МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

Худзік Володимир Федорович

Програмний модуль обміну повідомленнями телемедичної системи "HIAMS" на основі сокетів./ The socket-based communication module for "HIAMS" telemedicine system

спеціальність: 6.050102 - Комп'ютерна інженерія
освітньо-професійна програма - Комп'ютерні системи та мережі

Випускна кваліфікаційна робота

Виконав: студент групи КСМ-41/1
Худзік Володимир Федорович
_____

Науковий керівник:
Піцун О. Й.
_____

Випускну кваліфікаційну роботу
допущено до захисту:

"___" _____ 20___ р.

Завідувач кафедри
О. М. Березький

ТЕРНОПІЛЬ - 2019

РЕЗЮМЕ

Дипломний проект містить 77 сторінок пояснюючої записки, 38 рисунків, 9 таблиць, 3 додатки. Обсяг графічного матеріалу 2 аркуші формату А3.

Метою бакалаврської роботи є розробка програмного модуля обміну повідомленнями телемедичної системи «HIAMS» на основі сокетів.

В бакалаврській роботі на основі порівняльного аналізу існуючих телемедичних систем обміну повідомленнями виділено їх переваги та недоліки, що дозволило сформувати перелік обов'язкового функціоналу.

Враховуючи ріст популярності застосування інформаційних технологій в медицині, розроблено алгоритм обміну повідомленнями у програмному модулі. Для забезпечення можливості зручного перегляду інформації та взаємодії з системою на пристроях з допомогою бібліотеки SignalR розроблено адаптивний графічний інтерфейс із різним розширенням.

Розроблений програмний модуль складається з клієнтської та серверної частини. У клієнтській частині відображається інформація про користувачів та форма реєстрації та авторизації користувача. У серверній частині можна переглядати інформацію про зареєстрованих користувачів за такими полями: ПІБ, адреса електронної пошти, номер телефону та інші.

Можливість швидкого обміну повідомленнями між користувачами підвищить продуктивність програмного модуля.

Ключові слова: ВЕБ – РЕСУРС, РЕЄСТРАЦІЯ КОРИСТУВАЧІВ, АВТОРИЗАЦІЯ КОРИСТУВАЧІВ, АДАПТИВНИЙ ГРАФІЧНИЙ ІНТЕРФЕЙС.

RESUME

The degree project contains 77 pages of explanatory notes, 38 figures, 9 tables, and 3 appendixes. The volume of graphic material is 2 sheets of A3 format.

The purpose of the degree project the socket-based communication module for «HIAMS» telemedicine system.

In the degree project, on the basis of a comparative analysis of existing telemedicine messaging systems, their advantages and disadvantages have been identified, which made it possible to form a list of obligatory functional.

Given the increasing popularity of information technology applications in medicine, an algorithm for messaging in the software module has been developed. To provide convenient viewing of information and interaction with the system on devices using the library SignalR designed adaptive graphical interface with different extensions.

The developed software module consists of a client and a server part. The client part displays information about users and the form of user registration and authorization. In the server part, you can view information about registered users by the following fields: first name, email address, phone number, and others.

The ability to quickly exchange messages between users will increase the productivity of the software module.

Key words: WEB - RESOURCE, REGISTRATION OF USERS, USER AUTHORIZATION, ADAPTIVE GRAPHIC INTERFACE

ЗМІСТ

ВСТУП

Спілкування – невід’ємна частина життя. Доступність пересування майже по всій земній кулі часом розкидує близьких друзів і родичів на значні відстані. Не зважаючи на інтенсивний розвиток мобільної телефонії, подекуди це залишається дорогим задоволенням. І тут в нагоді стане такий винахід людської думки як месенджер.

Призначення телемедицини полягає в передаванні медичної інформації між віддаленими один від одного пунктами, де знаходяться пацієнти, лікарі, інші провайдери медичної допомоги, між окремими медичними закладами.

Використання телекомунікацій для зв'язку медичних спеціалістів з клініками, лікарнями, лікарями, які надають першу допомогу, пацієнтами, які знаходяться на відстані, з метою діагностики, лікування, консультації та неперервного навчання стало доступне за допомогою системи телемедицини.

Телемедичні системи є помічниками в боротьбі за життя і здоров'я людини і слугують ефективним інструментом для вирішення медичних завдань:

– надання допомоги лікарям, що працюють у віддалених стаціонарних або тимчасово розгорнутих медичних пунктах;

– передавання знань і досвіду фахівців провідних медичних лікувальних і учбових центрів лікарям-практикам, проведення віддалених кваліфікаційних іспитів і сертифікацій;

– надання необхідного і достатнього набору функцій для вирішення завдань діагностики, лікування і реабілітації хворих, навчання і підвищення кваліфікації медичних працівників, а також збирання і поширення управлінської інформації.

Технологічною альтернативою традиційної медичної допомоги є телемедицина. Проте, погані заходи безпеки в службах телемедицини можуть мати негативний вплив на якість наданої допомоги, незалежно від досліджуваного хронічного стану.

Чат – засіб обміну повідомленнями з комп'ютерної мережі в режимі реального часу, а також програмне забезпечення, що дозволяє організовувати таке спілкування. Характерною особливістю є комунікація, а саме в реальному часі або близька до цього, що відрізняє чат від форумів та інших повільних засобів.

Передача повідомлень в режимі реального часу і є головна особливість і перевага чатів. Незважаючи на бурхливе зростання інформаційної індустрії, тема текстових чатів все ще не втратила своєї актуальності і їх часто використовують на підприємствах чи офісах, коли важливо швидко отримувати або відправляти важливу інформацію.

Метою дипломного проекту є розробка програмного модуля обміну повідомлення телемедичної системи «HIAMS» на основі сокетів.

Актуальність роботи полягає у вдосконаленні засобів комунікації між медичними працівниками, а також в отриманні консультації щодо стану здоров’я за найменший проміжок часу.

1 АНАЛІЗ ПРОГРАМНИХ ЗАСОБІВ КОМУНІКАЦІЇ

1.1 Програмні засоби обміну повідомленнями

Сучасні програми вимагають обміну повідомленнями в режимі реального часу з низькою затримкою. Незалежно від того, створюєте веб-додаток чи мобільний користувачі очікують, що зможуть взаємодіяти з даними якомога ближче до реального часу, зводячи до мінімуму вплив на загальну взаємодію з користувачем [1].

Існує багато додатків, з якими ви, ймовірно, взаємодієте сьогодні, які вимагають такого типу підключення і в той же час забезпечують взаємодію користувачів в режимі реального часу або майже в реальному часі. Якщо ви спілкувалися з Twitter або Facebook, ви стикалися з потоками активності майже в реальному часі, які постійно оновлювалися по всій мережі, на вашому мобільному пристрої або в браузері без необхідності натискати кнопку оновлення.

Сокети забезпечують двосторонній зв'язок типу «точка-точка» між двома процесами. Вони є основними компонентами міжсистемної і міжпроцесної взаємодії. Кожен сокет являє собою кінцеву точку зв'язку, з якою може бути пов'язано деякий ім'я. Він також має певний тип, і один або декількох пов'язаних з ним процесів [2].

Сокети існують в областях зв'язку (доменах). Домен сокета – це абстракція, яка визначає структуру адресації і набір протоколів. Сокети можуть з'єднуватися тільки з сокетами в тому ж самому домені зображено на рисунку 1.1.

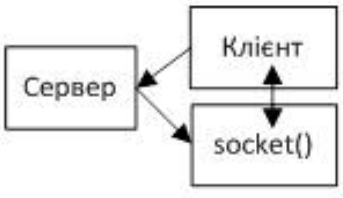

Рисунок 1.1 – Протокол обміну по сокетах

Найкраще те, що WebSockets повністю сумісний і кроссплатформенний на рівні браузера, підтримує порти 80/443, а також міждоменні з'єднання. Протокол WebSocket складається з двох частин: клієнта і сервера.

Існує кілька різновидів програмної реалізації чатів:

– HTTP або веб-чати. Такий чат виглядає як звичайна веб-сторінка, де можна прочитати останні кілька десятків фраз, написані учасниками чату та модераторами. Сторінка чату автоматично оновлюється з заданою періодичністю.

– Чати, що використовують технологію Adobe Flash. Замість періодичної перезавантаження сторінки між клієнтом і сервером відкривається сокет, що дозволяє моментально відправляти або отримувати повідомлення, витрачаючи менше трафіку.

– Програми-чати для спілкування в локальних мережах (наприклад, Vypress Chat, Intranet Chat, Pichat). Часто є можливість передачі файлів.

– Чати, реалізовані поверх сторонніх протоколів.

– Чати, працюючі по схемі клієнт-сервер, це дозволяє використовувати їх в мережах зі складною конфігурацією, а також керувати клієнтськими додатками (наприклад, Mychat, Jabber) чат клієнт сервер мережу.

Метою дипломного проекту є розробка програмного модуля обміну повідомлення телемедичної системи «HAMS» на основі сокетів.

Актуальність роботи полягає у знаходженні нових шляхів популяризації медичних досліджень з можливістю спілкування кількох людей одночасно.

ASP.NET Core може працювати поверх крос-платформного середовища .NET Core, яка може бути розгорнута на основних популярних операційних системах: Windows, Mac OS X, Linux. І таким чином, за допомогою ASP.NET Core ми можемо створювати крос-платформні додатки [4].

Для розгортання веб-додатків можна використовувати традиційний IIS, або крос-платформний веб-сервер Kestrel.

Якщо підсумувати, то можна виділити наступні ключові відмінності ASP.NET Core від попередніх версій ASP.NET:

– Новий легкий і модульний конвеєр HTTP-запитів.

– Можливість розгортати додаток як на IIS(Internet Information Services), так і в рамках свого власного процесу.

– Використання платформи .NET Core і її функціональності.

– Поширення пакетів платформи.

– Інтегрована підтримка для створення та використання пакетів.

– Єдиний стек веб-розробки, що поєднує Web UI і Web API.

– Конфігурація для спрощеного використання в хмарі.

– Вбудована підтримка для впровадження залежностей.

– Можливість розширення.

– Кросплатформеність – можливість розробки і розгортання додатків ASP.NET на Windows, Mac і Linux.

WPF(Windows Presentation Foundation) є частиною екосистеми .NET і розвивається разом з фреймворком .NET і має ті ж версії.

Перша версія WPF 3.0 вийшла разом з .NET 3.0 і операційною системою Windows Vista в 2006 році. З того часу платформа послідовно розвивається.

Остання версія WPF 4.6 вийшла паралельно з .NET 4.6 в липні 2015 року, ознаменувавши дев'ятиріччя даної платформи [5].

Однією з важливих особливостей є використання мови розмітки інтерфейсу XAML(eXtensible Application Markup Language), заснованого на XML(eXtensible Markup Language). Можливість створювати насичений графічний інтерфейс, використовуючи або декларативне оголошення інтерфейсу, або код на керованих
мовах C # і VB.NET, або поєднувати і те, і інше.

WPF входить до складу платформи .NET. Говорячи про .NET як про технологічну платформу, не можна не відзначити той факт, що вона забезпечує одночасну підтримку проектування і реалізації програмного забезпечення з використанням різних мов програмування. При цьому підтримуються десятки мов програмування, починаючи від найперших (зокрема, COBOL і FORTRAN) і

закінчуючи сучасними (наприклад, C # і Visual Basic) [6].

Ранні мови програмування активно використовуються, зокрема, для забезпечення сумісності з раніше створеними додатками зображено на рисунку 1.2.

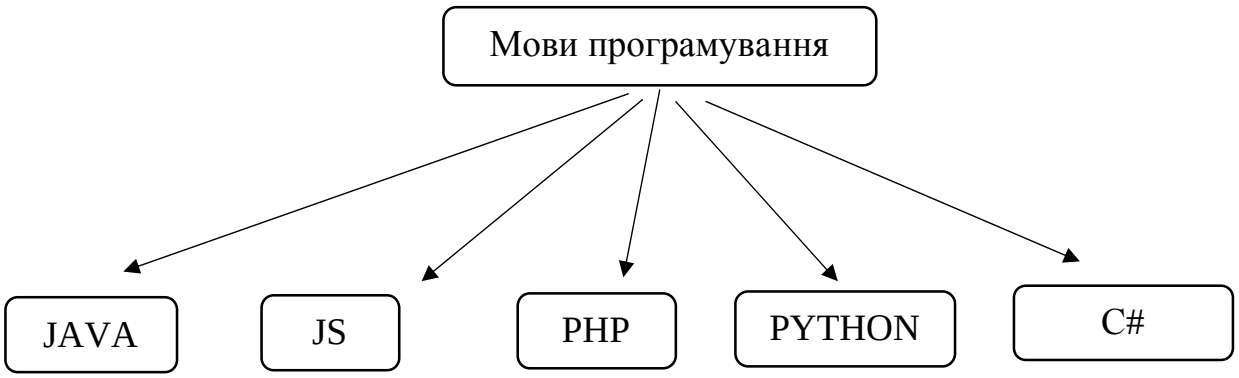

Рисунок 1.2 – Мови програмування

Більшість мов програмування є об’єктно-орієнтованими, однак функціональний підхід до розробки програмних модулів обміну повідомленнями набуває популярності. На сьогоднішній момент мова програмування C # одна з найпотужніших, що швидко розвиваються і затребуваних мов в IT(Information Technologies) галузі. На даний момент на даній пишуться найрізноманітніші програми: від невеликих десктопних програмок до великих веб-сервісів, які обслуговують щодня мільйони користувачів [7].

