            <asp:TextBox ID ="Entry_field2" runat="server"></asp:TextBox>
          </td>
          <td>
            <asp:Label ID="Label3" Font-Size="16px" runat="server"
Text="Ім'я"></asp:Label> <br/>
            <asp:TextBox ID ="Entry_field3" runat="server"></asp:TextBox>
          </td>
          </tr>
          <tr>
            <td><asp:Button ID="Button1" runat="server" Text="Додати"
OnClick="Button1_Click" /></td>
          <td style="width: 108px">
            <asp:Label ID="Label4" Font-Size="16px" runat="server" Text="По
батькові"></asp:Label> <br/>
            <asp:TextBox ID ="Entry_field4" runat="server"></asp:TextBox>
          </td>
          <td style="width: 108px">
            <asp:Label ID="Label5" Font-Size="16px" runat="server"
Text="Стать"></asp:Label> <br/>
            <asp:TextBox ID ="Entry_field6" runat="server"></asp:TextBox>
          </td>
          <td style="width: 108px">
            <asp:Label ID="Label6" Font-Size="16px" runat="server" Text="Рік
народження"></asp:Label> <br/>
            <asp:TextBox ID ="Entry_field7" runat="server"></asp:TextBox>
          </td>
        </tr>
        <tr style="text-align:center;">
          <td><asp:Button ID="Button2" runat="server" Text="Пошук"
OnClick="Button2_Click" /></td>
          <td style="text-align:center;"><asp:DropDownList
ID="DropDownList2" runat="server">
            <asp:ListItem>За кодом</asp:ListItem>
            <asp:ListItem>За прізвищем</asp:ListItem>
                              </asp:DropDownList>
        </td>
          <td><asp:TextBox ID="Entry_field5"
```

```
runat="server"></asp:TextBox></td>
        <td colspan="2"><asp:Button ID="Button3" runat="server" Text="Вси"
OnClick="Button3_Click" /></td></tr>
   </table>
   </form>
</body>
</html>
```

Patients.aspx.cs

```
using System;
using System.Linq;
using System.Collections.Generic;
using System.Web;
using System.Web.UI.WebControls;
using System.Web.UI;
namespace WebApplication1
{
   public partial class WebForm3 : System.Web.UI.Page
   {
      protected void Page_Load(object sender, EventArgs e)
      {
      }
      protected void Button1_Click(object sender, EventArgs e)
      {
         try
         {
            SqlDataSource2.InsertCommand = "INSERT INTO Patients VALUES ("
+ TextBox1.Text + ",'" + Entry_field2.Text + "','" + Entry_field3.Text + "', '" +
Entry_field4.Text + "', '" + Entry_field6.Text + "','" + Entry_field7.Text + "')";
            SqlDataSource2.Insert();
         }
         catch (Exception ex)
         {
            Response.Write("Помилка" + ex.Message);
         }
         Entry_field1.Text = "";
         Entry_field2.Text = "";
         Entry_field3.Text = "";
         Entry_field4.Text = "";
         Entry_field6.Text = "";
         Entry_field7.Text = "";
      }
      protected void Button2_Click(object sender, EventArgs e)
      {
         try
```

```
        {
            switch (DropDownList2.SelectedIndex)
            {
                case 0:
                    {

