Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
ФІС
(назва факультету)
Програмна інженерія
(повна назва кафедри)

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту (роботи)

магістр
(освітній ступінь (освітньо-кваліфікаційний рівень))

на тему: Розробка програмного забезпечення для визначення предметів та розпізнавання тексту з використанням машинного навчання та технології Voice Over

Виконав: студент (ка) VI курсу, групи СПм-61
спеціальності (напряму підготовки)
121 Інженерія програмного забезпечення
(шифр і назва спеціальності (напряму підготовки))

	(підпис)	Біломазур К В (прізвище та ініціали)
Керівник	(підпис)	(прізвище та ініціали)
Нормоконтроль	(підпис)	(прізвище та ініціали)
Рецензент	(підпис)	(прізвище та ініціали)

м. Тернопіль – 2019

АНОТАЦІЯ

Магістерська робота на тему «Розробка програмної системи для визначення предметів та розпізнавання тексту з використанням машинного навчання та підтримкою технології Voice Over» Біломазура Костянтина Володимировича. – Тернопільський національний технічний університет імені Івана Пулюя, Факультет комп'ютерно-інформаційних систем і програмної інженерії, Кафедра програмної інженерії, група СПм–61 // Тернопіль, 2019.

С. – 97, рис. – 22, табл. – 12, слайдів. – 10, додат. – 3, бібліогр. – 19.

Дипломна робота присвячена створенню програмного забезпечення для розпізнавання тексту та визначення предметів з використанням машинного навчання.

Об’єктом дослідження є оптимізація процесу перенесення тексту з друкованого чи рукописного формату в електронний з використанням машинного навчання. Озвучування отриманого результату для користувачів за допомогою технології Voice Over.

Предметом дослідження: є програмна система розпізнавання тексту та предметів з використанням машинного навчання

Мета роботи: реалізувати програмну систему для розпізнавання тексту та предметів у поєднанні із технологією Voice Over.

При створені програмного забезпечення використовувались такі технології розробки як об’єктно-орієнтоване програмування, середовище розробки Xcode, для взаємодії з базою даних - Core Data. Система реалізована на мові програмування Swift.

Ключові слова: програмна система, Xcode, Swift, Core Data, Voice Over.

ABSTRACT

Magister work « Developing a software system for object detection and text recognition using machine learning and Voice Over technology support» Bilomazur Kostyntyn. – Ternopil Ivan Pul'uj National Technical University, Faculty of Computer Information Systems and Software Engineering, Software engineering department, group SPm-61 // Ternopil, 2019.

Pages. – 97, pictures. – 22, tables. – 12, slides – 10, add. – 3, bibl.ref. – 19.

The diploma thesis is devoted to the creation of software for recognizing text and identifying subjects using machine learning.

The object of the study is to optimize the process of transferring text from print or handwriting to electronic using machine learning. Voice output for users with Voice Over technology.

Subject of study: there is a software system for recognizing text and objects using machine learning

Purpose: to implement a software system for recognizing text and objects in combination with Voice Over technology.

When software was creating, we used such development technologies as object-oriented programming, the Xcode development environment, to interact with the database - Core Data. The system is implemented in Swift programming language.

Key words: program system, Xcode, Swift, Core Data, Voice Over.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПЗ – програмне забезпечення.

ПС – програмна система, комплекс програмного забезпечення.

ПК – персональний комп’ютер, робоча машина для розробки та виконання програм.

ООП – (Об’єктно-орієнтоване програмування) парадигма програмування, в якій основою є класи та об’єкти, які між собою взаємодіють.

MVC – Модель – Представлення – Контролер (Model-View-Controller) архітектурний шаблон.

MVVM – Модель – Представлення – Модель Представлення (Model-View-View Model) архітектурний шаблон.

UML – (Unified Modeling Language) уніфікована мова графічного представлення та об’єктного моделювання в області розробки програмного забезпечення парадигми об’єктно-орієнтованого програмування.

ОП – охорона праці.

ВДТ – відео дисплейний термінал.

ДСанПіН - Державні санітарні правила і норми.

НПАОП - Нормативно-правовий акт з охорони праці.

ЗМІСТ

ВСТУП 8
1. ПРОЕКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ 9
1.1 Аналіз вимог до програмної системи 9
1.1.1 Аналіз предметної області 9
1.1.1.1 Штучний інтелект 9
1.1.1.2 Машинне навчання 11
1.1.1.3 Розпізнавання тексту 13
1.1.1.4 Розпізнавання предметів 15
1.1.2 Постановка задачі 17
1.1.3 Пошук акторів та варіантів використання 18
1.1.4 Опис ключових варіантів використання 22
1.2 Проектування програмної системи 28
1.2.1 Вибір моделі розробки 28
1.2.2 Вибір архітектурного шаблону 33
1.2.3 Побудова діаграми класів 36
2. РЕАЛІЗАЦІЯ ПРОГРАМНОЇ СИСТЕМИ 50
2.1 Вибір мови програмування 50
2.2 Вибір інструменту для збереження даних 52
2.3 Реалізація основних класі та методів системи 54
2.4 Використання системи 57
2.4.1 Системні вимоги 57
2.4.2 Використання та тестування системи 58
3. ОРГАНІЗАЦІЙНО-ЕКОНОМІЧНА ЧАСТИНА 62
3.1 Загальний підхід до визначення економічної ефективності розробки. 62
3.2 Розрахунок вартості процесу розробки та оцінка економічної ефективності проекту 64
4. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ 75
4.1 Охорона праці 75
4.2 Електробезпека користувачів ПК 78

ВИСНОВКИ..... 84
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ..... 86
ДОДАТКИ..... 89

ВСТУП

У сучасному світі є чимало людей з обмеженими можливостями певного виду. Одною з таких проблем є – поганий зір. Ця вада має надзвичайно сильний вплив на людське життя, адже за допомогою зору людина отримує 80-90% інформації про навколишній світ. За даними ВООЗ, проблеми з зором мають близько 300 мільйонів людей у світі. З них 43% страждають порушеннями рефракції - короткозорістю (міопією), далекозорістю (гіперметропією) та астигматизмом. Для того, щоб допомогти людям використовувати сучасні технології та застосунки Apple розробили технологію VoiceOver.

Також доволі частою є ситуація, коли існує певний написаний чи надрукований текст у паперовому форматі, який потрібно перетворити у електронний формат, проредагувати та відправити комусь у мережі Інтернет. Для цього необхідно вручну переписувати даний текст, а це займає деякий час. Інколи буває проблема із розпізнавання написаного тексту людським оком. З використанням штучного інтелекту є можливість суттєво оптимізувати даний процес. Ком'ютер самостійно опрацює даний текст і переведе його у електронний формат. Користувач зможе спокійно поділитися даним текстом у мережі.

Об'єднавши використання штучного інтелекту та технології Voice Over можна досягнути наступне – спростити процес переведення тексту з паперового формату у електронний та допомогти людям з поганим зором у розпізнаванні тексту та предметів. Процес відтворення результат у звуковому форматі допомагає не тільки користувачам, які мають поганий зір, а й іншим, адже вам не потрібно дивитися не екран телефону, щоб знати результат опрацювання.

Описана нижче магістерська робота відображає процес проектування та реалізації системи розпізнавання тексту та предметів з використанням машинного навчання та підтримкою технології Voice Over.

1. ПРОЕКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

1.1 Аналіз вимог до програмної системи

1.1.1 Аналіз предметної області

1.1.1.1 Штучний інтелект

Штучний інтелект - це моделювання процесів інтелекту людини за допомогою машин, особливо комп'ютерних систем. Ці процеси включають навчання (здобуття інформації та правила користування інформацією), міркування (використання правил для отримання приблизних чи певних висновків) та самокорекцію. Окремі програми, які використовують штучний інтелект включають експертні системи, розпізнавання мови та машинне бачення[1].

Штучний інтелект можна класифікувати як слабкий або сильний. Слабкий штучний інтелект, також відомий як вузький, - це система, яка розроблена і підготовлена для певного завдання. Віртуальні особисті помічники, такі як Apple Siri, є формою слабкого штучного інтелекту. Сильний штучний інтелект, також відомий як штучний загальний інтелект, - це система з узагальненими когнітивними здібностями людини. Коли постає незнайоме завдання, сильна система штучного інтелекту здатна знайти рішення без втручання людини.

Оскільки витрати на апаратне забезпечення, програмне забезпечення та персонал для штучного інтелекту можуть бути дорогими, багато виробників включають компоненти штучного інтелекту у свої стандартні пропозиції, а також доступ до платформ штучного інтелекту як сервісу. Штучний інтелект як послуга дозволяє людям та компаніям експериментувати з ним для різних бізнес-цілей та вибирати кілька платформ, перш ніж взяти на себе зобов’язання. Популярні хмарні пропозиції штучного інтелекту включають

послуги Amazon AI, помічник IBM Watson, Microsoft Cognitive Services та служби Google AI Google.

Хоча інструменти штучного інтелекту представляють цілий ряд нових функціональних можливостей для бізнесу, використання штучного інтелекту викликає етичні питання. Це пояснюється тим, що алгоритми глибокого навчання, які лежать в основі багатьох найсучасніших інструментів штучного інтелекту, є настільки ж розумними, як і дані, які вони отримують під час навчання. Оскільки людина вибирає, які дані слід використовувати для підготовки програми штучного інтелекту, потенціал людської упередженості притаманний і повинен ретельно контролюватися.

Деякі експерти галузі вважають, що термін штучний інтелект занадто тісно пов'язаний із популярною культурою, що спричиняє широку громадськість нереалістичних побоювань щодо штучного інтелекту та неймовірних очікувань щодо того, як він змінить робоче місце та життя загалом. Дослідники та маркетологи сподіваються, що розширений інтелект етикету, який має більш нейтральну конотацію, допоможе людям зрозуміти, що штучний інтелект просто покращить продукти та послуги, а не замінить людей, які ними користуються.

Коли більшість людей чує термін штучний інтелект, перше, про що вони зазвичай думають - це роботи. Це тому, що великобюджетні фільми та романи плетуть розповіді про людиноподібні машини, які завдають хаосу на Землі. Але нічого не могло бути далі від істини.

Штучний інтелект заснований на принципі, що людський інтелект можна визначити таким чином, що машина може легко імітувати його та виконувати завдання, від найпростіших до тих, які ще складніші. Цілі штучного інтелекту включають навчання, міркування та сприйняття. По мірі прогресу технологій попередні орієнтири, які визначали штучний інтелект, застаріли.

Наприклад, машини, які обчислюють основні функції або розпізнають текст за допомогою оптимального розпізнавання символів, вже не

вважаються втіленими штучним інтелектом, оскільки ця функція зараз сприймається як належна як функція, що притаманна комп'ютеру. Штучний інтелект постійно розвивається, щоб принести користь багатьом різним галузям. Машини підключаються за допомогою міждисциплінарного підходу, який базується на математиці, інформатиці, лінгвістиці, психології тощо.

1.1.1.2 Машинне навчання

Машинне навчання - категорія алгоритму, яка дозволяє програмним застосункам стати більш точними при прогнозуванні результатів, не будучи явно запрограмованими. Основною передумовою машинного навчання є побудова алгоритмів, які можуть приймати вхідні дані та використовувати статистичний аналіз для прогнозування виходу, оновлюючи результати, коли нові дані стануть доступними[2].

Процеси, що беруть участь у машинному навчанні, аналогічні процесам видобутку даних та прогнозування моделювання. Обидва вимагають пошуку даних, щоб шукати шаблони та відповідно коригувати дії програми. Багато людей знайомі з машинним навчанням за допомогою покупок в Інтернеті та подають рекламу, пов’язану з їх покупкою. Це відбувається тому, що механізми рекомендацій використовують машинне навчання, щоб персоналізувати доставку реклами в режимі реального часу.

Крім персоналізованого маркетингу, інші поширені випадки використання машинного навчання включають виявлення шахрайства, фільтрацію спаму, виявлення загрози мережевій безпеці, прогнозне обслуговування та створення новин. Алгоритми машинного навчання часто класифікуються як наглядові чи непідконтрольні. Контрольовані алгоритми вимагають від науковця даних або аналітика даних з навичками машинного навчання, щоб забезпечити як вхід, так і бажаний результат, на додаток до надання відгуків про точність прогнозів під час навчання алгоритму.

Дані вчені визначають, які змінні чи особливості моделі слід аналізувати та використовувати для розробки прогнозів. Після того як навчання закінчиться, алгоритм застосує те, що було вивчено, до нових даних. Непідтримувані алгоритми не потребують навчання з бажаними даними про результати. Натомість вони використовують ітеративний підхід, який називається глибоким навчанням, щоб переглянути дані та дійти висновків.

Непідконтрольні алгоритми навчання - їх також називають нейронними мережами - використовуються для складніших завдань обробки, ніж керовані системи навчання, включаючи розпізнавання зображень, мовлення в текст та створення природних мов. Ці нейронні мережі працюють, комбінуючи мільйони прикладів навчальних даних та автоматично ідентифікуючи часто тонкі кореляції між багатьма змінними. Після навчання, алгоритм може використовувати свій банк асоціацій для інтерпретації нових даних. Ці алгоритми стали здійсненними лише в епоху великих даних, оскільки для них потрібні великі обсяги навчальних даних[2].

Машинне навчання сьогодні використовується в широкому діапазоні застосувань. Один з найвідоміших прикладів - News Feed Facebook. News Feed використовує машинне навчання, щоб персоналізувати канал кожного члена. Якщо учасник часто перестає прокручувати читання або сподобатися публікаціям певного друга, News News почне показувати більше активності цього друга раніше у стрічці.

Програмне забезпечення просто використовує статистичний аналіз та прогностичну аналітику для виявлення шаблонів даних користувача та використання цих моделей для заповнення News News. Якщо член більше не зупиняється, щоб читати, сподобатися чи коментувати публікації друга, нові дані будуть включені до набору даних, а News News буде відповідно коригуватися.

Машинне навчання також входить до масиву корпоративних додатків. Системи управління взаємовідносинами з клієнтами (CRM) використовують

навчальні моделі для аналізу електронної пошти та оперативних членів торгової групи, щоб спочатку відповісти на найважливіші повідомлення. Більш просунуті системи можуть навіть рекомендувати потенційно ефективні відповіді.

Постачальники бізнес-аналітики та аналітики використовують машинне навчання у своєму програмному забезпеченні, щоб допомогти користувачам автоматично визначати потенційно важливі точки даних. Системи людських ресурсів (HR) використовують моделі навчання для визначення характеристик ефективних працівників і спираються на ці знання, щоб знайти найкращих претендентів на відкриті посади.

Машинне навчання також відіграє важливу роль у самостійному керуванні автомобілями. Нейронні мережі глибокого навчання використовуються для ідентифікації об'єктів та визначення оптимальних дій для безпечного керування транспортним засобом вниз по дорозі.

1.1.1.3 Розпізнавання тексту

Розпізнавання тексту використовується в рукописному або надрукованому тексті в машино-закодованому тексті - від сканованого документа, фотографії документа, фотографії сцени або від тексту підзаголовків, накладених на зображення[3].

Оптичне розпізнавання символів - це розпізнавання друкованих чи записаних текстових символів комп'ютером. Це передбачає сканування фотографії тексту за символом, аналіз відсканованого зображення, а потім переклад зображення символу кодами символів, наприклад як ASCII, що зазвичай використовується в процесі обробки даних[4].

Етапи, які повинні бути пройдені перед автоматичним розпізнавання тексту:

- Зняття перекосу – це коли документ не вирівняний належним чином при скануванні, його може знадобитися назвати на кілька градусів за годинниковою стрілкою або проти годинникової стрілки.
- Очистка - видаліть позитивні та негативні плями, розгладжуючи краї.
- Бінаризація - перетворення зображення з кольорового або відтінку сірого в чорно-біле (його називають «двійковим зображенням», оскільки є два кольори). Завдання бінаризації виконується як простий спосіб відокремлення тексту (або будь-якого іншого потрібного компонента зображення) від фону.
- Видалення лінії - очищає неглифні коробки та лінії.
- Аналіз макета або “зонування” - визначає стовпці, абзаци, підписи тощо як окремі блоки.
- Виявлення рядків і слів - встановлює базову лінію для слів і форм символів, при необхідності розділяє слова.
- Ізоляція символів або "сегментація" - для оптичного розпізнавання символів кілька символів, які з'єднані через артефакти зображень, повинні бути розділені; поодинокі символи, розбиті на кілька частин через артефакти, повинні бути з'єднані.
- Номалізація співвідношення сторін і масштаб.

Існує два основних типи алгоритму основного оптичного розпізнавання символів, який може створити список ранжируваних символів.

Матричне узгодження включає порівняння зображення зі збереженим гліфом на основі пікселя на піксель; він також відомий як "відповідність шаблону", "розпізнавання візерунків" або "кореляція зображення". Це залежить від того, щоб вхідний гліф був правильно ізольований від решти зображень, а збережений гліф був схожим шрифтом і в тому ж масштабі. Цей прийом найкраще працює з машинописним текстом і не працює добре, коли

виникають нові шрифти. Це методика раннього фізичного використання фотоелементів на основі ОЦР, а не безпосередньо.

Екстракція функції розбиває гліфи на "особливості", такі як лінії, замкнуті петлі, напрям лінії та перетини ліній. Особливості вилучення зменшують розмірність подання та роблять процес розпізнавання обчислювально ефективним. Ці функції порівнюються з абстрактним векторним зображенням символу, яке може зводитися до одного або декількох прототипів гліфів. Загальні методи виявлення функцій у комп’ютерному зорі застосовні до цього типу відеомагнітофонів, що зазвичай спостерігається у «інтелектуальному» розпізнаванні рукописного тексту та справді найсучасніших програм оптичного розпізнавання символів. Найближчі класифікатори сусідів, такі як алгоритм k-найближчих сусідів, використовуються для порівняння особливостей зображення зі збереженими функціями гліфів та виберіть найближчу відповідність.

Точність OCR може бути збільшена, якщо вихід обмежений лексикою - переліком слів, дозволених у документі. Точність оптичного розпізнавання символів може бути збільшена, якщо вихід обмежений лексикою - переліком слів, дозволених у документі. Це можуть бути, наприклад, усі слова англійською мовою або більш технічний лексикон для певного поля. Цей прийом може бути проблематичним, якщо документ містить слова, які не містяться в лексиконі, Tesseract використовує свій словник для впливу на крок сегментації символів для покращення точності. Знання граматики сканованої мови також може допомогти визначити, чи є певне слово наприклад, дієслово або іменник, що дозволяє підвищити точність. Алгоритм відстані Левенштейна також використовувався при обробці оптичного розпізнавання символів для подальшої оптимізації результатів.

1.1.1.4 Розпізнавання предметів

Розпізнавання предметів - це загальний термін для опису сукупності пов'язаних із цим завдань комп’ютерного зору, які передбачають ідентифікацію об'єктів на цифрових фотографіях чи відеозаписах[5].

Класифікація зображень передбачає прогнозування класу одного об’єкта в зображенні. Локалізація об'єкта означає визначення місця розташування одного або декількох об'єктів у зображенні та нанесення рясного ряду навколо їх розміру. Виявлення об'єктів поєднує ці дві задачі та локалізує та класифікує один або кілька об’єктів у зображенні.

Таким чином, ми можемо розрізняти ці три завдання з комп’ютерного зору.

- Класифікація зображень - дає можливіть передбачити тип або клас об'єкта на зображенні. На вхід дається зображення з одним об’єктом, наприклад фотографії. На виході ми отримуємо мітку класу (наприклад, одне або більше цілих чисел, які відображаються на мітках класів).
- Локалізація об'єктів – дає можливіть знайти присутність об’єктів на зображенні та вказати їх розташування за допомогою обмежувального поля. На вхід дається зображення з одним або декількома об'єктами, наприклад фотографією. На виході ми отримуємо одну або кілька обмежувальних коробок (наприклад, визначених точкою, шириною та висотою).
- Виявлення об'єктів – дає можливість знайти присутність на зображенні об'єктів із обмежувальним вікном та типами або класами розташованих об'єктів. На вхід дається зображення з одним або декількома об'єктами, наприклад фотографією. На виході ми отримуємо одне або кілька обмежувальних полів (наприклад, визначених точкою, шириною та висотою) та мітка класу для кожного обмежувального поля.

