Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему *Метод автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання*

Галузь знань *12 – Інформаційні технології*
Шифр і назва галузі знань

Спеціальність *122 – Комп'ютерні науки*
Шифр і назва спеціальності

Освітня програма *Комп'ютерні науки*
Назва освітньої програми

Виконав: *студент 4 курсу, група КН-18-1* (Курс, група виконавця) _____ (Підпис) *Н.С.Домбровський* (Ініціали, прізвище)

Керівник: *старший викладач кафедри КН* (Науковий ступінь, посада) _____ (Підпис) *Т.К.Скрипник* (Ініціали, прізвище)

Нормоконтроль: *к.т.н., доцент кафедри КН* (Науковий ступінь, посада) _____ (Підпис) *Р.О. Багрій* (Ініціали, прізвище)

До захисту допускаю:
Зав. кафедри КН, д.т.н., професор _____ (Підпис) *О.В. Бармак* (Ініціали, прізвище)
1 червня 2022 р.

Хмельницький 2022

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет *інформаційних технологій*
Кафедра *комп'ютерних наук*
Освітній ступінь *бакалавр*
Галузь знань *12 – Інформаційні технології*
Спеціальність *122 – Комп'ютерні науки*

ЗАТВЕРДЖУЮ
Завідувач кафедри *комп'ютерних наук*

(підпис)
д.т.н., професор О.В. Бармак
«*25*» *березня* *2022* року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

1. Тема кваліфікаційної роботи бакалавра: *«Метод автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання»*

2. Завдання видано студенту *Домбровському Назарію Сергійовичу*
(прізвище, ім'я, по батькові)

3. Керівник роботи *старший викладач кафедри КН Скрипник Тетяна Казимирівна*
(посада, прізвище, ім'я, по батькові)

4. Затверджено наказом університету від «*01*» *березня* *2022* р. № *18*

5. Зміст пояснювальної записки (перелік задач) та вихідні дані:

Мета роботи – розробка методу автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання. Зокрема, реалізація програмного продукту із набором функцій: підбір можливих діагнозів; формування експертного висновку; робота з експертними відомостями.

Виконавець: *студент 4 курсу, група КН-18-1* (Курс, група виконавця) _____ (Підпис) *Н.С.Домбровський* (Ініціали, прізвище)

Керівник: *старший викладач кафедри КН* (Науковий ступінь, посада) _____ (Підпис) *Т.К.Скрипник* (Ініціали, прізвище)

Анотація

Тема кваліфікаційної роботи бакалавра: *«Метод автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання»*

Виконавець кваліфікаційної роботи бакалавра: *студент групи КН-18-1 Домбровський Назарій Сергійович*

Керівник кваліфікаційної роботи бакалавра: *старший викладач кафедри КН Скрипник Тетяна Казимирівна*

Кваліфікаційна робота бакалавра містить:

Пояснювальна записка				Кількість додатків
Сторінок	Рисунків	Таблиць	Джерел інформації	
57	38	25	35	4

Метою кваліфікаційної роботи бакалавра є розробка методу автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання. Для розробки інформаційної системи було використано мову програмування C#, платформу .NET, а також систему керування базами даних MS SQL Server.

Розроблена система призначена для працівників медичних закладів, програмний застосунок допомагатиме в автоматизації роботи лікаря, пришвидшить діагностику здоров'я пацієнта та встановлення діагнозу.

Зокрема, необхідно реалізувати ряд функцій, а саме: облік візитів, бази пацієнтів, симптомів, діагнозів та встановлення експертного висновку щодо діагнозу пацієнта за симптомами захворювання.

Ключові слова: експертна система, терапевт, інформаційна система, хвороба, діагноз, анамнез.

Виконавець: *студент 4 курсу, група КН-18-1* (Курс, група виконавця) _____ (Підпис) *Н.С.Домбровський* (Ініціали, прізвище)

Зміст

Перелік скорочень

Скорочення, термін, позначення	Пояснення
КРБ	Кваліфікаційна робота бакалавра
КН	Комп’ютерні науки
ОЗ	Охорона здоров’я
ІТ	Інформаційні технології
БД	База даних
ЛПЗ	Лікувально-профілактичний заклад
ПП	Програмний продукт
СКБД	Система керування базами даних
ЛПУ	Лікувально-профілактична установа
ПМД	Первинна медична допомога
ЕМК	Електронна медична картка
ЦБД	Центральна база даних
МІС	Медична інформаційна система
ДП	Державне підприємство
ПІБ	Прізвище, ім’я, по батькові
ЕМД	Епізод медичної допомоги
СППР	Система підтримки прийняття рішень
ЕС	Експертна система

Вступ

Здоров’я людини є однією із найбільших її цінностей, тому воно має колосальне значення не лише для особистості, а й для суспільства та держави в цілому. Рівень сфери охорони здоров’я в кожній країні є показником її розвитку, благополуччя громадян та їх забезпеченості. Сьогодення внесло значні переосмислення в сферу охорони здоров’я, особливо зважаючи на пандемію COVID-19, що збільшила обсяги роботи лікарів та завантаження ЛПЗ загалом.

Важливість здоров’я громадянина закріплена і в Конституції України, на законодавчому рівні. Професія медичного працівника сьогодні є стратегічно важливою, адже у період військового стану та пандемії, що досі триває й набуває нових обертів медичний працівник – безцінний, формує стратегічний капітал, а близько 65% бюджету у світі, що спрямований в сферу ОЗ спрямований на кадрове забезпечення.

Коронавірусна хвороба змусила переглянути ставлення громадян до лікарів та до рівня їх завантаженості. Під час пандемії було впроваджено низку нових ІС для моніторингу роботи лікаря та ведення документаційної роботи.

В кожній сфері бізнесу використовуються ІТ-засоби: для автоматизації обчислень, збереження документів та інших даних, формування експертного висновку, тощо.

Галузь охорони здоров’я також потребує діджиталізації, адже сьогодні лікар завантажений не лише лікуванням пацієнтів, а веденням карток, направлень на обстеження організму хворого, видачею рецептів, веденням обмінних карт, тощо. Тому сьогодні фактично неможливо не впроваджувати в ЛПЗ програмне забезпечення медичних інформаційних систем.

Сьогодні засоби штучного інтелекту надають можливість впроваджувати використання комп’ютера в процес визначення діагнозу пацієнта. Це надає ряд значних переваг:

– швидше діагностування хвороби;

– персоналізоване лікування пацієнта;

– постійне оновлення бази діагнозів та симптомів, наприклад, нові штами коронавірусної хвороби, що мають нові симптоми.

ЛПЗ використовують та створюють великі обсяги даних, що часто бувають неструктурованими та із помилками, що виникають механічно. Такі огріхи в роботі можуть спричинити ряд ускладнень, як і з визначенням діагнозів, так і в оформленні супровідної документації, із якою часто взаємодіють інші спеціалісти.

Створення єдиного інформаційного простору, автоматизований документообіг, використання штучного інтелекту для визначення хвороби пацієнта – рішення, не лише пришвидшить й полегшить роботу лікарів, а й підвищить якість їх роботи, мінімізує ймовірність виникнення помилки на етапі ведення карток пацієнтів, направлень та видачі рецептів й призначення лікувань.

Об’єктом дослідження кваліфікаційної роботи бакалавра є метод автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання, предметом – доступні методи, алгоритми та засоби створення експертних систем для визначення діагнозу за симптомами захворювань, мета роботи – розробка методу автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання та його програмна реалізація.

Розділ 1 Характеристика предметної області: аналіз моделей, методів та реалізацій

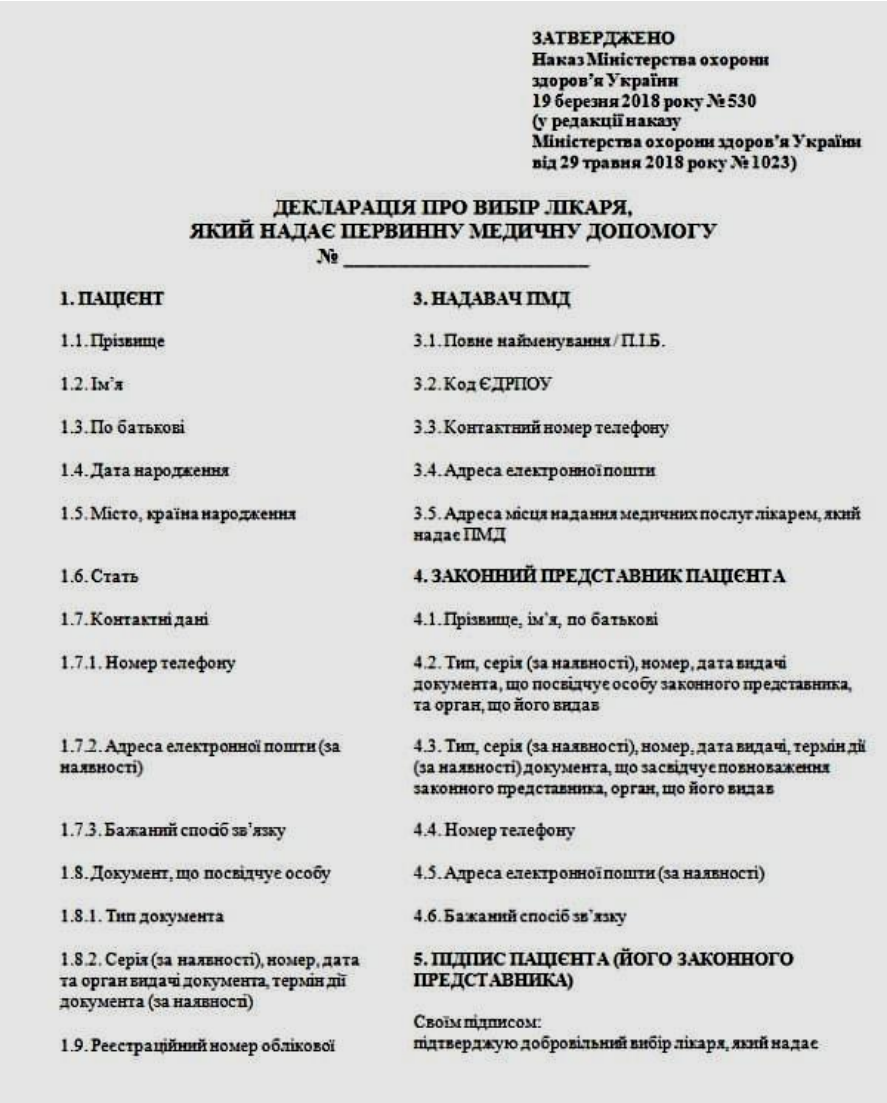
1.1 Аналіз інформаційних моделей

Сьогодні складно уявити хоч одну сферу бізнесу чи послуг, що обходиться без залучення ІТ. Однак медицина та ЛПЗ України ще не повністю перейшла на сучасні технології задля автоматизації та пришвидшення роботи та документообігу. Існує чимало МІС [1] як і для лікарів, так і для пацієнтів, однак дійсно пришвидшити роботу лікаря може автоматизоване визначення діагнозу хворого та автоматична обробка даних симптомів.

На жаль, досі жодна українська МІС не надає такої можливості, навіть повноцінне ПЗ на цю тему недоступне для лікаря, адже платні підписки не фінансуються медичними закладами, а швидкодія їх роботи та складний, громіздкий інтерфейс і зовсім не налаштовують на продуктивну роботу.

Для розширення знань щодо предметної області, було проведено її детальний аналіз. Людина, що звертається по медичну допомогу, тобто пацієнт [2], проходить чимало етапів перед тим, як дізнатись діагноз та перейти до безпосереднього лікування захворювання. Спершу майбутній пацієнт реєструється на візит до сімейного лікаря [3], що проводить поверхневу діагностику та визначає, чи необхідно звернутись до іншого спеціаліста. Лікар-терапевт – це людина, що займається лікуванням багатьох органів і систем, однак не може повністю замінити кардіолога, невролога, дерматолога та інших вузькопрофільних фахівців, так як кожен з них має свої завдання і методи лікування.

Медична реформа в Україні [4] передбачає обов’язкове оформлення декларації з сімейним лікарем для отримання медичних послуг. Для того, щоб мати змогу записатись на прийом до лікаря, необхідно заключити договір [5] (рисунок 1.1)



ЗАТВЕРДЖЕНО
Наказ Міністерства охорони здоров'я України
19 березня 2018 року № 530
(у редакції наказу Міністерства охорони здоров'я України від 29 травня 2018 року № 1023)

ДЕКЛАРАЦІЯ ПРО ВИБІР ЛІКАРЯ, ЯКИЙ НАДАЄ ПЕРВИННУ МЕДИЧНУ ДОПОМОГУ
№ _____

1. ПАЦІЄНТ

1.1. Прізвище

1.2. Ім'я

1.3. По батькові

1.4. Дата народження

1.5. Місто, країна народження

1.6. Стать

1.7. Контактні дані

1.7.1. Номер телефону

1.7.2. Адреса електронної пошти (за наявності)

1.7.3. Бажаний спосіб зв'язку

1.8. Документ, що посвідчує особу

1.8.1. Тип документа

1.8.2. Серія (за наявності), номер, дата та орган видачі документа, термін дії документа (за наявності)

1.9. Реєстраційний номер облікової

3. НАДАВАЧ ПМД

3.1. Повне найменування / П.І.Б.

3.2. Код ЄДРПОУ

3.3. Контактний номер телефону

3.4. Адреса електронної пошти

3.5. Адреса місця надання медичних послуг лікарем, який надає ПМД

4. ЗАКОННИЙ ПРЕДСТАВНИК ПАЦІЄНТА

4.1. Прізвище, ім'я, по батькові

4.2. Тип, серія (за наявності), номер, дата видачі документа, що посвідчує особу законного представника, та орган, що його видав

4.3. Тип, серія (за наявності), номер, дата видачі, термін дії (за наявності) документа, що засвідчує повноваження законного представника, орган, що його видав

4.4. Номер телефону

4.5. Адреса електронної пошти (за наявності)

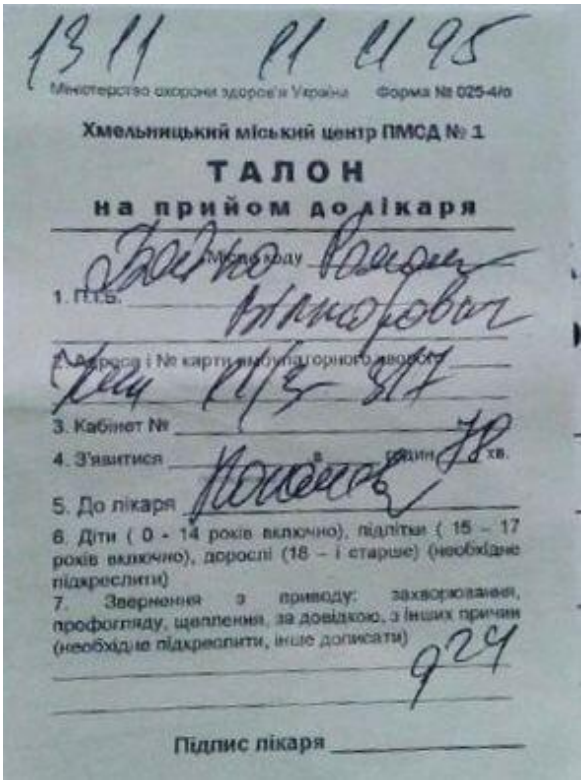
4.6. Бажаний спосіб зв'язку

5. ПІДПИС ПАЦІЄНТА (ЙОГО ЗАКОННОГО ПРЕДСТАВНИКА)

Своїм підписом:
підтверджую добровільний вибір лікаря, який надає

Рисунок 1.1 – Зразок оформлення декларації про вибір сімейного лікаря [5]

Якщо договір підписаний – громадянин може записати на прийом. Сьогодні переважна більшість ЛПЗ використовують медичні інформаційні системи для реєстрації на прийом та ведення обліку візитів. Раніше це здійснювалось через реєстратуру, однак це не гарантувало запис в той день, паперові талони [6] (рисунок 1.2) часто губились.


Міністерство охорони здоров'я України Форма № 025-4/о

Хмельницький міський центр ПМСД № 1

ТАЛОН

на прийом до лікаря

1. П.І.Б.

2. Адреса і № карти амбулаторного хворого

3. Кабінет №

4. З'явитися о годині хв.

5. До лікаря

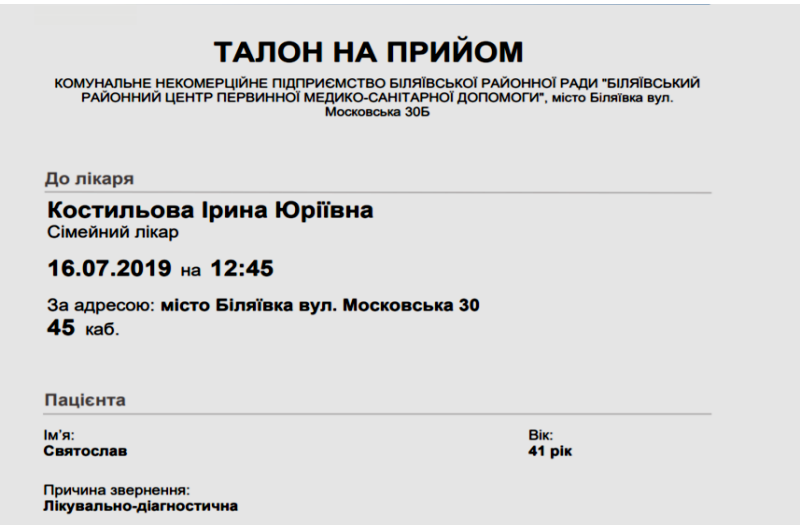
6. Діти (0 - 14 років включно), підлітки (15 – 17 років включно), дорослі (18 – і старше) (необхідне підкреслити)

7. Звернення з приводу: захворювання, профогляду, щеплення, за довідкою, з інших причин (необхідне підкреслити, інше дописати)

Підпис лікаря

Рисунок 1.2 – Зразок паперового талону на запис до лікаря [6]

ЛПЗ міста Хмельницького використовують медичну інформаційну систему «Medics» [7]. Ця МІС забезпечує роботу як лікарів, так і пацієнтів. Можна записатись на прийом, переглянути доступних лікарів у різних ЛПЗ та побачити виконані записи. Приклад запису до лікаря наведено на рисунку 1.3.


ТАЛОН НА ПРИЙОМ

КОМУНАЛЬНЕ НЕКОМЕРЦІЙНЕ ПІДПРИЄМСТВО БІЛЯЇВСЬКОЇ РАЙОННОЇ РАДИ "БІЛЯЇВСЬКИЙ РАЙОННИЙ ЦЕНТР ПЕРВИННОЇ МЕДИКО-САНІТАРНОЇ ДОПОМОГИ", місто Біляївка вул. Московська 30Б

До лікаря

Костильова Ірина Юріївна
Сімейний лікар

16.07.2019 на **12:45**

За адресою: **місто Біляївка вул. Московська 30**
45 каб.

Пацієнта

Ім'я:
Святослав

Вік:
41 рік

Причина звернення:
Лікувально-діагностична

Рисунок 1.3 – Зразок електронного талону на запис до лікаря [7]

Всю інформацію про перебіг хвороби пацієнта, призначене лікування та результати лабораторних досліджень раніше записувались в паперовій медичній картці [8] (рисунок 1.4).

Рисунок 1.4 – Паперова медична картка [8]

Раніше картка пацієнта мала паперовий вигляд і мала зберігатися лише в реєстратурі, де часто губилась, однак у сучасних умовах діджиталізації все більше ЛПЗ надають перевагу створенню електронних карток та перенесенню туди усієї інформації.

Визначення діагнозу лікарем часто може бути хибним через брак відповідних обстежень та інших даних від пацієнта. Вирішенню цієї проблеми сприяє залучення методів експертних систем [9]. ЕС допомагають швидко аналізувати дані за вказаними параметрами та формувати висновок, опираючись на математичні обчислення. Для методу автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання будуть застосовані принципи ЕС, що визначатимуть діагноз пацієнта за введеними лікарем симптомами.

Технології автоматизованого визначення діагнозу базуються на експертних системах, що є одним із найпоширеніших типів систем штучного

інтелекту [10]. Свій активний розвиток ЕС почали із 1960-х років і розглядалися як штучний інтелект спеціального типу, призначеного для успішного вирішення складних задач у вузькій предметній галузі, такий як медична діагностика захворювань.

Підсумовуючи, було досліджено предметну область КРБ та визначено основні поняття, що зустрічатимуться в подальшій роботі.

1.2 Огляд теоретичних підходів до розв'язку подібних задач

Експертні системи – це напрям в області штучного інтелекту, що стрімко розвивається. З допомогою ЕС можуть бути вирішені чи не будь-які задачі, із якими стикається людина в процесі виробництва чи іншої діяльності. Експертна система [11] – це набір програм або програмне забезпечення, що виконує функції експерта при рішенні якої-небудь задачі в області його компетенції. Протягом багатьох років ЕС викликають інтерес розробників та математиків, адже їх властивості можна застосувати до будь-якої сфери діяльності людини.

Для створення експертних систем знання щодо предметної області поміщують в бази даних, при цьому правильно їх формалізувавши. Здійснюється це для того, щоб система могла будь-які дані інтерпретувати для подальших обчислень та доповнень. ЕС в результаті роботи виводять користувачеві висновок, пораду, проводять складний аналіз даних. Призначені ЕС для вирішення задач, що потребують залучення людського мислення, проведення експертизи фахівцем.

Значною перевагою використання таких засобів штучного інтелекту є здатність вирішення задач, що містять неструктуровані дані.

Найбільша перевага експертних систем [12] – здатність накопичувати здобуті знання, зберігати та використовувати в подальшій роботі без додаткових внесень інформації в базу даних під час роботи. Для створення ЕС діагностики захворювань використовують базу даних та інтерпретатор, що аналізує дані, введені користувачем.

Під час створення ЕС можна зауважити наступні труднощі:

1. Часто буває складно інтерпретувати знання від людини в дані, зрозумілі для компілятора, тому до створення ЕС необхідно залучати висококваліфікованих експертів для нормалізації даних.

2. Для перетворення даних та вирішення задач представлення часто необхідно витрачати велику кількість часу.

Основою будь-якої ЕС є база даних: її представлення формулює трійка, де F - сукупність явних фактів, що зберігаються в БД в явному вигляді, R – сукупність правил виведення, котрі забезпечують можливість на основі відомих знань набувати нові, P – сукупність процедур, що визначають, яким чином слід застосовувати правила виведення.

База даних зберігає знання експертів, формалізовані в зрозумілому для компілятора вигляді. Сьогодні використовують декілька основних форм представлення [13]:

Правило «трійки»: об’єкт, його атрибут та значення. Дана форма представлення інформації визначає певний об’єкт, що володіє рядом атрибутів та заповнює їх значення, що можуть приймати їх з відомого набору.

1. Продукційні правила: мають наступний вигляд «ЯКЩО пацієнт хворий COVID-19, стадія захворювання початкова, ТО температура висока з імовірністю = 0.95 та головний біль є з імовірністю = 0.8».

2. Фрейм: містить іменовану таблицю з певною кількістю слотів, що мають власні імена та в процесі роботи машини виводу висновку деякі значення. В значеннях можуть використовуватись константи, посилання на фрейми більш високого чи більш низького рівня, а також обчислювальні процедури.

3. Семантична мережа: є орієнтованим графом, вершини якого представляють події, об’єкти, а дуги описують відносини між вершинами.

