

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ОБЛІКУ ЛІКАРСЬКИХ ЗАСОБІВ
УНІВЕРСАЛЬНОЇ АПТЕЧКИ
SOFTWARE FOR THE ACCOUNTING OF MEDICINES IN A UNIVERSAL
FIRST AID KIT

Виконала: студентка 4 курсу, групи КНТ-130

Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ПАНИЧУК К.С.

(ПРІЗВИЩЕ та ініціали)

Керівник

ОЛІЙНИК А.О.

(ПРІЗВИЩЕ та ініціали)

Рецензент

КОЦУР М.І.

(ПРІЗВИЩЕ та ініціали)

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

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
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.
Сергій СУББОТІН
« » 2024 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТКИ

ПАНИЧУК Катерини Сергіївни

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення обліку лікарських засобів універсальної аптечки. Software for the Accounting of Medicines in a Universal First Aid Kit

керівник проєкту (роботи) д.т.н., професор, ОЛІЙНИК Андрій Олександрович,

(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від «24» квітня 2024 року №168

2. Строк подання студентом проєкту (роботи) 10 червня 2024 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Вибір та обґрунтування структури системи, яка проєктується. 3. Основні рішення з реалізації компонентів системи. 4. Керівництво програміста. 5. Керівництво оператора. 6. Тестування і результати.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-6 Основна частина	ОЛІЙНИК А.О., професор		
Нормоконтроль	ЛИПОВЕЦЬ М.В., асистент		

7. Дата видачі завдання « 15 » квітня 2024 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 предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3-6
6	Створення керівництв програміста та оператора.	5 тиждень	Розділи 4, 5
7	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 6
8	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
9	Нормоконтроль та рецензування.	7 тиждень	
10	Захист роботи.	8 тиждень	

Студентка

Підпис
(підпис)

Катерина ПАНИЧУК
(Ім'я, ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис)

Андрій ОЛІЙНИК
(Ім'я, ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 134 с., 21 табл., 77 рис., 4 дод., 27 джерел.

ОБЛІК, ЛІКАРСЬКІ ЗАСОБИ, АПТЕЧКА, ПАКОВАННЯ, РОЗПІЗНАВАННЯ, МОБІЛЬНИЙ ЗАСТОСУНОК, .NET MAUI, PYTHON.

Об'єкт дослідження – процес обліку лікарських засобів аптечки.

Предмет дослідження – методи обліку лікарських засобів аптечки.

Мета роботи – створення програмного забезпечення, яке надає можливість обліку лікарських засобів універсальної аптечки, тобто дозволяє зберігати інформацію про наявні лікарські засоби, наприклад, їхні назви, кількість, терміни придатності тощо.

Метод дослідження – облік вручну, сканування штрих-кодів, розпізнавання зображень пакувань лікарських засобів.

Результати. Створено мобільний застосунок, який дозволяє здійснювати облік лікарських засобів, а також контролювати курс лікування. Було забезпечено безпеку передачі зображень за допомогою шифрування.

Висновки. Створено програмне забезпечення, для якого було розроблено модифіковану модель згорткової нейронної мережі, точність якої складає 92%, та яка правильно розпізнала усі 5 тестових зображень пакувань лікарських засобів. Вона показала гарні результати на рівні 6 моделей, які засновані на вже відомих, а саме: InceptionV3, Xception, ResNet50V2, MobileNetV2, NASNetMobile та DenseNet169. Крім того, було впроваджено методи шифрування даних AES, RSA та комбінацію цих алгоритмів. За результатами аналізу було зроблено висновок, що гібридне шифрування є найкращим для розробленого програмного забезпечення.

Сфера застосування – будь-яка сфера, де користувач має аптечку і має потребу слідкувати за станом ліків та правильним їх застосуванням.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 134 pages, 21 tables, 77 figures, 4 appendixes, 27 sources.

ACCOUNTING, MEDICINES, FIRST AID KIT, PACKAGING, RECOGNITION, MOBILE APPLICATION, .NET MAUI, PYTHON.

The object of research is the process of accounting for medicines in the universal first aid kit.

The subject of research is methods of accounting for medicines in the universal first aid kit.

The purpose of this work is to create software that provides the ability to account for medicines of a first aid kit, i.e. allows you to store information about available medicines, such as their names, quantity, expiration dates, etc.

The research methods were manual accounting, barcode scanning, and image recognition of drug packages.

Results. A mobile application has been created that allows you to account for medicines and monitor the course of treatment. The security of image transmission was ensured through encryption.

Conclusions. The software was created for which a modified convolutional neural network model was developed, which is 92% accurate and correctly recognized all 5 test images of drug packages. It showed good results at the level of 6 models based on already known ones, namely: InceptionV3, Xception, ResNet50V2, MobileNetV2, NASNetMobile and DenseNet169. In addition, AES, RSA and a combination of these algorithms were implemented. Based on the results of the analysis, it was concluded that hybrid encryption is the best for the developed software.

The scope of application is any area where the user has a first aid kit and needs to monitor the condition of medicines and their proper use.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз предметної області.....	10
1.1 Огляд відомих методів.....	10
1.2 Огляд наявних програмних засобів, що виконують аналогічні завдання	13
1.3 Постановка завдання роботи.....	18
1.4 Висновки за розділом 1	19
2 Вибір та обґрунтування структури системи, яка проєктується.....	20
2.1 Вибір мови програмування	20
2.2 Порівняння мов програмування	27
2.3 Вибір середовища розробки.....	29
2.4 Порівняння середовищ розробки	33
2.5 Вибір системи управління базами даних.....	34
2.6 Порівняння систем управління базами даних.....	37
2.7 Структурна схема програми.....	37
2.8 Опис модулів розроблюваної програми	38
2.9 Висновки за розділом 2	39
3 Основні рішення з реалізації компонентів системи	40
3.1 Вибір шаблону проєктування	40
3.2 Опис модулів розроблюваної системи.....	42
3.3 Створення моделі для розпізнавання	45
3.4 Проєктування бази даних	61
3.5 Основні рішення щодо шифрування зображень.....	62
3.6 Основні рішення щодо розробки інтерфейсу.....	67
3.7 Обробка виняткових ситуацій	70
3.8 Висновки за розділом 3	70
4 Керівництво програміста.....	72
4.1 Призначення та умови застосування програми	72
4.2 Характеристика програми	73
4.3 Звертання до програми	75

	7
4.4 Вхідні та вихідні дані.....	75
4.5 Повідомлення програмісту.....	76
4.6 Висновки за розділом 4	77
5 Керівництво оператора	78
5.1 Призначення програми	78
5.2 Умови виконання програми	78
5.3 Виконання програми.....	78
5.4 Повідомлення оператору	86
5.5 Висновки за розділом 5	87
6 Тестування і результати.....	88
6.1 Опис методики проведення тестування.....	88
6.2 Результати використання розробленого програмного забезпечення	89
6.3 Висновки за розділом 6	94
Висновки	95
Перелік джерел посилань	97
Додаток А Технічне завдання	100
Додаток Б Опис програми	111
Додаток В Текст програми.....	118
Додаток Г Слайди презентації	129

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

AES	– Advanced Encryption Standard;
API	– Application Programming Interface;
CFB	– Cipher Feedback;
HTTP	– Hypertext Transfer Protocol;
IDE	– Integrated Development Environment;
RSA	– Rivest, Shamir, Adleman;
SQL	– Structured Query Language;
.NET MAUI	– .NET Multi-Platform App UI;
БД	– база даних;
ОС	– операційна система;
СУБД	– система управління базами даних;
$\oplus$	– операція виключне АБО.

ВСТУП

Лікарські засоби відіграють надзвичайно важливу роль у житті людини, допомагаючи зберегти здоров'я та покращити якість життя. Однак, часто люди пропускають приймання ліків або переплачують за ті, які вже є у них вдома, але вони не пам'ятають про це. Всесвітня організація охорони здоров'я класифікувала відсутність дотримання режиму лікування як основну глобальну проблему [1]. За оцінками, від 20% до 50% пацієнтів не приймають ліки належним чином [1]. Причини цього доволі різноманітні, але найчастішими причинами є ненавмисні, такі як розгубленість або проста забудькуватість. Ці проблеми можуть мати серйозні наслідки для здоров'я та збільшити витрати на лікування. У зв'язку з цим, з'являється потреба у зручному та ефективному засобі обліку лікарських засобів.

Програмне забезпечення обліку лікарських засобів універсальної аптечки є актуальним рішенням для вирішення цих проблем. Воно дозволяє додавати, видаляти та редагувати лікарські засоби аптек, стежити за їхніми термінами придатності, а також дає можливість налаштувати нагадування про приймання ліків. Це особливо корисно для людей, які приймають багато лікарських засобів або мають хронічні захворювання.

Використання мобільного застосунку для обліку є зручним та доступним, оскільки зазвичай мобільний телефон завжди знаходиться поруч із власником. Це також має значний вплив на екологію, бо дозволяє зменшити кількість небезпечних відходів, пов'язаних з неправильним зберіганням та викиданням прострочених ліків.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Для того, щоб аптечка була ефективною та безпечною, необхідно правильно вести облік лікарських засобів, які вона містить. Це є надзвичайно важливим, оскільки від правильного застосування та належного стану цих засобів залежить життя та здоров'я людини. Предметною областю обліку лікарських засобів універсальної аптечки є фармацевтична галузь. Для розробки необхідно мати знання про різні лікарські засоби, їхню дію, класифікацію, форми випуску, терміни придатності, дозування та способи застосування.

1.1 Огляд відомих методів

Завдання обліку лікарських засобів аптечки не є новим, оскільки люди вже давно використовують аптечки для зберігання медикаментів. Розвиток технологій зробив його виконання більш зручним, ефективним і доступним для користувачів.

Існує кілька методів виконання завдання обліку лікарських засобів аптечки, а саме: облік вручну, сканування штрих-кодів, розпізнавання зображень пакувань лікарських засобів.

1.1.1 Облік вручну

Облік вручну полягає у введенні даних про лікарські засоби вручну. Користувач може ввести назву, термін придатності, кількість та іншу інформацію про кожен лікарський засіб відповідно до потреб. Цей метод простий, але може бути часо- та трудомістким.

Переваги:

- гнучкість введення даних та можливість врахувати додаткову інформацію;

- не вимагає додаткового обладнання або технологій.

Недоліки:

- трудомісткість та часові затрати на ручне введення даних;
- ймовірність помилок при введенні даних.

1.1.2 Сканування штрих-кодів

Мобільні застосунки можуть використовувати камеру смартфона для сканування штрих-кодів, розташованих на пакуваннях лікарських засобів. Після сканування застосунок автоматично знаходить лікарський засіб в БД і додає його до списку ліків. Цей метод зручний та швидкий, оскільки дозволяє уникнути ручного введення даних.

Переваги:

- швидке та зручне додавання лікарських засобів до списку ліків;
- мінімізація помилок введення даних.

Недоліки:

- потреба в якісній камері телефону для надійного сканування;
- залежність від наявності штрих-кодів на пакуваннях лікарських засобів;
- необхідність мати базу даних з лікарськими засобами та значеннями їхніх штрих-кодів.

1.1.3 Розпізнавання зображень пакувань лікарських засобів

Мобільний застосунок може використовувати систему розпізнавання зображень для ідентифікації лікарських засобів. Користувач може зробити фотографію пакування лікарського засобу, і застосунок автоматично додасть цей засіб до списку ліків. Цей метод дозволяє легко додавати нові засоби до списку, але точність розпізнавання може залежати від якості зображення, моделі та бази даних розпізнавання.

Переваги:

- легкість використання та зручність для користувача;
- можливість швидкого додавання засобів до списку ліків за допомогою фотографії.

Недоліки:

- точність розпізнавання залежить від якості зображення, моделі та бази даних розпізнавання;
- потреба в постійному оновленні бази даних та покращенні моделі для розпізнавання нових лікарських засобів.

1.1.4 Порівняльна таблиця відомих методів

Вище було наведено відомі методи обліку лікарських засобів. Вони мають свої переваги і недоліки. В табл. 1.1 наведено порівняння цих методів.

Таблиця 1.1 — Порівняння методів обліку

Критерії порівняння	Облік вручну	Сканування штрих-кодів	Розпізнавання зображень пакувань
Ефективність	Низька	Висока	Висока
Зручність використання	Середня	Висока	Висока
Надійність та точність	Залежить від користувача	Висока	Висока
Цінова доступність	Висока	Середня	Середня
Масштабованість	Низька	Висока	Середня
Необхідність мати камеру	Немає	Висока	Висока
Необхідність мати БД усіх лікарських засобів	Низька	Висока	Середня
Потреба оновлення БД	Немає	Висока	Висока
Потреба оновлення моделі розпізнавання	Немає	Низька	Висока
Складність оновлення БД	Немає	Висока	Середня
Складність оновлення моделі розпізнавання	Немає	Низька	Висока

Для подальшої реалізації було обрано методи обліку вручну та розпізнавання зображень паковань, бо вони дозволяють зробити процес обліку більш доступним та зручним для широкого кола користувачів. Людина зможе обрати метод обліку, який буде відповідати її потребам, а саме вводити дані вручну, розпізнавати лікарський засіб за його пакуванням чи поєднувати ці методи.

Розпізнавання зображень може бути більш універсальним методом, ніж сканування штрих-кодів, оскільки воно дозволяє вести облік різноманітних лікарських засобів, незалежно від типу штрих-коду або його наявності, пошкоджень пакування, відмінностей в дизайні або локалізації.

1.2 Огляд наявних програмних засобів, що виконують аналогічні завдання

Аптечки є невід'ємною частиною нашого життя і відіграють важливу роль у забезпеченні невідкладної медичної допомоги. Це невеликі контейнери, які містять різноманітні медичні засоби та інструменти, необхідні для невеликих медичних втручань (наприклад, шприци) або першої допомоги.

Люди використовують аптечки в різних сферах своєї діяльності. Вони можуть бути присутніми вдома, на роботі, у школах, автомобілях, в подорожах та при спортивних заняттях. Аптечки також широко застосовуються в галузі туризму, екстремальних видів спорту та військової справи.

Важливо правильно доглядати за аптечками та якісно вести облік медичних засобів, що в них містяться. Це необхідно для забезпечення безпеки та ефективного використання ліків у випадку нагальної потреби. Зберігання лікарських препаратів відповідно до рекомендацій виробника, перевірка термінів придатності та заміна препаратів, які втратили свою ефективність, є ключовими аспектами догляду за аптечкою. Також важливо

мати чіткий перелік медичних засобів та оновлювати його вчасно, щоб мати повну інформацію про наявні ресурси у разі необхідності.

Наразі вже існують програмні засоби, які забезпечують виконання обліку лікарських засобів. Ці програми дозволяють точно відстежувати наявність та кількість препаратів, контролювати терміни придатності, а також оновлювати аптечки відповідно до потреб.

Такі програмні засоби зазвичай надають можливість створювати переліки лікарських засобів, вносити інформацію про кожен засіб та вказувати кількість одиниць, що є в наявності. Деякі з них навіть дозволяють сканувати штрих-коди на пакуваннях препаратів для автоматичного заповнення даних. Крім того, програми можуть відображати сповіщення про наближення до закінчення терміну придатності або нестачу певних засобів, що дозволяє своєчасно поповнювати запаси.

Кожна програма має свої переваги та недоліки, тому перед вибором необхідно ретельно оцінити їхню функціональність, зручність використання та інші параметри.

1.2.1 Мобільний застосунок «Мои лекарства: учет аптечки»

«Мои лекарства: учет аптечки» — це мобільний застосунок (рис. 1.1), що дозволяє легко вести облік ліків у домашній аптечці [2]. Він допомагає відслідковувати прострочені ліки, створювати список ліків, які потрібно купити в найближчий час, та нагадує про необхідність придбання [2].

Основні переваги мобільного застосунку:

- групування за терміном придатності (прострочені та непрострочені);
- сортування за назвою, терміном придатності та датою додавання;
- пошук серед доданих ліків;
- експорт даних.

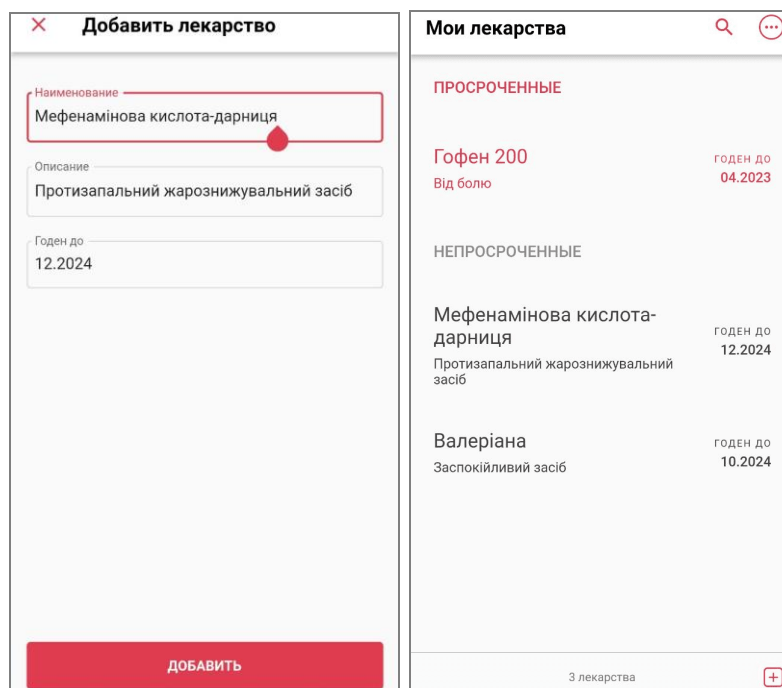


Рисунок 1.1 — Мобільний застосунок «Мои лекарства: учет аптечки» [2]

1.2.2 Мобільний застосунок «Медицинский справочник-аптечка»

«Медицинский справочник-аптечка» — це мобільний застосунок (рис. 1.2), який призначений для обліку лікарських засобів аптечки та надає можливість ознайомитися з інструкціями, показаннями та протипоказаннями, дозуваннями та способами застосування [3]. Застосунок містить опис більше ніж 450 різних хвороб та майже 4000 лікарських засобів. Він є корисним як для професійних медиків, так і для студентів медичних вузів [3].

Основні переваги мобільного застосунку:

- сортування лікарських засобів та хвороб за алфавітом;
- пошук будь-якого препарату чи хвороби в довіднику за буквами;
- можливість вести облік декількох аптечок;
- групування лікарських засобів в аптечці;
- фільтрування лікарських засобів;
- можливість додати зображення лікарського засобу;
- можливість поділитися аптечкою (надіслати вебсторінку з ліками);
- можливість працювати без доступу до Інтернету.

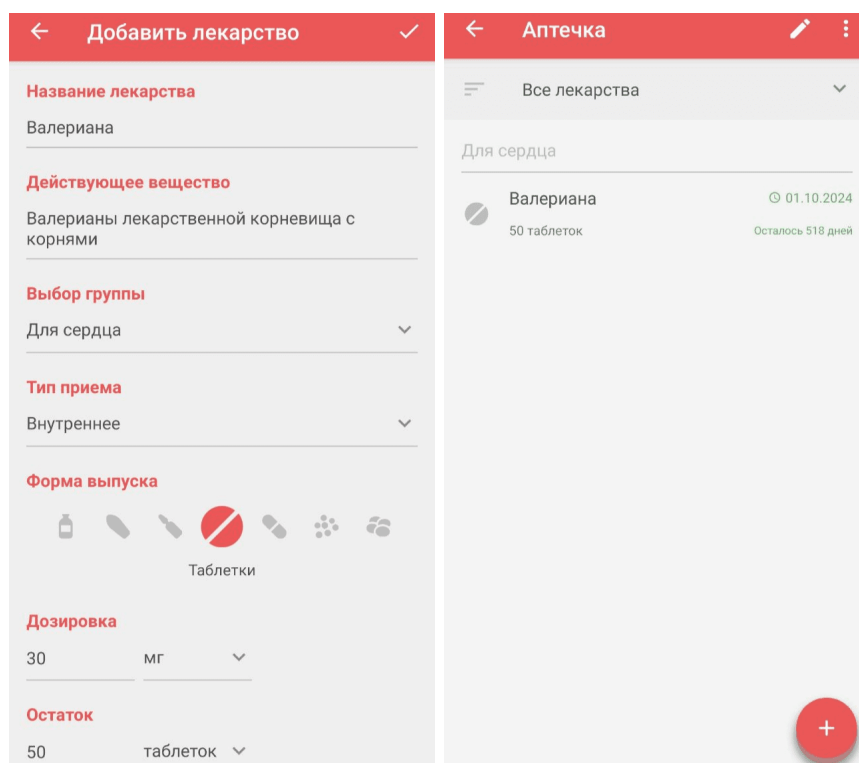


Рисунок 1.2 – Мобільний застосунок «Медицинский справочник-аптека» [3]

1.2.3 Мобільний застосунок «Аптека. Учет медикаментов»

«Аптека. Учет медикаментов» — це мобільний застосунок (рис. 1.3) для ведення обліку лікарських засобів аптечок [4]. Програма дозволяє додавати ліки, отримувати сповіщення про закінчення терміну придатності, виконувати пошук, сканувати штрих-коди пакувань, стежити за ліками в кількох аптечках, таких як домашня, автомобільна, дитяча тощо [4].

Основні переваги мобільного застосунку:

- фільтрування та сортування лікарських засобів;
- можливість вести облік декількох аптечок;
- можливість додати зображення лікарського засобу;
- можливість сканування штрих-коду;
- пошук серед доданих ліків;
- статистика;
- експорт даних.

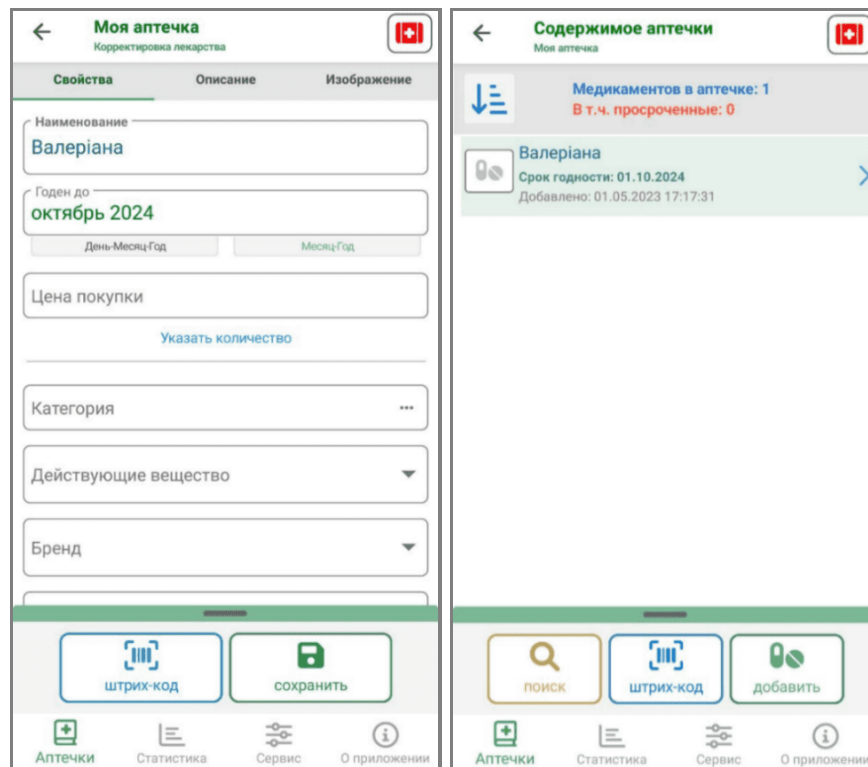


Рисунок 1.3 — Мобільний застосунок «Аптечка. Учет медикаментов» [4]

1.2.4 Порівняльна таблиця наявних та розроблюваного програмних продуктів

Вище було наведено найбільш популярні на сьогоднішній день мобільні застосунки, які дозволяють вести облік лікарських засобів аптек. Але крім основних переваг, усі вони мають і недоліки. Наприклад, відсутність нагадувань про необхідність отримання рецепту від лікаря для певних ліків, відслідковування курсу лікування та нагадувань про приймання ліків, якщо це необхідно користувачу, імпорту власних аптек.

Також існують переваги, характерні для усіх програм: наявність пошуку, сповіщень про закінчення терміну придатності.

У ході виконання роботи буде реалізовано мобільний застосунок, який буде мати наступні переваги, на відміну від наявних аналогів: нагадування про необхідність отримання рецепту від лікаря, відслідковування курсу лікування та нагадування про приймання ліків, якщо

це необхідно користувачу, розпізнавання деяких лікарських засобів за пакуванням.

В таблиці 1.2 проведено порівняння наявних мобільних застосунків та програми, яка буде розроблена в ході виконання роботи.

Таблиця 1.2 — Порівняння мобільних застосунків

Можливості	«Мои лекарства: учет аптечки»	«Медицинский справочник-аптечка»	«Аптечка. Учет медикаментов»	Власний застосунок
Багатоплатформність	+	—	+	+
Українська локалізація	—	—	—	+
Сортування ліків	+	+	+	+
Фільтрування ліків	—	+	+	—
Групування ліків	+	+	+	+
Пошук ліків	+	—	+	—
Експорт даних	+	—	+	—
Створення декількох аптек	—	+	+	—
Сканування штрих-коду	—	—	+	—
Відслідковування курсу лікування	—	—	—	+
Сповіщення	+	+	+	+
Розпізнавання ліків за пакуваннями	—	—	—	+

1.3 Постановка завдання роботи

Мобільний застосунок повинен забезпечувати виконання нижче перерахованих функцій:

- додавання нових лікарських засобів;
- змінення інформації про лікарський засіб;
- видалення лікарського засобу;

- сортування переліку ліків;
- відслідковування курсу лікування;
- відображення того, чи достатня кількість препарату наявна для курсу лікування;
- відображення терміну придатності та індикація того, що наближується його закінчення;
- сповіщення про наближення терміну придатності до закінчення, необхідність отримання рецепту від лікаря для придбання лікарського засобу, настання часу приймати ліки;
- надання можливості зробити фотографію лікарського засобу, її шифрування, розшифрування та подальше розпізнавання.

1.4 Висновки за розділом 1

В першому розділі було розглянуто відомі методи виконання завдання обліку лікарських засобів універсальної аптечки та мобільні застосунки, що реалізують цей облік.

У ході аналізу аналогів було виявлено їхні переваги й недоліки та визначено функціонал, який буде реалізовано у розроблюваному мобільному застосунку.

2 ВИБІР ТА ОБҐРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ, ЯКА ПРОЄКТУЄТЬСЯ

2.1 Вибір мови програмування

Вибір мови програмування для реалізації програмного забезпечення є важливим етапом у процесі розробки. Існує кілька популярних мов програмування, які часто використовуються для створення мобільних застосунків, таких як Java, Swift, Kotlin і C# [5]. Кожна з цих мов має свої особливості, переваги та недоліки.

При розробці серверної частини проєкту також виникає питання про вибір мови програмування. Існують декілька популярних мов, що використовуються для реалізації серверних застосунків, зокрема Java, C++, C#, та Python [5]. Кожна з цих мов має свої переваги і може бути використана для створення потужних та ефективних серверних програм.

2.1.1 Java

Java — це потужна та надійна об'єктно-орієнтована мова програмування, яка може слугувати багатьом цілям, включаючи створення мобільних, серверних та вебзастосунків [6]. Вона також підходить для вбудованих систем і машинного навчання. Java була випущена в 1995 році компанією Sun Microsystems (зараз є власністю Oracle), і більшість її функцій мають відкритий вихідний код [6]. Мова запозичила багато своїх концепцій з мов Smalltalk та C++ [6].

Java була першою мовою, обраною для розробки мобільних застосунків для Android, оскільки вона є платформонезалежною — це означає, що Java-застосунки можуть працювати на будь-якій платформі, яка підтримує віртуальну машину Java (Java Virtual Machine) [6]. Вона також має власне середовище виконання (Java Runtime Environment) та API [6].

Крім того, Java вважається дуже безпечною завдяки своєму відкритому ключу, автентифікації на основі шифрування, а також системі захисту від вірусів і несанкціонованого доступу [6]. Вона також підтримує багатопоточність (програми можуть виконувати кілька завдань одночасно), що робить програми та системи на Java більш ефективними та швидкими [6]. Основна увага приділяється перевірці помилок під час компіляції, таким чином усуваючи вразливі до помилок стани [6].

Переваги [7]:

- добре підходить для великих, складних завдань;
- існує велика кількість ресурсів та бібліотек, тому розробнику не доведеться створювати новий код для вирішення проблем.

Недоліки [7]:

- потрібно більше часу на написання коду для легких, швидких завдань;
- має деякі суттєві недоліки, які були вирішені іншими мовами, такими як Kotlin.

2.1.2 Kotlin

У травні 2019 року Google оголосив Kotlin офіційною (рекомендованою) мовою для створення застосунків для Android [6]. Це статично типізована мова програмування, яка виходить за рамки розробки програм для Android [6]. Kotlin також стає все більш популярною у серверній та клієнтській веброботі. Оскільки ця мова має відкритий вихідний код і постійно розвивається, то незабаром можна буде побачити її застосування у вбудованих системах, iOS та macOS [6].

Kotlin має як об'єктно-орієнтовані, так і функціональні конструкції, що дозволяє програмістам використовувати її в об'єктно-орієнтованому, функціональному програмуванні або поєднувати елементи обох підходів [6]. Однією з найважливіших переваг використання Kotlin при розробці

застосунків для Android є те, що ця мова набагато лаконічніша за Java — за приблизними підрахунками, код на Kotlin на 40% коротший [6].

Крім того, Kotlin повністю сумісний з Java. Це означає, що можна швидко викликати код на Kotlin з Java і навпаки, що робить адаптацію коду набагато простішою та менш ризикованою [6]. Для ще більшого спрощення процесу міграції (конвертації) програмісти мають у своєму розпорядженні вбудований в IDE автоматизований конвертер з Java в Kotlin [6].

Kotlin також надає більшу безпеку типів. Наприклад, вона підтримує ненульові типи, що зменшує ймовірність появи `NullPointerException` [6]. Серед інших можливостей Kotlin — інтелектуальне приведення типів, функції вищого порядку, функції розширення та лямбди з приймачами [6].

Переваги [7]:

- ця мова сумісна з Java, що дозволяє легко розширювати та вдосконалювати будь-яке програмування на Java за допомогою Kotlin;
- вона забезпечує підтримку на рівні нативної мови, при цьому працює на різних платформах.

Недоліки [7]:

- оскільки Kotlin є більш новою мовою, про неї може бути менше інформації. Це ускладнює розробку та вирішення проблем.

2.1.3 Swift

Swift — це мова програмування з відкритим вихідним кодом і безпекою типів, розроблена компанією Apple для роботи з платформами iOS, OS X, tvOS, watchOS і навіть Linux [6]. Вона була анонсована у 2014 році, а через рік була випущена для загального користування та додана до Xcode [6]. Ця гнучка та масштабована мова дозволяє легко адаптувати безпечний шаблон програмування для додавання функцій до будь-якого застосунку [6].

Код Swift набагато більше нагадує англійську мову, ніж код ObjC, що робить його легшим для читання та написання і, таким чином, менш

схильним до людських помилок [6]. Серед інших важливих особливостей — потужна система обробки помилок, прості у використанні узагальнення, швидка та лаконічна ітерація над низкою колекцій, функціональний шаблон програмування та контроль потоку потужності [6].

