

Міністерство освіти і науки України  
Одеський національний політехнічний університет  
Навчально-науковий інститут комп'ютерних систем  
Кафедра системного програмного забезпечення

*Степаненко Денис Сергійович,*  
студент групи НАС-163

## **КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА**

*Мобільне застосування для розпізнавання символів догляду  
за текстильними виробами*

Спеціальність:  
121 – Інженерія програмного забезпечення

Спеціалізація:  
Інженерія програмного забезпечення

Керівник:  
*Тройніна Анастасія Сергіївна,*  
канд. техн. наук, доцент

## ЗМІСТ

ВСТУП .....	8
1 СПЕЦИФІКАЦІЯ ВИМОГ ДО ПРОГРАМНОГО ПРОДУКТУ .....	10
1.1 Визначення проблеми користувача.....	10
1.2 Опис предметної області .....	10
1.3 Аналіз можливих рішень.....	16
1.4 Аналіз існуючих аналогів.....	19
1.5 Функціональні вимоги.....	27
1.6 Нефункціональні вимоги.....	33
2 ПЛАН ВИКОНАННЯ ПРОЕКТУ.....	38
2.1 Оцінка тривалості розробки.....	38
2.2 План виконання робіт .....	41
2.3 Аналіз ризиків проекту.....	43
3 ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ .....	46
3.1 Опис архітектури та модулів проекту.....	46
3.2 Структура організації даних .....	47
3.3 Проектування нейронної мережі .....	50
3.4 Проектування контролеру .....	52
3.5 Проектування інтерфейсу користувача .....	56
4 ПРОГРАМНА РЕАЛІЗАЦІЯ РОЗРОБЛЮВАНОЇ СИСТЕМИ.....	67
4.1 Формування набору даних та навчання нейронної мережі .....	67
4.2 Реалізація нейронної мережі.....	74
4.3 Реалізація контролеру.....	76
4.4 Реалізація інтерфейсу користувача .....	79

5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ .....	88
5.1 Матриця відповідності вимог та сценарії тестування .....	88
5.2 Експеримент .....	91
6. ОХОРОНА ПРАЦІ .....	93
6.1 Загальна характеристика робочого місця .....	93
6.2 Загальний стан охорони праці на фірмі з інформаційних послуг .....	96
6.3 Аналіз умов праці на робочому місці програміста проектної групи .....	97
6.4 Індивідуальне завдання. Розрахунок інтегральної бальної оцінки важкості праці ( $I_p$ ) на робочому місці програміста .....	100
6.5 Заходи з охорони праці. Рекомендації .....	104
6.6 Обґрунтування запропонованих заходів з охорони праці .....	106
6.7 Висновки .....	107
ВИСНОВКИ .....	108
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ .....	109
ДОДАТОК А. ЛІСТИНГ ПРОГРАМИ .....	113

Міністерство освіти і науки України  
Одеський національний політехнічний університет  
Навчально-науковий інститут комп'ютерних систем  
Кафедра системного програмного забезпечення

Рівень вищої освіти: ПЕРШИЙ (бакалаврський)

Спеціальність: 121 – Інженерія програмного забезпечення

Спеціалізація: Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри

\_\_\_\_\_ Крісілов В.А.

«\_\_» \_\_\_\_\_ 20\_\_ р.

**ЗАВДАННЯ**  
**НА КВАЛІФІКАЦІЙНУ РОБОТУ**

*Степаненко Денис Сергійович, група НАС-163*

1. Тема роботи: *Тема роботи за наказом*

Керівник роботи: *Тройніна Анастасія Сергіївна, канд. техн. наук, доцент*  
затверджені наказом ректора від «17» березня 2020 р. № 158-в.

2. Зміст роботи: формування вимог до програмного продукту, планування виконання робіт, проектування програмної системи, програмна реалізація розроблюваної системи, тестування мобільного застосунку, заходи з охорони праці.

3. Перелік ілюстративного матеріалу: згідно зі слайдами презентації.

## 4. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
6	Москалюк А.Ю., доц.	11.05.2020	22.05.2020

## 5. Дата видачі завдання: «11» травня 2020 р.

## КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання	Примітка
1	Аналіз предметної області та існуючих рішень	03.02.2020 – 21.02.2020	виконав
2	Специфікація вимог до програми	24.02.2020 – 13.03.2020	виконав
3	Проектування програмної системи	16.03.2020 – 10.04.2020	виконав
4	Програмна реалізація та тестування системи	13.04.2020 – 08.05.2020	виконав
5	Завдання з охорони праці	11.05.2020 – 22.05.2020	виконав
6	Оформлення пояснювальної записки та графічного матеріалу	25.05.2020 – 01.06.2020	виконав

Здобувач вищої освіти \_\_\_\_\_ Д. С. Степаненко

Керівник роботи \_\_\_\_\_ А. С. Тройніна

## ЗАВДАННЯ

на розробку розділу «Охорона праці»

*Степаненка Дениса Сергійовича, група НАС-163*

Навчально-науковий інститут комп'ютерних систем

Кафедра системного програмного забезпечення

Тема роботи: Мобільне застосування для розпізнавання символів догляду за текстильними виробами

Зміст розділу:

1. Загальна характеристика робочого місця
2. Загальний стан охорони праці на фірмі з інформаційних послуг
3. Аналіз умов праці на робочому місці програміста проектної групи
4. Індивідуальне завдання. Розрахунок інтегральної бальної оцінки важкості праці на робочому місці програміста
5. Заходи з охорони праці. Рекомендації
6. Обґрунтування запропонованих заходів з охорони праці
7. Висновки

Керівник роботи

Консультант з охорони праці

\_\_\_\_\_ А. С. Тройніна

\_\_\_\_\_ А. Ю. Москалюк

«\_\_» \_\_\_\_\_ 20\_\_ р.

«\_\_» \_\_\_\_\_ 20\_\_ р.

## АНОТАЦІЯ

Метою цієї роботи є зниження затрат часу на тлумачення правил догляду за речами шляхом автоматичного розпізнавання символів догляду за ними.

Основними інструментами розробки є мова програмування Python, хмарна технологія автоматичного машинного навчання Google Cloud Vision та фреймворк розробки користувацьких інтерфейсів Kivy.

Як результат було створено практичне багатоплатформне мобільне застосування, яке пропонує новий підхід до вирішення конкретної задачі в галузі комп'ютерного зору.

Ключові слова: автоматичне машинне навчання, глибинне навчання, багатоміточна класифікація, комп'ютерний зір, Google Cloud AutoML, Google Cloud Vision, Kivy, Material Design.

## ABSTRACT

The aim of this study is to reduce the time spent on interpreting the textile care labels by means of automatic recognition of the corresponding care symbols.

The underlying development tools are the Python programming language, the Google Cloud Vision automated machine learning cloud technology and the Kivy user interface development framework.

As a result, a practical cross-platform mobile application was made, suggesting a new approach to solving a specific problem in a computer vision area.

Keywords: automated machine learning, deep learning, multi-label classification, computer vision, Google Cloud AutoML, Google Cloud Vision, Kivy, Material Design.

## ВСТУП

Ще з давніх часів люди носять одяг та користуються різноманітними текстильними виробами. Одяг завжди був базовою потребою кожної людини, на рівні з їжею, – тисячі років він надає людям тепло, безпеку, підвищує функціональність, сигналізує про соціальний статус тощо.

У наш час ця потреба вже давно не стоїть під питанням – одяг має більшість населення планети. Кількість унікальних матеріалів, з яких його створюють, за різними джерелами становить від 50 до 90 одиниць [1], які поділяються на відносно велику кількість категорій [2].

Крім різноманітності, так само й значно виросла кількість речей на одну людину, особливо в розвинених країнах. В 2016 американська компанія ClosetMaid провела опитування серед 1000 жінок в США, результати якого свідчать про те, що середньостатистична американка має 103 речі в своєму гардеробі [3].

Окремим, але в той же час пов'язаним, фактом варто привести, що 11 мільйонів тон речей, взуття та текстильних матеріалів, які підлягають переробці, опиняються на сміттєзвалищах щорічно, і це число з кожним роком зростає [4].

З оглядом на це, слід наголосити на тому, що речі, за якими правильно доглядають, не викидаються і слугують своїм власникам довше. Така, на першу думку, проста складова побуту несе за собою руйнівні наслідки в світовому масштабі. Цей негативний вплив особливо помітно в бідних країнах з текстильної промисловістю, в яких, нажаль, в результаті й опиняється більша частина відходів.

Підсумовуючи, актуальність проблеми полягає в тому, що догляд за речами – це одночасно питання і економічних заощаджень, і зовнішнього вигляду, і гігієни, і впливу на екологію, а значить має неабиякий вплив на життя в цілому.

Розмірковуючи про можливі причини неправильного догляду за речами одразу виникає думка про відносну складність інтерпретації інструкцій догляду за ними певними групами людей. Звісно, це зумовлено самою потребою в особливому підході до різних матеріалів.



Виникає потреба вирішення цього питання. Загальновідомо, що технологічний прогрес покращує рівень життя та робить його довшим – цей факт не підлягає оскарженню [5]. В наш час велику роль в цьому процесі грає сфера інформаційних технологій, спеціалісти з якої щоденно крок за кроком вирішують і вивчають існуючі проблеми людства.

Тому цілком логічно розглядати застосування методів з цієї області до вищезазначеного питання. Конкретніше, це є нагодою для використання рішень у рамках комп'ютерного зору. Релевантними успішними прикладами їх застосування є системи аналізу відео з камер спостереження, сканери QR- та штрих-кодів, розпізнавання тексту з документів, аналіз рентгенівських знімків тощо.

Слід зазначити, що ця галузь є доволі новою в контексті штучного інтелекту, в той же час будучи для неї фундаментальною [6]. Ці й сумісні з ними теми сьогодні вважаються одними із найскладніших для дослідження, все ще інтенсивно вивчаються вченими. Одна з причин цьому – парадокс Моравека: «на високому рівні міркування потребує відносно малих обчислень, але з низьким рівнем сенсомоторних навичок вимагає величезних обчислювальних ресурсів» [7].

Але потреба в використанні комп'ютерного зору тільки більшає: як наслідок, в останні декілька років набирає популярності новітній підхід – автоматичне машинне навчання. Ця технологія бере на себе найскладніші аспекти розробки і експлуатації штучного інтелекту задля облегшення і пришвидшення його інтеграції в повсякденне життя [8].

Аргументовано актуальністю проблеми і можливостями її вирішення методами сфери комп'ютерного зору, основним завданням цієї роботи є розробка багатоплатформного мобільного застосунку з функцією розпізнавання символів догляду за речами.

Робота складається зі вступу, специфікації вимог, плану виконання роботи, проектування, програмної реалізації, тестування, охорони праці та висновків.

## **1 СПЕЦИФІКАЦІЯ ВИМОГ ДО ПРОГРАМНОГО ПРОДУКТУ**

### **1.1 Визначення проблеми користувача**

У дослідженні, яке проводилося у Державному університеті Колорадо, як експеримент розроблялася та тестувалася на студентах програма відповідального догляду за речами [9]. Результати опитувань до та після експерименту показали, що молодь загалом не дуже освічена у питаннях догляду за речами, а також що навчальні ініціативи є ефективним методом покращення знань, хоч і доволі складним у впровадженні.

Виходячи з цього, логічно зробити припущення, що молодь в роки початку самостійного життя – як правило, в перші роки навчання в університеті – може бути основною групою людей, для яких інтерпретація символів викликає труднощі. Багато з них не має достатнього досвіду в догляді за речами, а в епоху колосальної кількості інформації та сенсорного перевантаження [10] деякі можуть вважати це не вартим належної уваги.

Крім цього, у роботі науковців з Лондонського університету мистецтв було зазначено, що більшість споживачів ігнорує інструкції, які зазначено символами на етикетках, сортуючи свій одяг для прання по кольору, а не по матеріалу виробу. Як наслідок це призводить до пошкодження речей [11], що є ще одним підтвердженням поширеності проблеми.

Перед тим як проводити аналіз можливих рішень, треба детально розглянути предметну область – самі символи.

### **1.2 Опис предметної області**

В 1963 році було засновано міжнародну асоціацію по розробці символів догляду за текстильними виробами GINETEX [12], головний офіс якої знаходиться в Парижі. Асоціацією була розроблена спеціальна система для маркування текстильних виробів, яка використовується в багатьох країнах світу і затверджена стандартом. Вона надає споживачам інформацію щодо правильного догляду за

речами. Складається з зареєстрованих у 22 країнах товарних знаків, є власністю асоціації. Остання ревізія стандарту була проведена Міжнародною організацією зі стандартизації, знайти її можна за кодом ISO3758:2012 [13].

Нажаль, не всі виробники у світі використовують цей стандарт, оскільки для його комерційного використання потрібно платити річний внесок та проходити сертифікацію. Через уникнення проблем із законом на етикетках часто опиняються або незначні варіації символів, або змішані символи з різних стандартів, або застарілі символи, які офіційно вже не використовуються.

Крім цього, існують ще й стандарти інших країн: Японії та США [14]. Детально буде розглядатись, а також розпізнаватись в застосунку, тільки стандарт асоціації GINETEX, оскільки він є міжнародним та найбільш поширеним.

Варто зазначити, що з 1973 року існувала ще канадська система, яка у 2003 році була замінена на міжнародну.

У 2015 році Японія відмовилася від японського стандарту (JIS L0217-1995) і приєдналася [15] до списку країн, в яких притримуються стандарту ISO3758:2012 (в Японії JIS L0001-2014). Приклад порівняння двох стандартів представлено на рис. 1.1.

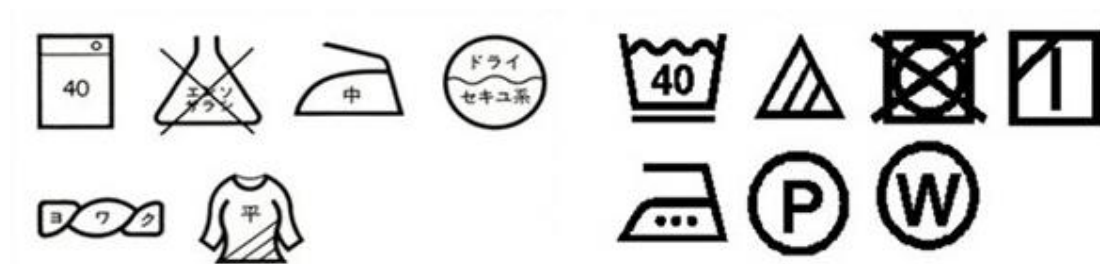


Рисунок 1.1 – Застарілі японські (зліва) та сучасні міжнародні символи (справа)

Американські ж символи не так значно відрізняються від міжнародних, як попередні японські. Приклад їх порівняння представлено на рис. 1.2.



Рисунок 1.2 – Американські (зверху) та міжнародні символи (знизу)

Тепер розглянемо символи стандарту GINETEX детальніше; далі для уникнення повторень стандарт зазначатися не буде.

Всього символів в стандарті 43. Вони поділяються на наступні п'ять категорій і розміщуються на етикетках саме в такому порядку:

- прання (12 шт.);
- відбілювання (3 шт.);
- сушка (13 шт.);
- прасування (4 шт.);
- професійний догляд (11 шт.), для скорочення далі – хімчистка.

Для кожної з категорій є базовий символ – певна форма, обрамлення, на базі якого будуються його варіації. Ці символи представлені на рис. 1.3.



Рисунок 1.3 – Базові символи стандарту GINETEX;  
зліва направо: прання, відбілювання, сушка,  
прасування, хімчистка


Треба зазначити, що базові символи відбілювання та барабанної сушки використовуються самі по собі. Базові символи звичайної (небарабанної) сушки та хімчистки окремо майже не використовуються. Базові символи прання та прасування окремо не використовуються ніколи.

Далі розглянемо самі символи. Інформацію та зображення символів візьмемо з офіційного сайту [16]. Всі права зберігаються за правовласником.

Одна риска під символом – це модифікатор, який означає делікатний догляд, дві – особливо делікатний. По аналогії з крапками у символах – їх кількість означає той чи інший режим експлуатації побутового приладу (наприклад, праски).




На табл. 1.1 зображено й описано семантично згруповані символи прання.

Таблиця 1.1 – Символи прання

Символ	Значення
	Прання заборонено
	Ручне прання, температура до 40°C
	Звичайне, делікатне та особливо делікатне прання при температурі води до 30°C
	Звичайне, делікатне та особливо делікатне прання при температурі води до 40°C
	Звичайне та делікатне прання, при температурі води до 60°C
	Звичайне прання, температура до 70°C
	Звичайне прання, температура до 95°C







На табл. 1.2 зображено й описано символи відбілювання.

Таблиця 1.2 – Символи відбілювання

Символ	Значення
	Відбілювання заборонено
	Можна відбілювати засобами без вмісту хлору
	Можна відбілювати будь-якими засобами

На табл. 1.3 зображено й описано символи сушки.





Таблиця 1.3 – Символи сушки

Символ	Значення
	Барабанну сушку заборонено
	Барабанну сушку дозволено
	Звичайна барабанна сушка, температура до 60°C
	Делікатна барабанна сушка, температура до 80°C
	Звичайну сушку дозволено
	Сушка у вертикальному положенні
	Сушка у вертикальному положенні в тіні
	Сушка без віджиму у вертикальному положенні
	Сушка без віджиму у вертикальному положенні в тіні
	Сушка у горизонтальному положенні
	Сушка у горизонтальному положенні в тіні
	Сушка без віджиму у горизонтальному положенні
	Сушка без віджиму у горизонтальному положенні в тіні

Можемо помітити, що направлення rischi в квадраті відповідає позиції, в якій рекомендується сушити виріб – вертикально чи горизонтально. Дві rischi в квадраті означають, що виріб не можна вижимати. Діагональна риска у верхньому лівому куту означає сушку в тіні.

На табл. 1.4 зображено й описано символи прасування.










Таблиця 1.4 – Символи прасування

Символ	Значення
	Прасування заборонено
	Прасування при температурі до 110°C
	Прасування при температурі до 150°C
	Прасування при температурі до 200°C

На більшості прасок маркування температурних режимів відповідає кількості крапок на відповідних символах.

У табл. 1.5 розглянуто останню категорію – хімчистку.

Таблиця 1.5 – Символи хімчистки

Символ	Значення
	Мокру професійну чистку заборонено
	Звичайна мокра професійна чистка
	Делікатна мокра професійна чистка
	Особливо делікатна мокра професійна чистка
	Сушу професійну чистку заборонено
	Звичайна суха професійна чистка з використанням вуглеводнів
	Делікатна суха професійна чистка з використанням вуглеводнів
	Звичайна суха професійна чистка з використанням перхлоретилену
	Делікатна суха професійна чистка з використанням перхлоретилену

Розглянувши всі символи, у підтвердження визначеної в попередньому підрозділі проблеми наведемо більш детальну інформацію. На табл. 1.6 представлено результати дослідження, проведеного у 2019 році міжнародною дослідницькою компанією Ipsos у семи країнах Європи. У дослідженні взяло участь 6000 осіб віком від 18 до 65 років [17].

Таблиця 1.6 – Відсоткова частка європейців, які правильно асоціюють символ з відповідним правилом догляду

Прання	Відбілювання	Сушка	Прасування	Хімчистка
91%	33%	32%	97%	21%

Крім цього, серед учасників тільки 25% осіб розрізняють символи з рискою та без риси. Крапки в символах сушки або прасування розрізняє 54% опитаних.

Тепер розглянемо можливі підходи до вирішення зазначених проблем.

### 1.3 Аналіз можливих рішень

На даний момент існує всього одна наукова робота, метою якої є розпізнавання символів догляду за речами. Це стаття науковців з Чеського технічного університету [18], яку було опубліковано в рамках Міжнародної конференції з аналізу та розпізнавання документів (ICDAR) – однієї з найбільш провідних у цій галузі – у вересні 2019 року. Робота, серед всього іншого, базується на методах оптичного розпізнавання символів (англ. optical character recognition або OCR), а точніше на використанні нейронних мереж згорткового типу.

Підходи, використані у вищезгаданій роботі, є комбінованими:

- розпізнавання тексту, який описує правила догляду за виробом, та його співставлення з відомими комбінаціями слів зі спеціального багатомовного словника;

- розпізнавання графічних образів символів;

- розпізнавання складу волокна, з якого виготовлено виріб, із використанням вищезгаданого багатомовного словника.

Нейронна мережа була навчена на наборі даних, створеним авторами роботи по причині відсутності такого у відкритому доступі. Набір даних складається з 63 фотографій етикеток [19].

Результати роботи чеських науковців наступні (адаптований переклад): «Експерименти показують, що проблема розпізнавання етикеток для догляду за одягом є складною навіть для найсучасніших моделей. За допомогою запропонованого методу вдалося досягнути розпізнавання символів з такою точністю: склад волокна – 96,8%, відбілювання – 67,7%, прасування – 64,5%, сушка – 74,2%, прання – 58,1%.»

На рис. 1.4 представлено приклад роботи вищеописаної системи.





Рисунок 1.4 – Розпізнавання тексту та символів на етикетці

Можна було би зробити припущення, що вищеописані методи не є достатньо ефективними для вирішення проблеми. Але зрозуміло, що висновки робити ще зарано – і не тільки через те, що ця робота є першою в своєму роді.

Наприклад, можливо, що результати описаного вище підходу стануть кращими при розширенні набору даних. Ще важливо зазначити, що в роботі не була вказана кількість унікальних символів, які охоплюються системою. Також можливо, що різноманітність символів з 63 фотографій не є оптимальною для підтвердження або спростовування гіпотез.

Тому на даному етапі варто розглянути якомога більше варіантів вирішення проблеми – це допоможе визначити найбільш ефективні методи для подальшої роботи, а також для майбутніх досліджень і експериментів.

Одним з альтернативних підходів є використання методів розпізнавання об'єктів (англ. object detection/object recognition) [20]. Цей підхід вимагає грамотного виділення ознак за допомогою машинного або глибинного навчання. Але такі підходи потребують більших наборів даних, мають залежність від архітектури моделей та ще ряду факторів, тому досягнення плідних результатів в розглядуваному питанні вимагає неабиякого досвіду та ресурсів.

Можливо також виділення ознак за допомогою ручної графічної розмітки зображень – обрамлення об'єктів та/або їх найбільш інформативних ознак. Але при збільшенні набору даних кількість зусиль тільки на перевірку гіпотези ефективності методу розпізнавання об'єктів була би зовеликою в рамках цієї роботи. На кожній етикетці знаходиться 5 символів, тож навіть якщо виділяти

тільки їх, а не – більш детально – їх складові , то для набору даних з 63 фотографій це вже вимагало би 315 обрамлень. Навіть якщо не брати до уваги питання проектування і реалізації нейронної мережі, можна точно сказати, що цей метод вимагає відносно великих затрат часу на формування набору даних.

Натомість, потенційно ефективним методом для вирішення проблеми є багатоміточна класифікація (англ. multi-label classification [21], узгодження щодо сталого перекладу українською наразі немає). На рис. 1.5 представлено візуальне пояснення відмінностей цього методу від звичайної багатокласової класифікації (англ. multi-class classification) [22].

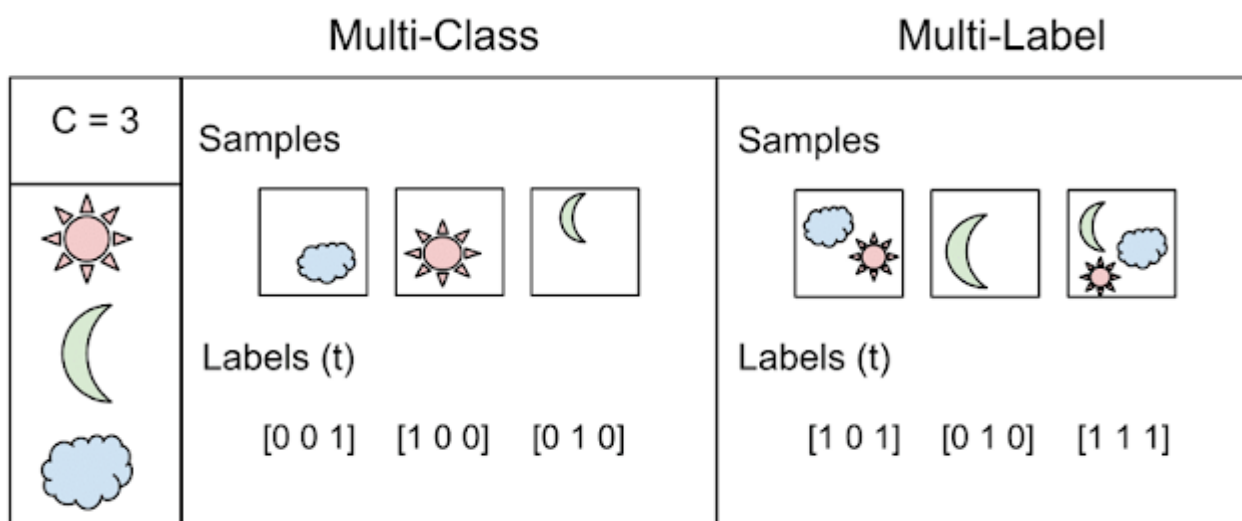


Рисунок 1.5 – Відмінності між багатокласовою (зліва) та багатоміточною (справа) класифікацією

Цей варіант також має доволі високу варіацію можливостей виділення ознак. Для дослідження гіпотези і досягнення мети цієї роботи він може бути найбільш ефективним через те, що потребує меншої кількості ресурсів на реалізацію, особливо у комбінації з автоматичним машинним навчанням, яке було описано раніше.

Для повноти наостанок важливо зазначити, що на всесвітньовідомій платформі для змагань з науки про дані Kaggle по розглядуваній темі ще вдалося знайти приватне змагання, організоване турецькою компанією Vestel у 2019 році

[23, 24]. Код програм учасників та набір даних із 1000 фотографій, який вони використовували, не було опубліковано.

Перейдемо до огляду існуючих рішень – мобільних застосунків, які пов’язані з темою цієї роботи.

#### **1.4 Аналіз існуючих аналогів**

Для початку проведемо невеликий аналіз формату, в якому вищеописана функція розпізнавання символів буде поставлятися до кінцевих користувачів.

Програмне забезпечення, яке призначене для вирішення розглядуваної проблеми, має бути доступним для якомога більшої кількості людей. Очевидним і практично єдиним рішенням, яке задовільнило би цю умову, є розробка мобільного застосунку. За даними багаточисельних досліджень, станом на початок 2020 року частка людей, які володіють смартфоном, складає приблизно 44,98% – 3,5 мільярдів людей [25]. Станом на травень 2020 року частка смартфонів, які працюють на операційній системі Android становить 72,6%, а частка iOS – 26,72% [26]. Загалом на ці дві операційні системи припадає 99,32% смартфонів.

Також важливо враховувати застарілі та невідтримувані версії цих операційних систем. В той же час, якщо взяти до уваги ціну пральної машини і зіставити її із ціною смартфонів на базі iOS та відносною дешевизною й поширеністю смартфонів на базі Android, виходить, що доступ до сучасного мобільного застосунку серед власників побутової техніки для догляду за речами будуть мати майже всі. Саме тому обраний формат постачання – мобільний застосунок – є надзвичайно ефективним.

Далі для зручності подальшого опису і порівняння дамо застосунку ім’я, яке максимально точно описувало би його призначення. Нехай це буде «Laundry Scan»: англійською ця назва влучно передає призначення програмного забезпечення, при цьому залишаючись в достаток короткою.

Також слід визначитися з короткою характеристикою застосунку. Laundry Scan – це безкоштовний багатоплатформний мобільний застосунок, який містить

список символів догляду за речами та дозволяє розпізнавати їх за допомогою камери.

Тепер можемо перейти до порівняння аналогів.

За результатами пошуку було знайдено чотири мобільних застосунки для смартфонів на базі ОС Android та два для iOS.

Декілька Android застосунків були дуже схожими між собою, тому для порівняння обрано було тільки унікальні.

Всього в порівнянні розглядалося чотири застосунки: по два на кожную мобільну платформу.

#### 1.4.1 Застосунок «Laundry Pro»

На рис. 1.6 представлено скріншоти програми, взяті з відповідної сторінки в крамниці застосунків Google Play [27].