Об'єктно – орієнтований підхід дозволяє вирішити завдання з побудови великих, але в той же час гнучких, масштабованих і розширюваних додатків. C# продовжує активно розвиватися і з кожною новою версією з'являється все більше цікавих функціональностей, як, наприклад, лямбда, динамічне зв'язування, асинхронні методи [8].

Порівняльна характеристика даних мов програмування наведена у таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика мов програмування

Параметр	PHP	JS	C#	Java	Python
Узагальнене програмування	+	+	+	+	+
Ручне керування пам’яттю	-	-	-	-	-
Функціональність	+	+/-	+	-/+	+
Типізація	неявна	неявна	явна	явна	неявна
Анонімні функції	+	+	+	-	+/-
Наявність інтерфейсів	+	+	+	+	+/-

Аналізувавши вищенаведену таблицю можна зробити висновок, що розглянуті мови програмування володіють схожим функціоналом, однак для написання серверної частини програмного модуля обміну повідомленнями найоптимальнішою є мова програмування C#.

C# – об'єктно-орієнтована і в цьому плані багато перейняла у Java і C++. Наприклад, C# підтримує поліморфізм, успадкування, перевантаження операторів, статичну типізацію [9] .

C# – мова з Сі-подібним синтаксисом і близька в цьому відношенні до C ++ і Java. Тому, якщо ви знайомі з одним з цих мов, то опанувати C# буде легше [10].

Білл Гейтс сказав, що платформа .NET – це найкраще, що створила компанія Microsoft [11]. Можливо, він мав рацію. Фреймворк .NET представляє потужнуплатформу для створення додатків. Можна виділити наступні її основні риси:

– Підтримка декількох мов. Основою платформи є загальномовне середовище виконання Common Language Runtime (CLR), завдяки чому .NET підтримує кілька мов: піряд з C # це також VB.NET, C ++, F#, а також діалекти інших мов, прив'язані до .NET, наприклад, Delphi. NET. При компіляції код

компілюється в збірку спільною мовою CIL (Common Intermediate Language) – асемблер платформи .NET. Тому ми можемо зробити окремі модулі однієї програми на окремих мовах [12].

– Кросплатформленість.

– Потужна бібліотека класів. .NET представляє єдину для всіх підтримуваних мов бібліотеку класів. І який б додаток ми не збиралися писати на C # – текстовий редактор, чат або складний веб-сайт так чи інакше ми задіємо бібліотеку класів .NET.

– Різноманітність технологій. Базова бібліотека класів є основою для цілого стека технологій, які розробники можуть задіяти при побудові тих чи інших додатків.

Для побудови графічних додатків з багатим насиченим інтерфейсом – технологія WPF. Для створення веб-сайтів – ASP.NET.

Також ще слід відзначити таку особливість мови C# і фреймворка .NET, як автоматичне прибирання сміття.

Реляційні бази даних, такі як MySQL, PostgreSQL та SQLite3, представляють і зберігають дані в таблицях і рядках. Тим часом нереляційні бази даних, такі як MongoDB, представляють дані в колекціях документів JSON(JavaScript Object Notation) [13].

Реляційні бази даних використовують мову структурованих запитів, що робить їх гарним вибором для додатків, в яких використовується управління декількома транзакціями. Структура реляційної бази даних дозволяє пов'язувати інформацію з різних таблиць за допомогою зовнішніх ключів (або індексів), які використовуються для унікальної ідентифікації будь-якого атомарного фрагмента даних в цій таблиці.

Інші таблиці можуть посилатися на цей зовнішній ключ, щоб створити зв'язок між їх частинами даних і частиною, на яку вказує зовнішній ключ. Це зручно для додатків, які мають велике значення для аналізу даних.

При розробці додатків, часто стоять завдання щодо передачі даних з мобільного додатку на сервер або інший пристрій. Це можуть бути різні файли, такі як фотографії або відео, дані по датчикам або ж будь-яка інша інформація.

При такому завданні необхідно реалізувати якийсь механізм обміну даними. Найчастіше будуються архітектури, при яких є певний заздалегідь клієнт і сервер [14].

Адже необхідно знати, куди ми будемо відправляти дані і хто їх буде приймати. Крім вибору, хто буде сервером, а хто клієнтом, необхідно ще знати (або визначити), в якому вигляді будуть надходити дані і як їх можна інтерпретувати. Часто сервер виступає просто сховищем файлів або даних зображено на рисунку 1.3.

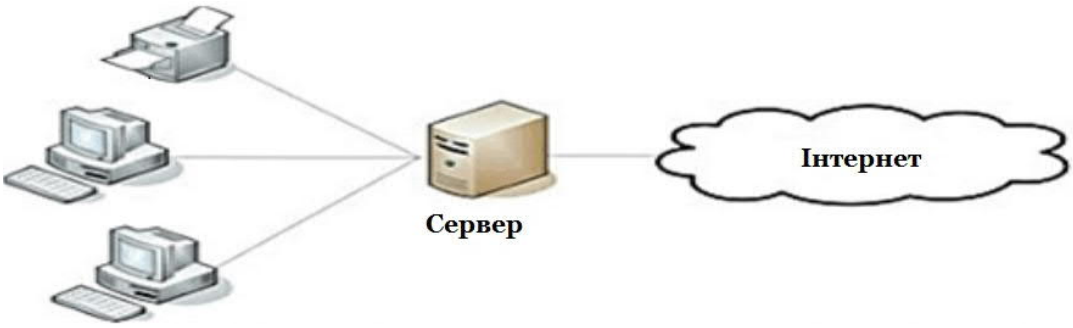

Рисунок 1.3 – Архітектура «клієнт–сервер»

Переваги клієнт-серверної архітектури:

– відсутність дублювання коду програми-сервера програмами клієнтами;

– так як всі обчислення виконуються на сервері, то вимоги до комп'ютерів, на яких встановлено клієнт, знижуються;

– всі дані зберігаються на сервері, який, як правило, захищений набагато краще за більшість клієнтів.

На сервері простіше організувати контроль повноважень, щоб вирішувати доступ до даних тільки клієнтам з відповідними правами доступу [15].

Недоліки клієнт–серверної архітектури:

– непрацездатність сервера може зробити непрацездатною всю мережу;

– підтримка роботи даної системи вимагає окремого фахівця–системного адміністратора;

– висока вартість обладнання.

Непрацездатним сервером слід вважати сервер, продуктивності якого не вистачає на обслуговування всіх клієнтів, а також сервер, що знаходиться на ремонті, профілактиці.

Отже, у даному підрозділі проведено огляд та аналіз сучасних програмних засобів розробки додатків, що дозволило виділити їх переваги та недоліки та встановити, що мова програмування C# найкраще підходить для реалізації поставленої задачі.

1.2 Структура телемедичних систем

Телемедицина – це дистанційне надання медичних послуг, таких як оцінка стану здоров'я або консультації, через телекомунікаційну інфраструктуру. Це дозволяє медичним працівникам оцінювати, діагностувати і лікувати пацієнтів, використовуючи звичайні технології, такі як відеоконференції і смартфони, без необхідності особистого візиту.

Системи телемедицини можуть бути розділені на три основні категорії:

– дистанційне спостереження за пацієнтами;

– зберігання та пересилання;

– інтерактивна телемедицина.

Віддалений моніторинг пацієнтів, також відомий як телемоніторинг, дозволяє проводити моніторинг пацієнтів з хронічними захворюваннями в своїх будинках за допомогою мобільних медичних пристроїв, які збирають дані про рівень цукру в крові, кров'яний тиск або інших життєво–важливих показниках.

Зберігання та пересилання, також відома як асинхронна телемедицина, дозволяє провайдерам обмінюватися інформацією про пацієнта, такої як результати лабораторних досліджень, з лікарем в іншому місці.

Інтерактивна телемедицина дозволяє лікарям і пацієнтам спілкуватися в режимі реального часу. Вона являє собою сукупність засобів та комплексів, які реалізують потенціал сучасних інформаційних та телекомунікаційних технологій в охороні здоров'я, а також відповідні фінансове та правове забезпечення.

Системи телемедицини використовуються в усіх галузях сучасної медицини:

– клінічна медицина – діагностика, визначення тактики лікування, трактування результатів обстежень, допомога в прийнятті рішень;

– організація – створення інформаційних мереж, координація дій різних установ, ведення реєстрів тощо;

– навчальний процес – дистанційне навчання, використання матеріалів телеконсультацій у педагогічному процесі.

В основі телемедицини є безліч технічних засобів, в тому числі і телефон, радіо, модеми, спеціалізовані сканери, спеціалізовані системи відображення відеографічної інформації, а також пристрої суміщення комп'ютерних та спеціалізованих медичних приладів.

Медичну допомогу можна надавати як в реальному часі (наприклад, за допомогою інтерактивного відео), так і з затримкою (передавання текстових чи графічних даних, фотографій та відеокліпів).

Послуги віддаленого аналізу і моніторингу, а також електронне зберігання даних значно скорочують витрати на медичне обслуговування, заощаджуючи гроші для вас, ваших пацієнтів і страхових компаній.

За допомогою телемедицини можна зменшити кількість непотрібних візитів до відділення невідкладної допомоги та усуває витрати на транспортування для регулярних оглядів. Вона пропонує кращий доступ до більшої кількості фахівців, а також можливість направити своїх пацієнтів до

потрібних лікарів, незалежно від їх місцезнаходження. Це нововведення пропонує пацієнт-орієнтовані підходи, такі як підвищення своєчасності надання медичної допомоги. Це дуже важливо для якісного догляду за пацієнтами.

Пацієнти можуть швидко вирішити проблеми зі здоров'ям за допомогою консультацій з невідкладної допомоги в реальному часі і дізнатися про варіанти лікування протягом декількох хвилин.

Слід відзначити, що єдиного методу, який підходить для вирішення всіх задач телемедицини, не існує технічні характеристики кожної системи визначаються виходячи з потреб користувачів [15].

Організаційні фактори можуть бути вирішальними факторами успіху або провалу програми телемедицини. Інновації в технології телемедицини повинні супроводжуватися інноваціями в організаційній комунікації і структурі.

Результати цього дослідження можуть бути застосовані до існуючих або запланованих програм. Приклади включають перевизначення ролей і обов'язків певного персоналу, підвищення ефективності і зниження складності процесу планування консультацій, уточнення і формалізація керівництва і прийняття рішень, а також розвиток універсальної термінології телемедицини.

1.3 Порівняльний аналіз телемедичних систем

Комп’ютеризація суспільства та необхідність поширення та обміну інформацією між науковцями стали причиною формування телемедичної системи.

Систему телемедицини можна розділити на такі категорії:

– Телеконсультація – це діагностичні та терапевтичні рішення за участю віддаленого лікаря або іншого медичного працівника через телекомунікаційні пристрої (наприклад, консультації між лікарем і по телефону).

– Телемануляція – це віддалена маніпуляція, якою лікар може виконують обстеження (наприклад, ендоскопію).

– Теледіагностика – це дистанційна діагностика, при якій здійснюється акт діагностування організму.

Слід зазначити, що телемедицина – це практика використання комунікаційних технологій для зв'язку медичних працівників з їх пацієнтами і один з одним на великих відстанях. Це проявляється по-різному, в основному зосереджена навколо передачі даних і зв'язку. Сучасні технології дозволяють розробляти більше телемедичних систем без витрат з'єднання, на короткій або великій відстані.

З цієї точки зору, телемедицина визначається як надання медичної допомоги та обміну медичними знаннями на відстані за допомогою телекомунікаційних засобів. Таким чином, мета телемедицини надавати експертну медичну допомогу віддаленим ділянкам.

Ухвалення останніх ініціатив в області телемедицини може допомогти вашій практиці добитися численних переваг. Перелік переваг зображено на рисунку 1.4.