Ще одним розширенням цієї розбивки завдань комп'ютерного зору є сегментація об'єктів, яка також називається "сегментація екземпляра об'єкта"

або "семантична сегментація", де екземпляри розпізнаних об'єктів позначаються виділенням конкретних пікселів об'єкта замість грубої рамки обмеження.

1.1.2 Постановка задачі

Завданням магістерської роботи є розробка програмного забезпечення для визначення предметів та розпізнавання тексту з використанням машинного навчання та підтримкою технології VoiceOver. Результатом розробки повинен бути мобільний застосунок для платформи iOS.

Інтерфейс застосунку повинен бути нескладним та інтуїтивно зрозумілим для будь-якого користувача. Основні функції додатку, а саме – розпізнавання тексту та визначення предметів повинні бути чітко виокремлені, так як вони являються ключовими у системі. Також візуальна частина застосунку та її компоненти повинні бути виконані згідно сучасних вимог та стандартів. Навігація по додатку повинна бути продуманою та вивіреною. Застосунок повинен працювати плавно та без жодних перебоїв. Усе повинно бути виконано з врахуванням дизайну досвіду користувача.

Після аналізу предметної області було визначено основний функціонал системи, що повинен бути реалізований в процесі розробки програмної системи. Отже, ключовими модулями застосунку є: модуль розпізнавання тексту, модуль розпізнавання предметів, модуль збережених текстів та модуль збережених предметів.

Розпізнавання повинно проводитися з використанням машинного навчання. Інструмент, який буде використовуватися для роботи з машинним навчання, буде визначено на одному з наступних етапів проектування програмної системи.

Під підтримкою технології VoiceOver розуміється, що після успішного розпізнавання тексту чи предмету, отриманий результат повинен бути озвучений для користувача.

Порядок задач та кроків, які необхідно вирішити:

- Визначення акторів та варіантів використання системи
- Обрати модель розробки застосунку
- Обрати архітектурний шаблон додатку
- Побудувати діаграму класів
- Обрати середовище розробки та мову програмування
- Обрати бібліотеку для роботи з машинним навчання
- Обрати спосіб збереження даних
- Реалізувати систему
- Повести тестування системи

1.1.3 Пошук акторів та варіантів використання

Ознайомившись з поставленою задачею та вимогами системи було визначено, що в даній системі буде лише один актор – Користувач. Оскільки технічним завданням до системи не передбачено користувачів системи з різним рівнем доступу до системи та різним функціоналом.

Користувач – повинен мати змогу проводити розпізнавання предметів, розпізнавання тексту, перегляд, додавання та видалення предметів та тексту, що були успішно виявлені.

Після визначення основного актора системи можна приступити до визначення основних варіантів використання системи. Результат виявлення варіантів використання представлено в таблиці 1.1.

Таблиця 1.1 Виявлення варіантів використання

Актор	Назва	Опис

Користувач	Розпізнати предмет з використанням фото з галереї	Користувач повинен мати змогу вибрати картинку чи зображення з необхідним предметом з галереї свого мобільного телефону для того, щоб система провела розпізнавання предмету на основі обраного зображення

Продовження таблиці 1.1

Користувач	Розпізнати предмет з використанням фото зробленого камерою	Користувач виконує фото предмету, використовуючи, камеру свого мобільного телефону, щоб система провела розпізнавання предмету на основі зробленого зображення
Користувач	Розпізнати текст з використанням фото з галереї	Користувач повинен мати змогу вибрати картинку чи зображення із текстом з галереї свого мобільного телефону для того, щоб система провела розпізнавання предмету на основі обраного зображення
Користувач	Розпізнати текст з використанням фото зробленого камерою	Користувач виконує фото тексту, використовуючи, камеру свого мобільного телефону, щоб система провела розпізнавання тексту на основі зробленого зображення
Користувач	Прослухати результат розпізнавання предмету	Користувач, після того як система виконала успішне розпізнавання предмету, повинен мати можливість натиснувши на екран почути результат розпізнавання предмету
Користувач	Прослухати результат розпізнавання тексту	Користувач, після того як система виконала успішне розпізнавання тексту, повинен мати можливість натиснувши на екран почути результат

		розпізнавання тексту
Користувач	Зберегти результат розпізнавання предмету	Користувач, після того як система виконала успішне розпізнавання предмету, повинен мати можливість зберегти результат
Користувач	Зберегти результат розпізнавання тексту	Користувач, після того як система виконала успішне розпізнавання тексту, повинен мати можливість зберегти результат розпізнавання в базу даних

Продовження таблиці 1.1

Користувач	Переглянути збережені результати розпізнавання предметів	Користувач повинен мати змогу здійснити перегляд усіх збережених результатів успішного розпізнавання предметів
Користувач	Переглянути збережені результати розпізнавання тексту	Користувач повинен мати змогу здійснити перегляд усіх збережених результатів успішного розпізнавання тексту
Користувач	Переглянути конкретний результат розпізнавання предметів	Користувач повинен мати змогу здійснити перегляд конкретного збереженого результату успішного розпізнавання предметів
Користувач	Переглянути конкретний результат розпізнавання тексту	Користувач повинен мати змогу здійснити перегляд конкретного збереженого результату успішного розпізнавання тексту
Користувач	Редагувати конкретний результат розпізнавання предметів	Користувач повинен мати змогу здійснити редагування конкретного збереженого результату успішного розпізнавання предметів

Користувач	Редагувати конкретний результат розпізнавання тексту	Користувач повинен мати змогу здійснити редагування конкретного збереженого результату успішного розпізнавання тексту
Користувач	Видалити конкретний результат розпізнавання предметів	Користувач повинен мати змогу здійснити видалення конкретного збереженого результату успішного розпізнавання предметів
Користувач	Видалити конкретний результат розпізнавання тексту	Користувач повинен мати змогу видалити перегляд конкретного збереженого результату успішного розпізнавання тексту виконаного програмною системою

Продовження таблиці 1.1

Користувач	Сортувати збережені результати розпізнавання предметів	Користувач повинен мати змогу сортувати збережені результати успішного розпізнавання предметів за датою їх збереження у базі даних
Користувач	Сортувати збережені результати розпізнавання тексту	Користувач повинен мати змогу сортувати збережені результати успішного розпізнавання тексту за датою їх збереження у базі даних

Після успішного виявлення варіантів використання системи було побудовано діаграму варіантів використання системи (рисунок 1.1). На даній діаграмі відображено основні варіанти використанні описані вище.

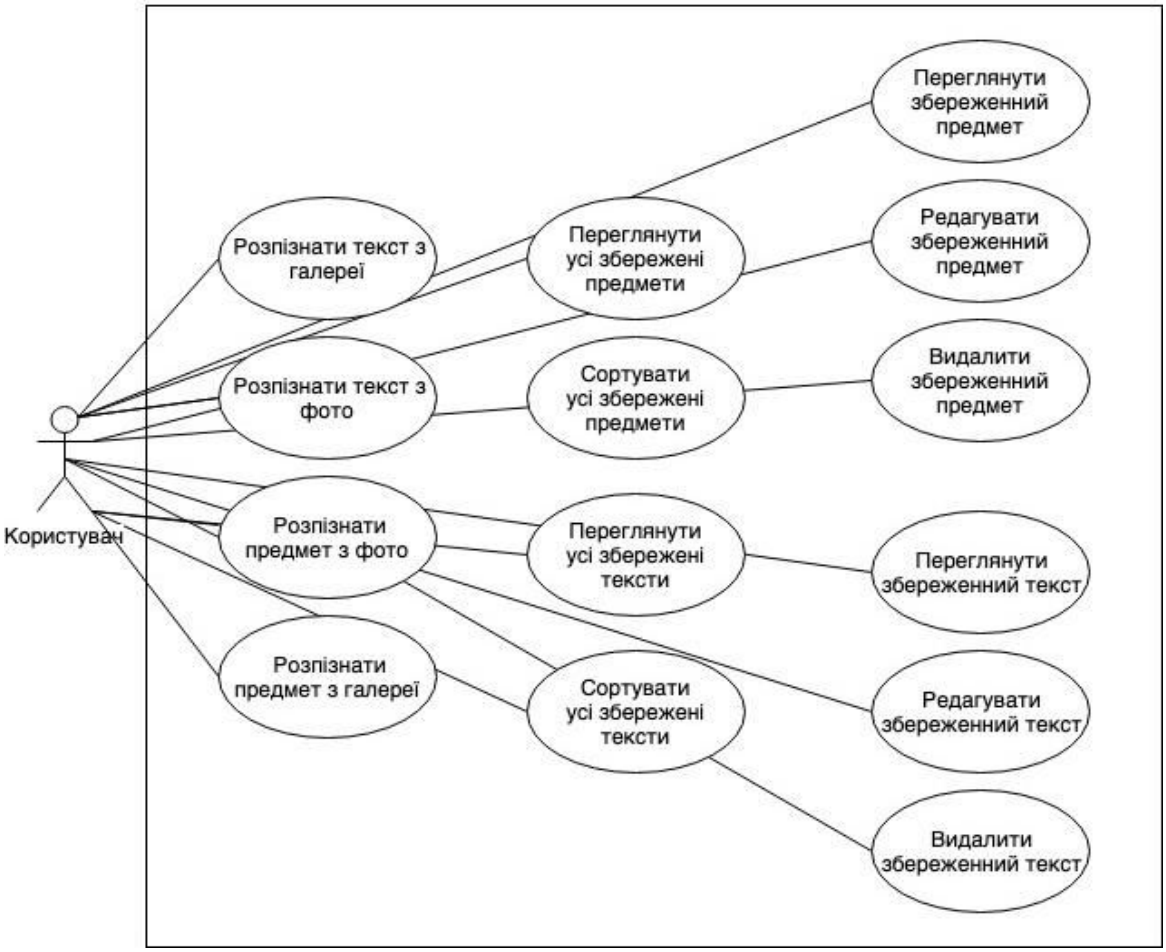

Рисунок 1.1 – Діаграма варіантів використання системи

1.1.4 Опис ключових варіантів використання

Після визначення усіх варіантів використання програмної системи було визначено ключові варіанти використання, а саме: розпізнати текст з фото, розпізнати предмет з фото, озвучити розпізнаний текст, озвучити результат розпізнавання предмету, переглянути усі збережені предмети, переглянути усі збережені тексти.

Варіант використання «Розпізнати текст з фото» описаний у таблиці 1.2

Таблиця 1.2 Опис варіанту використання «Розпізнати текст з фото»

Дійові особи	Користувач, Система
Цілі	Розпізнати текст, який знаходиться на фото зробленому

	користувачем
Передумова	Пристрій користувача повинен мати камеру. Користувач повинен дозволити застосунку використовувати камеру його телефона.
Успішний сценарій: 1. Користувач відкриває застосунок. 2. Користувач натискає клавішу «Розпізнати текст». 3. Система видає користувачу опцію «Камера». 4. Користувач обирає дану опцію. 5. Система просить дозвіл у користувача на використання камери його телефону. 6. Користувач підтверджує, що він дає дозвіл на користування камерою. 7. Система відкриває екран, який відображає, те що показує камера користувача. Користувач робить фото. 8. Система перевіряє валідність фото. 9. Система обробляє фото та визначає текст, що на ньому зображений. 10. Система відображає результат на екрані та озвучує його користувачу.	
Результат	Система успішно розпізнала текст з фото яке надав користувач
Альтернативні сценарії	
5а	Пристрій користувача не містить камери. Результат: Система повідомляє користувача про неможливість виконання даної операції.

Продовження таблиці 1.2

6а	Користувач не підтвердив дозвіл на використання програмною системою камери його телефону. Результат: система не має можливості використовувати камеру. Наступні кроки сценарію не буде виконано
8а	Система виявила, що фото виконане користувачем – невірне. Результат: система повідомляє користувача про це. Система надає можливість повторити операцію виконання фото(пункт 7) ще раз.

10а	Результат розпізнавання був проведений не успішно. Результат: система повідомляє користувача про це. Система надає можливість повторити операцію виконання фото(пункт 7) ще раз.

Варіант використання «Розпізнати предмет з фото» описаний у таблиці 1.3

Таблиця 1.3 Опис варіанту використання «Розпізнати предмет з фото»

Дійові особи	Користувач, Система
Цілі	Розпізнати предмет, який знаходиться на фото, зробленому користувачем
Передумова	Пристрій користувача повинен мати камеру. Користувач повинен дозволити застосунку використовувати камеру його телефона.
Успішний сценарій: 1. Користувач відкриває застосунок. 2. Користувач натискає клавішу «Розпізнати предмет». 3. Система видає користувачу опцію «Камера». 4. Користувач обирає дану опцію. 5. Система просить дозвіл у користувача на використання камери його телефону. 6. Користувач підтверджує, що він дає дозвіл на користування камерою. 7. Система відкриває екран, який відображає, те що показує камера користувача. Користувач робить фото. 8. Система перевіряє валідність фото. 9. Система обробляє фото та визначає предмет, що на ньому зображений. 10. Система відображає результат на екрані та озвучує його користувачу.	

Продовження таблиці 1.3

Результат	Система успішно розпізнала предмет з фото, яке надав користувач
Альтернативні сценарії	
5а	Пристрій користувача не містить камери. Результат: Система повідомляє користувача про неможливість виконання даної операції.
6а	Користувач не підтвердив дозвіл на використання програмною системою камери його телефону. Результат: система не має можливості використовувати камеру. Наступні кроки сценарію не буде виконано
8а	Система виявила, що фото виконане користувачем –

	невірне. Результат: система повідомляє користувача про це. Система надає можливість повторити операцію виконання фото(пункт 7) ще раз.
10а	Результат розпізнавання був проведений не успішно. Результат: система повідомляє користувача про це. Система надає можливість повторити операцію виконання фото(пункт 7) ще раз.

Варіант використання «Озвучити розпізнаний текст» описаний у таблиці 1.4.

Таблиця 1.4 Опис варіанту використання «Озвучити розпізнаний текст»

Дійові особи	Користувач, Система
Цілі	Озвучити текст, який був успішно розпізнаний системою
Передумова	Користувач повинен увімкнути у налаштуваннях «Доступності» опцію «VoiceOver». Користувач повинен надати доступ до мікрофону
Успішний сценарій: 1. Користувач відкриває застосунок. 2. Користувач використовує функцію «Розпізнати текст». 3. Система видає користувачу результат розпізнавання тексту на екрані. 4. Користувач натискає на екран. 5. Система просить дозвіл у користувача на використання мікрофону його телефону. 6. Користувач підтверджує, що він дає дозвіл на користування мікрофоном. 7. Система перевіряє та озвучує текст, який був розпізнаний системою.	

Продовження таблиці 1.4

Результат	Система успішно озвучила розпізнаний текст для користувача
Альтернативні сценарії	
***а**	Користувач не увімкнув опцію «VoiceOver» на своєму пристрої. Результат: система не має можливості озвучувати розпізнаний текст для користувача.
2а	Система не змогла успішно розпізнати текст. Результат: система повертає користувача на попередній екран.
5а	Користувач не підтвердив дозвіл на використання

	програмною системою мікрофону його телефону. Результат: система не має можливості використовувати мікрофон. Наступні кроки сценарію не буде виконано
7а	Система виявила, що розпізнаний текст невірний для його озвучування. Результат: система повідомляє про це користувача та надає можливість повторити операцію(пункт 2), ще раз.

Варіант використання «Озвучити розпізнаний предмет» описаний у таблиці 1.5.

Таблиця 1.5 Опис варіанту використання «Озвучити розпізнаний предмет»

Дійові особи	Користувач, Система
Цілі	Озвучити текст, який був успішно розпізнаний системою
Передумова	Користувач повинен увімкнути у налаштуваннях «Доступності» опцію «VoiceOver». Користувач повинен надати доступ до мікрофону
Успішний сценарій: 1. Користувач відкриває застосунок. 2. Користувач використовує функцію «Розпізнати предмет». 3. Система видає користувачу результат розпізнавання предмету на екрані. 4. Користувач натискає на екран. 5. Система просить дозвіл у користувача на використання мікрофону його телефону. 6. Користувач підтверджує, що він дає дозвіл на користування мікрофоном.	

Продовження таблиці 1.5

7. Система перевіряє та озвучує інформацію про предмет, який був розпізнаний системою.	
Результат	Система успішно озвучила інформацію про розпізнаний предмет для користувача
Альтернативні сценарії	
***а**	Користувач не увімкнув опцію «VoiceOver» на своєму пристрої. Результат: система не має можливості озвучувати інформацію про розпізнаний предмет для користувача.
2а	Система не змогла успішно розпізнати предмет.

	Результат: система повертає користувача на попередній екран.
5а	Користувач не підтвердив дозвіл на використання програмною системою мікрофону його телефону. Результат: система не має можливості використовувати мікрофон. Наступні кроки сценарію не буде виконано
7а	Система виявила, що інформація про розпізнаний предмет невірна для його озвучування. Результат: система повідомляє про це користувача та надає можливість повторити операцію(пункт 2), ще раз.

Варіант використання «Переглянути усі збережені предмети» описаний у таблиці 1.6.

Таблиця 1.6 Опис варіанту використання «Переглянути усі збережені предмети»

Дійові особи	Користувач, Система
Цілі	Переглянути усі збережені користувачем предмети, що раніше було розпізнані системою та збережені ним до бази даних
Передумова	У базі даних повинні бути наявні збережені предмети
Успішний сценарій: 1. Користувач відкриває застосунок. 2. Користувач натискає клавішу «Переглянути збережені предмети». 3. Система відкриває новий екран «Збережені предмети». 4. Система перевіряє доступ до бази даних. 5. Система показує індикатор завантаження та дістає усі наявні в базі даних збережені предмети.	

Продовження таблиці 1.6

6. Система перевіряє валідність стягнутих даних. 7. Система відображає дані у списку. 8. Користувач переглядає отримані дані.	
Результат	Система успішно продемонструвала користувачу список збережених даних.
Альтернативні сценарії	
4а	Система не отримала доступ до бази даних. Результат: система показує відповідне повідомлення та закриває екран.
5а	Система не вдалось завантажити дані з бази даних.

	Результат: система показує відповідне повідомлення та дає можливість користувачеві спробу повторити завантаження.
6а	Система виявила, що стягнуті дані невірні. Результат: система показує відповідне повідомлення та дає можливість користувачеві спробу повторити завантаження.
7а	Системі не вдалося правильно відобразити дані у списку. Результат: система показує відповідне повідомлення та дає можливість користувачеві спробу повторити завантаження.

Варіант використання «Переглянути усі збережені тексти» описаний у таблиці 1.7.

Таблиця 1.7 Опис варіанту використання «Переглянути усі збережені тексти»

Дійові особи	Користувач, Система
Цілі	Переглянути усі збережені користувачем тексти, що раніше було розпізнані системою та збережені ним до бази даних
Передумова	У базі даних повинні бути наявні збережені тексти
Успішний сценарій: 1. Користувач відкриває застосунок. 2. Користувач натискає клавішу «Переглянути збережені тексти». 3. Система відкриває новий екран «Збережені тексти». 4. Система перевіряє доступ до бази даних.	

Продовження таблиці 1.7

5. Система показує індикатор завантаження та дістає усі наявні в базі даних збережені тексти. 6. Система перевіряє валідність стягнутих даних. 7. Система відображає дані у списку. 8. Користувач переглядає отримані дані.	
Результат	Система успішно продемонструвала користувачу список збережених даних.
Альтернативні сценарії	
4а	Система не отримала доступ до бази даних. Результат: система показує відповідне повідомлення та

	закриває екран.
5а	Система не вдалось завантажити дані з бази даних. Результат: система показує відповідне повідомлення та дає можливість користувачеві спробу повторити завантаження.
6а	Система виявила, що стягнуті дані невірні. Результат: система показує відповідне повідомлення та дає можливість користувачеві спробу повторити завантаження.
7а	Системі не вдалося правильно відобразити дані у списку. Результат: система показує відповідне повідомлення та дає можливість користувачеві спробу повторити завантаження.

