НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

«На правах рукопису»
УДК 004.42

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
«___»_____2024 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Інженерія програмного забезпечення інформаційних систем»

зі спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Програмне забезпечення для налаштування та виконання голосових команд в операційній системі Windows»

Виконав:
студент ІІ курсу, групи ІП-22мп
Соболевський Владислав Олександрович _____

Керівник:
доцент кафедри ІПІ, к.т.н., доц.,
Баклан Ігор Всеволодович _____

Рецензент:
доцент кафедри ІСТ, к.т.н., доц.,
Голубєв Леонтій Петрович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2024 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – другий (магістерський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення інформаційних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Едуард ЖАРІКОВ

«___»_____2023р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Соболевському Владиславу Олександровичу

1. Тема дисертації «Програмне забезпечення для налаштування та виконання голосових команд в операційній системі Windows», науковий керівник дисертації Баклан Ігор Всеволодович, доцент кафедри ІПІ, к.т.н. затверджені наказом по університету від «11» листопада 2023 р. № 5168-с

2. Термін подання студентом дисертації «16» січня 2024 р.

3. Об’єкт дослідження – програмні рішення для налаштування та виконнання голосових команд в операційній системі Windows

4. Предмет дослідження – методи, моделі, інструменти для представлення і перетворення знань щодо програмного забезпечення для налаштування та виконнання голосових команд в операційній системі Windows

5. Перелік завдань, які потрібно розробити – аналіз проблеми та існуючих рішень; визначення архітектурного дизайну; Реалізація та прототипування програмного рішення; тестування запропонованого програмного рішення

6. Орієнтовний перелік графічного (ілюстративного) матеріалу – 4 плакати

7. Орієнтовний перелік публікацій – одна публікація

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9.Дата видачі завдання «1» листопада 2022 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання	Примітка
1	Огляд літератури	1.11.2022	
2	Аналіз проблеми та існуючих рішень	11.05.2023	
3	Визначення архітектурного дизайну	16.07.2023	
4	Реалізація та прототипування програмного рішення	12.09.2023	
5	Тестування запропонованого програмного рішення	20.11.2023	
6	Розробка стартап-проєкту	16.12.2023	
7	Оформлення пояснювальної записки	17.12.2023	
8	Подання дисертації на попередній захист	19.12.2023	
9	Подання дисертації на захист	16.01.2024	

Студент Владислав СОБОЛЕВСЬКИЙ

Науковий керівник Ігор БАКЛАН

РЕФЕРАТ

Розмір пояснювальної записки – 131 аркушів, містить 10 ілюстрацій, 66 таблиць, 4 додатки, 16 посилання на джерела.

Актуальність теми. Використання голосового управління операційною системою значно розширює можливості користувачів до зручного використання персональних комп’ютерів, але потребує подальших досліджень для підвищення ефективності підходів їх реалізації. У роботі розглянуто проблему необхідності розробки та вдосконалення програмного забезпечення з метою підвищення варіативності його використання.

Мета дослідження. Основною метою є є підвищення варіативності використання програмного забезпечення для виконнання голосових команд в операційній системі Windows з можливістю гнучкого налаштування користувацьких команд.

Об’єкт дослідження: програмні рішення для налаштування та виконнання голосових команд в операційній системі Windows.

Предмет дослідження: методи, моделі, інструменти для представлення і перетворення знань щодо програмного забезпечення для налаштування та виконнання голосових команд в операційній системі Windows.

Для реалізації поставленої мети **сформульовані наступні завдання**:

- аналіз існуючих рішень;
- визначення недоліків аналогів та можливостей для покращення;
- визначення архітектурного дизайну;
- реалізація та прототипування програмного рішення;
- тестування запропонованого програмного рішення.

Наукова новизна: результатом роботи є вдосконалення існуючих підходів до створення голосових асистентів та розробка відповідного програмного

забезпечення, що надасть можливість гнучкого налаштування голосових команд та може бути використано на практиці.

Практичним значенням цієї роботи **є** програмне забезпечення для налаштування та виконнання голосових команд в операційній системі Windows на основі розпізнавання мовлення, яке може бути використане пересічними користувачами операційних систем для зручної експлуатації під час роботи з персональним комп’ютером.

Зв’язок з науковими програмами, планами, темами. Робота виконувалась на кафедрі інформатики та програмної інженерії Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського".

Апробація. Наукові положення дисертації пройшли апробацію на V науково-практичній конференції «SoftTech-2023» та опубліковані у матеріалах конференції.

Ключові слова: ОПЕРАЦІЙНА СИСТЕМА, ГОЛОСОВЕ КЕРУВАННЯ, ПЛАТФОРМА, APPLE SIRI, MICROSOFT CORTANA, AMAZON ALEXA, SAMSUNG BIXBY, .NET, C#, PYTHON, NUMPY, TENSORFLOW, WPF, РОЗПІЗНАВАННЯ МОВИ, КОРИСТУВАЧ, ГОЛОСОВИЙ ПОМІЧНИК, RIDER, PYCHARM, АРХІТЕКТУРА, НЕЙРОННА МЕРЕЖА.

ABSTRACT

Explanatory note size – 131 pages, contains 10 illustrations, 66 tables, 4 applications, 16 references.

Topicality. The use of voice control in operating systems significantly expands the capabilities of users for convenient use of personal computers but requires further research to improve the effectiveness of their implementation approaches. This work examines the problem of developing and improving software to increase its usage variability.

The aim of the study. The main goal is to enhance the variability of software usage for executing voice commands in the Windows operating system, with the possibility of flexible customization of user commands.

The object of research: software solutions for setting up and executing voice commands in the Windows operating system.

The subject of research: methods, models, and tools for representing and transforming knowledge about software for setting up and executing voice commands in the Windows operating system.

To achieve this goal, the **following tasks** were formulated:

- analysis of existing solutions;
- identification of shortcomings in analogs and opportunities for improvement;
- definition of architectural design;
- implementation and prototyping of the software solution;
- testing of the proposed software solution.

The scientific novelty the result of the work is the improvement of existing approaches to creating voice assistants and the development of corresponding software, which will provide the possibility of flexible configuration of voice commands and can be used in practice.

The practical value of this work is the software for setting up and executing voice commands in the Windows operating system based on speech recognition, which can be used by average operating system users for convenient operation during work with a personal computer.

Relationship with working with scientific programs, plans, topics. Work was performed at the Department of Informatics and Software Engineering of the National Technical University of Ukraine «Kyiv Polytechnic Institute. Igor Sikorsky».

Approbation. The scientific provisions of the dissertation were tested at the fifth Scientific and Practical Conference "SoftTech-2023" and published in the conference proceedings.

Keywords: OPERATING SYSTEM, VOICE CONTROL, PLATFORM, APPLE SIRI, MICROSOFT CORTANA, AMAZON ALEXA, SAMSUNG BIXBY, .NET, C#, PYTHON, NUMPY, TENSORFLOW, WPF, SPEECH RECOGNITION, USER, VOICE ASSISTANT, RIDER, PYCHARM, ARCHITECTURE, NEURAL NETWORK.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ОС – операційна система

ПЗ – програмне забезпечення

IDE - integrated development environment. Комплексне програмне рішення для розробки програмного забезпечення.

JSON - JavaScript Object Notation, нотація об’єктів JavaScript

NuGet – менеджер пакетів для платформи .Net

BRNN – Bidirectional recurrent neural networks, бідирекційна рекурентна нейронна мережа

ASR – Automatic Speech Recognition, автоматичне розпізнавання мовлення

CTC - Connectionist temporal classification

WPF - Windows Presentation Foundation

ВСТУП

У сучасному світі інформаційних технологій, створення ефективного та зручного програмного забезпечення стає дедалі важливішою задачею. Одним з перспективних напрямків розробки є голосове управління операційною системою, яке дозволяє користувачам взаємодіяти з пристроями на кшталт комп’ютер, смартфон та іншими за допомогою голосових команд. Моя магістерська робота має на меті розробку саме такого програмного забезпечення, що надасть користувачам можливість керувати операційною системою за допомогою голосових команд та інструменти для їх гнучкого налаштування.

Основною задачею цієї роботи є реалізація десктопного програмного забезпечення, що допоможе користувачам підвищити зручність експлуатації їх пристроїв. Це програмне забезпечення відкриє можливість користувачам за допомогою голосових команд викликати команди операційної системи, надасть можливість запуску інших програм та дозволить створювати персоналізовані голосові інструкції, що дозволить надзвичайно легко створити набір команд для дій, що користувач вважає найважливішими або ж часто використовує.

Актуальність голосового управління операційною системою полягає в його потенціалі покращити ефективність та зручність взаємодії користувача з пристроями. Голосове керування дає змогу користувачам виконувати завдання без використання клавіатури або миші. Це особливо важливо для людей з обмеженими можливостями або фізичними обмеженнями. Голосове керування може поліпшити їхній доступ до технологій і забезпечити рівність у використанні комп'ютерних систем. Воно може значно збільшити швидкість та продуктивність користувача, дозволяючи виконувати завдання швидше та безпосередньо через голосові команди. Голосове керування дозволяє швидко виконувати операції, такі як запуск програм, переходи між вкладками або виконання пошуку інформації без необхідності ручного вводу. Розвиток голосових технологій,

таких як глибоке навчання і штучний інтелект, створює нові можливості для поліпшення голосового управління. Спікери стають все точнішими та надійнішими, розпізнавання мови стає більш натуральним, а функції інтеграції з іншими додатками та сервісами розширюються.

З огляду на ці фактори, голосове керування операційними системами є актуальним напрямком розвитку програмного забезпечення, що відкриває нові можливості для зручної, ефективної та економної взаємодії з технічними системами.

Метою дипломного проєкту є розширення можливостей керування операційною системою за допомогою голосового асистента.

Дослідження що виконуються:

- аналіз інсуючих рішень та виявлення їх проблем;
- дослідження аналізаторів мови;
- розробка та реалізація методів та алгоритмів для вирішення виявлених проблем;
- розробка програмного забезпечення для користувачів, що бажають покращити свій досвід взаємодії з операційною системою за допомогою керуванням голосовими командами.

Об’єктом дипломного проєкту є програмне забезпечення для голосового керування операційною системою.

Предметом дипломного проєкту є методи та програмне забезпечення для керування операційною системою та модель штучного інтелекту для аналізу мови.

