МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК
СЕКЦІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЕКТУВАННЯ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: «Інформаційна технологія підтримки прийняття рішень при діагностуванні гострих респіраторних інфекцій»

за спеціальністю 122 «Комп'ютерні науки»,
освітньо-наукова програма «Інформаційні технології проектування»

Виконавець роботи: студент групи ІТ.м.н-71 Бабенко Аліна Вікторівна

Кваліфікаційну роботу захищено на засіданні ЕК з оцінкою _____ «___» травня 2019 р.

Науковий керівник _____ к.т.н., доц., Шендрик В.В.
(підпис)

Голова комісії _____ Шифрін Д.М.
(підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент __
(підпис)

Суми-2019

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук
Секція інформаційних технологій проектування
Спеціальність 122 «Комп'ютерні науки»
Освітньо-наукова програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ

Зав. секцією ІТП

_____ В. В. Шендрик

«__» _____ 2019 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові

Бабенко Аліна Вікторівна

(прізвище, ім'я, по батькові)

1 Тема проекту *Інформаційна технологія підтримки прийняття рішень при діагностуванні гострих респіраторних інфекцій*

затверджена наказом по університету від «05» лютого 2019 р. №0237-III

2 Термін здачі студентом закінченого проекту «_13__» ___травня___ 2019 р.

3 Вхідні дані до проекту Дані пацієнта, дані анамнезу та очікування пацієнта щодо антибіотико терапії.

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) Аналіз г інфекцій, реалізація Інформаційна технологія підтримки прийняття рішень при діагностуванні гострих респіраторних інфекцій

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Презентація до дипломної роботи із слайдів

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання _____.

Керівник _____

(підпис)

Завдання прийняв до виконання _____

(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів випускної проекту	Термін виконання етапів проекту	Примітка
1	Дослідження предметної області	01.02 – 17.02.19	
3	Аналіз функціональних вимог	27.02 – 29.02.19	
4	Аналіз нефункціональних. вимог	30.02 – 04.03.19	
5	Розробка дизайну сайту	08.03 – 15.03.19	
6	Розробка бази даних	15.03. – 25.03.19	
7	Розробка інформаційної системи	26.03 – 15.04.19	
8	Текстове наповнення сайту	16.04 – 18.04.19	
9	Функціональне тестування	19.05 – 23.04.19	
10	Нефункціональне тестування	23.06 – 25.04.19	
11	Налагодження сайту	25.07 – 30.04.19	

Магістрант _____ Бабенко А.В.

Керівник роботи _____ к.т.н., доц. Шендрик В.В.

РЕФЕРАТ

Тема магістерської роботи: «Інформаційна технологія підтримки прийняття рішень при діагностуванні гострих респіраторних інфекцій».

Пояснювальна записка містить вступ, 4 розділи, висновки, додатки та список літератури, включає 87 сторінку, 16 таблиць, 24 ілюстрації, 54 джерел та 5 формул.

В першому розділі наведений огляд актуальності напрямку дослідження. Визначається аналіз сучасного стану застосування інформаційних технологій для вирішення задач в обраній предметній області, визначаються наявні переваги та недоліки систем-аналогів, обґрунтовується актуальність роботи.

Другий розділ включає в себе формулювання мети, задач та вибору методів дослідження. Крім того, виконується планування робіт для реалізації проекту.

Третій розділ призначений для опису поетапного проектування інформаційної системи та розробки архітектури системи. В даному розділі наведений структурно-функціональний аналіз проекту, описані основні принципи роботи розробленої інформаційної системи.

У четвертому розділі описується розробка інформаційної системи, що є результатом практичного застосування запропонованих методологій дослідження. Детально наводяться етапи розробки та приклади використання функцій системи.

Результатом проведеної роботи є розроблена інформаційна технологія підтримки прийняття рішень при діагностуванні гострих респіраторних інфекцій.

Ключові слова: веб-розробка, сайт, гострі респіраторні інфекції, діагностика ГРІ, багаторівнева система розпізнавання.

ЗМІСТ

ВСТУП

У медичних установах все більш актуальним стає створення автоматизованих робочих місць, наявність яких робить роботу лікаря-фахівця більш продуктивною. Тому клінічні системи підтримки прийняття рішень (СППР) стали невід'ємною складовою сучасних інформаційних технологій охорони здоров'я. Вони допомагають при інтерпретації, діагностиці та лікуванні. СППР можуть бути вбудованими по всьому спектру безпеки пацієнтів, надаючи нагадування, рекомендації та попередження для медичних працівників [1].

Актуальність. Гострі респіраторні інфекції (ГРІ) – найбільш поширені інфекційні хвороби, що вражають усі вікові групи населення. Експерти Всесвітньої організації охорони здоров’я відзначають, що в останні роки ГРІ мають постійну тенденцію до збільшення та посідають перше місце серед причин тимчасової втрати працездатності – навіть у міжепідемічний період на них хворіє 1/6 частина населення планети. В Україні щорічно на ГРІ хворіють 10–14 млн осіб, що становить 25–30% усієї та близько 75–90% інфекційної захворюваності в Україні. Це призводить до значних економічних збитків та зумовлює актуальність ГРІ [2].

Оскільки симптоматика ГРІ досить неоднозначна, динамічна та приводить до складності в ідентифікації інфекції лікарем, виникає необхідність створення інформаційної технології підтримки прийняття рішень, яка дозволяє діагностувати ГРІ за симптомами та передбачає подальше анкетування хворого для уточнення діагнозу та моніторингу ефективності лікування.

За опрацьованими даними було виявлено відсутність аналогів у вигляді автоматичного програмного продукту, який дозволяє діагностувати ГРІ за симптомами та передбачає подальше анкетування щодо зміни симптомів та загального стану хворого.

Метою роботи є розробка інформаційної технології підтримки прийняття рішень при діагностуванні ГРІ з можливістю анкетування пацієнта для уточнення діагнозу.

Об’єкт дослідження – є процес діагностування гострих респіраторних інфекції.

Предмет дослідження – є інформаційне забезпечення з діагностування захворювань.

Задачами дослідження є розробка інформаційного технології підтримки прийняття рішень з діагностування ГРІ. Дана робота потребує вирішити наступні задачі:

- дослідити актуальність проблеми;
- ознайомитися з процесом прийняття рішення щодо діагностування ГРІ лікарем;
- огляд сучасних СППР в медицині;
- створити математичну модель для діагностування захворювання;
- розробити модель бази даних;
- спроектувати та розробити СППР;
- провести тестування СППР.

Наукова новизна роботи полягає в удосконаленні процесу діагностування ГРІ шляхом розроблення нової інформаційної технології діагностування ГРІ.

Практична цінність – запропонована інформаційна технологія підтримки прийняття рішень дозволить діагностувати ГРІ за симптомами та уточнити зміни симптомів та загального стану хворого за допомогою анкетування.

Впровадження даної ІТ дозволить у значній мірі спростити та прискорити процес постановки діагнозу лікарем, а також мінімізувати кількості лікарських помилок щодо виявлення гострих респіраторних інфекцій та призначення антибіотикотерапії.

Апробація: Науково-технічна конференція «Інформатика, математика, автоматика :: 2019», 23-26 квітня 2019 р. [3].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Системи підтримки прийняття рішень у медицині

Сьогодні поняття підтримка прийняття рішень це процес розпізнавання ситуації, де потрібно прийняти рішення та визначити мету, спланувати та сформувати способи її реалізації, обрати варіант з використанням експертних знань та методів математичного програмування, змоделювати наслідків прийнятих рішень для їх оцінки. В якості рішення розглядається варіант тієї чи іншої дії, а вирішення ситуації прийняття рішення – знаходження варіанту рішення із застосуванням СППР, налаштованої на предметну область. Для реалізації СППР використовують інформаційні технології, які забезпечують процес прийняття рішення на всіх його етапах [4].

СППР в клінічній медицині можуть виконувати такі функції:

- диференціальна діагностика та вибір лікування;
- підготовка рішень незалежно від вираженості клінічних проявів хвороби, що передбачає діагностику при ранніх формах захворювань та при умові стертої клінічній картинки;
- облік фонових станів (супутніх захворювань) пацієнта, що особливо важливо при підборі лікування;
- аналіз динаміки патологічного процесу з прогнозом потенційно можливих несприятливих ситуацій (при обліку проведеної терапії, включаючи побічні ефекти медичних препаратів);
- оцінка стану в режимі «реального» часу, що може бути досягнуто при актуалізації за рахунок інформації, що надходить з приладових комплексів і персональних систем [5].

СППР дозволяють лікарю не тільки перевірити висунуту гіпотезу на користь обраного захворювання, а й звернутися за порадою в важких для прийняття рішення ситуаціях. Тому найчастіше СППР використовуються саме для допомоги при

постановці діагнозу, призначення і, при необхідності, коригування призначеного лікування.

Медичні СППР направлені на підтримку прийняття рішень при медичній діагностиці покриваються широкий спектр захворювань починаючи з офтальмологічних захворювань та закінчуючи генетичними захворюваннями, такими як різновиди хвороби Хантінгтона. У таблиці 1.1 наведена класифікація областей застосування медичних діагностичних СППР в залежності від типу методів розробки моделей СППР.

Таблиця 1.1 – Області застосування медичних діагностичних СППР

Метод	Області застосування
Дерево рішень	Діагностування серцевих захворювань [6]
Нечітка логіка	вад розвитку кортикального розвитку [7], діагностування ниркового каменю та ниркової інфекції [8], малярії [9], хвороб шлунково-кишкового тракту [10], хвороби Альцгеймера [11]
Нейронні мережі	Діагностуваня первинних головних болей [12], хвороб серця [13], захворювань щитовидної залози [14], хвороби Хантінгтона (генетичне захворювання нервової системи) [15], мітохондріальних захворюваннях [16], гострого нефриту [17], хвороби серця [18], раку молочної залози, серцевої аритмії, дерматологічних захворювань і хвороби Паркінсона [19], офтальмологічних захворювань [20]

З таблиці видно, що штучні нейронні мережі (ШНМ) являються найбільш поширеним методом підтримки прийняття рішень при діагностуванні різних видів захворювань.

1.2 Алгоритм постановки діагнозу

Для прийняття діагностичних рішень лікарю потрібно враховувати безліч факторів: симптоми захворювань, їх форми, патогенез, клінічні прояви з урахуванням індивідуальних особливостей пацієнтів. Все це тримати в голові та приймати безпомилкові рішення стає все складніше. Неперервно з'являються нові знання, тоді як час на прийняття лікарем відповідного рішення не збільшується. Як результат, збільшується кількість лікарських помилок, які в деяких країнах доходять до 30% [21].

Процес виявлення ГРІ лікарем виконується відповідно до вимог Міністерства охорони здоров'я України згідно алгоритму [2], що представлений на рис. 1.1.

У наш час діагностичний процес, вироблений лікарем, починається зі збору даних для прийняття рішення. Тому вхідними даними для прийняття рішення при діагностуванні ГРІ є дані про пацієнта, збір анамнезу та очікування пацієнта щодо антибіотикотерапії та дані про алергії пацієнта, що наведені у таблиці 1.2. Дані пацієнта такі як номер пацієнта, вік, попередні хвороби зберігаються у медичній картці пацієнта, які зберігаються в паперовому вигляді, оскільки електронні медичні картки ще не впроваджено. В наш час такий підхід збереження та обробки даних застарілий та недостатньо додає інформацію, оскільки складно розібрати почерк лікарів та знайти дату останнього прояву певних хвороб. Все це призводить до значних незручностей у роботі сімейних лікарів та в певній мірі ускладнює процес постановки діагнозу. Тому очевидно, що для оптимізації та підвищення ефективності роботи лікаря сучасні ІТ в медицині вимагають постійного оновлення даних, зберігання даних пацієнтів у достатньо відкритому вигляді, уніфікації особистих карток пацієнтів. Це зумовлює створення бази даних (БД) в рамках даної СППР для структурованого збереження даних пацієнтів.

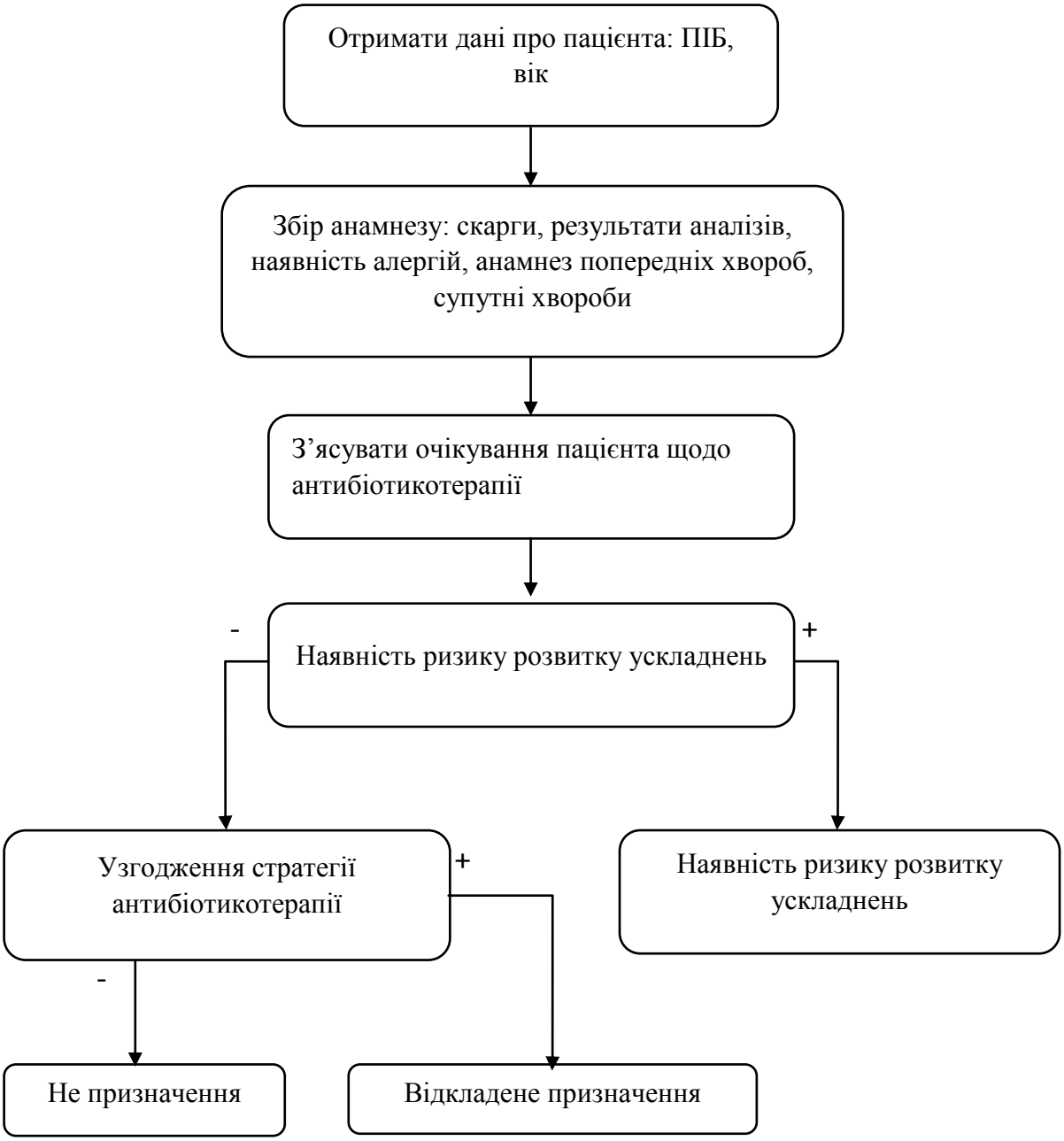

Рисунок 1.1 – Алгоритм дії лікаря

Оскільки все більше людей відмовляються від прийому антибіотиків, щоб зменшити проблему резистентності до них, не менш важливим етапом серед дій лікаря є з'ясування очікувань пацієнта щодо антибіотикотерапії та виявлення наявності у пацієнта алергій на медичні препарати.

Таблиця 1.2 – Вхідні дані для прийняття рішення з діагностування ГРІ

Назва	Джерело
Якісні	
Скарги	Пацієнт
Попередні хвороби	Карта пацієнта
Кількісні	
ID пацієнта	Карта пацієнта
Вік	Карта пацієнта
Альтернативні	
Погодження або відмова атибіотикотерапії	Пацієнт
Наявність алергій на препарати	Пацієнт

Рішення щодо вибору ймовірного захворювання формується на основі протоколів ГРІ та досвіду лікарю. Дані про хвороби, їх відповідні симптоми та додаткові фактори впливу представлених у базі даних. На виході система повинна надати список ймовірних захворювань та відповідні рекомендації щодо призначення антибіотиків.

1.3 Огляд сучасних систем підтримки прийняття рішень в медицині

У наш час широко розвиваються СППР в медицині:

- «Доктор Елекс» - це медична інформаційна система, яка розроблена для автоматизації всіх ключових позицій роботи сучасної клініки: реєстратури, лікаря, лабораторії, діагностики, звітності, управління.
- - «Мусin» - це експертна система, призначена для роботи в області діагностики і лікування. Система прогнозує ймовірний діагноз на основі зазначених їй симптомів, та надає рекомендації щодо курсу медикаментозного лікування діагностованих захворювань [22].

- IndiGO (Archimedes) – це система, що забезпечує обробку різної природи даних клінічної, фізіологічної природи та відомостей про управління процесом лікування та формує індивідуалізовані протоколи для діагностики та лікування з урахуванням факторів ризику, історії хвороби, відомостей про отриманий лікуванні, біомаркерів і т.п. по конкретному пацієнту [23].
- Auminence - система диференціальної діагностики, аналізує відомості про симптоми і ін. інформацію, виявляє значущі патерни та формує діагностичний план.
- СППР Isabel Healthcare - система підтримки прийняття діагностичних рішень на основі відомостей про симптоми, використовує веб-інтерфейс, що підтримує процес аналізу симптомів у процесі диференціальної діагностики. [24].
- VisualDx-система підтримки діагностичних рішень з використанням принципів диференціальної діагностики. Володіє рядом унікальних особливостей, зокрема, найбільшої в світі бібліотекою медичних зображень [25].
- OncoFinder (Онкофайндер) – програма для аналізу внутрішньо-клітинних сигнальних шляхів та підбору найбільш підходящих терапевтичних препаратів при проявах різних типів раку. Аналізуючи індивідуальні особливості пацієнта, система підбирає йому найбільш адекватний тип терапії препаратами [26].

Для діагностування захворювань часто використовують математичну статистику. Прикладом такої системи є робота Комлевої О.О. Система підтримки прийняття рішень під час діагностування бронхіту за допомогою методів математичної статистики. В даній роботі було розглянуто модуль DiaBronchitis підтримки прийняття рішень при діагностуванні бронхіту. Також описано алгоритм для формування діагностичних ознак з використанням критерію Стьюдента та з використанням алгоритму перевірки вигляду розподілу даних з ціллю визначення можливості їх використання в системі DiaBronchitis [27].