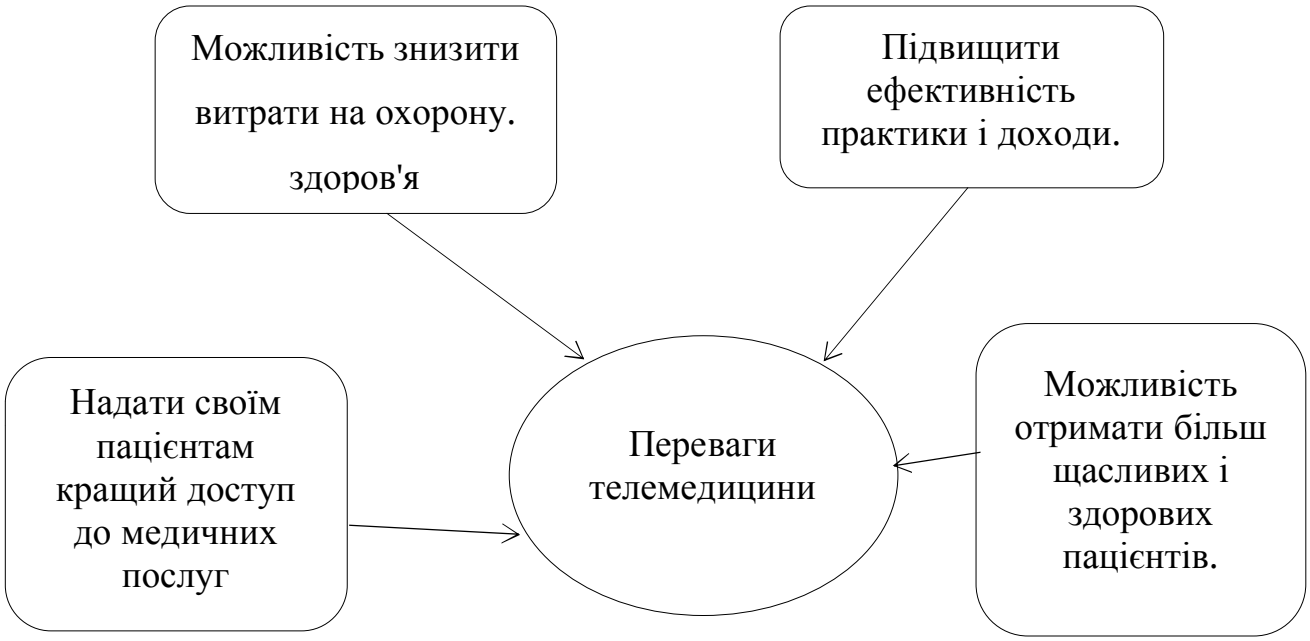

Рисунок 1.4 – Переваги телемедицини

У той час як досягнення в галузі медицини підвищили ефективність використання технологій, бувають випадки, коли відбувається збій системи.

Впровадження нової системи вимагає навчання, і іноді співробітникам важко прийняти цю зміну. Керівники практики, медсестри, лікарі і багато інших повинні навчитися використовувати систему, щоб практики могли бачити переваги. Хоча спочатку телемедицина коштує дорого, системи охорони здоров'я повинні з часом отримувати позитивний повернення інвестицій завдяки більшій кількості пацієнтів і меншій кількості персоналу [15].

Хоча телемедицина приносить багато переваг, у неї є і недоліки. Хоча протягом наступного десятиліття ця область буде рости в геометричній прогресії, вона буде пов'язана з практичними і технологічними проблемами.

Недоліки телемедичної системи зображена на рисунку 1.5.

Рисунок 1.5 – Недоліки телемедицини

Chiron Health, eVisit – приклади темемедичної системи. За допомогою систем телемедицини лікарі можуть економити більше часу в день, а також розширити свої послуги і забезпечити більш якісніший і швидший догляд за пацієнтами [16].

Chiron Health полегшує успішну реалізацію програми телемедицини в вашій практиці завдяки індивідуальними планами, оптимізованим робочим процесам і простих функцій пацієнта.

Як тільки призначена зустріч, пацієнти автоматично отримують електронний лист, щоб налаштувати обліковий запис.

Інтерфейс входу в систему зображено на рисунку 1.6.

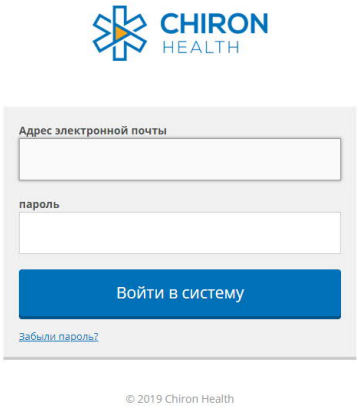

Рисунок 1.6 – Інтерфейс входу в систему телемедицини Chiron Health

Додаток системи телемедицини Chiron Health доступний у платній версії – 150 американських доларів/місяць [16].

Ввівши адресу електронної пошта та пароль отримуємо доступ до персональної сторінки зображено на рисунку 1.7.

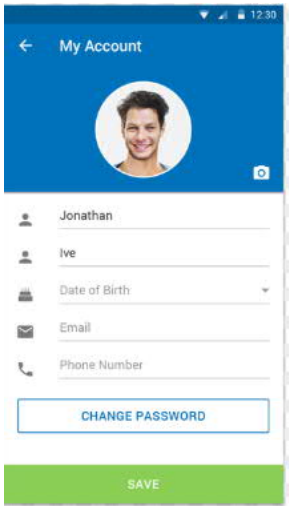

Рисунок 1.7 – Інтерфейс персональної сторінки користувача.

Також можливість отримати запланований розклад візитів та інший функціонал програми зображено на рисунку 1.8.

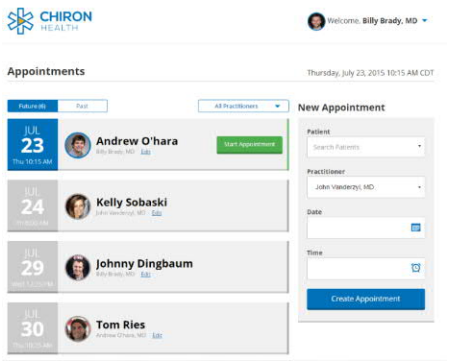

Рисунок 1.8 – Інтерфейс запланованих візитів

Незалежно від того, чи перебуває пацієнт перед комп'ютером або смартфоном, передова технологія дозволяє бачити все в найдрібніших деталях. Тому телемедицину можна вважати важливою складовою сьогодення.

Порівняльна телемедичних систем наведена у таблиці 1.2.

Таблиця 1.2 – Порівняльна таблиця телемедичних систем

Особливості	Опис	Chiron Health	eVisit
Необмежена кількість відвідувань	Можливість бачити своїх пацієнтів по онлайн відео	+	+
Планування	Просте планування для неінтегрованих або автономних відвідувань	+	+
Автоматичні нагадування про зустрічі	Пацієнти отримують нагадування по електронній пошті / SMS для відвідувань	+	+
Мобільні додатки для пацієнтів	Можливіть підключатися з будь–якого місця за допомогою додатків для iOS і Android.	+	+

Отже, до загальних недоліків телемедичних систем можна віднести: ймовірність помилки, оскільки технологія не завжди може вловити те, що може зробити людський дотик, також бувають випадки, коли відбувається збій системи.

1.4 Висновки та постановка задачі

В даному розділі проведено аналітичний огляд підходів до створення системи телемедицини, що дозволило обрати найоптимальнішу мову програмування для отримання поставленої мети, визначити основні структурні елементи, провести порівняльний аналіз сучасних систем телемедицини та виділити їх переваги та недоліки.

Метою даного дипломного проекту є розробка програмного модуля обміну повідомленнями телемедичної системи «HIAMS» на основі сокетів.

Для досягнення поставленої мети потрібно виконати наступні задачі:

– провести огляд підходів до створення системи телемедицини;

– провести порівняльну характеристику сучасних систем;

– розробити програмний модуль обміну повідомленнями телемедичної системи;

– порівняти результати роботи розробленого програмного модуля з іншими відомими системами.

В даному розділі було досліджено основні та найбільш популярні засоби для створення телемедичної системи. Аналіз сучасних мов програмування дозволив виділити мову програмування C# як найоптимальніший варіант для написання програмного модуля. Проведений аналіз сучасних телемедичних систем дозволив виділити їх переваги і недоліки та врахувати їх при написанні власного модуля.

2. АЛГОРИТМИ РОБОТИ МОДУЛЮ ОБМІНУ ПОВІДОМЛЕННЯМИ В ТЕЛЕМЕДИЧНІЙ СИСТЕМІ

2.1 Структура модулю системи реєстрації та авторизації

Алгоритм – набір інструкцій, що описують порядок дій виконавця для досягнення певного результату. У старому трактуванні замість слова «порядок» використовувалося слово «послідовність», але в міру розвитку паралельності в роботі комп'ютерів слово «послідовність» стали замінювати більш загальним словом «порядок». Незалежні інструкції можуть виконуватися в довільному порядку, паралельно, якщо це дозволяється [17].

Алгоритм являється основною частиною програмного продукту, адже саме алгоритмом описуються основні можливості та функції програмного продукту.

Системи телемедицини потребують сервісів аутентифікації, які є достатньо сильними для забезпечення конфіденційності даних, а також приватності працівників та пацієнтів [18].

Щоби надати користувачам доступ до деяких його ресурсів для загального доступу розділах, або вам необхідно персоніфікувати якусь інформацію, наприклад програми для завантаження, новини, то вам необхідний модуль авторизації. Завдання даного модуля полягає в реєстрації користувачів, визначенні імені користувача і аутентифікації користувача на кожній сторінці.

Ідея даного модуля полягає у виділенні користувачеві якогось маркера доступу (авторизації) після введення імені та правильного пароля. Після успішної авторизації, для того, щоб з'ясувати, що за користувач зайшов на сторінку, на ній виробляється аутентифікація користувача (на основі виданого маркера), після чого логіка системи дозволяє визначити всю необхідну інформацію про користувача: рівні доступу, активність і іншу додаткову інформацію. Маркер доступу містить в собі зашифровану комбінацію імені та пароля користувача і зберігається протягом часу сеансу на робочої станції [19].

Модуль авторизації виявляє не криптостійкі паролі і повідомляє про це користувача. Інформація про користувачів зберігається в базі даних, відкритим текстом паролі не зберігається, структура таблиць міститься в самому модулі авторизації.

Авторизація відбувається після успішної автентифікації вашої особи, що надає вам повний доступ до таких ресурсів, як інформація, файли, бази даних, кошти тощо. Однак авторизація перевіряє ваші права на надання доступу до ресурсів лише після визначення можливості системи і до якої міри. Іншими словами, авторизація – це процес, який визначає, чи має аутентифікований користувач доступ до конкретних ресурсів. Хорошим прикладом цього є перевірка і підтвердження ідентифікатора та паролів співробітників через аутентифікацію, наступним кроком буде визначення того, який працівник має доступ до якого поверху, і це здійснюється через авторизацію.

Доступ до системи захищений автентифікацією та авторизацією, і вони часто використовуються спільно один з одним. Хоча в обох є різні концепції, вони мають вирішальне значення для інфраструктури веб-сервісу, особливо коли йдеться про надання доступу до системи. Розуміння кожного терміну є дуже важливим і ключовим аспектом безпеки [20]. Принцип аутентифікації зображено на рисунку 2.1.

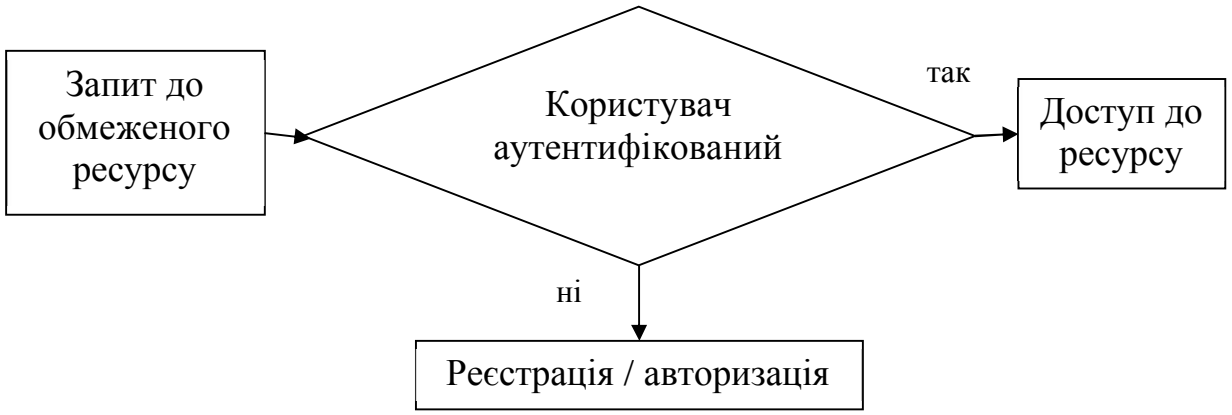

Рисунок 2.1 – Основний принцип аутентифікації

Аутентифікація – це перевірка ваших облікових даних, таких як ім'я користувача або ідентифікатор користувача та пароль, щоб підтвердити вашу особу. В результаті чого система перевіряє, чи є ви тим, що ви заявляєте, що використовуєте свої облікові дані. У публічних або приватних мережах система аутентифікує ідентифікацію користувача за допомогою паролів для входу [11].

Зазвичай аутентифікація здійснюється ім'ям користувача та паролем, хоча існують інші способи аутентифікації [21].

Фактори аутентифікації визначають багато різних елементів, які система використовує для перевірки своєї ідентичності, перш ніж надавати індивідуальний доступ до будь-якого ресурсу. Ідентичність людини може визначатися тим, що людина знає, і коли йдеться про безпеку, необхідно перевірити принаймні два або всі три фактори автентифікації, щоб дати комусь дозвіл на систему. На основі рівня безпеки фактори автентифікації можуть відрізнятися від одного з наступних:

– однофакторна аутентифікація;

– двофакторна аутентифікація;

– багатофакторна аутентифікація.

Однофакторна аутентифікація – найпростіша форма методу аутентифікації, яка вимагає пароля для надання користувачеві доступу до певної системи, наприклад, телемедицини. Користувач може запитувати доступ до системи, використовуючи лише одну з облікових даних для підтвердження своєї ідентичності. Наприклад, лише запит пароля на ім'я користувача може бути способом перевірки облікових даних для входу з використанням однофакторної автентифікації [22].

