МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра інтелектуальних технологій

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

НА ТЕМУ

Програмна система надання рекомендацій щодо плану дієт та фізичних вправ для хворих на діабет 2-го типу

Галузь знань **12 «Інформаційні технології»**

Спеціальність **122 «Комп'ютерні науки»**

Освітня програма **«Комп'ютерні науки»**

Освітній рівень: бакалавр

Виконав: студент 4 курсу, групи КН- 42

Чорненький Д.О.
(прізвище та ініціали)

Керівник Циганок В.В.
(прізвище та ініціали)
проф., д.т.н, с.н.с
(науковий ступінь, звання)

Випускна кваліфікаційна робота бакалавра допущена до захисту
рішенням кафедри *інтелектуальних технологій*
Протокол № 13 від 05.06.2023 р.
зав. кафедри_____доц. Іларіонов О.Є.

Київ – 2023

Анотація

Чорненький Денис Олегович виконав випускну кваліфікаційну роботу на тему «Програмна система надання рекомендацій щодо плану дієт та фізичних вправ для хворих на діабет 2-го типу» за спеціальністю 122 – «Комп'ютерні науки».

У випускній кваліфікаційній роботі проведено аналіз поточних проблем, з якими стикаються хворі на діабет, вже створених додатків та вебсайтів для надання медичної допомоги, залучено методи інтелектуальних технологій для точного та швидкого реагування на відхилені від норми ситуації. Окрім цього, розроблено інформаційне та програмне забезпечення, що виконує процедури оцінки поточних даних пацієнтів за життєво важливими критеріями в умовах прогнозування та рекомендує на основі отриманих даних найкращий спосіб полегшення протікання визначеної хвороби.

Ключові слова: інтелектуальні технології, програмне забезпечення, діабет 2-го типу, життєво важливі критерії, оцінка, рекомендації.

Summary

The degree project: «A software system for providing recommendations for a diet and exercise plan for type 2 diabetes patients» has completed by **Denys Chornenkyi** specialty 122 – «Computer Sciences».

In this graduation thesis, an analysis of the current problems faced by patients with diabetes, already created applications and websites for the provision of medical care was carried out, methods of intelligent technologies were involved for accurate and quick response to abnormal situations. Information and software was developed, which performs procedures for evaluating current patient data according to vital criteria in forecasting conditions and recommends the most relevant solution based on the received data.

Keywords: intelligent technologies, software, type 2 diabetes, vital criteria, assessment, recommendations.

ЗМІСТ

ВСТУП

Інформаційні технології використовуються в усіх сферах життя, від звичайних автоматичних дверей магазину, до визначення температури через смартфон. Цінність інтелектуальних технологій зростає з кожним днем і охоплює всі сфери життя.

Використання інтелектуальних технологій у медицині, звичайно, не новинка, можна навіть сказати, що це досить протестоване середовище, оскільки існує чимало додатків та систем для невідкладної допомоги. Але, все ще є і будуть ті сфери в медицині, де “розумні” програмні продукти можуть змінити ситуацію та значно покращити її. Однією із таких сфер є допомога хворим на цукровий діабет за допомогою застосування програмної системи надання рекомендацій стосовно плану дієт та фізичних вправ для хворих на діабет.

Цукровий діабет - небезпечне хронічне захворювання, яке може погіршити "якість" життя. У більшості випадків хвороба корінним чином змінює життя пацієнта. Хворим на діабет “доводиться жити” з цією хворобою. Якщо говорити про цукровий діабет 2 типу, то він може бути більш небезпечним, оскільки, стан пацієнта, зазвичай, погіршується повільно, але неухильно. Запобіжні заходи щодо дієти, застереження щодо руху тіла, уникнення травм, обов’язкові фізичні вправи, дотримання розкладу приймання ліків і дієти – ось деякі основні проблеми для хворих на цукровий діабет, а також часте відвідування лікаря та виділення часу на інші медичні заходи, наприклад, медичні аналізи. Це більш проблематично для літніх пацієнтів [1].

З іншого боку, лікування цукрового діабету також є важким, високовартісним і комплексним завданням для медичного персоналу. Існує ряд важливих даних, які потрібно знати про пацієнта та хворобу, періодично записувати, наприклад, план дієти, план фізичних вправ, час частих прийомів, щоденне вимірювання рівня цукру в крові пацієнта тощо. Усі ці записи допомагають лікарям прийняти оптимальне рішення щодо пацієнта, щоб зробити його життя кращим. Пацієнту практично неможливо щодня відвідувати

медпункт і повідомляти лікаря про рівень цукру в крові та інші необхідні речі. Інтелектуальна система підтримки прийняття рішень в медицині при цукровому діабеті 2 типу є хорошим розв'язання проблеми, яке може допомогти пацієнтам і медичному персоналу також. Медичні працівники високо цінують таку систему, яка надає відповідні та різноманітні поради щодо здоров’я пацієнтам хворим на діабет, особливо старшому поколінню. Така система може допомогти зменшити їхні непотрібні візити до лікарні. Система повинна надавати пацієнту достатньо корисної інформації, а також допомагати швидко прийняти оптимальне рішення щодо захворювання пацієнта. Повинен бути простий спосіб збереження записів і відображення таких результатів, які легко зрозуміти та допомагають прийняти рішення. Між медичним працівником і пацієнтами не повинно бути безперервного каналу зв’язку. Навпаки, пацієнти хочуть постійного каналу зв'язку з лікарями. Їм необхідна така система, яка може допомогти їм віддалено оновлювати свої записи, щоб можна було зменшити витрати на відвідування медичного центру [2].

Окрім того, пацієнтам потрібна порада щодо дієти та фізичних вправ, тому дана дипломна робота спрямована на розробку інтелектуальної системи підтримки прийняття рішень для допомоги пацієнтам.

Об’єктом дослідження даної роботи є хворі на цукровий діабет 2-го типу та групи пацієнтів, які користуються інтелектуальною системою підтримки прийняття рішень.

Предметом дослідження роботи є розробка та застосування програмної системи надання рекомендацій щодо плану дієт та фізичних вправ для хворих на діабет 2-го типу, ефективність та користь системи для хворих.

Метою дослідження є інформаційна підтримка хворих на цукровий діабет та медичних працівників за допомогою надійної та інтелектуальної програми для отримання медичних послуг дистанційно, а саме:

- поліпшення якості життя хворих на діабет; (вони можуть отримати основні інструкції та базові методи лікування щодо свого здоров’я, залишаючись вдома)

- зменшення кількості візитів пацієнтів до лікарень або центрів надання медичної допомоги завдяки ефективному обміну інформацією між постачальниками медичних послуг та пацієнтами;
- розширення співпраці між медичним персоналом і пацієнтам. (Останні зможуть отримати доступ до інформації та отримувати рекомендації в будь-який час, використовуючи власний ПК)

ПЕРШИЙ РОЗДІЛ. АНАЛІЗ ТА ПОСТАНОВКА ЗАДАЧІ. НАДАННЯ РЕКОМЕНДАЦІЙ ЩОДО РЕАБІЛІТАЦІЇ ХВОРИХ

1.1 Аналіз хвороби цукрового діабету 2-го типу та її лікування

1.1.1 Поняття про цукровий діабет

Цукровий діабет є одним з основних хронічних захворювань, пов’язаних з аномально високим рівнем цукру (глюкози) у крові, і це стає проблемою охорони для будь-якої країни. Цукровий діабет займає 7-му позицію в топі серед причин смерті. Цією хворобою у світі страждає понад 180 мільйонів людей, і очікується, що до 2030 року ця цифра подвоїться. Інсулін - це гормон, який виділяє підшлункова залоза для контролю рівня цукру в крові. Інсулін розщеплює глюкозу на частини, щоб вона могла розчинятися в м’язах, жирі та клітинах печінки людського тіла, де її можна використовувати як паливо. У разі діабету кров не може генерувати достатньо інсуліну для розчинення глюкози або не може використовувати власний інсулін (м’язи, жир і клітини печінки не реагують на інсулін нормально), що підвищує рівень цукру в крові. Це може спричинити серйозні ускладнення для здоров’я, такі як сліпота, хвороба серця, ниркова недостатність, втрата ваги та погане загоєння ран, особливо на ногах [6].

Розглянемо два основних типи діабету:

- Тип 1. Діабет 1-го типу, організм виробляє мало або зовсім не виробляє інсулін, зазвичай у 5-10% випадків діагностується у дітей та молодих людей. Багатьом пацієнтам, які страждають на діабет 1-го типу, діагностують у віці старше 20 років. При цьому типі діабету необхідне введення інсуліну. Його можна контролювати за допомогою дієти та фізичних вправ. Діабет 1 типу частіше призводить до ниркової недостатності. Ниркова недостатність розвивається у 20-40% людей, хворих на діабет 1 типу у віці 50 років.
- Тип 2. Це частіше, ніж діабет 1 типу, зазвичай у 90-95%, його діагностують у дорослих, але у молодих людей також діагностують це захворювання. При цьому типі підшлункова залоза не виробляє достатньо

інсуліну для підтримки нормального рівня цукру в крові. Це важкий тип діабету, тому що здебільшого люди не знають, що страждають від цього. Трьома основними причинами діабету 2-го типу є неправильне харчування протягом усього життя, малорухливий спосіб життя та надмірна вага. Діабет 2-го типу стає все більш поширеним через такі фактори ризику, як літній вік, ожиріння, відсутність фізичних вправ, сімейна історія діабету, серцеві захворювання. Деякі зміни в способі життя, особливо збільшення фізичної активності, наприклад фізичні вправи та помірне зниження ваги, можуть зменшити ймовірність захворювання на діабет 2-го типу на цілих 58%. Застосування метформіну (препарат, що використовується для лікування діабету 2-го типу) разом із веденням способу життя та здорового харчування знижує ризик розвитку діабету 2 типу на 31%, але метформін майже неефективний у пацієнтів старше 60 років. Для лікування людей похилого віку більш важливим є правильне харчування, фізичні вправи та приймання ліків. Що ми й спробуємо реалізувати в нашій роботі [1, 3].

Таблиця 1.1 Загальні характеристики двох типів діабету

Характеристики	Тип 1	Тип 2
Вік початку захворювання	До 30 років	Після 30 років (найчастіше)
Ожиріння	Ні	Досить поширено
Схильність до кетоацидозу	Так	Ні
Ступінь резистентності інсуліну в плазмі	Від дуже низького до неможливо визначити	Змінний. Може бути низьким – в нормі, підвищеним. Все залежить від ступеня резистентності до інсуліну.
Пов’язаний зі специфічними антигенами HLA-D	Так	Ні
Антитіла до острівцевого апарату підшлункової залози при перевірці	Так	Ні
Острівкові патології	Втрата більшості бета клітин	Вигляд острівці в межах норми. В такому випадку це поширене явище
Лежачий стиль життя з хворобою дає можливість її постійного розвитку	Так	Так

Як можна бачити, з 2-м типом цукрового діабету, з дотриманням конкретних правил та процедур, можна справлятися самостійно. Саме тому ця

дипломна робота пов’язана з пацієнтами саме з захворюванням діабетом 2-го типу, як покращити якість їхнього життя та як вони можуть отримати доступ до медичних послуг вдома. Пацієнти повинні отримувати перелік фізичних вправ та список дієти, мати можливість написати власному лікарю та цілодобово оновлювати свої життєво важливі дані. Це може повністю змінити їхній розпорядок дня. Дипломна робота зменшить непотрібні візити пацієнтів до лікарів для клінічних тестів, дієти та плану фізичних вправ [7].

1.1.2 Актуальність дослідження у сфері реабілітації хворих

Завдяки розробці програмної системи, медичне лікування захворювань, таких як діабет, у домашніх умовах, особливо для пацієнтів похилого віку, стане реальністю. У такій системі охорони здоров’я при цукровому діабеті існують дослідження, які пов’язані з наданням необхідної інформації онлайн та аналізу в першу чергу необхідних візитів до медичного персоналу, тому, завдяки даним дослідженням та на основі відповідної статистики і будуть будуватись функції програмної системи.

Самообслуговування пацієнта з особистою програмою допомоги.

Користувачі стаціонарного самообслуговування отримують підтримку інтелектуального програмної системи надання рекомендацій, що включає просту діагностику на основі нещодавно виміряних медичних даних. Самообслуговування пацієнта за допомогою програмної системи може зменшити кількість відвідувань лікарень. Додаток проекту має електронний порядок дня, якого потрібно дотримуватись. Електронний розпорядок дня надає необхідний перелік життєво важливих даних хворому на цукровий діабет, які йому необхідно постійно оновлювати, а також, наприклад, план вправ, які обов’язково потрібно робити хворому задля зменшення ризиків прогресування хвороби. Також надається список з певних страв, які пройшли рекомендації від лікарів і є затвердженні, консультацію лікаря-спеціаліста та регулярні огляди. Інформація, що зберігається в розпорядку дня (спостереженнях), доступна лікарю віддалено. Програма надає пацієнтам зручний інтерфейс, щоб допомогти

їм з дієтою, фізичними вправами, ліками, плануванням часу та медичним оглядом. Є обмеження проекту - це велике електронне табло, яке показує все на екрані. Він не портативний. Пацієнт повинен підійти до певного місця (будинок чи офіс), щоб отримати допомогу.

Підтримка рішень для учасників через моніторинг протікання хвороби та сигнали про тривогу.

Дана система підтримки прийняття рішень надає рекомендації медичному персоналу щодо діагнозу. Додаток надає послуги хворим на цукровий діабет для покращення якості їхнього життя шляхом забезпечення кращого зв'язку між пацієнтами з цукровим діабетом і лікарем. Дана програмна система може контролювати та отримувати дані про рівень глюкози в крові та передавати їх інтелектуальному агенту. (інтелектуальній системі прийняття рішень) Якщо є будь-який тривожний стан, він запускає тривогу, відображаючи відповідне сповіщення в інтерфейсі та безпосередньо надсилаючи тривожне повідомлення родичу цього пацієнта . Основною функцією додатка є обробка даних пацієнтів і автоматичне генерування сигналів тривоги за допомогою деяких методів, визначених алгоритмом надання рекомендацій, поєднання методів, заснованих на статистиці та моделях, передових методів для аналізу даних сукупних вимірювань рівня глюкози в крові. Програмна система надає 24 години на добу прості послуги пацієнтам і лікарям, що також є метою цього дослідження.

1.1.3 Визначення проблеми

Проблема спілкування та роботи з ризиком і невизначеністю є серйозною для безліч сфер, в тому числі й для медицини. Пацієнти з діабетом, особливо старше покоління, потребують більшого догляду, ніж інші, тому їм потрібен повноцінний відпочинок, догляд за всім тілом, наприклад, шкірою, суглобами, а також регулярність у лікуванні. Вони також потребують обережності у своєму харчуванні, неправильні ліки та неправильна їжа можуть спричинити серйозні захворювання. Тільки сам пацієнт або лікар, який не проводить постійного особистого контролю, не може надати необхідну йому допомогу.

Щоб покращити якість життя пацієнтів, необхідна злагоджена команда людей, які працюють разом. Команда складається з самого пацієнта, лікаря, медсестер, аптекарів та родичів. Пацієнтові важко відвідувати лікаря щодня або записатися на лабораторні дослідження щодня, навіть жоден медичний працівник не може відвідувати пацієнта щодня або протягом короткого періоду часу, щоб бути в курсі здоров'я пацієнта.

Більшість хворих на цукровий діабет щодня звертаються до своїх медичних працівників. Якість пацієнта з діабетом можна покращити, забезпечивши надійний і легкий доступ між постачальниками послуг і пацієнтом. Керівництво з охорони здоров'я при цукровому діабеті описує, що для якісної медичної допомоги при діабеті необхідна командна робота, в якій основний акцент робиться на хворих на діабет. Пацієнти потребують спеціального навчання, підтримки та нагляду, щоб вони могли краще піклуватися про себе, контролювати власну ситуацію та займатися самолікуванням. Пацієнти повинні легко спілкуватися з медичними працівниками. Пацієнт не може щодня відвідувати лікаря або записатися на лабораторні дослідження щодня. Вони можуть бути занадто хворими, щоб відвідувати лікарню, пацієнти можуть мати раціональну поведінку, з якою вони не погоджуються відвідувати медиків, навіть жоден медик не може відвідувати пацієнта щодня або протягом коротких періодів часу, щоб тримати себе в курсі про стан здоров'я пацієнта. Іноді лікарні чи клініки занадто переповнені, і пацієнтам доводиться довго чекати, щоб відвідати лікаря та провести лабораторні дослідження, це також важко для пацієнтів [6].

1.1.4 Фокус проблеми та викликів

- Повсякденний розпорядок життя пацієнта повністю залежить від розкладу всіх постачальників послуг. Він/вона повинен чекати медичних оглядів або будь-яких інших медичних вправ. Занадто багато візитів до лікарень та призначення лабораторій для медичних тестів заважають повсякденному життю.

- Хворі на діабет страждають від багатьох видів ускладнень. Їм потрібен особливий вид догляду (дієта, фізичні вправи), пацієнт сам або лікар не може надати необхідну йому медичну допомогу.
- Медичні працівники не можуть відвідувати пацієнта щодня або протягом короткого періоду часу, щоб бути в курсі здоров'я пацієнта. Вони можуть не завжди добре знати про стан пацієнта, особливо якщо постачальник послуг повинен виконувати різні дії. У такій ситуації постачальникам послуг важко оновлювати інформацію про стан здоров'я пацієнтів.

Ці проблеми відіграють важливу роль у нашому дослідженні. Усунувши ці проблеми, ми можемо заощадити багато медичних ресурсів і багато дорогоцінного часу медперсоналу, родичів і пацієнтів. Цей час може бути використаний для інших конструктивних заходів як пацієнтами, так і постачальниками послуг. Пацієнт може витрачати більше часу на соціальну діяльність замість того, щоб проводити більшу частину часу в депресивному середовищі лікарні [8, 9].

1.1.5 Питання дослідження

1. Як підвищити якість життя хворих на цукровий діабет, забезпечивши інтелектуальну програму для самообслуговування використовуючи комп'ютер в будь-який час?
2. Як можна оновлювати стан здоров'я пацієнтів і сповіщати родичів, якщо є будь-який тривожний стан?
3. Як технологія може підтримати медичний персонал і допомогти лікарям у прийнятті рішень щодо захворювань пацієнтів?

Оскільки діабет є хронічним захворюванням, хворі на цукровий діабет потребують більше догляду, ніж інші, їм потрібна особлива увага щодо дієти та фізичних вправ, щоб підтримувати себе. Хворий на діабет потребує регулярних оглядів у медичних працівників. Ці регулярні огляди, медичні аналізи, терапія, інструкції з дієти, фізичні вправи та вправи заважають пацієнтам жити. Згідно з нашим першим досліджуваним питання, ми спробуємо забезпечити вирішення

деяких із цих проблем за допомогою інтелектуальної програми, за допомогою яких якість життя пацієнтів підвищиться і вони зможуть отримати до неї доступ в будь-який момент часу. Пацієнти можуть приймати інструкції щодо дієти та фізичних вправ, залишаючись вдома, і вимірюючи рівень цукру в крові та оновлюючи медичну карту, не відвідуючи лікарню. Згідно з нашим другим дослідницьким питанням, пацієнт оновить дані свого статусу в системі, щоб медичний персонал та родичі були спокійні за стабільно нормальний стан конкретного пацієнта, але якщо виникнуть тривожні умови, автоматичне повідомлення (електронна пошта) надійде родичу про стан хворого. У нашому третьому дослідницькому питанні, на основі даних, які пацієнти оновлюють щодня, і попередньої зареєстрованої історії захворювання або інших пацієнтів з діабетом, це допоможе постачальникам послуг (лікарям) у прийнятті рішень щодо пацієнтів [10].

1.2 Аналіз бізнес-процесів та бізнес архітектура

Процес - це послідовність пов'язаних операцій або завдань, що потрібні для досягнення результату. Під бізнес-процесом у широкому значенні розуміється структурована послідовність дій з виконання певного виду діяльності на всіх етапах життєвого циклу предмета діяльності - від створення концептуальної ідеї через проєктування до реалізації й результату (здача в експлуатацію об'єкта, постачання продукції, надання послуг, закінчення певної фази діяльності), тобто певний системно замкнений процес. Бізнес-процес являє собою сукупність бізнес-операцій, певну кількість внутрішніх видів діяльності, що починаються з одного або більше входів і закінчуються створенням продукції, необхідної клієнту (клієнт - не обов'язково зовнішній відносно підприємства споживач, це може бути підрозділ організації або конкретний працівник).

Поняття бізнес-процесу є багатозначним, і усі визначення об'єднує насамперед акцентування уваги на тому, що бізнес-процеси є безперервними, мають певні входи (постачання ресурсів, виникнення ідеї бізнесу, ідеї нового

продукту, послуги тощо) і виходи у вигляді продукту, що задовольняє потреби споживачів. Таким чином опишемо наші процеси, які охоплюють всю організацію програми, зверху до низу.

Бізнес-процеси нашої програми підтримки пацієнтів хворих на діабет умовно поділяють на 2 типи: front-end та back-end.

Front-end – це процеси, які видно пацієнтам, їх родичам, і які здатні впливати на їх вибір. Сюди включено залучення, обслуговування та утримання пацієнтів. Кожен із них поділяється на додаткові процеси. Обслуговування включає екстрені повідомлення, консультації, розробку плану дієти, підтвердження зустрічі з лікарем, фізичні вправи. Утримання клієнтів – це бізнес-процес нашої програми, що включає роботу з актуальними даними пацієнтів щодо їх стану, рекомендаціями, обробку дієт та фізичних вправ.

Back-end - це процеси, що впливають на роботу нашої програми для лікарів, але їх не видно таким користувачам як пацієнт, або його родич. Сюди відносяться:

- Формування рекомендацій. Сюди відносяться аналітика, прогнозування, спілкування з лікарями, розробка нових дієт та фізичних вправ.
- Обробка прийомів. Це перевірка наявності вільного часу певного лікаря, прийом вхідних дзвінків та повідомлень, підтвердження, дзвінок клієнтів.

Оптимізація бізнес-процесів програмного модуля рекомендацій (помічника) є необхідною та важливою для успішної роботи. Навіть якщо ці процеси не видно кінцевому споживачеві.

TOGAF використовує корпоративну архітектуру як спосіб покращити бізнес-можливості – ось чому перша фаза розробки архітектури пов'язана з бізнес-архітектурою. ADM (методологія розробки архітектури) починається з погляду бізнесу - сильні бізнес-вимоги визначаються у запиті на виконання архітектурних робіт на попередньому етапі і далі уточнюються у заяві про архітектурні роботи та концепцію архітектури на етапі чіткого архітектурне завдання, яке надаватиметься в ітераціях ADM.

Ключовим завданням фази бізнес-архітектури є розробка цільової бізнес-

архітектури, яка демонструє, як підприємство реалізує архітектурне бачення та відповідає на робочі запити щодо архітектури. Його друга мета полягає в тому, щоб спочатку визначити компоненти дорожньої карти архітектури-кандидата, щоб подолати розрив між базовою та цільовою бізнес-архітектурою. TOGAF розглядає знання бізнес-архітектури як необхідну умову для роботи з архітектурою в інших областях, таких як дані, програми та технології.

Побудуємо бізнес-архітектура (див. рис. 1) програмного модуля формування рекомендацій, що демонструє цінність для бізнесу та окупність інвестицій у архітектуру для ключових зацікавлених сторін бізнес-моделі, такі як моделі дій або процесів, моделі варіантів використання та класів або схеми з'єднань вузлів.