                        SqlDataSource2.SelectCommand = "SELECT * FROM Patients
WHERE Patient_id='" + Entry_field5.Text + "'";
                        SqlDataSource2.DataBind();
                        GridView2.DataBind();
                        break;
                    }
                case 1:
                    {
                        SqlDataSource2.SelectCommand = "SELECT * FROM Patients
WHERE Surname='" + Entry_field5.Text + "'";
                        SqlDataSource2.DataBind();
                        GridView2.DataBind();
                        break;
                    }
            }
        }
        catch (Exception ex)
        {
            Response.Write("Помилка" + ex.Message);
        }
        Entry_field5.Text = "";
    }
    protected void Button3_Click(object sender, EventArgs e)
    {
        SqlDataSource2.SelectCommand = "SELECT * FROM Patients";
        SqlDataSource2.DataBind();
        GridView2.DataBind();
    }
  }
}
```

Referral.aspx

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="Referral.aspx.cs" Inherits="WebApplication1.WebForm4" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
   <title>Таблиця звернення</title>
```

```
</head>
<body style="background-color: #FFFFCC">
   <form id="form1" runat="server">
      <h3 style="text-align:center">Звернення</h3>
      <asp:SqlDataSource ID="SqlDataSource3" runat="server"
ConnectionString="<%$ ConnectionStrings:Paid_polyclinicConnectionString
%>"
         SelectCommand="SELECT * FROM [Referral]"
         DeleteCommand="DELETE FROM Referral WHERE (Referral_id =
@Referral_id)"
         InsertCommand="INSERT INTO Referral(Referral_id, Doc_id, Patient_id,
Date_of_referral, Diagnosis, Cost_of_treatment) VALUES (@Referral_id,
@Doc_id, @Patient_id, @Date_of_referral, @Diagnosis, @Cost_of_treatment)"
         UpdateCommand="UPDATE Referral SET Referral_id = @Referral_id,
Doc_id = @Doc_id, Patient_id = @Patient_id, Date_of_referral =
@Date_of_referral, Diagnosis = @Diagnosis, Cost_of_treatment =
@Cost_of_treatment"></asp:SqlDataSource>
      <br />
      <asp:LinkButton ID="LinkButton1" runat="server"
PostBackUrl="~/Default.aspx">Повернутися на головну
сторінку</asp:LinkButton>
            <br /><br /><br />
      <asp:GridView ID="GridView3" runat="server"
AutoGenerateColumns="False" DataKeyNames="Referral_id"
DataSourceID="SqlDataSource3" BackColor="LightGoldenrodYellow"
BorderColor="Tan" BorderWidth="1px" CellPadding="2" ForeColor="Black"
GridLines="None">
         <AlternatingRowStyle BackColor="PaleGoldenrod" />
         <Columns>
            <asp:BoundField DataField="Referral_id" HeaderText="Код
звернення" ReadOnly="True" SortExpression="Referral_id" />
            <asp:BoundField DataField="Doc_id" HeaderText="Код лікаря"
SortExpression="Doc_id" />
            <asp:BoundField DataField="Patient_id" HeaderText="Код
пацієнта" SortExpression="Patient_id" />
            <asp:BoundField DataField="Date_of_referral" HeaderText="Дата
звернення" SortExpression="Date_of_referral" />
            <asp:BoundField DataField="Diagnosis" HeaderText="Діагноз"
SortExpression="Diagnosis" />
            <asp:BoundField DataField="Cost_of_treatment"
HeaderText="Вартість лікування" SortExpression="Cost_of_treatment" />
            <asp:CommandField ShowEditButton="True"
CancelText="Відмінити" DeleteText="Видалити" EditText="Виправити"
InsertText="Вставити" NewText="Створити" SelectText="Вибір"
```

```
UpdateText="Оновити" />
            <asp:CommandField ShowDeleteButton="True"
CancelText="Відмінити" DeleteText="Видалити" EditText="Виправити"
InsertText="Вставити" NewText="Створити" SelectText="Вибір"
UpdateText="Оновити" />
        </Columns>
        <FooterStyle BackColor="Tan" />
        <HeaderStyle BackColor="Tan" Font-Bold="True" />
        <PagerStyle BackColor="PaleGoldenrod" ForeColor="DarkSlateBlue"
HorizontalAlign="Center" />
        <SelectedRowStyle BackColor="DarkSlateBlue"
ForeColor="GhostWhite" />
        <SortedAscendingCellStyle BackColor="#FAFAE7" />
        <SortedAscendingHeaderStyle BackColor="#DAC09E" />
        <SortedDescendingCellStyle BackColor="#E1DB9C" />
        <SortedDescendingHeaderStyle BackColor="#C2A47B" />
    </asp:GridView>
    <table style="width: 594px" >
    <tr style="text-align:center;">
        <td style ="width: 115px">

        </td>
        <td>
            <asp:Label ID="Label1" Font-Size="16px" runat="server" Text="Код
звернення"></asp:Label> <br/>
            <asp:TextBox ID ="Entry_field1" runat="server"></asp:TextBox>
        </td>
        <td>
            <asp:Label ID="Label2" Font-Size="16px" runat="server" Text="Код
лікаря"></asp:Label> <br/>
            <asp:TextBox ID ="Entry_field2" runat="server"></asp:TextBox>
        </td>
        <td>
            <asp:Label ID="Label3" Font-Size="16px" runat="server" Text="Код
пацієнта"></asp:Label> <br/>
            <asp:TextBox ID ="Entry_field3" runat="server"></asp:TextBox>
        </td>
        </tr>
        <tr>
            <td> <asp:Button ID="Button1" runat="server" Text="Додати"
OnClick="Button1_Click" /></td>
        <td style="width: 108px">
            <asp:Label ID="Label4" Font-Size="16px" runat="server"
Text="Дата звернення"></asp:Label> <br/>
```

```
            <asp:TextBox ID ="Entry_field4" runat="server"></asp:TextBox>
        </td>
        <td style="width: 108px">
            <asp:Label ID="Label5" Font-Size="16px" runat="server"
Text="Діагноз"></asp:Label> <br/>
            <asp:TextBox ID ="Entry_field6" runat="server"></asp:TextBox>
        </td>
        <td style="width: 108px">
            <asp:Label ID="Label6" Font-Size="16px" runat="server"
Text="Вартість лікування"></asp:Label> <br/>
            <asp:TextBox ID ="Entry_field7" runat="server"></asp:TextBox>
        </td>
      </tr>
      <tr style="text-align:center;">
        <td><asp:Button ID="Button2" runat="server" Text="Пошук"
OnClick="Button2_Click" /></td>
        <td style="text-align:center;"><asp:DropDownList
ID="DropDownList2" runat="server">
            <asp:ListItem>За кодом</asp:ListItem>
            <asp:ListItem>За датою</asp:ListItem>
            <asp:ListItem>За діагнозом</asp:ListItem>
                        </asp:DropDownList>
      </td>
        <td><asp:TextBox ID="Entry_field5"
runat="server"></asp:TextBox></td>
        <td colspan="2"><asp:Button ID="Button3" runat="server" Text="Всі"
OnClick="Button3_Click" /></td></tr>
   </table>
   </form>
</body>
</html>
```