1.2 Проектування програмної системи

1.2.1 Вибір моделі розробки

Для розробки даної програмної системи вибір буде із двох моделей розробки: ітеративної та каскадної. Проведемо розгляд кожної із них.

Ітеративна модель розробки - це спосіб розбити розробку програмного забезпечення великого додатка на менші шматки. В ітераційній розробці функціональний код обдумується, розробляється і перевіряється на повторних циклах. З кожною ітерацією можна обдумати, розробити та протестувати додаткові функції до тих пір, поки не буде повністю функціональне програмне забезпечення, готове до розгортання серед клієнтів[6].

Зазвичай ітеративна розробка використовується в поєднанні з поступовою розробкою, в якій довший цикл розробки програмного забезпечення розбивається на більш дрібні сегменти, які будуються один на одному.

Ітеративна модель є ключовою практикою методологій розвитку Agile. У методах Agile коротший цикл розвитку, який називають ітерацією чи спринтом, є віковим (обмежений певним збільшенням часу, наприклад, два

тижні). В кінці ітерації очікується робочий код, який можна продемонструвати для замовника. Метою ітеративної роботи є надання більшої гнучкості для змін.

Переваги ітеративної моделі:

- Створює робоче програмне забезпечення швидко та рано протягом життєвого циклу програмного забезпечення.
- Більш гнучка - дешевше змінювати сферу застосування та вимоги.
- Простіше тестувати та налагоджувати під час меншої ітерації.
- Простіше керувати ризиком, оскільки під час його ітерації виявляються та обробляються ризикові частини.
- Кожна ітерація є важливою віхою.

Недоліки ітеративної моделі:

- Кожна фаза ітерації є жорсткою і не перетинаються.
- Можуть виникнути проблеми, пов’язані з архітектурою системи, оскільки не всі вимоги зібрані наперед протягом усього життєвого циклу програмного забезпечення.

На рисунку 1.2 схематично відображено життєвий цикл розробки програмного забезпечення з використанням ітеративної моделі розробки.

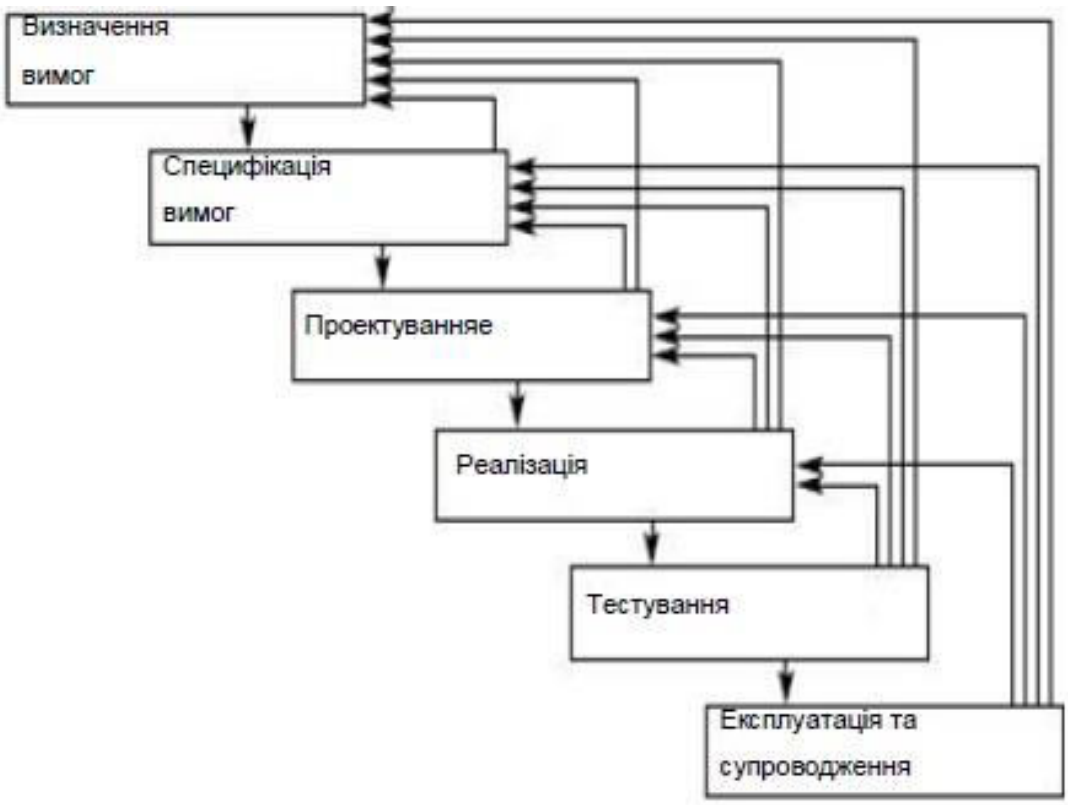

Рисунок 1.2 – Діаграма ітеративної моделі розробки

Каскадна модель - це лінійний послідовний підхід до життєвого циклу розробки програмного забезпечення, який популярний у розробці програмного забезпечення та розробці продуктів. Каскадна модель підкреслює логічне просування кроків. Подібно до напрямку, що вода протікає через край скелі, окремі кінцеві точки або цілі встановлюються для кожної фази розвитку і після завершення їх не можна переглянути[7].

Каскадна модель складається з семи етапів, що не перетинаються:

Визначення вимог: потенційні вимоги, терміни та вказівки щодо проекту аналізуються та розміщуються у функціональній специфікації. Цей етап обробляє визначення та планування проекту без згадування конкретних процесів.

Аналіз: специфікації системи аналізуються для створення моделей продукції та бізнес-логіки, які керуватимуть виробництвом. Це також, коли фінансові та технічні ресурси перевіряються на предмет доцільності.

Проектування: документ із специфікацією дизайну створений для викладення технічних вимог проектування, таких як мова програмування, апаратне забезпечення, джерела даних, архітектура та послуги.

Кодування / реалізація: вихідний код розробляється з використанням моделей, логіки та вимог, визначених на попередніх етапах. Зазвичай система розробляється в менших компонентах або одиницях, перш ніж впроваджуватися разом.

Тестування: перевірка якості, модуль, система та бета-тести проходять для повідомлення про проблеми, які, можливо, потребують вирішення. Це може спричинити вимушене повторення етапу кодування для налагодження.

Експлуатація / розгортання: продукт або додаток вважається повністю функціональним і розгорнуто в реальному середовищі. Технічне обслуговування:

Коригування: адаптивне та досконале обслуговування проводиться необмежено для вдосконалення, оновлення та покращення кінцевого продукту. Це може включати випуск оновлень та нових версій.

Переваги каскадної моделі:

- Попередні етапи документації та планування дозволяють великим групам або групам, що переходять на зміну, залишатися в курсі та рухатися до спільної мети.
- Легка для розуміння, супроводу та розділення завдань.
- Сприяє департаменталізації та управлінському контролю на основі розкладу або строків.
- Дозволяє легко здійснити ранні зміни в дизайні чи технічних характеристиках.
- Чітко визначає віхи та терміни.

Недоліки каскадної моделі:

- Проектування не адаптивне - часто, коли виявляється недолік, весь процес потрібно починати заново.

- Ігнорує можливість отримувати відгуки користувачів або клієнтів посередині процесу та вносити зміни на основі отриманих результатів.
- Затримує тестування до кінця життєвого циклу розвитку.
- Не враховує виправлення помилок.
- Знижує ефективність, не дозволяючи процесам перетинатися.
- Не підходить для складних проектів, що мають високий ризик, довго тривають або об'єктно орієнтовані.

На рисунку 1.3 схематично відображено життєвий цикл розробки програмного забезпечення з використанням каскадної моделі розробки.

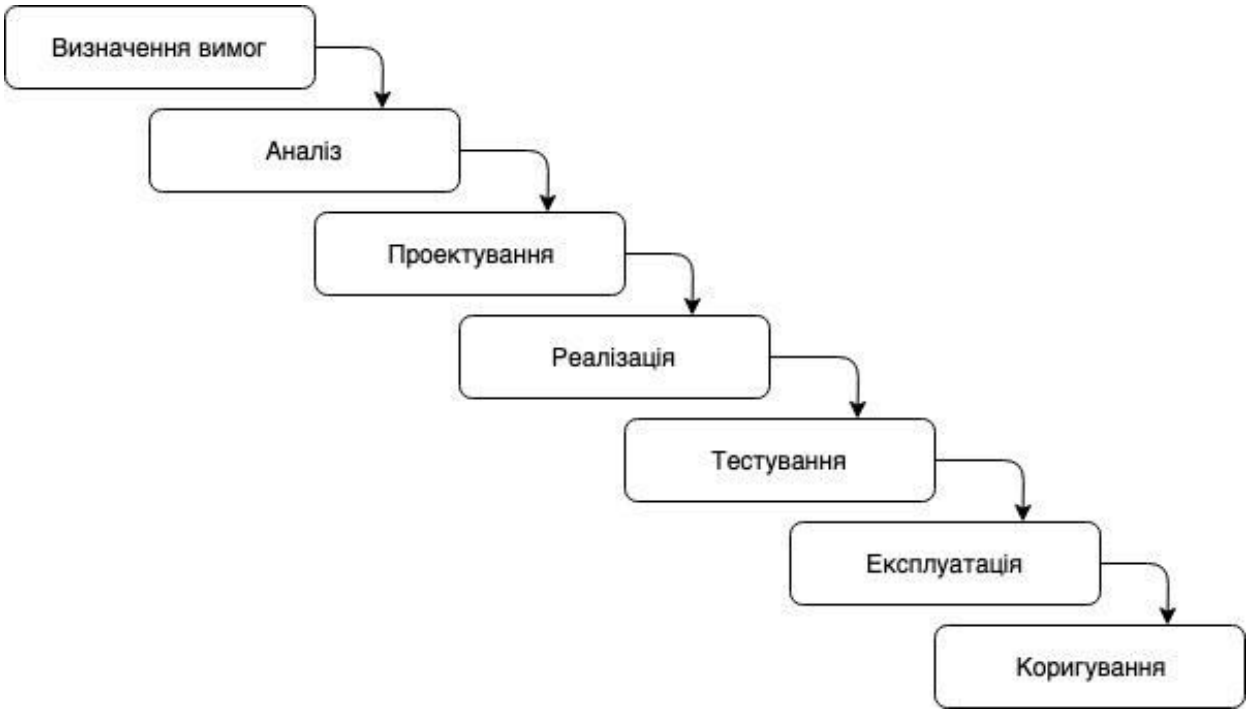

Рисунок 1.3 – Діаграма каскадної моделі розробки

Проаналізувавши усі переваги та недоліки кожної з моделей, а також концепцію та особливість системи, яку потрібно розробити, було обрано каскадну модель розробки. Основною причиною такого вибору стало те, що під час розробки даної системи не будуть вноситись зміни, адже технічне завдання та весь необхідний функціонал чітко окреслений та незмінний. А

також застосунок не є об’ємним та не має великої необхідності у тестуванні та постійній підтримці.

1.2.2 Вибір архітектурного шаблону

Для вибору архітектурного шаблону програмного застосунку було вибрано два найчастіше використовуваних шаблони, для розробки застосунків для платформи iOS, а саме – MVC (Модель-Представлення-Контролер) та MVVM (Модель-Представлення-Модель Представлення). Розглянемо кожен із них.

Модель-Представлення-Контролер – шаблон дизайну призначає об’єктам у програмі одну з трьох ролей: модель, перегляд або контролер. Шаблон визначає не тільки рольові об'єкти, які виконують певну роль у програмі, він визначає спосіб спілкування об'єктів між собою[8].

Кожен з трьох типів об'єктів відокремлений від інших абстрактними межами і спілкується з об'єктами інших типів через ці межі. Колекцію об'єктів певного типу даного шаблону у додатку іноді називають шаром - наприклад, шаром моделі.

Модель-Представлення-Контролер – основний шаблон для хорошого дизайну для застосування Cocoa. Переваги прийняття цього зразка численні. Багато об'єктів у цих додатках мають тенденцію до використання більш багаторазово, а їхні інтерфейси, як правило, краще визначені. Програми, що мають такий дизайн, також легше розширюються, ніж інші програми. Більше того, багато технологій та архітектури Cocoa базуються на даному шаблонові і вимагають, щоб ваші власні об’єкти грали одну з представлених вище ролей ролей.

Модель - відповідає за дані домену або рівень доступу до даних, який маніпулює цими даними.

Представлення - відповідає за рівень презентації, для iOS компонентів все, що починається з префіксу "UI".

Контролер - клей або посередник між Моделлю та Представленням, як правило, відповідальний за зміну Моделі, реагуючи на дії користувача, які виконуються у Представленні, та оновлення Представлення із змінами, що виникають у Моделі.

Схематичне відображення діаграми шаблону Модель-Представлення-Контролер показано на рисунку 1.4.

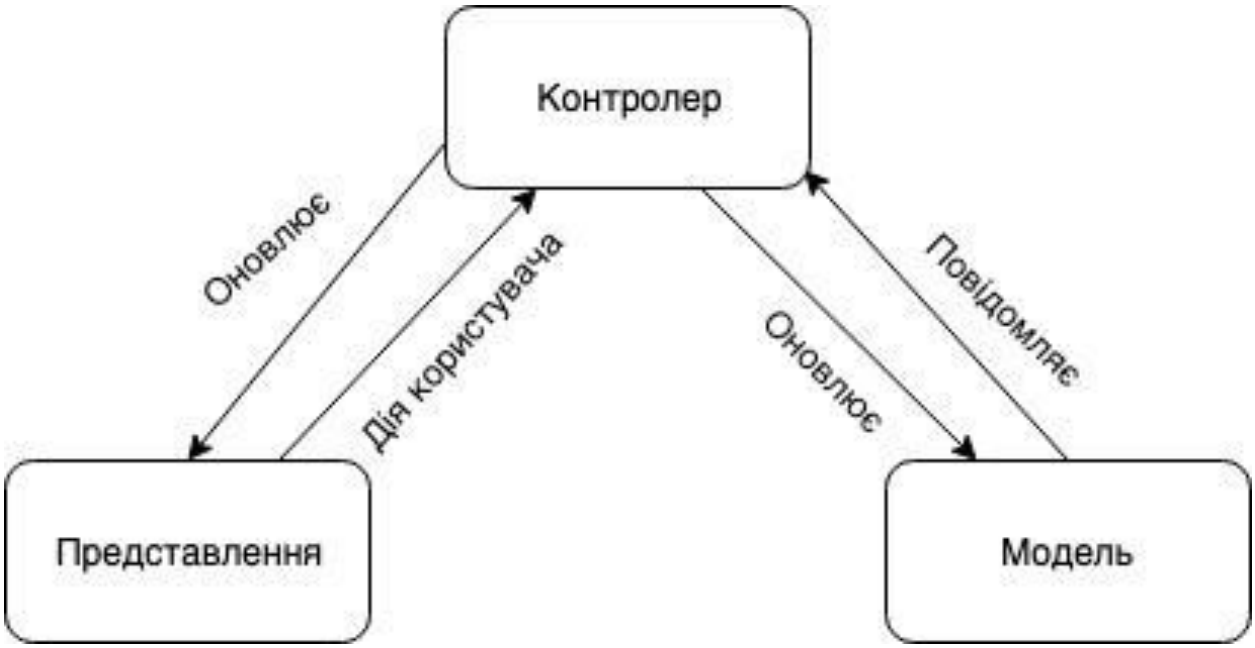

Рисунок 1.4 – Діаграма шаблону «Модель-Представлення-Контролер»

Модель-Представлення-Модель Представлення - модель дизайну, схожа на «Модель-Представлення-Контролер», що реалізованна в iOS, але забезпечує кращу розв'язку інтерфейсу користувача та бізнес логіки. Ця розв'язка призводить до отримання тонких, гнучких та легких для читання класів контролерів перегляду в iOS[9].

Даний шаблон, як зрозуміло із назви, містить додатковий компонент - Модель Представлення. Саме через це він забезпечує кращу інкапсуляцію. Бізнес-логіка та робочі процеси містяться майже виключно у моделі представлення. Розглянемо як змінилися ролі компонентів у даному шаблоні на відміну від попереднього.

Модель - відповідає за дані домену або рівень доступу до даних, який маніпулює цими даними. А, отже, виконує ту саму роль, що і в шаблоні MVC.

Представлення - відповідає за рівень презентації, для iOS компонентів все, що починається з префіксу "UI".

Контролер – відповідає за представлення отриманих з Моделі Представлення даних на самому представленні. У даному шаблоні, Контролер не спілкується з Моделлю на пряму, а через Модель Представлення.

Модель Представлення – інкапсулює у собі логіку роботи з даними. Отримує оновлення від Моделі та оновлює в разі необхідності Представлення чи Контролер. Засіб за допомогою якого Модель Представлення оновлює Представлення чи Контролер називається біндінгом.

Схематичне відображення діаграми шаблону «Модель-Представлення-Модель Представлення» показано на рисунку 1.5.

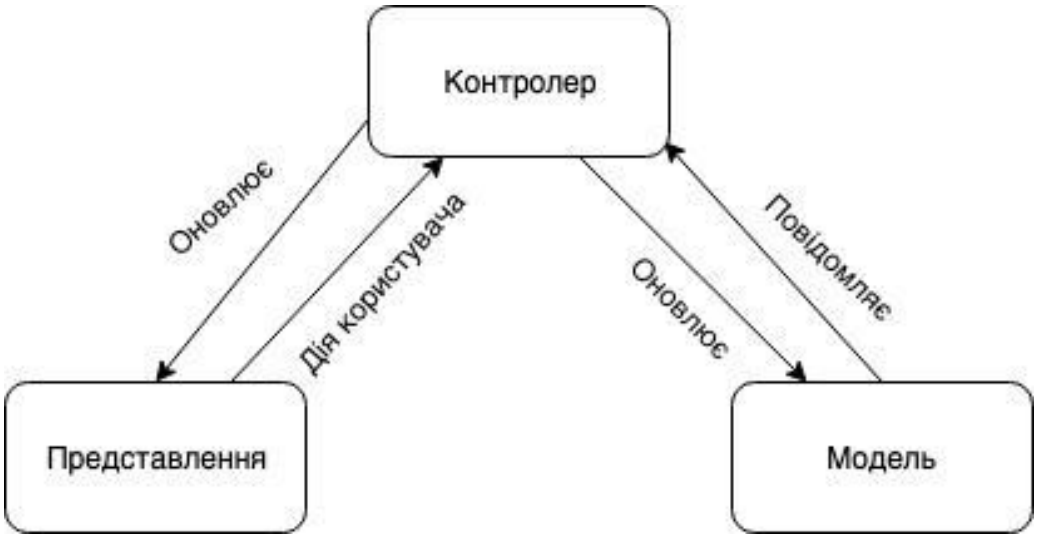

Рисунок 1.5 – Діаграма шаблону «Модель-Представлення-Модель Представлення»

Застосунок, який необхідно реалізувати, не містить велику кількість вікон та не містить комплексної та складної роботи з даними. Тому немає великої потреби застосовувати складніший шаблон (Модель-Представлення-Модель

Представлення), тому що, в даному випадку це принесе лише додатково витрачений час на розробку застосунку. Тому, прийнято рішення вибрати Модель Представлення Контролер в якості архітектурного шаблону.

1.2.3 Побудова діаграми класів

Основними сутностями програмної системи є розпізнаний текст та розпізнаний предмет. Тому для кожної з цих сутностей буде відповідати певний клас. RecognizedText – розпізнаний текст та RecognizedSubject – розпізнаний предмет. З точки зору архітектури MVC, ці класи виступають моделями. Проведемо опис кожного з цих класів.