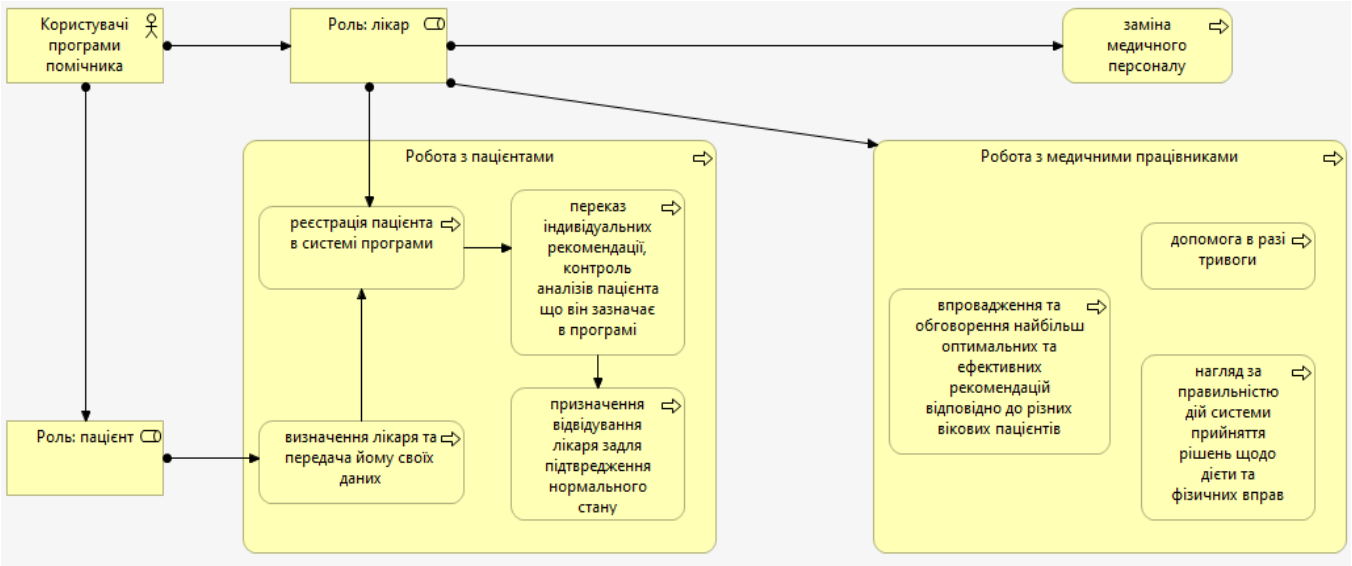


Рисунок 1.1 — Бізнес архітектура відділу прийому

На даному рисунку було зосереджено увагу на користувачів створюваної програми. Ми відокремили основні бізнес-процеси у роботі даної частини програми та тих хто їх виконує:

- Робота з пацієнтами:

Пацієнт – визначення лікаря та передача йому своїх даних.

Лікар:

1. Реєструє пацієнта в системі програми.

2. Переписує індивідуальні рекомендації та проводить контроль аналізів хворого, що останній зазначає в програмі декілька разів на день.

3. Призначає відвідування лікаря задля підтвердження нормального стану та дійсності аналізів в межах норми.

- Заміна медичного персоналу – Лікар
- Робота з медичними працівниками:

Лікар:

1. Впровадження та обговорення оптимальних та ефективних рекомендацій відповідно до різних вікових пацієнтів

2. Допомога в разі тривоги

3. Нагляд за правильністю дій системи прийняття рішень щодо дієти та фізичних вправ.

1.3 Порівняльний аналіз з вже існуючими програмами

При розробці програмної системи, спрямованої на допомогу людям, вкрай важливо звертати увагу на успішні аналоги, які вже набули популярності. Вивчаючи та аналізуючи вже існуючі успішні програми, ми можемо використати цінні інсайти та перейняти корисну інформацію для підвищення ефективності та результативності нових ініціатив. Такий підхід економить час, ресурси та мінімізує ризик провалу. До того ж використання перевірених аналогів дозволяє виявити найкращі практики, уникнути потенційних пасток і адаптувати успішні стратегії до конкретних умов. Це сприяє прийняттю рішень на основі фактичних даних і підвищує ймовірність розробки ефективних програм, які можуть позитивно трансформувати охорону здоров'я і поліпшити результати для хворих в усьому світі. Для створення якісної та надійної програмної системи, опишемо головні критерії популярних медичних програм та виконаємо порівняльний аналіз в таблиці 1.2.

Таблиця 1.2 Таблиця критеріїв порівняння створеної програми ж все існуючими

Критерії порівняння	Програми				
	Програмна система надання рекомендацій щодо дієти та фізичних вправ	First Aid	Full Term	Snug Safety	Pill Reminder and Med Tracker
Призначення	Для спрощення життя пацієнтів хворих на цукровий діабет другого типу та заощадження їх власного часу, а також часу їх лікарів.	Для допомоги впоратися з будь-якими випадками, від незначних аварій до надзвичайних ситуацій.	Якщо людині доведеться народжувати в несподіваному у місці.	Для людей похилого віку, хворих на епілепсію та людей із захворюваннями, які обмежують рухливість.	Для людей які вагаються чи правильно зрозуміли їх призначення. А також для тих в кого надважливий прийом ліків та його не можна пропускати.
Опис	Програмна система сама дає поради щодо дієти та фізичних вправ, не заважаючи медичним працівникам. Він сповіщатиме родичів через пошту, коли система аналізуватиме тривожні стани пацієнта.	Додаток надає відео та анімовані посібники на додаток до прямих ліній зв’язку зі службами екстреної допомоги.	Ця програма спростить принаймні один аспект процесу. Інтерфейс користувача було розроблено таким чином, щоб бути максимально зрозумілим з урахуванням зручності читання.	Забезпечуючи щоденну реєстрацію хворих. Snug Safety був розроблений спеціально для людей, які живуть самі.	Програма надає функцію, яка сповіщає опікунів і близьких, якщо хворий пропустив дозу.

Критерії порівняння	Програми				
	ПК	Ipad	Ipad	Ipad	Ipad
Зручність та зрозумілість	+	+-	+-	+-	+-
Користь	+	+	+	+	+
Швидкість	+	-	-	+	+
Точність в визначенні загрози	+	-	-	+	-
Наявність СППР для рекомендації плану дієти та фіз.вправ	+	-	-	-	-
Можливість системи надіслати звістку про надзвичайну	+	-	-	+	+
Можливість лікарю проаналізувати причину екстреної ситуації	+ (Завдяки аналізу всіх збережених даних пацієнтів)	-	-	-	-
Ціна	Безкоштовно	Присутні платні функції	Присутні платні функції	Присутні платні функції	Присутні платні функції

Згідно з таблицею 1.2 було проведено функціональний аналіз, сформовано структуру критеріїв оцінювання та порівняння, наведено порівняльну таблицю. Завдяки цьому, можна зробити висновок, що програмний модуль формування рекомендацій для діабетиків, розроблений в дипломній роботі, успішно пройшов всі критерії оцінювання та показала себе якнайкраще. Їх характерні простота, зрозумілість, швидкість, користь клієнту (адже програма раціонально розподіляє час хворого та в разі чого точно визначає рівень загрози, повідомляючи про це необхідних людей), конфіденційність, збереження надважливих даних пацієнтів, які вони оновлюють декілька разів на день задля аналізу і реконструкції плану дієт/фізичних вправ лікарями для майбутніх користувачів [11].

1.4 Постановка задачі на розробку програмної системи надання рекомендацій щодо плану дієт та фізичних вправ

1.4.1 Інтелектуальна система прийняття рішень

Інтелектуальна система підтримки прийняття рішень у сфері медицини при цукровому діабеті може забезпечити краще рішення за допомогою комп’ютерної програми допомоги. Програма допомоги надає інтерфейс та послуги користувачам (лікарям, пацієнтам, родичам) вдома. Система підтримки рішень в системі електронної охорони здоров’я при діабеті 2-го типу діятиме як експертна система. Експертна система - це комп'ютерна програма, яка допомагає розв'язувати проблеми або приймати рішення, зберігання яких актуальне. Інтелектуальна система підтримки прийняття рішень у програмі забезпечить цілодобову медичну допомогу пацієнтам і постачальникам послуг у будь-який час. Пацієнт зможе оновити стан свого здоров’я, план дієти, розпорядок фізичних вправ та коментарі від лікаря, залишаючись вдома. У разі надзвичайних ситуацій автоматичне повідомлення буде надіслано постачальникам послуг, наприклад, родичам пацієнтів.

1.4.2 Task flow програми

Лікар реєструє пацієнта за ПІБ, дата народження, адреса хворого, особистий номер, електронну адресу родича, електронну адресу пацієнта. Після процесу входу пацієнт може ввести рівень цукру в крові, тиск та отримати план дієти та фізичних занять.

Процес містить аналіз введених даних пацієнтів, історії оновлених даних, які посилатимуться щодня по багато разів. Система знайде комбінацію ліків дієти та фізичних вправ і підтвердить її. Якщо введені дані показують тривожний стан пацієнта, то система миттєво сповіщає про це родича.

Результати – інструкції щодо дієти та фізичних вправ. Система зберігає оновлення пацієнта в сховищі історії. Її може переглянути як пацієнт, так і лікар.

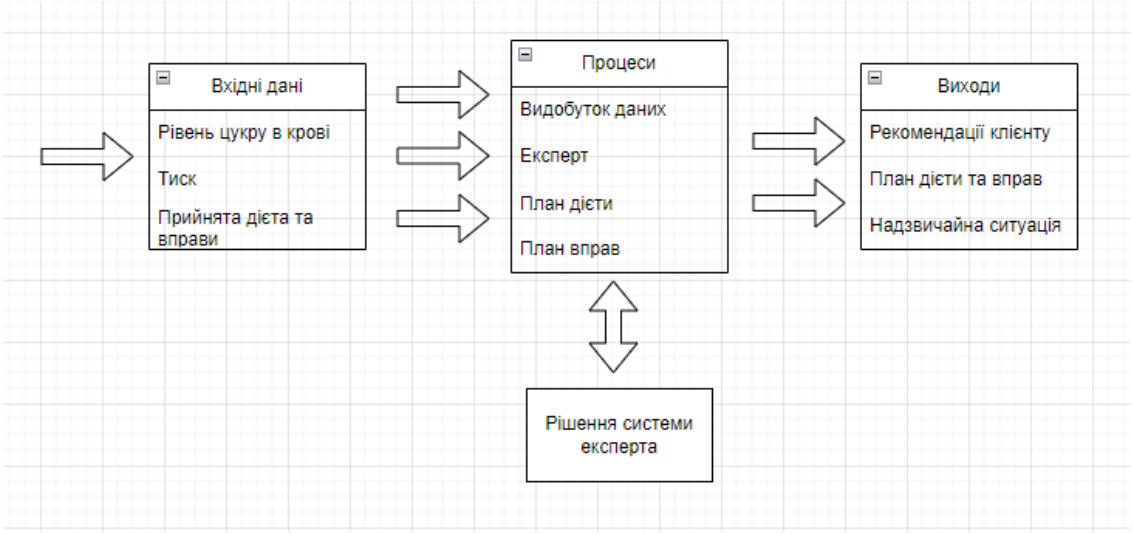


Рисунок 1.2 — Потік завдань Програми Помічника

1.4.3 Автоматичне повідомлення на пошту

Короткі повідомлення є хорошим засобом для віддаленого збору даних пацієнтів. Він успішно використовується в різних програмах для різних цілей, включаючи нагадування про зустрічі, нагадування про контрацепцію та навчання діабету. Використання такого нагадування в медицині має певні переваги, наприклад покращення зручності для пацієнтів, швидкі результати з меншою паперовою роботою та меншими витратами. Однією з ключових особливостей програми для діабетиків є автоматичне надсилання повідомлень в разі загрози. Коли пацієнт оновлює свої дані, щоб отримати вказівки щодо

здоров’я, система перевіряє стан пацієнтів. Якщо стан в межах не нормального, автоматичне повідомлення надійдуть до постачальників послуг (тобто родича), щоб повідомити про стан здоров’я пацієнта. Пацієнти лиш повинні під'єднатися до нашої програми через свій ПК та як і завжди оновити свої дані. Програмний модуль формування рекомендацій для діабетиків підключається через бази даних і має різні порогові значення, наприклад, рівень цукру, вона перевіряє введені дані пацієнтів, якщо введені дані дорівнюють або перевищують попередньо визначені тривожні значення, автоматичне повідомлення буде надіслано постачальникам послуг для інформування про стан здоров’я пацієнта. Це є унікальною функцією, яка виграє у всіх конкурентів. (детальніше в таблиці 1.2) Кожного разу рівень цукру в крові, який вводять пацієнти, порівнюватиметься з попередньо встановленими значеннями.

1.4.4 Основні вимоги до програмного забезпечення

Визначимо вимоги до програмного забезпечення, які описують внутрішню роботу системи, її поведінку: введення даних, маніпулювання даними, обробка даних та інші специфічні функції, які має виконувати система:

1. Можливість введення та видалення дієти.
2. Можливість додавання та видалення лікаря.
3. Можливість введення та видалення фізичних вправ.
4. Можливість переглянути, додати чи видалити спостереження пацієнта.
5. Можливість переглянути, додати чи видалити прийом до лікаря.
6. Можливість пацієнту відправити повідомлення лікарю на електронну адресу
7. Можливість додати чи видалити пацієнта, поступово ввівши його дані.

Визначимо вимоги до програмного забезпечення, які задають критерії для оцінки якості його роботи, або так звані вимоги до графічного інтерфейсу користувача:

1. Елементи інтерфейсу користувача мають супроводжуватися допоміжними заголовками, задля інтуїтивної зрозумілості їх призначення.

2. Інтерфейс користувача має бути зручним та простим, щоб кожен новий користувач відразу розумів свої можливості використовуючи відповідні елементи інтерфейсу.

3. Використання даного інтерфейсу має бути доступним в усіх версіях операційної системи Windows.

Визначимо спосіб структурування програмної або обчислювальної системи, абстракцію елементів системи на певній фазі її роботи:

1. Мова інтерфейсу – Українська мова.

2. Наявність документаційних коментарів (для класів – призначення класу; для методів – призначення методу, опис параметрів та значення).

3. Використання потрібних інтерфейсів.

1.5 Висновок до першого розділу

Своїм проектом, я хочу розробити програмну систему для допомоги пацієнтам хворим на цукровий діабет 2-го типу. Завдяки зрозумілому інтерфейсу та послугам користувачів, бувши вдома, інтелектуальна система підтримки прийняття рішень забезпечить цілодобову медичну допомогу пацієнтам і постачальникам послуг у будь-який час. Пацієнт зможе оновити стан свого здоров’я, отримати план дієти, розпорядок фізичних вправ та коментарі від лікаря, бувши не в лікарні. У разі надзвичайних ситуацій автоматичне повідомлення буде надіслано родичу пацієнта. Було проведено аналіз предметної області, визначено головні проблеми та питання, що турбують пацієнтів, а також створено таблицю порівнянь за основними критеріями в даній області (таблиця 1.2), створено бізнес-архітектуру (рисунок 1.1), а також описано основні вимоги до програмного забезпечення.

ДРУГИЙ РОЗДІЛ. РОЗРОБКА АРХІТЕКТУРИ ТА ОПИС ФУНКЦІЙ ПРОГРАМНОЇ СИСТЕМИ НАДАННЯ РЕКОМЕНДАЦІЙ ЩОДО РЕАБІЛІТАЦІЇ ХВОРИХ

2.1 Функціональні можливості програми

Застосунок, створений для спрощення життя хворих на діабет 2-го типу, містить кілька класів, кожен з яких має певні функціональні можливості. Основною метою програми є управління записами пацієнтів, включаючи особисту інформацію, історію хвороби та рекомендації щодо дієти. Розберемо деякі з цих класів:

1. Першим класом є клас Program, який є точкою входу в програму. Він містить метод Main, який відповідає за відображення головного меню та обробку даних, введених користувачем. Користувач може вибирати між кількома опціями, такими як додавання нового запису своїх показників, на основі яких згенеруються рекомендації щодо дієти та фізичних вправ, а також можливість написати листа своєму особистому лікарю.

2. Клас Pacient представляє один запис про пацієнта. Він містить такі властивості, як ім'я, дата народження, номер телефону, адреса та історія хвороби. Клас також має методи для додавання та оновлення історії хвороби та генерування рекомендацій щодо дієти на основі стану здоров'я пацієнта.

3. Клас Randomizer відповідає за генерацію випадкових даних, таких як імена, номери телефонів, адреси та дати. Він використовує клас Random для генерації випадкових чисел і вибирає значення з попередньо визначених масивів для побудови випадкових даних. Даний клас застосовується у випадку тестування нових функцій для адміністратора. Це значно економить час та одночасно не застосовує інформацію реальних клієнтів застосунку.

Використовуючи дерево функцій (рисунок 2.1) опишемо повноцінний вигляд застосунку. Почнемо з вершини дерева функцій, що являє собою головний модуль - функції програмної системи, і продовжимо адміністративною та

клієнтською частиною [4].

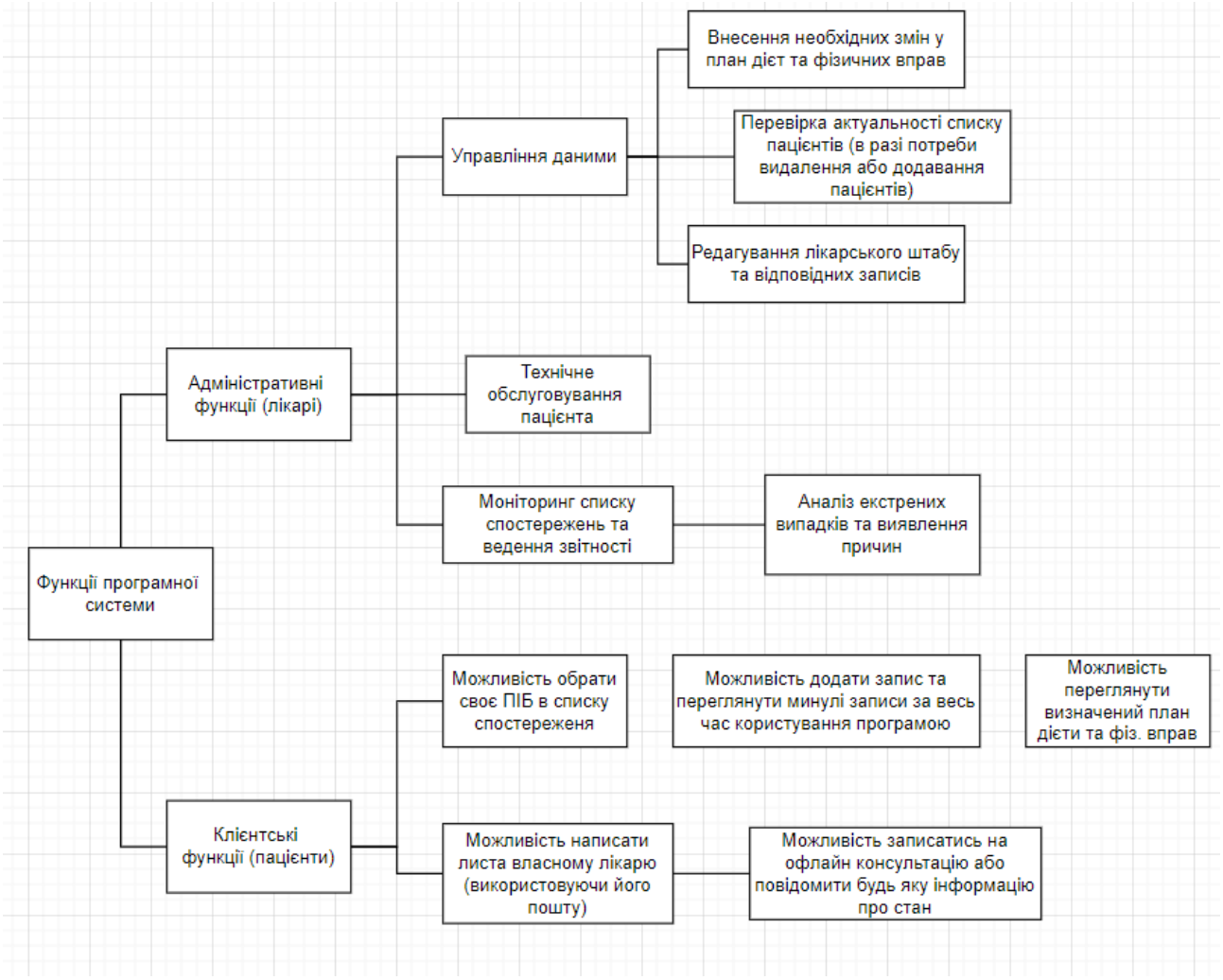


Рисунок 2.1 – Дерево функцій програмної системи рекомендацій щодо плану дієт та фізичних вправ

2.2 Методи інтелектуального аналізу програми

Інтелектуальні методи аналізу, що використовуються в програмі, - це системи, з використанням певного алгоритму, які є різновидом штучного інтелекту, що використовує твердження "якщо - то" для прийняття рішень. Відповідний алгоритм надання рекомендацій ґрунтується на медичних знаннях і розроблений таким чином, щоб відображати найкращі практики лікування різних медичних станів. Завдяки використанню програмної системи, що використовує під час своєї роботи алгоритм надання рекомендацій, програма здатна надавати персоналізовані плани дієти, пристосовані до унікальних потреб

і особливостей кожного пацієнта.

Для визначення дієти для конкретних пацієнтів програма використовує систему, з використанням алгоритму рекомендацій. Клас Patient має метод getDietRecommendations, який приймає на вхід історію хвороби пацієнта і повертає список рекомендованих продуктів. Метод використовує алгоритм для визначення рекомендованих продуктів на основі стану здоров'я пацієнта. Наприклад, якщо у пацієнта високий рівень холестерину, метод рекомендує продукти з низьким вмістом насичених жирів і холестерину. Додаток взаємодіє з базою даних для зберігання та пошуку записів про пацієнтів. База даних реалізована з використанням механізму баз даних SQLite, який є легкою та швидкою системою реляційних баз даних. Додаток використовує пакет System.Data.SQLite для підключення до бази даних і виконання SQL-запитів. Коли програма запускається, вона перевіряє, чи існує файл бази даних, і створює його, якщо його немає. Потім програма завантажує всі записи про пацієнтів з бази даних в пам'ять і зберігає їх у вигляді списку. Коли користувач додає або оновлює запис про пацієнта, програма записує зміни до бази даних, щоб зберегти їх.

Для повного розуміння роботи інтелектуальної системи прийняття рішень, наведемо покрокового алгоритм призначення програмою дієти пацієнту з діабетом 2 типу:

1. Спочатку з бази даних витягується історія хвороби пацієнта, включаючи його вік, вагу, зріст і рівень глюкози в крові. Ця інформація використовується для визначення тяжкості діабету.

2. На основі цієї інформації програма визначає, які з набору попередньо визначених дієтичних планів буде використовувати алгоритм.

3. Наприклад, якщо пацієнт перебуває на інсулінотерапії і має високий рівень глюкози в крові, алгоритм програми обмежує споживання вуглеводів, щоб контролювати рівень цукру в крові.

4. З іншого боку, якщо пацієнт приймає пероральний препарат, який стимулює вироблення інсуліну, алгоритм збалансовує споживання вуглеводів і

білків, щоб підтримувати рівень цукру в крові.

5. Програма також враховує дієтичні вподобання та культурні особливості пацієнта, пропонуючи продукти, які слід їсти або яких слід уникати.

6. Як результат, програма генерує персоналізований план дієти для пацієнта на основі результатів роботи алгоритму рекомендацій і відображає його в інтерфейсі користувача.

2.3 Архітектура інформаційно-програмної системи

Програмний застосунок має багаторівневу архітектуру, яка дозволяє розділити завдання і сприяє модульності та масштабованості. Його архітектура складається з:

1. Рівня представлення, який відповідає за представлення інформації користувачеві та отримання вхідних даних від користувача. У цьому додатку рівень представлення складається з інтерфейсу Windows Forms, який дозволяє користувачеві взаємодіяти з додатком. Інтерфейс користувача включає декілька форм, які дозволяють користувачеві взаємодіяти з системою, включаючи головне вікно, форму пацієнта та форму дієти. Головне вікно надає доступ до форми пацієнта, яка дозволяє користувачеві переглядати та вносити нову інформацію про себе. Форма взаємодії з лікарем, дозволяє користувачеві обмінюватись інформацією про будь-які аспекти життя з власним лікарем за допомогою електронної пошти [12].