Типова експертна система, побудована на основі продукційних правил працює за схемою наведеною на рисунку 1.5.

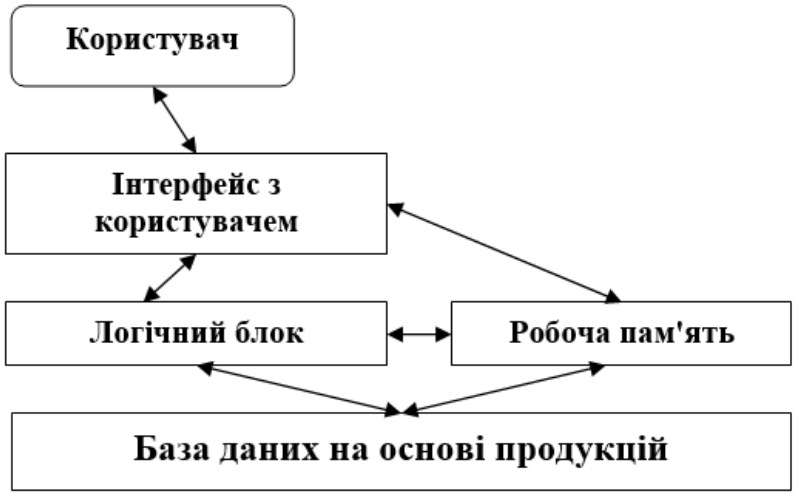


Рисунок 1.5 – Типова схема роботи продукційної системи

Використання нових інформаційних технологій для створення експертних систем медичної діагностики, що поєднують знання і досвід лікарів , є важливим завданням.

Для реалізації методу автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання та програмного застосунку на його основі будуть використані принципи розробки ЕС.

1.3 Аналіз існуючих програмних рішень

Один із найпопулярніших ресурсів для визначення діагнозу онлайн – «Symptomate» (рисунок 1.6) [14]. Роботу цього веб-застосування підтримують потужні засоби експертних систем.

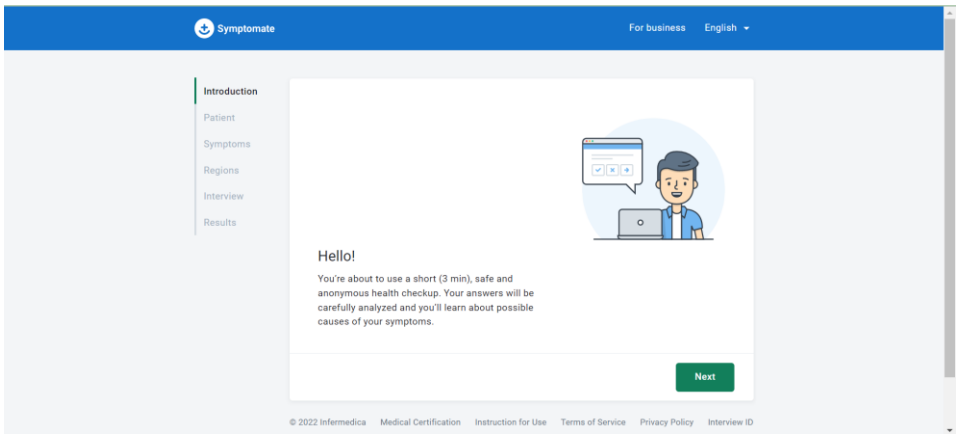


Рисунок 1.6 – Головна сторінка сайту «Symptomate» [14]

Система пропонує автоматизовані пропозиції щодо розширення списку симптомів, однак вони не завжди пов’язані із реальною хворобою пацієнта.

Розпочавши роботу із веб-застосунком користувач обирає стать та вік. Система одразу пропонує обрати найпоширеніші захворювання та звички, які потенційно можуть виникнути в людей визначеного віку та статі (рисунок 1.7).

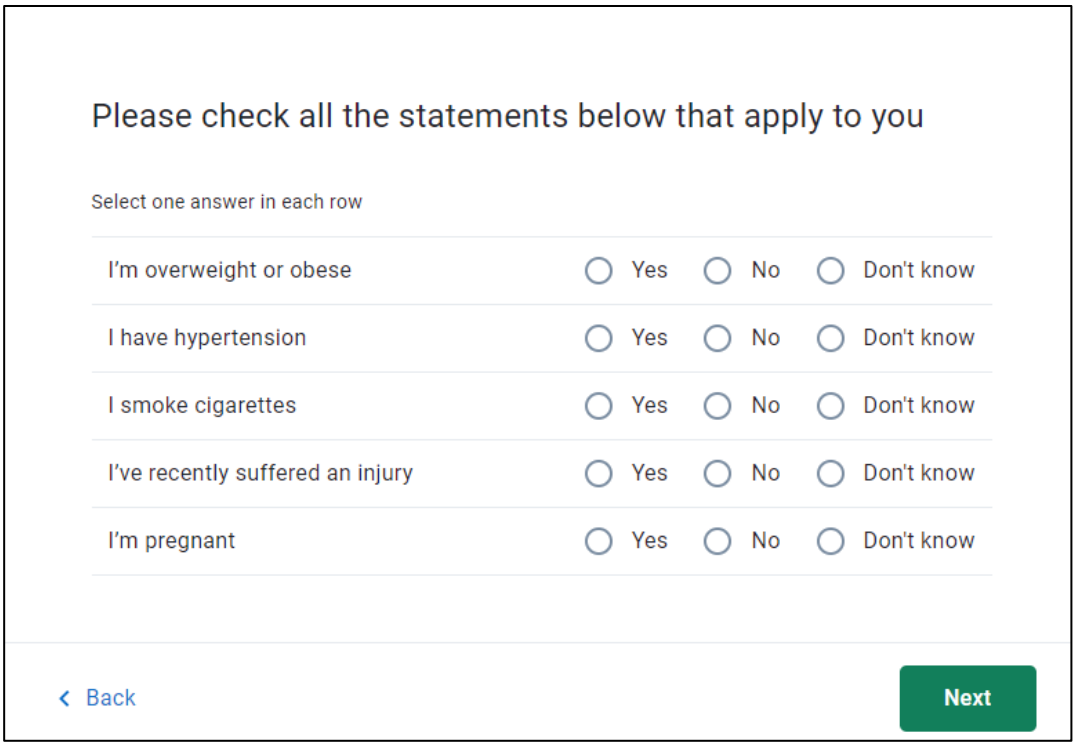


Рисунок 1.7 – Приклад роботи сайту «Sympthomate» [14]

Далі система пропонує ввести усі симптоми, які помітив у себе користувач (рисунок 1.8).

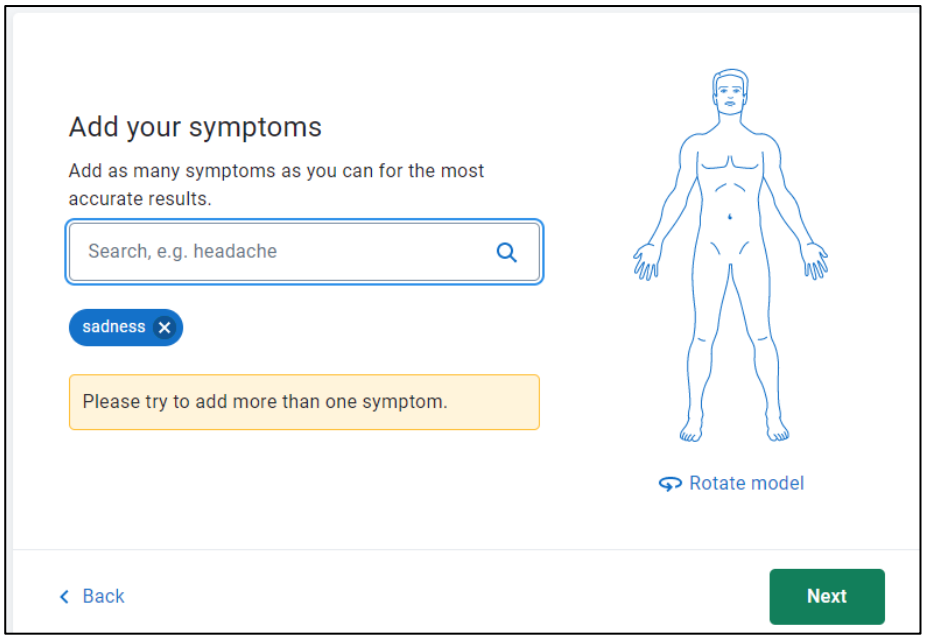


Рисунок 1.8 – Приклад роботи сайту «Sympthomate» [14]

Також користувачеві запропоновано обрати регіон, в якому проживає, адже це може вплинути на результати дослідження (рисунок 1.9).

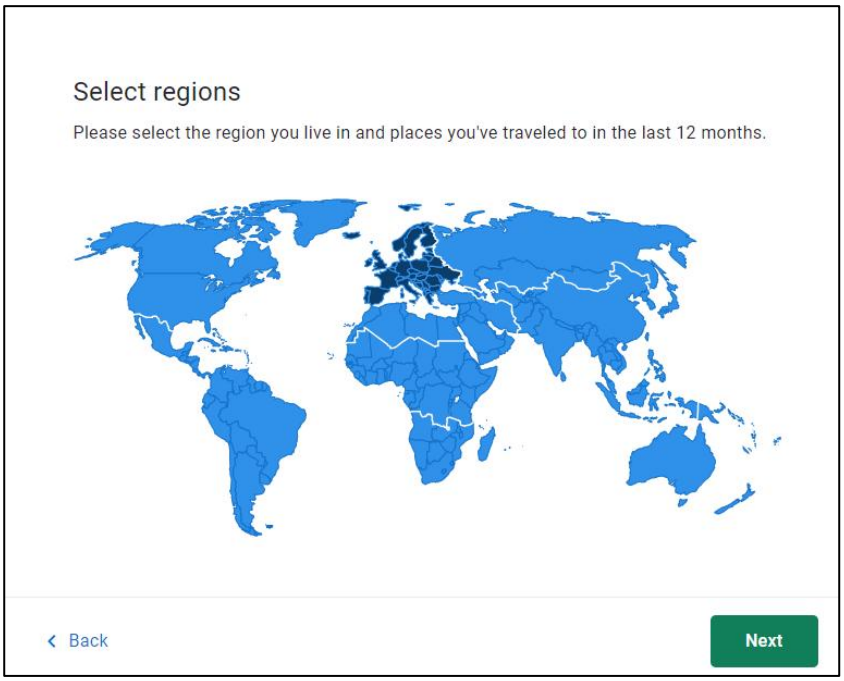


Рисунок 1.9 – Приклад роботи сайту «Sympthomate» [14]

Після введення необхідних даних система пропонує обрати симптоми, що також можуть спостерігатись при певному захворюванні, яке намагається визначити користувач (рисунок 2.5).

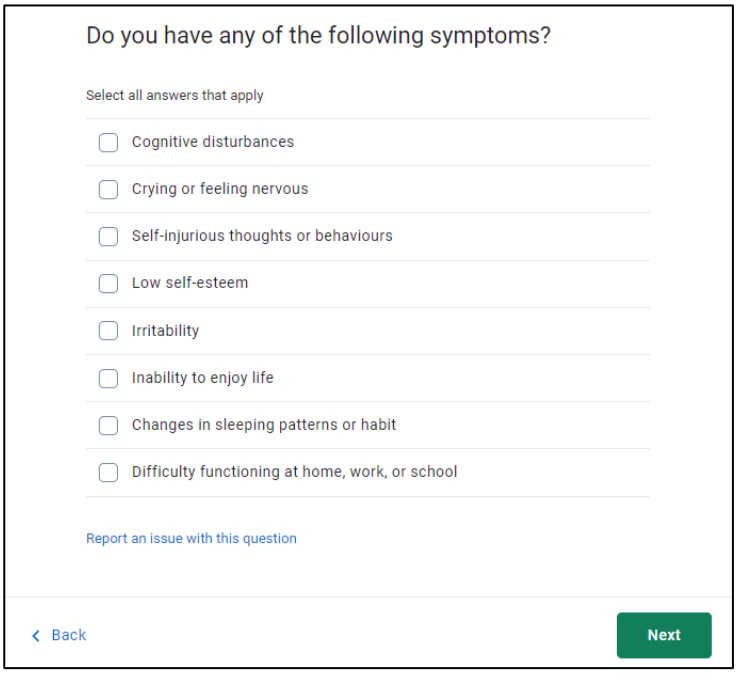


Рисунок 1.10 – Приклад роботи сайту «Sympthomate» [14]

В результаті проходження тестування система формує експертний висновок (рисунок 1.11).

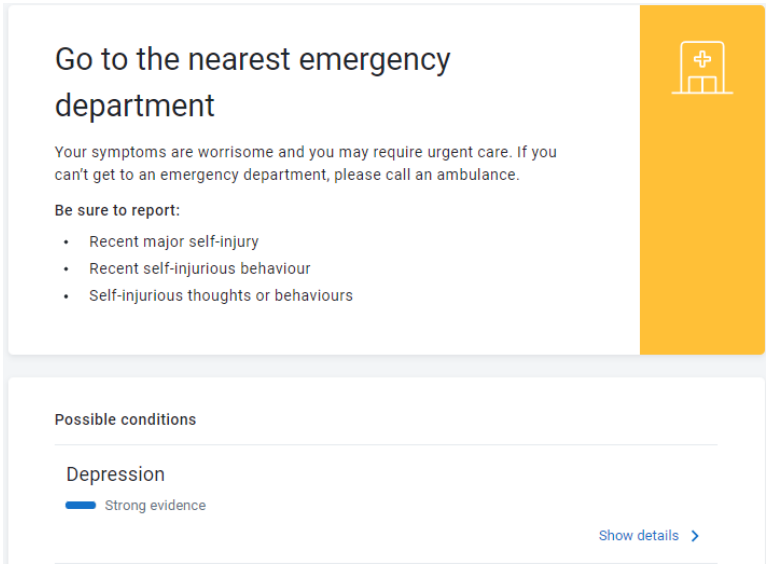


Рисунок 1.11 – Приклад роботи сайту «Sympthomate» [14]

Це застосування не набуло широкого використання в українських ЛПЗ. Недоліком цієї системи є висока вартість використання, а в результаті роботи,

експертний висновок сформовано із надто великими розбіжностями в результатах.

Також було досліджено систему онлайн-діагностики здоров'я від мережі аптек «АНЦ» (рисунок 1.12) [15].

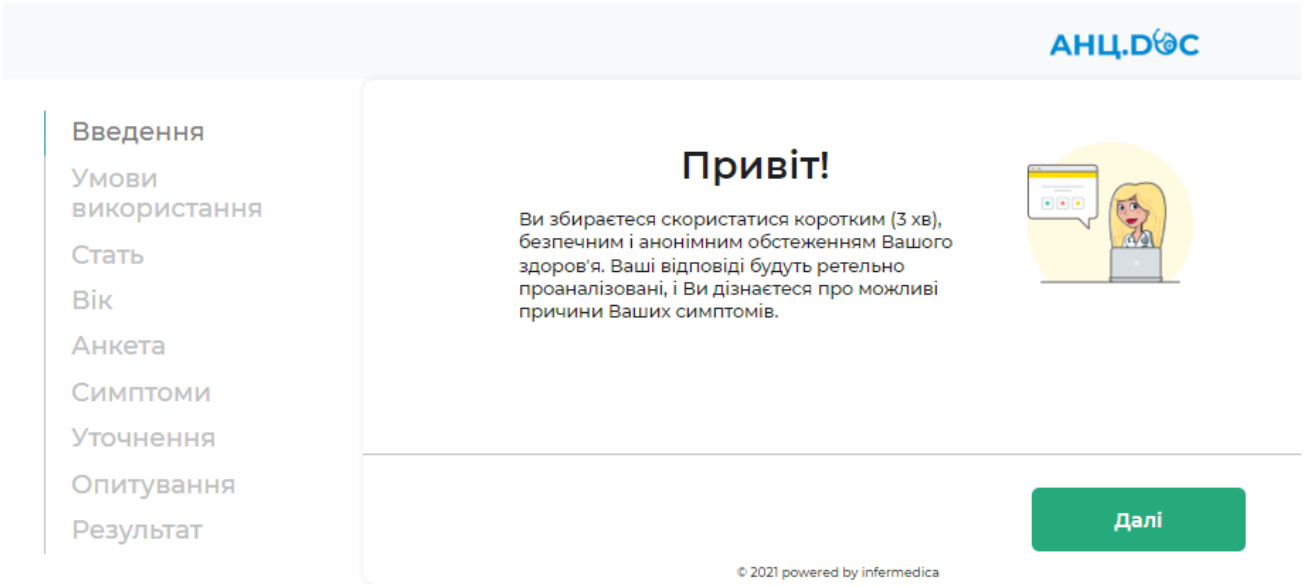


Рисунок 1.12 – Головна сторінка сайту «АНЦ Діагностика» [15]

Після введення статі та віку особи, що потребує пройти діагностику система генерує запитання щодо самопочуття, хронічних захворювань та шкідливих звичок (рисунок 1.13).

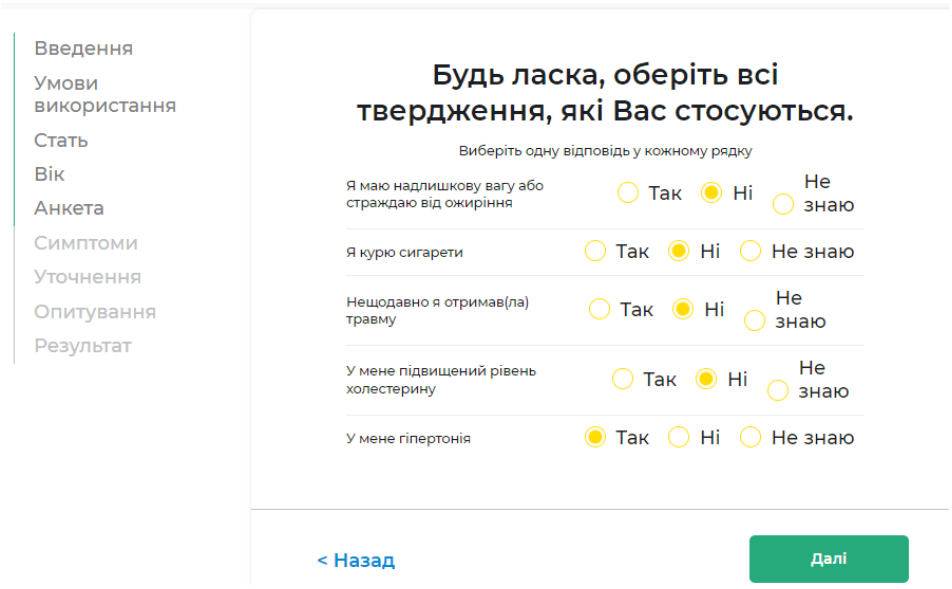


Рисунок 1.13 – Робота сайту «АНЦ Діагностика» [15]

Далі користувач має обрати симптоми, також реалізовано пошук по частинах тіла у вигляді графічного застосунку (рисунок 1.14).

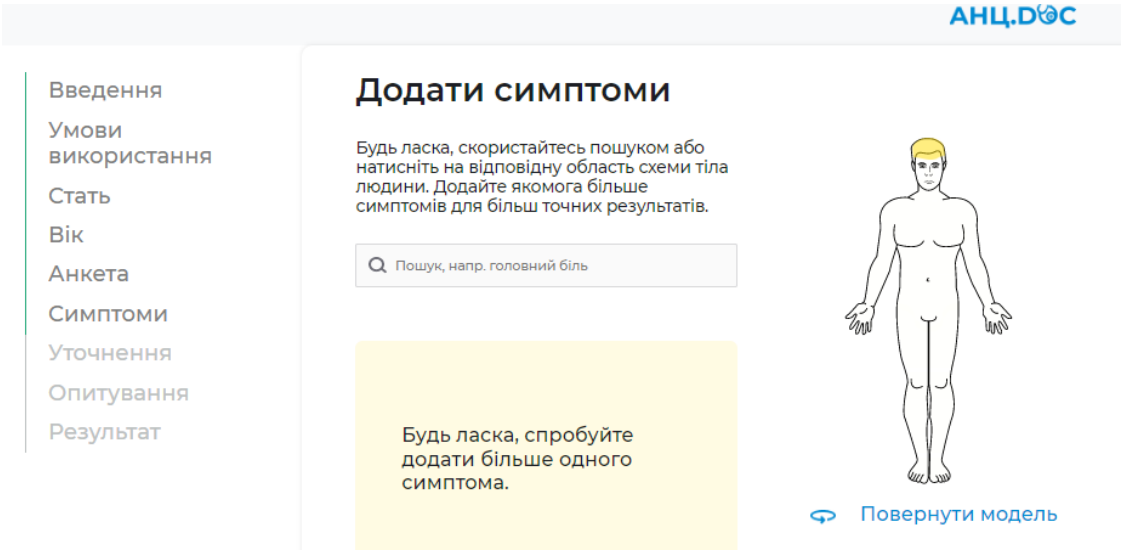


Рисунок 1.14 – Робота сайту «АНЦ Діагностика» [15]

Система не виводить потенційну хворобу після введення усіх симптомів та відповіді на додаткові питання – лише рекомендацію, чи варто звернутись до лікаря (рисунок 1.15).

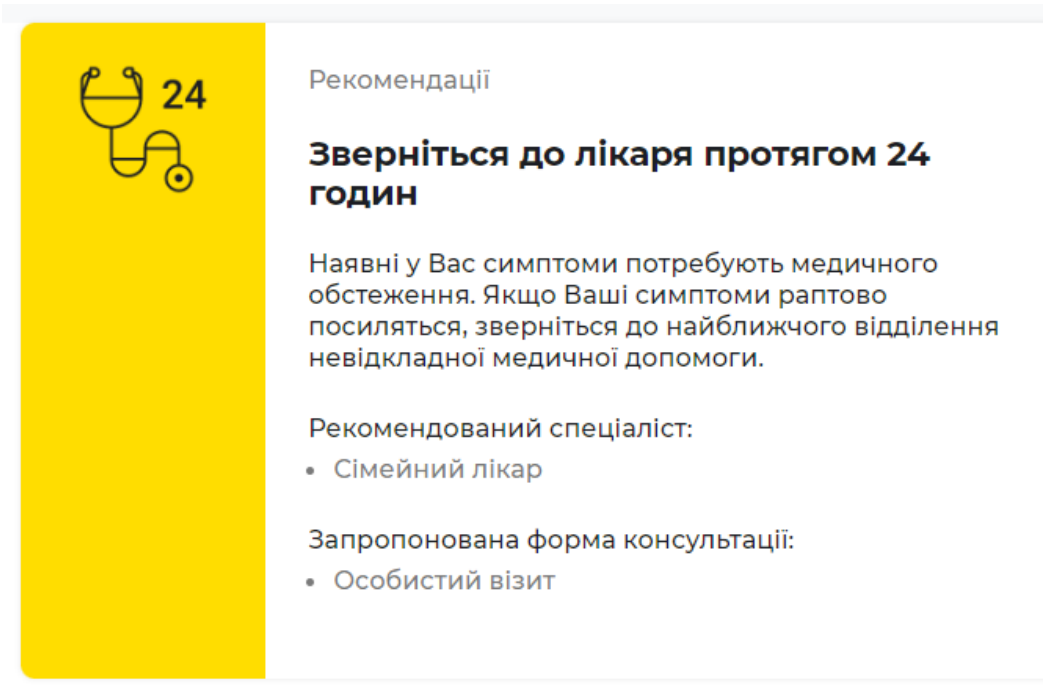


Рисунок 1.16 – Робота сайту «АНЦ Діагностика» [15]

МІС в сучасній медицині, зокрема використання експертних систем у визначенні діагнозу пацієнта – перспективна та передова технологія, що дозволить значно пришвидшити роботу лікарів та здоров’я людей, адже час на діагностування хвороби скорочується.