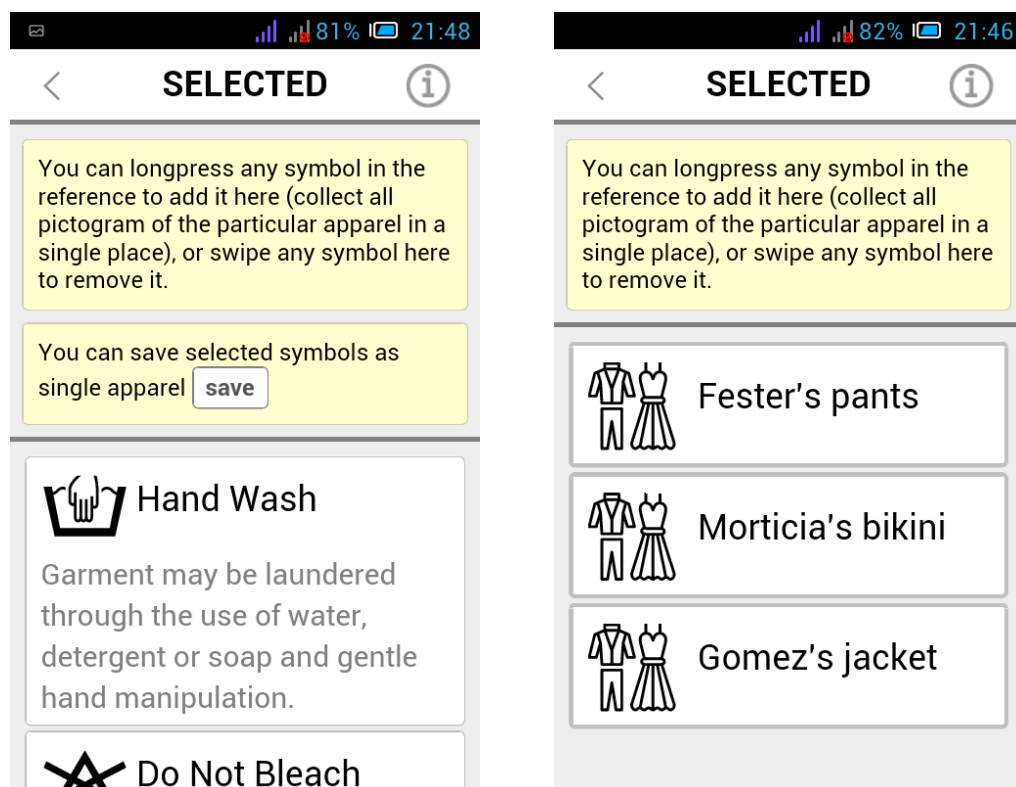


Рисунок 1.6 – Скріншоти роботи застосунку Laundry Pro

Основні характеристики:

- розроблено для ОС Android;
- поширюється безкоштовно, містить рекламу;
- містить список символів з поясненнями;
- функції розпізнавання символів не має.

Хоча застосунок і має додаткову функцію зберігання наборів символів до списку, загальний набір можливостей доволі обмежений, застосунок містить рекламу та має застарілий інтерфейс, який потребує багато клацань для пошуку та консолідації інформації.

#### 1.4.2 Застосунок «My Care Label»

На рис. 1.7 представлено скріншоти роботи програми, які було взято з відповідної сторінки в Google Play [28].

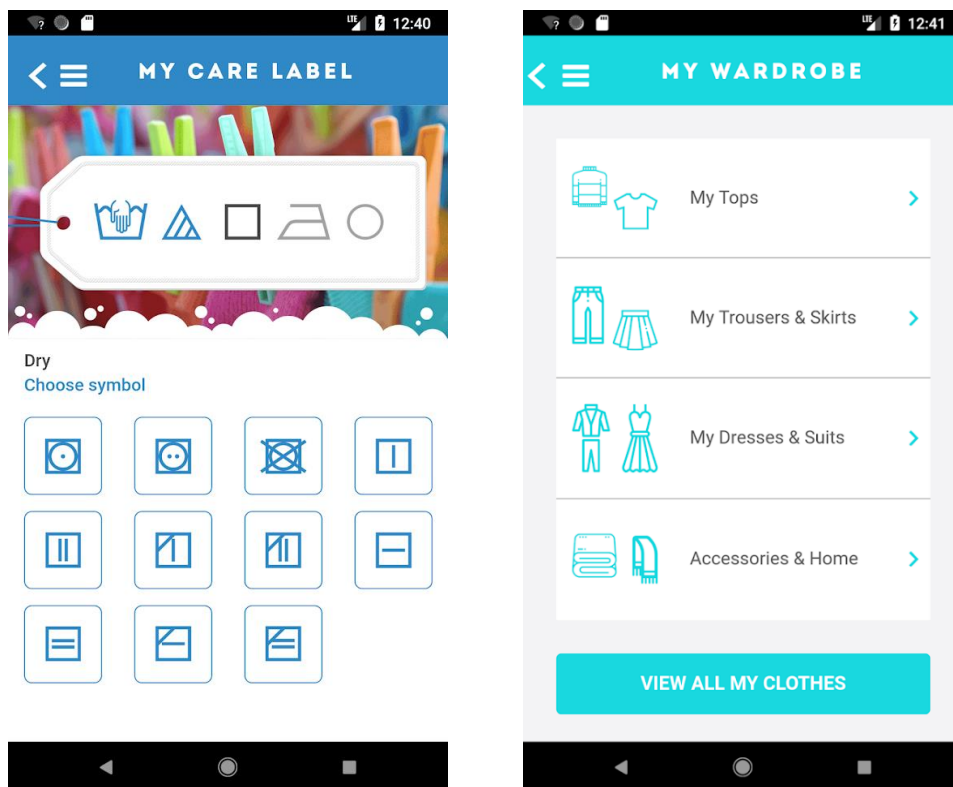


Рисунок 1.7 – Скріншоти роботи застосунку My Care Label

Основні характеристики:

- розроблено для ОС Android;
- поширюється безкоштовно, не містить реклами;
- містить список символів з поясненнями;
- має функцію зберігання наборів символів до списку;
- функції розпізнавання символів не має.

Крім вже згаданого, хоча це прямо і не пов'язано з цією роботою, варто зазначити, що застосунок містить загальні рекомендації щодо догляду за речами – наприклад, видалення плям. Цю функцію показано на рис. 1.8.

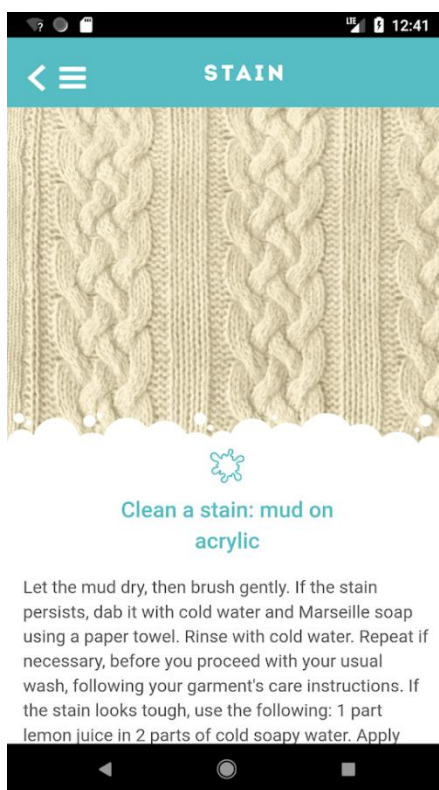


Рисунок 1.8 – Рекомендації щодо догляду за речами у застосунку My Care Label

Набір можливостей доволі широкий, застосунок поширюється безкоштовно, не містить реклами, має відносно сучасний інтерфейс, додаткові функції та, крім відсутності функції сканування символів, очевидних мінусів не має.

### 1.4.3 Застосунок «Laundry Day»

На рис. 1.9 представлено скріншоти роботи вбудованого сканера для розпізнавання символів, які було взято з відповідної сторінки в App Store [29].



Рисунок 1.9 – Робота функції сканування в застосунку Laundry Day

Основні характеристики:

- розроблено для ОС iOS;
- поширюється платно, за ціною \$0,99;
- містить список символів з поясненнями;
- має функцію розпізнавання символів.

Набір можливостей відносно широкий, інтерфейс оригінальний та сучасний, але застосунок є платним. Варто зазначити, що серед відгуків є як і позитивні, так і негативні, і останні нерідко стосуються функції розпізнавання символів. Немає відгуків новіше 2016 року, застосунок не оновлювався з першого випуску, який відбувся у 2015 році. Сайт автора наразі недоступний.



На рис. 1.10 представлено скріншот екрану символів.



Рисунок 1.10 – Екран символів у застосунку Laundry Day

#### 1.4.4 Застосунок «Complete Laundry Care»

Основні характеристики:

- розроблено для ОС iOS;
- поширюється безкоштовно, не містить реклами;
- містить список символів з поясненнями;
- функції розпізнавання символів не має.

Крім цього, серед додаткових функцій застосунку є ще рекомендації по догляду за речами, а також таймер для прання та сушки, який не налаштовується.

Набір можливостей доволі широкий, інтерфейс трохи негармонічний – використовується комбінація застарілих і сучасних рішень.

На рис. 1.11 наведено скріншот роботи мобільного застосунку, який було взято з відповідної сторінки в App Store [30].





Рисунок 1.11 – Екран символів у застосунку Complete Care Label

На рис. 1.12 представлено додаткові функції, які згадувались раніше.

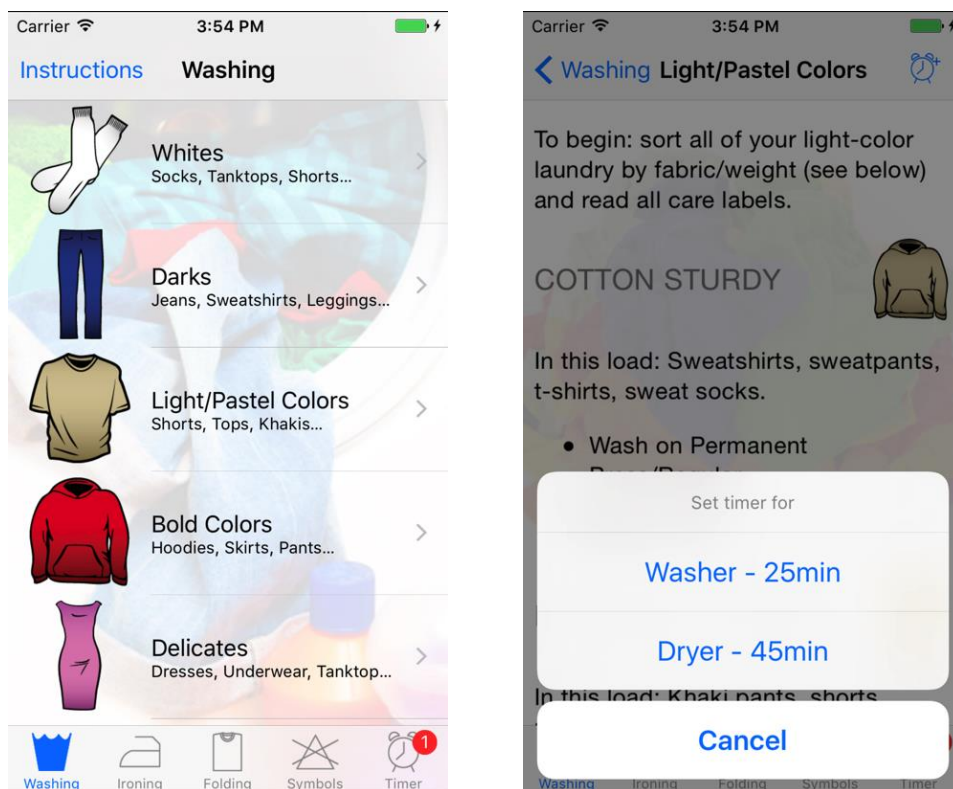


Рисунок 1.12 – Додаткові функції застосунку Complete Care Label

Проаналізувавши всі чотири аналоги, занесемо результати до табл. 1.7.

Таблиця 1.7 – Порівняння аналогів

Назва Критерій	Laundry Pro	My Care Label	Laundry Day	Complete Laundry Care	<i>Laundry Scan</i>
Платформа	Android	Android	iOS	iOS	Android та iOS
Список символів	✓	✓	✓	✓	✓
Можливість сканувати символи	✗	✗	✓	✗	✓
Безкоштовний	Реклама	✓	✗	✓	✓

Можна зробити висновок, що мобільний застосунок Laundry Scan хоч і не містить додаткових функцій, але в цілому є найкращим серед аналогів. Він є: багатоплатформним, містить список символів, дозволяє сканувати символи та поширюється безкоштовно, не містить реклами.

Також застосунок буде мати найбільш сучасний серед аналогів інтерфейс, що теж є дуже важливо, бо впливає, наприклад, на швидкість навігації в інтерфейсі програми.

## 1.5 Функціональні вимоги

Функціональні вимоги описують те, що система повинна робити [31].

Для визначення функціональних пріоритетів скористуємося методом аналізу вимог MoSCoW [32]:

- Must have (Повинні зробити) – критичні та обов’язкові для проекту вимоги, які потрібно зробити в першу чергу;
- Should have (Бажано зробити) – важливі, але не критичні вимоги, які вже не так залежать від часу, коли їх буде реалізовано;
- Could have (Можна було би зробити) – покращення, які можна було би зробити за достатньої кількості ресурсів: часу, грошей тощо;
- Won’t have (Не будемо робити) – найменш критичні вимоги, які не впливають на успішність проекту або реалізація яких наразі не є доцільною.

У табл. 1.8 наведено функціональні вимоги до мобільного застосунку за цим методом.

Таблиця 1.8 – Функціональні вимоги проекту за методом MoSCoW

№	Функції	M	S	C	W
1	Можливість працювати на ОС Android та iOS	+			
2	Наявність списку символів	+			
3	Можливість працювати з камерою	+			
4	Можливість розпізнавати символи, які використовуються найчастіше	+			
5	Можливість розпізнавати всі символи стандарту		+		
6	Прийнятна швидкість розпізнавання	+			
7	Наявність рекомендацій по догляду за одягом			+	

Ресурси, яких потребує компіляція набору даних достатнього розміру для розпізнавання всіх символів, не дають змогу задовільнити вимогу №7.

Крім того, особливо схожі за сенсом (а значить і за виглядом) символи краще відображати разом задля підвищення точності результатів, бо через розмір набору даних точність моделі значно погіршиться для кожного з них.

Наприклад, символи звичайного, делікатного та особливо делікатно прання за однієї температури слід помітити для нейронної мережі одною міткою, а при відображенні результатів на екрані показувати для користувача всі три символи.

Наявність рекомендацій по догляду за одягом є доволі корисною для кінцевого користувача функцією, але в рамках цієї роботи вона розглядатися не буде через тривіальність її реалізації; те саме стосується і функції збереження набору символів.

Далі розглянемо варіанти використання на діаграмі прецедентів, яку також називають Use Case діаграмою.

На діаграмі прецедентів відображаються відношення між акторами та прецедентами в системі [33].

Цю діаграму використовують для наглядного зображення вимог до системи та їх відношень до інших елементів системи, при цьому реалізація та декомпозиція цих вимог не розглядається – вони мають бути якомога більш короткими та зрозумілими.

На діаграмі опишемо двох акторів: Користувача та Нейронну мережу.

Користувач може переглядати інформацію про застосунок, переглядати список символів або робити фотографію етикетки з символами.

Нейронна мережа робить передбачення по фотографії та намагається розпізнати на ній символи

Далі або показується помилка, або символи, які вдалося розпізнати.

На рис. 1.13 представлена Use Case діаграма мобільного застосунку Laundry Scan.

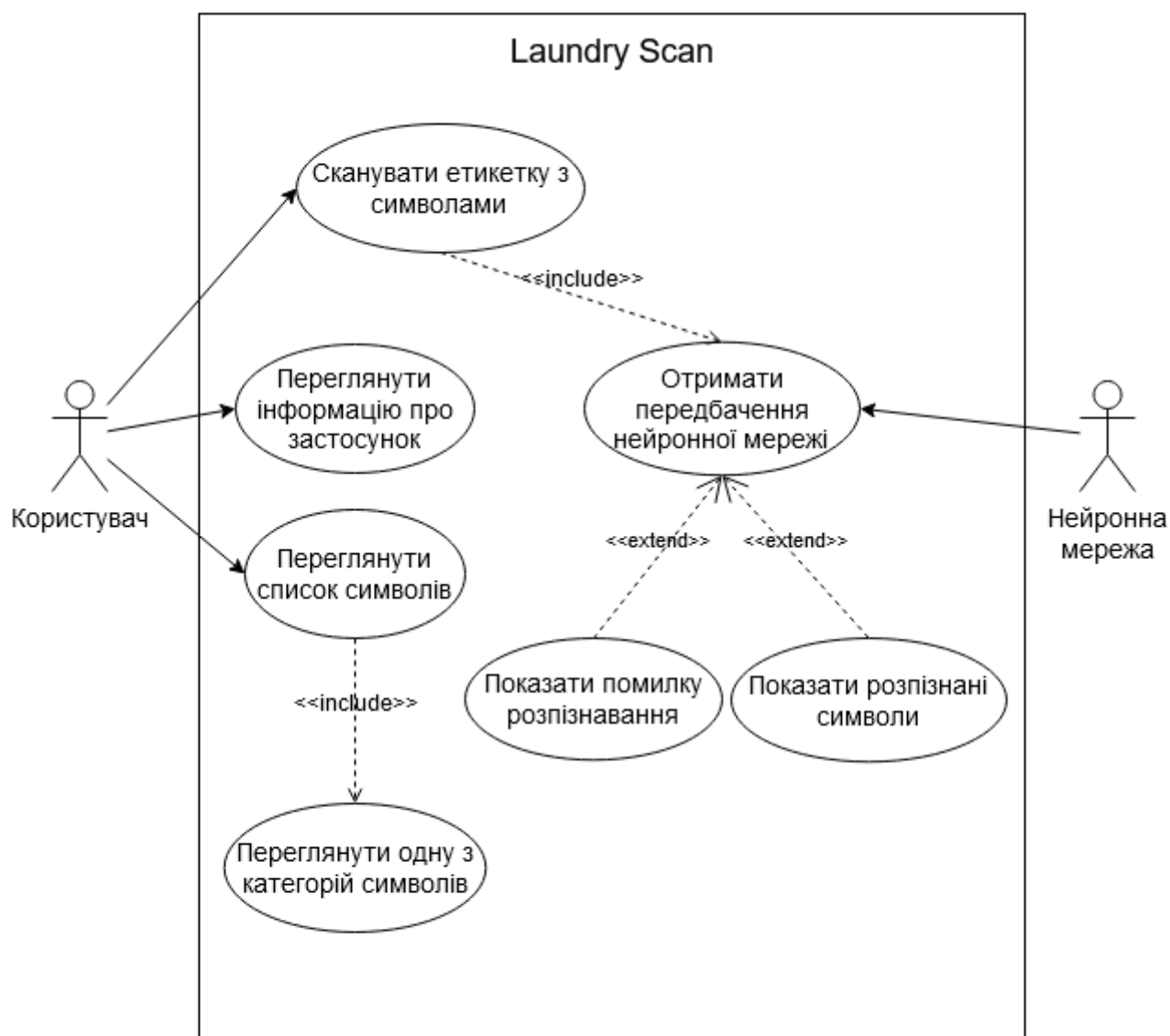


Рисунок 1.13 – Use Case діаграма застосунку

Опишемо кожний з варіантів використання.

У табл. 1.9 представлено опис прецеденту «Переглянути список символів».

Таблиця 1.9 – Прецедент №1. «Переглянути список символів»

Опис	Прецедент №1. Користувач переглядає список символів
Актори	Користувач
Передумова	Користувач бажає почати роботу із застосунком
Післяумова	Користувач переглядає одну з категорій символів

Продовження таблиці 1.9

Основний сценарій	Користувач відкриває мобільний застосунок і бачить перед собою головний екран, на якому відображається перша категорія символів. Це значить, що Прецедент №2 («Переглянути одну з категорій символів») вже було викликано.
-------------------	--

Між попереднім прецедентом і прецедентом «Переглянути одну з категорій символів» вказано відношення «include», що означає обов’язкове виконання прецеденту, в сторону якого направлено відношення. Цей прецедент описано у табл. 1.10.

Таблиця 1.10 – Прецедент №2. «Переглянути одну з категорій символів»

Опис	Користувач переглядає одну з категорій символів.
Актори	Користувач
Передумова №1	Користувач переглядає список символів.
Передумова №2	Користувач натиснув на кнопку, яка відображає відповідну категорію символів.
Основний сценарій	Користувач починає роботу у мобільному застосунку (Прецедент №1), автоматично починаючи переглядати першу категорію символів.
Альтернативний сценарій	Користувач бажає відкрити іншу категорію, натискає на одну з кнопок, отримуючи на дисплеї відповідні символи.

Наступним розглянемо прецедент «Переглянути інформацію про застосунок» (табл. 1.11).

Таблиця 1.11 – Прецедент №3. «Переглянути інформацію про застосунок»

Опис	Користувач переглядає інформацію про застосунок.
Актори	Користувач

Продовження таблиці 1.11

Передумова	Користувач натискає кнопку, яка відображає інформацію про застосунок.
Післяумова	Відображається інформація про застосунок.
Основний сценарій	Користувач натискає кнопку, яка відображає інформацію про застосунок. Інформація відображається.

Далі розглянемо основну функцію мобільного застосунку – розпізнавання символів. Відповідний прецедент описано у табл. 1.12.

Таблиця 1.12 – Прецедент №4. «Сканувати етикетку з символами»

Опис	Користувач бажає зробити фотографію і розпізнати символи на ній.
Актори	Користувач, Нейронна мережа
Передумова	Користувач натискає кнопку, яка відкриває інтерфейс камери, та робить фотографію.
Післяумова	Користувач зробив фотографію, Нейронна мережа дає передбачення по наявності на ній символів.
Основний сценарій	Користувач робить фотографію. Викликається Прецедент №5 ( «Отримати передбачення нейронної мережі»).

Через зв'язок «включення» прецедент «Отримати передбачення нейронної мережі» після виклику прецеденту №4 викликається автоматично. Його розглянуто у табл. 1.13.

Таблиця 1.13 – Прецедент №5. «Отримати передбачення нейронної мережі»

Опис	Система дає передбачення на наявність символів на фотографії.
------	---

Продовження таблиці 1.13

Передумова	Користувач забажав зробити фотографію, щоб Нейронна мережа розпізнала на ній символи.
Післяумова №1	На екрані відображаються знайдені на фотографії символи.
Післяумова №2	На екрані відображається помилка.
Основний сценарій	Нейронній мережі вдалося знайти символи на фотографії. Результати відображаються на екрані. Викликається Прецедент №7 («Показати розпізнані символи»).
Альтернативний сценарій	Нейронній мережі не вдалося знайти жодного символу на фотографії. На екрані відображається помилка. Викликається Прецедент №6 («Показати помилку розпізнавання»).

Оскільки прецедент «Отримати передбачення нейронної мережі» пов'язаний з прецедентами «Показати помилку розпізнавання» та «Показати розпізнані символи» зв'язком «розширення», при виклику основного прецеденту викликається один з двох дочірніх.

Прецедент «Показати помилку розпізнавання» описано в табл. 1.14.

Таблиця 1.14 – Прецедент №6. «Показати помилку розпізнавання»

Опис	Нейронна мережа відображає помилку розпізнавання.
Актори	Нейронна мережа
Передумова	Нейронна мережа не розпізнала на фотографії жодного символу.
Післяумова	На екрані відображається помилка.
Основний сценарій	Нейронній мережі не вдалося знайти жодного символу на фотографії. На екрані відображається помилка.

Прецедент «Показати розпізнані символи» описано в табл. 1.15.



Таблиця 1.15 – Прецедент №7. «Показати розпізнані символи»

Актори	Нейронна мережа
Опис	Нейронна мережа відображає розпізнані символи.
Актори	Нейронна мережа
Передумова	Нейронна мережа розпізнала певні символи на фотографії.
Післяумова	На екрані відображаються розпізнані символи.
Основний сценарій	Нейронній мережі вдалося знайти певні символи на фотографії. На екрані відображаються результати.

Сформулювавши функціональні вимоги, перейдемо до нефункціональних.

## **1.6 Нефункціональні вимоги**

### **1.6.1 Вимоги до середовища функціонування**

Смартфон, на якому буде працювати Laundry Scan, має працювати на ОС Android 5.0 і вище або на ОС iOS версії 9.0 і вище.

Камера смартфона має бути справною. Оскільки кожна з версій обох ОС має певні вимоги до апаратного середовища, за умови працюючої системи будь-яка камера на такому пристрої задовольняє умовам.

Для гарантованої роботи застосунку на смартфоні має бути мінімум 100 Мб вільного простору на внутрішній пам'яті.

### **1.6.2 Вимоги до умов експлуатації**

Для користування функцією сканування користувач має надати відповідний дозвіл застосунку.

Для підвищення результативності розпізнавання символів мають бути виконані наступні умови:

— фотографія зроблена при нормальному освітленні;

- фокус на символах;
- символи чітко видно;
- етикетка розташована по центру фотографії та займає її третину.

### **1.6.2 Вимоги до набору даних та моделі нейронної мережі**

Як вже було зазначено раніше, через те, що проблема недостатньо досліджена і ефективні методи її вирішення невідомі, треба покрити максимум випадків за мінімум ресурсів. Тому для виконання цих умов опишемо набір даних, який будемо використовувати для навчання нейронної мережі, і сформулюємо обмеження на очікувані результати:

- набір даних має містити 200 фотографій етикеток з символами;
- з них для навчання нейронної мережі використовуються тільки ті, символи на яких визначені міжнародним стандартом;
- фотографії, на яких погано видно символи, не використовуються;
- фотографії, які зустрічаються в наборі даних найрідше, задля збереження точності передбачень інших міток, використовуватися у навчанні не будуть [34];
- як було зазначено раніше, деякі схожі символи будуть помічені як однакові, а для користувача застосунок результати будуть «розгортатись» до двох-трьох символів;
- виходячи з відносно малого набору даних (хоча майже в три рази більшого, ніж в раніше проаналізованій науковій роботі), елементи навчального набору даних буде розділено не за класичною схемою 80% тренувальної вибірки (з яких 20% валідаційної) і 20% тестової вибірки, а за наступною схемою: 60% тренувальної, 20% валідаційної і 20% тестової вибірки [35];
- процес навчання мережі повинен бути високорівневим;
- навчання має проходити швидко;
- навчена модель має працювати на мобільних пристроях;
- точність правильного розпізнавання обраних символів має складати мінімум 75%;

– швидкість розпізнавання одної фотографії має не перевищувати одної секунди.

### **1.6.3 Вимоги до графічного інтерфейсу**

Графічний інтерфейс – дуже важлива частина цього проекту. Вимоги до нього наступні:

- має бути інтуїтивним;
- має містити якомога меншу кількість елементів;
- кількість клацань для навігації в застосунку має бути мінімальною;
- інтерфейс має бути сучасним і звичним для користувача;
- застосунок має правильно відображатися на різних пристроях та бути доступним для людей з вадами зору або для людей, які використовують збільшений шрифт на своїх пристроях;
- символи стандарту GINETEX мають виглядати гармонічно відносно один одного, бути розбірливими.

### **1.6.4 Вимоги до середовища та інструментів розробки**

Операційна система для написання коду та маніпуляцій з набором даних має бути сучасною та мати доступ до інтернету.

Операційна система для компіляції виконуваних файлів мобільного застосунку – сучасні GNU/Linux або macOS з доступом до інтернету.

Смартфон на базі Android версії 5.0 і вище або iOS версії 9.0 і вище для тестування застосунку та для створення набору даних за допомогою камери.

Для розробки і тестування мобільного застосунку має використовуватись інструментарій для роботи зі смартфоном через інтерфейс командного рядка.

Текстовий редактор та/або інтегроване середовище розробки має містити всі необхідні інструменти для роботи з обраною мовою програмування, а також містити можливості для зручної роботи з набором даним – розміткою фотографій.

Вимоги до мови програмування:

- має бути зручною і лаконічною в задачах маніпуляції даних;
- має містити готові інтерфейси для роботи з нейронними мережами;
- має бути швидкою для розробки;
- має мати інтуїтивний синтаксис, бути простою для читання та удосконалення коду;
- має бути швидкою при компіляції в байт-код;
- має бути багатоплатформною.

### **1.6.5 Обґрунтування обраних інструментів розробки**

Реалізація і експлуатація нейронної мережі:

- через свою багатофункціональність, документацію, швидкість та результативність – одна з найбільш розвинених хмарних технологій автоматичного машинного навчання Google Cloud AutoML, а саме її частина Vision Image Classification;

- хмарне сховище для набору даних Google Cloud Platform Storage – для роботи з Google Cloud AutoML Vision та зберігання набору даних;
- бібліотека машинного навчання та роботи з нейронними мережами TensorFlow Lite версії 1.10-1.14, а саме пакунок tfliteruntime, – для експлуатації навченої в хмарі моделі на смартфонах.