Крім того, Swift постачається з автономним IDE під назвою Xcode [6]. Він має функцію автоматичного підрахунку посилань (Automatic Reference Counting) [6]. Завдяки цьому iOS-розробнику не потрібно турбуватися про те, скільки пам'яті займе застосунок [6].

Переваги [7]:

- легко читається як мова;
- завдяки своїй інтерактивній природі, Swift спрощує виявлення помилок під час кодування, тим самим заощаджуючи загальний час розробки та налагодження;
- Swift забезпечує чудовий користувацький досвід і часто вимагає менше пам'яті на пристрої.

Недоліки [7]:

- продукти Apple чудово працюють в екосистемі Apple. Однак сумісність з іншими операційними системами залишається проблемою. Застосунок, написаний на Swift, не так легко переноситься на Android та інші платформи.

2.1.4 C#

C# — це сучасна об'єктно- та компонентно-орієнтована мова програмування з безпекою типів, яка дозволяє програмістам створювати безпечні та надійні застосунки на платформі .NET [6]. Як і Objective-C, C# походить із сімейства мов C, що робить її знайомою для розробників C, C++, Java та JavaScript [6].

Багато можливостей C# роблять можливим створення надійних та довговічних застосунків. Зокрема, функція збору сміття автоматично звільняє

пам'ять, зайняту недоступними невикористаними об'єктами [6]. Типи Nullable захищають від змінних, які не посилаються на виділені об'єкти [6]. У той же час, обробка винятків забезпечує структурований та розширюваний підхід до виявлення помилок та відновлення. І, нарешті, лямбда-вирази дозволяють програмістам писати код відповідно до методів функціонального програмування [6].

Програми на C# працюють на .NET — віртуальній системі виконання, яка називається Common Language Runtime, та наборі бібліотек класів [6].

Переваги [7]:

- забезпечує майже нативний рівень продуктивності та швидкості на iOS та Android;
- C# має простішу підтримку коду, що допомагає звести до мінімуму час, необхідний для безперервної розробки.

Недоліки [7]:

- можливе відносне відставання у викликах API;
- це не найкращий вибір для застосунків зі складним користувацьким інтерфейсом.

2.1.5 C++

Витоки C++ лежать у мові програмування, відомій як C, яка була розроблена у 1972 році [8]. У 1980-х роках співробітник Bell Labs Б'ярн Страуструп почав розробляти вдосконалену версію мови C, яку він спочатку назвав «C з класами» [8]. Пізніше вона перетворилася на C++ [8].

Хоча C++ спочатку була розроблена для кодування операційних систем, зараз це універсальна, кросплатформна мова [8]. Вона використовується для написання коду в найрізноманітніших контекстах, підтримує об'єктно-орієнтоване, процедурне та загальне програмування, і є настільки поширеною, що її часто називають «швейцарським кишеньковим ножом» серед мов кодування [8].

З технічної точки зору, C++ — це скомпільована мова програмування, яка дозволяє користувачеві організувати дані та код у зрозумілу для нього програму [8]. Потім код перекладається платформою кодування або компілятором в інструкції, які може виконувати комп'ютер, що робить цю мову дуже швидкою та ефективною у виконанні [8].

Переваги [8]:

- це портативна мова програмування, тому її можна використовувати для запуску однієї і тієї ж програми на різних операційних системах;
- є багатопарадигмальною, тому дає програмістам основу, в якій вони можуть кодувати в різних стилях;
- є об'єктно-орієнтованою мовою, що полегшує сортування даних і означає, що код можна використовувати повторно;
- оскільки вона тісно пов'язана з машинними мовами, C++ полегшує низькорівневі маніпуляції;
- є досить сумісною з мовою програмування C;
- програміст керує пам'яттю при використанні C++, що дає більше контролю;
- це мова з великою масштабованістю;
- існує велика спільнота користувачів C++, а це означає, що існує велика підтримка та багато безкоштовних підручників і посібників.

Недоліки [8]:

- використання вказівників займає багато пам'яті і може бути дуже складним для розуміння;
- використання об'єктно-орієнтованої мови пов'язане з деякими проблемами безпеки;
- відсутня функція збирача сміття для автоматичної фільтрації непотрібних даних;
- не підтримує вбудовані потоки коду;

- коли дані вказують на одне і те ж місце з двох різних початкових точок, програма має тенденцію плутати код;
- система об'єктної орієнтації вважається досить базовою у порівнянні з іншими мовами;
- синтаксис дуже складний, а стандартна бібліотека невелика.

2.1.6 Python

Мова програмування, відома як Python, була розроблена у 1991 році голландським програмістом Гвідо ван Россумом [8]. Це універсальна, кросплатформна, об'єктно-орієнтована мова програмування, яка є високорівневою, інтерактивною та має динамічну семантику [8]. Дві її найбільш помітні особливості – динамічна типізація та динамічне зв'язування – роблять її дуже популярним вибором [8].

У своїй базовій формі Python використовується як мова сценаріїв та автоматизації [8]. Крім того, що вона використовується для розробки складних застосунків, це також мова, яка використовується для «склеювання» існуючих компонентів у коді або програмному забезпеченні разом [8]. Особливістю мови є те, що її синтаксис дуже простий для читання, що полегшує її вивчення і спрощує обслуговування та налагодження [8].

Переваги [8]:

- підвищена продуктивність під час кодування;
- найпростіша мова програмування;
- вона динамічно типізується і автоматично присвоює тип даних під час виконання;
- це інтерпретована мова програмування, тому код виконується рядок за рядком, і якщо виявляється проблема, то вся програма зупиняється і повідомляє, що сталася помилка;
- є дуже портативною мовою і може бути легко запущена на різних платформах;

– стандартна бібліотека Python є великою.

Недоліки [8]:

– оскільки вона виконує свій код рядок за рядком, то, як правило, працює досить повільно;

– зазвичай використовується як мова програмування на стороні сервера. Вона не дуже ефективна в мобільних обчисленнях і браузерах, оскільки працює досить повільно;

– код на Python потрібно ретельно тестувати, перш ніж використовувати його належним чином, оскільки помилки під час виконання є поширеним явищем, а типи даних змінних можуть змінитися в будь-який момент.

2.2 Порівняння мов програмування

На основі вищезазначеної інформації було проведено порівняльний аналіз мов програмування для розробки мобільних застосунків, результат якого наведено в таблиці 2.1.

Таблиця 2.1 — Порівняльна характеристика мов програмування

Критерії порівняння	Java	Kotlin	Swift	C#
Зрозумілість синтаксису	—	+	+	+
Підтримка спільноти	+	+	+	+
Простота вивчення	—	+	+	+
Підтримка IDE	+	+	+	+
Бібліотеки сторонніх розробників	+	+	+	+
Підтримка багатопоточності	+	+	+	+
Підвищена продуктивність	+	+	+	+
Розробка під Android	+	+	—	+
Розробка під iOS	—	—	+	+
Кросплатформна розробка	—	+	—	+
Інструментарій та екосистема	+	+	+	+

За результатами порівняльної таблиці 2.1 можна зробити висновок, що мова програмування C# разом з фреймворком .NET MAUI є найкращим

варіантом для кросплатформної розробки мобільних застосунків. C# є потужною мовою з багатим набором інструментів і бібліотек для розробки на різних платформах, включаючи Android та iOS. .NET MAUI надає зручні інструменти для створення кросплатформних інтерфейсів користувача, що дозволяє ефективно використовувати спільний код для різних платформ. Такий підхід дозволяє зберегти час та зусилля при розробці мобільних застосунків і забезпечує їхню швидку та ефективну роботу на різних пристроях. Отже, для розробки мобільного застосунку було обрано мову програмування C# з фреймворком .NET MAUI.

Під час аналізу мов програмування для розробки серверної частини програми було створено порівняльну характеристику, наведену в таблиці 2.2.

Таблиця 2.2 — Порівняльна характеристика мов програмування

Критерії порівняння	Java	C#	C++	Python
Зрозумілість синтаксису	—	+	—	+
Підтримка спільноти	+	+	+	+
Простота вивчення	—	+	—	+
Підтримка IDE	+	+	+	+
Бібліотеки сторонніх розробників	+	+	+	+
Підтримка багатопоточності	+	+	+	+
Підвищена продуктивність	+	+	+	±
Вебфреймворки	+	+	—	+
Бібліотеки машинного навчання	—	+	+	+
Кросплатформна розробка	+	+	+	+
Інструментарій та екосистема	+	+	+	+

За результатами порівняльної таблиці 2.2 видно, що C# та Python мають найбільше переваг серед інших мов, проте в контексті веброботки та розпізнавання Python є найкращим вибором для реалізації серверної частини. З використанням фреймворку Flask, Python надає простий та ефективний спосіб створення серверної частини мобільного застосунку. Крім того, завдяки бібліотекам Tensorflow, Keras та PyCryptodome, Python є потужним інструментом для розробки систем шифрування та розпізнавання образів. Отже, для розробки серверної частини програми було обрано мову

програмування Python з використанням Flask, Tensorflow, Keras та PyCryptodome.

2.3 Вибір середовища розробки

При розробці мобільного застосунку та серверної частини вибір правильного середовища розробки відіграє важливу роль. Існує кілька популярних середовищ, які надають розробникам необхідні інструменти та можливості для створення потужних програмних продуктів. Серед таких середовищ можна виділити Android Studio, Xcode, Eclipse та Visual Studio [9].

2.3.1 Android Studio

Android Studio — це офіційне IDE від Google для розробки застосунків для Android [10]. Це потужний інструмент, який пропонує широкий спектр функцій, що допомагають розробникам створювати високоякісні програми [10].

Android Studio створена на базі платформи IntelliJ IDEA і включає в себе різноманітні інструменти для розробки, тестування та налагодження Android-застосунків [10]. Вона також включає в себе емулятор Android, який дозволяє тестувати програми на різних пристроях і з різними розмірами екранів [10].

Переваги [10]:

- універсальне середовище розробки з редактором коду, візуальним редактором макетів та інструментами налагодження;
- платформа з відкритим вихідним кодом;
- велика база користувачів;
- інтеграція з сервісами Google;
- економічна ефективність.

Недоліки [10]:

- фрагментація;
- крута крива навчання;
- пристрої на Android є більш вразливими до загроз безпеки, а це означає, що розробники повинні вживати додаткових заходів для забезпечення безпеки застосунків.

2.3.2 Xcode

Xcode — це офіційне середовище розробки від Apple для розробки застосунків для iOS [10]. Це комплексний інструмент, який включає в себе все, що потрібно розробникам для створення високоякісних застосунків для пристроїв на iOS [10]. Xcode включає в себе редактор коду, дизайнер графічного інтерфейсу, налагоджувач та симулятор iOS [10]. Він також включає в себе широкий спектр інструментів для тестування, профілювання та оптимізації застосунків [10].

Переваги [10]:

- Xcode надає комплексне середовище розробки з редактором коду, візуальним редактором макетів та інструментами налагодження;
- тісна інтеграція з сервісами Apple;
- застосунки для iOS відомі своїм високоякісним користувацьким досвідом, що може допомогти програмам виділитися на ринку;
- пристрої iOS менш вразливі до загроз безпеки, а це означає, що застосунки, розроблені на Xcode, є більш безпечними;
- Xcode пропонує спрощений процес розробки застосунків завдяки таким функціям, як Interface Builder, який дозволяє розробникам створювати користувацькі інтерфейси без написання коду.

Недоліки [10]:

- iOS має меншу частку ринку, ніж Android, що означає меншу аудиторію для застосунків для iOS;
- обмежені можливості апаратного забезпечення;

– iOS — це закрита платформа, що означає, що розробники мають обмежений доступ до вихідного коду і можуть зіткнутися з обмеженнями під час розробки застосунків.

2.3.3 Eclipse

Eclipse — одне з найпопулярніших середовищ розробки. Це кросплатформний інструмент з потужним користувацьким інтерфейсом, який підтримує перетягування [9]. IDE також має ряд важливих функцій, таких як інструменти статичного аналізу, налагодження та профілювання [9]. Eclipse орієнтована на корпоративну розробку і дозволяє розробникам легко працювати над масштабованим програмним забезпеченням з відкритим вихідним кодом [9].

Хоча Eclipse найкраще асоціюється з Java, вона також підтримує декілька мов програмування [9]. Крім того, користувачі можуть додавати до IDE власні плагіни для підтримки проєктів з розробки програмного забезпечення [9].

Переваги [11]:

- безкоштовний та з відкритим вихідним кодом;
- кросплатформність;
- багато плагінів для інших мов, таких як Java, та додаткова функціональність, наприклад, статичний аналіз коду.

Недоліки [11]:

- може вимагати багато ресурсів (як процесора, так і оперативної пам'яті) на великих проєктах;
- не дуже стабільний – на великих проєктах може час від часу виходити з ладу;
- дещо застарілий інтерфейс.

2.3.4 Visual Studio

Visual Studio IDE — це інтегрований інтерфейс розробки Microsoft, створений, щоб допомогти розробникам програмного забезпечення [9]. IDE використовує функції штучного інтелекту, щоб вчитися на основі редагування коду, що полегшує автоматичне завершення рядків коду [9].

Однією з головних переваг Visual Studio, яку цінують багато розробників, є те, що вона сприяє спільній розробці між командами в режимі реального часу [9]. Ця функція дуже важлива, особливо під час процесу налагодження. IDE також дозволяє користувачам спільно використовувати сервери, коментарі та термінали [9].

Visual Studio має можливість підтримувати розробку мобільних застосунків, вебзастосунків та ігор. Вона також підтримує мови Python, Node.js, ASP.NET та Azure. За допомогою Visual Studio розробники можуть легко створити середовище розробки в хмарі [9].

Завдяки підтримці багатьох мов Visual Studio має функції, які бездоганно інтегруються з фреймворками Django та Flask [9]. Її можна використовувати як IDE для Python на ОС Mac, Windows та Linux [9].

Переваги [11]:

- розумний редактор, який надзвичайно корисний;
- швидке налагодження;
- потужні функції тестування.

Недоліки [11]:

- продуктивність може бути повільною на старому обладнанні;
- має велику кількість функцій та інструментів, що може бути складним для новачків.

2.4 Порівняння середовищ розробки

На основі вищезазначеної інформації було проведено порівняльний аналіз середовищ розробки, результат якого наведено в таблиці 2.3.

Таблиця 2.3 — Порівняльна характеристика середовищ розробки

Критерії порівняння	Android Studio	Xcode	Eclipse	Visual Studio
Кросплатформність	—	—	+	+
Автозавершення коду	+	+	+	+
Підтримка репозиторіїв	+	+	+	+
Простота встановлення	+	+	—	+
Перевірка та аналіз коду	+	+	+	+
Швидкий та безпечний рефакторинг коду	+	+	+	+
Наявність зручного користувацького інтерфейсу	+	+	—	+
Швидкий та зручний пошук	+	+	+	+
Підсвічування синтаксису	+	+	+	+
Підтримка пакетів	+	+	+	+
Підтримка C#	—	—	+	+
Підтримка Python	—	—	+	+

За результатами порівняльної таблиці 2.3 можна зробити висновок, що середовище Visual Studio є найкращим варіантом для розробки програмного забезпечення. Visual Studio надає широкі можливості для розробки як мовою C#, так і мовою Python. За допомогою розширень, таких як Python Tools for Visual Studio, можна зручно писати, відлагоджувати та виконувати код на обох мовах в єдиному робочому середовищі. Visual Studio також відомий своїми зручними інструментами для рефакторингу коду, аналізу та перевірки коду, що сприяє покращенню якості програмного забезпечення. Крім того, він надає вбудований Android Emulator, що дозволяє тестувати та налагоджувати розроблені застосунки для платформи Android безпосередньо в середовищі розробки. Отже, для розробки програмного забезпечення було обрано Visual Studio.

2.5 Вибір системи управління базами даних

При розробці мобільних застосунків одним з ключових аспектів є вибір системи управління базами даних (СУБД), яка забезпечить ефективне та надійне зберігання даних. Серед розповсюджених СУБД, які використовуються в мобільній розробці, варто відзначити MySQL, PostgreSQL, MongoDB та SQLite [12].

2.5.1 MySQL

MySQL — одна з найпопулярніших систем реляційних баз даних [12]. Спочатку це було рішення з відкритим вихідним кодом, зараз MySQL належить Oracle [12]. Сьогодні MySQL є основою прикладного програмного забезпечення LAMP (Linux, Apache, MySQL та Perl/PHP/Python) [12]. Маючи C та C++ під капотом, MySQL добре працює з такими системними платформами, як Windows, Linux, MacOS та іншими [12].

Переваги [12]:

- редакцію MySQL для спільноти можна завантажити безкоштовно;
- простий синтаксис і помірна складність;
- сумісність з хмарами.

Недоліки [12]:

- проблеми з масштабуванням;
- часткова підтримка відкритого коду. Хоча MySQL має частину з відкритим вихідним кодом, здебільшого працює під ліцензією Oracle [12]. Це обмежує спільноту MySQL з точки зору покращення СУБД [12];
- MySQL не підтримує деякі стандартні можливості SQL [12]. З іншого боку, MySQL має деякі розширення та окремі функції, які не відповідають стандартам SQL [12].

2.5.2 PostgreSQL

Система управління базами даних PostgreSQL ділить свою популярність з MySQL [12]. Це об'єктно-реляційна СУБД, де об'єкти, що визначаються користувачем, і табличний підхід поєднуються для побудови більш складних структур даних [12]. Вона спрямована на посилення стандартів відповідності та розширюваності [12]. Отже, вона може обробляти будь-яке робоче навантаження, як для одномашинних продуктів, так і для складних застосунків [12]. Належить і розробляється PostgreSQL Global Development Group, але все ще залишається повністю відкритим кодом [12]. Ця СУБД доступна для використання з такими платформами, як Microsoft, iOS, Android та багатьма іншими [12].

Переваги [12]:

- чудова масштабованість;
- підтримка користувацьких типів даних;
- легко інтегровані сторонні інструменти;
- відкритий вихідний код та підтримка спільноти.

Недоліки [12]:

- суперечлива документація;
- відсутність інструментів звітності та аудиту.

2.5.3 MongoDB

MongoDB — це безкоштовна нереляційна СУБД з відкритим вихідним кодом, яка також має комерційну версію [12]. Хоча MongoDB спочатку не призначалася для обробки структурованих даних, її можна використовувати для застосунків, які використовують як структуровані, так і неструктуровані дані [12]. У MongoDB бази даних підключаються до застосунків за допомогою драйверів баз даних [12]. Кілька типів даних обробляються одночасно і використовують для цього внутрішній кеш [12].

Переваги [12]:

- простий доступ до даних, їхнє зберігання, введення та пошук;
- легка сумісність з іншими моделями даних;
- горизонтально масштабоване рішення.

Недоліки [12]:

- значне споживання пам'яті;
- не має достатнього рівня безпеки даних;
- складний процес інтерпретації на інші мови запитів.

2.5.4 SQLite

SQLite — це самодостатня, безсерверна система керування базами даних, яка не потребує конфігурації [12]. Часто використовується як вбудована база даних, популярна для невеликих мобільних і настільних застосунків [12].

Переваги [12]:

- невеликий розмір і легка портативність;
- мінімальне споживання ресурсів;
- надійна та зручна у використанні.

Недоліки [12]:

- використовує блокування на основі файлів, що обмежує його здатність керувати кількома одночасними операціями запису;
- відсутність розширених можливостей. У SQLite відсутні деякі складні функції, які є в інших системах керування базами даних, такі як збережені процедури, тригери або функції, визначені користувачем [12];
- обмежена масштабованість.

2.6 Порівняння систем управління базами даних

На основі вищезазначеної інформації було проведено порівняльний аналіз систем управління базами даних, результат якого наведено в табл. 2.4.

Таблиця 2.4 — Порівняльна характеристика СУБД

Критерії порівняння	MySQL	PostgreSQL	MongoDB	SQLite
Функціональність	+	+	+	+
Швидкодія	+	+	+	+
Масштабованість	+	+	+	—
Надійність	+	+	—	+
Зручність використання	+	+	—	+
Підтримка SQL	+	+	—	+
Вбудованість	—	—	—	+
Сумісність з мобільними платформами	+	+	+	+
Відкритість та спільнота	+	+	+	+

За результатами порівняльної таблиці 2.4 можна зробити висновок, що SQLite є найкращим варіантом для мобільного застосунку з точки зору локального зберігання даних. Оскільки SQLite є вбудованою СУБД, вона легка у використанні і не вимагає окремого сервера. Вона має низькі вимоги до ресурсів, що робить її ідеальним варіантом для мобільних пристроїв. SQLite також підтримує транзакції, індексацію та кешування даних, що сприяє швидкодії та ефективності роботи з базою даних. Отже, для мобільного застосунку було обрано СУБД SQLite.

2.7 Структурна схема програми

Розроблюване програмне забезпечення складається з двох компонентів: мобільний застосунок та серверна частина. Структуру цих компонентів та їхній взаємозв'язок зображено на рисунку 2.1.

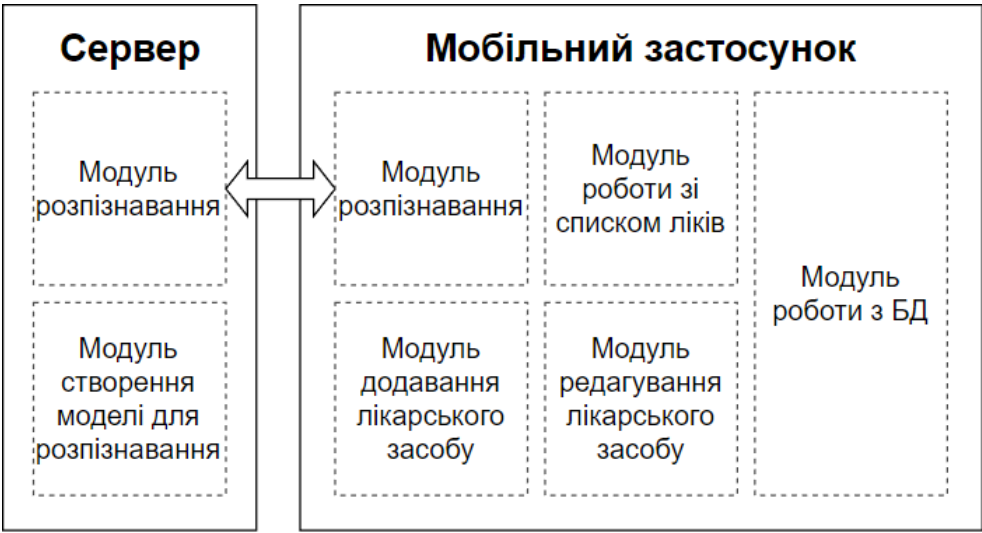


Рисунок 2.1 — Структурна схема програмного забезпечення

2.8 Опис модулів розроблюваної програми

Опис модулів, з яких буде складатися програмне забезпечення, з визначенням їхніх основних функцій наведено в таблицях 2.5-2.6.

Таблиця 2.5 — Опис модулів мобільного застосунку з визначенням основних функцій

Модуль	Опис
Модуль роботи зі списком ліків	Модуль призначений для відображення та сортування переліку ліків, видалення лікарського засобу, визначення того, чи близький термін придатності до закінчення.
Модуль додавання лікарського засобу	Модуль призначений для додавання лікарського засобу, налаштування нагадувань про приймання ліків, визначення того, чи вистачить ліків для курсу лікування.
Модуль редагування лікарського засобу	Модуль призначений для редагування інформації про лікарський засіб, налаштування нагадувань про приймання ліків.
Модуль розпізнавання	Модуль призначений для створення зображення пакування лікарського засобу за допомогою камери пристрою, його шифрування, надсилання до серверної частини, отримання результату розпізнавання, додавання лікарського засобу з використанням цього результату.
Модуль роботи з БД	Модуль призначений для створення та відкриття БД, отримання переліку ліків з неї, додавання, оновлення та видалення інформації про лікарський засіб в БД.

Таблиця 2.6 — Опис модулів серверної частини з визначенням основних функцій

Модуль	Опис
Модуль розпізнавання	Модуль призначений для отримання з мобільного застосунку HTTP-запиту, який містить зображення пакування лікарського засобу, його розшифрування, розпізнавання ліків, надсилання результату до застосунку.
Модуль створення моделі для розпізнавання	Модуль призначений для створення моделей розпізнавання за допомогою навчальної вибірки, тестування моделей, виведення статистики та збереження моделей.

2.9 Висновки за розділом 2

У цьому розділі було проведено аналіз і вибір інструментарію розробки програми, було визначено та описано структуру майбутнього програмного забезпечення. Було проведено порівняльний аналіз мов програмування та встановлено, що для розробки мобільного застосунку буде використано мову програмування C# з фреймворком .NET MAUI, а для серверної частини — мову програмування Python з використанням Flask, Tensorflow, Keras та PyCryptodome.

Було проведено порівняльний аналіз середовищ розробки, в ході якого було встановлено, що найкращим варіантом є середовище Visual Studio, тому що воно надає широкі можливості для розробки як мовою C#, так і мовою Python.

Було проведено порівняльний аналіз систем управління базами даних, в ході якого було встановлено, що найкращим варіантом є SQLite, тому що вона є вбудованою, має низькі вимоги до ресурсів, що робить її ідеальним вибором для мобільних пристроїв.

3 ОСНОВНІ РІШЕННЯ З РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Вибір шаблону проєктування

Перед тим, як перейти до розробки програмного забезпечення, було визначено, який шаблон проєктування буде використано.

Шаблон проєктування — це багато разів застосовувана архітектурна конструкція, що надає рішення для загальної проблеми проєктування в рамках конкретного контексту й описує значимість цього рішення [13].

MVVM (Model-View-ViewModel) — шаблон проєктування, що дозволяє розділити архітектуру на три функціональні частини [14]:

- модель представляє модель предметної області, яка може включати модель даних, а також бізнес-логіку та логіку перевірки. Модель взаємодіє з моделлю представлення і не знає про представлення [14];

- представлення представляє користувацький інтерфейс програми і містить обмежену, суто презентаційну логіку, яка реалізує візуальну поведінку [14]. Представлення повністю відсторонене від бізнес-логіки. Іншими словами, це клас, який ніколи не містить даних і не маніпулює ними безпосередньо [14]. Воно взаємодіє з моделлю представлення через зв'язування даних і не знає про модель;

- модель представлення є сполучною ланкою між представленням і моделлю. Вона реалізує та надає загальнодоступні властивості та команди, які використовує представлення шляхом зв'язування даних [14]. Якщо відбуваються будь-які зміни стану, модель представлення сповіщає представлення через події сповіщення [14].

Такий поділ дозволяє розподілити відповідальності та зменшити залежності між компонентами. Це полегшує тестування, підтримку та розширення програмного забезпечення. Крім того, такий підхід сприяє більш чистому та структурованому коду, що покращує розуміння програми та орієнтування в ній.

Для проєктування архітектури мобільного застосунку було використано шаблон проєктування Model-View-ViewModel. Схему архітектури наведено на рисунку 3.1.

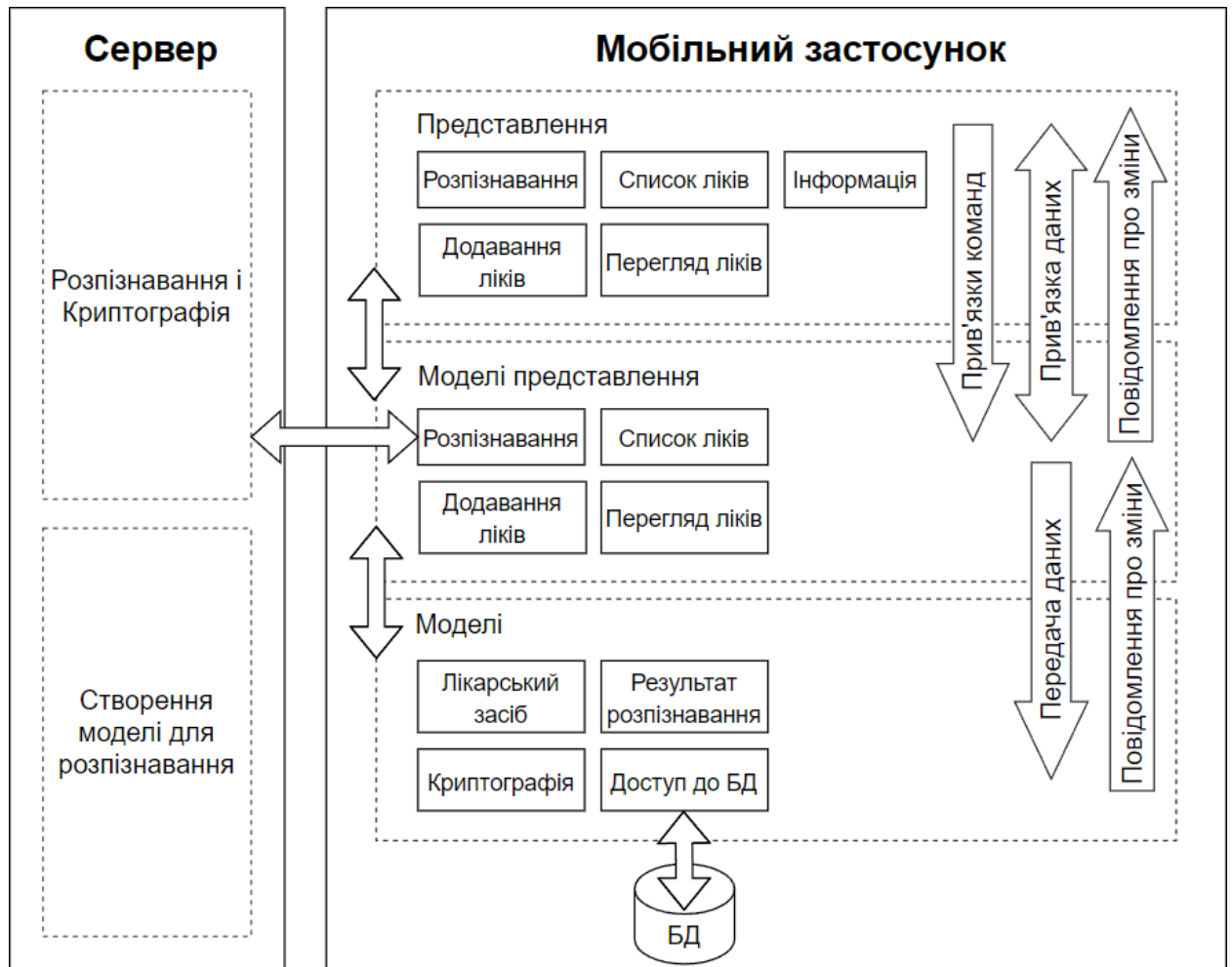


Рисунок 3.1 — Архітектура системи

При створенні архітектури передбачається, що система згодом буде еволюціонувати і що абсолютно неможливо наперед знати все, що необхідно для проєктування системи. Дизайн системи може змінюватися і допрацьовуватися в ході реалізації програми при виявленні нових відомостей і в ході тестування на відповідність вимогам. Архітектура повинна бути створена з орієнтуванням на такі зміни, щоб мати можливість адаптувати її до вимог, які на початку процесу проєктування відомі не в повному обсязі.

3.2 Опис модулів розроблюваної системи

Архітектура мобільного застосунку містить три функціональні частини: модель, представлення, модель представлення. Модулі частини, яка містить моделі, описані в таблиці 3.1. Діаграму класів, які репрезентують ці модулі, наведено на рисунку 3.2.