Як підсумок, проаналізовано популярні системи семантичного аналізу тексту. Було встановлено, застосунків такого типу є свої недоліки й переваги, що обов’язково варто врахувати під час реалізації власного методу метод автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання та створенні програмного продукту на його основі.

1.4 Аналіз сучасних засобів створення програмного забезпечення

Для створення методу автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання. було проаналізовано платформи .NET, Java та доступні фреймворки.

Платформа .NET [16] призначена об'єктно-орієнтованої розробки, яка славиться використанням меншої кількості коду скрізь, де це можливо. Багаторазовий код і багаторазово використовувані компоненти дозволяють розробникам витрачати набагато менше часу на розробку додатків. Коротший життєвий цикл розробки має на увазі дешевше програмне забезпечення в результаті.

.NET підтримує модульну конструкцію, що сприяє простішому розгортанню. Те саме стосується налагодження та обслуговування: можна розібрати додаток, виправити його частинами і знову зібрати. Такий підхід заощаджує, коли весь код не потрібно перевіряти, щоб знайти рядок з помилками.

Сама наявність екосистеми .NET [17] сама по собі є великою перевагою: використовуючи всього одну мову програмування, розробник може створювати

різні програми: настільні, серверні, мобільні (багато хто вважає це досить умовним, але все ж таки), веб-додатки та ігрові.

Сам вибір між технологіями навряд чи має сенс, якщо немає чіткої концепції майбутнього програмного продукту. Чотири критичні властивості можуть полегшити пошук правильного бачення вашого проекту. Рекомендується пам'ятати про вищезгадані порівняльні характеристики Java та .NET при розгляді властивостей вашої майбутньої програми.

Масштаб проекту [18]. Прості інструменти, готові до використання шаблони та платформи з відкритим вихідним кодом відповідають невеликим проектам, коли максимально швидка доставка MVP та зворотний зв'язок з користувачами є більш важливими, ніж будь-яка просунута функціональність. Комбінація кількох мов програмування та фреймворків має сенс для проекту середнього розміру, що потребує великих технологічних знань. ERP-системи, соціальні мережі, онлайн-маркети та інші масштабні рішення вимагають високорівневих технологічних стеків, які здатні забезпечити складну функціональність. Схоже, що остання категорія проектів відповідає як Java, і .NET.

Масштабованість. Кожен проект повинен враховувати потенціал масштабування свого технологічного стека. Це може бути як вертикальна масштабованість, коли додаток додаються нові функції, так і горизонтальна, коли на сервер додаються нові обчислювальні одиниці. Чим складніший проект, тим ширша масштабованість потрібна. І знову ж таки, і Java, і .NET забезпечують достатній потенціал масштабованості для систем масштабу підприємства.

Було розглянуто фреймворки .NET Framework 4.7 та .NET Core.

Microsoft .NET Core [19] – це безкоштовна платформа розробки загального призначення з відкритим вихідним кодом для створення хмарних програм на Windows, Linux і macOS, в той час як Microsoft .Net Framework [20] – це платформа розробки програмного забезпечення для створення і запуску програм Windows .Net Framework включає інструменти розробника, мови

програмування та бібліотеки, які використовуються для розробки настільних та веб-додатків. Ключова різниця між .NET Core та .NET Framework - це платформа для додатків .NET на Windows, тоді як NET Core – це остання версія .NET Framework, яка є кросплатформовою та відкритою платформою, оптимізованою для сучасних додатків та робочих процесів розробників. У таблиці 1.1 наведено порівняння [21] .NET Core та .NET Framework

Таблиця 1.1 – Порівняння .NET Core та .NET Framework

№П/П	Назва атрибуту	.NET Core	.NET Framework
1	Платформа або фреймворк	Коли ми говоримо про .NET Core, його визначають як платформу, на яку спираються такі фреймворки, як ASP.NET Core та Universal Windows Platform, та розширюють функціональні можливості платформи .NET Core.	.Net Framework - це повноцінне середовище розробки. Вона надає всі основні вимоги для розробки додатків, такі як інтерфейс користувача, можливість підключення до БД, сервіси, API і т.д..
2	З відкритим вихідним кодом	.NET Core – це платформа з відкритим вихідним кодом.	До складу платформи .Net Framework входять деякі компоненти з відкритим кодом.
3	Крос-платформність	Заснована на концепції "створи один раз, запускай де завгодно". Оскільки вона є кросплатформною, вона сумісна з різними операційними системами, включаючи Windows, Linux та Mac OS.	.NET Framework сумісний лише з Windows OS (операційна система)
4	Моделі додатків	Модель програм .Net Core включає ASP.NET та універсальні програми windows.	Модель програм .NET Framework включає WinForms, ASP.NET та WPF.
5	Встановлення	.Net Core є кросплатформним, тому його необхідно встановлювати самостійно.	.NET Framework має єдине упаковане середовище установки та виконання для windows.
6	Підтримка мікросервісів	.NET Core має підтримку мікросервісів. NET Core дозволяє змішувати технології, які можуть бути мінімалізовані для кожного мікросервісу.	Коли ми говоримо про .NET Framework, він не дозволяє будувати та розгортати мікросервіси кількома мовами.
7	Продуктивність	Ядро .NET забезпечує високу	.NET Framework менш

	та масштабованість	масштабованість та продуктивність у порівнянні з .NET Framework завдяки своїй архітектурі.	масштабується і забезпечує низьку продуктивність порівняно з .NET Core.
8	Безпека	Така функція як безпека доступу до коду відсутня в .NET Core, тому .NET Framework має перевагу в цьому випадку.	У .NET Framework є така функція, яка називається безпекою доступу до коду.
9	Орієнтація на пристрої	.NET Core орієнтований на розробку програм у різних галузях, таких як ігри, мобільні пристрої, штучний інтелект і т.д.	.NET Framework обмежений Windows OS.
10	Сумісність	Mobile.NET Core сумісний з різними операційними системами – Windows, Linux та Mac OS.	З іншого боку, .NET Framework сумісний лише з ОС Windows.
11	Мобільна розробка	Мобільні програми стають все більш важливими для бізнесу. .NET Core має деяку підтримку мобільних програм. Він сумісний з Xamarin та іншими платформами з відкритим кодом для мобільних додатків.	.NET Framework взагалі не підтримує їхню розробку, і це проблема.

Враховуючи специфіку предметної області та поставлену задачу, було обрано платформу .NET та фреймворк .NET Framework 4.5 для створення програмного продукту методу автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання.

1.5 Мета, задачі та вимоги до реалізації інформаційної системи

Метою кваліфікаційної роботи є розробка методу автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання, та його програмна реалізація.

Прикладна програмна реалізація створеного методу має бути здійснена у вигляді застосунку на платформі .NET, що виконує наступні основні функції:

– первинна реєстрація симптомів та їх параметрів;

– підбір можливих діагнозів та їх підтвердження шляхом формування додаткових запитів на діагностування;

– формування експертного висновку із поясненням діагностування захворювань за наявними симптомами;

– робота з експертними відомостями (продукційними правилами діагностування захворювань по симптомах);

– робота з допоміжними даними (хворобами, симптомами, візитами пацієнтів, даними лікарів, деклараціями й індивідуальними медичними картками пацієнтів ЛПЗ).

Також для досягнення мети необхідно здійснити наступні етапи: характеризувати предметну область, дослідити існуючі алгоритми та методи побудови експертних систем, спроєктувати модель інформаційної системи із відповідним математично-алгоритмічним забезпеченням, реалізувати програмний застосунок на базі створених методів та дослідити ефективність створеного програмного продукту.

Розділ 2 Проєктування інформаційної системи

2.1 Аналіз та автоматизація обробки потоків даних

2.1.1 Методи автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання

Програмний продукт на базі методу автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання розрахований на одну групу користувачів – працівників ЛПЗ: як і лікарів-терапевтів, так і вузькопрофільних спеціалістів.

Групи функцій користувача наведено на рисунку 2.1.

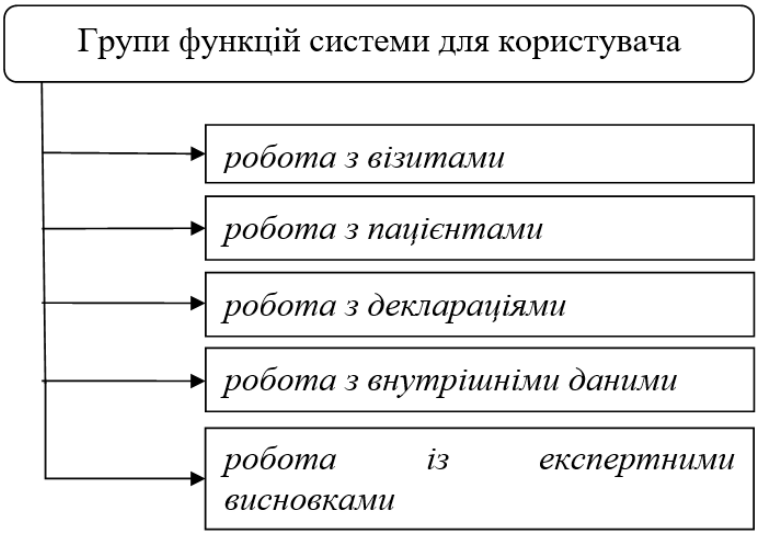


Рисунок 2.1 – Групи функцій користувача програмного застосунку

Схему методу автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання наведено на рисунку 2.2.

На першому кроці роботи методу визначають вхідні дані – симптоми та їх вираженість, що вводитимуться до відомостей користувачем.

Етап 1 – Аналіз отриманих даних, а саме: перевірка повноти відомостей, тобто системою буде перевірено, чи правильно користувач ввів дані у відомість та чи достатньо симптомів для подальшого аналізу.

Далі буде виконано обчислення результатів за допомогою суматорів та формування результатів.

Якщо всі дані були введені правильно, формується експертний висновок, в протилежному випадку – користувачеві буде запропоновано ввести більше симптомів для дослідження.

Етап 3 – передача експертного висновку в карту пацієнта та призначення лікування. На цьому етапі всі дані передаються до бази даних та визначається лікування для пацієнта.

Сформований експертний висновок щодо діагнозу пацієнта виводиться в окреме поле для перегляду та передається для запису в БД.

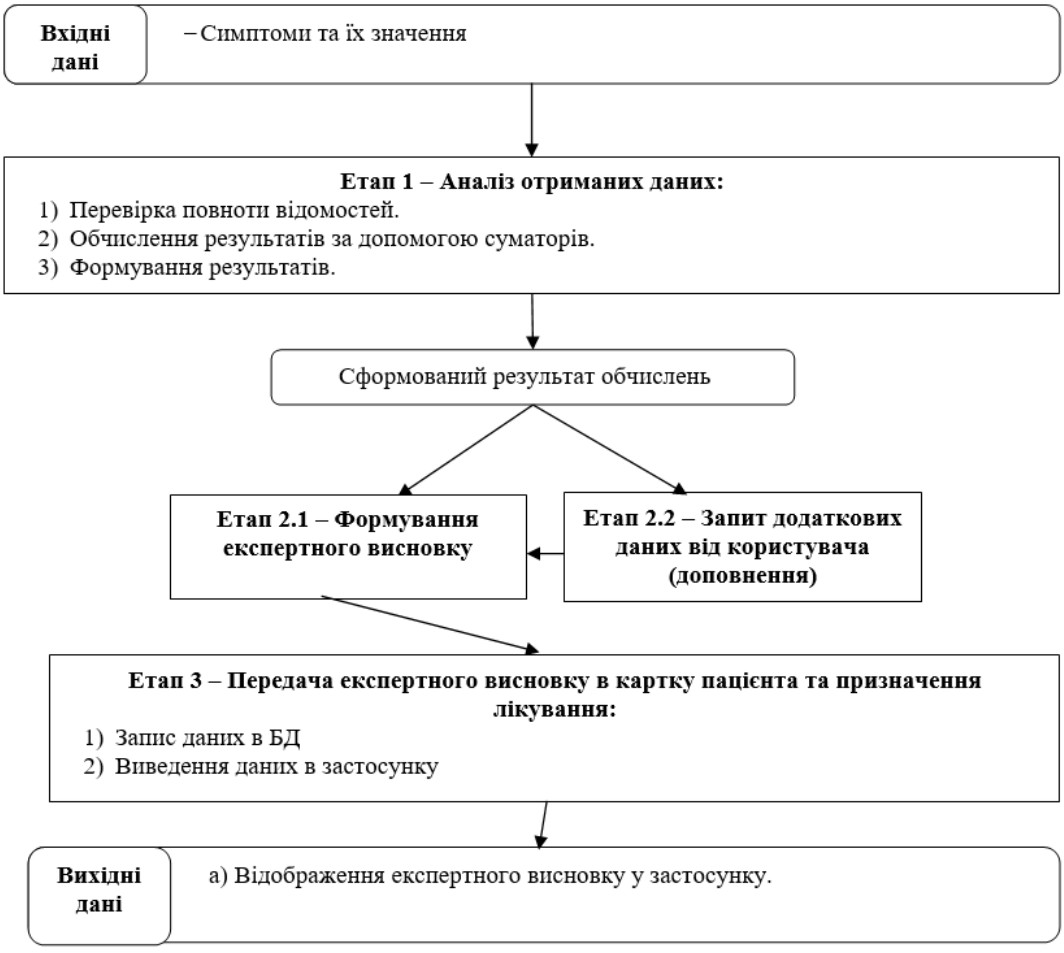


Рисунок 2.2 – Схема методу автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання

Таким чином, було представлено схему роботи методу автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання.

2.1.2 Математично-алгоритмічне забезпечення

В ряді робіт [22, 23] пропонуються підходи, що засновані кількісних операціях. Алгоритм максимальної оцінки інформації [24] (2.1) побудовано на логічних операціях. У цьому випадку здійснюється постійне порівняння інформації від пацієнта з інформацією напрацьованою експертами. Задача ймовірнісного діагнозу може бути сформульована як задача визначення ймовірності даної хвороби по заданому набору симптомів, якщо відомі ймовірні наявності окремих симптомів при певних захворюваннях. Для вирішення цієї задачі в обчислювальній діагностиці застосовується:

$$P\left(\frac{D_i}{E}\right) = \frac{P(E/D_i) \cdot P(D_i)}{\sum_{k=1}^{n} P(E/D_k) \cdot P(D_k)}, \qquad (2.1)$$

де $P(D_i/E)$ – ймовірність, що за наявності комплексу симптомів E є хвороба D; $P(D_i)$ – ймовірність захворювання при випадковому виборі; $P(\frac{E}{D_k})$ – відома з медичного досвіду ймовірність наявності симптомів комплексу E при хворобі D. $\sum_{k=1}^{n} P(E/D_k) \cdot P(D_k)$ – сума добутків ймовірностей кожної з розглянутих хвороб на ймовірність даного комплексу симптомів при кожній з цих хвороб.

Обчислювана величина відображатиме той показник, який накопичено в даний час і виражений кількісно у вигляді ймовірності даного симптомокомплексу при певному захворюванні. Ця ймовірність може бути виражена числом від 0 до 1,0. Також, вони можуть бути виражені приблизно таким чином: симптомокомплекси мають ймовірність P = 1.0; симптомокомплекси з високою частотою P = 0.7-0.9; симптомокомплекси з середньою частотою P = 0.5-0.6; симптомокомплекси з низькою частотою P =

0.3-0.4; симптомокомплекси з дуже низькою частотою P = 0.1-0.2; ймовірність P = 0, означає, що симптомокомплекс при даному захворюванні ніколи не зустрічається.

Сенс введення в діагностику величини $P(D_i)$ полягає в тому, що вона непостійна і залежить від географічних, сезонних, епідеміологічних та інших факторів, які лікар повинен враховувати при встановленні діагнозу.

Для реалізації методу автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання., та реалізації програмного продукту на його основі із функціями, заданими в розділі 1.5 будуть використані формули та методи, що описані вище.

2.2 Інформаційна структура системи

2.2.1 Проектна архітектура системи та взаємозв'язок компонентів

Метод інтелектуальної обробки даних симптомів і автоматизованого прийняття рішень щодо діагностування захворювань пацієнтів ЛПЗ складається із двох підсистем та БД (рисунок 2.3).

Першою є підсистема, необхідна для роботи із БД, у якій зберігаються вищеописані дані. Ця підсистема виконує такі функції для роботи із БД як: вивід даних з БД, їх редагування та видалення. Окрім того, БД зберігатиме і проміжні результати для подальшої роботи із експертною системою.

Друга підсистема – призначена для роботи ЕС. Саме з її допомогою будуть генеруватись експертні висновки, оброблюватись введені користувачем дані щодо симптомів.

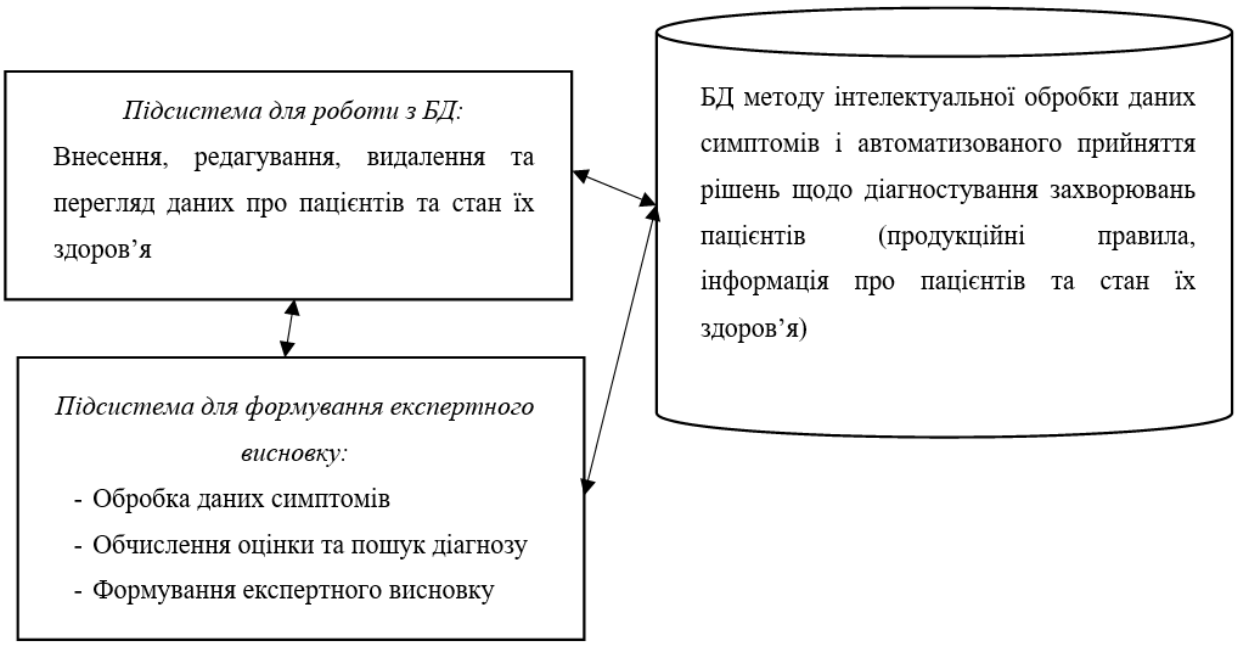


Рисунок 2.3 – Структура компонентів методу автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання

2.2.2 Інформаційна модель

База даних, із якою безперервно працюватиме наш застосунок буде зберігати не лише інформацію про пацієнтів, їх лікарські картки, тощо, а й являтиме собою базу знань для роботи експертної системи: в ній зберігатимуться дані про діагнози та симптоми (з діапазонами значень), що їм відповідають, види панелей аналізів та значення, тощо.

Таблиця «Cities» (таблиця 2.1) зберігає назву міста.

Таблиця 2.1 – Атрибути таблиці «Cities»

№п/п	Назва атрибуту	Тип даних	Опис
1.	ID	int	Первинний ключ, числовий ідентифікатор
2.	Name	varchar(255)	Назва міста

Таблиця «Polyclinics» (таблиця 2.2) призначена для збереження даних щодо ЛПЗ: назву, вулицю, номер будинку та номер телефону.

Таблиця 2.2 – Атрибути таблиці «Polyclinics»

№п/п	Назва атрибуту	Тип даних	Опис
1.	ID	int	Первинний ключ, числовий ідентифікатор
2.	Name	varchar(255)	Назва поліклініки
3.	FK_street	int	Вторинний ключ-посилання на таблицю «Streets». Адреса поліклініки, вулиця
4.	House	int	Номер будинку поліклініки
5.	Phone	varchar(255)	Номер телефону чергового поліклініки

Таблиця «Streets» (таблиця 2.3) призначена для збереження назв вулиць та міст, у яких вони знаходяться.

Таблиця 2.3 – Атрибути таблиці «Streets»

№п/п	Назва атрибуту	Тип даних	Опис
1.	ID	int	Первинний ключ, числовий ідентифікатор
2.	Name	varchar(255)	Назва вулиці
3.	FK_city	int	Вторинний ключ-посилання на таблицю «Cities» Місто, у якому знаходиться вулиця

Таблиця «Sex» (таблиця 2.4) створена для збереження інформації про статі.

Таблиця 2.4 – Атрибути таблиці «Sex»

№п/п	Назва атрибуту	Тип даних	Опис
1.	ID	int	Первинний ключ, числовий ідентифікатор
2.	Name	varchar(255)	Назва статі

Таблиця «Soc_statuses» (таблиця 2.5) призначена для зберігає назви соціальних статусів: як-от студент, пенсіонер, тощо.

Таблиця 2.5 – Атрибути таблиці «Soc_status»

№п/п	Назва атрибуту	Тип даних	Опис
1.	ID	int	Первинний ключ, числовий ідентифікатор
2.	Name	varchar(255)	Назва соціального статусу

Таблиця «Positions» (таблиця 2.6) призначена для збереження даних про назву посади лікаря.

Таблиця 2.6 – Атрибути таблиці «Positions»

№п/п	Назва атрибуту	Тип даних	Опис
1.	ID	int	Первинний ключ, числовий ідентифікатор
2.	Name	varchar(255)	Назва посади лікаря

Таблиця «Specializations» (таблиця 2.7) призначена для збереження назв спеціалізацій лікарів ЛПЗ.

Таблиця 2.7 – Атрибути таблиці «Specializations»

№п/п	Назва атрибуту	Тип даних	Опис
1.	ID	int	Первинний ключ, числовий ідентифікатор
2.	Name	varchar(255)	Назва спеціалізації лікаря

Таблиця «Statuses» (таблиця 2.8) призначена для збереження інформації про стани візитів.

Таблиця 2.8 – Атрибути таблиці «Statuses»

№п/п	Назва атрибуту	Тип даних	Опис
1.	ID	int	Первинний ключ, числовий ідентифікатор
2.	Name	varchar(255)	Назва статусу візиту

Таблиця «Purposes_of_visits» (таблиця 2.9) зберігає назви цілей візитів.

Таблиця 2.9 – Атрибути таблиці «Purposes_of_visits»

№п/п	Назва атрибуту	Тип даних	Опис
1.	ID	int	Первинний ключ, числовий ідентифікатор
2.	Name	varchar(255)	Назва цілі візиту

Таблиця «Disease» (таблиця 2.10) зберігає інформацію про назви хвороб.