Іншим досить поширеним методом класифікації різного типу захворювань є метод дерев рішень. Даний метод являє собою ефективний інструмент обробки медико-біологічних даних. Для створення СППР часто використовують дерева рішень у випадку діагностики пацієнтів із захворюваннями серця. Це дослідження досліджує застосування низки методів для різних типів дерев рішень, які шукають

кращу продуктивність при діагностиці серцевих захворювань. У цьому дослідженні використовується широко використовуваний набір еталонних даних [28].

Оскільки медична інформація є неповною, суперечливою або неточною, ефективним являється використання методів з використанням нечіткої логіки:

- Локтюхин В. Н., Черепнин А. А., Підтримка прийняття рішень на основі нейро-нечіткої технології при діагностиці захворювань шлунково кишкового тракту. [29].
- Крашений І. Е., Метод аналізу томографічних зображень мозку на основі нечіткої логіки для діагностування хвороби Альцгеймера. В даній роботі представлено новій алгоритм синтезу систем нечіткого логічного висновку для діагностування хвороби Альцгеймера, що дозволяє обирати систему з максимальною ефективністю на основі кривої помилок [30]
- Алайон С., Робертсон Р., Варфилд С., Руиз-Алзола Д., Медична система для діагностування вад розвитку кортикального розвитку. У даній роботі запропонована система нечітких правил. Система збирає наявні експертні знання про кортикальні вади та допомагає медичному спостерігачеві отримати правильний діагноз. Крім того, система дозволяє вивчати вплив різних факторів, які беруть участь у прийнятті рішення [31].

У результаті був проведений аналіз предметної області та визначено які дослідження проводилися з діагностики ГРІ. Аналіз предметної області показав, що на даний момент в медицині широко розповсюджені СППР, які дають можливість приймати рішення при діагностуванні різних захворювань. Але усі вище наведені СППР мають недоліки: багато систем є комерційними продуктами та вартість їх використання та підтримки – занадто велика, а також є необхідність у складному налаштуванні систем під потреби конкретної організації охорони здоров'я. За опрацьованими даними було виявлено відсутність аналогів у вигляді автоматичного програмного продукту, який дозволяє діагностувати саме ГРІ за симптомами та передбачає надання рекомендацій щодо антибіотикотерапії.

2 ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1 Мета та задачі дослідження

Метою роботи є підвищення ефективності роботи сімейного лікаря шляхом створення, впровадження та використання інформаційної технології підтримки прийняття рішень при діагностуванні. Тобто, потрібно створити систему, яка допоможе користувачу виявити найбільш ймовірне захворювання.

Було визначено актуальність розроблюваної системи, досліджено аналоги на аналізі яких можна визначати перші функціональні вимоги, саме вони визначають те, як система повинна працювати. Отже до функціональних вимог [32] до системи належить те, що вона повинна:

– мати розмежований доступ для звичайних користувачів та адміністратора системи, під профілем користувача розуміється особистий профіль лікаря;

– забезпечити в профілі адміністратора редагування даних пацієнтів, хвороб, симптомів та додаткових факторів, а також створення нових користувачів та редагування їх даних.

– забезпечити можливість введення симптомів та відповідних коефіцієнтів, введення додаткових факторів для подальшої підтримки прийняття рішення при діагностуванні ГРІ;

– забезпечити можливість надання підтримки прийняття рішень щодо діагностування ГРІ у вигляді;

– забезпечити можливість введення даних результату опитування пацієнта щодо антибіотикотерапії, а саме про підтвердження або відмову антибіотикотерапії та про наявність алергій на препарати;

– забезпечити можливість надання рекомендацій щодо антибіотикотерапії;

– забезпечити можливість додавання особистих карток пацієнтів з наступною інформацією: ім’я, прізвище, дата народження, стать, адреса;

– забезпечити можливість редагування та видалення даних пацієнтів;

До нефункціональних вимог [33] до системи належить те, що вона повинна:

- бути представлення у вигляді веб-орієнтованої ІС;
- має бути зручним та зрозумілим у використанні.

Для розробки інформаційного забезпечення потрібно обрати методи та інструменти, провести планування робіт, а саме: ідентифікувати мету проекту методом SMART, визначити план змісту робіт, побудувати календарний план та визначити ризики проекту.

Наступним етапом потрібно спроектувати СППР, що включає збір вимог, проектування моделі бази даних та інтерфейсу. Завершальними стадіями проекту будуть реалізація інтерфейсу, основного функціоналу СППР та бази даних, з подальшим тестуванням та розробленням супровідної документації.

2.2 Аналіз методів дослідження

Одним з найвідоміших методів математичної статистики, який використовується для вирішення завдання медичної діагностики, є метод Байеса [34]. Суть методу Байеса полягає в наступному: на основі наявних апріорних ймовірностей захворювань та заданих умовних ймовірностей прояву симптомів для пацієнта проводиться розрахунок ймовірності захворювань [35]. У медичній діагностиці зустрічаються випадки, для яких апріорні ймовірності для будь-яких об’єктів не відомі. В такому випадку використання Баєсовської стратегії неможливо [36].

Для діагностування захворювань часто використовують математичну статистику. Використання методів математичної статистики зобов'язує виконувати попередній аналіз даних, на основі яких виконується подальша діагностика. Перевагою методів, заснованих на математичній обробці даних, є спроба структурувати інформацію про відомі захворювання.

До недоліків статистичних методів відносяться наступні:

1. Цей клас методів орієнтований на великі обсяги інформації, проте, навіть при її наявності вона часто може бути не систематизована, що унеможливлює або обмежує роботу статичних методів.

2. Так для накопичення, обробки та зберігання інформації потрібні обчислювальні та тимчасові затрати.

3. Поява нової інформації веде до редагування моделі та перерахунку наявних ймовірностей.

4. В системах, побудованих на математичної статистики, результати діагностики виводяться на основі математичних операцій, які можуть кардинально відрізнятися від звичайних дій та логіки лікаря-терапевта. У зв'язку з цим відсутня можливість пояснення прийнятого рішення, що не надає можливість організувати роботу механізму пояснення рішень в експертних системах.

5. Відсутня можливість враховувати всю складність об'єктів діагностики та те, як одні чинники можуть впливати на інші.

Сьогодні НМ стали досить поширеними серед систем діагностування у медицині, оскільки реалізація СППР, використовуючи НМ, істотно зменшує витрату часу на створення та значно знижує вимоги до експертів в розглянутій предметній області [37]. Рішення задач отримується на підставі навчання на обраних прикладах. Не зважаючи на суттєві переваги, НМ мають декілька недоліків: використовується повільний та ітераційний процес навчання, необхідно ініціалізувати ваги для входів, задати параметри та топологію мережі.

З огляду на різноманіття та нечітку природу ознак ГРІ, коли один і той же симптом може відноситися до різних захворювань, важливо використовувати підхід, який враховує дані конкретні умови - нечітку логіку. Системи нечіткої логіки чудові в поводженні з такою неоднозначною і неточною інформацією, поширеною в медичній діагностиці. Методи основані на нечіткій логіці мають ряд переваг [38]:

– дозволяють описувати якісні та кількісні показники у лінгвістичної формі;

– надають можливість вичислювати значення ступенів приналежності об’єктів до нечіткій множин, враховуючи концепцію невизначеності;

Однак системи на основі нечіткої логіки мають досить складну архітектура, навіть при використанні ієрархічної бази знань. Для того щоб забезпечити точність рішень, потрібно збільшувати кількість термів у лінгвістичних змінних, що значно ускладнюють роботу з діагностичною системою.

У медицині широкого пізнання набув метод, тісно згруповані послідовності подій, або випадків хвороби, або інших пов'язаних зі здоров'ям феноменів з ясно окресленим розподілом образу у часі, або в просторі, або в часі і просторі. Термін правильно використовується для опису агрегації щодо нечастих подій або хвороб, наприклад, лейкозів, розсіяного склерозу. Ідея розділення на кластери заснована на гіпотезі про компактність образів у просторі ознак. Завдання розпізнавання та класифікації образів являються типовими для штучних нейронних. Архітектура такої мережі залежить від наявності та якості навчальної вибірки.

Можно виділити найпоширеніші архітектури:

- багатошарова мережа прямого поширення, у якій використовують зворетнє поширення помилки для навчання. При кожному навчальному прикладі проводиться налаштування вагових коефіцієнтів, це зумовлює велику кількість епох для отримання локального мінімуму. Пошук глобального мінімуму похибки є метою навчання.
- двошарова мережа, яка містить шар радіально-базисних нейронів, де проводиться формування рішення. У синапсах одного з нейронів зберігається один обраний образ. Персептрон складає другий шар мережі. В якості ступеня впевненості чи ймовірності виявлення об’єкта певного класу є виходи нейронів другого шару.

2.3 Математичний метод підтримки прийняття рішення з діагностування ГРІ

Медичний діагноз зазвичай передбачає ретельне обстеження хворого, щоб перевірити наявність та вплив деяких ознак, які мають відношення до ймовірного підозрюваного захворювання для подальшого прийняття рішення. Симптом такий

як нежить, може буди сильним для одного пацієнта, але водночас бути помірним або навіть легким для іншого. В такому випадку досвід лікаря допомагає поєднати набір симптомів так, щоб з'ясувати правильне діагностичне рішення.

При використанні персептрона для аналізу ознак на всіх рівнях, формування багаторівневої системи розпізнавання включає ряд етапів:

Eman 1. Множина кластерів позначається $C = \{c_z, z = \overline{1, Z}\}$. У роботі виділено декілька кластерів, а саме пансинусит, гострий пансинусит, ларинготрахеїт Множина ознак представлена як дві множини:

$$S =< SM, SC > \quad (2.1)$$

де SM- множина ознак, що вказують на певний кластер (Жар пітливість, озноб, безсоння, слабкість у тілі та інші) $SM = \{sm_m, m = \overline{1, M}\}$;

SC – підмножина ознак, яка розділяє кластер на мікрокластери: $SC = \{sc_n, n = \overline{1, N}\}$.

Ядро кластеру позначено D_{sm}, тому у межах одного кластера ознаки підмножини SC повинні мати мінімальну дисперсію D_{sc}, що вказує на те, що об’єкти одного класу з такими ознаками майже не відрізняються. Дана підмножина ознак використовується при ідентифікації кластера на першому рівні розпізнавання.

Eman 2. Для аналізу усіх ознак на розподільчу властивість сформовано таблиці-відношення (табл.2.1), де рядки – це об’єкти, стовпці – ознаки.

Таблиця 2.1 – Фрагмент заповненої таблиці відношення

		Симптоми				
		Жар	Сильний кашель	Головний біль	Слабкість у тілі	Кашель
Захворювання	Пансинусит	1	0	1	1	0
	Хронічний пансинусит	1	0	1	1	0
	Ларинготрахеїт	1	0	0	0	1

	Гострий ларинготрахеїд	1	1	1	0	1

Результатом є відношення значень ознак до відповідних об’єктів (2.2):

$$R \subseteq H \times S \qquad (2.2)$$

Відповідність симптомів та захворювань відзначено таким чином, що 0 відповідає відсутності симптому, 1 його наявності. Таким чином, виходить матриця, що складається з векторів-симптомів:

$$\begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Для того, щоб розрізнювати образи за кожною з наявних ознак S_k, потрібно сформувати верхньотрикутні матриці DM_n, де всі елементи знизу від головної діагоналі дорівнюють нулю. Кількість матриць повинна відповідати кількості ознак та мати розмір $I \times I$.

Далі кожну ознаку S_n потрібно проаналізувати на наявність властивості до розрізнення образів та поставити відповідну сукупність пар об’єктів, що різняться n-ю ознакою.

Для кожної ознаки S_n потрібно визначити порогову величину, тобто границю, яка розрізнює об’єктів DT_n. Завдяки границі розділення можна виділити рядки в таблиці, де значення ознаки S_k помітно відрізняються. Якщо $\Delta s_{nij} \geq DT_n$, то ознака розрізнює об’єкти H_i та H_j. Якщо $\Delta s_{nij} < DT_n$, то дана ознака враховується на першому етапі до ядра кластера SC: $s_{kij} \in SC$. Під час тестування та налагодження системи є змога задати значення границь DT_n.

Для різних об’єктів порівнюються значення s_{ni} та s_{nj} у циклі обробки таблиці відношення. Якщо $\Delta s_{nij} > DT_n$, то результат заноситься до n-ї матриці розрізнення

DM_n у вигляді одиниць. На основі даних в матрицях DM_n можна визначити кількість ознак, необхідну для розпізнавання кожного захворювання.

Eman 3. Для того, щоб сформувати ядра кластерів потрібно в матрицях DM_n виділити позиції нулів та заповнити масив CF, призначений для урахування ознак з низькою розподільною здатністю. Для заповнення масиву CF потрібно визначити кількість пар об'єктів, що мають спільну ознаку. Якщо $\Delta s_{nij} < DT_n$, то ознака s_n має низьку розподільчу здатність.

Eman 4. Для точного розпізнавання захворювань потрібно виділити підмножини. Для цього спочатку перевіряється умова розрізнення, шляхом по елементної диз'юнкції матриць DM_k. У результаті отримаємо матрицю R, що показує розрізнення DT_k. Об'єкти, які неможливо розрізнити позначено нулями. Якщо об'єкти помітно схожі, то можливо корегувати значення DT_k.

Шляхом по елементною кон'юкцією матриць DM_k виконуємо пошук об'єктів, що різняться у визначеній множині ознак. У результаті отримаємо матрицю RY, що показує розрізнення у визначеній множині. У матриці знаходяться ознаки, що різняться для різних об'єктів.

У даному випадку такою парою об'єктів є ларинготрафеїд та гострий ларинготрафеїд, для якої виділяємо розрізнюючу ознаку «сильний кашель».

Деякі випадки такі, як пасинусит та хронічний пансинусит мають однакові симптоми, тому без додаткових факторів неможливо розрізнити дані об'єкти. Єдиний фактор, Для цього вводимо додаткові фактори ризику, що заповнюються аналогічно до симптомів, за виключенням вагових коефіцієнтів, які задані автоматично, тоді як симптоми оцінюються лікарем.

Далі для визначення підмножин створюємо таблицю RR, де кожен рядок – це об'єкт, що зіставлений відповідно зі стовпцем, тобто ознакою та позначений одиницею. Таблиця RR заповнюється за допомогою циклу обробки матриці RY. У разі повторень номерів ознак у рядку об'єкта записується одиниця у матрицю RR.

Eman 5. На даному етапі потрібно виділити кластери та мікрокластери, з'ясувати їх кількість та класифікувати. Далі потрібно сформувати перший шар на

основі таблиці *RR*, де вид ядерної функції активації множини SC входів для кожного кластеру (2.3):

$$f(x) = e^{-a\|x-c\|^{-2}}, \qquad (2.3)$$

де $\bar{c}$ – вектор центрів множини одновимірних радіально-симетричних функцій,

$\|x - \bar{c}\|$ – норма вектору відхилень вхідної змінної від центрів одновимірних функцій, що розташовані на координатах простору ознак,

a – параметр зв'язаний з радіусом розсіювання вхідних змінних r відношенням: $a = \frac{1}{2r^2}$

Евклідова відстань використовується для розрахунку норми вектору (2.4) [39].

$$\|x - \bar{c}\| = \sqrt{(x_1 - c_1)^2 + (x_2 - c_2)^2 + \cdots + (x_m - c_m)^2} \qquad (2.4)$$

Вихід кожного нейрона першого шару активізує окремий блок другого шару нейронної мережі, що розпізнає образи на рівні мікрокластерів.

Для формування другого шару нейронної мережі було використано метод опорних векторів, де кількість розпізнавальних нейронів *I* з'єднаних синапсисами зі своїми підмножинами ознак, вказаних у таблиці *RR*.

Висновок визначається як відсоткове значення для кожного підозрюваного захворювання.

Дана система має два функціональні блоки, де перший блок розпізнає кластери, а другий блок розпізнає мікрокластери всередині кластера. Зв'язок виходу d_i показує ступінь ймовірності правильного розпізнавання образу h_i з ознаками *so*, *sc*, що вказано у виразі:

$$d_i(so, sc) = \psi(\sum_{n=1}^{N_i} w_n\, so_n + e^{-a\|sc_n - c\|^{-2}}) \qquad (4)$$

де w_n – ваги нейронів другого шару,

so_n – ступінь впевненості (значення ознак, що розділяють мікрокластери),

sc_n – ступінь впевненості (значення ознак, що розділяють кластери),

N_i – кількість ознак, які розрізняють образ h_i,

$\psi(*)$– активаційна характеристика нейрона вихідного шару.

Навчання даної системи проводиться в три етапи:

Eman 1. Формування підмножини ознак кожного кластера, де координати об’єктів навчальної вибірки використовуються для формування ядерної функції (2.4) для кожного кластеру.

Eman 2. Формування прикладів для навчання та тестування нейронів другого шару. Кожен приклад складається з двох масивів даних:

- масив значень ступеня впевненості в наявності ознак із підмножини SO. Масив заповнений значенням зміщення, що були отримані від ядерного нейрона першого шару, який розпізнає відповідний кластер;
- масив шаблонних значень виходів нейронів другого шару.

Eman 3. Навчання нейронів другого шару з використанням правила Уідроу-Хоффа [40]

2.4 Вибір засобів реалізації

Для розробки ІТ, описаної в даній роботі, була обрана технологія об'єктно-орієнтованого програмування (ООП). Даний підхід до написання програм ґрунтується на об'єктах, а не на функціях і процедурах, тобто головними є об'єкти, а не дії, дані, а не логіка. ООП спрощує процес організації та створення структури програми, надає можливість вносити зміни об’єкту без впливу на інші частини програми. Так як з плином часу програми стають все більшими, а їх підтримка все більш важкою, дані аспекти ООП стають все більш актуальними [41].

Для розробки основних функціональних модулів підтримки прийняття рішень при діагностуванні ГРІ та надання рекомендацій щодо антибіотикотерапії була обрана мова об'єктно-орієнтованого програмування Python, яка володіє високою гнучкістю та динамічністю [42], а також набором бібліотек для створення

нейронних мереж. Для створення та навчання нейронної мережі для підтримки прийняття рішень при діагностуванні ГРІ з використанням інтерактивної оболонки Atom [43]. Atom має повний доступ до файлової системи, природні для операційної системи меню та панель команд, при цьому ідеально пристосований для веб-програмування: можливість додавати власні функції для редагування CSS, HTML і JavaScript.

Для розробки інтерфейсу ІТ обрано мову JavaScript, а саме бібліотеку React, яка в останні роки набула особливої популярність, використовуючи так званий компонентний підхід, де компонент - це одиниця, з якої збирається інтерфейс. Компоненти можна перевикористати, успадковувати один від одного, компонувати, що забезпечує

Також для розробки бази даних проекту було обрано система управління базами даних MySQL. MySQL є гарним рішенням для створення малих і середніх додатків [44]. Можна виділити такі можливості сервера MySQL:

- простота у встановленні та використанні;
- підтримка великої кількості користувачів, які одночасно працюють з базою даних.
- можливість зберігати великі обсяги інформації
- висока швидкість виконання команд;
- підтримка вкладених запитів [45].