Таблиця 3.1 — Опис модулів частини, яка містить моделі

Модуль	Структура	Функції
Доступ до БД	class DBConstants class CommonData class DrugDatabase	Зчитування інформації з БД; Запис інформації в БД; Оновлення даних у БД; Видалення даних з БД.
Лікарський засіб	class Drug	Зберігання даних про лікарський засіб; Редагування даних.
Результат розпізнавання	class Prediction	Зберігання даних про результат розпізнавання; Редагування даних.
Криптографія	class Cryptography	Шифрування зображень.

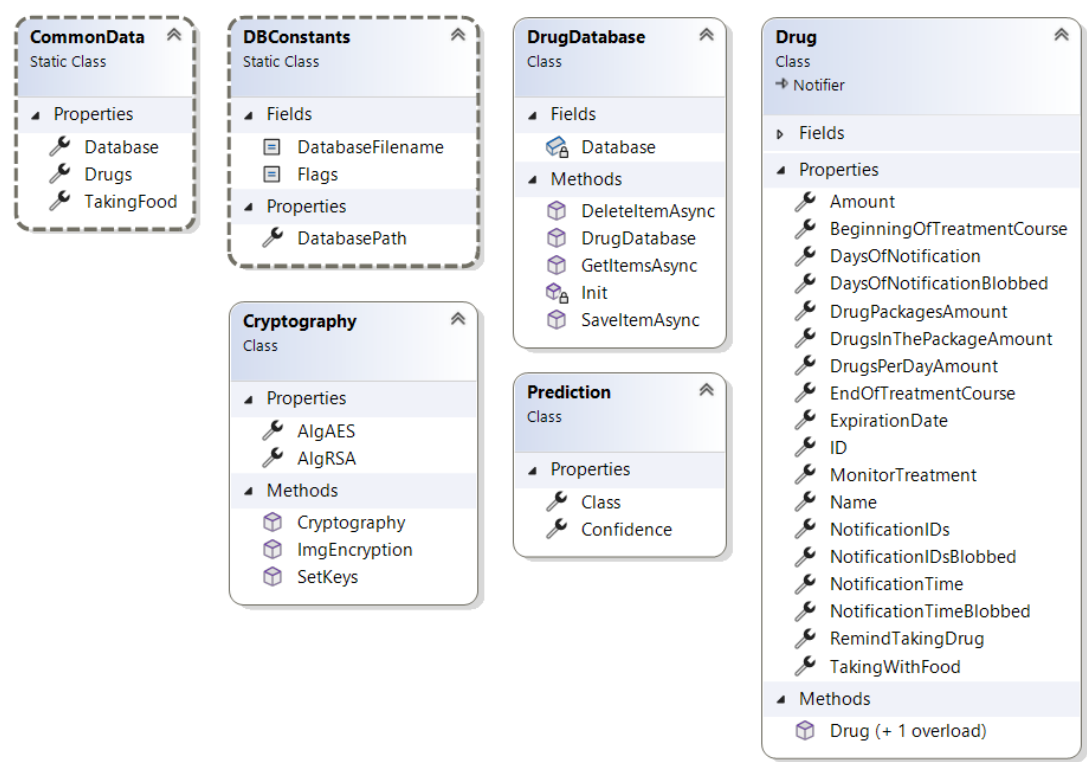


Рисунок 3.2 — Діаграма класів, які належать до моделей

Модулі частини, що містить моделі представлення, описані в табл. 3.2.
Діаграму класів, які репрезентують ці модулі, наведено на рисунку 3.3.

Таблиця 3.2 — Опис модулів частини з моделями представлення

Модуль	Структура	Функції
Список ліків	class AllDrugsViewModel	Отримання переліку ліків; Сортування списку з обраними параметрами; Видалення обраного лікарського засобу.
Додавання ліків	class AbstractViewModel class AddDrugViewModel	Отримання інформації від користувача; Налаштування сповіщень; Створення запису в БД.
Перегляд ліків	class AbstractViewModel class EditDrugViewModel	Отримання обраного лікарського засобу; Налаштування сповіщень; Оновлення інформації про лікарський засіб в БД після редагування.
Розпізнавання	class ScannerViewModel	Отримання ключів шифрування від сервера; Отримання зображення пакування лікарського засобу з камери, його шифрування та надсилання до сервера; Отримання результату розпізнавання з серверної частини.

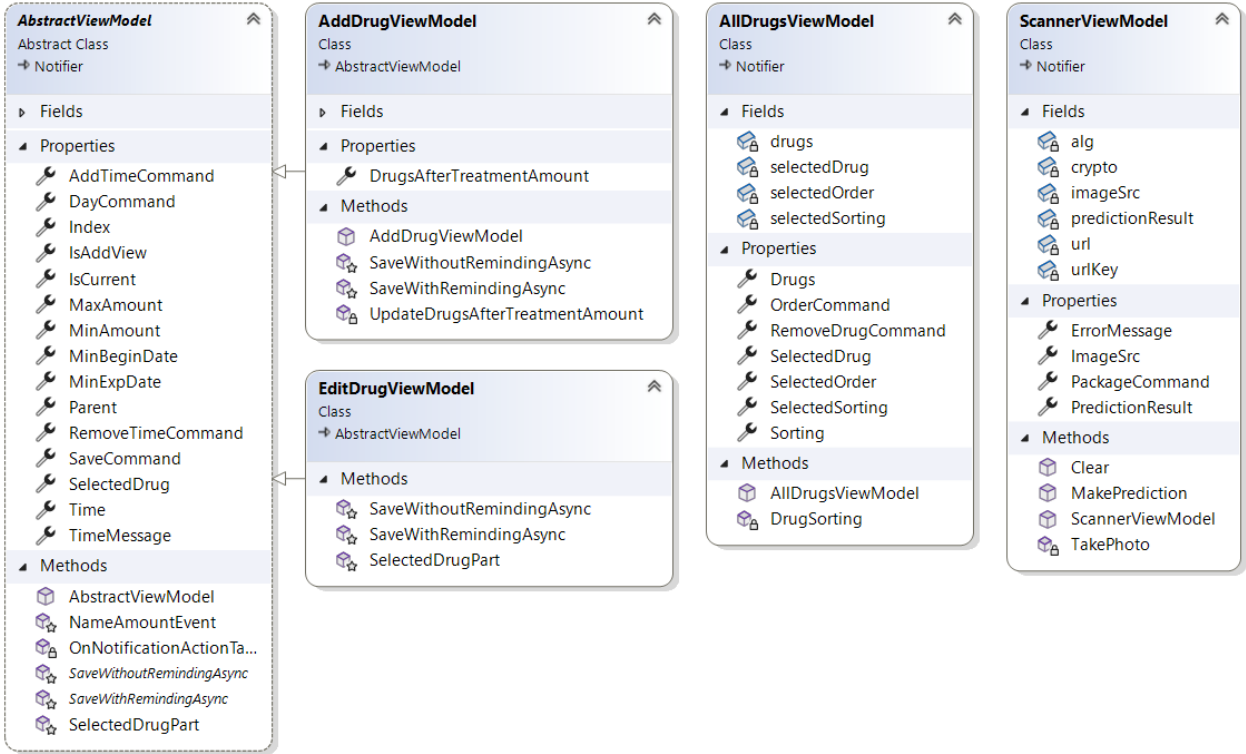


Рисунок 3.3 — Діаграма класів, які належать до моделей представлень

Модулі частини, яка містить представлення, описані в таблиці 3.3. Діаграму класів, які репрезентують ці модулі, наведено на рисунку 3.4.

Таблиця 3.3 — Опис модулів частини, яка містить представлення

Модуль	Структура	Функції
Список ліків	class AllDrugsView	Відображення переліку ліків із застосованим сортуванням; Встановлення параметрів сортування; Виділення лікарського засобу та можливість його видалення або переходу до редагування; Перехід до додавання лікарського засобу; Індикація термінів придатності.
Додавання ліків	class DrugView	Введення інформації про лікарський засіб; Відображення кількості ліків, яка залишиться після курсу лікування; Збереження інформації (додавання до переліку ліків).
Перегляд ліків	class DrugView	Відображення інформації про ліки; Редагування інформації про лікарський засіб; Збереження (оновлення) інформації.
Розпізнавання	class ScannerView	Використання камери для створення зображення пакування; Відображення зображення; Відображення результату розпізнавання; Перехід до додавання лікарського засобу з використанням результатів розпізнавання.
Інформація	class InfoView	Відображення інформації про застосунок та розробника.

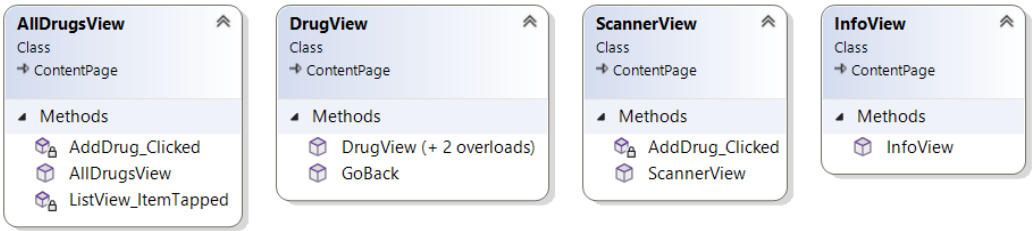


Рисунок 3.4 — Діаграма класів, які належать до представлень

Серверна частина складається з трьох модулів: криптографії, розпізнавання та створення моделі для розпізнавання. Опис цих модулів наведено в таблиці 3.4.

Таблиця 3.4 — Опис модулів серверної частини

Модуль	Структура	Функції
Створення моделі для розпізнавання	MLModel.py pretrained.py visualization.py	Попередня обробка зображень пакувань ліків; Створення структур моделей для розпізнавання; Навчання моделей на навчальній вибірці; Перевірка моделей на тестовій вибірці; Виведення статистики; Збереження створених моделей.
Розпізнавання	MLWebServer.py	Надсилення ключів шифрування до застосунку; Отримання із застосунку HTTP-запиту, який містить зображення пакування лікарського засобу; Розпізнавання ліків за допомогою створеної моделі; Надсилення результату до застосунку.
Криптографія	Cryptography.py	Розшифрування зображень.

3.3 Створення моделі для розпізнавання

Для реалізації в програмному забезпеченні функції розпізнавання лікарських засобів за зображеннями їхніх пакувань було створено модифіковану модель згорткової нейронної мережі та ще 6 моделей на основі вже відомих, а саме: InceptionV3, Xception, ResNet50V2, MobileNetV2, NASNetMobile та DenseNet169. Результати навчання цих моделей було використано для порівняння, аналізу та покращення моделей.

Для розпізнавання було обрано наступні лікарські засоби: «Флюколд-N», «Гофен 200», «Но-шпа», «Ортофен-здоров'я форте», «Фосфалюгель».

3.3.1 Модифікована модель згорткової нейронної мережі

Згорткова нейронна мережа — це алгоритм глибинного навчання, що може приймати вхідне зображення, призначати важливість (ваги та зсуви) різним об'єктам на зображенні та вміти відрізняти їх один від одного [15].

Метою операції згортки є виділення високорівневих характеристик з вхідного зображення. Мережі не повинні обмежуватися лише одним шаром згортки. Зазвичай, перший шар згортки відповідає за виділення

низькорівневих характеристик, таких як межі, колір, орієнтація градієнта і т. ін. З додаванням шарів архітектура (рис. 3.5) адаптується і до високорівневих характеристик, забезпечуючи мережу, яка має цілісне розуміння зображень з набору даних, подібно до того, як це робила б людина [15].

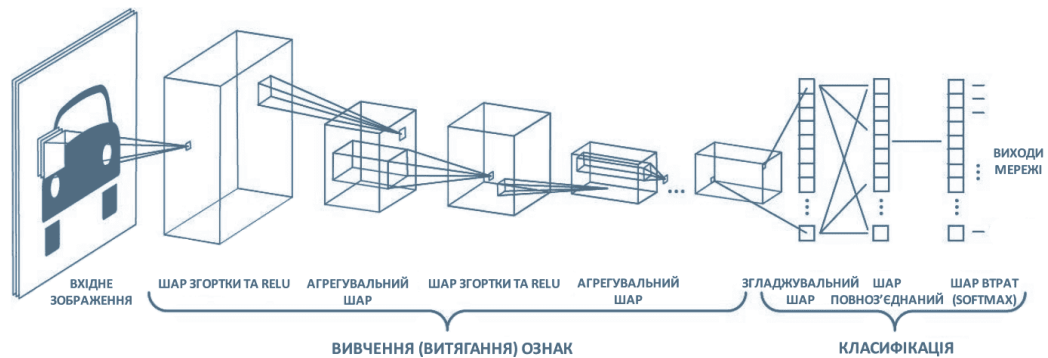


Рисунок 3.5 — Згорткова нейронна мережа [15]

Функціонування нейрона можна виразити у математичній формі за допомогою формул (3.1-3.3) [16]:

$$y = f(v), \quad (3.1)$$

$$v = b + \sum_{i=1}^n x_i w_i, \quad (3.2)$$

$$w_i \sim U \left[-\sqrt{\frac{6}{fan_{in} + fan_{out}}}, \sqrt{\frac{6}{fan_{in} + fan_{out}}} \right], \quad (3.3)$$

де y – вихідний сигнал нейрона;

f – функція активації;

b – вектор зсуву;

x_i – i -те значення вхідного сигналу;

w_i – i -те значення ваги вхідного сигналу (ініціалізація ваг Xavier/Glorot);

U – рівномірний розподіл;

fan_{in} – кількість вхідних одиниць у ваговому тензорі;

fan_{out} – кількість вихідних одиниць у ваговому тензорі.

Ініціалізатори ваг — це стратегії для встановлення початкових значень вагової матриці для шару нейронної мережі [16]. У 2010 році Глорот і Бенджіо запропонували формулу для ініціалізації ваг в залежності від кількості вхідних і вихідних одиниць (нейронів). Ініціалізатор Глорота добре працює в багатьох випадках і досі залишається популярним, тому його було використано в розробленій моделі [16].

Створена модель містить згорткові шари, шари максимізаційного агрегування, згладжування, виключення та повноз'єднані шари. Крім того, зображення нормалізуються та випадково обертаються, масштабуються, горизонтально перевертаються для збільшення різноманітності даних (у всіх створених моделях). Функцією активації було обрано ELU (Exponential Linear Unit) через те, що вона показала найкращі результати серед наявних функцій. Вихідний шар використовує функцію активації Softmax (5 класів). В повноз'єднаних шарах було використано L2-регуляризацію (заохочує менші, більш рівномірно розподілені ваги, додаючи штраф), яка покращила отриману модель. Структура цієї моделі наведена на рисунку 3.6, а результат навчання — на рисунку 3.7.

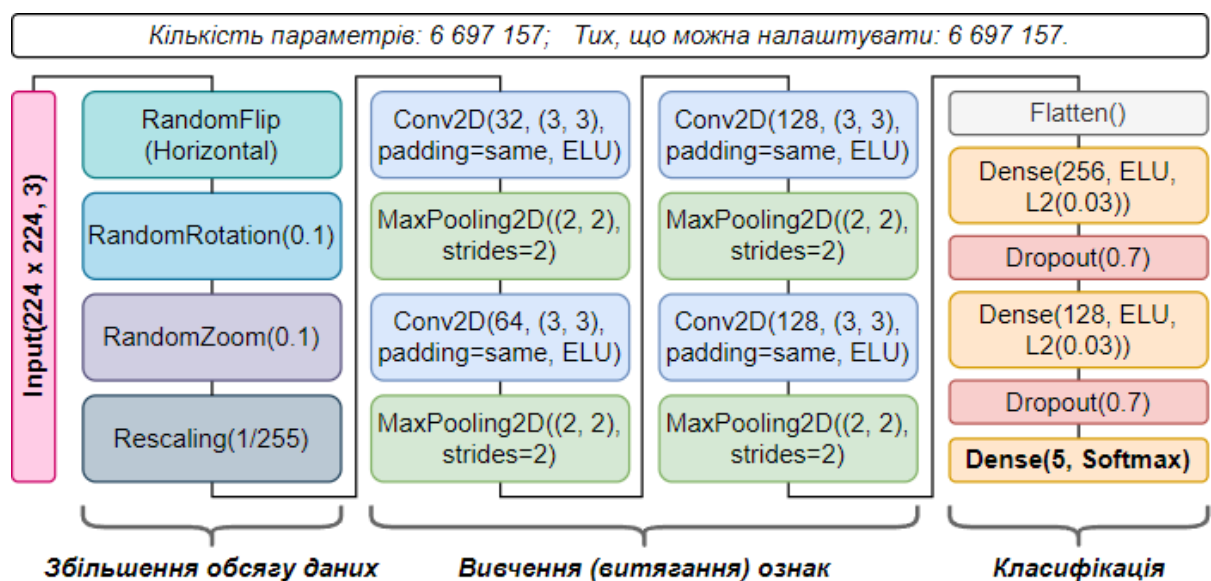


Рисунок 3.6 — Структура моделі

Own

Час навчання: 230.4; К-сть епох: 35; Втрата тестування: 3.63; Точність тестування: 0.92

I: Флюколд-N (0.99); II: Гофен 200 (0.99); III: Но-шпа (0.91);

IV: Ортофен-здоров'я форте (1.00); V: Фосфалюгель (0.98)

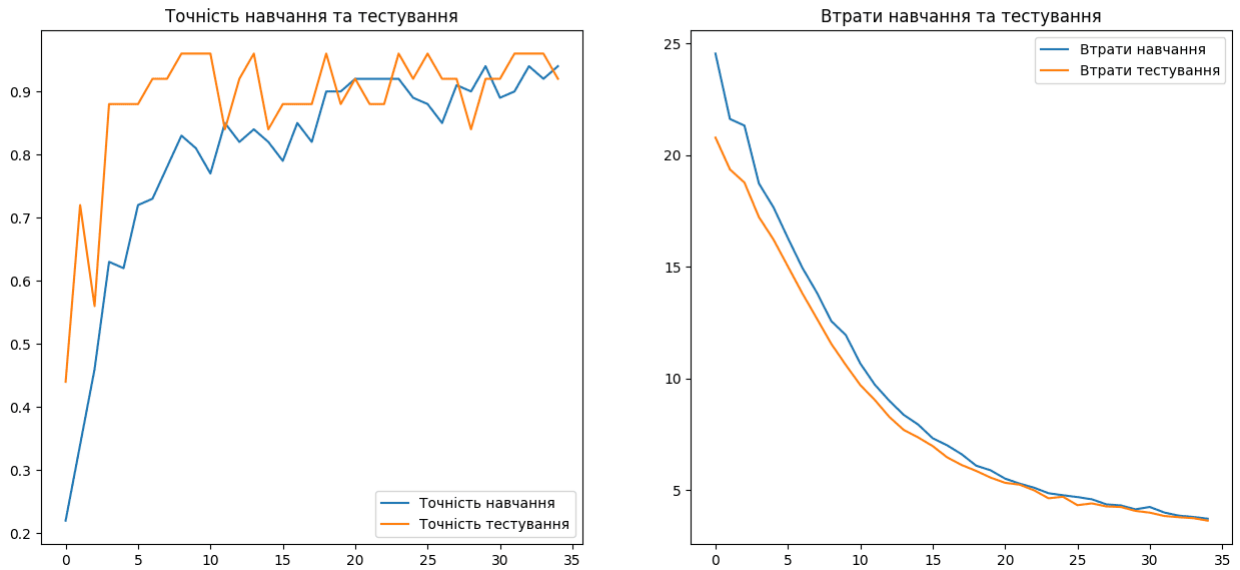


Рисунок 3.7 — Результат навчання моделі

3.3.2 Модель на основі InceptionV3

Inception V3 — це модель глибокого навчання на основі згорткових нейронних мереж, яка використовується для класифікації зображень [17]. Inception V3 є покращеною версією базової моделі Inception V1, яка була представлена як GoogLeNet у 2014 році. Як випливає з назви, вона була розроблена командою Google [17].

Inception V3 має 42 шари, її структура наведена на рисунку 3.8.

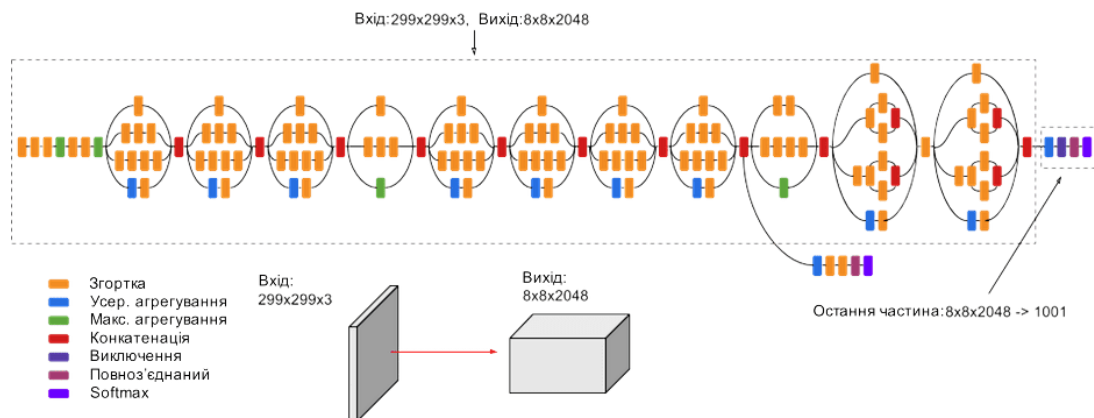


Рисунок 3.8 — Модель Inception V3 [17]

Створена модель містить Inception V3, шар глобального усереднювального агрегування та повноз'єднані шари. Структура цієї моделі наведена на рисунку 3.9, а результат навчання — на рисунку 3.10.

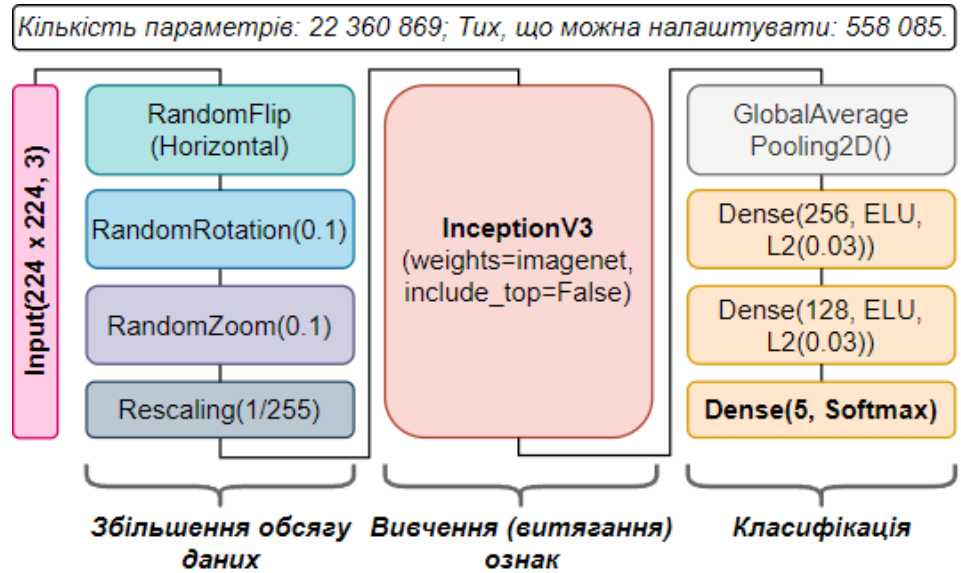


Рисунок 3.9 — Структура моделі з Inception V3

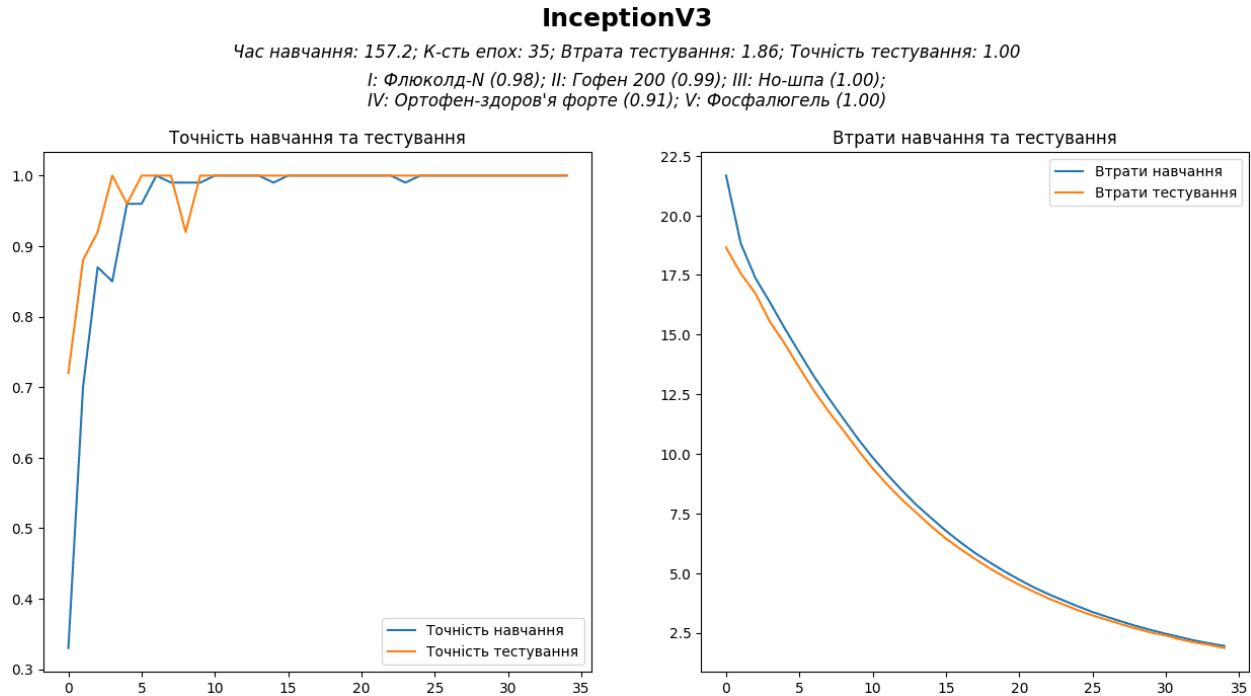


Рисунок 3.10 — Результат навчання моделі з Inception V3

3.3.3 Модель на основі Xception

Xception — це архітектура глибокої згорткової нейронної мережі, яка використовує глибокі розділені згортки (Depthwise Separable Convolutions) [18]. Її розробили дослідники Google. Google представив інтерпретацію модулів Inception в згорткових нейронних мережах як проміжний крок між звичайною згорткою і операцією глибокої розділеної згортки (глибинна згортка, за якою слідує точкова згортка) [18].

Структура Xception наведена на рисунку 3.11.

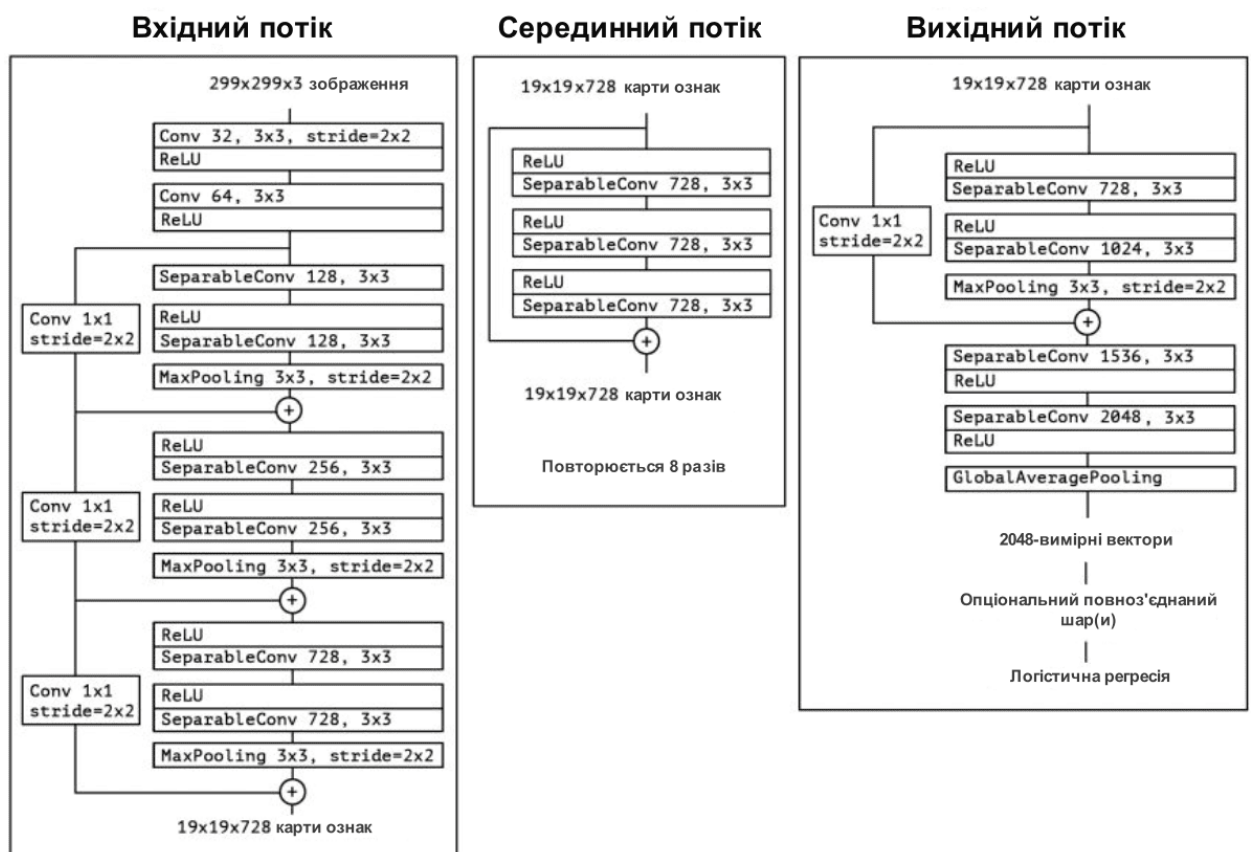


Рисунок 3.11 — Модель Xception [18]

Створена модель містить Xception, шар глобального усереднювального агрегування та повноз'єднані шари. Структура цієї моделі наведена на рисунку 3.12, а результат навчання — на рисунку 3.13.

