

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
ДІАГНОСТУВАННЯ ШКІРНИХ ЗАХВОРЮВАНЬ
RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS FOR
DIAGNOSIS OF SKIN DISEASES

Виконав: студент(ка) 2 курсу, групи КНТ-120м
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Мироненко Н.В.

(прізвище та ініціали)

Керівник Льовкін В.М.

(прізвище та ініціали)

Рецензент Скрупський С.Ю.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 20 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Мироненко Надії Володимирівни

(прізвище, ім'я, по батькові)

- Тема проєкту (роботи) Дослідження та програмна реалізація методів діагностування шкірних захворювань
Research and Software Implementation of Methods for Diagnosis of Skin Diseases
керівник проєкту (роботи) Льовкін Валерій Миколайович, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від “5” листопада 2021 року № 435
- Строк подання студентом проєкту (роботи) грудень 2021 року
- Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання
- Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали і методи. 3. Розробка архітектури програми. 4. Основні рішення щодо реалізації компонентів системи. 5. Експлуатація, тестування та експериментальне дослідження програми. Додатки
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Слайди презентації.

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Льовкін В.М., доцент		
Нормоконтроль	Камінська Ж.К., асистент		

7. Дата видачі завдання “ 1 ” вересня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз та дослідження предметної області	2-3 тижні	Розділ 1
3	Розробка архітектури програми	4 тиждень	Розділ 2
4	Розробка програми	5-6 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми	7 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	8 тиждень	Додатки
7	Захист роботи	9 тиждень	

Студент(ка)

(підпис) МIRONENKO H.B.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) ЛЬОВКІН В.М.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра: 138 с., 7 табл., 70 рис., 3 дод., 61 джерело.

ДЕРМАТОСКОПІЯ, ДІАГНОСТУВАННЯ, ЗГОРТКОВА НЕЙРОННА МЕРЕЖА, НАБІР ДАНИХ HAM10000, ШКІРНЕ ЗАХВОРЮВАННЯ, PHP, PYTHON, RESNET50.

Об'єкт дослідження – процес діагностування шкірних захворювань.

Предмет дослідження – методи діагностування шкірних захворювань.

Мета роботи – розробка програмного забезпечення діагностування шкірних захворювань, що дозволить знизити ризик встановлення помилкового діагнозу та почати лікування на початковій стадії.

Матеріали, методи та технічні засоби: набір даних HAM10000, згорткова нейронна мережа ResNet50, об'єктно-орієнтоване програмування, мови програмування PHP, Python, JavaScript, мова розмітки HTML, мова таблиць стилів CSS, система керування базою даних MySQL.

Результати. Проведено аналіз предметної області, розглянуто програми-аналоги, запропоновано метод діагностування шкірних захворювань, визначено функціональні вимоги до розроблюваної системи, спроектовано та виконано програмну реалізацію застосунку, проведено тестування створеної системи.

Висновки. Наукова новизна роботи полягає в запропонованому методі діагностування шкірних захворювань, який характеризується використанням модифікованої моделі ResNet50, що дозволяє підвищити точність винесення діагнозу. Розроблений метод використовується у створеному вебзастосунку для діагностування шкірних захворювань.

Галузь використання – медичне діагностування. Система призначена для використання лікарями та пацієнтами.

ABSTRACT

Explanatory note to the master's thesis: 138 pp., 7 tables, 70 figures, 3 appendices, 61 sources.

DERMATOSCOPY, DIAGNOSIS, CONVOLVED NEURAL NETWORK, DATA SET HAM10000, SKIN DISEASE, PHP, PYTHON, RESNET50.

The object of research is the process of diagnosing skin diseases.

The subject of research - methods of diagnosing skin diseases.

The purpose of the work is to develop software for diagnosing skin diseases, which will reduce the risk of misdiagnosis and start treatment at an early stage.

Materials, methods and technical means: HAM10000 data set, ResNet50 convolutional neural network, object-oriented programming, PHP programming languages, Python, JavaScript, HTML markup language, CSS style sheet language, MySQL database management system.

Results. The analysis of the subject area is carried out, the programs-analogues are considered, the method of diagnosing skin diseases is offered, the functional requirements to the developed system are defined, the program implementation of the application is designed and executed, the created system is tested.

Conclusions. The scientific novelty of the work lies in the proposed method of diagnosing skin diseases, which is characterized by the use of a modified model ResNet50, which allows to increase the accuracy of diagnosis. The developed method is used in the created web application for diagnosing skin diseases.

Field of use - medical diagnosis. The system is intended for use by doctors and patients.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз проблеми та постановка завдань дослідження	10
1.1 Шкірні захворювання	10
1.2 Огляд програмних засобів.....	14
1.3 Висновки до першого розділу.....	19
2 Матеріали і методи.....	21
2.1 Методи розпізнавання зображень у медичному діагностуванні.....	21
2.2 Згорткові нейронні мережі в дерматології	22
2.3 Архітектура моделі нейронної мережі ResNet50	24
2.4 Аналіз набору даних шкірних захворювань.....	27
2.5 Метод діагностування шкірних захворювань	33
2.6 Висновки до другого розділу	38
3 Розробка архітектури програми.....	39
3.1 Вибір засобів проєктування	39
3.1.1 Вибір засобів розробки методу діагностування.....	39
3.1.2 Вибір засобів розробки вебзастосунку	42
3.1.3 Використання менеджера пакетів.....	51
3.1.4 Налаштування локального середовища розробки	52
3.2 Визначення функціональних вимог до програмної системи.....	54
3.3 Проєктування структури бази даних.....	57
3.4 Проєктування архітектури програмного забезпечення.....	60
3.5 Визначення структури програмного забезпечення	61
3.6 Висновки до третього розділу	62
4 Основні рішення щодо реалізації компонентів системи.....	64
4.1 Алгоритм функціонування програми.....	64
4.2 Опис модулів реалізованої програмної системи.....	66
4.3 Висновки до четвертого розділу.....	73

5	Експлуатація, тестування та експериментальне дослідження програми	74
5.1	Опис застосування програми	74
5.2	Умови виконання програми	74
5.3	Інструкція по експлуатації програми	74
5.4	Повідомлення та обробка виняткових ситуацій	86
5.5	Методика та результати тестування.....	88
5.5.1	Навчання та оцінка точності нейронної мережі	88
5.5.2	Тестування вебзастосунку.....	92
5.6	Висновки до п'ятого розділу.....	97
	Висновки	98
	Перелік джерел посилання	100
	Додаток А Технічне завдання	106
	Додаток Б Текст програми	112
	Додаток В Слайди презентації.....	130

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

AJAX	– Asynchronous Javascript And Xml;
AKIES	– Actinic Keratoses and Intraepithelial Carcinoma;
BCC	– Basal Cell Carcinoma;
BKL	– Benign Keratosis-like Lesions;
CNN	– Convolutional Neural Network;
CSS	– Cascading Style Sheets;
DF	– Dermatofibroma;
DNN	– Deep Neural Networks;
HAM	– Human Against Machine;
HTML	– HyperText Markup Language;
IDE	– Integrated Development Environment;
JS	– JavaScript;
LAMP	– Linux, Apache, MySQL, PHP;
MEL	– Melanoma;
MVC	– Model-View-Controller;
NV	– Melanocytic Nevi;
PDO	– PHP Data Objects;
VASC	– Vascular Lesions;
БД	– база даних;
СКБД	– система керування базами даних;
ІІІ	– штучний інтелект.

ВСТУП

Захворювання шкіри є одними з найбільш поширених і небезпечних. Незважаючи на те, що всі новоутворення на шкірі легко побачити, часто люди звертаються до лікарів тільки тоді, коли захворювання знаходиться у пізній стадії. Однак своєчасне виявлення та діагностування таких проблем є вирішальним фактором, який впливає на можливість проведення ефективної терапії та забезпечення нормальної якості життя хворої людини.

За статистикою, лише близько 40% немеланомних пухлин шкіри в Україні діагностуються на ранніх стадіях (стадія 1 і 2), що може гарантувати позитивний прогноз одужання пацієнтів, а діагностувати меланому на початковій стадії, коли захворювання піддається лікуванню найкраще, взагалі вдається не часто [1].

У 30% випадків рак шкіри розвивається з родимки і в 70% маскується під неї [2]. За офіційними даними за 2020 рік в Україні рак шкіри займає перше місце серед усіх типів раку, а кількість пацієнтів, яким поставлено даний діагноз, перевищує 180 тисяч [3].

Тому важливим аспектом ефективного лікування є виявлення та діагностування захворювань на ранніх стадіях. Однак, не завжди встановлений діагноз є правильним. Причиною цього може бути низька кваліфікація лікаря, подібність зовнішнього вигляду захворювань, неправильно зібрані аналізи, людський фактор.

Для того, щоб забезпечити більш точну класифікацію захворювання, потрібно використовувати сучасні підходи для діагностування. Одним з них є запровадження штучного інтелекту (ШІ), зокрема, нейронної мережі. У кооперації з лікарем це дозволить значно скоротити кількість неправильних діагнозів і врятувати здоров'я та життя пацієнтів.

Отже, метою даної роботи є розробка програмного забезпечення діагностування шкірних захворювань, що дозволить знизити ризик встановлення помилкового діагнозу та почати лікування на початковій стадії.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Шкірні захворювання

Шкіра – найбільший орган людського організму, який діє як водонепроникний, ізолюючий щит, захищаючи тіло від перепадів температур, пошкоджень сонячним світлом та шкідливого хімічного впливу, шляхом виділення антибактеріальних речовин, які запобігають інфекціям, і вироблення вітаміну D для перетворення кальцію в здорові кістки, а також шкіра – це величезний датчик, наповнений нервами, які підтримують зв'язок мозку з зовнішнім світом [4].

Як і інші органи, шкіра страждає від усіх типів патологічних змін, включаючи спадкові, запальні, доброякісні та злоякісні (неопластичні), ендокринні, гормональні, травматичні та дегенеративні процеси [5]. Це викликає різні види захворювань, деякі з них можуть охоплювати невелику ділянку у верхній частині шкіри, а деякі проникати глибоко в шкіру або поширюватися на більшу площу і викликати суттєве погіршення здоров'я та навіть призводити до смерті людини.

Серед найбільш поширених шкірних захворювань можна виділити наступні:

- актинічний кератоз – поширена вікова зміна шкіри, головним чинником виникнення якої вважають ураження неприкритих одягом ділянок шкіри обличчя, рук, голови, ніг внаслідок сонячного опромінення впродовж багатьох років; вогнища хвороби становлять численні піднесені над шкірою та схильні до лущення ділянки шорсткої шкіри 1-3 см в діаметрі від рожевого до коричневого кольору [6]. Притаманне для актинічного кератозу повільне прогресування та схильність вогнищ залишатися в незмінному стані тривалий час зумовлює малосимптомний перебіг порушення та вихід на перший план косметичного дефекту шкіри, однак наявна небезпека його трансформації в базальноклітинний рак шкіри [6];

– базаліома (базальноклітинний рак шкіри) є найпоширенішою формою раку шкіри і найбільш поширеною формою всіх видів раку, найбільш часто пухлина локалізується на обличчі, шиї або волосяній частині голови і на початковому етапі виглядає як невелика папула (не більше 4 мм) з плоскою або випуклою поверхнею [7]. За статистикою, основна частина пацієнтів, у яких зустрічається даний недуг, – люди старшого і похилого віку, проте цей вид раку рідко прогресує, поширюючись на інші органи і тканини, що дозволяє більш ніж в 90 % випадків повністю побороти хворобу [7];

– себорейний кератоз – це неракові (доброякісні) нарости шкіри, які розвиваються у деяких людей і часто з'являються, як правило, на спині або грудях, але можуть виникати і на інших частинах тіла, ростуть повільно групами або поодинокі, можуть бути від світло-коричневого до коричневого або чорного кольору, тому їх можна помилково прийняти за бородавки, родимки або рак шкіри [8];

– доброякісна фіброзна гістіоцитома (дерматофіброма) відноситься до доброякісних пухлин, які, як правило, мають розмір до 1 см у діаметрі, але можуть досягати 10 см, найчастіше відмічається ураження кінцівок, а тому ураження голови, шиї й обличчя є нетиповим для даного типу пухлини, яка часто представлена як поверхнєве, папулонодулярне новоутворення із мінливою пігментацією, що повільно росте, але загалом даний тип пухлини має сприятливий прогноз, проте останнім часом з'являються повідомлення про переродження пухлин даного типу в більш агресивні форми новоутворень, що найчастіше пов'язано з некоректно проведеним хірургічним або іншими видами лікування [9];

– меланома – злоякісна пухлина, яка розвивається з меланоцитів (пігментні клітини, що продукують меланіни), клінічно проявляється пухлиноподібним щільним вузликом чорного забарвлення, іноді з синім відтінком, від 0,5 см до 2-3 см у діаметрі, округлої чи овальної форми з нерівними краями, і розповсюджується через лімфу і кров майже в усі органи шляхом появи метастазів, які зазвичай з'являються протягом першого року

від початку захворювання і спершу поширюються в лімфатичні вузли, пізніше – в легені, печінку, кістки, головний мозок [10]. У 50-70% випадків меланома шкіри виникає в ділянці пігментної плями, родимки; за даними Всесвітньої організації охорони здоров'я (ВООЗ), щорічно у всьому світі реєструється приблизно 48 000 смертей, пов'язаних з меланомою [10];

– меланоцитарний невус, або родимки – це доброякісні новоутворення шкіри різних розмірів (від кількох міліметрів до декількох сантиметрів) із чіткими межами, округлої, овальної чи неправильної форми, із плоскою, припіднятою чи папіломатозною поверхнею, які можуть відрізнятися за кольором – від світло-коричневого до темно-коричневого та чорного, і виникають при вадах розвитку меланінутворюючих клітин – епідермальних та дермальних меланоцитів [11]. Переважна більшість меланоцитарних невусів є доброякісними новоутвореннями, вони ростуть пропорційно тілу людини і не зумовлюють неприємних відчуттів, за винятком косметичних проблем, водночас у кожної людини можуть з'явитися родимки з підвищеним ризиком переродження у злоякісну пухлину [11];

– судинні ураження шкіри: почервоніння шкіри спочатку мають вигляд одиничних зон розширених капілярів, які за відсутності належного лікування збільшуються і переходять у форму купероза; наступна більш розвинена стадія процесу (розацеа) характеризується застійними явищами в капілярах і дрібних судинах шкіри, що приводять до розширення вен на рівні венозних сплетінь та запалення шкіри і можуть виглядати як судинні зірочки, рубінові тільця, телеангіектазії, дрібні «блакитні вени», винні плями [12]. На жаль, судинні ураження шкіри не проходять самостійно і потребують лікування: чим раніше вжиті заходи з лікування судин, тим легше впоратися з проблемою, тим більше гарантій успішної корекції [12].

Процес діагностування шкірних захворювань відбувається за наступним планом:

- збір анамнезу пацієнта: лікар з'ясовує у пацієнта, що його турбує, як давно, які можуть бути причини, якого способу життя дотримується пацієнт;
- візуальний огляд: лікар візуально оцінює стан шкіри для визначення подальших дій;
- інструментальний огляд: проводиться огляд ділянки шкіри методом дерматоскопії [13].

Дерматоскопія – це один із найпрогресивніших та найсучасніших методів діагностування новоутворень, який не передбачає хірургічного втручання, оскільки для проведення процедури використовують дерматоскоп – збільшувальний прилад, який дає можливість побачити, в якому стані структура родимки, і чи є вона доброякісною [14].

Окрім безболісності, у процедури є і інший великий плюс: її можна робити всім, навіть дітям та вагітним жінкам, бо у цього метода діагностування немає ніяких протипоказань [14].

Таку процедуру призначають у випадку, коли є підозра на рак шкіри, а також коли треба діагностувати бородавки, кератоми, папіломи, гемангіоми та інші утворення на шкірі [14]. Завдяки дерматоскопії можна виявити злоякісні утворення на ранніх стадіях та провести успішне лікування [14].

Дерматоскопія необхідна у таких випадках:

- новоутворення різко збільшилося;
- з'явився біль, свербіж чи лущення на місці утворення;
- родимка змінила колір і/або форму;
- родимка має розмір більше 0,5 сантиметра;
- при пошкодженнях новоутворень;
- якщо у когось з родичів було виявлено меланому [14].

Процес проведення процедури дуже простий: на проблемну ділянку наноситься спеціальний гель, який скеровує світло під потрібним кутом, після цього до тіла притискають зовнішнє скло апарату та роблять кадр (знімок), за допомогою якого лікар може визначити, яка структура та розміри

утворення і вже на основі цього поставити остаточний діагноз та призначити лікування; оскільки дана процедура повністю безболісна, вона не займає більше п'ятих хвилин, до того ж дає дуже чіткий та правильний результат і дозволяє контролювати навіть найменші зміни новоутворень, і тому дерматоскопія – це найкращий метод діагностування новоутворень в сучасній медицині [14].

1.2 Огляд програмних засобів

Для того, щоб розробити власне програмне забезпечення діагностування шкірних захворювань, потрібно спочатку розглянути існуючі додатки, провести їх аналіз та визначити доцільність розробки нового застосунку.

Одним із застосунків для онлайн перевірки родимок є сервіс «Проверь родинку онлайн» (рис. 1.1).



Рисунок 1.1 – Сервіс «Проверь родинку онлайн»

Сервіс призначено для проведення самообстеження на наявність факторів ризику розвитку раку шкіри, зокрема меланоми. Дозволяє в домашніх умовах:

- провести самодіагностику;
- вчасно виявити новоутворення чи зміну вже існуючого утворення (родимки, кератоми);
- своєчасно звернутися до лікаря [15].

Сервіс надає онлайн анкету з 6 питаннями з фотозображеннями основних ознак злоякісності утворень шкіри, у якості зображень використані фотоматеріали родимок, що відповідають показникам норми та патології (зміна кольору родимки, форми, розміру, контуру, структури) [15].

Користувач може порівняти свої показники з показниками фотоматеріалів, при виявленні відхилень ухвалити рішення про відвідування фахівця, а також, відповівши на запитання, користувач отримує перелік рекомендацій щодо профілактики меланоми [15].

До недоліків використання сервісу відносяться наступні:

- немає індивідуального підходу до визначення захворювання (перелік питань є загальним і не може використовуватися для точного визначення захворювання без огляду лікарем);
- варіанти відповідей лише так/ні, але на деякі питання складно відповісти однозначно;
- після проходження тесту немає вірогідних діагнозів, а тільки список лікарень для проведення діагностування;
- немає можливості онлайн-запису на прийом.

Отже, даний сервіс носить лише інформаційний характер, і призначений тільки для того, щоб змусити людину задуматися про похід до лікаря, тобто не може бути використаний у якості повноцінного медичного діагностування.

Тест від медичної мережі «Добробут» (рис. 1.2) передбачає набір запитань про звички та образ життя людини, на основі яких визначається

ризик виникнення різних видів раку, оскільки дослідження доводять безпосередній вплив деяких факторів (наприклад, куріння або неправильного харчування) на виникнення злоякісних новоутворень, такі фактори і називають факторами ризику, а групи людей, схильних до їх впливу – групами ризику [16]. Тест дозволяє виокремити такі групи та визначити імовірність виникнення захворювання.

The screenshot shows a web browser window with the URL dobrobut.com/botany/bot/calculation/skin/mole. The website header includes the logo 'ДОБРОБУТ' and navigation links for 'Українська' and 'Поділитися'. A progress bar at the top indicates the current step: 'Результат розрахунку' (Calculation result), with other steps being 'Антропометричні дані', 'Результат розрахунку', 'Продукти харчування', 'Шкідливі звички', 'Зовнішній вигляд', and 'Загальні питання'. The main heading is 'Кількість родимок (невусів) на вашому тілі?'. Below this, there are four input fields for selecting the number of moles: '0', '0 - 25', '25 - 50', and '> 50'. At the bottom, there are two buttons: '<< Назад' and 'Далі >>'. The footer contains copyright information '© 2019. Всі права захищені.', a link to 'Список літератури', and a link to 'Умова користування'.

Рисунок 1.2 – Тест від медичної мережі «Добробут»

Недоліки сервісу такі ж, як і для попереднього застосунку, однак на відміну від «Проверь родинку онлайн», даний сервіс надає перелік можливих захворювань з імовірними ризиками.

Ще одним популярним застосунком для діагностування шкірних захворювань є додаток Skinive (рис. 1.3). Цей сервіс спочатку був створений білоруськими розробниками як телеграм-бот, але тепер вже з'явився і повноцінний додаток, у якому штучний інтелект дозволяє виявити новоутворення шкіри та класифікувати їх [17].

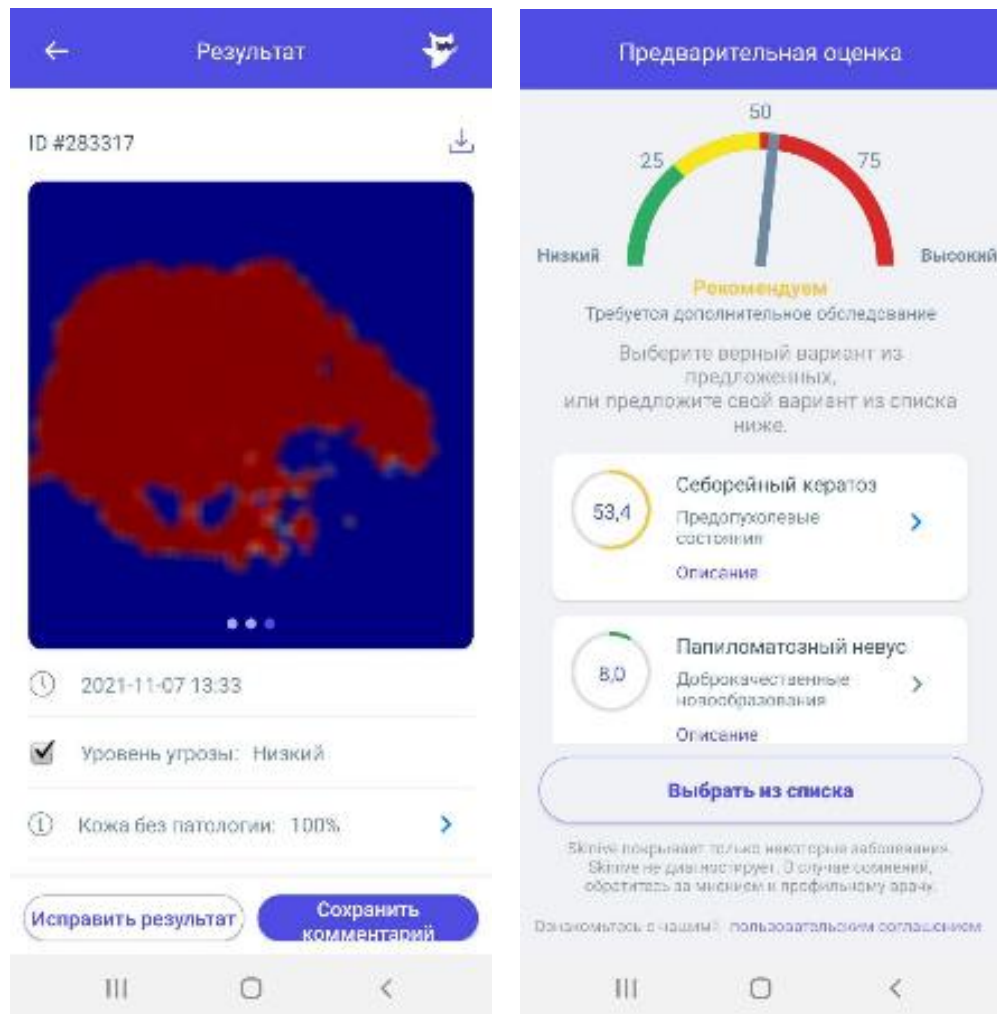


Рисунок 1.3 – Додаток Skinive

Для виявлення новоутворення треба при хорошому освітленні з відстані до 10 сантиметрів сфотографувати ділянку шкіри у додатку Skinive, після чого додаток самостійно визначить відсоток імовірності захворювання та порадить, до якого лікаря звертатися з цією проблемою: до дерматолога, косметолога чи дерматоонколога, а у разі виявлення високого ризику захворювання знімок оброблятиметься вручну, після чого буде відправлена рекомендація [17].

В основі роботи додатка лежить авторитетне Стенфордське дослідження з визначення меланоми на знімках шкіри, на основі якого спочатку розробники вирішили спробувати реалізувати алгоритм діагностування, а успішний результат дав основу для можливості визначення

інших поширених видів раку – базальноклітинного та плоскоклітинного [17].
На чітких знімках достовірність прогнозу сягає 90% [17].

До недоліків застосунку можна віднести:

- безкоштовна версія додатка надається лише на 3 місяці;
- обмеження у кількості завантажених зображень для безкоштовної версії;
- відсутня можливість онлайн-запису до лікаря;
- необхідно вправно зробити знімок, адже за умови використання недостатньо чіткого знімку діагноз може бути встановлено неправильно.

Також для аналізу шкіри використовується додаток Miiskin (рис. 1.4).

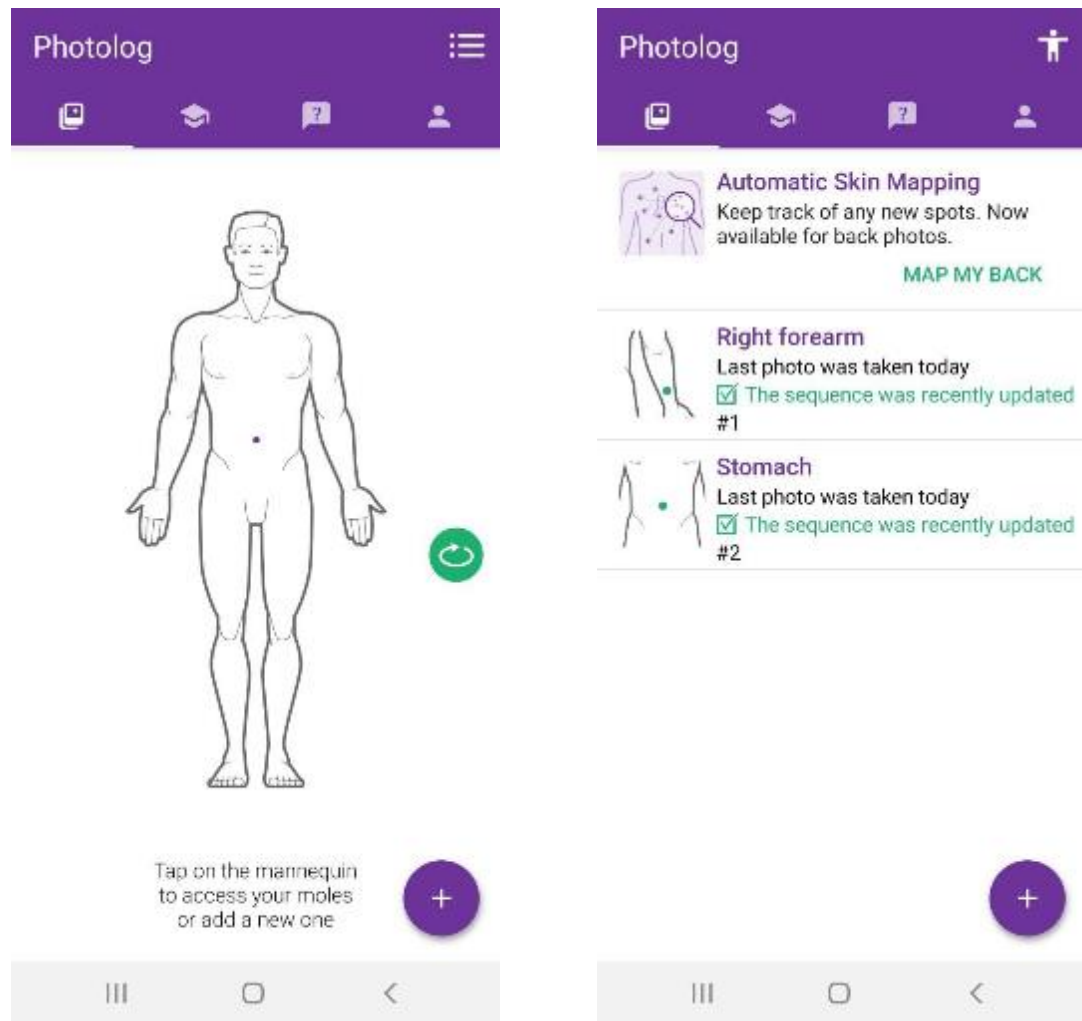


Рисунок 1.4 – Додаток Miiskin

Ця програма дозволяє виявити всі підозрілі зміни шкіри, які для звичайного ока невидимі, і можуть бути виявлені за допомогою карти родимок (фактично це збільшена цифрова фотографія), яку зазвичай створюють дерматологи під час клінічного обстеження шкіри всього тіла за допомогою цифрової дерматоскопії [17].

Оскільки фотографії з високою роздільною здатністю дають набагато більше інформації, ніж звичайні цифрові фотографії, то розробники Miiskin хочуть дати можливість користуватися цією технологією звичайним споживачам, тому було розроблено додаток для створення збільшених фотографій великих ділянок шкіри [17].

Програма зберігає фотографії окремо від звичайної бібліотеки смартфона і дозволяє оцінювати на шкірі зміни родимок з часом [17].

До недоліків відноситься те, що додаток не встановлює імовірний діагноз, а просто дозволяє відстежувати зміни новоутворення з плином часу. До того ж таке відстеження важко виконувати непрофесіоналу через відсутність необхідних знань і досвіду роботи з даним типом захворювання.

Отже, розглянуті застосунки не надають повноцінного функціоналу для застосування у сфері діагностування шкірних захворювань, тому є потреба у створенні власного програмного забезпечення, яке могло б об'єднати переваги та виправити недоліки існуючих.

1.3 Висновки до першого розділу

У даному розділі було розглянуто проблему діагностування шкірних захворювань, виділено та описано симптоми деяких з них, проведено аналіз програм-аналогів. Також розглянуто метод дерматоскопії, який використовується для визначення захворювань.

Через те, що шкірні захворювання викликають не тільки косметичний дефект, а можуть становити серйозну небезпеку для життя та здоров'я людини, важливо їх виявити та почати лікування на ранній стадії. Тобто

задача підвищення точності діагностування захворювань шкіри є актуальною та важливою на сьогодні.

Для вирішення цієї задачі потрібно виконати наступні завдання:

- аналіз предметної області та розробка технічного завдання;
- розробка методу діагностування захворювань шкіри;
- проєктування програмного забезпечення;
- розробка програмного забезпечення;
- проведення тестування програмного забезпечення.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Методи розпізнавання зображень у медичному діагностуванні

Розпізнавання медичних зображень відіграє істотну роль у клінічному лікуванні та завданнях навчання, однак традиційні методи досягли максимальної продуктивності, крім того, використовуючи їх, багато часу та зусиль потрібно витратити на виокремлення та вибір класифікаційних ознак [18].