Практична робота дипломного проєкту – програмне забезпечення, що надає інтерфейс для голосового керування операційною системою та налаштування персоналізованих інструкцій.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

У ході дипломного проєкту розроблено застосунок для налаштування та виконнання голосових команд в операційній системі Windows у вигляді десктопного додатку. Саме така форма вибрана як найбільш оптимальна, оскільки користувач зможе взаємодіяти з застосунком без використання мережі Інтернет. Також, це дозволяє користувачу зручно отримувати оновлення версії програмного застосунку.

Головними перевагами виклику команд за допомогою голосу є:

- вивільнення часу, що витрачається мануальні маніпуляції з операційною системою за допомогою пристоїв введення таких як миша та клавіатура;
- можливість паралельно виконувати дії з операційною системою, як мануально, так і за допомогою голосових команд;
- спрощення досвіду використання персональних комп’ютерів з операційною системою Windows для людей з обмеженеми можливостями.

Переважна більшість подібних програмних рішень надає лише обмежений перелік функціоналу, а також, зазвичай, їх працездатність напряму залежить від наявності підключення до мережі Інтернет, що заважає користувачам у надзвичайних ситуаціях.

Тому дипломний проєкт пропонує рішення цієї проблеми.

1.2 Змістовний опис і аналіз предметної області

Голосові асистенти, яких часто називають віртуальними асистентами або цифровими помічниками, - це програмні агенти, які можуть виконувати завдання

або послуги на основі голосових команд. Вони використовують обробку природної мови (NLP), щоб розуміти і відповідати на запити користувача. Ось короткий огляд поширених застосувань голосових помічників:

- пошук інформації: користувачі можуть ставити запитання, а голосовий асистент може надавати відповіді, здійснюючи пошук в Інтернеті або отримуючи доступ до збереженої інформації. Наприклад, "Яка сьогодні погода?" або "Хто виграв гру вчора ввечері?";
- керування завданнями: встановлюйте нагадування, створюйте списки справ або керуйте подіями календаря. Наприклад, "Нагадай мені купити молока о 17:00" або "Заплануй зустріч з Сарою на наступний вівторок о 15:00";
- спілкування: надсилайте текстові повідомлення, телефонуйте або навіть надсилайте електронні листи, просто використовуючи свій голос. "Напиши Джону: "Буду через 10 хвилин" або "Подзвони мамі";
- відтворення мультимедіа: відтворення музики, подкастів або відео. Поширені команди включають "Відтворити останній епізод [назва подкасту]" або "Відтворити мій плейлист для тренування";
- керування розумним будинком: керування розумними пристроями у вашому домі, такими як світло, термостати, замки тощо. "Вимкни світло у вітальні" або "Встанови термостат на 22 градуси";
- навігація та подорожі: прокладайте маршрути, перевіряйте умови дорожнього руху або знаходьте найближчі об'єкти. "Знайди найближчу заправку" або "Які затори на дорогах до аеропорту?";
- покупки: деякі голосові помічники дозволяють користувачам замовляти товари, відстежувати посилки або навіть бронювати місця. "Замов більше прального порошку" або "Забронюй столик на двох у [назва ресторану]";
- розваги: деякі голосові помічники можуть грати в ігри, жартувати або

зачитувати історії, щоб розважити користувачів;

- інтеграція зі сторонніми додатками: багато голосових помічників можна інтегрувати зі сторонніми додатками, що дозволяє користувачам робити такі речі, як бронювання поїздки за допомогою сервісу rideshare, замовлення доставки їжі або управління фінансами за допомогою голосових команд;
- доступність: для людей з обмеженими можливостями або тих, кому традиційні інтерфейси можуть бути складними, голосові помічники можуть запропонувати альтернативний спосіб взаємодії з технологіями та доступу до інформації;
- освіта: їх можна використовувати для навчання, відповідей на запитання, допомоги у виконанні домашніх завдань або вивчення мов;
- рутинна настройка: деякі просунуті голосові помічники дозволяють користувачам налаштовувати рутини. Наприклад, програма "Доброго ранку" може увімкнути світло, повідомити прогноз погоди, а потім відтворити новини.

Серед відомих голосових помічників - Siri від Apple, Alexa від Amazon, Google Assistant, Cortana від Microsoft і Bixby від Samsung. Кожен з них має власний набір функцій, можливостей та екосистему підтримуваних пристроїв і сервісів. Очікується, що з розвитком технологій можливості голосових помічників розширюватимуться, роблячи їх ще більш інтегрованими в наше повсякденне життя.

Оскільки кінцевий варіант програмного забезпечення націлене на те, щоб задовольнити потреби користувача, важливою частиною роботи є дослідження та аналіз конкурентів та потреб кінцевого споживача. Кожного дня сотні мільйонів людей користуються гаджетами, а отже взаємодіють з операційною системою. Запити користувачів до підвищення зручності використання пристроїв ростуть прямо пропорційно розвитку та прогресу як IT-сфери, так і безпосередньо сфери

послуг. Серед оглянутих далі проєктів, присутні основні найбільші «голосові асистенти» та засоби, що дозволяють виконання команд за допомогою голосового вводу.

Основними недоліками, з якими стикаються користувачі подібних програмних рішень є їх значна залежність від мережі інтернет, та повна відсутність або ж значна обмеженість можливості користувацьких налаштувань дій системи на голосові команди, що надає користувач. Цю проблему можна вирішити деякою зміною підходу до реалізації такого типу програмного забезпечення та створенням зручного механізму «кастомізації» користувацьких команд.

1.3 Аналіз успішних IT-проєктів

У даному розділі здійснимо огляд існуючих програмних рішень, що пропонують можливості голосового керування операційною системою та «голосових асистентів», проведемо порівняльний аналіз даних програмних засобів, та висвітимо сильні і слабкі сторони, для визначення областей, що потребують покращення та доопрацювання.

Перед тим як приступити до розробки застосунку голосового керування операційною системою, розглянемо застосунки, що намагалися або частково вирішують існуючі задачі та проблеми, виокремимо та проаналізуємо їх переваги та недоліки. Для аналізу обрано чотири найбільш поширених програмних рішень на ринку: Amazon Alexa, Apple Siri, Microsoft Cortana та Samsung Bixby.

1.3.1 Аналіз відомих технічних рішень

Amazon Alexa - це голосовий помічник, розроблений компанією Amazon, який використовує штучний інтелект і розпізнавання голосу для надання користувачам низки функцій і послуг. Alexa може виконувати різноманітні

завдання - від простих запитань до керування домашніми пристроями, медіаплеєрами, замовлення товарів в Інтернеті, відтворення музики, отримання новин та інформації про погоду, нагадувань та багато іншого. Вона також підтримує інтеграцію зі сторонніми додатками та сервісами для розширення функціональності.

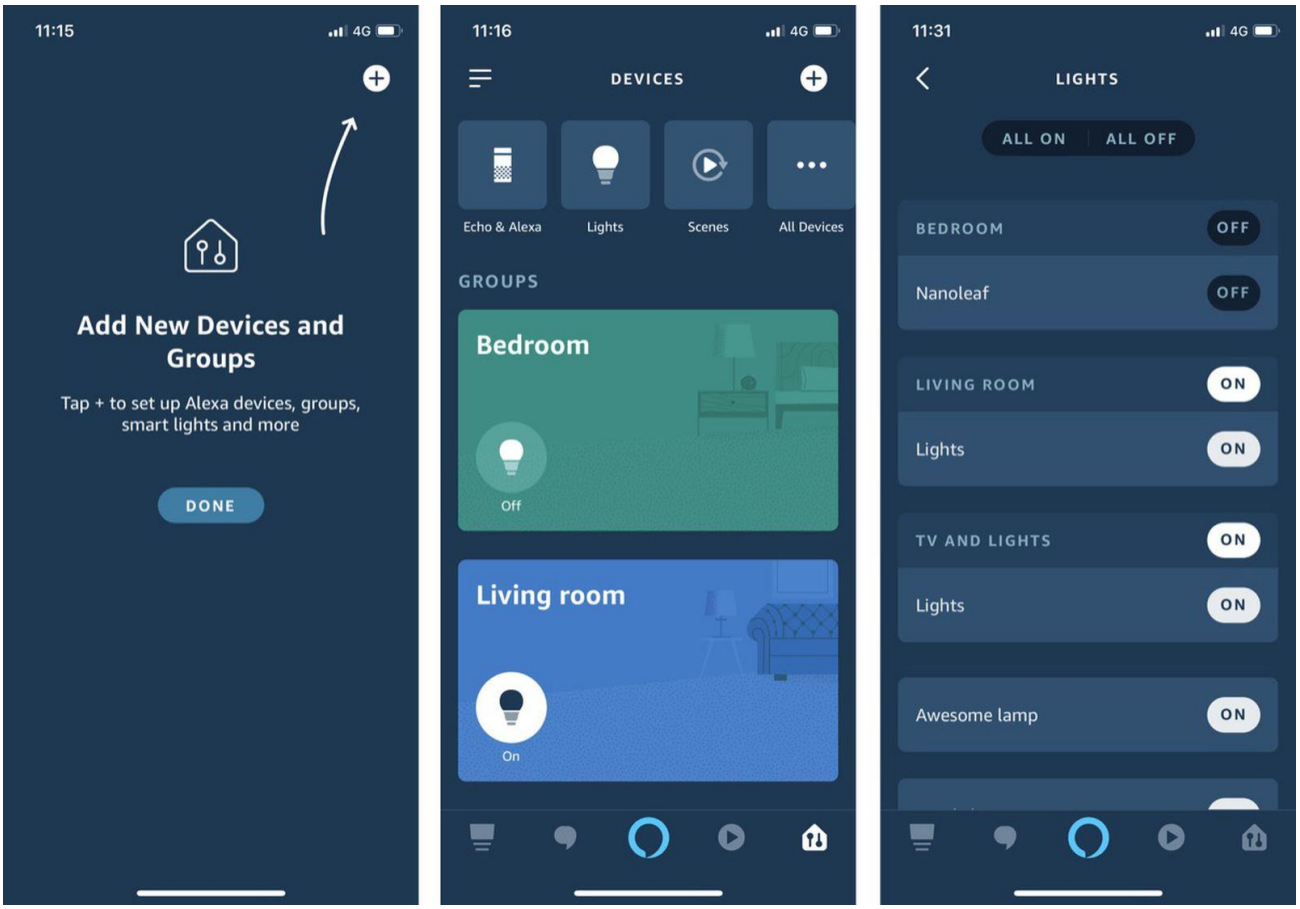


Рисунок 1.1 – Інтерфейс Amazon Alexa

Alexa базується на Amazon Echo - аудіопристрої для розумного будинку з вбудованим мікрофоном для збору голосових команд користувача. Коли користувач вимовляє ключове слово "Alexa", пристрій активується і починає слухати команди. Завдяки технології розпізнавання мови та обробці природної мови, Alexa розуміє команду або запит користувача і дає відповідь або виконує необхідну дію.