2. Рівня бізнес-логіки, що містить бізнес-правила та логіку програмної системи. Даний рівень обробляє запити від рівня представлення і взаємодіє з рівень доступу до даних, що відповідає за взаємодію з базою даних, отримання та оновлення даних.. У програмі рівень бізнес-логіки включає головні класи функціональних можливостей, які містять логіку призначення дієт і управління даними про пацієнтів. Рівень бізнес-логіки також включає кілька інтелектуальних методів аналізу, таких як системи, для визначення відповідного плану дієти для пацієнта. Система використовує алгоритм для визначення плану

дієти для пацієнта на основі його віку, ваги та рівня активності. Рівень доступу до даних, що відповідає за взаємодію з базою даних, отримання та оновлення даних. В програмній системі рівень доступу до даних складається з класу DBHelper, який обробляє з'єднання з базою даних і виконує команди SQL. Процес призначення дієти передбачає збір інформації про пацієнта, такої як вага, зріст, вік і стан здоров'я (див. таблицю 2.1), а потім аналіз цих даних для визначення відповідної дієти. Застосунок використовує систему та її алгоритм рекомендацій, щоб проаналізувати дані пацієнта і призначити дієту відповідно до його потреб. Відповідні значення алгоритму базуються на стандартних медичних рекомендаціях для різних захворювань і враховують такі фактори, як вік, стать і рівень активності пацієнта.

Таблиця 2.1. Відношення зареєстрованих пацієнтів

Id	ПІБ	Дата народження	Адреса пацієнта	Телефон	Пошта пацієнта	Пошта родича
2	Шевченко Олександр Іванович	1988.07.30	вулиця Скіфська, буд. 190, кв. 74	769-70-06	test.pacient.email@gmail.com	denys.ch.uk13@gmail.com
3	Руденко Матвій Олександрович	1973.12.04	вулиця Миколи Костомарова, буд. 126, кв. 23	683-77-21	test.pacient.email@gmail.com	denys.ch.uk13@gmail.com
4	Ковальчук Матвій Вікторович	1996.05.25	вулиця Болгарська, буд. 149	146-77-37	test.pacient.email@gmail.com	denys.ch.uk13@gmail.com
6	Коваленко Андрій Сергійович	1988.07.30	вулиця Болгарська, буд. 136, кв. 17	272-59-82	test.pacient.email@gmail.com	denys.ch.uk13@gmail.com
7	Коваленко Андрій Володимирович	1961.01.06	провулок Лінецького, буд. 135	379-66-34	test.pacient.email@gmail.com	denys.ch.uk13@gmail.com
8	Кравченко Олександр Володимирович	1968.08.26	провулок Євгена Чикаленка, буд. 95, кв. 16	436-76-24	test.pacient.email@gmail.com	denys.ch.uk13@gmail.com
9	Савченко Артем Михайлович	1940.08.28	вулиця Ринкова, буд. 99	350-11-49	test.pacient.email@gmail.com	denys.ch.uk13@gmail.com

2.4 Бізнес-процеси програми

Програмна система призначена для управління планом харчування пацієнта на основі його персональної інформації, такої як вік, вага та рівень активності. Система використовує алгоритм для визначення відповідного плану харчування для пацієнта. Значення алгоритму залежать від віку, ваги та рівня активності пацієнта, та необхідні для визначення відповідних споживань калорій та білків. Потім система зберігає план дієти в базі даних для подальшого використання. Додаток надає користувацький інтерфейс для управління інформацією про пацієнта та його дієтичним планом. Користувач може додавати та переглядати інформацію про себе, наприклад, переглядати власну статистику спостережень, зроблених з початку користування додатком та постійно мати можливість оновити її, додавши новий запис на основі власних даних. Користувач також може переглянути рекомендований і призначений план харчування, та зв’язатись з лікарем за допомогою вбудованої функції листування. Система надає набір попередньо визначених типів дієт, включаючи низькокалорійну, висококалорійну та високобілкову дієти. Система прийняття рішень може вибрати відповідний тип дієти для кожного пацієнта, і автоматично розрахувати відповідне споживання калорій і білків на основі особистої інформації пацієнта [5, 12].

Архітектура інформаційно-програмної системи є добре структурованою і призначена для обробки різних бізнес-процесів лікарні. Програмний додаток є модульним, з різними компонентами, що виконують різні функції, що полегшує його підтримку та оновлення в майбутньому.

Щодо інтелектуальних методів аналізу, таких як система прийняття рішень, то вона використовує алгоритм рекомендацій, для визначення відповідної дієти для кожного пацієнта. Система, що використовує даний алгоритм - це тип штучного інтелекту, який використовує алгоритм для прийняття рішень або висновків про певну область. У цьому випадку певні результати та значення під час процесу обрахування алгоритму ґрунтуються на

історії хвороби пацієнта, поточному стані здоров'я та дієтичних уподобаннях, а також на встановлених рекомендаціях щодо харчування. Використовуючи ці значення, додаток може надавати персоналізовані дієтичні рекомендації для кожного пацієнта.

Щоб призначити дієту для пацієнта, додаток дотримується покрокового алгоритму, який можна узагальнити наступним чином:

- Аналізується історія хвороби та поточний стан здоров'я пацієнта, включаючи будь-які алергії, захворювання та ліки, які він приймає.
- Враховуються дієтичні вподобання пацієнта, такі як вегетаріанство, веганство або харчові відрази.
- Вік, стать, вага та рівень активності пацієнта враховуються для визначення його потреб у калоріях та рекомендованого споживання поживних речовин.
- На основі цієї інформації додаток генерує список рекомендованих продуктів і розмірів порцій, беручи до уваги дієтичні вподобання пацієнта і будь-які медичні обмеження.
- Додаток також формує список продуктів, яких слід уникати, на основі історії хвороби пацієнта і будь-яких відомих харчових алергій.
- Рекомендована дієта відображається на екрані та може бути роздрукована для ознайомлення.

Отже, інформаційно-програмний комплекс - це добре розроблений і структурований додаток, який здатний керувати різними бізнес-процесами, включаючи управління пацієнтами, аналіз дієти та фізичних вправ хворих, аналіз отриманих даних щодо вмісту цукру в крові, а також підтримка зв’язку хворих з лікарями. Використання інтелектуальних методів аналізу, таких як системи, з використанням алгоритму рекомендацій, дозволяє додатку надавати персоналізовані дієтичні рекомендації для кожного пацієнта на основі його історії хвороби, поточного стану здоров'я та дієтичних уподобань. Модульний дизайн і використання архітектури MVC роблять додаток більш зручним для обслуговування, тестування і розширення. Загалом, система є ефективним

інструментом для покращення догляду за пацієнтами та оптимізації роботи в дистанційному форматі.

2.5 Математичне та алгоритмічне забезпечення програмної системи

Програмна система надання рекомендацій щодо плану дієти та фізичних вправ використовує декілька математичних та алгоритмічних методів для аналізу даних пацієнта та генерації персоналізованих планів харчування. Детально проаналізуємо математичне та алгоритмічне забезпечення, що використовується в програмі:

1. Одним з ключових математичних методів, що використовуються в системі, є розрахунок індексу маси тіла (ІМТ) пацієнта. ІМТ розраховується шляхом ділення ваги людини в кілограмах на квадрат її зросту в метрах. Отримане значення потім порівнюється зі стандартними значеннями ІМТ, щоб визначити, чи має людина недостатню вагу, нормальну вагу, надлишкову вагу або ожиріння. У програмному коді це реалізовано в методі calculateBMI.
2. Для розрахунку ІМТ використовується рівняння Харріса-Бенедикта, яке враховує вагу, зріст, вік і стать пацієнта. Отримане значення потім множиться на коефіцієнт активності, щоб оцінити загальні добові витрати енергії пацієнта. Це реалізовано в методі calculateBMR.
3. Після обчислення ІМТ пацієнта система використовує алгоритм для створення персоналізованого плану харчування на основі цілей пацієнта (наприклад, схуднення, збільшення або підтримання ваги) та дієтичних обмежень (наприклад, вегетаріанська дієта, безглютенова дієта). Алгоритм враховує TDEE пацієнта і створює дефіцит або надлишок калорій залежно від мети пацієнта. Потім програма розраховує відповідні співвідношення макроелементів (вуглеводів, білків і жирів), виходячи з типу фігури та мети пацієнта. Цей алгоритм реалізовано в методі generateDietPlan.

Окрім описаних математичних методів, програма також використовує кілька

алгоритмів для обробки даних пацієнта і генерації планів харчування. Наприклад:

1. Застосунок має систему з використанням алгоритму рекомендацій, щоб визначити, які продукти підходять пацієнту, виходячи з його дієтичних обмежень. Ця система реалізована в методі checkFood, який приймає на вхід продукт харчування і перевіряє, чи відповідає він дієтичним обмеженням пацієнта. Якщо їжа не підходить, система пропонує альтернативні продукти, які відповідають обмеженням пацієнта.
2. Програма також використовує алгоритм, заснований на даних, щоб рекомендувати їжу на основі вподобань пацієнта. Система веде базу даних продуктів харчування та їх поживної цінності, яка використовується для формування списку рекомендованих продуктів на основі вподобань та дієтичних обмежень пацієнта. Цей алгоритм реалізовано в методі suggestFood.
3. Інший алгоритм, що використовується в програмі - метод Монте-Карло, який використовується для генерації випадкових комбінацій продуктів, що відповідають дієтичним обмеженням пацієнта та співвідношенню макроелементів. Монте-Карло - це статистичний метод, який використовується для розв'язання проблем шляхом моделювання випадкових подій, і часто використовується в ситуаціях, коли детерміновані методи важко застосувати або коли важко отримати точний розв'язок. Метод Монте-Карло генерує велику кількість випадкових комбінацій продуктів і вибирає ту комбінацію, яка найближче відповідає вимогам пацієнта.

Отже, в застосунку реалізовано різні комбінації математичних методів та алгоритмів для аналізу даних пацієнта та формування персоналізованих планів харчування. Ці методи та алгоритми враховують ІМТ, ІМТ, цілі та дієтичні обмеження пацієнта, щоб створити план, який відповідає його харчовим потребам. Використання систем з алгоритмом рекомендацій, алгоритмів на основі даних і моделювання за методом Монте-Карло гарантує, що програма

здатна генерувати різноманітні та персоналізовані дієтичні плани, пристосовані до потреб кожного пацієнта. В методі randomDiet() метод Монте-Карло генерує рандомізований план дієти, моделюючи різні можливі комбінації прийомів їжі та вибираючи ту, яка найкраще відповідає харчовим потребам пацієнта. Метод randomDiet() спочатку обчислює добову потребу пацієнта в калоріях і поживних речовинах на основі його віку, статі, ваги і зросту. Потім він створює порожній об'єкт List, щоб зберігати страви на день. Потім метод використовує метод Монте-Карло для моделювання різних можливих комбінацій прийомів їжі на день. Для цього він випадковим чином вибирає продукти зі списку foodList і додає їх до страв, доки не буде виконано вимоги щодо калорійності та поживності страв. Метод повторює цей процес доти, доки не згенерує три прийоми їжі на день, які відповідають потребам пацієнта [1].

Система також використовує різні математичні функції та алгоритми для розрахунку та аналізу інформації про харчування:

- Метод calculateBMI() обчислює індекс маси тіла пацієнта, який є мірою вмісту жиру в організмі на основі даних про зріст і вагу.
- Метод calculateBMR() обчислює основний метаболізм пацієнта, тобто кількість енергії, що витрачається в стані спокою.
- Метод calculateCalories() обчислює загальну кількість калорій, яку повинен споживати пацієнт, виходячи з його BMR та рівня активності.
- Метод calculateNutrient() обчислює кількість певної поживної речовини, наприклад, білка або жиру, яку пацієнт повинен споживати, виходячи з його потреб у калоріях.

Окрім цих функцій, система також використовує різні структури даних та алгоритми для організації та аналізу даних:

- Об'єкт Dictionary використовується для зберігання та пошуку продуктів харчування та відповідної інформації про їхню поживну цінність.
- Об'єкт List використовується для зберігання страв на день, згенерованих методом randomDiet(). Бібліотека LINQ використовується для запитів та маніпулювання даними в базі даних SQLite.

Ці структури даних та алгоритми дозволяють системі ефективно зберігати, отримувати та аналізувати великі обсяги даних.

2.6 Розробка інформаційного забезпечення

Розробка інформаційного забезпечення для програмної системи рекомендацій обов’язково передбачає створення надійної та логічної бази даних, що на вході зможе зчитувати повну інформація про людину (ПІБ, дата народження, адреса хворого, особистий номер, електронну адресу родича) та перевіряти чи пацієнт ще не був зареєстрований в системі раніше. Після чого взаємодіяти з іншими додатками для формування вихідної інформації, яка складається з рекомендованих пацієнтам планів дієт та фізичних вправ, інформація про прийоми та таблиця спостережень, а також можливість її перегляду.

Визначимо основні додатки та побудуємо їх карту (рисунок 2.2)

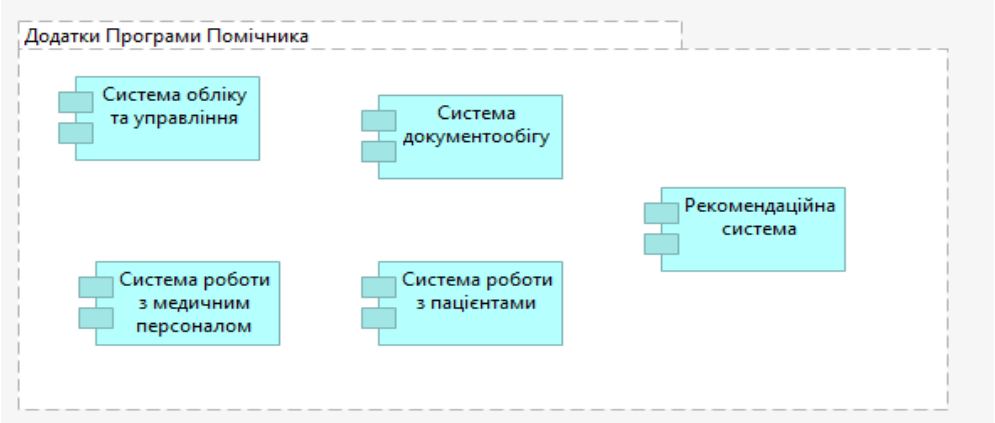


Рисунок 2.2 — Карта додатків

Розглянемо їх взаємодію. (рисунок 2.3)

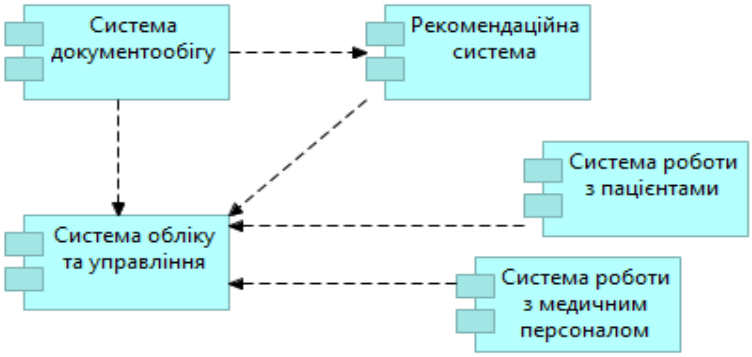


Рисунок 2.3 — Взаємодія додатків

Розглянемо інтеграцію ІС система роботи з пацієнтами хворими на діабет 2-го типу. Система роботи з пацієнтами формує звітність про те, які є плани дієт та спеціально розроблені фізичні вправи, а також індивідуальні рекомендації на початковому прийомі у лікаря. Завдяки якому клієнти дізнаються про наш програмний модуль та отримують змогу скористуватись послугами.

Відповідна діаграма зображені на рисунку 2.4

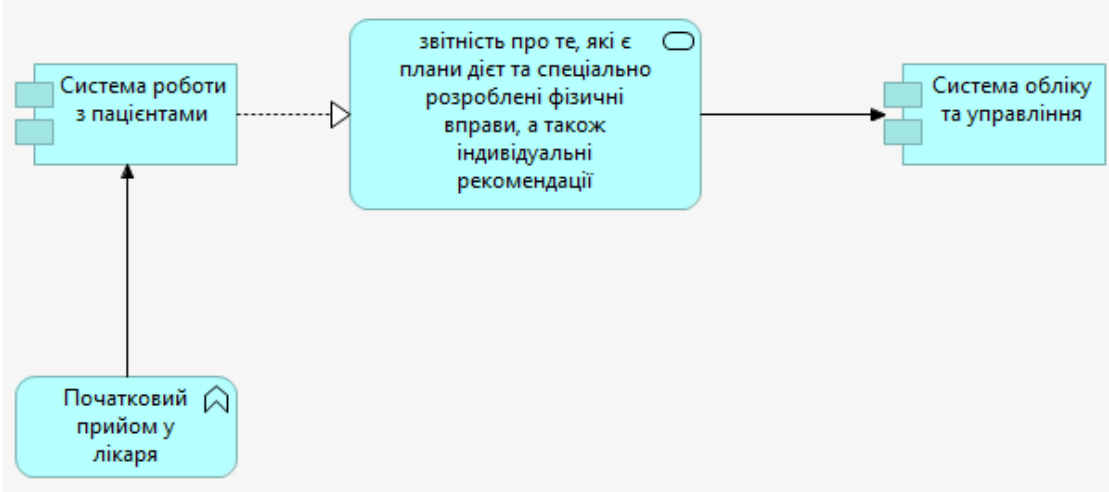


Рисунок 2.4 — Діаграма інтеграції ПЗ

Розробимо структуру рекомендаційної, вона буде складатися з наступних модулів:

- Модуль введення даних пацієнтів
- Модуль обробки даних пацієнтів

- Модуль формування розкладу дієт та вправ задля позитивних змін стану пацієнтів

Відповідна діаграма зображена на рис 2.5

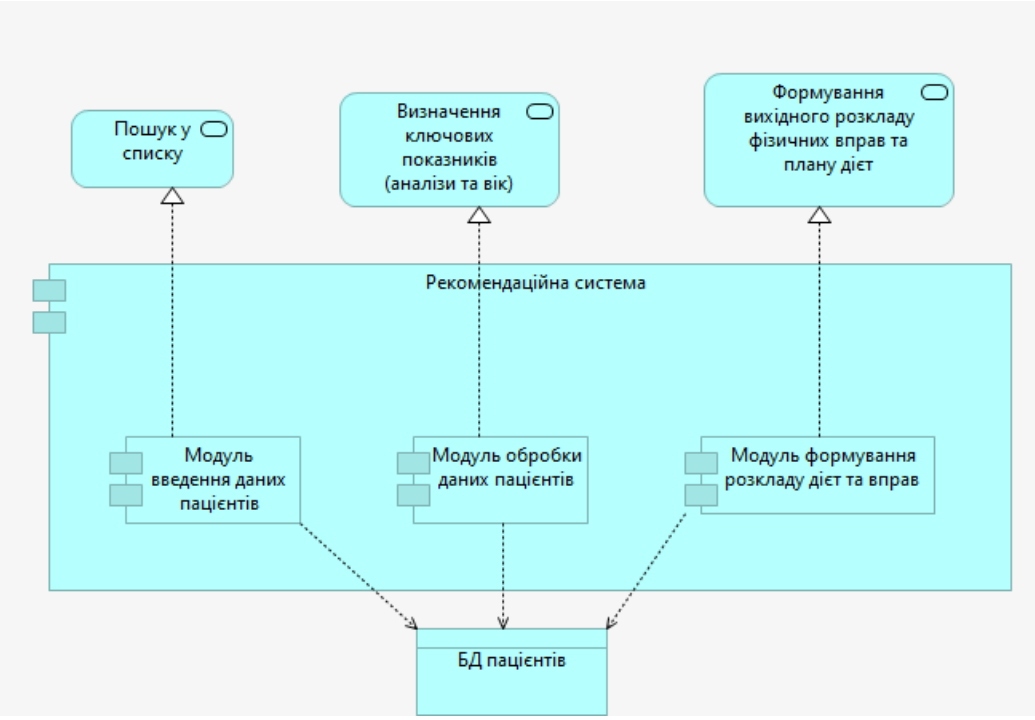


Рисунок 2.5 — Діаграма представлення структури прикладної програмного модуля формування рекомендацій

При розробці структури бази даних інформацію по даній роботі раціонально представити у вигляді сукупності таблиць.

Таблиця dieta містить інформацію про ті продукти, які рекомендовано вживати хворим.

id_dieta	name_dieta	description_dieta
1	Дієта №1	Опис дієти №1
2	Дієта №2	Опис дієти №2

Рисунок 2.6 — Специфікація таблиці dieta

Таблиця doctor містить інформацію про ПІБ лікаря, його посаду та особистий телефон.

id_doctor	name_doctor	job_doctor	phone_doctor
1	Бойко Лев Сергійович	Лікар	489-29-95
2	Коваль Максим Володимирович	Лікар	760-90-94
4	Бондар Лев Олександрович	Старший лікар	751-73-23
NULL	NULL	NULL	NULL

Рисунок 2.7 — Специфікація таблиці doctor

Таблиця exercize містить інформацію про фізичні вправи, спеціально підібрані для хворих.

id_exercize	name_exercize	description_exercize
1	Вправа №1	Опис вправи №1
3	Вправа №3	Опис вправи №3

Рисунок 2.8 — Специфікація таблиці exercize

Таблиця pacient містить особисту інформацію про пацієнтів хворих на діабет. Це стосується ПІБ хворого, дати народження, адреси проживання, мобільного телефону та електронної пошти родича.

id_pacient	name_pacient	date_of_birth	address_pacient	phone_pacient	email_relatives
2	Шевченко Олександр Іванович	1988-07-30 00:00:00	вулиця Скіфська, буд. 190, кв. 74	769-70-06	denys.ch.uk13@gmail.com
3	Руденко Матвій Олександрович	1973-12-04 00:00:00	вулиця Миколи Костомарова, буд. 126, кв. 23	683-77-21	denys.ch.uk13@gmail.com
4	Ковальчук Матвій Вікторович	1996-05-25 00:00:00	вулиця Болгарська, буд. 149	146-77-37	denys.ch.uk13@gmail.com
6	Коваленко Андрій Сергійович	1986-01-14 00:00:00	вулиця Болгарська, буд. 136, кв. 17	272-59-82	denys.ch.uk13@gmail.com
7	Коваленко Андрій Володимирович	1961-01-06 00:00:00	провулок Лінецького, буд. 135	379-66-34	denys.ch.uk13@gmail.com
8	Поліщук Лев Васильович	1996-03-02 00:00:00	вулиця Темерязева, буд. 4	980-55-33	denys.ch.uk13@gmail.com
NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 2.9 — Специфікація таблиці pacient

Таблиця visit містить інформацію про прийом та рекомендації, які потрібно обговорити при зустрічі з пацієнтом.

id_visit	id_pacient	id_doctor	recommendations
1	2	1	Рекомендації щодо чо...
3	2	1	
4	2	1	
5	6	1	
6	2	4	Рекомендації до чогось
NULL	NULL	NULL	NULL

Рисунок 2.10 — Специфікація таблиці visit

Таблиця watch містить інформацію про останні дані пацієнтів, які були оновлені (який саме хворий, дата та час заповнення, рівень цукру в крові, тиск).

id_watch	id_pacient	date	sugar	pressure1	pressure2	id_dieta	id_exersize
1	2	2022-12-09 00:00:00	5.0	120	80	1	1
3	2	2022-12-09 01:07:57	5.0	120	80	1	1
4	2	2022-12-09 01:21:36	5.0	120	80	1	1
5	2	2022-12-09 01:29:52	5.0	120	80	1	1
6	2	2022-12-09 01:30:23	5.0	120	80	1	1
7	2	2022-12-09 01:37:55	5.0	120	80	1	1
8	2	2022-12-09 01:43:04	5.5	120	80	1	1
9	2	2022-12-09 01:43:57	4.4	120	80	1	1
10	4	2022-12-09 01:55:36	5.0	120	80	1	1
11	2	2022-12-09 01:59:48	5.0	120	80	1	1
12	6	2022-12-09 02:07:11	5.0	120	80	1	1
13	2	2022-12-09 02:20:57	0.6	120	80	1	1
14	2	2022-12-09 02:22:32	1.0	120	80	1	1
15	2	2022-12-09 02:22:56	1.0	120	80	1	1
16	2	2022-12-09 02:24:49	5.0	120	80	1	1
17	2	2022-12-09 02:24:57	20.0	120	80	1	1
18	2	2022-12-09 02:26:19	20.0	120	80	1	1
19	2	2022-12-09 02:27:36	5.0	90	80	1	1
20	2	2022-12-09 02:27:47	2.0	80	80	1	1
21	2	2022-12-09 02:31:31	5.0	80	80	1	1
23	2	2022-12-10 02:43:05	5.0	120	80	1	1
25	6	2022-12-10 01:56:58	5.0	120	80	1	1
27	2	2022-12-11 02:41:36	13.0	120	80	1	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 2.11 — Специфікація таблиці watch

Виходячи з структури, яку додатку база даних містить різний тип добре розробленої та пов’язаної інформації. БД містить детальну інформацію про історію пацієнтів з діабетом, оптимальні рішення від різних медичних експертів для різних станів, відповідні вправи та плани дієти відповідно до пацієнта, а також записи відповідних медичних працівників.