Клас RecognizedText – буде містити наступні властивості: id, title, text, createdDate, image, accuracy. Властивість id – публічна властивість класу RecognizedText, типу String, виступає в якості унікального ідентифікатора конкретного розпізнаного тексту з усіх інших сутностей даного типу. Властивість title – публічна властивість класу RecognizedText, типу String, дана властивість буде створюватися на основі властивості text, а саме буде його початковою частиною. Дана властивість міститься у цьому класі для того, щоб було зручніше відображати даний компонент для користувача. Властивість text – публічна властивість класу RecognizedText, типу String, дана властивість є ключовим у даному класі, а саме вона зберігає результат розпізнавання тексту комп'ютером. Властивість createdDate – публічна властивість класу RecognizedText, типу Date, дана властивість зберігає дату та час коли системою був розпізнаний текст. Властивість image – публічна властивість класу RecognizedText, типу Image, дана властивість зберігає зображення на основі якого система проводила розпізнавання тексту. Властивість accuracy - публічна властивість класу RecognizedText, типу Double, дана властивість зберігає точність, з якою система визначила текст. Також даний клас має метод createTitle(). Метод createTitle() – приватний метод класу RecognizedText, даний метод на вхід приймає змінну типу String

та повертає значення типу String. Дана функція використовується для визначання значення властивості title, на основі властивості text. UML діаграму класу відображено на рисунку 1.6.

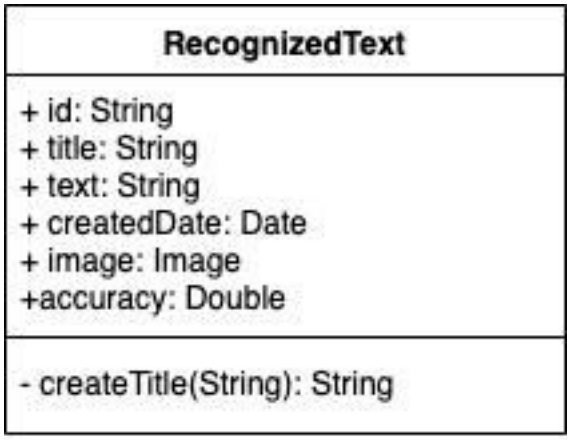

Рисунок 1.6 – UML діаграма класу «RecognizedText»

Клас RecognizedSubject – буде містити наступні властивості: id, result, createdDate, image, accuracy. Властивість id – публічна властивість класу RecognizedSubject, типу String, виступає в якості унікального ідентифікатора конкретного розпізнаного предмету з усіх інших сутностей даного типу. Властивість result – публічна властивість класу RecognizedSubject, типу String, дана властивість є ключовим у даному класі, а саме вона зберігає результат розпізнавання предмету комп'ютером. Властивість createdDate – публічна властивість класу RecognizedSubject, типу Date, дана властивість зберігає дату та час коли системою був розпізнаний текст. Властивість image – публічна властивість класу RecognizedSubject, типу Image, дана властивість зберігає зображення на основі якого система проводила розпізнавання тексту. Властивість accuracy - публічна властивість класу RecognizedSubject, типу Double, дана властивість зберігає точність, з якою система визначила текст. UML діаграму класу відображено на рисунку 1.7, аркуш 38.

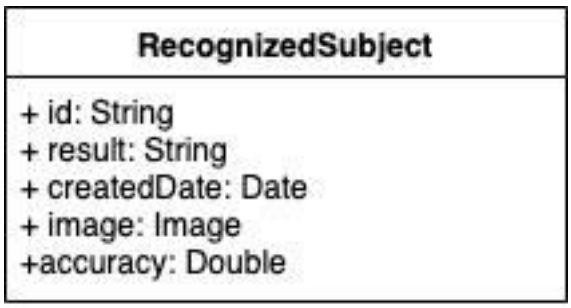

Рисунок 1.7 – UML діаграма класу «RecognizedSubject»

Оскільки, користувач повинен мати змогу провести збереження, перегляду, оновлення та видалення розпізнаного тексту чи об’єкту, нам потрібно створити сервіси, які інкапсулюють у собі це функціонал. Тому створимо наступні сервіси: RecognizedSubjectStorageService та RecognizedTextStorageService. Кожен із цих сервісів повинен містити у собі логіку роботи з збереженням, переглядом, оновленням та видаленням відповідних об’єктів. Оскільки з плином часу спосіб збереження даних може змінюватися, для того, щоб не створювати зайвої залежності шара Відображення від шара роботи з базою даних чи іншим засобом для збереження даних, варто створити інтерфейси RecognizedSubjectStorageServiceInterface та RecognizedTextStorageServiceInterface.

Інтерфейс – це набір методів та властивостей, що дозволяє проводити «спілкування» різних компонентів програмної системи між собою, уникаючи залежності від реалізації даного компонента.

Інтерфейс RecognizedSubjectStorageServiceInterface буде містити наступні методи: getAllRecognizedSubjects(), getRecognizedSubjectById(), sortRecognizedSubjects(), updateRecognizedSubject(), removeRecognizedSubject(). Метод getAllRecognizedSubjects() – публічний метод інтерфейсу RecognizedSubjectStorageServiceInterface, який не містить вхідних параметрів, а на виході повертає список розпізнаних предметів. Даний метод використовується для того, щоб витягнути усі збережені розпізнані предмети з бази даних. Метод getRecognizedSubjectById() –

публічний метод інтерфейсу RecognizedSubjectStorageServiceInterface, який у якості вхідних параметрів приймає значення типу String – id розпізнаного предмету, який ми хочемо знайти у базі даних та повертає конкретний розпізнаний об’єкт типу RecognizedSubject. Метод sortRecognizedSubjects() – публічний метод інтерфейсу RecognizedSubjectStorageServiceInterface, який приймає у якості параметру значення типу Bool – дане значення показує чи сортування потрібно провести від старших записів до новіших чи навпаки від новіших записів до старших, та повертає список посортованих значень типу RecognizedSubject. Метод здійснює звернення в базу даних із проханням здійснити сортування усіх записів відносно дати їх додавання у цю базу даних. Метод updateRecognizedSubject() – публічний метод інтерфейсу RecognizedSubjectStorageServiceInterface, який приймає у якості параметру, значення типу RecognizedSubject та не повертає жодного значення. Метод здійснює оновлення вибраного запису у базі даних. Метод removeRecognizedSubject() – публічний метод інтерфейсу RecognizedSubjectStorageServiceInterface, який приймає у якості параметру значення типу String – id розпізнаного об’кту, який потрібно видалити з бази даних, та не повертає жодного значення. UML діаграму класу відображено на рисунку 1.8.

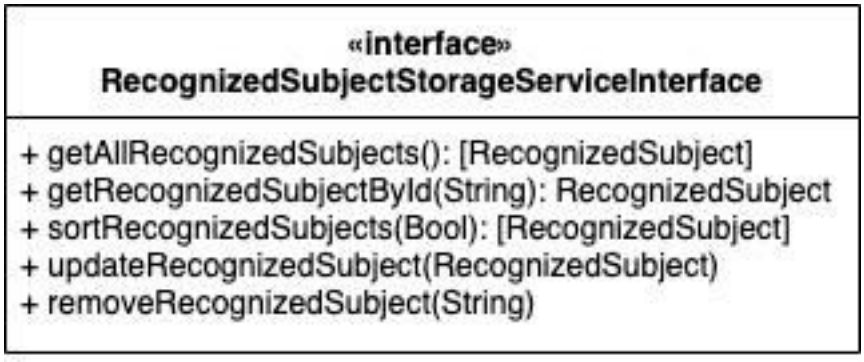

Рисунок 1.8 – UML діаграма інтерфейсу
«RecognizedSubjectStorageServiceInterface»

Інтерфейс RecognizedTextStorageServiceInterface буде містити наступні методи: getAllRecognizedTexts(), getRecognizedTextById(),

sortRecognizedTexts(), updateRecognizedText(), removeRecognizedText(). Метод getAllRecognizedTexts() – публічний метод інтерфейсу RecognizedTextStorageServiceInterface, який не містить вхідних параметрів, а на виході повертає список розпізнаних текстів. Даний метод використовується для того, щоб витягнути усі збережені розпізнані тексти з бази даних. Метод getRecognizedTextById () – публічний метод інтерфейсу RecognizedTextStorageServiceInterface, який у якості вхідних параметрів приймає значення типу String – id розпізнаного тексту, який ми хочемо знайти у базі даних та повертає конкретний розпізнаний об'єкт типу RecognizedText. Метод sortRecognizedTexts() – публічний метод інтерфейсу RecognizedTextStorageServiceInterface, який приймає у якості параметру значення типу Bool – дане значення показує чи сортування потрібно провести від старших записів до новіших чи навпаки від новіших записів до старших, та повертає список посортованих значень типу RecognizedText. Метод здійснює звернення в базу даних із проханням здійснити сортування усіх записів відносно дати їх додавання у цю базу даних. Метод updateRecognizedText() – публічний метод інтерфейсу RecognizedTextStorageServiceInterface, який приймає у якості параметру, значення типу RecognizedSubject та не повертає жодного значення. Метод здійснює оновлення вибраного запису у базі даних. Метод removeRecognizedText() – публічний метод інтерфейсу RecognizedTextStorageServiceInterface, який приймає у якості параметру значення типу String – id розпізнаного об'кту, який потрібно видалити з бази даних, та не повертає жодного значення. UML діаграму класу відображено на рисунку 1.9, аркуш 41.

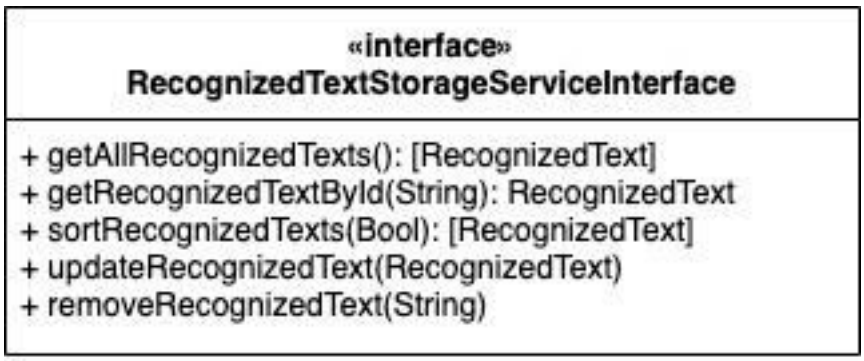

Рисунок 1.9 – UML діаграма інтерфейсу
«RecognizedTextStorageServiceInterface»

Повернемося до сервісів. RecognizedSubjectStorageService та RecognizedTextStorageService – сервіси, що повинні імлементовувати, тобто реалізовувати дані інтерфейси. А це в свою чергу значить, що у даних сервісах будуть усі ті самі методи, що в описаних вище інтерфейсах. На даному етапі важко відзначити, які у даних класів будуть власні властивості та методи, адже спосіб збереження даних буде обрано на етапі реалізації програмної системи.

Розглянемо наступний компонент MVC – Контролери. Як уже було зазначено на етапі вибору архітектурного шаблону для реалізації програмної системи Контролер – це посередник між Представленням та Моделлю. В даній програмній системі будуть наступні Контролери: MainTabBarController, TextRecognizeMenuViewController, RecognizedTextsViewController, TextRecognizeViewController, SubjectRecognizeMenuViewController, RecognizedSubjectsViewController, SubjectRecognizeViewController.

Клас MainTabBarController – виступає у якості контейнера для двох інших Контролерів – TextRecognizeMenuViewController та SubjectRecognizeMenuViewController. Він містить наступні властивості: textRecognizeMenuViewController та subjectRecognizeMenuViewController. Властивість textRecognizeMenuViewController – приватна властивість класу MainTabBarController, яка містить у собі об’єкт класу

TextRecognizeMenuViewController. Властивість subjectRecognizeMenuViewController – приватна властивість класу MainTabBarController, яка містить у собі об’єкт класу SubjectRecognizeMenuViewController. Клас MainTabBarController містить у собі також метод didLoad, у якому здійснюються усі операції, що виконаються при додаванні цього класу у пам’ять. UML діаграму класу відображено на рисунку 1.10.

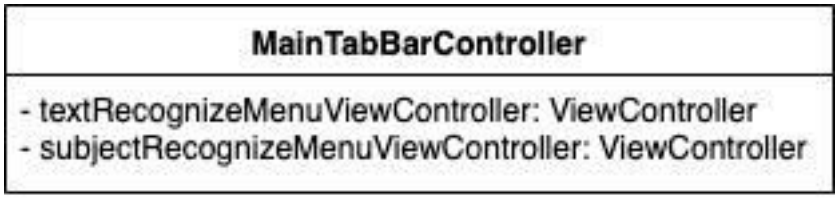

Рисунок 1.10 – UML діаграма класу «MainTabBarController»

Клас TextRecognizeMenuViewController – є кореневим класом для усіх класів, що працюють із розпізнаванням тексту. Він містить наступні властивості: recognizeTextButton, reconizedTextsButton. Властивість recognizeTextButton – приватна властивість класу TextRecognizeMenuViewController типу Button. Ця властивість містить у собі клавішу, що відповідає за відкривання екрану розпізнавання тексту. Змінюючи цю властивість можна редагувати вигляд та функціонування клавіші. Властивість reconizedTextsButton – приватна властивість класу TextRecognizeMenuViewController типу Button. Ця властивість містить у собі клавішу, що відповідає за відкривання екрану розпізнаних текстів. Змінюючи цю властивість можна редагувати вигляд та функціонування клавіші. Також цей клас містить методи: didLoad, recognizeTextButtonTapped, reconizedTextsButtonTapped. Метод didLoad містить у собі усі операції, що виконаються при додаванні цього класу у пам’ять. Метод recognizeTextButtonTapped містить у собі операції, які будуть виконані у результаті натискання клавіші recognizeTextButton, а саме відкривання екрану розпізнавання тексту. Метод reconizedTextsButtonTapped містить у

собі операції, які будуть виконані у результаті натискання клавіші reconizedTextsButton, а саме відкривання екрану, що відображає список розпізнаних текстів. UML діаграму класу відображено на рисунку 1.10.

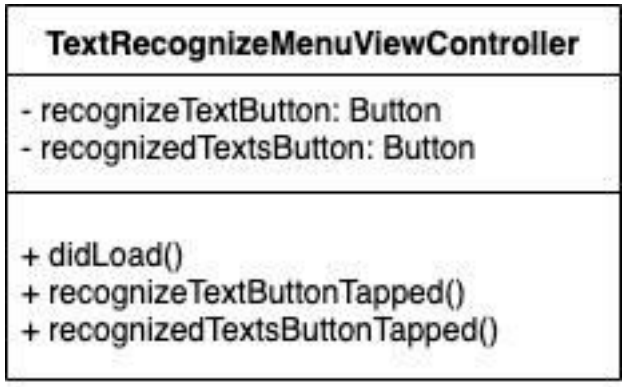

Рисунок 1.11 – UML діаграма класу «TextRecognizeMenuViewController»

Клас SubjectRecognizeMenuViewController – є кореневим класом для усіх класів, що працюють із розпізнаванням предметів. Він містить наступні властивості: recognizeSubjectButton, reconizedSubjectsButton. Властивість recognizeSubjectsButton – приватна властивість класу SubjectRecognizeMenuViewController типу Button. Ця властивість містить у собі клавішу, що відповідає за відкривання екрану розпізнавання тексту. Змінюючи цю властивість можна редагувати вигляд та функціонування клавіші. Властивість reconizedSubjectsButton – приватна властивість класу SubjectRecognizeMenuViewController типу Button. Ця властивість містить у собі клавішу, що відповідає за відкривання екрану розпізнаних текстів. Змінюючи цю властивість можна редагувати вигляд та функціонування клавіші. Також цей клас містить методи: didLoad, recognizeSubjectButtonTapped, reconizedSubjectButtonTapped. Метод didLoad містить у собі усі операції, що виконаються при додаванні цього класу у пам’ять. Метод recognizeSubjectButtonTapped містить у собі операції, які будуть виконані у результаті натискання клавіші recognizeSubjectButton, а саме відкривання екрану розпізнавання тексту. Метод reconizedSubjectsButtonTapped містить у собі операції, які будуть виконані у

результаті натискання клавіші reconizedSubjectsButton, а саме відкривання екрану, що відображає список розпізнаних текстів. UML діаграму класу відображено на рисунку 1.11.

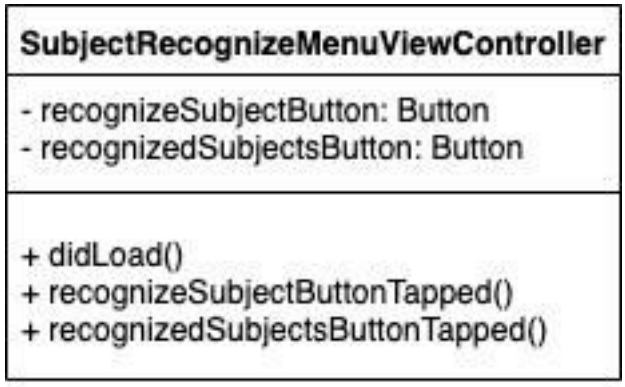

Рисунок 1.11 – UML діаграма класу «SubjectRecognizeMenuViewController»

Клас TextRecognizeViewController – відповідає за управління екраном на якому відбувається процес розпізнавання тексту. Даний клас містить наступні властивості: recognizedTextImageView, recognizedTextTitleLabel, recognizedTextView, actionButton, viewState, recognizedTextStorageService. Властивість recognizedTextImageView – приватна властивість класу TextRecognizeViewController типу ImageView. Ця властивість містить у собі представлення зображення отримане для виконання процесу розпізнавання. Властивість recognizedTextTitleLabel – приватна властивість класу TextRecognizeViewController типу Label. Ця властивість мітить у собі компонент відповідальний за відображення заголовку розпізнаного тексту. Властивість recognizedTextView – приватна властивість класу TextRecognizeViewController типу TextView. Ця властивість мітить у собі компонент відповідальний за відображення розпізнаного тексту. Властивість actionButton – приватна властивість класу TextRecognizeViewController типу Button. Ця властивість мітить у собі компонент відповідальний за дію, яка повинна бути виконана у результаті натискання на неї. Властивість viewState – публічна властивість класу TextRecognizeViewController. Дана властивість має три можливих значення редагування, розпізнавання та перегляд. В

залежності від значення цієї властивості змінюється поведінка інших компонентів цього класу. Властивість recognizedTextStorageService – приватна властивість класу TextRecognizeViewController, дана властивість містить у собі логіку з роботою з відповідним сервісом. Даний клас містить у собі наступні методи: didLoad(), actionButtonTapped(), viewTapped(). Метод didLoad містить у собі усі операції, що виконаються при додаванні цього класу у пам’ять. Метод actionButtonTapped() містить собі операції, що виконаються при натисканні на клавішу actionButton. Ця дія залежить від значення властивості viewState. Якщо значення редагування то клавіша відправить запит на збереження змін, якщо значення розпізнавання то клавіша відправить запит на збереження результатів розпізнавання, якщо значення перегляд то клавіша надає можливість користувачу вносити змінювати результат розпізнавання. Метод viewTapped() відповідає за запуск озвучування результату розпізнавання тексту. UML діаграму класу відображено на рисунку 1.12.