Окрім вбудованого пристрою Echo, Alexa також можна використовувати на мобільних пристроях за допомогою спеціального додатку, а також інтегрувати в низку сторонніх смарт-пристроїв, таких як телевізори, робочі станції та автомобілі, що підвищує її доступність та зручність використання [1].

Amazon Alexa є лідером серед голосових помічників і одним з найбільш визнаних і використовуваних рішень у своїй галузі.

Переваги Amazon Alexa:

- багата функціональність Alexa пропонує широкий спектр функцій і послуг, включаючи відтворення музики, керування домашніми пристроями, створення нагадувань, замовлення товарів, отримання прогнозу погоди та новин, пошук в Інтернеті та планування подій у календарі;
- інтеграція зі сторонніми програмами та сервісами: Alexa можна інтегрувати з різноманітними додатками та сервісами, які можна використовувати для розширення її функціональності та взаємодії з іншими популярними платформами та пристроями;
- багата база знань: Alexa має доступ до багатої бази даних і бази знань, що дозволяє їй швидко і точно відповідати на різноманітні запити користувачів;
- розпізнавання мови і природна мова: Alexa використовує потужну технологію розпізнавання мови і обробки природної мови, щоб розуміти користувачів і точно виконувати їхні команди.

Недоліки Amazon Alexa

- залежність від інтернету Більшість функцій Alexa вимагають підключення до інтернету, що може обмежити її використання, якщо у вас немає доступу до мережі;
- проблеми з конфіденційністю використання голосового помічника,

такого як Alexa, вимагає надсилання голосових даних на сервери Amazon для обробки та розпізнавання. Це може призвести до порушення конфіденційності та безпеки даних;

- обмежена мовна підтримка: У порівнянні з іншими голосовими помічниками, мовна підтримка Alexa може бути обмеженою. Вона більше підходить для англійської та інших поширених мов;
- конкуренція з іншими голосовими помічниками: Вона конкурує з іншими голосовими помічниками, такими як Siri, Google Assistant і Cortana, які пропонують широкий спектр функцій і послуг на ринку.

Apple Siri - це голосовий помічник, розроблений компанією Apple для своїх пристроїв, включаючи iPhone, iPad, Mac, Apple Watch і HomePod. Siri використовується для надання користувачам низки функцій і послуг за допомогою голосових команд.

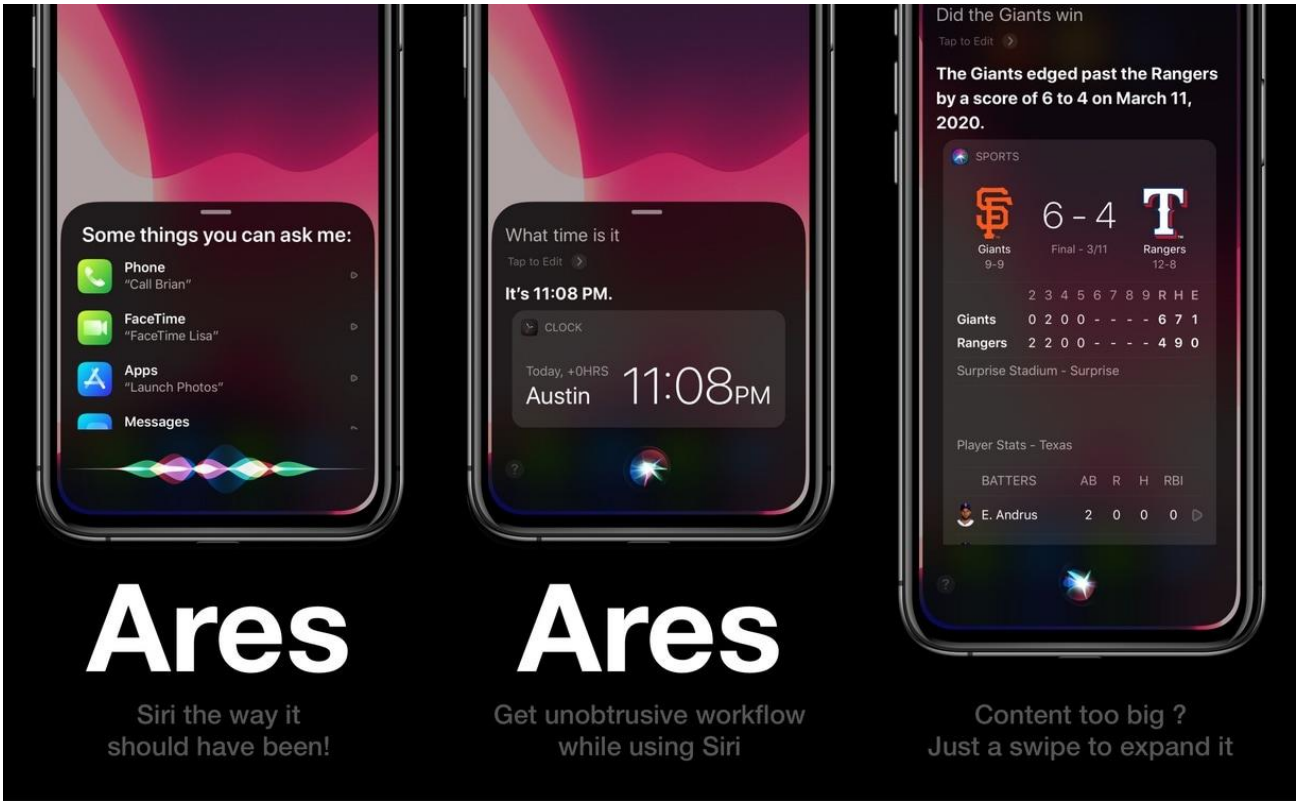


Рисунок 1.2 – Інтерфейс Apple Siri

Siri може виконувати різноманітні завдання, такі як надсилання повідомлень, здійснення телефонних дзвінків, відтворення музики, призначення зустрічей, пошук в Інтернеті, отримання новин та керування домашніми пристроями. Siri також може інтегруватися з програмами з App Store і взаємодіяти з програмами сторонніх розробників.

Однією з унікальних особливостей Siri є її здатність розпізнавати та адаптуватися до голосу користувача, щоб надавати персоналізовані відповіді та пропозиції. Siri також може використовувати контекстну інформацію, щоб краще розуміти запити користувачів і надавати точніші відповіді.

На додаток до розпізнавання голосу, Siri підтримує введення тексту, що дозволяє користувачам взаємодіяти з помічником, набираючи текст на клавіатурі.

Siri постійно оновлюється та вдосконалюється з оновленнями iOS та macOS, розширюючи свої можливості та функціональність [2].

Переваги Apple Siri:

- Apple Siri тісно інтегрована з продуктами та сервісами Apple, включаючи iPhone, iPad, Mac, Apple Watch і HomePod. Тож ви можете безперешкодно взаємодіяти зі своїми пристроями та користуватися перевагами екосистеми Apple;
- природна мова та персоналізація: Siri може розпізнавати природну мову та адаптуватися до голосу користувача, щоб надавати персоналізовані відповіді та пропозиції. Вона також може використовувати контекстну інформацію для кращого розуміння запитів;
- багата функціональність: Siri може відтворювати музику, робити нагадування, керувати програмами, надсилати повідомлення, шукати інформацію та багато іншого. Вона також може працювати зі сторонніми програмами, щоб розширити свої функціональні можливості;

- надійність та оновлення: Siri регулярно оновлюється через оновлення iOS та macOS для покращення функціональності та додавання нових можливостей.

Недоліки Apple Siri

- обмежена платформа: Siri доступна лише на пристроях Apple, таких як iPhone, iPad і Mac. Це означає, що не існує універсального рішення, яке можна використовувати на всіх платформах;
- мовна підтримка: Siri має обмежену мовну підтримку порівняно з іншими голосовими помічниками. Вона підходить для англійської та інших поширених мов, тому підтримка менш поширених мов може бути обмеженою;
- залежність від Інтернету: деякі функції Siri потребують підключення до Інтернету і можуть бути недоступні, якщо ви не підключені до мережі;
- обмежене налаштування: Siri не пропонує багато налаштувань або розширених параметрів, що може обмежити можливості пристосування асистента до потреб користувача.

Microsoft Cortana - це голосовий помічник, розроблений компанією Microsoft і доступний на різних пристроях, включаючи комп'ютери Windows, смартфони Windows Phone і деякі інші платформи. Cortana призначена для надання користувачам широкого спектру функцій і послуг за допомогою голосових команд, текстових запитів і призначена для надання користувачам широкого спектру функцій і послуг.

Cortana може виконувати різноманітні завдання, такі як нагадування користувачам про події, пошук в Інтернеті, надсилання повідомлень, керування календарями, прогнозами погоди та управління програмами. Вона також може інтегруватися з різними службами Microsoft, такими як Outlook, OneDrive і Skype, і безперешкодно взаємодіяти з екосистемою Microsoft.

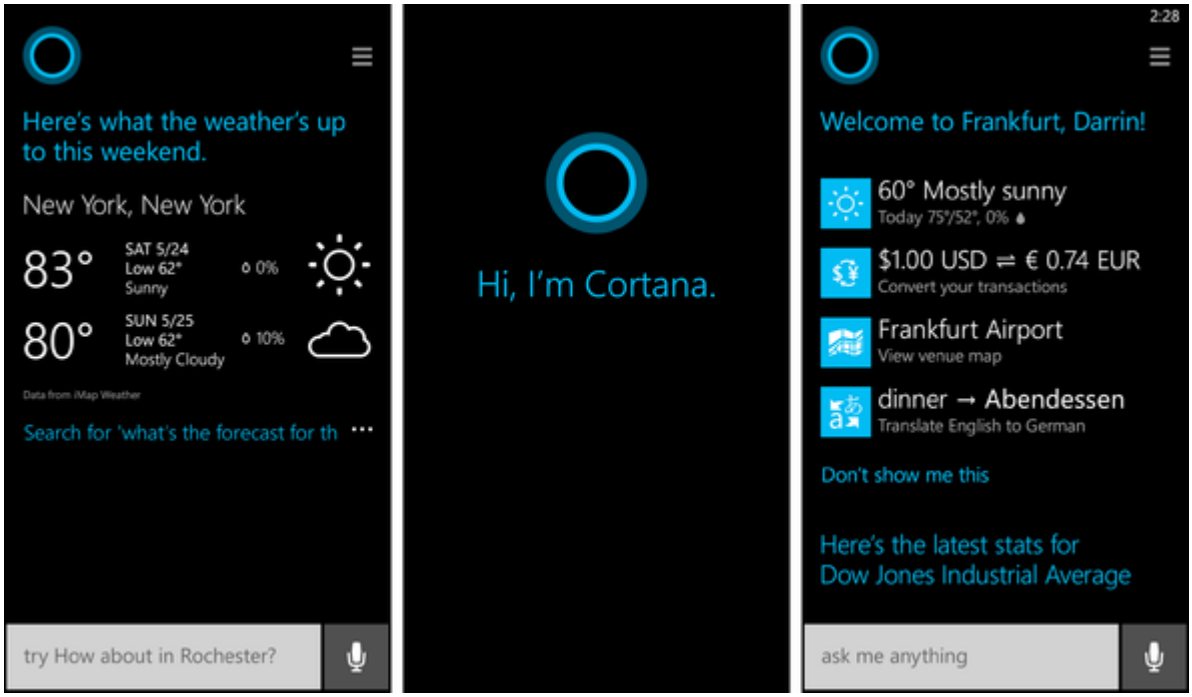


Рисунок 1.3 – Інтерфейс Microsoft Cortana

Однією з унікальних особливостей Cortana є її здатність навчатися та адаптуватися до потреб окремих користувачів. Вона може надавати персоналізовані рекомендації та відповіді, беручи до уваги вподобання та інформацію користувача.