Також розробка інформаційного забезпечення передбачає створення концептуальної, логічної та фізичної моделей, що відображають сутності та атрибути предметної області програмної системи (рисунок 2.12, 2.13, 2.14 відповідно). Основна мета проектування моделі даних полягає в тому, щоб переконатися, що об'єкти даних, запропоновані функціональною командою, представлені точно. Спочатку ми повинні почати з концептуальної моделі даних, і в міру того, як з'являється все більше і більше інформації, ми додаємо більше деталей, щоб удосконалити її від концептуальної до логічної моделі. Як

результат, коли ми точно знаємо, як реалізувати базу даних нашої системи, ми можемо вдосконалити нашу логічну модель у фізичну модель даних, яка може безпосередньо відображати діаграму та фактичну систему бази даних.

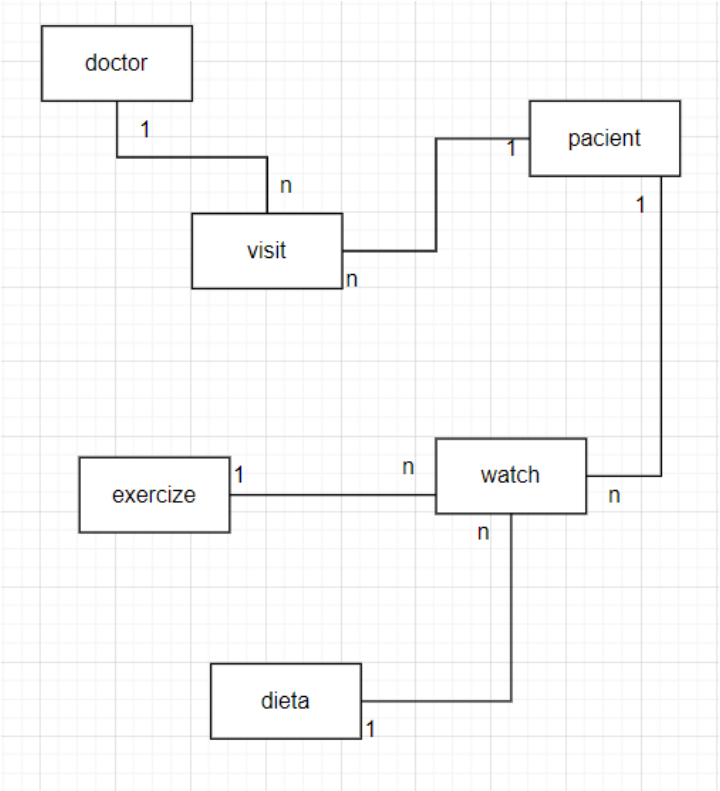


Рисунок 2.12 – Концептуальна модель БД програмної системи рекомендацій

На рисунку 2.12 ми показали основні сутності, з який складається БД програмної системи рекомендацій, а саме: «doctor», «visit», «pacient», «watch», «exercize», «dieta». Кожна з них має індивідуальні атрибути, наприклад, сутність “doctor”, має персональний номер лікаря, його ім’я, посаду та номер телефону. Сутність «watch» містить атрибути, що відповідають за персональний номер пацієнта, персональний номер спостереження, номери дієти та фізичних вправ, а також інформацію про життєво важливі показники (рівень цукру, тиску) та дату, коли ці показники були оновлені.

Наступним кроком, буде створення логічної та фізичної моделей даних. Логічна модель (рисунок 2.13) є деталізованою версією концептуальної та визначає структуру самих даних і зв'язки між різними атрибутами, таблицями та записами, тому, на відміну від концептуальної моделі, тут, ми вже вказуємо

первинні та вторинні ключі, які пов’язують таблиці між собою. Створення логічної моделі даних здійснюється за допомогою інструментів візуалізації, щоб проілюструвати зв'язки між об'єктами в базі даних.

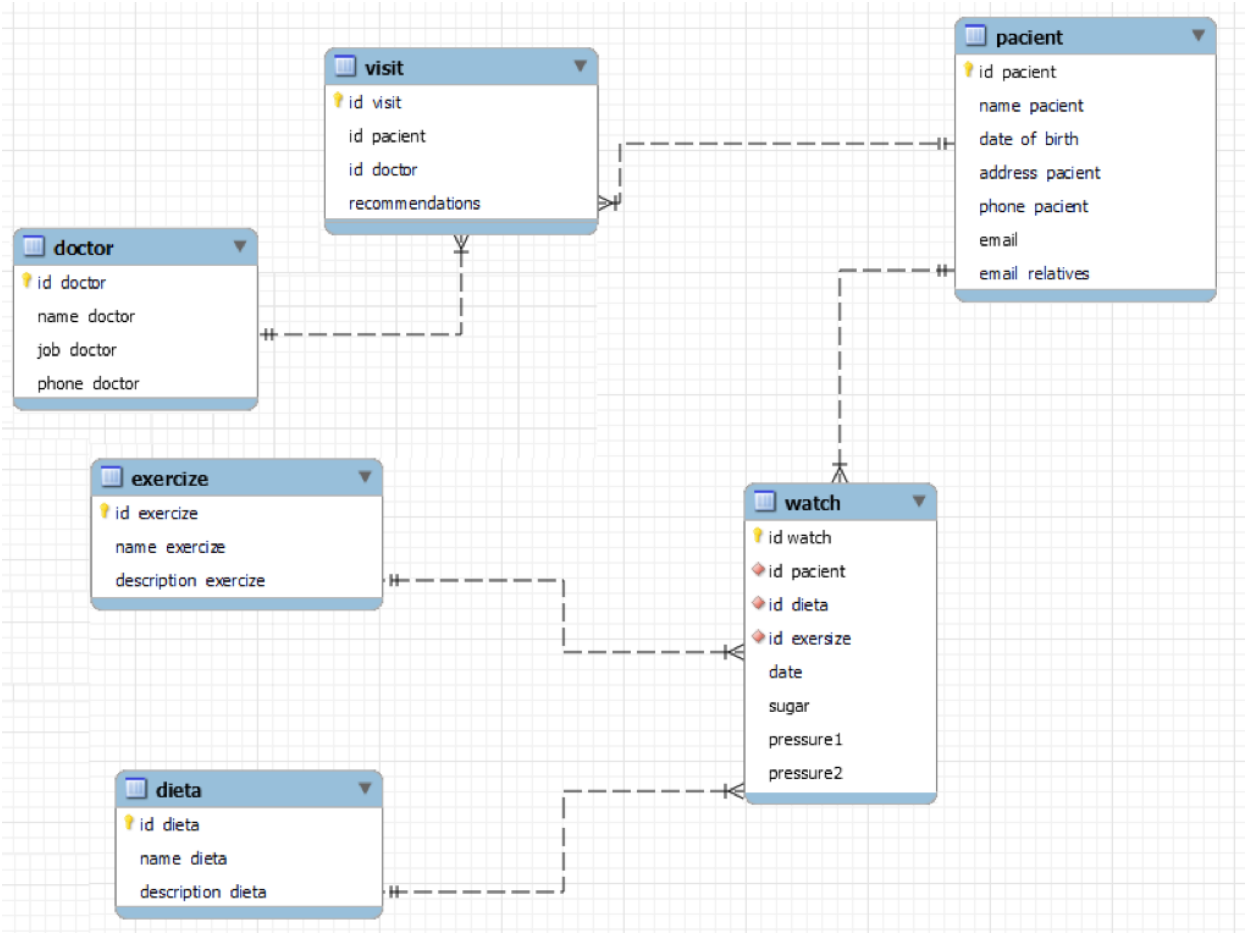


Рисунок 2.13 – Логічна модель БД програмної системи рекомендацій

Фізична модель визначає, як дані зберігаються та управляються на фізичному жорсткому диску пристроїв, на яких працює база даних. Фізична модель дещо відрізняється і значною мірою залежить від структури логічної моделі. (рисунок 2.14)

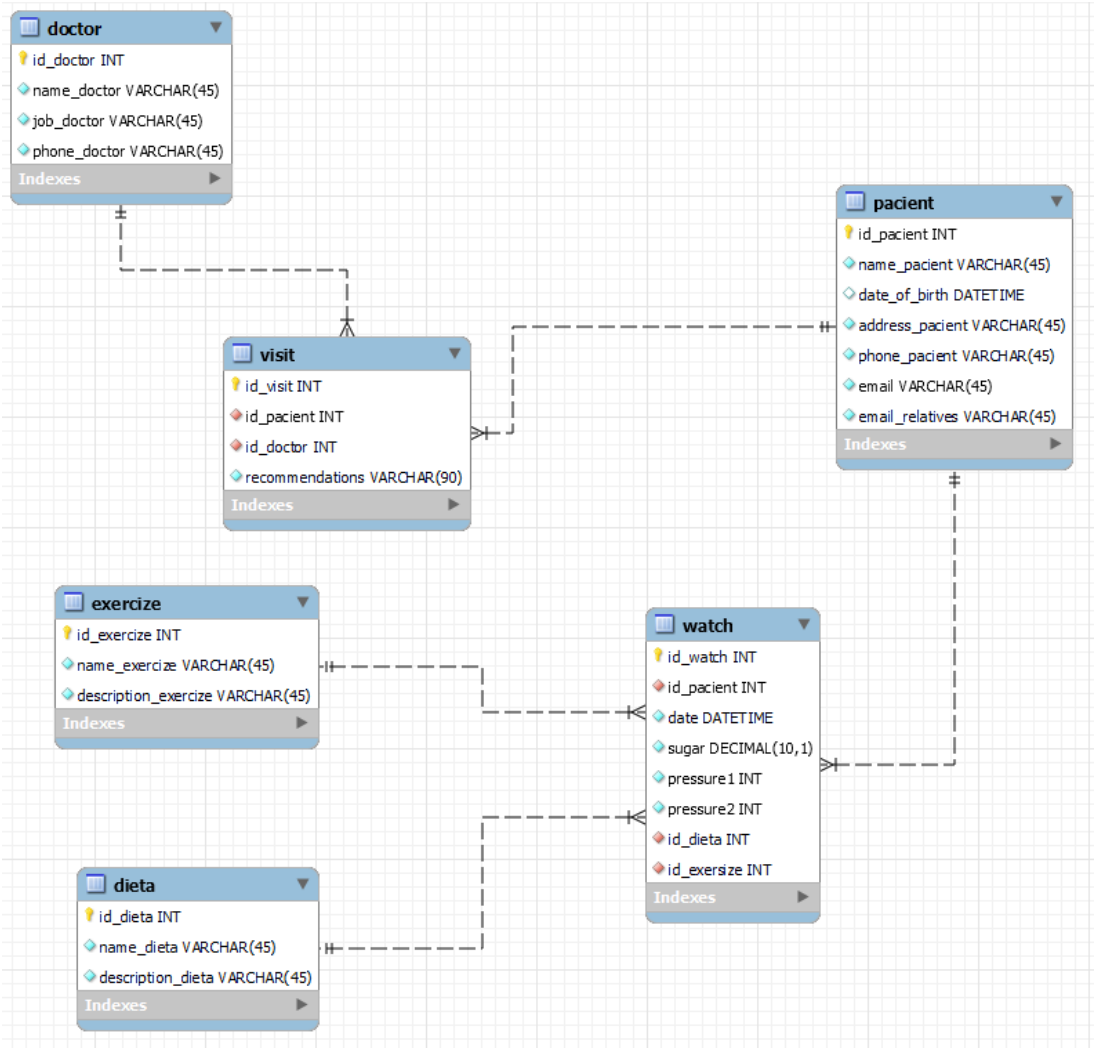


Рисунок 2.14 – Фізична модель БД програмної системи рекомендацій

Фізична модель бази даних показує, як дані зберігаються на жорсткому диску пристрою. Це фактичний код, який буде використовуватися для створення структури бази даних програмної системи рекомендацій, яка включає відповідні таблиці і те, як вони пов'язані одна з одною. У MongoDB це досягається шляхом створення режиму мангуста, однак в даному випадку, оскільки реалізація відбувається через MySQL, будуть використовуватися SQL запити для створення бази даних з таблицями.

Таблиця 2.2 Сутності БД та їх зв’язки

№	Сутності БД, між якими утворюється зв’язок	Тип зв’язку	Опис зв’язку між зазначеними сутностями
1	Doctor - visit	Один до багатьох	Одна лікар надає багато консультацій та рекомендацій різним пацієнтам
2	Pacient - visit	Один до багатьох	Один пацієнт має багато візитів у лікаря
3	Pacient - watch	Один до багатьох	Один пацієнт має список спостережень, що складається з багатьох індивідуальних записів.
4	Exercize - watch	Один до багатьох	В залежності від багатьох індивідуальних записів пацієнта, програмна система рекомендує один найкращий список фізичних вправ, розроблений під конкретного хворого.
5	Dieta - watch	Один до багатьох	В залежності від багатьох індивідуальних записів пацієнта, програмна система рекомендує одну найкращу дієту, розроблену під цього пацієнта.

Таким чином, ми виконали проектування моделі даних та переконалися, що об'єкти даних, запропоновані функціональною командою, представлені точно.

2.7 Висновок до другого розділу

В даному розділі дипломної роботи, було проведено аналіз методів інтелектуального аналізу та функціональні можливості застосунку. В результаті чого було створений для спрощення життя хворих на діабет 2-го типу відповідні класи, кожен з яких має певні функціональні можливості. Основною метою програми є управління записами пацієнтів, включаючи особисту інформацію, історію хвороби та рекомендації щодо дієти. Також описано структуру інформаційної технології обробки даних, архітектуру інформаційно-програмної системи та, звичайно ж, математичне та алгоритмічне забезпечення системи програмного застосунку. Додаток використовує алгоритм, заснований на даних,

щоб рекомендувати їжу на основі вподобань пацієнта, де система веде базу даних продуктів харчування та їх поживної цінності, яка використовується для формування списку рекомендованих продуктів на основі вподобань та дієтичних обмежень пацієнта.

ТРЕТІЙ РОЗДІЛ. РЕАЛІЗАЦІЯ ПРОГРАМНОЇ СИСТЕМИ НАДАННЯ РЕКОМЕНДАЦІЙ ЩОДО РЕАБІЛІТАЦІЇ ХВОРИХ ТА ЇЇ ТЕСТУВАННЯ

3.1 Структура програмного коду застосунку

Програмний код застосунку складається з 6 спеціальних форм (Form1.cs, Form2.cs, …, Form6.cs) в яких реалізовано файлове підключення до бази даних MySQL, методи безпеки, автентифікації користувачів, алгоритми систем прийняття рішень та додаткових файлів запуску застосунку (program.cs, DBconnect.cs), генерації випадкових вхідних даних для перевірки працездатності системи (Randomizer.cs), а також файли для реалізації інтерфейсу (про це пізніше):

1. DBconnect.cs

- Метою цієї частини програмного коду є створення з'єднання з базою даних MySQL. Тут використовується бібліотека MySQL Connector/NET для взаємодії з базою даних MySQL. У коді визначено клас з іменем "DBconnect" з загальнодоступним статичним методом з іменем "connect". Цей метод повертає об'єкт MySqlConnection, який створюється за допомогою рядка з'єднання, що складається з імені хоста, імені бази даних, імені користувача та пароля.

2. Form1.cs

- Метою частини коду Form2.cs є надання Windows Form для автентифікації користувачів. У коді міститься клас з іменем "Form1", який успадковується від класу Form (дана форма містить елемент управління DataGridView, який відображає список користувачів з бази даних MySQL), декілька статичних полів для зберігання об'єктів, пов'язаних з базою даних, таких як MySqlConnection, MySqlCommand і MySqlDataAdapter (ці поля використовуються різними методами класу, такими як "selectAll" і "button2_Click").

- Метод selectAll відповідає за вибірку всіх рядків з таблиці user в базі даних MySQL і заповнення даними елемента управління DataGridView. Метод

button2_Click викликається, коли користувач натискає кнопку "Вхід". Він перевіряє, чи збігаються введені ім'я користувача та пароль з обліковими даними адміністратора, і якщо збігаються, то відкриває новий екземпляр Form2 з правами адміністратора. В іншому випадку він шукає користувача з введеними обліковими даними в елементі управління DataGridView і відкриває новий екземпляр Form2 з привілеями звичайного користувача.

3. Form2.cs

- Ця частина коду визначає Form2 інтелектуальної системи прийняття рішень та панель адміністратора. Вона імпортує кілька просторів імен, включаючи System, MySql.Data.MySqlClient і Microsoft.VisualBasic. Також оголошуються змінні admin, userID, form1, form4, form5, form6 і email_pacient.
- Конструктор для Form2 отримує три аргументи: form1, admin та userID. Вони зберігаються в однойменних змінних-членах класу. Конструктор також викликає метод InitializeComponent().
- Методи *updateDieta ma updateExersize* отримують два цілочисельні аргументи і оновлюють відповідні значення в таблиці ISPPR.watch бази даних MySQL. Ці методи виконують SQL-запит для оновлення вказаних значень для заданого id_pacient (ідентифікатор пацієнта). Метод оновлення оновлює дієти і вправи пацієнтів на основі їхнього віку. Спочатку він отримує ідентифікатори дієт і вправ з об'єктів dataGridView1 і dataGridView3 відповідно. Потім він циклічно переглядає кожен рядок об'єкта dataGridView4 (який містить інформацію про пацієнта) і обчислює вік пацієнта на основі його дати народження. Якщо вік пацієнта потрапляє в певний діапазон, його дієта і план фізичних вправ оновлюються до відповідних значень з масивів id_dieta і id_vprava за допомогою методів updateDieta і updateExersize. Метод *pacientsToDictionary()* створює новий словник під назвою pacients, де кожен ключ є цілим числом, що представляє ідентифікатор пацієнта, а кожне значення є рядком, що представляє інформацію про пацієнта (наприклад, ім'я, вік, адреса і т.д.). Цей метод ітераційно переглядає рядки об'єкта dataGridView4, отримує відповідну інформацію з кожної комірки рядка і додає пару ключ-значення до

словника пацієнтів. Метод *doctorsToDictionary()* подібний до pacientsToDictionary(), але створює словник з назвою doctors, де кожен ключ - це ідентифікатор лікаря, а кожне значення - рядок, що представляє інформацію про лікаря. Метод *exercisesToDictionary*() подібний до pacientsToDictionary() та doctorsToDictionary(), але створює словник з назвою exercises, де кожен ключ є ідентифікатором вправи, а кожне значення - рядком, що представляє інформацію про вправу. *dietasToDictionary*():схожий на попередні три методи, але створює словник з назвою dietas, де кожен ключ є ідентифікатором дієти, а кожне значення - рядком, що представляє інформацію про дієту. Метод *selectAll():* вибирає всі рядки з вказаної таблиці в базі даних MySQL і заповнює даними вказаний об'єкт DataGridView. Спочатку метод встановлює з'єднання з базою даних за допомогою методу DBconnect.connect(), створює об'єкт MySqlCommand з оператором SELECT * FROM для вказаної таблиці, виконує команду і зберігає результат в об'єкті DataTable. Метод створює об'єкт BindingSource, який прив'язується до DataTable, і встановлює властивість DataSource вказаного DataGridView у значення BindingSource. Метод *delete():* дозволяє користувачеві видаляти вибрані рядки з вказаної таблиці в базі даних MySQL. Спочатку перевіряє, чи є вибрані рядки у вказаному об'єкті DataGridView. Якщо жодного рядка не вибрано, він виводить вікно повідомлення з проханням вибрати рядок. Якщо рядки вибрано, метод виконує ітерацію по кожному вибраному рядку, отримує ідентифікатор пов'язаного з ним запису у вказаній таблиці і створює оператор DELETE для цього запису. Метод виводить вікно повідомлення з проханням підтвердити видалення, і якщо користувач підтверджує видалення, він виконує оператор DELETE і видаляє рядок з об'єкта DataGridView. Метод *Form2_Load()* викликається при завантаженні форми і викликає метод selectAll() для кожного з шести об'єктів DataGridView, щоб заповнити їх даними з відповідних таблиць бази даних MySQL. Метод також викликає методи pacientsToDictionary() і doctorsToDictionary() для заповнення словників пацієнтів і лікарів відповідно [1].

- Обробник події *button1_Click* відкриває діалогове вікно і пропонує

користувачеві ввести назву та опис нової дієти, які потім вставляються в таблицю dieta бази даних. Функції selectAll і dietasToDictionary викликаються для оновлення dataGridView1 і словника dietas новими даними. Обробник події *button2_Click* видаляє вибраний рядок з dataGridView1 і відповідний запис з таблиці dieta в базі даних. Функція dietasToDictionary викликається для оновлення словника dietas новими даними. Обробник події *button4_Click* відкриває діалогове вікно і пропонує користувачеві ввести ім'я, посаду і номер телефону нового лікаря, які потім вставляються в таблицю doctor бази даних. Функції selectAll і doctorsToDictionary викликаються для оновлення dataGridView2 і словника лікарів новими даними. Обробник події *button3_Click* видаляє вибраний рядок з dataGridView2 і відповідний запис з таблиці doctor в базі даних. Викликається функція doctorsToDictionary для оновлення словника лікарів новими даними. Обробник події *Form2_FormClosed* викликається при закритті форми Form2 і показує форму Form1. *button6_Click* обробляє натискання кнопки "Додати вправу". Він пропонує користувачеві ввести назву та опис вправи, а потім вставляє ці дані в таблицю "ISPPR.exercise" в базі даних за допомогою оператора SQL INSERT. Потім метод викликає метод selectAll, щоб оновити елемент управління "dataGridView3" новими даними вправи, і метод exercisesToDictionary, щоб оновити внутрішній словник назв та ідентифікаторів вправ.*button5_Click* обробляє натискання кнопки "Видалити вправу". Він викликає метод delete, передаючи ім'я таблиці "exercise" і елемент управління "dataGridView3", щоб видалити вибрану вправу з бази даних. Потім він оновлює внутрішній словник вправ, викликаючи метод exercisesToDictionary. *button8_Click* обробляє натискання кнопки "Додати пацієнта". Він пропонує користувачеві ввести інформацію про нового пацієнта, включаючи ім'я, дату народження, адресу, номер телефону, електронну пошту та електронну пошту родича. Потім він вставляє ці дані в таблицю "ISPPR.pacient" в базі даних за допомогою оператора SQL INSERT. Потім метод викликає метод selectAll, щоб оновити елемент управління "dataGridView4" новими даними про пацієнта, і метод pacientsToDictionary, щоб оновити внутрішній словник імен та

ідентифікаторів пацієнтів. *button7_Click* обробляє натискання кнопки "Видалити пацієнта". Він викликає метод delete, передаючи ім'я таблиці "pacient" і елемент управління "dataGridView4", щоб видалити вибраного пацієнта з бази даних. Потім він оновлює внутрішній словник пацієнтів, викликаючи метод pacientsToDictionary. Обробники подій *button10_Click і button12_Click* обробляють натискання кнопок "Додати візит" і "Додати дієту" відповідно. Вони приховують поточну форму і показують нову форму (Form4 або Form5), щоб дозволити користувачеві ввести додаткову інформацію про візит або дієту. Обробники *button11_Click та button9_Click* обробляють натискання кнопок "Видалити годинник" та "Видалити візит" відповідно. Вони обидва викликають метод delete для видалення вибраного запису з таблиці "годинник" або "візит" відповідно [1, 12].