Використання традиційних методів машинного навчання, таких як методи опорних векторів (Support Vector Machine, SVM), в класифікації медичних зображень почалося дуже давно, однак у цих методів є такі недоліки: продуктивність далека від практичних стандартів, а їх розвиток останніми роками відбувається досить повільно, крім того, визначення та вибір функцій займають багато часу і змінюються залежно від різних об'єктів [18].

Глибока нейронна мережа (Deep Neural Networks, DNN) – це новий метод машинного навчання, який довів свій потенціал для різних завдань класифікації [18].

DNN – це нейронна мережа з певним рівнем складності, нейронна мережа з більш ніж двома шарами; багато експертів визначають глибокі нейронні мережі як мережі, які мають вхідний шар, вихідний шар і принаймні один прихований шар між ними [19].

DNN складається з кількох пов'язаних одиниць (вузлів), які вважаються найменшими частинами нейронної мережі, і виконують роль нейронів в мозку людини: коли нейрон отримує сигнал, він запускає процес, сигнал передається від одного нейрона до іншого на основі отриманого входу, таким чином утворюється складна мережа, яка вчиться на зворотному зв'язку [20].

Завдання вирішується шляхом обробки різних шарів (групи вузлів) між вхідним і вихідним шарами, тому чим більша кількість шарів підлягає обробці, тим глибша мережа [20].

DNN, особливо згорткові нейронні мережі (Convolutional Neural Network, CNN), широко використовуються для вирішення завдань класифікації зображень і показують значну продуктивність у даній сфері: деякі дослідження CNN щодо класифікації медичних зображень досягли результатів, які конкурують з експертами-людьми [18].

За останні два десятиліття CNN стали відомими як безцінний інструмент класифікації біомедичних зображень і були запропоновані як інструмент клінічної діагностики в таких дисциплінах, як радіологія, гістологія, офтальмологія та дерматологія [21].

CNN — це глибокі нейронні мережі з архітектурою, спеціально розробленою для аналізу зображень, які зазвичай навчаються за допомогою навчання з учителем, тобто це означає, що CNN використовують розмічені дані, наприклад, дерматоскопічні зображення з відповідним діагнозом, щоб дізнатися зв'язок між вхідними даними та мітками, тому CNN можуть застосовувати вивчені операції до невідомих зображень і класифікувати їх на основі виокремлених ознак [22]. Оскільки діагностика в клінічній дерматології та дерматопатології значною мірою базується на розпізнаванні візуальних моделей, використання CNN може допомогти розробити додаткові та/або покращені клінічно значущі бази даних (БД) [22].

Примітно, що згорткова нейронна мережа домінує з найкращими результатами у різних задачах класифікації зображень [18].

2.2 Згорткові нейронні мережі в дерматології

CNN вважається новим інструментом для дерматологів, який допомагає краще діагностувати ураження. Робота, яку проводить CNN, щоб отримати діагностичний результат із зображення, подібна до того, як

дерматолог використовує свої навички та знання: діагностика уражень дерматологом зазвичай включає вхідне зображення (ураження шкіри), що подається через мережу обробки (навички та знання дерматолога, який аналізує його та синтезує доступну інформацію) для отримання класу (або діагнозу) [23].

Візуальна природа дерматології добре підходить для цифрової візуалізації уражень, і CNN мають величезний потенціал у даній сфері, оскільки це багатогранний засіб аналізу даних, що включає складну математику і вимагає величезних обчислювальних можливостей для поєднання біології, математики та інформатики [23].

Основою CNN є комп'ютерний метод, який здатний розрізняти різні класи зображень на основі унікальних особливостей, які можна надійно використовувати для ідентифікації класу зображень, потім цю базу розширюють, додаючи більш абстрактні особливості за допомогою низки згорткових об'єднань та вихідних шарів [23].

До основних шарів CNN належать:

- згортковий шар: цей шар є першим шаром, який використовується для вилучення різних елементів із вхідних зображень, шляхом виконання математичних операцій згортки між вхідним зображенням і фільтром певного розміру $M \times M$, пересуваючи який по вхідному зображенню, розраховується добуток між фільтром і частинами вхідного зображення відповідно до розміру фільтра ($M \times M$), отримані таким чином вихідні дані називають картою ознак, яка дає інформацію про зображення, наприклад, кути та краї, і пізніше ця карта ознак подається на інші шари, щоб дізнатися інші особливості вхідного зображення;

- шар субдискретизації: основна мета цього шару – зменшити розмір згорнутої карти ознак, щоб заощадити витрати на обчислення, що виконується шляхом зменшення зв'язків між шарами і незалежно працює для кожної карти ознак;

- повнозв'язний шар використовується для з'єднання нейронів між двома різними шарами і зазвичай розміщується перед вихідним шаром та утворює останні кілька шарів архітектури CNN, завдяки чому вхідне зображення з попередніх шарів сплющується і подається на повнозв'язний шар, потім отриманий вектор проходить ще кілька шарів, де зазвичай відбуваються операції з математичними функціями, подібно до того, як дерматолог синтезує різні підказки у попередній діагноз [24].

Переваги CNN у діагностиці уражень шкіри включають:

- точність: оскільки клініко-діагностична точність для меланоми залежить від досвіду і підготовки лікаря, то використання CNN дозволить покращити цей показник, бо за даними джерела [23] CNN можуть працювати так само, як сертифіковані дерматологи за обмежених обставин, і їх точність буде покращуватися в майбутньому;
- швидкість: наразі CNN вимагають від декількох секунд до хвилин, щоб поставити діагноз, коли стикаються з зображенням ураження шкіри, що дозволяє вхідні дані, алгоритми та результати роботи оброблювати поза робочим часом і вони можуть бути доступними для всіх, хто має доступ до Інтернету;
- адаптивність: алгоритми можуть бути адаптивними, і можуть вчитися, додаючи нові зображення з плином часу;
- низька вартість: передбачається, що CNN зможуть діагностувати ураження за частину вартості візиту до дерматолога [23].

2.3 Архітектура моделі нейронної мережі ResNet50

ResNet (residual neural network) – залишкова нейронна мережа, це означає, що вона має глибоку залишкову структуру навчання, додаються з'єднання швидкого доступу, які пропускають один або декілька шарів і виконують співставлення ідентифікаторів поточного про шарку та попереднього залишкового [25].

Поява мережі ResNet обумовлена наступною проблемою: для того, щоб вирішити складну проблему розпізнавання, додаються кілька шарів у глибокі нейронні мережі, що призводить до покращення точності та продуктивності, додавання нових шарів дозволяє мережі поступово вивчати більш складні функції, наприклад, у разі розпізнавання зображень перший шар може навчитися виявляти краї, другий – розпізнавати текстури, а третій – виявляти об'єкти тощо, але було виявлено, що існує максимальний поріг глибини для традиційної моделі згорткової нейронної мережі, тобто це говорить про те, що при додаванні більшої кількості шарів продуктивність мережі погіршується, що стало суттєвою проблемою навчання дуже глибоких мереж і було вирішено шляхом введенням ResNet або залишкових мереж [26].

Модель ResNet-50 – згорткова нейронна мережа, яка має глибину 50 шарів, серед яких виділяють наступні:

- Conv – ядро згортки, тобто матриця ваг, яка «ковзає» над зображенням, поелементно виконуючи операцію множення з тією частиною вхідних даних, над якою воно зараз знаходиться, потім підсумовує всі отримані значення в один вихідний піксель і повторює цю процедуру з кожною локацією, над якою воно «ковзає», перетворюючи двовимірну матрицю в іншу все ще двовимірну матрицю ознак, де ознаки на виході є зваженими сумами (де ваги є значеннями самого ядра) ознак на вході, розташованих приблизно в тому ж місці, що і вихідний піксель на вхідному шарі [27];

- BatchNormalization – це шар, який використовується для нормалізації виходу попередніх шарів, він слугує ефективним механізмом для уникнення перенавчання моделі і додається до послідовної моделі, щоб стандартизувати вхідні або вихідні дані, також його можна використовувати в декількох точках між шарами моделі [28];

- Activation – функція активації: визначає вихідне значення нейрона в залежності від результату зваженої суми входів і порогового

значення [29]. У моделі ResNet-50 у якості функції активації використовуються ReLu – це найбільш часто використовувана функція активації при глибокому навчанні, яка повертає 0, якщо приймає негативний аргумент, а при позитивному аргументі, функція повертає його значення [29];

- MP (MaxPooling) – це операція об'єднання, яка вибирає максимальний елемент із області карти ознак, охопленої фільтром, таким чином, вихідний результат після шару MaxPooling буде картою ознак, що містить найбільші значення попередньої карти ознак [30];

- identity_block – блок ідентифікації є стандартним блоком, що використовується в ResNet-50 і відповідає випадку, коли активація входу має той самий розмір, що і активація виходу (рис. 2.1) [25];

- conv_block – згортковий блок використовується, коли вхідні та вихідні розміри не збігаються (рис. 2.2) [25];

- AveragePool2D – це операція об'єднання, яка обчислює середнє значення елементів, присутніх в області карти ознак, охопленої фільтром [30].

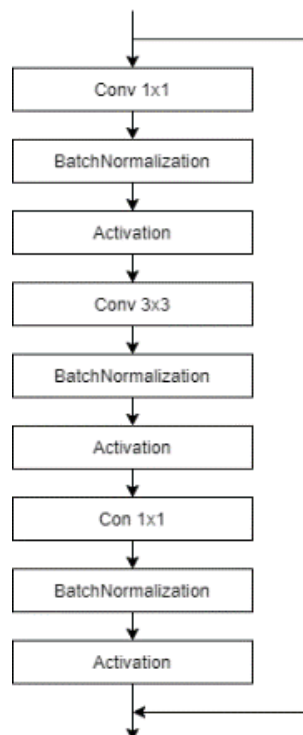


Рисунок 2.1 – Блок ідентифікації ResNet50

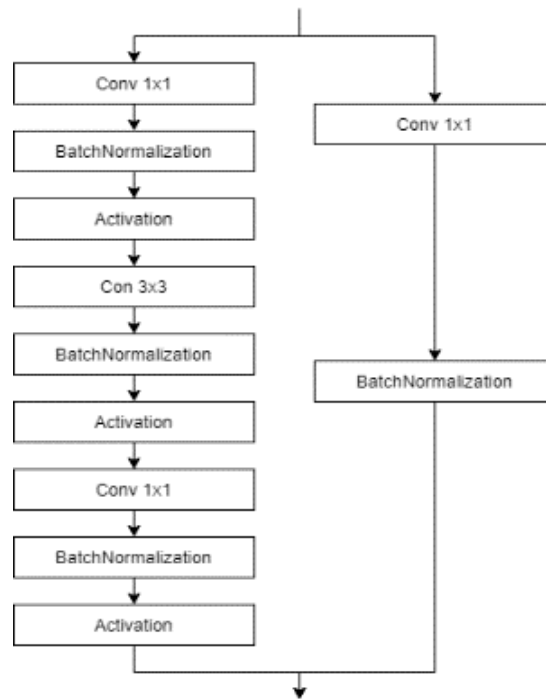


Рисунок 2.2 – Блок згортки ResNet50

2.4 Аналіз набору даних шкірних захворювань

Для розробки та проведення навчання нейронної мережі необхідний набір даних, який містить зображення шкірних захворювань. У якості такого набору було обрано «Human Against Machine with 10000 training images» (HAM10000). Цей набір даних є загальнодоступним і може бути завантажений з офіційних ресурсів, наприклад, таких як Harvard Dataverse або Kaggle.

Він складається з 10015 зображень, отриманих методом дерматоскопії. Усі зображення розподілені на дві папки HAM10000_images_part1 і HAM10000_images_part2, у яких відповідно 5000 і 5015 зображень.

Кожне наявне зображення відноситься до одного з наступних класів, які позначаються відповідними мітками:

- actinic keratoses and intraepithelial carcinoma / Bowen's disease (AKIES) – актинічні кератози та інтраепітеліальна карцинома / хвороба Боуена;

- basal cell carcinoma (BCC) – базаліома;
- benign keratosis-like lesions (BKL) – доброякісні кератозоподібні ураження (сонячні лентигіни / себореїні кератози та лишайники);
- dermatofibroma (DF) – дерматофіброма;
- melanoma (MEL) – меланома;
- melanocytic nevi (NV) – меланоцитарні невуси;
- vascular lesions (VASC) – судинні ураження (ангіоми, ангіокератоми, піогенні гранульоми та крововиливи) [31].

Приклади зображень та діагнозів наведено на рис. 2.3.



Рисунок 2.3 – Приклади зображень з набору даних HAM10000

Також до набору входять два .csv файли, які містять опис пікселів кожного зображення, і .csv файл, який містить додаткову інформацію про пацієнта та складається з наступних колонок:

- image_id – ідентифікатор зображення;
- dx – мітка діагнозу;
- dx_type – метод, який використовувався для визначення типу захворювання;
- age – вік пацієнта;
- sex – стать пацієнта;
- localization – локалізація захворювання.

Діаграма діагнозів та кількість зображень для кожного діагнозу наведені на рис. 2.4 та у табл. 2.1.

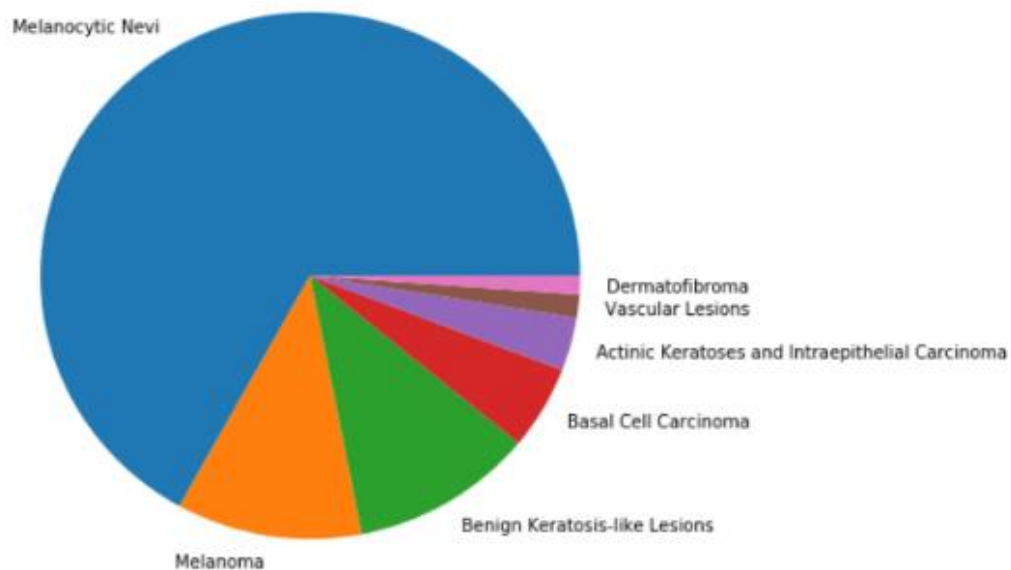


Рисунок 2.4 – Діаграма діагнозів набору HAM10000

Таблиця 2.1 – Кількість зображень для кожного діагнозу

Мітка діагнозу	Кількість зображень
NV	6705
MEL	1113
BKL	1099

Продовження таблиці 2.1

Мітка діагнозу	Кількість зображень
BCC	514
AKIES	327
VASC	142
DF	115

Для визначення типу захворювання використовувалися такі методи, як гістопатологія (histo, 53%), подальше обстеження (follow up, 37%), консенсус експертів (consensus, 9%), підтвердження конфокальною мікроскопією (confocal, 1%). Порівняльна діаграма за кожним з методів наведена на рис. 2.5.

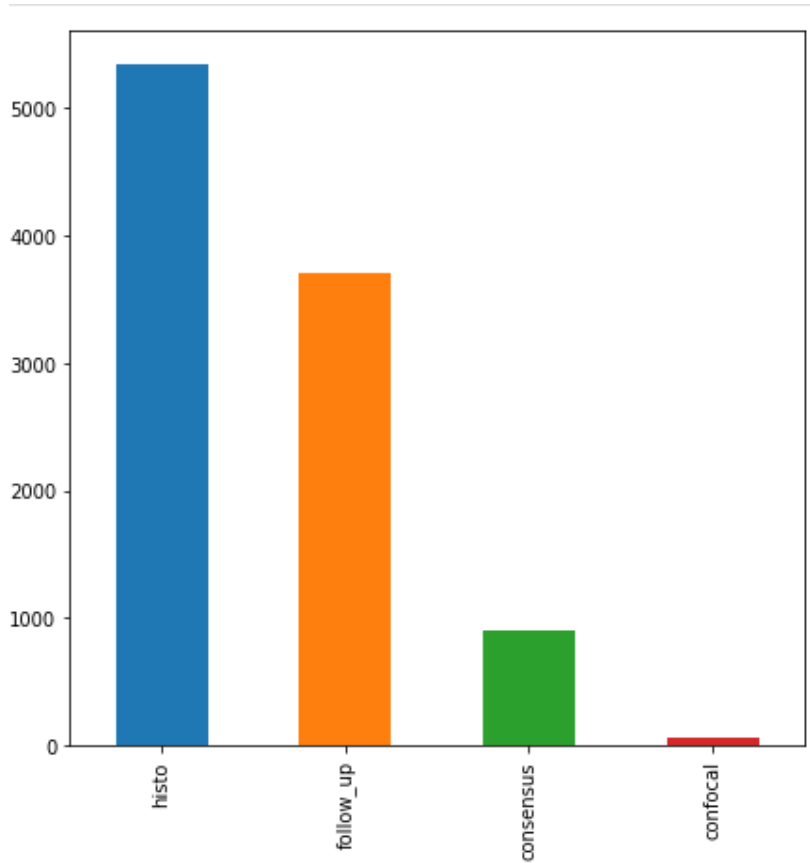


Рисунок 2.5 – Діаграма методів визначення типу захворювання

Вік пацієнтів варіюється від 0 до 85 років, відповідна діаграма наведена на рис. 2.6.

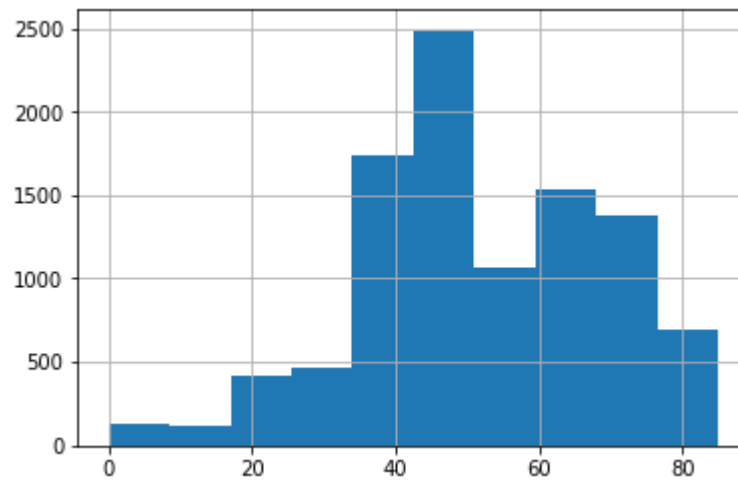


Рисунок 2.6 – Діаграма віку пацієнтів

Серед усіх наявних в наборі зображень шкірних хвороб 54% належать чоловікам (male), 45% – жінкам (female) та 1% зображень, де стать зазначена як unknown (рис. 2.7).

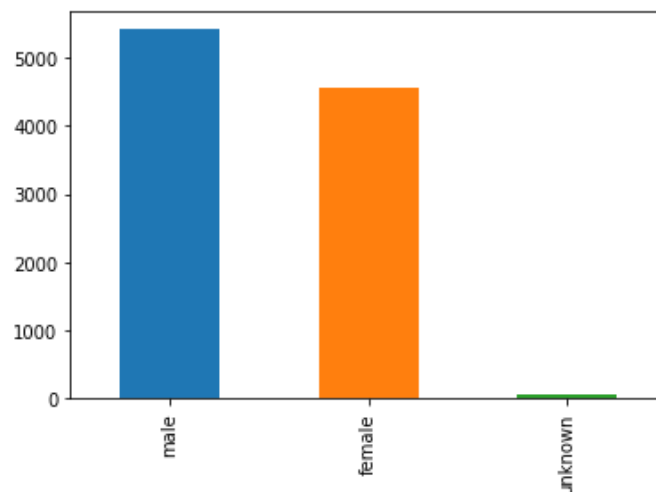


Рисунок 2.7 – Діаграма статі пацієнтів

Значення локалізації захворювань наведено у табл. 2.2, діаграма – на рис. 2.8.

Таблиця 2.2 – Локалізація шкірних захворювань

Місце локалізації	Позначка	Кількість зображень
спина	back	2192
нижня кінцівка	lower extremity	2077
тулуб	trunk	1404
верхня кінцівка	upper extremity	1118
живіт	abdomen	1022
обличчя	face	745
грудна клітина	chest	407
стопа	foot	319
шия	neck	168
шкіра голови	scalp	128
рука	hand	90
вухо	ear	56
статеві органи	genital	48
долоні, підошви стоп, шкіра під нігтями рук або ніг	acral	7
невідомо	unknown	234

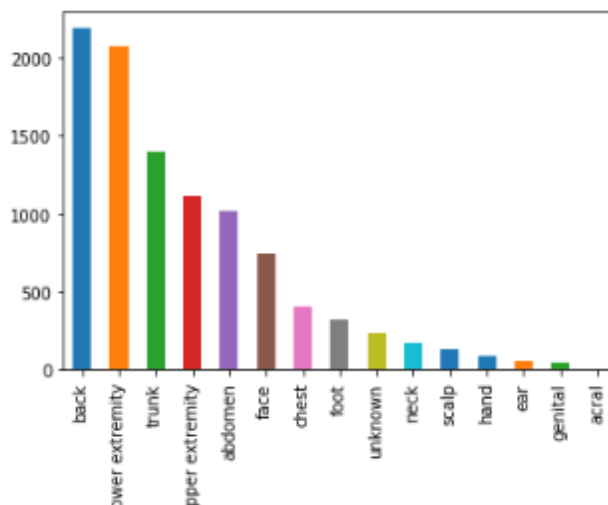


Рисунок 2.8 – Діаграма локалізації шкірних захворювань

2.5 Метод діагностування шкірних захворювань

Задача діагностування полягає у визначенні приналежності об'єкта (захворювання) наведеного на зображенні до певного зафіксованого класу об'єктів (діагнозів). На вхід подаються зображення у вигляді масивів пікселів (значень інтенсивності кожного каналу зображення), які описуються цілими числами у діапазоні від 0 до 255. На виході повертається номер класу діагнозу з наступного переліку:

- 0 – AKIES;
- 1 – BCC;
- 2 – BKL;
- 3 – DF;
- 4 – NV;
- 5 – MEL;
- 6 – VASC.

Даний перелік діагнозів може бути розширений за наявності відповідних зображень у вибірці даних.

Для вирішення задачі діагностування потрібно виконати наступні етапи:

- формування вибірки даних;
- побудова модифікованої моделі;
- навчання моделі;
- перевірка точності розпізнавання моделі.

На першому етапі треба сформувати та підготувати вибірку даних. Для того, щоб досягти високої точності розпізнавання нейронною мережею, необхідно виконати попередню обробку даних, а саме змінити початковий розмір вхідних зображень. Зміна розміру зображень є важливим етапом попередньої обробки даних, оскільки в основному, моделі машинного навчання тренуються швидше на менших зображеннях, більше того, багато

архітектур моделей глибокого навчання вимагають, щоб зображення були однакового розміру [32].

Як видно з табл. 2.1 обраний набір даних є незбалансованим, тобто кількість зображень одного класу (NV) значно перевищує кількість зображень інших класів. У такому випадку не вдасться отримати високу точність діагностування, тому потрібно досягти балансу кількості зображень штучним чином.

Для збалансування набору даних використовуються такі підходи:

- undersampling – це техніка, яка дозволяє збалансувати нерівномірні набори даних, зберігаючи всі дані в класі меншості та зменшуючи розмір класу більшості; це один з кількох методів, який науковці можуть використовувати для отримання більш точної інформації зі незбалансованих наборів даних, хоча він має недоліки, такі як втрата потенційно важливої інформації, даний метод залишається поширеним і важливим для обробки даних;

- oversampling – це техніка, яка дозволяє збалансувати нерівномірні набори даних шляхом збільшення кількості екземплярів в класах меншості, дана техніка доцільна, коли науковці не мають достатньої інформації [33].

Оскільки використання техніки undersampling приведе до зменшення кількості зображень, що в даному випадку матиме негативний вплив на процес навчання моделі, було обрано техніку oversampling. Для того, щоб зображення не повторювалися, випадковим чином потрібно застосувати методи зміни зображень, такі як обертання, зсув по горизонталі та вертикалі, поворот, масштабування.

Кількість зображень для кожного діагнозу після застосування техніки oversampling буде мати вигляд, наведений у табл. 2.3.

Таблиця 2.3 – Кількість зображень після застосування техніки oversampling

Діагноз	Кількість зображень
AKIES	4905
BCC	5140
BKL	5495
DF	5750
MEL	5565
NV	6705
VASC	5680
Загальна кількість	39240

Також потрібно розділити початкову вибірку на наступні вибірки даних:

- навчальна – фактичний набір даних, який використовується для навчання моделі;
- валідаційна – вибірка даних, що використовується для забезпечення оцінки відповідності моделі для навчального набору даних під час налаштування гіперпараметрів;
- тестова – вибірка даних, яка використовується для забезпечення оцінки побудованої моделі, що відповідає набору навчальних даних. Застосовуються лише після того, як модель повністю пройшла етап навчання [34].

Далі потрібно провести нормалізацію даних для кожної вибірки. Дана процедура виконується наступним чином: від значення кожного пікселя зображення віднімається середнє значення та ділиться на стандартне відхилення.

На другому етапі потрібно побудувати модель нейронної мережі шляхом модифікації попередньо навченої моделі ResNet50.

Структура модифікованої моделі наведена на рис. 2.9.

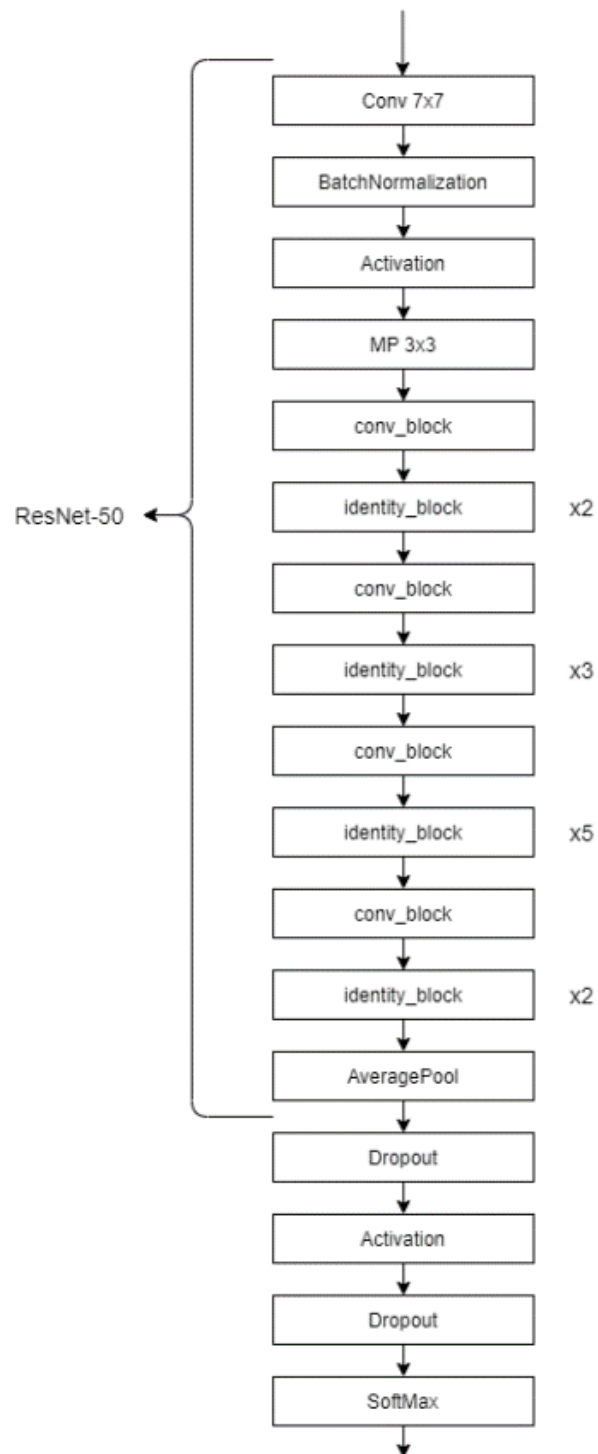


Рисунок 2.9 – Структура модифікованої моделі

До моделі ResNet50 вирішено додати наступні шари:

- Dropout (0.5);
- Activation (шар зі 128 нейронами та функцією активації ReLu);
- Dropout (0.5);
- SoftMax (шар зі 7 нейронами та функцією активації SoftMax).

Dropout – метод регуляризації штучних нейронних мереж, призначений для запобігання перенавчання мережі та підвищення швидкості навчання [35]. Суть методу полягає в тому, що в процесі навчання із загальної мережі виділяється підмережа, шляхом вибору нейронів випадковим чином з імовірністю, яка називається коефіцієнтом дропаута, далі для обраної підмережі здійснюється навчання, потім випадковим чином обирається нова підмережа і навчання продовжується, більш навчені нейрони отримують в мережі більшу вагу [35].

SoftMax – функція активації, яка обчислює відносні ймовірності.

На наступному етапі необхідно провести навчання нейронної мережі. Для цього можна використати техніки трансферного навчання та fine-tuning.

Трансферне навчання – це дуже популярна техніка у глибокому навчанні, яка передбачає повторне використання попередньо навченої моделі для нової проблеми і дозволяє тренувати глибокі нейронні мережі з порівняно невеликою кількістю даних [36].

Fine-tuning – перенавчання окремих шарів нейронної мережі, яке дозволяє залежно від обсягу доступних даних та складності проблеми «заморозити» (не змінювати ваги під час зворотного поширення) перших кількох шарів нейронної мережі і тренувати лише останні кілька шарів, бо початкові шари згорткових нейронних мереж просто вивчають загальні особливості зображень, а більш глибокі – вже вивчають конкретні форми та частини об'єктів, тому можна зосередити навчання саме на них [37].

Після проведення навчання потрібно виконати перевірку точності прогнозування модифікованої моделі нейронної мережі. Більш детально практична частина проведення етапу навчання та оцінки точності моделі описана у пункті 5.5.1.