Cortana також підтримує текстове та голосове введення, що дозволяє користувачам взаємодіяти з асистентом за допомогою клавіатури.

Корпорація Майкрософт постійно вдосконалює Cortana, випускаючи оновлення та додаткові функції, щоб покращити її роботу та задовольнити потреби користувачів.

Переваги Microsoft Cortana:

- інтеграція з екосистемою Microsoft: Cortana глибоко інтегрована з продуктами і службами Microsoft, такими як Windows, Outlook, OneDrive і Skype. Це дає змогу безперешкодно взаємодіяти з пристроями та використовувати

екосистему Microsoft;

- персоналізація та навчання: Cortana навчається та адаптується до потреб окремих користувачів, надаючи персоналізовані відповіді та рекомендації. Вона може враховувати вподобання, календар, контакти та іншу інформацію, щоб краще відповідати потребам користувача;
- розширені функції та взаємодія з додатками Cortana пропонує ряд функцій, таких як нагадування, пошук інформації та управління додатками. Вона також може працювати зі сторонніми додатками для розширення функціональності та доступу до спеціалізованих сервісів;
- кросплатформеність: Cortana доступна на різних платформах, включаючи Windows, Android та iOS. Тому її можна використовувати на різних пристроях і операційних системах.

Недоліки Microsoft Cortana

- обмежена мовна підтримка: Cortana підтримує обмежену кількість мов порівняно з іншими голосовими помічниками. Вона підходить для англійської та інших поширених мов, підтримка менш поширених мов може бути обмеженою;
- залежність від екосистеми Microsoft: Деякі функції Cortana можуть бути обмежені або не працювати належним чином без продуктів і служб Microsoft. Це може обмежити її використання на пристроях інших виробників;
- обмежене проникнення на деяких платформах: Cortana може бути менш поширеною та менш розвиненою на деяких платформах, таких як Android та iOS, порівняно з конкуруючими продуктами, такими як Google Assistant і Siri;
- менш часті оновлення: порівняно з іншими голосовими помічниками, Cortana може рідше оновлюватися і отримувати нові функції.

Samsung Bixby - це голосовий асистент, розроблений компанією Samsung. Він вперше представлений у 2017 році разом з випуском смартфонів Samsung

Galaxy S8 та S8+. Bixby розроблений для надання користувачам зручного голосового управління своїми пристроями та сервісами Samsung.

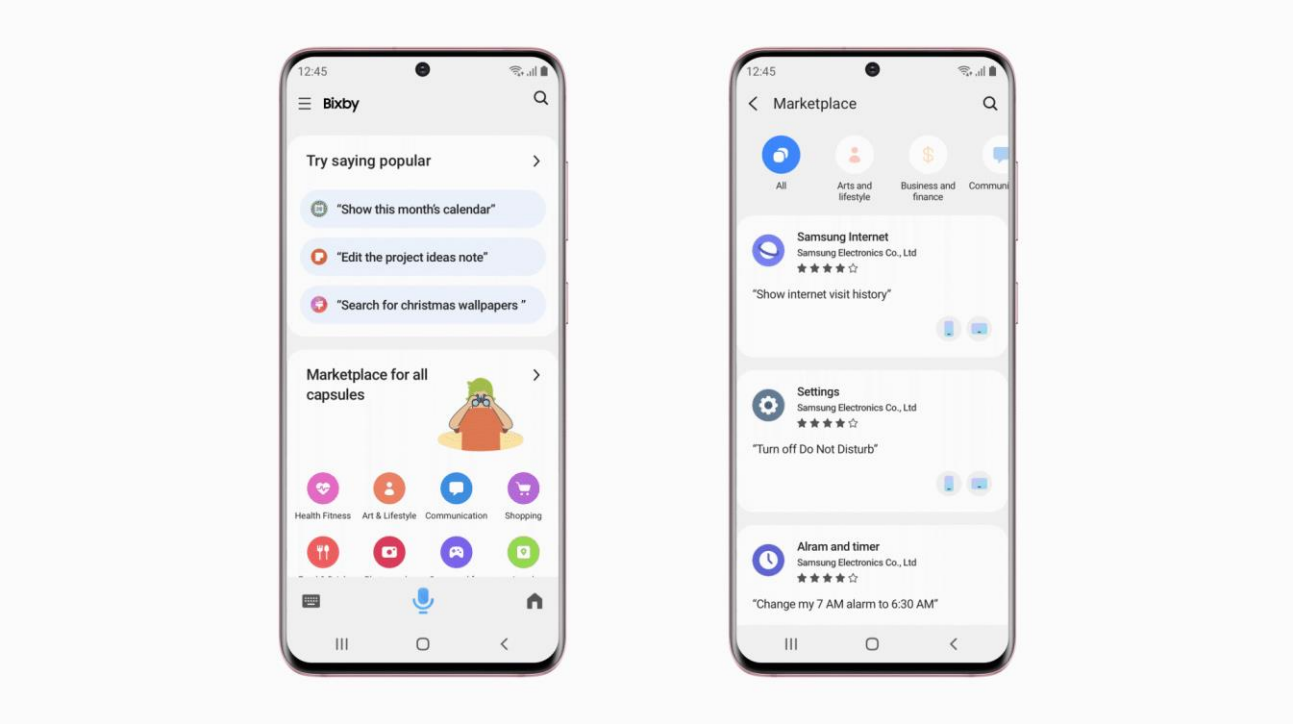


Рисунок 1.4 – Інтерфейс Samsung Bixby

Bixby пропонує широкий спектр функцій та можливостей, включаючи відкриття додатків, надсилання повідомлень, керування налаштуваннями пристрою, пошук інформації в Інтернеті, виконання завдань у мобільних додатках та багато іншого. Він також може виконувати запити з фотографій, розпізнавати об'єкти та надавати додаткову інформацію.

Однією з особливостей Bixby є його здатність до контекстного розуміння та продовження діалогу. Він може запитувати додаткову інформацію, щоб точніше розуміти запити користувача та надавати більш точні відповіді.

Bixby також має інтеграцію з екосистемою Samsung, дозволяючи користувачам керувати підключеними пристроями та послугами Samsung, такими як телевізори, побутова техніка та інші розумні пристрої.

Крім голосового вводу, Bixby також підтримує текстовий ввід та взаємодію через вбудований асистент Bixby Home.

Samsung продовжує вдосконалювати Bixby, випускаючи оновлення та додаткові функції, щоб поліпшити його ефективність та надати більше можливостей користувачам [3].

Переваги Samsung Bixby

- інтеграція з екосистемою Samsung Bixby глибоко інтегрований з продуктами та послугами Samsung, що дозволяє йому безперешкодно взаємодіяти з пристроями та використовувати екосистему Samsung;
- розуміння контексту Bixby може продовжувати діалог, розуміючи контекст запиту і запитуючи додаткову інформацію. Це може забезпечити більш точні відповіді та персоналізований користувацький досвід;
- розширені функції керування: за допомогою Bixby ви можете керувати різними аспектами пристрою та сервісами Samsung, наприклад, відкривати програми, виконувати дії в мобільних додатках і налаштовувати параметри пристрою;

Недоліки Samsung Bixby

- обмежена популярність: Порівняно з іншими голосовими помічниками, такими як Google Assistant і Amazon Alexa, Bixby може мати меншу популярність і проникнення на ринок. Це може обмежити можливості та підтримку сторонніх розробників;
- обмежена мовна підтримка: Порівняно з іншими голосовими помічниками, Bixby підтримує обмежену кількість мов. Він підходить для англійської та інших поширених мов, тому підтримка менш поширених мов може бути обмеженою;
- обмежена функціональність поза екосистемою Samsung: Bixby може мати обмежену функціональність і підтримку за межами екосистеми Samsung;

Наприклад, інтеграція зі сторонніми сервісами та програмами може бути менш розвиненою, ніж у конкурентів;

- недоступний на деяких пристроях: Не всі пристрої Samsung підтримують Bixby. Деякі старіші моделі та дешевші варіанти пристроїв можуть не підтримувати Bixby, що може обмежити доступність цього голосового помічника.

1.4 Аналіз вимог до програмного забезпечення

Основуючись на аналізі успішних аналогів, можна зробити висновок, що застосування повинне мати наступний функціонал:

- введення голосової команди;
- формування результату голосової команди;
- аудіовідповідь на запити з отримання інформації;
- отримання переліку користувацьких налаштувань голосових команд;
- додавання команди до переліку користувацьких команд;
- редагування користувацької команди;
- видалення користувацької команди;
- повідомлення про проблему з розпізнаванням команди.

1.4.1 Розроблення функціональних вимог

В ході аналізу вимог до ПЗ розділено виявлені вимоги на функціональні та нефункціональні. Функціональні вимоги описують поведінку системи та її внутрішню роботу. Для нашого застосування розроблено наступні функціональні вимоги, описані нижче:

Таблиця 1.1 – Опис функціональної вимоги REQ01

Номер	REQ01
Назва	Введення голосової команди

Продовження таблиці 1.1

Опис	Користувач має змогу ввести голосову команду за допомогою мікрофона

Таблиця 1.2 – Опис функціональної вимоги REQ02

Номер	REQ02
Назва	Формування результату голосової команди
Опис	Користувач має змогу отримати результат виконання голосової команди, що передана програмі за допомогою мікрофона, за умови валідності команди.