Комбінування обраних методів дослідження та технологій забезпечить ефективний розподіл ресурсів (часу), що необхідно для досягнення поставленої мети у визначені терміни.

3 МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕННЯ З ДІАГНОСТУВАННЯ ГРІ

3.1 Інформаційна технологія підтримки прийняття рішень при діагностуванні гострих респіраторних інфекцій

Процес підтримки прийняття рішень при діагностуванні ГРІ та наданні рекомендацій щодо антибіотикотерапії складається з декількох етапів:

Етап 1. Введення даних

На даному етапі вводяться загальні дані пацієнта з особистої медичної картки, що зберігається в БД. При першому візиті пацієнта, створюється особиста карта та зберігається в БД системи. Після введення даних пацієнта, вводяться дані анамнезу, що являють собою набір симптомів. Симптоми обираються з існуючих в БД та відповідно кожному симптому виставляється оцінка, тобто ступінь впливу симптому на постановку діагнозу. Оцінки виставляються користувачем (лікарем), спираючись на його досвід. Далі для уточнення постановки діагнозу ГРІ вносяться додаткові фактори впливу (Паління, алергії та інші), які впливають на пошук діагностичного рішення.

Етап 2. Кластеризація даних

За допомогою методу описаного в розділі 2.3 виконується кластеризація об’єктів. Кластеризація виконується за допомогою двошарової нейронної мережі. Дана система має два функціональні блоки, де перший блок розпізнає кластери, а другий блок розпізнає мікрокластери всередині кластера. На входи мережа отримує набір симптомів та відповідні вагові коефіцієнти. Приклад структури мережі показаний на рис. 3.1. Для розуміння роботи системи показані 2 ядра для 2 кластерів та 2 персептрона другого шару, які на рівні мікрокластера розпізнають захворювання.

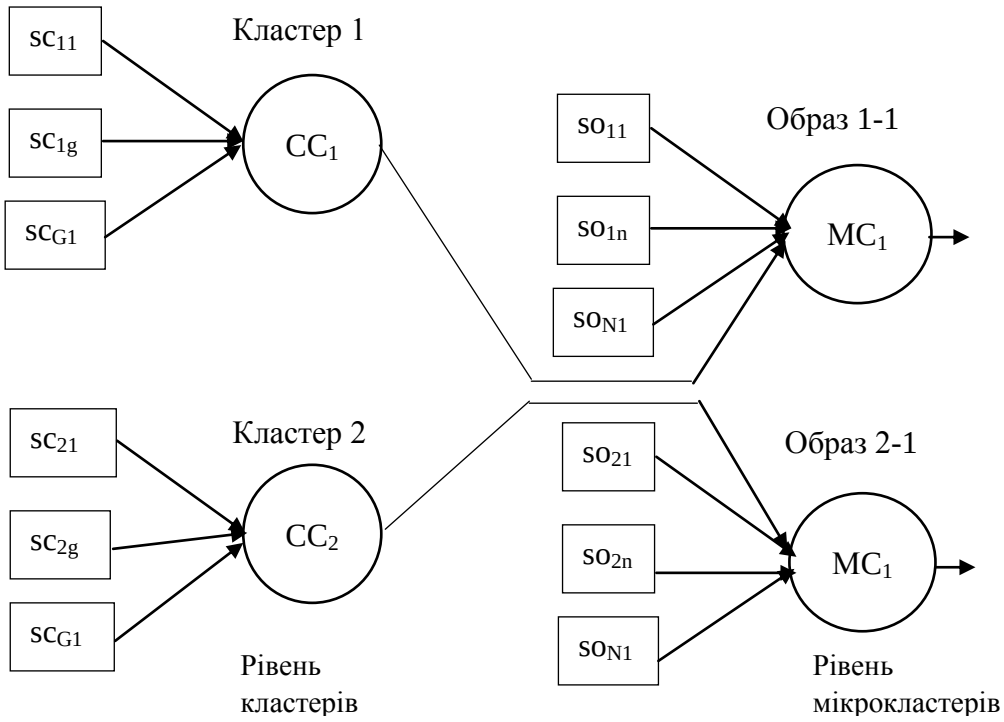

Рисунок 3.1 – Структура модулю діагностування ГРІ

Етап 3. Визначення рекомендацій щодо антибіотикотерапії.

Рекомендації щодо призначення антибіотикотерапії визначаються за допомогою методу дерева рішень. Дерево рішень модулю надання рекомендацій щодо антибіотикотерапії наведений на рисунку 3.2. Дане дерево має три функціональні вузи: підтверження або відмова антибіотикотерапії, наявність алергічних реакцій на медичні препарати та навність ускладнень. Також дерево має 4 листки:

- Не призначення антибіотикотерапії при відмові пацієнта;
- Направлення на аналіз на чутливість до медичних препаратів, а саме антибіотиків. Дана дія виконується при умовах, що пацієнт погодився на антибіотикотерапію та повідомив про наявність алергій.
- Відкладене призначення антибіотикотерапії виконується за умов , що пацієнт погодився на приймати антибіотики, не має алергій на препарати та відсутні ускладнення.

- Негайне призначення антибіотикотерапії виконується за умов , що пацієнт погодився на приймати антибіотики, не має алергій на препарати, але має ускладнення.

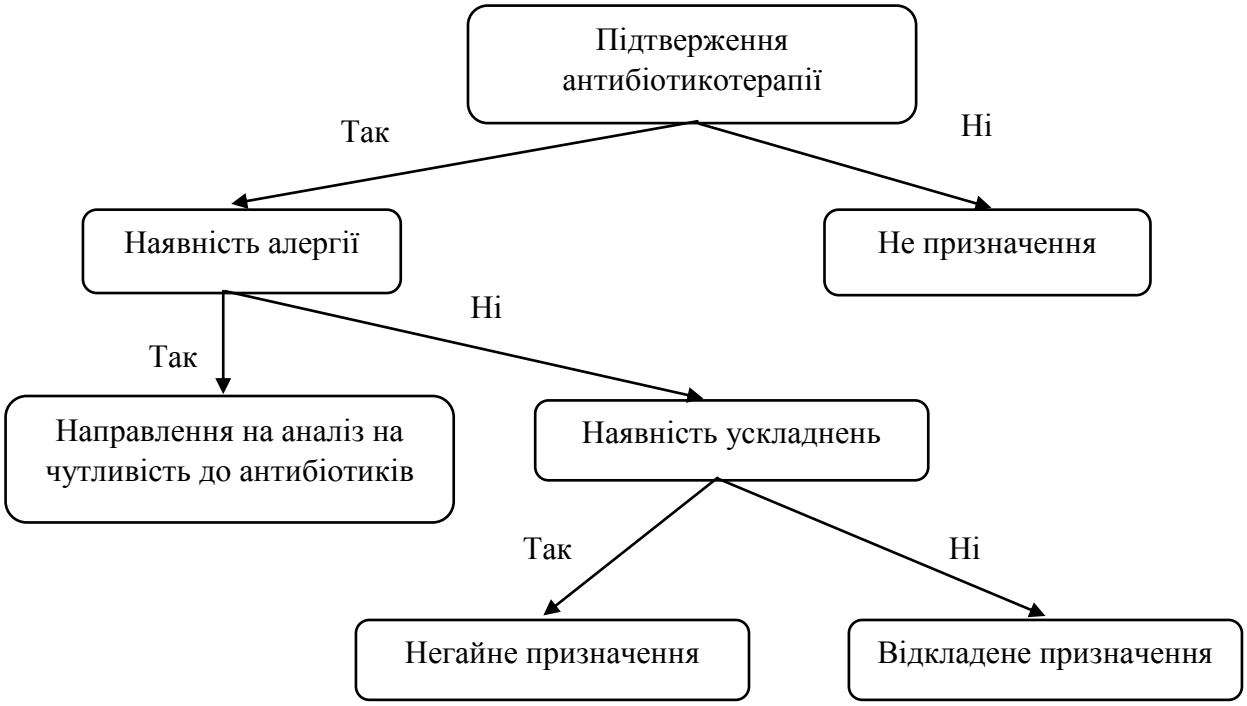

Рисунок 3.2 – Дерево рішень для надання рекомендацій щодо антибіотикотерапії

Eman 4. Збереження даних

Зі списку ймовірних захворювань обирається один діагноз, який на думку лікаря є наймовірнішим, та зберігається у особистій картці пацієнта у БД, для подальшого врахування даного захворювання при наступному прийомі у лікаря. Оскільки часті випадки захворювання свідчать про хронічну його стадію, збереження результату діагностування у особистій електронній картці користувача сприяє ефективному подальшому діагностуванню ГРІ.

3.2 IDEF0 діаграма

За допомогою IDEF0 діаграми показано логіку основних процесів в ІС [46] (рис. 3.1). Особливістю є те, що акцент у даній нотацій робиться на взаємодію компонентів, а не на послідовність подій у системі [47].

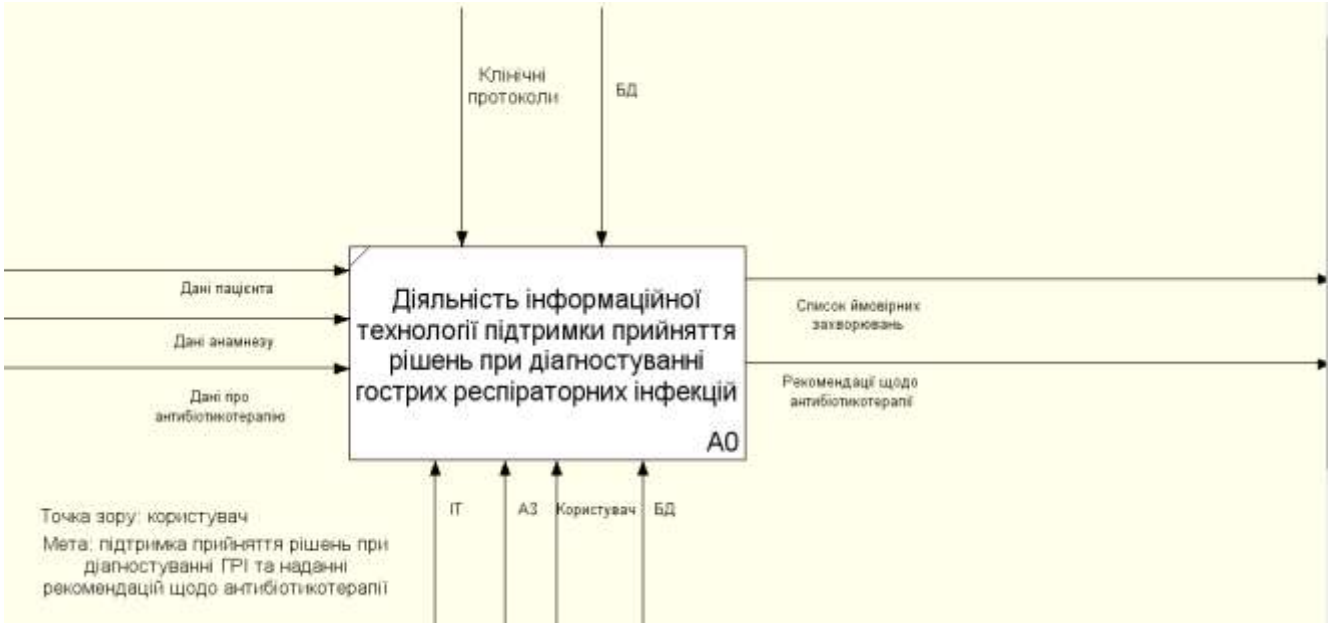

Рисунок 3.1 – IDEF0 діаграма

З діаграми видно, що система отримує на вході особисті дані пацієнта, список симптомів та передбачення щодо антибіотикотерапії, а на виході видає список можливих діагнозів та певні рекомендації користувачам щодо антибіотикотерапії. На декомпозиції першого рівня (рис. 3.2) продемонстровано, що спочатку відбувається обробка вхідних даних з їх предствленням у системі, потім відбувається обробка активностей доданих користувачем до міні-профілю тварини, а потім відбува-ються дії щодо обробки активності візиту до лікаря, вона додається до профілю адміністратора з можливістю відповіді, щодо неї на особисту пошту користувача.

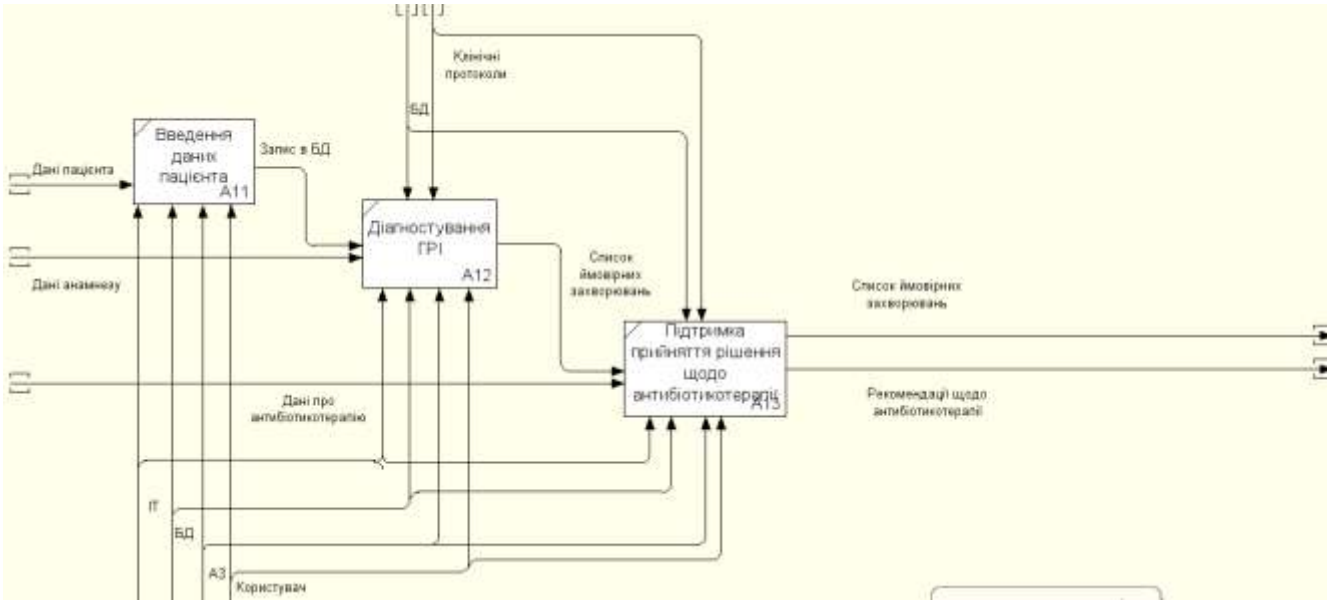

Рисунок 3.2 – Декомпозиція IDEF0 діаграма

Наведені діаграми розроблені з урахуванням точки зору звичайного користувача (лікаря) на компоненти ІС. Ціль у даному випадку полягає у підтримки прийняття рішень при діагностуванні ГРІ та наданні рекомендацій щодо антибіотикотерапії.

3.3 Схема бази даних

Для підтримки прийняття рішень сімейним лікарем необхідно зберігати інформацію у структурованому вигляді, тобто у БД. База даних для розроблюваної ІТ представлена на рис. 3.2. [48].

Для збереження даних було створено наступні таблиці:

- Users – інформація про користувачів.
- Patient – інформація про пацієнтів.
- Disease – інформація про хвороби.
- Symptom – інформація про симптоми.
- Add_Info – інформація про додаткові фактори ризику.

- Symptom_Disease – об’єднуюча таблиця, містить інформацію про відношення симптомів до хвороб.
- Add_Info_Disease – об’єднуюча таблиця, містить інформацію про відношення додаткових факторів ризику до хвороб.
- Patient_Disease – об’єднуюча таблиця, містить інформацію про пацієнтів та відповідних захворювань.

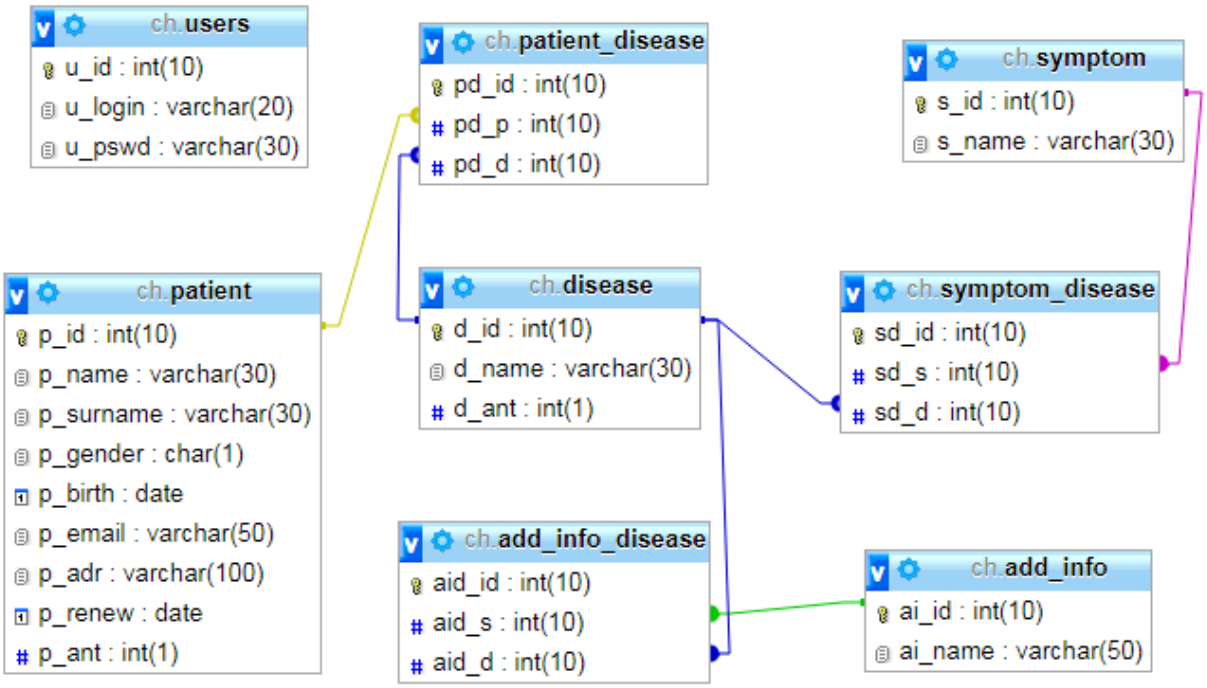

Рисунок 3.2 – Схема бази даних

В таблиці 3.1 наведений фрагмент таблиці «Users».

Таблиця 3.1 – Фрагмент таблиці «Users»

ID користувача	Логін	Пароль
1	admin	111
2	drmom	dr1

В таблиці 3.2 наведений фрагмент таблиці «Patient».

Таблиця 3.2 – Фрагмент таблиці «Patient»

ID пацієнта	1	2
Ім’я	Максим	Наталія
Прізвище	Теторенко	Московченко
Стать	м	Ж
Дата народження	12.12.1999	11.11.1996
e-mail	tetor@gmail.com	natamoskov@gmail.com
Адреса	м.Суми, вул. СКД 5, кв. 4	м.Суми, вул. Харківська 6, кв. 6
Дата оновлення даних	5.05.2019	12.05.2019
Список захворювань	1	4

В таблиці 3.3 наведений фрагмент таблиці «Disease».