Після перевірки моделі вона може бути використана для діагностування шкірних захворювань за наступним алгоритмом:

- на першому кроці лікар завантажує зображення шкірного захворювання, отримане методом дерматоскопії, у систему;

- на другому кроці відбувається підготовка зображення до розпізнавання: змінюється його розмір, потім воно представляється у вигляді багатомірного масиву, проводиться нормалізація зображення;
- на третьому кроці підготовлений масив передається у модель нейронної мережі. У результаті повертається номер класу, до якого належить зображення. За отриманим номером визначається назва діагнозу;
- на четвертому кроці отриманий діагноз виводиться на екран для лікаря.

2.6 Висновки до другого розділу

У даному розділі було розглянуто методи діагностування шкірних захворювань. Особливе місце серед них займає використання згорткової нейронної мережі.

Для вирішення задачі діагностування виділено наступні етапи:

- формування вибірки даних;
- побудова модифікованої моделі;
- навчання моделі;
- перевірка точності розпізнавання моделі.

Для розробки власного методу діагностування було вирішено модифікувати попередньо навчену модель ResNet50, для навчання та тестування обрано набір даних HAM10000, який містить зображення отримані методом дерматоскопії. Також було описано техніки, які мають використовуватися на різних етапах розробки методу.

Наукова новизна роботи полягає в запропонованому методі діагностування шкірних захворювань, який характеризується використанням модифікованої моделі ResNet50, що дозволяє підвищити точність винесення діагнозу. Розроблений метод використовується у створеному вебзастосунку для діагностування шкірних захворювань.

3 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

3.1 Вибір засобів проєктування

3.1.1 Вибір засобів розробки методу діагностування

Для розробки методу діагностування шкірних захворювань було обрано мову програмування Python.

Python — інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією; це мова програмування, яка підтримує створення застосунків широкого спектру і вважається чудовим вибором для проєктів ШІ, машинного навчання та глибокого навчання [38].

Причини використання мови Python в машинному навчанні:

- велика кількість бібліотек та фреймворків: мова Python постачається з багатьма бібліотеками та фреймворками, які полегшують кодування і також економлять значну кількість часу;
- простота: код Python є стислим і читабельним, що корисно для проєктів машинного та глибокого навчання, а завдяки простому синтаксису розробка додатків за допомогою Python відбувається швидко, якщо порівнювати з багатьма іншими мовами програмування;
- велика спільнота підтримки: Python є мовою програмування з відкритим кодом і відрізняється чудовою підтримкою з багатьох ресурсів і якісною документацією, також має велику та активну спільноту розробників, які надають свою допомогу на будь-якому етапі розробки;
- швидкість розробки: Python має зрозумілий та «дружній» синтаксис, крім того, численні фреймворки та бібліотеки сприяють більш швидкій розробці програмного забезпечення, використовуючи готові рішення, можна багато чого зробити за допомогою кількох рядків коду;
- гнучка інтеграція: проєкти Python можна інтегрувати з іншими системами, створеними на різних мовах програмування;

- швидке тестування коду: Python надає багато інструментів для тестування, тому розробники можуть швидко перевірити правильність і якість коду;
- продуктивність: деякі розробники стверджують, що Python відносно повільна порівняно з іншими мовами програмування, однак незважаючи на те, що швидкість не є однією з сильних сторін Python, вона забезпечує рішення, відоме як Cython, яке є розширеним варіантом Python, розроблене для досягнення продуктивності коду, такої ж, як і мова C;
- інструменти візуалізації: Python поставляється з широким спектром бібліотек, які пропонують хороші інструменти візуалізації, а оскільки у ШІ, машинному навчанні та глибокому навчанні важливо подавати дані у форматі, зрозумілому людині, то Python є ідеальним вибором для реалізації цієї функції [38].

Порівняння мов програмування, які використовуються для розробки в галузі ШІ наведено у табл. 3.1.

Таблиця 3.1 – Порівняння мов програмування Python, R та Java

Критерії	Python	R	Java
Об'єктно-орієнтованість	Так	Так	Так
Обробка великого обсягу даних	Добре	Гірше	Добре
Простота вивчення	Просто	Складно	Складно
Складність розробки	Просто	Складно	Просто, але потребує більше рядків коду, ніж Python

Продовження таблиці 3.1

Критерії	Python	R	Java
Бібліотеки	Багато	Менше	Багато
Швидкість виконання коду	Швидко	Повільніше	Швидко
Популярність	Велика	Менша	Велика

Оскільки мова програмування Python є найбільш практичною для вирішення задачі побудови нейронної мережі, саме вона була обрана для подальшої розробки.

У процесі розробки було використано наступні бібліотеки:

- TensorFlow є найпопулярнішим інструментом в галузі обробки даних на сьогоднішній день; це бібліотека програмного забезпечення з відкритим кодом для чисельних розрахунків з використанням графів потоку даних, вузли яких представлені у вигляді математичних операцій, в той час як ребра графу представляються багатовимірними масивами даних (тензорами), що передаються між вузлами [39];
- Keras – відкрита нейромережна бібліотека, написана мовою Python, яка була спроектована для можливості виконання швидких експериментів з мережами глибинного навчання і є зручною в користуванні, модульною та розширюваною [40];
- NumPy – відкрита бібліотека, що додає підтримку n-вимірних масивів у Python і величезну кількість швидких зручних функцій для роботи з ними, наприклад, поелементні арифметичні операції над масивами: додавання, віднімання, множення, поділ, зведення в ступінь, тригонометричні функції, гіперболічні функції [41];
- Pandas – дуже популярна бібліотека для обробки даних на Python, яка працює на основі NumPy та надає спеціальні структури даних для маніпулювання з таблицями та тимчасовими рядами [41];

- Matplotlib – бібліотека для візуалізації даних, що використовується для побудови 2D та 3D графіків [41];
- Scikit Learn – один з найбільш широко використовуваних пакетів Python для Data Science та Machine Learning, який дозволяє виконувати безліч операцій та надає безліч алгоритмів, а також пропонує чудову документацію про свої класи, методи та функції, і опис використовуваних алгоритмів [42]. Scikit Learn підтримує попередню обробку даних, зменшення розмірності, вибір моделі, регресії, класифікації, кластерний аналіз [42];
- os – це бібліотека для роботи з операційною системою, яка містить методи, що дозволяють визначати тип операційної системи, отримувати доступ до змінних оточення, керувати директоріями та файлами [43].

3.1.2 Вибір засобів розробки вебзастосунку

Оскільки було вирішено реалізувати програмний продукт у вигляді вебзастосунку, то в якості мови розробки обрано PHP.

PHP (Hypertext Preprocessor – гіпертекстовий препроцесор) – скриптова мова програмування, яка використовується для створення серверної частини вебдодатків, і на сьогодні входить до числа найкращих і найпопулярніших інструментів програмування для веброзробки: за даними джерела W3Techs, 78.8% усіх вебсайтів використовують PHP для свого сервера [44].

Серед переваг використання PHP можна виділити наступні:

- відкритий вихідний код: однією з найважливіших переваг PHP є те, що ця мова програмування доступна для всіх, тобто її можна завантажити і легко використовувати для розробки вебдодатків;
- масштабованість: дуже важлива перевага, оскільки вебсайт повинен розвиватися і мати можливість обробляти збільшення трафіку, а

вебсайти, створені за допомогою PHP, можна легко розширити, додавши більше серверів, коли це необхідно;

- висока швидкість: оскільки PHP використовує власну пам'ять, зменшуються і робоче навантаження, і час завантаження, тому в результаті швидкість обробки досить висока, що є перевагою не тільки для розробників, але і для клієнтів, які швидше отримують продукт, а також швидкість завантаження має вирішальне значення для SEO, а це дуже важливий фактор, якщо сайт використовується для маркетингу та продажів;

- незалежність від платформи: іншим важливим фактором є те, що, програми на основі PHP можуть працювати на будь-якій ОС, наприклад, UNIX, Windows, Linux тощо;

- зручність та простота використання: ця мова проста у використанні та у вивченні, тобто якщо програміст знає мову програмування C, він може швидко засвоїти мову PHP для розробки додатків;

- стабільність: на відміну від інших мов, PHP дуже стабільна протягом багатьох років і може забезпечити користувачу допомогу та постійну підтримку, у тому числі щодо різних версій мови;

- гнучкість: дана мова програмування дуже гнучка, і може легко використовуватися для поєднання з різними іншими мовами програмування;

- велика спільнота розробників: ще однією перевагою PHP є велика спільнота розробників, які працюють з нею, вони не тільки щодня використовують PHP, але й постійно вдосконалюють, вносять нові позитивні зміни та покращення, тому програмісти можуть вчитися і впроваджувати передові рішення якомога швидше, і, як наслідок, усі ці фактори призводять до появи нових фреймворків, компонентів та інших інструментів, які роблять PHP ще кращим [44].

З огляду на те, що для розробки серверної частини вебзастосунку можуть використовуватися різні мови програмування, у табл. 3.2 наведено порівняння основних критеріїв мов програмування PHP, Ruby, Perl.

Таблиця 3.2 – Порівняння мов програмування

Критерії	PHP	Ruby	Perl
Об'єктно-орієнтованість	Так	Так	Так
Продуктивність	Висока	Нижча	Нижча
Складність розробки	Просто	Просто	Складно
Простота вивчення	Просто	Складно	Складно
Популярність	Велика	Нижче	Нижче
Спільнота підтримки	Велика	Менша	Менша
Безкоштовна ліцензія	Так	Так	Так

Як видно з табл. 3.2 мова програмування PHP має значні переваги над іншими мовами, а також вона є найбільш поширена і найчастіше використовується для розробки вебзастосунків, тому саме вона була обрана для подальшої роботи.

Оскільки вебзастосунок оперує великою кількістю інформації, необхідно підключити БД. У даному випадку була використана система керування БД (СКБД) MySQL.

MySQL – це безкоштовна БД з відкритим вихідним кодом, яка сприяє ефективному управлінню БД, підключаючи їх до програмного забезпечення; це стабільне, надійне та потужне рішення з такими розширеними функціями, як:

- безпека даних: MySQL відома у всьому світі як найбезпечніша та найнадійніша СКБД, яка використовується в популярних вебдодатках, таких як WordPress, Drupal, Joomla, Facebook і Twitter, тому безпека даних і

підтримка обробки транзакцій, які супроводжують останню версію MySQL, можуть принести велику користь будь-якому бізнесу;

- масштабованість: MySQL пропонує неперевершену масштабованість, щоб полегшити керування глибоко вбудованими програмами, використовуючи меншу площу, навіть у великих сховищах, які накопичують терабайти даних;

- висока продуктивність: MySQL має чітку структуру системи зберігання даних, яка полегшує системним адміністраторам конфігурацію сервера БД MySQL для бездоганної роботи, незалежно від того, чи це вебсайт електронної комерції, який отримує мільйон запитів щодня, чи високошвидкісна система обробки транзакцій, MySQL розроблено для задоволення навіть найвимогливіших додатків, забезпечуючи при цьому оптимальну швидкість, повнотекстові індекси та унікальний кеш-пам'яті для підвищення продуктивності;

- цілодобовий час роботи: MySQL забезпечує безперебійну роботу 24x7 і пропонує широкий спектр рішень високої доступності, таких як спеціалізовані кластерні сервери та конфігурації реплікації «головний/підпорядкований»;

- комплексна підтримка транзакцій: MySQL очолює список надійних механізмів транзакційних БД, доступних на ринку, а завдяки таким функціям, як повна атомарна, послідовна, ізольована, довговічна підтримка транзакцій, підтримка багатоверсійних транзакцій і необмежене блокування на рівні рядків, це рішення відмінно підходить для забезпечення повної цілісності даних;

- повний контроль робочого процесу: оскільки середній час завантаження та встановлення становить менше 30 хвилин, це забезпечує зручність використання з першого дня, а значить, незалежно від платформи (Linux, Microsoft, Macintosh або UNIX), MySQL – це комплексне рішення з функціями самокерування, які автоматизують все: від розширення простору та налаштування до проектування даних та адміністрування БД;

– гнучкість відкритого вихідного коду: безпечна обробка та надійне програмне забезпечення MySQL поєднуються, щоб забезпечити ефективні транзакції для великих проєктів, що робить технічне обслуговування, налагодження та оновлення швидкими та простими, одночасно покращуючи роботу кінцевого користувача [45].

Для адміністрування БД MySQL було обрано вебдодаток phpMyAdmin.

phpMyAdmin – це інструмент з відкритим вихідним кодом, написаний на мові програмування PHP, вважається найпопулярнішим додатком для керування БД MySQL [46]. По суті, це сторонній інструмент для керування таблицями та даними всередині БД, який підтримує різні типи операцій на MariaDB і MySQL, і дозволяє керувати адмініструванням MySQL через Інтернет [46].

phpMyAdmin підтримує такі функції:

- створення, редагування, перегляд та видалення БД, представлення даних, таблиць, стовпців та індексів;
- використання збережених процедур та запитів для відображення кількох наборів результатів;
- підтримка зовнішніх ключів та таблиць InnoDB;
- відстеження змін, внесених у БД, представлення і таблиці;
- створення PDF-графіка макета БД;
- експортування в різні формати, такі як XML, CSV, PDF, ISO/IEC 26300;
- підтримка mysqli, який є покращеним розширенням MySQL;
- взаємодія з 80 різними мовами;
- редагування, виконання та створення SQL запитів;
- перетворення збережених даних у будь-який формат, використовуючи набір попередньо визначених функцій;
- можливість резервного копіювання БД у різних формах [46].

До переваг використання phpMyAdmin відносяться наступні:

- phpMyAdmin може працювати на будь-якому сервері або будь-якій операційній системі;
- користувач може легко створювати, видаляти та редагувати БД, а також керувати всіма елементами за допомогою графічного інтерфейсу phpMyAdmin, що набагато простіше, ніж використання запитів SQL;
- phpMyAdmin допомагає контролювати права користувачів та працювати з кількома серверами одночасно;
- дозволяє створювати резервну копію БД і експортувати дані в різні формати, такі як XML, CSV, SQL, PDF, OpenDocument Text, Excel, Word, електронна таблиця тощо;
- дозволяє виконувати складні оператори та запити SQL, створювати та редагувати функції, тригери та події за допомогою графічного інтерфейсу phpMyAdmin [46].

Для роботи з БД було обрано бібліотеку PHP Data Objects (PDO).

PDO – це прошарок, який пропонує універсальний спосіб роботи з декількома БД [47].

Переваги використання PDO:

- об'єктно-орієнтованість: використання PDO дозволяє винести роботу з БД на об'єктно-орієнтований рівень та покращити переносимість коду;
- абстракція: використання PDO дозволяє здійснювати перехід від однієї СКБД до іншої зміною лише декількох параметрів у файлах конфігурації;
- зв'язування параметрів: використання пов'язаних параметрів надає велику гнучкість у формуванні запитів та дозволяє покращити захист від SQL ін'єкцій;
- отримання даних у вигляді об'єктів: PDO дозволяє отримувати дані у вигляді об'єктів [47].

Для створення вебсторінок застосунку було використано мову розмітки HyperText Markup Language (HTML).

HTML – це мова опису структури сторінок документів, яка дозволяє звичайний текст формувати в абзаци, заголовки, списки та інші структури, створювати посилання на інші сторінки; це текстова мова, в якій інструкції з форматування (теги) вбудовуються в розділи документа, які містять конкретну інформацію і повідомляють браузерам, як формувати і представляти інформацію на екрані [48].

Важливим чинником, який вплинув на розвиток мови HTML, став її вибір за основу для гіпертекстової бази даних звичайного текстового файлу, який можна створювати у будь-якому текстовому редакторі на будь-якій апаратній платформі у середовищі будь-якої операційної системи [48].

За основу моделі розмітки документів у HTML прийнята тегова модель, яка описує документ як сукупність контейнерів, кожен з яких починається і закінчується тегами, тобто документ HTML є не чим іншим, як звичайним ASCII-файлом з доданими до нього керуючими HTML-кодами (тегами) [48].

Теги HTML-документів в основному є простими і зрозумілими для використання, оскільки вони створені за допомогою загальноновживаних слів англійської мови, зрозумілих скорочень і позначень [48].

Для зміни зовнішнього виду вебсторінок було використано мову стилів Cascading Style Sheets (CSS).

CSS – це спеціальна мова (мова стилів), за допомогою якої описують вигляд документів (як і де відображати елементи вебсторінки), написаних мовами розмітки даних, як правило використовується для документів, котрі розмічені мовою HTML, XHTML та XML [49].

Одна з головних переваг використання CSS – це можливість відокремити зміст сторінки від її оформлення, що дозволяє покращити сприйняття та доступність змісту, забезпечити більшу гнучкість та контроль за відображенням змісту в різних умовах, зробити зміст більш структурованим та простим, прибрати повторення та виправити інші проблеми, що власне і є основною метою створення цієї технології [49].

Використання CSS дозволяє:

- відображати один і той же документ в різних стилях;
- використовувати декілька дизайнів сторінки для різних пристроїв;
- зменшити час завантаження сторінок сайту за рахунок перенесення правил відображення в окремий CSS-файл, щоб дозволити браузеру завантажувати тільки структуру документа і дані, що зберігаються на сторінці, а правила стилів цих даних завантажувати тільки один раз і кешувати;
- швидко змінювати дизайн: не потрібно правити кожну сторінку, а лише змінити CSS-файл;
- створювати складну і пропрацьовану техніку дизайну [49].

Для створення динамічних ефектів на вебсторінці було використано мову програмування JavaScript (JS).

JS – мова програмування, що дозволяє реалізувати ряд складних рішень у вебдокументах, допомагає зробити сторінки більш інтерактивними і обробляє дії користувачів сайту; це об'єктно-орієнтована клієнтська мова, яка підтримується додатками, що працюють з дизайном сайту [50].

JS містить цілий ряд функцій, які надають наступні можливості:

- зберігати дані в змінних;
- активувати частину коду у відповідності з певними сценаріями, які здійснюються на сторінці сайту;
- створювати контент, який оновлюється автоматично;
- управляти мультимедійними можливостями (працювати з відео, анімувати зображення) [50].

JS стала ще більш популярною в середовищі розробників, коли з'явилася Asynchronous Javascript And Xml (AJAX) технологія, що призвело до нового етапу в розробці сайтів [50].

AJAX – це стек технологій, за допомогою яких можна у фоновому режимі отримувати дані з сервера та використовувати їх для оновлення

сторінки без перезавантаження, що робить сайти та вебдодатки більш швидкими при взаємодії користувачів з ними [51].

Переваги використання технології AJAX:

- зменшення трафіку (через зменшення обсягу даних між клієнтом і сервером);
- зменшення навантаження на сервер (не потрібно генерувати всю сторінку, а лише ту частину, яку потрібно оновити);
- створення швидких інтерфейсів (через те, що немає потреби в повному перезавантаженні сторінки, достатньо змінити вміст лише окремих блоків);
- збільшення інтерактивності (за допомогою AJAX можна миттєво відобразити результати та підвищити зручність використання ресурсу для користувача) [51].

Для створення якісної адаптивної верстки вебзастосунку та прискорення процесу розробки було використано інструмент Bootstrap.

Bootstrap — це безкоштовний набір інструментів з відкритим кодом, призначений для створення вебсайтів та вебдодатків, який містить шаблони CSS та HTML для форм, кнопок, навігації та інших компонентів інтерфейсу, а також додаткові розширення JS, що дозволяють спростити розробку динамічних вебзастосунків [52].

Фреймворк Bootstrap володіє масою переваг:

- швидкість роботи: створення макетів Bootstrap займає менше часу завдяки великому набору готових до використання елементів;
- гнучкість: додавання нових елементів не порушує загальну структуру завдяки сітці, яка динамічно змінюється;
- легка змінюваність: редагування стилів досягається за рахунок додавання нових CSS правил, які перевизначають існуючі, при цьому, не потрібно використовувати атрибути типу !important;
- велика кількість шаблонів: шаблони у Bootstrap дозволяють змінювати вже модифіковані елементи під потреби розробника;

- величезне співтовариство прихильників та розробників;
- широкий спектр застосування: Bootstrap використовується для створення тем майже для будь-якої CMS: Magento, Joomla, WordPress та інших;
- чудова офіційна документація [52].

3.1.3 Використання менеджера пакетів

Для підключення додаткових бібліотек і їх залежностей було використано менеджер пакетів Composer.

Composer – менеджер пакетів прикладного рівня для мови програмування PHP, що забезпечує стандартний формат для управління залежностями у програмному забезпеченні та необхідними бібліотеками [53].

Composer працює з командного рядка і дозволяє користувачам встановлювати PHP пакети, доступні на «Packagist», який є його основним сховищем, а також реалізує автозавантажувач класів для встановлених бібліотек, що полегшує використання коду від сторонніх розробників [53].

Composer використовується як складова частина багатьох популярних PHP проєктів з відкритим вихідним кодом, наприклад: Magento, Laravel, Symfony [53].

За допомогою Composer було встановлено наступні пакети:

- vlucas/phpdotenv – автоматично завантажує змінні середовища з файлу .env до функції getenv(), масивів \$_ENV та \$_SERVER. Файл .env – це файл, який містить користувацькі змінні конфігурації, які необхідні програмі, без необхідності змінювати файли .htaccess або віртуальні хости Apache, а це означає, що не потрібно редагувати будь-які файли за межами проєкту, а всі змінні середовища завжди встановлюються незалежно від того, як запускається проєкт [54];
- spatie/dropbox-api – мінімальна реалізація Dropbox API v2 для використання у вебдодатках, розроблених на мові програмування PHP;

- google/apiclient – клієнтська бібліотека Google API, що дозволяє працювати з такими сервісами як Gmail, Google Drive або YouTube. Ця клієнтська бібліотека офіційно підтримуються Google.

3.1.4 Налаштування локального середовища розробки

Для того, аби мати можливість розробляти та тестувати вебдодаток локально, було встановлено набір програмного забезпечення LAMP (Linux, Apache, MySQL, PHP).

LAMP – це акронім від назв Linux, Apache, MySQL та PHP. Ці технології разом становлять надійний набір програмного забезпечення для розробки високопродуктивних вебдодатків, де кожен компонент грає свою важливу роль [55]:

- Linux – безкоштовна операційна система з відкритим кодом, яка на сьогодні є однією з найбільш поширених в усьому світі, широко використовується у всіх галузях завдяки більш гнучким функціям та можливостям налаштування порівняно з іншими операційними системами;

- Apache – надійний та багатофункціональний вебсервер з відкритим вихідним кодом, на якому зараз працює величезна кількість вебсайтів Інтернету, призначений для обробки запитів та надання вебресурсів за протоколом HTTP, щоб будь-який користувач міг отримати доступ до програми за допомогою URL-адреси;

- MySQL – реляційна СКБД з відкритим вихідним кодом, призначена для зберігання даних застосунків;

- PHP – скриптова мова програмування з відкритим кодом, разом з Apache допомагає створювати динамічні вебсторінки [55].

Розробка вебзастосунку проводилася у інтегрованому середовищі розробки (Integrated Development Environment, IDE) PhpStorm.

PhpStorm – це IDE, яке було розроблено JetBrains для PHP та веброзробників, і яке підтримує різні версії PHP, забезпечує запобігання

помилкам «на льоту», найкраще автозавершення та рефакторинг коду, налагодження конфігурації та розширений редактор HTML, CSS та JS [56].

Функції та переваги використання PhpStorm:

- підтримуються багато фреймворків PHP: при використанні фреймворків PhpStorm є ідеальним вибором, оскільки воно має вбудовану інтеграцію з багатьма популярними фреймворками, такими як Laravel, Drupal, WordPress, CakePHP, Symfony та багатьма іншими;
- інтелектуальна допомога в кодуванні: хоча PHP є основним напрямком PhpStorm, більшість розробників також працюють з іншими мовами веброзробки, такими як HTML, CSS та JS, а дане IDE полегшує розробникам роботу з іншими мовами, забезпечуючи доповнення коду та підтримку виділення синтаксису;
- підтримка БД: за допомогою PhpStorm легко впроваджувати БД і код SQL, тому що IDE надає різні інструменти та допоміжні функції коду, що дозволяють підключатися до БД, редагувати таблиці всередині, виконувати різні запити і навіть аналізувати схеми за допомогою UML-діаграм;
- інтеграція Git та GitHub: наявність функцій розгалуження та злиття Git і GitHub в IDE є дуже важливим функціоналом для спільноти програмістів, також підтримуються інші методи контролю версій, наприклад підтримка FTP, що дозволяє завантажувати код на віддалений сервер за кілька секунд;
- налагодження, тестування та профілювання: PhpStorm також надає безліч опцій для налагодження коду за допомогою Visual Debugger, що дозволяє перевіряти змінні, встановлювати точки зупину та оцінювати вирази під час виконання, а також налагоджувати віддалені програми, налагоджувати сторінку в кількох сесіях одночасно та багато іншого;
- великий форум спільноти: це IDE є дуже популярним, а значить існує великий форум спільноти, де розробник може поділитися своєю роботою, побачити чужий код і отримати допомогу від професіоналів у разі виникнення проблеми [56].

3.2 Визначення функціональних вимог до програмної системи

Після проведення аналізу предметної області та порівняння програм-аналогів, було визначено наступні вимоги до розроблюваного програмного забезпечення:

а) вимоги лікаря:

- 1) авторизація за допомогою електронної пошти та пароля;
- 2) перегляд списку пацієнтів на поточний день;
- 3) перегляд інформації про конкретного пацієнта;
- 4) додавання записів про прийом (скарги хворого, аналізи, діагноз, призначення);
- 5) завантаження зображення захворювання з комп'ютера та/або хмарних сховищ, таких як Google Drive та Dropbox;
- 6) діагностування захворювання за завантаженим зображенням;
- 7) перегляд медичної карти хворого;
- 8) пошук даних у медкартці;
- 9) запис пацієнта на наступний прийом з можливістю вибору дати та часу;
- 10) перегляд та редагування особистої інформації лікаря;
- 11) налаштування хмарних сховищ (Google Drive та Dropbox);

б) вимоги пацієнта:

- 1) реєстрація та авторизація пацієнтів;
- 2) діагностування захворювання за завантаженим зображенням;
- 3) перегляд інформації про наявні записи на прийом до лікаря;
- 4) створення нових записів на прийом з можливістю вибору лікаря, дати та часу;
- 5) видалення непотрібних записів на прийом;
- 6) перегляд медичної карти;

- 7) пошук даних у медичній карті;
- 8) перегляд та редагування особистої інформації;
- в) загальні вимоги до системи:
- 1) використання розробленого методу для здійснення діагностування шкірного захворювання за завантаженим зображенням;
 - 2) обробка помилкових дій користувача з подальшим інформування шляхом виводу відповідних повідомлень;
 - 3) обробка виняткових ситуацій, таких як помилки при виконанні запитів до БД;
 - 4) розподіл доступу до сторінок системи за ролями користувачів;
 - 5) взаємодія з системою на різних пристроях (мобільні телефони, планшети, комп'ютери);
 - 6) зручний та інтуїтивно зрозумілий інтерфейс.

Діаграма прецедентів для лікаря наведена на рис. 3.1.

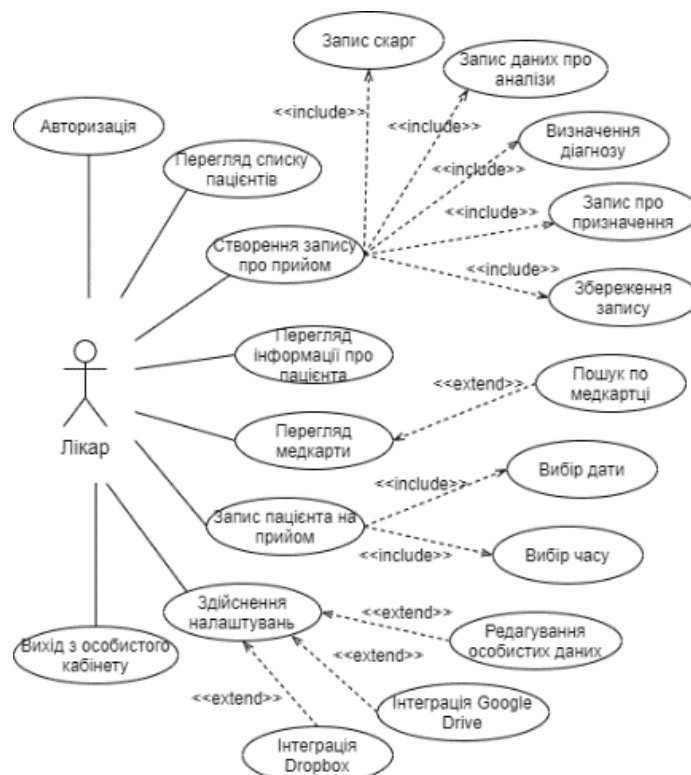


Рисунок 3.1 – Діаграма прецедентів для лікаря

Діаграма прецедентів для пацієнта наведена на рис. 3.2. Діаграма прецедентів для визначення діагнозу наведена на рис. 3.3.