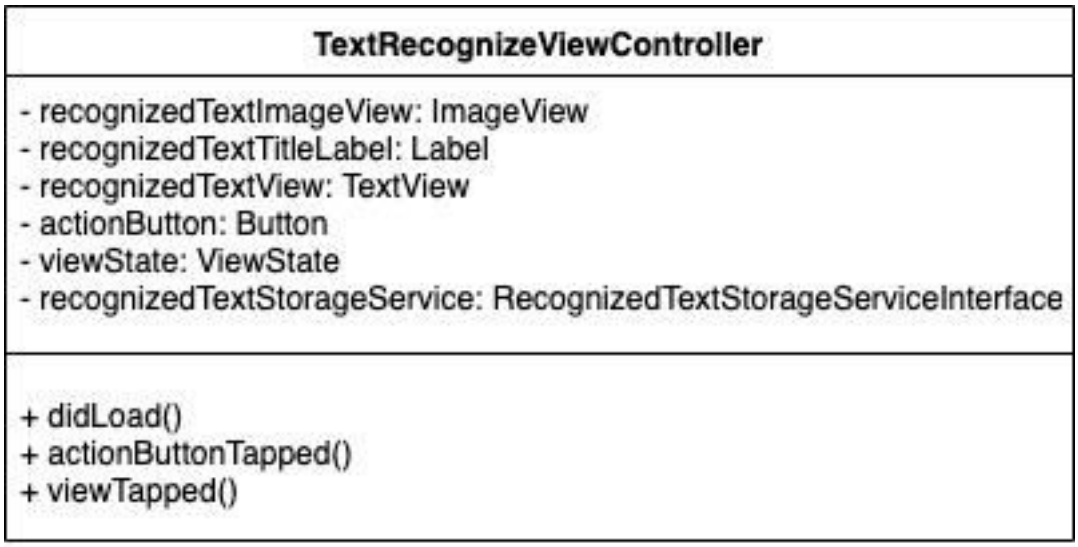

Рисунок 1.12 – UML діаграма класу «TextRecognizeViewController»

Клас SubjectRecognizeViewController – відповідає за управління екраном на якому відбувається процес розпізнавання тексту. Даний клас містить наступні властивості: recognizedSubjectImageView,

recognizedResultLabel, actionButton, viewState. Властивість recognizedSubjectImageView – приватна властивість класу SubjectRecognizeViewController типу ImageView. Ця властивість містить у собі представлення зображення отримане для виконання процесу розпізнавання. Властивість recognizedResultLabel – приватна властивість класу SubjectRecognizeViewController типу Label. Ця властивість мітить у собі компонент відповідальний за відображення результату розпізнавання. Властивість actionButton – приватна властивість класу SubjectRecognizeViewController типу Button. Ця властивість мітить у собі компонент відповідальний за дію, яка повинна бути виконана у результаті натискання на неї. Властивість viewState – публічна властивість класу SubjectRecognizeViewController. Дана властивість має три можливих значення редагування, розпізнавання та перегляд. В залежності від значення цієї властивості змінюється поведінка інших компонентів цього класу. Даний клас містить у собі наступні методи: didLoad(), actionButtonTapped(), viewTapped(). Метод didLoad() містить у собі усі операції, що виконаються при додаванні цього класу у пам’ять. Метод actionButtonTapped() містить собі операції, що виконаються при натисканні на клавішу actionButton. Ця дія залежить від значення властивості viewState. Якщо значення редагування то клавіша відправить запит на збереження змін, якщо значення розпізнавання то клавіша відправить запит на збереження результатів розпізнавання, якщо значення перегляд то клавіша надає можливість користувачу вносити змінювати результат розпізнавання. Метод viewTapped() відповідає за запуск озвучування результату розпізнавання предмету. UML діаграму класу відображено на рисунку 1.13, аркуш 47.

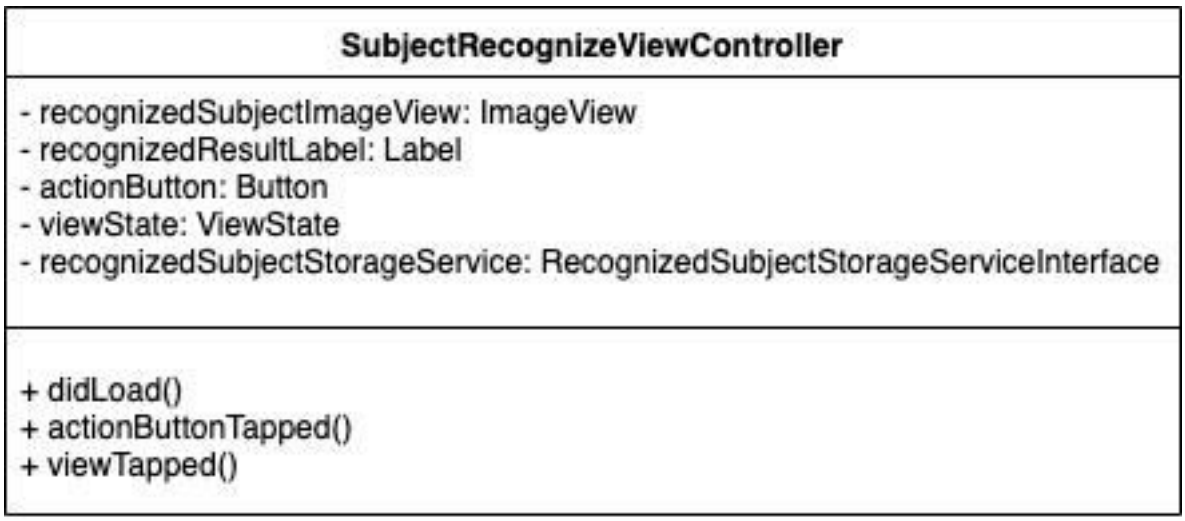

Рисунок 1.13 – UML діаграма класу «SubjectRecognizeViewController»

Клас RecognizedTextsViewController – відповідає за управління екраном, що відповідає за відображення списку усіх розпізнаних текстів. Даний клас містить наступні властивості: tableView, recognizedTextStorageService. Властивість tableView – приватна властивість класу RecognizedTextsViewController, типу TableView, відповідає за відображення списку розпізнаних текстів. Властивість recognizedTextStorageService – приватна властивість класу RecognizedTextsViewController, дана властивість містить у собі логіку з роботою з відповідним сервісом. Також клас RecognizedTextsViewController містить наступні методи: didLoad(), downloadData(), removeTextWithId(), openTextInfoScreen(). Метод didLoad() містить у собі усі операції, що виконаються при додаванні цього класу у пам’ять. Метод downloadData() викликає відповідний метод recognizedTextStorageService для завантаження з бази даних списку розпізнаних текстів. Метод removeTextWithId() викликає відповідний метод recognizedTextStorageService для видалення конкретного запису з бази даних. Метод openTextInfoScreen() відкриває екран для детального відображення конкретного елемента списку. UML діаграму класу відображено на рисунку 1.14, аркуш 48.

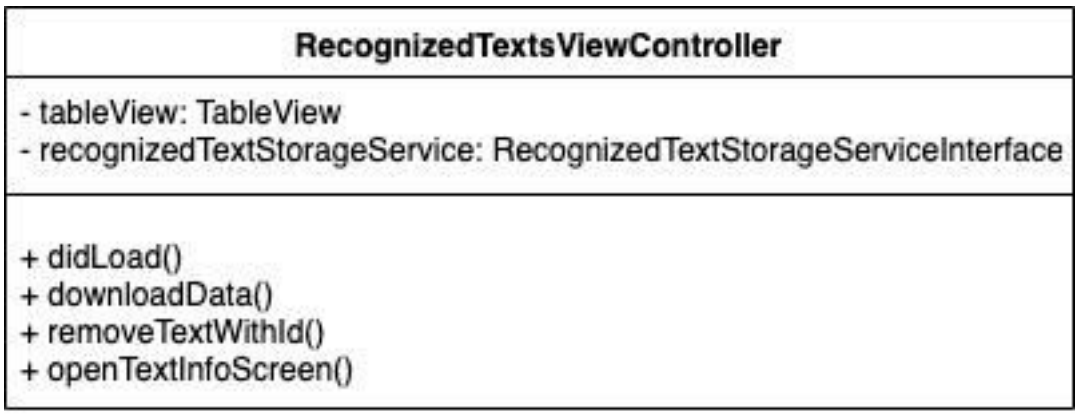

Рисунок 1.14 – UML діаграма класу «RecognizedTextsViewController»

Клас RecognizedSubjectsViewController – відповідає за управління екраном, що відповідає за відображення списку усіх розпізнаних предметів. Даний клас містить наступні властивості: tableView, recognizedSubjectStorageService. Властивість tableView – приватна властивість класу RecognizedSubjectsViewController, типу TableView, відповідає за відображення списку розпізнаних предметів. Властивість recognizedSubjectStorageService – приватна властивість класу RecognizedSubjectsViewController, дана властивість містить у собі логіку з роботою з відповідним сервісом. Також клас RecognizedSubjectsViewController містить наступні методи: didLoad(), downloadData(), removeSubjectWithId(), openSubjectInfoScreen(). Метод didLoad() містить у собі усі операції, що виконаються при додаванні цього класу у пам’ять. Метод downloadData() викликає відповідний метод recognizedTextStorageService для завантаження з бази даних списку розпізнаних текстів. Метод removeSubjectWithId() викликає відповідний метод recognizedTextStorageService для видалення конкретного запису з бази даних. Метод openSubjectInfoScreen() відкриває екран для детального відображення конкретного елемента списку. UML діаграму класу відображено на рисунку 1.15, аркуш 49.

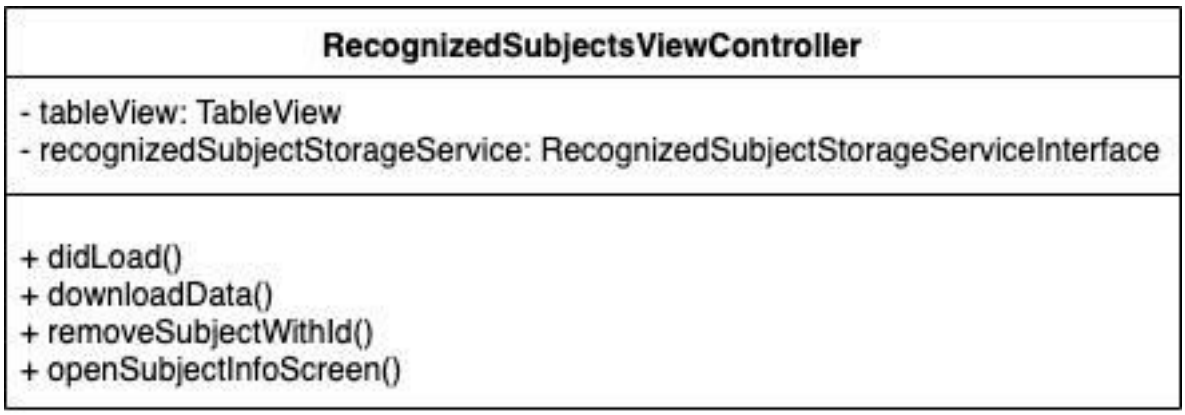

Рисунок 1.15 – UML діаграма класу «RecognizedSubjectsViewController»

UML діаграму класів системи та зв’язків між цими класами відображено на рисунку 1.16.

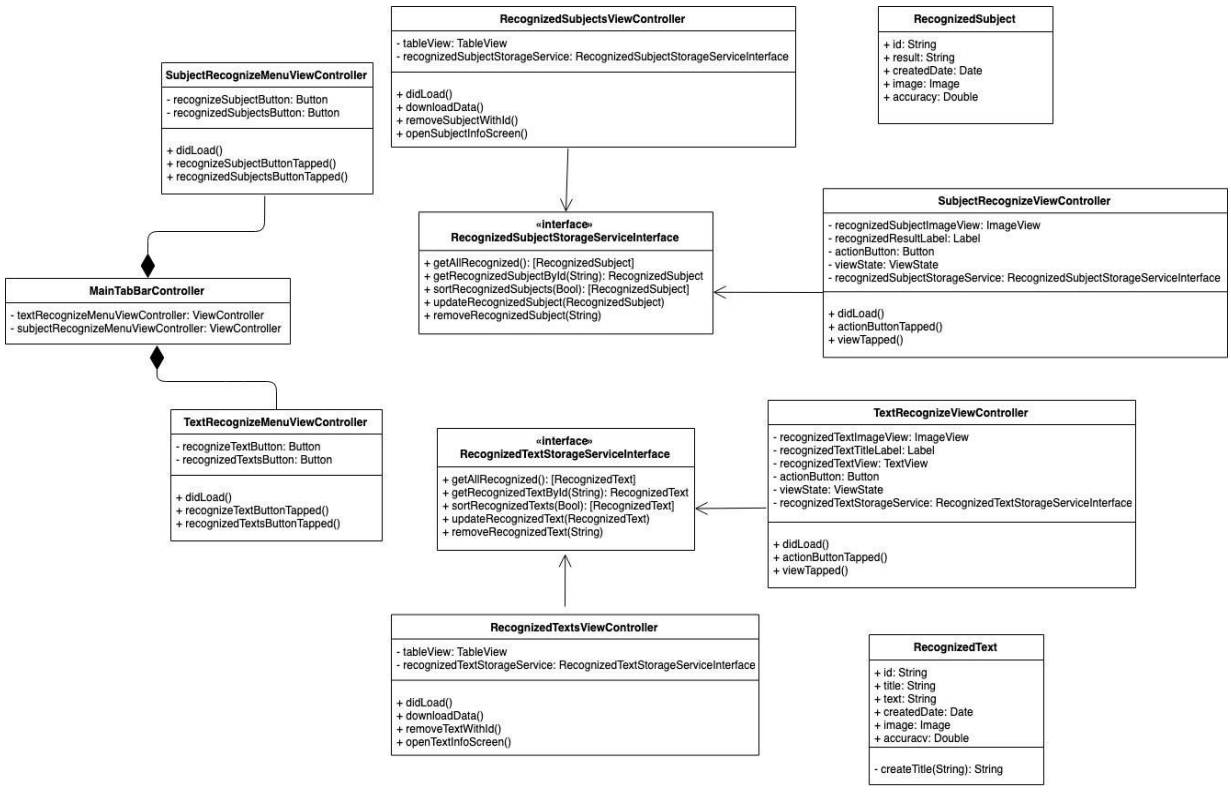

Рисунок 1.16 – UML діаграма класів системи

2. РЕАЛІЗАЦІЯ ПРОГРАМНОЇ СИСТЕМИ

2.1 Вибір мови програмування

Дана програмна система розробляється для мобільної операційної системи – iOS. Для реалізації такої програмної системи можна використати одну із двох мов програмування – Objective-C та Swift. Для того, щоб обрати один із двох можливих варіантів потрібно проаналізувати переваги та недоліки кожної із них.

Для початку розглянемо мову Objective-C.

Objective-C був створений Бредом Коксом і Томом Лаввом у 1984 році як розширення C. Це додало мові C обміну повідомленнями стилю SmallTalk та об’єктної орієнтації[10].

Переваги мови Objective-C:

- Взаємодія з C ++ та Objective-C ++
- Динамічні особливості, такі як шиплячий метод
- Краща підтримка для написання двійкових фреймворків.

Недоліки Objective-C:

- Оскільки Objective-C побудований поверх C, йому не вистачає простору імен. Усі класи в додатку Objective-C повинні бути унікальними на рівні додатку. Отже, щоб уникнути зіткнення, існує умова префіксації назв класів. З цієї причини у нас є префікс 'NS' для класу в Foundation Framework та префікс 'UI' для класів в UIKit.
- Явні покажчики.
- Можливість надсилати повідомлення на об’єкт нуля без збоїв і відсутність суворого введення призводить до помилок, які важко відстежити та виправити.
- Мова синтаксично багатослівна і складна, але це очікується, враховуючи, що це досить стара мова.

Swift - це молода мова, випущена в 2014 році. Вона створена, щоб бути безпечною та ефективною із сучасними синтаксисом та особливостями. Swift став відкритим джерелом в грудні 2015 року[10].

Переваги мови Swift:

- Swift безпечний завдяки статичному набору тексту та використанню додаткових пристроїв та додаткових ланцюжків.
- Підтримка просторів імен, чіткий синтаксис змінності, функціональні зразки та стислий синтаксис.
- Інтерактивна розробка з використанням ігрових майданчиків.
- Swift є ефективним і знаходить своє місце в додатках на сервері. Переваги використання Swift на стороні сервера пояснюються в цій бесіді в Realm Academy Крісом Бейлі, де він вказує на перевагу, яку має Swift перед іншими рамками на сервері та хмарі. За його словами, Swift високоефективний і має низький слід пам'яті, що робить його ідеальним вибором для розвитку сервера.
- Swift тепер стабільний, його ABI заблоковано.
- Стандартний код бібліотеки Swift містить близько 42,5% його коду в Swift. Розділення різних мов, що використовуються у стандартній бібліотеці, показано на малюнку нижче. Цей код Swift - це, мабуть, найкращий код Swift, на який розробники можуть звернутися, щоб поліпшити кодування Swift. Це дуже добре висвітлюється в цій розмові.
- SwiftUI - це лише декларативна база Swift для створення інтерфейсу користувача для декількох платформ із вбудованою підтримкою темного режиму та іншими функціями доступності. SwiftUI повністю сумісний з UIKit, і SwiftUI View може вбудувати UIView / NSView, який також може вбудувати представлення SwiftUI.
- Попередній перегляд SwiftUI доступний прямо з Xcode без запуску проекту на тренажері (даючи миттєвий візуальний зворотній

зв'язок). Також пристрій попереднього перегляду можна вмикати на льоту, додавши модифікатор для пристрою попереднього перегляду, без клопоту зі створення та запуску проекту на іншому симуляторі.

Недоліки Swift:

- Більший час компіляції.
- Немає прямого способу використання C ++ бібліотек.
- Стабільність формату модулів все ще не досягнута і потрібна розробникам, які хочуть ділитися своїм кодом як бінарні фреймворки.

Єдиною суттєвою перевагою мови Objective-C є сумісність цієї мови із мовами C++ та Objective-C++. Оскільки у контексті розроблювальної програмної системи дана перевага не є впливовою було вирішено обрати Swift у якості мови програмування, адже він швидший, сучасніший та простіший для розуміння і використання.

2.2 Вибір інструменту для збереження даних

Для збереження даних в додатках для мобільної платформи iOS з використанням мови програмування Swift можна використати наступні засоби збереження даних: Core Data та UserDefaults. Для того, щоб обрати необхідний засіб, потрібно проаналізувати усі недоліки та переваги кожного із них.

Core Data використовується, щоб зберегти постійні дані програми для використання в режимі офлайн, кешувати тимчасові дані та додавати функцію скасування функцій у ваш додаток на одному пристрої. За допомогою редактора Модель даних Core Data Ви визначаєте типи та відносини даних та генеруєте відповідні визначення класів. Потім основні дані можуть керувати екземплярами об'єктів під час виконання, надаючи такі функції.

Core Data резюмують деталі відображення ваших об’єктів у магазині, що дозволяє легко зберігати дані з Swift та Objective-C без адміністрування бази даних безпосередньо[11].

Менеджер скасування змін у Core Data відстежує зміни і може їх відкручувати окремо, групами або всі відразу, що полегшує додавання скасування та повторну підтримку у вашому додатку.

Виконуйте потенційно завдання, що блокують користувальницький інтерфейс, наприклад розбір JSON на об'єкти у фоновому режимі. Потім ви можете кешувати або зберігати результати, щоб зменшити зворотні сторони сервера.

Core Data також допомагають зберігати ваші уявлення та дані синхронізовано, надаючи джерела даних для подань таблиці та колекції.

Core Data включають механізми модифікації вашої моделі даних та міграції даних користувачів у міру розвитку вашої програми[11].

Клас UserDefaults пропонує програмний інтерфейс для взаємодії з системою за замовчуванням. Система за замовчуванням дозволяє додатку налаштувати свою поведінку відповідно до вподобань користувача. Наприклад, ви можете дозволити користувачам вказувати бажані одиниці вимірювання або швидкість відтворення медіа. Програми зберігають ці налаштування, призначаючи значення набору параметрів у базі даних за замовчуванням користувача[12].

Параметри називаються за замовчуванням, оскільки вони зазвичай використовуються для визначення стану програми за замовчуванням програми під час запуску або того, як воно діє за замовчуванням.

Під час виконання ви використовуєте об'єкти UserDefaults, щоб читати типові параметри, які використовується вашим додатком, із бази даних за замовчуванням користувача.

UserDefaults кешує інформацію, щоб уникнути необхідності відкривати базу даних за замовчуванням користувача кожного разу, коли вам потрібно значення за замовчуванням. Коли ви встановите значення за замовчуванням,

воно змінюється синхронно у вашому процесі та асинхронно для постійного зберігання та інших процесів.

Клас UserDefaults надає зручні методи доступу до поширених типів, таких як поплавці, парні, цілі числа, булеві значення та URL-адреси. Ці методи описані в Налаштування значень за замовчуванням. Об'єктом за замовчуванням повинен бути список властивостей - тобто екземпляр (або для колекцій, комбінація екземплярів) NSData, NSString, NSNumber, NSDate, NSArray або NSDictionary[12].