Таблиця 1.3 – Опис функціональної вимоги REQ03

Номер	REQ03
Назва	Аудіовідповідь на запити з отримання інформації
Опис	Користувач має змогу отримати аудіовідповідь на запит з отримання інформації з відкритих джерел за наявності підключення до мережі Інтернет та шуканої інформації.

Таблиця 1.4 – Опис функціональної вимоги REQ04

Номер	REQ04
Назва	Отримання переліку користувацьких налаштувань голосових команд
Опис	Користувач має змогу отримати перелік користувацьких налаштувань голосових команд, та можливість до подальшої взаємодії з ними.

Таблиця 1.5 – Опис функціональної вимоги REQ05

Номер	REQ05
Назва	Додавання команди до переліку користувацьких команд
Опис	Користувач має змогу додати команду до переліку користувацьких голосових команд.

Таблиця 1.6 – Опис функціональної вимоги REQ06

Номер	REQ06
Назва	Редагування користувацької команди
Опис	Користувач має змогу редагувати команду з переліку користувацьких голосових команд.

Таблиця 1.7 – Опис функціональної вимоги REQ07

Номер	REQ07
Назва	Видалення користувацької команди
Опис	Користувач має змогу видалити команду з переліку користувацьких голосових команд.

Таблиця 1.8 – Опис функціональної вимоги REQ08

Номер	REQ08
Назва	Повідомлення про проблему з розпізнаванням команди
Опис	Користувач має змогу отримати повідомлення про помилку розпізнавання голосової команди.

Таблиця 1.9 – Опис функціональної вимоги REQ09

Номер	REQ09
Назва	Додавання локалізації до переліку доступних локалізацій
Опис	Користувач має змогу додати локалізацію до переліку доступних локалізацій

Таблиця 1.10 – Опис функціональної вимоги REQ10

Номер	REQ10
Назва	Вибір робочої локалізації
Опис	Користувач має змогу обрати локалізацію з якою працює система.

На основі описаних функціональних вимог, сформовано відповідні варіанти використання. На рисунку 1.5 зображено Use Case діаграму для користувачів.

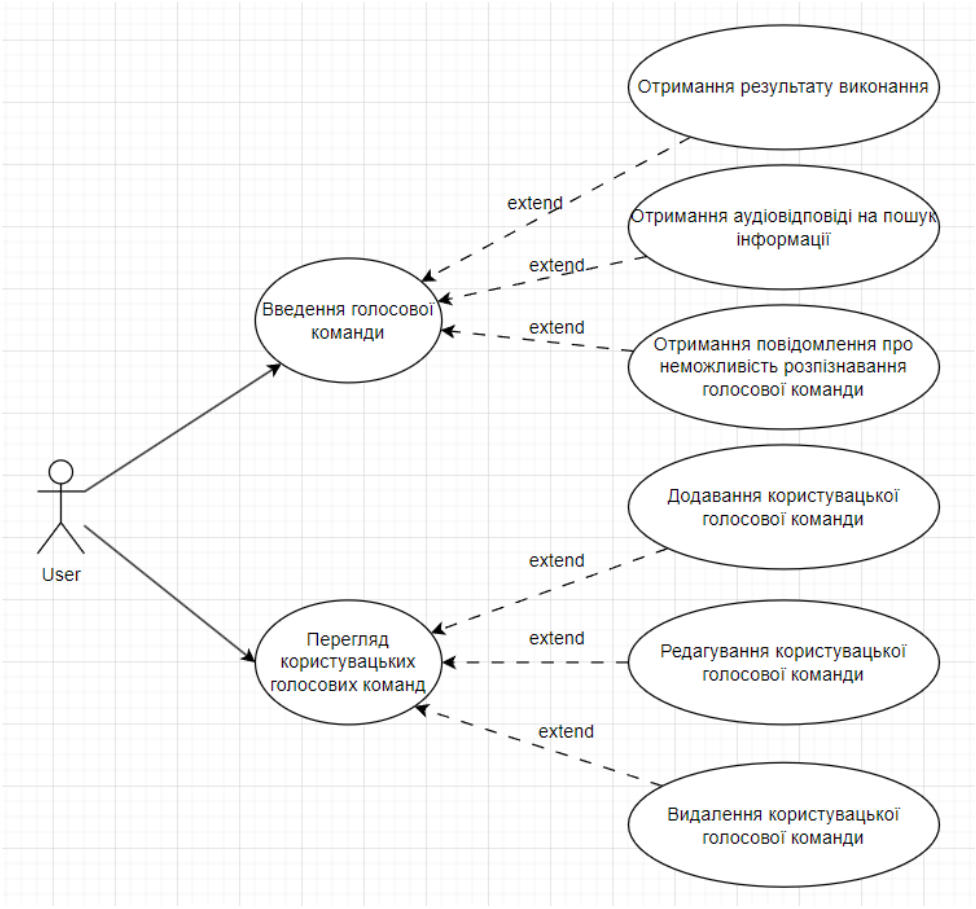


Рисунок 1.5 – Діаграма варіантів використання для користувача

Для даної діаграми опишемо відповідні варіанти використання:

Таблиця 1.11 – Варіант використання UC01

Назва	Введення голосової команди
Опис	Користувач передає голосову команду у систему
Учасник	Користувач
Передумови	Використання голосового вводу
Постумови	Голосова команда отримана системою
Основний сценарій	Користувач вводить голосову команлу, якщо ця команда входить до переліку користувацьких команд або ж це валідна незареєстрована команда, то команда успішно виконується, у іншому ж випадку, користувач отримує

Продовження таблиці 1.11

	повідомлення про неможливість виконання переданої команди

Таблиця 1.12 – Варіант використання UC02

Назва	Перегляд користувацьких голосових команд
Опис	Користувач отримує перелік передналаштованих голосових команд
Учасник	Користувач
Передумови	Користувач знаходиться у головному меню
Постумови	Користувач знаходиться на екрані передналаштованих голосових команд
Основний сценарій	Користувач натискає на «User Commands» та переходить на екран передналаштованих голосових команд; Користувач отримує можливість до перегляду, додавання, видалення та редагування голосових команд.

Таблиця 1.13 – Варіант використання UC03

Назва	Створення користувацької команди
Опис	Користувач додає нову передналаштовану голосову команду у систему
Учасник	Користувач
Передумови	Користувач знаходиться на екрані передналаштованих голосових команд
Постумови	Користувацьку голосову команду додано

Продовження таблиці 1.13

Основний сценарій	Користувач натискає на кнопку «Add», обирає бажаний тип команди, заповнює поля ідентифікатора голосової команди та токени виконуваної операції, після чого натискає кнопку «Save»

Таблиця 1.14 – Варіант використання UC04

Назва	Редагування користувацької команди
Опис	Користувач змінює уже існуючу передналаштовану голосову команду
Учасник	Користувач
Передумови	Користувач знаходиться на екрані передналаштованих голосових команд
Постумови	Користувацьку голосову команду редаговано
Основний сценарій	Користувач натискає на кнопку «Edit», вносить зміни до користувацької команди, після чого натискає кнопку «Save»

Таблиця 1.15 – Варіант використання UC05

Назва	Видалення користувацької команди
Опис	Користувач видаляє уже існуючу передналаштовану голосову команду
Учасник	Користувач
Передумови	Користувач знаходиться на екрані передналаштованих голосових команд
Постумови	Користувацьку голосову команду видалено

Продовження таблиці 1.15

Основний сценарій	Користувач натискає на кнопку «Delete», що призводить до видалення користувацької команди

Таблиця 1.16 – Варіант використання UC06

Назва	Додавання додаткової локалізації
Опис	Користувач додає нову локалізацію до системи
Учасник	Користувач
Передумови	Користувач знаходиться у головному меню
Постумови	Нова локалізація додана до системи
Основний сценарій	Користувач переходить до налаштувань локалізації, передає шлях до файлів локалізації, після чого встановлює значення стандартних операцій відповідних цій локалізації.

1.4.2 Розроблення нефункціональних вимог

Програмне забезпечення повинне відповідати наступним нефункціональним вимогам:

– мова інтерфейсу – англійська;

– зручний інтерфейс, зрозумілий і приємний для будь-якого середньостатистичного користувача;

– доступ до Інтернет (лише для пошуку інформації з відкритих джерел);

– підтримка операційної системи Windows.

1.5 Постановка задачі

Мета дипломного проєкту – створення програмного рішення у вигляді десктопного застосунку для налаштування та виконання користувацьких голосових команд у операційній системі Windows. Основною метою проєкту є створити такий застосунок, основними особливостями якого будуть можливість створення користувацьких голосових команд, що будуть виконуватись операційною системою, за допомогою чого буде розширено спектр взаємодії користувача з операційною системою та значно обмежити залежність функціоналу програмного рішення від наявності підключення до мережі Інтернет.

Для досягнення цієї мети сформовано наступні задачі:

- введення голосової команди;
- формування результату голосової команди;
- аудіо відповідь на запити з отримання інформації;
- отримання переліку користувацьких налаштувань голосових команд;
- додавання команди до переліку користувацьких команд;
- редагування користувацької команди;
- видалення користувацької команди;
- повідомлення про проблему з розпізнаванням команди.

Висновки до розділу

В даному розділі сформульовано мету проведення розробки десктопного застосунку, актуальність, основні задачі та цілі. Проведене дослідження існуючого програмного забезпечення, з'ясовано їх головні переваги та недоліки. Виконано аналіз відомих технічних рішень, вимог до програмного забезпечення.

Сформовано перелік функціональних та нефункціональних вимог до програмного забезпечення.

2 РОЗРОБКА МЕТОДУ РОЗПОДІЛЕНОГО МАШИННОГО НАВЧАННЯ

Нейронні мережі, що становлять основу машинного навчання, - це обчислювальні моделі, натхненні людським мозком. Вони складаються з шарів взаємопов'язаних вузлів або "нейронів", які обробляють дані за допомогою серії перетворень. Машинне навчання, підмножина штучного інтелекту, дозволяє комп'ютерам вчитися і робити прогнози або рішення на основі даних. Воно охоплює різні методи, включаючи контрольоване, неконтрольоване і навчання з підкріпленням, кожен з яких підходить для різних завдань [4].

Розподілене машинне навчання є ключовим елементом сучасних технологій обробки даних. Ця методика спрямована на використання множини обчислювальних ресурсів, таких як кластери серверів чи окремі обчислювальні вузли, з метою ефективної обробки величезних об'ємів даних. Основою машинного навчання є систематичний підхід до оптимізації параметрів моделі на основі даних [5].