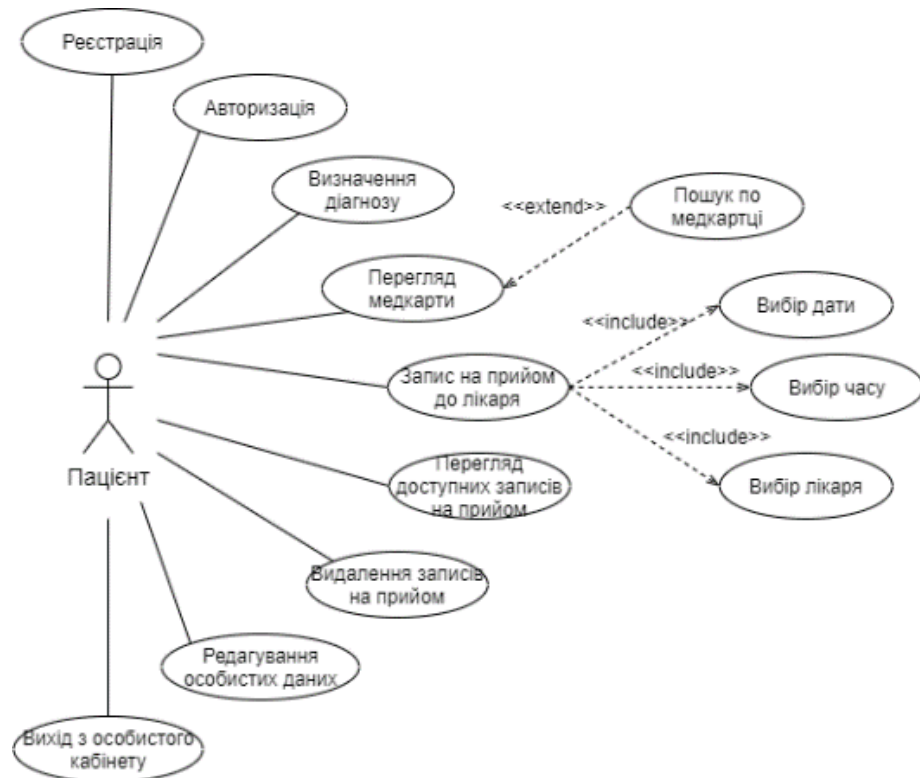


Рисунок 3.2 – Діаграма прецедентів для пацієнта

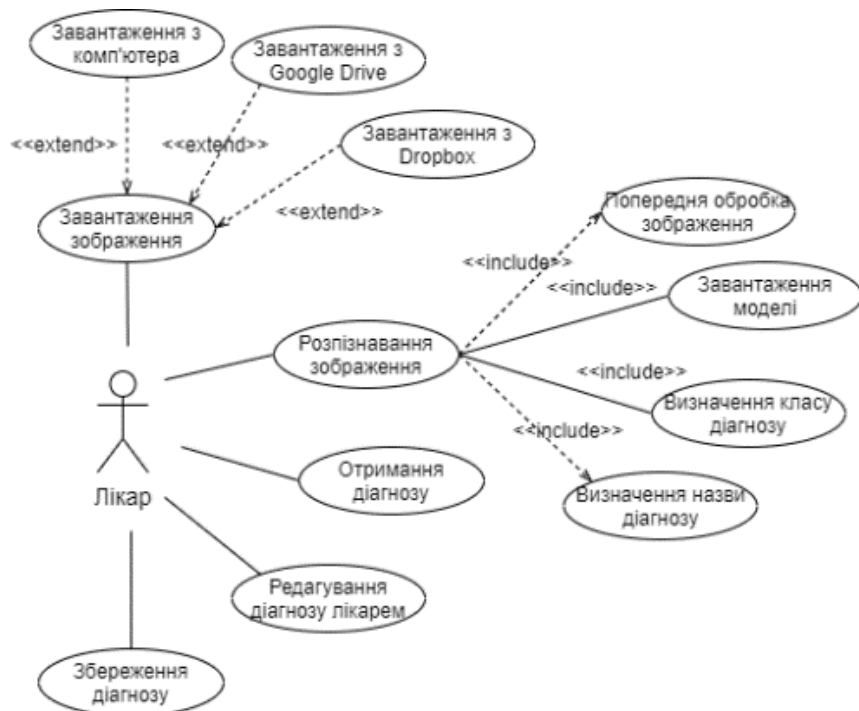


Рисунок 3.3 – Діаграма прецедентів для визначення діагнозу лікарем

3.3 Проектування структури бази даних

Оскільки розроблюване програмне забезпечення повинно працювати з великою кількістю даних, то необхідно забезпечити зберігання цих даних.

Для цього було спроектовано БД, що складається з наступних таблиць:

- patient містить інформацію про пацієнта;
 - doctor містить інформацію про лікаря;
 - appointment містить інформацію про запис на прийом;
 - appointment_type містить інформацію про тип послуги;
 - appointment_repetition містить інформацію про тип прийому;
 - timetable містить інформацію про розклад лікаря;
 - medical_card містить інформацію про медичну картку пацієнта;
 - image містить інформацію про завантажені зображення захворювань;
 - settings містить інформацію про налаштування хмарних сховищ.
- Опис атрибутів кожної таблиці БД наведено у табл. 3.3.

Таблиця 3.3 – Опис атрибутів таблиць БД

Таблиця	Атрибут	Тип атрибуту	Опис атрибуту
patient	id	int	Ідентифікатор
	surname	varchar	Прізвище
	name	varchar	Ім'я
	fatherName	varchar	По батькові
	email	varchar	Електронна пошта
	password	varchar	Пароль
	genderId	int	Ідентифікатор статі
	dateOfBirthday	date	Дата народження
	phone	varchar	Телефон
	address	varchar	Адреса

Продовження таблиці 3.3

Таблиця	Атрибут	Тип атрибуту	Опис атрибуту
doctor	id	int	Ідентифікатор
	surname	varchar	Прізвище
	name	varchar	Ім'я
	fatherName	varchar	По батькові
	email	varchar	Електронна пошта
	password	varchar	Пароль
appointment	id	int	Ідентифікатор
	doctorId	int	Ідентифікатор лікаря
	patientId	int	Ідентифікатор пацієнта
	type	int	Тип послуги
	repetition	int	Тип прийому
	date	date	Дата прийому
appointment_type	time	time	Час прийому
	id	int	Ідентифікатор
appointment_repetition	name	varchar	Назва послуги
	id	int	Ідентифікатор
timetable	name	varchar	Назва типу прийому
	id	int	Ідентифікатор
	doctorId	int	Ідентифікатор лікаря
settings	date_time	text	Розклад
	id	int	Ідентифікатор
	doctorId	int	Ідентифікатор лікаря
	type	varchar	Тип сховища
	status	varchar	Статус
	credentials	text	Дані для інтеграції хмарних сховищ

Продовження таблиці 3.3

Таблиця	Атрибут	Тип атрибуту	Опис атрибуту
image	id	int	Ідентифікатор
	name	varchar	Назва
	size	int	Розмір
	type	varchar	Тип
medical_card	id	int	Ідентифікатор
	patientId	int	Ідентифікатор пацієнта
	date	date	Дата прийому
	diagnose	varchar	Діагноз
	imageId	int	Ідентифікатор зображення
	complaint	text	Скарги хворого
	analyzes	text	Результати аналізів
	prescription	text	Призначення лікаря

Логічна структура розробленої БД наведена на рис. 3.4.

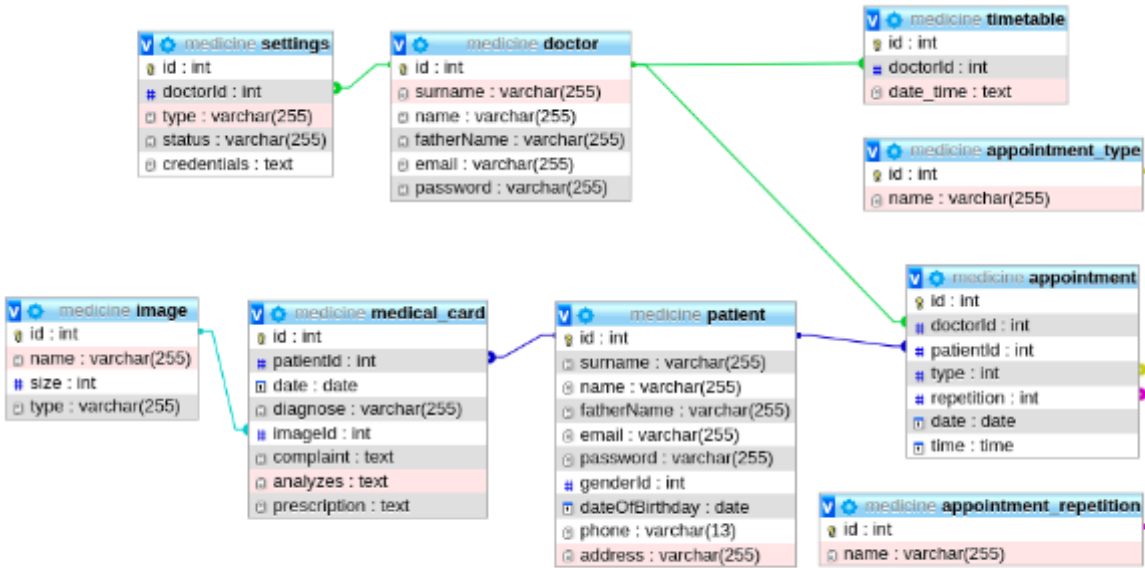


Рисунок 3.4 – Логічна структура розробленої БД

3.4 Проектування архітектури програмного забезпечення

З огляду на те, що було вирішено розробити програмне забезпечення у вигляді вебзастосунку, то у якості архітектурного шаблону використовувався один з найпопулярніших шаблонів – Model-View-Controller (MVC).

MVC – методологія або архітектурний шаблон, який використовується для ефективного відокремлення бізнес-логіки від користувацького інтерфейсу програми і складається з наступних компонентів:

- модель: відповідає за керування даними, оскільки вона зберігає та витягує сутності, що використовуються додатком, зазвичай з БД, і містить логіку, таку як отримання інформації, оновлення, запис, видалення;
- вид: відповідає за відображення даних, що надаються моделлю, тобто це графічний інтерфейс кінцевого користувача, за допомогою якого користувач може взаємодіяти з додатком;
- контролер: містить бізнес-логіку та забезпечує зв'язок між моделлю та видом, а саме отримує запити від користувача, викликає модель для виконання необхідних операцій та відправляє дані у вид [57].

Схема шаблону MVC наведена на рис.3.5.

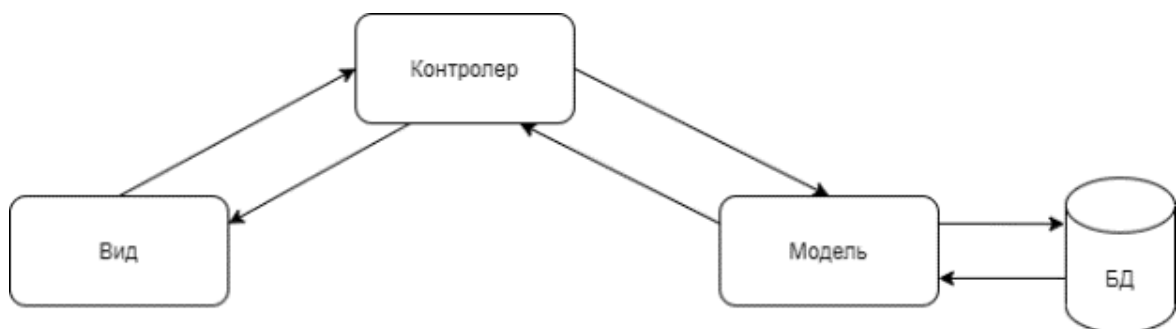


Рисунок 3.5 – Схема шаблону MVC

3.5 Визначення структури програмного забезпечення

Оскільки для розробки програмного забезпечення було обрано шаблон MVC, структурна схема система представлена у вигляді наведеному на рис. 3.6.

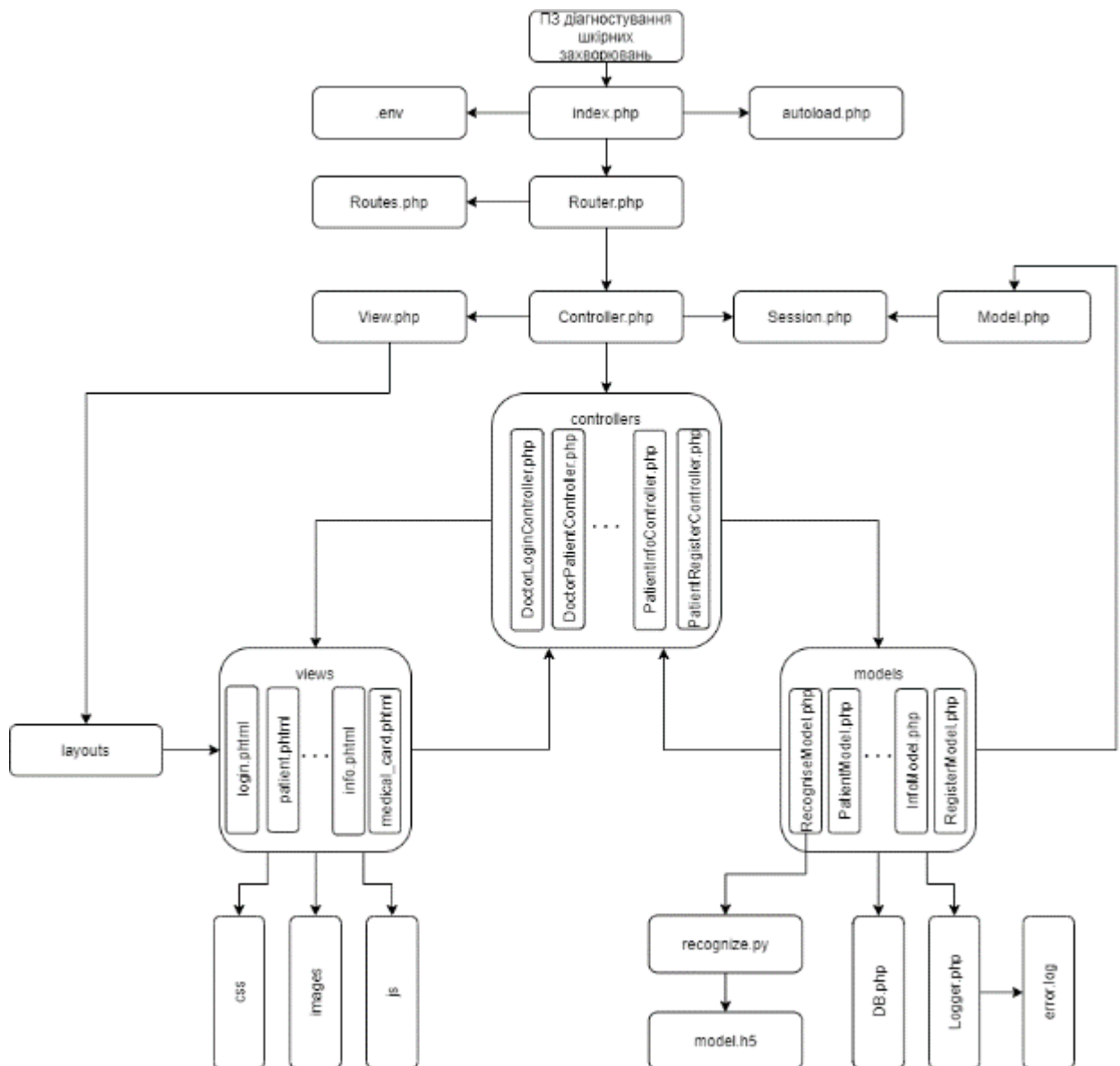


Рисунок 3.6 – Структурна схема системи

Файл index.php є вхідною точкою для усіх сторінок вебзастосунку. У ньому підключається файл автозавантаження класів, завантажується дані файлу .env і створюється об'єкт класу роутера.

Роутер використовує доступні роути з файлу `Routes.php`, створює об'єкти контролерів та викликає функції цих контролерів відповідно до отриманої URL-адреси.

Файли `Contoller.php` та `Model.php` містять базові класи для контролерів та моделей відповідно. У даних класах описано спільні базові методи для всіх класів спадкоємців. Файл `View.php` містить клас `View`, призначений для формування відповідного виду шляхом об'єднання потрібного `layout` та отриманого файлу виду.

Класи моделей та контролерів використовують клас `Session` для збереження та отримання даних із сесії, наприклад, для передачі повідомлень про помилки чи успішне виконання дії.

Папки `controllers`, `models` та `views` містять відповідно розроблені контролери, моделі та види.

Види використовують файли `.css` та `.js`, а також картинки з папки `images`.

Класи моделей використовують клас `DB` для взаємодії з БД, а також клас `Logger` для запису помилок у лог (файл `error.log`).

У моделі `RecogniseModel.php` викликається скрипт `recognize.py`, у якому підключається модифікована модель нейронної мережі `model.h5`. Скрипт отримує шлях до зображення, яке потрібно розпізнати, та повертає назву діагнозу.

3.6 Висновки до третього розділу

У даному розділі було обрано засоби проектування для подальшої розробки програмного забезпечення, а саме:

- для нейронної мережі: мову програмування Python, бібліотеки TensorFlow, Keras, NumPy, Pandas, Matplotlib, Scikit Learn, os;

- для розробки вебзастосунку: мови програмування PHP та JS, мову розмітки HTML, мову стилів CSS, інструмент Bootstrap, менеджер пакетів Composer;
- для роботи з БД: СКБД MySQL, інструмент phpMyAdmin, бібліотеку PDO;
- у якості середовища розробки – PhpStorm.

Також було визначено основні функціональні вимоги до розроблюваної системи, виділено дві групи користувачів, а саме лікарі та пацієнти, та побудовано діаграми прецедентів на основі вимог кожної групи користувачів.

Оскільки система передбачає роботу з великою кількістю даних, було спроектовано БД, описано таблиці та відповідні атрибути.

Для розробки системи обрано архітектурний шаблон MVC, на основі якого побудовано структурну схему розроблюваного програмного забезпечення та описано основні її складові.

4 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

4.1 Алгоритм функціонування програми

На основі функціональних вимог, визначених у попередньому розділі, було побудовано алгоритм функціонування програми (рис. 4.1).

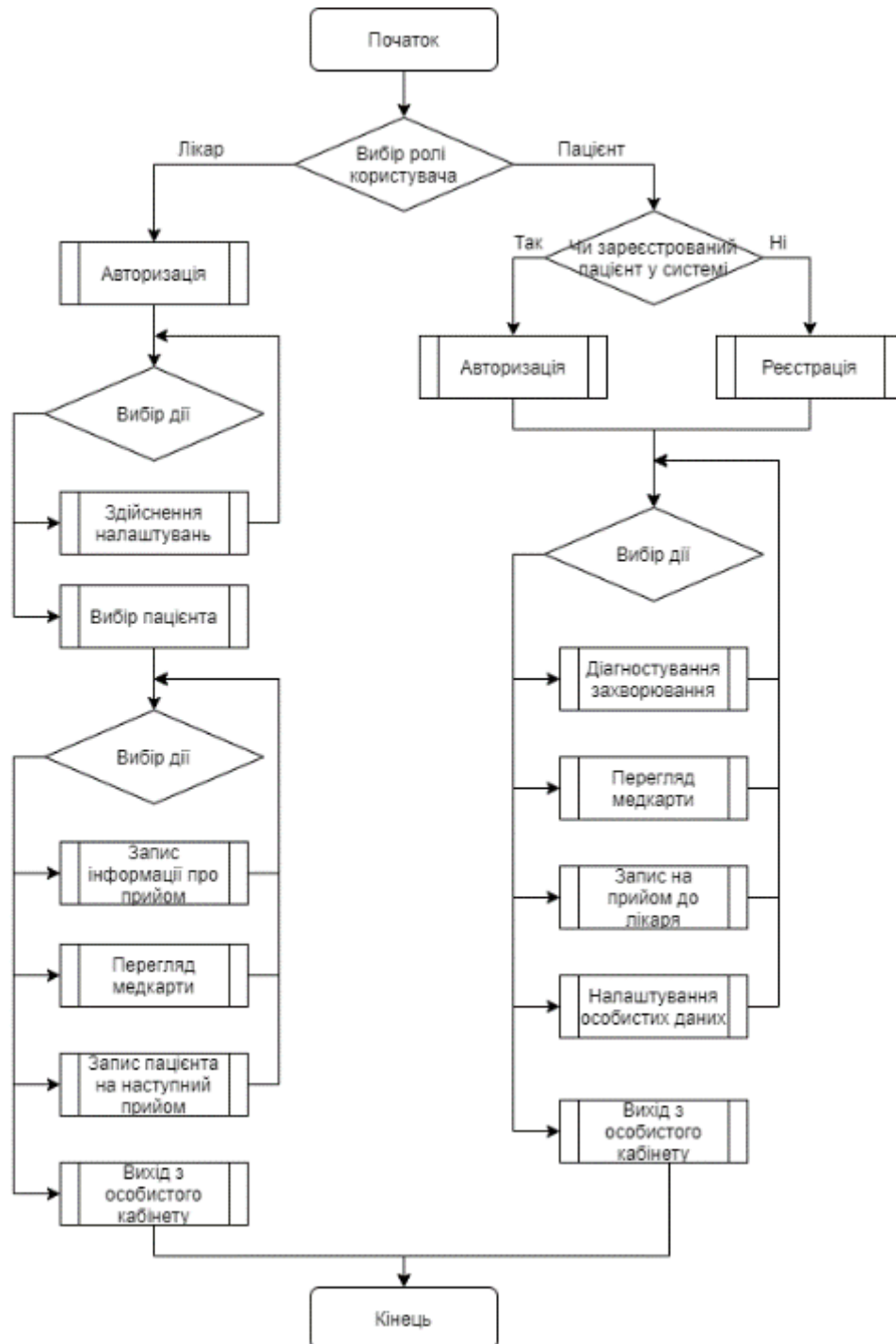


Рисунок 4.1 – Алгоритм функціонування програми

Для початку роботи з системою лікар має пройти авторизацію, після чого йому буде доступний перелік пацієнтів на поточний день. Лікар може або обрати пацієнта або перейти до сторінки налаштувань.

Налаштування дозволяють виконати наступні дії:

- відредагувати особисту інформацію;
- налаштувати інтеграцію Google Drive;
- налаштувати інтеграцію Dropbox.

Якщо лікар обрав пацієнта, то його буде перенаправлено на сторінку прийому, де він може виконувати такі дії:

- додати новий запис про прийом;
- перейти на сторінку медичної карти пацієнта;
- перейти на сторінку запису пацієнта на наступний прийом.

Усі сторінки вебзастосунку поєднані між собою, тобто користувач може з легкістю перейти від однієї до іншої.

Для роботи пацієнта з системою йому потрібно пройти авторизацію або реєстрацію (у разі відсутності облікового запису). Після цього пацієнт може переходити між наступними сторінками:

- сторінка діагностування захворювання за завантаженим зображенням;
- сторінка з записами на прийом (з можливістю створювати та видаляти записи);
- сторінка перегляду медичної карти;
- сторінка редагування особистої інформації.

Робота з системою закінчується після того, як користувач вийшов з облікового запису, для поновлення роботи необхідно повторно пройти авторизацію.

4.2 Опис модулів реалізованої програмної системи

У даному пункті розглянуто основні папки, класи та методи, які було створено у процесі розробки програмного забезпечення.

Папка `core` містить класи `Controller`, `DB`, `Logger`, `Model`, `Router`, `Routes`, `Session`, `View`.

Клас `Controller` – базовий клас для усіх контролерів системи, містить такі методи:

- `renderView($view, $params = [], $layout = '')` – для формування виду;
- `redirect($url)` – для здійснення перенаправлення;
- `render404()` – для формування сторінки 404;
- `getParams()` – для отримання параметрів з глобальних масивів `$_GET` і `$_POST`;
- `method()` – повертає метод (`get` чи `post`);
- `checkLogin($role)` – для перевірки авторизації користувача.

Клас `DB` призначено для роботи з БД, він містить функцію `getConnection()`, у якій створюється та повертається об'єкт класу `PDO`.

Клас `Logger` призначено для запису повідомлень про помилки у лог (файл `error.log`), містить метод `getLogger()` для створення об'єкта логера на основі класу `MonologLogger` та метод `error($message)` – для запису помилок.

Клас `Model` є базовим для класів моделей та містить наступні методи:

- `validate($model)` – для валідації даних;
- `addError($attribute, $rule)` – для додавання валідаційних помилок;
- `errorMessage()` – для додавання повідомлення про помилки;
- `login($object)` – для запису даних у сесію після авторизації користувача;
- `logout($role)` – для очищення сесії після виходу користувача з особистого кабінету.

Клас Router призначено для здійснення маршрутизації, надає можливість створення об'єкта потрібного контролера та виклику відповідного методу. Даний клас містить такі методи:

- `__construct()` – конструктор класу;
- `run()` – для отримання контролера, створення об'єкта контролера та виклику потрібного методу;
- `getURL()` – для отримання та попередньої обробки URL-адреси.

Клас Routes містить метод `getRoutes`, який повертає масив контролерів та методів для відповідної URL-адреси.

Клас Session призначено для роботи з сесіями та містить наступні методи:

- `__construct()` – конструктор класу;
- `getSession()` – для отримання класу сесії;
- `setFlash($key, $message)` – для встановлення flash повідомлень;
- `getFlash($key)` – для отримання flash повідомлень;
- `set($key, $value)` – для запису значення в сесію;
- `get($key)` – для отримання значення з сесії;
- `remove($key)` – для видалення значення з сесії;
- `__destruct()` – деструктор класу.

Клас View призначено для формування видів системи, містить такі методи:

- `setLayout($layout)` – для встановлення layout;
- `renderView($view, $params = [])` – для формування виду;
- `layoutContent()` – для підключення layout та підготовки flash повідомлень;
- `renderOnlyView($view, $params = [])` – для встановлення змінних для виду.

Папка `controllers` містить файли контролерів та відповідні підпапки `doctor` та `patient`.

Кожен контролер містить метод `__construct()` – конструктор класу, який використовується для ініціалізації атрибутів об'єкта та метод `index` – головний метод, який викликається при завантаженні відповідної сторінки, оперує даними з моделі та підключає потрібний вид. Оскільки усі контролери містять у собі описані вище методи, то у подальшому буде вказано лише додаткові методи кожного контролера.

Клас `DoctorAppointmentController` призначено для роботи лікаря з записами на прийом. Метод `save($patientId)` призначено для збереження даних про запис пацієнта на прийом.

Клас `DoctorLoginController` призначено для здійснення авторизації лікаря.

Клас `DoctorMainController` призначено для обробки сторінки зі списком пацієнтів на поточний день.

Клас `DoctorMedicalCardController` призначено для обробки сторінки медичної карти.

Клас `DoctorPatientController` призначено для обробки сторінки прийому пацієнта, містить такі методи:

- `saveCardRecord($patientId)` – для обробки запису про прийом;
- `recognise()` – для розпізнавання зображення захворювання;
- `saveImage()` – для збереження зображення.

Клас `DoctorSettingsController` призначено для обробки сторінки з налаштуваннями, містить метод `saveCredentials($type)` для збереження даних для інтеграції хмарних сховищ.

Клас `PatientDiagnoseController` призначено для обробки сторінки діагностування захворювання за завантаженим зображенням. Містить методи `saveImage()` для збереження зображення та `recognise()` для розпізнавання зображення.

Клас `PatientInfoController` призначено для обробки сторінки з особистою інформацією пацієнта.

Клас `PatientLoginController` призначено для обробки сторінки авторизації пацієнта.

Клас `PatientMainController` призначено для обробки сторінки пацієнта, на якій можна здійснити запис на прийом до лікаря. Містить наступні методи:

- `timetable()` – для отримання розкладу лікаря;
- `saveAppointment()` – для збереження запису на прийом;
- `deleteAppointment($id)` – для видалення запису про прийом.

Клас `PatientMedicalCardController` призначено для обробки сторінки медичної карти пацієнта.

Клас `PatientRegisterController` призначено для обробки реєстрації пацієнта.

Папка `models` містить розроблені моделі та складається з наступних папок: `base` (базові моделі, які описують сутність лікаря та пацієнта), `doctor` (моделі для лікаря), `patient` (моделі для пацієнта), `general` (загальні моделі для лікаря і пацієнта).

Клас `DoctorModel` описує сутність лікаря і містить наступні методи:

- `getSurname()` – для отримання прізвища;
- `setSurname($surname)` – для встановлення прізвища;
- `getName()` – для отримання імені;
- `setName($name)` – для встановлення імені;
- `getFatherName()` – для отримання по батькові;
- `setFatherName($fatherName)` – для встановлення по батькові;
- `getEmail()` – для отримання електронної пошти;
- `setEmail($email)` – для встановлення електронної пошти;
- `getPassword()` – для отримання пароля;
- `setPassword($password)` – для встановлення пароля;
- `getTableName()` – для отримання назви таблиці БД;
- `getRole()` – для отримання значення ролі користувача.

Клас `PatientModel` описує сутність пацієнта і містить наступні методи:

- `getSurname()` – для отримання прізвища;
- `setSurname($surname)` – для встановлення прізвища;
- `getName()` – для отримання імені;
- `setName($name)` – для встановлення імені;
- `getFatherName()` – для отримання по батькові;
- `setFatherName($fatherName)` – для встановлення по батькові;
- `getEmail()` – для отримання електронної пошти;
- `setEmail($email)` – для встановлення електронної пошти;
- `getPassword()` – для отримання пароля;
- `setPassword($password)` – для встановлення пароля;
- `getGenderId()` – для отримання ідентифікатора статі;
- `setGenderId($genderId)` – для встановлення ідентифікатора статі;
- `getDateOfBirthday()` – для отримання дати народження;
- `setDateOfBirthday($dateOfBirthday)` – для встановлення дати

народження;

- `getPhone()` – для отримання номера телефону;
- `setPhone($phone)` – для встановлення номера телефону;
- `getAddress()` – для отримання адреси;
- `setAddress($address)` – для встановлення адреси;
- `getTableName()` – для отримання назви таблиці БД;
- `getRole()` – для отримання значення ролі користувача.

Клас `DropboxModel` використовується для роботи з `Dropbox API`, містить такі методи:

- `getDropboxClient()` – для створення об'єкта `Dropbox Client`;
- `getAccessToken()` – для отримання токена доступу;
- `loadImage($link)` – для завантаження зображення зі сховища

`Dropbox`.

Клас `GoogleDriveModel` використовується для роботи з Google Drive API, містить такі методи:

- `getClient($token)` – для створення об'єкта класу `Google_Client`;
- `loadImage($fileId, $token)` – для завантаження зображення зі сховища Google Drive;
- `getImageContent($fileId)` – для отримання контенту зображення;
- `getMetadata($fileId)` – для отримання метаданих зображення;
- `getCredentials()` – для отримання даних доступу до сховища.

Клас `InfoModel` містить методи для роботи з особистими даними користувача:

- `rules()` – повертає правила для здійснення валідації при редагуванні особистих даних;
- `getInfo($id)` – повертає значення особистих даних;
- `editInfo($id, $params)` – зберігає оновлені дані.

Клас `MedicalCardModel` містить метод `getInfo()` для отримання даних медичної карти пацієнта.

Клас `DoctorPatientModel` призначено для обробки даних, отриманих під час прийому пацієнта лікарем, містить метод `loadImage($imageMetadata)` для завантаження зображення та `saveRecord($params)` для збереження запису про прийом.

Клас `SettingsModel` містить метод `saveCredentials($doctorId, $params, $type)` для збереження доступів до хмарних сховищ та метод `getCredentials($doctorId)` для отримання доступів до хмарних сховищ.

Клас `doctor/MainModel` містить метод `getListOfPatients($doctorId)` для обробки сторінки з переліком пацієнтів на поточний день.

Клас `patient/MainModel` містить методи для створення запису на прийом на сторінці пацієнта:

- `getAllDoctors()` – для отримання переліку лікарів;
- `getAvailableTime($doctorId)` – для отримання переліку вільних дат і часу;

- `getTimetable($doctorId)` – для отримання розкладу лікаря;
- `getAppointments($doctorId, $date)` – для отримання переліку зайнятих дат і часу;
- `saveAppointment($params)` – для збереження запису;
- `getAppointmentsForPatient($patientId)` – для отримання переліку записів для конкретного пацієнта;
- `deleteAppointment($id)` – для видалення запису.

Клас `RegisterModel` призначено для обробки реєстрації пацієнта, містить метод `rules()`, який повертає перелік правил для валідації, та метод `register()` – для збереження даних користувача.

Клас `AppointmentInfoModel` описує додаткові параметри, які вказуються при створенні запису на прийом. Містить метод `getTypes()` та `getRepetition()` для отримання послуги та типу прийому.