Якщо ви хочете зберегти будь-який інший тип об'єкта, вам слід, як правило, архівувати його, щоб створити екземпляр NSData. Значення, повернені з UserDefaults, незмінні, навіть якщо ви встановили як об'єкт, що змінюється.

Наприклад, якщо ви встановите змінну рядок як значення для "MyStringDefault", рядок, який ви отримаєте пізніше, використовуючи рядок (метод forKey :) буде незмінним. Якщо встановити змінну рядок як значення за замовчуванням і пізніше змінити рядок, значення за замовчуванням не відображатиме змінене значення рядка, якщо ви не зателефонуєте набір (_: forKey :) знову.

Оскільки UserDefaults найчастіше використовується для збереження малого об’єму інформації, а в розроблюваній програмній системі потрібно буде зберігати списки з розпізнаними даними користувача тому як засіб збереження даних було обрано Core Data.

2.3 Реалізація основних класі та методів системи

Реалізація програмної системи відбулася на мові Swift, з використанням середовища розробки Xcode. Для збереження даних було використано бібліотеку для роботи з базою даних Core Data. Для роботи з машинним навчанням було використано бібліотеку Vision, моделі для розпізнавання зображень було взято з офіційного джерела Apple.

Розглянемо декілька основних класів та методів в системі.

Одним із основних методів є метод, що відповідає за розпізнавання предмету по зображенню. Даний метод у якості параметру приймає зображення із типом CGImage та замикання із типом (VNClassificationObserbation?, Error?) -> (). Причиною використання замикання ж те, що процес розпізнавання зображення відбувається асинхронно, тобто результат виконання функції не буде отримано послідовно після виклику цієї функції. Код цієї функції наведено у лістингу 2.1.

```
func recognizeImage(image: CGImage,
                    completion: @escaping
                ((VNClassificationObservation?, Error?) -> ())) {
guard let model = try? VNCoreMLModel(for: Resnet50().model) else {
     return
}

let request = VNCoreMLRequest(model: model) { (finishedRequest, error)
     if let results = finishedRequest.results as?
[VNClassificationObservation] {
          guard let result = results.first else {
               completion(nil, nil)
               return
          }
          completion(result, nil)
     }

     if let error = error {
          completion(nil, error)
     }
}
try? VNImageRequestHandler(cgImage: image, options:
[:]).perform([request])
}
```

Лістинг 2.1 – Реалізація методу recognizeImage()

У процесі розробки було виявлено необхідність для дотримання ключових концепції об’єктно-орієнтованого програмування винести у окремий компонент клас, що відповідає за проведення озвучування результату розпізнавання предметів та тексту. Даний клас отримав назву VoiceOverService. Для реалізації класу було використано одну із вбудованих бібліотек AVFoundation.

Даний клас містить наступні властивості: synthesizer та rate. Властивість
synthesizer це екземпляр класу AVSpeechSynthesizer, що містить у собі логіку роботи із озвучуванням тексту. Властивість rate – це публічна властивість класу, вона відповідає за темп озвучування, публічна вона через те, щоб дозволити користувачу самому налаштовувати темп озвучування ззовні. Також клас містить функцію say(String). Ця функція власне і виконує озвучування тексту. Код цієї функції наведено у лістингу 2.2.

```
class VoiceOverService {

	private let synthesizer = AVSpeechSynthesizer()

	var rate: Float = AVSpeechUtteranceDefaultSpeechRate

	func say(_ phrase: String) {
		let utterance = AVSpeechUtterance(string: phrase)
		utterance.rate = rate
		utterance.voice = AVSpeechSynthesisVoice(language: "en-US")

		synthesizer.speak(utterance)
	}

}
```

Лістинг 2.2 – Реалізація класу VoiceOverService

Для роботи із базою даних було створено базовий сервіс BaseStorageService, який містить у собі логіку роботи із Core Data, основними налаштуваннями для цієї бібліотеки. Лістинг коду основної частини цього сервісу відображено у лістингу 2.3.

```
class BaseStorageService {
	static let shared = BaseStorageService()
	private var persistentContainer: NSPersistentContainer = {
		let container = NSPersistentContainer(name: "List")
		container.loadPersistentStores() { (_, error) in
			if let error = error {
				fatalError("Unresolved error, \((error as NSError).userInfo)")
			}
		}
		return container
	}()

	var managedContext: NSManagedObjectContext {
		return persistentContainer.viewContext
		}
	private init() {}
}
```

Лістинг 2.3 – Реалізація класу VoiceOverService

2.4 Використання системи

2.4.1 Системні вимоги

Оскільки програмна система розробляється лише для операційної системи iOS, то це ключовою системною вимогою є пристрій на якому

наявна така операційна система. Через те, що підтримка старіших версій досить накладна та оскільки на таких версіях часто некоректно працюють навіть стандартні бібліотеки, було прийнято рішення поставити мінімальну версію iOS – 12.

Також, для коректної роботи програмної системи та використання усього наявного функціоналу на пристрої повинні працювати камера та динаміки.

Для того, щоб коректно працював функціонал озвучування тексту та результатів розпізнавання, потрібно увімкнути у налаштуваннях смартфона функцію Voice Over.

2.4.2 Використання та тестування системи

Для тестування програмної систему були використані як і реальні пристрої так і симулятори. Для повноцінного тестування були використані пристрої із різними версіями iOS, а саме iOS 12.4 та 13.2.

Надалі буде продемонстровано тестування на симуляторі iPhone 11 Pro Max із версією iOS 13.2.

Після входу в систему користувач бачить екран Subject Menu, на якому йому пропонується дві опції – розпізнати предмет чи переглянути список наявних предметів. Зображення даного екрану на рисунку 2.1.

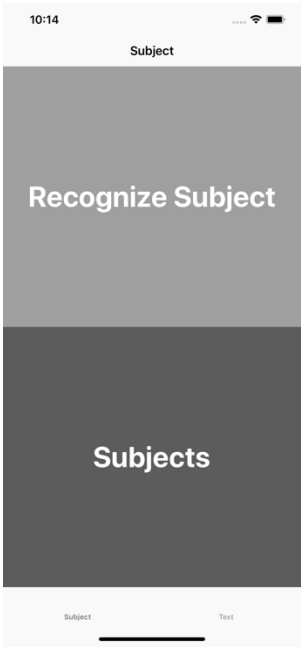

Рисунок 2.1 – Екран «Subject Menu»

Розглянемо опцію розпізнавання тексту (Recognize Subject). Користувач у Subject Menu натискає на клавішу із найменуванням Recognize Subject, перед ним появляється новий екран та показується вспливаюче вікно з вибором способу отримання зображення, а саме за допомогою камери телефона чи галереї телефона. На рисунку 2.2 відображено результат описаний вище.

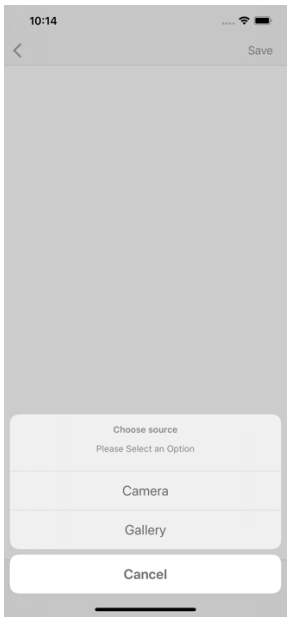

Рисунок 2.2 – Екран «Recognize Subject»

Розглянемо опцію «Gallery» та виберемо наперед заготовлену картинку олівця. Після вибору картинки система опрацювала її та показала наступний результат, який відображено на рисунку 2.3.

Система вдало опрацювала результат і показала, що вона вважає, що це олівець на 79%. Тому можна вважати, що даний функціонал у застосунку працює правильно.

Рисунок 2.3 – Результат розпізнавання зображення олівця

Наступний етап – тестування клавіші збереження (Save). Натиснемо дану клавішу. Система зберігає отриманий результат до локальної бази даних та повідомляє про успішний результат, це можна побачити на рисунку 2.4.

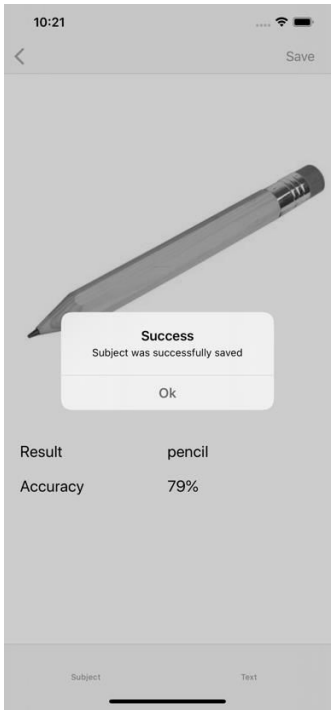

Рисунок 2.4 – Результат збереження результат розпізнавання до бази даних

Тепер варто перевірити чи показується щойно збережений результат на екрані збережених предметів. Для цього повернемося на початковий екран та виберемо опцію «Предмети» (Subject). Система витягує збережені результати у локальній базі даних і як видно із рисунку 2.5, показує збережений результат.

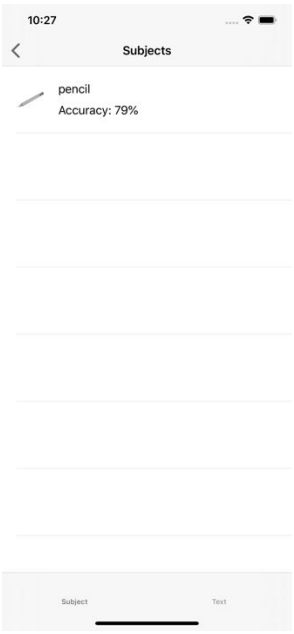

Рисунок 2.5 – Екран «Subjects»

3. ОРГАНІЗАЦІЙНО-ЕКОНОМІЧНА ЧАСТИНА

3.1 Загальний підхід до визначення економічної ефективності розробки.

Головною метою розділу виступає встановлення економічної доцільності розробки удосконалення програмного забезпечення, що буде використовуватись для розпізнавання тексту та визначення предметів.

Розробка нового програмного продукту вимагає свого управління і контролю з боку менеджера. Таким чином, складання та організація економічної частини є актуальною проблемою сучасного менеджменту.

Планування потребує будь-яке підприємство, будь-яке виробництво, економіка в цілому. Планування – оцінка можливостей, необхідністі і обсягів випуску конкурентоспроможної продукції, визначення місткості ринку і його

конкретного сегмента, оцінка попиту на продукцію, що випускається підприємством, результативності його роботи на ринку.

Економічне обґрунтування доцільності розробки програмного забезпечення. Зокрема розраховується комплексний показник якості проектного рішення, який показує його переваги в порівнянні з аналогами. А також на основі показника якості та ціни споживання проектного рішення та його аналога визначається коефіцієнт конкурентоздатності, який показує спроможність даного проектного рішення конкурувати з аналогами.

Приступаючи до розробки такого програмного забезпечення перш за все потрібно обґрунтувати його з точки зору економічної доцільності. Дане обґрунтування необхідне для того, щоб розробка була економічно ефективною і визначити чи варто розробляти новий програмний продукт, чи вигідніше модернізувати застарілий.

Економічний ефект розробленого продукту визначається на основі економічних показників, які дають можливість прогнозувати результат від впровадження даної програми. Враховуючи інтенсивний розвиток комп'ютерної техніки і широкий вибір програмного забезпечення, на сьогодні економічний аналіз є невід'ємною частиною попереднього аналізу, а кошти, що затрачаються, повинні бути еквівалентними тому ефекту, який принесе конкретне нововведення.

Оцінка вартості дослідницьких розробок базується на витратному підході: використанні первісної вартості об'єктів, виходячи з фактичних витрат на розробку та доведення до комерційного використання з урахуванням амортизації.

Згідно Статті 8 Закону № 3792-12 передбачено, що твори наукового характеру та комп’ютерні програми є об’єктами авторського права[13]. У разі реєстрації виконаної роботи як інтелектуальної власності та авторського права у державній службі інтелектуальної власності України необхідно буде сплатити збори за державну реєстрацію.

Для отримання відмінних результатів експериментів та доцільності розробки такого спеціалізованого ПЗ потрібні відповідні затрати на дослідження та розробку. Це і становитиме основу витрат, які будуть здійснені протягом підготовки та виконання реалізації даного рішення. Уявно модель витрат можна поділити на дві основні частини: витрати, пов’язанні на дослідження предметної області, побудову математичних моделей, отримання попередніх результатів експериментів, та частину реалізації програмної системи, архітектури та тестування.

Беручи до уваги залежність якості кінцевого продукту від кваліфікації інженерів програмного забезпечення, потрібно сконцентрувати увагу на якості та результативності розробки. Для цього потрібно провести ґрунтовний аналіз предметної області, залучити найновіші та інноваційні технології, провести ґрунтовне тестування та оцінку результатів розробки. Для забезпечення хорошої результативності при розробці доводиться йти на додаткові заходи заохочення та стимулювання у вигляді преміювання працівників.

До створення ПЗ можуть бути залучені позаштатні інженери програмного забезпечення як зареєстровані, так і не зареєстровані підприємцями. В обох випадках співпраця з ними здійснюється на підставі цивільно-правового договору, найчастіше – договорами підряду. Щодо оподаткування виплат за договором підряду, то все залежить від того, чи зареєстрований виконавець підприємцем. Важливим етапом розробки ПЗ є його тестування, яке виконується тестером за певну винагороду. Якщо тестери не перебувають з підприємством у трудових відносинах, оплата виконується на основі договору підряду на виконання робіт з тестування ПЗ. Сам же результат розробки не оподатковується, адже не є комерційним проектом і не спрямований на продаж.

3.2 Розрахунок вартості процесу розробки та оцінка економічної ефективності проекту

Для оцінки нематеріальних активів використовують міжнародні стандарти оцінки розрахунку вартості об’єктів інтелектуальної власності, розроблені TIAVIS (The International Assets Vabulation Standarts Commitie).

Виконання розробки програмного забезпечення з огляду економічної моделі можна виконувати двома способами: процедурним та об’єктно-орієнтованим. Обидва підходи потребують залучення ресурсів у вигляді програмістів-розробників, тестерів, керівника проекту, експерта у предметній області. Різниця виникає в самій схемі розробки, тривалості періоду розробки та відповідній вартості. Процедурний підхід для розробки ПЗ в основі якого лежать процедури і функції передбачає розробку ПЗ як монолітного композиту, що в подальшому, як правило, вимагає великих витрат на супровід та модернізацію. Об’єктно-орієнтований підхід, що ґрунтується на основі об’єктів певних класів, що описують певну область, описують певну поведінку (методи) та володіють властивостями (атрибутами), орієнтовані на варіанти використання та покроковий процес розробки.

Для початку робіт необхідно скласти технічне завдання на розробку, яке є основним документом, що регламентує подальшу роботу, та містить докладний опис необхідних функцій програми, інтерфейс, технології, інше. Вартість складання технічного завдання переважно складає до 10% від планованої вартості розробки. Роботу зі складання технічного завдання веде керівник проекту разом із програмістами та консультуючись із замовником.

Усі програмісти, що працюють у штаті підприємства-розробника мають встановлено певний посадовий оклад. Місячний оклад, денна заробітна плата, трудомісткість (днів) і основна заробітна плата кожного учасника технологічного процесу представлено у таблиці 3.1. Всі суми наведені в національній валюті – в гривні.

Таблиця 3.1 – Розрахункова вартість технологічного процесу розробки

Посада	Місячни	Денна	Об’єктно-	Процедурний

	й оклад, грн.	зар. плата, грн.	орієнтований підхід		підхід	
			Днів	Сума, грн.	Днів	Сума, грн.
Керівник проекту	30300,00	1377,00	14	19278,00	18	24786,00
Програміст	25500,00	1161,00	14	16254,00	18	20898 ,00
Тестер	16120,00	732,00	5	3660,00	5	3660,00
Всього			33	39192 грн.	41	49344 грн.

Витрати на науково-дослідницьку роботу та здійснення розробки програмних продуктів і об'єктно-орієнтованим, і процедурним способом включають:

Основна заробітна плата:

$$ЗП_{осн\ 1} = 39192 \text{ грн}; \qquad ЗП_{осн\ 2} = 49344 \text{ грн}.$$

Додаткова заробітна плата обчислюється як:

$$ЗП_{дод} = 0{,}2 \cdot ЗП_{осн} \tag{3.1}$$

$$ЗП_{дод\ 1} = 0{,}2 \cdot 39192 = 7838 \text{ грн};$$

$$ЗП_{дод\ 2} = 0{,}2 \cdot 49344 = 9868 \text{ грн}.$$

Таким чином загальний фонд заробітної плати, що обчислюється за формулою:

$$ФЗП = ЗП_{осн} + ЗП_{дод}. \tag{3.2}$$

$$ФЗП_1 = 39192 + 7838 = 47030 \text{ грн.};$$

$$ФЗП_2 = 49344 + 6332 = 55676 \text{ грн}.$$

Крім того, слід визначити відрахування на соціальні заходи:

- єдиний соціальний внесок – 3,6 %;
- військовий збір – 1,5 %;
- ПДФО (прибутковий податок) – 15 %.

Отже, сума відрахувань на соціальні заходи буде становити:

$$\text{Відр}_{\text{ЄСВ1}} = 0{,}036 \cdot \text{ФЗП} = 1693{,}08 \text{ грн.};$$

$$\text{Відр}_{\text{ЄСВ2}} = 0{,}036 \cdot \text{ФЗП} = 1993{,}53 \text{ грн.};$$

$$\text{Відр}_{\text{вз1}} = 0{,}015 \cdot \text{ФЗП} = 705{,}45 \text{ грн.};$$

$$\text{Відр}_{\text{вз2}} = 0{,}015 \cdot \text{ФЗП} = 830{,}64 \text{ грн.}$$

$$\text{Відр}_{\text{ПДФО1}} = 0{,}15 \cdot \text{ФЗП} = 7054{,}5 \text{ грн.};$$

$$\text{Відр}_{\text{ПДФО2}} = 0{,}15 \cdot \text{ФЗП} = 8306{,}4 \text{ грн..}$$

Нарахування на фонд оплати праці, які включають відрахування до Пенсійного фонду, фонду з тимчасової втрати працездатності, фонду з безробіття і фонду страхування від нещасних випадків на виробництві; для бюджетної організації тариф на фонд оплати праці встановлено на рівні 36,3%.

Зокрема, видання програмного забезпечення – 36,77%.

Нарахування на Фонд оплати праці (ФОП): $\text{ФОП}_{\text{ЄСВ}} = 0{,}3677 \cdot \text{ФЗП}$

$$\text{ФОП}_{\text{ЄСВ1}} = 0{,}3677 \cdot \text{ФЗП}_1 = 17292{,}93 \text{ грн.};$$

$$\text{ФОП}_{\text{ЄСВ2}} = 0{,}3677 \cdot \text{ФЗП}_2 = 20472{,}06 \text{ грн}$$

Всього витрат:

$$\text{В}_{\text{ЗП1}} = \text{ФЗП}_1 + \text{ФОП}_{\text{ЄСВ1}} = 47030 + 17292{,}93 = 64322{,}93 \text{ грн.};$$

$$\text{В}_{\text{ЗП2}} = \text{ФЗП}_2 + \text{ФОП}_{\text{ЄСВ2}} = 55676 + 20472{,}06 = 76148{,}06 \text{ грн.};$$

Також важливою складовою витрат є матеріальні витрати. Матеріальні витрати визначаються як добуток кількості витрачених матеріалів та їх ціни:

$$M_{в.і} = q_{\text{і}} \cdot p_{\text{і}}\,, \tag{3.3}$$

де: $q_{\text{і}}$ – кількість витраченого матеріалу і–го виду;

$p_{\text{і}}$ – ціна матеріалу і–го виду.

Звідси, загальні матеріальні витрати можна визначити:

$$З_{м.в} = \sum M_{В.і} \tag{3.4}$$

Проведені розрахунки занесемо у таблицю:

Таблиця 3.2 – Зведені розрахунки матеріальних витрат

Найменування ресурсу	Кількість, шт.	Ціна одиниці, грн	Загальна сума, грн
Флешки	2	227,00	454,00
Папір для друку А4, арк	500	0,2	100,00
Тонер для принтера	1	60,00	60,00
Всього			614,00

Отже загальна сума матеріальних витрат становить 614 гривень.