Таблиця 2.10 – Атрибути таблиці «Disease»

№п/п	Назва атрибуту	Тип даних	Опис
1.	ID	int	Первинний ключ, числовий ідентифікатор
2.	Name	varchar(255)	Назва хвороби

Таблиця «Diagnos» (таблиця 2.11) зберігає

Таблиця 2.11 – Атрибути таблиці «Diagnos»

№п/п	Назва атрибуту	Тип даних	Опис
1.	ID	int	Первинний ключ, числовий ідентифікатор.
2.	FK_First_name	varchar(255)	Вторинний ключ-посилання на таблицю «Patients» для співставленням із відповідним прізвищем.
3.	FK_Last_name	varchar(255)	Вторинний ключ-посилання на таблицю «Patients» для співставленням із відповідним ім’ям..
4.	FK_Surname	varchar(255)	Вторинний ключ-посилання на таблицю «Patients» для співставленням із відповідним патронімом.
5.	FK_sex	int	Вторинний ключ-посилання на таблицю «Sex» для співставленням із відповідною статтю.
6.	FK_soc_status	int	Вторинний ключ-посилання на таблицю «Soc_status» для співставленням із відповідним

			соціальним статусом.
7.	FK_polyclinic	int	Вторинний ключ-посилання на таблицю «Polyclinics» для співставленням із відповідною поліклінікою.
8.	FK_street	int	Вторинний ключ-посилання на таблицю «Street» для співставленням із відповідною вулицею, на якій проживає пацієнт.
9.	House	varchar(5)	Номер будинку у якому проживає пацієнт
10.	Apartment	varchar(5)	Номер квартири у якій проживає пацієнт
11.	Date_of_birth	DATE	Дата народження пацієнта
12.	Phone	varchar(255)	Номер телефону пацієнта
13.	Date_of_death	DATE	Дата смерті пацієнта (якщо вказана)

Таблиця «Offices» (таблиця 2.12) зберігає дані про кабінети поліклініки.

Таблиця 2.12 – Атрибути таблиці «Offices»

№п/п	**Назва атрибуту**	**Тип даних**	**Опис**
1.	ID	int	Первинний ключ, числовий ідентифікатор.
2.	Number	varchar(255)	Номер кабінету
3.	Name	varchar(255)	Назва кабінету
4.	FK_ polyclinic	int	Вторинний ключ-посилання на таблицю «Polyclinics» для співставленням із відповідною поліклінікою

Таблиця «Patients» (таблиця 2.13) зберігає дані про пацієнтів.

Таблиця 2.13 – Атрибути таблиці «Patients»

№п/п	**Назва атрибуту**	**Тип даних**	**Опис**
1.	ID	int	Первинний ключ, числовий ідентифікатор.
2.	FK_First_name	int	Вторинний ключ-посилання на таблицю «Patients» для співставленням із відповідним прізвищем.
3.	FK_Last_name	int	Вторинний ключ-посилання на

			таблицю «Patients» для співставленням із відповідним ім’ям..
4.	FK_Surname	int	Вторинний ключ-посилання на таблицю «Patients» для співставленням із відповідним патронімом.
5.	FK_sex	int	Вторинний ключ-посилання на таблицю «Sex» для співставленням із відповідною статтю.
6.	FK_soc_status	int	Вторинний ключ-посилання на таблицю «Soc_status» для співставленням із відповідним соціальним статусом.
7.	FK_polyclinic	int	Вторинний ключ-посилання на таблицю «Polyclinics» для співставленням із відповідною поліклінікою.
8.	FK_street	int	Вторинний ключ-посилання на таблицю «Street» для співставленням із відповідною вулицею, на якій проживає пацієнт.
9.	House	varchar(5)	Номер будинку у якому проживає пацієнт
10.	Apartment	varchar(5)	Номер квартири у якій проживає пацієнт
11.	Date_of_birth	DATE	Дата народження пацієнта
12.	Phone	varchar(255)	Номер телефону пацієнта
13.	Date_of_death	DATE	Дата смерті пацієнта (якщо вказана)

Таблиця «Doctors» (таблиця 2.14) призначена для збереження інформації про лікарів.

Таблиця 2.14 – Атрибути таблиці «Doctors»

№п/п	**Назва атрибуту**	**Тип даних**	**Опис**
1.	ID	int	Первинний ключ, числовий ідентифікатор.
2.	Last_name	varchar(255)	Прізвище лікаря
3.	First_name	varchar(255)	Ім’я лікаря
4.	Surname	varchar(255)	По батькові лікаря
5.	Date_of_birth	DATE	Дата народження лікаря

6.	FK_sex	int	Вторинний ключ-посилання на таблицю «Sex» для співставленням із відповідною статтю.
7.	FK_specialization	int	Вторинний ключ-посилання на таблицю «Spetializations» для співставленням із відповідною спеціалізацією.
8.	FK_position	int	Вторинний ключ-посилання на таблицю «Position» для співставленням із відповідною посадою лікаря.
9.	Work_with_children	tinyint(1)	Чи працює лікар з дітьми та підлітками
10.	Work_with_retired	tinyint(1)	Чи працює лікар з пацієнтами похилого віку
11.	Hire_date	DATE	Дата найму лікаря на роботу
12.	Fire_date	DATE	Дата звільнення лікаря (якщо існує)
13.	Experience	int(3)	Попередній стаж лікаря
14.	Phone	varchar(255)	Контактний телефон лікаря

Таблиця «Visits» (таблиця 2.15) призначена для збереження даних про візит пацієнта до терапевта.

Таблиця 2.15 – Атрибути таблиці «Visits»

№п/п	**Назва атрибуту**	**Тип даних**	**Опис**
1.	ID	int	Первинний ключ, числовий ідентифікатор.
2.	Date_of_beginnning	datetime	Дата й час початку прийому
3.	FK_purpose	int	Вторинний ключ-посилання на таблицю «Purpose» для співставленням із відповідною статтю.
4.	FK_doctor	int	Вторинний ключ-посилання на таблицю «Doctors» для співставленням із відповідним лікарем
5.	FK_office	int	Вторинний ключ-посилання на таблицю «Offises» для співставленням із відповідним кабінетом
6.	Date_of_ending	datetime	Дата й час завершення прийому
7.	FK_status	int	Вторинний ключ-посилання на таблицю «Status» для

			співставленням із відповідним статусом прийому
8.	FK_patient	int	Вторинний ключ-посилання на таблицю «Patients» для співставленням із відповідним пацієнтом
9.	Comment	text	Коментар лікаря щодо прийому

Таблиця «Declarations» (таблиця 2.16) призначена для збереження даних про декларації, укладені між пацієнтом та лікарем-терапевтом.

Таблиця 2.16 – Атрибути таблиці «Declarations»

№п/п	Назва атрибуту	Тип даних	Опис
1.	ID	int	Первинний ключ, числовий ідентифікатор.
2.	FK_patient	int	Вторинний ключ-посилання на таблицю «Patients» для співставленням із відповідним пацієнтом
3.	FK_doctor	int	Вторинний ключ-посилання на таблицю «Doctors» для співставленням із відповідним лікарем
4.	Date_of_concluding	DATE	Дата укладання договору
5.	Date_of_termination	DATE	Дата закінчення договору
6.	Date_of_extention	DATE	Дата продовження договору

Таблиця «Blood_types» (таблиця 2.17) призначена для збереження даних про групи крові.

Таблиця 2.17 – Атрибути таблиці «Blood_types»

№п/п	Назва атрибуту	Тип даних	Опис
1.	ID	int	Первинний ключ, числовий ідентифікатор.
2.	Name	varchar(255)	Назва групи крові та резус-фактор

Таблиця «Illness» (таблиця 2.18) містить назви хвороб.

Таблиця 2.18 – Атрибути таблиці «Illness»

№п/п	Назва атрибуту	Тип даних	Опис
1.	ID	int	Первинний ключ, числовий ідентифікатор.
2.	Name	varchar(255)	Назва хвороби
3.	Value_up	float	Верхнє значення вираженості симптому
4.	Value_down	float	Нижнє значення вираженості симптому

Таблиця «Sympthoms» (таблиця 2.19) містить інформацію про симптоми.

Таблиця 2.19 – Атрибути таблиці «Sympthoms»

№п/п	Назва атрибуту	Тип даних	Опис
1.	ID	int	Первинний ключ, числовий ідентифікатор.
2.	Name	varchar(255)	Назва симптому

Таблиця «Diagnoses» (таблиця 2.20) призначена для збереження даних щодо діагнозів пацієнтів.

Таблиця 2.20 – Атрибути таблиці «Diagnoses»

№п/п	Назва атрибуту	Тип даних	Опис
1.	ID	int	Первинний ключ, числовий ідентифікатор.
2.	Name	varchar(255)	Назва симптому
3.	FK_Sympthoms	int	Вторинний ключ-посилання на таблицю «Sympthoms» для співставленням із відповідним симптомом
4.	SympthomsSeverity	float	Вираженість кожного симптому при певній хворобі
5.	FK_Illnes	int	Вторинний ключ-посилання на таблицю «Illness» для співставленням із відповідною хворобою

Структура бази даних наведена на рисунку 2.4

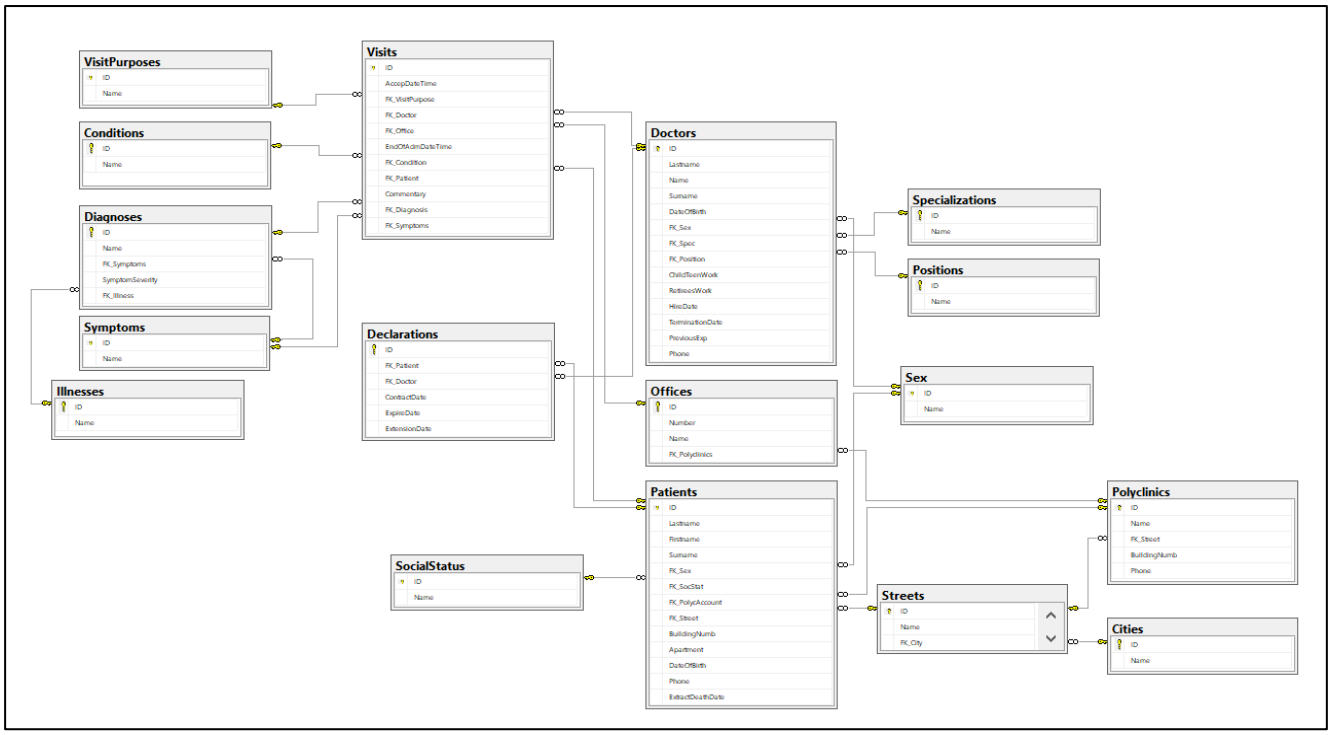


Рисунок 2.4 – Структура бази даних методу автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання.

Таким чином, в базі даних зберігається вся інформація, яка потрібна для роботи системи методу автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання.

2.3 Вибір засобів розробки інформаційної системи

Задля створення методу автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання. Та програмного застосунку на його основі необхідно обрати інструменти та засоби, що забезпечать зручне проєктування системи та її надійну роботу в побільшому. Необхідно визначити, якою мовою програмування буде написаний застосунок, СКБД, фреймворк та редактор програмного коду.

2.3.1 Вибір мови програмування

Було проведено порівняння двох мов програмування: C# та Java. Основні відмінності між цими мовами програмування:

1. Середовище виконання: Java [25] працює на JRE (Java Runtime Environment), тоді як C# працює на CLR (Common Language Runtime).

2. Парадигма програмування: Java – суворо об'єктно-орієнтована мова, у той час як C# – об'єктно-орієнтована, функціональна, сильно типізована і компонентно-орієнтована.

3. Перевантаження операторів: Java не підтримує навантаження операторів, у той час як C# [26] підтримує навантаження кількох операторів.

4. Вказівники: Java не підтримують вказівники, тоді як C# підтримує вказівники лише в небезпечному режимі.

5. Масиви: Масиви Java є спеціалізацією Object, тоді як масиви C# є спеціалізацією System.

Переваги C# [27]:

– C# забезпечує підтримку лямбд та дженериків;

– інтегровані в мову запити (LINQ);

– безпечні методи розширення;

– властивості з методами GET/SET;

– управління пам'яттю;

– найкраща у своєму класі кросплатформна підтримка;

– зворотна сумісність.

Переваги Java [28]:

– JIT-компілятор робить програму повільною;

– Java пред'являє високі вимоги до пам'яті та обробки даних;

– мова не підтримує низько рівневі конструкції програмування, такі як вказівники;

– користувач не може контролювати складання сміття, оскільки Java не надає таких функцій, як delete() або free().

Зважаючи на особливість предметної області та поставлену задачу, було обрано мову програмування C#.

2.3.2 Вибір фреймворку

Проведено аналіз .NET Framework та.NET Core. Виокремлено наступні переваги та недоліки кожного з фреймворків. Варто віддати перевагу .NET Core [29], якщо:

– проект потребує крос-платформної інтеграції;

– проект потребує розробки мікросервісів;

– проект залежить від cli (інтерфейс командного рядка), оскільки .NET Core підходить для cli;

.NET Framework варто обрати, якщо:

– програми вже працюють на .NET Framework;

– програми потрібують такі технології як: workflow, webforms або wcf, які відсутні в .NET Core;

– програми створені лише під Windows.

Зважаючи на вищеописані переваги й недоліки, доцільно обрати .NET Framework для створення методу автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання.

2.3.3 Вибір редактору програмного коду

Для вибору редактору програмного коду було проведено аналіз між Visual Studio та Eclipse.

Обидві IDE [30] (інтегровані середовища розробки) добре опрацьовані та мають своїх шанувальників. Проте вони значно відрізняються один від одного і мають деякі функції, які є взаємовиключними.

Обидва середовища пропонують безліч функцій, які гарантують їх зручність у використанні, Perspective - одна з таких функцій організації в Eclipse, що складається з набору вікон та налаштувань макета. IDE також дозволяє переключатися між перспективами під час роботи над різними аспектами

проекту. Visual Studio, однак, відстає за кількістю таких функцій. Однак VS дозволяє імпортувати/експортувати установки. Крім того, в ній є сторонній плагін, який за своїми можливостями певною мірою нагадує Perspective.

Visual Studio [31] допомагає перевизначити точку виконання. Це корисно в тих випадках, коли було написано код, який перестав працювати так, як було очікувано, і тому є необхідність повернутися до точки виконання. Eclipse відстає у цій функції, і найкраще, що він може зробити у цьому контексті, то це дозволить повернутися до вершини поточного методу (кадру стека). Крім того, Visual Studio дозволяє працювати у двох різних екземплярах IDE одночасно, а також дозволяє налагоджувати певний сервіс та його клієнтів одночасно.

Проаналізувавши переваги та недоліки редакторів коду було обрано MS Visual Studio 2019.

2.3.4 Вибір СКБД

SQL [32] – це абревіатура мови структурованих запитів. Вона використовується для доступу, маніпулювання та вилучення інформації з бази даних.

MySQL [33] – це система управління реляційними базами даних (RDBMS) з відкритим вихідним кодом, що базується на мові структурованих запитів (SQL). Вона працює на таких платформах, як Linux, UNIX та Windows.

SQL Server [34] належить та розроблений корпорацією Microsoft. Основною функцією SQL Server є зберігання та доступ до даних у міру їхньої затребуваності іншими програмами, незалежно від того, чи працюють вони на інших комп'ютерах, підключених до мережі, або на комп'ютері, на якому знаходиться сервер.

В таблиці 2.21 наведено детальне порівняння [35] цих СКБД.

Таблиця 2.21 – Порівняння MS SQL Server та MySQL

№п/п	MS SQL Server	MySQL

1.	Підтримує такі мови програмування як: C++, JAVA, Ruby, Visual Basic, Delphi, R і т.д.	MySQL пропонує розширену підтримку таких мов як Perl, Tcl, Haskey і т.д.
2.	Потребує великий обсяг оперативного простору для зберігання даних.	Потребує менший обсяг оперативного простору для зберігання даних.
3.	Дозволяє зупиняти виконання запитів.	Не дозволяє скасувати запит на середині процесу.
4.	Не блокує базу даних під час резервного копіювання даних.	Блокує базу даних під час резервного копіювання даних.
5.	Потужні версії не є безкоштовними.	З відкритим кодом. Доступний безкоштовно.
6.	Має високий рівень захисту та не допускає жодних маніпуляцій із файлами бази даних під час роботи.	Дозволяє маніпулювати файлами бази даних під час роботи.
7.	Доступний у кількох редакціях, таких як Enterprise, Standard, Web, Workgroup або Express.	Доступний у MySQL Standard Edition, MySQL Enterprise Edition та MySQL Cluster Grade Edition.

Зважаючи на особливості кожної із СКБД та особливості реалізації програмного застосунку було обрано MS SQL Server.

Підсумовуючи, для розробки інформаційної технології методу автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання було обрано платформу .NET, мову програмування C#, СКБД Microsoft SQL Server та середовище розробки Visual Studio 2017.

Розділ 3 Програмна реалізація інформаційної системи

3.1 Структура та функціональне призначення програмних складових системи

Для реалізації програмного застосунку було створено діаграму класів, відповідно до поставлених задач (рисунок 3.1).

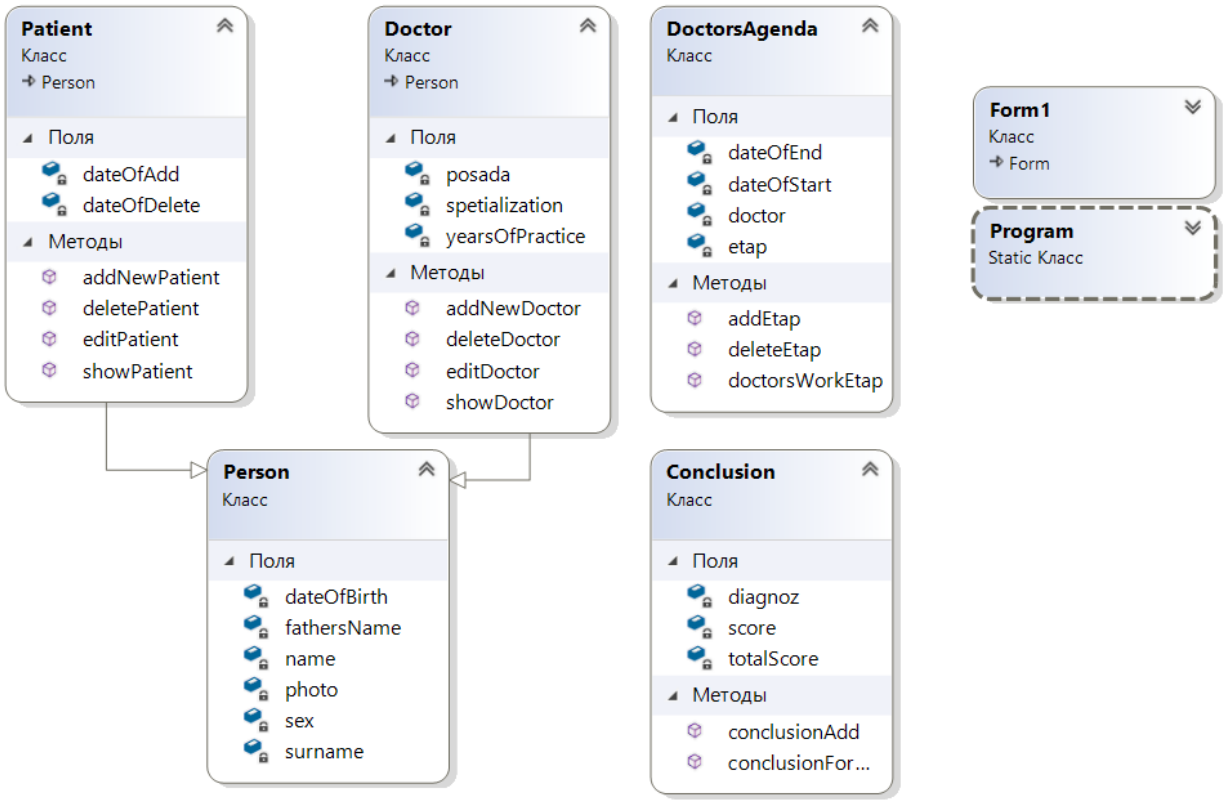


Рисунок 3.1 – Діаграма класів методу автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання.

Основним класом розробленої інформаційної системи є клас Form1, він відповідає за взаємодію з користувачем, містить користувацький інтерфейс та агрегує в собі об’єкти всіх інших класів діаграми для заповнення даними таблиць.

Клас Person є базовим класом, який містить набір властивостей, притаманний і лікарям і пацієнтам. Від нього успадковуються класи Patient та Doctor.

У класі Patient реалізована логіка пов’язана з пацієнтом. Клас містить конструктор класу та методи додавання нового лікаря (newPatient), видалення (deletePatient) та оновлення даних про пацієнта (changePatient).

У класі Doctor реалізована логіка пов’язана з лікарем. Клас містить конструктор класу та методи додавання нового лікаря (addNewDoctor), видалення лікаря (deleteDoctor) та оновлення даних про лікаря (changeDoctor). Також є функція пошуку лікаря за видом діяльності (showDoctor). Також з діяльністю лікаря є клас із назвою

DoctorsAgenda. У цьому класі зберігається інформація про етапи роботи лікаря та візити. Кожен етап має дату та час початку та дату і час завершення. Також при створенні об’єкту класу DoctorsWorksEtap, на вхід параметром подається екземпляр класу Doctor. У класі DoctorsAgenda реалізовано методи додавання та видалення робочого етапу (addWorksEtap та deleteWorksEtap відповідно).

Клас Conclusion є по суті вхідним параметром класу для заповнення інформації. Саме в ньому реалізовано методи й поля, що необхідні для створення експертного висновку та роботи ЕС в цілому.

Програмне застосування (рисунок 3.2) розділене на декілька модулів:

- робота із симптомами;
- робота з діагнозами;
- пацієнти ;
- візити ;
- декларації;
- формування експертного висновку.