Клас `Image` призначено для роботи з зображеннями, містить методи такі методи:

- `loadImage($imageMetadata)` – для завантаження зображення;
- `saveImageInFolder($image, $metadata)` – для збереження зображення у папку;
- `saveImageInDB($image)` – для збереження інформації про зображення у БД.

Клас `LoginModel` призначено для обробки авторизації користувачів, містить метод `rules()`, який повертає перелік правил для валідації форм авторизації та `loginUser()` – здійснює перевірку даних користувача у БД.

Клас `RecogniseModel` містить метод `callNeuralNetwork($imageName)` для виклику моделі нейронної мережі.

Папка `pub` містить такі папки, як `css` (файли стилів), `image` (файли зображень), `js` (файли JS), `python` (скрипт Python та модель нейронної мережі `.h5`).

Папка `var` містить лог файл `error.log`, у який записуються помилки, отримані в процесі роботи системи.

В папці `vendor` містяться файли сторонніх бібліотек (наприклад, `spatie/dropbox-api` та `google/apiclient`), а також файли для автозавантаження класів.

4.3 Висновки до четвертого розділу

У даному розділі було визначено алгоритм функціонування розроблюваного програмного забезпечення.

Також виділено та детально описано основні папки, класи та функції, які було створено для реалізації програмного забезпечення.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Опис застосування програми

Розроблене програмне забезпечення призначене для використання у медичній сфері, зокрема для діагностування шкірних захворювань, а також дозволяє виконувати додаткові функції, пов'язані з веденням медичної карти пацієнта і оформленням записів на прийом.

5.2 Умови виконання програми

Для розміщення розробленої програми у мережі Інтернет необхідно виділити вебвузол і встановити на ньому:

- вебсервер Apache;
- інтерпретатор PHP 7.4;
- менеджер пакетів Composer;
- БД MySQL.

Для використання розробленої системи необхідні:

- процесор від 1.0 ГГц;
- обсяг оперативної пам'яті від 4 ГБ;
- веббраузер (Google Chrome, Mozilla Firefox, Opera та ін.);
- наявність підключення до мережі Інтернет.

5.3 Інструкція по експлуатації програми

Взаємодія з системою відбувається через мережу Інтернет за допомогою браузера. Для роботи лікаря з програмним забезпеченням йому необхідно пройти процес авторизації (рис. 5.1).

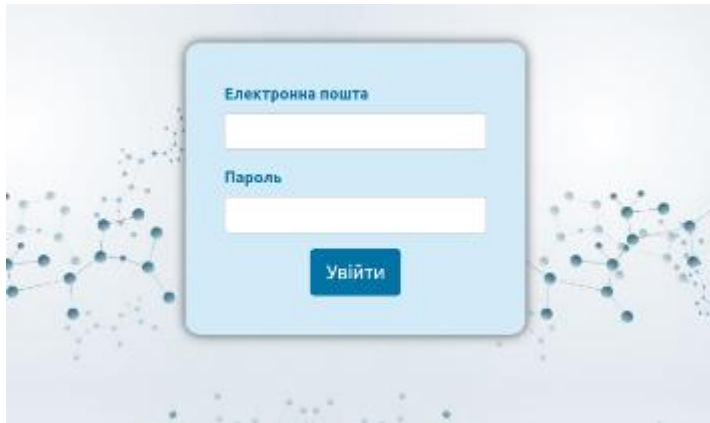


Рисунок 5.1 – Сторінка авторизації лікаря

Після цього лікаря буде перенаправлено на сторінку зі списком пацієнтів на поточний день (рис. 5.2). Він може або обрати потрібного пацієнта або перейти до сторінки налаштувань.



Рисунок 5.2 – Сторінка з переліком пацієнтів

На сторінці налаштувань лікар може виконати редагування особистої інформації (рис. 5.3), підключити інтеграцію Google Drive (рис. 5.4) чи Dropbox (рис 5.5).

The screenshot shows a web interface for editing personal information. On the left, a sidebar contains three buttons: 'Особиста інформація' (selected), 'Google Drive', and 'DropBox'. The main form area has a blue header 'Електронна пошта'. Below it are input fields for 'test@gmail.com', a password field with masked characters, and fields for 'Прізвище' (Sergiienko), 'Ім'я' (Yaroslav), and 'По батькові' (Oleksandrovich). A 'Зберегти' (Save) button is at the bottom right. The background features a molecular structure pattern.

Рисунок 5.3 – Редагування особистої інформації

The screenshot shows the 'Google Drive Integration' form. The sidebar on the left has 'Особиста інформація', 'Google Drive' (selected), and 'DropBox'. The main form has a blue header 'Інтеграція з Google Drive'. Below it is a toggle switch labeled 'Увімкнути інтеграцію' which is turned on. There are input fields for 'Developer Key', 'Client ID', and 'App ID'. A 'Зберегти' (Save) button is at the bottom right. The background features a molecular structure pattern.

Рисунок 5.4 – Налаштування інтеграції Google Drive

The screenshot shows the 'Dropbox Integration' form. The sidebar on the left has 'Особиста інформація', 'Google Drive', and 'DropBox' (selected). The main form has a blue header 'Інтеграція з Dropbox'. Below it is a toggle switch labeled 'Увімкнути інтеграцію' which is turned on. There is a large text area for 'Access Token'. A 'Зберегти' (Save) button is at the bottom right. The background features a molecular structure pattern.

Рисунок 5.5 – Налаштування інтеграції Dropbox

Після того, як лікар обере потрібного пацієнта, його буде перенаправлено на сторінку, призначену для оформлення запису про прийом (рис. 5.6). Тут лікар може додати скарги пацієнта, інформацію про аналізи, визначити діагноз та записати зроблені призначення.

Дата прийому: 02.11.2021

Скарги хворого

← → Paragraph B I [Formatting icons]

Р [Text area] POWERED BY TINY

Аналізи

← → Paragraph B I [Formatting icons]

Р [Text area] POWERED BY TINY

Діагноз

Діагноз

Обраний файл [File upload icon] [Google Drive icon] [Dropbox icon]

Обрати файл

Призначення

← → Paragraph B I [Formatting icons]

Р [Text area] POWERED BY TINY

Зберегти

Кравчук Роман Євгенійович

Дата народження: 07.09.1981

Стать: Чоловік

Телефон: +380630377780

Email: test@gmail.com

Адреса: Запоріжжя, вул. Поштова, 1

Медкартка

Талон

Рисунок 5.6 – Сторінка створення запису про прийом

Для визначення діагнозу за допомогою використання нейронної мережі лікарю необхідно завантажити дерматоскопічне зображення захворювання. Для цього можна використати файл зображення, який

зберігається на комп'ютері (рис. 5.7), у Google Drive (рис. 5.8) чи Dropbox (рис. 5.9).

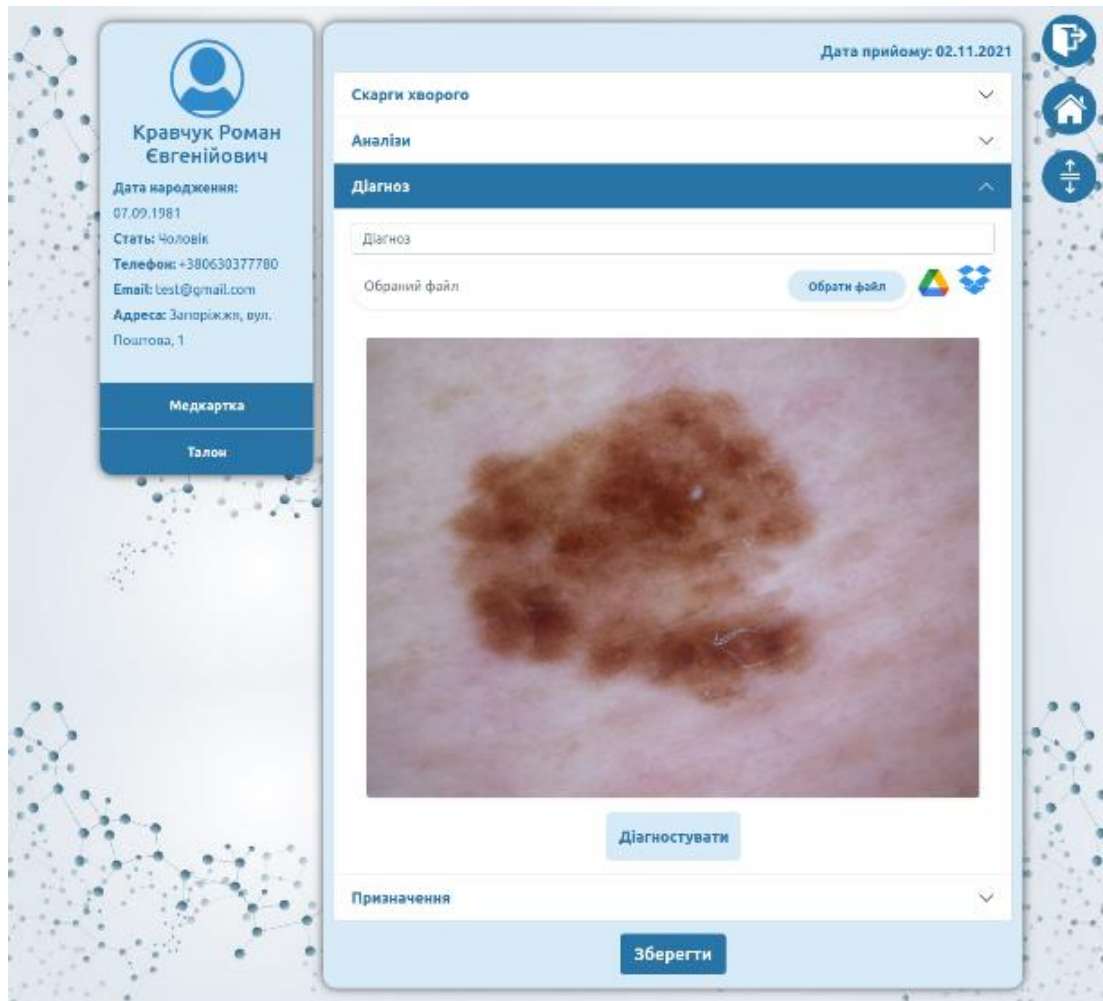


Рисунок 5.7 – Завантаження зображення з комп'ютера

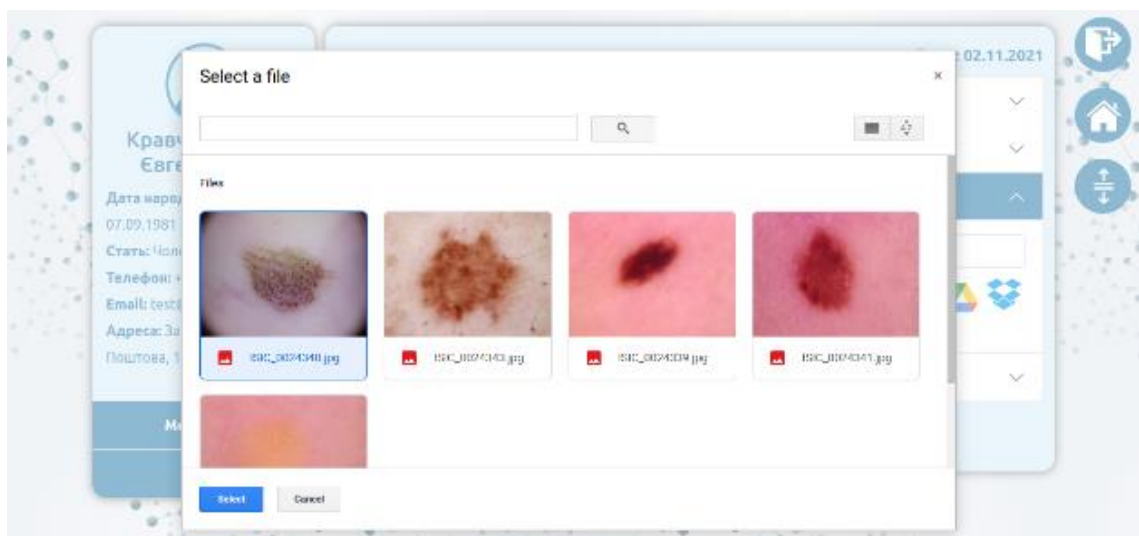


Рисунок 5.8 – Завантаження зображення з Google Drive

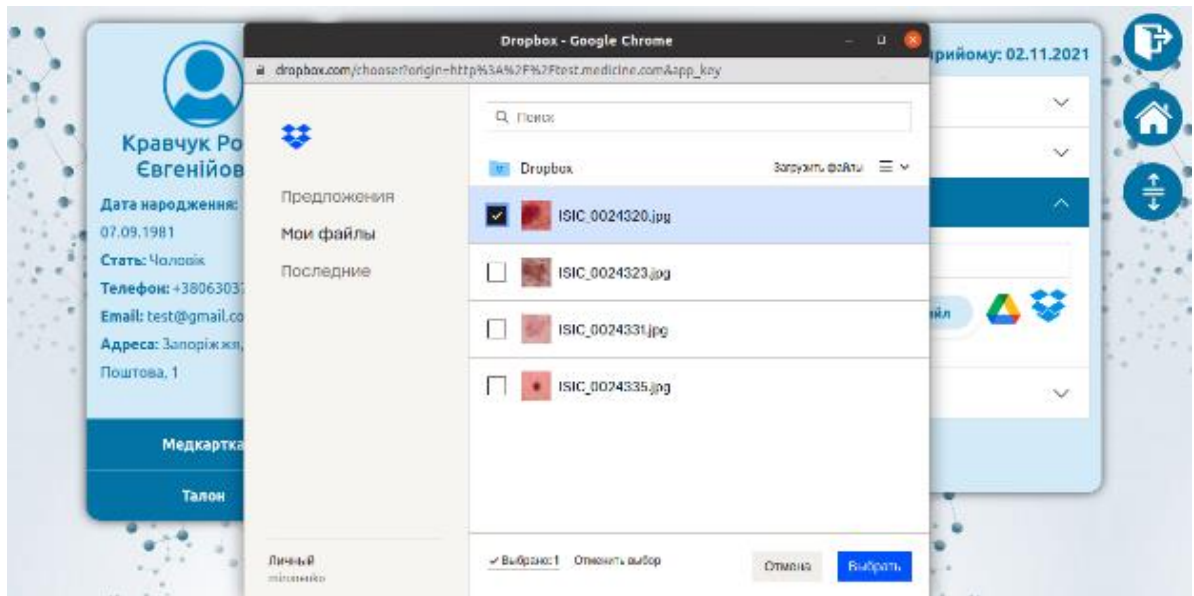


Рисунок 5.9 – Завантаження зображення з Dropbox

Після завантаження зображення необхідно натиснути кнопку «Діагностувати», після чого буде визначено попередній діагноз за допомогою нейронної мережі (рис. 5.10).

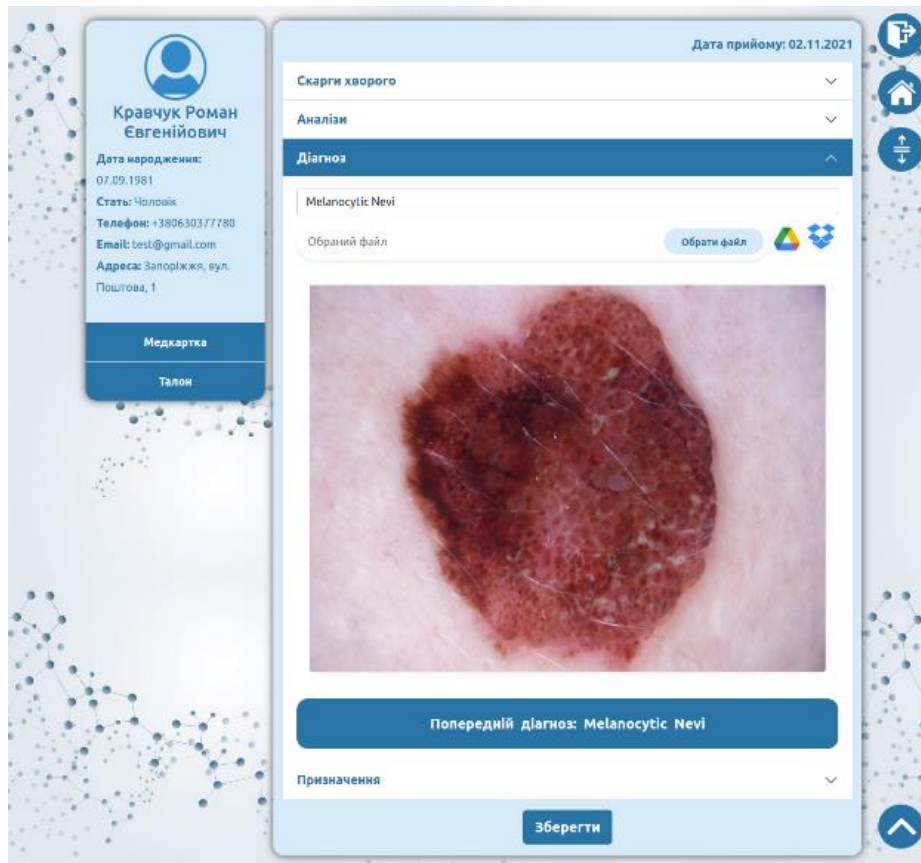


Рисунок 5.10 – Визначення діагнозу за допомогою нейронної мережі

Лікар також має змогу переглянути медичну карту пацієнта (рис. 5.11).



Рисунок 5.11 – Сторінка медичної карти пацієнта

Ще одна функція, яку може виконати лікар, – запис пацієнта на наступний прийом (рис. 5.12). Лікар має можливість обрати доступну дату та час, а також визначити додаткові характеристики прийому.

Кравчук Роман Євгенійович
 Дата народження: 07.09.1981
 Стать: Чоловік
 Телефон: +380630377781
 Email: test@gmail.com
 Адреса: Запоріжжя, вул. Поштова, 1

Медкартка
Прийом

Послуги: Консультації
 Тип прийому: Повторний

04.11.2021

13:00	13:30	14:00	14:30	15:00	15:30	16:00	16:30
17:00	17:30						

05.11.2021

08:00	08:30	09:00	09:30	10:00	10:30	11:00	11:30
12:00	12:30						

08.11.2021

08:00	08:30	09:00	09:30	10:00	10:30	11:00	11:30
12:00	12:30						

09.11.2021

13:00	13:30	14:00	14:30	15:00	15:30	16:00	16:30
17:00	17:30						

10.11.2021

08:00	08:30	09:00	09:30	10:00	10:30	11:00	11:30
12:00	12:30						

Рисунок 5.12 – Сторінка запису пацієнта на прийом

Для роботи пацієнта з системою йому необхідно пройти авторизацію (за умови наявності облікового запису) чи реєстрацію (за відсутності облікового запису). Сторінка авторизації / реєстрації наведена на рис. 5.13.

Рисунок 5.13 – Сторінка авторизації / реєстрації пацієнта

Пацієнт може завантажити зображення захворювання та виконати діагностування (рис. 5.14), після чого йому буде запропоновано записатися на прийом до лікаря.

Рисунок 5.14 – Діагностування захворювання пацієнтом

Сторінка «Талон» дозволяє пацієнту переглянути доступні записи на прийом (рис. 5.15), створити новий запис з вибором дати, часу та лікаря (рис 5.16), а також видалити непотрібні записи (рис. 5.17).

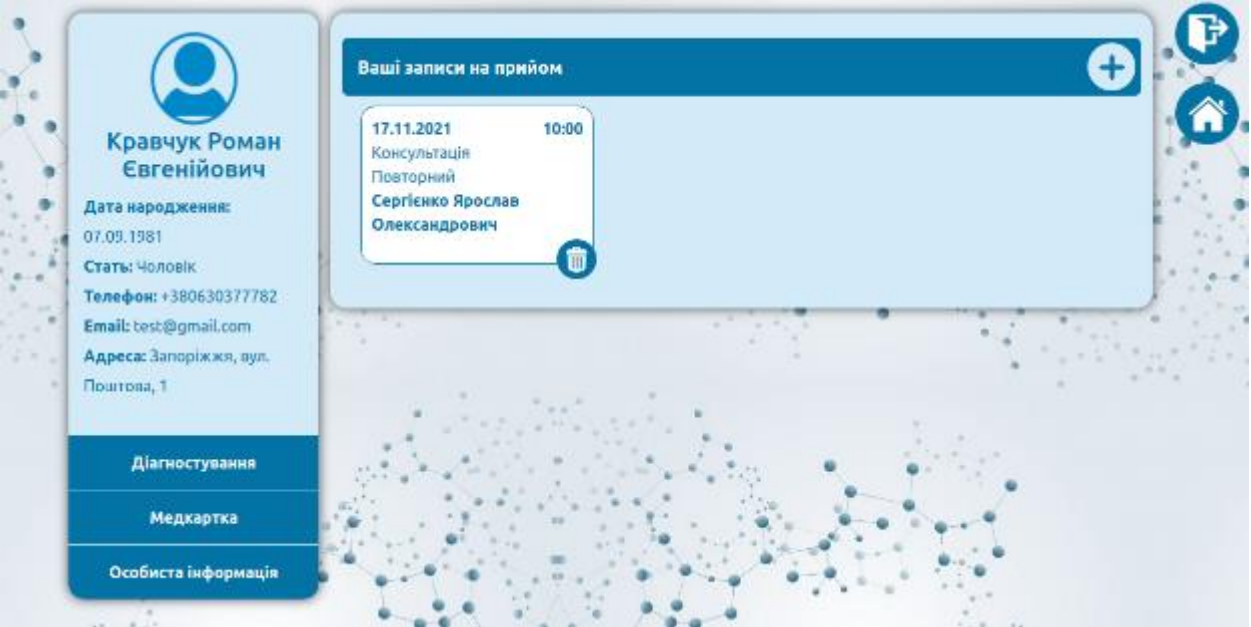


Рисунок 5.15 – Сторінка перегляду записів на прийом для пацієнта

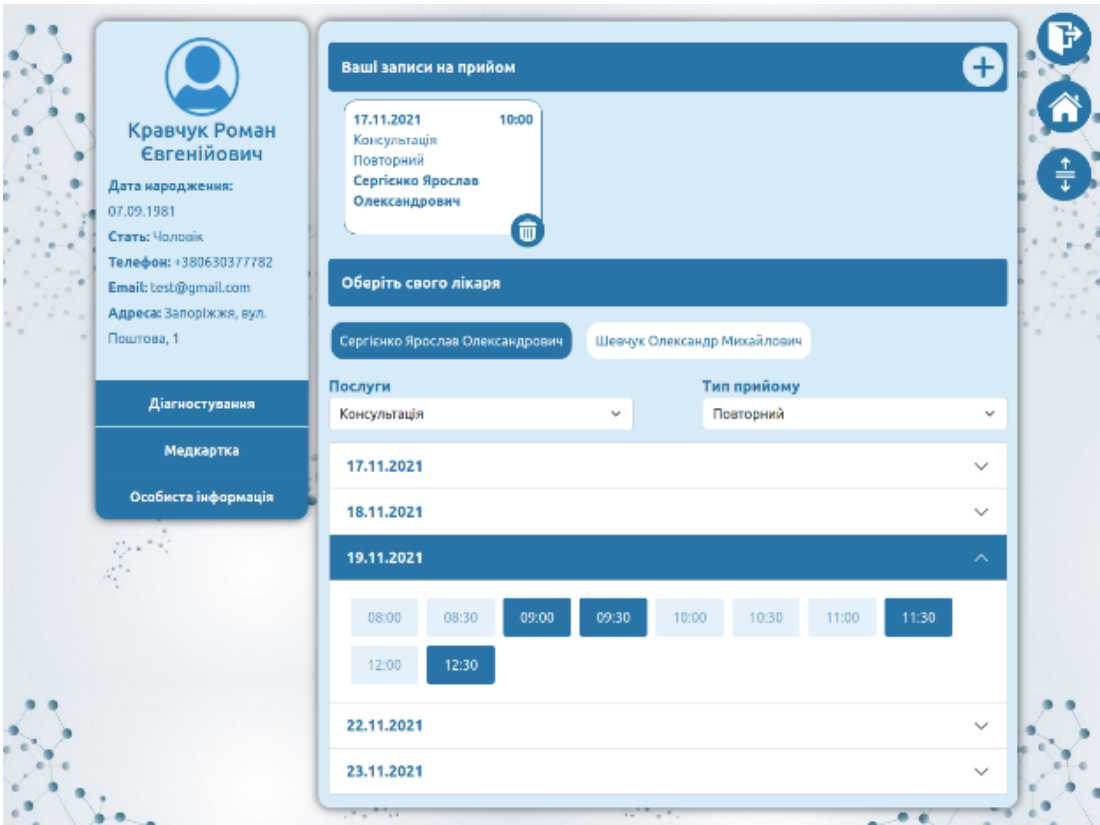


Рисунок 5.16 – Створення нового запису на прийом

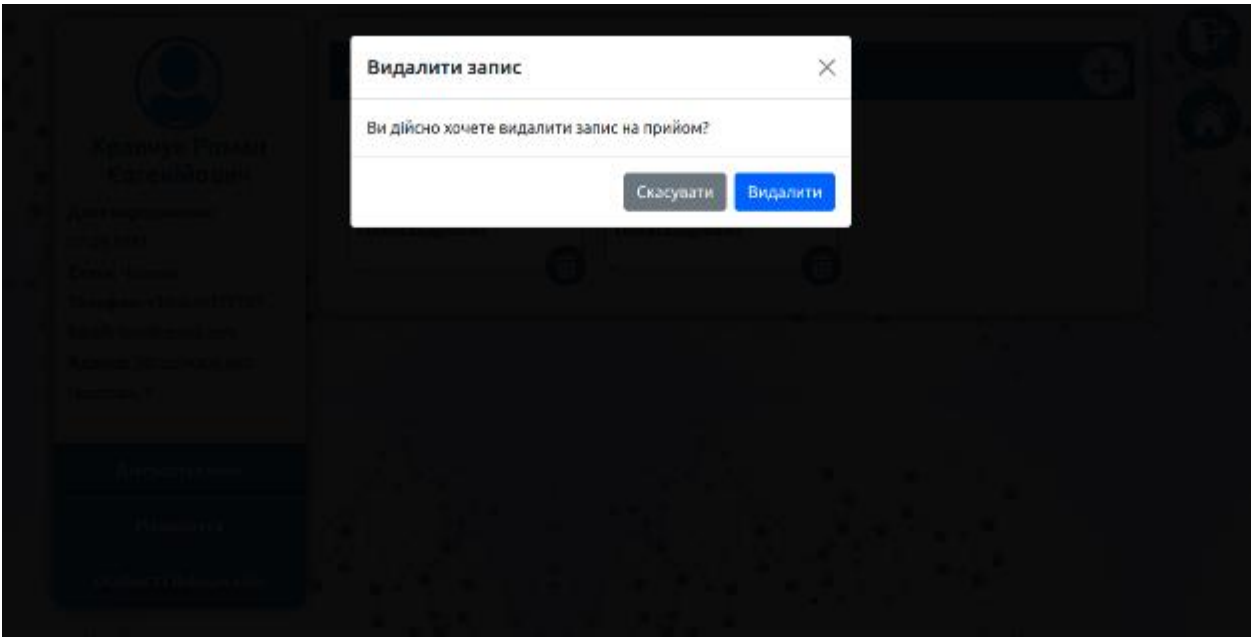


Рисунок 5.17 – Видалення запису на прийом

За потреби пацієнт може відредагувати особисті дані (рис. 5. 18).

Кравчук Роман Євгенійович
Дата народження: 07.09.1981
Стать: Чоловік
Телефон: +380630377782
Email: test@gmail.com
Адреса: Запоріжжя, вул. Поштова, 1

Діагностування
Медкарта
Талон

Прізвище
Кравчук

Ім'я
Роман

По батькові
Євгенійович

Стать
Чоловік

Електронна пошта
test@gmail.com

Пароль

Дата народження
09/07/1981

Телефон
+380630377782

Адреса проживання
Запоріжжя, вул. Поштова, 1

Зберегти

Рисунок 5.18 – Сторінка редагування особистих даних пацієнта

Також пацієнт може переглянути свою медичну карту і виконати пошук необхідних даних (рис. 5.19).



Рисунок 5.19 – Сторінка медичної карти пацієнта

5.4 Повідомлення та обробка виняткових ситуацій

Для обробки виняткових ситуацій у кодї програми використовується спеціальна конструкція `try/catch`, яка допомагає виявити помилки, обробити їх відповідним чином та продовжити виконання програми.

Усі помилки записуються у спеціальний файл `error.log`, який знаходиться у папці `var`. Це зручно для подальшої розробки та підтримки застосунку, адже дозволяє розробнику зекономити час та ресурси, які можуть бути витрачені на виконання відлагодження та пошук помилки.

Для користувачів системи також передбачено ряд повідомлень про успішне та помилкове виконання роботи. Для цього використовуються `flash` повідомлення, тобто повідомлення з'являється після виконання певної дії та зникає після перезавантаження сторінки чи переходу на іншу сторінку. Наприклад, повідомлення про успішне збереження запису про прийом (рис. 5.20), повідомлення про успішне редагування даних (рис. 5.21), повідомлення про спробу завантажити файл, тип якого не підтримується (рис. 5.22).