Referral.aspx.cs

```
using System;
using System.Linq;
using System.Collections.Generic;
using System.Web;
using System.Web.UI.WebControls;
using System.Web.UI;
namespace WebApplication1
{
   public partial class WebForm4 : System.Web.UI.Page
   {
      protected void Page_Load(object sender, EventArgs e)
      {
```

```
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        try
        {
            SqlDataSource3.InsertCommand = "INSERT INTO Referral VALUES ("
+ TextBox1.Text + ",'" + Entry_field2.Text + "','" + Entry_field3.Text + "', '" +
Entry_field4.Text + "', '" + Entry_field6.Text + "','" + Entry_field7.Text + "')";
            SqlDataSource3.Insert();
        }
        catch (Exception ex)
        {
            Response.Write("Помилка" + ex.Message);
        }
        Entry_field1.Text = "";
        Entry_field2.Text = "";
        Entry_field3.Text = "";
        Entry_field4.Text = "";
        Entry_field6.Text = "";
        Entry_field7.Text = "";
    }
    protected void Button2_Click(object sender, EventArgs e)
    {
        try
        {
            switch (DropDownList2.SelectedIndex)
            {
                case 0:
                    {

                        SqlDataSource3.SelectCommand = "SELECT * FROM Referral
WHERE Referral_id='" + Entry_field5.Text + "'";
                        SqlDataSource3.DataBind();
                        GridView3.DataBind();
                        break;
                    }
                case 1:
                    {
                        SqlDataSource3.SelectCommand = "SELECT * FROM Referral
WHERE Date_of_referral='" + Entry_field5.Text + "'";
                        SqlDataSource3.DataBind();
                        GridView3.DataBind();
                        break;
                    }
```

```
                case 2:
                    {
                        SqlDataSource3.SelectCommand = "SELECT * FROM Referral
WHERE Diagnosis='" + Entry_field5.Text + "'";
                        SqlDataSource3.DataBind();
                        GridView3.DataBind();
                        break;
                    }
            }
        }
        catch (Exception ex)
        {
            Response.Write("Помилка" + ex.Message);
        }
        Entry_field5.Text = "";
    }
    protected void Button3_Click(object sender, EventArgs e)
    {
        SqlDataSource3.SelectCommand = "SELECT * FROM Referral";
        SqlDataSource3.DataBind();
        GridView3.DataBind();
    }
}
}
```