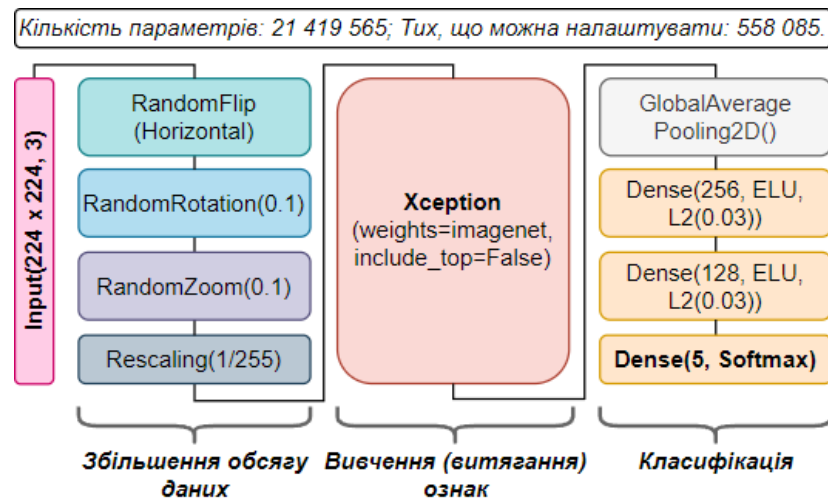


Рисунок 3.12 — Структура моделі з Xception

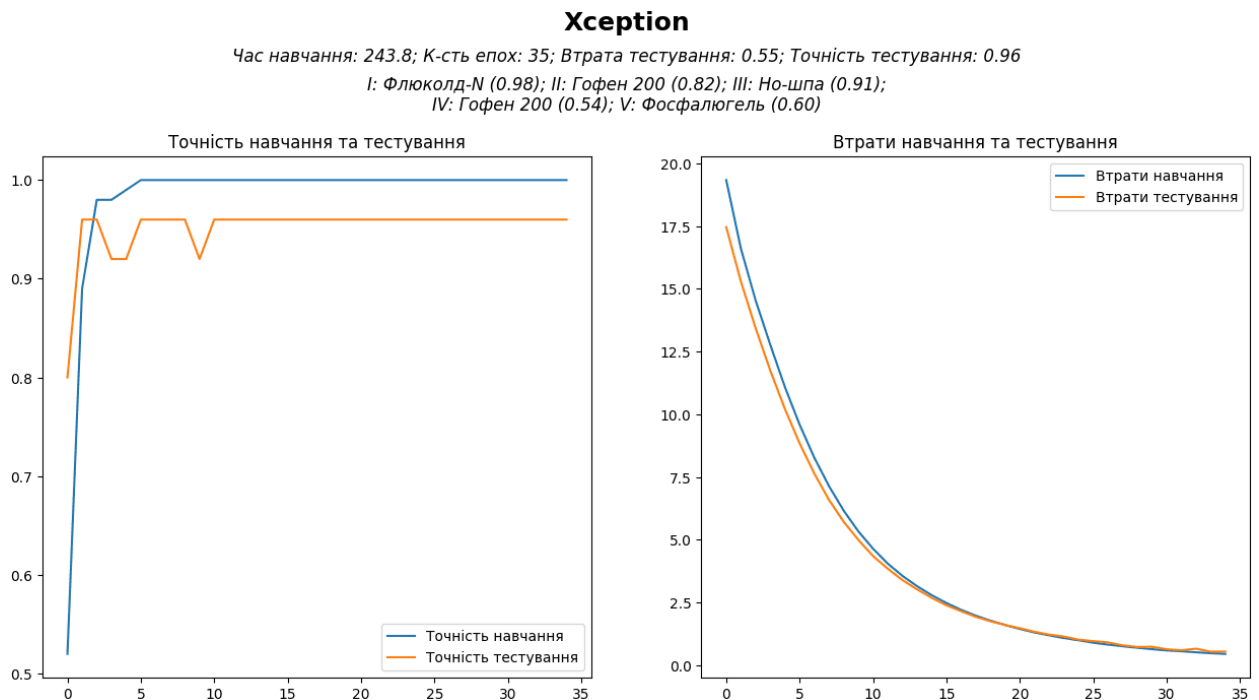


Рисунок 3.13 — Результат навчання моделі з Xception

3.3.4 Модель на основі ResNet50V2

ResNet50 — це 50-шарова згорткова нейронна мережа (48 згорткових шарів, один шар максимізаційного агрегування і один шар усереднювального агрегування) [19]. Залишкові нейронні мережі — це тип штучних нейронних мереж, які формують мережі шляхом складання залишкових блоків [19].

Початковою архітектурою ResNet була ResNet34, яка складалася з 34 шарів. Вона забезпечила новий спосіб додавання більшої кількості згорткових шарів до згорткової нейронної мережі, не стикаючись з проблемою зникання градієнтів, використовуючи концепцію пропускових з'єднань (зрізань) [19]. Пропускове з'єднання «перескакує» через деякі шари, перетворюючи звичайну мережу на залишкову [19].

Архітектура ResNet (рис. 3.14) дотримується двох основних правил проєктування. По-перше, кількість фільтрів у кожному шарі однакова і залежить від розміру вихідної карти ознак. По-друге, якщо розмір карти ознак зменшується вдвічі, вона містить вдвічі більше фільтрів, щоб зберегти часову складність кожного шару [19].

50-шарова ResNet використовує конструкцію «вузьких місць» для будівельного блоку. Залишковий блок використовує згортки 1×1 , відомі як «вузьке місце», що зменшує кількість параметрів і множень матриць [19]. Це дозволяє набагато швидше навчати кожен шар. Вона використовує стек з трьох шарів замість двох [19].

Назва шару	Розмір	18 шарів	34 шарів	50 шарів	101 шарів	152 шарів
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Рисунок 3.14 — Моделі ResNet [19]

Створена модель містить ResNet50V2, шар глобального усереднювального агрегування та повноз'єднані шари. Структура цієї моделі наведена на рисунку 3.15, а результат навчання — на рисунку 3.16.

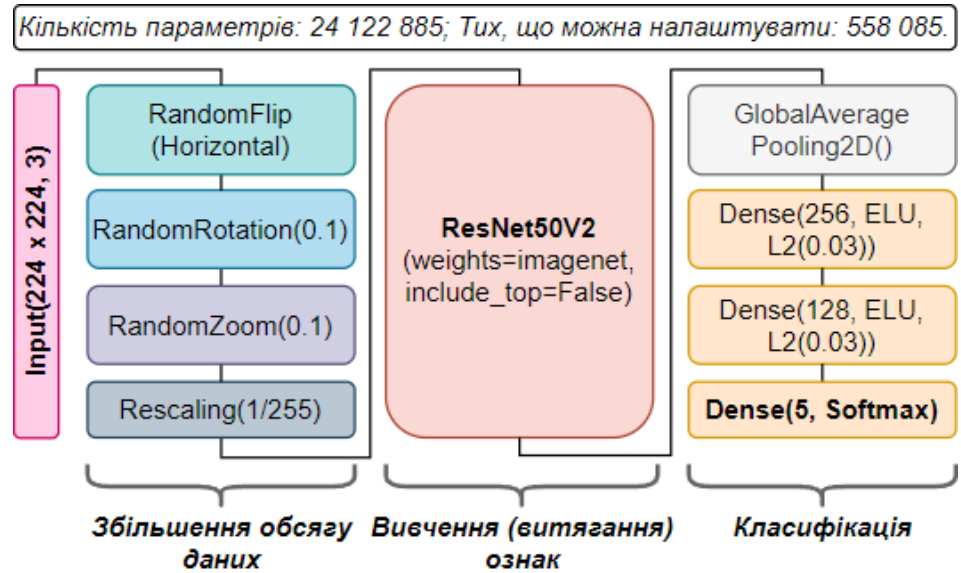


Рисунок 3.15 — Структура моделі з ResNet50V2

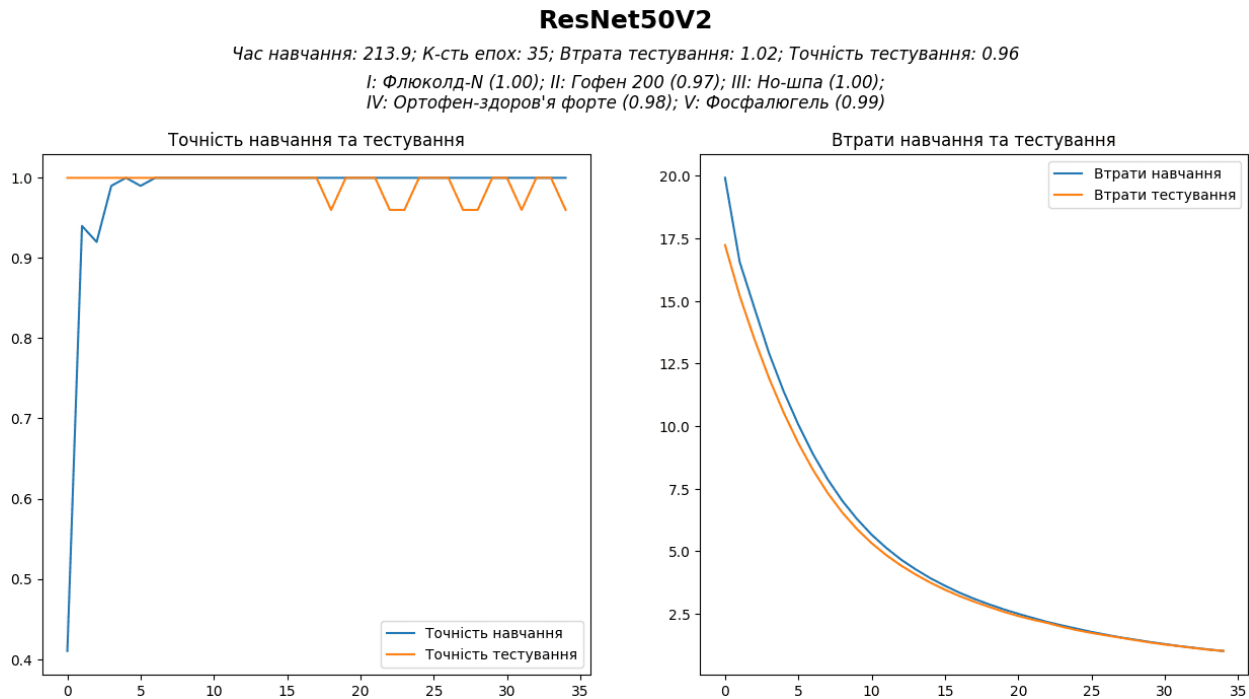


Рисунок 3.16 — Результат навчання моделі з ResNet50V2

3.3.5 Модель на основі MobileNetV2

Моделі MobileNet дуже малі та мають низьку затримку, їх можна легко розгортати на мобільних та вбудованих периферійних пристроях [20].

У першій версії автори використовували глибинну розділену згортку для скорочення обчислень. У другій версії автори ще більше скоротили час обчислень, зменшивши кількість параметрів, що навчаються, без значного падіння точності або майже без нього. Це було досягнуто завдяки новому блоку згортки під назвою «Інвертовані залишки та лінійні вузькі місця» [20]. У цьому блоці ознаки з представлення меншої розмірності масштабуються. Потім застосовується глибинна розділена згортка, після чого ознаки стискаються назад у попереднє представлення меншої розмірності [20].

MobileNetV2 починається з базового двовимірного шару згортки (рис. 3.17). Потім один за одним додається серія шарів «вузьких місць», які мають коефіцієнт розширення (t), різні кроки (s), вихідні канали (c) і кількість повторень (n) [20].

Вхід	Оператор	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	-

Рисунок 3.17 — Модель MobileNetV2 [20]

Створена модель містить MobileNetV2, шар глобального усереднювального агрегування та повноз'єднані шари. Структура цієї моделі наведена на рисунку 3.18, а результат навчання — на рисунку 3.19.

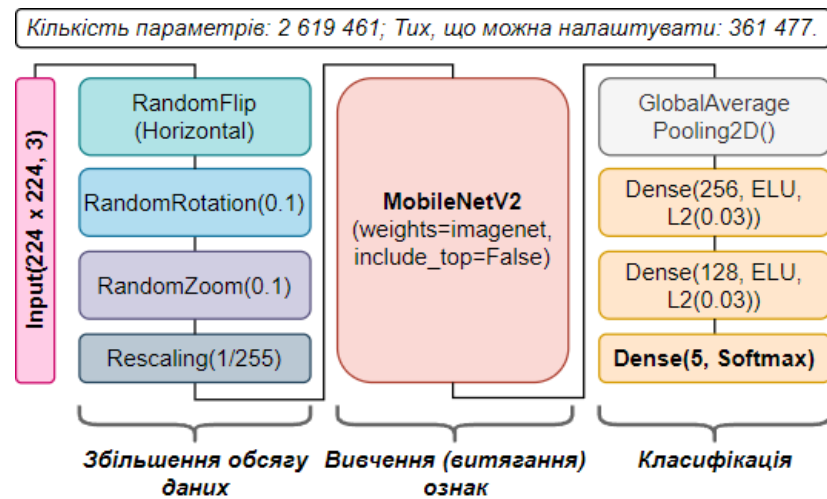


Рисунок 3.18 — Структура моделі з MobileNetV2

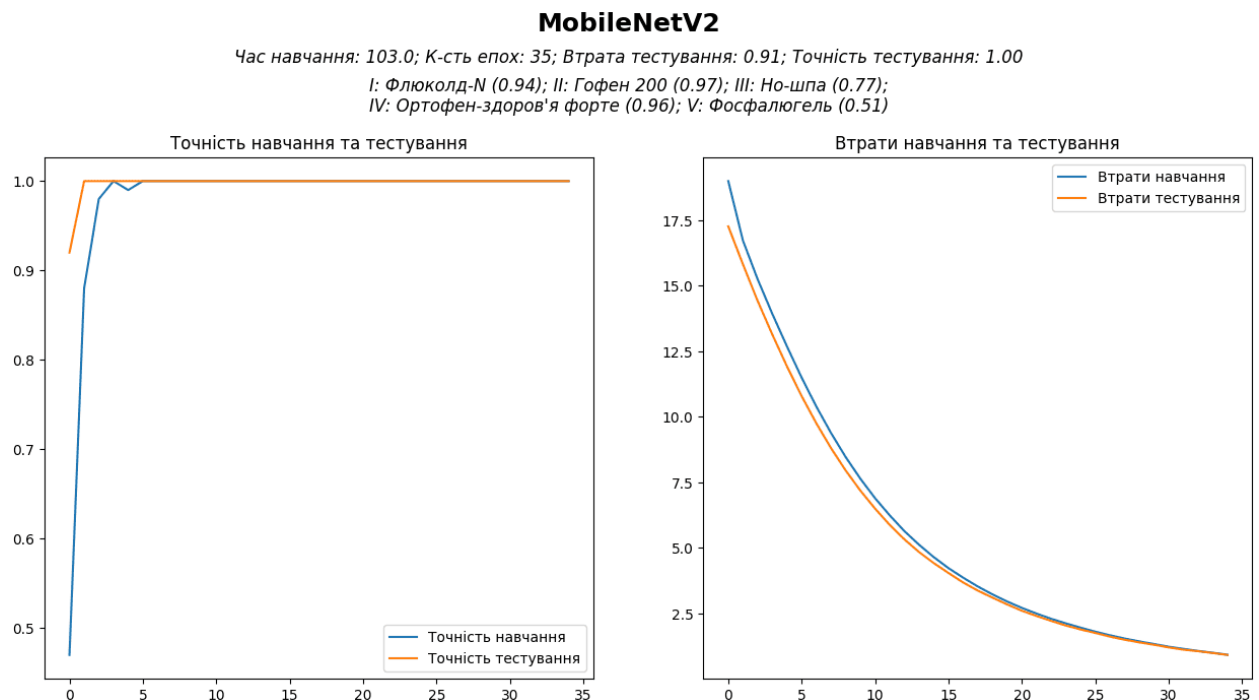


Рисунок 3.19 — Результат навчання моделі з MobileNetV2

3.3.6 Модель на основі NASNetMobile

NasNetMobile — це архітектура згорткової нейронної мережі, яка була розроблена для мінімізації обчислювальної складності та досягнення швидшого виконання на мобільних пристроях [21]. Вона вважається вдосконаленням, яке дозволить пришвидшити обробку моделі без втрати точності, що досягається за рахунок відходу від стандартного послідовного

підходу до обробки, натомість покладаючись на паралельну обробку декількох шарів з використанням архітектури блоків і кроків. Блоки — це заздалегідь визначені шари мережі («зовнішні каркаси»), а кроки — це підмножини кожного блоку, які містять один або два шари [21]. Перевага такої архітектури полягає в прийнятому алгоритмі пошуку, оскільки він дозволяє моделі додатково розбивати шари для виконання пропусків в мережі, відповідно до розміру фільтра на шар [21].

Архітектура моделі (рис. 3.20) складається з 769 шарів, включаючи вхідний шар, 371 шар згортки, 188 шарів активації та 144 шари редукції, а також включає шари глобального усереднювального агрегування та пакетну нормалізацію. Модель має різні розміри ядер для блоків, що допомагає збалансувати різноманітність шарів для прискорення алгоритму пошуку [21].

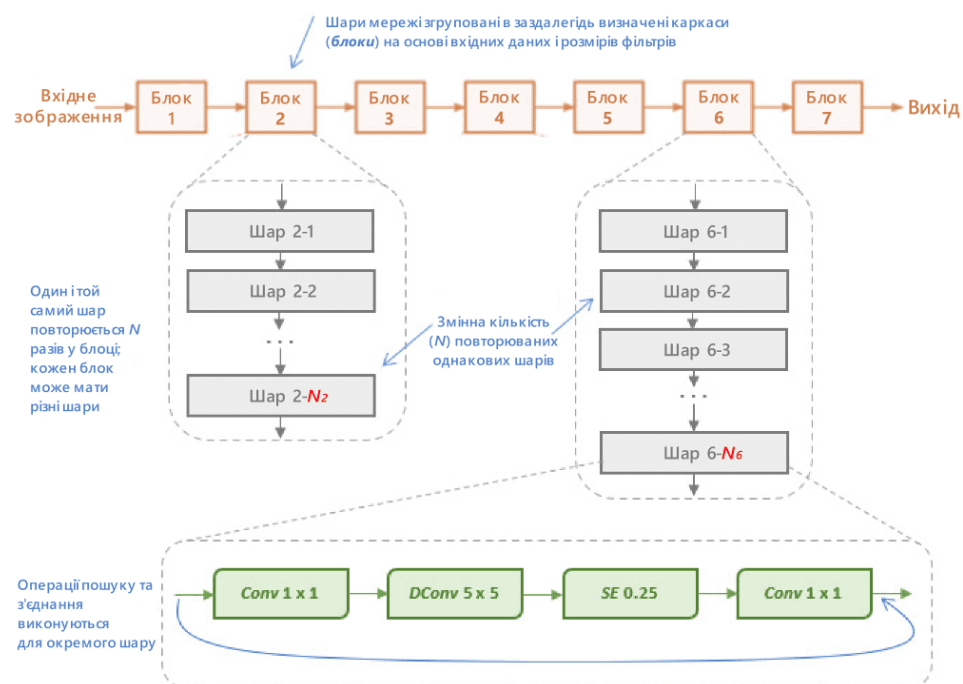


Рисунок 3.20 — Модель NasNetMobile [21]

Створена модель містить NasNetMobile, шар глобального усереднювального агрегування та повноз'єднані шари. Структура цієї моделі наведена на рисунку 3.21, а результат навчання — на рисунку 3.22.

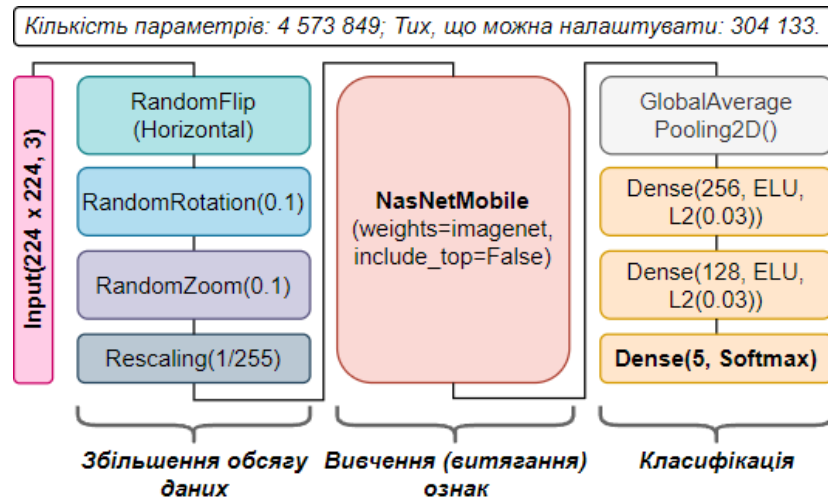


Рисунок 3.21 — Структура моделі з NasNetMobile

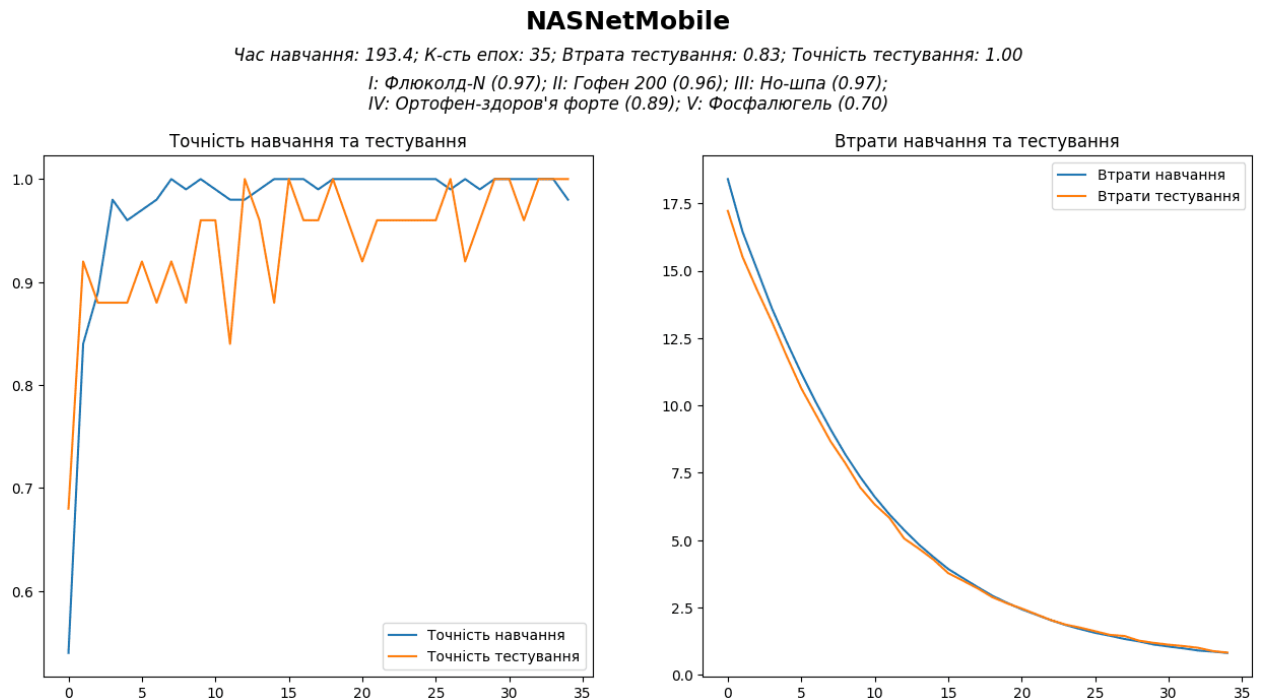


Рисунок 3.22 — Результат навчання моделі з NasNetMobile

3.3.7 Модель на основі DenseNet169

DenseNet169 — одна з архітектур сімейства DenseNet зі 169 шарами, яка широко використовується для задач класифікації [22]. Вона має набагато менше параметрів, що навчаються, порівняно з іншими архітектурами DenseNet з меншою кількістю шарів. DenseNet169 та інші архітектури DenseNet здатні подолати проблему зникання градієнтів, мають сильну

стратегію поширення ознак, мінімізують кількість параметрів, що навчаються, і заохочують повторне використання ознак, що робить їх сімейством дуже надійних архітектур глибокого навчання [22].

Архітектура (рис. 3.23) включає згорткові шари, шари максимізаційного агрегування, повноз'єднані та проміжні шари. Модель використовує функцію активації ReLU (Rectified Linear Units) для всієї архітектури, а для останнього шару — активацію Softmax [22].

Шар	Розмір ядра	Параметри	Розмір тензора
Згортка	7×7 (Conv)	Stride = 2, ReLu	112×112
Агрегування	3×3 (MaxPool)	Stride = 2	56×56
Повноз'єднаний шар 1	$1 \times 1 \times 6$ (Conv)	Dropout = 0.2	56×56
	$3 \times 3 \times 6$ (Conv)		
Проміжний шар 1	1×1 (Conv)	Stride = 2	56×56
	2×2 (AvgPool)		28×28
Повноз'єднаний шар 2	$1 \times 1 \times 12$ (Conv)	Dropout = 0.2	28×28
	$3 \times 3 \times 12$ (Conv)		
Проміжний шар 2	1×1 (Conv)	Stride = 2	28×28
	2×2 (AvgPool)		14×14
Повноз'єднаний шар 3	$1 \times 1 \times 32$ (Conv)	Dropout = 0.2	14×14
	$3 \times 3 \times 32$ (Conv)		
Проміжний шар 3	1×1 (Conv)	Stride = 2	14×14
	2×2 (AvgPool)		7×7
Повноз'єднаний шар 4	$1 \times 1 \times 32$ (Conv)	Dropout = 0.2	7×7
	$3 \times 3 \times 32$ (Conv)		
Класифікація	1×1 (Global AvgPool)		1×1
	1000D (fully-connected softmax)		

Рисунок 3.23 — Модель DenseNet169 [22]

Створена модель містить DenseNet169, шар глобального усереднювального агрегування та повноз'єднані шари. Структура цієї моделі наведена на рисунку 3.24, а результат навчання — на рисунку 3.25.

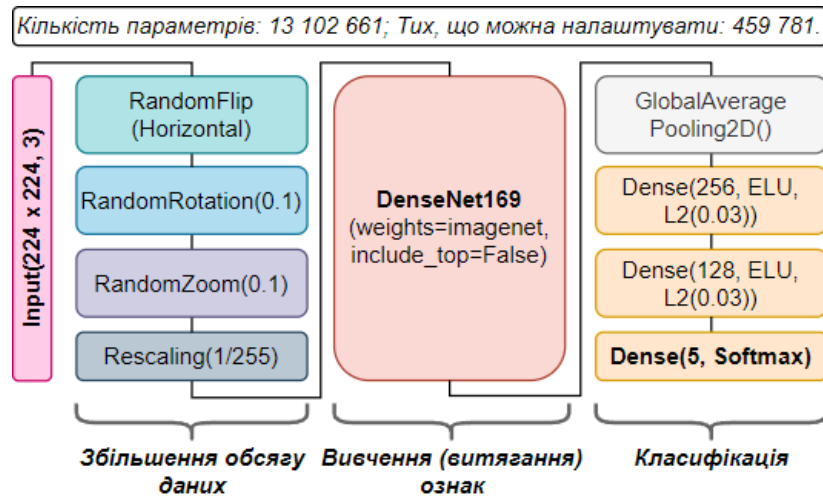


Рисунок 3.24 — Структура моделі з DenseNet169

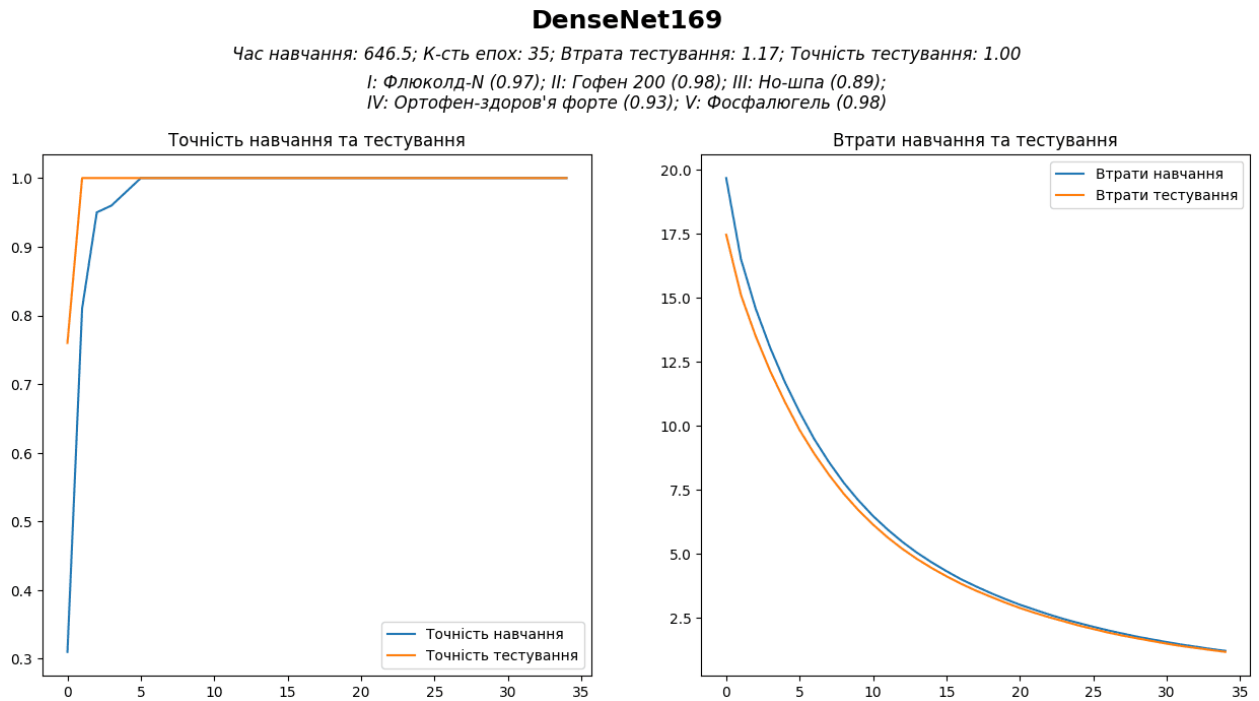


Рисунок 3.25 — Результат навчання моделі з DenseNet169

3.3.8 Порівняння моделей

Для навчальної вибірки було використано 100 зображень паковань 5 лікарських засобів (20 зображень на один лікарський засіб), а для тестової вибірки — 25 зображень паковань 5 лікарських засобів (5 зображень на один лікарський засіб). В таблиці 3.5 наведено результати навчання моделей за 35 епох. Для перевірки моделей було використано 5 зображень паковань

(рис. 3.26), які не проходили через моделі під час навчання. Кількість класів, які було правильно визначено для прикладів наведено в таблиці 3.5, а ймовірності — в таблиці 3.6.

Таблиця 3.5 — Порівняння результатів навчання моделей

Модель	Час навч. (с)	Втрата тест.	Точність тест. (%)	Розмір (Мб)	К-сть правильних класів прикладів
Own	230,4	3,63	92	78,3	5 з 5
InceptionV3	157,2	1,86	100	97,5	5 з 5
Xception	243,8	0,55	96	90,8	4 з 5
ResNet50V2	213,9	1,02	96	101	5 з 5
MobileNetV2	103,0	0,91	100	18,0	5 з 5
NASNetMobile	193,4	0,83	100	39,9	5 з 5
DenseNet169	646,5	1,17	100	67,7	5 з 5



Рисунок 3.26 — Приклади для перевірки моделей

Таблиця 3.6 — Ймовірності класифікації зображень у відсотках

Модель	Флюколд-N	Гофен 200	Но-шпа	Ортофен-здоров'я форте	Фосфалюгель
Own	99	99	91	100	98
InceptionV3	98	99	100	91	100
Xception	98	82	91	—	60
ResNet50V2	100	97	100	98	99
MobileNetV2	94	97	77	96	51
NASNetMobile	97	96	97	89	70
DenseNet169	97	98	89	93	98

З отриманих даних видно, що створена модифікована модель згорткової нейронної мережі успішно пройшла навчання на наборі даних лікарських засобів та продемонструвала здатність до точного розпізнавання

паковань лікарських засобів з високою достовірністю. Вона показала гарні результати на рівні моделей, які засновані на вже відомих моделях, тому її було використано в серверній частині програми.

3.4 Проєктування бази даних

Дані розроблюваного програмного забезпечення зберігаються в базі даних, яку було створено, та з якою відбувається взаємодія за допомогою SQLite. Концептуальну модель бази даних зображено на рисунку 3.27.