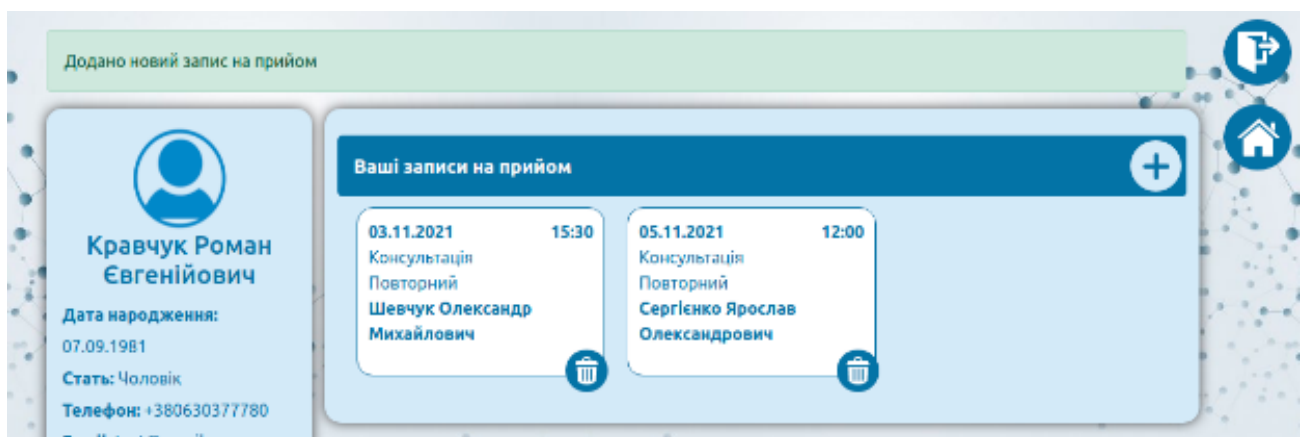


Рисунок 5.20 – Повідомлення про запис на прийом

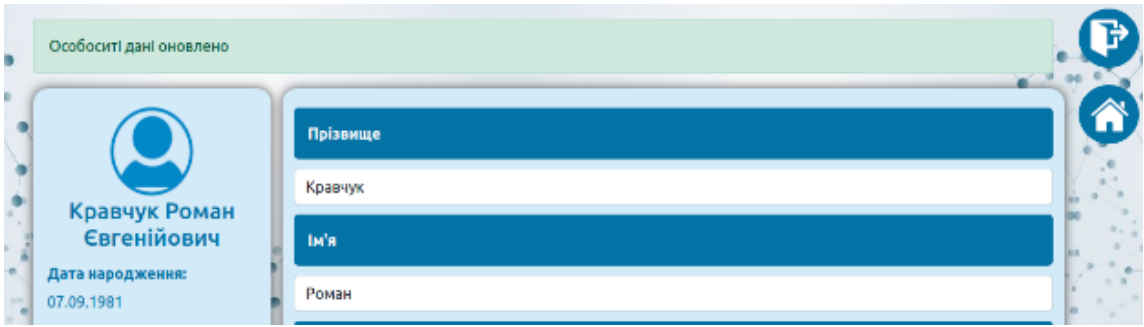


Рисунок 5.21 – Повідомлення про редагування особистих даних

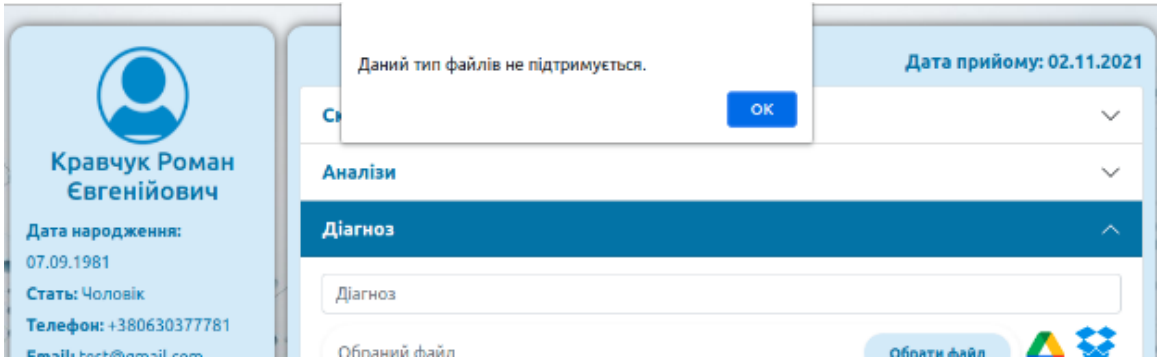


Рисунок 5.22 – Повідомлення про неправильний тип файлу

Також важливою частиною є валідація форм, які відправляють користувачі. Для цього було розроблено відповідні правила та повідомлення (рис. 5.23).

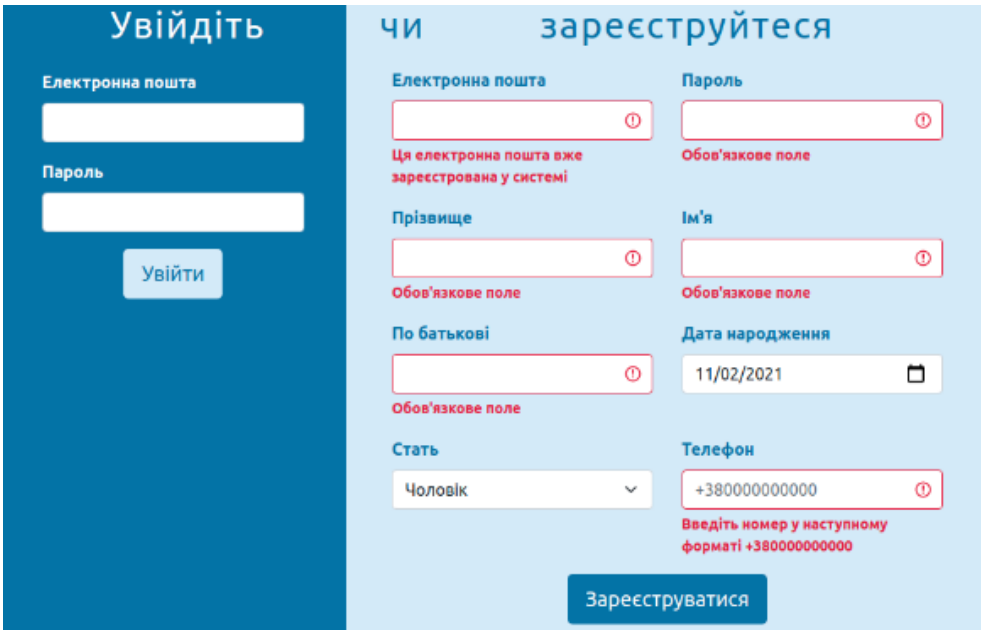


Рисунок 5.23 – Валідація форми реєстрації

5.5 Методика та результати тестування

5.5.1 Навчання та оцінка точності нейронної мережі

Для навчання та оцінки точності нейронної мережі було використано наступні налаштування:

- кількість параметрів для тренування – 9 194 503;
- кількість епох – 60;
- розмір міні-вибірки – 16;
- оптимізатор – Adam;
- нижня межа швидкості навчання – 0.00001;
- верхня межа швидкості навчання – 0.001;
- функція для зменшення швидкості навчання –

ReduceLROnPlateau;

- метрика точності – accuracy;
- функція втрат – categorical_crossentropy.

Результати навчання нейронної мережі наведено на рис. 5.24.

```
Epoch 1/60
1570/1569 [=====] - 104s 66ms/step - loss: 1.8020 - acc: 0.6424 - val_loss: 1.0240 - val_acc: 0.7512
Epoch 2/60
1570/1569 [=====] - 94s 60ms/step - loss: 0.9633 - acc: 0.7502 - val_loss: 0.8172 - val_acc: 0.8102
Epoch 3/60
1570/1569 [=====] - 95s 60ms/step - loss: 0.8218 - acc: 0.7996 - val_loss: 0.7881 - val_acc: 0.8243
Epoch 4/60
1570/1569 [=====] - 91s 58ms/step - loss: 0.7460 - acc: 0.8279 - val_loss: 0.6170 - val_acc: 0.8624
Epoch 5/60
1570/1569 [=====] - 94s 60ms/step - loss: 0.6595 - acc: 0.8535 - val_loss: 0.5827 - val_acc: 0.8861
.....
Epoch 55/60
1570/1569 [=====] - 87s 55ms/step - loss: 0.0581 - acc: 0.9950 - val_loss: 0.1858 - val_acc: 0.9745
Epoch 56/60
1570/1569 [=====] - 86s 55ms/step - loss: 0.0581 - acc: 0.9948 - val_loss: 0.1564 - val_acc: 0.9761
Epoch 57/60
1570/1569 [=====] - 88s 56ms/step - loss: 0.0569 - acc: 0.9949 - val_loss: 0.1593 - val_acc: 0.9755
Epoch 58/60
1570/1569 [=====] - 88s 56ms/step - loss: 0.0559 - acc: 0.9947 - val_loss: 0.1663 - val_acc: 0.9755
Epoch 59/60
1570/1569 [=====] - 87s 55ms/step - loss: 0.0563 - acc: 0.9949 - val_loss: 0.1630 - val_acc: 0.9756
Epoch 60/60
1570/1569 [=====] - 87s 55ms/step - loss: 0.0566 - acc: 0.9946 - val_loss: 0.1706 - val_acc: 0.9753
```

Рисунок 5.24 – Результати навчання нейронної мережі

Також було побудовано графіки метрики точності (рис. 5.25) та функції втрат (рис. 5.26) для тестової та валідаційної вибірок, отриманих в процесі проведення навчання. Дані графіки показують, як змінювалися

значення точності та помилки для використаних вибірок залежно від кількості проведених епох.

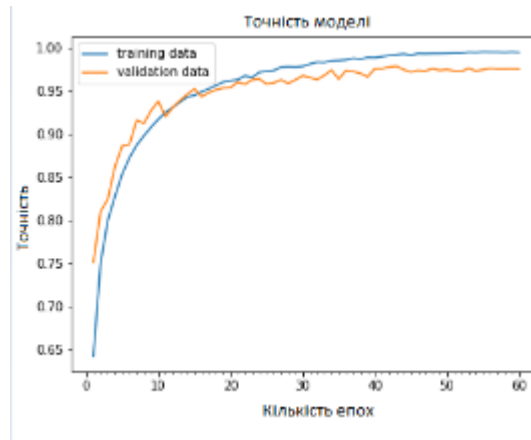


Рисунок 5.25 – Графік метрики точності для тестової та валідаційної вибірки

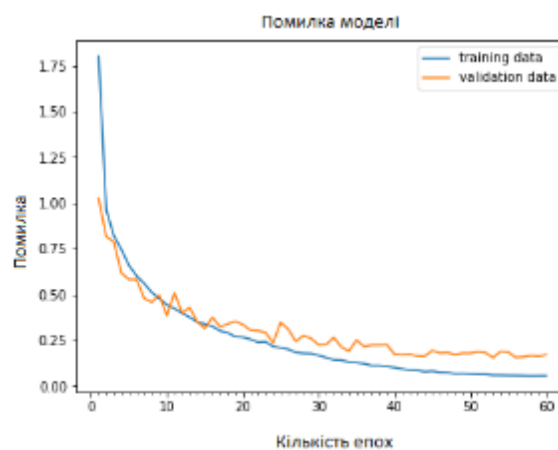


Рисунок 5.26 – Графік функції втрат для тестової та валідаційної вибірки

Для оцінки точності розпізнавання нейронної мережі було використано функцію `evaluate` з бібліотеки Keras та тестовий набір даних і отримано результат 96.31% (рис. 5.27).

```
2003/2003 [=====] - 2s 844us/step
Test: accuracy = 0.963055
```

Рисунок 5.27 – Оцінка точності розпізнавання нейронної мережі

За результатами тестування було визначено, що точність модифікованої моделі (96.31%) перевищує результати використовуваних раніше моделей для набору даних HAM10000. Порівняння існуючих результатів наведено у табл. 5.1.

Таблиця 5.1 – Порівняння точності розпізнавання різних моделей на наборі даних HAM10000

Модель	Джерело	Точність розпізнавання
VGG16	[58]	78.00%
ResNet	[58]	90.50%
AlexNet	[59]	88.81%
ResNet	[59]	85.20%
VGG-16	[59]	86.09%
DenseNet	[59]	85.25%
MobileNet	[59]	82.62%
ResNet34	[60]	78.84%
ResNet101	[60]	78.96%
SEResNet50	[60]	80.72%
VGG16	[60]	81.65%
VGG19	[60]	79.02%
EfficientNetB5	[60]	77.14%
MobileNet	[60]	84.73%
ResNeXt101	[61]	93.20%
Ensemble	[61]	92.83%
Розроблений метод з модифікованою моделлю ResNet50		96.31%

Також було побудовано матрицю розбіжностей, яка допомагає виявити кількість екземплярів зображень, які було правильно діагностовано, та кількість помилкових діагностувань за кожним діагнозом (рис. 5.28). Рядок – це істинні значення, а стовбець – значення передбачені нейронною мережею.

AKIES	[	66	0	0	0	0	0	0]
BCC	[	0	105	0	0	1	0	0]
BKL	[	0	0	196	2	1	0	0]
DF	[	0	0	0	22	0	0	0]
NV	[	0	4	22	23	1286	13	1]
MEL	[	0	0	0	1	5	222	0]
VASC	[	0	0	0	1	0	0	32]
		AKIES	BCC	BKL	DF	NV	MEL	VASC

Рисунок 5.28 – Матриця розбіжностей

На рис. 5.29 наведено діаграму, яка показує частку правильно класифікованих зображень для кожного діагнозу. Для даної діаграми значення розраховується, як відношення кількості правильно класифікованих зображень до загальної кількості зображень індивідуально для кожного з діагнозів.

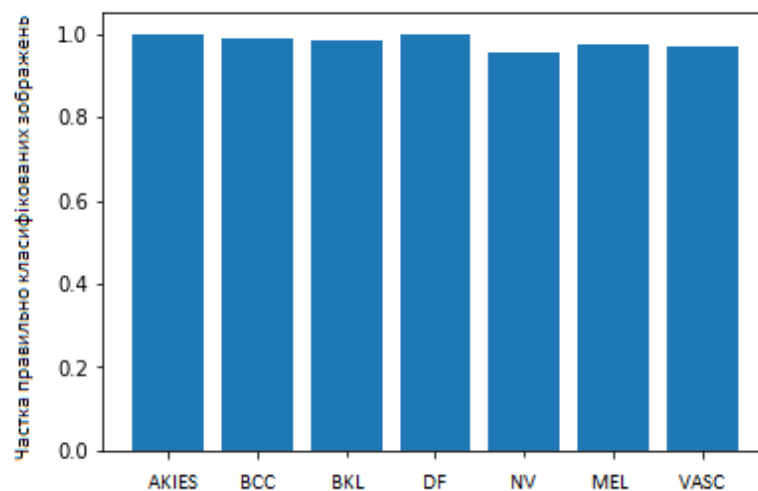


Рисунок 5.29 – Діаграма правильно класифікованих зображень

На рис. 5.30 наведено діаграму, яка показує частку неправильно класифікованих зображень для кожного діагнозу. Для даної діаграми значення розраховується, як відношення кількості неправильно класифікованих зображень до загальної кількості зображень індивідуально для кожного з діагнозів.

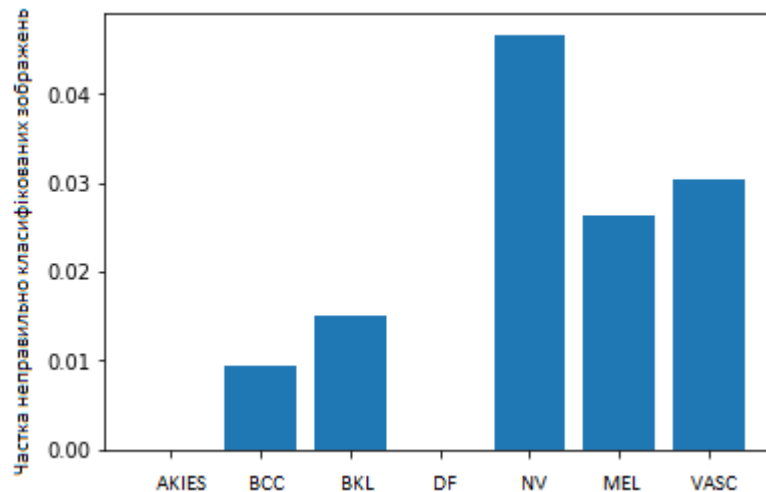


Рисунок 5.30 – Діаграма неправильно класифікованих зображень

5.5.2 Тестування вебзастосунку

Для перевірки якості роботи вебзастосунку було проведено такі види тестування:

- тестування верстки;
- функціональне тестування;
- тестування безпеки.

Оскільки програмне забезпечення може застосовуватися на різних видах пристроїв, потрібно було забезпечити його коректне відображення на різних розмірах екранів. Перевірку верстки було проведено для наступних пристроїв:

- Galaxy S5 (розширення екрану 360x640);
- Pixel 2 (411x731);

- Pixel 2 XL (411x823);
- iPhone 5/SE (320x568);
- iPhone 6/7/8 (375x667);
- iPhone 6/7/8 Plus (414x736);
- iPhone X (375x812);
- iPad (768x1024);
- iPad Pro (1024x1366);
- Surface Duo (540x720).

Тестування показало, що розроблене програмне забезпечення правильно відображається на усіх типах використовуваних пристроїв. Приклади тестування верстки наведено на рис. 5.31-5.33.

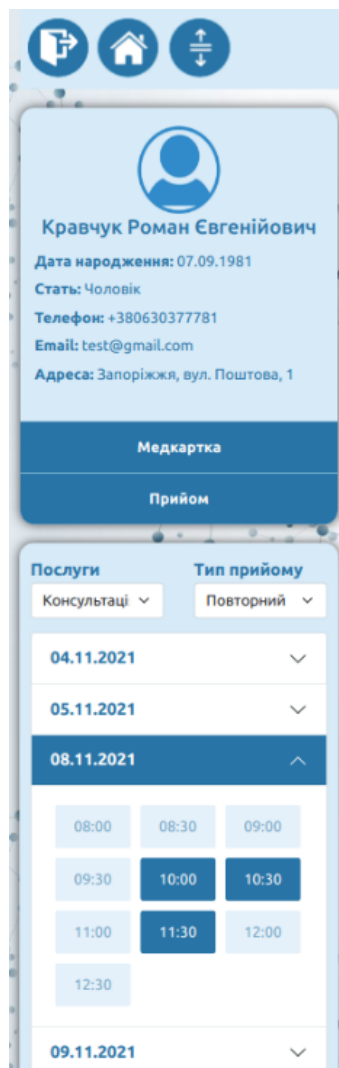


Рисунок 5.31 – Тестування на Galaxy S5 (360x640)

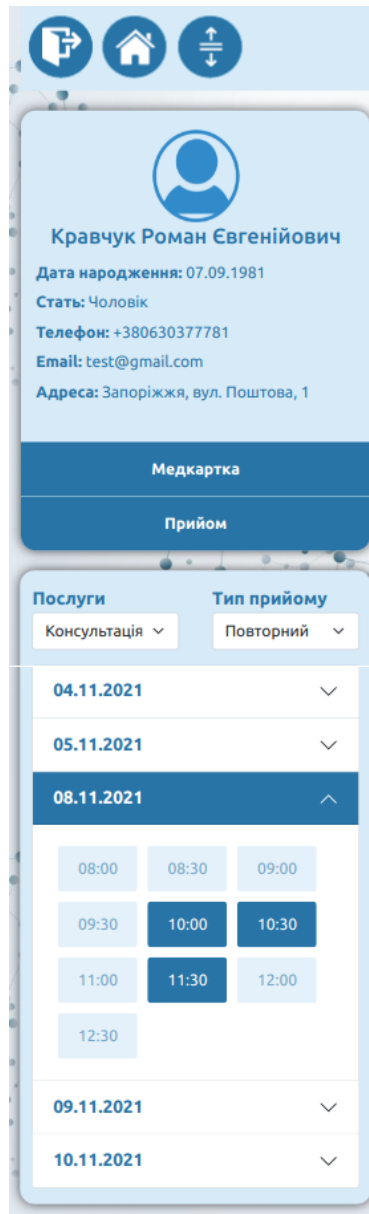


Рисунок 5.32 – Тестування на iPhone 6/7/8 (375x667)

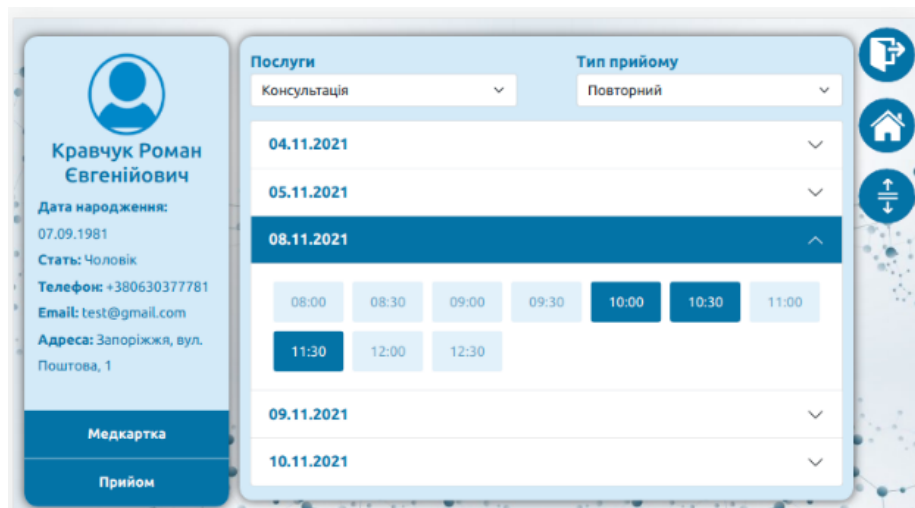


Рисунок 5.33 – Тестування на iPad Pro 1024x1366

Також було проведено функціональне тестування розробленого програмного забезпечення. Даний вид тестування передбачає перевірку реалізації функціональних вимог. Результати тестування показали, що всі вимоги визначені у підрозділі 3.2 було виконано в повному обсязі. Також на даному етапі було перевірено обробку помилкових дій користувача при роботі з системою, а саме:

- перевірка на порожні значення для форм;
- перевірка унікальності електронної пошти при реєстрації;
- перевірка існування електронної пошти і пароля при авторизації;
- перевірка дати народження на потрапляння у діапазон дат;
- перевірка номера телефона на наявність лише цифр;
- перевірка номера телефона на відповідність заданому формату;
- перевірка типу завантажуваного файлу зображення;
- перевірка розміру завантажуваного файлу зображення.

У разі виконання помилкових дій користувачем система повинна повідомити його про це відповідним повідомленням.

У разі, якщо користувач змінить поточну URL-адресу на неіснуючу його буде перенаправлено на 404 сторінку (рис. 5.34).

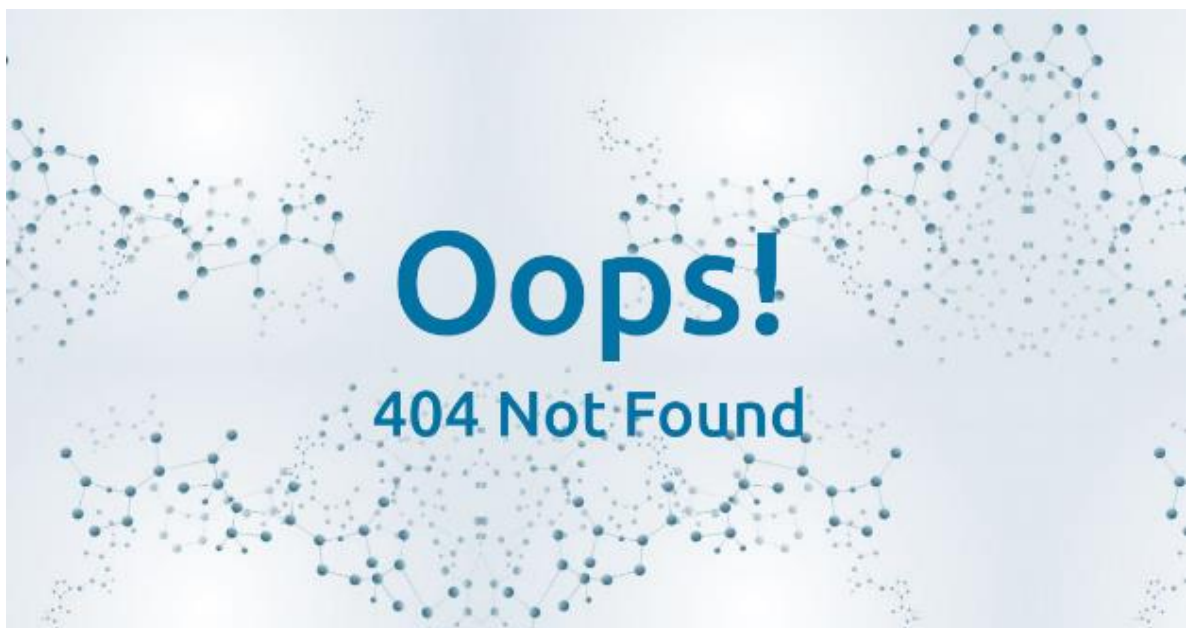


Рисунок 5.34 — Сторінка 404

Також було проведено тестування безпеки програмного забезпечення, яке передбачає перевірку розподіленого доступу до сторінок системи для різних груп користувачів.

Оскільки до роботи в системі допускаються лише користувачі, які пройшли аутентифікацію, було перевірено можливість отримати доступ до сторінки за допомогою введення URL-адреси (наприклад, /doctor/medical_card/8) без попереднього проходження авторизації. У даному випадку доступ було заблоковано, користувача перенаправлено на сторінку авторизації, а значить додаток правильно оброблює такі типи запитів і є безпечним для зберігання приватних даних (наприклад, таких як медичні діагнози).

Для безпечної роботи системи також було використано PHP функції, які допомагають знизити ризик виникнення помилок і проблем при роботі застосунку. Зокрема було застосовано наступні функції:

- `filter_input` – дозволяє отримати значення зовнішньої змінної за її назвою та відфільтрувати її за допомогою одного або кількох вбудованих фільтрів;
- `password_hash` – створює хеш пароля;
- `htmlspecialchars_decode` – перетворює спеціальні HTML-сутності у відповідні символи;
- `stripslashes` – видаляє екранування символів;
- `filter_var` – фільтрує змінну за допомогою відповідного фільтра;
- `password_verify` – дозволяє перевірити відповідність пароля хешу.

Ще одним важливим способом для підвищення безпеки системи є застосування бібліотеки PDO, яка надає можливість використання підготовлених запитів до БД.

5.6 Висновки до п'ятого розділу

У даному розділі було наведено опис застосування розробленого програмного забезпечення, визначено умови виконання та розглянуто основні сторінки системи з інструкцією для їх експлуатації.

Також було проведено наступні види тестування:

- тестування верстки;
- тестування безпеки;
- функціональне тестування.

Розроблене програмне забезпечення вдало пройшло всі види тестування, система коректно функціонує і може використовуватися за призначенням.

ВИСНОВКИ

У процесі виконання дипломної роботи було розроблено метод діагностування шкірних захворювань та створено програмне забезпечення, яке використовує даний метод. Для цього виконано наступні завдання:

- проведено аналіз предметної області та розроблено технічне завдання;
- розроблено метод діагностування захворювань шкіри;
- спроектовано програмне забезпечення;
- розроблено програмне забезпечення;
- проведено тестування програмного забезпечення.

На етапі проведення аналізу предметної області було визначено основні її особливості, виокремлено та описано поширені хвороби шкіри і метод їх діагностування, також наведено порівняння програм-аналогів.

Оскільки було проаналізовано методи розпізнавання медичних зображень, зокрема отриманих за допомогою дерматоскопії, і визначено, що одним з найбільш ефективних є використання згорткових нейронних мереж, саме цей метод і було обрано для подальшого дослідження.

Також на даному етапі було розроблено технічне завдання, визначено підстави для розробки, призначення, основні вимоги до програми, стадії та етапи, порядок контролю та приймання.

На етапі розробки методу діагностування шкірних захворювань було визначено особливості використання нейронних мереж для розпізнавання медичних зображень, обрано набір даних HAM10000, вирішено модифікувати попередньо навчену модель ResNet50, а також описано техніки, які мають використовуватися на різних етапах розробки методу.

На наступному етапі було обрано такі засоби проектування:

- для нейронної мережі: мову програмування Python, бібліотеки TensorFlow, Keras, NumPy, Pandas, Matplotlib, Scikit Learn, os;

- для розробки вебзастосунку: мови програмування PHP та JS, мову розмітки HTML, мову стилів CSS, інструмент Bootstrap, менеджер пакетів Composer;
- для роботи з БД: СКБД MySQL, інструмент phpMyAdmin, бібліотеку PDO;
- у якості середовища розробки – PhpStorm.

Також на даному етапі було визначено функціональні вимоги для різних груп користувачів (лікарі та пацієнти) і побудовано відповідні діаграми прецедентів, спроектовано та описано структуру БД. Для розробки обрано архітектурний шаблон MVC, на основі якого побудовано структурну схему системи.

На етапі розробки програмного забезпечення визначено алгоритм функціонування програми, а також наведено детальний опис основних папок, класів та методів, які було реалізовано в процесі розробки.

На етапі тестування виконано перевірку роботи програмного забезпечення, наведено опис застосування та умови виконання, розроблено інструкцію з експлуатації, побудовано та описано графіки навчання та оцінки точності модифікованої моделі нейронної мережі, а також проведено різні види тестування створеного вебзастосунку. Система вдало пройшла всі види тестування, вона коректно функціонує і може використовуватися за призначенням.

Наукова новизна роботи полягає в запропонованому методі діагностування шкірних захворювань, який характеризується використанням модифікованої моделі ResNet50, що дозволяє підвищити точність винесення діагнозу. Після проведення навчання та тестування точність розпізнавання склала 96.31%.

Отже, усі поставлені завдання було виконано та реалізовано програмне забезпечення, яке відповідає усім визначеним вимогам.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Рак шкіри (пухлини шкіри) [Електрон. ресурс] – Режим доступу: <https://spizhenko.clinic/uk/zabolevaniya/opuholi-kozhi-uk>.
2. Рак кожи [Электрон. ресурс] – Режим доступа: <https://oncology-spb.ru/zabolevaniya/kozha-i-myagkie-tkani/rak-kozhi>.
3. Рак в Украине 2020. Информация к размышлению – анализ данных канцер-реестра [Электрон. ресурс] – Режим доступа: <https://chemoteka.com.ua/blog/article/rak-v-ukraine-2020-informaciya-k-razmyshleniyu-analiz-dannyh-kancer-reestra-197>.
4. Skin [Electronic resource] – Access mode: <https://www.nationalgeographic.com/science/article/skin-1>.
5. Skin disease [Electronic resource] – Access mode: <https://www.britannica.com/science/human-skin-disease>.
6. Актинічний кератоз: новий підхід до лікування [Електрон. ресурс] – Режим доступу: <https://www.umj.com.ua/article/199151/aktinichnij-keratoz-novij-pidhid-do-likuvannya>.
7. Базаліома (базальноклітинний рак шкіри) [Електрон. ресурс] – Режим доступу: <https://spizhenko.clinic/uk/zabolevaniya/opuholi-kozhi-uk/bazalioma-bazalnokletochnyj-rak-kozhi>.
8. Seborrheic Keratosis [Electronic resource] – Access mode: <https://www.webmd.com/skin-problems-and-treatments/picture-of-seborrheic-keratosis>.
9. Остафійчук, В. В. Клінічний випадок трансформації дерматофіброми у фібросаркому / В. В. Остафійчук, С. І. Коровін, М. С. Кротеви́ч // Клінічна онкологія . – К: МОРІОН, 2019. – С. 110–114.
10. Що треба знати про меланому [Електрон. ресурс] – Режим доступу: <https://moz.gov.ua/article/health/scho-treba-znati-pro-melanomu>.