Таблиця 3.3 – Фрагмент таблиці «Disease»

ID захворювання	Назва	Необхідність антибіотикотерапії
1	Гострий фарингіт	1
2	Гострий ларинготрахеїт	0

В таблиці 3.4 наведений фрагмент таблиці «Symptom».

Таблиця 3.4 – Фрагмент таблиці «Symptom»

ID захворювання	Назва
1	Нежить
2	Сухий кашель

В таблиці 3.5 наведений фрагмент таблиці «Add_Info».

Таблиця 3.5 – Фрагмент таблиці «Add_Info»

ID додаткового фактору впливу	Назва
1	Слабкий імунітет
2	Хронічні інфекції носоглотки

В таблиці 3.6 наведений фрагмент таблиці «Symptom_Disease».

Таблиця 3.6 – Фрагмент таблиці «Symptom_Disease»

ID відношення	ID захворювання	ID симптом
1	1	2
2	2	5

В таблиці 3.7 наведений фрагмент таблиці «Add_Info_Disease».

Таблиця 3.7 – Фрагмент таблиці «Add_Info_Disease»

ID відношення	ID захворювання	ID додаткового фактору впливу
1	2	1

2	2	2

В таблиці 3.8 наведений фрагмент таблиці «Patient_Disease».

Таблиця 3.8 – Фрагмент таблиці «Patient_Disease»

ID відношення	ID пацієнта	ID захворювання
1	4	3
2	5	2

Отже, у БД зберігаються картки пацієнтів, перелік захворювань та відповідні симптоми та додаткові фактори ризику, а також інформація про користувачів системи.

3.3 Use Case діаграма

Діаграма варіантів використання (рис. 3.3) показує функціональні можливості системи та взаємодію між акторами (користувачами) та компонентами системи [49]. Така діаграма моделює можливості системи у зрозумілому представленні.

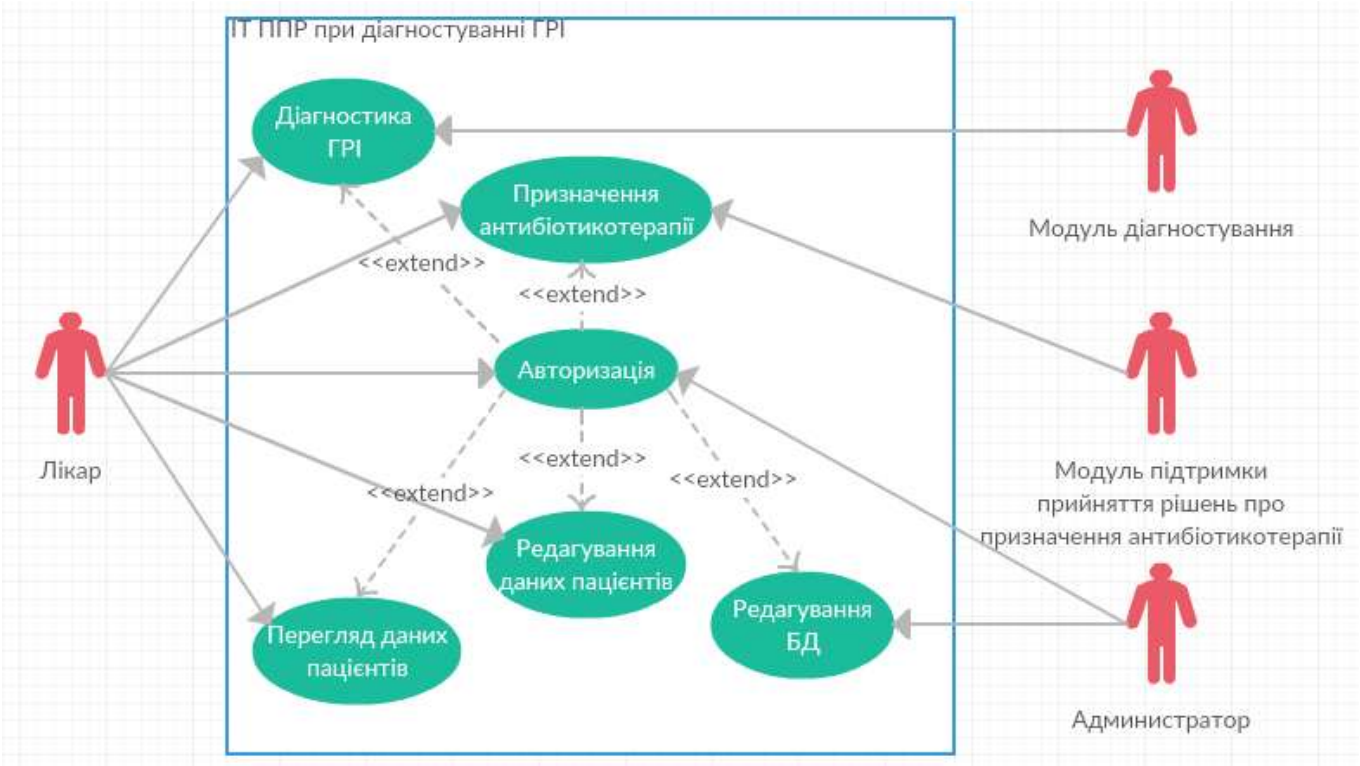

Рисунок 3.3 – Use Case діаграма

Отже, у ІС є розподіл ролей (розмежований доступ) на звичайного користувача та адміністратора системи. Крім того, певні функції доступні лише за певним доступом. Так звичайний користувач переглядає, додає та редагує особисті картки пацієнтів та може перейти до діагностування ГРІ та надання рекомендацій щодо антибіотикотерапії, у той час як адміністратор системи може редагувати дані пацієнтів, захворювань, симптомів, додаткових факторів та додавати нових користувачів системи.

4 РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ ПРИ ДІАГНОСТУВАННІ ГРІ

4.1 Головна сторінка лікаря

Привітальна сторінка (рис. 4.1) містить короткий опис додатку та має дві кнопки переходу до авторизації користувача.

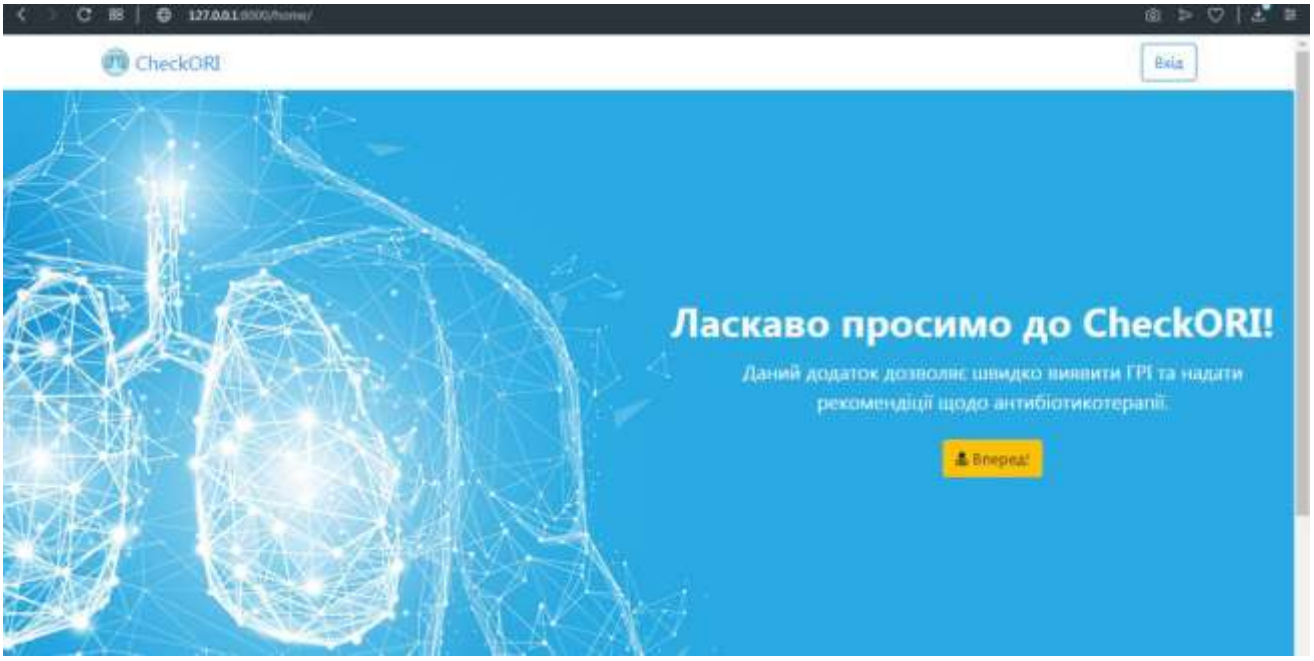

Рисунок 4.1 – Сторінка привітання

На рисунку 4.2 показано вікно авторизації користувача.

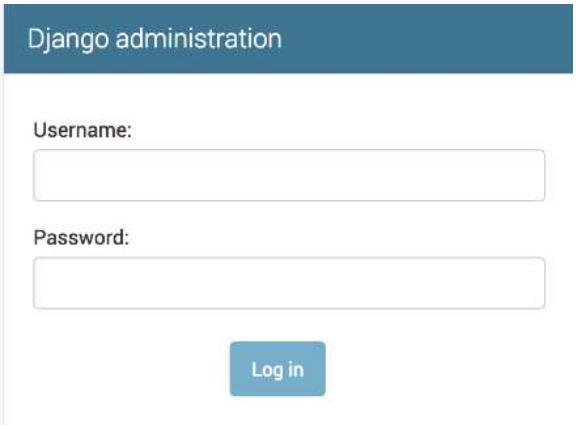

Рисунок 4.2 – Вікно авторизації

Головна сторінка лікаря (рис. 4.3) дозволяє йому переходити до діагностування ГРІ, переглянути список пацієнтів та додати нового пацієнта.

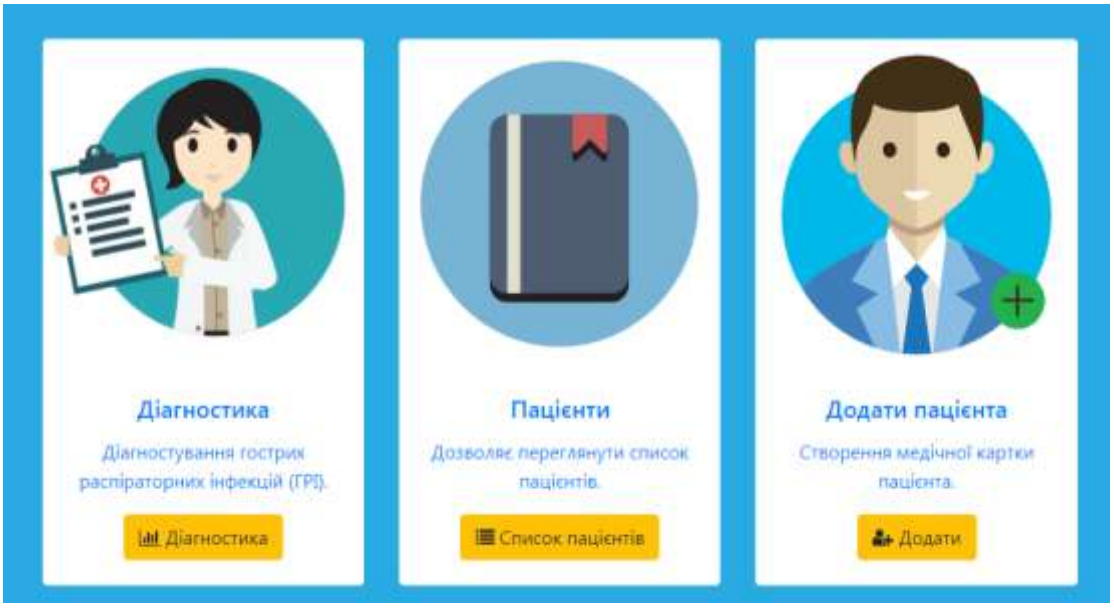

Рисунок 4.3 – Головна сторінка лікаря

Лікар може переглядати список пацієнтів, редагувати дані наявних пацієнтів та видаляти записи (рис.4.4).

Рисунок 4.4 – Перелік пацієнтів

4.2 Сторінка діагностування ГРІ

Процес діагностування складається з чотирьох етапів (рис. 4.5), є змога здійснювати перехід по кожному етапу.

Рисунок 4.5 – Етапи діагностування

Перший етап – додавання пацієнта. Якщо даного пацієнта не має в БД, то потрібно створити карту пацієнта (рис.4.6).

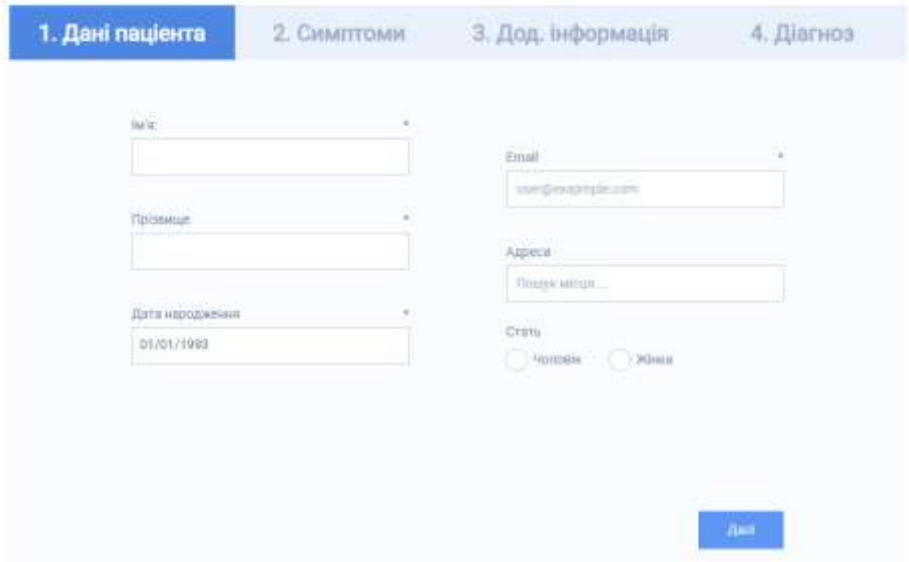

Рисунок 4.6 – Додавання даних нового пацієнта

Приклад введення даних пацієнта наведений на рис. 4.7.

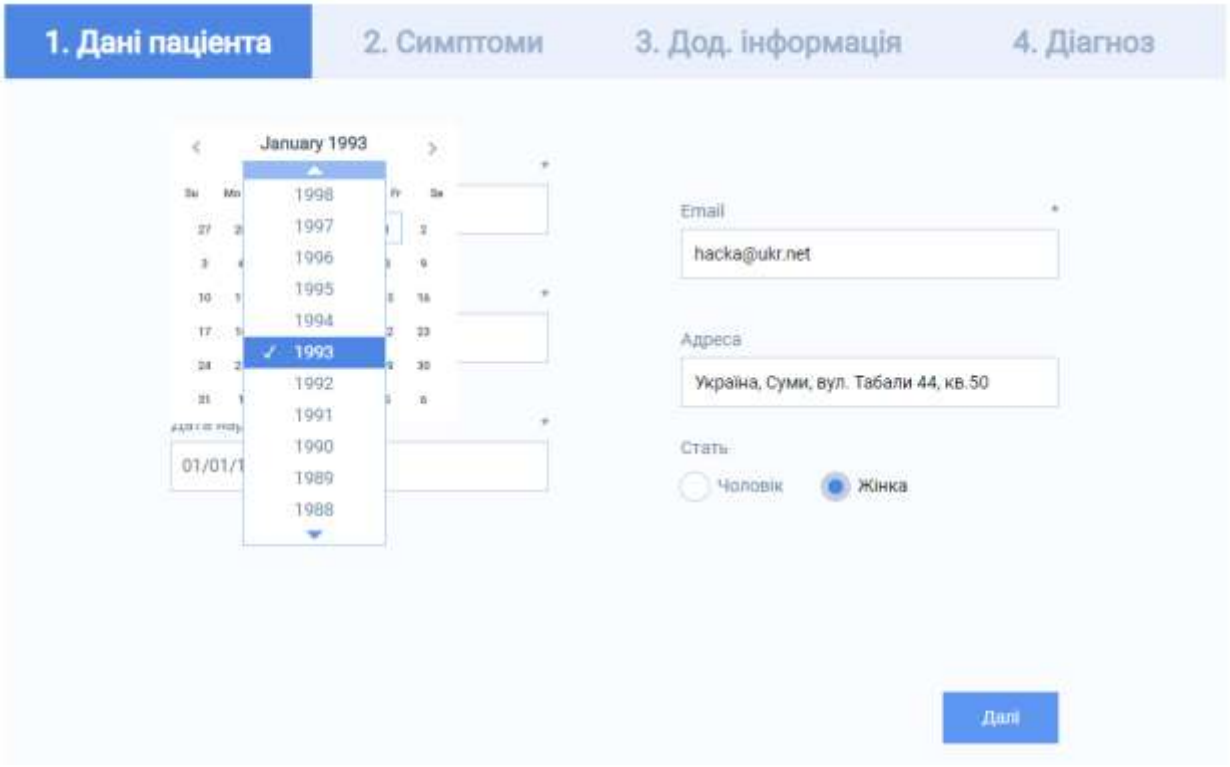

Рисунок 4.7 – Заповнення даних пацієнта

На другому етапі обираються симптоми хвороб та відповідно виставляються оцінки кожного симптому. Оцінка симптомів визначається за шкалою , представленою у вигляді ползункаю (рис.4.8).