Реалізація графічного інтерфейсу:

- через сучасність, адаптивність, звичність для користувачів – система графічного дизайну Material Design;
- через багатоплатформність, гнучкість, зручність та швидкість розробки – фреймворк для розробки користувацьких інтерфейсів Kivu версії 2.0 і вище;
- для реалізації вказівок Material Design – бібліотека KivyMD;
- для відображення символів GINETEX – векторний шрифт з офіційного сайту асоціації, що дозволить змінювати розмір символів в інтерфейсі без зниження якості зображення.

Середовище та інструменти розробки:

- мова програмування Python (версія 3.6 та вище, бажано 3.8) – задовольняє всім умовам (компіляція в байт-код за допомогою інструментарію Cython);
- через швидку оболонку, потребу для компіляції виконуваних програм під Android і доступ до інструментів GNU/Linux – ОС Xubuntu
- VS Codium (базується на Visual Studio Code) – багатофункціональний текстовий редактор та інтегроване середовище розробки з великою кількістю необхідних інструментів та розширень;
- програмне забезпечення buildozer – для компіляції Python коду під різні платформи;
- програмне забезпечення для роботи зі смартфонами на ОС Android через інтерфейс командного рядка adb.

## 2 ПЛАН ВИКОНАННЯ ПРОЕКТУ

### 2.1 Оцінка тривалості розробки

Проведемо оцінку тривалості розробки за методом бальної оцінки варіантів використання – UCP (Use Case Points) [36], яка складається з наступних пунктів:

- а) нескоригована вага варіантів використання (Unadjusted Use Case Weight, UUCW);
- б) нескоригована вага акторів (Unadjusted Actor Weight, UAW);
- в) коефіцієнт технічної складності (Technical Complexity Factor, TCF);
- г) коефіцієнт складності навколишньої середовища (Environmental Complexity Factor, ECF).

Коли всі показники буде розраховано, дізнаємось UCP за наступною формулою:

$$UCP = (UUCW + UAW) * TCF * ECF \quad (2.1)$$

Для розрахунку UUCW спочатку занесемо вагу кожного з варіантів використання у таблицю 2.1:

Таблиця 2.1 – Ваги варіантів використання за кількістю транзакцій

Варіант використання	Кількість транзакцій	Вага
Переглянути список символів	2	5
Переглянути інформацію про застосунок	1	5
Зробити фотографію етикетки з символами	4	10

Розрахуємо нескориговану вагу за формулою:

$$UUCW = (n \text{ простих прецедентів} * 5) + (n \text{ середніх прецедентів} * 10) + (n \text{ складних прецедентів} * 15), \quad (2.2)$$

$$UUCW = (2 * 5) + (1 * 10) = 20$$

Далі розрахуємо UAW (табл. 2.2).

Таблиця 2.2 – Ваги акторів за складністю

Актор	Складність
Система	Простий
Користувач	Складний

$$\begin{aligned}
 UAW &= (n \text{ простих акторів} * 1) + (n \text{ середніх акторів} * 2) + \\
 &\quad (n \text{ складних акторів} * 3) , \\
 UAW &= 1 + 3 = 4
 \end{aligned}
 \tag{2.3}$$

Розрахуємо TCF: оцінимо технічні фактори, занесемо їх до табл. 2.3:

Таблиця 2.3 – Оцінка технічних факторів

Фактор	Опис	Вага	Оцінка
T1	Розподіленість системи	2	0
T2	Швидкість відгуку	1	5
T3	Ефективність кінцевого користувача	1	4
T4	Складність внутрішніх обчислень	1	2
T5	Можливість повторно використовувати код	1	3
T6	Простота інсталяції	0.5	0
T7	Простота використання	0.5	3
T8	Переносимість на інші платформи	2	5
T9	Простота в обслуговуванні	1	0
T10	Паралельні обчислення	1	0
T11	Засоби безпеки	1	0
T12	Доступ до третіх сторін	1	0
T13	Потреби користувача в навчанні	1	1

$$\begin{aligned}
 TF &= 5 + 4 + 2 + 3 + 0.5 * 3 + 2 * 5 + 1 = 23,5 , \\
 TCF &= 0,6 + (TF/100) = 0,835
 \end{aligned}
 \tag{2.4}$$

Аналогічно розрахуємо ECF (табл. 2.4):

Таблиця 2.4 – Оцінка зовнішніх факторів

Фактор	Опис	Вага	Оцінка
F1	Знайомство з використовуваним процесом розробки	1,5	2
F2	Досвід розробки подібного ПЗ	0,5	0
F3	Досвід команди в ООП	1	2
F4	Здібності головного аналітика	1,5	3
F5	Мотивація команди	1	3
F6	Стабільність вимог	1	3
F7	Часткова зайнятість членів команди	-1	2
F8	Складність мови програмування	-1	0

$$EF = 1,5 * 2 + 2 + 1,5 * 3 + 3 + 3 - 2 = 13,5,$$

$$ECF = 1,4 + (-0,03 * EF) = 0,995 \quad (2.5)$$

І нарешті підрахуємо UCP:

$$UCP = (UUCW + UAW) * TCF * ECF = 19,94 \quad (2.6)$$

Для того, щоб визначити очікувану тривалості розробки, потрібно знайти якій кількості робочих годин відповідає один UCP. Треба підрахувати кількість чинників з множини F1-F8 (табл. 3.4), оцінки яких за абсолютним значенням перевищують 3.

В даному випадку таких чинників два, отже якщо один UCP (за рекомендацією автора системи) дорівнює 20 робочим годинам, то можемо розрахувати очікувану тривалість розробки наступним чином:

$$AUCP = 2 * UCP * TCF * ECF,$$

$$AUCP = 2 * 20 * 0,835 * 0,995 = 33 \quad (2.7)$$

Очікувана тривалість розробки програмного продукту – 33 повних дні, 800 годин або 100 робочих днів за умови 8-ми годинного робочого дня.



## 2.2 План виконання робіт

Розділимо проект на більш дрібні задачі та представимо їх ієрархічно за допомогою підходу структури декомпозиції робіт (WBS). Оцінимо об'єм кожної з робіт і занесемо всю отриману інформацію до табл. 2.5.

Таблиця 2.5 – Структура декомпозиції та строки виконання робіт

№	Назва етапу	Початок	Кінець	Днів
1	Визначення вимог	27.03.2020	10.04.2020	14
1.1	Визначення проблеми користувача	27.03.2020	28.03.2020	1
1.2	Опис предметної області	28.03.2020	30.03.2020	2
1.3	Аналіз можливих рішень	30.03.2020	03.04.2020	4
1.4	Аналіз існуючих аналогів	03.04.2020	04.04.2020	1
1.5	Функціональні вимоги	04.04.2020	07.04.2020	3
1.6	Нефункціональні вимоги	07.04.2020	10.04.2020	3
2	Планування	10.04.2020	03.05.2020	23
2.1	Оцінка тривалості розробки	10.04.2020	11.04.2020	1
2.2	Планування виконання робіт	11.04.2020	13.04.2020	2
2.3	Аналіз ризиків	13.04.2020	14.04.2020	1
2.4	Визначення архітектури та опис модулів	14.04.2020	17.04.2020	3
2.5	Структура організації даних	17.04.2020	15.04.2020	2
2.6	Проектування нейронної мережі	15.04.2020	24.04.2020	5
2.7	Проектування контролеру	24.04.2020	28.04.2020	4
2.8	Проектування інтерфейсу користувача	28.04.2020	03.05.2020	5
3	Реалізація	03.05.2020	31.05.2020	28
3.1	Формування набору даних та навчання нейронної мережі	03.05.2020	10.05.2020	7

Продовження таблиці 2.5

№	Назва етапу	Початок	Кінець	Днів
3.2	Реалізація нейронної мережі	10.05.2020	16.05.2020	6
3.3	Реалізація контролеру	16.05.2020	24.05.2020	8
3.4	Реалізація інтерфейсу користувача	24.05.2020	31.05.2020	7
4	Тестування	31.05.2020	10.06.2020	10
4.1	Сценарії тестування	31.05.2020	02.06.2020	2
4.2	Експеримент	02.06.2020	10.06.2020	8
	Загальний обсяг	27.03.2020	10.04.2020	75

На рис. 2.1 представлено діаграму Ганта на основі даних з табл. 2.5.

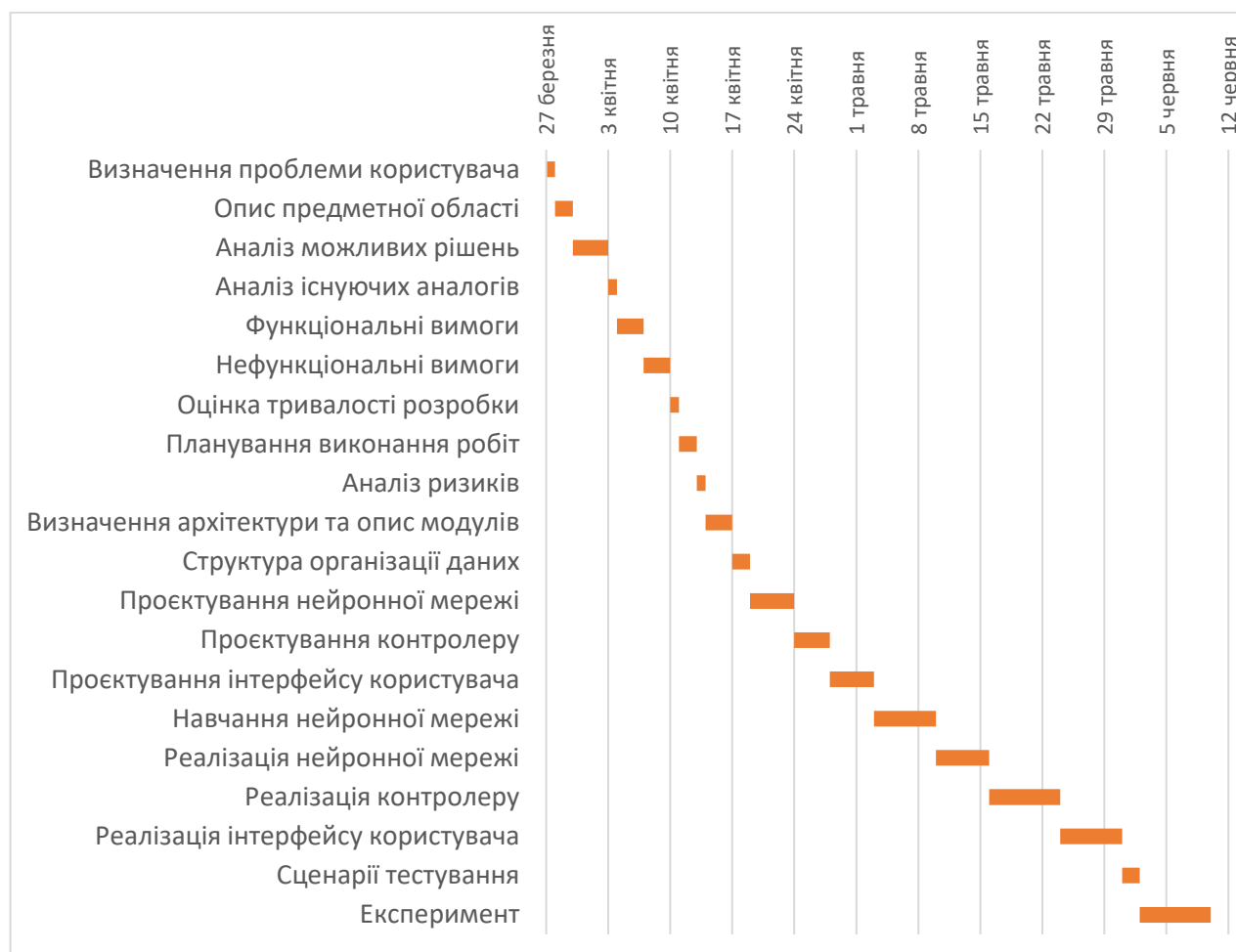


Рисунок 2.1 – Діаграма Ганта

На діаграмі можемо побачити, що часу на роботи проекту поступово витрачається все більше.

Основними причинами цього є використання експериментальних технологій та відсутність досвіду в галузі машинного навчання. Хоча, в той же час, подібний розподіл можна вважати нормальним через присутність в кожному проекті непередбачуваних факторів: тонкощів реалізації, упущень в плануванні тощо.

### 2.3 Аналіз ризиків проекту

Аналіз ризиків – важлива складова планування проекту. Розібравши їх вчасно, можна заощадити багато ресурсів та внести ясності у плани, специфікації.

Сформулюємо п'ять найбільших ризиків.

На рис. 2.2 описано ризик відсутності релевантного досвіду розробки. Управління ризиком – не завищувати вимоги до програмної системи.

Номер: R-1	Категорія: Технічні
Причина: Відсутність релевантного досвіду розробки	Умови: Розробка багатоплатформного мобільного застосування; експлуатація нейронних мереж
Наслідки: Система не працює належним чином	Вплив: Більші затрати часу на навчання та відловлювання помилок, менші – на розробку системи
Ймовірність: Дуже ймовірний	Ступінь впливу: Критична
Близькість: Дуже скоро	Ранг: 9
Вихідні дані: «план проекту», «вимоги»	

Рисунок 2.2 – Картка опису ризику R-1

На рис. 2.3 описано ризик неправильного оцінювання очікуваної тривалості розробки. Управління ризиком – аналіз ризиків та правильне формування вимог.

Номер: R-2	Категорія: Організаційні
Причина: Помилка в оцінці тривалості роботи	Умови: Розробка Laundry Scan
Наслідки: Система не відповідає вимогам	Вплив: Деморалізація, збільшення часу на розробку
Ймовірність: Дуже ймовірний	Ступінь впливу: Критична
Близькість: Дуже скоро	Ранг: 9
Вихідні дані: «план проекту», «вимоги»	

Рисунок 2.3 – Картка опису ризику R-2

На рис. 2.4 описано ризик неточного формулювання вимог до програмного продукту. Управління ризиком – збільшення часу на формування вимог, акцент на вимірюваність вимог.

Номер: R-3	Категорія: Технічні
Причина: Неточне формулювання вимог	Умови: Планування розробки програмного продукту
Наслідки: Система не буде готова в зазначений термін	Вплив: Збільшення трудомісткості розробки
Ймовірність: Середня ймовірність	Ступінь впливу: Критична
Близькість: Скоро	Ранг: 6
Вихідні дані: «вимоги», «план проекту»	

Рисунок 2.4 – Картка опису ризику R-3

На рис. 2.5 описано ризик неефективності обраної технології для розпізнавання символів. Управління ризиком – попереднє ознайомлення з можливостями та обмеженнями обраної технології.

Номер: R-4	Категорія: Технічні
Причина: Неефективна технологія машинного навчання	Умови: Розробка моделі нейронної мережі
Наслідки: Можливостей системи недостатньо для досягнення мети	Вплив: Низька результативність технології
Ймовірність: Середня ймовірність	Ступінь впливу: Середня
Близькість: Скоро	Ранг: 4
Вихідні дані: «програмна реалізація», «вимоги»	

Рисунок 2.5 – Картка опису ризику R-4

На рис. 2.6 описано ризик незацікавленості кінцевого користувача в розроблюваному продукті. Управління ризиком – опитування, більш детальне дослідження проблем користувачів, адаптація вимог.

Номер: R-5	Категорія: Зовнішні
Причина: Кінцевий користувач не зацікавлений у використанні програмного продукту	Умови: Тестування мобільного застосунку
Наслідки: Даремна витрата часу на розробку	Вплив: Марне витрачання ресурсів, деморалізація
Ймовірність: Низька ймовірність	Ступінь впливу: Середня
Близькість: Скоро	Ранг: 3
Вихідні дані: «тестування», «вимоги»	

Рисунок 2.6 – Картка опису ризику R-5

Проаналізувавши ризики, можна зробити висновок, що неправильно сформовані вимоги до програмного продукту часто можуть нести руйнівні наслідки для всього проекту.

### 3 ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

#### 3.1 Опис архітектури та модулів проекту

При розробці мобільного застосунку Laundry Scan доцільно буде використовувати архітектурний шаблон MVC (Model-View-Controller, Модель-Вигляд-Контролер).

В розроблюваній програмній системі буде використовуватись дещо змінений зв'язок між цими трьома компонентами. Зображення зв'язків між класами представлено на рис. 3.1.

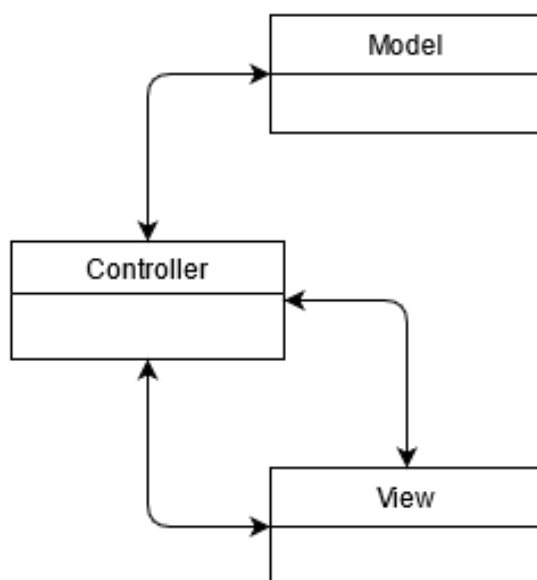


Рисунок 3.1 – Зв'язок між класами при використанні архітектурного шаблону MVC

Model працює тільки з даними: отримує вхідні дані, опрацьовує їх і вертає результати своєї роботи. В розглядуваній системі модель буде представлена нейронною мережею.

View відповідає за відображення цих даних та може мінімально контролювати стан інтерфейсу, якщо це доречно в контексті певної реалізації.

Controller фактично виконує функцію посередника між попередніми двома класами: може передавати повідомлення від користувача програми до різних її внутрішніх модулів, тим самим змінюючи стан всієї системи. Наприклад, цей клас за отримання певного сигналу може створювати або видаляти елементи інтерфейсу. Також Контролер може перекомпонувати дані, коли їх відображення тісно пов'язано з компонентами класу View – така комбінація є і більш зрозумілою для програміста, і менш затратною для швидкодії системи.

Модель буде представлено класом NeuralNet. Контролер – класом App. View – менеджером екранів графічного інтерфейсу – ScreenManager.

### 3.2 Структура організації даних

Дана програмна система не містить бази даних як такої, але для відображення символів, які є основним елементом, навколо якого функціонує програма, все ж потрібно зберігати певний статичний набір даних.

Згадаємо, що для відображення самих символів використовується векторний шрифт із гліфами, що в нашому випадку є графічними представленнями символів. На рис. 3.2 показане візуальне пояснення того, як це працює.



Рисунок 3.2 – Код літери q відображається як символ «особливо делікатне прання при температурі 40 градусів за Цельсієм»

Тому для відображення одного символу у відповідній йому категорії, нам треба знати про кожний символ наступне:

- категорію, до якої він належить;
- ім'я самого символу;
- літеру, яка відповідає за цей символ у шрифті;
- текстовий опис значення символу.

Одна з можливих структур такого переліку даних представлена за допомогою діаграми «сутність-зв'язок» (Entity-relationship diagram) на рис. 3.3.

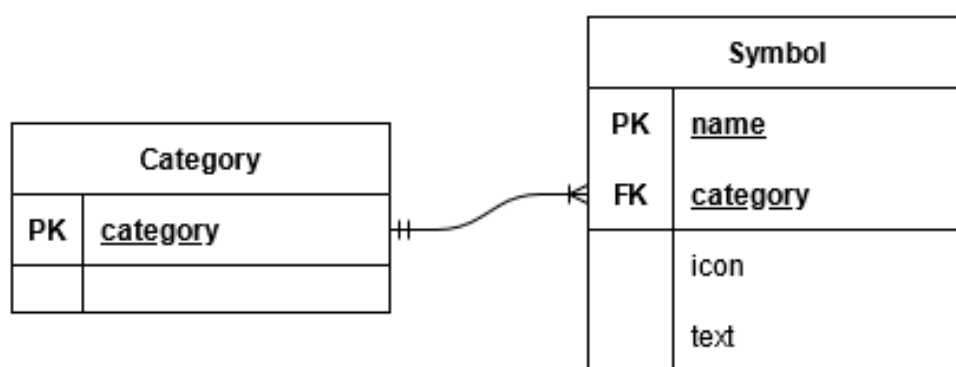


Рисунок 3.3 – ER-діаграма сутності категорії символів,  
зв'язок – «один до багатьох»

Тобто є багато категорій, в кожній з яких є багато символів і даних про них. Відповідні поля розглянуто в табл. 3.1.

Таблиця 3.1 – Поля сутності Symbol

Назва	Опис
category	Ім'я категорії символу
name	Ім'я символу
icon	Літера, яка відповідає гліфу символу у шрифті
text	Значення символу, його опис

Так само, як і для списку символів, для відображення треба зберігати і дані про категорії. Сутність даних про категорію наведена на рис. 3.4.



Category metadata	
PK	<u>name</u>
	icon
	text

Рисунок 3.4 – Сутність даних про категорію символів

Поля сутності розглянуто в табл. 3.2.

Таблиця 3.2 – Поля сутності Category metadata

Назва	Опис
name	Ім'я категорії для звертання до неї у коді
icon	Літера, яка відповідає гліфу категорії у шрифті
text	Ім'я категорії для відображення (українською)

Ця сутність відокремлена від попередніх, бо таким шляхом можна оминати багато зайвих операцій. Інформація про символи буде запрошуватись набагато частіше за категорії. Звісно, при роботі з текстом в наш час такі «навантаження» є майже непомітними, але подібний підхід на порядок облегшує розробку, бо задля отримання потрібного поля треба вказувати менше елементів в ієрархії.

Також потрібно представляти дані про фотографії, які будуть використовуватись для навчання нейронної мережі (див. рис. 3.5).

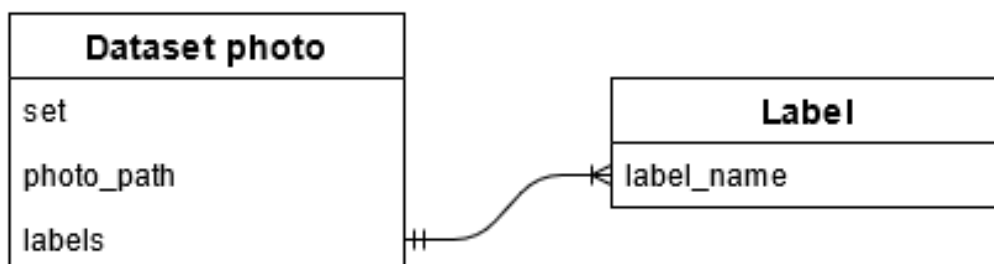


Рисунок 3.5 – ER-діаграма сутності фотографії з набору даних для навчання нейронної мережі, зв'язок – «один до багатьох»

Поля сутності розглянуто в табл. 3.3.

Таблиця 3.3 – Поля сутності Dataset photo

Назва	Опис
set	Тип набору даних, до якого входить фотографія (тренування, валідація, тестування)
photo_path	Шлях до фотографії
labels	Перелік міток – наявність відповідних символів на фотографії

Далі детально розглянемо кожний з трьох основних класів. Поля та методи буде описано менш детально через те, що вони тісно пов'язані з програмною реалізацією – особливо клас View.

### 3.3 Проектування нейронної мережі

Єдиною функцією класу нейронної мережі є розпізнавання символів на етикетці. Конкретніше, він приймає фотографію у вигляді матриці, в якій представлено по масиву пікселів на кожний з каналів RGB, і вертає найкращі результати – знайдені на зображенні символи категорій у скомпонованому вигляді.

Поля та методи класу показані на рис. 3.5.

NeuralNet
+ model_path: String + labels_path: String + threshold: Integer - _initialized: Bool - _interpreter: Object - _input_details: Object - _output_details: Object - _labels: List
- _init_(model_path, labels_path, threshold): None - _init_model(): None - _filter_results(prediction: numpy.ndarray): Dict[String, Dict] + make_prediction(img: numpy.ndarray): numpy.ndarray

Рисунок 3.5 – Поля та методи класу NeuralNet

Поля класу розглянуто в табл. 3.4.

Таблиця 3.4 – Поля класу NeuralNet

Назва	Опис
model_path	Шлях до файлу моделі нейронної мережі.
labels_path	Шлях до файлу з переліком міток, які підтримує модель.
threshold	Порогове значення (у відсотках) впевненості моделі у передбаченнях. Використовується для фільтрації результатів.
_initialized	Прапорець, який відображає чи створено об'єкт класу нейронної мережі. Використовується задля завантаження моделі тільки при першому звертанні до неї, а не під час запуску програми.
_interpreter	Об'єкт інтерпретатора моделі.
_input_details	Об'єкт першого шару нейронної мережі, її «вхід».
_output_details	Об'єкт останнього шару нейронної мережі, її «вихід».
_labels	Перелік підтримуваних моделлю міток. Завантажуються при створенні об'єкта нейронної мережі. з файлу, шлях до якого вказано у labels_path.

У табл. 3.5 розглянемо методи класу.

Таблиця 3.5 – Методи класу NeuralNet

Назва	Опис аргументів	Опис методу
__init__	Аргументи model_path, labels_path та threshold описані в табл. 3.4	Створює об'єкт класу і поля для внутрішнього використання.
_init_model	—	Створює інтерпретатор моделі, записує перший і останній шар мережі, виділяє пам'ять, створює список із підтримуваних міток.

Продовження таблиці 3.5

Назва	Опис аргументів	Опис методу
<code>_filter_results</code>	<code>prediction</code> – представлення фотографії у вигляді багатовимірної матриці	Фільтрує результати передбачення моделі за пороговим значенням <code>threshold</code> . Обирає найкращі мітки категорій і вертає словник з результатами.
<code>make_prediction</code>	<code>img</code> – представлення фотографії у вигляді багатовимірної матриці	Активує модель, яка приймає <code>img</code> і вертає список числових значень, які означають результат передбачення для кожної з підтримуваних міток відповідно. Вертає результати, відфільтровані за допомогою методу <code>_filter_results</code> .

Як можемо бачити, цей клас не містить великої кількості методів та полей, тому що виконує складні задачі на високому рівні абстракції – приймає фотографію і вертає результати розпізнавання.

### 3.4 Проектування контролеру

Вигляд та Модель виконують доволі спеціалізовану роль:

- Модель вертає «числа», дані;
- Вигляд має ці дані відображати.

Тому клас `App` містить відносно багато різноманітних функцій для роботи з двома іншими класами – фактично розширює їх функціонал. Наприклад, перетворює дані у віджети для відображення.

Поля та методи класу представлено на рис. 3.6.

App
+ nn: NeuralNet + transition: CardTransition - _permissions: List[String]
- __init__(nn: NeuralNet): None - _ensure_permissions(): Bool - _set_icons(): None - _create_tabs(): None - _create_list_item(category: String, symbol: String): OneLineIconListIter - _show_error: Bool - _show_prediction(prediction: Dict[String,Dict]): Bool + build(): ScreenManager + on_start(): None + populate_tab(instance_tabs: MDTabs, instance_tab: Tab, instance_tab_label: MDLabel, tab_text: StringProperty): None + show_info(): None + prepare_scanner(): Bool + scan_photo(scanner: Widget): Bool

Рисунок 3.6 – Поля та методи класу App

Поля класу App описано в табл. 3.6.

Таблиця 3.6 – Поля класу App

Назва	Опис
nn	Об'єкт класу NeuralNet, до якого App буде звертатись
transition	Об'єкт класу CardTransition – анімація переходу, яка буде використовуватись між екранами мобільного застосунку
_permissions	Список дозволів, які треба запросити і отримати у системи, щоб, наприклад, працювати з камерою пристрою

Приватні методи загальних функцій застосунку описано в табл. 3.7.

Таблиця 3.7 – Приватні методи загальних функцій класу App

Назва	Опис аргументів	Опис методу
__init__	Аргумент nn описано в табл. 3.6	Створює об'єкт App і зберігає адрес об'єкту NeuralNet, до якого App пізніше буде звертатись.

Продовження таблиці 3.7

Назва	Опис аргументів	Опис методу
<code>_ensure_permissions</code>	–	Перевіряє наявність потрібних дозволів ( <code>_permissions</code> ) і робить запит на них у користувача. Вертає чи всі дозволи в наявності – <code>True</code> або <code>False</code> .
<code>_set_icons</code>	–	Реєструє шрифт символів (пари «символ» – «гліф») для подальшого використання в програмі.
<code>_create_tabs</code>	–	Створює і додає до екрану об'єкти класу <code>Tab</code> , що відповідають окремим категоріям символів. Викликає метод <code>populate_tab</code> , заповнюючи символами ту категорію, яку бачить користувач після запуску застосунку.
<code>_create_list_item</code>	<code>category</code> – ім'я категорії символу; <code>symbol</code> – ім'я символу	Створює і вертає заповнений інформацією об'єкт класу <code>OneLineIconListItem</code> , який відповідає одному символу в їх переліку.