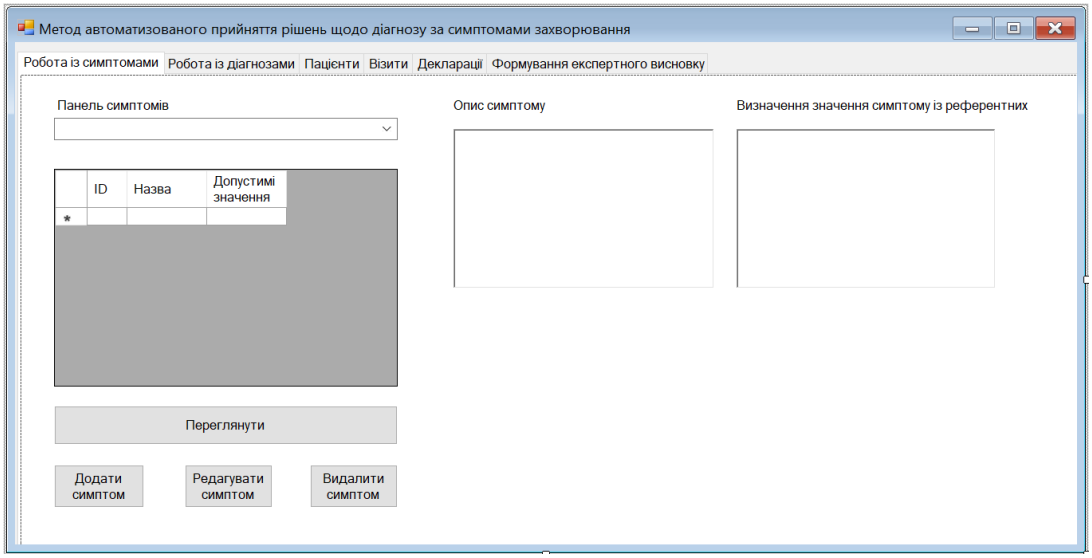


Рисунок 3.2 – Вигляд програмного продукту

Перший модуль «Робота із симптомами» забезпечує можливість користувачеві вводити нові симптоми, редагувати та видаляти існуючі. Усі симптоми класифіковано та внесено до окремих панелей.

Другий модуль «Робота з діагнозами» (рисунок 3.3) схожий із першим модулем: реалізовано функціонал, що дозволяє вносити нові діагнози, їх пояснення та симптоми, що їх характеризують.

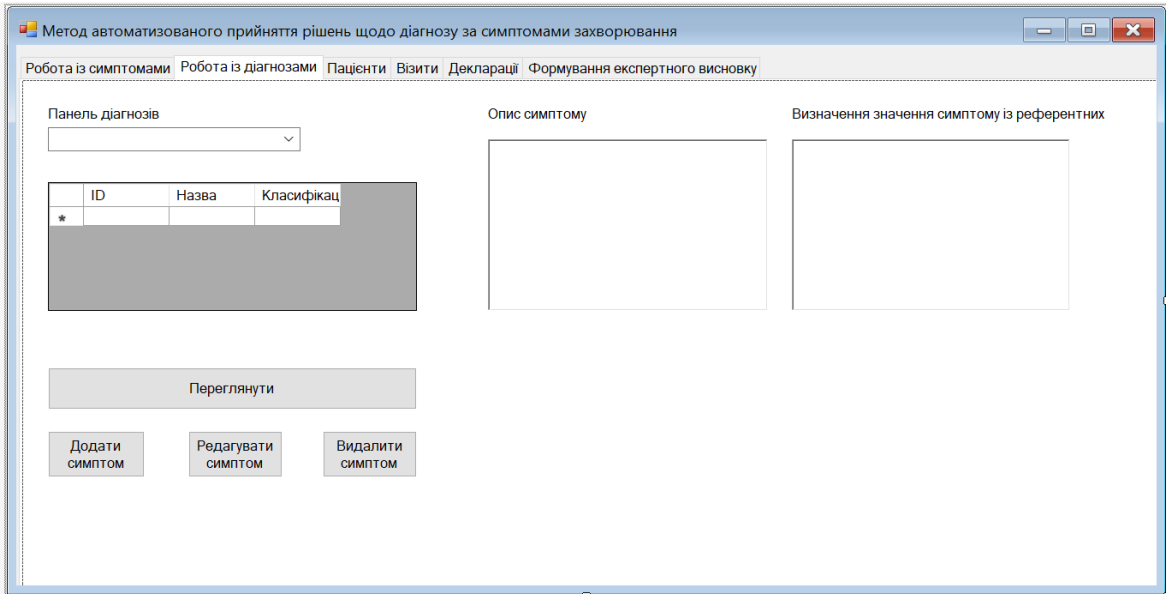


Рисунок 3.3 – Модуль «Робота з діагнозами»

Модулі «Пацієнти», «Візити», «Декларації» є допоміжними та виводять інформацію з БД, дозволяючи її переглядати, редагувати та видаляти.

Головний модуль «Формування експертного висновку» (рисунок 3.4) реалізовано задля аналізу симптомів, що ввів користувач та формування висновку щодо діагнозу.

Користувач обирає панель симптомів, обирає симптоми, що виведені в таблицю та вводить значення їх вираженості. На основі введених даних роботу розпочинає метод ЕС.

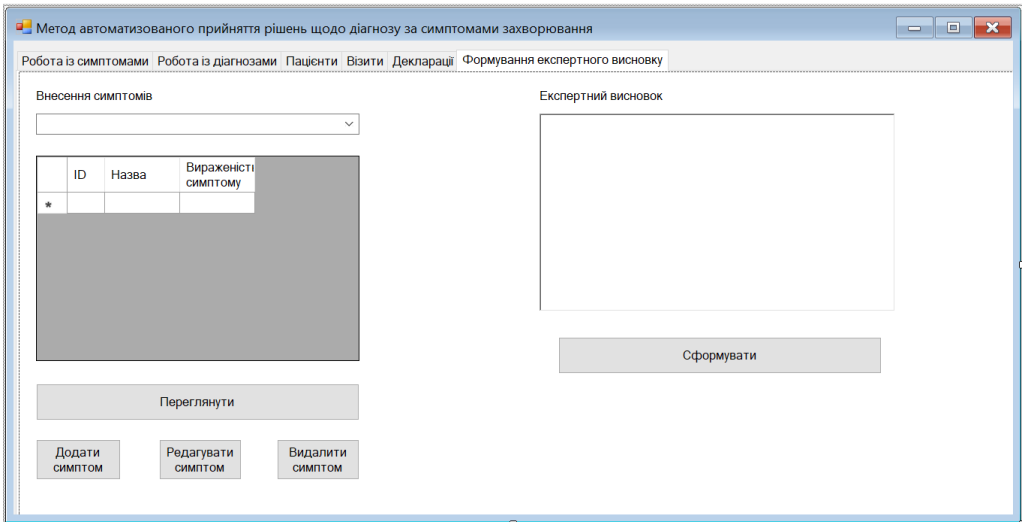


Рисунок 3.4 – Модуль «Формування експертного висновку»

Таким чином, дана схема повністю відповідає за функціональність методу автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання.

3.2 Особливості реалізації програмних складових системи

Перший модуль, із яким працюватиме користувач інформаційної системи – «Робота з симптомами». На ній лікар може виконати наступні дії:

– переглянути усі наявні панелі симптомів;

– переглянути симптоми з панелей;

– додати новий симптом, відредагувати або видалити існуючий;

– переглянути інформацію по кожному симптому.

На даному модулі розміщені елементи comboBox та dataGridView: перший елемент відповідатиме за вибір панелі, а таблиця відображатиме перелік симптомів відповідно до обраного розділу. Завантаження симптомів відбувається за допомогою методу, програмний код якого наведено нижче.

```
public void loadSymthoms()
        {
            string connectString = "Server=.\\SQLEXPRESS;Initial
Catalog=doctor;Integrated Security=true;";
            SqlConnection myConnection = new SqlConnection(connectString);
            myConnection.Open();

            string query1 = "SELECT * FROM Symthoms ORDER BY ID";
            SqlCommand command = new SqlCommand(query1, myConnection);

            SqlDataReader reader1 = command.ExecuteReader();

            List<string[]> dataa = new List<string[]>();

            while (reader1.Read())
            {
                dataa.Add(new string[3]);
                dataa[dataa.Count - 1][0] = reader1[0].ToString();
                dataa[dataa.Count - 1][1] = reader1[1].ToString();
                dataa[dataa.Count - 1][2] = reader1[2].ToString();
                    }
            reader1.Close();
            myConnection.Close();

            foreach (string[] s in dataa)
                dataGridView1.Rows.Add(s);
        }
```

Результат виконання програмного коду наведено на рисунку 3.5.

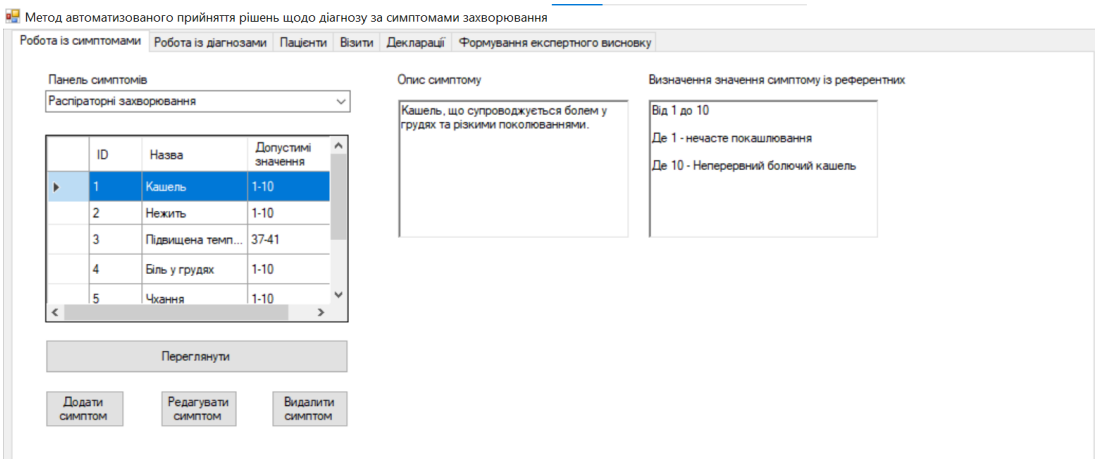


Рисунок 3.5 – Результат виконання програмного коду

Таким чином, модуль забезпечує лікарю роботу із симптомами та їх значенням для подальшої роботи експертної системи.

Наступний модуль призначений для роботи з діагнозами, працює сходим чином із попереднім. Має наступні можливості:

– переглянути усі наявні панелі діагнозів;

– переглянути діагнози з панелей;

– додати новий діагноз, відредагувати або видалити існуючий;

– переглянути інформацію по кожному діагнозу.

Лістинг методу завантаження діагнозів на форму наведено нижче:

```
if (comboBox1.SelectedIndex >= 0)
            {
                string connectString = "Server=.\\SQLEXPRESS;Initial
Catalog=doctor;Integrated Security=true;";
                SqlConnection myConnection = new SqlConnection(connectString);
                myConnection.Open();

                string query1 = "SELECT * FROM Diagnoses ORDER BY ID";
                SqlCommand command = new SqlCommand(query1, myConnection);

                SqlDataReader reader1 = command.ExecuteReader();

                List<string[]> dataa = new List<string[]>();

                while (reader1.Read())
                {
                    dataa.Add(new string[3]);
                    dataa[dataa.Count - 1][0] = reader1[0].ToString();
                    dataa[dataa.Count - 1][1] = reader1[1].ToString();
                    dataa[dataa.Count - 1][2] = reader1[2].ToString();
                }
                reader1.Close();
                myConnection.Close();

                foreach (string[] s in dataa)
                    dataGridView3.Rows.Add(s);
            }
```

На рисунку 3.6 наведено результат виконання лістингу, наведеного нижче.

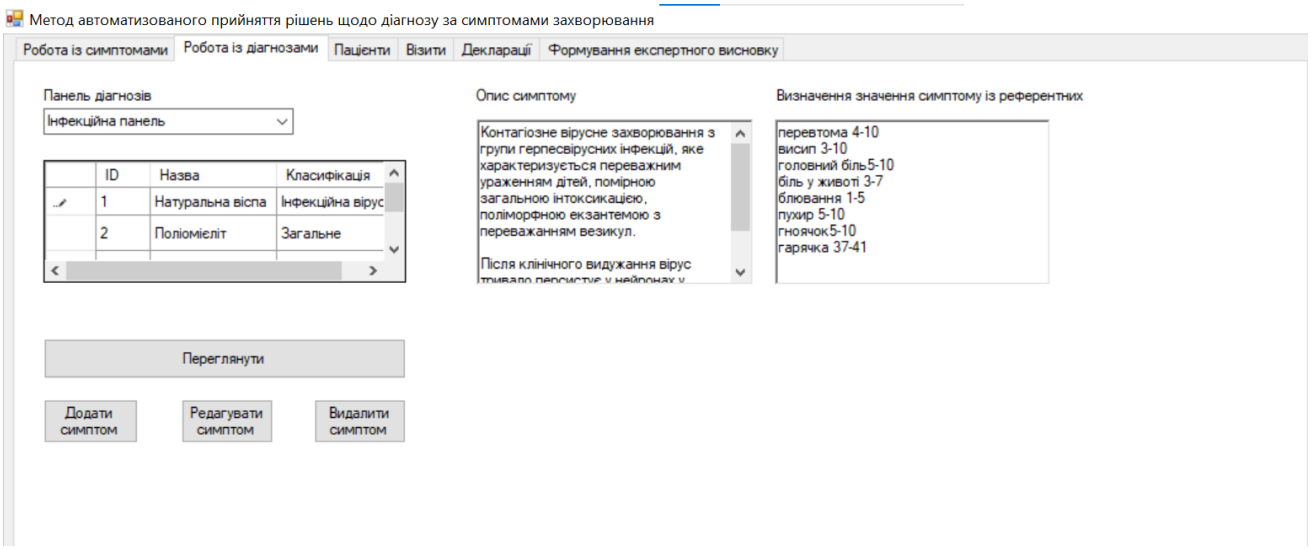


Рисунок 3.6 – Результат роботи програмного коду

Наступний модуль «Пацієнти» призначений для роботи лікар із базою пацієнтів. Зокрема, забезпечуються наступні функції:

– перегляд даних щодо пацієнтів;

– редагування записів таблиці;

– видалення інформації про пацієнта.

Дані завантажуються в таблицю схожим чином, як і в методах, наведених вище, однак розглянемо принцип виведення певного запису до окремих textbox-ів. Лістинг наведено нижче:

```
private void dataGridPatients_CellContentClick(object sender, DataGridViewCellEventArgs
e)
        {
            textBoxName.Text = dataGridView2.CurrentRow.Cells[1].Value.ToString();
            textBoxFirsnN.Text = dataGridView2.CurrentRow.Cells[2].Value.ToString();
            textBoxPatronim = dataGridView2.CurrentRow.Cells[3].Value.ToString();
            textBoxDeclaration = dataGridView2.CurrentRow.Cells[4].Value.ToString();
            textBoxVisit = dataGridView2.CurrentRow.Cells[5].Value.ToString();
        }
```

Результат виконання програмного коду наведено на рисунку3.7.

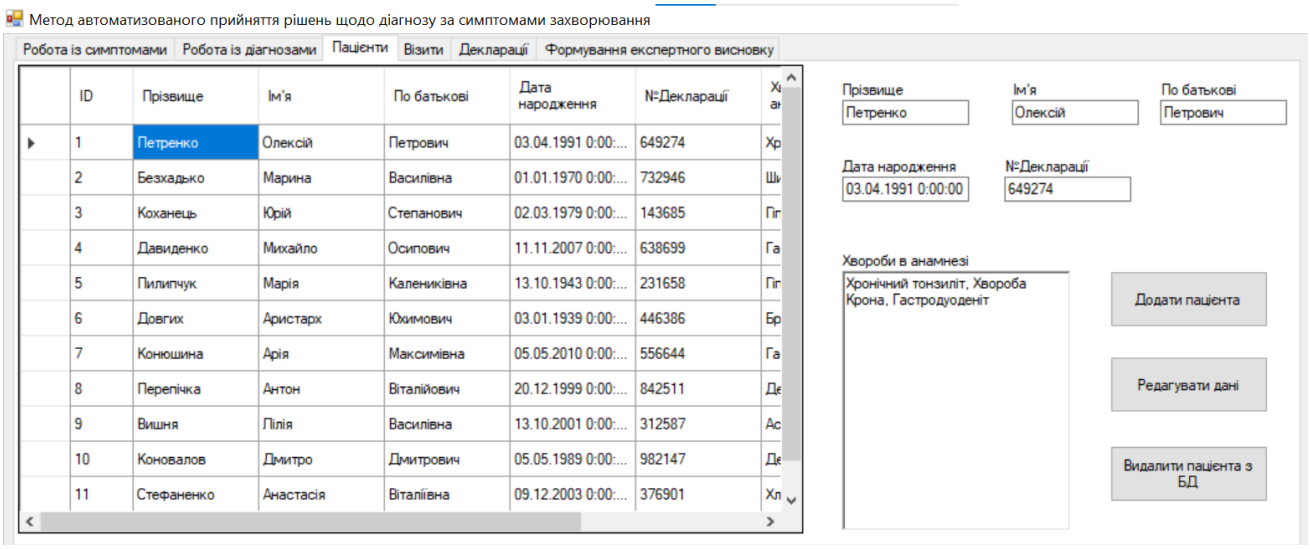


Рисунок 3.7 – Результат виконання програмного коду

Програмні модулі «Візити» та «Декларації» працюють за аналогічними методами.

Було реалізовано головний модуль інформаційної системи – «Формування експертного висновку». Тут користувач має можливість ввести всі необхідні симптоми для подальшого аналізу та отримати згенерований експертний висновок.

Користувачеві необхідно обрати панель, до якої належить симптом, або ввести його назву в пошукове поле вище. Після цього з’явиться відповідна до панелі таблиця із симптомами та полями для введення їх вираженості.

Метод, що повертає значення оцінки приналежності множини симптомів до певного діагнозу наведено нижче:

```
class Conclusion : DbContext
{
    public ConclusionDbContext()

 {
  public DbSet<Disease> Diseases { get; set; }
    public DbSet<DiseaseCategory> DiseasesCategories { get; set; }
       public DbSet<Plant> Patientss { get; set; }
    public DbSet<Specie> Species { get; set; }
    public DbSet<Symptom> Symptoms { get; set; }
    public DbSet<DiseaseSymptom> DiseaseSymptoms { get; set; }
    public DbSet<History> Histories { get; set; }
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<DiseaseSymptom>()
         HasKey(bc => new { bc.DiseaseId, bc.SymptomId });
```

```
modelBuilder.Entity<DiseaseSymptom>()
        .HasOne(bc => bc.Disease)
        .WithMany(b => b.DiseaseSymptoms)
        .HasForeignKey(bc => bc.DiseaseId);
        modelBuilder.Entity<DiseaseSymptom>()
        .HasOne(bc => bc.Symptom)
        .WithMany(c => c.DiseaseSymptoms)
        .HasForeignKey(bc => bc.SymptomId);
    }
    protected override void OnConfiguring(DbContextOptionsBuilder
   optionsBuilder)
    {
        base.OnConfiguring(optionsBuilder);
        optionsBuilder.UseSqlite(@$"FileName={Path.GetDirectoryName(Assembly.GetEx
        ecutingAssembly().Location)}\doctor.db");
 }
 }
        }
```

Результат виконання програмного коду наведено на рисунку 3.8.

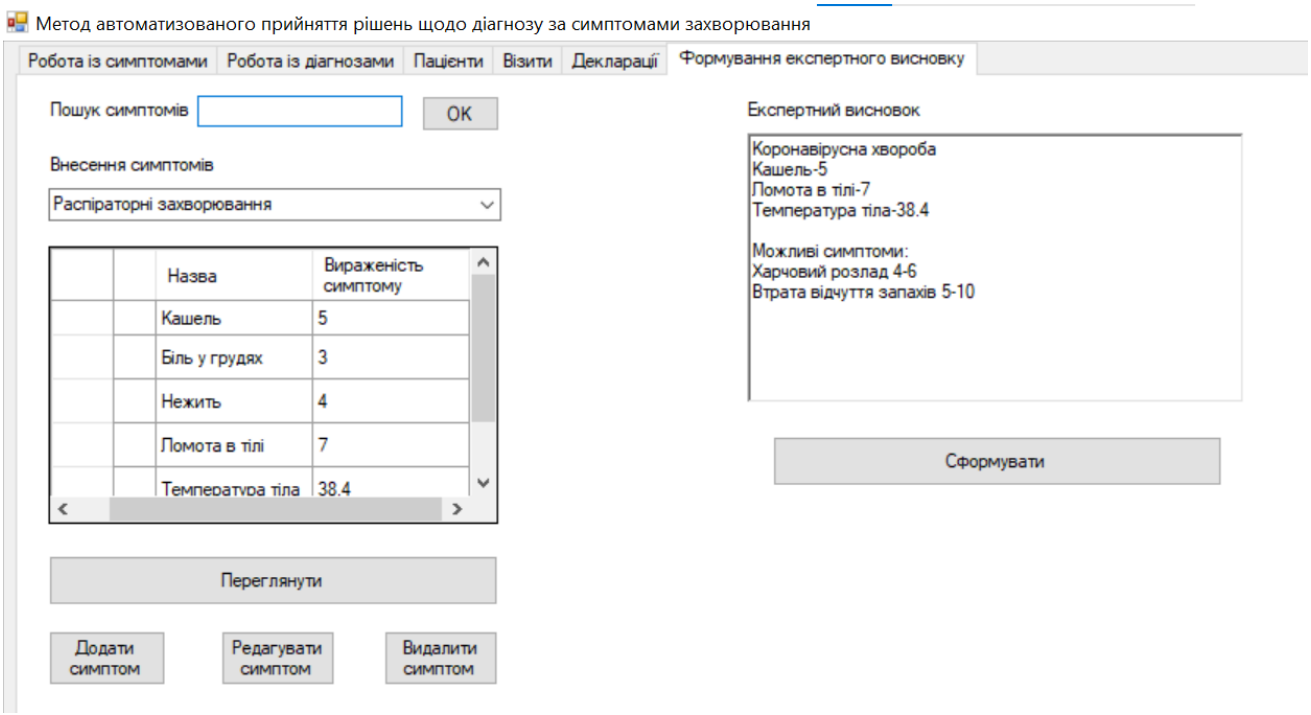


Рисунок 3.8 – Результат виконання програмного коду

Таким чином на основі методу автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання було реалізовано програмний застосунок, що містить всі зазначені в розділі 1.5 функції, має простий інтерфейс та високу швидкодію роботи.

3.3 Тестування інформаційної системи

Для верифікації програмного продукту необхідно провести ряд тестувань. Дослідження містить тест-кейси та модульне тестування інтерфейсу користувача. Кожен із дослідів перевіряє правильність роботи у тих місцях, де помилка найбільш вірогідна: некоректне введення даних в продукційні відомості, недотримана послідовність виконання етапів для визначення діагнозу – одні з таких точок, котрі необхідно найбільш ретельно протестувати.

Перший тест-кейс (таблиця 3.1) створено для перевірки роботи попередження щодо некоректно завантажених даних щодо пацієнтів. Вхідними даними стане рядок із записом про пацієнта, що не містить певних полів (№ декларації, прізвище, ім’я).