Двофакторна аутентифікація вимагає двоступеневого процесу верифікації, який вимагає не тільки ім'я користувача та пароля, але й інформацію, яку знає лише користувач. Використовуючи ім'я користувача та пароль разом із конфіденційною інформацією, хакерам набагато важче вкрасти цінні та особисті дані [23].

Багатофакторна аутентифікація – найдосконаліший метод аутентифікації, який вимагає двох або більше рівнів безпеки від незалежних категорій аутентифікації для надання користувачеві доступу до системи [24]. Ця форма аутентифікації використовує фактори, які не залежать один від одного для усунення будь-якого впливу даних. У фінансових організаціях, банках і правоохоронних органах часто використовують багатофакторну аутентифікацію.

Іноді не потрібно ідентифікувати користувача, а досить лише виконання процедури аутентифікації. Це відбувається у випадку, коли необхідно підтвердити зареєстрованого користувача при здійсненні дій, що вимагають додаткового захисту, наприклад запиту до бази даних.

У випадку якщо користувач не є зареєстрований у системі, надається можливість зареєструватися.

Користувач заходить на веб-базовану систему, відкриває форму реєстрації, вводить персональні дані, натискає кнопку реєстрації, якщо всі дані введено вірно, то створюється новий користувач, якщо ж ні то процедура введення даних повторюється.

У БР.КСМ. 07103/15.00.00.000 С2 наведено блок-схему алгоритму реєстрації користувачів.

В основі міжмережевих взаємодій по протоколам TCP(Transmission Control Protocol) і UDP(User Datagram Protocol) лежать сокети. В .NET сокети представлені класом System.NET.Sockets.Socket, який надає низькорівневий інтерфейс для прийому і відправки повідомлень по мережі.

Сокет – це один кінець двостороннього каналу зв'язку між двома програмами, що працюють в мережі. Поєднуючи разом два сокета, можна передавати дані між різними процесами (локальними або віддаленими) [25].

Реалізація сокетов забезпечує інкапсуляцію протоколів мережевого і транспортного рівнів [26].

Спочатку сокети були розроблені для UNIX в Каліфорнійському університеті в Берклі. Перш ніж ресурс використовувати, його потрібно

відкрити, задавши відповідні дозволи та інші параметри. Як тільки ресурс відкритий, з нього можна зчитувати або в нього записувати дані. Після використання ресурсу користувач повинен викликати метод Close (), щоб подати сигнал операційній системі про завершення його роботи з цим ресурсом на рисунку 2.3.

Рисунок 2.3 – Схема роботи серверного сокета

Інтерфейс IPC(Inter-Process Communication) для взаємодії між різними процесами побудований поверх методів введення-виведення. Це полегшує відправку та отримання даних. Кожен цільовий об'єкт задається адресою сокета, отже, ця адреса має бути вказана в клієнті, щоб встановити з'єднання з метою.

Типи сокетів зображено на рисунку 2.4.

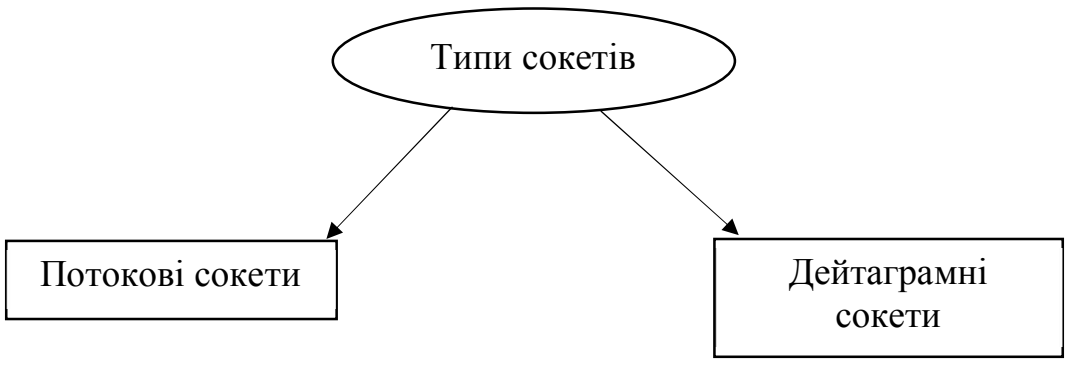

Рисунок 2.4 – Типи сокетів

Потоковий сокет – це сокет з встановленим з'єднанням, що складається з потоку байтів, який може бути двонаправленим. Через цю кінцеву точку додаток може і передавати, і отримувати дані.

Потоковий сокет гарантує виправлення помилок, обробляє доставку і зберігає послідовність даних. На нього можна покластися в доставці упорядкованих даних.

Потоковий сокет також підходить для передачі великих обсягів даних, оскільки накладні витрати, пов'язані з встановленням окремого з'єднання для кожного повідомлення, що відправляється, може виявитися неприйнятним для невеликих обсягів даних. Потокові сокети досягають цього рівня якості за рахунок використання протоколу TCP. TCP забезпечує надходження даних на іншу сторону в потрібній послідовності і без помилок.

Для цього типу сокетів шлях формується до початку передачі повідомлень. Тим самим гарантується, що обидві сторони, які взаємодіють між собою приймають і відповідають на повідомлення. Якщо додаток відправляє одержувачу два повідомлення, то гарантується, що ці повідомлення будуть отримані в тій же послідовності.

Однак, окремі повідомлення можуть дробитися на пакети, і способів визначити межі записів не існує. При використанні TCP цей протокол бере на себе розбиття переданих даних на пакети відповідного розміру, відправку їх в мережу і складання їх на іншій стороні. Додаток знає тільки, що воно відправляє на рівень TCP певне число байтів і інша сторона отримує ці байти. У свою чергу TCP ефективно розбиває ці дані на пакети відповідного розміру, отримує ці пакети на іншій стороні, виділяє з них дані і об'єднує їх разом [27].

Дейтаграмні сокети іноді називають сокетами без організації з'єднань. Ніякого явного з'єднання між ними не встановлюється, повідомлення відправляється вказаному сокету і, відповідно, може виходити від зазначеного сокета.

Потокові сокети в порівнянні з дейтаграмними дійсно дають більш надійний метод, але для деяких додатків накладні витрати, пов'язані з

установкою явного з'єднання, неприйнятні (наприклад, сервер часу доби, що забезпечує синхронізацію часу для своїх клієнтів). Зрештою на встановлення надійного з'єднання з сервером
потрібен час, яке просто вносить затримки в обслуговування, і завдання серверного додатка не виконується. Для скорочення накладних витрат потрібно використовувати дейтаграмні сокети.

Використання дейтаграмним сокетов вимагає, щоб передачею даних від клієнта до сервера займався UDP. У цьому протоколі на розмір повідомлень накладаються деякі обмеження, і на відміну від потокових сокетів, які вміють надійно відправляти повідомлення сервера-адресату, дейтаграмні сокети надійність не забезпечують. Якщо дані загубилися десь в мережі, сервер не повідомить про помилки.

На основі дослідження основних етапів розробки баз даних було розроблено структуру бази даних для функціонування системи телемедицини та описано структуру таблиць, з яких складається база даних.

Отже, у даному підрозділі на основі результатів дослідження основних принципів аутентифікації та авторизації розроблено алгоритм авторизації у системі телемедицини. В результаті описано послідовність дій користувача під час авторизації в системі.

2.2 Розробка структури бази даних

Головним завданням бази даних – гарантоване збереження значних обсягів інформації та надання доступу до неї користувачеві або ж прикладній програмі.

Бази даних можна класифікувати за різними ознаками. Таким чином база даних складається з двох частин:

– збереженої інформації;

– системи управління нею.

База даних – упорядкований набір логічно взаємопов'язаних даних, що використовується спільно, та призначений для задоволення інформаційних потреб користувачів.

Історично системи управління базами даних орієнтувалися на вирішення завдань, пов'язаних у першу чергу з транзакційною обробкою структурованої інформації. Безумовно, найкращим, перевіреним часом рішенням тут була і залишається реляційна модель СУБД. Однак в останні роки область застосування баз даних незмінно розширювалася. З одного боку, потрібно керувати більш широким набором форматів даних, переходячи до вирішення спільних проблем управління корпоративною інформацією. З іншого – саме СУБД(Database Management System) беруть на себе основні функції інтеграції даних і додатків корпоративних систем.

Класифікація БД за ступенем розподіленості:

– централізовані (зосереджені);

– розподілені.

Окреме місце в теорії та практиці займають просторові, тимчасові, або темпоральні і просторово-часові бази даних.

Ієрархічні бази даних можуть бути представлені як дерево, що складається з об'єктів різних рівнів. Верхній рівень займає один об'єкт, другий – об'єкти другого рівня. Мережеві бази даних подібні до ієрархічних, за винятком того, що в них є покажчики в обох напрямках, які з'єднують споріднену інформацію.

Незважаючи на те, що ця модель вирішує деякі проблеми, пов'язані з ієрархічною моделлю, виконання простих запитів залишається досить складним процесом. Також, оскільки логіка процедури вибірки даних залежить від фізичної організації цих даних, то ця модель не є повністю незалежною від програми.

Іншими словами, якщо необхідно змінити структуру даних, то потрібно змінити і додаток. За технологією обробки даних бази даних поділяються на централізовані й розподілені. Централізована база даних зберігається у пам'яті однієї обчислювальної системи. Якщо ця обчислювальна система є компонентом

мережі ЕОМ (Електро́нна обчи́слювальна маши́на), можливий розподілений доступ до такої бази. Такий спосіб використання баз даних часто застосовують у локальних мережах ПК.

Розподілена база даних складається з декількох, можливо пересічних або навіть дублюючих одна одну частин, які зберігаються в різних ЕОМ обчислювальної мережі. Робота з такою базою здійснюється за допомогою системи управління розподіленою базою даних (СУРБД).

За способом доступу до даних бази даних поділяються на:

– бази даних з локальним доступом;

– бази даних з віддаленим (мережевим) доступом.

Системи централізованих баз даних з мережевим доступом припускають різні архітектури подібних систем:

– файл-сервер;

– клієнт-сервер.

Архітектура систем БД з мережевим доступом передбачає виділення однієї з машин мережі в якості центральної (сервер). На такій машині зберігається спільно використовувана централізована БД. Усі інші машини мережі виконують функції робочих станцій, за допомогою яких підтримується доступ користувальницької системи до централізованої бази даних. Файли бази даних відповідно до призначених для користувача запитів передаються на робочі станції, де в основному і проводиться обробка [15]. При великій інтенсивності доступу до одних і тих же даних продуктивність інформаційної системи падає.

У концепції клієнт-сервер мається на увазі, що крім зберігання централізованої бази даних центральна машина (сервер бази даних) повинна забезпечувати виконання основного обсягу обробки даних.

Ієрархічна структура являє собою сукупність елементів, пов’язаних між собою за визначеними правилами. В ієрархічній моделі дані подаються у вигляді деревовидної (ієрархічної) структури. Вона зручна для роботи з ієрархічно упорядкованою інформацією і громіздка для інформації зі складними логічними зв’язками.

Специфікою архітектури клієнт-сервер є використання мови запитів SQL. Алгоритм роботи бази даних зображено на рисунку 2.5.

Рисунок 2.5 – Схематичне зображення алгоритму роботи бази даних.

Основні типи для зберігання даних в базі даних:

– текстовий – використовується для зберігання звичайного тексту;

– числовий – призначений для зберігання дійсних чисел;

– дата/час – для зберігання дати та часу;

– грошовий – для зберігання грошових сум;

– логічний – для зберігання логічних даних (типу «так», «ні»);

– поле MEMO – спеціальний тип для зберігання великих обсягів тексту;

– гіперпосилання – спеціальне поле для зберігання адреси URL;

– поле об’єкту OLE – призначений для зберігання об’ємів OLE,

– наприклад мультимедійних;

– рахівник – використовується для запису чисел.

Класифікація баз даних зображено на рисунку 2.6.

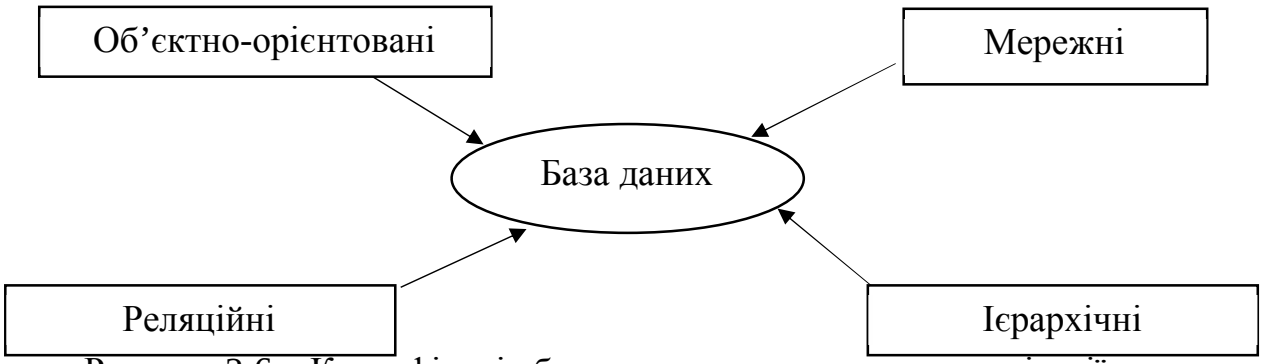

Рисунок 2.6 – Класифікація баз даних за структурою організації даних

У БР.КСМ. 07103/15.00.00.000 С1 наведено структурну схему розробленої бази даних.