Приватні методи роботи з результатами передбачення описано в табл. 3.8.

Таблиця 3.8 – Приватні методи для роботи з результатами передбачення

Назва	Опис аргументів	Опис методу
<code>_show_error</code>	–	Створює і додає на екран вікно з повідомленням про помилку розпізнавання. Видаляє об'єкт через певний час. Завжди вертає <code>False</code> , бо представляє собою помилку.

Продовження таблиці 3.8

Назва	Опис аргументів	Опис методу
<code>_show_prediction</code>	<code>prediction</code> – словник результатів передбачення	Створює на екрані заголовки категорій результатів передбачення. Заповнює їх, розгортає результати. Наприклад, мітка «барабанну сушка дозволено» перетворюється у два символи: звичайна та делікатна сушка. Завжди вертає <code>True</code> , бо представляє собою успішне розпізнавання.

В табл. 3.9 описано методи, через які відбувається взаємодія між класами `Model` та `View`.

Таблиця 3.9 – Публічні методи класу `App`

Назва	Опис аргументів	Опис методу
<code>build</code>	—	Створює графічну частину: реєструє шрифт та завантажує клас Вигляд. Вертає об'єкт класу <code>ScreenManager</code> .
<code>on_start</code>	—	Викликається автоматично наприкінці створення графічного інтерфейсу. Викликає метод <code>_create_tabs</code> для створення вкладок.
<code>prepare_scanner</code>	—	Викликає метод <code>_ensure_permissions</code> , вертає стан наявності дозволів: <i>False</i> або <i>True</i> .

Продовження таблиці 3.9

scan_photo	scanner – об'єкт Widget, для роботи з об'єктом Camera	Зберігає фотографію, зменшує її до потрібних розмірів, отримує передбачення та відображає результат або помилку розпізнавання.
populate_tab	instance_tabs – батьківський до Tab об'єкт; instance_tab – вкладка категорії символів Tab; instance_tab_label – об'єкт заголовку вкладки, tab_text – об'єкт текстового поля заголовку	Перевіряє чи заповнена вкладка символами відповідної категорії. Якщо ні, то створює і додає до контенту вкладки відповідні символи потрібної категорії, викликаючи метод _create_list_item. Якщо вкладку було заповнено, прапорець populated об'єкту instance_tab ставиться в положення True.
show_info	–	Створює вікно інформації для користувача.

Описавши структуру даних та класи, які відповідають за логіку розроблюваного застосунку, перейдемо до проектування графічного інтерфейсу.

### 3.5 Проектування інтерфейсу користувача

Будемо використовувати діаграми Salt GUI Tree для наочності проектування структури графічного інтерфейсу мобільного застосунку. Вона має деревовидний вигляд та відображає ієрархію класів, які будуть використовуватись.

Мобільний застосунок буде мати три екрани: для перегляду списку символів, для фотографування етикетки, що буде розпізнаватись, та для перегляду



результатів розпізнавання, якщо воно було успішним. Перший екран є основним – його користувач буде бачити одразу після запуску застосунку.

Всі екрани будуть дочірніми до об'єкту, за допомогою якого буде контролюватись взаємодія з екранами (див. рис. 3.7). Наприклад, через нього можна буде змінювати поточний екран.

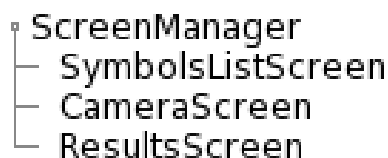


Рисунок 3.7 – Деревовидна діаграма екранів застосунку

Діаграма класів менеджера екранів показана на рис. 3.8.

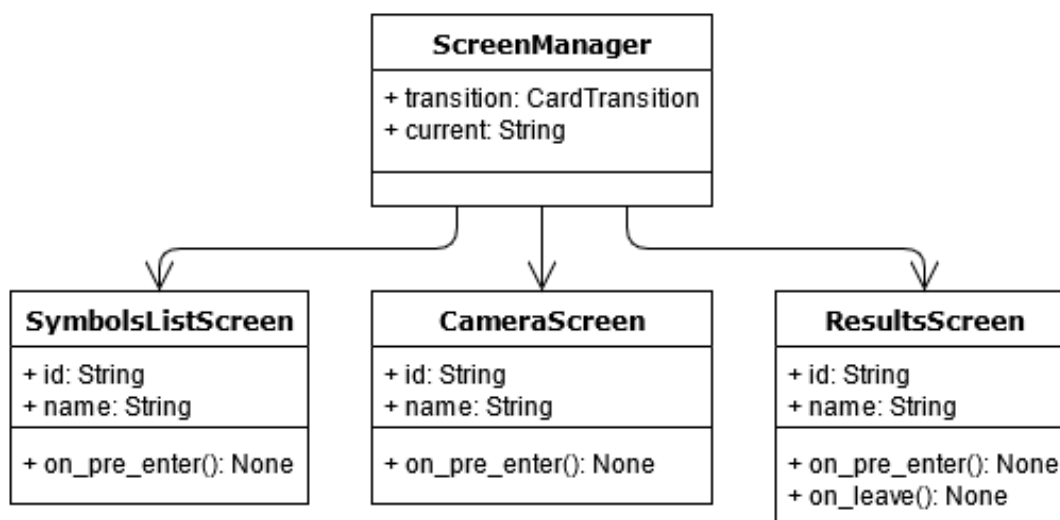


Рисунок 3.8 – Діаграма класів ScreenManager

У табл. 3.10 розглянуто поля класу ScreenManager.

Таблиця 3.10 – Поля класу ScreenManager

Назва	Опис
transition	Об'єкт класу CardTransition – анімація зміни екранів
current	Поточний екран, значення name одного з екранів.

У табл. 3.11 розглянуто поля класу SymbolsListScreen.

Таблиця 3.11 – Поля класу SymbolsListScreen

Назва	Опис
id	Ідентифікатор об'єкту для звертання до нього з класу App
name	Ім'я екрану, яке потрібно вказувати при переході з одного екрану до іншого через ScreenManager.

Поля класу CameraScreen ті самі, що вказано в табл. 3.11.

У табл. 3.12 розглянуто методи класу CameraScreen.

Таблиця 3.12 – Методи класу CameraScreen

Назва	Опис
on_pre_enter	Викликається перед зміною екрану на наступний. В даному випадку метод потрібен для перевірки і запиту дозволів на користування камерою.

Поля класу ResultsScreen ті самі, що вказано в табл. 3.11.

На кожному з екранів знизу буде панель з кнопкою, яка буде відкривати вікно з інформацією та інструкціями для користувача. Вікно також буде містити кнопку для того, щоб його закрити.

Якщо екран не є основним, на цій панелі буде відображатись кнопка, яка поверне користувача на попередній екран.

Структура цієї панелі показана на рис. 3.9. Панель буде додаватись до основного контейнеру елементів кожного екрану.



Рисунок 3.9 – Деревовидна діаграма нижньої панелі

Екран перегляду списку символів буде мати панель вкладок, які відповідають категоріям символів. Потрібна категорія буде відображатись при переході до вкладки, яку вибере користувач.

Структура списку символів показана на рис. 3.10. На прикладі показано один символ з іконкою.



Рисунок 3.10 – Деревовидна діаграма списку символів

Структура цього екрану представлена на рис. 3.11. На прикладі показана одна вкладка з одним символом.

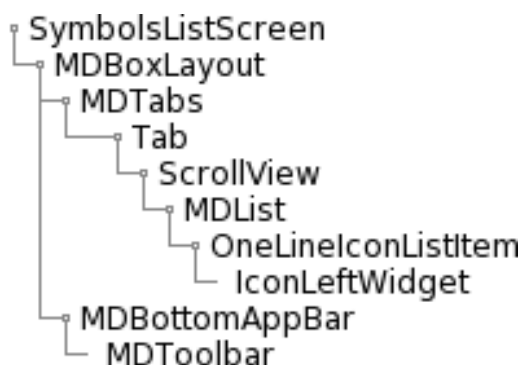


Рисунок 3.11 – Деревовидна діаграма екрану для перегляду списку символів

Контейнер контенту екрану – MDBoxLayout. Також нам треба зробити так, щоб контейнер із символами можна було скролити, якщо якісь символи виходять за екран. Для цього будемо використовувати контейнер, який можна скролити – ScrollView.

Діаграму класів цього екрану представлено на рис. 3.12.

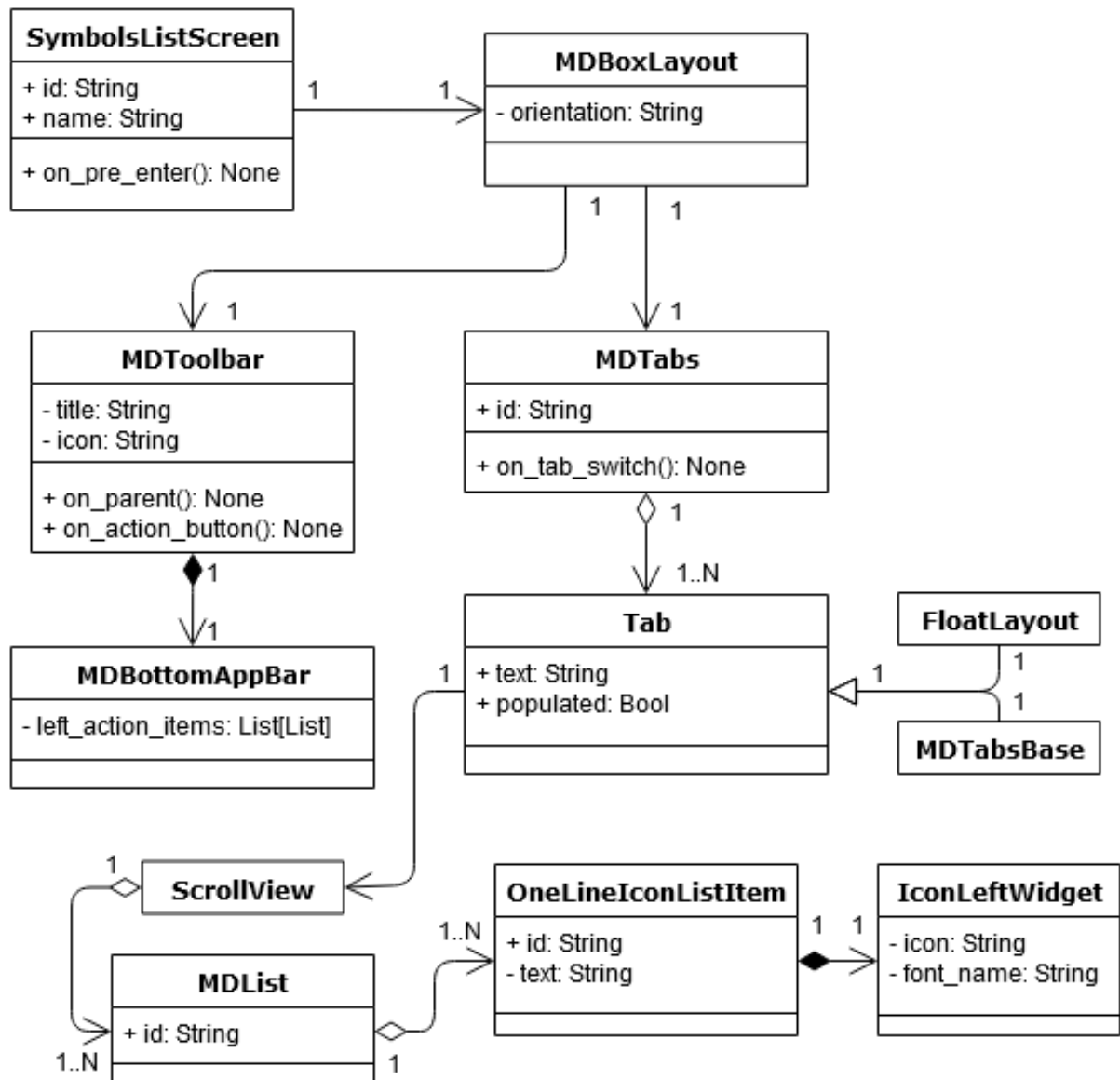


Рисунок 3.12 – Діаграма класів екрану SymbolsListScreen

У табл. 3.13 розглянуто поля класу MDBoxLayout – контейнеру контенту.

Таблиця 3.13 – Поля класу MDBoxLayout

Назва	Опис
orientation	Орієнтація відображуваного контенту

У табл. 3.14 розглянуто поля класу MDToolbar.

Таблиця 3.14 – Поля класу MDToolbar

Назва	Опис
title	Заголовок панелі
icon	Іконка великої кнопки (камера)

У табл. 3.15 розглянуто методи класу MDBottomAppBar – нижньої панелі.

Таблиця 3.15 – Методи класу MDToolbar

Назва	Опис
on_parent	Викликається автоматично при додаванні до MDToolbar. Потрібен для додавання іконок до панелі.
on_action_button	Виконує дію, коли користувач натисне на велику кнопку з іконкою камери, – зміна екрану або сканування фотографії.

У табл. 3.16 розглянуто поля класу MDToolbar.

Таблиця 3.16 – Поля класу MDToolbar

Назва	Опис
left_action_items	Перелік кнопок і методів їх активації

У табл. 3.17 розглянуто поля класу MDTabs – набору вкладок.

Таблиця 3.17 – Поля класу MDTabs

Назва	Опис
id	Ідентифікатор для звертання до об'єкту в коді

У табл. 3.18 розглянуто методи класу MDTabs.

Таблиця 3.18 – Методи класу MDTabs

Назва	Опис
on_tab_switch	Викликається при зміні поточної вкладки. Потрібен для заповнення вкладки символами перед її появою на екрані.

У табл. 3.19 розглянуто поля класу Tab – вкладки категорії символів.

Таблиця 3.19 – Поля класу Tab

Назва	Опис
text	Заголовок вкладки
populated	Чи заповнена вкладка символами: <i>True</i> або <i>False</i>

У табл. 3.20 розглянуто поля класу MDList – списку символів.

Таблиця 3.20 – Поля класу MDList

Назва	Опис
id	Ідентифікатор для звертання до об'єкту в коді

У табл. 3.21 розглянуто поля класу OneLineIconListItem – одного символу в переліку символів.

Таблиця 3.21 – Поля класу OneLineIconListItem

Назва	Опис
id	Ідентифікатор для звертання до об'єкту в коді
font_name	Значення, опис символу

У табл. 3.22 розглянуто поля класу IconLeftWidget – самого символу, іконки.

Таблиця 3.22 – Поля класу IconLeftWidget

Назва	Опис
icon	Літера, яка відповідає гліфу символу у шрифті
font_name	Ім'я використовуваного шрифту з гліфами символів

Далі розглянемо екран для сканування фотографії (див. рис. 3.12).

Він буде відображати зображення з камери пристрою і перевіряти фотографію на наявність символів, якщо користувач натисне кнопку камери.

Але якщо користувач не надав системних дозволів для користування камерою, то замість цього екрану він побачить повідомлення про помилку.



Рисунок 3.12 – Деревовидна діаграма екрану з камерою

Оскільки віджет Camera просто показує зображення з фотокамери, яке типово в горизонтальній орієнтації, нам буде потрібен пустий віджет Widget для того, щоб показувати в ньому це зображення у вертикальній орієнтації – в розвернутому вигляді.

У табл. 3.23 розглянуто поля класу Widget – списку символів.

Таблиця 3.23 – Поля класу Widget

Назва	Опис
id	Ідентифікатор об'єкту. Потрібен для захоплення зображення в правильній орієнтації

Діаграму класів цього екрану представлено на рис. 3.13.

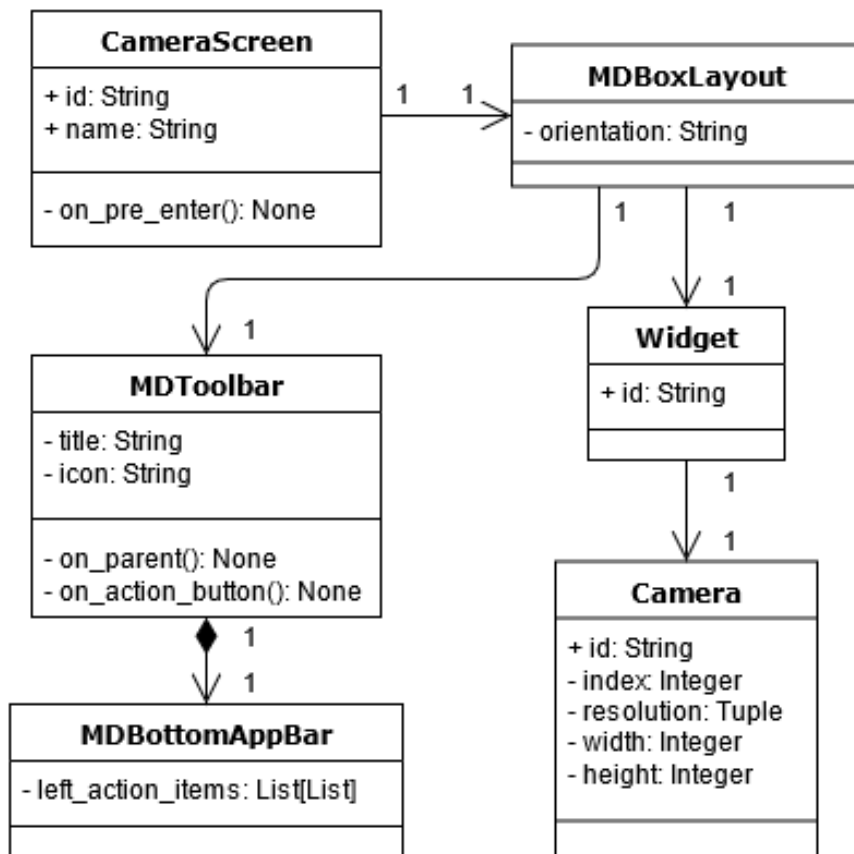


Рисунок 3.13 – Діаграма класів екрану CameraScreen

У табл. 3.24 розглянуто поля класу Camera.

Таблиця 3.24 – Поля класу Camera

Назва	Опис
id	Ідентифікатор об'єкту
index	Номер використовуваної камери – передня чи задня
resolution	Роздільна здатність зображення з камери
width	Ширина показуваного на екрані зображення
height	Висота показуваного на екрані зображення

Продовжимо розглядати екрани.

Якщо на фотографії знайшлися символи, то екран камери зміниться на екран результатів передбачення. На цьому екрані будуть відображатись переліки самих символів, і над кожним із переліків будуть заголовки відповідних категорій.



На рис. 3.14 представлена структура екрану результатів

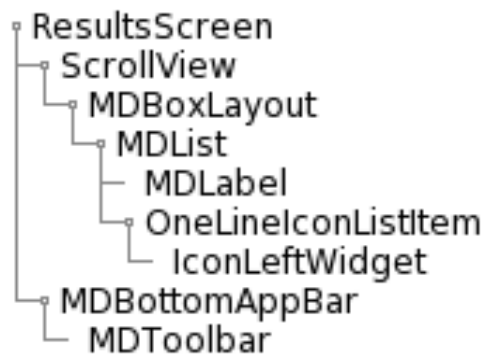


Рисунок 3.14 – Деревовидна діаграма екрану результатів передбачення

На рис. 3.15 представлена діаграма класів цього екрану.

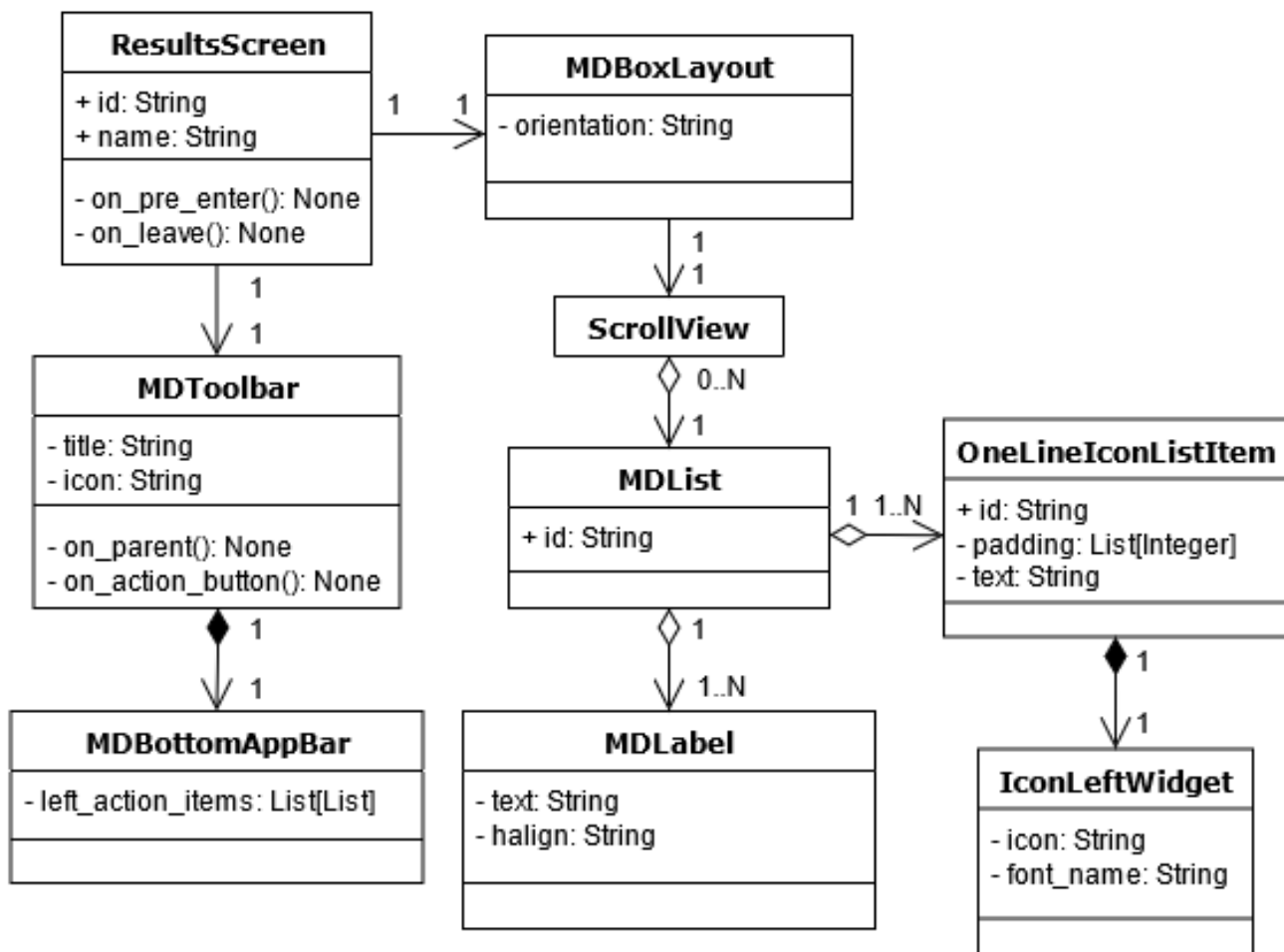


Рисунок 3.15 – Діаграма класів екрану ResultsScreen

Якщо нейронна мережа нічого не знайшла на фотографії, то користувач побачить на попередньому екрані повідомлення з помилкою і рекомендаціями до її виправлення. Вікно зникне автоматично через декілька секунд.

На табл. 3.25 описано поля класу MDLabel – заголовок категорії в результатах.

Таблиця 3.25 – Поля класу MDLabel

Назва	Опис
text	Текст заголовку категорії символів
halign	Вирівнювання по горизонталі – заголовок має бути по центру

Розглянувши питання архітектури – деталі всіх модулів програмної системи, відношення між ними, їх елементи, – перейдемо до програмної реалізації мобільного застосунку.

## 4 ПРОГРАМНА РЕАЛІЗАЦІЯ РОЗРОБЛЮВАНОЇ СИСТЕМИ

### 4.1 Формування набору даних та навчання нейронної мережі

Як було зазначено в документації технології Google Cloud AutoML Vision [34], для того, щоб нейронна мережа правильно розпізнавала мітки, треба переконатися, що для кожної мітки існує мінімум 10 фотографій.

Але для нашої системи це, і цього буде замало, тому що міток на одній фотографії частіше за все п'ять, а вищезгадана рекомендація відноситься до менш складних систем.

За документацією формат набору даних має бути наступний:

`set, photo_path, [labels]`

Розглянемо цей формат:

- `set` може бути TRAIN, VALIDATION або TEST відповідно. Раніше було обрано приблизне співвідношення цих категорій 60%/20%/20% відповідно;

- `photo_path` – це шлях до хмарного сховища Google Cloud Platform Storage. Спочатку розмітимо фотографії, а потім завантажимо їх у хмару;

- `labels` – перелік символів на фотографіях. Повний перелік сформованих імен символів представлено в додатку А.

На рис. 4.1 представлено приклад міток, які будуть використовуватись.

```
"WASH": {
  "WASH_no": {
    "icon": "z", "text": "Прання заборонено"
  },
  "WASH_hand": {
    "icon": "t", "text": "Ручне прання, температура до 40°C"
  }, ...
}
```

Рисунок 4.1 – Приклад списку міток/символів

Перед іменем файлу в `photo_path` кожного запису допишемо шлях до сегменту хмарного сховища: `gs://label-scan/`.

Тож зібрані 200 фотографій розмітимо по цій схемі.

На рис. 4.2 представлено приклад розмітки фотографії з набору даних.

```
TRAIN,gs://label-  
scan/122.jpg,WASH_30,BLEACH_no,DRY_tumble_no,IRON_yes,PROF_dry_no
```

Рисунок 4.2 – Приклад розміченої фотографії

Сама фотографія зображена на рис. 4.3.



Рисунок 4.3 – Фотографія 122.jpg, приклад з набору даних

Оскільки для розпізнавання ми будемо використовувати тільки стандарт GINETEX, чіткі фотографії та найпопулярніші символи, залишається 116 фотографій. Статистику по міткам представлена на табл. 4.1.

Таблиця 4.1 – Статистика кількості зображень по міткам та категоріям

Мітки	Зображень	Тренування	Перевірка	Тестування
BLEACH_no	116	74	22	20
DRY_tumble_no	89	57	16	16
DRY_tumble_yes	24	15	5	4
IRON_yes	116	74	22	20
PROF_dry_no	64	37	14	13
PROF_dry_p	45	32	7	6
WASH_30	64	40	12	12
WASH_40	38	24	8	6
WASH_hand	14	10	2	2

Завантажимо фотографії до сховища. Інтерфейс вищезгаданого сервісу показано на рис. 4.4.

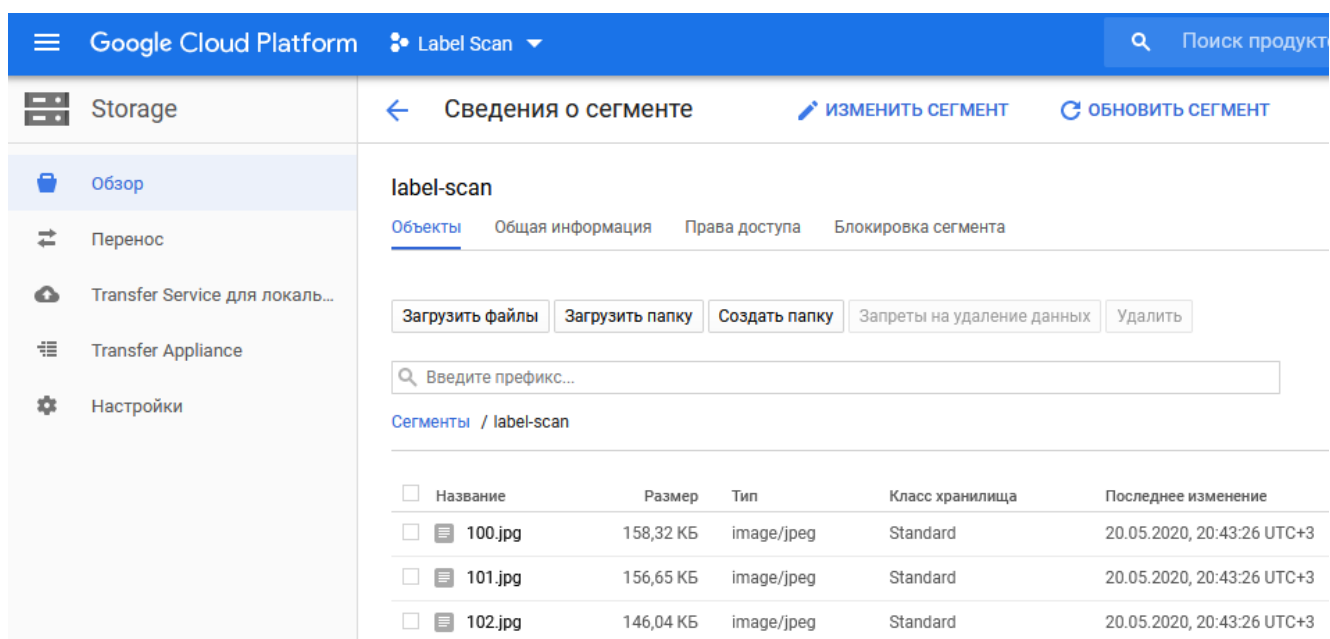


Рисунок 4.4 – Інтерфейс Google Cloud Platform Storage

Далі завантажимо набір даних зі сховища до Google Cloud AutoML Vision. Імпорт відбувається за допомогою завантаження CSV файлу з розміченими в раніше описаному форматі фотографіями.