Таблиця 3.1 – Тест-кейс AT001

Тест-кейс ID: AT001	**Пріоритет: 2**	**Створено:** 27.04.2022, Домбровський Н.С.
Назва: Перевірка роботи методу попередження користувача про пошкоджені записи таблиці «Patients» **Вхідні дані:** Таблиця «Patients», запис з id = 10 (рядок 11)		
Кроки		**Очікуваний результат**
Передумова: на вкладці «Пацієнти» користувач обирає рядок 12, в якому пошкоджено деякі поля. 1. Запустити додаток та обрати вкладку «Пацієнти» 2. Обрати рядок повідомлення із відсутніми записами в БД (рядок 11) 3. Натиснути на будь-яке поле цього запису 4. Порівняти фактичний результат з очікуваним 5. Закрити діалог із попередженням про помилку 6. Обрати рядок 1, де всі поля запису вірно заповнено 7. Натиснути на будь-яке поле цього запису 8. Порівняти фактичний результат з очікуваним		Поява діалогового вікна із попередженням про помилку з наступним текстом: «Пошкоджено запис бази даних!» Користувач обрав рядок, всі дані виведені до окремих полів на формі.

Результат виконання тест-кейсу: пройдено успішно

Результат виконання тест-кейсу наведено на рисунку 3.9.

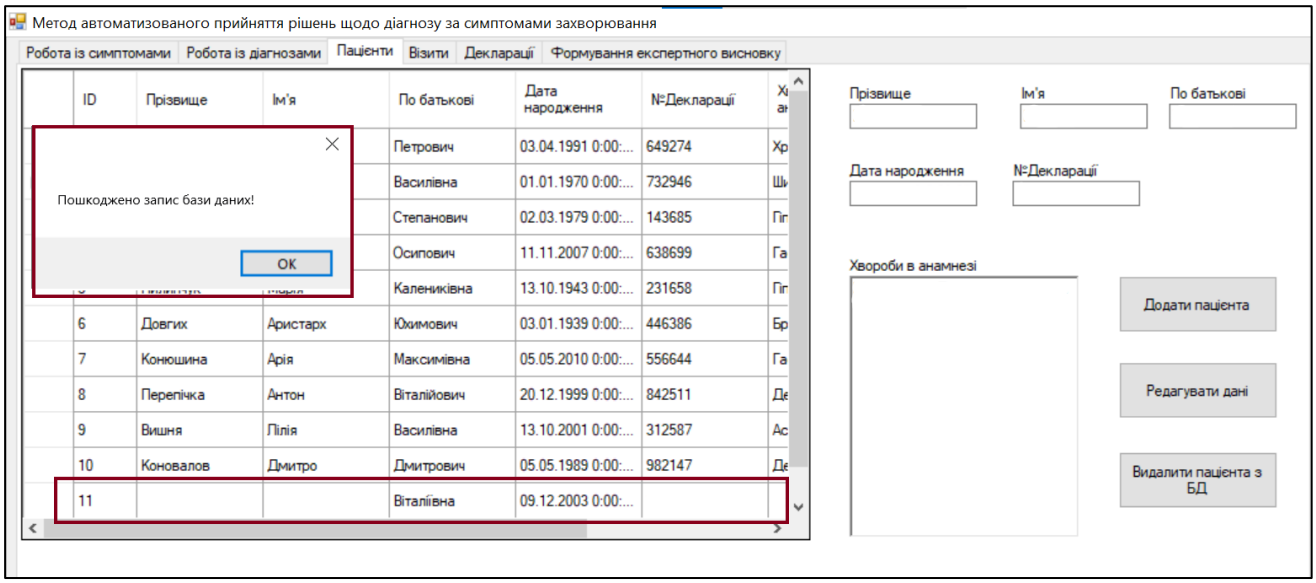


Рисунок 3.9 – Результат виконання тест-кейсу AT0001

Також необхідно перевірити правильність роботи методів, що забезпечують роботу із симптомами та діагнозами, а саме можливість перегляду, внесення, редагування та видалення симптомів та діагнозів. Було створено тест-кейс (таблиця 3.2) для перевірки методу внесення нового симптому.

Таблиця 3.2 – Тест-кейс AT002

Тест-кейс ID: AT002	**Пріоритет: 2**	**Створено:** 27.04.2022, Домбровський Н.С.
Назва: Перевірка роботи методу внесення нового симптому до бази даних з форми застосунку, вкладка «Робота із симптомами» **Вхідні дані:** введені в форму симптоми		
Кроки		**Очікуваний результат**
Передумова: на вкладці «Робота із симптомами» користувач обирає форму внесення нового симптому 1. Користувач обирає вкладку «Робота із симптомами» 2. Натискає кнопку «Додати симптом» 3. На новій формі, призначені для		Після внесення інформації щодо симптому, дані відображаються на формі «Робота із симптомами» в списку відповідної панелі симптомів

роботи із внесенням симптомів користувач вносить необхідні дані: панель симптомів, назву та референтні значення цього симптому 4. Натискає кнопку «Додати симптом» 5. Порівняти фактичний результат з очікуваним	Після внесення інформації щодо симптому, дані відображаються на формі «Робота із симптомами» в списку відповідної панелі симптомів
Результат виконання тест-кейсу: пройдено успішно	

Результат виконання тест-кейсу наведено на рисунку 3.10.

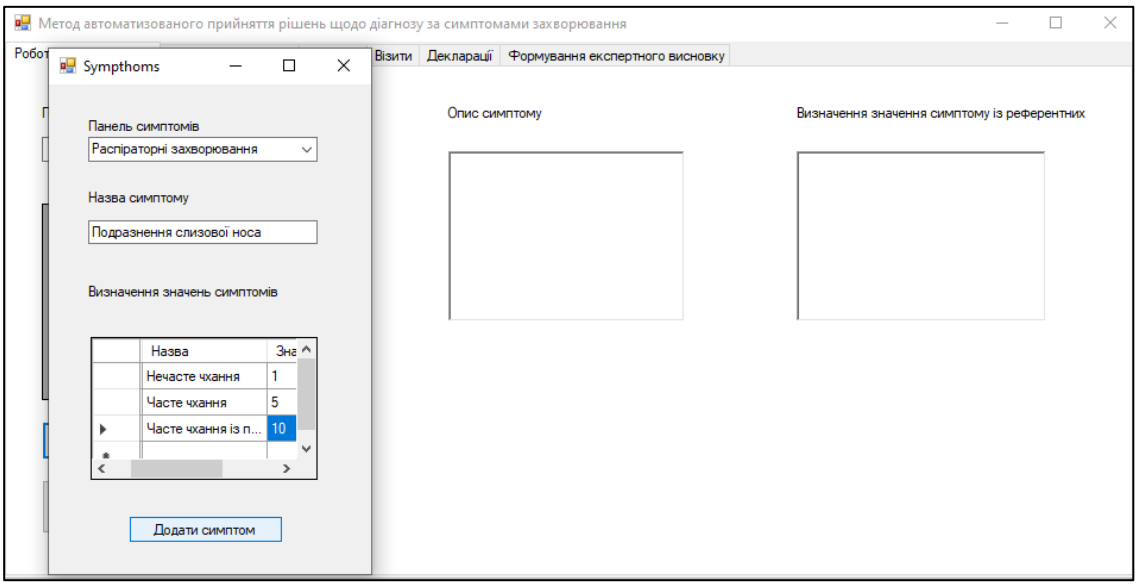


Рисунок 3.10 – Результат виконання тест-кейсу

Також необхідно переконатись, що метод автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання працює вірно. Для цього було реалізовано тест-кейс (таблиця 3.3) із перевіркою експертного висновку за вказаними симптомами. Було розглянуто хворобу Крона із наступними супровідними симптомами:

– Біль в животі – 7;
– Дисфрагія – 8;
– Підвищена температура тіла – 38.1;
– Втрата маси тіла – 3;
– Загальна слабкість – 7;

Таблиця 3.3 – Тест-кейс AT003

Тест-кейс ID: AT003	**Пріоритет: 1**	**Створено:** 27.04.2022, Домбровський Н.С.
Назва: Перевірка роботи методу визначення діагнозу за даними симптомів **Вхідні дані:** введені в форму симптоми		
Кроки		**Очікуваний результат**
Передумова: на вкладці «Формування експертного висновку» користувач вводить необхідні симптоми та отримує експертний висновок щодо діагнозу 1. Натискає кнопку «Додати симптом» 2. На новій формі, призначені для роботи із внесенням симптомів користувач вносить необхідні дані: панель симптомів, назву та референтні значення цього симптому 3. Натискає кнопку «Додати симптом» 4. Порівняти фактичний результат з очікуваним		Після внесення симптомів до відомості, користувач отримує експертний висновок із вірогідним захворюванням хвороба Крона. Після внесення симптомів до відомості, користувач отримує експертний висновок із вірогідним захворюванням хвороба Крона.
Результат виконання тест-кейсу: пройдено успішно		

Результат виконання тест-кейсу наведено на рисунку 3.11.

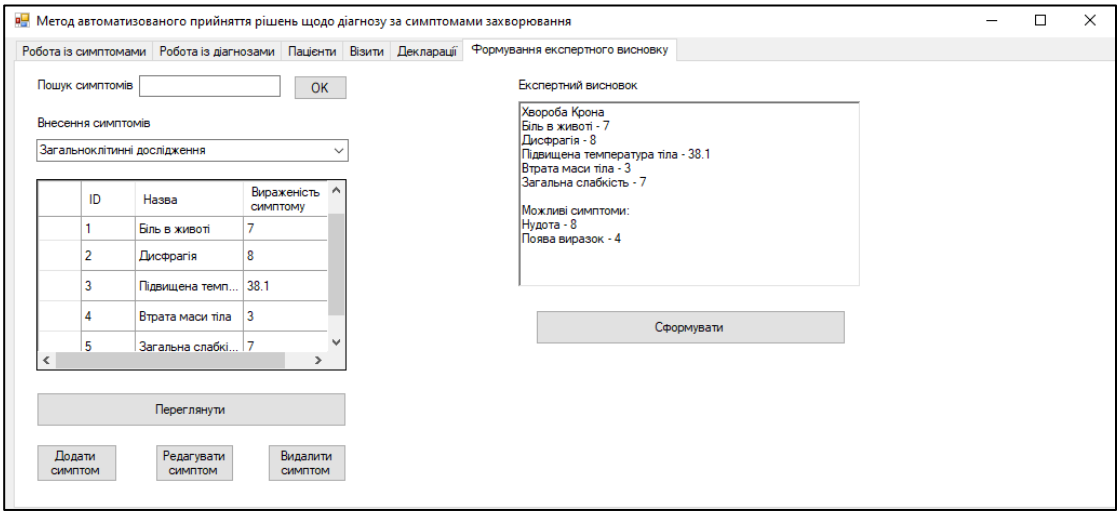


Рисунок 3.11 – Результат виконання тест-кейсу

Із вищенаведеного тест-кейсу було встановлено, що система працює вірно, окрім визначення найбільш вірогідної хвороби додає симптоми, що можуть виникнути при встановленому захворюванні.

Окрім тест-кейсів необхідно покрити програмний продукт GUI тестами, призначення яких – перевірити якість взаємодії додатку та користувача. На рисунку 3.12 продемонстровано результат виконання GUI-тестування.

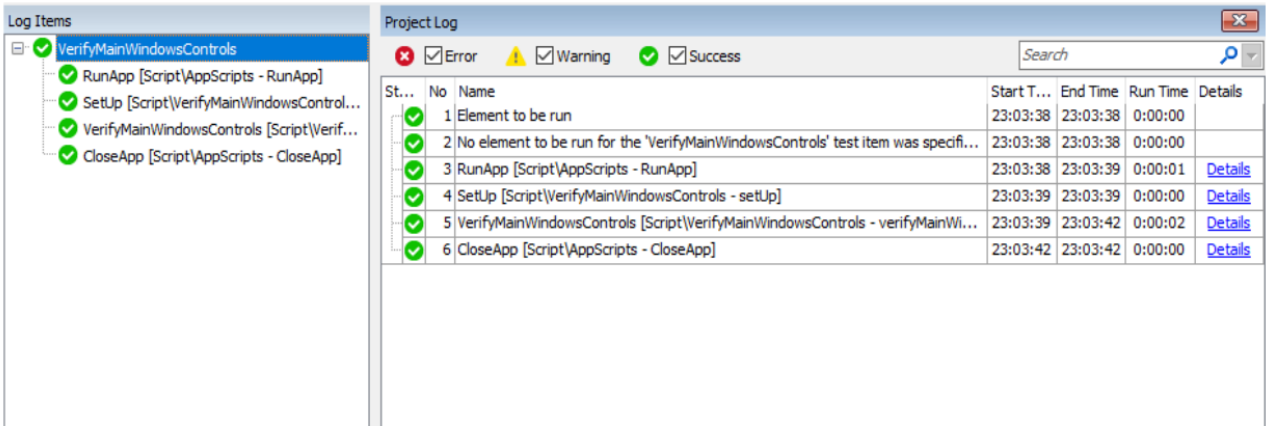


Рисунок 3.12 – Результат виконання GUI-тестування системи

В цьому розділі було проведене комплексне тестування системи, досліджено функціональність системи рядом тест-кейсів та GUI-тестуванням. Таким чином, було визначено працездатність програмного продукту та можливість його подальшого використання.

3.4 Інструкція користувача

Як тільки користувач розпочинає роботу із програмним продуктом, перша вкладка, із якою можна взаємодіяти – «Робота з симптомами». Цей модуль забезпечує наступні функції:

– перегляд існуючих симптомів та перегляд інформації про них;

– редагування та видалення існуючих симптомів;

– внесення нового симптому.

Для перегляду інформації про певний симптом необхідно натиснути на відповідне поле, де знаходиться запис (рисунок 3.13).

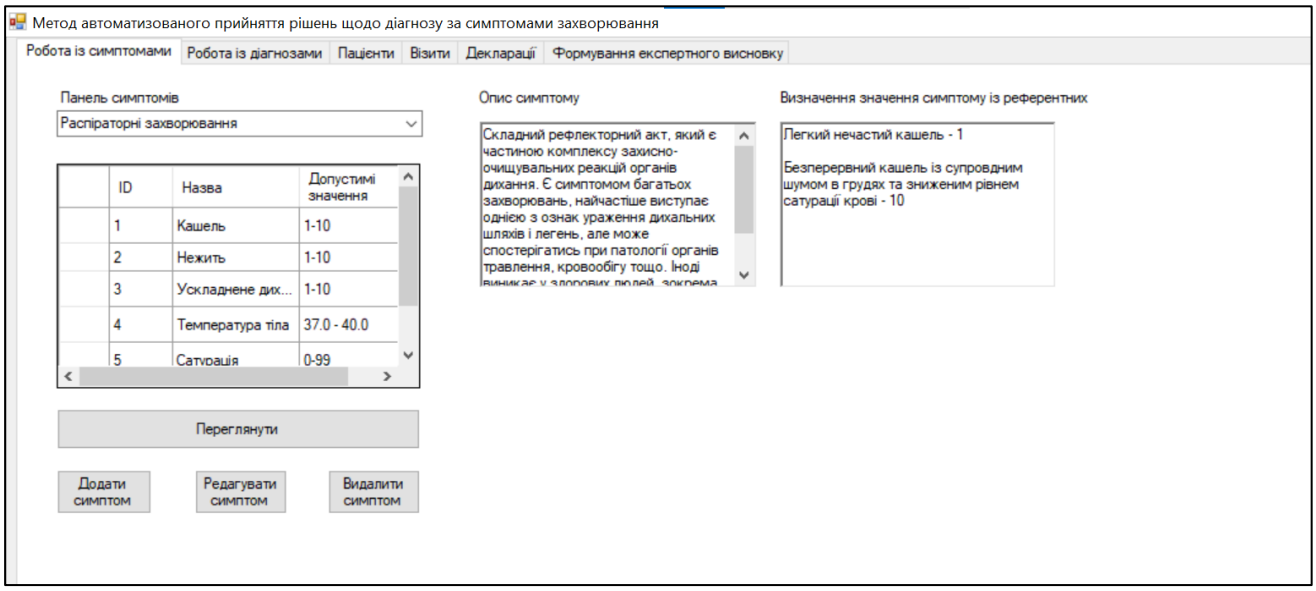


Рисунок 3.13 – Результат перегляду обраного симптому

Якщо користувачеві необхідно додати до БД новий симптом, обирає кнопку «Додати симптом», після чого з’являється відповідна форма із полями для внесення необхідної інформації (рисунок 3.14). Якщо користувач ввів всі необхідні дані, новий симптом з’являється в таблиці відповідної панелі.

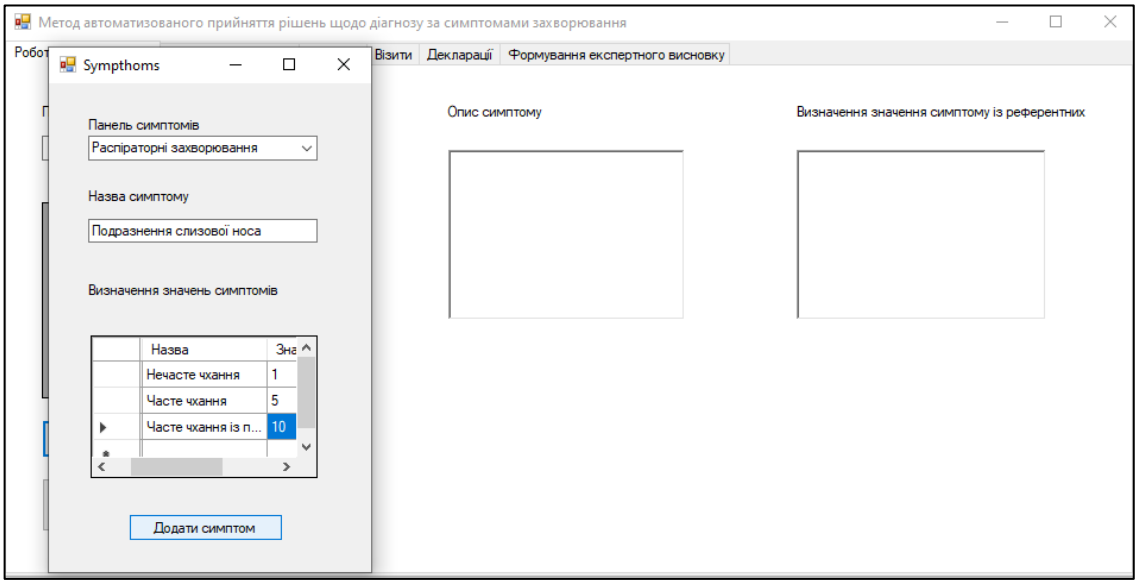


Рисунок 3.14 – Внесення нового симптому до БД

Такі ж дії користувач може зробити і з діагнозами. Для цього необхідно перейти на вкладку «Робота з діагнозами» (рисунок 3.15) та обрати відповідний діагноз із запропонованих панелей.

На цьому модулі користувач може зробити наступні дії:

– перегляд існуючих діагнозів та інформації про них;

– редагування та видалення існуючих діагнозів;

– внесення нового діагнозу.

Для того, щоб переглянути інформацію по кожному діагнозу та симптоми, що характерні при певному захворюванні необхідно натиснути на відповідний запис в таблиці. Роботу програмного модуля «Робота з діагнозами» наведено на рисунку 3.15.

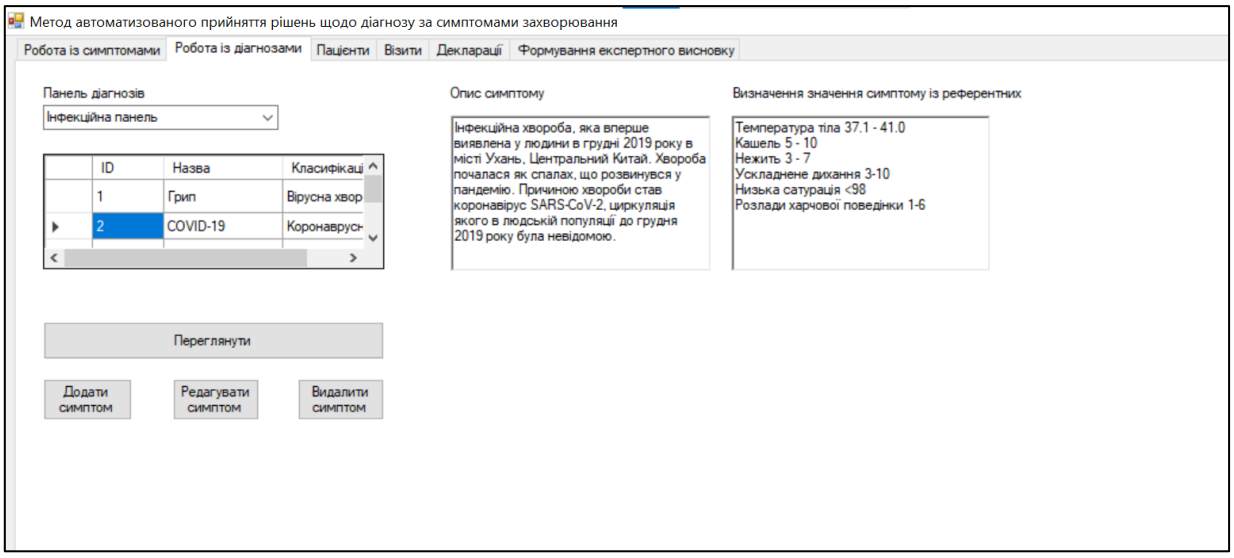


Рисунок 3.15 – Робота із модулем «Робота з діагнозами»

Якщо користувачеві необхідно здійснити будь-які маніпуляції із базою пацієнтів – на вкладці «Пацієнти» реалізовано наступні функції:

– перегляд інформації про пацієнтів, що звертались за медичною допомогою до лікаря;

– реєстрація нового пацієнта;

– редагування та видалення інформації щодо пацієнтів.

Якщо необхідно переглянути інформацію про пацієнта (ПІБ, дату народження, № декларації, хвороби в анамнезі, дату останнього візиту) – необхідно натиснути на відповідний запис у таблиці (рисунок 3.16).

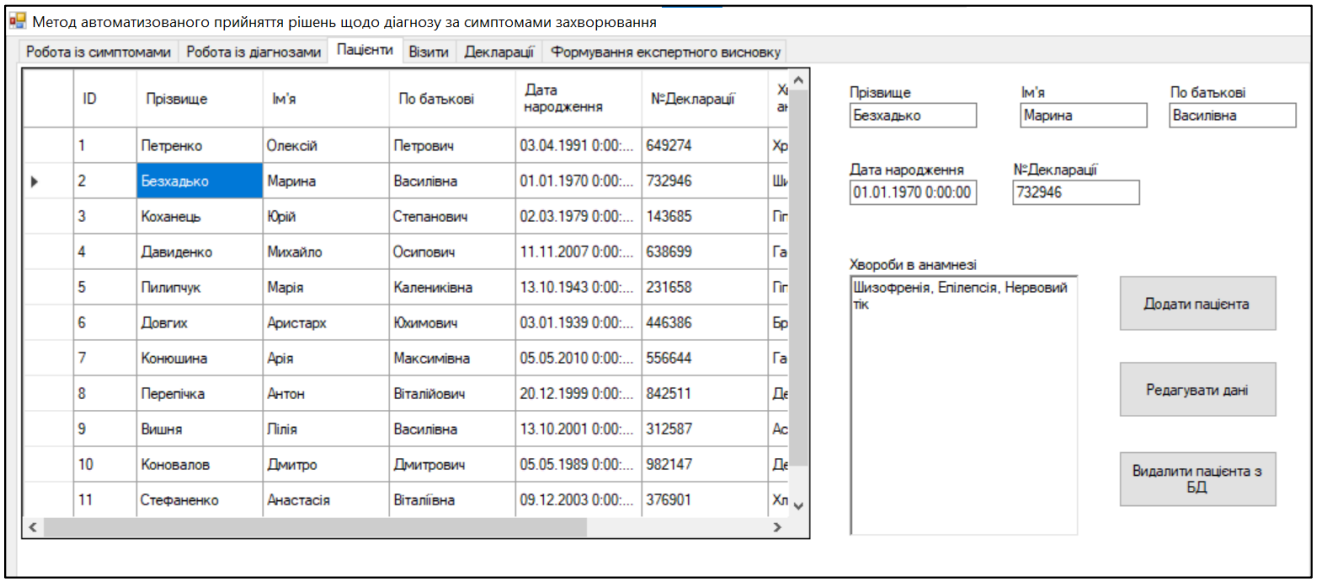


Рисунок 3.16 – Перегляд інформації про пацієнтів

Якщо необхідно зареєструвати нового користувача, у відповідні поля вносяться дані та натискається кнопка «Додати пацієнта» (Рисунок 3.17).