Запит на дані, який видається клієнтом (робочою станцією), породжує пошук і вилучення даних на сервері. Витягнуті дані транспортуються по мережі від сервера до клієнта.

Позитивною якістю мережних моделей даних є можливість їх ефективної реалізації показників витрат пам’яті і оперативності. Недоліком мережної моделі даних є висока складність і жорсткість схеми бази даних, яка побудована на її основі.

Мережна модель означає подання даних у вигляді довільного графа. У мережній структурі кожний елемент може бути пов’язаний будь-яким іншим елементом.

Ієрархічна структура являє собою сукупність елементів, пов’язаних між собою за визначеними правилами. В ієрархічній моделі дані подаються у вигляді деревовидної (ієрархічної) структури. Вона зручна для роботи з ієрархічно

упорядкованою інформацією і громіздка для інформації зі складними логічними зв’язками.

Реляційна модель даних подає дані у вигляді множини таблиць. Таблиця являє собою набір рядків та стовпців, де рядки називаються записами, а кожний елемент стовпчика – полем, або пояснювальним написом. Подання інформації у вигляді таблиці, яка складається зі стовпців, розташованих у певному порядку зліва направо, в математиці називається відношенням. Звідси і назва моделі – реляційна. Структуровані таким чином дані, можуть зберігатися у вигляді баз даних. У кожному рядку таблиці міститься дані про один об’єкт даних (наприклад, про хвороби пацієнтів, інформація про лікарів тощо). У базі даних кожна таблиця визначається сукупністю її стовпців і рядків.

Структура бази даних визначається встановленням зв’язків між таблицями. Стовпець відповідає певному елементу даних – атрибуту, який є найпростішою структурою даних і відображає властивість певної суті. Наприклад, прізвище, ім’я, по-батькові пацієнта, номер історії хвороби, оцінка стану здоров’я тощо є атрибутами суті «пацієнт». Кожний стовпець повинен мати ім’я відповідного елемента даних (атрибута).

Отже, реляційні таблиці будують за певним критерієм структурування. Необхідність такого структурування зумовлена прагненням систематизувати величезні масиви даних й автоматизувати пошук і селекцію їх компонентів.

У результаті дослідження основних етапів розробки баз даних було розроблено структуру бази даних для функціонування системи телемедицини з яких складається база даних.

2.3 Алгоритм обміну повідомленнями на основі сокетів

Існує два види сокетів – серверний та клієнтський. Наприклад, у випадку розробки мережевого чату кожний користувач підключається до сервера та

відправляє йому повідомлення. Сервер зберігає список підключень та відправляє усім користувачам(клієнтам) повідомлення, що прийшло від одного з користувачів.

SignalR Core представляє бібліотеку від компанії Microsoft, яка призначена для створення додатків, що працюють в режимі реального часу. Зокрема, її можна використовувати разом з ASP.NET Core. SignalR використовує двонаправлений зв'язок для обміну повідомленнями між клієнтом і сервером, завдяки чому сервер може відправляти в режимі реального часу всім клієнтам деякі дані.

Для обміну повідомленнями між клієнтом і сервером SignalR використовує ряд механізмів:

– websockets;

– server-side events;

– long polling.

Виходячи з можливостей клієнта і сервера інфраструктура SignalR вибирає найкращий механізм для взаємодії. Зокрема, найбільш оптимальним є WebSockets, відповідно якщо і клієнт, і сервер дозволяють використовувати цей механізм, то взаємодія йде через WebSockets.

Встановивши одного разу з’єднання, клієнт та сервер мають можливість вільно обмінюватись фреймами даних між собою у будь-якому напрямі без обмежень на формат даних (рисунок 2.5). При цьому TCP-з’єднання залишається відкритим. Життєвий циклу клієнт-серверного з’єднання на основі веб-сокетів зображено на рисунку 2.7.

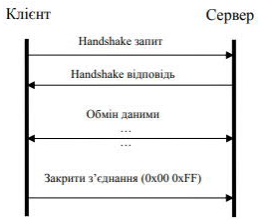

Рисунок 2.7 – Схема життєвого циклу клієнт-серверного з’єднання на основі веб-сокетів

Після встановлення з’єднання сторони можуть вільно обмінюватись даними, поки сеанс не завершиться з ініціативи однієї зі сторін або не буде отримано

статусу помилки. Обмін інформацією здійснюється за допомогою послідовності фреймів у спеціальному форматі і може виконуватись лише в межах відкритого сеансу.

Згідно з моделлю взаємодії відкритих систем (OSI), веб-сокети, як і HTTP протокол, належать до протоколів прикладного рівня. Хоча, технологічно, веб-сокети – це надбудова над HTTP протоколом, однак їх треба розглядати як окремий незалежний протокол. Клієнт-серверна взаємодія на основі веб-сокетів зображено на рисунку 2.8.

Рисунок 2.8 – Схема клієнт-серверної взаємодії на основі веб-сокетів

Для реалізації клієнт-серверної взаємодії веб-сокети використовують з’єднання на основі TCP та реалізують безпечний та прозорий програмний інтерфейс для їх використання.

У даному підрозділі проаналізовано сучасні підходи до реалізації асинхронної клієнт-серверної взаємодії веб-програм обміну повідомленнями.

Розглянуто основні алгоритми двонаправленої взаємодії між клієнтом та сервером.

Веб-сокети є однією з найперспективніших веб-технологій, яку вже зараз використовують багато розробників. Вона відмінно підходить для взаємодії в режимі реального часу, в тому числі в системах обміну повідомленнями.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТЕЛЕМЕДИЧНОЇ СИСТЕМИ HIAMS НА ОСНОВІ СОКЕТІВ

3.1 Структура серверної частини програмного модуля

Більшість інформації обробляється на серверній частині, зокрема вибірка інформації з бази даних, пошук по базі даних за ключовими словами, вибірка інформації із списку, збереження записів користувачів із форми зворотного зв'язку тощо. Структуру серверної частини системи обміну повідомленнями наведено на рисунку 3.1.

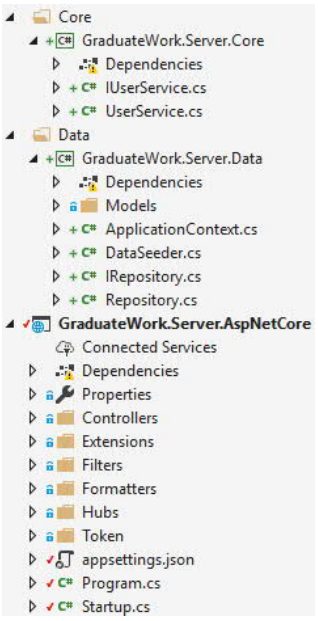

Рисунок 3.1 – Структура серверної частини розробленої системи обміну повідомленнями

Структура програмної частини поділяється на три рівні:

– Presentation Layer.

– Business Logic Layer.

– Data Access Layer.

Presentation layer (рівень представлення) – це рівень, з яким безпосередньо взаємодіє користувач. Рівень включає компоненти для користувача інтерфейсу,

механізм отримання запитів від користувача. А також моделі представлень, контролери, об'єкти контексту запиту.

Проект «GraduateWork.Server.AspNetCore» відповідає за рівень представлення. У серверній частині програмного модуля створено декілька контролерів, які отримують дані з клієнтської частини у вигляді запиту, обробляють ці дані та посилають результат обробки у форматі JSON.

За допомогою JSON можна пов'язувати між собою масиви та об'єкти, створюючи складні структури даних. JSON не залежить від мови програмування, він більш зручний і легше обробляється. Рівень представлення зображено на рисунку 3.2.

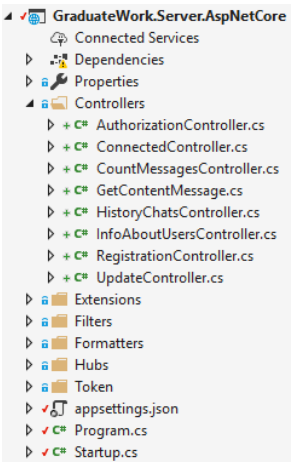

Рисунок 3.2 – Рівень представлення у проекті

Business layer (рівень бізнес логіки) містить набір компонентів, які відповідають за обробку отриманих від рівня представлення даних, реалізує всю необхідну логіку додатка, все обчислення, взаємодіє з базою даних і передає рівнем подання результат обробки. Бібліотека «GraduateWork.Server.Core» відповідає за рівень бізнес логіки. Рівень бізнес логіки зображено на рисунку 3.3.

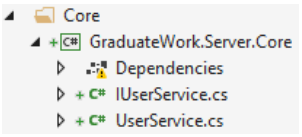

Рисунок 3.3 – Рівень бізнес логіки у проекті

Data Access layer (рівень доступу до даних) зберігає моделі, що описують використовувані сутності, також тут розміщуються специфічні класи для роботи з різними технологіями доступу до даних, наприклад, клас контексту даних Entity Framework. Тут також зберігаються репозиторії, через які рівень бізнес-логіки взаємодіє з базою даних. Також на даному рівні використовується патерн «Репозиторій», який дозволяє абстрагуватися від конкретних підключень до джерел даних, з якими працює програма, і є проміжною ланкою між класами.

Рівень доступу до даних зображено на рисунку 3.4.

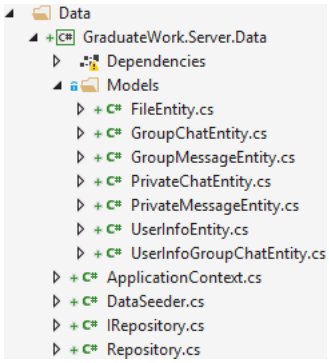

Рисунок 3.4 – Рівень доступу до даних

ASP.NET Core є крос-платформленним, високопродуктивним, з відкритим вихідним кодом для побудови сучасних, хмарних, підключених до Інтернету програм.

Одним з характерних моментів платформи ASP.NET Core є застосування патерну MVC(модель-вигляд-контролер).

У MVC моделі представлені двома основними типами:

– моделі уявлень, які використовуються уявленнями для відображення і передачі даних;

– моделі домену, які описують логіку управління даними.

Модель може містити дані, зберігати логіку управління цими даними. У той же час модель не повинна містити логіку взаємодії з користувачем і не має визначати механізм обробки запиту. Крім того, модель не повинна містити логіку відображення даних в поданні.

Вигляд (view) відповідає за візуальну частину або призначений для користувача інтерфейс, нерідко html-сторінка, через який користувач взаємодіє з додатком. Також уявлення може містити логіку, пов'язану з відображенням даних. У той же час вигляд не повинен містити логіку обробки запиту користувача або управління даними.

Контролер (controller) представляє центральний компонент MVC, який забезпечує зв'язок між користувачем та програмою, поданням і сховищем даних. Він містить логіку обробки запиту користувача. Контролер отримує введені користувачем дані і обробляє їх. І в залежності від результатів обробки відправляє користувачеві певний висновок, наприклад, у вигляді подання, наповненого даними моделей.

Розробка клієнтської сторони у даному програмному модулі здійснюється за допомогою технології WPF(Windows Presentation Foundation).

Однією з важливих особливостей даної технології є використання мови декларативною розмітки інтерфейсу XAML. Ви можете створювати насичений графічний інтерфейс, використовуючи або декларативне оголошення інтерфейсу, або код на керованих мовах C # і VB.NET, або поєднувати і те, і інше.

XAML (eXtensible Application Markup Language) – мова розмітки, яка використовується для ініціалізації об'єктів в технологіях на платформі .NET.

3.2 Структура клієнтської частини програмного модуля

Клієнтська частина реалізовує патерн MVVM (Model-View-ViewModel). Даний патерн дозволяє відокремити логіку додатку від візуальної частини (представлення). Даний патерн є архітектурним рішенням, тобто він задає загальну архітектуру програми.

У додатку А наведено лістинг файлу «ConnectionManager.cs» клієнтської частини програмного модуля.

Структура клієнтської частини системи обміну повідомленнями наведено на рисунку 3.5.

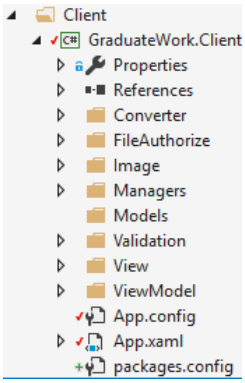

Рисунок 3.5 – Структура клієнтської частини системи обміну повідомленнями

Шаблон MVVM має три основних компоненти.

Модель (model), яка представляє бізнес-логіку додатка. Представлення (view) - графічний інтерфейс (вікна, списки, кнопки). Рівень представлення зображено на рисунку 3.6.

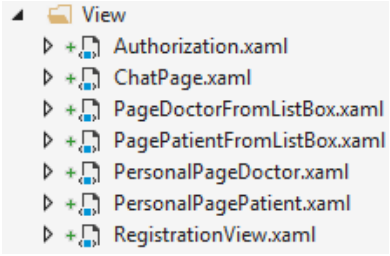

Рисунок 3.6 – Рівень представлення

Модель-представлення (view-model) пов'язує модель та представлення через механізм прив'язки даних. Якщо в моделі змінюються значення властивостей автоматично відбувається зміна відображуваних даних в представленні, хоча безпосередньо модель і представлення не пов'язані. View Model також містить логіку по отриманню даних з моделі, які потім передаються в уявлення. І також ViewModel визначає логіку по оновленню даних в моделі.