Інтерфейс зображено на рис. 4.5.

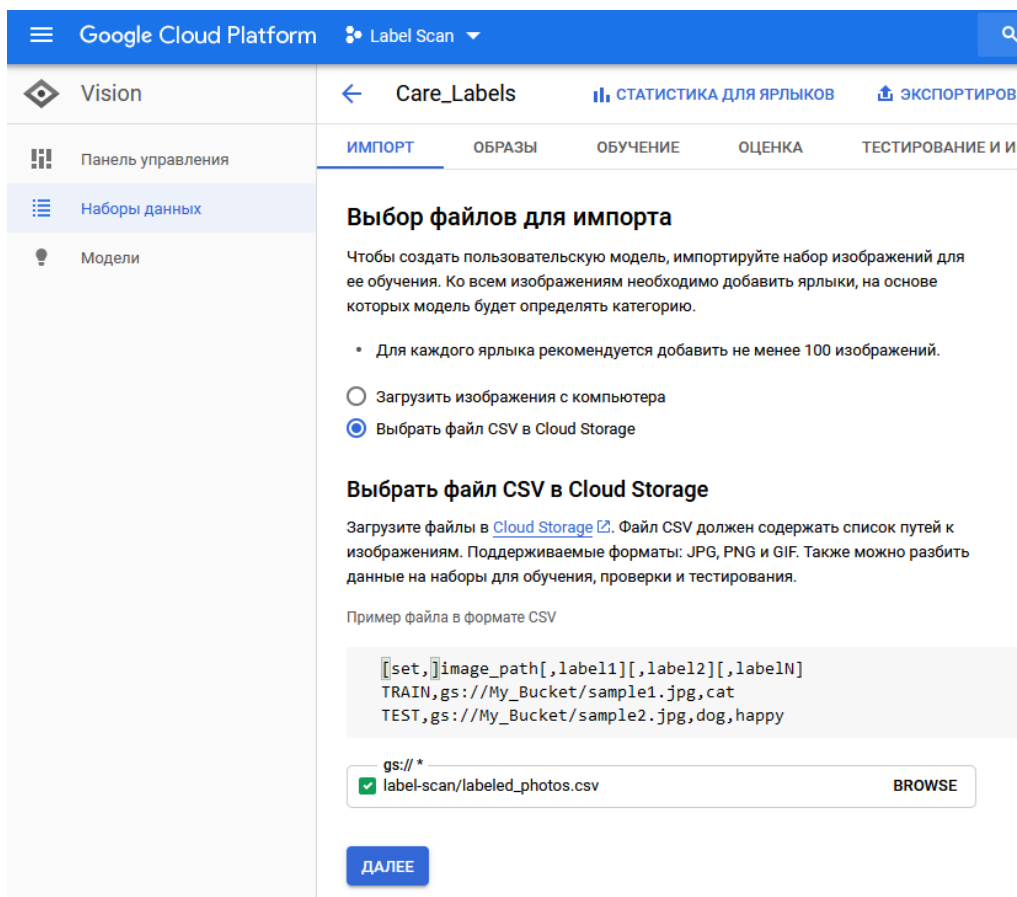


Рисунок 4.5 – Импортування набору даних до Vision

Після завершення операції переходимо у вкладку «Обучение» (навчання), та створюємо нову модель типу Edge – моделі цього типу працюють на мобільних пристроях та експортуються із сервісу (див. рис. 4.6).

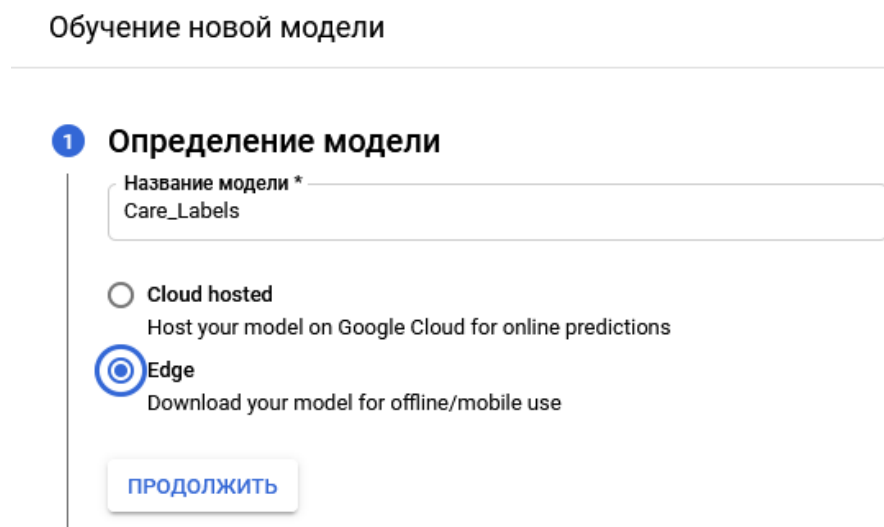


Рисунок 4.6 – Вибір типу моделі

Ще доступна опція Cloud Hosted – вона дозволяє не тільки навчати, а й розгортати модель у хмарі. Таким чином можна звертатися до неї через API. Але ця опція по вичерпанню певних лімітів буде коштувати грошей, тому її ми не розглядаємо.

Далі обираємо точність моделі – високу. Різниця у швидкості невелика, зате точність передбачень моделі буде вище.

✓ **Определение модели**

2 **Цель оптимизации модели:**

Цель	Размер пакета	Точность	Latency for Google Pixel 2
<input checked="" type="radio"/> Higher accuracy	6 MB	Higher	360 ms
<input type="radio"/> Best trade-off	3.2 MB	Medium	150 ms
<input type="radio"/> Faster predictions	0.6 MB	Lower	56 ms

Please note that prediction latency estimates are for guidance only. Actual latency will depend on your network connectivity.

**ПРОДОЛЖИТЬ**

Рисунок 4.7 – Вибір точності моделі

Після цього виберемо кількість годин роботи вузлів (див. рис. 4.8).

Кількості безкоштовних годин вистачає приблизно на 15 навчань в даному випадку. Обираємо рекомендовану кількість, – якщо мережа припинить ефективно навчатись, навчання припиниться і години роботи вузлів зніматись не будуть.

Навчання займає приблизно півгодини, тож навіть рекомендовані 5 годин не використались повністю.

### 3 Set a node hour budget

Укажите **максимальное** количество часов работы узла, которое вы хотите выделить для обучения модели.

We recommend using [5 node hours](#) for your dataset. However, you can train for as little as 1 node hours. You may also eligible to train with free node hours.

[Справочник по тарифам](#)

Определение бюджета \*  node hours

Estimated completion date: июн. 19, 2020 1 PM GMT+3

Рисунок 4.8 – Вибір кількості годин роботи вузлів для навчання

Точність навченої моделі вказана на рис. 4.9. Це фактор, який включає в себе багато чинників задля відображення загальної ефективності моделі.

Средняя точность ?

0,969

Рисунок 4.9 – Точність навченої моделі

На рис. 4.10 показані результати по міткам.

Все ярлыки	_____	PROF_dry_no	_____
BLEACH_no	_____	PROF_dry_p	_____
DRY_tumble_no	_____	WASH_30	_____
DRY_tumble_yes	_____	WASH_40	_____
IRON_yes	_____	WASH_hand	_____

Рисунок 4.10 – Результати по міткам



Можлива причина погіршення результатів по мітці DRY\_tumble\_yes показана на рис. 4.11.

Ложноотрицательные срабатывания: "DRY\_tumble\_yes"



Рисунок 4.11 – Проблемна етикетка

Зображень для тестування було всього декілька, що дуже мало для навчання ефективної моделі. Але можливо, іконка бренду «заплутала» нейронну мережу.

Обираємо формат експорту моделі – TF Lite. Є ще варіанти: TensorFlow.js, CoreML, Container, Coral. Але нам потрібен саме варіант TF Lite – з такою моделлю можна буде працювати на мобільних пристроях.

### Use your model

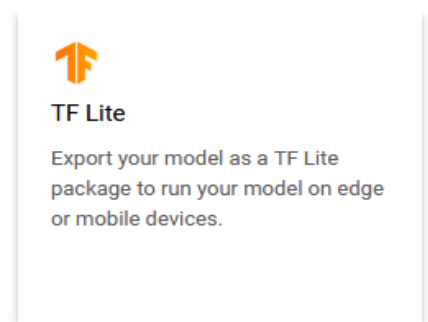


Рисунок 4.12 – Формат моделі для експорту

Експортуємо модель (рис. 4.13).

### Export TF Lite package

The **Tensorflow Lite (.tflite)** format allows you to run embedded devices.

1. Export your model as a TF Lite package.

Destination folder on Cloud Storage

EXPORT

Рисунок 4.13 – Експорт моделі формату TF Lite

Отримуємо файл моделі `model.tflite` та файл з переліком міток `dict.txt` (див. додаток А). Порядок, в якому вони представлені, важливий, бо в такому ж порядку мережа буде вертати передбачення.

## 4.2 Реалізація нейронної мережі

Опишемо основну функцію класу `NeuralNet` – дати передбачення – `make_prediction`. Її алгоритм представлено на рис. 4.14. Повний код класу представлено в додатку А.

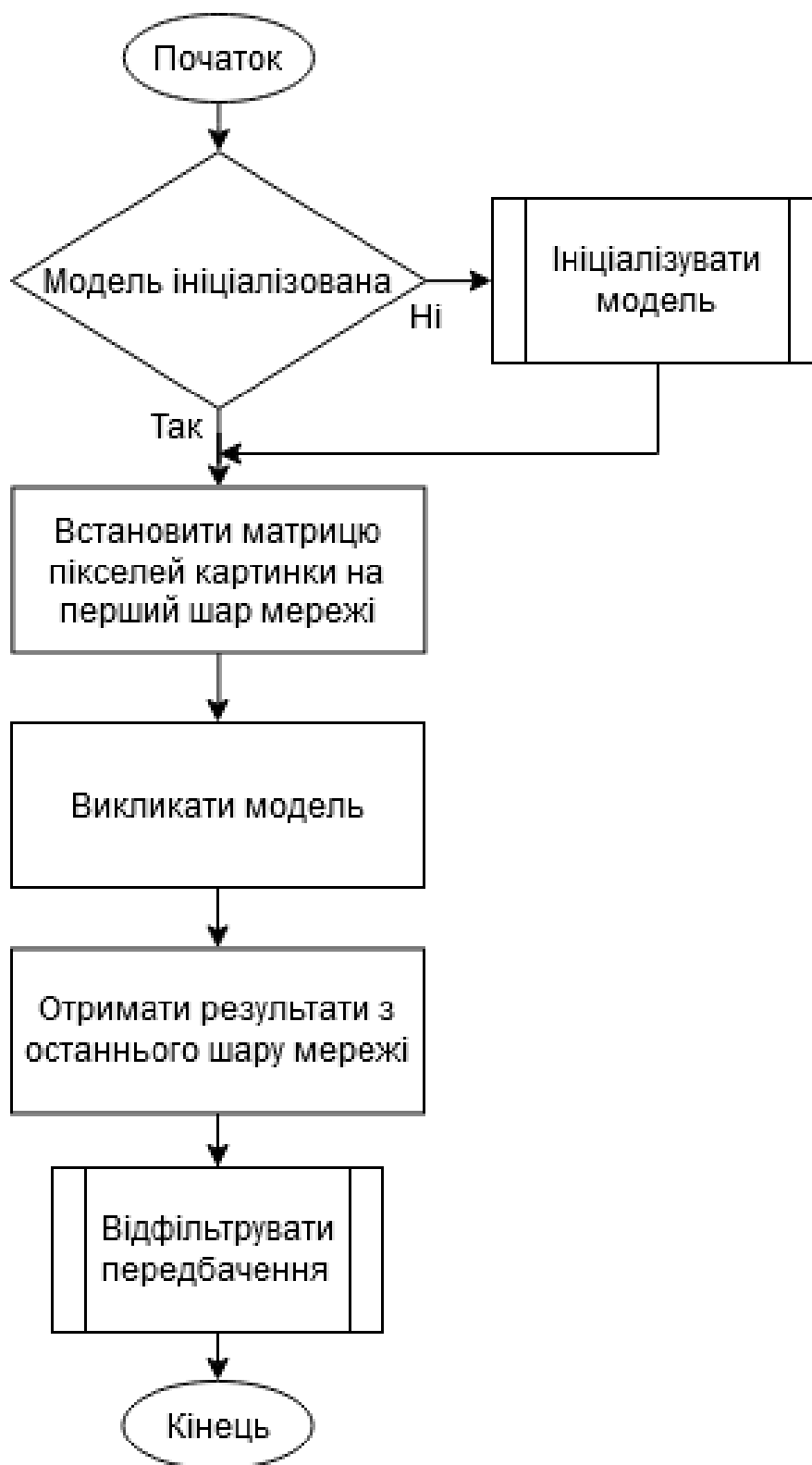


Рисунок 4.14 – Схема алгоритму функції передбачення

Як вже було зазначено раніше, клас виконує доволі вузьку роль інтерпретатора та працює на високому рівні абстракції.

### 4.3 Реалізація контролеру

Розглянемо основну функцію Контролеру – інтерпретація передбачень нейронної мережі.

На рис. 4.15 представлено алгоритм цієї функції. Кожна мітка з передбачення виконує одну з 3 умов, результатом кожної з яких є додавання символів на екран.

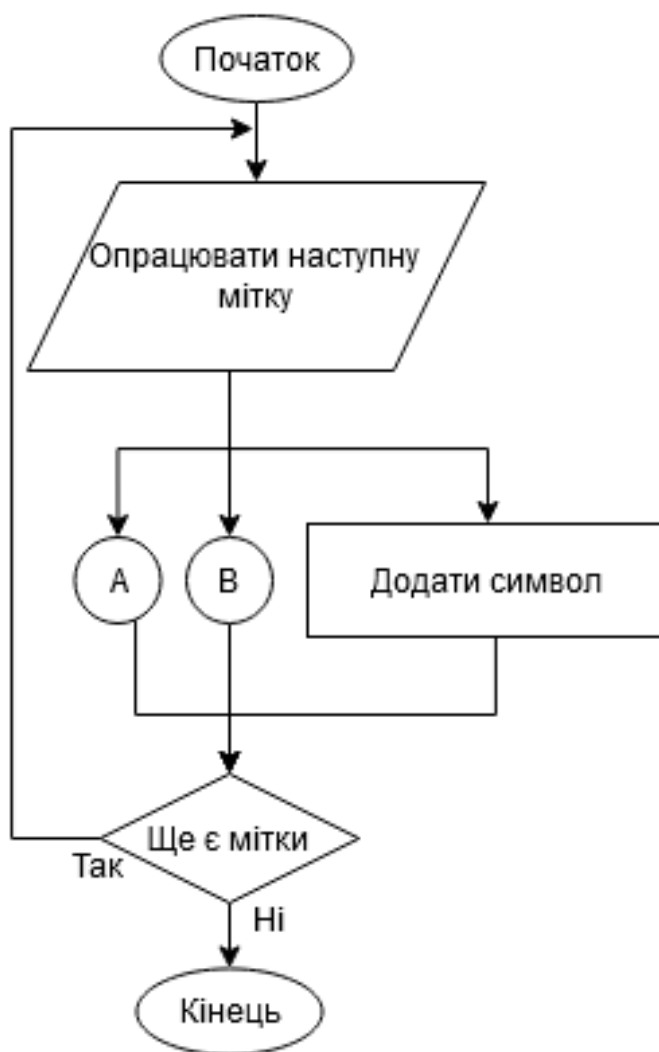


Рисунок 4.15 – Схема алгоритму функції відображення передбачення

Приклад для випадку С: мітка DRY\_tumble\_no просто перетвориться у символ DRY\_tumble\_no.

На рис. 4.16 представлено одне з відгалужень вищеописаного алгоритму.



Рисунок 4.16 – Схема алгоритму відображення передбачення для міток, які закінчуються на «yes»

Приклад: DRY\_tumble\_yes перетвориться у DRY\_tumble\_yes, DRY\_tumble\_normal та DRY\_tumble\_mild.

На рис. 4.17 представлено інше відгалуження вищеописаного алгоритму на рис. 4.15.

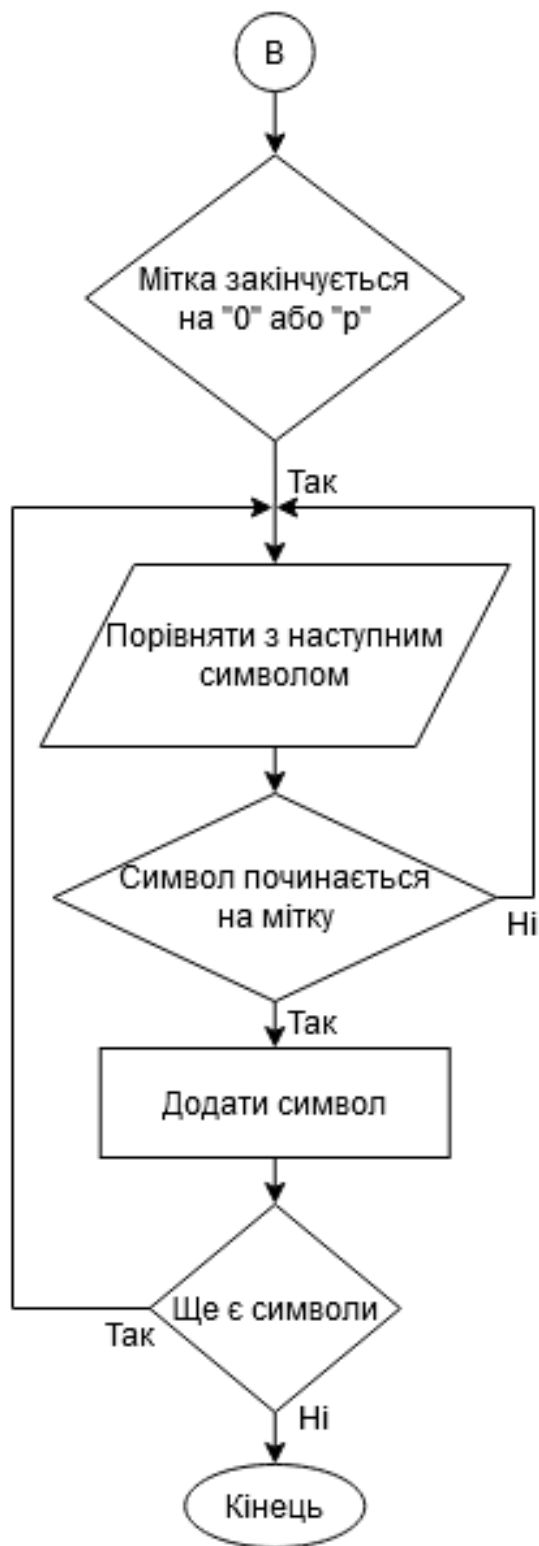


Рисунок 4.17 – Схема алгоритму відображення передбачення для міток, які закінчуються на «0» або «p»

Приклад: PROF\_dry\_p перетвориться у PROF\_dry\_p та PROF\_dry\_p\_mild.

## 4.4 Реалізація інтерфейсу користувача

Для реалізації графічного інтерфейсу використовуємо фреймворк Kivy. Він дозволяє описувати елементи інтерфейсу спеціальною мовою KvLang.

Обрана система дизайну мобільного застосунку – Material Design. Для програмування графічного інтерфейсу з утриманням специфікацій Material Design і використанням Kivy використовуємо бібліотеку KivyMD, яка надає відповідні віджети.

Приклад коду наведено на рис. 4.18. Повний програмний код графічного інтерфейсу наведено в додатку А.

```
<Tab@FloatLayout+MDTabsBase>:
    text:
        "[size=30sp][font=ginetex.ttf]{glyph}[/font][[/size]\n \
        [size=15sp][b]{title}[/b][[/size]]" \
        .format(glyph=self.glyph,title=self.title)
    populated: False

    ScrollView:
        MDList:
            padding: (0, dp(20))
            id: symbols
```

Рисунок 4.18 – Реалізація вкладки мовою KvLang

На рис. 4.19 приведено частину методу, який створює всі вкладки і заповнює типову (першу) вкладку символами.

```
first_tab = None
for category, meta in gs.category_meta.items():
    tab = Tab(name=category, glyph=meta["icon"], title=meta["translation"])
    if not first_tab:
        first_tab = tab
    self.root.ids.symbols_list_screen.ids.tabs.add_widget(tab)
self.populate_tab(instance_tab=first_tab)
```

Рисунок 4.19 – Частина методу \_create\_tabs

На рис. 4.20 наведено приклади графічного інтерфейсу екрану символів.

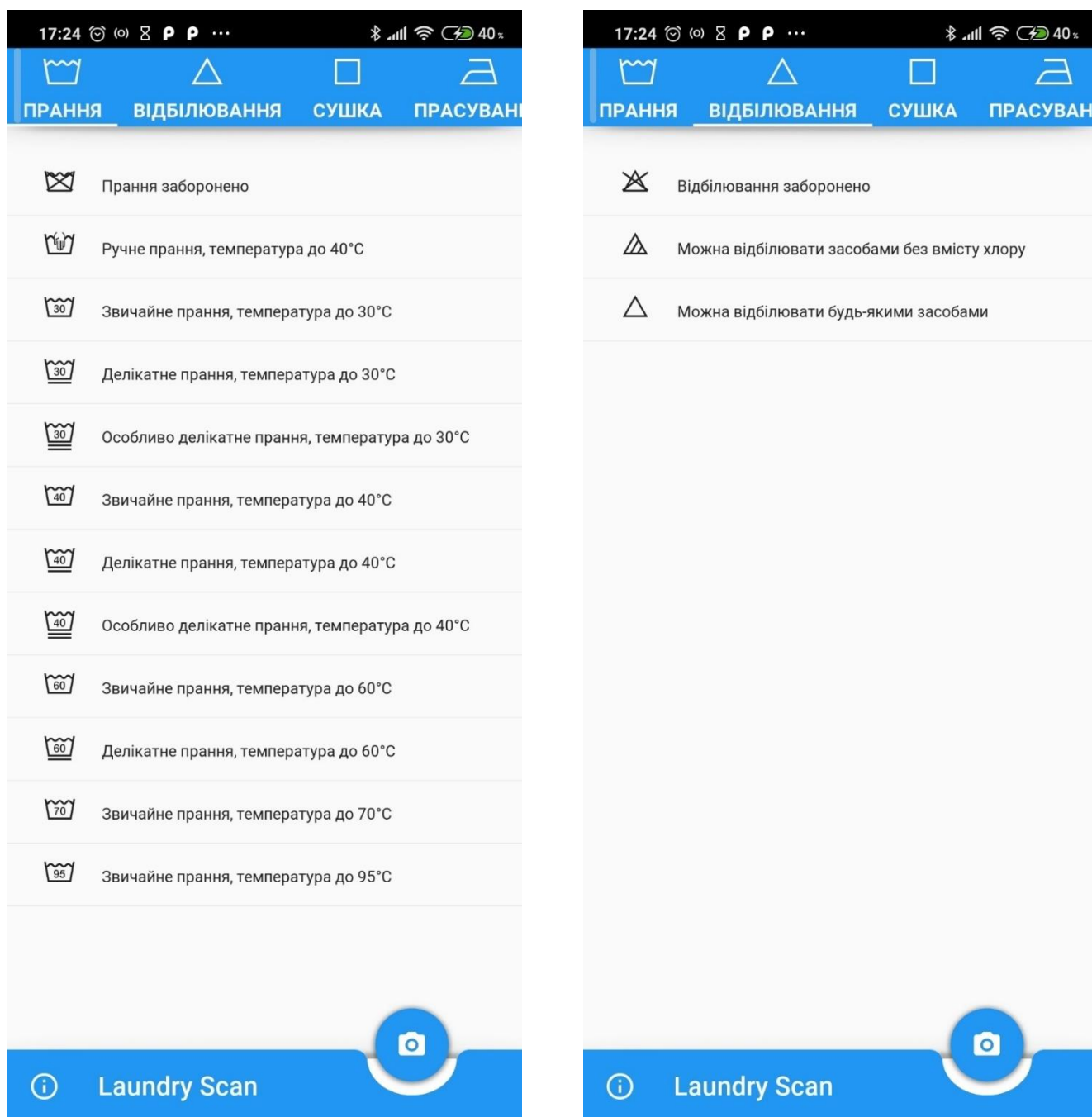


Рисунок 4.20 – Екран символів; категорії прання та відбілювання

Також користувач може з будь-якого екрану відкрити вікно інструкцій та інформації про застосунок.

Приклад показано на рис. 4.21.



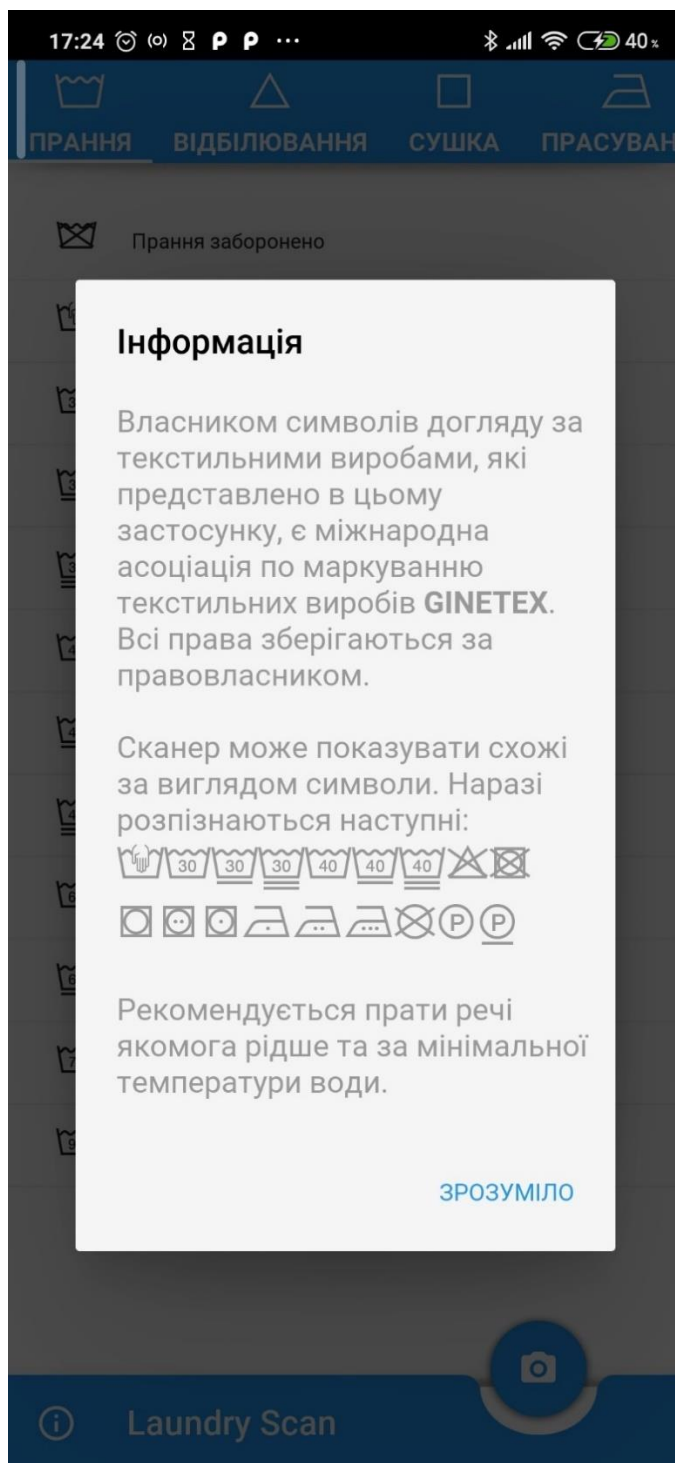


Рисунок 4.21 – Вікно з інформацією

Якщо натиснути по затемненій області або по кнопці «зрозуміло», вікно закриється. Це реалізовано за допомогою віджету MDDialog.