4. Form3.cs

- Дана форма програмного застосунку визначає користувацький інтерфейс інтелектуальної системи прийняття рішень в екстрених ситуаціях. Форма 3 дозволяє користувачеві вибрати пацієнта зі списку, що випадає, і надіслати електронною поштою сповіщення про термінову потребу в медичній допомозі одержувачу. Код також дозволяє користувачеві перейти до іншої форми для подальших дій. Клас успадковується від класу "Form" у просторі імен "System.Windows.Forms" та має три приватні поля: form1, form2 та form5, які є екземплярами класу "Form".
- Конструктор класу "Form3" отримує аргумент типу "Form" з іменем "form1" та ініціалізує поле "form1" цим аргументом. Метод "Form3_Load" ініціалізує поле "form2" новим екземпляром класу "Form2", а потім викликає метод "Show" для відображення форми користувачеві. Після цього викликається метод "Close" для негайного закриття форми. Елемент управління comboBox1 заповнюється іменами пацієнтів, що зберігаються у статичному словнику з іменем pacients класу Form1. Властивість Text елементу управління comboBox1 встановлюється в значення першого елементу колекції Items.
- Метод *button1_Click* відправляє електронний лист за протоколом

SMTP через Gmail. Лист містить попередження про термінову необхідність надання медичної допомоги пацієнту. Відправник і одержувач листа встановлені на одну адресу Gmail. Метод *"button2_Click"* створює новий екземпляр "Form5" і показує його користувачеві. Метод "Form3_FormClosed" викликається, коли форма "Form3" закривається, що робить поле "form1" видимим.

5. Form4.cs

- Дана форма програмного застосунку демонструє, як Form4 можна використовувати для додавання нових записів про відвідування до бази даних в системі ISPPR.
- Оператори використання імпортують необхідні простори імен для функціонування форми. Клас визначено як Form4, який успадковується від класу Form. Конструктор Form4 отримує параметр Form, який зберігається у змінній form2.
- Обробник події *Form4_Load* заповнює поля comboBox1 та comboBox2 іменами пацієнтів та лікарів відповідно, які отримуються зі словників Form1.pacients та Form1.doctors. Обробник події *button1_Click* витягує ідентифікатори вибраних пацієнтів і лікарів з відповідних словників, а потім будує SQL-команду для вставки нового запису про візит в базу даних. Команда виконується з використанням об'єктів mysql_query і mysql_connection, визначених у Form1. Після успішного додавання запису відображається вікно повідомлення для інформування користувача, а форма закривається. Обробник події *Form4_FormClosed* показує форму Form2, яка була передана як параметр у конструкторі.

6. Form5.cs

- Form5 також має конструктор, який приймає екземпляр Form2 як параметр і встановлює його у поле form2 та визначає обробники подій і логіку для елементів управління.
- Метод *sendMail* створює екземпляр SmtpClient і встановлює налаштування SMTP за допомогою об'єкта NetworkCredential, який містить адресу електронної пошти та пароль відправника. Потім він створює новий об'єкт

MailMessage і встановлює його властивості From, To, Subject і Body. Метод *checkForHelp*, який приймає три параметри: sugar (double), pressure1 (int) і pressure2 (int). Цей метод повертає булеве значення, яке вказує на те, чи потребує пацієнт невідкладної медичної допомоги. Метод перевіряє декілька умов, пов'язаних з рівнем цукру та кров'яним тиском, і повертає true, якщо будь-яка з них виконується. Метод, *numericUpDown2_ValueChanged*, спрацьовує, коли значення numericUpDown2 змінюється. Він встановлює мінімальне значення numericUpDown3, яке дорівнює новому значенню numericUpDown2. Метод, *numericUpDown3_ValueChanged*, спрацьовує, коли змінюється значення numericUpDown3. Він перевіряє, чи нове значення numericUpDown3 більше за поточне значення numericUpDown2. Якщо так, то він встановлює значення numericUpDown3 рівним numericUpDown2 [12].

- У класі Form5 є обробник події *FormClosed*, який показує екземпляр форми form2, коли форму Form5 закрито. Обробник події Form5_Load ініціалізує елемент управління DateTimePicker, встановлює значення comboBox1 на ім'я пацієнта, відключає comboBox1, заповнює comboBox2 і comboBox3 значеннями словників Form1.dietas і Form1.exercises відповідно, і встановлює вибраний елемент обох словників на перший рядок у відповідних списках. *ComboBox1_Click, comboBox1_SelectedIndexChanged ma button1_Click*, є обробниками подій для елементів управління comboBox1, comboBox2 та comboBox3 відповідно. *ComboBox1_SelectedIndexChanged і button1_Click* містять логіку для отримання вибраних значень з комбобоксів і використання їх для виконання запиту до бази даних для отримання дати народження вибраного пацієнта. Потім код обчислює вік пацієнта на основі отриманої дати народження та поточної дати, і відображає вік пацієнта у вікні повідомлення.

7. Form6.cs

- Дана форма програмного застосунку визначає клас з іменем Form6, який успадковується від класу Windows Form. Призначенням цього класу є надсилання електронного повідомлення на вказану адресу електронної пошти з використанням облікового запису Gmail.

- Клас має наступні інструкції використання:

a) System: містить фундаментальні типи та класи, які визначають загальновживані типи даних значень та посилань, події та обробники подій, інтерфейси, атрибути та винятки обробки.
b) System.Collections.Generic: містить інтерфейси та класи, що визначають загальні колекції, які дозволяють ефективно зберігати та вилучати елементи, не порушуючи при цьому типологічну безпеку.
c) System.ComponentModel: містить класи, які використовуються для реалізації поведінки компонентів та елементів керування під час виконання та проектування.
d) System.Data: надає доступ до класів, що представляють архітектуру ADO.NET, яка використовується для доступу та маніпулювання даними з джерела даних.
e) System.Drawing: надає доступ до базової графічної функціональності GDI+.
f) System.Linq: надає класи та інтерфейси, які підтримують запити, що використовують Language-Integrated Query (LINQ).
g) System.Net: надає простий програмний інтерфейс для багатьох протоколів, що використовуються у сучасних мережах, зокрема HTTP, FTP та SMTP.
h) System.Net.Mail: містить класи, які дозволяють програмам надсилати електронну пошту за допомогою простого протоколу передачі пошти (SMTP).
i) System.Text: містить класи, які представляють кодування тексту ASCII, Unicode, UTF-7 та UTF-8.
j) System.Threading.Tasks: містить типи, які спрощують роботу з написання паралельного та асинхронного коду.
k) System.Windows.Forms: надає класи для створення Windows-додатків, які використовують всі переваги багатого користувацького інтерфейсу, доступного в операційній системі Microsoft Windows [12, 13].

- Клас має конструктор, який отримує два параметри: form2, який є

екземпляром класу Form, та email_pacient, який є рядком, що зберігає адресу електронної пошти одержувача. Конструктор ініціалізує поля form2 і email_pacient значеннями відповідних параметрів і викликає метод InitializeComponent.

- У класі є метод *sendMail*, який надсилає повідомлення електронної пошти вказаному одержувачу за допомогою облікового запису Gmail. Метод спочатку отримує адресу електронної пошти одержувача з елемента управління textBox1. Потім він створює екземпляр класу SmtpClient, встановлює властивість UseDefaultCredentials в false і створює екземпляр класу NetworkCredential, який зберігає адресу електронної пошти і пароль відправника. Потім він встановлює властивості Port, EnableSsl, Host і Credentials екземпляра SmtpClient і створює екземпляр класу MailMessage, який зберігає адресу електронної пошти відправника, адресу електронної пошти одержувача, тему листа і тіло листа. Як результат, він викликає метод *Send* екземпляра SmtpClient для надсилання електронного повідомлення і виводить вікно повідомлення, яке показує, чи було надіслано повідомлення успішно чи ні.

- Клас має три обробники подій: обробник події *button1_Click* викликається, коли користувач клацає на елементі управління button1. Він викликає метод sendMail, показує екземпляр form2 і закриває поточну форму. Обробник події *Form6_Load* викликається при завантаженні форми. Він ініціалізує елемент управління textBox1 адресою електронної пошти за замовчуванням, а елемент управління textBox2 - адресою електронної пошти одержувача. Обробник події *Form6_FormClosed* викликається при закритті форми. Він показує екземпляр форми form2.

8. Randomizer.cs

- Дана форма програмного застосунку визначає клас під назвою Randomizer, який містить декілька статичних методів, що генерують випадкові дані. Клас Randomizer має статичну змінну rnd типу Random, яка використовується для генерації випадкових чисел. Він також містить декілька статичних масивів рядків, включаючи names1, names2, names3, jobs, and streets,

які містять різні типи даних, такі як імена, назви посад та назви вулиць.

- Перший метод *randomName()* повертає випадкове ім'я, згенероване шляхом конкатенації випадкового елемента з масивів names1, names2 та names3. Другий метод *randomPhone()* генерує випадковий телефонний номер шляхом конкатенації цифр і символу тире після кожної другої цифри. Згенерований номер телефону буде мати формат 1-234-567. Третій метод *randomDate()* генерує випадкову дату між 1 січня 1950 року та 31 грудня 2000 року у форматі yyyy-MM-dd. Четвертий метод *randomJob()* повертає випадкову назву посади з масиву jobs. П'ятий метод *randomAddress()* генерує випадкову адресу вулиці шляхом конкатенації випадкового елемента з масиву streets, випадкового номера будинку від 1 до 200 та необов'язкового номера квартири, якщо випадкове логічне значення, згенероване rnd.Next(0,2), дорівнює 0.

Ідея коду полягає в тому, щоб надати простий застосунок, який має можливість генерувати випадкові дані про пацієнтів для тестування в медичній клініці або подібних установах. А в звичному режимі, допомагати хворим за допомогою свого функціоналу. Програма дозволяє реєструвати нескінченну кількість пацієнтів, щоб надалі системи прийняття правильних рішень могли допомагати в рекомендаціях дієти та плану фізичних вправ в залежності від індивідуальних даних пацієнтів. Крім того, програмний код забезпечує функціонал на випадок екстрених випадків.

У підсумку, наданий код є добре структурованим і функціональним консольним додатком, який генерує випадкові дані про пацієнтів. Методологія коду базується на ООП, де кожен клас представляє об'єкт, який взаємодіє з іншими об'єктами для виконання певних завдань. Алгоритми, що використовуються в коді, покладаються на генерацію випадкових чисел та маніпуляції з рядками. Мета коду - створити простий і зручний інструмент для роботи з даними пацієнтів для використання в медичних установах.

3.2 Структура програмного коду інтерфейсу застосунку

Програма за допомогою свого зручного і простого інтерфейсу надає зберігання інформації про пацієнта звичайним способом і об’єднує пацієнтів і медичних працівників. Обидві служби використовують одну комунікаційну платформу та обробляють ту саму інформацію в потрібний час для прозорої спільної роботи між кожним членом організації:

- Моніторинг. Безперервний (онлайн) моніторинг життєво важливих показників, наприклад рівень глюкози в крові, тиск.
- Комунікабельність/доступність. Усі виміряні показники життєдіяльності повинні спочатку надсилатися до бази даних, і якщо значення є ненормальним у порівнянні з попередньо встановленим значенням у базі даних, сигнал має бути направлено для надсилання тривоги заздалегідь визначеним постачальникам послуг.
- Знання та прийняття рішень. Журнал спостережень в якому можна зберігати історію постійно оновлюваних даних пацієнтів як профіль для підтримки прийняття рішень та порад від постачальників послуг, а також для отримання інформації про фактичний стан здоров’я громадянина. Ставлення діагнозів, виявлення тенденцій і реагування на них.
- Заспокоює рідних. Програма надає психологічну підтримку для спілкування рідним, оскільки в разі чого вони миттєво отримають повідомлення про надзвичайну ситуацію.

Розглянемо детальніше, які елементи програмної системи відповідають за реалізацію надійного інтерфейсу:

1. Designer1.cs

Інтерфейс додатку створено за допомогою Windows Forms, який є фреймворком графічного інтерфейсу користувача (GUI), що надається компанією Microsoft. Код інтерфейсу написаний на мові C#, а сам інтерфейс спроектований за допомогою дизайнерського інструменту Visual Studio. Інтерфейс складається з форми з іменем Form1, яка містить екран для входу

користувача. Форма має фонове зображення логотипу та два текстових поля для введення імені користувача та паролю. Форма також містить кнопку для надсилання інформації для входу та подання у вигляді сітки даних для відображення даних.

Файл Designer1.cs містить автоматично згенерований код інтерфейсу, який включає метод InitializeComponent, що ініціалізує всі компоненти форми і встановлює їх властивості, такі як розмір, розташування і текст. Цей метод викликається в конструкторі форми. Метод обробника події *button2_Click* викликається, коли користувач натискає кнопку "ОК". Цей метод перевіряє, чи збігаються введені ім'я користувача та пароль з обліковими даними, що зберігаються в базі даних, і якщо збігаються, то відображає дані у вигляді сітки даних. Метод *Form1_Load* викликається при завантаженні форми, він ініціалізує представлення сітки даних і встановлює його властивості. Метод Form1_VisibleChanged викликається, коли форма стає видимою, і очищає текстові поля та подання сітки даних.

2. Designer2.cs

Інтерфейс, створений кодом, є додатком Windows Form, який відображає на формі шість DataGridView (таблиць), кожна з яких містить різні дані. Перші три DataGridView відображаються в лівій частині форми, а останні три DataGridView відображаються в правій частині форми. DataGridView заповнюються даними з бази даних за допомогою коду в методі Form2_Load. Дані можна оновити, натиснувши на кнопку "Оновити дані", яка викликає метод UpdateData. Кнопка "Додати дієту" використовується для додавання нової дієти до бази даних.

Кнопка 2 видаляє план дієти, кнопка 3 видаляє лікаря, кнопка 4 додає лікаря, кнопка 5 видаляє вправу, а кнопка 6 додає вправу. Кнопка 7 видаляє пацієнта, кнопка 8 додає пацієнта, кнопка 9 видаляє візит до лікаря, а кнопка 10 додає візит до лікаря. Кнопки розташовані у вигляді сітки за допомогою властивості TabIndex. Властивість TabIndex використовується для визначення порядку вибору кнопок, коли користувач натискає клавішу Tab. Порядок

розташування кнопок важливий з точки зору доступності, оскільки він дозволяє користувачам переміщатися по інтерфейсу, використовуючи лише клавіатуру.

Інтерфейс також містить 13 кнопок, кожна з яких представлена елементом управління Button. Кнопки розташовані під сітками даних і, використовуються для виконання різних дій над даними, таких як додавання або видалення даних, надсилання електронного листа лікарю тощо. Елементи керування кнопок мають декілька властивостей, які задаються в коді, включаючи розташування на формі, розмір кнопки, текстовий напис і унікальний індекс вкладки. Індекс вкладки визначає порядок доступу до кнопок за допомогою клавіатури. Інтерфейс також має деякі додаткові властивості, такі як розмір вікна, можливість розгортання та згортання вікна, а також позиція вікна при першому відображенні. Вікно також має обробники подій, які реагують на різні дії, наприклад, коли вікно завантажується або закривається.

Загалом, інтерфейс добре організований і простий у використанні, з чіткими підписами для кнопок і сіток даних. Код був написаний з використанням Microsoft .NET Framework та інтерфейсу прикладного програмування (API) Windows Forms.

3. Designer3.cs

У цьому коді визначено частковий клас Form3, який представляє інтерфейс Windows Form. Інтерфейс складається з елементу управління ComboBox, двох елементів управління Button та Label. Клас Form3 містить метод Dispose(), який звільняє ресурси, що використовуються формою. Він також має метод InitializeComponent(), який ініціалізує компоненти форми, встановлюючи їх властивості та обробники подій.

Елемент управління ComboBox відображає список варіантів для вибору користувачем. Елемент управління button з іменем button1 розташований у нижній частині форми і не має ніякого конкретного призначення, окрім як бути стандартним елементом управління кнопкою. Кнопковий елемент управління з іменем button2 розташований у верхній частині форми, і при натисканні на нього викликається метод обробника події button2_Click(), який, як видається, додає

спостереження до запису про пацієнта. Загалом, цей код генерує простий інтерфейс для додавання спостережень до картки пацієнта.

4. Designer4.cs

Цей код визначає інтерфейс Windows Form для керування медичними візитами. Форма містить наступні елементи управління:

- ComboBox1 - список пацієнтів.
- ComboBox2 - список лікарів.
- TextBox1 - текстове поле для введення рекомендацій.
- Button1 - кнопка для додавання візиту.
- Label1 - мітка для відображення тексту "Пацієнт" (що означає "Пацієнт" українською мовою).
- Label2 - мітка для відображення тексту "Лікар" (українською мовою "Лікар").
- Label3 - мітка для відображення тексту "Рекомендації" (українською мовою - "Рекомендації").

Форма призначена для того, щоб користувач міг вибрати пацієнта і лікаря зі списків, ввести рекомендації в текстове поле і додати новий візит, натиснувши кнопку "Додати відвідування" (що означає "Додати візит"). Форма також має обробники подій "FormClosed" та "Load".

5. Designer5.cs

В даному коді комбіновані поля дозволяють користувачеві вибирати елементи зі списку варіантів. Перше комбіноване поле (comboBox1) є найважливішим, оскільки воно має обробник подій як для натискання на нього, так і для зміни вибраного індексу. Це означає, що його вміст може впливати на інші елементи керування на формі або викликати якусь іншу дію.

Пікер дати (dateTimePicker1) дозволяє користувачеві вибрати дату. Він налаштований на використання користувацького формату, що може бути важливим для перевірки даних або інтеграції з іншими системами. Числові перемикачі дозволяють користувачеві вводити числові значення. Перший числовий перемикач (numericUpDown1) має десятковий знак після коми і

максимальне значення 20. Другий числовий перемикач (numericUpDown2) має максимальне значення 200 і обробник події, коли його значення змінюється.

В інтерфейсі присутні такі компоненти

- comboBox1: комбіноване поле, яке використовується для вибору пацієнта.
- dateTimePicker1: Пікер для вибору дати та часу спостереження.
- comboBox2: Комбіноване поле для вибору типу дієти.
- comboBox3: Комбіноване поле для вибору типу вправ.
- numericUpDown1: Числовий перемикач вгору-вниз, який використовується для введення значення глюкози в крові.
- numericUpDown2: Числовий перемикач, який використовується для введення значення систолічного артеріального тиску.
- numericUpDown3: Числовий перемикач, який використовується для введення значення діастолічного артеріального тиску.
- button1: Кнопка для додавання спостереження.
- label1: Мітка для відображення тексту "Пацієнт".
- label2: Мітка для відображення тексту "Дата та час".
- label3: Мітка з текстом "Цукор".
- label4: Мітка для відображення тексту "Верхній тиск".
- label5: Мітка для відображення тексту "Нижній тиск".
- label6: Помітка з текстом "Дієта".
- label7: Мітка для відображення тексту "Вправа".

Ці компоненти впорядковано за допомогою властивостей Location, Size та TabIndex, а властивість AutoSize для міток встановлено у значення true. Події TextChanged і Click використовуються для обробки вводу користувача та його дій на інтерфейсі.

6. Designer6.cs

В даному коді є три текстові поля, які дозволяють користувачеві вводити дані: textBox1 для адреси електронної пошти лікаря, textBox2 для адреси електронної пошти пацієнта і textBox4 для вмісту електронного листа. Текстове

поле для вмісту листа є багаторядковим, що означає, що користувач може ввести кілька рядків тексту.

Існує *textBox3*, який відображає текст за замовчуванням "Тема" (що означає "Тема" українською мовою), вказуючи на те, що це текстове поле призначене для теми електронного листа. Однак це текстове поле не дозволяє користувачеві вводити дані безпосередньо.

Мітки *label1 і label2* надають користувачеві інструкції щодо того, що вводити в текстові поля textBox1 і textBox2 відповідно. Кнопка *button1* є елементом керування, який запускає подію, що надсилає електронного листа лікарю. Після того, як користувач введе необхідну інформацію в текстові поля, він може натиснути цю кнопку, щоб надіслати листа.

Форма має шість текстових полів для введення інформації: "textBox1" для електронної пошти одержувача, "textBox2" для електронної пошти відправника, "textBox3" для імені відправника, "textBox4" для теми листа і "textBox5" для пароля до поштової скриньки відправника. У текстовому полі пароля відображаються символи "*", щоб приховати справжній пароль від сторонніх очей.

На формі також є кнопка з текстом "Надіслати листа" (Send Email), яка підключена до обробника події, що виконується, коли користувач натискає на кнопку. Обробник події натискання кнопки не визначено у файлі Designer6.part2.cs, а визначено у відповідному cs файлі для цієї форми, який, називається Form6.cs.

Отже, інтерфейс програми складається з декількох форм, які надають користувачеві різні функціональні можливості. Структура кожної форми створюється за допомогою Windows Forms Designer у Visual Studio, який дозволяє розробнику перетягувати елементи керування на форму та налаштовувати їх властивості. Кожна форма складається з рядка заголовка з назвою форми та трьох кнопок: згорнути, розгорнути/відновити та закрити. Під рядком заголовка знаходиться область вмісту, яка містить різні елементи керування, такі як написи, текстові поля, кнопки, перемикачі та прапорці.

Елементи управління розташовані таким чином, щоб зробити форму легкою у використанні та розумінні. Форми пов'язані одна з одною за допомогою різних подій, таких як натискання кнопок, завантаження форми та її закриття. Коли користувач взаємодіє з елементом управління на одній формі, додаток реагує на це виконанням методу або оновленням стану змінної, що може призвести до змін в інших формах або даних.

Загалом, інтерфейс програми розроблений таким чином, щоб бути зручним та інтуїтивно зрозумілим, з чіткими інструкціями та зворотним зв'язком, щоб допомогти користувачеві на кожному кроці процесу. Форми організовані логічно, відповідні функції згруповані разом, а елементи керування позначені та розташовані таким чином, щоб було легко зрозуміти їх призначення та використання.

3.3 Опис процесу впровадження та здійснення експериментальної експлуатації інформаційної системи

Процес впровадження та пілотування програмної системи для літніх людей з діабетом 2 типу можна розділити на кілька етапів:

- Збір вимог: Цей етап передбачає розуміння потреб та вимог цільової аудиторії. У цьому випадку цільовою аудиторією є люди похилого віку з діабетом 2 типу. Важливо зібрати інформацію про їхні потреби та вподобання, щоб створити програмний додаток, пристосований до їхніх потреб.
- Розробка дизайну: На основі зібраних вимог розробляється дизайн програмної системи. Дизайн повинен враховувати користувацький інтерфейс, простоту використання та систему підтримки прийняття рішень. Дизайн повинен бути простим, інтуїтивно зрозумілим і зручним у використанні для цільової аудиторії.
- Розробка: Програмна система надання рекомендацій розробляється з використанням мов програмування та фреймворків, визначених на етапі проектування. Розробка повинна відповідати найкращим практикам кодування,

включаючи коментування, тестування та контроль версій.

- Тестування: Програма система тестується на зручність використання, функціональність та надійність. Важливо протестувати програмний додаток на різних пристроях, щоб забезпечити сумісність.
- Пілотування: Програмна система пілотується з невеликою групою користувачів для перевірки його ефективності та збору відгуків. Пілотна група повинна бути репрезентативною для цільової аудиторії, а пілотування повинно проводитися протягом певного періоду часу, щоб зібрати дані про ефективність системи підтримки прийняття рішень.
- Оцінка: Дані пілотного проекту аналізуються, а програмний додаток оцінюється на предмет ефективності та зручності використання. На основі зворотного зв'язку та оцінки програмний додаток модифікується для підвищення його ефективності та зручності використання.
- Впровадження: Програмна система надання рекомендацій впроваджується і стає доступним для цільової аудиторії. Важливо забезпечити навчання та підтримку користувачів, щоб вони могли ефективно використовувати програмний додаток.
- Обслуговування: Програмна система регулярно підтримується, щоб гарантувати, що він залишається ефективним та актуальним. Обслуговування може включати виправлення помилок, оновлення системи підтримки прийняття рішень або додавання нових функцій.