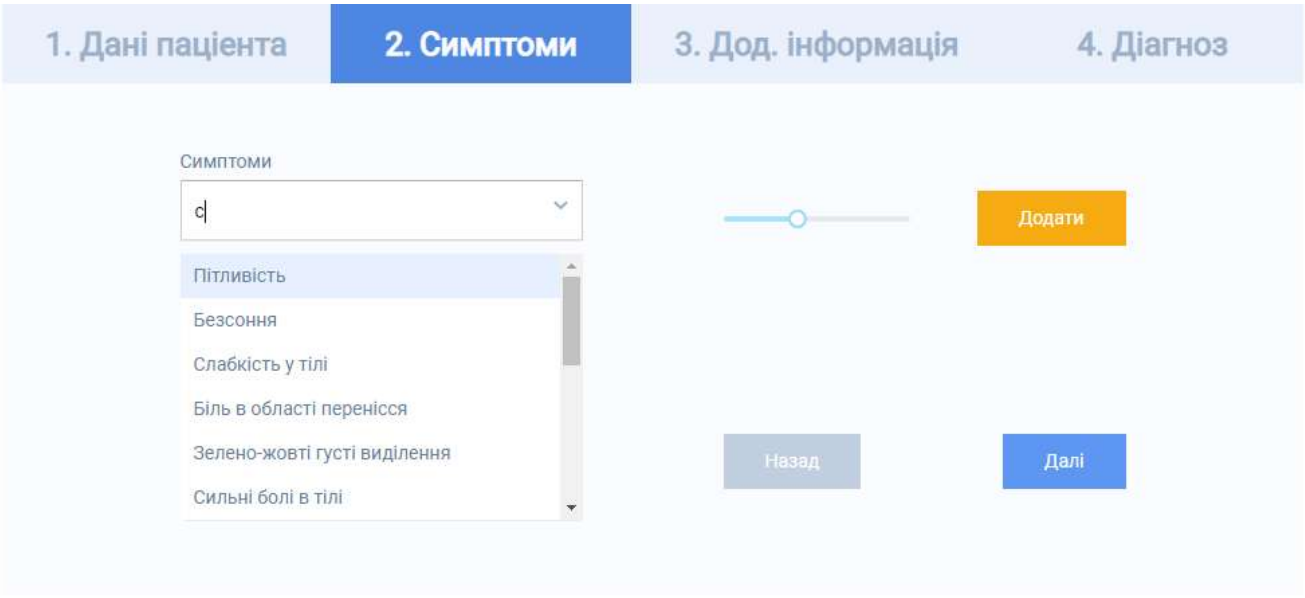

Рисунок 4.8 – Вікно додавання симптомів

Обрані симптоми формують список (рис.4.9), у якому відображаються всі обрані симптоми з виставленими відповідними оцінками. При некоректному введені оцінки симптому потрібно видалити симптом зі списку обраних симптомів та додати його знову. Після введення симптомів захворювання потрібно підтвердити введення, натиснувши кнопку «Далі».

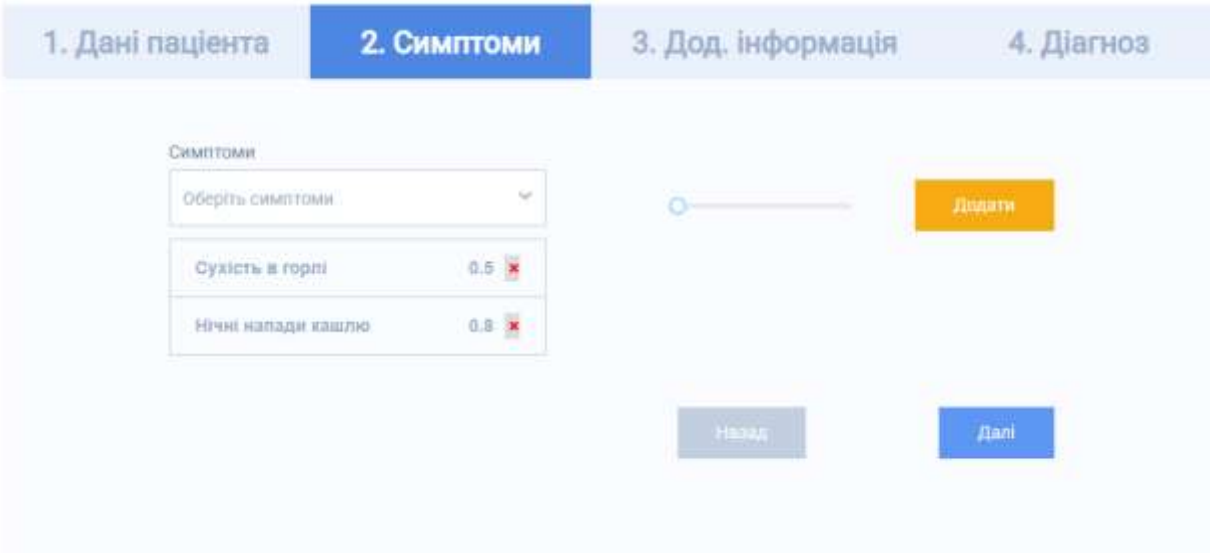

Рисунок 4.9 – Вікно додавання симптомів

На третьому етапі додаються додаткові фактори впливу та обирається передбачення щодо антибіотикотерапії (рис.4.10). Для отримання рекомендацій щодо антибіотикотерапії потрібно відповісти на поставлені питання про очікування пацієнта щодо прийняття ним антибіотиків та наявність алергій на медичні препарати, виконані у вигляді «radio button».

Далі для уточнення підтримки прийняття рішення при постановці діагнозу ГРІ потрібно обрати додаткові фактори ризику. Вибір додаткових факторів розроблений за допомогою елементу Multiselect.

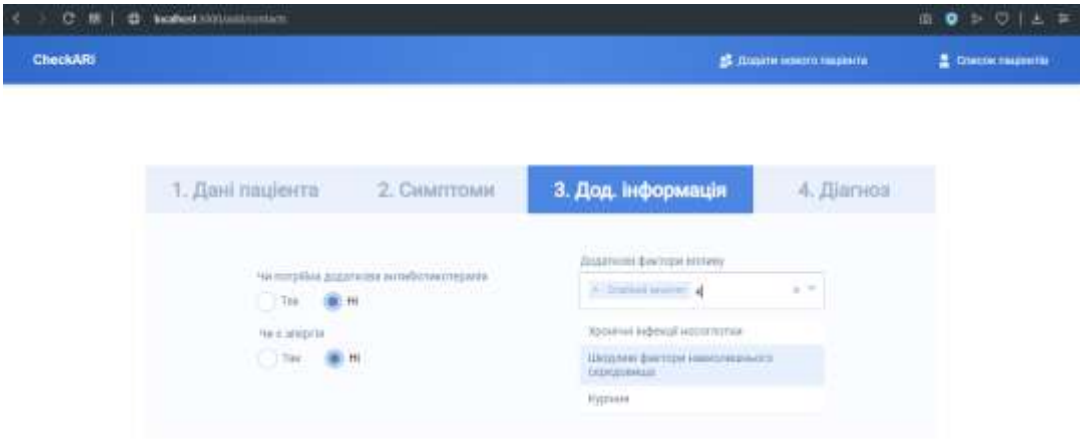

Рисунок 4.10 – Вікно додавання додаткових факторів впливу та вибору побажань щодо антибіотикотерапії

На четвертому етапі виводиться список можливих захворювань та рекомендації щодо антибіотикотерапії (рис.4.11).

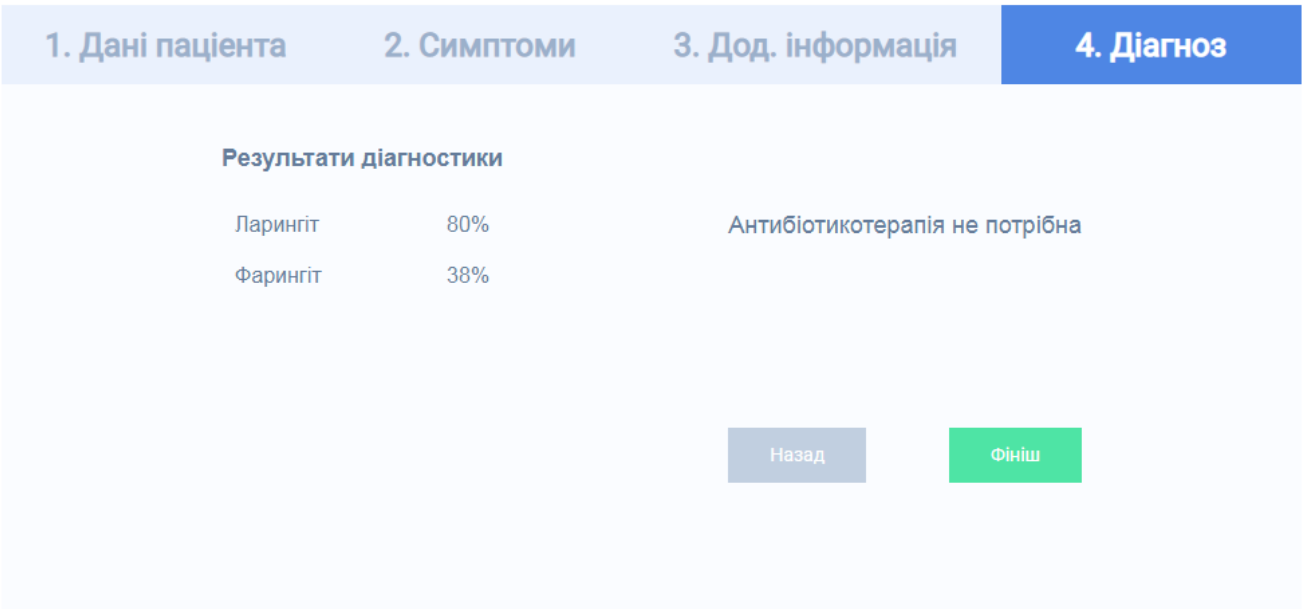

Рисунок 4.11 – Вікно діагнозу

Отже, за допомогою даної ІТ лікар, покроково зоповнюючи дані, отримує підтримку прийняття рішень при діагностуванні ГРІ у вигляді списку ймовірних захворювань та рекомендації щодо антибіотикотерапії.

4.3 Профіль адміністратора системи

Адміністратор має змогу редагувати дані таблиць БД, а саме додавати, редагувати та видаляти захворювання, відповідні симптоми та додаткові фактори ризику. (рис.4.12).

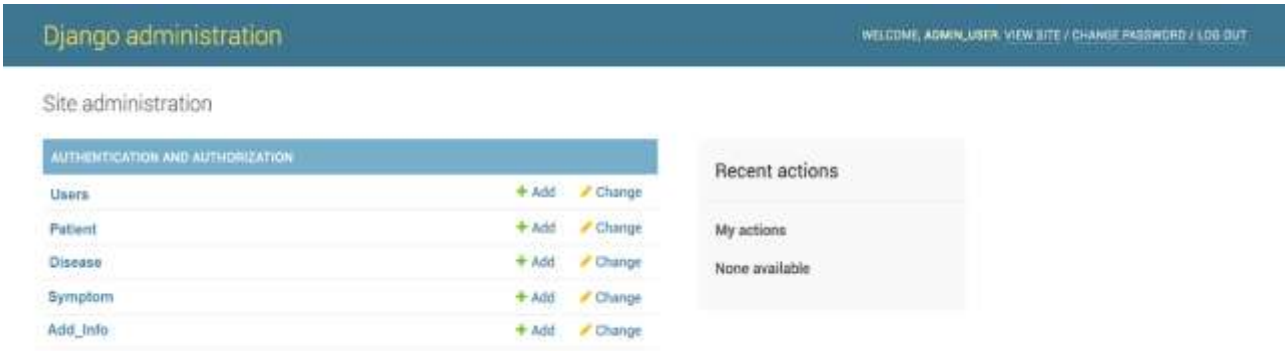

Рисунок 4.12 – Кабінет адміністратора

Адміністратор системи має можливість додавати нових користувачів (лікарів), приклад створення нового користувача наведений на рис.4.13.

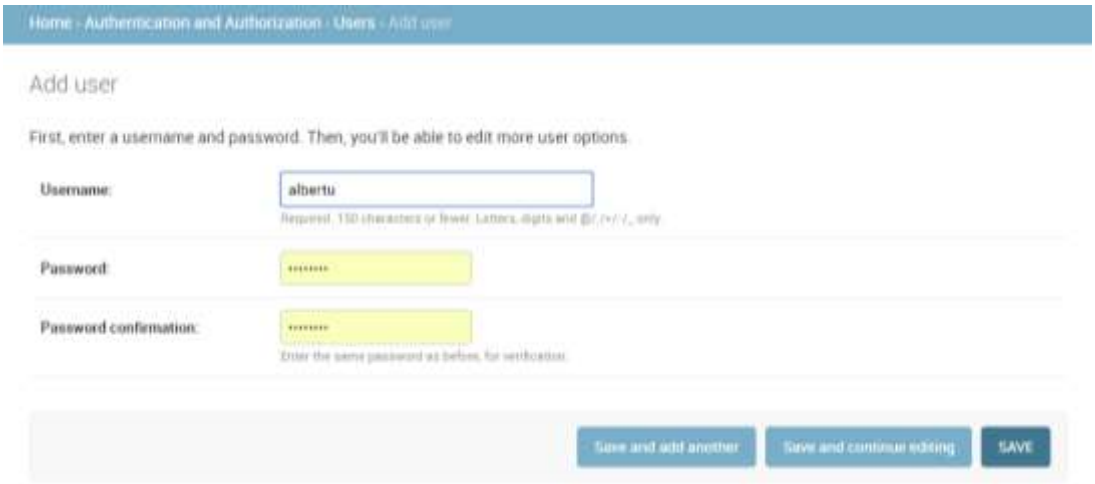

Рисунок 4.13 – Додавання користувача

Розроблена ІТ надає підтримку прийняття рішення при діагностуванні ГРІ та надає рекомендації щодо подільшої антибіотикотерапії., що в значній мірі дозволить спростити та прискорити процес постановки діагнозу лікарем, а також мінімізувати ризики лікарських помилок..

ВИСНОВКИ

У дипломній роботі було визначено актуальність використання інформаційних технологій для діагностування гострих респіраторних захворювань, досліджено аналоги на аналізі яких було визначено функціональні та нефункціональні вимоги. Також було здійснено планування, що включає в себе планування робіт, складання бюджету та визначення ризиків проекту.

Крім того, було обрано метод синтезу багаторівневої архітектури нейронної мережі для вирішення задач кластеризації для створення даної діагностичної системи, де є кластери та мікрокластери. Також було побудовано дерево рішень для підтримки прийняття рішення щодо подальшого призначення антибіотиків. Отже, розроблена модель багаторівневої нейронної мережі є простим та ефективним засобом для вирішення практичного завдання діагностики гострих респіраторних інфекцій.

Проаналізувавши існуючі аналоги та визначивши їх очевидні недоліки, було визначено, що існуючі системи не повною мірою задовольняють потреби кінцевого споживача. Крім того, з огляду на швидкоплинний розвиток технологій, доцільно постійно оновлювати та вдосконалювати існуючі системи. Усі ці причини є безсумнівним доказом актуальності розробки інформаційної технології підтримки прийняття рішень при діагностуванні ГРІ. Вагомим аргументом є також те, що вузькоспеціалізовані системи орієнтовані на виявлення ГРІ відсутні цілковито.

Результатом є розроблена інформаційна технологія діагностики гострих респіраторних інфекцій, яка в значній мірі підвищить ефективність роботи сімейного лікаря шляхом надання підтримки прийняття рішень при діагностуванні ГРІ та рекомендацій щодо подальшої антибіотикотерапії.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Yang Gong, Rohit Agrawal, Leming Zhou, Tahereh Saheb, Kam Cheong Wong, and Robert Robinson. Reasons For Physicians Not Adopting Clinical Decision Support Systems: Critical Analysis // JMIR Med Inform — 2018 Apr-Jun [Електронний ресурс]. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/ PMC5932331/

2. Гострі респіраторні інфекції. Уніфікований клінічний протокол первинної медичної допомоги дорослим та дітям.

3. Збірник IMA 2019

4. Сітнікова О. А., Почебут М. В., Розробка системи підтримки прийняття рішень для багатопрофільної медичної допомоги, Вісник НТУ «ХПІ», 2015

5. Г.Г. Асєєв, Концепція систем підтримки прийняття рішень

6. Mudasir Manzoor Kirmani, Syed Immamul Ansarullah, Prediction of Heart Disease using Decision Tree a Data Mining Technique, India [Електронний ресурс] . URL: https://pdfs.semanticscholar.org/ebec/6bc7fea03a81ae504702d89d52519b42d47a.pdf

7. Silvia Alayón, Richard Robertson, Simon K. Warfield, and Juan Ruiz-Alzola, A Fuzzy System for Helping Medical Diagnosis of Malformations of Cortical Development, 2007 [Електронний ресурс]. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2099697/

8. Farzad Firouzi Jahantigh, Kidney Diseases Diagnosis by Using Fuzzy Logic, 2015. [Електронний ресурс]. URL: http://ieomsociety.org/ieom_2015 /papers/416.pdf

9. James Ben Hayfron-Acquah, Joseph Kobina Panford, Designing Algorithm for Malaria Diagnosis using Fuzzy Logic for Treatment (AMDFLT) in Ghana, 2014. [Електронний ресурс]. URL: https://www.researchgate.net/publication/263030394_Designing_Algorithm_for_Malaria_Diagnosis_using_Fuzzy_Logic_for_Treatment_AMDFLT_in_Ghana

10. Song Weicai, a, Wu Yanxia, Application of Fuzzy Neural Network in Diagnosis of Gastrointestinal System Diseases, 2017.

11. Krashenyi I, Ramírez J, Popov A, Górriz JM, Fuzzy Computer-Aided Alzheimer's Disease Diagnosis Based on MRI Data, Kyiv, Ukraine, 2016. [Електронний ресурс] . URL: https://www.ncbi.nlm.nih.gov/pubmed/26971942

12. Karina Borges Mendes, Ronald Moura Fiuza, Maria Teresinha Arns Steiner, Diagnosis of Headache using Artificial Neural Networks, Brazil, 2010. [Електронний ресурс] . URL: https://pdfs.semanticscholar.org/d42e/12d7f7230177ee14881fd02777ecec11b300.pdf

13. Jae Kwon Kim and Sanggil Kang, Neural Network-Based Coronary Heart Disease Risk Prediction Using Feature Correlation Analysis, 2017. [Електронний ресурс] . URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5606055/

14. Astha Rastogi , Monika Bhalla, A Study of Neural Network in Diagnosis of Thyroid Disease, 2008. [Електронний ресурс] . URL: https://pdfs.semanticscholar.org/d2e5/1e850d5f23ae42688c5733fc4e0abe2259a4.pdf

15. B.W. Jervis ; M.R. Saatchi ; A. Lacey ; T. Roberts ; E.M. Allen ; N.R. Hudson ; S. Oke ; M. Grimsley, Artificial neural network and spectrum analysis methods for detecting brain diseases from the CNV response in the electroencephalogram, 1994. [Електронний ресурс]. URL: https://ieeexplore.ieee.org/document/331584

16. Inês Pimenta de Castro, L. Miguel Martins, Roberta Tufi, Mitochondrial quality control and neurological disease: an emerging connection, 2010. [Електронний ресурс]. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/ PMC2871738/

17. Ömer Yoldaş, Mesut Tez, Artificial neural networks in the diagnosis of acute appendicitis, 2011. [Електронний ресурс]. URL: https://www.researchgate.net/publication/51639593_Artificial_neural_networks_in_the_diagnosis_of_acute_appendicitis

18. Jose Rodriguez, Carlos Mendez, Maria Blanco, Salvador Tapia Breast Cancer Detection by Means of Artificial Neural Networks, 2017. [Електронний ресурс]. URL: https://www.intechopen.com/books/advanced-applications-for-artificial-neural-networks/breast-cancer-detection-by-means-of-artificial-neural-networks

19. David Gil, Magnus Johnson, Diagnosing Parkinson by using Artificial Neural Networks and Support Vector Machines, Spain. [Електронний ресурс]. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.182.7856&rep=rep1&type=pdf

20. Литвин А.А., Литвин В.А., Системы поддержки принятия решений в хирургии, 2014.