Якщо натиснути на кнопку з іконкою камери, то застосунок запросить дозвіл на користування камерою (див. рис. 4.22).

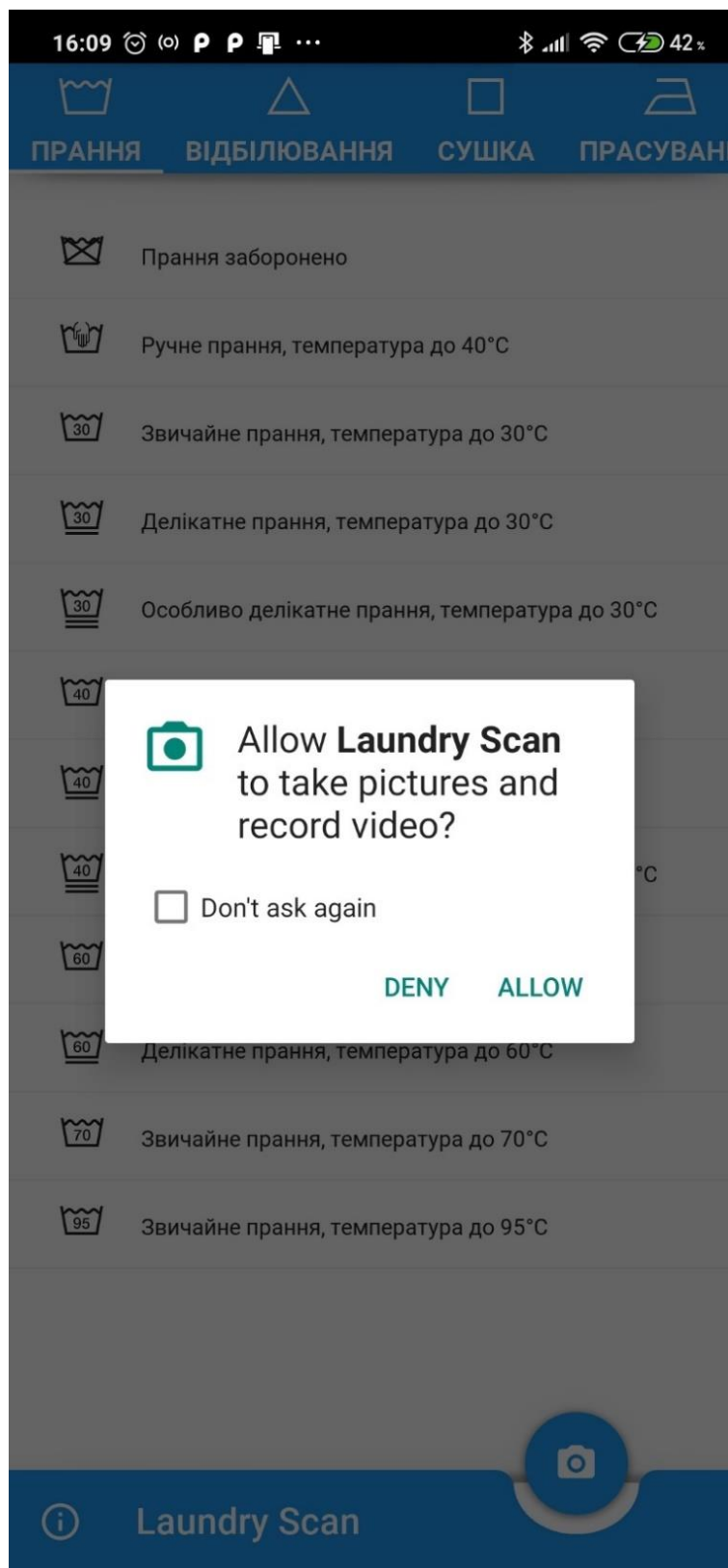


Рисунок 4.22 – Запит на дозвіл користування камерою

Якщо користувач не надав дозвіл, то на екрані з'явиться помилка, яку показано на рис. 4.23.

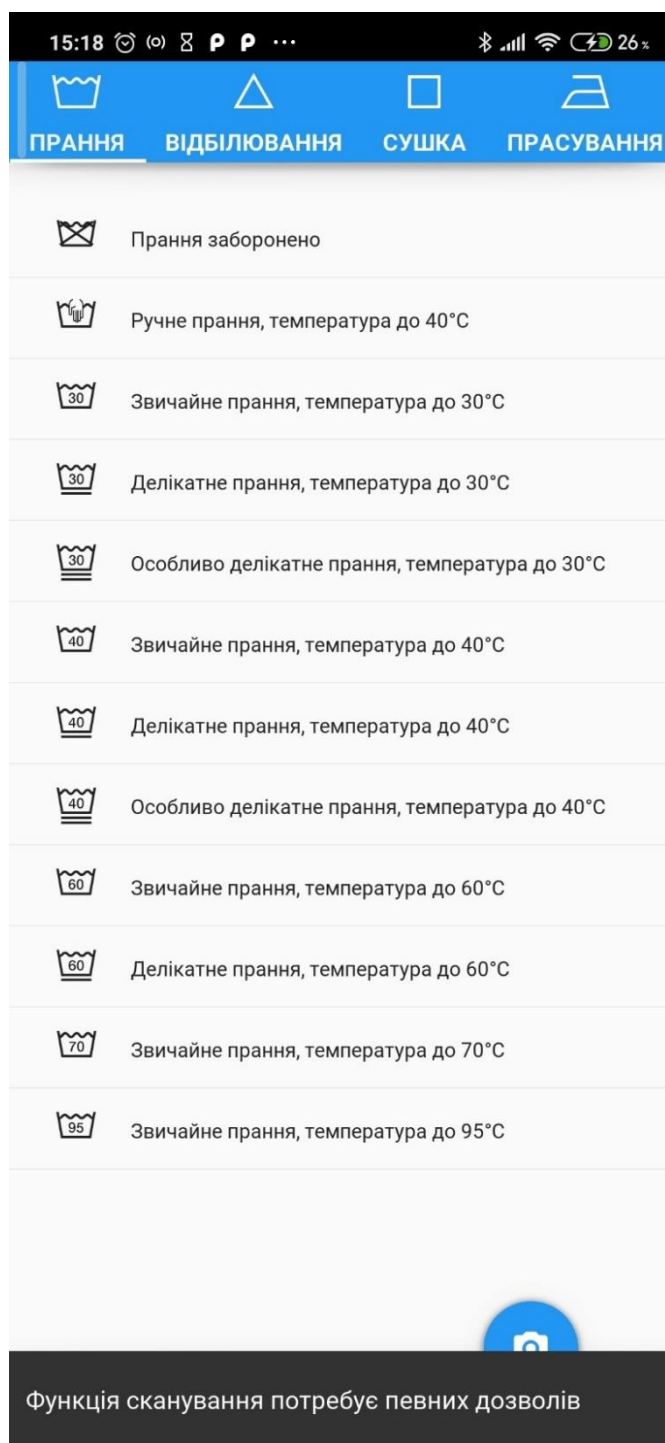


Рисунок 4.23 – Помилка наявності дозволу  
на користування камерою

Користувач може спробувати відкрити діалогове вікно про запит на дозвіл ще раз, якщо забажає.

Якщо користувач надав дозвіл, то екран символів зміниться на екран з камерою (див. рис. 4.24).

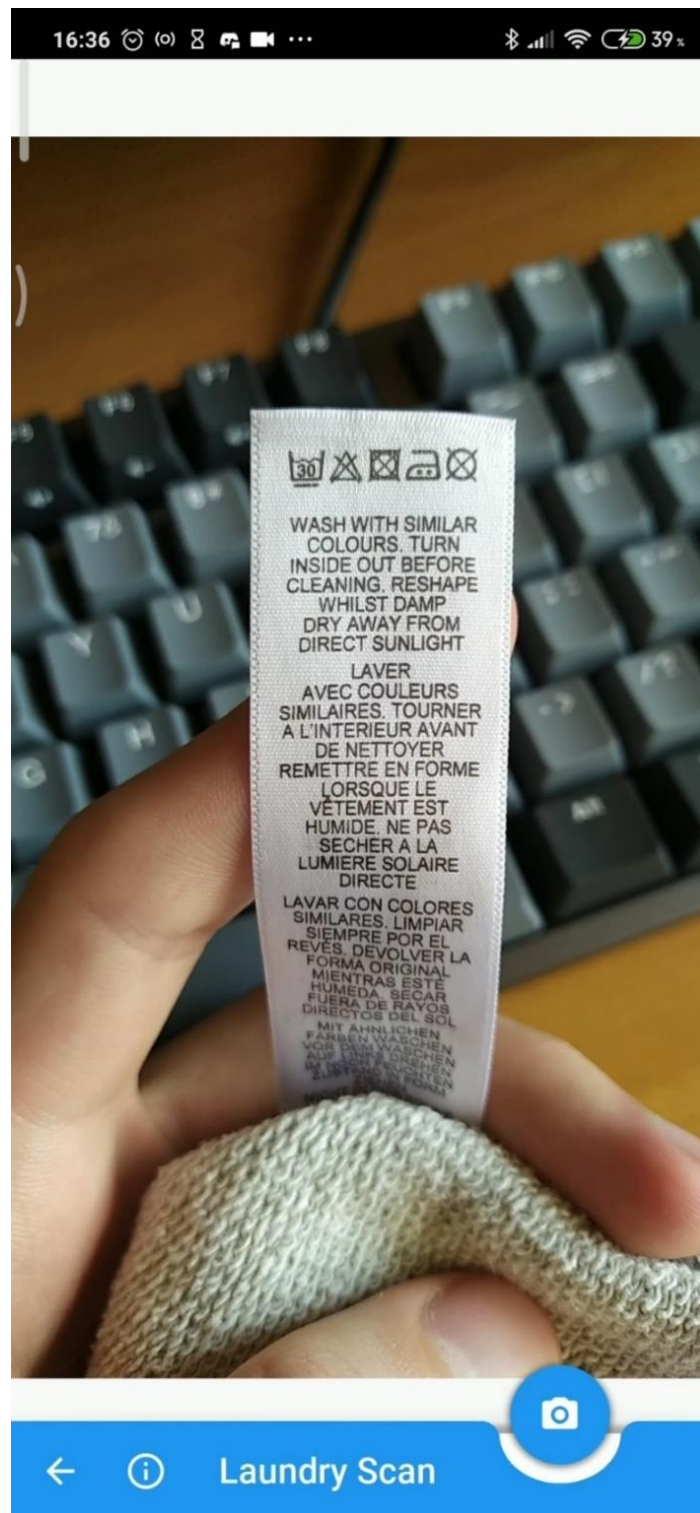


Рисунок 4.24 – Фотографування етикетки на екрані з камерою

Коли користувач натисне на кнопку з іконкою камери, зробиться фотографія, яка буде розпізнаватись нейронною мережею. Якщо результат розпізнавання успішний, то екран камери зміниться на екран результатів, на якому можна буде побачити передбачення по категоріям (див. рис. 4.25).

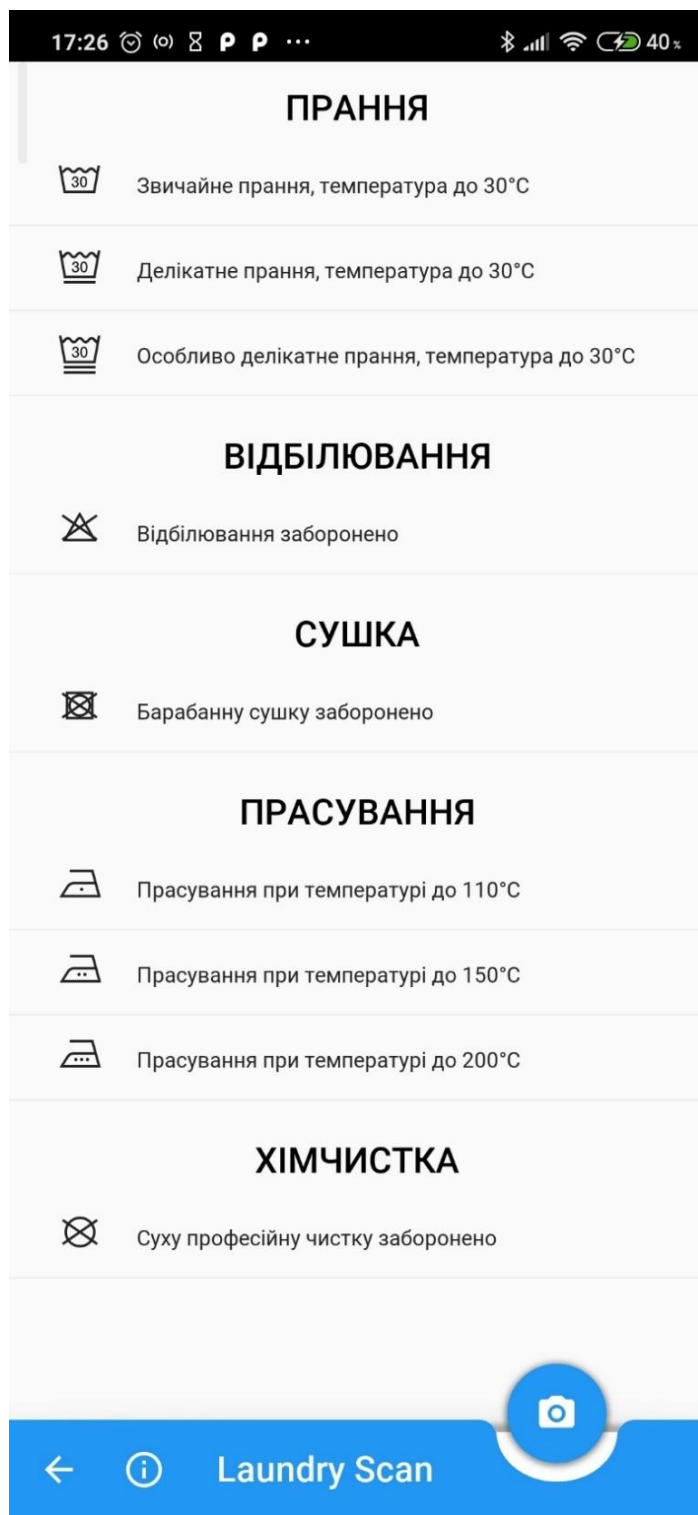


Рисунок 4.25 – Екран результату передбачення

Якщо користувач захоче повернутися назад до камери або до списку символів, він може натиснути на іконку зі стрілкою.

Також користувач може сфотографувати не етикетку, а щось інше (див. рис. 4.26).



Рисунок 4.26 – Фотографування не етикетки на екрані з камерою

Або ж на фотографії етикетку може бути погано видно. В такому разі на екрані з'явиться повідомлення з помилкою та рекомендаціями (див. рис. 4.27).

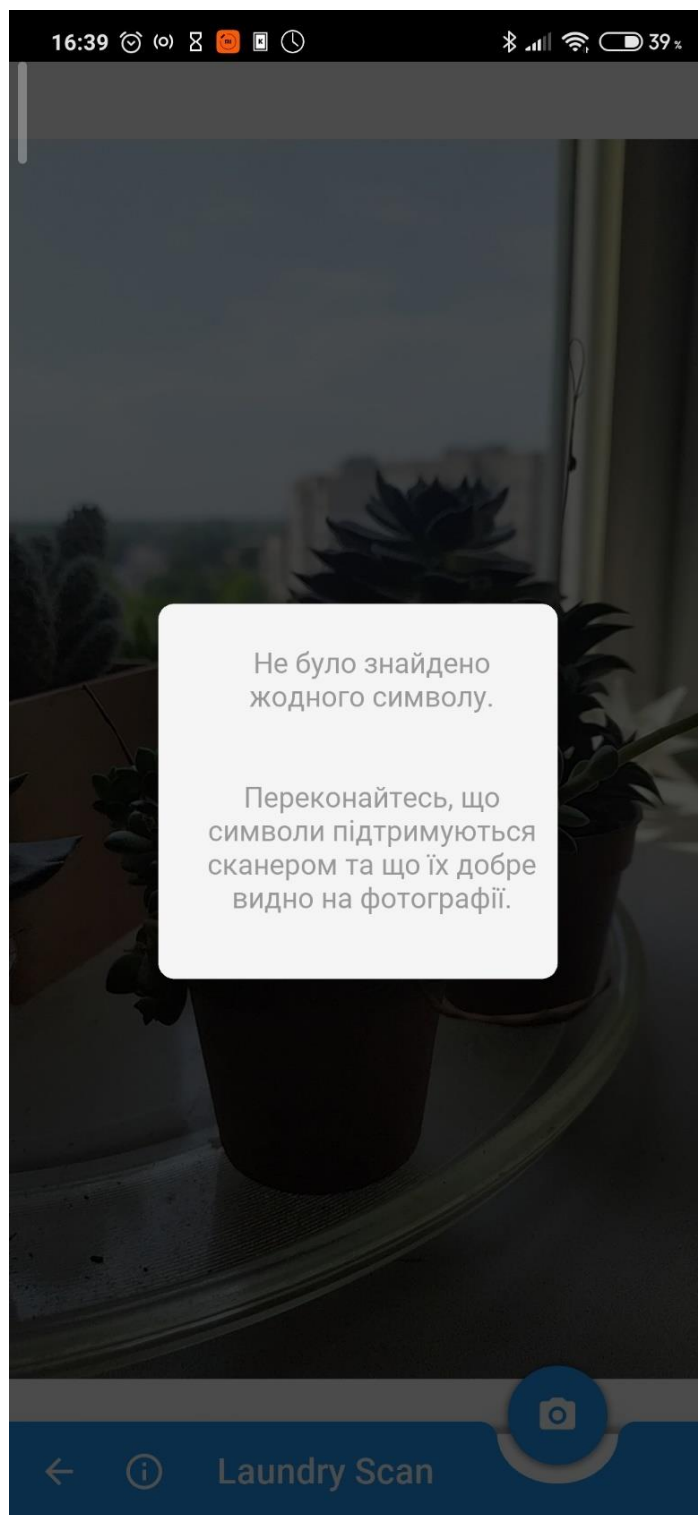


Рисунок 4.27 – Помилка розпізнавання

Інтерфейс було виконано в одному стилі. Навігація зроблена інтуїтивно. Текстові та графічні елементи інтерфейсу не залежать від розмірів екрану.

Елементи завантажуються поступово та динамічно, тому застосунок працює швидко.



## 5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

### 5.1 Матриця відповідності вимог та сценарії тестування

Матриця відповідності вимог містить відповідність функціональних вимог до підготовлених сценаріїв тестування. У табл. 5.1 представлена матриця відповідності вимог.

Таблиця 5.1 – Матриця відповідності вимог

Вимоги Test Case	Переглянути категорію символів	Перейти до екрану з камерою	Відсканувати етикетку
Test Case #1	X		
Test Case #2		X	
Test Case #3			X

Сценарій тестування випадок (Test Case) описує порядок кроків і умов для перевірки функціонування функції. В наступних таблицях будуть описані сценарії тестування. В табл. 5.2 описано перший тестовий сценарій.

Таблиця 5.2 – Опис тестового сценарію №1

Ідентифікатор	Test Case №1
Назва	Перевірка працездатності функції «Переглянути категорію символів»
Передумови	1. Користувач знаходиться на головному екрані – екрані символів
Опис кроків	1. Користувач змінює вкладку на відмінну від типової



## Продовження таблиці 5.2

Очікуваний результат №1	1. Контент вкладки заповнюється під час анімації настільки швидко, щоб користувач не помітив цього.
-------------------------	---

У табл. 5.3 описано другий тестовий сценарій – «Перейти до екрану з камерою».

Таблиця 5.3 – Опис тестового сценарію №2

Ідентифікатор	Test Case №2
Назва	Перевірка працездатності функції «Перейти до екрану з камерою»
Передумови	1. Користувач знаходиться на головному екрані
Опис кроків	1. Користувач натиснув кнопку з іконкою камери. 2. Система перевіряє наявність дозволів на користування камерою.
Очікуваний результат №1	1. Дозволи в наявності 2. Головний екран змінюється на екран з камерою
Очікуваний результат №2.1	1. Дозволів немає 2. Система запрошує дозволи у Користувача 3. Користувач надає дозволи 4. Головний екран змінюється на екран з камерою
Очікуваний результат №2.2	1. Дозволів немає 2. Система запрошує дозволи у Користувача 3. Користувач не надає дозволів 4. На головному екрані з'являється повідомлення про помилку наявності дозволів

У табл. 5.4 описано третій тестовий сценарій – «Відсканувати етикетку».

Таблиця 5.4 – Опис тестового сценарію №3

Ідентифікатор	Test Case №3
Назва	Перевірка працездатності функції «Відсканувати етикетку»
Передумови	1. Користувач знаходиться на екрані з камерою
Опис кроків	1. Користувач натиснув кнопку з іконкою камери. 2. Система робить фотографію. 3. Система намагається розпізнати символи на фотографії.
Очікуваний результат №1	1. На фотографії немає символів. 2. На екрані відображається повідомлення про помилку розпізнавання.
Очікуваний результат №2.1	1. На фотографії є символи. Всі символи на фотографії підтримуються моделлю. 2. Екран камери змінюється на екран з результатами 3. За результатами на екрані відображаються відповідні символи та їх категорії
Очікуваний результат №2.2	1. На фотографії є символи, які не підтримуються моделлю. 2. Екран камери змінюється на екран з результатами. 3. За результатами на екрані відображаються помилкові символи та їх категорії.

Можемо бачити, що всі тести було пройдено. Але, нажаль, Test Case №3 результат №2.2 є неочікуваним для користувача, тому такий сценарій не є правильним. На даний момент це є обмеженням системи.

Щоб дати відповідь, що символ не підтримується – треба вміти його розпізнавати. Тому непопулярні символи можуть викликати таке непорозуміння, якщо користувач не прочитав інструкцію користування.

## 5.2 Експеримент

Для того, щоб перевірити ефективність розробленого мобільного застосунка, необхідно провести рандомізований експеримент.

Для цього потрібно випадковим чином сформувати дві експериментальні групи:

– I група буде шукати символи для догляду за одягом в одному з додатків без функції сканування етикетки;

– II група буде використовувати для цього функцію сканування Laundry Scan.

Оскільки основна аудиторія застосунку – молодь, студенти, то було опубліковано об'яву у декількох студентських групах в месенджері Telegram.

Об'ява містила посилання на анонімну форму, прив'язану до таблиці. Учасники експерименту залишили адреси своїх поштових скриньок. За 7 днів всього було зібрано 27 адрес. З них випадковим чином було обрано 20 – по 10 на кожну групу.

Першій групі треба було засікти, скільки часу вони витратили на пошук трьох символів – прання, сушки та прасування – з етикетки на їх вибір. Вони користувались застосунком Laundry Pro. Другій – те саме, тільки з використанням функції розпізнавання Laundry Scan.

Кожний учасник мав спочатку ознайомитись з інструкцією до свого застосунку. Після цього протягом дня вони мали зробити вищевказане завдання для 5 етикеток та відправити на пошту відповідь – медіану з ряду чисел, які вони отримали. Потім було взято середнє арифметичне для кожної із груп і округлене до цілих. Отримані дані наведено в табл. 5.5.

Таблиця 5.5 – Результати обох груп, секунди

Група	1 уч.	2 уч.	3 уч.	4 уч.	5 уч.	6 уч.	7 уч.	8 уч.	9 уч.	10 уч.	Середнє арифметичне
I гр.	10	12	13	10	11	15	14	11	12	14	12,2 $\approx$ 12
II гр.	7	9	7	6	7	8	7	6	8	8	7,3 $\approx$ 7

За результатами двох груп було побудовано гістограму, яку представлено на рис. 5.1.

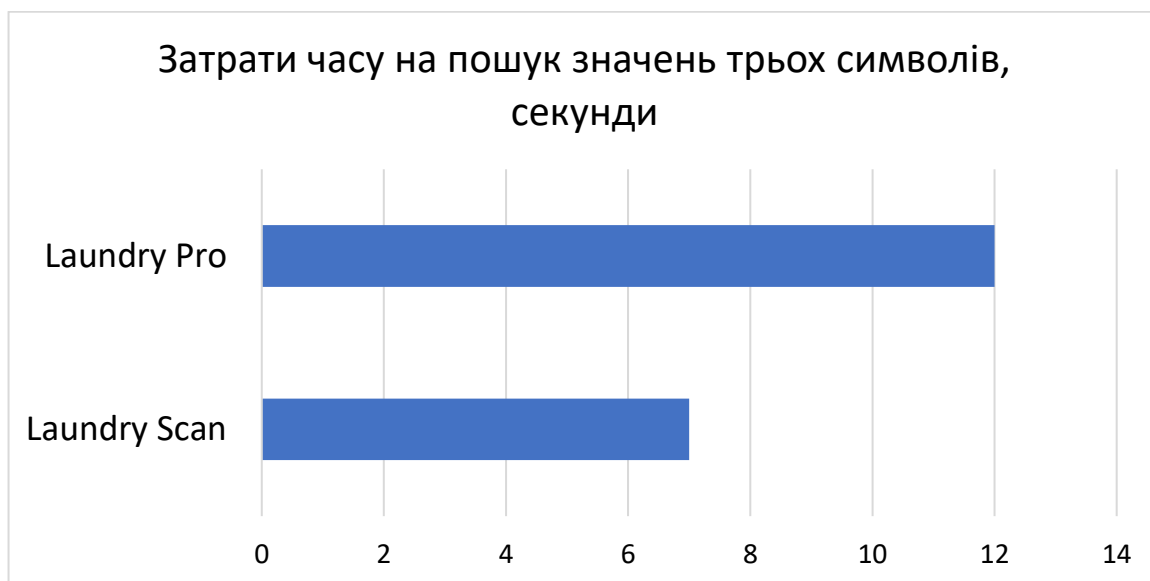


Рисунок 5.1 – Гістограма з порівнянням результатів обох груп

Як можемо бачити, застосунок Laundry Scan справляється із задачею швидше на 5 секунд, або на 42%. Але головне – він виконує поставлену задачу автоматично. Користувачу потрібно зробити два кліки, щоб отримати потрібну йому інформацію. Для інших застосунків ця кількість, звісно, набагато більша.

Це говорить про те, що користувачі будуть більше користуватися програмним продуктом, який «робить їх життя легшим», що з часом виллється у великі заощадження часу та грошей.

## **6. ОХОРОНА ПРАЦІ**

Охорона праці – це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини в процесі трудової діяльності.

Основними завданнями в галузі охорони праці є підготовка, прийняття та реалізація заходів, які спрямовані на:

- забезпечення належних, безпечних і здорових умов праці;
- утримання в належному стані виробничого устаткування, будівель і споруд;
- пропаганду охорони праці;
- облік, аналіз та оцінку стану умов і безпеки праці;
- забезпечення страхування працівників від нещасного випадку на виробництві та профзахворювання.

У попередніх розділах було розглянуто процес створення програмного забезпечення – мобільного застосунку. Була розглянута проблема, поставлені задачі, визначена специфікація вимог, план проекту, виконана реалізація та тестування. Розробка подібного програмного забезпечення здійснюються проектною групою з фахівцями різних напрямків – у тому числі програмістів. Отже, розглянемо з точки зору охорони праці робоче місце програміста проектної групи.

### **6.1 Загальна характеристика робочого місця**

Головними елементами робочого місця програміста є письмовий стіл і крісло. Основним робочим положенням є положення сидячи. Робоче місце для виконання робіт у положенні сидячи організується відповідно до ДСТУ 8604:2015. Робоча поза сидячи викликає мінімальне стомлення програміста. Раціональне планування робочого місця передбачає чіткий порядок і сталість розміщення предметів, засобів

праці і документації. Те, що потрібно для виконання робіт найчастіше, розташоване в зоні легкої досяжності робочого простору.

Моторне поле – простір робочого місця, в якому можуть здійснюватися рухові дії людини. Максимальна зона досяжності рук – це частина моторного поля робочого місця, обмеженого дугами, описуваними максимально витягнутими руками при їх русі у плечовому суглобі. Оптимальна зона – це частина моторного поля робочого місця, обмеженого дугами, описуваними передпліччями при русі в ліктьових суглобах з опорою в точці ліктя і з відносно нерухомим плечем. Зони досяжності рук у горизонтальній площині: а - зона максимальної досяжності; б - зона досяжності пальців при витягнутій руці; в - зона легкої досяжності долоні; г - оптимальний простір для грубої ручної роботи; д - оптимальний простір для тонкої ручної роботи.