11. Меланоцитарні невуси та ознаки їх злоякісної трансформації [Електрон. ресурс] – Режим доступу: <https://www.bsmu.edu.ua/blog/4073-melanotsitarni-nevusi-ta-oznaki-ih-zloyakisnoi-transformatsii/>.
12. Судинні ураження шкіри [Електрон. ресурс] – Режим доступу: <https://mama-tato.com.ua/article/a-1381.html>.
13. Диагностика, назначение лечения [Электрон. ресурс] – Режим доступа: <https://medcity.ua/services/diagnostika-postanovka-diagnoza-i-naznachenie-lecheniya200/>.
14. Дерматоскопія [Електрон. ресурс] – Режим доступу: <https://alpha-medical.com.ua/uk/analizi/dermatoskopiya-2/>.
15. Сервис «Проверь родинку онлайн» [Электрон. ресурс] – Режим доступа: <https://smarteka.com/practices/servis-prover-rodinku-onlajn>.
16. Добробут [Електрон. ресурс] – Режим доступу: <https://www.dobrobut.com/cancer/test/?lang=ua>.
17. Как определять рак кожи с помощью смартфона? Лучшие приложения и бот для Telegram [Электрон. ресурс] – Режим доступа: <https://yablyk.com/432998-how-to-detect-skin-cancer-on-iphone-android-smartph>.
18. Yadav, S. S. Deep convolutional neural network based medical image classification for disease diagnosis [Text] / S. S. Yadav, S. M. Jadhav // Journal of Big Data. – 2019. – №6. – P. 113–131.
19. Deep Neural Network [Electronic resource] – Access mode: <https://www.techopedia.com/definition/32902/deep-neural-network>.
20. Deep Neural Network: An Easy Introduction for 2021 [Electronic resource] – Access mode: <https://www.jigsawacademy.com/blogs/ai-ml/deep-neural-network/>.
21. Convolutional Neural Network for Skin Lesion Classification: Understanding the Fundamentals Through Hands-On Learning [Electronic resource] – Access mode: <https://www.frontiersin.org/articles/10.3389/fmed.2021-644327>.

22. Haggemüller, S. Skin cancer classification via convolutional neural networks: systematic review of studies involving human experts [Text] / S. Haggemüller, R. Maron, A. Hekler. // European Journal of Cancer. – 2021. – №156. – P. 202–216.

23. Convolutional neural networks in dermatology [Electronic resource] – Access mode: <https://dermnetnz.org/topics/convolutional-neural-networks-in-dermatology>.

24. Basic CNN Architecture: Explaining 5 Layers of Convolutional Neural Network [Electronic resource] – Access mode: <https://www.upgrad.com/blog/basic-cnn-architecture/>.

25. Keras Implementation of ResNet-50 (Residual Networks) Architecture from Scratch [Electronic resource] – Access mode: <https://machinelearningknowledge.ai/keras-implementation-of-resnet-50-architecture-from-scratch/>.

26. Introduction to Resnet or Residual Network [Electronic resource] – Access mode: <https://www.mygreatlearning.com/blog/resnet/>.

27. Как работает сверточная нейронная сеть: архитектура, примеры, особенности [Электрон. ресурс] – Режим доступа: <https://neurohive.io/ru/osnovy-data-science/glubokaya-svertochnaja-nejronnaja-se>.

28. Everything You Should Know About Dropouts And BatchNormalization In CNN [Electronic resource] – Access mode: <https://analyticsindiamag.com/everything-you-should-know-about-dropouts-and-batchnormalization-in-cnn/>.

29. Функции активации нейросети: сигмоида, линейная, ступенчатая, ReLu, tahn [Электрон. ресурс] – Режим доступа: <https://neurohive.io/ru/osnovy-data-science/activation-functions/>.

30. CNN | Introduction to Pooling Layer [Electronic resource] – Access mode: <https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/>.

31. The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions [Electronic resource] –

Access mode: <https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/DBW86T>.

32. You Might Be Resizing Your Images Incorrectly [Electronic resource] – Access mode: <https://blog.roboflow.com/you-might-be-resizing-your-images-incorrectly/>.

33. What Is Undersampling? [Electronic resource] – Access mode: <https://www.mastersindatascience.org/learning/statistics-data-science/undersampl>.

34. About Train, Validation and Test Sets in Machine Learning [Electronic resource] – Access mode: <https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7>.

35. Srivastava, N. Dropout: A Simple Way to Prevent Neural Networks from Overfitting [Text] / N. Srivastava, G. Hinton, A. Krizhevsky. // Journal of Machine Learning Research. – 2014. – №15. – С. 1929–1958.

36. What Is Transfer Learning? Exploring the Popular Deep Learning Approach. [Electronic resource] – Access mode: <https://builtin.com/data-science/transfer-learning>.

37. Fine-tuning Convolutional Neural Network on own data using Keras Tensorflow [Electronic resource] – Access mode: <https://cv-tricks.com/keras/fine-tuning-tensorflow/>.

38. Why Python is Good for Machine Learning [Electronic resource] – Access mode: <https://www.section.io/engineering-education/why-python-is-good-for-machine-learning/>.

39. TensorFlow [Электрон. ресурс] – Режим доступа: http://cloud-5.bitp.kiev.ua/?page_id=598.

40. Keras [Электрон. ресурс] – Режим доступа: <https://artsandculture.google.com/entity/g11c1r2rvnp?hl=uk>.

41. Numpy, Pandas, matplotlib – необходимый минимум для старта в Machine Learning [Электрон. ресурс] – Режим доступа: <https://proglib.io/p/numpy-pandas-matplotlib-neobhodimyy-minimum-dlya-starta-v-machine-learning-2021-06-18>.

42. Что такое Scikit Learn - гайд по популярной библиотеке Python для начинающих [Электрон. ресурс] – Режим доступа: <https://datastart.ru/blog/read/chto-takoe-scikit-learn-gayd-po-populyarnoy-python-dlya-nachinayuschih>.

43. Библиотека os [Электрон. ресурс] – Режим доступа: <https://all-python.ru/osnovy/os.html>.

44. Is PHP Still Relevant to Build Websites in 2021: Benefits and Real Examples [Electronic resource] – Access mode: <https://gbksoft.com/blog/build-websites-with-php/>.

45. 8 Major Advantages of Using MySQL [Electronic resource] – Access mode: <https://www.datamation.com/storage/8-major-advantages-of-using-mysql/>.

46. phpMyAdmin [Electronic resource] – Access mode: <https://www.javatpoint.com/phpmyadmin>.

47. PHP PDO – работаем с базами данных правильно [Электрон. ресурс] – Режим доступа: <https://p0vidl0.info/php-pdo-rabotaem-s-bazami-dannyx-pravilno.html>.

48. Основні поняття мови HTML та структура документів [Електрон. ресурс] – Режим доступу: <https://sites.google.com/site/vivcaemoweb-dizajndistancijno/html/lekcja-3-osnovni-ponatta-movi-html-ta-struktura-dokument>.

49. Що таке CSS [Електрон. ресурс] – Режим доступу: https://css.in.ua/article/shcho-take-html_10.

50. JavaScript [Електрон. ресурс] – Режим доступу: <https://astwellsoft.com/uk/blog/tehnology/javascript.html>.

51. Что такое AJAX? Создание асинхронных запросов [Электрон. ресурс] – Режим доступа: <https://itchief.ru/javascript/ajax-introduction>.

52. Фреймворк Bootstrap – лідер, про який не можна не сказати [Електрон. ресурс] – Режим доступу: <http://programer.org.ua/frejmwork-bootstrap-lider-pro-yakij-ne-mozhna-ne-skazati/>.

53. Composer [Електрон. ресурс] – Режим доступу: <https://uk.freejournal.org/2265640/1/composer.html>.

54. vlucas/phpdotenv [Electronic resource] – Access mode: <https://github.com/vlucas/phpdotenv>.
55. LAMP Stack [Electronic resource] – Access mode: <https://www.ibm.com/cloud/learn/lamp-stack-explained>.
56. What is PhpStorm? [Electronic resource] – Access mode: <https://monovm.com/blog/what-is-phpstorm/>.
57. Введение в PHP MVC Framework [Электрон. ресурс] – Режим доступа: <https://upread.ru/art.php?id=738>.
58. Garg, R. Decision Support System for Detection and Classification of Skin Cancer Using CNN [Text] / R. Garg, S. Maheshwari, A. Shukla. // Innovations in Computational Intelligence and Computer Vision. – 2021. – №1. – P. 578–586.
59. Ali, S. An enhanced technique of skin cancer classification using deep convolutional neural network with transfer learning models [Text] / S. Ali, S. Miah, J. Haque. // Machine Learning with Applications. – 2021. – №5. – P. 1–8.
60. Lucius, M. Deep neural frameworks improve the accuracy of general practitioners in the classification of pigmented skin lesions [Text] / M. Lucius, M. Belvisi, L. Radizza. // Diagnostics. – 2020. – №10. – P. 1–15.
61. Chaturvedi, S. A multi-class skin Cancer classification using deep convolutional neural networks [Text] / S. Chaturvedi, J. Tembhurne, T. Diwan. // Computer Science. – 2020. – №79. – P. 1–22.

ДОДАТОК А
Технічне завдання

Вступ

Метою магістерської роботи є розробка програмного забезпечення діагностування шкірних захворювань, що дозволить знизити ризик встановлення помилкового діагнозу та почати лікування на початковій стадії.

A.1 Підстави для розробки

Підставою для розробки є завдання на магістерську роботу на тему «Дослідження та програмна реалізація методів діагностування шкірних захворювань», затверджене наказом № 435 від 5 листопада 2021 року за Національним університетом «Запорізька політехніка».

A.2 Призначення розробки

Програмне забезпечення призначене для використання у медичній сфері, зокрема для діагностування шкірних захворювань, а також дозволяє виконувати додаткові функції, пов'язані з веденням медичної карти пацієнта і оформлення записів на прийом.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно реалізовувати наступні вимоги:

а) вимоги лікаря:

- 1) авторизація за допомогою електронної пошти та пароля;
- 2) перегляд списку пацієнтів на поточний день;
- 3) перегляд інформації про конкретного пацієнта;
- 4) додавання записів про прийом (скарги хворого, аналізи, діагноз, призначення);

5) завантаження зображення захворювання з комп'ютера та / або хмарних сховищ, таких як Google Drive та Dropbox;

6) діагностування захворювання за завантаженим зображенням;

7) перегляд медичної карти хворого;

8) пошук даних у медкартці;

9) запис пацієнта на наступний прийом з можливістю вибору дати та часу;

10) перегляд та редагування особистої інформації лікаря;

11) налаштування хмарних сховищ (Google Drive та Dropbox);

б) вимоги пацієнта:

1) реєстрація та авторизація пацієнтів;

2) діагностування захворювання за завантаженим зображенням;

3) перегляд інформації про наявні записи на прийом до лікаря;

4) створення нових записів на прийом з можливістю вибору лікаря, дати та часу;

5) видалення непотрібних записів на прийом;

6) перегляд медичної карти;

7) пошук даних у медичній карті;

8) перегляд та редагування особистої інформації;

в) загальні вимоги до системи:

1) використання розробленого методу для здійснення діагностування шкірного захворювання за завантаженим зображенням;

2) обробка помилкових дій користувача з подальшим інформування шляхом виводу відповідних повідомлень;

3) обробка виняткових ситуацій, таких як помилки при виконанні запитів до БД;

- 4) розподіл доступу до сторінок системи за ролями користувачів;
- 5) взаємодія із системою на різних пристроях (мобільні телефони, планшети, комп'ютери);
- 6) зручний та інтуїтивно зрозумілий інтерфейс.

A.3.2 Вимоги до надійності

Програмне забезпечення повинно відповідати таким вимогам безпеки:

- підтримка функцій захисту від несанкціонованого доступу (доступ до різних сторінок для лікарів та пацієнтів);
- обробка помилкових дій користувача з відповідними повідомлення (валідація форм користувачів, перехід на неіснуючі сторінки);
- обробка виняткових ситуацій (наприклад, помилки при роботі з БД).

A.3.3 Умови експлуатації

Для розміщення розробленої програми у мережі Інтернет необхідно виділити вебвузол і встановити на ньому:

- вебсервер Apache;
- інтерпретатор PHP 7.4;
- менеджер пакетів Composer;
- БД MySQL.

A.3.4 Вимоги до складу та параметрів технічних засобів

Для використання розробленої системи необхідні:

- процесор від 1.0 ГГц;
- обсяг оперативної пам'яті від 4 ГБ;

- веббраузер (Google Chrome, Mozilla Firefox, Opera та ін.);
- наявність підключення до мережі Інтернет.

А.3.5 Вимоги до транспортування та збереження

Вимоги до транспортування і збереження програмного забезпечення залежать від вимог до відповідного носія даних.

Транспортування програми може здійснюватися через мережу Інтернет. Відповідальність за збереження після транспортування несе компанія, яка надає послуги хостінга.

Якщо програмний продукт зберігатиметься на флеш-накопичувачі, то необхідно забезпечити відсутність впливу потужних магнітних полів, а також забезпечити допустимі кліматичні умови при транспортуванні та зберіганні накопичувача.

А.3.6 Вимоги до програмної документації

Програма повинна поставлятися з такими документами, як «Текст програми» та «Технічне завдання».

А.3.7 Вимоги до маркування

Розроблене програмне забезпечення може зберігатися на жорсткому диску або поширюватися і зберігатися на компакт-дисках.

На упаковці має бути вказано назву програми – «Дослідження та програмна реалізація методів діагностування шкірних захворювань».

Також програма може надаватися у вигляді програмного виробу на USB-флеш-накопичувачі або через мережу Інтернет.

A.4 Стадії та етапи розробки

Виконання дипломної роботи повинно складатися з наступних етапів:

- аналіз предметної області та розробка технічного завдання;
- розробка методу діагностування захворювань шкіри;
- проєктування програмного забезпечення;
- розробка програмного забезпечення;
- проведення тестування програмного забезпечення.

Робота має бути виконана у терміни встановлені згідно з календарним планом.

A.5 Порядок контролю та приймання

Дипломна робота допускається до захисту керівником і нормоконтролером, затверджується завідувачем кафедрою і захищається в присутності членів екзаменаційної комісії.

ДОДАТОК Б
Текст програми

```
//pub/index.php
<?php

use app\core\Router;

include_once dirname(__DIR__) . '/vendor/autoload.php';

define('ROOT', dirname(__FILE__));

$dotenv = \Dotenv\Dotenv::createImmutable(dirname(__DIR__));
$dotenv->load();

$router = new Router();
echo $router->run();
```

```
//core/Router.php
<?php
```

```
namespace app\core;
```

```
class Router
```

```
{
```

```
    /**
```

```
     * @var array
```

```
     */
```

```
    protected $routes;
```

```
    /**
```

```
     * @var Controller
```

```
     */
```

```
    protected $controller;
```

```
    /**
```

```
     * Router construct
```

```
     */
```

```
    public function __construct()
```

```
    {
```

```
        $this->routes = Routes::getRoutes();
```

```
        $this->controller = new Controller();
```

```
    }
```

```
    /**
```

```
     * @return string
```

```
     */
```

```
    public function run(): string
```

```
    {
```

```
        $url = $this->getURL();
```

```
        foreach ($this->routes as $urlPattern => $callback) {
```

```
            if (preg_match("~^$urlPattern$~", $url)) {
```

```
                $parameters = array_slice(explode('/', $url),
```

```
2);
```

```
                $controllerObject = new $callback[0];
```

```
                $actionName = $callback[1];
```

```

        return
    call_user_func_array(array($controllerObject, $actionName),
    $parameters);
    }
}

    return $this->controller->render404();
}

/**
 * @return string
 */
protected function getUrl(): string
{
    if(!empty($_SERVER['REQUEST_URI'])) {
        return trim($_SERVER['REQUEST_URI'], '/');
    }
    return '';
}
}

//core/Controller.php
<?php

namespace app\core;

use app\core\Session;
use app\core\View;

class Controller
{
    /**
     * @param $view
     * @param string $layout
     * @param array $params
     * @return string
     */
    public function renderView($view, $params = [], $layout =
''): string
    {
        $viewObject = new View();
        if ($layout) {
            $viewObject->setLayout($layout);
        }
        return $viewObject->renderView($view, $params);
    }

    /**
     * @param $url
     */
    public function redirect ($url): void
    {
        header('Location: ' . $url);
    }

    /**
     * @return string

```

```

    */
    public function render404(): string
    {
        $viewObject = new View();
        http_response_code(404);
        return $viewObject->renderOnlyView('_404');
    }

    /**
     * @return array
     */
    public function getParams(): array
    {
        $params = [];

        if ($this->isGet()) {
            foreach ($_GET as $key => $value) {
                $params[$key] = filter_input(INPUT_GET, $key,
FILTER_SANITIZE_FULL_SPECIAL_CHARS);
            }
        }

        if ($this->isPost()) {
            foreach ($_POST as $key => $value) {
                $params[$key] = filter_input(INPUT_POST, $key,
FILTER_SANITIZE_FULL_SPECIAL_CHARS);
            }
        }

        return $params;
    }

    /**
     * @return bool
     */
    public function isGet(): bool
    {
        return $this->method() === 'get';
    }

    /**
     * @return bool
     */
    public function isPost(): bool
    {
        return $this->method() === 'post';
    }

    /**
     * @return string
     */
    public function method(): string
    {
        return strtolower($_SERVER['REQUEST_METHOD']);
    }

    /**
     * @param $role

```

```

        * @return int
        */
public function checkLogin($role): int
{
    $userId = Session::getSession()->get($role);
    if ($userId) {
        return $userId;
    }

    if ($role == 'doctor') {
        $this->redirect('/doctor/login');
    } else {
        $this->redirect('/login');
    }

    return 0;
}
}

//core/Model.php
<?php

namespace app\core;

class Model
{
    public const RULE_REQUIRED = 'required';
    public const RULE_EMAIL = 'email';
    public const RULE_EMAIL_NOT_FOUND = 'email_not_found';
    public const RULE_PASSWORD_INVALID = 'password_invalid';
    public const RULE_PHONE = 'phone';
    public const RULE_DATE = 'dateOfBirthday';
    public const RULE_UNIQUE = 'unique';

    /**
     * @var array
     */
    public $errors = [];

    /**
     * @param $data
     */
    public function loadData($data): void
    {
        foreach ($data as $key => $value) {
            if (property_exists($this, $key)) {
                $this->{'set'.ucfirst($key)}($value);
            }
        }
    }

    /**
     * @param $model
     * @return bool
     */
    public function validate(&$model): bool

```

```

        {
            foreach ($this->rules() as $attribute => $rules) {
                $value = $model->{'get'.ucfirst($attribute)}();
                foreach ($rules as $rule) {
                    $ruleName = $rule;
                    if (!is_string($rule)) {
                        $ruleName = $rule[0];
                    }

                    if ($ruleName === self::RULE_REQUIRED &&
!$value) {
                        $model->addError($attribute,
self::RULE_REQUIRED);
                    }

                    if ($ruleName === self::RULE_EMAIL &&
!filter_var($value, FILTER_VALIDATE_EMAIL)) {
                        $model->addError($attribute,
self::RULE_EMAIL);
                    }

                    if ($ruleName === self::RULE_PHONE &&
!preg_match('/^\+[0-9]{12}$/ ', $value)) {
                        $model->addError($attribute,
self::RULE_PHONE);
                    }

                    if ($ruleName === self::RULE_DATE &&
!preg_match('/^[0-9]{4}-[0-9]{2}-[0-9]{2}$/ ', $value)) {
                        $model->addError($attribute,
self::RULE_DATE);
                    }

                    if ($ruleName == self::RULE_UNIQUE) {
                        $uniqueAttr = $rule['attribute'] ??
$attribute;

                        $tableName = $model->getTableName();
                        $statement = DB::getConnection()-
>prepare("SELECT * FROM $tableName WHERE $uniqueAttr = :attr");
                        $statement->bindValue(':attr', $value);
                        $statement->execute();
                        $record = $statement->fetchObject();
                        if ($record) {
                            $model->addError($attribute,
self::RULE_UNIQUE);
                        }
                    }
                }
            }

            return empty($model->errors);
        }

/**
 * @param $attribute
 * @param $rule
 */
public function addError($attribute, $rule): void
{

```

```

        $message = $this->errorMessage()[$rule] ?? '';
        $this->errors[$attribute][] = $message;
    }

    /**
     * @return array
     */
    public function errorMessage(): array
    {
        return [
            self::RULE_REQUIRED => 'Обов\'язкове поле',
            self::RULE_EMAIL    => 'Введіть коректну адресу
електронної пошти',
            self::RULE_PHONE    => 'Введіть номер у наступному
форматі +380000000000',
            self::RULE_DATE     => 'Введіть дату у наступному
форматі мм/дд/рррр',
            self::RULE_UNIQUE   => 'Ця електронна пошта вже
zareestrovana u systemi',
            self::RULE_EMAIL_NOT_FOUND => 'Вказана електронна
пошта не zareestrovana u systemi',
            self::RULE_PASSWORD_INVALID => 'Неправильний пароль'
        ];
    }

    /**
     * @param $object
     * @return bool
     */
    public function login($object): bool
    {
        $session = Session::getSession();
        if ($session) {
            $session->set($object->getRole(), $object->id);
            return true;
        }

        return false;
    }

    /**
     * @param $role
     */
    public function logout($role): void
    {
        Session::getSession()->remove($role);
    }
}

```

```

//core/View.php
<?php

```

```

namespace app\core;

use app\core\Session;

class View

```

```

{
    /**
     * @var string
     */
    private $layout = 'main';

    /**
     * @param $layout
     */
    public function setLayout($layout): void
    {
        $this->layout = $layout;
    }

    /**
     * @param $view
     * @param array $params
     * @return string
     */
    public function renderView($view, $params = []): string
    {
        $layoutContent = $this->layoutContent();
        $viewContent = $this->renderOnlyView($view, $params);
        return str_replace('{{content}}', $viewContent,
$layoutContent);
    }

    /**
     * @return string
     */
    protected function layoutContent(): string
    {
        $flashMessages = Session::getSession()-
>getFlash('success');
        $layout = $this->layout;
        ob_start();
        include_once dirname(__DIR__)
"/views/layouts/$layout.phtml";
        return ob_get_clean();
    }

    /**
     * @param $view
     * @param array $params
     * @return string
     */
    public function renderOnlyView($view, $params = []): string
    {
        foreach ($params as $key => $value) {
            $$key = $value;
        }

        ob_start();
        include_once dirname(__DIR__) . "/views/$view.phtml";
        return ob_get_clean();
    }
}

```

```

//controllers/doctor/DoctorAppointmentController.php
<?php

namespace app\controllers\doctor;

use app\core\Controller;
use app\models\general\SidebarModel;
use app\models\patient>MainModel;
use app\models\general\AppointmentInfoModel;

class DoctorAppointmentController extends Controller
{
    /**
     * @var int
     */
    protected $doctorId;

    /**
     * @var SidebarModel
     */
    protected $sidebarModel;

    /**
     * @var MainModel
     */
    protected $mainModel;

    /**
     * @var AppointmentInfoModel
     */
    protected $appointmentInfo;

    /**
     * DoctorAppointmentController construct
     */
    public function __construct()
    {
        $this->doctorId = $this->checkLogin('doctor');
        $this->sidebarModel = new SidebarModel();
        $this->mainModel = new MainModel();
        $this->appointmentInfo = new AppointmentInfoModel();
    }

    /**
     * @param $patientId
     * @return string
     */
    public function index($patientId): string
    {
        return $this->renderView(
            'doctor/appointment',
            [
                'patientId' => $patientId,
                'sidebarInfo' => $this->sidebarModel->getSidebarInfo($patientId),
                'types' => $this->appointmentInfo->getTypes(),
            ]
        );
    }
}

```

```

        'repetition'      =>      $this->appointmentInfo-
>getRepetition(),
        'timetable'      =>      $this->mainModel-
>getAvailableTime($this->doctorId)
    ]
    );
}

/**
 * @param $patientId
 */
public function save($patientId): void
{
    $params = $this->getParams();
    $params['patientId'] = $patientId;
    $params['doctorId'] = $this->doctorId;

    $this->mainModel->saveAppointment($params);
    $this->redirect("/doctor/appointment/{ $patientId}");
}
}

//controllers/patient/PatientRegisterController.php
<?php

namespace app\controllers\patient;

use app\core\Controller;
use app\core\Session;
use app\models\base\PatientModel;
use app\models\patient\RegisterModel;

class PatientRegisterController extends Controller
{
    const REGISTER_SUCCESS_MESSAGE = 'Реєстрація пройшла
    успішно!';

    /**
     * @var PatientModel
     */
    protected $patientModel;

    /**
     * @var RegisterModel
     */
    protected $registerModel;

    /**
     * PatientRegisterController construct
     */
    public function __construct()
    {
        $this->patientModel = new PatientModel();
        $this->registerModel = new RegisterModel();
    }

    /**

```

```

        * @return string
        */
public function register(): string
{
    if($this->getParams()) {
        $this->patientModel->loadData($this->getParams());

        if ($this->registerModel->validate($this->
>patientModel) && $this->registerModel->register($this->patientModel))
        {
            Session::getSession()->setFlash('success',
self::REGISTER_SUCCESS_MESSAGE);
            $this->redirect('/');
        }

        return $this->renderView(
            'patient/login',
            [
                'errors' => $this->patientModel->errors
            ]
        );
    }

    return $this->renderView('patient/login');
}
}

```

```

//models/base/Patient.php
<?php

```

```

namespace app\models\base;

use app\core\Model;

class PatientModel extends Model
{
    /**
     * @var string
     */
    private $surname;

    /**
     * @var string
     */
    private $name;

    /**
     * @var string
     */
    private $fatherName;

    /**
     * @var string
     */
    private $email;

    /**

```

```

    * @var string
    */
private $password;

/**
 * @var int
 */
private $genderId;

/**
 * @var string
 */
private $dateOfBirthday;

/**
 * @var string
 */
private $phone;

/**
 * @var string
 */
private $address;

/**
 * @return string
 */
public function getSurname(): string
{
    return $this->surname;
}

/**
 * @param $surname
 */
public function setSurname($surname): void
{
    $this->surname = $surname;
}

/**
 * @return string
 */
public function getName(): string
{
    return $this->name;
}

/**
 * @param $name
 */
public function setName($name): void
{
    $this->name = $name;
}

/**
 * @return string

```

```

    */
    public function getFatherName(): string
    {
        return $this->fatherName;
    }

    /**
     * @param $fatherName
     */
    public function setFatherName($fatherName): void
    {
        $this->fatherName = $fatherName;
    }

    /**
     * @return string
     */
    public function getEmail(): string
    {
        return $this->email;
    }

    /**
     * @param $email
     */
    public function setEmail($email): void
    {
        $this->email = $email;
    }

    /**
     * @return string
     */
    public function getPassword(): string
    {
        return $this->password;
    }

    /**
     * @param $password
     */
    public function setPassword($password): void
    {
        $this->password = $password;
    }

    /**
     * @return int
     */
    public function getGenderId(): int
    {
        return (int)$this->genderId;
    }

    /**
     * @param $genderId
     */
    public function setGenderId($genderId): void

```

```

{
    $this->genderId = (int)$genderId;
}

/**
 * @return string
 */
public function getDateOfBirthday(): string
{
    return $this->dateOfBirthday;
}

/**
 * @param $dateOfBirthday
 */
public function setDateOfBirthday($dateOfBirthday): void
{
    $this->dateOfBirthday = $dateOfBirthday;
}

/**
 * @return string
 */
public function getPhone(): string
{
    return $this->phone;
}

/**
 * @param $phone
 */
public function setPhone($phone): void
{
    $this->phone = $phone;
}

/**
 * @return string|null
 */
public function getAddress(): ?string
{
    return $this->address;
}

/**
 * @param $address
 */
public function setAddress($address): void
{
    $this->address = $address;
}

/**
 * @return string
 */
public function getTableName(): string
{
    return 'patient';
}

```

```

    }

    /**
     * @return string
     */
    public function getRole(): string
    {
        return 'patient';
    }
}

//models/patient/InfoModel.php
<?php

namespace app\models\patient;

use app\core\DB;
use app\core\Logger;
use app\core\Model;

class InfoModel extends Model
{
    /**
     * @var Logger
     */
    private $logger;

    /**
     * InfoModel construct
     */
    public function __construct()
    {
        $this->logger = new Logger();
    }

    /**
     * @return array
     */
    public function rules(): array
    {
        return [
            'email' => [self::RULE_REQUIRED, self::RULE_EMAIL],
            'password' => [self::RULE_REQUIRED],
            'surname' => [self::RULE_REQUIRED],
            'name' => [self::RULE_REQUIRED],
            'fatherName' => [self::RULE_REQUIRED],
            'phone' => [self::RULE_REQUIRED, self::RULE_PHONE],
            'genderId' => [self::RULE_REQUIRED],
            'dateOfBirthday' => [self::RULE_REQUIRED,
self::RULE_DATE],
            'address' => []
        ];
    }

    /**
     * @param $id
     * @return array

```

```

        */
        public function getInfo($id): array
        {
            try {
                $db = DB::getConnection();
                $data = $db->prepare('SELECT * FROM patient WHERE id
= :id');

                $data->execute(array('id' => $id));
                return (array)$data->fetch(\PDO::FETCH_ASSOC);
            } catch (\Exception $exception) {
                $this->logger->error($exception->getMessage());
            }

            return [];
        }

        /**
         * @param $id
         * @param $params
         * @return bool
         */
        public function editInfo($id, $params): bool
        {
            try {
                $db = DB::getConnection();
                $data = $db->prepare('UPDATE patient SET surname =
:surname, name = :name, fatherName = :fatherName, genderId =
:genderId, email = :email, password = :password, dateOfBirthday =
:dateOfBirthday, phone = :phone, address = :address WHERE id = :id');
                $data->execute(['surname' => $params['surname'],
'name' => $params['name'], 'fatherName' => $params['fatherName'],
'genderId' => $params['genderId'], 'email' => $params['email'],
'password' => $params['password'], 'dateOfBirthday' =>
$params['dateOfBirthday'], 'phone' => $params['phone'], 'address' =>
$params['address'], 'id' => $id]);
                return true;
            } catch (\Exception $exception) {
                $this->logger->error($exception->getMessage());
            }

            return false;
        }
    }

    //view/patient/login.phtml
    <div class="login-page">
        <div class="login-container">
            <form action="login" method="post" class="login-form
login">

                <div class="mb-3 login-field login-container-field">
                    <label for="email" class="form-label">Электронна
пошта</label>

                    <input name="email" type="email" id="email"
class="form-control <?=isisset($errorsLogin['email']) ? 'is-invalid' :
'' ?>">

                    <div class="invalid-feedback">
                        <?=? reset($errorsLogin['email'])?>