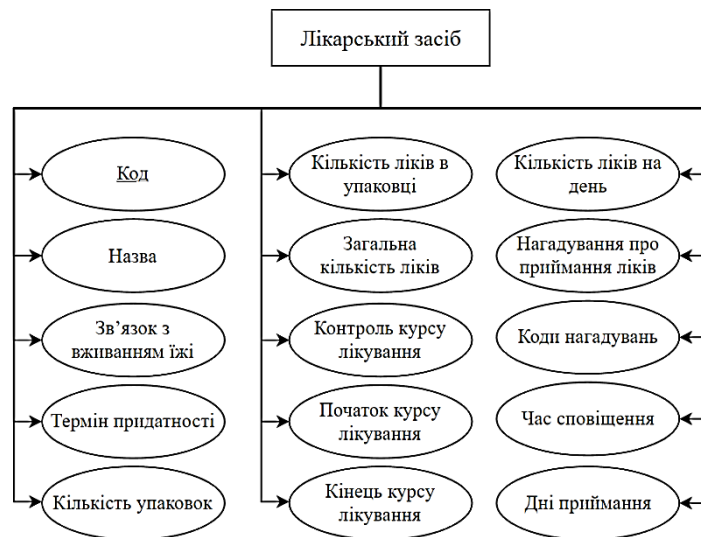


Рисунок 3.27 — Концептуальна модель бази даних

Сутність «Лікарський засіб» — це об'єкт в базі даних, який визначає основні атрибути, що характеризують лікарські засоби, призначені для лікування та профілактики захворювань. Вона включає інформацію про назву препарату, термін придатності, кількість пакувань і ліків в них, курс лікування, нагадування тощо.

Ця сутність є важливим елементом програмного забезпечення обліку лікарських засобів універсальної аптечки, яка сприяє точному та ефективному веденню обліку ліків.

Опис полів сутності та їхніх типів наведено у таблиці 3.7.

Таблиця 3.7 — Опис полів сутності «Лікарський засіб»

№	Назва поля	Тип даних	Розмір
1	ID	INTEGER	1-8 байтів
2	Name	TEXT	Залежить від довжини рядка та обмежень
3	TakingWithFood	TEXT	Залежить від довжини рядка та обмежень
4	ExpirationDate	TEXT	Залежить від довжини рядка та обмежень
5	DrugPackagesAmount	INTEGER	1-8 байтів
6	DrugsInThePackageAmount	REAL	8 байтів
7	DrugsPerDayAmount	REAL	8 байтів
8	Amount	REAL	8 байтів
9	MonitorTreatment	INTEGER	1-8 байтів
10	BeginningOfTreatment-Course	TEXT	Залежить від довжини рядка та обмежень
11	EndOfTreatmentCourse	TEXT	Залежить від довжини рядка та обмежень
12	RemindTakingDrug	INTEGER	1-8 байтів
13	NotificationIDs	TEXT	Залежить від довжини рядка та обмежень
14	NotificationTime	TEXT	Залежить від довжини рядка та обмежень
15	DaysOfNotification	TEXT	Залежить від довжини рядка та обмежень

3.5 Основні рішення щодо шифрування зображень

Для реалізації в програмному забезпеченні функції шифрування зображень було створено модулі, які виконують шифрування та розшифрування даних за допомогою симетричного алгоритму AES з режимом CFB, асиметричного алгоритму RSA та комбінації цих алгоритмів. Результати впровадження цих методів було використано для порівняння, аналізу та вибору найкращого підходу до захисту візуальної інформації.

3.5.1 AES

AES — це симетричний алгоритм, що використовує один і той самий 128-, 192- або 256-бітний ключ для шифрування та дешифрування (безпека системи AES зростає експоненціально зі збільшенням довжини) [23]. Навіть при 128-бітному ключі завдання злому AES шляхом перевірки кожного з 2^{128} можливих значень ключа (атака «грубою силою») настільки трудомістке, що

навіть найшвидшому суперкомп'ютеру знадобиться в середньому понад 100 трильйонів років, щоб зробити це [23]. Алгоритм AES послідовно застосовує низку математичних перетворень до кожного 128-бітного блоку даних. Оскільки обчислювальні вимоги цього підходу низькі, AES можна використовувати для швидкого шифрування великих обсягів даних [23].

Cipher Feedback (CFB), також відомий як зворотний зв'язок за шифротекстом, є режимом роботи блочного шифру, який перетворює його на поточковий шифр, що самосинхронізується [24]. Шифротекст — це зашифрований текст, переданий з відкритого тексту за допомогою алгоритму шифрування [24]. CFB використовує вектор ініціалізації, блоковий шифр як компонент генератора випадкових чисел. У режимі CFB попередній блок зашифрованого тексту шифрується, а вихідні дані об'єднуються з поточним блоком відкритого тексту для створення поточного блоку зашифрованого тексту [24]. Операція XOR (eXclusive OR) приховує шаблони відкритого тексту. З відкритим текстом не можна безпосередньо працювати, якщо не отримати блоки з початку або кінця зашифрованого тексту [24].

Схему шифрування та розшифрування даних за допомогою режиму CFB наведено на рисунку 3.28.

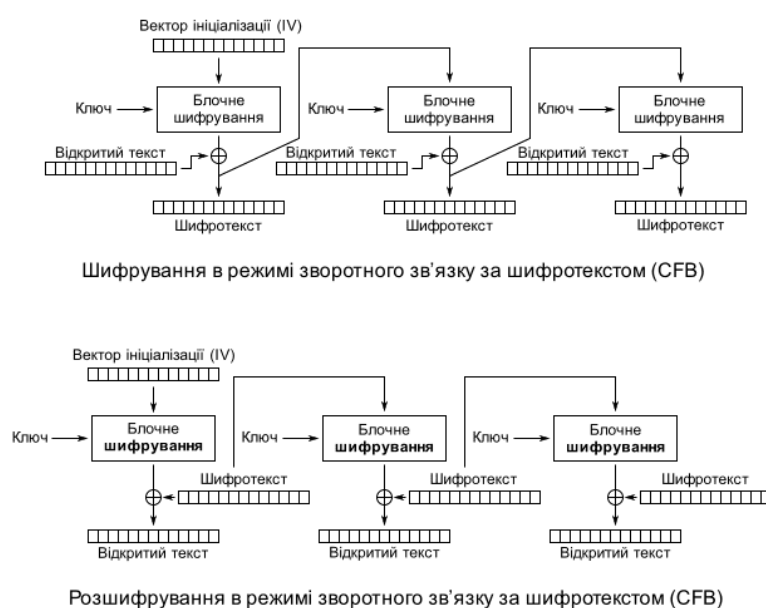


Рисунок 3.28 — Процес шифрування та розшифрування в режимі CFB [25]

Шифрування та розшифрування даних можна виразити у математичній формі за допомогою формул (3.4-3.6) [25]:

$$C_0 = IV, \quad (3.4)$$

$$C_i = E_k(C_{i-1}) \oplus P_i, \quad (3.5)$$

$$P_i = E_k(C_{i-1}) \oplus C_i, \quad (3.6)$$

де i – номер блоку;

k – ключ;

IV – вектор ініціалізації;

P_i – блок повідомлення (відкритий текст);

C_i – зашифрований блок (шифротекст);

E_k – функція, що виконує блокове шифрування.

3.5.2 RSA

RSA названо на честь вчених з МТІ (Рівеста, Шаміра та Адлемана, Массачусетський технологічний інститут), які вперше описали його в 1977 році. Це асиметричний алгоритм, який використовує публічно відомий ключ для шифрування, але вимагає іншого ключа, відомого тільки одержувачу, для розшифрування [23]. Відкритий ключ є результатом множення 2 величезних простих чисел. Тільки цей результат, довжиною 1024, 2048 або 4096 біт, оприлюднюється. Але розшифрування RSA вимагає знання двох простих множників цього добутку. Оскільки не існує відомого методу обчислення простих множників таких великих чисел, тільки творець публічного ключа може згенерувати приватний ключ, необхідний для розшифрування. RSA є більш трудомістким, ніж AES, і набагато повільнішим. Зазвичай його використовують для шифрування невеликих обсягів даних [23].

Шифрування та розшифрування даних можна виразити у математичній формі за допомогою формул (3.7-3.13) [26]:

$$n = p * q, \quad (3.7)$$

$$\varphi(n) = (p - 1)(q - 1), \quad (3.8)$$

$$1 < e \leq \varphi(n), \quad (3.9)$$

$$\text{НСД}(e, \varphi(n)) = 1, \quad (3.10)$$

$$d = e^{-1}(\text{mod } \varphi(n)), \quad (3.11)$$

$$C = M^e \text{mod } n, \quad (3.12)$$

$$M = C^d \text{mod } n, \quad (3.13)$$

де n – модуль;

p та q – великі прості числа;

$\varphi(n)$ – функція Ейлера;

e та d – відкрита й секретна експоненти;

C – зашифроване повідомлення (шифротекст);

M – повідомлення (відкритий текст).

3.5.3 Гібридне шифрування

Процес гібридного шифрування складається з чотирьох етапів: генерація симетричного ключа, шифрування даних за допомогою симетричного ключа, шифрування симетричного ключа за допомогою відкритого ключа та надсилання зашифрованих даних і зашифрованого ключа [27]. Симетричний ключ може бути згенерований випадковим чином або отриманий з паролльної фрази чи секретної угоди. За допомогою симетричного ключа дані можна зашифрувати будь-яким симетричним алгоритмом, наприклад, AES. Симетричний ключ може бути зашифрований будь-яким асиметричним алгоритмом, таким як RSA, з використанням відкритого ключа одержувача. Зашифровані дані та зашифрований ключ можуть бути надіслані разом одержувачу, який може розшифрувати їх за допомогою свого приватного ключа та симетричного алгоритму [27].

3.5.4 Отримані результати

В мобільному застосунку за допомогою бібліотеки System.Security.Cryptography було створено клас Cryptography для шифрування зображень паковань лікарських засобів за допомогою алгоритмів AES та RSA і розшифрування спільного ключа, який було зашифровано RSA при комбінованому підході. Після шифрування зображення надсилаються до серверної частини.

В серверній частині застосунку за допомогою бібліотеки PyCryptodome було створено модуль Cryptography для розшифрування отриманих зображень для їхнього подальшого розпізнавання та шифрування спільного ключа за допомогою RSA.

Ключі передаються між мобільним застосунком та серверною частиною з використанням HTTP-запитів. Тому безпека ключа симетричного шифрування сильно залежить від безпеки каналу зв'язку, тоді як ключі під час асиметричного та гібридного шифрування можуть передаватися навіть через незахищені канали.

Характеристики впроваджених методів шифрування для аналізу та порівняння наведені в таблиці 3.8.

Таблиця 3.8 — Порівняння впроваджених методів шифрування

Критерії порівняння	AES	RSA	AES та RSA
Ключі	Один спільний ключ	Приватний і публічний ключі	Комбінація ключів
Розмір ключа	256 біт	2048 біт	256 та 2048 біт
Швидкодія	1,09 с.	51,47 с.	1,44 с.
Складність управління ключами	Висока	Низька	Низька
Передача ключів	Складна	Легка	Легка
Обчислювальні витрати	Менші	Більші	Середні

З таблиці видно, що кожен з методів має свої переваги та недоліки. AES відомий своєю швидкістю та ефективністю в шифруванні великих

обсягів даних, тоді як RSA забезпечує безпечний обмін ключами та високий рівень безпеки, але може бути менш ефективним для великих обсягів даних. Комбінація цих методів з використанням AES для шифрування даних та RSA для безпечного обміну ключами, дозволяє поєднати швидкодію та ефективність симетричного шифрування з безпекою та надійністю асиметричного обміну ключами. Тому можна зробити висновок, що гібридне шифрування найкраще підходить для шифрування даних, які передаються мережею в клієнт-серверному застосунку.

3.6 Основні рішення щодо розробки інтерфейсу

Під час розробки мобільного застосунку було створено наступні елементи графічного інтерфейсу користувача:

- вкладка «Мої ліки» (рис. 3.29), яка містить відсортований перелік ліків, параметри сортування та кнопку «Додати ліки» для переходу до додавання лікарського засобу;

- сторінка «Додати ліки» (рис. 3.30), яка містить поля для введення інформації про лікарський засіб, курс лікування, налаштування сповіщень про приймання ліків та кнопку «Зберегти»;

- сторінка «Переглянути ліки» (рис. 3.31), яка містить поля з інформацією про лікарський засіб, курс лікування, налаштування сповіщень про приймання ліків та кнопку «Зберегти»;

- вкладка «Розпізнавання» (рис. 3.29), яка надає можливість зробити фотографію пакування лікарського засобу для розпізнавання за допомогою кнопки «Пакування», відображає зроблену фотографію, результат розпізнавання та містить кнопку «Додати» для переходу до додавання лікарського засобу з використанням результату розпізнавання;

- вкладка «Інфо» (рис. 3.29), яка містить інформацію про застосунок та розробника.

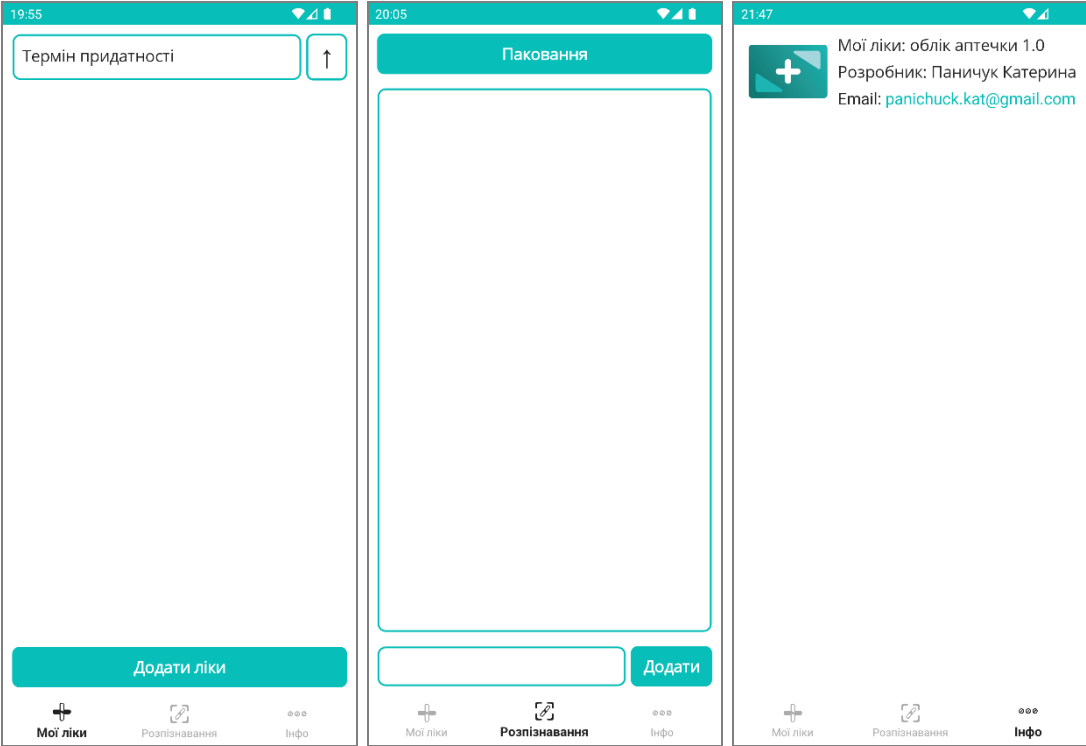


Рисунок 3.29 — Вкладки «Мої ліки», «Розпізнавання» та «Інфо»

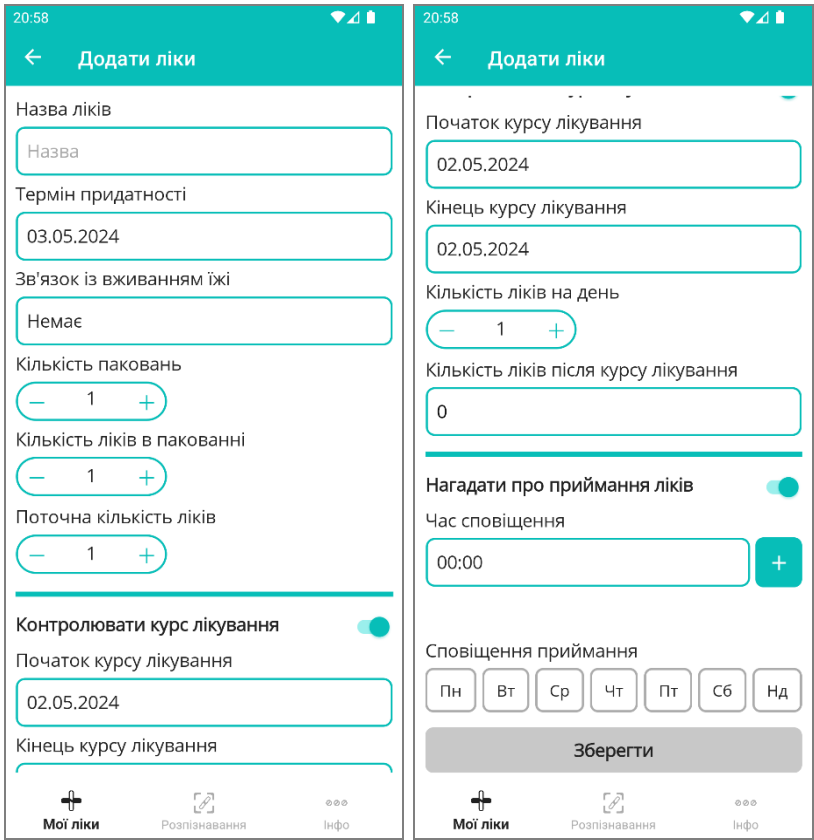


Рисунок 3.30 — Сторінка «Додати ліки»

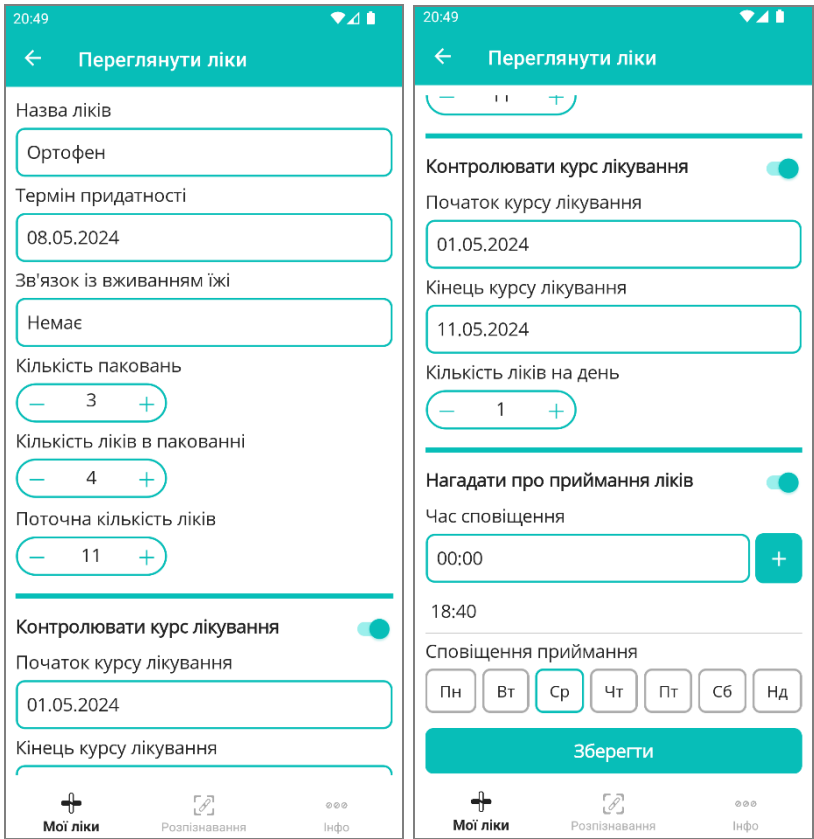


Рисунок 3.31 — Сторінка «Переглянути ліки»

Для застосунку було обрано 3 кольори, які наведено на рисунку 3.32, та намальовано логотип, зображення повідомлень та вкладок, які наведено на рисунку 3.33.



Рисунок 3.32 — Кольори для застосунку

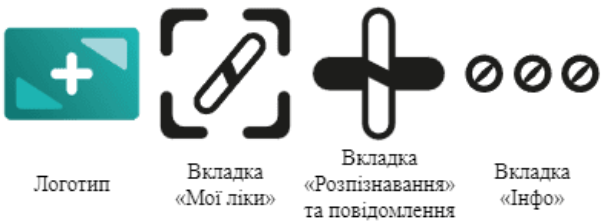


Рисунок 3.33 — Зображення для застосунку

Під час розробки інтерфейсу користувача було використано наступні елементи:

- мітки для відображення опису інших елементів, а також для виведення важливої інформації для користувача;
- кнопки для взаємодії з програмою;
- поля для введення текстової інформації, вибору дати та часу;
- перемикачі для вимкнення та ввімкнення опціональних полів;
- лічильники для введення числової інформації;
- зображення для відображення фотографії пакування;
- список, що розкривається, для вибору поля для сортування переліку ліків.

3.7 Обробка виняткових ситуацій

Для уникнення виняткових ситуацій в мобільному застосунку присутні різні перевірки та обмеження. Для деяких ситуацій було створено повідомлення, які містять текст помилки.

У застосунку присутні такі виняткові ситуації:

- помилки під час взаємодії з серверною частиною;
- відсутність інформації, необхідної для налаштування сповіщень про приймання лікарських засобів.

3.8 Висновки за розділом 3

У цьому розділі було описано обраний шаблон проектування, архітектуру розробленого програмного забезпечення. Також було наведено структури розроблених моделей для розпізнавання лікарських засобів за їхніми пакуваннями. За результатами навчання та тестування було вирішено використовувати модифіковану модель згорткової нейронної мережі, бо вона правильно розпізнає лікарські засоби з високою достовірністю і демонструє

показники на рівні моделей, заснованих на InceptionV3, Xception, ResNet50V2, MobileNetV2, NASNetMobile та DenseNet169.

Було розроблено концептуальну модель бази даних із сутністю «Лікарський засіб». Для цієї сутності було описано поля, їхні типи та розміри.

Потім було описано впроваджені методи шифрування даних AES, RSA та комбінацію цих алгоритмів. За результатами аналізу було зроблено висновок, що гібридне шифрування є найкращим для створеного застосунку.

Також було наведено основні рішення щодо розробки графічного інтерфейсу користувача, а саме: описано сторінки мобільного застосунку, їхнє призначення, використані елементи та кольори.

Крім того, було зазначено виняткові ситуація, повідомлення про які можуть виникнути під час роботи з програмним забезпеченням.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Призначення та умови застосування програми

4.1.1 Призначення програми

Призначенням програми є ведення обліку лікарських засобів універсальної аптечки з можливістю контролю курсу лікування.

4.1.2 Функції програми

Програма виконує наступні функції:

- додавання нових лікарських засобів;
- змінення інформації про лікарський засіб;
- видалення лікарського засобу;
- сортування переліку ліків;
- відслідковування курсу лікування;
- відображення того, чи достатня кількість препарату наявна для курсу лікування;
- відображення терміну придатності та індикація того, що наближується його закінчення;
- сповіщення про настання часу приймати ліки;
- надання можливості зробити фотографію лікарського засобу, її шифрування, розшифрування та подальше розпізнавання.

4.1.3 Умови застосування програми

Мінімальні вимоги до програмних та апаратних засобів для нормального функціонування мобільного застосунку наступні:

- операційна система — Android 5.0 і вище або iOS 11 і вище;
- процесор — 1,5 ГГц двоядерний процесор або вище;
- оперативна пам'ять — обсяг 2 Гб і більше;

- вільний дисковий простір — обсяг 250 Мб і більше;
- доступ до Інтернету;
- наявність камери.

Мінімальні вимоги до програмних та апаратних засобів для нормального функціонування серверної частини програми наступні:

- Python 3.9, Flask 2.2.3, Tensorflow 2.11.0, Keras 2.11.0, NumPy 1.23.5, Pillow 9.4.0, Matplotlib 3.7.2, PyCryptodome 3.19.0;
- процесор — двоядерний з частотою не менше 2 ГГц;
- оперативна пам'ять — обсяг 4 Гб і більше;
- вільний дисковий простір — обсяг 5 Гб і більше;
- доступ до Інтернету.

4.2 Характеристика програми

Мобільний застосунок реалізовано за допомогою фреймворку .NET MAUI мови програмування C# в середовищі розробки Microsoft Visual Studio.

Серверну частину реалізовано за допомогою мови програмування Python з використанням бібліотек Flask, Tensorflow, Keras, NumPy, Pillow, Matplotlib, PyCryptodome в середовищі розробки Microsoft Visual Studio.

4.2.1 Структура програми

Структура мобільного застосунку складається з 27 класів та 6 файлів XAML (eXtensible Application Markup Language), з якими пов'язані 6 класів. Класи моделей представлень та представлень у своїх назвах мають ViewModel та View відповідно. Усі файли програми структуровано за теками згідно з їхнім призначенням. Класи папки «General» призначені для підтримки взаємодії представлень з моделями представлень. Структуру файлів мобільного застосунку зображено на рисунку 4.1.

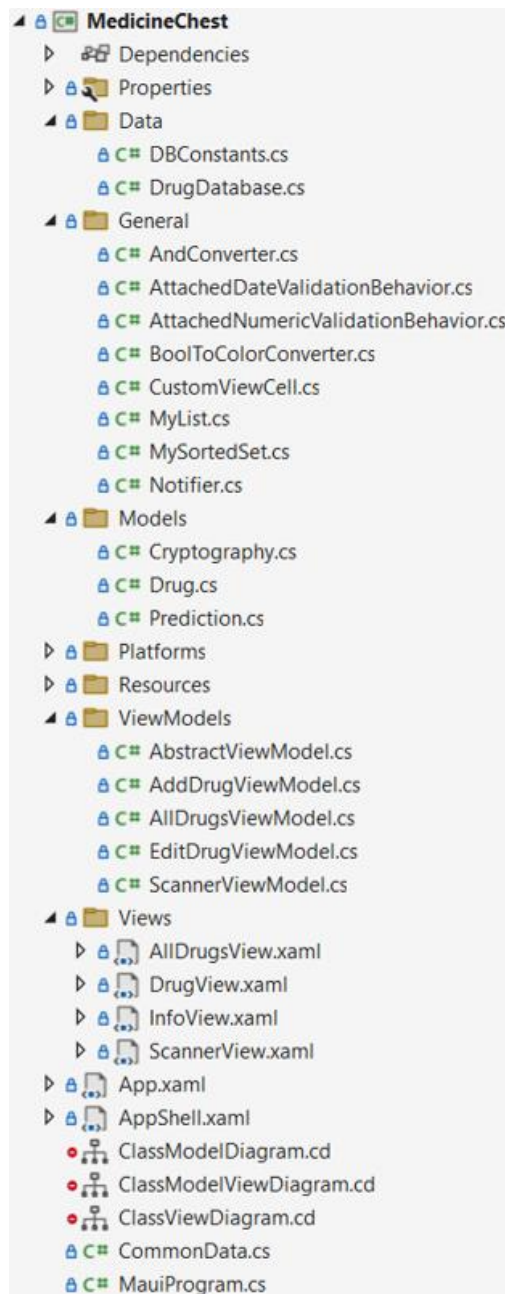


Рисунок 4.1 — Структура файлів мобільного застосунку

Структура серверної частини програми складається з 5 файлів, 3 з яких відповідають за створення моделі для розпізнавання ліків, а інші — за опрацювання HTTP-запитів, шифрування, розшифрування і розпізнавання.

Структуру файлів серверної частини програми зображено на рис. 4.2.

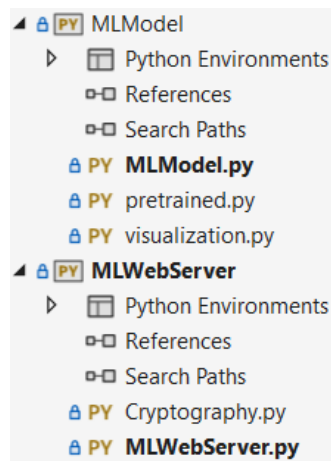


Рисунок 4.2 — Структура файлів серверної частини програми

4.3 Звертання до програми

Перед запуском програми потрібно упевнитися, що пристрій відповідає системним вимогам встановленої програми. Також потрібно перевірити наявність усіх файлів застосунку. Для коректної роботи HTTP-запитів потрібно впевнитися, що середовище розробки містить необхідні компоненти для роботи з Інтернетом.

Для запуску проєктів мобільного застосунку та серверної частини в Microsoft Visual Studio потрібно відкрити файли MedicineChest.sln та MLWebServer.sln відповідно.

Вихідними файлами мобільного застосунку та серверної частини є com.panikat.medichest-Signed.apk (для ОС Android) і MLModel.py, pretrained.py, visualization.py, MLWebServer.py та Cryptography.py.

4.4 Вхідні та вихідні дані

4.4.1 Вхідні дані

Вхідними даними програми є інформація про лікарські засоби, яка зберігається в базі даних, зображення, які використовуються в графічному інтерфейсі, модель розпізнавання, за допомогою якої виконується

розпізнавання в серверній частині, зображення, які використовуються для створення моделі.

4.4.2 Вихідні дані

Вихідними даними програми є інформація про лікарські засоби, збережена в базу даних, та створена модель розпізнавання.

4.5 Повідомлення програмісту

Під час взаємодії з мобільним застосунком програміст може побачити наведені повідомлення.

«Сповіщення не налаштовано! Налаштувати?» — повідомлення про те, що нагадування про приймання ліків ввімкнено, але час або дні тижня не налаштовано. Якщо обрати «Ні», то нагадування про приймання ліків буде вимкнено. Якщо обрати «Так», то можна буде повернутися до налаштування часу або днів тижня сповіщення.

«Помилка на стороні клієнта! Будь ласка, спробуйте пізніше або зверніться до служби підтримки» — повідомлення виникнення помилки на стороні клієнта під час надсилання запиту до серверної частини. Програмісту необхідно перевірити правильність налаштування зв'язку з серверною частиною в мобільному застосунку або підключення до Інтернету.

«Помилка на стороні сервера! Будь ласка, спробуйте пізніше або зверніться до служби підтримки» — повідомлення виникнення помилки на стороні сервера під час опрацювання запиту в серверній частині. Програмісту необхідно перевірити коректність роботи серверної частини.

«Помилка! Будь ласка, спробуйте пізніше або зверніться до служби підтримки» — повідомлення виникнення помилки під час надсилання запиту до серверної частини або його опрацювання. Програмісту необхідно перевірити правильність налаштування зв'язку з серверною частиною в

мобільному застосунку, підключення до Інтернету, коректність роботи серверної частини.

4.6 Висновки за розділом 4

У цьому розділі було наведено призначення та функції розробленого програмного забезпечення та мінімальні системні вимоги для його коректної роботи. Було вказано характеристику та структуру програми. Також у цьому розділі було наведено опис вхідних і вихідних даних та перелік всіх текстових повідомлень, які можуть з'являтися під час роботи програміста з програмним забезпеченням.

5 КЕРІВНИЦТВО ОПЕРАТОРА

5.1 Призначення програми

Призначенням програми є ведення обліку лікарських засобів універсальної аптечки з можливістю контролю курсу лікування.

5.2 Умови виконання програми

5.2.1 Вимоги програми

Мінімальні вимоги до програмних та апаратних засобів для нормального функціонування мобільного застосунку наступні:

- операційна система — Android 5.0 і вище;
- процесор — 1,5 ГГц двоядерний процесор або вище;
- оперативна пам'ять — обсяг 2 Гб і більше;
- вільний дисковий простір — обсяг 250 Мб і більше;
- доступ до Інтернету;
- наявність камери.