Розглянемо оптимальне розміщення предметів праці і документації в зонах досяжності рук:

- дисплей розміщається в зоні а (по центру);
- клавіатура – у зоні г/д;
- системний блок розміщається в зоні б (зліва);
- принтер знаходиться в зоні а (праворуч);
- документація в зоні легкої досяжності долоні – в (ліворуч);
- література і документація, необхідна при роботі;
- у висувних ящиках столу – література, яка не використовується на постійній основі.

При проектуванні письмового столу слід враховувати наступне:

- висота столу повинна бути вибрана з урахуванням можливості сидіти вільно, в зручній позі, при необхідності спираючись на підлокітники;
- нижня частина столу повинна бути сконструйована так, щоб програміст міг зручно сидіти, не був змушений підбирати ноги;
- поверхню столу повинна мати властивості, що виключають появу відблисків у поле зору програміста;

– конструкція столу повинна передбачати наявність висувних ящиків (не менше 3 для зберігання документації, лістингів, канцелярських принад, особистих речей).

Параметри робочого місця вибираються у відповідності з антропометричними характеристиками. При використанні цих даних у розрахунках варто виходити з максимальних антропометричних характеристик (М+2). При роботі в положенні сидючи рекомендуються такі параметри робочого простору: ширина не менше 700 мм, глибина не менше 400 мм, висота робочої поверхні столу над підлогою 700-750 мм.

Оптимальними розмірами столу є:

- висота 710 мм;
- довжина столу 1300 мм;
- ширина столу 650 мм.

Поверхня для листа повинна мати не менше 40 мм в глибину і не менше 600 мм завширшки. Під робочою поверхнею повинно бути передбачено простір для ніг:

- висота не менше 600 мм,
- ширина не менше 500 мм,
- глибина не менше 400 мм.

Важливим елементом робочого місця програміста є крісло. Воно виконується відповідно до ГОСТ 21.889-76. При проектуванні крісла виходять з того, що при будь-якому робочому положенні програміста його поза повинна бути фізіологічно правильно обґрунтованою, тобто положення частин тіла повинно бути оптимальним.

Для задоволення вимог фізіології, що впливають з аналізу положення тіла людини в положенні сидючи, конструкція робочого сидіння повинна відповідати таким основним вимогам:

- припускати можливість зміни положення тіла, тобто забезпечувати вільне переміщення корпусу і кінцівок тіла одна щодо одної;

— допускати регулювання висоти в залежності від зростання працюючої людини (у межах від 400 до 550 мм), мати злегка увігнуту поверхню, мати невеликий нахил назад.

Виходячи з вищесказаного, наведемо параметри столу програміста: висота столу 710 мм; довжина столу 1300 мм, ширина столу 650 мм; глибина столу 400 мм. Поверхня для листа: у глибину 40 мм; в ширину 600 мм. Важливим моментом є також раціональне розміщення на робочому місці документації, канцелярських принад, що має забезпечити робітнику зручну робочу позу, сприяти мінімальній кількості руху й мінімальній траєкторії переміщення працюючого та предмета праці на даному робочому місці.

Створення сприятливих умов праці і правильне естетичне оформлення робочих місць на виробництві має велике значення як для полегшення праці, так і для підвищення його привабливості, позитивно впливає на продуктивність праці. Забарвлення приміщень і меблів повинні сприяти створенню сприятливих умов для зорового сприйняття, гарного настрою. У службових приміщеннях, у яких виконується одноманітна розумова робота, що вимагає значної нервової напруги і великого зосередження, кольорова гамма повинна складатись зі спокійних тонів — малонасичених відтінків холодного зеленого або блакитного кольорів. При розробці оптимальних умов праці програміста необхідно враховувати освітленість, шум і мікроклімат.

## **6.2 Загальний стан охорони праці на фірмі з інформаційних послуг**

Служба охорони праці та функціонує відповідно до пп. НПАОП 0.00-4.35-04 «Типове положення про службу охорони праці».

На підприємстві (фірмі), де працює зазначена проектна група програмістів, створена система управління охороною праці (СУОП). Саме управління охороною праці — це збереження здоров'я і працездатність людини в процесі праці, поліпшення виробничого побуту, попередження виробничого травматизму і професійних захворювань.



Усі працівники при вступі на роботу:

- проходять вступний інструктаж з питань охорони праці, надання першої медичної допомоги, про правила поведінки при виникненні аварії, пожежі з реєстрацією у спеціальному журналі;
- попередній медогляд.

На підприємстві де працює зазначена проектна група програмістів, постійно проводиться контроль технічного стану, устаткування, приладів які використовуються програмістами у своїй діяльності.

Адміністрація та керівники проекту своєчасно інформують ІТП і працівників про зміст нововиданих наказів і розпоряджень, спрямованих на поліпшення безпечних методів праці.

На місцях виконання робіт є наочна пропаганда з охорони праці, техніки безпеки, протипожежного захисту, збереження зеленої зони та насаджень.

Згідно 0.00-4.29-96. «Положення про порядок забезпечення працівників спеціальним одягом, спеціальним взуттям та іншими засобами індивідуального захисту» на підприємстві здійснюється забезпечення працівників засобами індивідуального захисту, санітарно-побутовими пристроями, обладнанням та інвентарем, проводилося забезпечення робочих спецодягом, взуттям.

### **6.3 Аналіз умов праці на робочому місці програміста проектної групи**

Умови праці – це сукупність чинників виробничого середовища, що впливають на здоров'я і працездатність людини в процесі роботи.

У відповідності з ДНАОП 0.00-8.05-94. «Єдина державна система показників обліку умов та безпеки праці», умови і характер праці програміста можна віднести до легких фізичних робіт категорії 1б.

Отже розглянемо більш детально характер праці програміста у наступних пунктах аналізу умов праці.

### 6.3.1 Повітря робочої зони та мікрокліматичні параметри

Параметри мікроклімату на підприємстві відповідають нормам ДСН 3.3.6.042-99 «Мікроклімат виробничих приміщень». Для легких фізичних робіт категорії 1б норми мікроклімату наведені у табл. 6.1.

Таблиця 6.1 – Оптимальні параметри робочої зони

Період	Температура, °C	Відносна вологість, %	Швидкість руху повітря, м/с
Теплий	20-30	60-40	не більше 0,2
Холодний	22-25	60-40	не більше 0,2

Для виробничих умов у більшості випадків характерна сумарна дія метеорологічних факторів. Така дія може бути або антагоністичною, коли вплив одного чи декількох факторів послабляється чи повністю усувається іншими, або синергічною, коли вплив несприятливих факторів посилюють один одного.

Таким чином, збільшення швидкості повітря послабляє несприятливу дію підвищеної температури та посилює дію зниженої; підвищення вологості повітря послабляє дію як підвищеної, так і пониженої температури. Отже, в одних випадках поєднання метеорологічних факторів створює сприятливі умови для нормального протікання життєвих функцій організму, а в інших – несприятливі, що може призвести до порушення терморегуляції організму, при подовженій дії – до виникнення професійних захворювань.

### 6.3.2 Аналіз шуму в робочій зоні

Припустимі шумові показники робочих місць регламентуються ДСН 3.3.6-037-99 «Державні санітарні норми виробничого шуму, ультразвуку та інфразвуку». При нормуванні шумових характеристик робочих місць регламентують загальний шум на робочому місці незалежно від числа джерел шуму у приміщеннях і характеристик кожного окремо.

Характеристикою постійного шуму на робочих місцях є середньоквадратичні рівні звукового тиску в октавних лініях частот із середньгеометричними стандартними частотами, дані по яким приведено у табл. 6.2.

Табл. 6.2 – Припустимі рівні шуму на робочому місці

Рівні звукового тиску, дБ, в октавних смугах з середньгеометричними частотами, Гц									Рівні звуку, дБ
31,5	63	125	250	500	1000	2000	4000	8000	50
86	71	61	54	49	45	42	40	38	

### 6.3.3 Освітлення

На робочому місці є природне та штучне освітлення згідно з ДБН В.2.5-28-2006 Державні будівельні норми України «Природне і штучне освітлення» і ДСанПіН 3.3.2.-007-98.

ДБН В.2.5-28-2006 описує вісім розрядів зорової роботи, із яких перших шість характеризуються розмірами об'єкта розпізнавання. Найбільша нормована освітленість складає 5000 лк (розряд Іа), а найменша – 30 лк (розряд VIIв).

Правильне та раціональне освітлення приміщень та робочих місць – один із основних факторів, що забезпечують безпечну, здорову та високопродуктивну працю. Нераціональне освітлення (недостатнє або надто яскраве) може привести до погіршення зору та виникнення різноманітних офтальмологічних захворювань, а також зниження працездатності, втрати концентрації, втомлюваності і, як наслідок, виникнення помилок у роботі.

### 6.3.4 Електробезпека

На підприємстві (фірмі) взагалі та на обраному робочому місці (програміста) для захисту персоналу від поразки електричним струмом застосовується ізоляція струмопровідних частин електроустаткування, захисне заземлення і занулення

металевих частин устаткування, які можуть опинитися під напругою при порушенні ізоляції, захисне відключення.

Захист від проявів статичної електрики виконується відповідно до НПАОП 0.00-1.29-97. «Правила захисту від статичної електрики».

#### **6.4 Індивідуальне завдання. Розрахунок інтегральної бальної оцінки важкості праці ( $I_n$ ) на робочому місці програміста**

Категорія важкості характеризує стан організму людини, який формується під впливом умов праці. Категорію важкості праці можна визначити через інтегральну оцінку факторів робочого середовища.

Розрізняють чотири рівні впливу факторів робочого середовища на людину, необхідні для їх обліку та нормування:

- комфортна середа забезпечує оптимальну динаміку працездатності, хороше самопочуття і збереження її здоров'я;
- відносно дискомфортна робоча середа забезпечує при впливі протягом певного інтервалу часу задану працездатність і збереження здоров'я, але викликає у людини суб'єктивні відчуття та функціональні зміни, що не виходять за межі норми;
- екстремальна робоча середа призводить до зниження працездатності і викликає функціональні зміни, що виходять за межі норми, але не ведуть до патологічних змін або неможливості виконання роботи;
- понад екстремальна середа призводить до виникнення в організмі людини патологічних змін або неможливості виконання роботи.

Комплексну оцінку факторів робочого середовища проводять на основі методики фізіологічної класифікації важкості робіт.

Під вагою робіт розуміють сукупність впливу всіх факторів робочого середовища на здоров'я людини та її працездатність.

Інтегральну бальну оцінку важкості праці  $I_n$  на робочому місці можна визначити за такою методикою:

- а) сформувати таблицю виробничих факторів;
- б) визначити чисельні значення параметрів;
- в) провести оцінювання важкості кожного фактору по балам;
- г) вирахувати інтегральну бальну оцінку.

Інтегральну бальну оцінку важкості праці  $I_n$  на робочому місці можна визначити за наступною формулою:

$$I_n = 10 \cdot \left( X_{\text{оп}} + \sum x_i \cdot \frac{(6 - X_{\text{оп}})}{6(n - 1)} \right) \quad (6.1)$$

де  $X_{\text{оп}}$  – елемент умов праці, який одержав найбільшу оцінку;

$n$  – кількість врахованих елементів умов праці;

$\sum x_i$  – сума всіх елементів крім визначаючого  $X_{\text{оп}}$ .

В табл. 6.3 наведено категорії важкості та відповідні розміри доплати до тарифної ставки за інтегральною бальною оцінкою.

Таблиця 6.3 – Визначення категорії важкості праці та розміру доплати до тарифної ставки

Діапазон інтегральної бальної оцінки	Категорія важкості праці	Розмір доплати до тарифної ставки, %
До 18	I	4
19...33	II	8
34...45	III	12
45,7...53,9	IV	16
54...59	V	20

Визначимо інтегральну бальну оцінку важкості праці ( $I_n$ ) робочого місця за фактичними даними наведеними у Карті умов праці (табл. 6.4):

$$I_n = 10 * \left( 5 + (3 + 3 + 3 + 1 + 3 + 2 + 2 + 1 + 4 + 2 + 2) * \frac{(6 - 5)}{6 * (12 - 1)} \right) = 10 * \left( 5 + \frac{26 * 1}{6 * 11} \right) = 54 \quad (6.2)$$

За таблицею 6.3 визначаємо категорію важкості праці: це категорія V.

Також визначаємо розмір доплати до тарифної ставки, %: 20.

Таблиця 6.4 – Карта умов праці на робочому місці програміста

№ п/п	Назва фактору, одиниця вимірювання	Рівень фактору		Бальна оцінка
		Норматив	Факт.	
1.	Температура повітря на робочому місці у приміщенні, °С, теплий період року	20-22	25	3
2.	Швидкість руху повітря, м/с, теплий період року	до 0,3	0,6	3
3.	Відносна вологість повітря, %	40-60	75	3
4.	Інтенсивність теплового випромінювання, Вт/м <sup>2</sup>	100	25	1
5.	Шум, дБ	75	70	3
6.	Освітлення робочого місця, лк	150	275	2
7.	Мінімальний розмір об'єкта (точність зорових робіт), мм	0,25-2	0,8	2
8.	Тривалість повторювальних операцій, с	Більше 100	15	1
9.	Тривалість зосередженого спостереження, %	90	95	5
10.	Нервово-емоційна навантаження	Відповідальність за безпеку людей та особистий ризик		4

Продовження таблиці 6.4

№ п/п	Назва фактору, одиниця вимірювання	Рівень фактору		Бальна оцінка
		Норматив	Факт.	
11.	Робоче місце поза і переміщення у просторі	РМ стаціонарне, поза вільна, маса вантажу до 5 кг		2
12.	Режим праці і відпочинку	Обґрунтований, без додавання музики та гімнастики		2

Інтегральна бальна оцінка важкості праці  $I_n$  дозволяє визначити вплив умов праці на працездатність людини. Для цього спочатку визначається ступінь стомлення в умовних одиницях:

$$Y = \frac{I_n - 15,6}{0,64}, \quad (6.3)$$

$$Y = \frac{54 - 15,6}{0,64} = 60$$

де 15,6 і 0,64 – коефіцієнти регресії.

Працездатність людини визначається як величина, протилежна стомленню (в умовних одиницях):

$$R = 100 - Y, \quad (6.4)$$

$$R = 100 - 60 = 40$$

де  $R$  – умовні одиниці працездатності.

Результати розрахунків занесемо до табл. 6.5:

Таблиця 6.5 – Результати розрахунків

Змінна	Позначка	Значення
Категорія важкості праці		V
Інтегральна бальна оцінка важкості праці	$I_n$	54
Ступінь стомлення	$Y$	60
Працездатність робітників	$R$	40

### 6.5 Заходи з охорони праці. Рекомендації

Найбільшу бальну оцінку умов праці одержав параметр 9: Тривалість зосередженого спостереження – 5 балів.

Також наступні параметри умов праці оцінені доволі високо:

– Нервово-емоційне навантаження – 4 бали.

– Режим праці і відпочинку – 4 бали.

Таким чином, виділені на охорону праці кошти необхідно витратити на вдосконалення умов праці, які зазначено вище. Для оцінки привабливості проекту сформуємо нову карту умов праці – після майбутньої модернізації (табл. 6.6).

Таблиця 6.6 – Карта умов праці на робочому місці після модернізації

№ п/п	Назва фактору одиниця вимірювання	Рівень фактору		Бальна оцінка
		Норматив	Факт.	
1.	Температура повітря на робочому місці у приміщенні, °С, теплий період року	20-22	25	3
2.	Швидкість руху повітря, м/с, теплий період року	до 0,3	0,6	3
3.	Відносна вологість повітря, %	40-60	75	3
4.	Інтенсивність теплового випромінювання, Вт/м <sup>2</sup>	100	25	1
5.	Шум, дБ	75	70	3



Продовження таблиці 6.6

№ п/п	Назва фактору одиниця вимірювання	Рівень фактору		Бальна оцінка
6.	Освітлення робочого місця, лк	150	275	2
7.	Мінімальний розмір об'єкта (точність зорових робіт), мм	0,25-2	0,8	2
8.	Тривалість повторювальних операцій, с	Більше 100	15	1
9.	Тривалість зосередженого спостереження, %	90	55	3
10.	Нервово-емоційне навантаження	Складні дії по заданому плану з можливістю корекції		3
11.	Робоче місце, поза і переміщення у просторі	РМ стаціонарне, поза вільна, маса вантажу до 5 кг		2
12.	Режим праці і відпочинку	Не обґрунтований, без додавання музики та гімнастики		3

Визначимо інтегральну бальну оцінку важкості праці:

$$\begin{aligned}
 I_n &= 10 * \left( 3 + (3 + 3 + 3 + 1 + 3 + 2 + 2 + 1 + 3 + 2 + 2) * \right. \\
 &\quad \left. * \frac{(6 - 3)}{6 * (12 - 1)} \right) = 10 * \left( 3 + \frac{23 * 1}{6 * 11} \right) = 33,5
 \end{aligned}
 \tag{6.5}$$

За таблицею 7.3 визначаємо категорію важкості праці: це категорія III.

Також визначаємо розмір доплати до тарифної ставки, %: 12.

$$Y = \frac{33,5 - 15,6}{0,64} = 28 \quad (6.6)$$

де  $Y$  – ступінь стомлення в умовних одиницях.

$$R = 100 - 28 = 72 \quad (6.7)$$

де  $R$  – працездатність людини.

## 6.6 Обґрунтування запропонованих заходів з охорони праці

Для обґрунтування необхідно провести оцінку ефективності заходів. Для оцінки ефективності заходів з охорони праці проведено розрахунок зниження стомлення, зростання працездатності та продуктивності при зменшенні категорії важкості.

Зменшення втоми:

$$Y' = Y_H - Y = 28 - 60 = -32 \quad (6.8)$$

Збільшення працездатності:

$$R' = R_H - R = 72 - 40 = 30 \quad (6.9)$$

Для визначення того, яким чином зміна важкості праці впливає на працездатність людини та її продуктивність, необхідно провести розрахунок за формулою 3.10.

$$\Delta W = 100 * \left( R \frac{R_H}{R} - 1 \right) * 0,2 \quad (6.10)$$

де  $\Delta W$  – зростання продуктивності праці, %;

0,2 – емпіричний коефіцієнт, який показує вплив зростання рівня працездатності на продуктивність праці;

$R_n$ ,  $R$  – працездатність в умовних одиницях до і після впровадження заходів щодо охорони праці, які знизили важкість праці.

$$\Delta W = 100 * \left( \frac{72}{40} - 1 \right) * 0,2 = 16\% \quad (6.11)$$

Після впровадження нововведень, збільшення працездатності, зниження стомлення, продуктивність праці зросте на 16%.

## 6.7 Висновки

Було розглянуто робоче місце програміста проектної групи.

Проаналізовано умови праці на робочому місці програміста проектної групи відповідно з Державними стандартами України.

Розраховано інтегральну бальну оцінку важкості праці на робочому місці програміста.

Запропоновано та обґрунтовано заходи з охорони праці, які підвищать працездатність фахівців проектної групи, в тому числі програмістів.

## ВИСНОВКИ

Результат дипломної роботи – безкоштовне багатоплатформне мобільне застосування для розпізнавання символів догляду за текстильними виробами.

Під час виконання цієї роботи було проаналізовано актуальність розробки, вимоги користувача, предметну область, сформульовано специфікацію вимог, проаналізовано існуючі рішення та аналоги, розроблено план роботи над програмним продуктом, проаналізовано ризики, спроектовано та описано програмні модулі, структуру даних, графічний інтерфейс користувача, зібрано та розмічено набір даних, навчено на ньому нейронну мережу, реалізовано всі програмні модулі та графічний інтерфейс, сформульовано і опрацьовано сценарії тестування та проведено експериментальне випробування розробленого програмного продукту. Також була виконана частина по охороні праці, в якій були вивчені умови праці програміста проектної групи та запропоновані рекомендації щодо підвищення працездатності працівників.

У наслідок проведеного експерименту було доказано ефективність обраних методів: час на знаходження значень трьох символів було зменшено на 42% порівняно із застосунком без функції розпізнавання символів.

В майбутньому планується продовжити роботу над розпізнаванням символів догляду за текстильними виробами. А саме: розширити набір даних, удосконалити можливості нейронної мережі та оптимізувати роботу мобільного застосунку.

## ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Textile Fabric Types – different types of fabrics and their patterns.  
URL: <https://www.textileschool.com/171/textile-fabric-types-comprehensive-list-of-textile-fabrics/> (дата звернення: 25.05.2020)
2. What are the materials that are in different types a fabrics? How are fabrics different?  
URL: <http://scienceline.ucsb.edu/getkey.php?key=765> (дата звернення: 25.05.2020)
3. Full to the Brim. URL: <https://blog.closetmaid.com/2016/05/full-to-the-brim> (дата звернення: 25.05.2020)
4. 50 Recycling & Trash Statistics That Will Make You Think Twice About Your Trash. URL: <https://www.rubicon.com/blog/statistics-trash-recycling/> (дата звернення: 25.05.2020)
5. M. Roser, O. Esteban, H. Ritchie. Life Expectancy. 2013.  
URL: <https://ourworldindata.org/life-expectancy> (дата звернення: 25.05.2020)
6. B. Senftner. Re: Why computer vision is hard? URL: <https://www.quora.com/Why-computer-vision-is-hard> (дата звернення: 25.05.2020)
7. Парадокс Моравека, URL: [https://uk.wikipedia.org/wiki/Парадокс\\_Моравека](https://uk.wikipedia.org/wiki/Парадокс_Моравека) (дата звернення: 25.05.2020)
8. Automated machine learning.  
URL: [https://en.wikipedia.org/wiki/Automated\\_machine\\_learning](https://en.wikipedia.org/wiki/Automated_machine_learning) (дата звернення: 25.05.2020)
9. Mary Coats. Development of an Educational Program: Promoting Sustainability in Consumer Laundry Behavior. 2014.  
URL: [https://mountainscholar.org/bitstream/handle/10217/83883/Coats\\_colostate\\_0053N\\_12640.pdf](https://mountainscholar.org/bitstream/handle/10217/83883/Coats_colostate_0053N_12640.pdf) (дата звернення: 25.05.2020)
10. K. Fletcher, P. Goggin. The Dominant Stances on Ecodesign: A Critique. *Design Issues*. 2001. 17, №3. С. 15-25. DOI: <https://doi.org/10.1162/074793601750357150> (дата звернення: 25.05.2020)

11. Інформаційне перевантаження.

URL: [https://uk.wikipedia.org/wiki/Інформаційне\\_перевантаження](https://uk.wikipedia.org/wiki/Інформаційне_перевантаження) (дата звернення: 25.05.2020)

12. How Many Smartphones Are In The World?

URL: <https://www.bankmycell.com/blog/how-many-phones-are-in-the-world> (дата звернення: 25.05.2020)

13. Mobile Operating System Market Share Worldwide. May 2019 – May 2020.

URL: <https://gs.statcounter.com/os-market-share/mobile/worldwide> (дата звернення: 25.05.2020)

14. J. Kralicek, J. Matas, M. Busta. Care Label Recognition. *International Conference on Document Analysis and Recognition (ICDAR)*, 2019.

URL: [http://cmp.felk.cvut.cz/~matas/papers/kralicek-2019-care\\_labels-icdar.pdf](http://cmp.felk.cvut.cz/~matas/papers/kralicek-2019-care_labels-icdar.pdf) (дата звернення: 25.05.2020)

15. CL2018 Care Label Recognition Dataset,

URL: <http://cmp.felk.cvut.cz/carelabel/CL2018/> (дата звернення: 25.05.2020)

16. Object detection, URL: [https://en.wikipedia.org/wiki/Object\\_detection](https://en.wikipedia.org/wiki/Object_detection) (дата звернення: 25.05.2020)

17. Multi-label classification, [https://en.wikipedia.org/wiki/Multi-label\\_classification](https://en.wikipedia.org/wiki/Multi-label_classification) (дата звернення: 25.05.2020)

18. Multi-Label classification with One-Vs-Rest strategy,

URL: <https://prakhartechviz.blogspot.com/2019/02/multi-label-classification-python.html> (дата звернення: 25.05.2020)

19. Identification of Care Symbols Attached to Clothes,

URL: <https://www.kaggle.com/c/identification-care-symbols/overview/evaluation> (дата звернення: 25.05.2020)

20. M. Burak Bozbey Portfolio, <https://mburakbozbey.digital/#awards> (дата звернення: 25.05.2020)

21. Laundry Pro – care symbols,

URL: <https://play.google.com/store/apps/details?id=net.kloobi.laundry> (дата звернення: 25.05.2020)

22. My Care Label,

URL: <https://play.google.com/store/apps/details?id=com.cofreet.mycarelabel> (дата звернення: 25.05.2020)

23. Laundry Day - Care Symbol Reader, URL: <https://apps.apple.com/us/app/laundry-day-care-symbol-reader/id974530923> (дата звернення: 25.05.2020)

24. Complete Laundry Care, URL: <https://apps.apple.com/us/app/complete-laundry-care/id833637165> (дата звернення: 25.05.2020)

25. Who We Are, URL: <https://www.ginetex.net/article/GB/who-we-are-2> (дата звернення: 25.05.2020)

26. ISO 3758:2012(en) Textiles — Care labelling code using symbols, URL: <https://www.iso.org/obp/ui/#iso:std:iso:3758:ed-3:v1:en> (дата звернення: 25.05.2020)

27. Care Labels, URL: <https://www.coats.com/en/Information-Hub/Care-Labels> (дата звернення: 25.05.2020)

28. Japan Announces Adoption of New Care Labelling Symbols, URL: <https://hkmb.hktdc.com/en/1X0A2M15/hktdc-research/Japan-Announces-Adoption-of-New-Care-Labelling-Symbols> (дата звернення: 25.05.2020)

29. Care Symbols, URL: <https://www.ginetex.net/GB/labelling/care-symbols.asp> (дата звернення: 25.05.2020)

30. A Barometer for Textile Care Labelling in Europe, URL: <http://www.ginetex.net/article/GB/ginetex/news/a-barometer-for-textile-care-labelling-in-europe> (дата звернення: 25.05.2020)

31. Функціональні вимоги, URL: [https://uk.wikipedia.org/wiki/Функціональні\\_вимоги](https://uk.wikipedia.org/wiki/Функціональні_вимоги) (дата звернення: 25.05.2020)

32. MoSCoW method, URL: [https://en.wikipedia.org/wiki/MoSCoW\\_method](https://en.wikipedia.org/wiki/MoSCoW_method) (дата звернення: 25.05.2020)

33. Діаграма прецедентів, URL: [https://uk.wikipedia.org/wiki/Діаграма\\_прецедентів](https://uk.wikipedia.org/wiki/Діаграма_прецедентів) (дата звернення: 25.05.2020)

34. Preparing your training data,

URL: <https://cloud.google.com/vision/automl/docs/prepare> (дата звернення: 25.05.2020)

35. I. Ebenezer, Re: Is there a rule of thumb that explains the splitting of a limited dataset into two-three subsets?,

URL: [https://www.researchgate.net/post/Is\\_there\\_a\\_rule\\_of\\_thumb\\_that\\_explains\\_the\\_splitting\\_of\\_a\\_limited\\_dataset\\_into\\_two-three\\_subsets](https://www.researchgate.net/post/Is_there_a_rule_of_thumb_that_explains_the_splitting_of_a_limited_dataset_into_two-three_subsets) (дата звернення: 25.05.2020)

36. Use Case Points, URL: [https://en.wikipedia.org/wiki/Use\\_Case\\_Points](https://en.wikipedia.org/wiki/Use_Case_Points) (дата звернення: 25.05.2020)