У сфері обробки звуку машинне навчання відіграє ключову роль, особливо в автоматичному розпізнаванні мови (ASR). Системи ASR перетворюють розмовну мову на текст, аналізуючи звукові хвилі мовлення. Ці системи мають справу з різними проблемами, такими як різні акценти, мовленнєві патерни та фонові шуми. Алгоритми машинного навчання, особливо моделі глибокого навчання, значно підвищили точність та ефективність систем ASR [6].

Двонаправлені рекурентні нейронні мережі (BRNN) дуже ефективні в системах ASR. Вони обробляють дані як у прямому, так і в зворотному напрямку, фіксуючи контекст з обох кінців вхідної послідовності, що призводить до більш надійного розуміння мовного шаблону. Поряд з BRNN, коннекціоністська часова класифікація (CTC) часто використовується для узгодження вхідних мовних

фреймів з відповідним текстовим виводом без необхідності попередньої сегментації вхідних даних [7]. Крім того, методи доповнення даних, такі як додавання шуму, зміна висоти тону і швидкості, мають вирішальне значення для навчання надійних моделей ASR, оскільки вони допомагають моделі краще узагальнювати реальні сценарії [8].

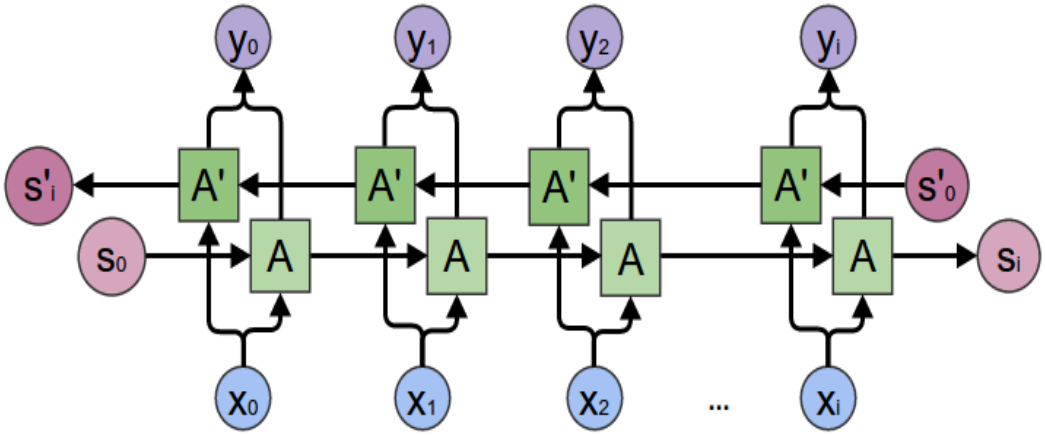


Рисунок 2.1 – Схема бідирекційної нейронної мережі

Важливим етапом розробки є попередня обробка даних. Застосовуючи інструментарій Librosa, що надає можливість перетворити аудіофайли на спектрограми, що представляють собою візуальний відображення частотного спектру звуку. Ці спектрограми служать вхідними даними для моделі.

TensorFlow, бібліотека машинного навчання з відкритим вихідним кодом, розроблена Google, широко використовується для побудови та навчання нейромережевих моделей, в тому числі для ASR. Гнучкість, масштабованість та велика колекція інструментів і бібліотек роблять її ідеальною для розробки складних моделей, таких як BRNN. TensorFlow також підтримує GPU-прискорення, що значно прискорює процес навчання великих нейронних мереж.

Common Voice, ініціатива Mozilla, - це проект, спрямований на створення безкоштовної та загальнодоступної бази даних голосових записів. Вона надає різноманітний набір зразків голосу різних дикторів на різних мовах, що робить її

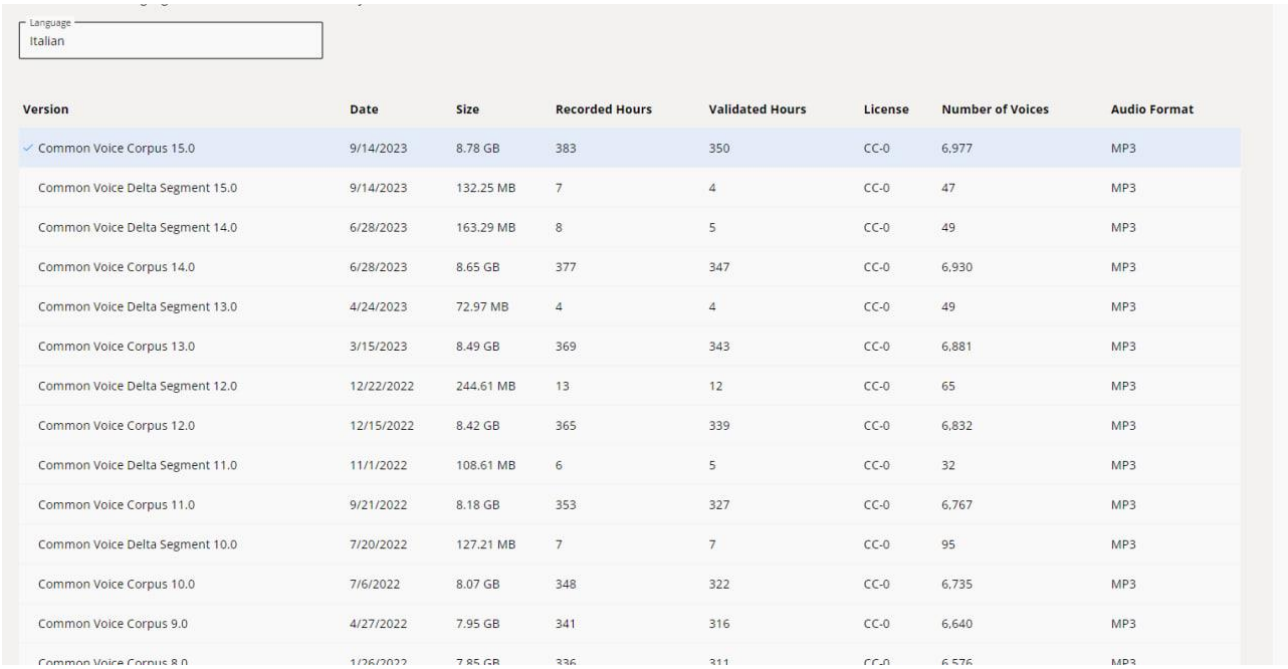
безцінним ресурсом для навчання і тестування моделей ASR. Використовуючи такі різноманітні набори даних, ASR-системи можна навчити розуміти широкий спектр мовних моделей і акцентів [9].

У якості набору даних для навчання моделі обрано саме Common Voice – вільний набір даних, аудіозаписи якого створюються волонтерами у мережі інтернет, а саме англомовний датасет, оскільки він містить найбільший обсяг доступних даних, рисунки 2.2-2.4 наведено для порівняння кількості [10].

Language
Ukrainian

Version	Date	Size	Recorded Hours	Validated Hours	License	Number of Voices	Audio Format
✓ Common Voice Corpus 15.0	9/14/2023	2.31 GB	105	94	CC-0	1,024	MP3
Common Voice Delta Segment 15.0	9/14/2023	251.17 MB	13	6	CC-0	135	MP3
Common Voice Delta Segment 14.0	6/28/2023	79.68 MB	4	4	CC-0	71	MP3
Common Voice Corpus 14.0	6/28/2023	2.07 GB	92	89	CC-0	889	MP3
Common Voice Delta Segment 13.0	4/24/2023	20.32 MB	2	4	CC-0	38	MP3
Common Voice Corpus 13.0	3/15/2023	1.99 GB	88	85	CC-0	818	MP3
Common Voice Delta Segment 12.0	12/22/2022	7.79 MB	1	9	CC-0	21	MP3
Common Voice Corpus 12.0	12/15/2022	1.97 GB	87	82	CC-0	780	MP3
Common Voice Delta Segment 11.0	11/1/2022	44.07 MB	3	4	CC-0	25	MP3
Common Voice Corpus 11.0	9/21/2022	1.96 GB	87	73	CC-0	759	MP3
Common Voice Delta Segment 10.0	7/20/2022	86.34 MB	5	4	CC-0	38	MP3
Common Voice Corpus 10.0	7/6/2022	1.92 GB	85	70	CC-0	734	MP3
Common Voice Corpus 9.0	4/27/2022	1.84 GB	80	67	CC-0	696	MP3
Common Voice Corpus 8.0	1/26/2022	1.77 GB	77	64	CC-0	684	MP3
Common Voice Corpus 7.0	7/28/2021	1.58 GB	67	57	CC-0	615	MP3
Common Voice Corpus 6.1	12/22/2020	1.13 GB	44	31	CC-0	459	MP3

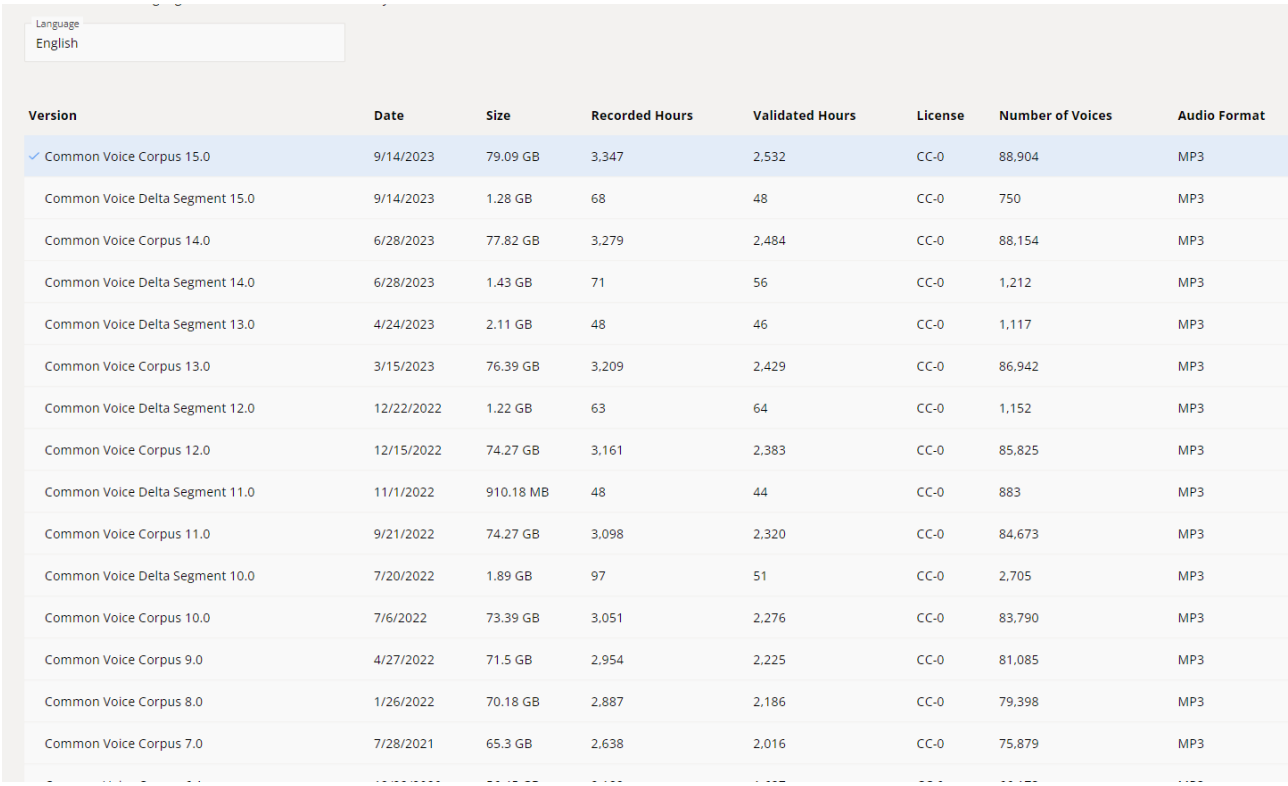
Рисунок 2.2 – Перелік доступних наборів даних українською мовою