Як результат, процес впровадження та пілотування програмної системи для літніх людей з діабетом 2 типу передбачає розуміння потреб цільової аудиторії, проектування та розробку програмної системи, тестування та пілотування програмної системи, а також оцінку та підтримку програмної системи. Важливо залучати цільову аудиторію до етапів проектування та тестування, щоб забезпечити ефективність та зручність використання програмної системи [10, 14].

3.4 Тестування програмної системи надання рекомендацій на працездатність

Програма, що рекомендує дієти та фізичні вправи, згідно з вимогами та цілями дослідження, повинна мати зручний і інтуїтивно зрозумілий інтерфейс для користувачів.

При старті програми відбувається автоматичне підключення до бази даних. Після чого можливо відразу з нею працювати

Рисунок 3.1 — Вхід в систему від імені головного лікаря (адміністратора)

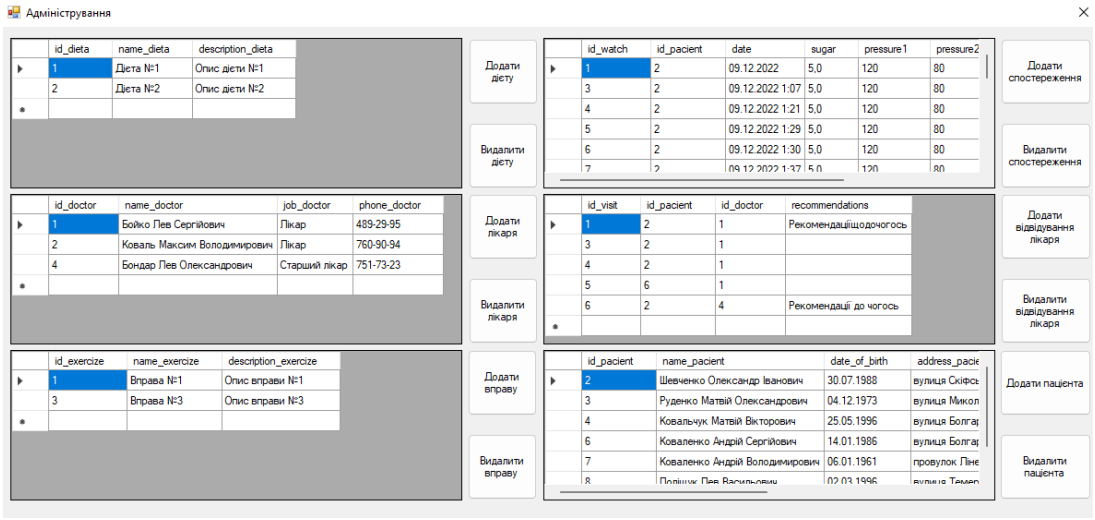


Рисунок 3.2 — Головне вікно програми від імені адміністратора

На головному вікні розташована вся доступна інформація про дієти,

фізичні вправи та їх кількість, повна інформація про пацієнтів та лікарів. Також можемо переглянути інформацію про прийом до лікаря, та найбільш актуальні спостереження пацієнтів. Адміністратор має можливість додавати та видаляти абсолютно всю інформацію, описану вище.

Спробуємо додати нового пацієнта в систему

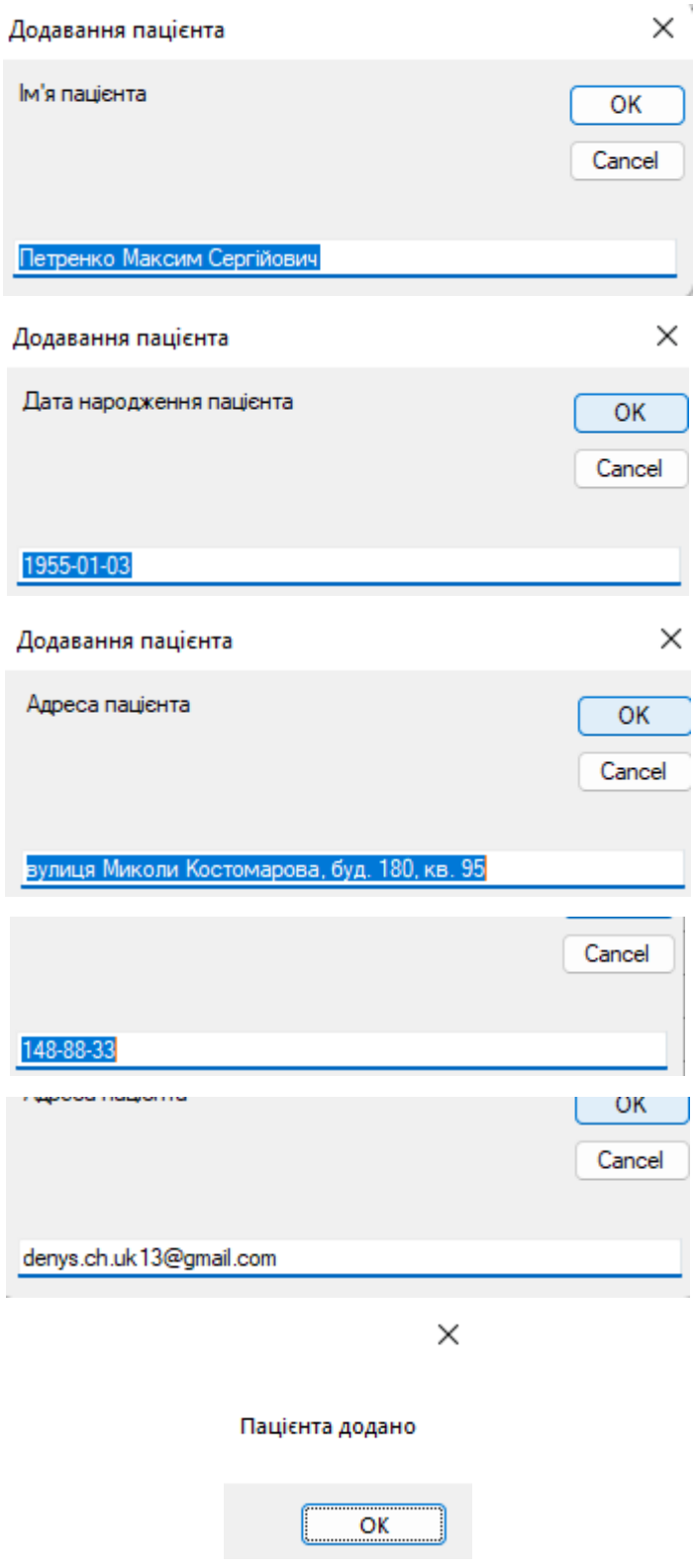


Рисунок 3.3-3.8 — Процес створення нового пацієнта

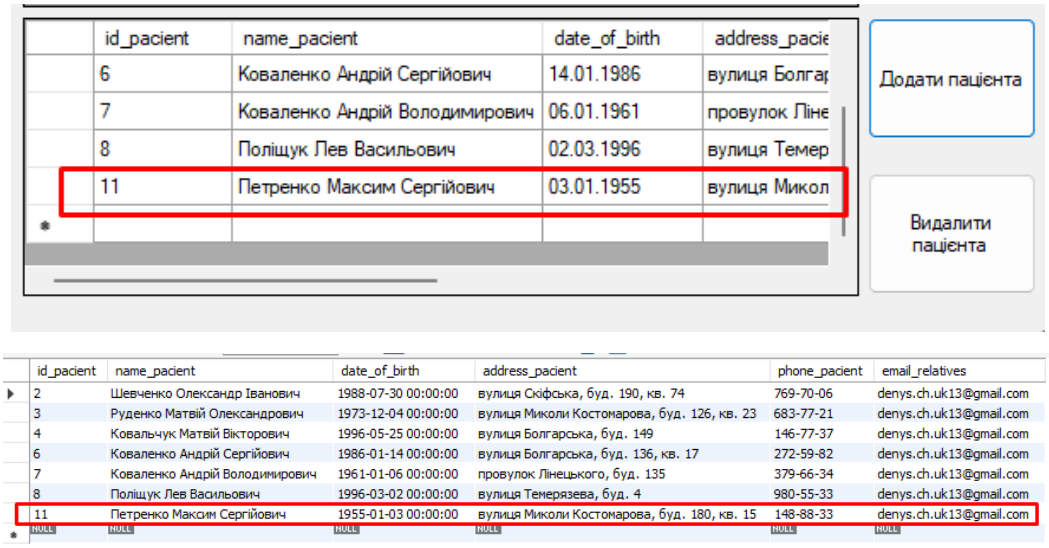


Рисунок 3.9-3.10 — Результат створення пацієнта та оновлення БД

Тепер спробуємо знайти від імені щойно створеного пацієнта та оновити дані в спостереження.

Рисунок 3.11 — Вхід від імені пацієнта

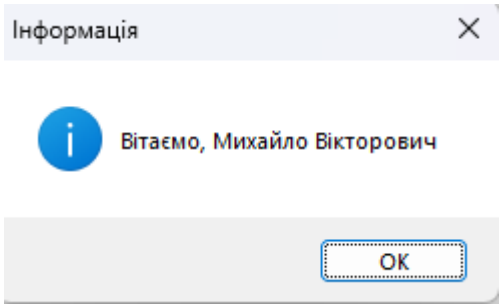


Рисунок 3.12 — Успішна авторизація пацієнта

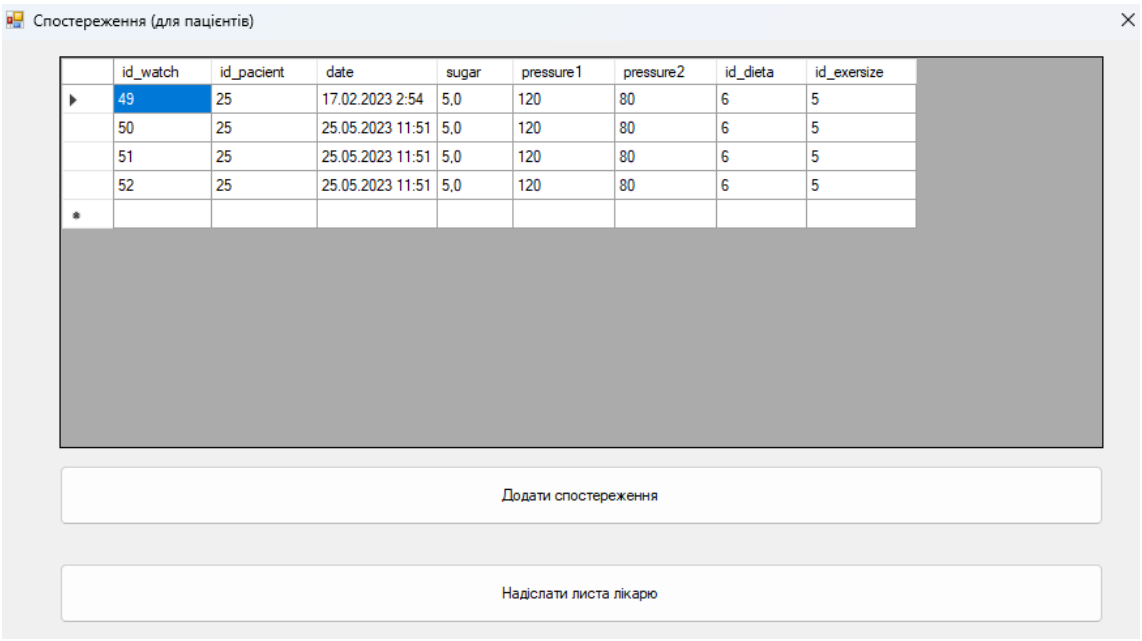


Рисунок 3.13 — Головне вікно програми від імені пацієнта

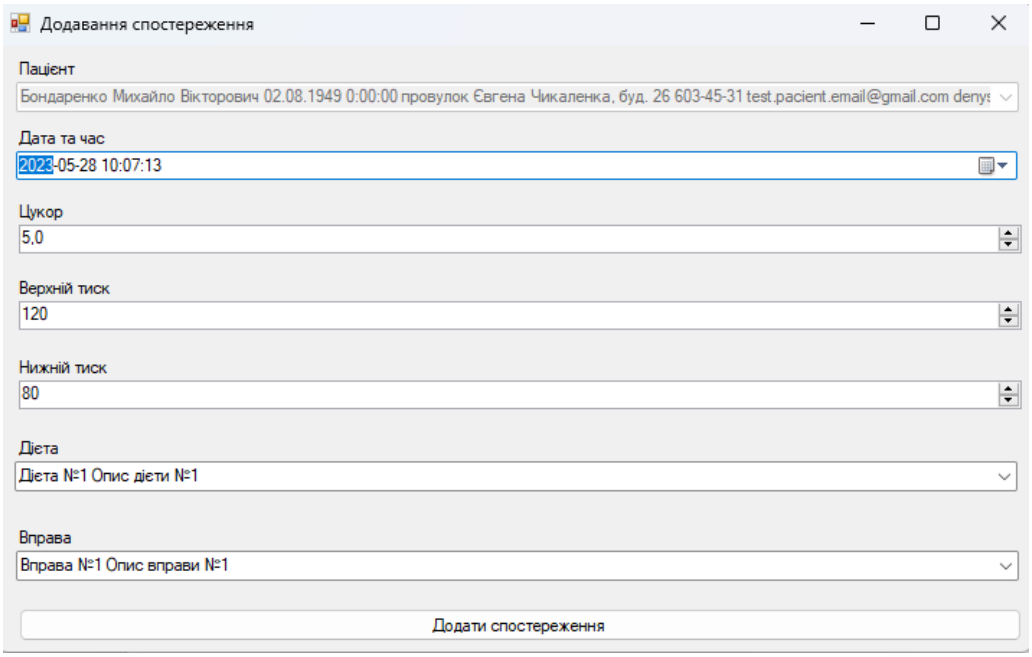


Рисунок 3.14 — Вказує необхідні дані про вміст цукру в крові, тиск

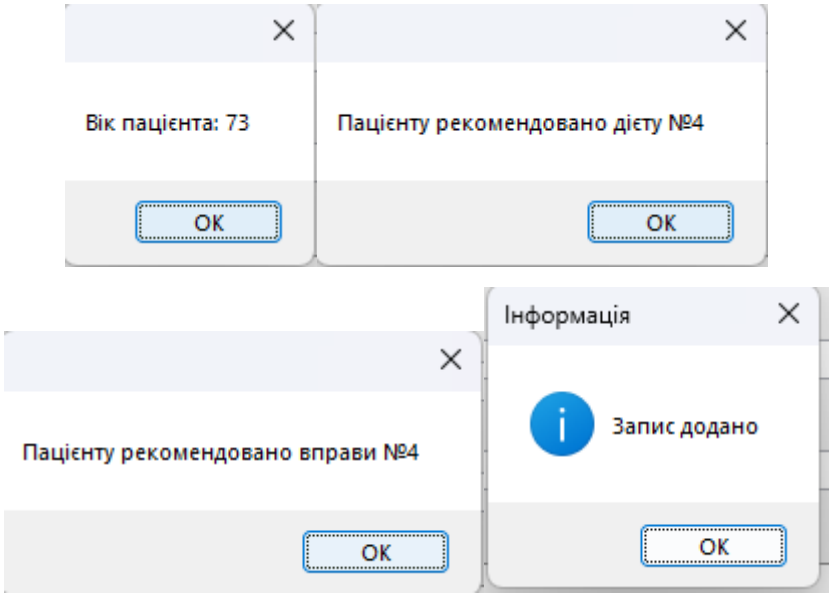


Рисунок 3.15-3.18 — Процес надання рекомендації після введення необхідних показників та в залежності від віку та індивідуальних значень

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id_watch	id_pacient	date	sugar	pressure1	pressure2	id_dieta	id_exersize
1	2	2022-12-09 00:00:00	5.0	120	80	1	1
3	2	2022-12-09 01:07:57	5.0	120	80	1	1
4	2	2022-12-09 01:21:36	5.0	120	80	1	1
5	2	2022-12-09 01:29:52	5.0	120	80	1	1
6	2	2022-12-09 01:30:23	5.0	120	80	1	1
7	2	2022-12-09 01:37:55	5.0	120	80	1	1
8	2	2022-12-09 01:43:04	5.5	120	80	1	1
9	2	2022-12-09 01:43:57	4.4	120	80	1	1
10	4	2022-12-09 01:55:36	5.0	120	80	1	1
11	2	2022-12-09 01:59:48	5.0	120	80	1	1
12	6	2022-12-09 02:07:11	5.0	120	80	1	1
13	2	2022-12-09 02:20:57	0.6	120	80	1	1
14	2	2022-12-09 02:22:32	1.0	120	80	1	1
15	2	2022-12-09 02:22:56	1.0	120	80	1	1
16	2	2022-12-09 02:24:49	5.0	120	80	1	1
17	2	2022-12-09 02:24:57	20.0	120	80	1	1
18	2	2022-12-09 02:26:19	20.0	120	80	1	1
19	2	2022-12-09 02:27:36	5.0	90	80	1	1
20	2	2022-12-09 02:27:47	2.0	80	80	1	1
21	2	2022-12-09 02:31:31	5.0	80	80	1	1
23	2	2022-12-10 02:43:05	5.0	120	80	1	1
25	6	2022-12-10 01:56:58	5.0	120	80	1	1
27	2	2022-12-11 02:41:36	13.0	120	80	1	1
28	11	2022-12-12 01:09:38	6.0	110	75	1	1

Рисунок 3.19 — Результат створення запиту та оновлення БД

Також пацієнт завжди має змогу звязатись зі своїм лікарем за допомогою вбудованої функції листування. Для цього потрібно лише знати електронну адресу лікаря. Після цього ввести свою пошту та пароль від неї, а також написати зміст листа та його опис.

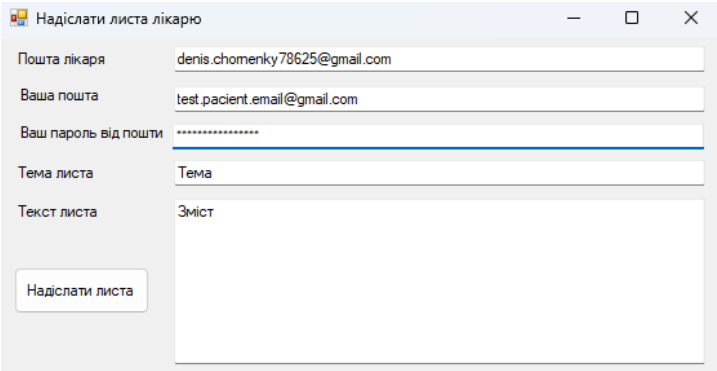


Рисунок 3.20 — Процес створення електронного листа лікарю

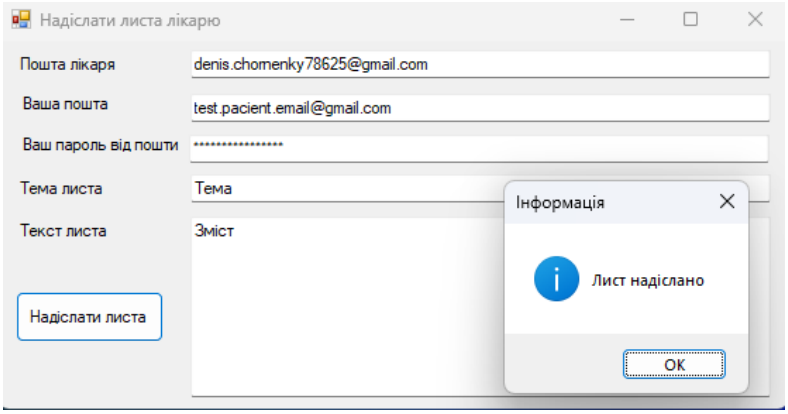
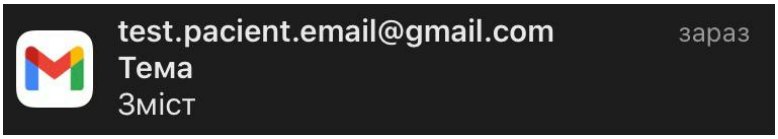


Рисунок 3.21 — Результат успішного надсилання листа



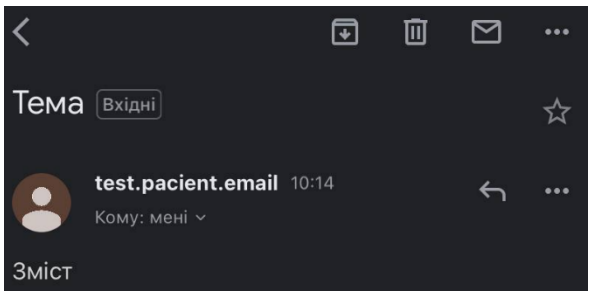


Рисунок 3.22-3.23 — Результат успішного надходження листа пацієнта на пошту лікаря

Коли пацієнт оновлює свої дані, щоб отримати вказівки щодо здоров’я, система перевіряє стан пацієнтів. Якщо стан в межах не нормального, автоматичне повідомлення надійдуть до постачальників послуг (тобто родича), щоб повідомити про стан здоров’я пацієнта. Протестуємо це.