Розрахунок витрат на електроенергію одиниці обладнання визначаються за формулою:

$$З_в = W * T * S,$$

(3.5)

де W – необхідна потужність, кВт; T – кількість годин роботи обладнання; S – вартість кіловат-години електроенергії, S = 2,50 грн/кВт·год.

$З_{в1}$ =0.7 * 264 * 2.50 = 462 грн;

$З_{в2}$ =0.7 * 328 * 2.50 = 574 грн;

Розрахунок суми амортизаційних відрахувань. Комп'ютери та оргтехніка належать до четвертої групи основних фондів. Для цієї групи річна норма амортизації дорівнює 60 %, вартість яких перевищує 1000 грн. і визначається:

$$A = \frac{C_Б \cdot N_A \cdot T_{ФАК}}{T_{ГОД}} \qquad (3.6)$$

де $C_Б$ – балансова вартість обладнання, грн; N_A – норма амортизаційних відрахувань в рік, %; $T_{ФАК}$ – фактичний час роботи

обладнання по написанню програми, год; $T_{ГОД}$ – річний робочий фонд часу, год.

У даній формулі норма відрахувань на амортизацію рівна N_A = 0,6. Балансова вартість обладнання вказана в таблиці 3.3 і рівна $C_Б$ = 16045 гривень. Річний робочий фонд часу приймемо за $T_{ГОД}$ = 2120 годин. З них реальний фактичний робочий час становить $T_{ФАК}$ = 264 години згідно об'єктно-орієнтованого підходу та $T_{ФАК}$ = 328 годин згідно процедурного підходу.

Таблиця 3.3 – Перелік необхідного обладнання

Найменування	Кількість, шт	Ціна, грн	Сума, грн
Комп'ютер	2	32555,00	65111,00
Тестовий пристрій	2	16255,00	32510,00
Принтер	1	3500,00	3500,00
Середовища розробки	2	безкоштовно	безкоштовно
Операційна система	2	безкоштовно	безкоштовно
Всього більше 1000 грн.			101121,00
Всього			101121 грн.

$$A_1 = (101121 \cdot 0{,}6 \cdot 264)/2120 = 7555{,}45 \text{ грн.}$$
$$A_2 = (101121 \cdot 0{,}6 \cdot 328)/2120 = 9387{,}08 \text{ грн.}$$

Накладні витрати пов'язані з обслуговуванням виробництва, утриманням апарату управління та створення необхідних умов праці та закупівлю ресурсів та обладнання для розробки наведені в таблиці 3.3.

Залежно від організаційно-правової форми діяльності господарюючого суб'єкта, накладні витрати можуть становити 20–60 % від суми основної та додаткової заробітної плати працівників.

$$НВ = 0{,}5 \cdot ЗП_{осн} \quad (3.7)$$

де *НВ* – накладні витрати.

Отже, накладні витрати:

$$НВ_1 = 39192 \cdot 0{,}5 = 19596 \text{ грн.}$$

$$НВ_2 = 49344 \cdot 0{,}5 = 24672 \text{ грн.}$$

Проведемо розрахунок вартості створюваного програмного продукту. Вартість продукції включає у собі собівартість і планований прибуток. Найважливішим моментом для розробника, з економічної точки зору, є процес встановлення ціни.

Можна отримати загальні значення витрат на розробку та реалізацію проекту, враховуючи всі вище описані затрати та нарахування. Цей вид витрат складається з сум витрат на оплату праці (всього витрати на оплату праці), матеріальні затрати, затрати на електроенергію, накладні витрати, витрати на обладнання, враховуючи амортизації обладнання на час виконання проекту.

Собівартість продукції – це сума грошових витрат підприємства (фірми) на виробництво і збут одиниці продукції, виконання робіт та надання послуг[14].

Повна собівартість програмного продукту дорівнює сумі усіх витрат на його виробництво(таблиця 3.4):

Св1 = 84712,38 грн.; Св2 = 105063,14 грн.

Таблиця 3.4 – Розрахунок собівартості

Зміст витрат	Сума, грн	
	Об’єктно-орієнтований підхід	Процедурний підхід
Витрати на оплату праці	39192	49344

Відрахування на соціальні заходи	17292,93	20472,06
Матеріальні витрати	614	614
Витрати на електроенергію	462	574
Амортизаційні відрахування	7555,45	9387,08
Накладні витрати	19596	24672
Собівартість	84712,38	105063,14

Приймемо прибуток на рівні 25%. Для нових інноваційних продуктів, що користуються високим попитом на ринку, ринкову вартість Вр можна встановити вищу.

Отже, вартість розробленого програмного забезпечення:

Вр1 = Св1 + 0,3 · Св1 = 84712,38+ 0,25 · 84712,38= 105890,47 грн.;

Вр2 = Св2 + 0,3 · Св2 = 105063,14+ 0,25 · 105063,14= 131382,92 грн.

Ефективність виробництва – це узагальнене і повне відображення кінцевих результатів використання робочої сили, засобів та предметів праці на підприємстві за певний проміжок часу.

Економічна ефективність (Е) полягає у відношенні результату виробництва до затрачених ресурсів:

$$E = \frac{П}{Св} \tag{3.8}$$

де $П$ – прибуток, $П = В - Св$; $Св$ – собівартість.

Плановий прибуток ($П_{пл}$) знаходимо за формулою:

$$П_{пл} = В_р - С_в \tag{3.9}$$

Розраховуємо плановий прибуток:

$$П_{пл1} = 105890,47 - 84712,38 = 21178,09 \text{ грн.}$$

$$П_{пл2} = 131382,92 - 105063,14 = 26265,78 \text{ грн.}$$

Отже, формула для визначення економічної ефективності набуде вигляду:

$$E_p = \frac{П_{пл}}{C_в}. \quad (4.10)$$

$$E_{р1} = 21178,09 / 84712,38 = 0,25.$$

$$E_{р2} = 26265,78 / 105063,14 = 0,25.$$

Якщо ринкова вартість програмного продукту рівна прийнятій, то економічна ефективність визначається встановленим рівнем прибутку. Поряд із економічною ефективністю розраховують термін окупності капітальних вкладень ($Т_{ок}$):

$$Т_{ок} = \frac{1}{E} \quad (3.10)$$

Термін окупності дорівнює:

$$Т_{ок} = 1 / 0,25 = 4 \text{ роки}$$

У нашому випадку $Т_{ок1} = Т_{ок2} = 1/0,25 = 4$ років, що є нормальним, оскільки допустимим вважається термін окупності до 5 років.

Даний розрахунок виконаний у розрахунку на 1 екземпляр програмного продукту без врахування його тиражування.

Чистий приведений дохід (ЧПД) визначається як різниця між сукупними доходами (сукупний грошовий потік) і сукупними витратами (сукупними інвестиціями) взятими за весь період життя інвестицій і дисконтова ними в кожному році на фактор часу. Дисконтування являє собою визначення вартості майбутніх грошових потоків у теперішній момент часу. Ефективним вважається той проект, який забезпечує максимум ЧПД, оскільки при цьому досягається найвища дохідність власників інвестицій.

Визначення ЧПД відбувається за формулою:

$$ЧПД = \sum_{i=1}^{t} ГП_i \alpha_{TBi} - \sum_{i=1}^{t} ІК_i \alpha_{TBi} \quad (3.11)$$

де $ГП_i$ – грошовий потік і-го розрахункового року;

$ІК_i$ – сума інвестицій і-го розрахункового року;

α_{TBi} – коефіцієнт дисконтування (коефіцієнт приведення інвестицій і грошового потоку до теперішньої вартості).

Грошовий потік – це фінансовий показник, що характеризує ефект інвестицій у вигляді грошових коштів, які повертаються інвестору.

Коефіцієнт дисконтування показує, яку величину грошових коштів ми отримаємо з урахуванням фактору часу та ризиків. Він дозволяє перетворити майбутню вартість у вартість на даний момент. Для розрахунку коефіцієнта дисконтування (коефіцієнта приведення) грошових потоків за роками періоду економічного життя інвестицій використовується формула:

$$\alpha = \frac{1}{(1+i)^n} \quad (3.12)$$

де i – ставка дисконтування або норма дисконту, $i = 0{,}2$;

n – час або кількість періодів (років), протягом якого планується отримання доходу.

$\alpha_0 = 1$, $\alpha_1 = \frac{1}{(1+0.2)^n} = 0{,}83$

Вважатимемо, що обидва програмних продукта одинаково забезпечують потреби і вимоги споживача, і тому придбання першої чи другої програми одинаково вплинуть на розмір його додаткових доходів на вкладений капітал.

Тому приймемо цю величину за постійну, а порівняння дохідності двох проектів проведемо тільки за витратами.

$ЧПД'_1 = ГП + 0{,}83 * ГП - 105890{,}47 - 0{,}83 * 21178{,}09 = 1{,}83ГП - 123468{,}28$ грн.;

$ЧПД'_2 = ГП + 0{,}83 * ГП - 131382{,}92 - 0{,}83 * 26265{,}78 = 1{,}83ГП - 153183{,}52$ грн.

Чим менші витрати, тим більша дохідність проекту.

Економія витрат у випадку придбання, супроводу і одноразової модернізації програмного продукту, створеного за об’єктно-орієнтованим підходом, становить 29715,24 грн. Однак варто зауважити, що розробка спрямована на короткотривалу підтримку та не прогнозує модернізації. У разі ж необхідності розробки такого ж або суміжного ПЗ можна вважати за доцільно розпочинати розробку з початкових етапів, що потягне за собою нові витрати у повній мірі. Тобто у разі короткотермінової підтримки все ж за доцільніше обирати об’єктно-орієнтовану модель розробки, адже вартість короткотермінової підтримки згідно цієї моделі не вплине значно на сукупну вартість розробки.

Здана в експлуатацію система не завжди цілком завершена, її треба змінювати протягом терміну експлуатації. Внаслідок змін система стає більш складною і погано керованою. Об’єктно-орієнтоване представлення програми дозволяє навіть середньому програмісту швидко і ефективно супроводжувати і модернізувати програми, що значно скорочує подальші витрати на супровід і модернізацію.

Сумарні дані економічного розрахунку розробки даного проекту наведені в таблиці 3.5.

Таблиця 3.5 – Загальні витрати

Вид витрат	Об’єктно-орієнтований підхід, грн	Процедурний підхід, грн
Зарплата основна	39192	49344
Зарплата додаткова	7838	9868

Фонд заробітної плати, грн.	47030	59212
Відрахування на ФОП	17292,93	20472,06
Разом на оплату праці	111352,93	138896,06
Матеріальні витрати	614,00	614,00
Електроенергія	462	574
Амортизація	7555,45	9387,08
Накладні витрати	19596	24672
Разом на ін. витрати	17867,1	19403,7
Собівартість	84712,38	105063,14
Вартість розробленого ПЗ	105890,47	131382,92
Економічна ефективність	0,25	0,25
Термін окупності, років	4	4
Загальні витрати на розробку	123468,28	153468,52
Економія ($ЗВ_2$-$ЗВ_1$)	**29715,24**	

Загальна вартість пропонованих робіт становить 153468,52 гривень для процедурного і 123468,28 гривень для об’єктно-орієнтованого підходів розробки. В даному випадку реалізації проекту варто вибрати об’єктно орієнтований підхід для розробки даного ПЗ, адже фінансово це більш вигідно. Також у даній методиці розробки кращі часові рамки та перспективи підтримки і модернізації. У оцінці вартості продукту варто враховувати можливість фінансування проекту, що дозволить гнучко та ефективно підійти до розробки та організації праці.

4. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Охорона праці

Результатом виконання даної магістерської роботи буде програмна система для розпізнавання тексту та визначення предметів, яка буде використовуватись на комп’ютеризованому робочому місці.

Площа та об’єм для одного робочого місця визначають згідно з вимогами ДСанПіН 3.3.2-007-98[15]. Площа має бути не менше 6,0 кв. м, об’єм — не менше 20,0 куб. м.

При розміщенні робочих місць з персональними комп'ютерами відстань між робочими столами з екранами має бути не менше 2,0 м, а відстань між бічними поверхнями ВДТ - не менше 1,2 м.

Робочі місця з персональними комп'ютерами в приміщеннях з джерелами шкідливих виробничих факторів розміщуються в ізольованих кабінах з організованим повітрообміном. При виконанні творчої роботи, що вимагає значного розумового напруження або високої концентрації уваги, робочі місця з персональними комп'ютерами рекомендується ізолювати один від одного перегородками висотою 1,5-2,0 м.

Екран повинен знаходитися від очей користувача на відстані 600-700 мм, але не ближче 500 мм з урахуванням розмірів алфавітно-цифрових знаків і символів.

Відповідно до НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» конструкція робочого столу повинна забезпечувати оптимальне розміщення на робочій поверхні використовуваного обладнання з урахуванням його кількості і конструктивних особливостей відносно характеру виконуваної роботи[16]. Проте допускається використання робочих столів різних конструкцій, які відповідають сучасним вимогам ергономіки. Коефіцієнт відбиття для поверхні робочого столу повинен становити 0,5-0,7.

Висота робочої поверхні столу для дорослих користувачів повинна регулюватися в межах 680-800 мм; при відсутності такої можливості висота робочої поверхні столу повинна складати 725 мм. Модульними розмірами робочої поверхні столу для ПК, на підставі яких повинні розраховуватися конструктивні розміри, слід вважати: ширину - 800, 1000, 1200 і 1400 мм, глибину - 800 і 1000 мм при нерегульованої його висоті, рівній 725 мм.

Робочий стіл повинен мати простір для ніг висотою не менше 600 мм, шириною - не менше 500 мм, глибиною на рівні колін - не менше 450 мм, на рівні витягнутої ноги - не менше 650 мм.

Конструкція робочого стільця (крісла) повинна забезпечувати підтримку раціональної робочої пози під час роботи на персональному комп'ютері, дозволяти змінювати позу з метою зниження статичного напруження м'язів шийно-плечової області і спини для попередження розвитку втоми. Тип робочого стільця (крісла) слід вибирати з урахуванням зростання користувача, характеру та тривалості роботи з ПК[17].

Робоче місце користувача ПК слід обладнати підставкою для ніг, має ширину не менше 300 мм, глибину не менше 400 мм, регулювання по висоті в межах до 150 мм і за кутом нахилу опорної поверхні підставки до 20 град. Поверхня підставки повинна бути рифленою і мати по передньому краю бортик висотою 10 мм.

Клавіатуру слід розташовувати на поверхні столу на відстані 100-300 мм від краю, зверненого до користувача, або на спеціальній, регульованою по висоті робочої поверхні, відокремленої від основної стільниці.

Профілактичні заходи для зниження нервовоемоційного напруження викладено у ДСанПіН 3.3.2.007-98. Раціональний режим праці та відпочинку передбачає дотримання певної тривалості безперервної роботи на персональному комп'ютері і перерв, регламентованих з урахуванням тривалості робочої зміни, виду трудової діяльності.

При виконанні протягом робочої зміни робіт, що відносяться до різних видів трудової діяльності, за основну роботу з ПК слід сприймати

таку, що займає не менше 50% часу протягом робочої зміни або робочого дня.

Для попередження передчасної стомлюваності користувачів ПК рекомендується організовувати робочу зміну шляхом чергування робіт з використанням персонального комп'ютера і без нього[18].

При виникненні у працюючих з ПК зорового дискомфорту та інших несприятливих суб'єктивних відчуттів, незважаючи на дотримання санітарно-гігієнічних і ергономічних вимог, рекомендується застосовувати індивідуальний підхід з обмеженням часу роботи з ПК.

У випадках, коли характер роботи вимагає постійної взаємодії з ВДТ (набір текстів або введення даних тощо) з напругою уваги та зосередженості, при виключенні можливості періодичного перемикання на інші види трудової діяльності, не пов'язані з ПК, рекомендується організація перерв на 10 -15 хвилин через кожні 45-60 хвилин роботи. Тривалість безперервної роботи з ВДТ без регламентованого перерви не повинна перевищувати однієї години[19].

При роботі з ПК в нічну зміну (з 22 до 6 год.) незалежно від категорії і виду трудової діяльності тривалість регламентованих перерв слід збільшувати на 30%.

Під час регламентованих перерв з метою зниження нервово-емоційного напруження, стомлення зорового аналізатора, усунення впливу гіподинамії та гіпокінезії, запобігання розвитку позотонічної (статичного) втоми доцільно виконувати спеціально розроблені комплекси вправ.

Користувачам ПК, виконують роботу з високим рівнем напруженості, показана психологічне розвантаження під час регламентованих перерв і в кінці робочого дня в спеціально обладнаних приміщеннях (кімната психологічного розвантаження).

Під час розробки, тестування та впровадження автоматизованої системи для розпізнавання тексту та визначення зображень були дотримані всі вимоги, норми та державні стандарти з охорони праці.

4.2 Електробезпека користувачів ПК

Умови й організацію праці при роботі з візуальними дисплейними терміналами усіх типів вітчизняного та зарубіжного виробництва на основі електронно-променевих трубок, що використовуються в електронно-обчислювальних машинах (ЕОМ*) колективного використання та персональних, визначають Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин (ДСанПІН 3.3.2.007-98), затверджені постановою головного держаного санітарного лікаря України від 10 грудня 1998 р. № 7.

Мінімальні вимоги безпеки та захисту здоров'я під час роботи, пов'язаної з використанням екранних пристроїв незалежно від їхнього типу та моделі, визначають Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями, затверджені наказом Міністерства соціальної політики України від 14 лютого 2018 р. № 207 (далі — НПАОП 0.00-7.15-18).

Виробники забезпечують супроводження електрообладнання інструкціями та інформацією про безпечність, складеними згідно з вимогами закону щодо порядку застосування мов. Зазначені інструкції та інформація, а також будь-яке маркування повинні бути чіткими, зрозумілими та дохідливими (п. 13 Технічного регламенту низьковольтного електричного обладнання, затвердженого постановою Кабінету Міністрів України від 16 грудня 2015 р. № 1067).

Порядок допуску до роботи з комп'ютерною технікою визначає Типове положення про порядок проведення навчання і перевірки знань з питань охорони праці (НПАОП 0.00-4.12-05), затверджене наказом Державного комітету України з нагляду за охороною праці від 26 січня 2005 р. № 15, відповідно до п. 3.1 та 3.4 якого передбачене проведення інструктажів та навчання з питань охорони праці, зокрема й електробезпеки.

Порядок присвоєння кваліфікаційної групи з електробезпеки на виробництві визначає НПАОП 40.1-1.21-98 Правила безпечної експлуатації електроустановок споживачів, затверджений наказом Державного комітету України по нагляду за охороною праці від 9 січня 1998 р. № 4, що поширюється на працівників, які обслуговують діючі електроустановки споживачів. Під обслуговуванням (технічним) розуміють комплекс робіт з підтримки працездатності обладнання в період його використання, до якого належать роботи з випробування обладнання, пристроїв, огляд обладнання, підтяжка контактних з'єднань (п. 3.1 розділу III Правил технічної експлуатації електроустановок споживачів, затверджених наказом Міністерства палива та енергетики України від 25 липня 2006 р. № 258 (далі — ПТЕЕС).

Під електроустановкою розуміють комплекс взаємопов'язаних машин, апаратів, ліній та допоміжного обладнання (разом з будівлями і приміщеннями, в яких їх встановлено), призначених для виробництва, трансформації, передавання, розподілу електричної енергії і перетворення її в інші види енергії (п. 1.1.3 Правил улаштування електроустановок, затверджених наказом Міністерства енергетики та вугільної промисловості України від 21 липня 2017 р. № 476 (далі — ПУЕ).

Роботодавець повинен забезпечити навчання і перевірку знань з питань охорони праці та безпечного використання екранних пристроїв до початку роботи з ними та проведення медичних оглядів працівників (п. 2, 6 розділу II НПАОП 0.00-7.15-18).