21. B.J. Copeland, MYCIN Artificial intelligence program [Електронний ресурс]. URL: https://www.britannica.com/technology/MYCIN

22. Программное обеспечение для определения рисков для пациентов INDIGO [Електронний ресурс]. URL: http://www.medicalexpo.ru/ prod/archimedes/product-81708-535035.html

23. Isabel Products [Електронний ресурс] . URL: https://www.isabelhealthcare.com/

24. Excellence in Clinical Decision-Making [Електронний ресурс]. URL: https://www.visualdx.com/

25. OncoFinder algorithm [Електронний ресурс] . URL: https://www.oncofinder.com/

26. Комлевої О.О. Система підтримки прийняття рішень під час діагностування бронхіту за допомогою методів математичної статистики.

27. Mai Shouman, Tim Turner, Rob Stocker, Using Decision Tree for Diagnosing Heart Disease Patients, 2011 [Електронний ресурс] . URL: https://www.semanticscholar.org/paper/Using-Decision-Tree-for-Diagnosing-Heart-Disease-Shouman-Turner/79e730959efb4e00f3b2fd162b3e0a1432caec59

28. Локтюхин В. Н., Черепнин А. А., Поддержка принятия решений на основе нейро-нечеткой технологии при диагностике заболеваний желудочно-кишечного тракта, 2009

29. Крашений І. Е., Метод аналізу томографічних зображень мозку на основі нечіткої логіки для діагностування хвороби Альцгеймера. Київ, 2017. [Електронний ресурс] . URL:http://ela.kpi.ua/jspui/bitstream/123456789/19688/ 1/Krashenyi_diss.pdf

30. Alayón S, Robertson R, Warfield SK, Ruiz-Alzola J., A fuzzy system for helping medical diagnosis of malformations of cortical development. 2007 [Електронний ресурс] . URL: https://www.ncbi.nlm.nih.gov/pmc/articles/ PMC2099697/

31. Functional Requirements Document [Електронний ресурс] . URL: https://www.tutorialspoint.com/business_analysis/business_analysis_functional_requirements_document.htm – Назва з екрану

32. Examples of Non-Functional Requirements [Електронний ресурс]. URL: https://simplicable.com/new/non-functional-requirements – Назва з екрану

33. Mark E. Glickman and David A. van Dyk, Basic Bayesian Methods [Електронний ресурс]. URL: http://www.glicko.net/research/glickman-vandyk.pdf

34. Правила вероятности [Електронний ресурс]. URL: http://statistica.ru/theory/pravila-veroyatnosti/ – Назва з екрану

35. Лорьер Ж.Л. Системы искусственного интеллекта. – М.: Мир, 1991. – 342 с.

36. Байдык Т.Н. Нейронные сети и задачи искусственного интеллекта. – К.: Наукова думка, 2001. – 263 с.

37. Zadeh L.A. Computing With Words. Principal Concepts and Ideas. – Berlin: Springer, 2012. – PP. pp–70.

38. Измерения в линейном пространстве [Електронний ресурс]. URL: http://twt.mpei.ac.ru/math/LARB/Euclidesp/LA_03040000.html – Назва з екрану

39. Искусственные нейронные сети [Електронний ресурс]. URL: http://www.kph.npu.edu.ua/!e-book/tpft/data/WOLG # 2/bookchap/ 978594807046.html – Назва з екрану

40. Ефименко И.В., Интеллектуальные системы поддержки принятия решений в медицине: ретроспективный обзор состояния исследований и разработок и перспективы, Москва, Россия [Електронний ресурс] . URL: http://docplayer.ru/80238763-Intellektualnye-sistemy-podderzhki-prinyatiya-resheniy-v-medicine-retrospektivnyy-obzor-sostoyaniya-issledovaniy-i-razrabotok-i-perspektivy.html

41. Advantages Of Python Over Other Programming Languages [Електронний ресурс]. URL: https://elearningindustry.com/advantages-of-python-programming-languages – Назва з екрану

42. Atom Documentation [Електронний ресурс]. URL: https://atom.io/docs

43. MySQL. [Електронний ресурс]. URL: https://www.mysql.com/ – Назва з екрану

44. Реляционные СУБД – сравнение MySQL и SQL сервер. [Електронний ресурс]. URL: https://www.hostinger.com.ua/ rukovodstva/reljacionnye-subd-sravnenie-mysql-i-sql-server/ – Назва з екрану

45. SMART [Електронний ресурс]: URL:: http://blog.trenings.ru/2011/10/smart

46. PMBOK - Work Breakdown Structure [Електронний ресурс]. URL: https://www.workbreakdownstructure.com/work-breakdown-structure-according-to-pmbok.php

47. Діаграма Ганта – ваш помічник у плануванні. Що таке діаграма Ганта і як її скласти? [Електронний ресурс]. URL: http://faqukr.ru/biznes/103951-diagrama-ganta-vash-pomichnik-u-planuvannishho.html

48. GanttProject [Електронний ресурс]. URL: https://www.ganttproject.biz/

49. Управление Рисками Проекта, PMBOK 2000

ДОДАТОК А. ПЛАНУВАННЯ РОБІТ

Ідентифікація мети ІТ-проекту методом SMART

Для деталізації мети був використаний метод SMART [45]. Результати деталізації розміщені у таблиці А.1.

Таблиця А.1– Деталізація мети методом SMART

Specific (конкретна)	Розробка Web-орієнтованої системи підтримки прийняття рішень з діагностування ГРІ та анкетуванням.
Measurable (вимірювана)	Результатом буде список можливих ГРІ та подальше анкетування для уточнення симптомів.
Achievable (досяжна, узгоджена)	Для досягнення мети необхідно обрати алгоритм діагностування ГРІ.
Relevant (реалістична)	Для успішної розробки ІТ забезпечено усі необхідні технології та засоби. Знання консультантів дозволяють досягти поставлену мету.
Time-framed (обмежена у часі)	Ціль має часове обмеження до 18 травня 2019 року.

Після проведення аналізу методом SMART можна визначити кінцеву мету. Мета проекту: розробка інформаційної технології підтримки прийняття рішень при діагностуванні ГРІ з можливістю анкетування пацієнта для уточнення діагнозу. Проект буде виконано вчасно, що підтверджується календарним планом проекту.

Планування змісту структури робіт IT-проекту

Структура декомпозиції робіт (WBS) у проектному менеджменті та системотехніці є орієнтованою на доконане виконання проекту декомпозицією проекту на менші частки. Структура декомпозиції робіт є ключовою часткою робіт по проекту, яка організовує командну роботу по проекту у керовані частини. Звід Знань по Проектному Менеджменту (PMBOK) визначає структуру декомпозиції робіт як "ієрархічну декомпозицію робіт, що має бути виконаною командою проекту та орієнтована на успішне завершення проекту" [4.

Елементом структури декомпозиції робіт може бути продукт, дані, Послуги або будь-яка комбінація вищезгаданого. Крім того, WBS надає необхідний каркас для детальної оцінки термінів та контролю, а також надає управління для розробки графіків робіт і контролю за їх виконанням.

Створена WBS-діаграма представлена на рисунку А.1.

Побудова календарного графіку виконання IT – проекту

Найпоширеніший формат графіка проекту в будь-якій галузі - це діаграма Ганта. Вона дозволяє менеджерам проекту і всій команді візуалізувати графіки часу і взаємозв'язок між окремими завданнями та етапами роботи над проектом [47].

Діаграма Ганта виконана у середовищі GanttProject [48]

] та зображена на рисунку А.2-3.

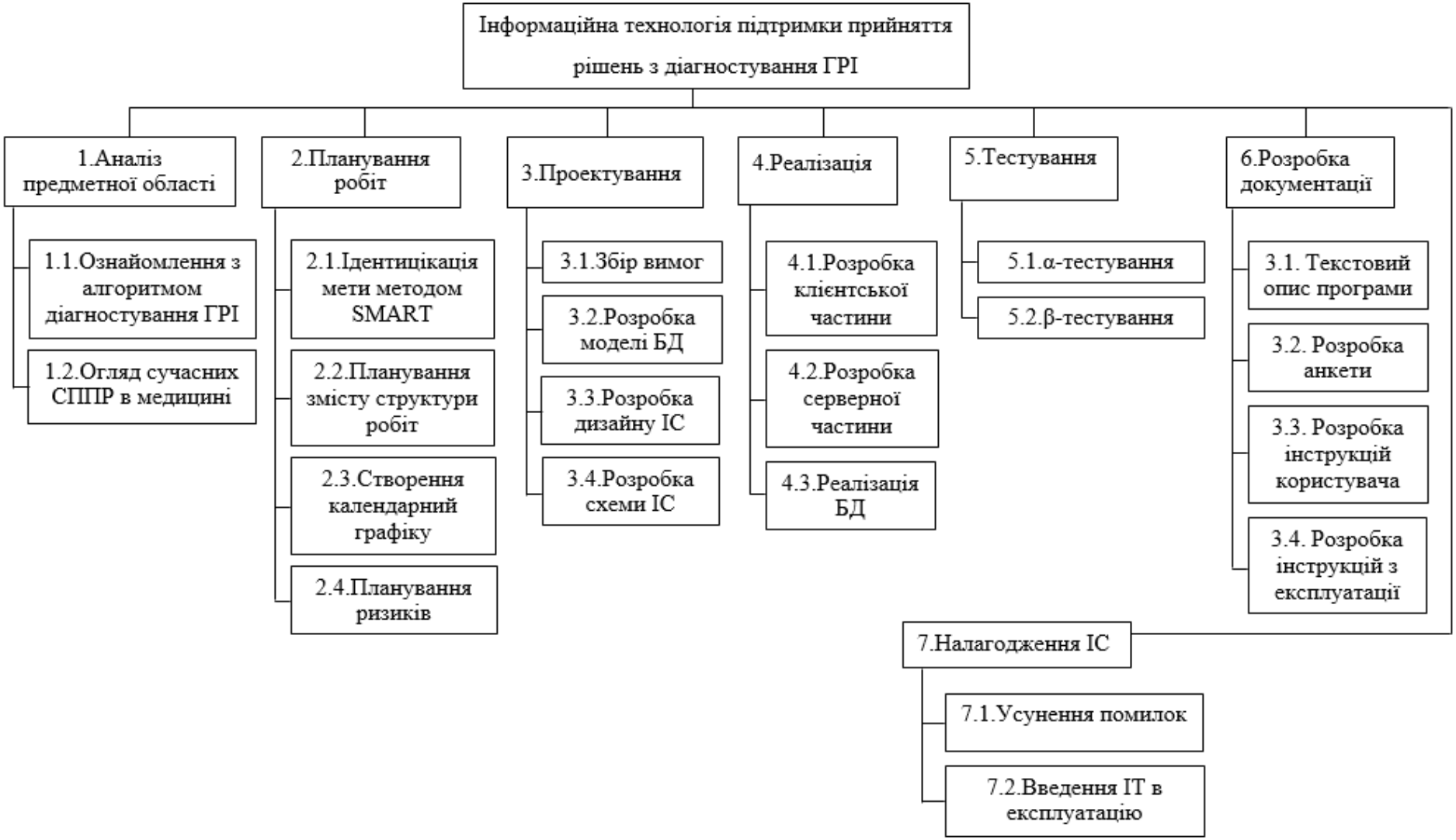

Рисунок А.1- WBS-діаграма

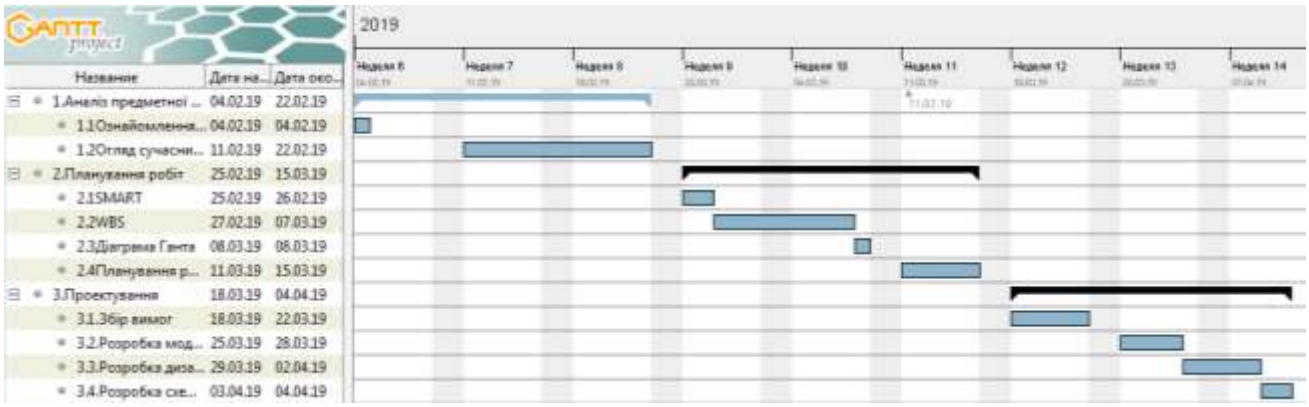

Рисунок А.2 – Фрагмент діаграми Ганта

Фрагмент діагами Ганта наведений на рисунку А.3.

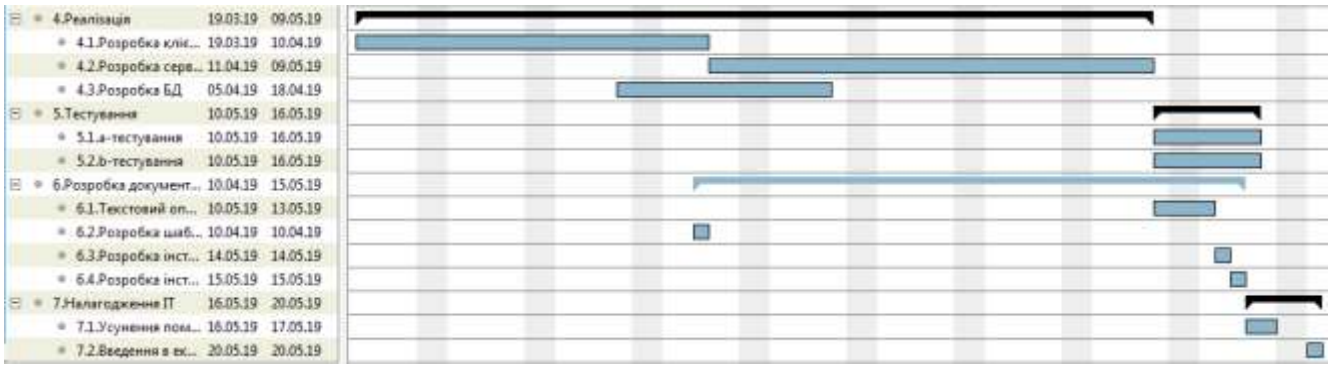

Рисунок А.3 – Фрагмент діаграми Ганта

Планування ризиків проекту

При розробці будь-якого проекту необхідно передбачити всі можливі ризики, які можуть трапитись та провести необхідні заходи для їх уникнення чи зменшенню збитків.

Управління ризиками проекту включає:

- Планування управління ризиками проекту;
- Ідентифікацію ризиків;
- Якісну і кількісну оцінку ризиків;
- Планування реагування на ризики, моніторинг і контроль.

Ідентифікація ризиків передбачає визначення ризиків, здатних вплинути на проект, і документальне оформлення їх характеристик.

Виділимо основні ризики під час розробки інформаційного забезпечення:

1. Часте внесення змін у ТЗ.
2. Неоптимальний розподіл часу.
3. Вибір неефективної технології розробки.
4. Збій апаратного і програмного забезпечення
5. Помилки проектування.
6. Недостатність кваліфікації та досвіду персоналу.
7. Відсутність резервних копій даних.
8. Виникнення незапланованих робіт та поява непередбачуваних витрат.
9. Поява альтернативного продукту.

Оцінюємо ризики за наступними показниками.

Ймовірність виникнення:	*Ступінь втрат:*
1 Слабкоймовірно	*1 Мінімальна*
2 Малоймовірно	*2 Низька*
3 Ймовірно	*3 Середня*
4 Вельми ймовірно	*4 Висока*
5 Майже можливо	*5 Максимальна*

Класифікуємо ризики за даними показниками та розрахуємо індекс ризику за формулою 4:

$$R = P_q \times I_q \qquad (4)$$

де R – індекс ризику (бали);

P_q – ймовірність виникнення ризиків відповідно до класифікації (бали);

I_q – величина втрат відповідно до класифікації ризику (бали)

Таблиця А.2 – Оцінка ймовірності виникнення, величини витрат та індексу ризиків

№	Ризик	R	P_q	I_q
R1	Часте внесення змін у ТЗ.	5	4	20
R2	Неоптимальний розподіл часу.	3	4	12
R3	Вибір неефективної технології розробки.	2	4	8
R4	Збій апаратного і програмного забезпечення	2	4	8
R5	Помилки проектування.	3	5	15
R6	Недостатність кваліфікації та досвіду персоналу.	4	4	16
R7	Відсутність резервних копій даних.	2	4	8
R8	Виникнення незапланованих робіт та поява непередбачуваних витрат.	3	3	9
R9	Поява альтернативного продукту	2	3	6

Якісна оцінка ризиків - процес подання якісного аналізу ідентифікації ризиків і визначення ризиків, що вимагають швидкого реагування. Така оцінка ризиків визначає ступінь важливості ризику і вибирає спосіб реагування [28].

Матриця ймовірності втрат використовує наступну класифікацію:

зелений колір – прийнятні ризики;

жовтий колір – виправданні ризики;

червоний колір – недопустимі ризики.

Ймовірність виникнення						
5				R1	
4				R6	
3			R8	R2	R5
2			R9	R3, R4, R7	
1					
	1	2	3	4	5
	Ступінь втрат				

Рисунок А.4 – Матриця ймовірності втрат

Кількісна оцінка ризиків часто супроводжує якісну оцінку і також вимагає процес ідентифікації ризиків. Кількісна і якісна оцінка ризиків можуть використовуватися окремо або разом, залежно від наявного часу і бюджету, необхідності в кількісній або якісній оцінці ризиків [29].

На підставі отриманого значення індексу ризики класифікують: за ступенем впливу(таблиця А.3) та рівнем ризику(таблиця А.4)

Таблиця А.3 – Шкала оцінювання за ступенем впливу

Назва	Межі	Ризики, що входять
Ігноруючі	1≤R≤4	-
Незначні	5≤R≤8	R3, R4, R7, R8, R9
Помірні	9≤R≤10	-
Істотні	12≤R≤19	R2, R5, R6
Критичні	20≤R≤25	R1

Таблиця А.4 – Шкала оцінювання за рівнем ризику

Назва	Межі	Ризики, що входять
Прийнятні	1≤R≤4	
Виправдані	5≤R≤10	R3, R4, R7, R8, R9
Недопустимі	12≤R≤25	R1, R2, R5, R6

Планування реагування на ризики - це розробка методів і технологій зниження негативного впливу ризиків на проект. За ступенем впливу істотні та критичні ризики потребують певних дій, які дадуть можливість уникнути їх.

Аспекти по запобіганню виникнення ризиків наведені у таблиці А.5.