5.2.2 Вимоги до користувача

Основною вимогою до користувача є володіння основами роботи з мобільним пристроєм.

5.3 Виконання програми

5.3.1 Запуск програми

Спочатку необхідно встановити мобільний застосунок за допомогою файлу com.panikat.medichest-Signed.apk для ОС Android. Після цього на мобільному пристрої з'явиться іконка застосунку з назвою «Мої ліки», яку зображено на рисунку 5.1.

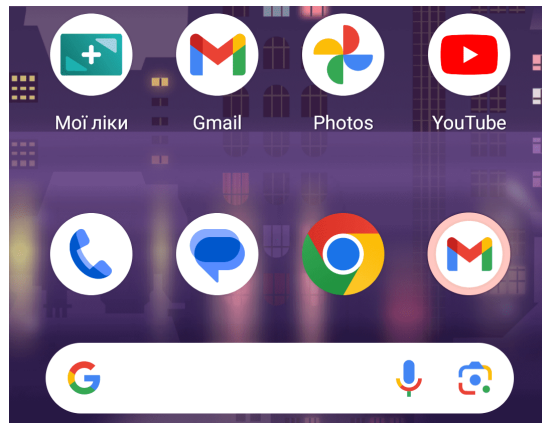


Рисунок 5.1 — Іконка встановленого мобільного застосунку

Після запуску застосунку з'явиться початковий екран, який містить іконку програми. Потім буде відкрита перша вкладка з назвою «Мої ліки», яку зображено на рисунку 5.2.

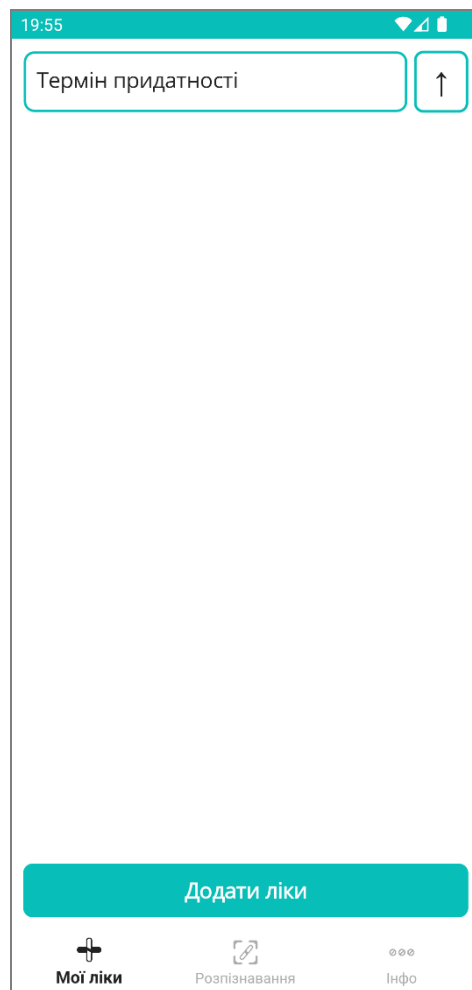


Рисунок 5.2 — Вкладка «Мої ліки»

5.3.2 Робота з програмою

Після запуску програми буде відкрита перша вкладка з назвою «Мої ліки», яку зображено на рисунку 5.2. Вона містить перелік лікарських засобів, в якому будуть ліки, після їхнього додавання, список, що розкривається та містить назви полів для сортування, кнопку зі стрілкою, яка встановлює порядок сортування, кнопку «Додати ліки» для додавання засобу. Приклад змінених параметрів сортування наведено на рисунку 5.3.

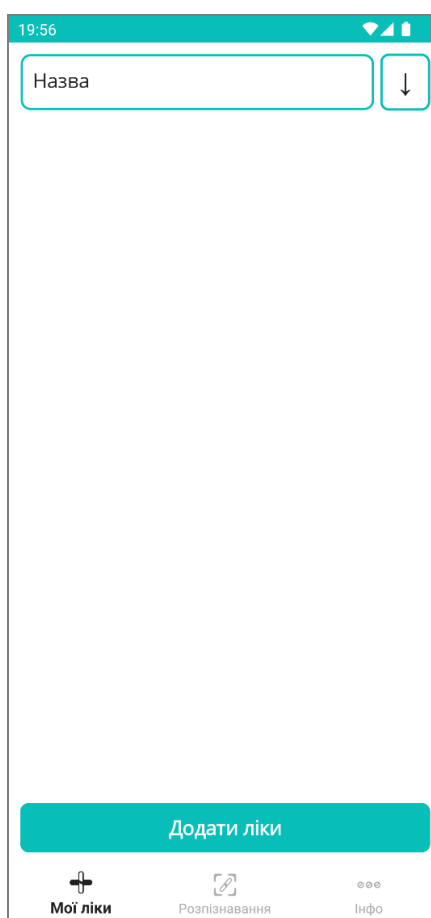


Рисунок 5.3 — Змінені параметри сортування

Після натискання на кнопку «Додати ліки» з'явиться сторінка додавання лікарського засобу (рис. 5.4). Вона містить наступні поля: «Назва ліків», «Термін придатності», «Зв'язок із вживанням їжі», «Кількість пакувань», «Кількість ліків в пакуванні», «Поточна кількість ліків», «Контролювати курс лікування», «Початок курсу лікування», «Кінець курсу

лікування», «Кількість ліків на день», «Кількість ліків після курсу лікування», «Нагадати про приймання ліків», «Час сповіщення», «Сповіщення приймання». Дані в ці поля вводяться за допомогою текстових полів, календарів, годинників, лічильників, перемикачів та кнопок. Контроль курсу лікування та нагадування про приймання ліків є опціональними, тому їхнє вимкнення приховає пов'язані поля.

Поле «Кількість ліків після курсу лікування» розраховується автоматично. Якщо значення є від'ємним, то воно буде червоним, що свідчить про недостатню кількість ліків для курсу лікування.

Для налаштування нагадувань необхідно обрати дні тижня за допомогою відповідних кнопок та додати час сповіщень за допомогою годинника та кнопки «+». Для видалення часу необхідно утримувати значення часу та в контекстному меню натиснути кнопку «Видалити».

The image displays two screenshots of a mobile application interface for adding a medication. Both screens have a teal header with a back arrow and the title 'Додати ліки'.

Left Screenshot:

- Назва ліків:** Валеріана
- Термін придатності:** 15.06.2024
- Зв'язок із вживанням їжі:** Після вживання їжі
- Кількість пакувань:** 2
- Кількість ліків в пакуванні:** 15
- Поточна кількість ліків:** 30
- Контролювати курс лікування:** (Switched on)
- Початок курсу лікування:** 10.05.2024
- Кінець курсу лікування:** (Empty field)

Right Screenshot:

- 10.05.2024** (Date field)
- Кінець курсу лікування:** 31.05.2024
- Кількість ліків на день:** 2
- Кількість ліків після курсу лікування:** -14
- Нагадати про приймання ліків:** (Switched on)
- Час сповіщення:** 20:30
- Сповіщення приймання:** Пн, Вт, Ср, Чт, Пт, Сб, Нд
- Зберегти** (Button)

Both screenshots show a bottom navigation bar with icons for 'Мої ліки', 'Розпізнавання', and 'Інфо'.

Рисунок 5.4 — Додавання лікарського засобу

Після натискання кнопки «Зберегти» лікарський засіб з'явиться в переліку ліків. Для видалення засобу зі списку необхідно утримувати значення засобу та в контекстному меню натиснути кнопку «Видалити» (рис. 5.5).

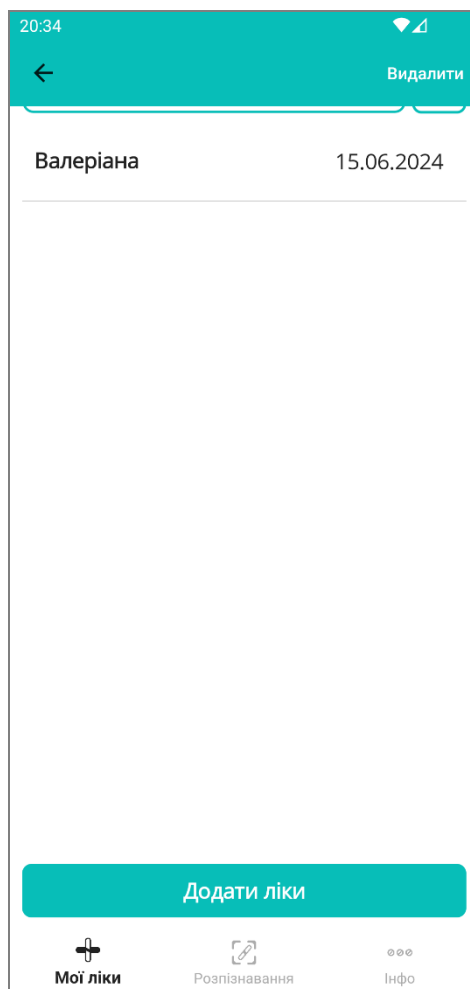


Рисунок 5.5 — Контекстне меню з кнопкою «Видалити»

У переліку ліків зазначено назву та термін придатності лікарського засобу. Якщо термін придатності виділено жовтим, то до його закінчення лишилося 30 днів або менше, якщо помаранчевим, то 7 днів або менше, якщо червоним, то термін придатності вже закінчився, і лікарський засіб вживати не можна.

Приклад кольорової індикації терміну придатності наведено на рисунку 5.6.

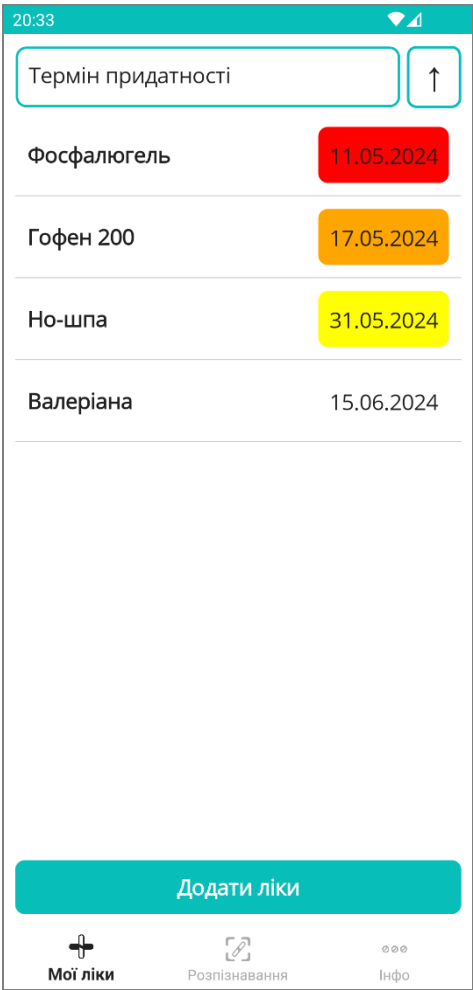


Рисунок 5.6 — Кольорова індикація терміну придатності

Для перегляду та редагування інформації про лікарський засіб потрібно натиснути на нього в переліку ліків. Відкриється сторінка перегляду ліків (рис. 5.7), яка містить ті ж поля, що і при додаванні, крім поля «Кількість ліків після курсу лікування». Для збереження внесених змін необхідно натиснути кнопку «Зберегти».

20:02

← Переглянути ліки

Назва ліків

Валеріана

Термін придатності

15.06.2024

Зв'язок із вживанням їжі

Після вживання їжі

Кількість пакувань

— 2 +

Кількість ліків в пакуванні

— 15 +

Поточна кількість ліків

— 30 +

Контролювати курс лікування

Початок курсу лікування

10.05.2024

Кінець курсу лікування

Мої ліки Розпізнавання Інфо

20:02

← Переглянути ліки

Контролювати курс лікування

Початок курсу лікування

10.05.2024

Кінець курсу лікування

31.05.2024

Кількість ліків на день

— 2 +

Нагадати про приймання ліків

Час сповіщення

00:00 +

10:00

20:30

Сповіщення приймання

Пн Вт Ср Чт Пт Сб Нд

Зберегти

Мої ліки Розпізнавання Інфо

Рисунок 5.7 — Перегляд інформації про лікарський засіб

Друга вкладка «Розпізнавання» (рис. 5.8) призначена для розпізнавання лікарських засобів за фотографіями пакувань. Вона містить кнопку «Пакування», місце для зображення, текстове поле для результату розпізнавання та кнопку «Додати». Після натискання кнопки «Пакування» відкриється застосунок камери для того, щоб зробити фотографію пакування. Потім зображення буде зашифровано та надіслано до серверної частини для розпізнавання. Отримана відповідь буде відображена в текстовому полі. Якщо результат відповідає лікарському засобу, то можна натиснути кнопку «Додати», після чого відкриється сторінка додавання ліків (рис. 5.4) з результатом розпізнавання у полі «Назва».

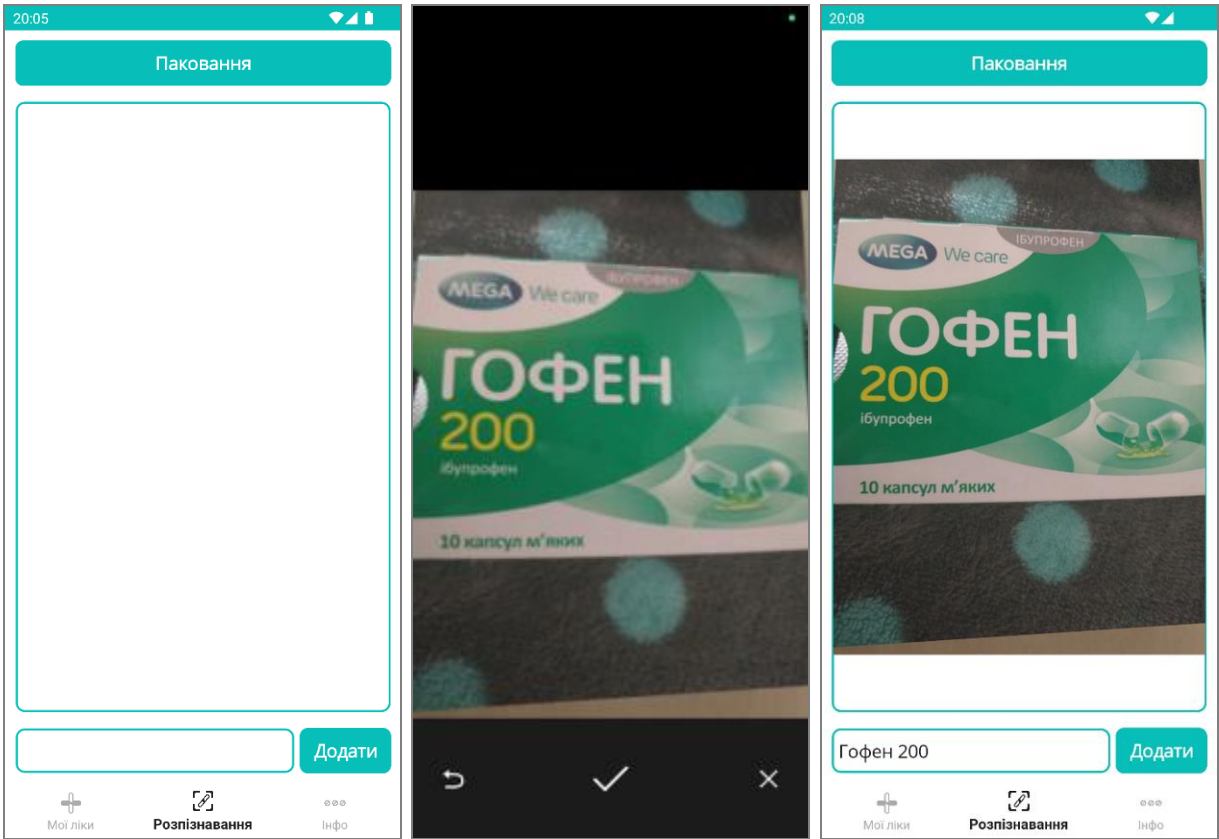


Рисунок 5.8 — Розпізнавання

Після налаштування нагадувань у вказані час та дні тижня буде показано повідомлення з назвою ліків та кнопкою «Вжито», як зображено на рисунку 5.9. Після натискання кнопки поточна кількість засобу зменшиться.

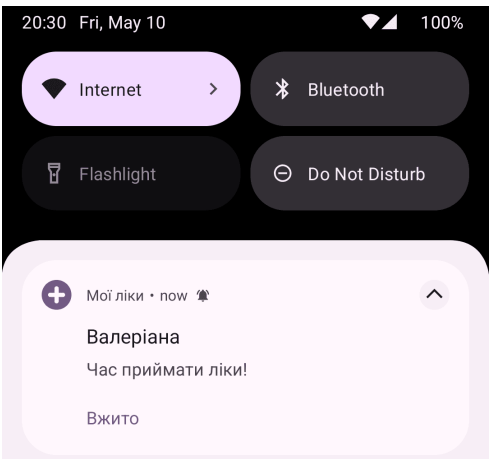


Рисунок 5.9 — Нагадування про приймання ліків

5.4 Повідомлення оператору

Під час взаємодії з програмою оператор може побачити наведені повідомлення.

«Сповіщення не налаштовано! Налаштувати?» — повідомлення про те, що нагадування про приймання ліків ввімкнено, але час або дні тижня не налаштовано (рис. 5.10). Якщо обрати «Ні», то нагадування про приймання ліків буде вимкнено. Якщо обрати «Так», то можна буде повернутися до налаштування часу або днів тижня сповіщення.

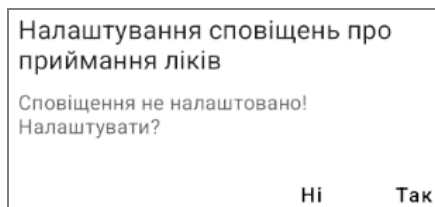


Рисунок 5.10 — Повідомлення про неналаштоване нагадування

«Помилка на стороні клієнта! Будь ласка, спробуйте пізніше або зверніться до служби підтримки» — повідомлення виникнення помилки на стороні клієнта під час надсилання запиту до серверної частини (рис. 5.11). Оператору варто спробувати пізніше або звернутися до служби підтримки.



Рисунок 5.11 — Повідомлення про помилку на стороні клієнта

«Помилка на стороні сервера! Будь ласка, спробуйте пізніше або зверніться до служби підтримки» — повідомлення виникнення помилки на стороні сервера під час опрацювання запиту в серверній частині (рис. 5.12). Оператору варто спробувати пізніше або звернутися до служби підтримки.



Рисунок 5.12 — Повідомлення про помилку на стороні сервера

«Помилка! Будь ласка, спробуйте пізніше або зверніться до служби підтримки» — повідомлення виникнення помилки під час надсилання запиту до серверної частини або його опрацювання (рис. 5.13). Оператору варто спробувати пізніше або звернутися до служби підтримки.



Рисунок 5.13 — Повідомлення про помилку під час роботи з сервером

5.5 Висновки за розділом 5

У цьому розділі було наведено призначення та функції розробленого програмного забезпечення і мінімальні системні вимоги для його коректної роботи. Була описана покрокова інструкція роботи оператора з розробленим мобільним застосунком, а саме було зазначено як додавати, видаляти, редагувати інформацію про лікарські засоби, налаштовувати сповіщення про приймання ліків та виконувати розпізнавання за пакуваннями. Було перераховано перелік текстових повідомлень, які можуть з'являтися під час роботи оператора з програмою. Розроблений інтерфейс мобільного застосунку є простим та зрозумілим, і немає потреби в додатковому навчанні оператора.

6 ТЕСТУВАННЯ І РЕЗУЛЬТАТИ

6.1 Опис методики проведення тестування

Тестування буде виконано за наступним сценарієм:

- запуск мобільного застосунку;
- додавання лікарського засобу;
- зазначення курсу лікування;
- перегляд кількості ліків, яка залишиться після курсу лікування;
- налаштування нагадувань про приймання ліків;
- редагування інформації про лікарський засіб;
- видалення лікарського засобу;
- перевірка наявності індикації термінів придатності;
- сортування переліку ліків;
- створення фотографії пакування лікарського засобу для подальшого шифрування, розшифрування та розпізнавання.

Під час тестування будуть перевірятися основні функціональні можливості застосунку, правильність додавання, відображення даних і т. ін.

Тестування буде проводитися на емуляторі мобільного пристрою з ОС Android з наступними характеристиками:

- модель: Google Pixel 5;
- версія ОС: Android 14.0 (API 34);
- розмір екрану: 6.0 дюймів;
- роздільна здатність екрану: 1080x2340 пікселів, 440 dpi;
- процесор: x86_64;
- пам'ять: 1 Гб;
- доступ до мережі;
- доступ до фронтальної та задньої камер;
- підтримка сенсорів.

6.2 Результати використання розробленого програмного забезпечення

Для тестування розробленого програмного забезпечення на емулятор мобільного пристрою з ОС Android було встановлено застосунок. Після його запуску була відкрита вкладка «Мої ліки» і було натиснуто кнопку «Додати ліки». На відкритій сторінці «Додавання ліків» було заповнено наявні поля. Введеної кількості ліків не вистачить на вказаний курс лікування, про що свідчить від’ємне значення червоного кольору в полі «Кількість ліків після курсу лікування» (рис. 6.1). Після натискання кнопки «Зберегти» лікарський засіб було додано до БД та переліку ліків на вкладці «Мої ліки» (рис. 6.2). Під час повторного запуску застосунку лікарський засіб відображається в списку.

20:01

← Додати ліки

Назва ліків

Валеріана

Термін придатності

15.06.2024

Зв'язок із вживанням їжі

Після вживання їжі

Кількість пакувань

— 2 +

Кількість ліків в пакуванні

— 15 +

Поточна кількість ліків

— 30 +

Контролювати курс лікування

Початок курсу лікування

10.05.2024

Кінець курсу лікування

+

Мої ліки

Розпізнавання

Інфо

20:01

← Додати ліки

10.05.2024

Кінець курсу лікування

31.05.2024

Кількість ліків на день

— 2 +

Кількість ліків після курсу лікування

-14

Нагадати про приймання ліків

Час сповіщення

20:30

+

10:00

20:30

Сповіщення приймання

Пн Вт Ср Чт Пт Сб Нд

Зберегти

+

Мої ліки

Розпізнавання

Інфо

Рисунок 6.1 — Додавання лікарського засобу

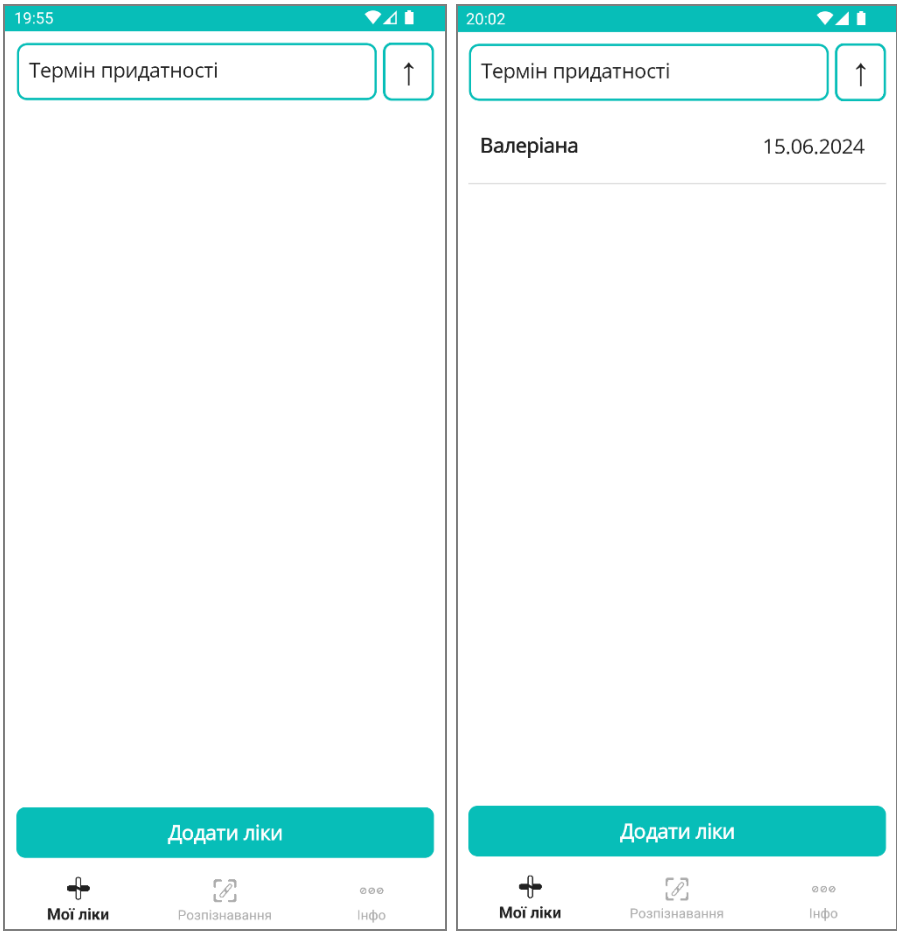


Рисунок 6.2 — Зміни в переліку ліків

Коли настав час приймання доданого лікарського засобу, то спрацювало нагадування, і повідомлення з’явилося на панелі повідомлень (рис. 6.3). Після натискання кнопки «Вжито» поточна кількість засобу зменшилась.

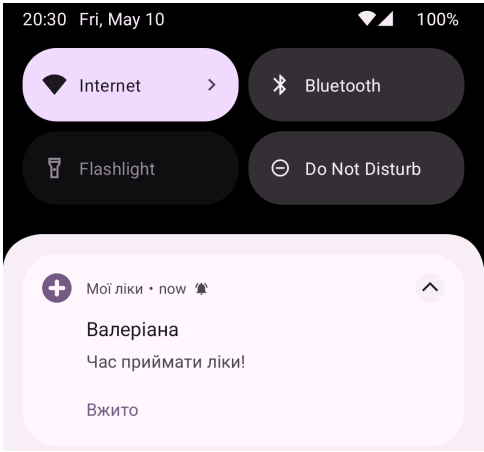


Рисунок 6.3 — Сповіщення про настання часу приймати ліки

Після натискання на лікарський засіб була відкрита сторінка «Переглянути ліки», на якій було внесено зміни та натиснуто кнопку «Зберегти» (рис. 6.4). Оновлена інформація була збережена в БД. Під час повторного запуску застосунку лікарський засіб містить відредаговані дані.

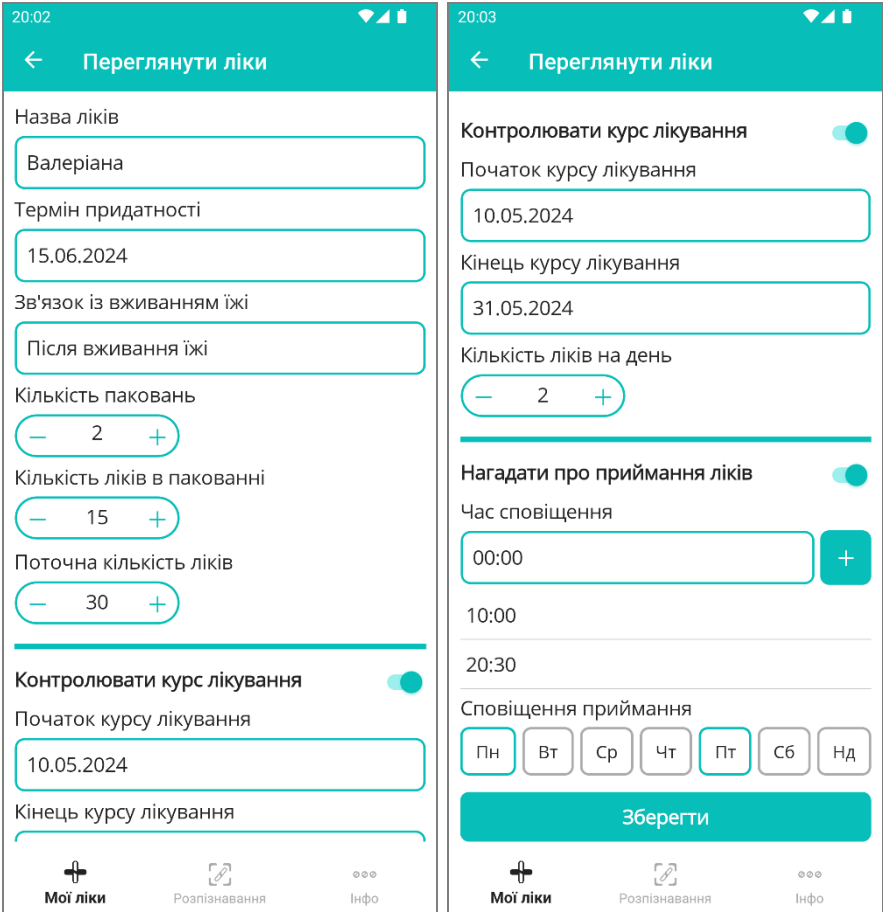


Рисунок 6.4 — Змінення інформації про лікарський засіб

Для видалення лікарського засобу зі списку та БД було утримано значення засобу та в контекстному меню натиснуто кнопку «Видалити» (рис. 6.5). Під час повторного запуску застосунку лікарського засобу не було в переліку.

В списку ліків чим ближчий термін придатності до закінчення, тим більше колір виділення змінюється від жовтого до червоного. Над переліком є параметри сортування. Після їхнього змінення порядок розташування лікарських засобів змінився (рис. 6.6).