Language
Italian

Version	Date	Size	Recorded Hours	Validated Hours	License	Number of Voices	Audio Format
✓ Common Voice Corpus 15.0	9/14/2023	8.78 GB	383	350	CC-0	6,977	MP3
Common Voice Delta Segment 15.0	9/14/2023	132.25 MB	7	4	CC-0	47	MP3
Common Voice Delta Segment 14.0	6/28/2023	163.29 MB	8	5	CC-0	49	MP3
Common Voice Corpus 14.0	6/28/2023	8.65 GB	377	347	CC-0	6,930	MP3
Common Voice Delta Segment 13.0	4/24/2023	72.97 MB	4	4	CC-0	49	MP3
Common Voice Corpus 13.0	3/15/2023	8.49 GB	369	343	CC-0	6,881	MP3
Common Voice Delta Segment 12.0	12/22/2022	244.61 MB	13	12	CC-0	65	MP3
Common Voice Corpus 12.0	12/15/2022	8.42 GB	365	339	CC-0	6,832	MP3
Common Voice Delta Segment 11.0	11/1/2022	108.61 MB	6	5	CC-0	32	MP3
Common Voice Corpus 11.0	9/21/2022	8.18 GB	353	327	CC-0	6,767	MP3
Common Voice Delta Segment 10.0	7/20/2022	127.21 MB	7	7	CC-0	95	MP3
Common Voice Corpus 10.0	7/6/2022	8.07 GB	348	322	CC-0	6,735	MP3
Common Voice Corpus 9.0	4/27/2022	7.95 GB	341	316	CC-0	6,640	MP3
Common Voice Corpus 8.0	1/26/2022	7.85 GB	336	311	CC-0	6,576	MP3

Рисунок 2.3 – Перелік доступних наборів даних італійською мовою



Language
English

Version	Date	Size	Recorded Hours	Validated Hours	License	Number of Voices	Audio Format
✓ Common Voice Corpus 15.0	9/14/2023	79.09 GB	3,347	2,532	CC-0	88,904	MP3
Common Voice Delta Segment 15.0	9/14/2023	1.28 GB	68	48	CC-0	750	MP3
Common Voice Corpus 14.0	6/28/2023	77.82 GB	3,279	2,484	CC-0	88,154	MP3
Common Voice Delta Segment 14.0	6/28/2023	1.43 GB	71	56	CC-0	1,212	MP3
Common Voice Delta Segment 13.0	4/24/2023	2.11 GB	48	46	CC-0	1,117	MP3
Common Voice Corpus 13.0	3/15/2023	76.39 GB	3,209	2,429	CC-0	86,942	MP3
Common Voice Delta Segment 12.0	12/22/2022	1.22 GB	63	64	CC-0	1,152	MP3
Common Voice Corpus 12.0	12/15/2022	74.27 GB	3,161	2,383	CC-0	85,825	MP3
Common Voice Delta Segment 11.0	11/1/2022	910.18 MB	48	44	CC-0	883	MP3
Common Voice Corpus 11.0	9/21/2022	74.27 GB	3,098	2,320	CC-0	84,673	MP3
Common Voice Delta Segment 10.0	7/20/2022	1.89 GB	97	51	CC-0	2,705	MP3
Common Voice Corpus 10.0	7/6/2022	73.39 GB	3,051	2,276	CC-0	83,790	MP3
Common Voice Corpus 9.0	4/27/2022	71.5 GB	2,954	2,225	CC-0	81,085	MP3
Common Voice Corpus 8.0	1/26/2022	70.18 GB	2,887	2,186	CC-0	79,398	MP3
Common Voice Corpus 7.0	7/28/2021	65.3 GB	2,638	2,016	CC-0	75,879	MP3

Рисунок 2.4 – Перелік доступних наборів даних англійською мовою

Висновки до розділу

У цьому розділі ми розглянули кілька ключових аспектів, що стосуються використання нейронних мереж та машинного навчання для автоматичного розпізнавання мови (ASR), зокрема в контексті бідирекційних рекурентних нейронних мереж (BRNN).

3 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір інструментів розробки

.NET (вимовляється як "дотнет") - це платформа розробки програмного забезпечення, створена компанією Microsoft. Вона надає середовище для створення, виконання і управління різноманітними програмами на різних платформах, зокрема настільних комп'ютерах.

Основні компоненти .NET:

- Common Language Runtime (CLR): CLR є виконавчою середовищем в рамках .NET. Вона відповідає за керування виконанням програмного коду, включаючи компіляцію, виконання, контроль типів, збірку сміття та обробку винятків;
- Base Class Library (BCL): BCL - це набір класів і компонентів, які надають базовий функціонал для розробки програм на .NET. Вона містить різні класи для роботи з файлами, мережами, базами даних, криптографією, колекціями даних та багато іншого;
- мовні специфікації: .NET підтримує різні мови програмування, включаючи C#, Visual Basic.NET, F# та інші. Кожна мова має свою власну синтаксичну структуру та використовує CLR для виконання програм;
- MSIL і JIT-компіляція: коли програмний код написаний на .NET мові компілюється, він перетворюється на Microsoft Intermediate Language (MSIL), який є проміжним форматом для виконання на CLR. Під час виконання MSIL перетворюється в нативний код за допомогою JIT-компіляції (Just-in-Time), що забезпечує високу продуктивність програм;
- управління пам'яттю: CLR включає в себе механізми зборки сміття, що автоматично видаляють непотрібні об'єкти з пам'яті. Це полегшує розробку

програм і зменшує ризик витоку пам'яті;

- доступ до ресурсів: .NET надає зручні механізми для роботи з різноманітними ресурсами, такими як файли, бази даних, мережі і служби. Вона підтримує стандартні протоколи комунікації, такі як HTTP, TCP/IP та інші.

Платформа .NET має кілька особливостей, які роблять її привабливою для розробників і дозволяють створювати потужні та масштабовані програми. Ось деякі особливості платформи .NET:

- мовна незалежність: .NET підтримує різні мови програмування, включаючи C#, Visual Basic.NET, F# та інші. Це дозволяє розробникам вибирати мову, яка найкраще підходить для їх потреб і навичок, і використовувати її для розробки програм на платформі .NET;
- кросплатформенність: завдяки .NET Core, платформа .NET стала кросплатформенною, що означає, що програми, написані на .NET, можуть працювати на різних операційних системах, таких як Windows, macOS і Linux. Це дозволяє розробникам створювати додатки, які можна використовувати на різних пристроях та платформах;
- засоби безпеки: .NET має вбудовану підтримку безпеки, включаючи механізми автентифікації, авторизації і шифрування. Це дозволяє розробникам легко захистити свої програми та дані від несанкціонованого доступу та зловживання;
- розширення: завдяки відкритій природі .NET, розробники можуть створювати власні бібліотеки класів, компоненти та розширення для використання в своїх програмах. Це сприяє перевикористанню коду, підвищенню продуктивності та розширенню можливостей розробки.

C# (вимовляється "Сі-шарп") - це об'єктно-орієнтована мова програмування, розроблена компанією Microsoft. Вона є однією з ключових мов для платформи .NET і широко використовується для розробки різноманітних додатків, від десктопних програм до веб-додатків і мобільних додатків.

Основні особливості C#:

- синтаксис: C# має синтаксис, схожий на синтаксис інших популярних мов програмування, таких як C++ і Java. Вона використовує фігурні дужки для відокремлення блоків коду, крапку з комою для розділення операторів, а також ключові слова і конструкції, які визначають структуру програми;

- об'єктно-орієнтованість: C# підтримує концепції об'єктно-орієнтованого програмування, такі як класи, об'єкти, спадкування, поліморфізм і інкапсуляція. Це дозволяє розробникам створювати структуровані, модульні і легкозрозумілі програми;

- керований код: C# компілюється в проміжний байт-код, відомий як Common Intermediate Language (CIL). Цей байт-код виконується в середовищі виконання, такому як Common Language Runtime (CLR), яке забезпечує керування пам'яттю, обробку виключень, збирання сміття та інші рутинні задачі, що спрощують розробку програм;

- можливості мови: C# надає розробникам багато корисних можливостей, таких як обробка винятків, делегати і події, лямбда-вирази, властивості, індексатори, оператори перевантаження та багато іншого. Це дозволяє писати елегантний і ефективний код з меншими зусиллями;

- розширення .NET: C# інтегрується з різноманітними бібліотеками і фреймворками платформи .NET, що дозволяє розробникам використовувати готові компоненти і розширення для реалізації певного функціоналу, що прискорює процес розробки;

- сильна типізація: C# використовує сильну типізацію, що означає, що всі змінні мають визначений тип і не можуть змінювати свій тип без використання явного приведення типів. Це допомагає виявляти помилки в ході компіляції і полегшує роботу з кодом;

- управління пам'яттю: C# використовує автоматичне керування пам'яттю, що дозволяє автоматично відслідковувати і звільняти пам'ять, виділену для об'єктів. Розробнику не потрібно вручну викликати операції виділення та звільнення пам'яті, що допомагає уникнути багатьох проблем, пов'язаних з управлінням пам'яттю, таких як витоки пам'яті і неправильне використання пам'яті;

- асинхронне програмування: C# має вбудовану підтримку асинхронного програмування за допомогою ключових слів async і await. Це дозволяє розробникам створювати асинхронні операції, які не блокують головний потік виконання і полегшують роботу з паралельними та асинхронними задачами;

- властивості і індексатори: C# надає можливість використовувати властивості, які дозволяють контролювати доступ до полів класу і виконувати додаткову логіку при читанні або записі значень. Крім того, ви можете використовувати індексатори для доступу до елементів колекцій за допомогою індексу;

- делегати і події: C# підтримує механізм делегатів, які дозволяють передавати функції як параметри і виконувати їх в іншиих частинах програми. Крім того, ви можете використовувати події для реалізації публікації-підписки, де об'єкти можуть підписатися на подію і реагувати на її виникнення;

- лямбда-вирази: C# підтримує лямбда-вирази, що дозволяють створювати анонімні функції без необхідності оголошення окремих методів. Лямбда-вирази дозволяють скоротити кількість коду і полегшують роботу з колекціями і делегатами.