Таблиця А.5 – Аспекти по запобіганню виникнення ризиків

№	Аспекти по запобіганню виникнення ризиків
Критичні	
R1	Погодити всі питання на початкових етапах, щоб мінімізувати кількість змін під час розробки.
Істотні	
R2	Здійснювати проміжний контроль результатів в ході виконання проекту.
R5	На етапі проектування тісно співпрацювати із замовником та на певних етапах демонструвати поточні результати.
R6	Проводити курси підвищення знань персоналу, тренінги тощо.

Отже, необхідно завчасно передбачати усі можливі варіанти для уникнення будь-яких ризиків та по можливості не допускати. Якщо все ж таки це трапилося – то своєчасно ліквідувати їх.

ДОДАТОК Б
ПРОГРАМНИЙ КОД

```
CREATE TABLE Symptom
        (s_id int(10) not null AUTO_INCREMENT,
        s_name varchar(30),
      s_info varchar(100),
  primary key (s_id)
  );

CREATE TABLE Add_Info
        (ai_id int(10) not null AUTO_INCREMENT,
        ai_name varchar(50),
      ai_info varchar(100),
  primary key (ai_id)
  );

CREATE TABLE Disease
        (d_id int(10) unsigned not null AUTO_INCREMENT,
        d_name varchar(30),
      d_info varchar(100),
  primary key (d_id)
  );

CREATE TABLE Symptom_Disease
        (sd_id int(10) unsigned not null AUTO_INCREMENT,
        sd_s int(10),
      sd_d int(10),
  primary key (sd_id),
  FOREIGN KEY (sd_s)
    REFERENCES Symptom (s_id),
  FOREIGN KEY (sd_d)
    REFERENCES Disease (d_id)
  );

CREATE TABLE Add_Info_Disease
        (aid_id int(10) not null AUTO_INCREMENT,
```

```
        aid_s int(10),
      aid_d int(10),
   primary key (aid_id),
   FOREIGN KEY (aid_s)
    REFERENCES Add_Info (ai_id),
   FOREIGN KEY (aid_d)
    REFERENCES Disease (d_id)
   );

CREATE TABLE Users
        (u_id int(10) not null AUTO_INCREMENT,
        u_login varchar(20),
      u_pswd varchar(30),
   primary key (u_id)

   );

CREATE TABLE Patient
                        (p_id int(10) not null AUTO_INCREMENT,
                                p_name varchar(30),
      p_surname varchar(30),
      p_gender char(1),
      p_birth date,
      p_email varchar(50),
      p_adr varchar(100),
      p_renew date,
      p_ds int(10),

   primary key (p_id),
   FOREIGN KEY (p_ds)
    REFERENCES Symptom_Disease (sd_id)



   );
ALTER TABLE Patient
ADD FOREIGN KEY (p_ds)
```

```
        REFERENCES Disease(d_id);


        from django.urls import path, include
        from . import views
        urlpatterns = [
            path(' ', views.index, name='index'),
            path('home/', views.index, name='index'),
            path('dr/', views.dr, name='dr'),
            path('diag/', views.diag, name='diag'),]
        from django.shortcuts import render
        def index(request):
            return render(request, 'ori/homePage.html')
        def dr(request):
            return render(request, 'ori/drPage.html')
        def diag(request):
            return render(request, 'ori/diagPage.html')
X = np.array([ [0,0,1],[0,1,1],[1,0,1],[1,1,1] ])
y = np.array([[0,1,1,0]]).T
syn0 = 2*np.random.random((3,4)) - 1
syn1 = 2*np.random.random((4,1)) - 1
for j in xrange(60000):
        l1 = 1/(1+np.exp(-(np.dot(X,syn0))))
        l2 = 1/(1+np.exp(-(np.dot(l1,syn1))))
        l2_delta = (y - l2)*(l2*(1-l2))
        l1_delta = l2_delta.dot(syn1.T) * (l1 * (1-l1))
syn1 += l1.T.dot(l2_delta)
syn0 += X.T.dot(l1_delta)
import numpy as np
def nonlin(x,deriv=False):
        if(deriv==True):
                return f(x)*(1-f(x))
        return 1/(1+np.exp(-x))
X = np.array([  [0,0,1],
        [0,1,1],
```

```
        [1,0,1],
        [1,1,1] ])
y = np.array([[0,0,1,1]]).T
np.random.seed(1)
syn0 = 2*np.random.random((3,1)) - 1
for iter in xrange(10000):
        l0 = X
        l1 = nonlin(np.dot(l0,syn0))
        l1_error = y - l1
        l1_delta = l1_error * nonlin(l1,True)
        syn0 += np.dot(l0.T,l1_delta)
weight_update = input_value * l1_delta
print "Output after the training:"
print l1
if expect:
        if allerg:
                x = 'Направлення на аналіз чутливості до атибіотиків'
        else:
                if compl:
                        x = 'Негайне призначення'
                else:
x = 'Відкладене призначення'
else:
x = 'Не призначення'
```

button.jxs

```
        import React, { Component } from "react";
        import PropTypes from "prop-types";
        import styles from "./Button.module.css";

        class Button extends Component {
          static defaultProps = {
            color: "plain",
            type: "button"
          };

          getStyle = () => {
            const { color } = this.props;
            switch (color) {
```

```
        case "plain":
          return styles.button_plain;
        case "green":
          return styles.button_green;
        case "faded":
          return styles.button_faded;    case "yellow":
          return styles.button_yellow;
        default:
          return styles.button;
      }
    };

    render() {
      const { title, handleClick, type } = this.props;
      const style = this.getStyle();
      return (
        <button onClick={handleClick} className={style} type={type}>
          {title}
        </button>
      );
    }
  }

  Button.propTypes = {
    title: PropTypes.string.isRequired,
    color: PropTypes.oneOf(["plain", "green", "faded"])
  };

export default Button;
```

CheckboxGroup.jxs

```
  import React, { Component, Fragment } from "react";
  import PropTypes from "prop-types";
  import styles from "./CheckboxGroup.module.css";

  class CheckboxGroup extends Component {
    changeList = option => {
      const { input } = this.props;
      let value = input ? input.value : "";
      value = Array.from(value);
      if (value.includes(option)) {
        const index = value.indexOf(option);
        value.splice(index, 1);
      } else {
        value.push(option);
      }
      if (input) {
        input.onChange(value);
      }
    };

    getOptions = () => {
```

```
    const { title, options, input } = this.props;
    const value = input ? input.value : [];
    const { changeList } = this;
    const list = options.map((option, index) => {
      const isChecked = value.includes(option);
      return (
        <div className={styles.item} key={option + index + !!isChecked}>
          <input
            className={styles.input}
            type="checkbox"
            name={title}
            defaultChecked={isChecked}
            onChange={() => changeList(option)}
            value={option}
            id={option + index}
          />
          <label className={styles.itemTitle} htmlFor={option + index}>
            <span />
            {option}
          </label>
        </div>
      );
    });
    return <Fragment>{list}</Fragment>;
  };

  render() {
    const { title } = this.props;
    const list = this.getOptions();
    return (
      <Fragment>
        <p className={styles.title}>{title}</p>
        <div />
        {list}
      </Fragment>
    );
  }
}

CheckboxGroup.propTypes = {
  title: PropTypes.string.isRequired,
  options: PropTypes.array
};

export default CheckboxGroup;
```

dateinput.jxs

```
import React, { Component, Fragment } from "react";
import DatePicker from "react-datepicker";
import moment from "moment";
import "react-datepicker/dist/react-datepicker.css";
import "./DateInput.css";
```

```
import styles from "./DateInput.module.css";

class DateInput extends Component {
  handleChange = value => {
    const onChange = this.props.input ? this.props.input.onChange : null;
    this.setState({
      startDate: value
    });
    if (onChange) {
      onChange(value.utc().format());
    }
  };
  handleBlur = () => {
    const { input } = this.props;
    const onBlur = input ? input.onBlur : null;
    const date = input.value ? moment(input.value) : moment();
    if (onBlur) {
      onBlur(date);
    }
  };

  render() {
    let { title, isRequired, isShowError, meta, input } = this.props;
    var { touched, error, warning } = meta ? meta : null;

    const date = input.value ? moment(input.value) : moment();
    return (
      <Fragment>
        <div className={styles.block}>
          <p className={styles.title}>{title}</p>
          {isRequired && <p className={styles.title}>*</p>}
        </div>
        <DatePicker
          showYearDropdown
          selected={date}
          onChange={this.handleChange}
          onBlur={this.handleBlur}
        />
        <p className={styles.error}>
          {((touched || isShowError) &&
            ((error && <span>{error}</span>) ||
              (warning && <span>{warning}</span>))) || <br />}
        </p>
      </Fragment>
    );
  }
}

export default DateInput;
```

input.jxs

```
import React, { Component } from "react";
```

```
import PlainInput from "./PlainInput";
import PhoneInput from "./PhoneInput";
import SearchInput from "./SearchInput";
import PasswordInput from "./PasswordInput";
import PlaceAutoComplete from "./PlaceAutoComplete";

class Input extends Component {
  static defaultProps = {
    width: 300,
    type: "text"
  };
  constructor(props) {
    super(props);
    const { type } = props;
    this.state = {
      type
    };
  }

  render() {
    let { type, width, placeholder, isShowError, isRequired } = this.props;
    let inputComponent;
    switch (type) {
      case "password":
        inputComponent = (
          <PasswordInput
            {...this.props}
            type={type}
            isRequired={isRequired}
            placeholder={placeholder}
            isShowError={isShowError}
          />
        );
        break;
      case "phone":
        inputComponent = (
          <PhoneInput
            {...this.props}
            isRequired={isRequired}
            placeholder={placeholder}
            isShowError={isShowError}
          />
        );
        break;
      case "placeAutoComplete":
        inputComponent = (
          <PlaceAutoComplete
            {...this.props}
            isRequired={isRequired}
            placeholder={placeholder}
            isShowError={isShowError}
          />
```

```
            );
            break;
          case "search":
            inputComponent = (
              <SearchInput
                {...this.props}
                isRequired={isRequired}
                placeholder={placeholder}
                isShowError={isShowError}
              />
            );
            break;
          default:
            inputComponent = (
              <PlainInput
                {...this.props}
                type={type}
                isRequired={isRequired}
                placeholder={placeholder}
                isShowError={isShowError}
              />
            );
        }
        return <div style={{ width }}>{inputComponent}</div>;
      }
    }
export default Input;
```

passwordInput.jxs

```
    import React, { Fragment, Component } from "react";
    import styles from "./Input.module.css";

    class PasswordInput extends Component {
      constructor(props) {
        super(props);
        const { type } = props;
        this.state = {
          type
        };
      }
      changePassVisability = () => {
        const { type } = this.state;
        const newType = type === "password" ? "text" : "password";
        this.setState({ type: newType });
      };

      render() {
        let {
          title,
          height,
          input,
          isRequired,
```

```
      isShowError,
      meta: { touched, error }
    } = this.props;
    const { type } = this.state;

    const isErrorShow = (touched || isShowError) && error;
    let style = isErrorShow ? styles.inputError : styles.password;

    const styleButton =
      type === "password" ? styles.passHideInVisiable : styles.passHideVisiable;

    return (
      <Fragment>
        <div className={styles.block}>
          <p className={styles.title}>{title}</p>
          {isRequired && <p className={styles.title}>*</p>}
        </div>
        <input
          {...input}
          onCopy={e => e.preventDefault()}
          className={style}
          style={{ height }}
          type={type}
          name={title}
        />
        <div className={styleButton} onClick={this.changePassVisability} />
        <p className={styles.error}>
          {(isErrorShow && (error && <span>{error}</span>)) || <br />}
        </p>
      </Fragment>
    );
  }
}

export default PasswordInput;
```

placeautocomplit.jxs

```
import React, { Component } from "react";
import PlacesAutocomplete, {
  geocodeByAddress,
  getLatLng
} from "react-places-autocomplete";
import styles from "./Input.module.css";

class PlaceAutoComplete extends Component {
  static defaultProps = {
    width: 300
  };

  handleChange = address => {
    const { onChange } = this.props.input;
    onChange(address);
```

```
};

handleSelect = address => {
  const { onChange } = this.props.input;

  geocodeByAddress(address)
    .then(results => {
      getLatLng(results[0]);
    })
    .then(latLng => {
      onChange(address);
    })
    .catch(error => console.error("Error", error));
};

render() {
  const { title, input, width } = this.props;
  const value = input ? input.value : "";
  return (
    <PlacesAutocomplete
      style={{ width }}
      value={value}
      onSelect={this.handleSelect}
      onChange={this.handleChange}
    >
      {({ getInputProps, suggestions, getSuggestionItemProps }) => (
        <div>
          <div className={styles.block}>
            <p className={styles.title}>{title}</p>
          </div>
          <input
            style={{ width }}
            autoComplete="street-address"
            className={styles.input}
            {...getInputProps({
              placeholder: "Пошук місця ..."
            })}
          />
          <div style={{ width }} className="autocomplete-dropdown-container">
            {suggestions.map(suggestion => {
              const className = suggestion.active
                ? styles.suggestionItemActive
                : styles.suggestionItem;
              return (
                <div
                  {...getSuggestionItemProps(suggestion, {
                    className: className
                  })}
                >
                  <span>{suggestion.description}</span>
                </div>
              );
```

```
                })}
              </div>
            </div>
          )}
        </PlacesAutocomplete>
      );
    }
  }

export default PlaceAutoComplete;
```

plaininput.jxs

```
import React, { Fragment } from "react";
import styles from "./Input.module.css";

const PlainInput = props => {
  let {
    title,
    height,
    input,
    type,
    isRequired,
    placeholder,
    isShowError,
    meta
  } = props;

  let touched, error;
  if (meta) {
    touched = meta.touched;
    error = meta.error;
  }
  const isErrorShow = (touched || isShowError) && error;
  let style = isErrorShow ? styles.inputError : styles.input;

  return (
    <Fragment>
      <div className={styles.block}>
        <p className={styles.title}>{title}</p>
        {isRequired && <p className={styles.title}>*</p>}
      </div>
      <input
        {...input}
        className={style}
        style={{ height }}
        type={type}
        placeholder={placeholder}
      />
      <p className={styles.error}>
        {(isErrorShow && (error && <span>{error}</span>)) || <br />}
      </p>
    </Fragment>
```

```
    );
  };

export default PlainInput;
```

searchinput.jxs

```
import React, { Component } from "react";
import styles from "./Input.module.css";

class SearchInput extends Component {
  render() {
    let { title, placeholder, handleSearch } = this.props;

    return (
      <div className={styles.searchWrapper}>
        <input
          onChange={e => handleSearch(e.target.value)}
          className={styles.input}
          placeholder={placeholder}
          name={title}
        />
        <div className={styles.search} onClick={this.handleSearch} />
      </div>
    );
  }
}

export default SearchInput;
```

multiselect.jxs

```
import React from "react";
import Select from "react-select";
import "react-select/dist/react-select.css";
import "./Multiselect.css";
import styles from "./Multiselect.module.css";

class Multiselect extends React.Component {
  handleSelectChange = value => {
    const { input } = this.props;
    const onChange = input ? input.onChange : null;
    value = value.split(",");
    if (onChange) {
      onChange(value);
    }
  };

  handleBlur = () => {
    const { input } = this.props;
    const onBlur = input ? input.onBlur : null;
    const value = input ? input.value : [];
    if (onBlur) {
      onBlur(value);
```

```
  }
 };

 render() {
  const {
   options,
   title,
   input,
   isShowError,
   meta: { touched, error }
  } = this.props;

  const value = input ? input.value : [];

  const isErrorShow = (touched || isShowError) && error;
  let style = isErrorShow ? styles.multiselectError : "";

  return (
   <div className={styles.section}>
    <h3 className={styles.title}>{title}</h3>
    <Select
     {...input}
     className={style}
     multi
     onBlur={this.handleBlur}
     onChange={this.handleSelectChange}
     options={options}
     placeholder={`Оберіть ${title.toLowerCase()}`}
     simpleValue
     onBlurResetsInput={false}
     value={value}
    />
    <p className={styles.error}>
     {(isErrorShow && (error && <span>{error}</span>)) || <br />}
    </p>
   </div>
  );
 }
}

export default Multiselect;
```

profile.jxs

```
import React from "react";
import Avatar from "../Avatar/Avatar";
import ProfileSection from "../ProfileSection/ProfileSection";
import styles from "./Profile.module.css";

class Profile extends React.Component {
 getPhones = () => {
  const { contacts } = this.props.user.user;
  const { phoneGroup } = contacts;
```

```
  let phones = {};
  for (var i = 0; i < phoneGroup.length; i++) {
    const phone = contacts[`phone${phoneGroup[i]}`];
    if (phone) {
      let key = `Phone#${phoneGroup[i]}:`;
      phones[key] = phone;
    }
  }
  return phones;
};

render() {
  const {
    account: { userName, password, avatar },
    profile: { firstName, lastName, birthDate, email, adress },
    contacts: { fax, facebook, company, gitHub },
    capabilities: { skills, hobbies, additionalInformation }
  } = this.props.user.user;

  const phones = this.getPhones();
  const img = avatar ? avatar.img : "";
  const sections = [
    <ProfileSection
      title="Дані пацієнта"
      key="Account"
      link={`/edit/${userName}/account`}
      values={{
        "User name:": userName,
        "Password:": password.replace(/./g, "*")
      }}
    />,
    <ProfileSection
      title="Personal"
      key="Personal"
      link={`/edit/${userName}/profile`}
      values={{
        "Ім'я:": firstName,
        "Прізвище:": lastName,
        "Дата народження:": birthDate
          ? new Date(birthDate).toLocaleString().split(",")[0]
          : "",
        "Email:": email,
        "Адреса:": adress
      }}
    />,
    <ProfileSection
      title="Contacts"
      key="Contacts"
      link={`/edit/${userName}/contacts`}
      values={{
        "Company:": company,
        "Fax:": fax,
```

```
                "Facebook Link:": (
                  <a
                    className={styles.link}
                    target="_blank "
                    rel="external"
                    href={`//${facebook}`}
                  >
                    {facebook}
                  </a>
                ),
                "GitHub Link:": (
                  <a
                    className={styles.link}
                    target="_blank "
                    rel="external"
                    href={`//${gitHub}`}
                  >
                    {gitHub}
                  </a>
                ),
                ...phones
              }}
            />,
            <ProfileSection
              title="Capabilities"
              key="Capabilities"
              link={`/edit/${userName}/capabilities`}
              values={{
                "Skills:": skills ? skills.join(",\n") : "",
                "Hobbies:": hobbies ? hobbies.join(",\n") : "",
                "Additional Information": additionalInformation
              }}
            />
          ];

          return (
            <section className={styles.section}>
              <div className={styles.avatarWrapper}>
                <Avatar width="200" img={img} />
              </div>
              <div className={styles.sections}>{sections}</div>
            </section>
          );
        }
      }

export default Profile;
```