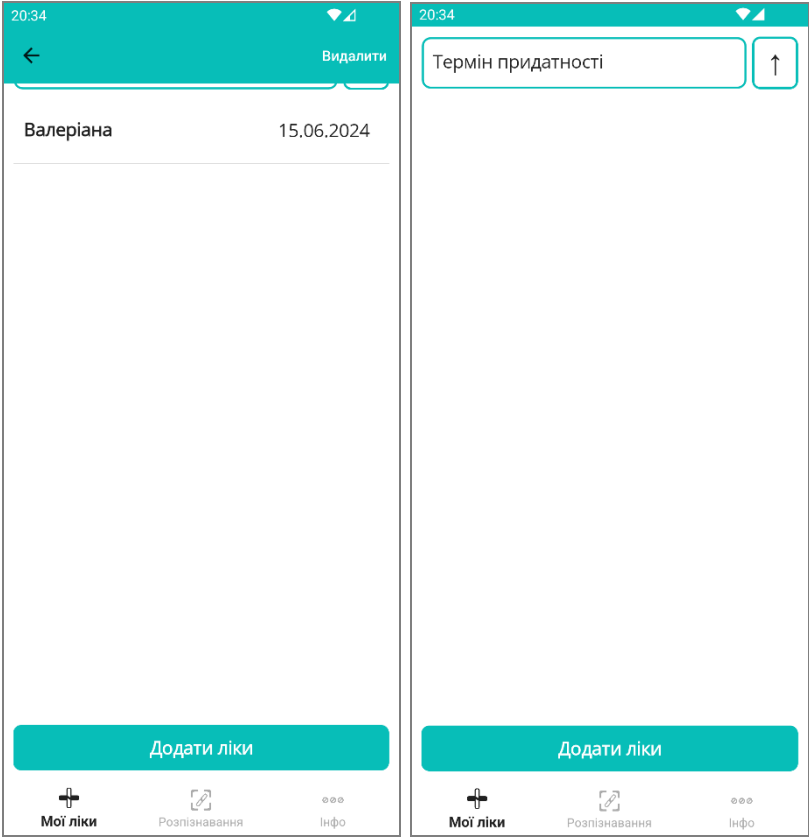


Рисунок 6.5 — Видалення лікарського засобу

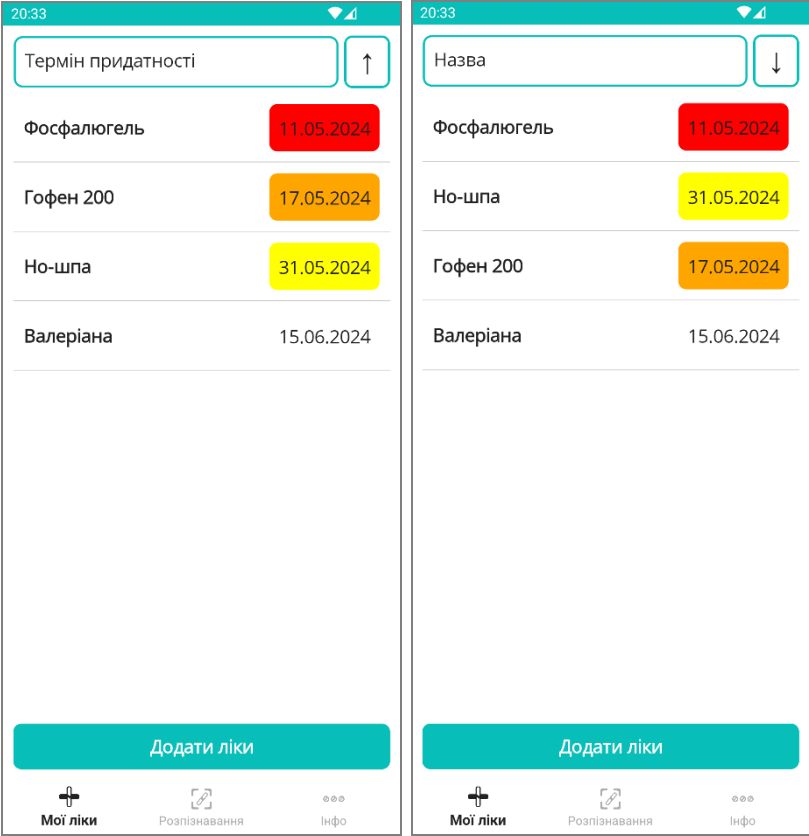


Рисунок 6.6 — Кольорова індикація терміну придатності і сортування списку

На другій вкладці «Розпізнавання» було натиснуто на кнопку «Пакування», зроблено фотографію пакування лікарського засобу. Потім в текстовому полі з'явився результат розпізнавання з назвою лікарського засобу (рис. 6.7). Після натискання на кнопку «Додати» було відкрито сторінку «Додати ліків» із заповненим полем «Назва ліків».

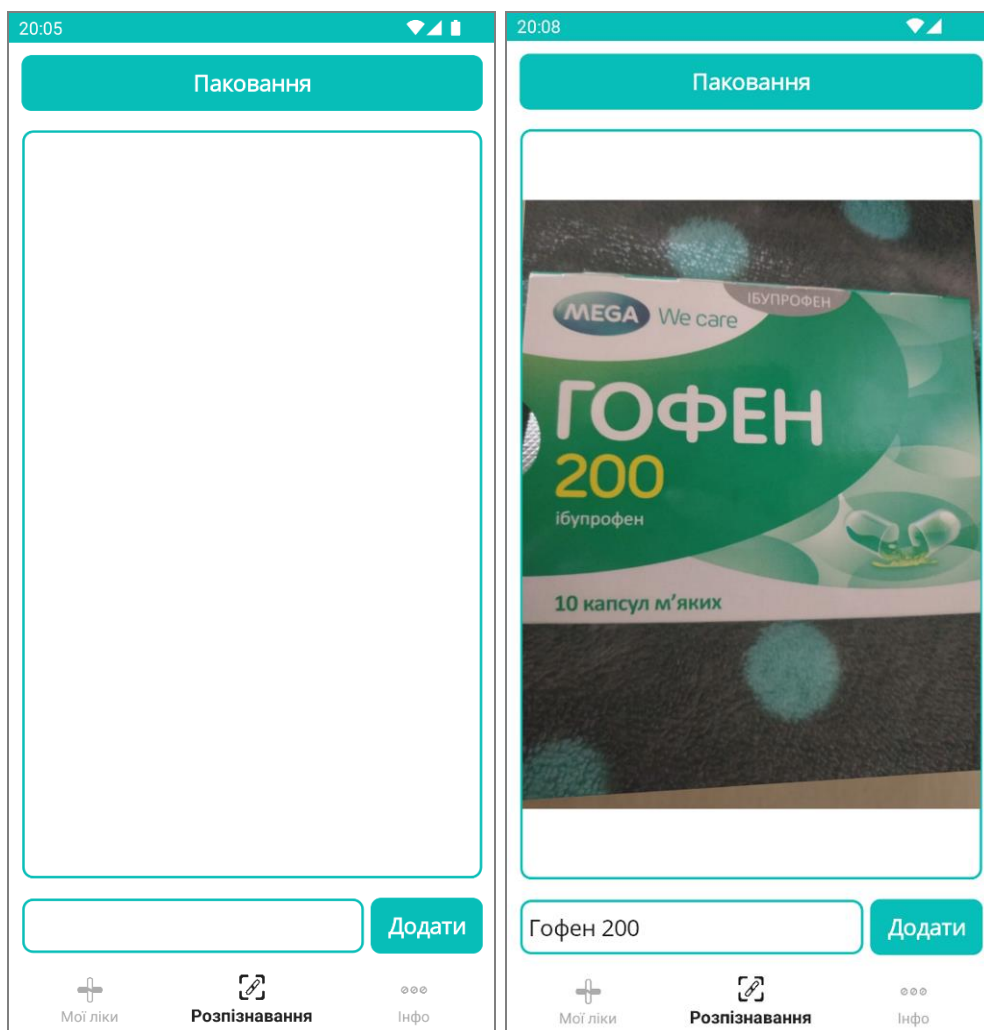


Рисунок 6.7 — Розпізнавання лікарського засобу за пакуванням

Під час тестування мобільного застосунку були виконані різні сценарії, включаючи перевірку функціональності, стабільності, взаємодії з серверною частиною, коректного зберігання та обробки даних. Результати тестування було зведено в таблицю 6.1.

Таблиця 6.1 — Результати тестування

Перевірка	Результат
Додавання нових лікарських засобів	Пройдено
Змінення інформації про лікарський засіб	Пройдено
Видалення лікарського засобу	Пройдено
Сортування переліку ліків	Пройдено
Відслідковування курсу лікування	Пройдено
Відображення того, чи достатня кількість препарату наявна для курсу лікування	Пройдено
Відображення терміну придатності та індикація того, що наближується його закінчення	Пройдено
Сповіщення про настання часу приймати ліки	Пройдено
Надання можливості зробити фотографію лікарського засобу, її шифрування, розшифрування та подальше розпізнавання	Пройдено
Сповіщення про наближення терміну придатності до закінчення	Не реалізовано
Сповіщення про необхідність отримання рецепту від лікаря для придбання лікарського засобу	Не реалізовано

6.3 Висновки за розділом 6

В цьому розділі було описано сценарій проведення тестування програмного забезпечення, а також характеристики пристрою, який було використано для цього.

У ході тестування було перевірено правильність функціонування програми, стабільність та відповідність вимогам.

ВИСНОВКИ

Під час виконання роботи було проведено аналіз предметної області, наведено актуальність та доцільність розробки програмного забезпечення.

У першому розділі було розглянуто відомі методи виконання завдання обліку лікарських засобів універсальної аптечки та мобільні застосунки, що реалізують цей облік. У ході аналізу аналогів було виявлено їхні переваги й недоліки та визначено функціонал для реалізації.

У другому розділі було проведено аналіз і вибір інструментарію розробки програми та було визначено структуру майбутнього програмного забезпечення. Було проведено порівняльний аналіз мов програмування та встановлено, що для розробки мобільного застосунку буде використано мову програмування C# з фреймворком .NET MAUI, а для серверної частини — мову програмування Python з використанням Flask, Tensorflow, Keras та PyCryptodome. Було проведено порівняльний аналіз середовищ розробки, в ході якого було встановлено, що найкращим варіантом є середовище Visual Studio. Також, було проведено порівняльний аналіз систем управління базами даних, в ході якого було встановлено, що найкращим варіантом є SQLite.

У третьому розділі було описано обраний шаблон проектування, архітектуру розробленого програмного забезпечення. Також було наведено структури розроблених моделей для розпізнавання лікарських засобів за їхніми пакуваннями. За результатами навчання та тестування було вирішено використовувати модифіковану модель згорткової нейронної мережі, точність якої складає 92%, та яка правильно розпізнала усі 5 тестових зображень пакувань лікарських засобів. Було розроблено концептуальну модель бази даних із сутністю «Лікарський засіб». Для цієї сутності було описано поля, їхні типи та розміри. Потім було описано впроваджені методи шифрування даних AES, RSA та комбінацію цих алгоритмів. За результатами аналізу було зроблено висновок, що гібридне шифрування є найкращим для розробленого програмного забезпечення. Також було наведено основні

рішення щодо розробки графічного інтерфейсу користувача та зазначено виняткові ситуації, повідомлення про які можуть виникнути під час роботи з програмним забезпеченням.

В створеному програмному забезпеченні було реалізовано усі зазначені в першому розділі функції, крім сповіщень про наближення терміну придатності до закінчення та необхідність отримання рецепту від лікаря для придбання лікарського засобу. Тестування програми довело, що увесь розроблений функціонал працює коректно.

В подальшому можливо покращити створене програмне забезпечення за допомогою додавання сповіщень про необхідність придбання лікарського засобу через те, що воно скоро закінчиться, реалізації сканування штрих-кодів або серійних номерів, покращення моделі розпізнавання за пакуваннями, механізмів забезпечення безпеки та розширення переліку розпізнавання, надання можливості перегляду інструкцій для застосування лікарських засобів тощо.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Mobile Apps for Increasing Treatment Adherence: Systematic Review [Text] / [V. Pérez-Jover, M. Sala-González, M. Guilabert et al.] // Journal of medical Internet research. – 2019. – Vol. 21, № 6.
2. Гордєєв, А. Мои лекарства: учет аптечки [Электрон. ресурс] / А. Гордєєв. – Режим доступа: <https://play.google.com/store/apps/details?id=com.andreygordeev.mydrugs>.
3. Медицинский справочник-аптечка [Электрон. ресурс]. – Режим доступа: <https://play.google.com/store/apps/details?id=ru.medical.spravochnik>.
4. Аптечка. Учет медикаментов. [Электрон. ресурс]. – Режим доступа: <https://play.google.com/store/apps/details?id=ru.decsoft.myaidkit>.
5. Шевченко, Р. Рейтинг мов програмування 2023. JavaScript/TypeScript завоюють світ, Python увійшов у топ-3, Salesforce Apex випередив 1C [Електрон. ресурс] / Р. Шевченко, І. Яновський. – Режим доступу: <https://dou.ua/lenta/articles/language-rating-2023/>.
6. Hrabchak, M. What is the Best Language for Mobile App Development [Electronic resource] / M. Hrabchak. – Access mode: <https://idapgroup.com/blog/best-programming-language-for-mobile-app-development/>.
7. Iannace, K. Pros and Cons of the 11 Best Mobile App Programming Languages [Electronic resource] / K. Iannace. – Access mode: <https://designli.co/blog/11-best-mobile-app-programming-languages-pros-and-cons/>.
8. Oddy, C. C++ vs Python [Electronic resource] / C. Oddy. – Access mode: <https://www.ko2.co.uk/c-plus-plus-vs-python/>.
9. Okeke, F. The 12 best IDEs for programming [Electronic resource] / F. Okeke. – Access mode: <https://www.techrepublic.com/article/best-ide-software/>.
10. Android Studio vs. Xcode – Pros and Cons of each Application Development Platform [Electronic resource]. – Access mode:

<https://www.iubenda.com/en/help/127491-android-studio-vs-xcode-pros-and-cons-of-each-application-development-platform>.

11. Welch, S. Best C++ IDEs [Electronic resource] / S. Welch. – Access mode: <https://www.udacity.com/blog/2020/05/best-c-ides.html>.

12. Comparing Database Management Systems: MySQL, PostgreSQL, MSSQL Server, MongoDB, Elasticsearch, and others [Electronic resource]. – Access mode: <https://www.altexsoft.com/blog/business/comparing-database-management-systems-mysql-postgresql-mssql-server-mongodb-elasticsearch-and-others/>.

13. Grishina, A. Software Design Patterns 101: What Is It and Why Must You Learn Them? [Electronic resource] / A. Grishina. – Access mode: <https://softteco.com/blog/software-design-patterns>.

14. Gallardo, E. G. What Is MVVM Architecture? [Electronic resource] / E. G. Gallardo. – Access mode: <https://builtin.com/software-engineering-perspectives/mvvm-architecture>.

15. Saha, S. A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way [Electronic resource] / S. Saha. – Access mode: <https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way/>.

16. Li, H. A Comparison of Weight Initializers in Deep Learning-Based Side-Channel Analysis [Text] / H. Li, M. Krček, G. Perin // Applied Cryptography and Network Security Workshops. ACNS 2020. Lecture Notes in Computer Science. – 2020. – Vol. 12418. – P. 126-143.

17. Narein, A. Inception V3 Model Architecture [Electronic resource] / A. Narein. – Access mode: <https://iq.opengenus.org/inception-v3-model-architecture/>.

18. Fabien, M. XCeption Model and Depthwise Separable Convolutions [Electronic resource] / M. Fabien. – Access mode: <https://maelfabien.github.io/deeplearning/xception/>.

19. ResNet-50: The Basics and a Quick Tutorial [Electronic resource]. – Access mode: <https://datagen.tech/guides/computer-vision/resnet-50/>.
20. Badgujar, S. Creating MobileNetsV2 with TensorFlow from scratch [Electronic resource] / S. Badgujar. – Access mode: <https://medium.com/analytics-vidhya/creating-mobilenetsv2-with-tensorflow-from-scratch-c85eb8605342>.
21. AgroAid: A Mobile App System for Visual Classification of Plant Species and Diseases Using Deep Learning and TensorFlow Lite [Text] / [M. Reda, R. Suwwan, S. Alkafri et al.] // Informatics. – 2022. – Vol. 9, № 3. – P. 55.
22. Fine-Tuned DenseNet-169 for Breast Cancer Metastasis Prediction Using FastAI and 1-Cycle Policy [Text] / [A. Vulli, P. N. Srinivasu, M. S. K. Sashank et al.] // Sensors (Basel). – 2022. – Vol. 22, № 8. – P. 2988.
23. Franklin, R. AES vs. RSA Encryption: What Are the Differences? [Electronic resource] / R. Franklin. – Access mode: <https://www.precisely.com/blog/data-security/aes-vs-rsa-encryption-differences>.
24. Froehlich, A. What is ciphertext feedback (CFB)? [Electronic resource] / A. Froehlich. – Access mode: <https://www.techtarget.com/searchsecurity/definition/ciphertext-feedback>.
25. Hammod, D. N. An Enhancement Method Based on Modifying CFB Mode for Key Generation in AES Algorithm [Text] / D. N. Hammod, M. H. Al-Rawi, H. S. Abdulah // Engineering and Technology Journal. – 2016. – Vol. 34, № 6, Part (B) Scientific. – P. 759-768.
26. Introduction to RSA [Electronic resource] – Access mode: <https://medium.com/@c0D3M/introduction-to-rsa-e8cb39af508e>.
27. How do you use AES and RSA together for hybrid encryption schemes? [Electronic resource] – Access mode: <https://www.linkedin.com/advice/0/how-do-you-use-aes-rsa-together-hybrid-encryption-schemes>.

ДОДАТОК А
Технічне завдання

Вступ

Мобільний застосунок «Мої ліки: облік аптечки» призначений для використання у різних сферах, де користувач має аптечку. Усі сфери застосування програми об'єднує потреба слідкувати за станом лікарських засобів та правильним їх застосуванням.

A.1 Підстави для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Програмне забезпечення обліку лікарських засобів універсальної аптечки», затверджене наказом Національного університету «Запорізька політехніка» № 168 від 24 квітня 2024 р.

A.2 Призначення розробки

Програма буде використовуватися для обліку лікарських засобів універсальної аптечки. Також буде присутня можливість використання для контролю курсу лікування.

A.2.1 Функціональне призначення

Програма має надавати можливості додавати, видаляти, сортувати лікарські засоби, редагувати інформацію про них, контролювати курс лікування, налаштовувати сповіщення про наближення терміну придатності до закінчення, необхідність отримання рецепту від лікаря для придбання лікарського засобу, настання часу приймати ліки, розпізнавати деякі засоби за їхніми пакуванням.

A.2.2 Експлуатаційне призначення

Програма має експлуатуватися користувачами на мобільних пристроях, які задовольняють мінімальні вимоги до програмних і апаратних засобів для нормального функціонування застосунку.

A.3 Вимоги до програми або програмного продукту

A.3.1 Вимоги до функціональних характеристик

A.3.1.1 Користувач

Користувач — це людина, яка бажає контролювати стан своєї аптечки та слідкувати за дотриманням курсу лікування.

A.3.1.2 Вимоги до складу виконуваних функцій

Мобільний застосунок має забезпечувати виконання наступних функцій:

- додавання нових лікарських засобів;
- змінення інформації про лікарський засіб;
- видалення лікарського засобу;
- сортування переліку ліків;
- відслідковування курсу лікування;
- відображення того, чи достатня кількість препарату наявна для курсу лікування;
- відображення терміну придатності та індикація того, що наближується його закінчення;
- сповіщення про наближення терміну придатності до закінчення, необхідність отримання рецепту від лікаря для придбання лікарського засобу, настання часу приймати ліки;

– надання можливості зробити фотографію лікарського засобу, її шифрування, розшифрування та подальше розпізнавання.

А.3.1.3 Вимоги до організації початкових даних

Вхідні дані в контексті програми — це ті, що знаходяться у БД на початку роботи програми та ті, які вводить користувач у процесі роботи. Вхідні дані мають бути організовані як система лікарських засобів.

Структура інформації про лікарський засіб:

- назва;
- зв'язок з вживанням їжі;
- термін придатності;
- кількість пакувань та ліків в них;
- загальна кількість ліків;
- контроль курсу лікування;
- початок та кінець курсу лікування;
- кількість ліків на день;
- нагадування про необхідність отримання рецепту від лікаря;
- нагадування про приймання ліків;
- час сповіщення;
- дні приймання.

А.3.1.4 Вимоги до організації вихідних даних

Вихідні дані — це та інформація, яку бачить користувач на екрані; яка оброблена алгоритмами та виведена за допомогою інтерфейсу.

До вихідних даних має входити кількість ліків, які залишаться після проходження курсу лікування.

A.3.1.5 Вимоги до тимчасових характеристик

Вимоги до тимчасових характеристик програми не пред'являються.

A.3.2 Вимоги до надійності

Імовірність безвідмовної роботи системи повинна бути не менше 99%.

A.3.2.1 Вимоги до забезпечення надійного функціонування програми

Надійне (стійке) функціонування програми має бути забезпечене виконанням сукупності організаційно-технічних заходів, перелік яких наведено нижче:

- організацією безперебійного живлення технічних засобів;
- використанням ліцензійного програмного забезпечення;
- регулярним виконанням вимог стандартів захисту інформації.

Виконання перевірок на наявність вірусів.

A.3.2.2 Збійні ситуації

Програмою мають оброблятися наступні збійні ситуації:

- відсутній зв'язок з базою даних;
- занесення до бази даних некоректних даних;
- неможливість коректного зчитування даних з бази даних;
- не заповнено обов'язкові поля;
- помилки під час зв'язку з серверною частиною.

У разі виникнення збою повинні видаватися коректні повідомлення із зазначенням подальших дій.

А.3.2.3 Час відновлення після відмови

Вимоги до часу відновлення після відмови не пред'являються.

А.3.3 Умови експлуатації

А.3.3.1 Кліматичні умови експлуатації

Кліматичні умови експлуатації, за яких повинні забезпечуватися вказані характеристики, повинні задовольняти вимоги, які пред'являються до технічних засобів у частині умов їх експлуатації.

А.3.3.2 Вимоги до видів обслуговування

Програма не вимагає проведення яких-небудь видів обслуговування.

А.3.3.3 Вимоги до чисельності та кваліфікації персоналу

Вимоги до чисельності та кваліфікації персоналу не пред'являються.

А.3.4 Вимоги до складу та параметрів технічних засобів

Мінімальні вимоги до програмних та апаратних засобів для нормального функціонування мобільного застосунку наступні:

- операційна система — Android 5.0 і вище або iOS 11 і вище;
- процесор — 1,5 ГГц двоядерний процесор або вище;
- оперативна пам'ять — обсяг 2 Гб і більше;
- вільний дисковий простір — обсяг 250 Мб і більше;
- доступ до Інтернету;
- наявність камери.

Мінімальні вимоги до програмних та апаратних засобів для нормального функціонування серверної частини програми наступні:

- Python 3.9, Flask 2.2.3, Tensorflow 2.11.0, Keras 2.11.0, NumPy 1.23.5, Pillow 9.4.0, Matplotlib 3.7.2, PyCryptodome 3.19.0;
- процесор — двоядерний з частотою не менше 2 ГГц;
- оперативна пам'ять — обсяг 4 Гб і більше;
- вільний дисковий простір — обсяг 5 Гб і більше;
- доступ до Інтернету.

А.3.5 Вимоги до інформаційної та програмної сумісності

Програма повинна бути написана на мові програмування C#. Для її створення має бути використане інтегроване середовище розробки Microsoft Visual Studio. При визначенні архітектури програмного забезпечення має застосовуватися шаблон проєктування Model-View-ViewModel. Для створення графічного інтерфейсу має бути використано фреймворк .NET MAUI.

Застосунок орієнтований на роботу в ОС Android та iOS, тому для коректної роботи програми необхідне стабільне функціонування ОС.

Серверна частина програми повинна бути написана на мові програмування Python з використанням бібліотек Flask, Tensorflow, Keras, NumPy, Pillow, Matplotlib, PyCryptodome. Для її створення має бути використане інтегроване середовище розробки Microsoft Visual Studio.

Серверна частина програми орієнтована на роботу в будь-якій операційній системі, яка підтримує Python та бібліотеки Flask, Tensorflow, Keras, NumPy, Pillow, Matplotlib, PyCryptodome.

А.3.6 Вимоги до маркування та пакування

Мобільний застосунок передається у вигляді файлу APK (Android Package Kit) для Android та файлу IPA (iOS App Store Package) для iOS. Вимоги до маркування відсутні.

А.3.7 Вимоги до транспортування та збереження

Вимоги до транспортування та збереження відсутні.

А.3.8 Спеціальні вимоги

Програма повинна забезпечувати взаємодію з користувачем за допомогою графічного інтерфейсу, зображеного на рисунках А.1-А.2.

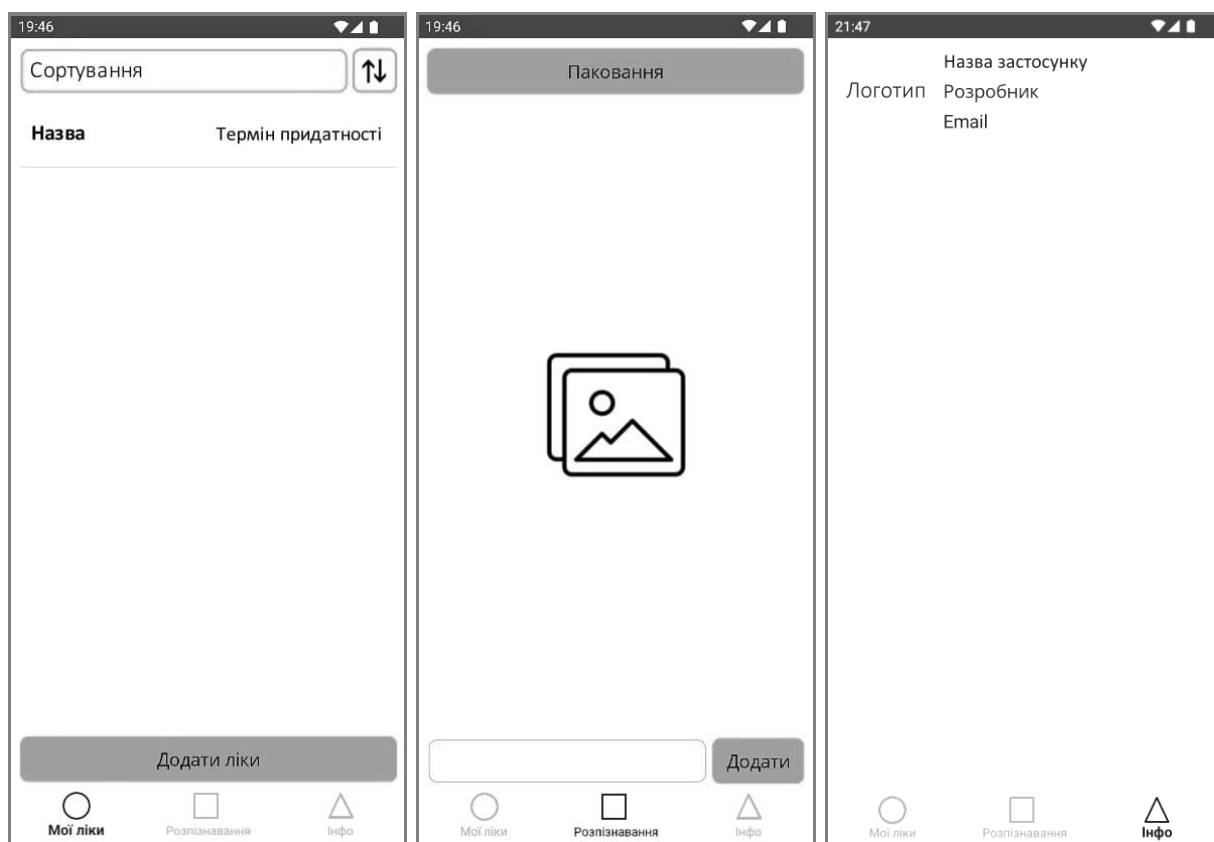


Рисунок А.1 — Сторінки з переліком ліків, розпізнаванням та інформацією

Рисунок А.2 — Сторінки додавання та перегляду ліків

А.4 Вимоги до програмної документації

Склад програмної документації повинен включати:

- технічне завдання;
- керівництво програміста;
- керівництво оператора;
- опис програми;
- програма і методика випробувань.

А.5 Техніко-економічні показники

Економічна ефективність від використання програми забезпечується за рахунок скорочення витрат часу на ведення паперового журналу, в якому

вручну заносяться назви ліків, дати придбання та споживання, для будь-якого користувача незалежно від його розташування та часу доби. Також ефективність досягається завдяки уникненню придбання зайвих ліків.

A.6 Стадії та етапи розробки

Розробка повинна бути проведена в 3 стадії:

- технічне завдання;
- технічний і робочий проєкти;
- впровадження.

На стадії «Технічне завдання» повинен бути виконаний етап розробки, узгодження і затвердження технічного завдання.

На стадії «Технічний і робочий проєкти» повинні бути виконані:

- розробка програми;
- розробка програмної документації;
- тестування програми.

На стадії «Впровадження» повинен бути виконаний етап підготовки та передачі програми.

Зміст робіт за етапами. На етапі розробки технічного завдання повинні бути виконані перераховані нижче роботи:

- постановка задачі;
- визначення та уточнення вимог до технічних засобів;
- визначення вимог до програми;
- визначення стадій, етапів і термінів розробки програми і документації до неї;
- узгодження і затвердження технічного завдання.

На етапі розробки програми повинна бути виконана робота з програмування (кодування) і налагодження програми.

На етапі розробки програмної документації розробляються відповідні документи.

На етапі тестування програми повинні бути виконані перераховані нижче види робіт:

- розробка, узгодження та затвердження порядку і методики випробувань;
- проведення приймально-здавальних випробувань;
- документації за результатами випробувань.

На етапі підготовки та передачі програми повинна бути виконана робота з підготовки та передачі програми і програмної документації в експлуатацію.

Виконавець повинен розробити мобільний застосунок «Мої ліки: облік аптечки», надати вихідні коди та документацію до розробленої програми не пізніше 26.05.2024.

A.7 Порядок контролю та приймання

Приймально-здавальні випробування програми повинні проводитися відповідно до розробленого виконавцем і узгодженого із замовником документа «Програма і методика випробувань».

Хід проведення приймально-здавальних випробувань замовник і виконавець документують в протоколі випробувань.

На підставі протоколу випробувань виконавець спільно із замовником підписують акт приймання-здачі програми в експлуатацію.

ДОДАТОК Б
Опис програми

Б.1 Загальні відомості

Мобільний застосунок має назву «Мої ліки: облік аптечки». Призначенням програми є ведення обліку лікарських засобів універсальної аптечки з можливістю контролю курсу лікування.

Користувачем застосунку є людина, яка бажає контролювати стан своєї аптечки та слідкувати за дотриманням курсу лікування.

Б.2 Функціональне призначення

Мобільний застосунок забезпечує виконання нижче перерахованих функцій:

- додавання та видалення лікарських засобів;
- змінення інформації про лікарський засіб;
- сортування переліку ліків;
- відслідковування курсу лікування;
- відображення того, чи достатня кількість препарату наявна для курсу лікування;
- відображення терміну придатності та індикація того, що наближується його закінчення;
- сповіщення про настання часу приймати ліки;
- надання можливості зробити фотографію лікарського засобу, її шифрування, розшифрування та подальше розпізнавання.

Б.3 Опис логічної структури програми

Програма складається з двох компонентів: мобільний застосунок та серверна частина. Структурну схему програмного забезпечення наведено на рисунку Б.1.

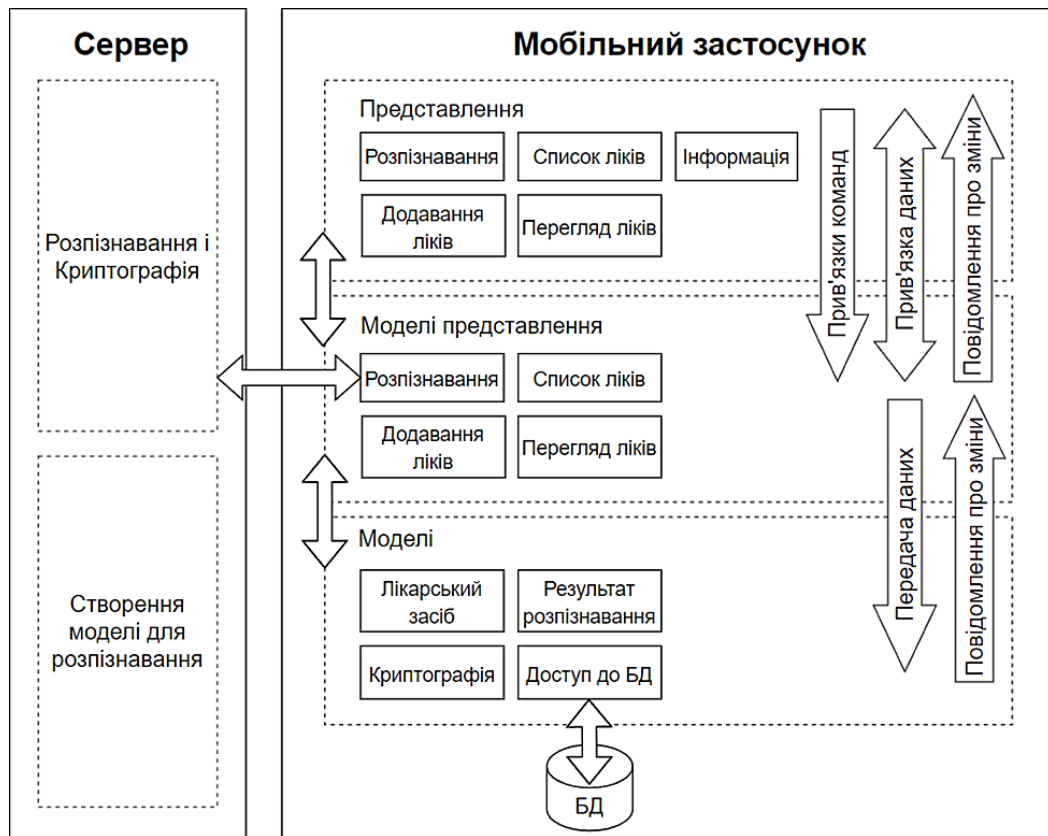


Рисунок Б.1 — Структурна схема програмного забезпечення

Мобільний застосунок містить три функціональні частини: модель, представлення, модель представлення. Модулі частини, яка містить моделі, описані в таблиці Б.1.