Сучасний комп'ютер не є електроустановкою, і вимоги ПУЕ та ПТЕЕС можуть бути правомірні тільки для мережі його електроживлення, тобто на саму комп'ютерну техніку не поширюються. Вимоги безпеки електрообладнання комп'ютерної техніки регламентують державні стандарти, зокрема, серії ДСТУ EN 60335 та ДСТУ EN 60950.

Технічні засоби загального (побутового) призначення не повинні використовуватися в умовах підвищеної небезпеки, тож експлуатація сучасної комп’ютерної техніки не належить до робіт підвищеної небезпеки.

Приміщення із робочими місцями користувачів комп’ютерів для забезпечення електробезпеки обладнання, а також для захисту від ураження електричним струмом самих користувачів ПК повинні мати достатні технічні засоби захисту відповідно до ГОСТ 12.1.009-76, НПАОП 40.1-1.07-01 “Правила експлуатації електрозахисних засобів”, НПАОП 40.1-1.21-98 “Правила безпечної експлуатації електроустановок споживачів”, НПАОП 40.1-1.32-01 “Правила будови електроустановок. Електрообладнання спеціальних установок”

З метою запобігання ушкодженням, що можуть статися через ураження електричним струмом, загоряння, коротке замикання тощо, розроблено загальний стандарт безпеки IEC 950. Загальним стандартом електробезпечності для країн Європейської співдружності є Cemark. Під час проектування систем електропостачання, монтажу силового електрообладнання та електричного освітлення будівель та приміщень для ПЕОМ необхідно дотримуватись вимог вищеназваних нормативно-правових актів, а також СН 357-77 "Инструкция по проектированию силового осветительного оборудования промышленных предприятий", затверджених Держбудом СРСР, ГОСТу 12.1.006, ГОСТу 12.1.030 "ССБТ. Электробезопасность. Защитное заземление, зануление", ГОСТу 12.1.019 "ССБТ. Электробезопасность. Общие требования и номенклатура видов защиты", ГОСТу 12.1.045, ВСН 59-88 Держкомархітектури СРСР "Электрооборудование жилых и общественных зданий. Нормы проектирования", Правил пожежної безпеки в Україні, ДСанПіН 3.3.2.007-98, розділів СНиП, що стосуються штучного освітлення і електротехнічних пристроїв, та вимог нормативно-технічної і експлуатаційної документації заводу-виробника ПЕОМ.

ЕОМ, периферійні пристрої ЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ, інше устаткування (апарати управління, контрольно-вимірювальні прилади, світильники тощо), електропроводи та кабелі за виконанням та ступенем захисту мають відповідати класу зони за ПУЕ, мати апаратуру захисту від струму короткого замикання та інших аварійних режимів.

Під час монтажу та експлуатації ліній електромережі необхідно повністю унеможливити виникнення електричного джерела загоряння внаслідок короткого замикання та перевантаження проводів, обмежувати застосування проводів з легкозаймистою ізоляцією і, за можливості, перейти на негорючу ізоляцію.

Лінія електромережі для живлення ЕОМ, периферійних пристроїв ЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ виконується як окрема групова трипровідна мережа, шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів.

Використання нульового робочого провідника як нульового захисного провідника забороняється. Нульовий захисний провід прокладається від стійки групового розподільчого щита, розподільчого пункту до розеток живлення. Не допускається підключення на щиті до одного контактного затискача нульового робочого та нульового захисного провідників. Площа перерізу нульового робочого та нульового захисного провідника в груповій трипровідній мережі повинна бути не менше площі перерізу фазового провідника.

Усі провідники повинні відповідати номінальним параметрам мережі та навантаження, умовам навколишнього середовища, умовам розподілу провідників, температурному режиму та типам апаратури захисту, вимогам ПУЕ.

У приміщенні, де одночасно експлуатується або обслуговується більше п'яти персональних ЕОМ, на помітному та доступному місці встановлюється аварійний резервний вимикач, який може повністю вимкнути електричне живлення приміщення, крім освітлення.

ЕОМ, периферійні пристрої ЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ повинні підключатися до електромережі тільки з допомогою справних штепсельних з'єднань і електророзеток заводського виготовлення. Штепсельні з'єднання та електророзетки крім контактів фазового та нульового робочого провідників повинні мати спеціальні контакти для підключення нульового захисного провідника. Конструкція їх має бути такою, щоб приєднання нульового захисного провідника відбувалося раніше ніж приєднання фазового та нульового робочого провідників. Порядок роз'єднання при відключенні має бути зворотним. Необхідно унеможливити з'єднання контактів фазових провідників з контактами нульового захисного провідника.

Неприпустимим є підключення ЕОМ та периферійних пристроїв ЕОМ до звичайної двопровідної електромережі, в тому числі – з використанням перехідних пристроїв.

Електромережі штепсельних з'єднань та електророзеток для живлення ЕОМ, периферійних пристроїв слід виконувати за магістральною схемою, по 3…6 з’єднань або електророзеток в одному колі. Штепсельні з'єднання та електророзетки для напруги 12 В та 36 В за своєю конструкцією повинні відрізнятися від штепсельних з'єднань для напруги 127 В та 220 В і мають бути пофарбовані в колір, який візуально значно відрізняється від кольору штепсельних з'єднань, розрахованих на напругу 127 В та 220 В.

Індивідуальні та групові штепсельні з'єднання та електророзетки необхідно монтувати на негорючих або важкогорючих пластинах з урахуванням вимог ПУЕ та Правил пожежної безпеки в Україні.

Електромережу штепсельних розеток для живлення ЕОМ, периферійних пристроїв ЕОМ при розташуванні їх уздовж стін приміщення

прокладають по підлозі поряд зі стінами приміщення, як правило, в металевих трубах і гнучких металевих рукавах з відводами відповідно до затвердженого плану розміщення обладнання та технічних характеристик обладнання.

При розташуванні в приміщенні за його периметром до 5 ЕОМ, використанні трипровідникового захищеного проводу або кабелю в оболонці з негорючого або важкогорючого матеріалу дозволяється прокладання їх без металевих труб та гнучких металевих рукавів.

Електромережу штепсельних розеток для живлення ЕОМ при розташуванні їх у центрі приміщення, прокладають у каналах або під знімною підлогою в металевих трубах або гнучких металевих рукавах. При цьому не дозволяється застосовувати провід і кабель в ізоляції з вулканізованої гуми та інші матеріали, що містять сірку. Відкрита прокладка кабелів під підлогою забороняється. Металеві труби та гнучкі металеві рукави повинні бути заземлені. Заземлення повинно відповідати вимогам НПАОП 40.1-1.21-98.

Для підключення переносної електроапаратури застосовують гнучкі проводи в надійній ізоляції.

Тимчасова електропроводка від переносних приладів до джерел живлення виконується найкоротшим шляхом без заплутування проводів у конструкціях машин, приладів та меблях. Доточувати проводи можна тільки шляхом паяння з наступним старанним ізолюванням місць з'єднання.

Є неприпустимими:

– експлуатація кабелів та проводів з пошкодженою або такою, що втратила захисні властивості за час експлуатації, ізоляцією; залишення під напругою кабелів та проводів з неізольованими провідниками;

– застосування саморобних подовжувачів, які не відповідають вимогам ПВЕ до переносних електропроводок;

– застосування для опалення приміщення нестандартного (саморобного) електронагрівального обладнання або ламп розжарювання;

ВИСНОВКИ

В результаті виконання дипломної роботи було спроектовано та реалізовано програмну систему для визначення предметів та розпізнавання предметів з використанням машинного навчання та підтримкою технології Voice Over на мові програмуванням Swift, з використанням IDE Xcode для мобільної платформи iOS версії 12.0 і вище.

На етапі проектування програмної системи було проаналізовано предметну область, проведено постановку задачі, визначено основні варіанти використання, проведено опис ключових варіантів використання, а саме розпізнавання тексту та предметів, озвучення отриманого результату, збереження отриманого результату у базу даних, перегляд збережених результатів. В якості моделі розробки було визначено – каскадну модель. Було визначено MVC, як архітектурний шаблон, побудовано на основі попередніх етам діаграму класів системи.

На етапі реалізації системи було проведено вибір мови програмування для реалізації програмної системи. Було обрано мову програмування Swift. В якості інструменту для збереження даних було обрано систему управління базами даних Core Data. Наступний кроком була власне реалізація системи та проведено тестування системи. В процесі тестування не було виявлено жодних помилок, а отже система реалізована, як належне. У пояснювальній записці було наведено основні класи та методи реалізованої системи, а також демонстрація процесу використання та тестування системи.

Доцільність розробки опирається на обґрунтуванні техніко-економічних показників. Так було розраховано витрати на розробку та впровадження, підтримку проекту. З точки зору організації процесу реалізації у вигляді програмного продукту було обґрунтовано, що об’єктно-

орієнтований підхід більш прийнятний для використання в такого типу проектах.

Задля дотримання державних стандартів та норм з Охорони праці у галузі розробки програмного забезпечення, було проаналізовано нормативно-правові акти, правила та норми праці при роботі з персональними комп'ютерами. Було створено та забезпечено всі необхідні умови освітлення приміщень при роботі з ВДТ для розробки та тестування програмного забезпечення.

Даний дипломний проект та його реалізація була виконана з метою оптимізувати процес перенесення тексту в електронну форму та допомогти користувачам з слабким зором в розпізнаванні тексту та предметів завдяки озвучуванню отриманого результату.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. AI (artificial intelligence) - TechTarget [Електронний ресурс] - Режим доступу: URL: https://searchenterpriseai.techtarget.com/definition/AI-Artificial-Intelligence

2. What is machine learning? - MathWorks [Електронний ресурс] - Режим доступу: URL: https://www.mathworks.com/discovery/machine-learning.html

3. Deep Learning Based OCR for Text in the Wild) - NanoNets [Електронний ресурс] - Режим доступу: URL: https://nanonets.com/blog/deep-learning-ocr/

4. Text Recognition - Medium [Електронний ресурс] - Режим доступу: URL: https://medium.com/@eltronicsvilla17/text-recognition-7286606e57ba

5. A Gentle Introduction to Object Recognition With Deep Learning - MachineLearningMastery [Електронний ресурс] - Режим доступу: URL: https://machinelearningmastery.com/object-recognition-with-deep-learning/

6. Iterative development - TechTarget [Електронний ресурс] - Режим доступу:URL:https://searchsoftwarequality.techtarget.com/definition/iterative-development

7. Waterfall model - TechTarget [Електронний ресурс] - Режим доступу: URL: https://searchsoftwarequality.techtarget.com/definition/waterfall-model

8. Model-View-Controller – Apple [Електронний ресурс] - Режим доступу: URL:developer.apple.com/library/archive/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html

9. IOS Design Patterns: MVC and MVVM – CaptechConsulting [Електронний ресурс] - Режим доступу: URL: https://www.captechconsulting.com/blogs/ios-design-patterns-mvc-and-mvvm

10. Swift vs Objective C in 2019 – Medium [Електронний ресурс] - Режим доступу: URL: https://medium.com/swiftify/swift-vs-objective-c-comparison-32aba9dad4e3

11. Core Data – Apple [Електронний ресурс] - Режим доступу: URL: https://developer.apple.com/documentation/coredata

12. User Defaults – Apple [Електронний ресурс] - Режим доступу: URL: https://developer.apple.com/documentation/foundation/userdefaults

13. Закон України «Про збір та облік єдиного внеску на загальнообов'язкове державне соціальне страхування» №2464-VI від 08.07.2010

14. Методичні вказівки для виконання розділу дипломної роботи щодо техніко-економічного обґрунтування вибору проектного рішення розробки та оцінки якості програмного забезпечення / Упор. Петрик М.Р., Кінах Я.І., Головатий А.І., Рогатинська Л.Р. – Тернопіль: Вид-во ТНТУ ім. І. Пулюя. – 2013. – 34 с

15. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин [Електронний ресурс] - Режим доступу: URL: https://zakon.rada.gov.ua/rada/show/v0007282-98

16. Наказ 14.02.2018 № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями»

[Електронний ресурс] - Режим доступу: URL: https://zakon.rada.gov.ua/laws/show/z0508-18#n14

17. Охорона праці в галузі. Конспект лекцій - ELARTU [Електронний ресурс] -Режим доступу: URL http://elartu.tntu.edu.ua/bitstream/123456789/17891/1/KonspektOPG.pdf

18. Психологічне забезпечення складних систем діяльності [Електронний ресурс] - Режим доступу: URL http://maup.com.ua/assets/files/lib/book/psihol_zabezp_skl.pdf

19. Формування поняття про професійні захворювання користувачів ПК в процесі підготовки фахівців С.В. Дембіцька, М.С. Фурман. 2018.

ДОДАТКИ

Міністерство освіти і науки України

Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії

Кафедра програмної інженерії

ЗАТВЕРДЖУЮ

Завідувач кафедрою

програмної інженерії

"___ "_____ 2019 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської дипломної роботи

на тему: «Розробка програмної системи для визначення предметів та розпізнавання зображень з використанням машинного навчання та підтримкою технології Voice Over»

Біломазуру Костянтину Володимировичу

Керівник роботи:

д.ф.-м.н., професор Петрик М. Р.

"___" _____ 2019р.

_____

Виконавець:

студент групи СПм-61

Біломазур Костянтин Володимирович

"___" _____ 2019р.

_____

м. Тернопіль – 2019

ЗМІСТ

Вступ

1. Підстави до розробки
2. Призначення до розробки
3. Вимоги до програмного продукту
   - 3.1 Функціональні характеристики
   - 3.2 Склад та параметри технічних засобів
   - 3.3 Інформаційна та програмна сполучність
4. Стадії розробки
5. Програмна документація
6. Порядок контролю та приймання

1 ПІДСТАВИ ДО РОЗРОБКИ

Розробка проводиться у відповідності до графіку навчального плану на 2019 рік, та згідно наказу на виконання дипломної роботи студента-магістра.

Тема проекту: «Розробка програмної системи для визначення предметів та розпізнавання зображень з використанням машинного навчання та підтримкою технології Voice Over».

2 ПРИЗНАЧЕННЯ РОЗРОБКИ

Машинне навчання дозволяє суттєво оптимізувати будь-який процес людської діяльності. В даному випадку він дозволяє розпізнати необхідний рукописний чи друкований текст та перенести його у електронний формат, що дозволить користувачеві зекономити час на переписування його вручну. Озвучування результату з використанням технології Voice Over, дозволить допомогти користувачам із поганим зором, а також усім користувачам прослухати результат розпізнавання не контактуючи із дисплеєм телефону.

Метою дипломної роботи є дослідження та розробка системи для розпізнавання тексту та предметів з використанням машинного навчання. У процесі виконання дипломної роботи буде проведено аналіз предметної області, визначення основних варіантів використання, проектування та реалізація системи. Також потрібно провести тестування отриманого програмного продукту на наявність помилок та відмінностей від поставленого завдання.

За результатами виконаної роботи потрібно реалізувати програмну систему для розпізнавання тексту та визначення предметів з використанням машинного навчання та підтримкою технології Voice Over.

3 ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

3.1 Функціональні характеристики

Програмне забезпечення має виконувати наступні дії:

- розпізнавати предмет використовуючи фото зроблене камерою чи зображення обране користувачем із галереї;
- розпізнавати текст використовуючи фото зроблене камерою чи зображення обране користувачем із галереї;
- озвучити отриманий результат;
- зберігати отриманий предмет чи текст у результаті розпізнавання до бази даних;
- видаляти предмет чи текст із бази даних;
- змінювати предмет чи текст, який знаходиться у базі даних:
- сортувати предмети чи тексти, які знаходяться у базі даних;
- показувати користувачу усі наявні предмети чи тексти, які знаходяться у базі даних

3.2 Склад та параметри технічних засобів

1) ПК або планшетний комп’ютер з 4024 Мб оперативної пам‘яті, встановленою системою MacOS. Не менше 1024 Мб вільного місця на жорсткому диску. Двоядерний процесор з тактовою частотою від 2.2 GHz і більше.

2) наявність встановленого Xcode version 10.2 і вище.

3.3 Інформаційна та програмна сполучність

Програмний продукт повинен коректно функціонувати на пристоях з операційною системою iOS version 12.0 і вище. Програмний інтерфейс застосунку повинен бути простим та інтуїтивно зрозумілим для користувача. Вигляд програмного додатку повинен відповідати усім сучасним тенденціям.

4. СТАДІЇ РОЗРОБКИ

В ходів реалізації роботи проект повинен пройти крізь наступні стадії розробки:

- аналіз предметної області;
- постановка задачі;
- пошук акторів та варіантів використання;
- опис ключових варіантів використання;
- проектування програмної системи;
- реалізація програмної системи;
- тестування програмної систмеми
- оформлення супровідної документації;
- здача роботи.

5. ПРОГРАМНА ДОКУМЕНТАЦІЯ

Для програмного продукту повинні бути розроблені наступні документи:

- Пояснювальна записка;
- Технічне завдання;
- Презентаційний матеріал;
- Додатки.

6. ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Розроблений програмний продукт має виконувати всі вимоги, що складаються з перерахованих у п. 3.1 характеристик.

Приймання проводиться спеціально створеною екзаменаційною комісією в термін до:

“__” грудня 2019р.

УДК 004.415.5
К. Біломазур ст. гр. СПм-61
Тернопільський національний технічний університет імені Івана Пулюя

Причини використання та підтримки технології VoiceOver в мобільному застосунку

У сучасному світі є чимало людей з обмеженими можливостями певного виду. Одною з таких проблем є – поганий зір. Ця вада має надзвичайно сильний вплив на людське життя, адже за допомогою зору людина отримує 80-90% інформації про навколишній світ. За даними ВООЗ, проблеми з зором мають близько 300 мільйонів людей у світі. З них 43% страждають порушеннями рефракції - короткозорістю (міопією), далекозорістю (гіперметропією) та астигматизмом [1].

Для того, щоб допомогти людям використовувати сучасні технології та застосунки Apple розробили технологію VoiceOver.

VoiceOver — це функція (технологія), вбудована в операційну систему Mac OS X від Apple Inc. Використовуючи VoiceOver, користувач може керувати своїм комп'ютером завдяки мові та клавіатурі. Ця функція була розроблена для того, щоб покращити керування комп'ютером користувачем з поганим зором. VoiceOver — вбудована програма читання екрану, яка озвучує інформацію на екрані комп'ютера: вимовляючи текст, що міститься в документах та вікнах [2].

VoiceOver на iOS взаємодіє з користувачем, використовуючи різні "жести", різні рухи, які ви робите одним або кількома пальцями на дисплеї. Багато жестів залежать від місця розташування - наприклад, ковзання пальцем по екрану розкриє візуальний вміст екрана, коли палець проходить по них. Це дає можливість сліпим користувачам досліджувати фактичний контент програми на екрані. Користувач може двічі торкнутись (подібно до подвійного клацання мишкою), щоб активувати вибраний елемент так само, як якщо б проникливий користувач торкнувся його.

VoiceOver також може вимкнути дисплей, але залишати сенсорний екран чутливим до дотику, економлячи енергію акумулятора. Apple називає цю функцію "Екранна завіса". Він також доступний на комп'ютерах Mac під управлінням ОС X.

Для того, щоб додати підтримку для технології VoiceOver у власному iOS застосунку необхідно у налаштуваннях кожного візуального компонента увімкнути властивість "Доступність". З використанням фреймворку "AVFoundation" можна розширити можливості технології VoiceOver та додати озвучення в процесі роботи застосунку, ставити оголошення на паузу та проводити інші маніпуляції.

Підсумовуючи весь матеріал можна дійти висновку, що поганий зір та сліпота є однією з актуальних проблем сучасного світу. Для того, щоб дозволити людям надалі користуватися сучасними додатками, не зважаючи на це обмеження, потрібно додавати в розроблений застосунок підтримку технології VoiceOver. Данна технологія дозволить, хоч і не в повній мірі, користуватися вашим програмним забезпеченням.

Список використаної літератури:

1. GLOBAL DATA ON VISUAL IMPAIRMENTS 2010 [Електронний ресурс]. – Режим доступу: https://who.int/blindness/GLOBALDATAFINALforweb.pdf/ – Назва з екрану.
2. Accessibility. Vision [Електронний ресурс]. – Режим доступу: https://www.apple.com/accessibility/mac/vision/ – Назва з екрану.