Рівень модель-представлення зображено на рисунку 3.7.

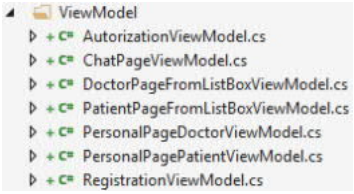

Рисунок 3.7 – Рівень модель-представлення

Запустивши програмний модуль користувач матиме змогу авторизуватися в програмі ввівши логін та пароль. Вікно авторизації зображено на рисунку 3.8.

Рисунок 3.8 – Вікно авторизації користувача

У випадку введення некоректних даних отримаємо повідомлення про це. Повідомлення про некоректно введені дані зображено на рисунку 3.9.

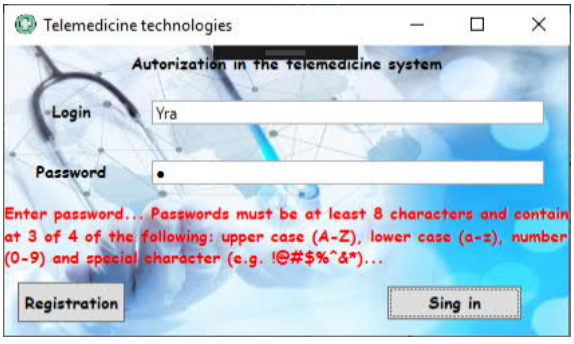

Рисунок 3.9 – Повідомлення про некоректно введені дані

У випадку, якщо користувач ще не зареєструвався у системі йому надається така можливість нажавши кнопку «Registration». В результаті відкриється форма реєстрації та можливість вводу особистих даних користувача.

Вікно реєстрації зображено на рисунку 3.10.

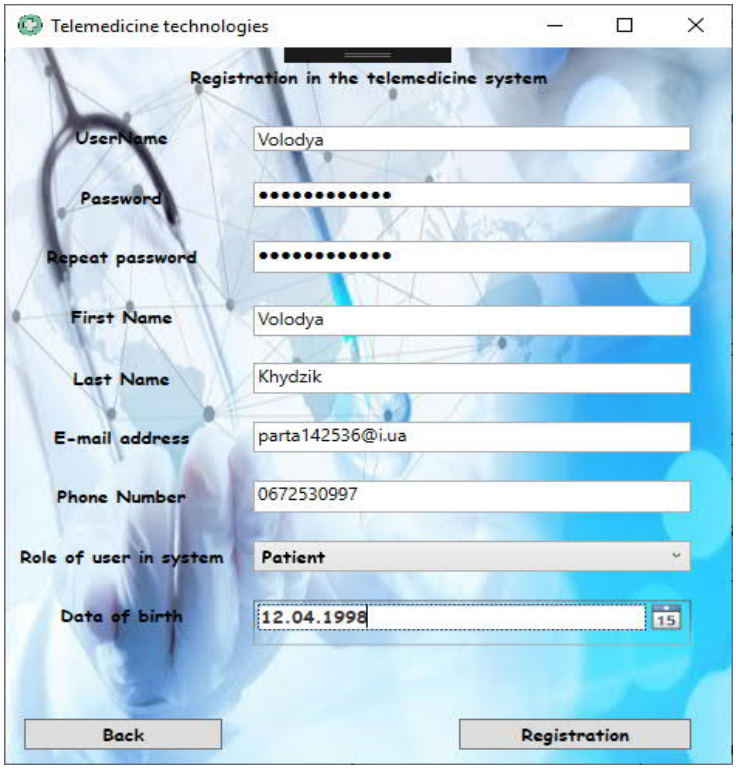

Рисунок 3.10 – Вікно реєстрації

Нажавши кнопку «Registration» отримаємо повідомлення про успішну реєстрацію та можливість ввести логін та пароль. Повідомлення про успішну реєстрацію зображено на рисунку 3.11.

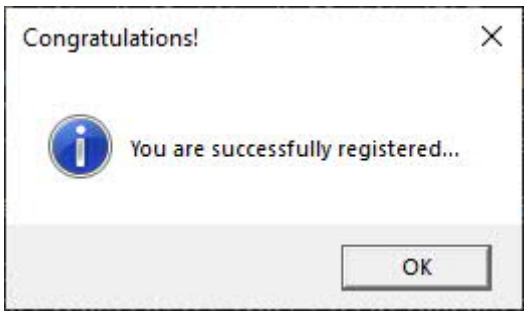

Рисунок 3.11 – Повідомлення про успішну реєстрацію

У випадку введення некоректних даних отримаємо повідомлення про це. Повідомлення некоректних даних при реєстрації зображено на рисунку 3.12.

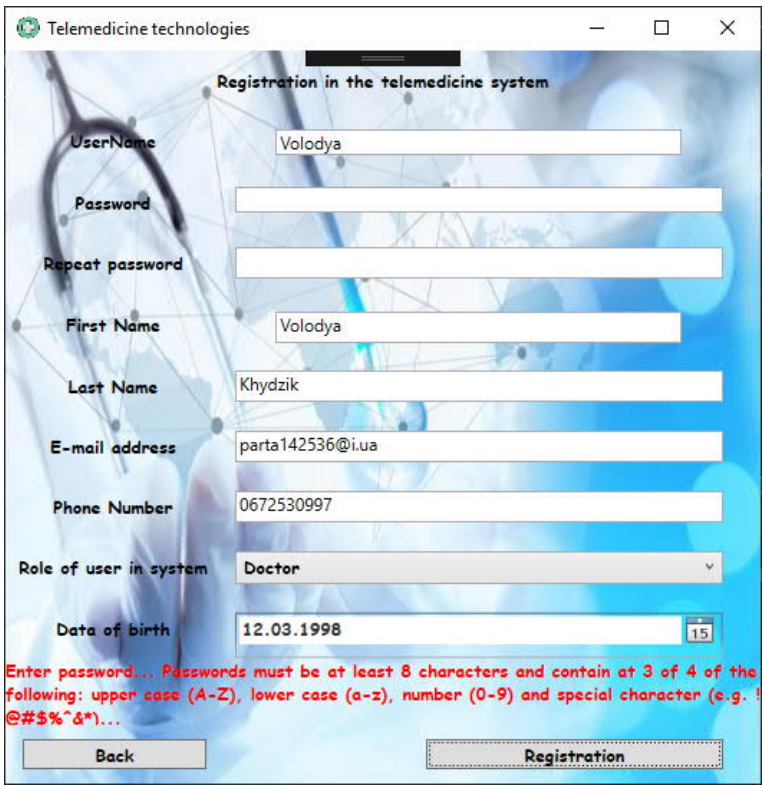

Рисунок 3.12 – Введення некоректних даних при реєстрації

У випадку введення коректних даних при авторизації відкриється головний інтерфейс персональної сторінки в залежності від ролі користувача. Персональна сторінка користувача з роллю «Doctor» зображено на рисунку 3.13.

Рисунок 3.13 – Персональна сторінка користувача з роллю «Doctor»

На персональній сторінці користувача зображена особиста інформація, а саме:

– First Name.

– Last Name.

– Data Of Birth.

– Email.

– Phone Number.

– Список докторів та список пацієнтів.

Нажавши на кнопку «Message» відкриється вікно та можливість обмінюватися повідомленнями.

Вибравши пацієнта із списку пацієнтів відкриється персональна сторінка вибраного пацієнта. Персональна сторінка користувача з роллю «Patient» для користувача з роллю «Doctor» зображено на рисунку 3.14.

Рисунок 3.14 – Персональна сторінка користувача з роллю «Patient» для користувача з роллю «Doctor»

До того ж, передбачено, що користувач з роллю «Doctor» має право змінювати історію хвороби. Ввівши діагнос хвороби та нажавши кнопку «Add to history of the disease» отримаємо повідомлення про редагування історії хвороби пацієнта зображено на рисунку 3.15.

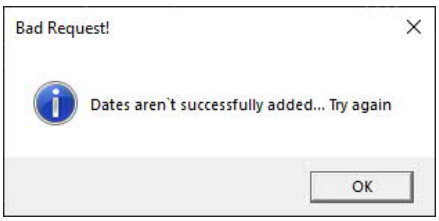

Рисунок 3.15 – повідомлення про редагування історії хвороби пацієнта

Та побачимо зміни історії хвороби на інтерфейсі персональної сторінки користувача з роллю «Patient» зображено на рисунку 3.16.

Рисунок 3.16 – Редагування історії хвороби на інтерфейсі персональної сторінки користувача з роллю «Patient»

Також на даному етапі функціоналу передбачено можливість написати користувачеві нажавши на кнопку «Write a message» зображено на рисунку 3.17.

Рисунок 3.17 – Інтерфейс вікна обміну повідомленнями з користувачем Fedir Khydzik

Вибравши користувача із списку «Doctors» відкриється персональна сторінка вибраного користувача із роллю «Doctor» зображено на рисунку 3.18.

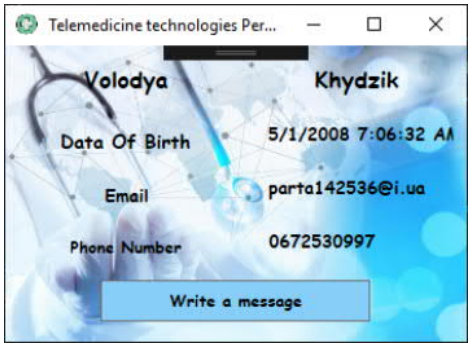

Рисунок 3.18 – Персональна сторінка вибраного користувача з списку із роллю «Doctor»

На даному етапі програмного модуля також передбачено можливість обміну повідомленнями. Інтерфейс обміну повідомленнями зображено на рисунку 3.19.

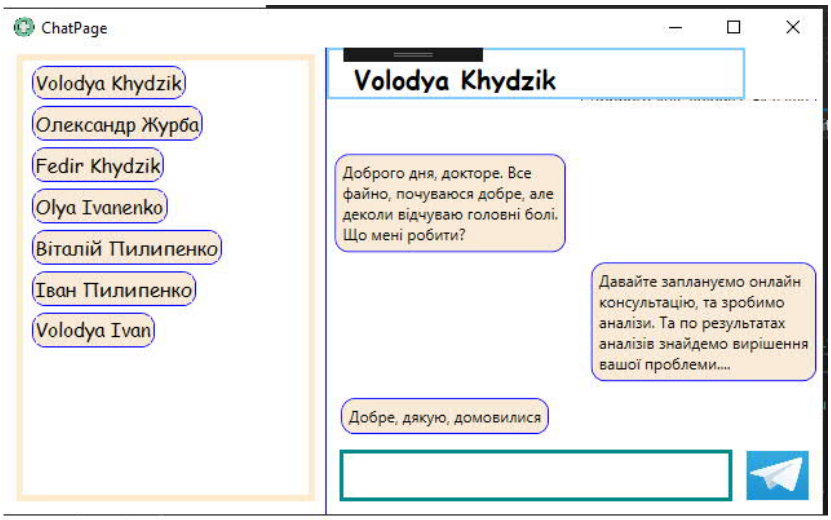

Рисунок 3.19 – Інтерфейс обміну повідомленнями з користувачем «Volodya Khydzik»

Авторизувавшись у системі з роллю «Patient» відкриється персональна сторінка користувача з роллю «Patient» зображено на рисунку 3.20.

Рисунок 3.20 – Персональна сторінка користувача з роллю «Patient»

Для користувача з даною роллю передбачено певні обмеження. Насамперед це те, що пацієнт має можливість обмінюватися повідомленнями лише з користувачами з роллю «Doctor» вибравши потрібного користувача із списку «Doctors» зображено на рисунку 3.21.

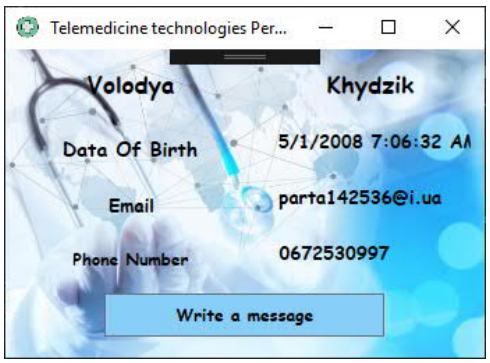

Рисунок 3.21 – Персональна сторінка вибраного користувача з списку із роллю «Doctor»

Також передбачено те, що користувач з роллю «Patient» може лише читати свою історію хвороби, а не змінювати її, що дозволено користувачам з роллю «Doctor».

3.3 Тестування програмного модулю

Модульне тестування – метод тестування програмного забезпечення, який полягає в окремому тестуванні кожного модуля коду програми. Модулем називають найменшу частину програми, яка може бути протестованою. Зазвичай unit-тести застосовують для того, щоб упевнитися, що код відповідає вимогам архітектури та має очікувану поведінку.