profilesection.jxs

```
import React from "react";
import { Link } from "react-router-dom";
import styles from "./ProfileSection.module.css";
```

```
        class ProfileSection extends React.Component {
          render() {
            const { title, link, values } = this.props;
            let data = [];
            if (values) {
              for (let key in values) {
                if (values.hasOwnProperty(key)) {
                  data.push(
                    <div key={key} className={styles.keyValue}>
                      <p className={styles.key}>{key}</p>
                      <p className={styles.values}>{values[key] ? values[key] : ""}</p>
                    </div>
                  );
                }
              }
            }

            return (
              <div className={styles.section}>
                <div className={styles.linkWrapper}>
                  <Link className={styles.link} to={link}>
                    {title}
                  </Link>
                </div>

                <div className={styles.values}>{data}</div>
              </div>
            );
          }
        }

export default ProfileSection;
```

radiogroup.jxs

```
        import React, { Component, Fragment } from "react";
        import PropTypes from "prop-types";
        import styles from "./RadioGroup.module.css";

        class RadioGroup extends Component {
          handleSelectChange = value => {
            const { input } = this.props;
            const onChange = input ? input.onChange : null;
            if (onChange) {
              onChange(value);
            }
          };

          handleBlur = () => {
            const { input } = this.props;
            const onBlur = input ? input.onBlur : null;
            const value = input ? input.value : "";
```

```
    if (onBlur) {
      onBlur(value);
    }
  };

  render() {
    const {
      options,
      title,
      input,
      input: { name, value }
    } = this.props;
    return (
      <Fragment>
        <p className={styles.title}>{title}</p>
        <div className={styles.radioSection}>
          {options.map((option, index) => {
            const isChecked = option === value;
            const className = isChecked ? styles.radioTitleChecked : "";
            return (
              <div key={option + value + index} className={styles.radioWrapper}>
                <input
                  {...input}
                  name={name}
                  onBlur={this.handleBlur}
                  onChange={this.handleSelectChange}
                  type="radio"
                  defaultChecked={isChecked}
                  id={title + option + index}
                  className={styles.input}
                  value={option}
                />
                <label htmlFor={title + option + index} className={className}>
                  <span />
                  {option}
                </label>
              </div>
            );
          })}
        </div>
      </Fragment>
    );
  }
}

RadioGroup.propTypes = {
  title: PropTypes.string.isRequired,
  options: PropTypes.array
};

export default RadioGroup;
```

table.jxs

```
import React, { Fragment } from "react";
import { withRouter, Link } from "react-router-dom";
import TableRow from "../TableRow/TableRow";
import Button from "../Button/Button";
import styles from "./Table.module.css";

class Table extends React.Component {
 constructor(props) {
  super(props);
  this.state = {
   applicantForDelete: ""
  };
 }

 showDeleteConfirmation = index => {
  this.setState({
   applicantForDelete: index
  });
 };

 removeAplicantForDelete = () => {
  this.setState({
   applicantForDelete: ""
  });
 };

 showInfo = user => {
  const { history } = this.props;
  this.setState({
   applicantForDelete: ""
  });
  history.push(`/profile/${user.userName}`);
 };

 render() {
  const dataColumns = ["Прізвище", "Ім'я ", "Дата оновлення", ''];
  const {  deleteUser } = this.props;
  const { applicantForDelete } = this.state;

  const tableHeaders = (
   <thead className={styles.tableHead}>
    <tr className={styles.tableRow}>
     {dataColumns.map((column, index) => {
      return (
       <th key={index} className={styles.tableRowColumn}>
        {column}
       </th>
      );
     })}
    </tr>
   </thead>
```

```
    );

    const users = [
      {first: "Максим" ,second: 'Теторенко', lastUpdated: '5 травня'},
      {first: "Наталія" ,second: 'Московчеко', lastUpdated: '12 травня'},
      {first: "Федір" ,second: 'Орипа', lastUpdated: '13 травня'}
    ];

    let tableBody = null;
    let noUsers = null;
    if (users.length > 0) {
      tableBody = (
        <tbody className={styles.tableBody}>
          {users.map((user, index) => (
            <TableRow
              key={index}
              indexKey={index}
              user={user}
              removeAplicantForDelete={this.removeAplicantForDelete}
              isButtonVisiable={!(index === applicantForDelete)}
              deleteUser={deleteUser}
              showDeleteConfirmation={() => this.showDeleteConfirmation(index)}
              isEven={!!(index % 2)}
            />
          ))}
        </tbody>
      );
    } else {
      noUsers = (
        <div className={styles.noUsers}>
          <h2 className={styles.noUsersTitle}>Пацієнти відсутні :(</h2>
          <Link to={`/add/account`}>
            <Button title="Створити нового пацієнта" />
          </Link>
        </div>
      );
    }
    return (
      <Fragment>
        <table className={styles.table}>
          {tableHeaders}
          {tableBody}
        </table>
        {noUsers}
      </Fragment>
    );
  }
}

export default withRouter(Table);
```

tablerow.jxs

```
import React from "react";
import TimeAgo from "javascript-time-ago";
import en from "javascript-time-ago/locale/en";
import TdName from "../TdName/TdName";
import styles from "./TableRow.module.css";

TimeAgo.locale(en);
const timeAgo = new TimeAgo("en-US");

class TableRow extends React.Component {
 handleDelete = event => {
  const { showDeleteConfirmation } = this.props;
  event.stopPropagation();
  showDeleteConfirmation();
 };

 handelClick = () => {
  const {
   removeAplicantForDelete,
   isButtonVisiable,
   showInfo,
   user
  } = this.props;
  if (isButtonVisiable) {
   showInfo(user.userName);
  } else {
   removeAplicantForDelete();
  }
 };



 render() {
  const {
   user,
   isButtonVisiable,
   indexKey,
   isEven,
   showInfo,
   removeAplicantForDelete,
   deleteUser
  } = this.props;

  let style = {},
   buttonSubmit = {};
  if (!isEven) {
   style = {
    background: "#e7f0ff",
    position: "relative",
    opacity: "1"
   };
```

```
}
if (!isButtonVisiable) {
  buttonSubmit = {
    display: "block"
  };
  style = {
    ...style,
    opacity: "0.5",
    transform: "translate(-70px, 0)"
  };
}
const buttons = isButtonVisiable && [
  <button
    key="edit"
    className={styles.edit}
    onClick={() => showInfo(user.userName)}
  />,
  <button
    key="delete"
    className={styles.delete}
    onClick={this.handleDelete}
  />
];
const rowClass = !isButtonVisiable
  ? styles.tableRowActive
  : styles.tableRow;


return (
  <tr
    key={indexKey}
    onClick={removeAplicantForDelete}
    className={rowClass}
    style={style}
  >
    <td key="name" className={styles.tableRowColumn}>
      {user.second}
    </td>

    <td key="company" className={styles.tableRowColumn}>
      {user.first}
    </td>

    <td key="lastUpdate" className={styles.tableRowColumn}>
      {user.lastUpdated}
    </td>

    <td key="buttons" className={styles.tableRowColumn}>
      <div className={styles.columnWrapper}>
        {buttons}

        <button
```

```
                  className={styles.submitDelete}
                  style={buttonSubmit}
                >
                  Видл.
                </button>
              </div>
            </td>
          </tr>
        );
      }
    }

export default TableRow;
```

tablearea.jxs

```
    import React, { Component, Fragment } from "react";
    import PropTypes from "prop-types";
    import styles from "./TextArea.module.css";

    class TextArea extends Component {
      render() {
        let {
          title,
          isShowError,
          input,
          meta: { touched, error }
        } = this.props;

        const isErrorShow = (touched || isShowError) && error;
        let style = isErrorShow ? styles.textareaError : styles.textarea;

        return (
          <Fragment>
            <h3 className={styles.title}>{title}</h3>
            <textarea className={style} {...input} />
            <p className={styles.error}>
              {(isErrorShow && (error && <span>{error}</span>)) || <br />}
            </p>
          </Fragment>
        );
      }
    }

    TextArea.propTypes = {
      title: PropTypes.string.isRequired
    };

export default TextArea;
```

usaveddialog.jxs

```
    import React from "react";
    import styles from "./UnsavedDialog.module.css";
```

```
const UnsavedDialog = props => {
  const { userInProcess, closeDialog, confirmDialog } = props;

  const isEmpty =
    Object.keys(userInProcess).length === 0 &&
    userInProcess.constructor === Object;

  if (isEmpty) {
    return null;
  }

  return (
    <div className={styles.wrapper}>
      <p className={styles.title}>
        You have an unsaved user data. Do you want to complete it?
        <span onClick={confirmDialog} className={styles.continue}>
          Continue
        </span>
      </p>
      <button onClick={closeDialog} className={styles.close} type="button" />
    </div>
  );
};

export default UnsavedDialog;
```

wizard.jxs

```
import React from 'react';
import { Switch, Route, Redirect } from 'react-router';
import { connect } from 'react-redux';
import { withRouter } from 'react-router';
import WizardNav from '../WizardNav/WizardNav';
import UnsavedDialog from '../UnsavedDialog/UnsavedDialog';
import Account from './WizardContainers/Account/Account';
import Profile from './WizardContainers/Profile/Profile';
import Contacts from './WizardContainers/Contacts/Contacts';
import Capabilities from './WizardContainers/Capabilities/Capabilities';
import styles from './Wizard.module.css';

class Wizard extends React.Component {
  constructor(props) {
    super(props);
    const { user } = props;
    this.state = {
      isDialogVisible: !user,
    };
  }

  confirmDialog = () => {
    const {
      applyUserInProcess,
```

```
    userInProcess,
    formErrors,
    history,
  } = this.props;
  applyUserInProcess(userInProcess);
  this.setState({
    isDialogVisible: false,
  });
};

closeDialog = () => {
  const { isDialogVisible } = this.state;
  const { deleteUserInProcess, formErrors } = this.props;
  if (isDialogVisible) {
    deleteUserInProcess();
    this.setState({
      isDialogVisible: false,
    });
  }
};

render() {
  const {
    onSubmit,
    handleSubmit,
    userInProcess,
    user,
    users,
    form,
    isEditMode,
  } = this.props;
  const { isDialogVisible } = this.state;

  let userName = user ? user.userName : null;
  const path = userName ? `/edit/${userName}` : '/add';
  return (
    <div className={styles.wrapper}>
      <WizardNav isEditMode={isEditMode} />
      <form
        onFocus={this.closeDialog}
        onClick={this.closeDialog}
        onSubmit={handleSubmit(onSubmit)}
      >
        <Switch>
          <Route
            path={`${path}/profile`}
            render={routeProps => (
              <Account
                {...routeProps}
                users={users}
                isEditMode={isEditMode}
                form={form}
```

```
                />
              )}
            />
            <Route
              path={`${path}/contacts`}
              render={routeProps => (
                <Contacts {...routeProps} isEditMode={isEditMode} form={form} />
              )}
            />
            <Route
              path={`${path}/capabilities`}
              render={routeProps => (
                <Capabilities
                  {...routeProps}
                  isEditMode={isEditMode}
                  form={form}
                />
              )}
            />
            <Route
              path={`${path}/account`}
              render={routeProps => (
                <Profile {...routeProps} isEditMode={isEditMode} form={form} />
              )}
            />

          </Switch>
        </form>
      </div>
    );
  }
}

const mapStateToProps = state => {
  return {
    userInProcess: state.userInProcess,
  };
};

export default connect(mapStateToProps)(withRouter(Wizard));
```

wizardnav.jxs

```
import React from "react";
import { connect } from "react-redux";
import { withRouter } from "react-router";
import canGoWizardNav from "../../utils/canGoWizardNav";
import styles from "./WizardNav.module.css";
import ProfileSection from "../ProfileSection/ProfileSection";

class WizardNav extends React.Component {
  goTo = link => {
    const { location, history, form, isEditMode } = this.props;
```

```
    history.push(link);

  };

  getClassName = link => {
    const { location } = this.props;
    const linkActive = location.pathname.split("/").pop();
    return link === linkActive ? styles.link_active : styles.link;
  };
  render() {
    return (
      <nav className={styles.nav}>
        <div className={styles.linksHalf}>
          <div
            className={this.getClassName("account")}
            onClick={() => this.goTo("account")}
          >
            1. Дані пацієнта
          </div>
          <div
            className={this.getClassName("profile")}
            onClick={() => this.goTo("profile")}
          >
            2. Симптоми
          </div>
        </div>
        <div className={styles.linksHalf}>
          <div
            className={this.getClassName("contacts")}
            onClick={() => this.goTo("contacts")}
          >
            3. Дод. інформація
          </div>
          <div
            className={this.getClassName("capabilities")}
            onClick={() => this.goTo("capabilities")}
          >
            4. Діагноз
          </div>
        </div>
      </nav>
    );
  }
}

const mapStateToProps = state => {
  return {
    form: state.form
  };
};
```

```
export default connect(mapStateToProps)(withRouter(WizardNav));

        import React from "react";
        import Wizard from "../../components/Wizard/Wizard";
        import styles from "./AddNewUser.module.css";

        class AddNewUser extends React.Component {
          componentWillUnmount() {
            const { resetFrom } = this.props;
            resetFrom();
          }

          componentDidMount() {
            const { history } = this.props;
          }

          render() {
            const {
              deleteUserInProcess,
              applyUserInProcess,
              onSubmit,
              handleSubmit,
              userInProcess,
              user,
              users,
              form,
              formErrors
            } = this.props;
            return (
              <section className={styles.section}>
                <div className={styles.wrapper}>
                  <h1 className={styles.title}></h1>
                  <Wizard
                    onSubmit={onSubmit}
                    handleSubmit={handleSubmit}
                    userInProcess={userInProcess}
                    user={user}
                    users={users}
                    form={form}
                    formErrors={formErrors}
                    deleteUserInProcess={deleteUserInProcess}
                    applyUserInProcess={applyUserInProcess}
                  />
                </div>
              </section>
            );
          }
        }

export default AddNewUser;
```

edit.jxs

```
import React, { Fragment } from "react";
import { connect } from "react-redux";
import { reset } from "redux-form";
import { Link } from "react-router-dom";
import Wizard from "../../components/Wizard/Wizard";
import styles from "./Edit.module.css";

class Edit extends React.Component {
 constructor(props) {
  super(props);
  const { location } = this.props;
  const userNameUrl = location.pathname.slice(6).split("/")[0];
  this.state = {
   isInitialValuesSet: false,
   userNameUrl
  };
 }

 Title = (text, userName) => {
  return (
   <div className={styles.titleWrapper}>
    <Link className={styles.back} to={`/profile/${userName}`}>
     User Profile
    </Link>
    <h1 className={styles.title}>{text}</h1>
    <p className={styles.empty} />
   </div>
  );
 };

 pageFounded = user => {
  const {
   userName,
   profile: { firstName, lastName }
  } = user.user;

  const title = this.Title(`Edit ${firstName} ${lastName}`, userName);
  return (
   <Fragment>
    {title}
    <Wizard
     {...this.props}
     isEditMode={true}
     user={this.user}
     oldUsername={userName}
    />
   </Fragment>
  );
 };

 componentWillUnmount() {
```

```
    const { dispatch, form } = this.props;
    dispatch(reset(form));
  }

  render() {
    const { users, setInitialState } = this.props;
    const { userNameUrl } = this.state;

    this.user = users.filter(user => user.userName === userNameUrl)[0];

    if (this.user && !this.isInitialValuesSet) {
      setInitialState(this.user);
      this.isInitialValuesSet = true;
    }

    const page = this.user
      ? this.pageFounded(this.user)
      : this.Title(`User ${userNameUrl} was not found`, userNameUrl);
    return (
      <section className={styles.section}>
        <div className={styles.wrapper}>{page}</div>
      </section>
    );
  }
}

export default connect(state => {
  return {
    users: state.users
  };
})(Edit);
```

Sagas.jxs

```
import { put, takeEvery, all } from "redux-saga/effects";
import { setUsers } from "./reducers/users.js";
import { setUsersInProcess } from "./reducers/userInProcess.js";
import { loadAllUsersFromIndexedDB, getUserInProcess } from "./utils/IndexedDB";

//ACTIONS
const LOAD_USERS = "LOAD_USERS";
const LOAD_USER_IN_PROCESS = "LOAD_USER_IN_PROCESS";

function* loadAllUsersSaga() {
  const users = yield loadAllUsersFromIndexedDB();
  yield put(setUsers(users));
}

function* loadUsersInProcessSaga() {
  const user = yield getUserInProcess();
  yield put(setUsersInProcess(user));
}
```

```
        function* watchUsers() {
          yield takeEvery(LOAD_USERS, loadAllUsersSaga);
          yield takeEvery(LOAD_USER_IN_PROCESS, loadUsersInProcessSaga);
        }

        export default function* rootSaga() {
          yield all([watchUsers()]);
}
```

App.py

```
        from django.apps import AppConfig


        class OriConfig(AppConfig):
    name = 'ori'

        from django.contrib import admin
        from django.utils.translation import ugettext_lazy as _
        from django.utils.text import capfirst
        from django.db.models.base import ModelBase
        from django.conf import settings
        from pymorphy import get_morph

        morph = get_morph(settings.PYMORPHY_DICTS['ru']['dir'])

        class I18nLabel():
            def __init__(self, function):
                self.target = function
                self.app_label = u''

            def rename(self, f, name = u''):
                def wrapper(*args, **kwargs):
                    extra_context = kwargs.get('extra_context', {})
                    if 'delete_view' != f.__name__:
                        extra_context['title'] = self.get_title_by_name(f.__name__, args[1], name)
                    else:
                        extra_context['object_name'] = morph.inflect_ru(name, u'вн').lower()
                    kwargs['extra_context'] = extra_context
                    return f(*args, **kwargs)
                return wrapper

            def get_title_by_name(self, name, request={}, obj_name = u''):
                if 'add_view' == name:
                    return _('Add %s') % morph.inflect_ru(obj_name, u'вн,стр').lower()
                elif 'change_view' == name:
                    return _('Change %s') % morph.inflect_ru(obj_name, u'вн,стр').lower()
                elif 'changelist_view' == name:
                    if 'pop' in request.GET:
                        title = _('Select %s')
                    else:
                        title = _('Select %s to change')
                    return title % morph.inflect_ru(obj_name, u'вн,стр').lower()
```

```
            else:
                return ''

        def wrapper_register(self, model_or_iterable, admin_class=None, **option):
            if isinstance(model_or_iterable, ModelBase):
                model_or_iterable = [model_or_iterable]
            for model in model_or_iterable:
                if admin_class is None:
                    admin_class = type(model.__name__+'Admin', (admin.ModelAdmin,), {})
                self.app_label = model._meta.app_label
                current_name = model._meta.verbose_name.upper()
                admin_class.add_view = self.rename(admin_class.add_view, current_name)
                admin_class.change_view = self.rename(admin_class.change_view, current_name)
                admin_class.changelist_view = self.rename(admin_class.changelist_view,
current_name)
                admin_class.delete_view = self.rename(admin_class.delete_view, current_name)
            return self.target(model, admin_class, **option)

        def wrapper_app_index(self, request, app_label, extra_context=None):
            if extra_context is None:
                extra_context = {}
            extra_context['title'] = _('%s administration') % _(capfirst(app_label))
            return self.target(request, app_label, extra_context)

        def register(self):
            return self.wrapper_register

        def index(self):
            return self.wrapper_app_index

    admin.site.register = I18nLabel(admin.site.register).register()
admin.site.app_index = I18nLabel(admin.site.app_index).index()
```