Таблиця Б.1 — Опис модулів частини, яка містить моделі

Модуль	Структура	Функції
Доступ до БД	class DBConstants class CommonData class DrugDatabase	Зчитування інформації з БД; Запис інформації в БД; Оновлення даних у БД; Видалення даних з БД.
Лікарський засіб	class Drug	Зберігання даних про лікарський засіб; Редагування даних.
Результат розпізнавання	class Prediction	Зберігання даних про результат розпізнавання; Редагування даних.
Криптографія	class Cryptography	Шифрування зображень.

Модулі частини, яка містить моделі представлення, описані в таблиці Б.2.

Таблиця Б.2 — Опис модулів частини з моделями представлення

Модуль	Структура	Функції
Список ліків	class AllDrugsViewModel	Отримання переліку ліків; Сортування списку з обраними параметрами; Видалення обраного лікарського засобу.
Додавання ліків	class AbstractViewModel class AddDrugViewModel	Отримання інформації від користувача; Налаштування сповіщень; Створення запису в БД.
Перегляд ліків	class AbstractViewModel class EditDrugViewModel	Отримання обраного лікарського засобу; Налаштування сповіщень; Оновлення інформації про лікарський засіб в БД після редагування.
Розпізнавання	class ScannerViewModel	Отримання ключів шифрування від сервера; Отримання зображення пакування лікарського засобу з камери, його шифрування та надсилання до сервера; Отримання результату розпізнавання з серверної частини.

Модулі частини, яка містить представлення, описані в таблиці Б.3.

Таблиця Б.3 — Опис модулів частини, яка містить представлення

Модуль	Структура	Функції
Список ліків	class AllDrugsView	Відображення переліку ліків із застосованим сортуванням; Встановлення параметрів сортування; Виділення лікарського засобу та можливість його видалення або переходу до редагування; Перехід до додавання лікарського засобу; Індикація термінів придатності.
Додавання ліків	class DrugView	Введення інформації про лікарський засіб; Відображення кількості ліків, яка залишиться після курсу лікування; Збереження інформації (додавання до переліку ліків).
Перегляд ліків	class DrugView	Відображення інформації про ліки; Редагування інформації про лікарський засіб; Збереження (оновлення) інформації.

Продовження таблиці Б.3

Модуль	Структура	Функції
Розпізнавання	class ScannerView	Використання камери для створення зображення пакування; Відображення зображення; Відображення результату розпізнавання; Перехід до додавання лікарського засобу з використанням результатів розпізнавання.
Інформація	class InfoView	Відображення інформації про застосунок та розробника.

Серверна частина складається з 3 модулів: криптографії, розпізнавання та створення моделі для розпізнавання. Опис цих модулів наведено в табл. Б.4.

Таблиця Б.4 — Опис модулів серверної частини

Модуль	Структура	Функції
Створення моделі для розпізнавання	MLModel.py pretrained.py visualization.py	Попередня обробка зображень пакувань ліків; Створення структур моделей для розпізнавання; Навчання моделей на навчальній вибірці; Перевірка моделей на тестовій вибірці; Виведення статистики; Збереження створених моделей.
Розпізнавання	MLWebServer.py	Надсилення ключів шифрування до застосунку; Отримання із застосунку HTTP-запиту, який містить зображення пакування лікарського засобу; Розпізнавання ліків за допомогою створеної моделі; Надсилення результату до застосунку.
Криптографія	Cryptography.py	Розшифрування зображень.

Б.4 Використані технічні засоби

Мобільний застосунок написано на мові програмування C#. Для його створення було використане інтегроване середовище розробки Microsoft Visual Studio. При визначенні архітектури програмного забезпечення застосовувався шаблон проєктування Model-View-ViewModel. Для створення графічного інтерфейсу було використано фреймворк .NET MAUI.

Застосунок орієнтований на роботу в ОС Android, тому для коректної роботи програми необхідне стабільне функціонування ОС.

Серверна частина програми була написана на мові програмування Python з використанням бібліотек Flask, Tensorflow, Keras, NumPy, Pillow, Matplotlib, PyCryptodome. Для її створення також було використано Microsoft Visual Studio.

Серверна частина програми орієнтована на роботу в будь-якій операційній системі, яка підтримує Python та бібліотеки Flask, Tensorflow, Keras, NumPy, Pillow, Matplotlib, PyCryptodome.

Мінімальні вимоги до програмних та апаратних засобів для нормального функціонування мобільного застосунку наступні:

- операційна система — Android 5.0 і вище;
- процесор — 1,5 ГГц двоядерний процесор або вище;
- оперативна пам'ять — обсяг 2 Гб і більше;
- вільний дисковий простір — обсяг 250 Мб і більше;
- доступ до Інтернету;
- наявність камери.

Мінімальні вимоги до програмних та апаратних засобів для нормального функціонування серверної частини програми наступні:

- Python 3.9, Flask 2.2.3, Tensorflow 2.11.0, Keras 2.11.0, NumPy 1.23.5, Pillow 9.4.0, Matplotlib 3.7.2, PyCryptodome 3.19.0;
- процесор — двоядерний з частотою не менше 2 ГГц;
- оперативна пам'ять — обсяг 4 Гб і більше;
- вільний дисковий простір — обсяг 5 Гб і більше;
- доступ до Інтернету.

Б.5 Виклик і завантаження

Для початку роботи з програмним забезпеченням необхідно встановити мобільний застосунок за допомогою файлу com.panikat.medichest-

Signed.apk для ОС Android. Після цього на мобільному пристрої з'явиться іконка застосунку з назвою «Мої ліки». Натискання на іконку призводить до запуску програми.

Для роботи серверної частини на ній необхідно запустити файл MLWebServer.py

Б.6 Початкові та вихідні дані

Вхідними даними програми є інформація про лікарські засоби, яка зберігається в базі даних, зображення, які використовуються в графічному інтерфейсі.

Вихідними даними програми є інформація про лікарські засоби, збережена в базу даних.

ДОДАТОК В
Текст програми

В.1 Текст файла MLWebServer.py

```
# -*- coding: utf8 -*-
from flask import Flask, request
import tensorflow as tf
import io
import numpy as np
from PIL import Image
from Cryptography import Cryptography

# Initialize the Flask application
app = Flask(__name__)
crypto = Cryptography()
model = tf.keras.models.load_model('MLOwn2')
classes = ['Гофен 200', 'Но-шпа', 'Ортофен-здоров'я форте',
'Флюколд-N', 'Фосфалюгель']

# Route http posts to this method
@app.route('/classify', methods=['POST'])
def classify():
    # Getting files from the request
    encrFile = request.files['encrImage']
    algFile = request.files['alg']
    if not encrFile or not algFile:
        return {'error': 'No file provided'}, 400

    # Reading the file as an image
    encrImg = io.BytesIO(encrFile.read())
    encrImg = encrImg.getvalue()
    alg = io.BytesIO(algFile.read())
    alg = alg.getvalue()
    alg = alg.decode()

    # Decryption
    decrImg = crypto.imgDecryption(alg, encrImg)

    # Image
    decrImg = Image.open(io.BytesIO(decrImg))
    decrImg = decrImg.convert('RGB')
    img = decrImg.resize((224, 224))

    # Passing the image to the predict function of
TensorFlow
    img_array = np.array(img)
    img_array = np.expand_dims(img_array, axis=0)
    prediction = model.predict(img_array)

    # Returning recognition results
    class_index = np.argmax(prediction)
    class_name = classes[class_index]
    confidence = prediction[0][class_index]
```

```

        return {'class': class_name, 'confidence':
float(confidence)}

@app.route('/keys', methods=['POST'])
def keys():
    algFile = request.files['alg']
    alg = io.BytesIO(algFile.read())
    alg = alg.getvalue()
    alg = alg.decode()
    if alg == 'AES':
        return crypto.keyAES + crypto.ivAES
    elif alg == 'RSA':
        public_key = crypto.keyRSA.publickey()
        return b'<RSAKeyValue><Exponent>' +
crypto.b64_enc(public_key.e, 3) + b'</Exponent><Modulus>' +
crypto.b64_enc(public_key.n, 256) + b'</Modulus></RSAKeyValue>'
    elif alg == 'Hybrid':
        keyFile = request.files['key']
        key = io.BytesIO(keyFile.read())
        key = key.getvalue()
        crypto.genRSAwithKey(key)
        cryptoKey = crypto.keyEncryption()
        return cryptoKey + crypto.ivAES

@app.errorhandler(400)
def bad_request(error):
    return {'error': 'Bad request'}, 404

# Start flask app
if __name__ == '__main__':
    app.run(port=8080)

```

B.2 Текст файлы Cryptography.py

```

import base64
import math
from Crypto.Cipher import AES, PKCS1_OAEP
from Crypto.PublicKey import RSA
from Crypto.Random import get_random_bytes

class Cryptography:
    def __init__(self):
        self.genAES()
        self.genRSA()

    def newAES(self):
        self.algAES = AES.new(self.keyAES, AES.MODE_CFB,
self.ivAES)

    def genAES(self):
        self.keyAES = get_random_bytes(32)

```

```

        self.ivAES = get_random_bytes(16)
        self.newAES()

    def newRSA(self):
        self.algRSA = PKCS1_OAEP.new(self.keyRSA)

    def genRSA(self):
        self.keyRSA = RSA.generate(2048)
        self.newRSA()

    def genRSAwithKey(self, key):
        self.keyRSA = RSA.import_key(key)
        self.newRSA()

    def b64_enc(self, n, l):
        n = n.to_bytes(l, 'big')
        return base64.b64encode(n)

    def imgDecryption(self, alg, img):
        if alg == 'AES' or alg == 'Hybrid':
            result = self.algAES.decrypt(img)
            self.genAES()
            return result
        elif alg == 'RSA':
            tmp = []
            for i in range(math.ceil(len(img) / 256)):
                tmp.extend(self.algRSA.decrypt(img[(i
256) : ((i + 1) * 256)]))
            self.genRSA()
            return bytes(tmp)

    def keyEncryption(self):
        self.genAES()
        tmp = self.algRSA.encrypt(self.keyAES)
        self.genRSA()
        return tmp

```

В.3 Текст файлу MLModel.py

```

import visualization
import pretrained
import tensorflow as tf
import numpy as np
from keras.models import Sequential
from keras import regularizers
from keras.callbacks import Callback
from keras.layers import Rescaling, Flatten, Dense,
Dropout, RandomFlip, RandomRotation, RandomZoom, Conv2D,
MaxPooling2D
from timeit import default_timer as timer

```

```

# For time count
class TimingCallback(Callback):
    def __init__(self, logs={}):
        self.logs=[]
    def on_train_begin(self, logs={}):
        self.logs=[]
    def on_epoch_begin(self, epoch, logs={}):
        self.starttime = timer()
    def on_epoch_end(self, epoch, logs={}):
        self.logs.append(timer()-self.starttime)

if __name__ == '__main__':
    # Start data
    epochs, batch_size = 35, 25
    image_size = (224, 224)
    data_dir, test_dir =
'D:/Labs/MOC/Data/Images/Dataset_all',
'D:/Labs/MOC/Data/Images/Dataset_test'
    pred_files = ['D:/Labs/MOC/Data/Images/flucold_n.jpg',
'D:/Labs/MOC/Data/Images/gofen_200.jpg',
'D:/Labs/MOC/Data/Images/no_shpa.jpg',
'D:/Labs/MOC/Data/Images/ortophen.jpg',
'D:/Labs/MOC/Data/Images/phosphalugel.jpg']
    pretrained_models, models = dict(), dict()
    use_ga = use_pretrained = False
    cb = TimingCallback()

    # Ask for using pretrained models
    print('Use pretrained models? (Y/N):')
    x = input()
    if x == 'y' or x == 'Y':
        use_pretrained = True

    # Load datasets from directories
    train_ds =
tf.keras.utils.image_dataset_from_directory(data_dir, seed=123,
image_size=image_size, batch_size=batch_size)
    val_ds =
tf.keras.utils.image_dataset_from_directory(test_dir, seed=123,
image_size=image_size, batch_size=batch_size)
    num_classes = len(train_ds.class_names)
    class_names = train_ds.class_names

    # Data augmentation and normalization
    data_augmentation = Sequential([
        RandomFlip('horizontal', input_shape=(*image_size,
3)),
        RandomRotation(0.1),
        RandomZoom(0.1),
        Rescaling(1./255)
    ])

    # Use buffered prefetching

```

```

AUTOTUNE = tf.data.AUTOTUNE
train_ds =
train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)

# Create models
models["Own"] = Sequential([
    data_augmentation,
    Conv2D(32, (3, 3), padding='same',
activation='elu'),
    MaxPooling2D((2, 2), strides=2),
    Conv2D(64, (3, 3), padding='same',
activation='elu'),
    MaxPooling2D((2, 2), strides=2),
    Conv2D(128, (3, 3), padding='same',
activation='elu'),
    MaxPooling2D((2, 2), strides=2),
    Conv2D(128, (3, 3), padding='same',
activation='elu'),
    MaxPooling2D((2, 2), strides=2),
    Flatten(),
    Dense(256, activation='elu',
kernel_regularizer=regularizers.l2(0.03)),
    Dropout(0.7),
    Dense(128, activation='elu',
kernel_regularizer=regularizers.l2(0.03)),
    Dropout(0.7),
    Dense(num_classes, activation='softmax')
])

if use_pretrained == True:
    pretrained.pretrained_models(models, image_size,
num_classes, data_augmentation)

for name, model in models.items():
    # Compile the model
    model.compile(optimizer='adam',
loss='sparse_categorical_crossentropy', metrics=['accuracy'])
    model.summary()

    # Train the model
    history = model.fit(
        train_ds,
        validation_data=val_ds,
        epochs=epochs,
        batch_size=batch_size,
        callbacks=[cb]
    )

    # Evaluate the model
    score = model.evaluate(val_ds, verbose=0)

    # Make prediction

```

```

        ind_preds = []
        for pred_file in pred_files:
            image =
tf.keras.preprocessing.image.load_img(pred_file,
target_size=image_size)
            image =
tf.keras.preprocessing.image.img_to_array(image)
            image = tf.expand_dims(image, 0)
            preds = model.predict(image)
            print(preds)
            index = np.argmax(preds)
            print(index)
            ind_preds.append([index, preds[0][index]])

# Visualize training results
visualization.show_results(history, name,
class_names, cb, score, ind_preds)

# Ask for saving
print('Save the model? (Y/N):')
x = input()
if x == 'y' or x == 'Y':
    model.save('ML' + name)

```

B.4 Текст файлы Cryptography.cs

```

using System.Security.Cryptography;
using System.Text;

namespace MedicineChest.Models {
    internal class Cryptography {
        public Aes AlgAES { get; set; }
        public RSACryptoServiceProvider AlgRSA {get; set;}
        public Cryptography()
        {
            AlgAES = Aes.Create();
            AlgAES.Mode = CipherMode.CFB;
            AlgAES.Padding = PaddingMode.None;
            AlgRSA = new RSACryptoServiceProvider(2048);
        }

        public void SetKeys(byte[] key, string alg) {
            switch (alg) {
                case "AES":
                    AlgAES.Key = key.Take(2 * key.Length /
3).ToArray();
                    AlgAES.IV = key.Skip(2 * key.Length /
3).ToArray();
                    break;
                case "RSA":
                    string xml =
Encoding.UTF8.GetString(key, 0, key.Length);

```

```

        AlgRSA.FromXmlString(xml);
        break;
    case "Hybrid":
        byte[] cryptoKey =
key.Take(256).ToArray();
        AlgAES.Key = AlgRSA.Decrypt(cryptoKey,
true);

        AlgAES.IV = key.Skip(256).ToArray();
        break;
    }
}

public byte[] ImgEncryption(string alg, byte[] img)
{
    switch (alg) {
        case "Hybrid":
        case "AES":
            using (ICryptoTransform encryptor =
AlgAES.CreateEncryptor())
                return
encryptor.TransformFinalBlock(img, 0, img.Length);
        case "RSA":
            List<byte> tmp = new List<byte>();
            for (int i = 0; i <
Math.Ceiling(img.Length / 128.0); i++)

tmp.AddRange(AlgRSA.Encrypt(img.Skip(i
128).Take(128).ToArray(), true));
            return tmp.ToArray();
        default:
            return null;
    }
}
}
}

```

B.5 Текст файла ScannerViewModel.cs

```

using System.Net.Http.Headers;
using System.Net.Http.Json;
using System.Security.Cryptography;
using System.Text;
using System.Windows.Input;
using MedicineChest.General;
using MedicineChest.Models;

namespace MedicineChest.ViewModels {
    public class ScannerViewModel : Notifier
    {
        private string imageSrc;
        private string predictionResult;
    }
}

```

```

        private static readonly string alg = "Hybrid";
        private Cryptography crypto = new Cryptography();
        private static readonly string url =
"http://10.0.2.2:8080/classify";
        private static readonly string urlKey =
"http://10.0.2.2:8080/keys";
        public string ImageSrc
        {
            get { return imageSrc; }
            set { SetProperty(ref imageSrc, value); }
        }
        public string PredictionResult
        {
            get { return predictionResult; }
            set { SetProperty(ref predictionResult, value); }
        }
        public ICommand PackageCommand { get; private set; }
        public Func<string, string, Task> ErrorMessage {
get; set; }

        public ScannerViewModel()
        {
            PackageCommand = new Command( execute:
MakePrediction, canExecute: () => true);
        }

        public async void MakePrediction()
        {
            try
            {
                byte[] key = await TakePhoto();
                if (!string.IsNullOrEmpty(ImageSrc) && key
!= null)
                {
                    crypto.SetKeys(key, alg);
                    byte[] origImgBytes =
File.ReadAllBytes(ImageSrc);
                    byte[] encrImgBytes =
crypto.Encryption(alg, origImgBytes);

                    using var client = new HttpClient();
                    using var encrImgContent = new
ByteArrayContent(encrImgBytes);
                    using var algStrContent = new
StringContent(alg, Encoding.UTF8, "text/txt");
                    encrImgContent.Headers.ContentType =
MediaTypeHeaderValue.Parse("image/jpeg");
                    using var content = new
MultipartFormDataContent
                    {

```

```

        { encrImgContent, "encrImage",
"encrImage.jpg" },
        { algStrContent, "alg", "alg.txt" }
    };

    using var response = await
client.PostAsync(url, content);

    if (response.IsSuccessStatusCode)
    {
        var prediction = await
response.Content.ReadFromJsonAsync<Prediction>();
        PredictionResult =
prediction.Class;// + " | " + interval.TotalSeconds.ToString() +
"s.");// + ": " + prediction.Confidence.ToString();
    }
    else
    {
        if (ErrorMessage != null)
        {
            var error =
(int)response.StatusCode;

            if (error > 399 && error < 500)
await ErrorMessage.Invoke("Помилка розпізнавання", "Помилка на
стороні клієнта! Будь ласка, спробуйте пізніше або зверніться до
служби підтримки.");

            else if (error > 499 && error <
600) await ErrorMessage.Invoke("Помилка розпізнавання", "Помилка
на стороні сервера! Будь ласка, спробуйте пізніше або зверніться
до служби підтримки.");

            else await
ErrorMessage.Invoke("Помилка розпізнавання", "Помилка! Будь
ласка, спробуйте пізніше або зверніться до служби підтримки.");
        }
        PredictionResult = "Помилка!";
    }
}
else throw new Exception();
}
catch (Exception)
{
    await ErrorMessage.Invoke("Помилка
розпізнавання", "Помилка! Будь ласка, спробуйте пізніше або
зверніться до служби підтримки.");
    PredictionResult = "Помилка!";
}
}

private async Task<byte[]> TakePhoto()
{
    Clear();
    if (MediaPicker.Default.IsCaptureSupported)
    {

```

```

        FileResult photo = await
MediaPicker.Default.CapturePhotoAsync();
        if (photo != null)
        {
            string localFilePath =
Path.Combine(FileSystem.CacheDirectory, photo.FileName);

            using Stream sourceStream = await
photo.OpenReadAsync();
            using FileStream localFileStream =
File.OpenWrite(localFilePath);

            await
sourceStream.CopyToAsync(localFileStream);
            ImageSrc = localFilePath;
        }
        using var client = new HttpClient();
        using var algStrContent = new
StringContent(alg, Encoding.UTF8, "text/txt");
        using var content = new
MultipartFormDataContent { { algStrContent, "alg", "alg.txt" }
};
        if (alg == "Hybrid")
        {
            crypto.AlgRSA = new
RSACryptoServiceProvider(2048);
            byte[] publickey =
crypto.AlgRSA.ExportRSAPublicKey();
            var keyContent = new
ByteArrayContent(publickey);
            keyContent.Headers.ContentType =
MediaTypeHeaderValue.Parse("text/txt");
            content.Add(keyContent, "key",
"key.txt");
        }
        using var response = await
client.PostAsync(urlKey, content);
        if (response.IsSuccessStatusCode)
            return await
response.Content.ReadAsByteArrayAsync();
        else return null;
    }
    else return null;
}

public void Clear()
{
    PredictionResult = string.Empty;
    ImageSrc = string.Empty;
}
}

```

ДОДАТОК Г
Слайди презентації



Рисунок Г.1 — Слайд № 1

Мета, об'єкт, предмет дослідження

- **Об'єкт дослідження** – процес обліку лікарських засобів універсальної аптечки.
- **Предмет дослідження** – методи обліку лікарських засобів універсальної аптечки.
- **Мета роботи** – створення програмного забезпечення, яке надає можливість обліку лікарських засобів універсальної аптечки, тобто дозволяє зберігати інформацію про наявні лікарські засоби, наприклад, їхні назви, кількість, терміни придатності тощо.

2

Рисунок Г.2 — Слайд № 2

Огляд наявних аналогів

Можливості	«Мої лікарства: універсальна аптечка»	«Медицинский справочник-аптека»	«Аптека. Учет медикаментов»	Власний застосунок
Багатоплатформність	+	–	+	+
Українська локалізація	–	–	–	+
Сортування ліків	+	+	+	+
Фільтрування ліків	–	+	+	–
Групування ліків	+	+	+	+
Пошук ліків	+	–	+	–
Експорт даних	+	–	+	–
Створення декількох аптечок	–	+	+	–
Сканування штрих-коду	–	–	+	–
Відслідковування курсу лікування	–	–	–	+
Сповіщення	+	+	+	+
Розпізнавання ліків за пакуваннями	–	–	–	+

3

Рисунок Г.3 — Слайд № 3

Вибір мови програмування

Для мобільного застосунку

Критерії порівняння	Java	Kotlin	Swift	C#
Зрозумілість синтаксису	–	+	+	+
Підтримка спільноти	+	+	+	+
Простота вивчення	–	+	+	+
Підтримка IDE	+	+	+	+
Бібліотеки сторонніх розробників	+	+	+	+
Підтримка багатопоточності	+	+	+	+
Підвищена продуктивність	+	+	+	+
Розробка під Android	+	+	–	+
Розробка під iOS	–	–	+	+
Кросплатформна розробка	–	+	–	+
Інструментарій та екосистема	+	+	+	+

Для серверної частини

Критерії порівняння	Java	C#	C++	Python
Зрозумілість синтаксису	–	+	–	+
Підтримка спільноти	+	+	+	+
Простота вивчення	–	+	–	+
Підтримка IDE	+	+	+	+
Бібліотеки сторонніх розробників	+	+	+	+
Підтримка багатопоточності	+	+	+	+
Підвищена продуктивність	+	+	+	±
Вебфреймворки	+	+	–	+
Бібліотеки машинного навчання	–	+	+	+
Кросплатформна розробка	+	+	+	+
Інструментарій та екосистема	+	+	+	+

4

Рисунок Г.4 — Слайд № 4

Вибір середовища розробки

Критерії порівняння	Android Studio	Xcode	Eclipse	Visual Studio
Кросплатформність	–	–	+	+
Автозавершення коду	+	+	+	+
Підтримка репозиторіїв	+	+	+	+
Простота встановлення	+	+	–	+
Перевірка та аналіз коду	+	+	+	+
Швидкий та безпечний рефакторинг коду	+	+	+	+
Наявність зручного користувацького інтерфейсу	+	+	–	+
Швидкий та зручний пошук	+	+	+	+
Підсвічування синтаксису	+	+	+	+
Підтримка пакетів	+	+	+	+
Підтримка C#	–	–	+	+
Підтримка Python	–	–	+	+

5

Рисунок Г.5 — Слайд № 5

Вибір системи управління базами даних

Критерії порівняння	MySQL	PostgreSQL	MongoDB	SQLite
Функціональність	+	+	+	+
Швидкодія	+	+	+	+
Масштабованість	+	+	+	–
Надійність	+	+	–	+
Зручність використання	+	+	–	+
Підтримка SQL	+	+	–	+
Вбудованість	–	–	–	+
Сумісність з мобільними платформами	+	+	+	+
Відкритість та спільнота	+	+	+	+

6

Рисунок Г.6 — Слайд № 6

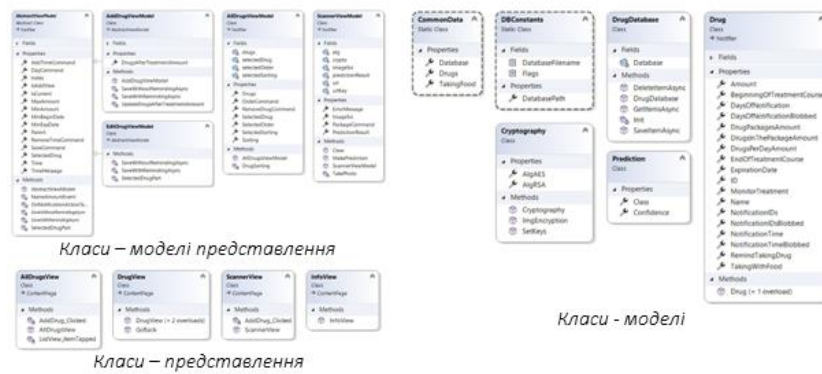
Схема структури програми



7

Рисунок Г.7 — Слайд № 7

Діаграми створених класів



8

Рисунок Г.8 — Слайд № 8

Схема функціонування програми



9

Рисунок Г.9 — Слайд № 9

Концептуальна модель бази даних



10

Рисунок Г.10 — Слайд № 10

Створені моделі для розпізнавання

Модель	Час навч. (с)	Втрата тест.	Точність тест. (%)	Розмір (МБ)	К-сть правильних класів прикладів
Owl	230,4	3,63	92	78,3	5 з 5
InceptionV3	157,2	1,86	100	97,5	5 з 5
Xception	243,8	0,55	96	90,8	4 з 5
ResNet50V2	213,9	1,02	96	101	5 з 5
MobileNetV2	103,0	0,91	100	18,0	5 з 5
NASNetMobile	193,4	0,83	100	39,9	5 з 5
DenseNet169	646,5	1,17	100	67,7	5 з 5



Приклади для перевірки моделей

11

Рисунок Г.11 — Слайд № 11

Впроваджені методи шифрування

Критерії порівняння	AES	RSA	<u>AES та RSA</u>
Ключі	Один спільний ключ	Приватний і публічний ключі	Комбінація ключів
Розмір ключа	256 біт	2048 біт	256 та 2048 біт
Швидкодія	1,09 с.	51,47 с.	1,44 с.
Складність управління ключами	Висока	Низька	Низька
Передача ключів	Складна	Легка	Легка
Обчислювальні витрати	Менші	Більші	Середні

12

Рисунок Г.12 — Слайд № 12

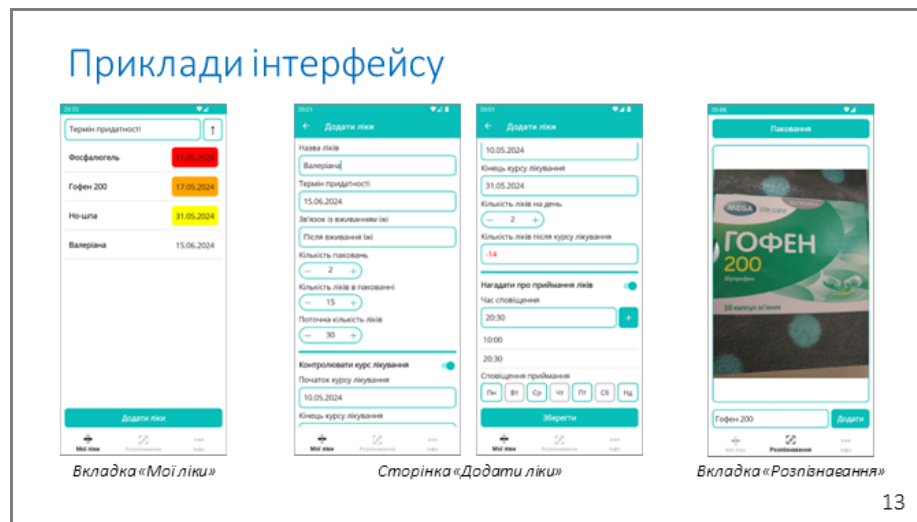


Рисунок Г.13 — Слайд № 13

Висновки

- В створеному програмному забезпеченні було реалізовано усі заплановані функції, крім сповіщень про наближення терміну придатності до закінчення та необхідність отримання рецепту від лікаря для придбання лікарського засобу. Тестування програми довело, що увесь розроблений функціонал працює коректно.
- В подальшому можливо покращити створене програмне забезпечення за допомогою:
 - додавання сповіщень про необхідність придбання лікарського засобу через те, що воно скоро закінчиться;
 - реалізації сканування штрих-кодів або серійних номерів;
 - покращення моделі розпізнавання за пакуваннями, механізмів забезпечення безпеки та розширення переліку розпізнавання;
 - надання можливості перегляду інструкцій для застосування лікарських засобів тощо.

14

Рисунок Г.14 — Слайд № 14

Наукові публікації та досягнення у творчих конкурсах

- I місце в університетському конкурсі студентських наукових робіт 2023/2024 н. р. (секція «Комп'ютерні науки, програмні засоби, системний аналіз та інформаційні технології», номінація «Програмні засоби»).
- III місце у першому турі Всеукраїнського конкурсу студентських наукових робіт зі штучного інтелекту 2023/2024 н. р.
- Фіналістка Конкурсу інноваційних стартап проєктів «Sikorsky Challenge 2023» (24 — 27 жовтня 2023 р.).
- Публікація тез «Програмне забезпечення обліку лікарських засобів універсальної аптечки» у збірнику тез доповідей щорічної науково-практичної конференції серед студентів, викладачів, науковців, молодих учених і аспірантів «Тиждень науки-2024».

15

Рисунок Г.15 — Слайд № 15