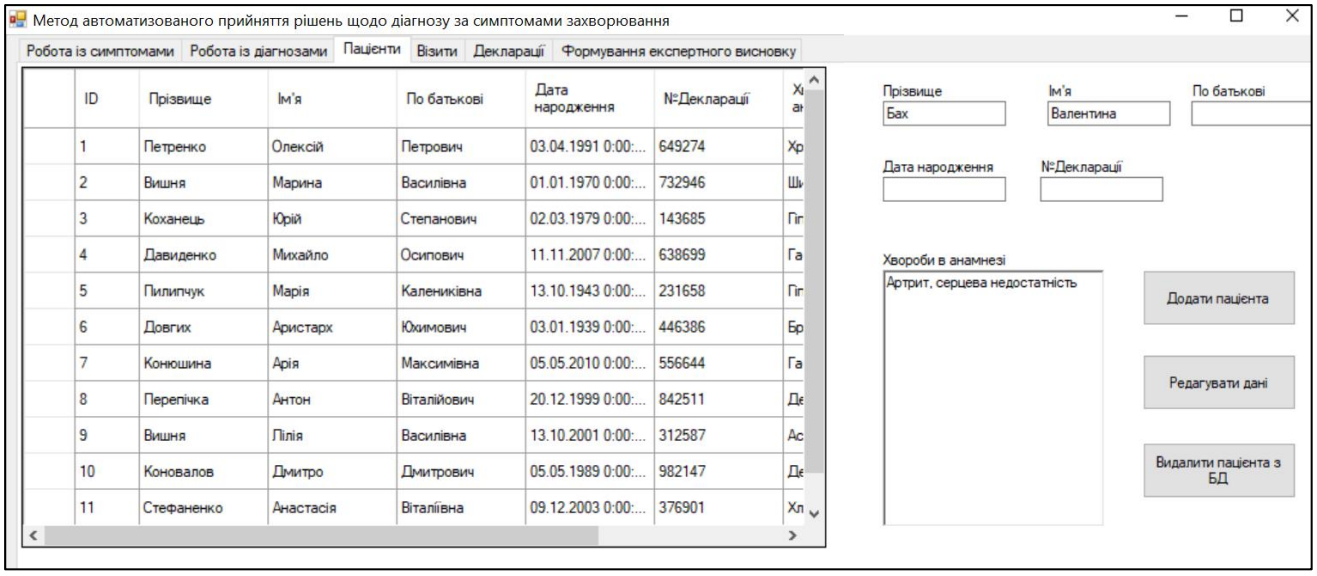


Рисунок 3.17 – Реєстрація нового пацієнта

Якщо мета користувача – формування експертного висновку щодо хвороби за симптомами захворювання, необхідно перейти на вкладку «Формування експертного висновку» та виконати наступні кроки:

1. обрати панель, до якої належить певний симптом;
2. із таблиці, що з’явилась обрати симптом та ввести його вираженість;
3. натиснути кнопку «Додати симптом»;
4. відповідно повторити кроки для кожного симптому;
5. для формування висновку натиснути кнопку «Сформувати».

Відповідно на рисунку 3.18 наведено результат виконання описаних кроків.

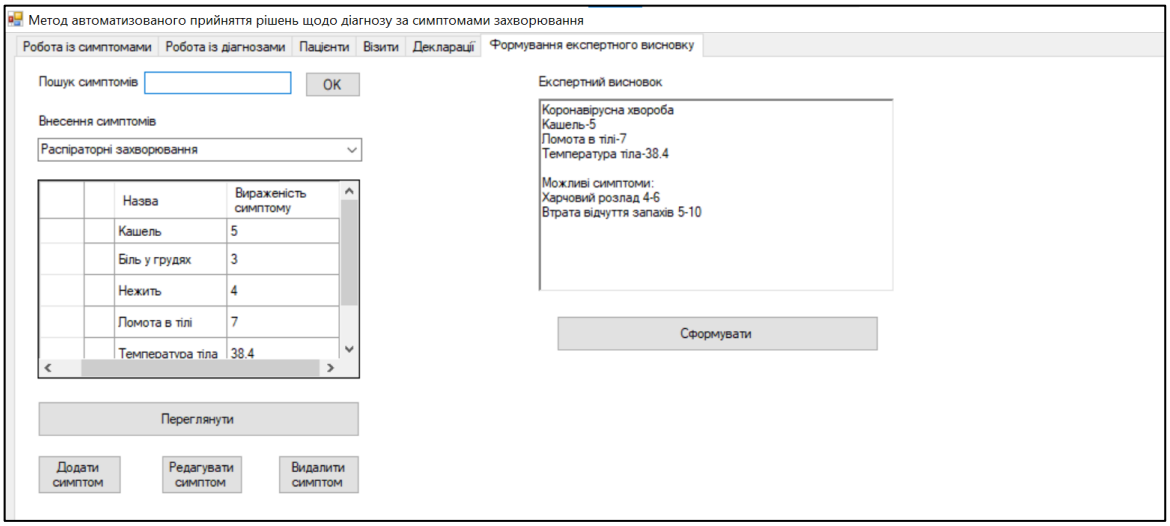


Рисунок 3.18 – Робота із формуванням експертного висновку

Таким чином було створено інструкцію користувача із описом усіх дій, з якими може стикнутись користувач під час роботи. Інтерфейс програмного застосунку є простим та зрозумілим, а завдяки покриттю програми перевірками, вірогідність механічної помилки користувача мінімальна.

3.5 Вимоги до розгортання інформаційної системи

Задля коректної роботи програмного застосунку методу автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання рекомендовано наступні технічні засоби.

Рекомендовані вимоги до апаратних засобів:

– Процесор: Intel Core I7;
– RAM: 8 Gb;
– Вільний дисковий простір: 128 Gb.

Вимоги до програмних засобів:

– ОС: Windows 7/10;
– СКБД MS SQL Server;
– .NET Framework 4.5.

Висновки

В результаті виконання кваліфікаційної роботи бакалавра, було реалізовано метод автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання Для розробки інформаційної системи було використано мову програмування C#, а також систему керування базами даних MS SQL Server.

Реалізований метод виконує наступні функції:

– первинна реєстрація симптомів та їх параметрів;

– підбір можливих діагнозів та їх підтвердження шляхом формування додаткових запитів на діагностування;

– формування експертного висновку із поясненням діагностування захворювань за наявними симптомами;

– робота з експертними відомостями (продукційними правилами діагностування захворювань по симптомах);

– робота з допоміжними даними (хворобами, симптомами, візитами пацієнтів, даними лікарів, деклараціями й індивідуальними медичними картками пацієнтів ЛПЗ).

Також для досягнення мети було виконано наступні етапи:

Характеристика предметної області, дослідження існуючих алгоритмів та методів побудови експертних систем, проєктування моделі інформаційної системи із відповідним математично-алгоритмічним забезпеченням, програмна реалізація створених методів та дослідження ефективності створеного програмного продукту.

Перелік посилань

1.Мойсак О. Д., Тимчик О. В. Основи медичних знань та охорони здоров'я, 2018.

2. Зінгерман, б. В.; Шкловський-корді, н. Є.; Вороб'єв, а. І. Про телемедицину "Пацієнт-лікар". Лікар та інформаційні технології , 2017, 1: 61-79.

3. Наливайко, Ольга Борисівна. Формування професійної культури майбутніх сімейних лікарів як педагогічна проблема. Сучасні інформаційні технології та інноваційні методики навчання в підготовці фахівців: методологія, теорія, досвід, проблеми, 2016, 46: 277-281.

4. Західна, О. Р.; Мидлик, Ю. І. Медична реформа в умовах децентралізації влади в Україні. Молодий вчений, 2017, 11: 1155-1158.

5. Декларація про вибір лікаря ПМД. URL: https://www.medsprava.com.ua/article/1086-deklaratsya-pro-vibr-lkarya-pervinno-medichno-dopomogi-blank-pravila-zapovnyuvannya

6. Зразок паперового талону для запису до лікаря. URL: https://minfin.com.ua/ua/2018/03/14/32771410/

7. Medics. URL: https://medics.ua/

8. Зразок паперової медичної картки. URL: https://www.chaspik.ua/medichna-51284/.

9. Марценюк, В. П.; Семенець, А. В. Медична інформатика. Інструментальні та експертні системи, 2003.

10. Бубела, М. О., В. О. Шевчук, Л. І. Поліщук. Аналітичний огляд перспективи застосування штучного інтелекту в медицині, ЦНТУ, 2021.

11. Білоус, Д. В. Використання експертних систем в медицині, 2021.

12. Козлова, О. В. Переваги експертних систем над традиційними системами штучного інтелекту. Системи озброєння і військова техніка, 2011, 1: 104-106.

13. Сєдих, Ольга Леонідівна; Овчарук, Володимир Олексійович. Дослідження методології побудови та принципів функціонування експертних систем, 2016.

14. Обстеження онлайн «Sympthomate». URL: https://symptomate.com/uk/

15. Система онлайн-діагностики здоров'я від мережі аптек «АНЦ». URL: https://anc.ua/doc/

16. Microsoft .NET. URL: https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet

17. Benefits of .NET. URL: https://scand.com/company/blog/net-vs-java-comparison/

18. .NET vs Java. URL: https://diceus.com/java-vs-net/#:~:text=Java%20is%20a%20platform%2Dagnostic,Code%20execution.

19. NET Core. URL: https://www.xelent.en/blog/

20. .NET Framework 4.5. URL: https://www.microsoft.com/en-en/download/details.aspx?id=30653

21. .NET Framework and .NET Core: pros and cons. URL: https://www.interviewbit.com/blog/net-core-vs-net-framework/#:~:text=NET%20Framework%20is%20a%20platform,app%20needs%20and%20developer%20workflows.

22. Хох, В. Д.; Мелешко, Є. В.; Якименко, М. С. Дослідження методів побудови експертних систем. Системи управління, навігації та зв'язку, 2016, 4: 48-52.

23. Міхалевська, Галина Іванівна; Міхалевський, Віталій Цезарійович. Основні концепції побудови експертних систем, 2010.

24. Дудзінський, Юрій Михайлович; Буковська, Марія Ігорівна. Навчальні експертні системи у медицині. Фізика та медицина у сучасному житті, 72.

25. Bishop J., Horspool R. N., Worrall B. Experience in integrating Java with C# and. NET. Concurrency and Computation: Practice and Experience, 2005, Т. 17, №. 5-6, С. 663-680.

26. C# Language. URL: https://www.codeguru.com/csharp/benefits-of-c/

27. Benefits of C#. URL: https://www.bairesdev.com/technologies/csharp/

28. Benefits of using Java. URL: https://data-flair.training/blogs/pros-and-cons-of-java/

29. Java VS C#. URL: https://hackr.io/blog/c-sharp-vs-java#:~:text=Key%20Differences%20Between%20C%23%20and%20Java,-Runtime%20Environment%3A%20Java&text=Programming%20Paradigm%3A%20Java%20is%20a,operator%20overloading%20for%20multiple%20operators.

30. Eclipse IDE. URL: https://uk.wikipedia.org/wiki/Eclipse

31. Visual Studio. URL: https://www.microsoft.com/rus/msdn/vs/default.mspx

32. Melton, Jim; Simon, Alan R. Understanding the new SQL: a complete guide. Morgan Kaufmann, 1993.

33. MySQL. URL: https://www.mysql.com/].

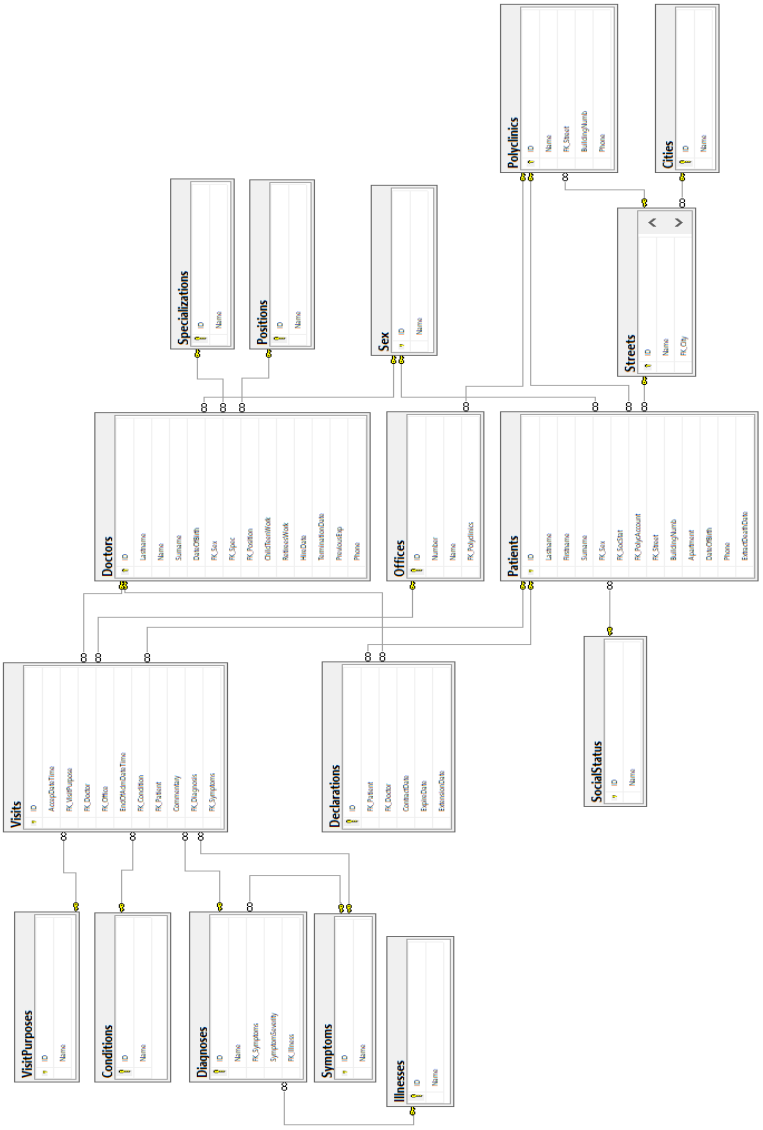
34. MS SQL Server. URL: https://www.microsoft.com/ru-ru/download/details.aspx?id=35579

35. Comparing MySQL and MS SQL Server. URL: https://www.geeksforgeeks.org/difference-between-mysql-and-ms-sql-server/

ДОДАТКИ

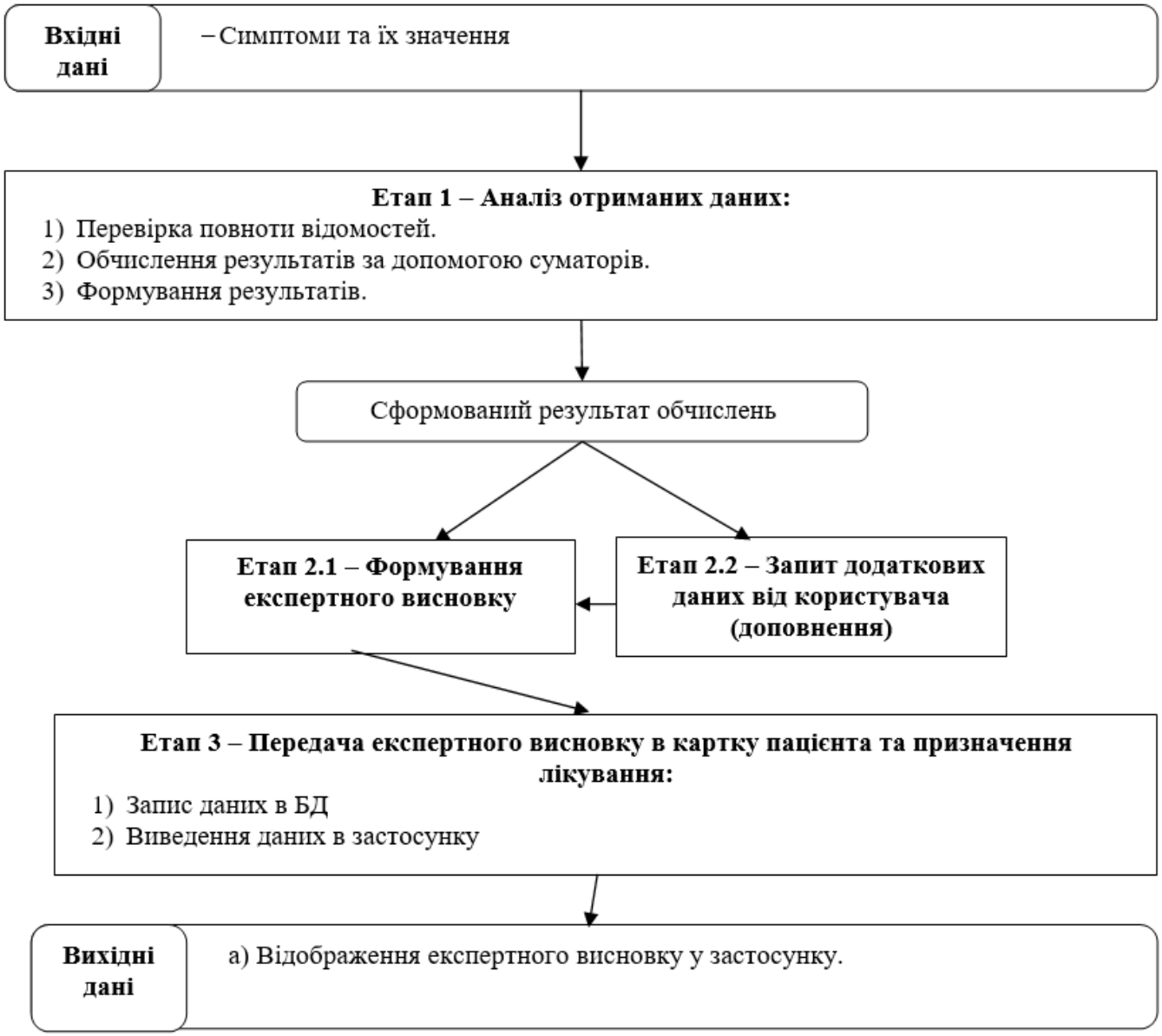
Додаток А

Структура бази даних метод автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання



Додаток Б

Схема методу автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання



Додаток В

Програмні коди

Form1.cs

```
using ExpertSystemCOVID.Models; using
System;

using System.Data; using System.Drawing;
using System.Linq; using System.Text; using
System.Threading;
using System.Threading.Tasks; using
System.Windows.Forms;

namespace ExpertSystemCOVID
{
public partial class Form1 : Form
{
Entities1 context = newEntities1();
NeuralNetwork Learn;

public Form1()
{
InitializeComponent();

dataGridView4.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
dataGridView4.AutoSizeRowsMode =
DataGridViewAutoSizeRowsMode.AllCells;

dataGridView1.DefaultCellStyle.SelectionBac
kColor =
dataGridView1.DefaultCellStyle.BackColor;
dataGridView1.DefaultCellStyle.SelectionFor
eColor =
dataGridView1.DefaultCellStyle.ForeColor;

dataGridView2.DefaultCellStyle.SelectionBac
kColor =
dataGridView1.DefaultCellStyle.BackColor;
dataGridView2.DefaultCellStyle.SelectionFor
eColor =
dataGridView1.DefaultCellStyle.ForeColor;

dataGridView3.DefaultCellStyle.SelectionBac
kColor =
dataGridView1.DefaultCellStyle.BackColor;
dataGridView3.DefaultCellStyle.SelectionFor
eColor =
dataGridView1.DefaultCellStyle.ForeColor;

var DataType = context.DataType; foreach
(var i in DataType)
comboBox1.Items.Add(i.Name);
comboBox1.SelectedIndex =
comboBox1.Items.Count - 1;

var ComboResult = context.Results; foreach
(var i in ComboResult)
comboBox2.Items.Add(i.Name);
comboBox2.SelectedIndex =
comboBox2.Items.Count - 1;

var epoch = context.Epoch; foreach (var i
in epoch)
using System.Collections.Generic; using
System.ComponentModel;


dataGridView1.Rows.Add(i.ID,  i.Name);

var parameters = context.Parameters;
foreach (var i in parameters)
dataGridView2.Rows.Add(dataGridView2.Rows.C
ount + 1, i.Name, i.UnitsMeasurement,
i.DataType.Name);

var results = context.Results; foreach (var
i in results)
dataGridView3.Rows.Add(dataGridView3.Rows.C
ount + 1, i.Name);

var trainingSamples =
context.TrainingSamples; foreach (var i in
trainingSamples)
dataGridView4.Rows.Add(i.ID  ,i.Name);


string[] row15 = new string[] { "Логічний",
"0", "1"};

string[] row25 = new string[] {
"Динаміний", "Спадає", "Зростає" };
object[] rows5 = new object[] { row15,
row25 };
foreach (string[] rowArray in rows5)
{
dataGridView6.Rows.Add(rowArray);
}

}

private void
dataGridView6_CellContentClick(object
sender, DataGridViewCellEventArgs
e)
{

}

private void button1_Click(object sender,
EventArgs e)
{
try
{
context.Epoch.Add(new Epoch
{
Name = textBox1.Text,
});
context.SaveChanges();
MessageBox.Show("Об'єкт успішно
збережено");

//Refresh dataGridView1.Rows.Clear(); var
epoch = context.Epoch; foreach (var i in
epoch)
dataGridView1.Rows.Add(i.ID,  i.Name);
```

```
}
catch
{
MessageBox.Show("Заповність всі поля");
}
}

private void button2_Click(object sender,
EventArgs e)
{

try
{
context.Parameters.Add(new  Parameters
{
Name = textBox2.Text, UnitsMeasurement =
textBox3.Text,
FK_DataType = context.DataType.Where(u =>
u.Name ==
comboBox1.SelectedItem.ToString()).FirstOrDe
fault().ID,
});
context.SaveChanges();
MessageBox.Show("Об'єкт успішно
збережено");
//Refresh dataGridView2.Rows.Clear();
var parameters = context.Parameters;
foreach (var i in parameters)
dataGridView2.Rows.Add(dataGridView2.Rows.C
ount + 1, i.Name, i.UnitsMeasurement,
i.DataType.Name);
}
catch
{
MessageBox.Show("Заповність всі поля");
}

private void button3_Click(object sender,
EventArgs e)
{

try
{
context.Results.Add(new Results
{
Name = textBox4.Text,
});
context.SaveChanges();
MessageBox.Show("Об'єкт успішно
збережено");

//Refresh dataGridView3.Rows.Clear(); var
results = context.Results; foreach (var i
in results)
dataGridView3.Rows.Add(dataGridView3.Rows.Cou
nt+1,   i.Name);

comboBox2.Items.Add(textBox4.Text);
}
catch
{
MessageBox.Show("Заповність всі поля");
}
}
Dictionary<int, DataGridView> sample = new
Dictionary<int, DataGridView>();
```