Модульне тестування проводиться викликаючи код, який необхідно перевірити за підтримки середовищ розробки. Всі знайдені дефекти, як правило виправляються в коді без формального їх опису в системі багів (Bug Tracking System). Один з найбільш ефективних підходів до модульного тестування – це підготовка автоматизованих тестів до початку основного кодування програмного забезпечення. Це називається розробка від тестування (test-driven development) або підхід тестування спочатку (test first approach). При цьому підході створюються і інтегруються невеликі шматки коду, навпроти яких запускаються тести, написані до початку кодування. Розробка ведеться до тих пір поки всі тести не будуть успішними.

Для автоматизації процесу розробники зазвичай звертаються до фреймворків юніт-тестування.

Фреймворки тестування:

– MS Test – фреймворк юніт-тестування від компанії Microsoft, який за замовчуванням включений в середовище розробки Visual Studio.

– NUnit – портований фреймворк з JUnit для платформи .NET.

– xUnit.net – ще один фреймворк тестування від творців NUnit для платформи .NET.

В результаті розробки системи було вирішено використовувати xUnit.net фреймворк та проведено тестування всіх модулей серверної частини.

Результат роботи unit-тесту в середовищі Visual Studio наведено на рисунку 3.8.

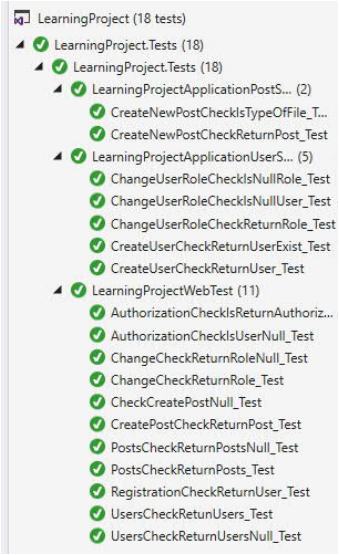

Рисунок 3.22 – Результати покриття unit-тестами серверну частину програмного модуля

Код тесту для перевірки некоректності авторизації такий:

```
    [Fact]
    public async Task AuthorizationCheckIsUserNull_Test() {
    User userIdent = null;
    var user = new LoginModel { UserName = "admin", Password =
"admin1" };
    var mock1 = new Mock<IUserService>();
    var mock2 = new Mock<ITokenFormation>();
    mock1.Setup(repo => repo.GetUser(user.UserName)).Returns(async
() => { return userIdent; });
    var authorizationController = new AuthorizationController
(mock1.Object, mock2.Object);
    var result = authorizationController.Authorization(user);
    Exception ex = await Assert.ThrowsAsync<Exception>(() =>
securityController.Authorization(user));
    Assert.Equal("Invalid username or password", ex.Message); }
```

Код методу тесту для перевірки правильності модуля реєстрації такий:

```
    [Fact]
    public void RegistrationCheckReturnUser_Test() {
    var registrationModel = new RegistrationUserModel { UserName =
"volodya", Password = "volodya1"};
    var mock = new Mock<IUserService>();
    User user = new User { Id = 1, UserName = "iVova", Password =
"Apple1"};
```

```
     mock.Setup(repo=>repo.CreateUserAsync(registrationModel.).Retu
rns(async () => { return user; });
     var registerController = new
RegistrationController(mock.Object);
     var result = registerController. RegistrationAsync
(registrationModel);
     Assert.IsType<User>(result.Result); }
```

Крім власне фреймворків для створення і проведення юніт-тестів при тестуванні часто бувають корисні такі фреймворки, які дозволяють імітувати або емулювати якусь функціональність або створювати мок-об'єкти. Подібних фреймворків існує безліч, і одним з найпопулярніших є Moq.

Розроблені тести дозволяють пересвідчитись у працездатності програмного модуля обміну повідомленнями.

4 ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ РОЗРОБКИ ПРОГРАМНОГО ЗАСОБУ

4.1 Розрахунок витрат на виконання проектного рішення

4.1.1 Розрахунок витрат на розробку програмного забезпечення

Витрати на розробку і впровадження програмних засобів (К) включають:

$$K = K_1 + K_2, \tag{4.1}$$

де K_1 – витрати на розробку програмних засобів, грн;

K_2 – витрати на відлагодження і дослідну експлуатацію програми рішення задачі на комп'ютері, грн.

Витрати на розробку програмних засобів включають:

– витрати на оплату праці розробників ($B_{ОП}$);

– витрати на відрахування у спеціальні державні фонди ($B_ф$);

– витрати на покупні вироби ($П_В$);

– витрати на придбання спецобладнання для проведення експериментальних робіт ($O_б$);

– накладні витрати (H);

– інші витрати ($I_В$).

4.1.2 Розрахунок витрат на оплату праці та соціальні заходи

Витрати на оплату праці включають заробітну плату (ЗП) всіх категорій працівників, безпосередньо зайнятих на всіх етапах проектування. Розмір ЗП

обчислюється на основі трудомісткості відповідних робіт у людино-днях та середньої ЗП відповідних категорій працівників.

У розробці програмного забезпечення задіяні спеціалісти-розробники, а саме: керівник проекту; студент-дипломник; консультант техніко-економічного розділу.

Вихідні дані для розрахунку витрат на оплату праці наведено у таблиці 4.1.

Таблиця 4.1 – Вихідні дані для розрахунку витрат на оплату праці

№ п/п	Посада виконавців	Місячний оклад, грн.
1	Керівник ДП, викладач	5950
2	Консультант техніко-економічного розділу, доцент	7293
3	Студент	1400

Витрати на оплату праці розробників проекту визначаються за формулою:

$$B_{ОП} = \sum_{i=1}^{N} \sum_{j=1}^{M} n_{ij} \cdot t_{ij} \cdot C_{ij} \ , \qquad (4.2)$$

де n_{ij} – чисельність розробників i-ої спеціальності j-го тарифного розряду, осіб;

t_{ij} – затрачений час на розробку проекту співробітником i-ої спеціальності j-го тарифного розряду, год;

C_{ij} – годинна ставка працівника i-ої спеціальності j-го тарифного розряду, грн.

Середньогодинна ставка працівника може бути розрахована за формулою:

$$C_{ij} = \frac{C_{ij}^{0}(1+h)}{РЧ_{i}}, \quad (4.3)$$

де C_{ij} – основна місячна заробітна плата розробника *i*-ої спеціальності *j*-го тарифного розряду, грн.;

h – коефіцієнт, що визначає розмір додаткової заробітної плати (при умові наявності доплат);

$РЧ_i$ – місячний фонд робочого часу працівника *i*-ої спеціальності *j*-го тарифного розряду, год. (приймаємо \168 год.).

Розрахунок витрат на оплату праці наведено у таблиці 4.2.

Таблиця 4.2 – Розрахунок витрат на оплату праці

№ п/п	Посада виконавців	Час розробки, год	Погодинна заробітна плата, грн/год.	Витрати на розробку, грн
1	Керівник ДП, викладач	20	44.27	885.4
2	Консультант техніко-економічного розділу, доцент	10	63.8	638
3	Студент	120	8.3	1000
Разом				2525.53

Величну відрахувань у спеціальні державні фонди визначають у відсотковому співвідношенні від суми основної та додаткової заробітних плат.

Згідно діючого нормативного законодавства сума відрахувань у спеціальні державні фонди складає 20,5 % від суми заробітної плати:

$$B_{ф} = \frac{2303.04 \cdot 20.5}{100} = 517.32 \text{ грн.}$$

4.1.3 Розрахунок витрат на матеріали та комплектуючі

Заглальна сума витрат на матеріальні ресурси (B_M) визначається за формулою:

$$B_M = \sum_{i=1}^{n} K_i \cdot Ц_i, \quad (4.4)$$

де K_i – витрата і-го типу матеріалу, натуральні одиниці вимірювання;

$Ц_i$ – ціна за одиницю і-го типу матеріалу, грн, і-тип матеріального ресурсу;

n – кількість типів матеріальних ресурсів.

Звідси, витрати на матеріальні ресурси дорівнюватимуть:

$В_м$ =254,1 грн.

Таблиця 4.3- Розрахунок витрат на матеріали та комплектуючі

№ п/п	Найменування купованих виробів	Одиниця виміру	Ціна, грн	Кількість купованих виробів	Сума, грн	Транспортні витрати (10% від суми)	Загальна сума, грн
1	Папір (формат А4)	уп	90	1	90,00	9,0	99,0
2	Ручка кулькова	шт	17	2	34	3,4	37.4
3	Олівець простий	шт	5	1	5	0,5	5.5
4	Диски CD-R	шт	7.0	1	7.0	0,7	7.7
5	Зошит, 96 арк	шт	15.0	1	15.0	1.5	16.5
6	Тонер для принтера	уп	80.0	1	80	8	88
Разом							254.1

4.1.4 Витрати на використання комп’ютерної техніки

Для розробки КС використовується електрообладнання, тому необхідно розрахувати витрати на електроенергію.

Витрати на використання комп’ютерної техніки включають витрати на амортизацію комп’ютерної техніки, витрати на користування програмним забезпеченням, витрати на електроенергію, що споживається комп’ютером.

Загальна сума витрат на електроенергію розраховується за формулою:

$$B_E = \sum_{i=1}^{n} P_i \cdot k_i \cdot T_i \cdot Ц \,, \qquad (4.5)$$

де P_i – паспортна потужність i-го електрообладнання, кВт;

k_i – коефіцієнт використання потужності i-го електрообладнання;

T_i – час роботи i-го устаткування за весь період розробки, год;

$Ц$ – ціна електроенергії, грн / кВт год;

i – тип електрообладнання;

n – кількість електрообладнання.

Для розробки проекту даної системи використовується один ноутбук потужністю Р = 0,5 кВт, який за весь період розробки працює 100 годин та друкуючий пристрій потужністю Р = 0,37 кВт, який працює 2 години.

Розрахунки на витрату електроенергії наведено у таблиці 4.4.

Таблиця 4.4 – Проміжні розрахунки на витрату електроенергії

Наймену-вання устаткування	Паспортна потужність, кВт	Коефіцієнт використання потужності	Час роботи обладнання для розробки, год	Ціна електро-енергії,	Сума, грн.
Ноутбук	0,5	0,98	100	0,90	44.1
Принтер	0,37	0,98	2	0,90	0,65
Разом					44.75

4.1.5 Накладні витрати

Накладні витрати проектних організацій включають три групи видатків: витрати на управління, загальногосподарські витрати, невиробничі витрати. Вони розраховуються за встановленими відсотками до витрат на оплату праці.

Середньостатистичний відсоток накладних витрат приймемо 150% від заробітної плати:

$$H_B = 1{,}5 \cdot B_{ОП}, \tag{4.6}$$

де H_B – накладні витрати.

Розрахунок накладних витрат для даного проекту:

$$H = \frac{2525.53 \cdot 150}{100} = 3788.29 \text{ грн.}$$

4.1.6 Обчислення інших витрати

Інші витрати є витратами, які не враховані в попередніх статтях. Вони становлять 10% від заробітної плати:

$$I = 2525.53 \cdot 0.1 = 252.55 \text{ грн.}$$

Витрати на розробку програмного забезпечення складають:

$$K = B_{ОП} + B_{ф} + B_{ПВ} + H + I, \tag{4.7}$$

$$K_1 = 2525.53 + 517.32 + 254.1 + 3788.29 + 525.55 = 7610.79 \text{ грн.}$$

Витрати на відлагодження та дослідну експлуатацію програмного продукту визначаємо за формулою:

$$K_2 = S_{м.г.} \cdot t_{від}, \quad (4.8)$$

де $S_{м.г.}$ – вартість однієї машино-години роботи ПК, грн./год;

$t_{від}$ – комп'ютерний час, витрачений на відлагодження і дослідну експлуатацію створеного програмного продукту, год.

Загальна кількість днів роботи на комп'ютері дорівнює 30 днів. Середній щоденний час роботи на комп'ютері – 2 години. Вартість години роботи комп'ютера дорівнює 5,2 грн. Розрахунок витрат на відлагодження та дослідну експлуатацію:

$$K_2 = 5.2 \cdot 60 = 312 \text{ грн.}$$

На основі отриманих даних складаємо кошторис витрат на розробку програмного забезпечення.

Загальні витрати ($B_{КС}$) розрахуємо за формулою:

$$B_{КС} = B_{ОП} + B_{ф} + B_{М} + B_{Е} + B_{АМ} + B_{Т} + H_{В}, \quad (4.9)$$

Таблиця 4.5 – Кошторис витрат на розробку програмного забезпечення

№ п/п	Найменування витрат	Сума витрат, *грн*.
1	Витрати на оплату праці	2525.53
2	Відрахування у спеціальні державні фонди	517.32
3	Витрати на куповані вироби	254.1
4	Накладні витрати	3788.29
5	Інші витрати	252.55
6	Витрати на відлагодження і дослідну експлуатацію програмного продукту	312,0
Разом		7649.79

4.2 Визначення експлуатаційних витрат

Для оцінки економічної ефективності розроблюваного програмного продукту слід порівняти його з аналогом, тобто існуючим програмним забезпеченням ідентичного функціонального призначення.

Експлуатаційні одноразові витрати по програмному забезпеченню і аналогу включають вартість підготовки даних і вартість роботи комп'ютера (за час дії програми):

$$E_{П} = E_{1П} + E_{2П}, \tag{4.10}$$

де $E_{П}$ – одноразові експлуатаційні витрати на ПЗ (аналог), грн.

$E_{1П}$ – вартість підготовки даних для експлуатації ПЗ (аналогу), грн.