## ДОДАТОК А. ЛІСТИНГ ПРОГРАМИ

### main.py

```
# -*- coding: utf-8 -*-

from kivy.clock import Clock
from kivy.lang import Builder
from kivy.metrics import dp
from kivy.uix.floatlayout import FloatLayout
from kivy.uix.screenmanager import CardTransition
from kivy.uix.widget import Widget
from kivymd.app import MDApp
from kivymd.icon_definitions import md_icons
from kivymd.uix.button import MDFlatButton
from kivymd.uix.dialog import MDDialog
from kivymd.uix.label import MDLabel
from kivymd.uix.list import IconLeftWidget, MDList, OneLineIconListItem
from kivymd.uix.snackbar import Snackbar
from kivymd.uix.tab import MDTabsBase

import ginetex_symbols as gs
from android.permissions import Permission, request_permissions

class NeuralNet:
    model_path = ""
    labels_path = ""
    threshold = 0

    _initialized = False
    _interpreter = None
    _input_details = None
    _output_details = None
    _labels = []

    def __init__(self, model_path: str, labels_path: str, threshold: int = 50) -
> None:
        self.model_path = model_path
        self.labels_path = labels_path
        self.threshold = threshold

    def _init_model(self) -> None:
        import tfLite_runtime.interpreter as tflite

        self._interpreter = tflite.Interpreter(model_path=self.model_path)
        self._input_details = self._interpreter.get_input_details()
        self._output_details = self._interpreter.get_output_details()
        self._interpreter.allocate_tensors()

        with open(file=self.labels_path, mode="r") as f:
            self._labels = [line.strip() for line in f.readlines()]

    def _filter_results(
        self, prediction: numpy.ndarray
    ) -> dict[str, dict[str, dict[str, str]]]:
        end_results = {}

        for label_num, score in enumerate(prediction):
            score = round(score / 100 * threshold)
            if score > self.threshold:
                name = self._labels[label_num]
                entry = {"name": name, "score": score}
```

```

        category_name = name.split(sep="_", maxsplit=1)[0]

        category_entries = end_results.get(category_name, None)
        if not category_entries or score > category_entries["score"]:
            end_results[category_name] = entry

    return end_results

def make_prediction(self, img: numpy.ndarray) -> numpy.ndarray:
    if not self._initialized:
        self._init_model()

    self._interpreter.set_tensor(self._input_details[0]["index"], [img])
    self._interpreter.invoke()

    prediction = self._interpreter.get_tensor(self._output_details[0]["index"]
[0])

    return self._filter_results(prediction=prediction)

class App(MDApp):
    nn = None

    transition = CardTransition()

    _permissions = [Permission.CAMERA]

    def __init__(self, nn: NeuralNet, *args, **kwargs) -> None:
        self.nn = nn
        super().__init__(*args, **kwargs)

    def _ensure_permissions(self) -> bool:
        granted = None

        def get_answers(permissions: list[str], answers: list[bool]) -> None:
            granted = all(answers)

        request_permissions(permissions=self._permissions, callback=get_answers)

        return granted

    def _set_icons(self) -> None:
        ginetex_icons = {}

        for category in gs.symbols.values():
            for symbol in category:
                ginetex_icons[symbol] = category[symbol]["icon"]

        md_icons.update(ginetex_icons)

    def _create_tabs(self) -> None:
        class Tab(FloatLayout, MDTabsBase):
            def __init__(self, name: StringProperty, glyph: str, title: str) -
> None:
                self.name = name
                self.glyph = glyph
                self.title = title
                super().__init__()

        first_tab = None

        for category, meta in gs.category_meta.items():

```

```

        tab = Tab(name=category, glyph=meta["icon"], title=meta["translation"])
    )

    if not first_tab:
        first_tab = tab

    self.root.ids.symbols_list_screen.ids.tabs.add_widget(tab)

    self.populate_tab(instance_tab=first_tab)

    def _create_list_item(self, category: str, symbol: str) -
> OneLineIconListItem:
        symbol_item = gs.symbols[category][symbol]

        list_item = OneLineIconListItem(
            text="[size=12sp]{info}[/size]".format(info=symbol_item["text"]),
            font_style="Body1",
            _no_ripple_effect=True,
        )

        list_item.add_widget(IconLeftWidget(icon=symbol, font_name="ginetex.ttf",)
    )

    return list_item

def _show_error(self) -> bool:
    dialog = MDDialog(
        text="Не було знайдено жодного символу.\n\n\nПереконайтесь, що символи
підтримуються сканером та що їх добре видно на фотографії.",
        size_hint=[0.6, 0.8],
        radius=[25, 25, 25, 25],
        padding=[15, 15, 15, 15],
    )
    dialog.ids.container.children[3].halign = "center"

    dialog.open()
    Clock.schedule_once(lambda _: dialog.dismiss(), 4)

    return False

def _show_prediction(
    self, prediction: dict[str, dict[str, dict[str, str]]]
) -> bool:
    results = self.root.ids.results_screen.ids.results

    for category, category_labels in gs.symbols.items():
        if category in prediction:
            section_list = MDList()
            section_list.add_widget(
                MDLabel(
                    text=gs.category_meta[category]["translation"],
                    font_style="H6",
                    halign="center",
                    size_hint_y=None,
                )
            )
            results_label = prediction[category]["name"]

            if results_label.endswith("yes"):
                for label in category_labels:
                    if not label.endswith("no") and label.startswith(
                        results_label.rsplit(sep="_", maxsplit=1)[0]
                    ):

```

```

        section_list.add_widget(
            self._create_list_item(category=category, symbol=l
abel)
        )

        elif results_label.endswith(("0", "p")):
            for label in category_labels:
                if label.startswith(results_label):
                    section_list.add_widget(
                        self._create_list_item(category=category, symbol=l
abel)
                    )

            else:
                section_list.add_widget(
                    self._create_list_item(category=category, symbol=results_l
abel)
                )

            results.add_widget(section_list)

        return True

def build(self) -> ScreenManager:
    self._set_icons()

    with open("main.kv", encoding="utf-8", mode="r") as f:
        return Builder.load_string(f.read())

def on_start(self) -> None:
    self._create_tabs()

def populate_tab(
    self,
    instance_tabs: MDTabs = None,
    instance_tab: Tab = None,
    instance_tab_label: MDLabel = None,
    tab_text: StringProperty = None,
) -> None:
    if not instance_tab.populated:
        for symbol in gs.symbols[instance_tab.name]:
            instance_tab.ids.symbols.add_widget(
                self._create_list_item(category=instance_tab.name, symbol=symb
ol)
            )

        instance_tab.populated = True

def show_info(self, *args) -> None:
    dialog = MDDialog(
        title="Інформація",
        text=u"[size=18sp]Власником символів догляду за текстильними виробами,
        які представлено в цьому застосунку, є міжнародна асоціація по маркуванню текстил
        ьних виробів [b]GINETEX[/b]. Всі права зберігаються за правовласником.\n\nСканер м
        оже показувати схожі за виглядом символи. Наразі розпізнаються наступні:\n[size=28
        sp][font=ginetex.ttf]twer89qod\nJasnbvUWE[/font][[/size]\n\nРекомендується прати ре
        чі якомога рідше та за мінімальної температури води.[/size]",
        size_hint=(0.8, 0.8),
        buttons=[
            MDFlatButton(
                text="ЗРОЗУМІЛО",
                text_color=self.theme_cls.primary_color,
                on_release=lambda _: dialog.dismiss(),
            )
        ]
    )

```

```

        ],
    )
    dialog.open()

def prepare_scanner(self) -> bool:
    self._ensure_permissions()
    if not self._ensure_permissions():
        SnackBar(
            text="Функція сканування потребує певних дозволів", duration=1.5
        ).show()

        return False

    return True

def scan_photo(self, scanner: Widget) -> bool:
    import cv2

    file_name = "photo.jpg"
    scanner.export_to_png(filename=file_name)

    img = cv2.imread("../" + file_name)
    resized_img = cv2.resize(img, (224, 224))

    prediction = self.nn.make_prediction(resized_img)

    return self._show_prediction(prediction) if prediction else self._show_err
or()

if __name__ == "__main__":
    App(
        nn=NeuralNet(
            model_path="app/model.tflite", labels_path="app/labels.txt", threshold
=40
        )
    ).run()

```

## main.kv

```

ScreenManager:
    transition: app.transition

    SymbolsListScreen:
        id: symbols_list_screen
        name: "SymbolsListScreen"
        on_pre_enter:
            root.transition.direction = "up"

    CameraScreen:
        id: camera_screen
        name: "CameraScreen"
        on_pre_enter:
            root.transition.direction = "down"

    ResultsScreen:
        id: results_screen
        name: "ResultsScreen"
        on_pre_enter:
            root.transition.direction = "down"

<SymbolsListScreen@Screen>:
    MDBoxLayout:

```

```

MDTabs:
    id: tabs

MDBottomAppBar:
    MDToolbar:
        left_action_items: self.parent.left_action_items
        on_action_button:
            if app.prepare_scanner():
                root.manager.current = "CameraScreen"

<CameraScreen@Screen>:
    MDBoxLayout:
        Widget:
            id: scanner
            Camera:
                id: camera
                index: 0
                resolution: (1920, 1080)

                size_hint: (None, None)
                width: root.height
                height: root.width
                allow_stretch: True
                keep_ratio: True

                center: scanner.center
                canvas.before:
                    PushMatrix
                    Rotate:
                        angle: -90
                        origin: self.center
                canvas.after:
                    PopMatrix

        MDBottomAppBar:
            MDToolbar:
                left_action_items:
                    [["arrow-
left", lambda _: setattr(root.manager, "current", "SymbolsListScreen")]] \
                    + self.parent.left_action_items
                on_action_button:
                    if app.scan_photo(scanner):
                        root.manager.current = "ResultsScreen"

<ResultsScreen@Screen>:
    on_leave:
        results.clear_widgets()

    ScrollView:
        MDBoxLayout:
            id: results
            adaptive_height: True
            padding: [0, 0, 0, dp(60)]

        MDBottomAppBar:
            MDToolbar:
                left_action_items:
                    [["arrow-
left", lambda _: setattr(root.manager, "current", "SymbolsListScreen")]] \
                    + self.parent.left_action_items
                on_action_button:
                    root.manager.current = "CameraScreen"

```

```

<MDBoxLayout>:
    orientation: "vertical"

<MDTabs>:
    tab_bar_height: dp(60)
    tab_indicator_height: dp(2)
    text_color_normal: app.theme_cls.bg_light
    color_indicator: app.theme_cls.bg_light
    elevation: 10
    tab_indicator_anim: True
    anim_threshold: 0
    on_tab_switch: app.populate_tab(*args)

<Tab@FloatLayout+MDTabsBase>:
    text:
        "[size=30sp][font=ginetex.ttf]{glyph}[/font][[/size]\n[size=15sp][b]{title}[/b][[/size]" \
        .format(glyph=self.glyph,title=self.title)
    populated: False

    ScrollView:
        MDList:
            padding: (0, dp(20))
            id: symbols

<MDToolbar>:
    type: "bottom"
    title: "Laundry Scan"
    icon: "camera"
    mode: "end"
    spacing: dp(5)
    on_parent: self.action_button.text_color = app.theme_cls.bg_light

<MDBottomAppBar>:
    left_action_items: [["information-outline", app.show_info]]

```

## ginetex\_symbols.py

```
# -*- coding: utf-8 -*-
```

```

symbols = {
    "WASH": {
        "WASH_no": {
            "icon": "z", "text": "Прання заборонено"
        },
        "WASH_hand": {
            "icon": "t", "text": "Ручне прання, температура до 40°C"
        },
        "WASH_30": {
            "icon": "w", "text": "Звичайне прання, температура до 30°C"
        },
        "WASH_30_mild": {
            "icon": "e", "text": "Делікатне прання, температура до 30°C"
        },
        "WASH_30_very_mild": {
            "icon": "r", "text": "Особливо делікатне прання, температура до 30°C"
        },
        "WASH_40": {
            "icon": "8", "text": "Звичайне прання, температура до 40°C"
        },
        "WASH_40_mild": {
            "icon": "9", "text": "Делікатне прання, температура до 40°C"
        }
    }
}

```

```

    },
    "WASH_40_very_mild": {
      "icon": "q", "text": "Особливо делікатне прання, температура до 40°C"
    },
    "WASH_60": {
      "icon": "4", "text": "Звичайне прання, температура до 60°C"
    },
    "WASH_60_mild": {
      "icon": "5", "text": "Делікатне прання, температура до 60°C"
    },
    "WASH_70": {
      "icon": "3", "text": "Звичайне прання, температура до 70°C"
    },
    "WASH_95": {
      "icon": "2", "text": "Звичайне прання, температура до 95°C"
    },
  },
  "BLEACH": {
    "BLEACH_no": {
      "icon": "o", "text": "Відбілювання заборонено"
    },
    "BLEACH_nonchlorine": {
      "icon": "i", "text": "Можна відбілювати засобами без вмісту хлору"
    },
    "BLEACH_any": {
      "icon": "u", "text": "Можна відбілювати будь-якими засобами"
    },
  },
  "DRY": {
    "DRY_tumble_no": {
      "icon": "d", "text": "Барабанну сушку заборонено"
    },
    "DRY_tumble_mild": {
      "icon": "s", "text": "Звичайна барабанна сушка, температура до 60°C"
    },
    "DRY_tumble_normal": {
      "icon": "a", "text": "Делікатна барабанна сушка, температура до 80°C"
    },
    "DRY_tumble_yes": {
      "icon": "J", "text": "Барабанну сушку дозволено"
    },
    "DRY_natural": {
      "icon": "p", "text": "Звичайну сушку дозволено"
    },
    "DRY_vertical": {
      "icon": "f", "text": "Сушка у вертикальному положенні"
    },
    "DRY_vertical_shade": {
      "icon": "k", "text": "Сушка у вертикальному положенні в тіні"
    },
    "DRY_vertical_drip": {
      "icon": "g", "text": "Сушка без віджиму у вертикальному положенні"
    },
    "DRY_vertical_drip_shade": {
      "icon": "l", "text": "Сушка без віджиму у верт. положенні в тіні"
    },
    "DRY_horizontal": {
      "icon": "h", "text": "Сушка у горизонтальному положенні"
    },
    "DRY_horizontal_shade": {
      "icon": "y", "text": "Сушка у горизонтальному положенні в тіні"
    },
    "DRY_horizontal_drip": {

```



```

        "icon": "j", "text": "Сушка без віджиму у горизонтальному положенні"
    },
    "DRY_horizontal_drip_shade": {
        "icon": "x", "text": "Сушка без віджиму у гор. положенні в тіні"
    },
},
"IRON": {
    "IRON_no": {
        "icon": "m", "text": "Прасування заборонено"
    },
    "IRON_low": {
        "icon": "n", "text": "Прасування при температурі до 110°C"
    },
    "IRON_moderate": {
        "icon": "b", "text": "Прасування при температурі до 150°C"
    },
    "IRON_hot": {
        "icon": "v", "text": "Прасування при температурі до 200°C"
    },
},
"PROF": {
    "PROF_wet_no": {
        "icon": "A", "text": "Мокру професійну чистку заборонено"
    },
    "PROF_wet_w": {
        "icon": "I", "text": "Звичайна мокра професійна чистка"
    },
    "PROF_wet_w_mild": {
        "icon": "O", "text": "Делікатна мокра професійна чистка"
    },
    "PROF_wet_w_very_mild": {
        "icon": "P", "text": "Особливо делікатна мокра професійна чистка"
    },
    "PROF_dry_no": {
        "icon": "U", "text": "Суху професійну чистку заборонено"
    },
    "PROF_dry_f": {
        "icon": "T", "text": "Звичайна суха професійна чистка (вуглеводні)"
    },
    "PROF_dry_f_mild": {
        "icon": "Z", "text": "Делікатна суха професійна чистка (вуглеводні)"
    },
    "PROF_dry_p": {
        "icon": "W", "text": "Звичайна суха професійна чистка (перхлоретилен)"
    },
    "PROF_dry_p_mild": {
        "icon": "E", "text": "Делікатна суха професійна чистка (перхлоретилен)"
    }
}
}

category_meta = {
    "WASH": {"icon": "1", "translation": "ПРАННЯ"},
    "BLEACH": {"icon": "u", "translation": "ВІДБІЛЮВАННЯ"},
    "DRY": {"icon": "p", "translation": "СУШКА"},
    "IRON": {"icon": "c", "translation": "ПРАСУВАННЯ"},
    "PROF": {"icon": "Q", "translation": "ХІМЧИСТКА"}
}

```

**dict.txt**

DRY\_tumble\_yes  
 PROF\_dry\_no  
 WASH\_30  
 PROF\_dry\_p  
 BLEACH\_no  
 IRON\_yes  
 WASH\_hand  
 DRY\_tumble\_no  
 WASH\_40

**labeled\_photos.csv**

TRAIN,gs://label-scan/1.jpg,WASH\_hand,BLEACH\_no,IRON\_yes,PROF\_dry\_p  
 TEST,gs://label-scan/2.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_p  
 VALIDATION,gs://label-scan/3.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_no  
 TRAIN,gs://label-scan/5.jpg,WASH\_hand,BLEACH\_no,IRON\_yes,PROF\_dry\_p  
 TRAIN,gs://label-scan/6.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_p  
 VALIDATION,gs://label-scan/7.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_p  
 TRAIN,gs://label-scan/8.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_p  
 VALIDATION,gs://label-scan/10.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_no  
 TRAIN,gs://label-scan/11.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_p  
 TRAIN,gs://label-scan/12.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_p  
 TRAIN,gs://label-scan/13.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_p  
 TRAIN,gs://label-scan/14.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_p  
 TRAIN,gs://label-scan/15.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_p  
 VALIDATION,gs://label-scan/18.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_no  
 TEST,gs://label-scan/19.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_yes,IRON\_yes,PROF\_dry\_no  
 VALIDATION,gs://label-scan/20.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_no  
 TRAIN,gs://label-scan/21.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_no  
 VALIDATION,gs://label-scan/22.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_no  
 TEST,gs://label-scan/23.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_yes,IRON\_yes,PROF\_dry\_no  
 TRAIN,gs://label-scan/25.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_no  
 TEST,gs://label-scan/26.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_no  
 TEST,gs://label-scan/27.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_no  
 VALIDATION,gs://label-scan/28.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_no  
 TRAIN,gs://label-scan/29.jpg,WASH\_40,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_no  
 TRAIN,gs://label-scan/30.jpg,WASH\_40,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_no  
 TRAIN,gs://label-scan/31.jpg,WASH\_hand,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_p  
 TRAIN,gs://label-scan/32.jpg,WASH\_hand,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_no  
 TRAIN,gs://label-scan/33.jpg,WASH\_hand,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_p  
 TRAIN,gs://label-scan/34.jpg,WASH\_hand,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_p  
 VALIDATION,gs://label-scan/36.jpg,WASH\_hand,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_p  
 TRAIN,gs://label-scan/39.jpg,WASH\_40,BLEACH\_no,DRY\_tumble\_yes,IRON\_yes,PROF\_dry\_p  
 TRAIN,gs://label-scan/40.jpg,WASH\_40,BLEACH\_no,DRY\_tumble\_yes,IRON\_yes,PROF\_dry\_no  
 TRAIN,gs://label-scan/52.jpg,WASH\_40,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_p  
 VALIDATION,gs://label-scan/54.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_yes,IRON\_yes,PROF\_dry\_no  
 TEST,gs://label-scan/55.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_p  
 VALIDATION,gs://label-scan/56.jpg,WASH\_40,BLEACH\_no,DRY\_tumble\_yes,IRON\_yes,PROF\_dry\_no  
 TRAIN,gs://label-scan/57.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_no  
 TEST,gs://label-scan/58.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_p

```
TRAIN,gs://label-scan/61.jpg,WASH_30,BLEACH_no,DRY_tumble_no,IRON_yes,PROF_dry_p  
VALIDATION,gs://label-  
scan/63.jpg,WASH_30,BLEACH_no,DRY_tumble_no,IRON_yes,PROF_dry_no  
VALIDATION,gs://label-  
scan/66.jpg,WASH_40,BLEACH_no,DRY_tumble_no,IRON_yes,PROF_dry_p  
TRAIN,gs://label-scan/71.jpg,WASH_30,BLEACH_no,DRY_tumble_no,IRON_yes,PROF_dry_p  
TRAIN,gs://label-  
scan/73.jpg,WASH_hand,BLEACH_no,DRY_tumble_no,IRON_yes,PROF_dry_no  
VALIDATION,gs://label-  
scan/74.jpg,WASH_hand,BLEACH_no,DRY_tumble_no,IRON_yes,PROF_dry_p  
TRAIN,gs://label-scan/76.jpg,WASH_30,BLEACH_no,DRY_tumble_no,IRON_yes,PROF_dry_no  
TRAIN,gs://label-scan/78.jpg,WASH_40,BLEACH_no,DRY_tumble_no,IRON_yes,PROF_dry_no  
TEST,gs://label-scan/79.jpg,WASH_40,BLEACH_no,DRY_tumble_no,IRON_yes,PROF_dry_no  
TEST,gs://label-scan/80.jpg,WASH_40,BLEACH_no,DRY_tumble_no,IRON_yes,PROF_dry_no  
TRAIN,gs://label-scan/82.jpg,WASH_40,BLEACH_no,DRY_tumble_no,IRON_yes,PROF_dry_no  
TRAIN,gs://label-scan/84.jpg,WASH_40,BLEACH_no,DRY_tumble_no,IRON_yes,PROF_dry_no  
TEST,gs://label-scan/88.jpg,WASH_40,BLEACH_no,DRY_tumble_yes,IRON_yes,PROF_dry_p  
TRAIN,gs://label-scan/89.jpg,WASH_40,BLEACH_no,DRY_tumble_yes,IRON_yes,PROF_dry_no  
TRAIN,gs://label-scan/90.jpg,WASH_30,BLEACH_no,DRY_tumble_yes,IRON_yes,PROF_dry_no  
TRAIN,gs://label-scan/91.jpg,WASH_40,BLEACH_no,DRY_tumble_yes,IRON_yes,PROF_dry_no  
TEST,gs://label-scan/92.jpg,WASH_40,BLEACH_no,DRY_tumble_yes,IRON_yes,PROF_dry_no  
TRAIN,gs://label-scan/93.jpg,WASH_40,BLEACH_no,DRY_tumble_yes,IRON_yes,PROF_dry_no  
TRAIN,gs://label-scan/94.jpg,WASH_40,BLEACH_no,DRY_tumble_yes,IRON_yes,PROF_dry_no  
TRAIN,gs://label-scan/97.jpg,WASH_30,BLEACH_no,DRY_tumble_no,IRON_yes,PROF_dry_p  
TRAIN,gs://label-scan/102.jpg,WASH_40,BLEACH_no,DRY_tumble_no,IRON_yes,PROF_dry_p  
TEST,gs://label-scan/103.jpg,WASH_30,BLEACH_no,DRY_tumble_no,IRON_yes,PROF_dry_no  
TRAIN,gs://label-scan/104.jpg,WASH_40,BLEACH_no,DRY_tumble_no,IRON_yes  
TRAIN,gs://label-scan/105.jpg,WASH_30,BLEACH_no,DRY_tumble_no,IRON_yes,PROF_dry_no  
TRAIN,gs://label-scan/106.jpg,WASH_40,BLEACH_no,DRY_tumble_no,IRON_yes,PROF_dry_no  
TRAIN,gs://label-scan/107.jpg,WASH_30,BLEACH_no,DRY_tumble_no,IRON_yes,PROF_dry_p  
TRAIN,gs://label-scan/109.jpg,WASH_30,BLEACH_no,DRY_tumble_no,IRON_yes,PROF_dry_no  
VALIDATION,gs://label-  
scan/110.jpg,WASH_30,BLEACH_no,DRY_tumble_yes,IRON_yes,PROF_dry_no  
TRAIN,gs://label-scan/111.jpg,WASH_30,BLEACH_no,DRY_tumble_no,IRON_yes,PROF_dry_p  
TRAIN,gs://label-scan/114.jpg,WASH_40,BLEACH_no,DRY_tumble_no,IRON_yes,PROF_dry_p  
VALIDATION,gs://label-  
scan/115.jpg,WASH_40,BLEACH_no,DRY_tumble_yes,IRON_yes,PROF_dry_no  
TRAIN,gs://label-scan/118.jpg,WASH_30,BLEACH_no,DRY_tumble_yes,IRON_yes,PROF_dry_p  
TRAIN,gs://label-scan/120.jpg,WASH_40,BLEACH_no,DRY_tumble_no,IRON_yes,PROF_dry_p  
TRAIN,gs://label-scan/121.jpg,WASH_30,BLEACH_no,DRY_tumble_no,IRON_yes,PROF_dry_no  
TRAIN,gs://label-scan/122.jpg,WASH_30,BLEACH_no,DRY_tumble_no,IRON_yes,PROF_dry_no  
TRAIN,gs://label-  
scan/123.jpg,WASH_hand,BLEACH_no,DRY_tumble_no,IRON_yes,PROF_dry_p  
TRAIN,gs://label-scan/124.jpg,WASH_30,BLEACH_no,DRY_tumble_no,IRON_yes,PROF_dry_no  
VALIDATION,gs://label-  
scan/126.jpg,WASH_30,BLEACH_no,DRY_tumble_yes,IRON_yes,PROF_dry_p  
TRAIN,gs://label-scan/127.jpg,WASH_40,BLEACH_no,DRY_tumble_yes,IRON_yes,PROF_dry_p  
TRAIN,gs://label-scan/128.jpg,WASH_40,BLEACH_no,DRY_tumble_yes,IRON_yes,PROF_dry_p  
TRAIN,gs://label-scan/129.jpg,WASH_30,BLEACH_no,DRY_tumble_no,IRON_yes,PROF_dry_no  
VALIDATION,gs://label-  
scan/130.jpg,WASH_30,BLEACH_no,DRY_tumble_no,IRON_yes,PROF_dry_no  
TEST,gs://label-scan/131.jpg,WASH_30,BLEACH_no,DRY_tumble_no,IRON_yes,PROF_dry_no  
TRAIN,gs://label-scan/133.jpg,WASH_30,BLEACH_no,DRY_tumble_no,IRON_yes,PROF_dry_no  
TEST,gs://label-scan/135.jpg,WASH_30,BLEACH_no,DRY_tumble_no,IRON_yes,PROF_dry_p  
VALIDATION,gs://label-  
scan/137.jpg,WASH_40,BLEACH_no,DRY_tumble_no,IRON_yes,PROF_dry_no  
TRAIN,gs://label-scan/138.jpg,WASH_40,BLEACH_no,DRY_tumble_no,IRON_yes  
TEST,gs://label-scan/140.jpg,WASH_40,BLEACH_no,DRY_tumble_no,IRON_yes,PROF_dry_no  
VALIDATION,gs://label-scan/141.jpg,WASH_40,BLEACH_no,DRY_tumble_no,IRON_yes  
TRAIN,gs://label-scan/143.jpg,WASH_40,BLEACH_no,DRY_tumble_no,IRON_yes,PROF_dry_p  
TRAIN,gs://label-scan/146.jpg,WASH_30,BLEACH_no,DRY_tumble_no,IRON_yes,PROF_dry_no  
TEST,gs://label-scan/147.jpg,WASH_30,BLEACH_no,DRY_tumble_no,IRON_yes,PROF_dry_no  
TRAIN,gs://label-scan/151.jpg,WASH_30,BLEACH_no,DRY_tumbleno,IRON yes,PROF dry p
```

VALIDATION,gs://label-  
 scan/152.jpg,WASH\_40,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_no  
 TRAIN,gs://label-scan/155.jpg,WASH\_40,BLEACH\_no,DRY\_tumble\_no,IRON\_yes  
 TRAIN,gs://label-scan/157.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_no  
 TRAIN,gs://label-scan/158.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_no  
 TRAIN,gs://label-scan/162.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_no  
 TRAIN,gs://label-scan/164.jpg,WASH\_hand,BLEACH\_no,DRY\_tumble\_no,IRON\_yes  
 TEST,gs://label-scan/168.jpg,WASH\_hand,BLEACH\_no,DRY\_tumble\_no,IRON\_yes  
 TEST,gs://label-scan/169.jpg,WASH\_40,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_no  
 TRAIN,gs://label-scan/171.jpg,WASH\_40,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_no  
 TEST,gs://label-scan/173.jpg,WASH\_hand,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_p  
 TRAIN,gs://label-  
 scan/175.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_yes,IRON\_yes,PROF\_dry\_no  
 TRAIN,gs://label-scan/176.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_no  
 TRAIN,gs://label-  
 scan/177.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_yes,IRON\_yes,PROF\_dry\_no  
 TRAIN,gs://label-scan/179.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_p  
 TEST,gs://label-scan/180.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_no  
 TRAIN,gs://label-  
 scan/181.jpg,WASH\_hand,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_p  
 TRAIN,gs://label-  
 scan/183.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_yes,IRON\_yes,PROF\_dry\_no  
 VALIDATION,gs://label-  
 scan/184.jpg,WASH\_40,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_p  
 TRAIN,gs://label-scan/187.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_no  
 VALIDATION,gs://label-scan/188.jpg,WASH\_40,BLEACH\_no,IRON\_yes,PROF\_dry\_p  
 TRAIN,gs://label-scan/191.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_p  
 TRAIN,gs://label-scan/193.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_p  
 TRAIN,gs://label-  
 scan/194.jpg,WASH\_40,BLEACH\_no,DRY\_tumble\_yes,IRON\_yes,PROF\_dry\_no  
 TRAIN,gs://label-scan/195.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_no,IRON\_yes,PROF\_dry\_no  
 TRAIN,gs://label-scan/196.jpg,WASH\_30,BLEACH\_no,DRY\_tumble\_yes,IRON\_yes