```

```

        </div>
    </div>
    <div class="mb-3 login-field login-container-field">
        <label          for="password"          class="form-
label">Пароль</label>
        <input          name="password"          type="password"
class="form-control <?= isset($errorsLogin['password']) ? 'is-invalid'
: '' ?>" id="password">
        <div class="invalid-feedback">
            <?= reset($errorsLogin['password'])?>
        </div>
    </div>
    <button type="submit" name="login" class="btn btn-lg
btn-login">Увійти</button>
    </form>
</div>

<div class="register-container">
    <form          class="register-form"          action="register"
method="post">
        <div class="mb-3 login-field">
            <label for="email" class="form-label">Електронна
пошта</label>
            <input name="email" type="email" id="email"
class="form-control <?= isset($errors['email']) ? 'is-invalid' : ''
?>">
            <div class="invalid-feedback">
                <?= reset($errors['email'])?>
            </div>
        </div>
        <div class="mb-3 login-field">
            <label          for="password"          class="form-
label">Пароль</label>
            <input          name="password"          type="password"
class="form-control <?= isset($errors['password']) ? 'is-invalid' : ''
?>" id="password">
            <div class="invalid-feedback">
                <?= reset($errors['password'])?>
            </div>
        </div>
        <div class="mb-3 login-field">
            <label          for="surname"          class="form-
label">Прізвище</label>
            <input name="surname" type="text" id="surname"
class="form-control <?= isset($errors['surname']) ? 'is-invalid' : ''
?>">
            <div class="invalid-feedback">
                <?= reset($errors['surname'])?>
            </div>
        </div>
        <div class="mb-3 login-field">
            <label          for="name"          class="form-
label">Ім'я</label>
            <input name="name" type="text" class="form-
control <?= isset($errors['name']) ? 'is-invalid' : '' ?>" id="name">
            <div class="invalid-feedback">
                <?= reset($errors['name'])?>
            </div>
        </div>
    </form>
</div>

```

```

        </div>
        <div class="mb-3 login-field">
            <label for="fatherName" class="form-label">По
батъкові</label>
            <input name="fatherName" type="text"
id="father_name" class="form-control <?= isset($errors['fatherName'])
? 'is-invalid' : '' ?>">
            <div class="invalid-feedback">
                <?= reset($errors['fatherName']) ?>
            </div>
        </div>
        <div class="mb-3 login-field">
            <label for="phone" class="form-
label">Телефон</label>
            <input class="form-control <?=
isset($errors['phone']) ? 'is-invalid' : '' ?>" type="tel" id="phone"
name="phone" placeholder="+380000000000" >
            <div class="invalid-feedback">
                <?= reset($errors['phone']) ?>
            </div>
        </div>
        <button type="submit" name="register" class="btn
btn-lg btn-register">Зареєструватися</button>
    </form>
</div>
</div>

```

```

//python/recognize.py
#!/usr/bin/env python
import sys
import os
import numpy as np
from PIL import Image
from tensorflow.keras.models import load_model

classes = ['Actinic Keratoses', 'Basal Cell Carcinoma', 'Benign
Keratosis Like Lesions',
'Dermatofibroma', 'Melanocytic Nevi', 'Melanoma',
'Vascular Lesions']

model = load_model(os.path.dirname(__file__) + '/model.h5')

img_path = os.path.dirname(__file__) + '/../image/patient/' +
str(sys.argv[1])
img = np.asarray(Image.open(img_path).resize((100,75)))
img = (img - np.mean(img)) / np.std(img)
x = img.reshape(1, *(75, 100, 3))

prediction = np.argmax(model.predict(x))
print(classes[prediction])

```

ДОДАТОК В
Слайди презентації

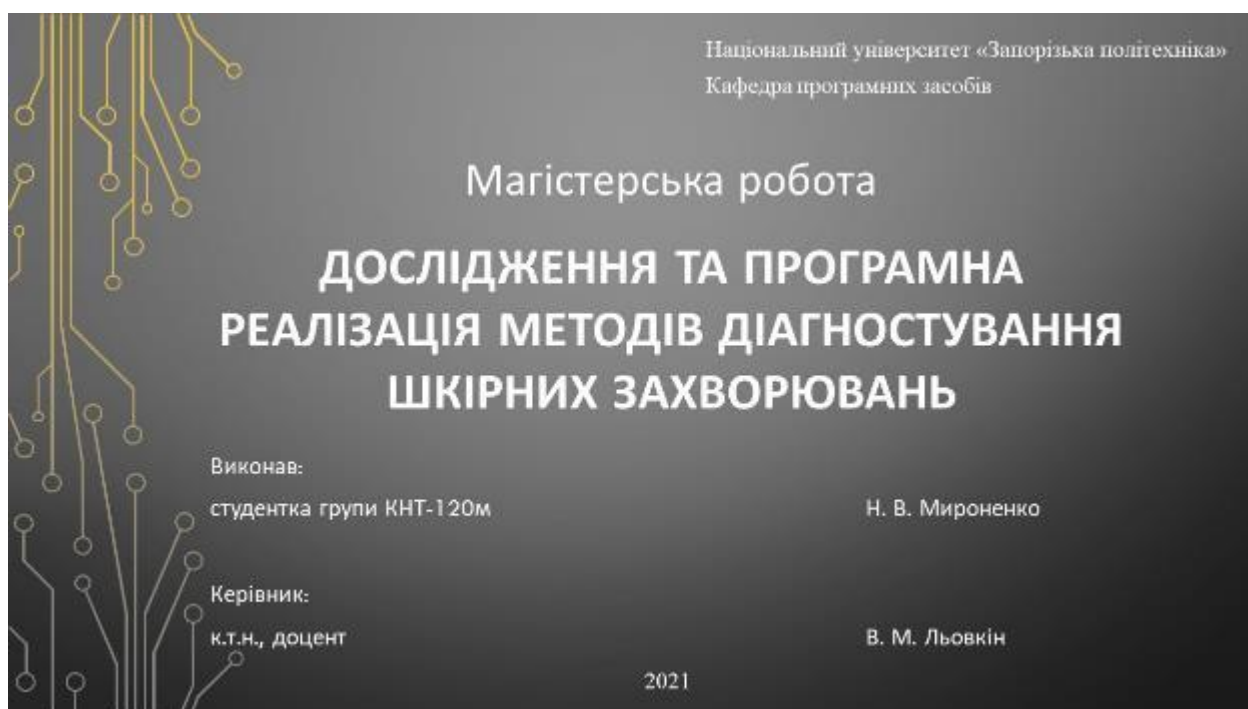


Рисунок В.1 – Слайд 1

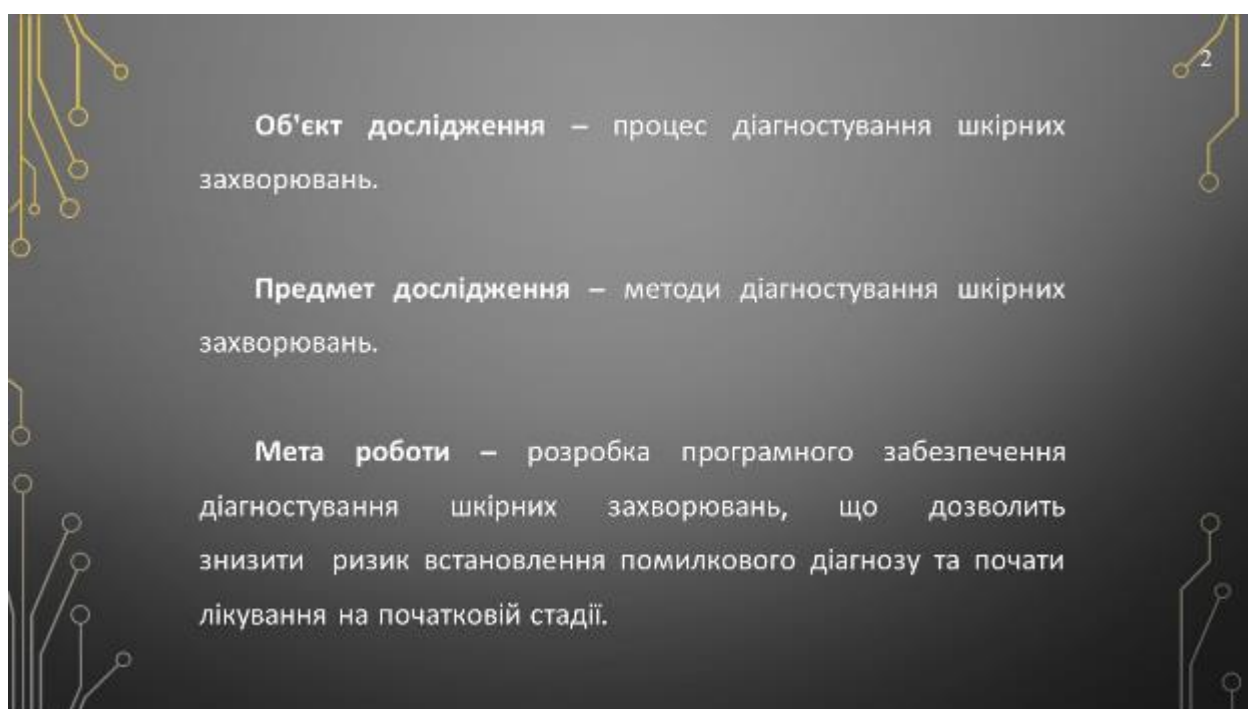


Рисунок В.2 – Слайд 2

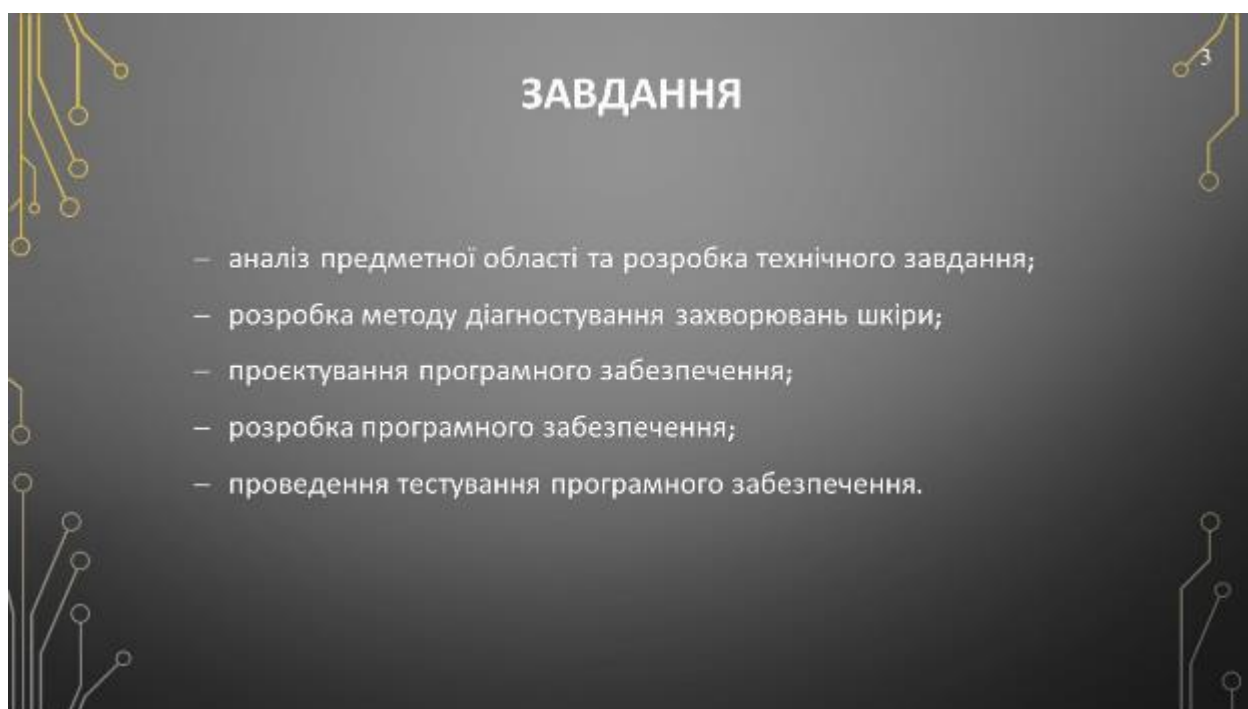


Рисунок В.3 – Слайд 3



Рисунок В.4 – Слайд 4

ВИБІР МОВИ ПРОГРАМУВАННЯ							
Критерії	Python	R	Java	Критерії	PHP	Ruby	Perl
Об'єктно-орієнтованість	Так	Так	Так	Об'єктно-орієнтованість	Так	Так	Так
Обробка великого обсягу даних	Добре	Гірше	Добре	Продуктивність	Висока	Нижча	Нижча
Простота вивчення	Просто	Складно	Складно	Складність розробки	Просто	Просто	Складно
Бібліотеки	Багато	Менше	Багато	Простота вивчення	Просто	Складно	Складно
Складність розробки	Просто	Складно	Просто, але потребує більше рядків коду, ніж Python	Популярність	Велика	Нижче	Нижче
Швидкість виконання коду	Швидко	Повільніше	Швидко	Спільнота підтримки	Велика	Менша	Менша
Популярність	Велика	Менша	Велика	Безкоштовна ліцензія	Так	Так	Так

Рисунок В.5 – Слайд 5



Рисунок В.6 – Слайд 6



Рисунок В.7 – Слайд 7

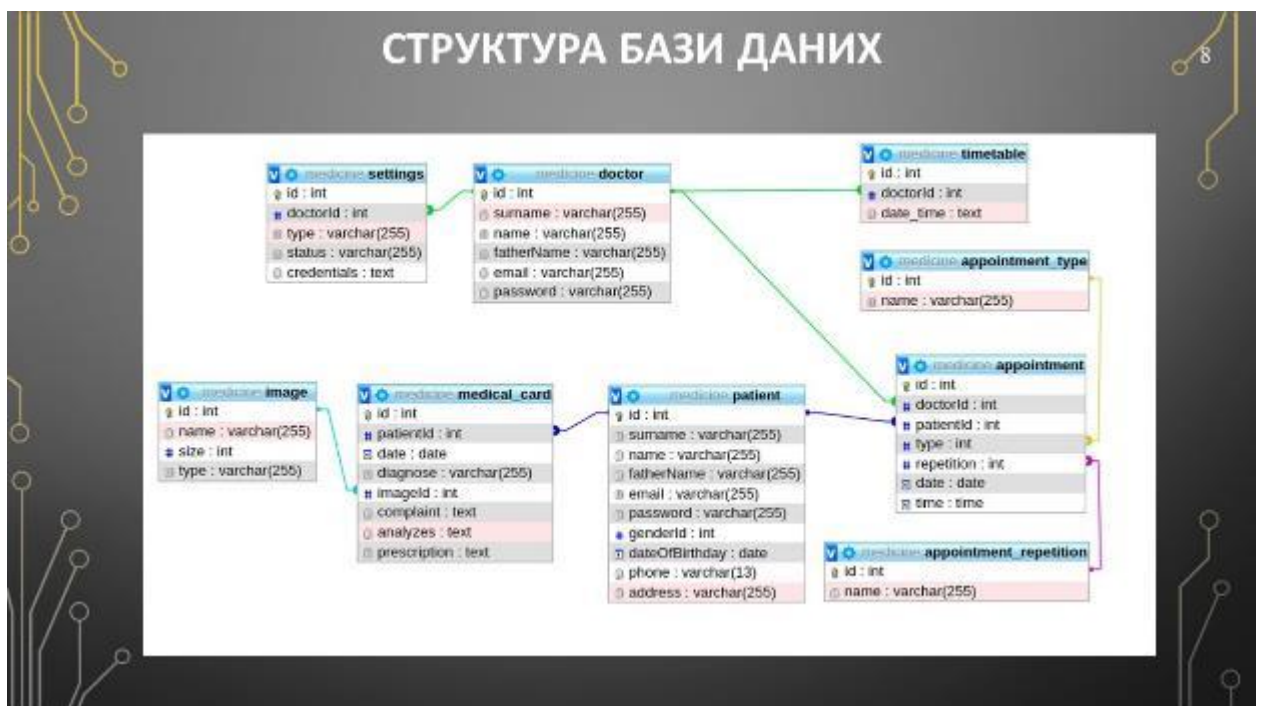


Рисунок В.8 – Слайд 8

Рисунок В.9 – Слайд 9



Рисунок В.10 – Слайд 10

РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТАЛЬНОГО ДОСЛІДЖЕННЯ

Модель	Точність розпізнавання
EfficientNetB3	77.14%
VGG16	78.00%
ResNet34	78.84%
ResNet101	78.96%
VGG19	79.02%
SEResNet50	80.72%
VGG16	81.65%
MobileNet	82.62%
MobileNet	84.73%
ResNet	85.20%
DenseNet	85.25%
VGG-16	86.09%
AlexNet	88.81%
ResNet	90.50%
Ensemble	92.83%
ResNeXt101	93.20%
Розроблений метод з модифікованою моделлю ResNet50	96.31%

Рисунок В.11 – Слайд 11

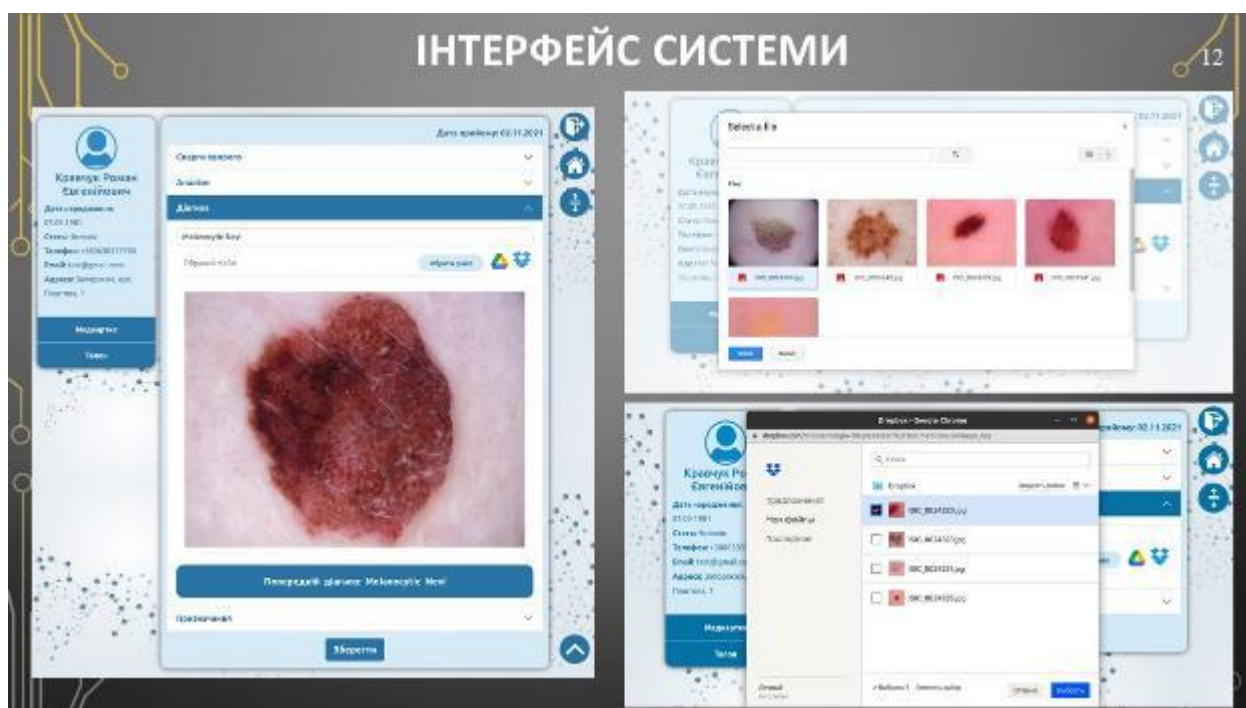


Рисунок В.12 – Слайд 12

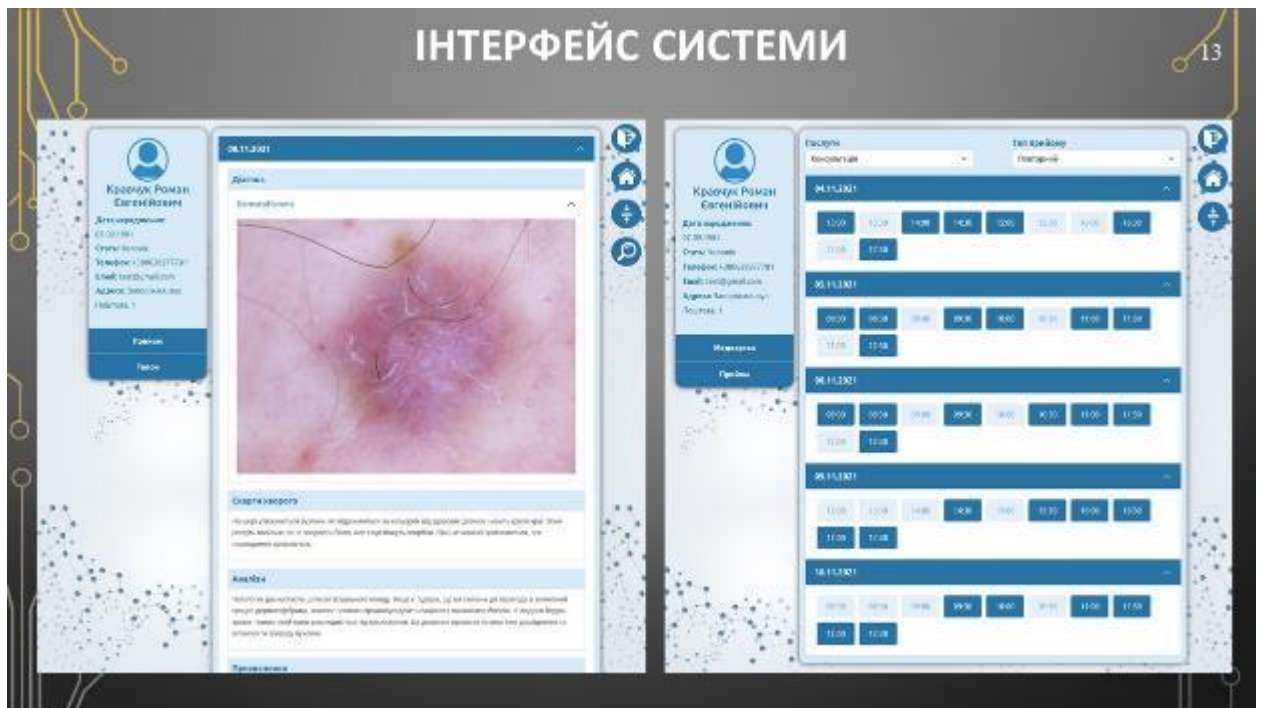


Рисунок В.13 – Слайд 13

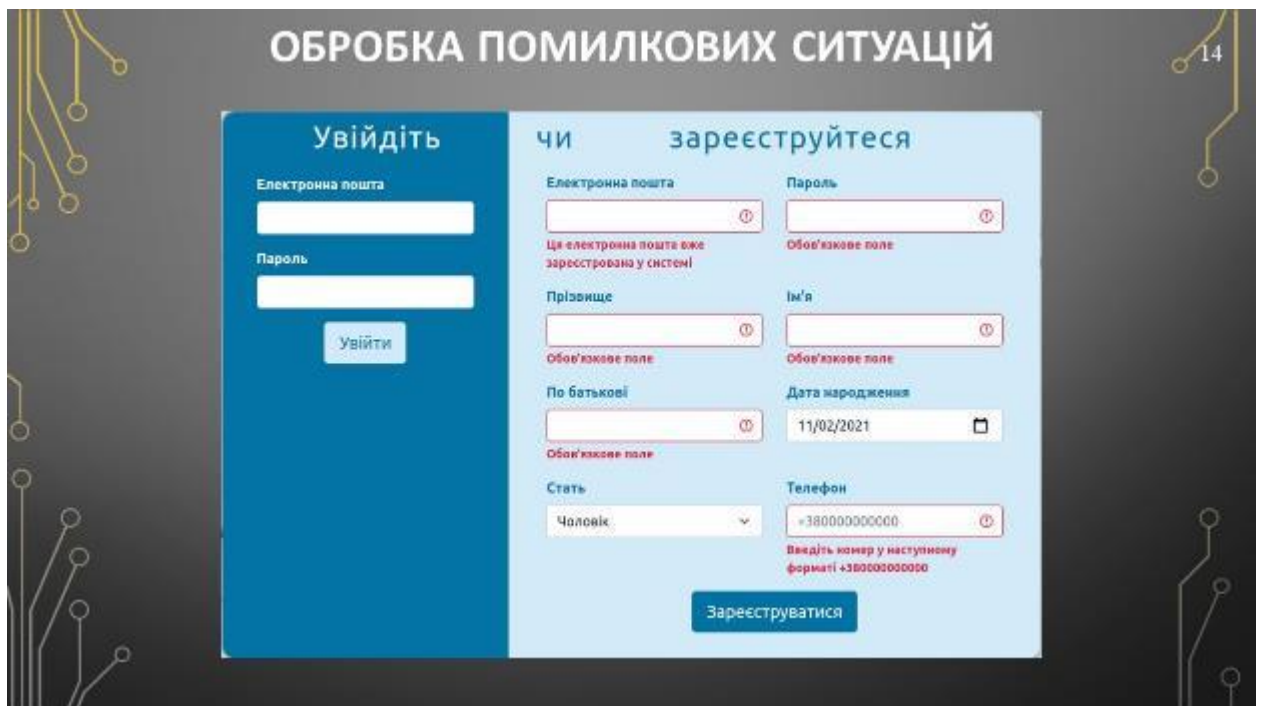


Рисунок В.14 – Слайд 14

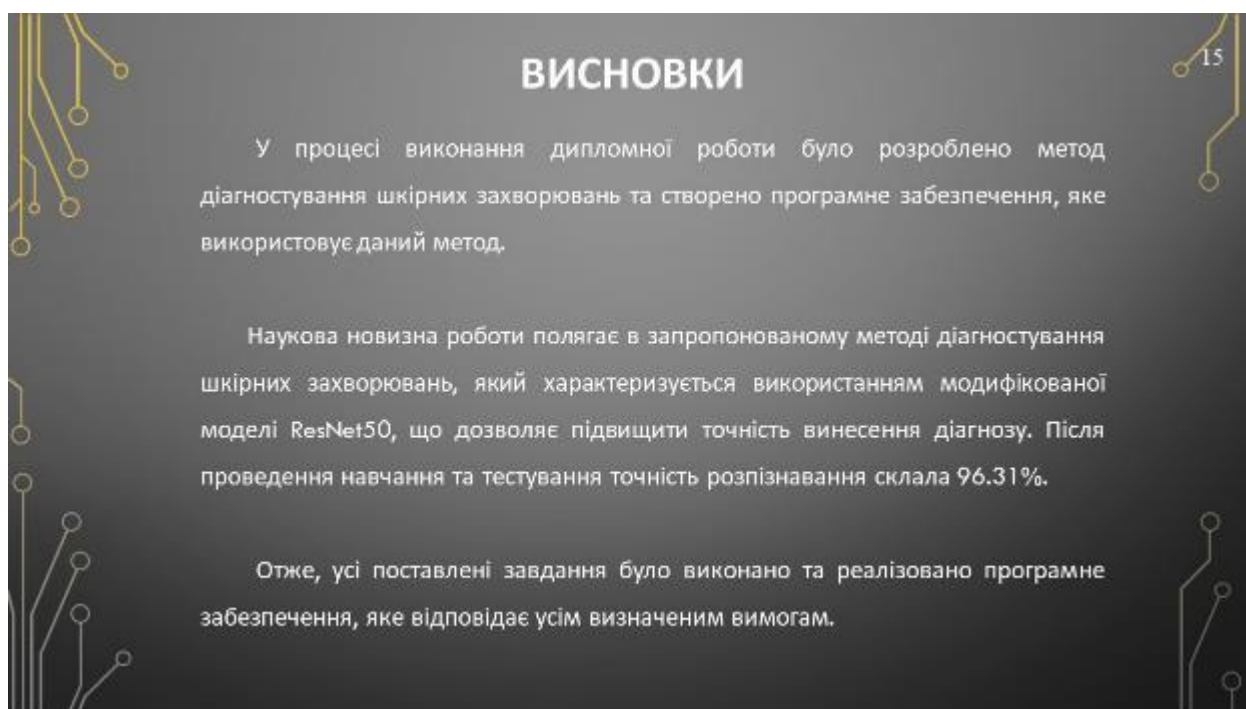


Рисунок В.15 – Слайд 15

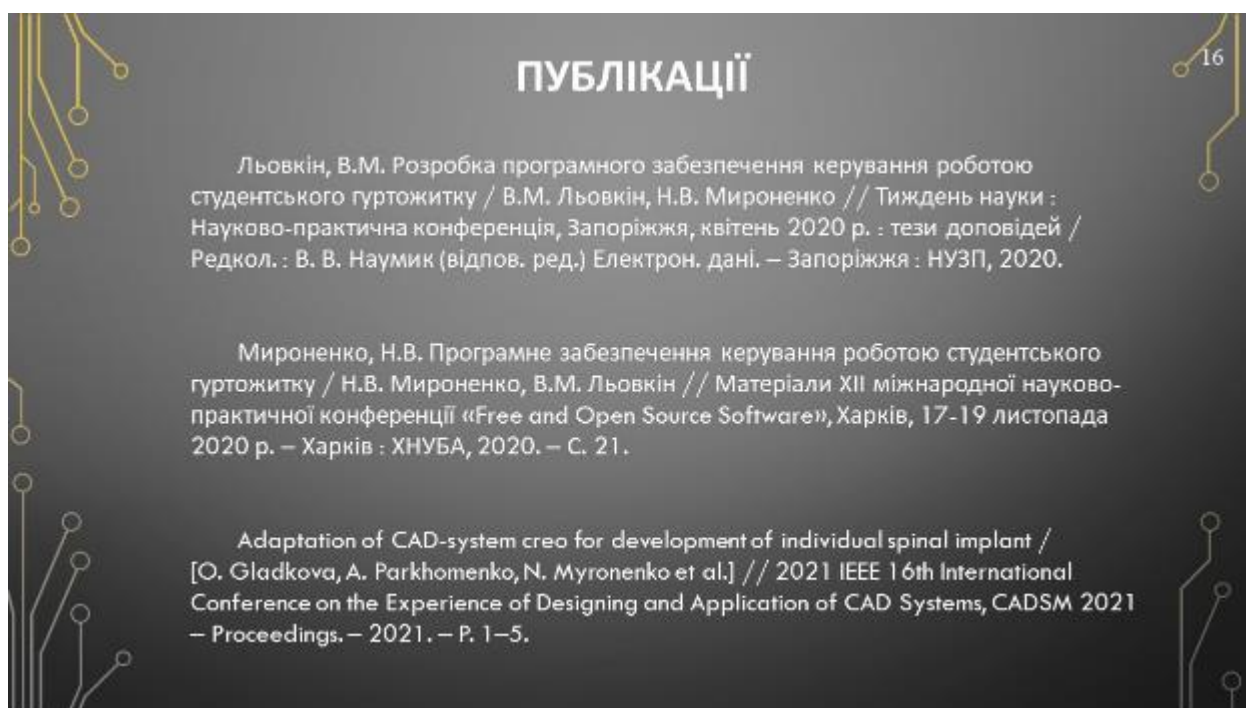


Рисунок В.16 – Слайд 16