$E_{2П}$ – вартість роботи комп'ютера для розробки програмного забезпечення (аналогу), грн.

Річні експлуатаційні витрати $B_{ЕП}$ визначаються за формулою:

$$B_{ЕП} = E_{П} \cdot N_{П}, \tag{4.11}$$

де $N_{П}$ – періодичність експлуатації ПЗ (аналогу), раз/рік.

Вартість підготовки даних для роботи на комп'ютері визначається за формулою:

$$E_{1П} = \sum_{i=1}^{n} n_i t_i c_i, \tag{4.12}$$

де i – категорії працівників, які приймають участь у підготовці даних (i=1,2,...n);

n_i – кількість працівників i-ої категорії, осіб.;

t_i – трудомісткість роботи співробітників *i*-ої категорії по підготовці даних, год.;

c_i – середньо годинна ставка працівника *i*-ої категорії з врахуванням додаткової заробітної плати, що знаходиться із співвідношення:

$$c_i = \frac{c_i^0 (1+b)}{m}, \qquad (4.13)$$

де c_i^0 – основна місячна заробітна плата працівника і-ої категорії, грн.

b – коефіцієнт, який враховує додаткову заробітну плату (приймемо 0,57);

m – кількість робочих годин у місяці, год.

Для роботи з даними як для поточного програмного забезпечення так і аналогу потрібен один працівник, основна місячна заробітна плата якого складає: с° = 1400 грн. Тоді:

$$c_1 = \frac{1400 \cdot (1+0)}{168} = 8.3 \text{ грн/год.}$$

Трудомісткість підготовки даних для програмного забезпечення складає 1 год., для аналога 1,5 год.

Таблиця 4.6 – Розрахунок витрат на підготовку даних та реалізацію програмного забезпечення на комп'ютері

№	Час роботи співробітників, *год.*	Середньогодинна заробітна плата, грн./год.	Витрати , грн.
	Проектне рішення		
1	1	8.3	8.3
	Аналог		
2	1,5	10,7	16,05

Витрати на експлуатацію комп'ютера визначається за формулою:

$$E_{2П} = t \cdot S_{МГ}, \quad (4.14)$$

де t – витрати машинного часу для реалізації програмного продукту (аналогу), год.;

$S_{МГ}$ – вартість однієї години роботи комп'ютера, грн./год.

$$E_{2П} = 1 \cdot 5.2 = 5.2 \text{ грн.}; \; E_{2а} = 1.5 \cdot 5.2 = 7.8 \text{ грн.};$$
$$E_{П} = 8.3 + 5.2 = 13.5 \text{ грн.}; \; E_{а} = 16.05 + 7.8 = 23.85;$$
$$B_{ЕП} = 13.5 \cdot 252 = 3402 \text{ грн.}; \; B_{еа} = 23.85 \cdot 252 = 6010.2 \text{ грн}$$

4.3 Розрахунок ціни споживання програмного продукту

Ціна споживання – це витрати на придбання і експлуатацію програмного продукту за весь строк його служби:

$$Ц_{С(П)} = Ц_{П} + B_{(Е)NPV}, \quad (4.15)$$

де $Ц_{П}$ – ціна придбання програмного продукту, грн.:

$$Ц_{П} = К(1 + \frac{П_{Р}}{100}) + К_{0} + К_{к} \quad (4.16)$$

де $К$ – кошторисна вартість;

$П_{Р}$ – рентабельність;

K_0 – витрати на прив'язку та освоєння програного забезпечення на конкретному об'єкті, грн.;

$K_к$ – витрати на доукомплектування технічних засобів на об'єкті, грн.

$$Ц_{П} = 10000 \cdot (1 + 0.3) = 13000 \text{ грн.}$$

$$Ц_{ПА} = 9143 \cdot (1 + 0.3) = 11885.9 \text{ грн.}$$

Вартість витрат на експлуатацію програмного забезпечення (за весь час його експлуатації), грн.:

$$B_{епрv} = \sum_{t=0}^{T} \frac{B_{ЕП}}{(1+R)^t}, \qquad (4.17)$$

де $B_{ЕП}$ – річні експлуатаційні витрати, грн.;

T – термін служби програмного забезпечення, років;

R – річна ставка проценту банку.

$$B_{впру} = \sum_{t=1}^{5} \frac{4006.8}{1+0.5} = 13356 \text{ грн.}$$

$$B_{впру} = \sum_{t=1}^{5} \frac{6010.2}{1+0.5} = 20034 \text{ грн.}$$

Тоді ціна споживання програмного забезпечення дорівнюватиме:

$$Ц_{СП} = 13000 + 13356 = 26356 \text{ грн.}$$

Аналогічно визначається ціна споживання для аналогу:

$$Ц_{са} = 11885.9 + 20034 = 25234.27 \text{ грн.}$$

4.4 Визначення показників економічної ефективності

Економічний ефект в сфері розробки програмного продукту:

$$E_{ПР} = Ц_{П} - Ц_{А} \quad (4.18)$$

$$E_{ПР} = 13000 - 11885.9 = 1114.1 \text{ грн.}$$

Річний економічний ефект в сфері експлуатації:

$$E_{КС} = B_{ЕА} - B_{ЕП} \quad (4.19)$$

$$E_{КС} = 6010.2 - 4006.8 = 2003.4 \text{ грн.}$$

Додатковий економічний ефект у сфері експлуатації:

$$\Delta E_{екс} = \sum_{t=1}^{T} E_{екс}(1+R)^{T-t} \quad (4.20)$$

$$\Delta B_{ВКС} = \sum_{t=1}^{5} 2003.4 \cdot (1+0.5)^{5-t} = 26419.83 \text{ грн.}$$

Сумарний ефект складає:

$$E = E_{ПР} + \Delta E_{ВКС} = 1114.1 + 26419.83 = 27533.93 \text{ грн.}$$

Таблиця 4.7 – Показники економічної ефективності програмного забезпечення

№	Найменування	Одиниці вимірювання	Значення показників	
			Базовий варіант	Новий варіант
1	Капітальні вкладення	грн.	-	7026.12
2	Ціна придбання	грн.	11885.9	13000
3	Річні експлуатаційні витрати	грн.	20034	13356
4	Ціна споживання	грн.	25234.27	26356
5	Економічний ефект в сфері проектування	грн.	-	1114.1
6	Економічний ефект в сфері експлуатації	грн.	-	2003.4
7	Додатковий ефект в сфері експлуатації	грн.	-	26419.83
8	Сумарний ефект	грн.	27533.93	

В даному розділі проведено розрахунок витрат на розробку програмного забезпечення. Здійснено порівняння з існуючим аналогом. На основі даних про зміст основних функцій, які повинен реалізувати програмний продукт, були визначені найбільш перспективні варіанти реалізації продукту. Згідно проведеного економічного обґрунтування зазначене програмного забезпечення є конкурентноздатним. Крім того, отримано економічний ефект у розмірі 1114.1грн. і тому розробка та впровадження цього програмного забезпечення є економічно доцільними.

ВИСНОВКИ

Отже, в результаті розробки бакалаврської роботи можна зробити наступні висновки:

1. На основі аналітичного підходу проведено порівняльний аналіз програмних засобів розробки модуля обміну повідомленнями та аналіз існуючих телемедичних систем, що дозволило виділити переваги та недоліки, та врахувати їх під час розробки власної системи.

2. Розроблено алгоритм обміну повідомленнями в режимі реального часу з використанням технології WPF, SignalR та бази даних PosgreSql, що дозволило виділити окремі функціональні модулі при проектуванні програмного модулю.

3. Розроблено структуру таблиць реляційної бази даних для зберігання інформації про користувачів системи та проведено її нормалізацію, що дозволило реалізувати сховище даних для розробленого програмного модулю.

4. Представлено структуру клієнтської та серверної частини системи «Hiams». Здійснено розробку unit-тестів для тестування критичних ділянок коду в автоматичному режимі. Здійснено порівняльний аналіз розробленої системи із існуючими аналогами, що дозволило виділити їх переваги та недоліки.

Основною перевагою розроблено у даній бакалаврській роботі модулю є наявність функціоналу для обміну повідомленнями і даними у телемедичній системі, зокрема для роботи з системою «Hiams».

Розроблений програмний модуль має практичне значення, що підтверджено довідкою про впровадження (додаток Б).

Результати дипломного проектування доповідалися на інтернет-конференції «Науково-практична конференція інтелектуальні ком’ютерні системи та мережі» (додаток В).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Електронна енциклопедія Вікіпедія: Веб-сайт URL: http://uk.wikipedia.org/wiki/Веб-сайт (дата звернення: 15.11.18).

2. Кузнецов М. В. ASP.NET. Практика создания Web-сайтов. СПб.: БХВ-Петербург, 2009 — 1264 с.

3. Хокінс С. Администрирование веб-сервера Apache и руководство по электронной коммерции. М.: Вильямс, 2001 — 336 с.

4. Six of the Best Open Source Data Mining Tools: веб-сайт. URL: https://thenewstack.io/six-of-the-best-open-source-data-mining-tools/ (дата звернення: 16.01.2019).

5. Єрмолаєв В. А. Програмне забезпечення ЕОМ. Структури даних та алгоритми. Запоріжжя: ЗНУ,2005 - 110с.

6. Хокінс С. Администрирование веб-сервера Apache и руководство по электронной коммерции. М.: Вильямс, 2001 — 336 с.

7. Entity framework: 18 особенностей движка URL: http://cccp-blog.com/laravel/laravel-frejmvork (дата звернення: 10.12.18)

8. Газизова Э.Р. Аутентификация. Теория и практика обеспечения безопасного доступа к информационным ресурсам. Учебное пособие для вузов. СПб.:БХВ-Петербург, 2009 – 559 с.

9. Мао В. Современная криптография: теория и практика. М.:Вильямс, 2001 – 240с.

10. Електронна енциклопедія Вікіпедія: Логін URL: http://ru.wikipedia.org/wiki/Логін (дата звернення: 05.01.19).

11. Аткинсон Л. Mysql. Библиотека профессионала. М Энергоатомиздат, 2002 – 496 с.

12. Дейт К. Руководство по реляционной СУБД DB2 М.: Финансы и статистика, 1988 – 320 с.

13. Мейер М. Теория реляционных баз данных М.: Мир, 1987 – 608 с.

14. Семантична модель: База даних URL: http://citforum.ru/database/advanced_intro/27.shtml (дата звернення: 15.11.18).

15. Chiron Health: Веб-сайт URL: https://chironhealth.com/pricing/ (дата звернення: 10.01.19).

16. Риккарди Грег. Системы баз данных. Теория и практика использования в Internet М.:Вильямс, 2001 – 240с.

17. Дейт К. Дж. Введение в системы баз данных СПб: Изд-во "Пітер", 2005 – 1315с.

18. Роберт І.В. Сучасні інформаційні технології в освіті: дидактичні проблеми, перспективи використання М.: Школа-Пресс, 1994 — 205с.

19. Мюллер Р.Дж. Базы данных и UML. Проектирование М.: Издательство “Лори”,2002 – 432с.: ил.

20. Острей О.Р. Діаграми класів UML як засіб моделювання інформаційної системи моніторингу освіти М.: - 2008. № 2. – С.85-89.

21. JSON и XML. Что лучше? : веб-сайт. URL: https://habr.com/post/31225/ (дата звернення: 12.02.19).

22. Формат JSON, метод toJSON? : веб-сайт. URL: https://learn.javascript.ru/json (дата звернення: 12.02.19).

23. Как легко понять логистическую регрессию: веб-сайт. URL: https://habr.com/ru/company/io/blog/265007/ (дата звернення: 12.02.2019).

24. Електронна енциклопедія Вікіпедія: JSON URL: https://uk.wikipedia.org/wiki/JSON (дата звернення: 13.02.19).

25. Компонентне або модульне тестування: веб-сайт. URL: http://www.protesting.ru/testing/levels/component.html (дата звернення: 14.02.19).

26. Електронна енциклопедія Вікіпедія: Модульне тестування URL: https://uk.wikipedia.org/wiki/Модульне тестування (дата звернення: 15.02.19).

27. Гаевский А. Ю. Самоучитель по созданию Web-страниц: HTML, JavaScript, Dynamic HTML К.: А.С.К., 2002 — 472с.

28. Методичні вказівки до написання техніко – економічного розділу для дипломних проектів на здобуття освітньо – кваліфікаційного рівня «Бакалавр»

напряму підготовки 6.050102 «Комп’ютерна інженерія» / І.Р. Паздрій. – Тернопіль: ТНЕУ, 2015. – 36с.

29. Методичні рекомендації до виконання дипломного проекту освітньо – кваліфікаційного рівня «Бакалавр» напряму підготовки 6.050102 «Комп’ютерна інженерія» фахового спрямування «Комп’ютерні системи та мережі» / Л.О. Дубчак О.М. Березький, Р.Б Трембач, Г.М. Мельник, Ю.М. Батько, С.В. Івасьєв / Під ред. О.М. Березького. – Тернопіль: ТНЕУ, 2016.- 60с.

30. Методичні вказівки до оформлення курсових проектів, звітів про проходження практики, випускних кваліфікаційних робіт для студентів спеціальності «Комп’ютерна інженерія» / І.В. Гураль, Л.О. Дубчак / Під ред. О.М. Березького. - Тернопіль: ТНЕУ, 2019. – 33 с.