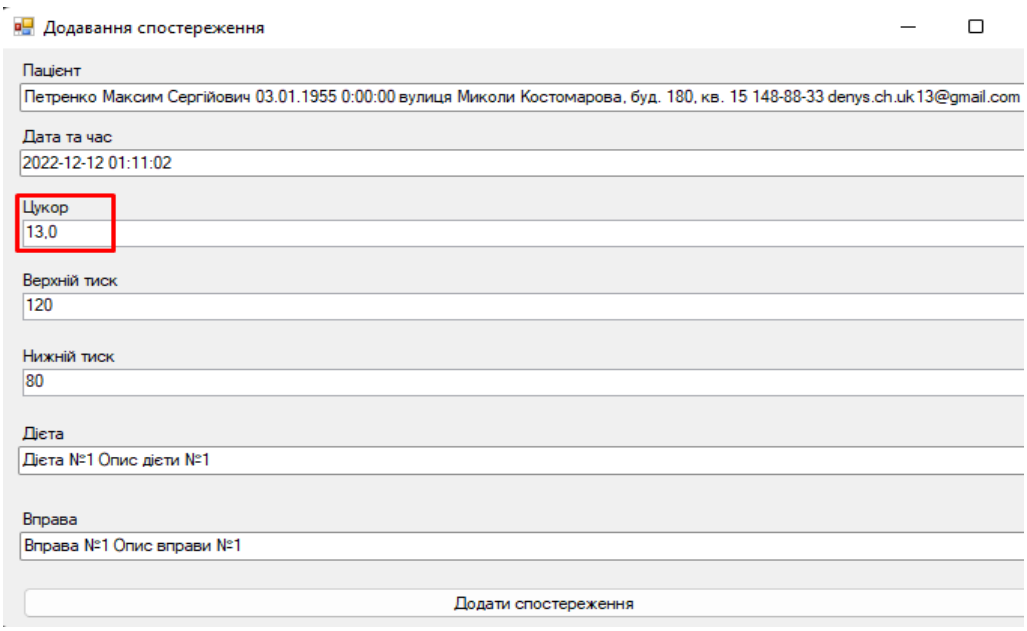


Рисунок 3.24 — Введення потенційно небезпечних даних про глюкозу

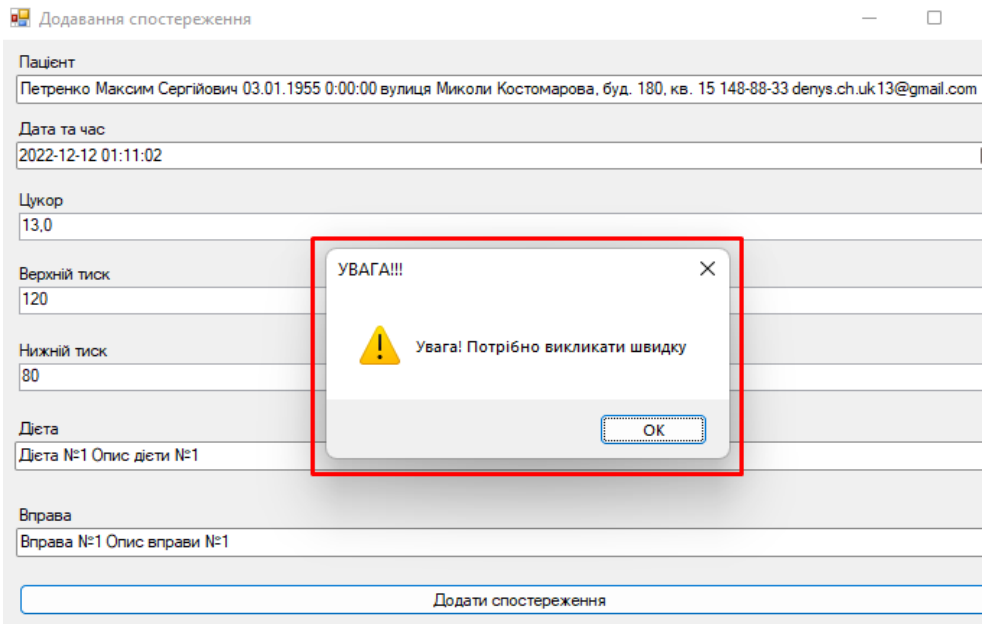


Рисунок 3.25 — Результат спрацювання СППР

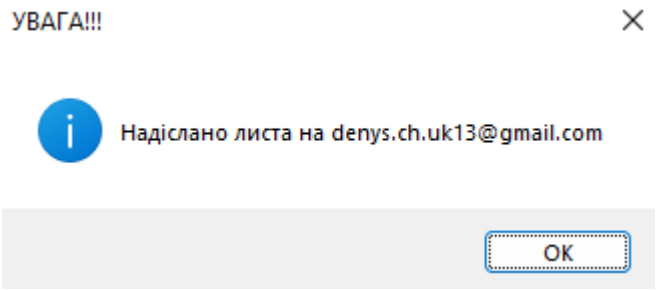


Рисунок 3.26 — Інформація про надсилання листа родичу на пошту

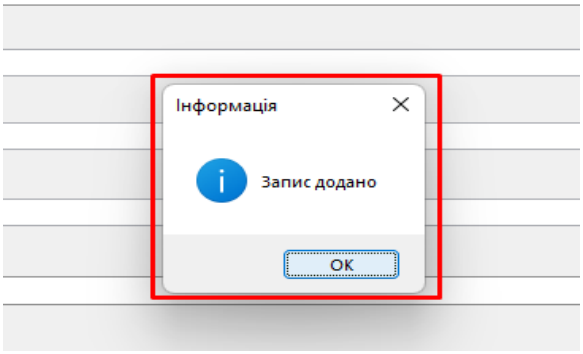


Рисунок 3.27 — Інформація про успішно доданий запит (необхідно для аналізу причину виникнення такої ситуації)

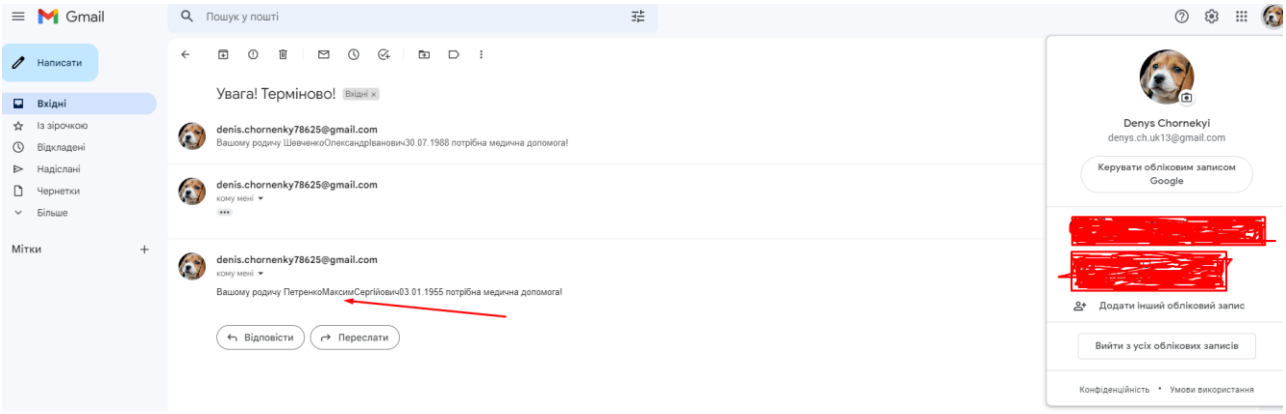
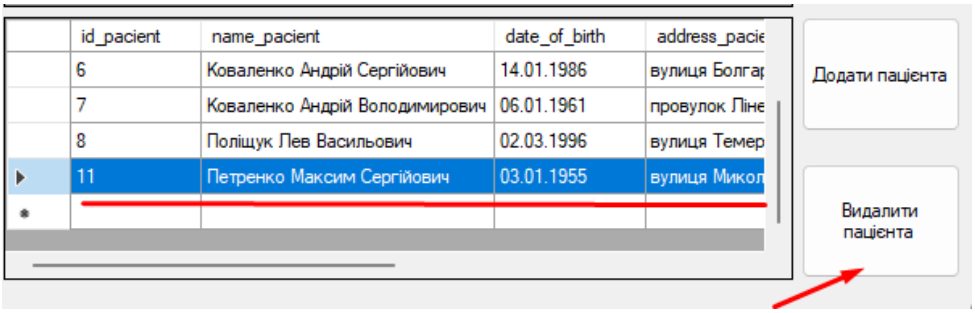


Рисунок 3.28 — Результат отримання екстреного повідомлення про загрозу створеному пацієнту

Видалимо цього пацієнта, протестувавши роботу відповідної функції (попередньо потрібно видалити про нього спостереження також за допомогою функції видалити спостереження).



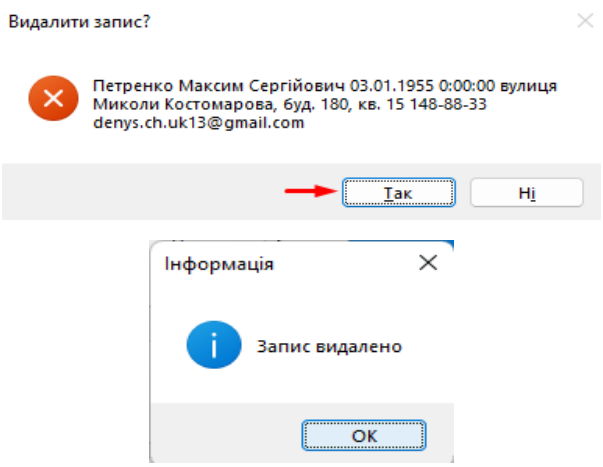


Рисунок 3.29-3.31 — Процес видалення пацієнта

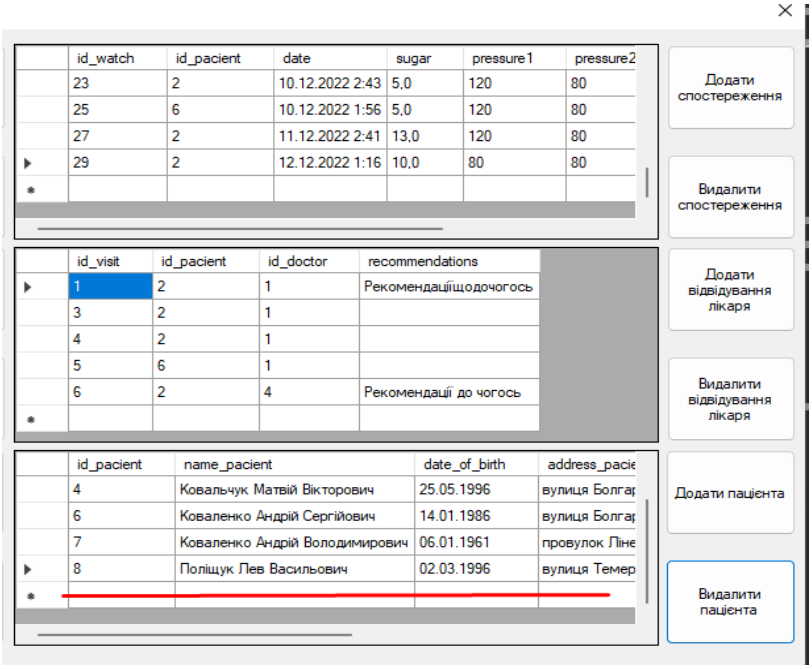


Рисунок 3.32 — Результат видалення пацієнта (залишились тільки попередньо створені)

Протестуємо як буде діяти система, коли вказана при реєстрації пошта родича хворого пацієнта не буде дійсною.

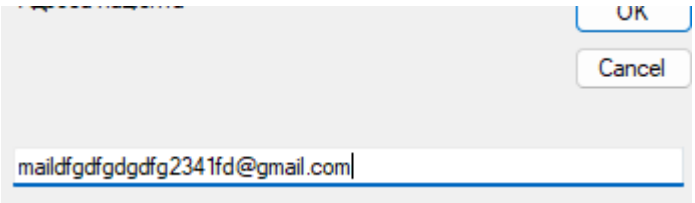


Рисунок 3.33 — Створення пацієнта з неіснуючою поштою

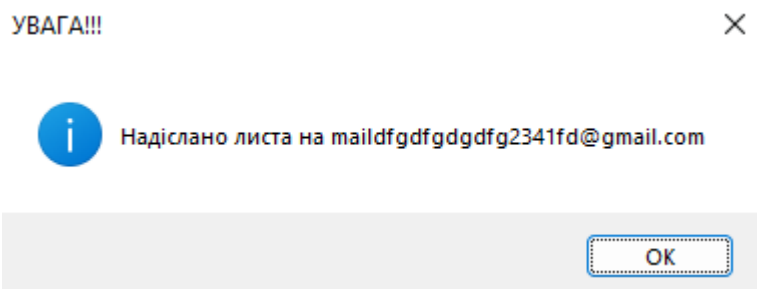


Рисунок 3.34 — Надсилання листа на неіснуючу пошту

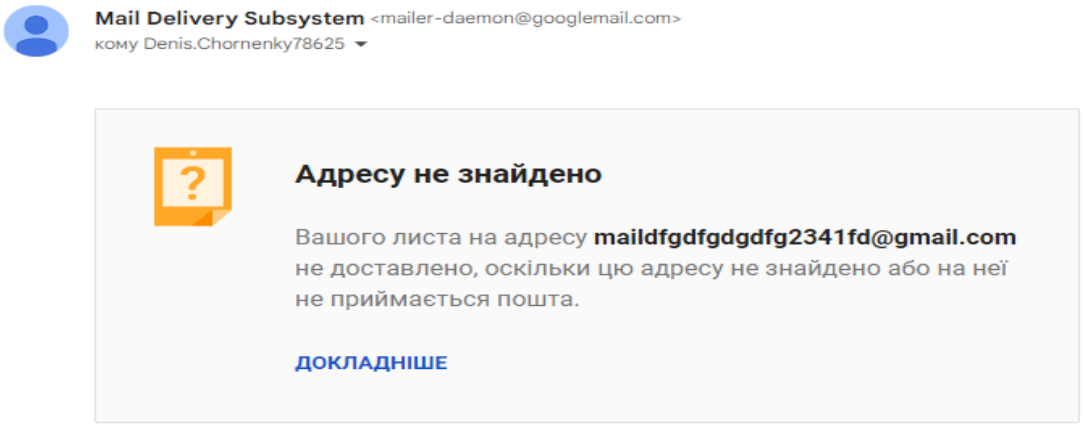


Рисунок 3.35 — Сповіщення про неможливість надіслати листа на пошту, що не існує (сповіщення прийшло на основну пошту, через яку працює система)

Отже, можна побачити що перевірка відповідності по пропонованім вимогам відповідає дійсності. Забезпечення впевненості в якості ПЗ, пошук очевидних помилок в програмному забезпеченні, які повинні бути виявлені до того, як їх виявлять користувачі програми були успішно протестовані.

3.5 Висновок до третього розділу

В розділі реалізації та тестування програмного застосунку було описано повну структуру програмної частини нашого додатка для хворих на діабет 2-го типу. Описано призначення, методи реалізації алгоритмів, систем прийняття рішень та функціоналу інтерфейсу даного застосунку. Після чого, проведено

тестування всього функціоналу задля виявлення помилок та підтвердження повної працездатності додатка перед тим як запускати продукт на ринок. Як результат, перевірка відповідності по пропонованім вимогам відповідає дійсності, а інтерфейс як і було зазначено – інтуїтивно зрозумілий та простий.

ВИСНОВКИ

Цукровий діабет є одним із хронічних захворювань, і хворі на нього потребують особливого догляду. Було проведено детальний огляд літератури на цукровий діабет, щоб дізнатися, як полегшити життя хворих на цукровий діабет і які проблеми вони мають, як можна покращити якість їхнього життя. Розробили систему підтримки прийняття рішень щодо діабету, ціллю такої системи стали особливо старші покоління. Існує ряд аспектів, про які варто звернути увагу, як-от розуміння хвороби, готовність пацієнтів виліковуватися та дотримуватися певних критеріїв, доступність постачальників послуг, доступність інформації як для лікарів, так і для пацієнтів, корисне представлення історії даних пацієнта та користувача дизайн інтерфейсу. Було створено система, яка може забезпечувати пацієнтам і лікарям умови, щоб вони могли ефективно та результативно досягати спільної цілі. Пацієнти можуть отримати базові інструкції щодо свого здоров’я вдома. Це може покращити якість їхнього життя. Доступність інформації та якісне представлення даних можуть допомогти лікарям прийняти оптимальне рішення для пацієнта та самого лікаря. Адже медики виступають проти непотрібних візитів пацієнтів до лікарні. За допомогою даної програми пацієнти можуть отримати основні інструкції вдома, їм не потрібно знову і знову відвідувати центри надання медичної допомоги, щоб отримати інструкції щодо підтримання здоров’я. До того ж вони можуть отримати доступ з власного ПК, залишаючись вдома. Це сприяє щоденному покращенню якості життя пацієнта. Пацієнти оновлюють інформацію про свій стан здоров'я, і вона автоматично зберігається в базі даних. Історія хвороби пацієнта допомагає лікарям приймати рішення щодо подальшого лікування. За результатами опитування та пропозицій пацієнтів, можна зробити висновок, що програмна система надання рекомендацій хворим на цукровий діабет 2-го типу може полегшити їм життя.

Поточна запропонована версія програми є лише першими кроками в даному дослідженні, з базовими функціями та максимально простим інтерфейсом. Проте, можна з упевненістю сказати, що реалізовано СППР з

програмним модулем формування рекомендацій, яка, залежно від даних пацієнта про його стан, надає рекомендації про план дієти та фізичних вправ. А також, в разі виходу індивідуальних показників від норми, - миттєво повідомить про екстрену ситуацію родича хворого, не забувши при цьому оновити поточні дані в загальний журнал спостережень задля проведення аналізу та покращення дій персоналу та самої програми в майбутньому.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. С. Кемпбелл // «Лікування діабету 2 типу у літніх пацієнтів», Журнал аптечної практики, – 2000. С. 260-277.
2. Нойміллер / П.С. Одегард / Дж.Р. Уайт / С.М. Сеттер / Р.К. Кемпбелл // «Погляд у майбутнє: фокус на інгібіторах DPP-4 для лікування діабету 2 типу та нових методах лікування», The Diabetes Educator – 2008. С 182-199.
3. M.J. Franz «Докази є: втручання у спосіб життя можуть запобігти діабету», American Journal of Lifestyle Medicine, – 2007. С. 111-120.
4. Mark J. Price // C# 8.0 and .NET Core 3.0 – Modern Cross-Platform Development: Build applications with C#, .NET Core, Entity Framework Core, ASP.NET Core. – 2021.
5. Karri Andersen // «I Have Diabetes: A Children's Book About Juvenile Diabetes», Little life lesson books, – 2012.

Електронні ресурси

6. Цукровий діабет: що потрібно знати для профілактики цієї хвороби. [Електронний ресурс] – Режим доступу до ресурсу: https://apteka-ds.com.ua/blog-item/tsukrovyi-diabet-shcho-potribno-znaty-dlia-profilaktyky-tsiiei-khvoroby
7. Exercise Tips for Type 2 Diabetes. [Електронний ресурс] – Режим доступу до ресурсу: https://www.webmd.com/diabetes/exercise-guidelines
8. ЛІКУВАННЯ ТА ДІАГНОСТИКА ЦУКРОВОГО ДІАБЕТУ 1-2 ТИПУ. [Електронний ресурс] – Режим доступу до ресурсу: https://viva.clinic/ua/endokrinologiya/lechenie-i-diagnostika-saharnogo-diabeta-1-2-stepeni/
9. ABOUT DIABETES DAY. [Електронний ресурс] – Режим доступу до ресурсу: https://worlddiabetesday.org/about/understandingdiabetes/?gclid=Cj0KCQiAnNacBhDvARIsABnDa69vst2P59R95Qy63XxQ4V14E2ybmY3N4Zz31YesQfjotqByEGrixfsaAq_VEALw_wcB

10. Type 2 Diabetes-Friendly Recipes and Tips. [Електронний ресурс] – Режим доступу до ресурсу: https://info.diatribe.org/brightspotsandlandmines-type-2-diabetes-food/?gclid=Cj0KCQiAnNacBhDvARIsABnDa68u4lsM61PKNqwD2gt1HW9ZoO-W-QTSOuUv2pgatZt6Fo7gFIUq440aAuB5EALw_wcB
11. The Best Diabetes Apps of 2022. [Електронний ресурс] – Режим доступу до ресурсу: https://www.healthline.com/health/diabetes/top-iphone-android-apps
12. Windows Forms app with C#. [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/visualstudio/ide/create-csharp-winform-visual-studio?view=vs-2022
13. C# Windows Forms Application Tutorial with Example. [Електронний ресурс] – Режим доступу до ресурсу: https://www.guru99.com/c-sharp-windows-forms-application.html
14. Prevention of Type 2 Diabetes. [Електронний ресурс] – Режим доступу до ресурсу:https://www.idfdiabeteschool.org/free-courses/prevention-t2d?utm_source=Adwords&utm_medium=Prevention&utm_campaign=English&gclid=Cj0KCQiAnNacBhDvARIsABnDa68jWHSP5RwqzWKb43OzH9J65Tj08STOSTfBRK_Dn5kfCy8W-lz4C3saAp65EALw_wcB

ДОДАТКИ

Form1.cs:

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Globalization;
using System.Windows.Forms;
using Microsoft.VisualBasic;
using MySql.Data.MySqlClient;

namespace ISPPR
{
   public partial class Form1 : Form
   {
      public static MySqlConnection mysql_connection;
      public static MySqlCommand mysql_query;
      public static MySqlDataReader mysql_result;
      public static MySqlDataAdapter MyDA;
      public static DataTable table;
      public static BindingSource bSource;
      public static Random rnd;
      public bool admin;
      public static int userID;

      private Form2 form2;
      private Form3 form3;

      public static Dictionary <int, string> pacients,doctors,exercizes,dietas;

      public static void selectAll(string tableDB, DataGridView dgv)
      {
         Form1.mysql_connection = DBconnect.connect();

         Form1.mysql_query = Form1.mysql_connection.CreateCommand();
         Form1.mysql_query.CommandText = "SELECT * FROM " + "user";
         Form1.mysql_connection.Open();
         Form1.mysql_result = Form1.mysql_query.ExecuteReader();
         Form1.mysql_connection.Close();

         Form1.MyDA = new MySqlDataAdapter();
         Form1.MyDA.SelectCommand = new MySqlCommand(Form1.mysql_query.CommandText,
Form1.mysql_connection);

         Form1.table = new DataTable();
         Form1.MyDA.Fill(Form1.table);

         Form1.bSource = new BindingSource();
         Form1.bSource.DataSource = Form1.table;

         dgv.DataSource = Form1.bSource;
      }


      private void Form1_VisibleChanged(object sender, EventArgs e)
      {
         selectAll("", dataGridView1);
      }

      private void button2_Click(object sender, EventArgs e)
      {
```

```
            string admPass = "";
            for (int i = 0; i < dataGridView1.RowCount - 1; i++)
            {
                int compareResult = String.Compare(dataGridView1.Rows[i].Cells[1].Value.ToString(), "Admin",
                                    StringComparison.OrdinalIgnoreCase);
                if (compareResult == 0)
                {
                    admPass = dataGridView1.Rows[i].Cells[2].Value.ToString();
                    break;
                }
            }
            //MessageBox.Show(admPass,"Пароль адміна");
            userID = 0;
            if ((textBox1.Text == "Admin")&&(textBox2.Text == admPass))
            {
                MessageBox.Show("Вхід під адміном", "Увага", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                this.Hide();
                form2 = new Form2(this,true,0);
                form2.Show();
            }
            else
            {
                for (int i = 0; i < dataGridView1.RowCount - 1; i++)
                {
                    int compareResultLogin = String.Compare(dataGridView1.Rows[i].Cells[1].Value.ToString(),
textBox1.Text,
                                        StringComparison.OrdinalIgnoreCase);
                    int compareResultPassword = String.Compare(dataGridView1.Rows[i].Cells[2].Value.ToString(),
textBox2.Text,
                                        StringComparison.Ordinal);
                    if ((compareResultLogin == 0)&& (compareResultPassword == 0))
                    {
                        userID = Int32.Parse(dataGridView1.Rows[i].Cells[3].Value.ToString());
                        break;
                    }
                }
                if (userID > 0)
                {
                                this.Hide();
                    form2 = new Form2(this, false, userID);
                    form2.Show();
                }
                else
                    MessageBox.Show("Такого користувача не існує", "Увага", MessageBoxButtons.OK,
MessageBoxIcon.Error);


            }
        }

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {

            selectAll("", dataGridView1);