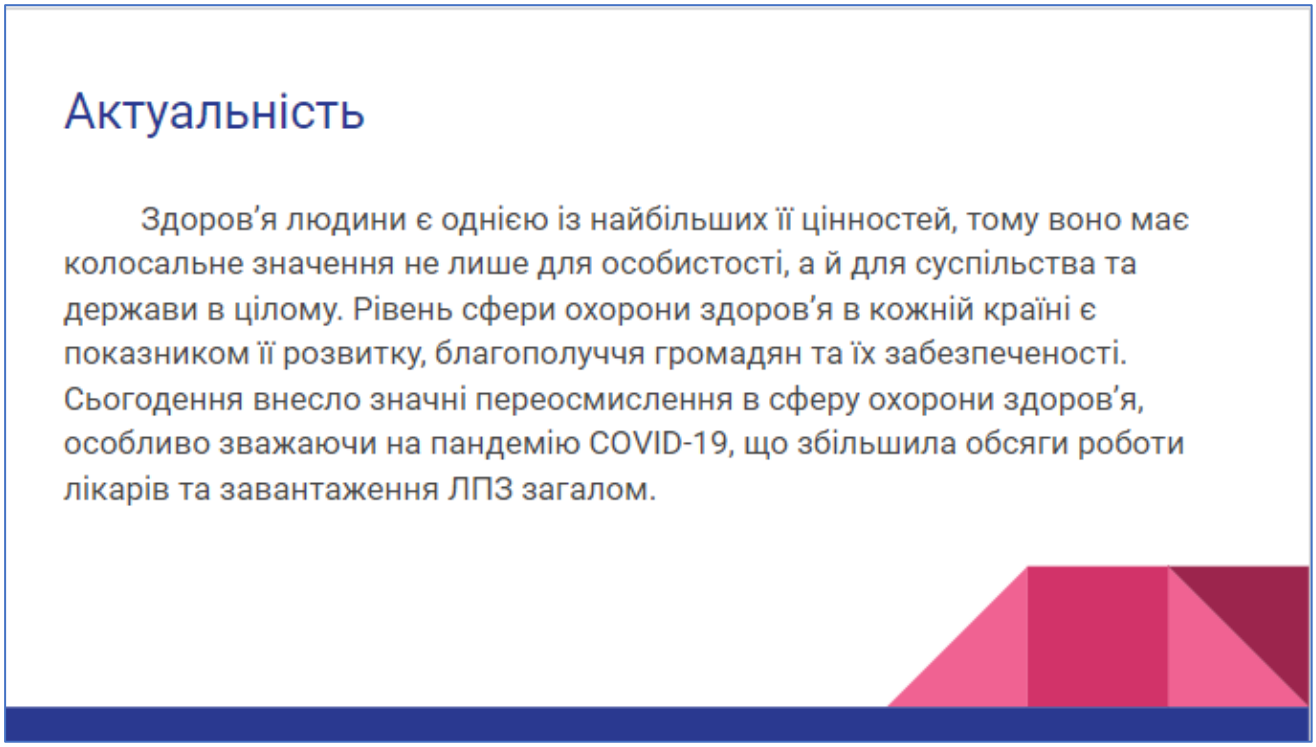
```
private void button4_Click(object sender,
EventArgs e)
{


try
{
context.TrainingSamples.Add(new
TrainingSamples
{
Name = textBox5.Text,
FK_Result = context.Results.Where(u =>
u.Name ==
comboBox2.SelectedItem.ToString()).FirstOrDe
fault().ID
});
context.SaveChanges();

for (int i = 0; i <
dataGridView5.Rows.Count; i++)
{
for (int j = 1; j <
dataGridView5.Columns.Count; j++)
{
string str1 =
dataGridView5.Columns[j].HeaderText.Substri
ng(0,
dataGridView5.Columns[j].HeaderText.IndexOf(
','));
string str2 = dataGridView5[0,
i].Value.ToString();
context.ParameterValues.Add(new
ParameterValues

Value = Convert.ToDouble(dataGridView5[j,
i].Value), FK_Parameter =
context.Parameters.Where(u => u.Name ==

FK_Epoch = context.Epoch.Where(u => u.Name
==

FK_TrainingSamples =
context.TrainingSamples.Where(u => u.Name ==
```

```
textBox5.Text).FirstOrDefault().ID
});
context.SaveChanges();

MessageBox.Show("Об'єкт успішно збережено");

//Refresh dataGridView4.Rows.Clear();
var trainingSamples =
context.TrainingSamples; foreach (var i in
trainingSamples)
dataGridView4.Rows.Add(i.ID,  i.Name);

}
catch
{
MessageBox.Show("Заповність всі поля");
}

}

private void button5_Click(object sender,
EventArgs e)
{
int ID =
Convert.ToInt32(dataGridView4.CurrentCell.Va
lue);

var delete = context.ParameterValues.Where(u
=> u.FK_TrainingSamples == ID); foreach(var
i in delete)
context.ParameterValues.Remove(i);

context.TrainingSamples.Remove(context.Trai
ningSamples.Where(u => u.ID ==
ID).FirstOrDefault());

context.SaveChanges();

dataGridView4.Rows.Clear();
var trainingSamples =
context.TrainingSamples; foreach (var i in
trainingSamples)
dataGridView4.Rows.Add(i.ID,  i.Name);
}


private void button6_Click(object sender,
EventArgs e)
{
List<double[]> inputsList = new
List<double[]>();



int[] arrID = new
int[dataGridView4.Rows.Count]; int p = 0;

foreach (DataGridViewRow row in
dataGridView4.Rows)
{
arrID[p] =
Convert.ToInt32(row.Cells[0].Value); p++;
}

for (int i = 0, j =0; i < arrID.Length; i++)
{
j = arrID[i];
var q = context.ParameterValues.Where(u =>
u.TrainingSamples.ID == j);

double[] Training = new double[q.Count()];
int z = 0;
foreach (var l in q)
{
Training[z] = (double)l.Value; z++;
}
inputsList.Add(Training);

dataGrid.DefaultCellStyle.SelectionForeColo
r =
dataGridView1.DefaultCellStyle.ForeColor;
}

private void
dataGridView4_CellDoubleClick(object
sender, DataGridViewCellEventArgs
e)
{
dataGridView5.Rows.Clear();
dataGridView5.Columns.Clear();
dataGridView5.Columns.Add("column" +
dataGridView5.Columns.Count + 1, "Епохи");



int ID = (int)dataGridView4[e.ColumnIndex,
e.RowIndex].Value;

var columnID =
context.ParameterValues.Where(u =>
u.FK_TrainingSamples == ID).GroupBy(u =>
u.FK_Parameter);
var rowID = context.ParameterValues.Where(u
=> u.FK_TrainingSamples == ID).GroupBy(u =>
u.FK_Epoch);

foreach (var i in columnID)
{
dataGridView5.Columns.Add("column" +
dataGridView5.Columns.Count + 1,
context.Parameters.Where(u => u.ID ==
i.Key).FirstOrDefault().Name);
}

foreach (var i in rowID)
{
dataGridView5.Rows.Add(context.Epoch.Where(
u => u.ID == i.Key).FirstOrDefault().Name);
}

for (int i = 0; i <
dataGridView5.Rows.Count; i++)
{
for (int j = 1; j <
dataGridView5.Columns.Count; j++)
{
```

```
string str1 =
dataGridView5.Columns[j].HeaderText; int
ID1 = context.Parameters.Where(u => u.Name
==
str1).FirstOrDefault().ID;
string str2 = dataGridView5[0,
i].Value.ToString();
int ID2 = context.Epoch.Where(u => u.Name ==
str2).FirstOrDefault().ID; dataGridView5[j,
i].Value = context.ParameterValues.Where(u
=>
u.FK_Parameter == ID1 && u.FK_Epoch == ID2
&&
u.FK_TrainingSamples ==
ID).FirstOrDefault().Value;

}
}

comboBox2.SelectedIndex =
context.TrainingSamples.Where(u => u.ID ==
ID).FirstOrDefault().Results.ID-1;
private void LoadData()
        {
            string connectString =
"Server=.\\SQLEXPRESS;Initial
Catalog=doctor;Integrated Security=true;";
            SqlConnection myConnection =
new SqlConnection(connectString);
            myConnection.Open();

            string query = "SELECT * FROM
Patients ORDER BY ID";
            SqlCommand command = new
SqlCommand(query, myConnection);

            SqlDataReader reader =
command.ExecuteReader();

            List<string[]> dataa = new
List<string[]>();

            while (reader.Read())
            {
                dataa.Add(new string[8]);
                dataa[dataa.Count - 1][0] =
reader[0].ToString();
                dataa[dataa.Count - 1][1] =
reader[1].ToString();
                dataa[dataa.Count - 1][2] =
reader[2].ToString();
                dataa[dataa.Count - 1][3] =
reader[3].ToString();
                dataa[dataa.Count - 1][4] =
reader[4].ToString();
                dataa[dataa.Count - 1][5] =
reader[5].ToString();
          }
                dataa[dataa.Count - 1][6] =
reader[6].ToString();
                dataa[dataa.Count - 1][7] =
reader[7].ToString();

            }
            reader.Close();
            myConnection.Close();

            foreach (string[] s in dataa)
                dataGridView2.Rows.Add(s);
        }


    public void loadSymthoms()
           {
               string connectString =
    "Server=.\\SQLEXPRESS;Initial
    Catalog=doctor;Integrated
    Security=true;";
               SqlConnection
    myConnection          =          new
    SqlConnection(connectString);
               myConnection.Open();

               string query1 = "SELECT
    * FROM Symthoms ORDER BY ID";
               SqlCommand command = new
    SqlCommand(query1, myConnection);

               SqlDataReader reader1 =
    command.ExecuteReader();

               List<string[]>  dataa  =
    new List<string[]>();

               while (reader1.Read())
               {
                   dataa.Add(new
    string[3]);
                   dataa[dataa.Count  -
    1][0] = reader1[0].ToString();
                   dataa[dataa.Count  -
    1][1] = reader1[1].ToString();
                   dataa[dataa.Count  -
    1][2] = reader1[2].ToString();
                       }
               reader1.Close();
               myConnection.Close();

               foreach  (string[]  s  in
    dataa)

    dataGridView1.Rows.Add(s);
           }
```

Додаток Г

Презентаційний матеріал



Завдання

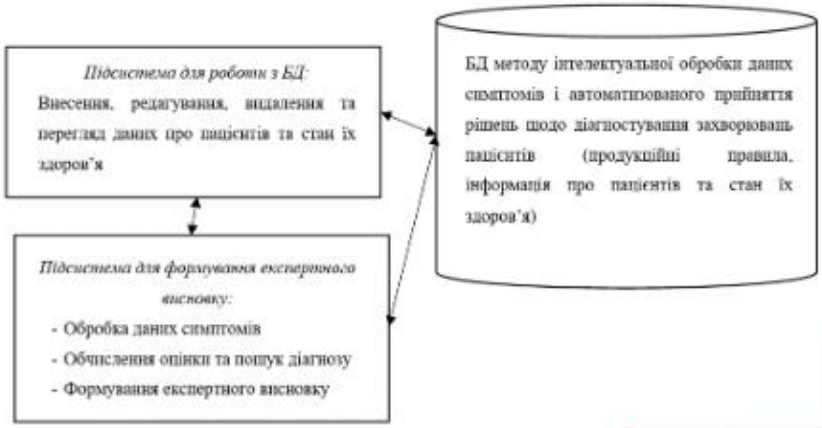
Метою кваліфікаційної роботи є розробка методу автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання, та його програмна реалізація.

Програмний застосунок має виконувати такі функції:

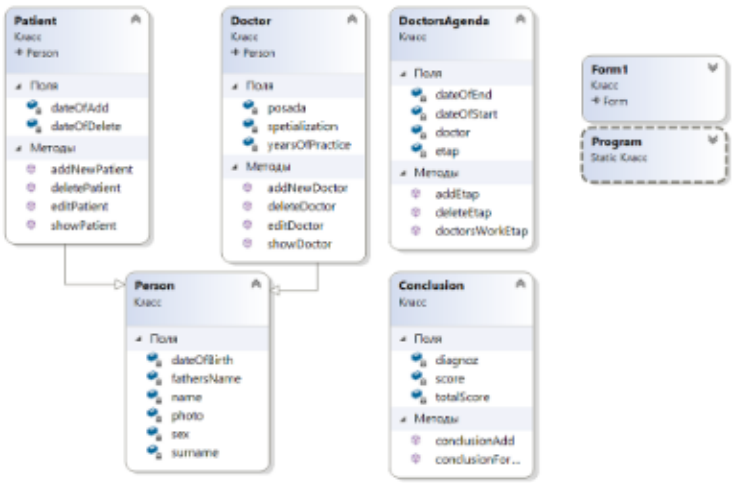
- первинна реєстрація симптомів та їх параметрів;
- підбір можливих діагнозів та їх підтвердження шляхом формування додаткових запитів на діагностування;
- формування експертного висновку із поясненням діагностування захворювань за наявними симптомами;
- робота з експертними відомостями (продукційними правилами діагностування захворювань по симптомах);
- робота з допоміжними даними (хворобами, симптомами, візитами пацієнтів, даними лікарів, деклараціями й індивідуальними медичними картками пацієнтів ЛПЗ).

Схема методу автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання

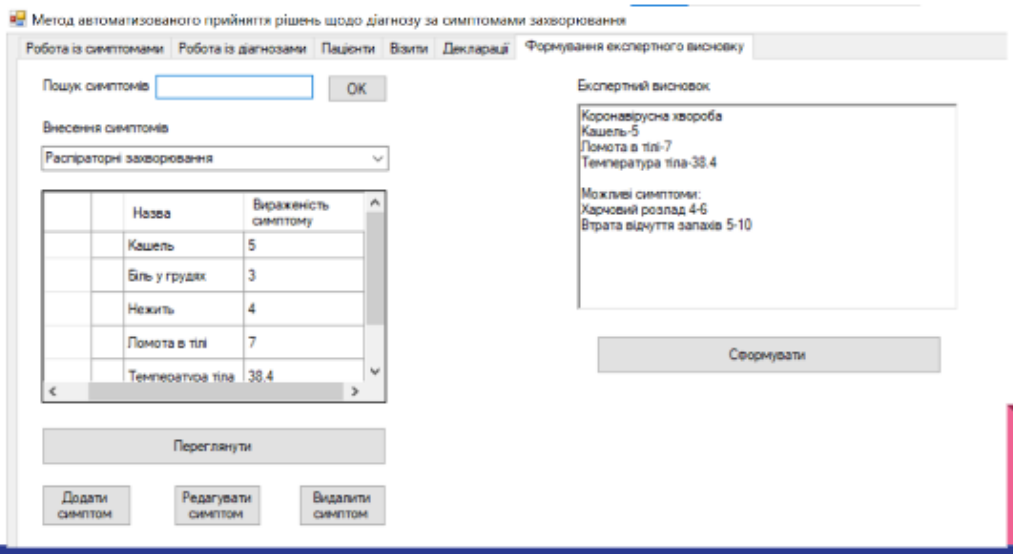
Структура компонентів методу автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання



Діаграма класів методу автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання



Результат роботи програмного застосунку



Mon May 30 07:47:18 EEST 2022, Петровський Сергій Степанович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальное совпадение с одним документом 4.0%

Словари проверки: en_US, ru_RU, ua_UA. **Ошибок в документах: 9%**

	Документ		Суммарное совпадение по Базе Данных	
ID: 104159 Название: КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА Метод автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання Добавлено в БД: 2022-05-30 Авторы: Н.С.Домбровський Руководители: Т.К.Скрипник Консультанты: Опоненты:	Символы	Лексемы	Символы	Лексемы
	43792	669	3388 (8%)	53 (8%)

Источник плагиата

ID	Описание	Наличие плагиата в документе	
		Символы	Лексемы

UNICHECK

Ім'я користувача:
Кафедра КН

ID перевірки:
1011369559

Дата перевірки:
30.05.2022 08:38:05 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
30.05.2022 08:39:18 EEST

ID користувача:
100005671

Назва документа: **Домбровський_ЗАПИСКА_short**

Кількість сторінок: **61** Кількість слів: **8658** Кількість символів: **64038** Розмір файлу: **3.07 MB** ID файлу: **1011254702**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

12.2%
Схожість

Найбільша схожість: **7.77%** з джерелом з Бібліотеки (ID файлу: **1011254530**)

1.41% Джерела з Інтернету	53	Сторінка 63
11.4% Джерела з Бібліотеки	87	Сторінка 63

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування	26 сторінок

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ КАФЕДРИ КОМП'ЮТЕРНИХ НАУК ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: *Метод автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання*

Автор: *студент групи КН-18-1 Домбровський Назарій Сергійович*

Спеціальність: *122 – Комп'ютерні науки*

Освітня програма: *освітньо-професійна*

Науковий керівник: *старший викладач кафедри КН Скрипник Т.К.*

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	***відповідає***
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі Домбровського Н.С. не є плагіатом, оскільки:

1. запозичення розміщені в розділі огляду існуючих підходів, що не описують безпосередньо авторську роботу і не стосуються її результатів;

2. усі запозичення фрагментарні;

3. до запозичень входять фрагменти програмного коду, що не мають авторства і містять поширені конструкції;

4. серед запозичень знаходяться загальновідомі терміни та скорочення;

5. обсяг запозичень, визначений системами виявлення збігів/ідентичності/схожості, складає:

- за системою Anti-Plagiarism: 4%;

- за системою Unicheck: 12.2 %.

Отже, запозичення є допустимими та відносяться до описаних вище і адресуються до першоджерел, що, з урахуванням наведених обґрунтувань, свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ *Тетяна СКРИПНИК*

Гарант ОП _____ *Олександр МАЗУРЕЦЬ*

Завідувач кафедри КН _____ *Олександр БАРМАК*

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

Кафедра комп'ютерних наук

РЕЦЕНЗІЯ

на кваліфікаційну роботу бакалавра

студента *гр. КН-18-1 Домбровського Назарія Сергійовича*
за темою: *Метод автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання*

1. Актуальність обраної теми

Наразі лікарі в медичних закладах надмірно навантажені рутинною роботою, такою як: реєстрація візиту пацієнта, зберігання даних про його здоров'я, заповнення його медичної картки. Крім того, їм потрібно призначати діагноз та певне лікування. Автоматизована система прийняття рішень щодо діагнозу за симптомами захворювання призначена для виконання цих завдань.

2. Повнота розкриття мети та завдань роботи

Протягом виконання кваліфікаційної роботи бакалавра мету та завдання роботи було розкрито у повній мірі. Мету та завдання можна вважати розкритими, адже було проведено повний аналіз предметної області, чітко визначено структуру інформаційної системи та реалізовано відповідний програмний продукт.

3. Зміст кожного розділу роботи

У розділах записки кваліфікаційної роботи було описано характеристику предметної області та постановку задачі, проєктування структури інформаційної систем методу автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання та відображення загальної інформації, а також проєктування методу та бази даних. Розділи також містять опис програмної реалізації інформаційних систем, а саме структуру, особливості реалізації та тестування.

4. Оцінка розробленої інформаційної системи, її практична цінність

Розроблений метод автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання та його програмна реалізація цілком може широко використовуватись у будь-якому лікувально-профілактичному закладі.

5. Якість оформлення кваліфікаційної роботи бакалавра

Записка оформлена якісно, з логічним викладенням матеріалу та наведенням вагомих аргументів. Написання літературно грамотне та доцільне.

6. Недоліки кваліфікаційної роботи бакалавра

Використовуються схожі, проте не тотожні терміни в пояснювальній записці. В переліку посилань використовуються застарілі джерела.

7. Загальний висновок (допускається чи не допускається до захисту), та оцінка на яку заслуговує кваліфікаційна робота.

Враховуючи рівень виконання та забезпечення усіх необхідних вимог, робота може бути допущена до захисту. Рекомендована оцінка ***«добре»***.

Рецензент _____ Т. Говорущенко

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

Кафедра комп'ютерних наук

ВІДГУК НАУКОВОГО КЕРІВНИКА

на кваліфікаційну роботу бакалавра

студента *гр. КН-18-1 Домбровського Назарія Сергійовича*

за темою *Методу автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання*

1. Актуальність теми

Наразі лікарі в медичних закладах надмірно навантажені рутинною роботою, такою як: реєстрація візиту пацієнта, зберігання даних про його здоров'я, заповнення його медичної картки. Крім того, їм потрібно призначати діагноз та певне лікування. Автоматизована система прийняття рішень щодо діагнозу за симптомами захворювання призначена для виконання цих завдань.

2. Відповідність роботи предметній області Стандарту спеціальності 122 Комп'ютерні науки

Відповідно до стандарту спеціальності 122 Комп'ютерні науки, метою роботи є розробки методів і технологій отримання, зберігання, обробки, передачі та використання інформації, інтелектуального аналізу даних і прийняття рішень, а саме метою прийняття рішень щодо діагнозу за симптомами захворювання призначений для виконання цих завдань. Для досягнення мети роботи використовуються математичні моделі, методи та алгоритми розв'язання теоретичних і прикладних задач, що виникають при розробці інформаційних технологій. Тому результати виконання кваліфікаційної роботи бакалавра відповідають стандарту бакалавра спеціальності 122 Комп'ютерні науки.

3. Професійні та особистісні якості бакалавра

Під час роботи над кваліфікаційною роботою бакалавра Домбровський Назарій Сергійович проявив себе кваліфікованим та дисциплінованим спеціалістом, вчасно та якісно виконував поставлені задачі. В процесі виконання поставлених задач проявив достатні для одержання успішного результату компетентності та результати навчання. Під час навчання опанував професійні навички за напрямком «Комп'ютерні науки».

4. Ступінь самостійності під час виконання кваліфікаційної роботи

Одержані в роботі результати є наслідком особистої діяльності студента, який самостійно виконував всі поставлені задачі.

5. Ступінь оволодіння методами дослідження

При реалізації кваліфікаційної роботи показав достатній рівень компетентностей та володіння необхідними інструментами та обладнанням, методами, методиками та технологіями предметної області комп'ютерних наук.

6. Повнота та якість розкриття теми роботи

Тема роботи в повній мірі обґрунтована й розкрита, проведено аналіз актуальності та відомих досліджень в межах обраної теми, поставлені завдання, які у роботі виконані, та розроблено відповідне програмне забезпечення.

7. Логічність, послідовність, аргументованість, літературна грамотність викладення матеріалу

Структура роботи та послідовність викладення логічні та відповідають поставленій меті. Викладення матеріалу послідовне, аргументоване, літературно грамотне.

8. Можливість практичного застосування кваліфікаційної роботи бакалавра, окремих її частин

Розроблений у роботі методу автоматизованого прийняття рішень щодо діагнозу за симптомами захворювання та його програмна реалізація може бути використана в лікувально-профілактичних закладах.

9. Висновок про можливість допуску кваліфікаційної роботи бакалавра до захисту, на яку оцінку заслуговує робота

Враховуючи достатній рівень виконання та забезпечення усіх необхідних вимог, робота може бути допущена до захисту. Рекомендована оцінка «добре».

Керівник _____ *старший викладач кафедри КН Т.К.Скрипник*