        }
    }
}
```

Form2.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using MySql.Data.MySqlClient;
using Microsoft.VisualBasic;
using System.Globalization;
using System.Net.Mail;
using System.Net;

namespace ISPPR
{
   public partial class Form2 : Form
   {
      public bool admin;
      public int userID;
      Form form1;
      private Form4 form4;
      private Form5 form5;
      private Form6 form6;

      string email_pacient;

      public Form2(Form form1,bool admin,int userID)
      {
         InitializeComponent();
         this.form1 = form1;
         this.admin = admin;
         this.userID = userID;
      }

      private void updateDieta(int id_pacient,int id_dieta)
      {
         Form1.mysql_query.CommandText = "update ISPPR.watch " +
            "set id_dieta = " + id_dieta.ToString()+
            " where id_pacient = " + id_pacient.ToString() +
            ";";
```

```
    //MessageBox.Show(Form1.mysql_query.CommandText);
    Form1.mysql_query = new MySqlCommand(Form1.mysql_query.CommandText, Form1.mysql_connection);
    Form1.mysql_connection.Open();
    Form1.mysql_result = Form1.mysql_query.ExecuteReader();
    Form1.mysql_connection.Close();

    //MessageBox.Show("Дієту змінено", "Інформація", MessageBoxButtons.OK, MessageBoxIcon.Information);
}

private void updateExersize(int id_pacient, int id_exersize)
{
    Form1.mysql_query.CommandText = "update ISPPR.watch " +
      "set id_exersize = " + id_exersize.ToString() +
      " where id_pacient = " + id_pacient.ToString() +
      ";";
    //MessageBox.Show(Form1.mysql_query.CommandText);
    Form1.mysql_query = new MySqlCommand(Form1.mysql_query.CommandText, Form1.mysql_connection);
    Form1.mysql_connection.Open();
    Form1.mysql_result = Form1.mysql_query.ExecuteReader();
    Form1.mysql_connection.Close();

    //MessageBox.Show("Дієту змінено", "Інформація", MessageBoxButtons.OK, MessageBoxIcon.Information);
}

public void updating()
{
    //id диет
    int[] id_dieta = new int[dataGridView1.RowCount - 1];
    for (int i=0;i< id_dieta.Length;i++)
    {
      if(dataGridView1.Rows[i].Cells[1].Value.ToString()=="Дієта №" +(i+1).ToString())
      {
        id_dieta[i] = Int32.Parse(dataGridView1.Rows[i].Cells[0].Value.ToString());
      }
    }

    //id упражнений
    int[] id_vprava = new int[dataGridView3.RowCount - 1];
    for (int i = 0; i < id_vprava.Length; i++)
    {
      if (dataGridView3.Rows[i].Cells[1].Value.ToString() == "Вправа №" + (i + 1).ToString())
      {
        id_vprava[i] = Int32.Parse(dataGridView3.Rows[i].Cells[0].Value.ToString());
```

```
    }
}
//проверка id диет
/*string tmp = "";
for (int i = 0; i < id_dieta.Length; i++)
    tmp += id_dieta[i].ToString();
MessageBox.Show(tmp);*/

//проверка id упражнений
/*string tmp = "";
for (int i = 0; i < id_vprava.Length; i++)
    tmp += id_vprava[i].ToString();
MessageBox.Show(tmp);*/

for (int i = 0; i < dataGridView4.RowCount - 1; i++)
{
    string ds = dataGridView4.Rows[i].Cells[2].Value.ToString();
    DateTime dt = DateTime.Parse(ds);
    TimeSpan span = DateTime.Now - DateTime.Parse(ds);
    DateTime zeroTime = new DateTime(1, 1, 1);
    int years = (zeroTime + span).Year - 1;
    //диета №1
    if(years>=30 && years<=45)
    {
        updateDieta(Int32.Parse(dataGridView4.Rows[i].Cells[0].Value.ToString()), id_dieta[0]);
        updateExersize(Int32.Parse(dataGridView4.Rows[i].Cells[0].Value.ToString()), id_vprava[0]);
    }
    //диета №2
    if (years >= 46 && years <= 60)
    {
        updateDieta(Int32.Parse(dataGridView4.Rows[i].Cells[0].Value.ToString()), id_dieta[1]);
        updateExersize(Int32.Parse(dataGridView4.Rows[i].Cells[0].Value.ToString()), id_vprava[1]);
    }
    //диета №3
    if (years >= 61 && years <= 70)
    {
        updateDieta(Int32.Parse(dataGridView4.Rows[i].Cells[0].Value.ToString()), id_dieta[2]);
        updateExersize(Int32.Parse(dataGridView4.Rows[i].Cells[0].Value.ToString()), id_vprava[2]);
    }
    //диета №4
    if (years >= 71 && years <= 80)
    {
        updateDieta(Int32.Parse(dataGridView4.Rows[i].Cells[0].Value.ToString()), id_dieta[3]);
```

```
            updateExersize(Int32.Parse(dataGridView4.Rows[i].Cells[0].Value.ToString()), id_vprava[3]);
        }
        //диета №5
        if (years >= 81 && years <= 100)
        {
            updateDieta(Int32.Parse(dataGridView4.Rows[i].Cells[0].Value.ToString()), id_dieta[4]);
            updateExersize(Int32.Parse(dataGridView4.Rows[i].Cells[0].Value.ToString()), id_vprava[4]);
        }

        //MessageBox.Show(DateTime.Now+" "+dataGridView4.Rows[i].Cells[2].Value.ToString()+" "+
years.ToString());

        /*for (int j = 1; j < dataGridView4.ColumnCount; j++)
            s += dataGridView4.Rows[i].Cells[j].Value.ToString() + " ";
        Form1.pacients.Add(Int32.Parse(dataGridView4.Rows[i].Cells[0].Value.ToString()), s);*/
    }
    selectAll("watch", dataGridView6);
    if (admin)
        MessageBox.Show("Дієти та вправи оновлено з урахуванням віку");
}

private void pacientsToDictionary()
{
    Form1.pacients = new Dictionary<int, string>();
    for (int i = 0; i < dataGridView4.RowCount - 1; i++)
    {
        string s = "";
        for (int j = 1; j < dataGridView4.ColumnCount; j++)
            s += dataGridView4.Rows[i].Cells[j].Value.ToString() + " ";
        Form1.pacients.Add(Int32.Parse(dataGridView4.Rows[i].Cells[0].Value.ToString()), s);
    }
}

private void doctorsToDictionary()
{
    Form1.doctors = new Dictionary<int, string>();
    for (int i = 0; i < dataGridView2.RowCount - 1; i++)
    {
        string s = "";
        for (int j = 1; j < dataGridView2.ColumnCount; j++)
            s += dataGridView2.Rows[i].Cells[j].Value.ToString() + " ";
        Form1.doctors.Add(Int32.Parse(dataGridView2.Rows[i].Cells[0].Value.ToString()), s);
    }
```

```
    }

    private void exercizesToDictionary()
    {
        Form1.exercizes = new Dictionary<int, string>();
        for (int i = 0; i < dataGridView3.RowCount - 1; i++)
        {
            string s = "";
            for (int j = 1; j < dataGridView3.ColumnCount; j++)
                s += dataGridView3.Rows[i].Cells[j].Value.ToString() + " ";
            Form1.exercizes.Add(Int32.Parse(dataGridView3.Rows[i].Cells[0].Value.ToString()), s);
        }
    }

    private void dietasToDictionary()
    {
        Form1.dietas = new Dictionary<int, string>();
        for (int i = 0; i < dataGridView1.RowCount - 1; i++)
        {
            string s = "";
            for (int j = 1; j < dataGridView1.ColumnCount; j++)
                s += dataGridView1.Rows[i].Cells[j].Value.ToString() + " ";
            Form1.dietas.Add(Int32.Parse(dataGridView1.Rows[i].Cells[0].Value.ToString()), s);
        }
    }

    public static void selectAll(string tableDB,DataGridView dgv)
    {
        Form1.mysql_connection = DBconnect.connect();

        Form1.mysql_query = Form1.mysql_connection.CreateCommand();
        Form1.mysql_query.CommandText = "SELECT * FROM "+ tableDB;
        Form1.mysql_connection.Open();
        Form1.mysql_result = Form1.mysql_query.ExecuteReader();
        Form1.mysql_connection.Close();

        Form1.MyDA = new MySqlDataAdapter();
        Form1.MyDA.SelectCommand = new MySqlCommand(Form1.mysql_query.CommandText,
Form1.mysql_connection);

        Form1.table = new DataTable();
        Form1.MyDA.Fill(Form1.table);
```

```
            Form1.bSource = new BindingSource();
            Form1.bSource.DataSource = Form1.table;

            dgv.DataSource = Form1.bSource;
        }

        private void delete(string tableDB, DataGridView dgv)
        {
            if (dgv.SelectedRows.Count == 0)
                MessageBox.Show("Оберіть рядок в таблиці", "Увага", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
            else
                foreach (DataGridViewRow row in dgv.SelectedRows)
                {
                    String delete = "";
                    for (int i = 1; i < dgv.ColumnCount; i++)
                    {
                        delete += dgv.Rows[row.Index].Cells[i].Value.ToString() + " ";
                    }
                    DialogResult dialogResult = MessageBox.Show(delete, "Видалити запис?", MessageBoxButtons.YesNo,
MessageBoxIcon.Stop);
                    if (dialogResult == DialogResult.Yes)
                    {
                        //string query = "delete from dieta WHERE id_dieta=" +
dgv.Rows[row.Index].Cells[0].Value.ToString();
                        string query = "delete from "+ tableDB + " WHERE id_"+ tableDB + "=" +
dgv.Rows[row.Index].Cells[0].Value.ToString();
                        Form1.mysql_query = new MySqlCommand(query, Form1.mysql_connection);
                        Form1.mysql_connection.Open();
                        Form1.mysql_result = Form1.mysql_query.ExecuteReader();
                        MessageBox.Show("Запис видалено", "Інформація", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                        Form1.mysql_connection.Close();

                        //удаление записи из таблицы пользователей
                        if (tableDB == "pacient")
                        {
                            query = "delete from user"+ " WHERE idpacient = " +
dgv.Rows[row.Index].Cells[0].Value.ToString();
                            Form1.mysql_query = new MySqlCommand(query, Form1.mysql_connection);
                            Form1.mysql_connection.Open();
                            Form1.mysql_result = Form1.mysql_query.ExecuteReader();
                            //MessageBox.Show("Запис видалено", "Інформація", MessageBoxButtons.OK,
```

```
MessageBoxIcon.Information);
                    Form1.mysql_connection.Close();
                }

                dgv.Rows.RemoveAt(row.Index);

            }
        }
    }
    private void Form2_Load(object sender, EventArgs e)
    {
        selectAll("dieta", dataGridView1);
        selectAll("doctor", dataGridView2);
        selectAll("exercize", dataGridView3);
        selectAll("pacient", dataGridView4);
        selectAll("visit", dataGridView5);
        selectAll("watch", dataGridView6);

        pacientsToDictionary();
        doctorsToDictionary();
        exercizesToDictionary();
        dietasToDictionary();

        updating();
        if (admin)
        {
            button13.Visible = false;
        }
        if(!admin)
        {
            dataGridView1.Visible = false;
            dataGridView2.Visible = false;
            dataGridView3.Visible = false;
            dataGridView4.Visible = false;
            dataGridView5.Visible = false;
            button1.Visible = false;
            button2.Visible = false;
            button3.Visible = false;
            button4.Visible = false;
            button5.Visible = false;
            button6.Visible = false;
            button7.Visible = false;
            button8.Visible = false;
```

```
            button9.Visible = false;
            button10.Visible = false;
            button11.Visible = false;
            button13.Visible = true;

            this.Text = "Спостереження (для пацієнтів)";
            dataGridView6.Left = 50;
            this.Width = 900;
            this.Height = 500;
            dataGridView6.Width = this.Width-100;
            dataGridView6.Height = this.Height - 200;
            button12.Left = 50;
            button12.Top = this.Height - 175;
            button12.Width = dataGridView6.Width;
            button12.Height = button12.Height - 25;

            button13.Left = 50;
            button13.Top = this.Height - 100;
            button13.Width = dataGridView6.Width;
            button13.Height = button12.Height;

            //отображение записей только теущего пользователя
            //MessageBox.Show(userID.ToString());

            Form1.mysql_connection = DBconnect.connect();

            Form1.mysql_query = Form1.mysql_connection.CreateCommand();
            Form1.mysql_query.CommandText = "SELECT * FROM watch where id_pacient = " +userID+";";
            //Form1.mysql_query.CommandText = "SELECT * from visit;";
            //MessageBox.Show(Form1.mysql_query.CommandText.ToString());
            Form1.mysql_connection.Open();
            Form1.mysql_result = Form1.mysql_query.ExecuteReader();
            Form1.mysql_connection.Close();

            Form1.MyDA = new MySqlDataAdapter();
            Form1.MyDA.SelectCommand = new MySqlCommand(Form1.mysql_query.CommandText,
Form1.mysql_connection);

            Form1.table = new DataTable();
            Form1.MyDA.Fill(Form1.table);

            Form1.bSource = new BindingSource();
            Form1.bSource.DataSource = Form1.table;
```

```
            dataGridView6.DataSource = Form1.bSource;

            string[] subs = Form1.pacients[userID].Split(' ');
            MessageBox.Show("Вітаємо, "+ subs[1]+" "+subs[2], "Інформація", MessageBoxButtons.OK,
MessageBoxIcon.Information);

            //извлечение почты - может быть пара дополнительных пробелов в адресе, поэтому через if
            email_pacient = subs[12];
            if (email_pacient.Length == 0)
               email_pacient = subs[10];
            //MessageBox.Show(email_pacient);
         }
      }
      private void button1_Click(object sender, EventArgs e)
      {
         string dieta = Interaction.InputBox("Назва дієти", "Додавання дієти", "Дієта №");
         string description = Interaction.InputBox("Опис дієти", "Додавання дієти", "Опис дієти №");

         Form1.mysql_query.CommandText = "insert into ISPPR.dieta(name_dieta,description_dieta) values('" + dieta +
"','" + description + "');";
         Form1.mysql_query = new MySqlCommand(Form1.mysql_query.CommandText, Form1.mysql_connection);
         Form1.mysql_connection.Open();
         Form1.mysql_result = Form1.mysql_query.ExecuteReader();
         MessageBox.Show("Дієту додано");
         Form1.mysql_connection.Close();
         selectAll("dieta", dataGridView1);
         dietasToDictionary();
      }
      private void Form2_FormClosed(object sender, FormClosedEventArgs e)
      {
         form1.Show();
      }
      private void button2_Click(object sender, EventArgs e)
      {
         delete("dieta", dataGridView1);
         dietasToDictionary();
      }
      private void button4_Click(object sender, EventArgs e)
      {
         string name = Interaction.InputBox("Ім'я лікаря", "Додавання лікаря", Randomizer.randomName());
         string job = Interaction.InputBox("Ім'я лікаря", "Додавання лікаря", Randomizer.randomJob());
         string phone = Interaction.InputBox("Ім'я лікаря", "Додавання лікаря", Randomizer.randomPhone());
```

```
            Form1.mysql_query.CommandText = "insert into ISPPR.doctor(name_doctor,job_doctor,phone_doctor)
values('" + name + "','" + job + "','" + phone + "');";
            Form1.mysql_query = new MySqlCommand(Form1.mysql_query.CommandText, Form1.mysql_connection);
            Form1.mysql_connection.Open();
            Form1.mysql_result = Form1.mysql_query.ExecuteReader();
            MessageBox.Show("Лікаря додано");
            Form1.mysql_connection.Close();
            selectAll("doctor", dataGridView2);
            doctorsToDictionary();
        }
        private void button3_Click(object sender, EventArgs e)
        {
            delete("doctor", dataGridView2);
            doctorsToDictionary();
        }

        private void button6_Click(object sender, EventArgs e)
        {
            string name = Interaction.InputBox("Назва вправи", "Додавання вправи", "Вправа №");
            string description = Interaction.InputBox("Опис вправи", "Додавання вправи", "Опис вправи №");

            Form1.mysql_query.CommandText = "insert into ISPPR.exercize(name_exercize,description_exercize)
values('" + name + "','" + description + "');";
            Form1.mysql_query = new MySqlCommand(Form1.mysql_query.CommandText, Form1.mysql_connection);
            Form1.mysql_connection.Open();
            Form1.mysql_result = Form1.mysql_query.ExecuteReader();
            MessageBox.Show("Вправу додано");
            Form1.mysql_connection.Close();
            selectAll("exercize", dataGridView3);
            exercizesToDictionary();
        }
        private void button5_Click(object sender, EventArgs e)
        {
            delete("exercize", dataGridView3);
            exercizesToDictionary();
        }
        private void button8_Click(object sender, EventArgs e)
        {
            string name = Interaction.InputBox("Ім'я пацієнта", "Додавання пацієнта", Randomizer.randomName());
            string date = Interaction.InputBox("Дата народження пацієнта", "Додавання пацієнта",
Randomizer.randomDate());
```

```
        string ds = date;
        DateTime dt = DateTime.Parse(ds);
        TimeSpan span = DateTime.Now - DateTime.Parse(ds);
        DateTime zeroTime = new DateTime(1, 1, 1);
        int years = (zeroTime + span).Year - 1;

        if ((years < 30) || (years > 100))
        {
           MessageBox.Show("Унікальний випадок!");
        }
        else {

        string address = Interaction.InputBox("Адреса пацієнта", "Додавання пацієнта",
Randomizer.randomAddress());
        string phone = Interaction.InputBox("Телефон пацієнта", "Додавання пацієнта",
Randomizer.randomPhone());
        string mail = Interaction.InputBox("Пошта пацієнта", "Додавання пацієнта", "pacient_mail @gmail.com");
        string mail_rel = Interaction.InputBox("Пошта родичів", "Додавання пацієнта", "mail@gmail.com");

        Form1.mysql_query.CommandText = "insert into
ISPPR.pacient(name_pacient,date_of_birth,address_pacient,phone_pacient,email,email_relatives)" +
           " values('" + name + "','" + date + "','" + address + "','" + phone + "','" + mail + "','" + mail_rel + "');";
        //MessageBox.Show(Form1.mysql_query.CommandText);
        Form1.mysql_query = new MySqlCommand(Form1.mysql_query.CommandText, Form1.mysql_connection);
        Form1.mysql_connection.Open();
        Form1.mysql_result = Form1.mysql_query.ExecuteReader();
        MessageBox.Show("Пацієнта додано");
        Form1.mysql_connection.Close();
        selectAll("pacient", dataGridView4);
           //додавання паролю
           int max = Int32.Parse(dataGridView4.Rows[dataGridView4.RowCount - 2].Cells[0].Value.ToString());
           string user = "user"+max.ToString();
           //string password = max.ToString();
           //випадковий пароль
           string password = "";
           Random rnd = new Random();
           password += ((char)(rnd.Next(65, 91))).ToString() + (char)(rnd.Next(97, 123)) + (char)(rnd.Next(97, 123))
              + (rnd.Next(0, 10)).ToString() + (rnd.Next(0, 10)).ToString() + (rnd.Next(0, 10)).ToString() + (rnd.Next(0,
10)).ToString() + (rnd.Next(0, 10)).ToString();

           MessageBox.Show(user+" "+password, "Інформація для пацієнта", MessageBoxButtons.OK,
MessageBoxIcon.Information);
```

```
        int idpacient = max;
        Form1.mysql_query.CommandText = "insert into ISPPR.user(login,password,idpacient)" +
        " values('" + user + "','" + password + "','" + idpacient + "');";
        Form1.mysql_query = new MySqlCommand(Form1.mysql_query.CommandText, Form1.mysql_connection);
        Form1.mysql_connection.Open();
        Form1.mysql_result = Form1.mysql_query.ExecuteReader();
        //MessageBox.Show("Пацієнта додано");
        Form1.mysql_connection.Close();

        pacientsToDictionary();
    }
}
private void button7_Click(object sender, EventArgs e)
{
    delete("pacient", dataGridView4);
    pacientsToDictionary();
}
private void button10_Click(object sender, EventArgs e)
{
    this.Hide();
    form4 = new Form4(this);
    form4.Show();
}
private void button12_Click(object sender, EventArgs e)
{
    this.Hide();
    form5 = new Form5(this);
    form5.Show();
}
private void button11_Click(object sender, EventArgs e)
{
    delete("watch", dataGridView6);
    //dietasToDictionary();
}
private void button9_Click(object sender, EventArgs e)
{
    delete("visit", dataGridView5);
    //dietasToDictionary();
}
private void dataGridView5_CellContentClick(object sender, DataGridViewCellEventArgs e)
{

}
```

```
        private void Form2_Shown(object sender, EventArgs e)
        {
        }
        private void Form2_VisibleChanged(object sender, EventArgs e)
        {
            selectAll("visit", dataGridView5);
            if(admin)
                selectAll("watch", dataGridView6);
        }
        private void button13_Click(object sender, EventArgs e)
        {
            //отправка письма врачу
            this.Hide();
            form6 = new Form6(this, email_pacient);
            form6.Show();
        }
    }
}
```

Form3.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Net;
using System.Net.Mail;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ISPPR
{
    public partial class Form3 : Form
    {
        Form form1,form2,form5;
        public Form3(Form form1)
        {
            InitializeComponent();
            this.form1 = form1;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            SmtpClient Smtp = new SmtpClient();
            Smtp.UseDefaultCredentials = false;
            var NetworkCredentials = new NetworkCredential()
            { UserName = "denis.chornenky78625@gmail.", Password = "*******" };
            Smtp.Port = 587;
            Smtp.EnableSsl = true;
            Smtp.Host = "smtp.gmail.com";
```

```
            Smtp.Credentials = NetworkCredentials;

            MailMessage msg = new MailMessage();
            msg.From = new MailAddress("denis.chornenky78625@gmail.com");
            msg.To.Add("denis.chornenky78625@gmail.com");
            msg.Subject = "Увага! Терміново!";
            msg.Body = "Вашому родичу потрібна медична допомога!";
            Smtp.Send(msg);
        }

        private void Form3_Load(object sender, EventArgs e)
        {
            form2 = new Form2(form1,false,0);
            form2.Show();
            form2.Close();

            comboBox1.Items.Clear();
            foreach (var key in Form1.pacients.Keys)
            {
                comboBox1.Items.Add(Form1.pacients[key]);
            }
            comboBox1.Text = comboBox1.Items[0].ToString();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            this.Hide();
            form5 = new Form5(this);
            form5.Show();
        }

        private void Form3_FormClosed(object sender, FormClosedEventArgs e)
        {
            form1.Show();
        }
    }
}
```

Form4.cs:

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ISPPR
{
    public partial class Form4 : Form
    {
        Form form2;
        public Form4(Form form2)
        {
            InitializeComponent();
            this.form2 = form2;
        }

        private void Form4_Load(object sender, EventArgs e)
        {
```

```
            comboBox1.Items.Clear();
            foreach (var key in Form1.pacients.Keys)
            {
                comboBox1.Items.Add(Form1.pacients[key]);
            }
            comboBox1.Text = comboBox1.Items[0].ToString();

            comboBox2.Items.Clear();
            foreach (var key in Form1.doctors.Keys)
            {
                comboBox2.Items.Add(Form1.doctors[key]);
            }
            comboBox2.Text = comboBox2.Items[0].ToString();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            int id_pacient = 0, id_doctor = 0;
            foreach (var key in Form1.pacients.Keys)
            {
                if(Form1.pacients[key]== comboBox1.Text)
                {
                    id_pacient = key;
                    break;
                }
            }

            foreach (var key in Form1.doctors.Keys)
            {
                if (Form1.doctors[key] == comboBox2.Text)
                {
                    id_doctor = key;
                    break;
                }
            }

            Form1.mysql_query.CommandText = "insert into ISPPR.visit(id_pacient,id_doctor,recommendations) " +
                "values('" + id_pacient + "','" + id_doctor + "','" + textBox1.Text + "');";
            Form1.mysql_query = new MySqlCommand(Form1.mysql_query.CommandText, Form1.mysql_connection);
            Form1.mysql_connection.Open();
            Form1.mysql_result = Form1.mysql_query.ExecuteReader();
            Form1.mysql_connection.Close();

            MessageBox.Show("Запис додано", "Інформація", MessageBoxButtons.OK, MessageBoxIcon.Information);
            this.Close();
        }

        private void Form4_FormClosed(object sender, FormClosedEventArgs e)
        {
            form2.Show();
        }
    }
}
```

Form5.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Net;
using System.Net.Mail;
```

```
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ISPPR
{
   public partial class Form6 : Form
   {
      Form form2;
      string email_pacient;
      public Form6(Form form2, string email_pacient)
      {
         InitializeComponent();
         this.form2 = form2;
         this.email_pacient = email_pacient;
      }

      public void sendMail()
      {
         string mailto=textBox1.Text;
         SmtpClient Smtp = new SmtpClient();
         Smtp.UseDefaultCredentials = false;
         var NetworkCredentials = new NetworkCredential()
         //{ UserName = "MAIL@gmail.com", Password = "PASSWORD" };
         { UserName = textBox2.Text, Password = textBox5.Text };
         Smtp.Port = 587;
         Smtp.EnableSsl = true;
         Smtp.Host = "smtp.gmail.com";
         Smtp.Credentials = NetworkCredentials;

         MailMessage msg = new MailMessage();
         //msg.From = new MailAddress("MAIL@gmail.com");
         msg.From = new MailAddress(textBox2.Text);
         msg.To.Add(mailto);
         //msg.Subject = "Увага! Терміново!";
         msg.Subject = textBox3.Text;
         //msg.Body = "Вашому родичу " + pacient + " потрібна медична допомога!";
         msg.Body = textBox4.Text;

         try
         {
            Smtp.Send(msg);
            MessageBox.Show("Лист надіслано", "Інформація", MessageBoxButtons.OK,
MessageBoxIcon.Information);
         }
         catch
         {
            MessageBox.Show("Не вдалося надіслати листа", "Помилка", MessageBoxButtons.OK,
MessageBoxIcon.Information);
         }
      }

      private void button1_Click(object sender, EventArgs e)
      {
         sendMail();
         form2.Show();
         this.Close();
      }

      private void Form6_Load(object sender, EventArgs e)
      {
         textBox1.Text = "denis.chornenky78625@gmail.com";
         //textBox2.Text = "MAIL@gmail.com";
         textBox2.Text = email_pacient;
```

```
            //textBox5.Text = "PASSWORD";
        }

        private void Form6_FormClosed(object sender, FormClosedEventArgs e)
        {
            form2.Show();
        }
    }
}
```