Windows Presentation Foundation (WPF) - це технологія, яка використовується для розробки графічних інтерфейсів користувача (GUI) у

десктопних додатках під управлінням операційної системи Windows. WPF є частиною платформи .NET і забезпечує потужні можливості для створення багатофункціональних і привабливих застосунків з використанням різних елементів управління, анімацій, графіки, стилів та інших візуальних ефектів [11].

Особливості WPF:

- розділення логіки та вигляду: WPF використовує патерн MVVM (Model-View-ViewModel), що дозволяє розділити логіку додатку від його вигляду. Це сприяє більшій організації коду, полегшує тестування та підтримку, а також дозволяє більш гнучко керувати відображенням даних;

- візуальна гнучкість: WPF надає широкі можливості для стилізації та налаштування елементів управління. Ви можете використовувати власні стилі, шаблони та ресурси, щоб забезпечити унікальний вигляд вашого додатку. Крім того, WPF підтримує векторну графіку, що дозволяє створювати резервовані за розміром елементи, які виглядають гарно незалежно від роздільної здатності екрану;

- анімація та перехід ефектів: WPF надає потужні засоби для створення анімації та переходів між різними станами елементів. Ви можете застосовувати анімацію до властивостей, створювати комплексні переходи між сторінками або елементами і створювати вражаючі візуальні ефекти;

- інтеграція зі змістом: WPF дозволяє вбудовувати різноманітний зміст у ваші додатки, такий як відео, зображення, веб-сторінки та інше. Ви можете використовувати контроли, такі як MediaElement і WebBrowser, для взаємодії з цими типами змісту;

- повністю програмований інтерфейс: WPF надає можливість створювати інтерфейс за допомогою коду, що дає більшу гнучкість і контроль

над елементами управління. Ви можете динамічно створювати, змінювати та керувати елементами управління відповідно до ваших потреб;

- підтримка введення: WPF забезпечує багатофункціональну підтримку введення, включаючи розпізнавання жестів, обробку клавіатури та миші, а також підтримку тачскріну. Ви можете легко реагувати на різні типи введення користувача та виконувати відповідні дії;

- потужна обробка даних: WPF має вбудовану підтримку для зв'язування даних, що дозволяє зручно прив'язувати дані до елементів управління. Ви можете використовувати зв'язування даних для автоматичного оновлення інтерфейсу при зміні даних.

Python - це інтерпретована, об'єктно-орієнтована мова програмування з високим рівнем абстракції. Вона розроблена в кінці 1980-х років і стала однією з найпопулярніших мов програмування в світі [12].

Особливості Python:

- простота та зрозумілість: Python відомий своєю простотою, зрозумілістю та читабельністю коду. Він має простий синтаксис, що нагадує англійську мову, і не вимагає великої кількості додаткового синтаксису. Це дозволяє розробникам писати код швидше і з меншими зусиллями;

- інтерпретованість: Python є інтерпретованою мовою, що означає, що він не потребує передкомпіляції. Код може бути виконаний безпосередньо інтерпретатором Python, що полегшує швидку розробку та налагодження програм;

- об'єктно-орієнтованість: Python підтримує об'єктно-орієнтоване програмування, що дозволяє розробникам створювати класи, об'єкти, спадкування, поліморфізм і інкапсуляцію. Це дозволяє писати модульний, структурований код і полегшує повторне використання коду;

- велика стандартна бібліотека: Python має велику стандартну бібліотеку, яка включає в себе багато корисних модулів і функцій. Ця бібліотека надає готові рішення для багатьох типових задач, таких як робота з файлами, мережеве програмування, робота з базами даних, обробка тексту, математичні обчислення та багато іншого;

- переносимість: Python підтримує переносимість між різними платформами, такими як Windows, macOS і Linux. Код, написаний на Python, може бути виконаний на різних операційних системах без необхідності зміни вихідного коду;

- широкі можливості: Python може бути використаний для різноманітних завдань, включаючи веб-розробку, наукові обчислення, аналіз даних, машинне навчання, автоматизацію, ігрову розробку, робототехніку та багато іншого. Ця універсальність робить Python привабливим вибором для багатьох розробників;

- динамічна типізація: Python використовує динамічну типізацію, що означає, що змінні не потребують явного вказівника типу при оголошенні. Тип змінної визначається автоматично на основі значення, яке вона містить. Це дає більшу гнучкість при програмуванні і полегшує роботу зі змінними;

- високорівнева мова програмування: Python надає високорівневі абстракції, що дозволяють розробникам писати код, що більш зосереджений на логіці програми, а не на деталях роботи з пам'яттю або низькорівневих операціях. Це полегшує створення швидкого і зрозумілого коду, а також підвищує продуктивність розробника;

- інтерактивний режим і скриптованість: Python підтримує інтерактивний режим виконання, де команди можна вводити і виконувати безпосередньо в інтерпретаторі Python. Це дозволяє швидко експериментувати з кодом, перевіряти функціональність і отримувати негайний результат. Крім того,

Python можна використовувати для написання скриптів, які автоматизують повторювані завдання або виконуються під час виконання операційної системи;

- система керування пам'яттю: Python використовує автоматичне управління пам'яттю, що дозволяє розробникам уникнути проблем, пов'язаних з вручним виділенням та звільненням пам'яті. Це зменшує кількість помилок, пов'язаних з пам'яттю, і полегшує процес розробки.

Ці особливості Python дозволяють розробникам швидко створювати ефективний та зрозумілий код, працювати з різними областями програмування і розширювати функціональність мови за допомогою багатьох доступних бібліотек та модулів [13].

NumPy (Numerical Python) - це бібліотека для мови програмування Python, яка надає підтримку для великих, багатовимірних масивів і матриць, разом з великою колекцією математичних функцій для роботи з цими масивами. NumPy є одним з основних інструментів для наукових обчислень та обробки даних у середовищі Python.

Основні особливості NumPy:

- масиви: NumPy надає об'єкт ndarray (N-dimensional array), який дозволяє зберігати та маніпулювати багатовимірними масивами даних. Масиви NumPy є ефективними, оскільки вони забезпечують однорідну типізацію, тобто всі елементи в масиві мають однаковий тип даних;
- математичні операції: NumPy надає велику кількість векторизованих математичних функцій, які можна застосовувати до масивів без необхідності використання циклів. Це дозволяє виконувати швидкі та зручні операції над масивами, такі як обчислення, сортування, перетворення та фільтрація даних;
- індексація та зрізи: NumPy надає багато можливостей для доступу до елементів масиву за допомогою індексів та зрізів. Ви можете використовувати ці можливості для вибору конкретних елементів, витягування підмасивів, зміни

значень та багато іншого;

- бродкастінг: NumPy використовує механізм бродкастінгу, що дозволяє виконувати операції над масивами різних розмірностей без необхідності розширення розмірів масивів вручну. Це спрощує та прискорює обробку даних та векторизовані обчислення.

Matplotlib - це бібліотека для мови програмування Python, яка надає потужні засоби для візуалізації даних та створення графіків. Вона дозволяє розробникам створювати різноманітні типи графіків, діаграм, гістограм, розподілених графіків та багато іншого.

Основні особливості Matplotlib:

- простота використання: Matplotlib надає простий та зрозумілий API, що дозволяє легко створювати графіки. Ця бібліотека має схожий синтаксис з MATLAB, що робить її зручною для користувачів, які вже знайомі з MATLAB;
- різноманітність типів графіків: Matplotlib підтримує багато типів графіків, таких як лінійні графіки, стовпчасті діаграми, кругові діаграми, точкові графіки, контурні графіки, 3D-графіки та багато інших. Ви можете використовувати ці типи графіків для візуалізації даних з різних областей;
- контроль над виглядом графіків: Matplotlib надає багато можливостей для настроювання вигляду графіків. Ви можете встановлювати заголовки, мітки осей, легенди, кольори, стилі ліній, розміри шрифтів та багато іншого. Це дозволяє створювати професійно виглядящі графіки з великою кількістю деталей;
- підтримка векторної та растрової графіки: Matplotlib дозволяє зберігати графіки у різних форматах, включаючи векторні формати (наприклад, PDF, SVG) і растрові формати (наприклад, PNG, JPEG). Це дозволяє зберігати графіки з високою якістю для використання у наукових публікаціях, презентаціях або веб-сайтах;

- інтеграція з іншими бібліотеками: Matplotlib можна поєднувати з іншими бібліотеками Python, такими як NumPy, Pandas, SciPy, Seaborn та іншими. Це дозволяє легко обробляти дані та створювати складні візуалізації з використанням комбінації цих бібліотек.

Matplotlib є потужним інструментом для візуалізації даних у Python, який надає велику гнучкість та можливості настроювання для створення якісних графіків та діаграм. Він широко використовується у наукових дослідженнях, аналізі даних, статистиці, веб-розробці та інших областях.

TensorFlow - це відкрите програмне забезпечення для чисельних обчислень, яке спеціалізується на роботі з нейромережами та глибинним навчанням. Він розроблений командою Google Brain і широко використовується у багатьох областях, включаючи комп'ютерне зору, обробку природної мови, розпізнавання мови, геноміку, рекомендаційні системи та інші.

Основні особливості TensorFlow:

- графове обчислення: TensorFlow базується на графовій моделі обчислень, де обчислення представлені у вигляді графа. Вершини графа представляють операції, а ребра - дані, що протікають між операціями. Це дозволяє ефективно виконувати паралельні обчислення та оптимізувати виконання складних моделей;
- глибоке навчання: TensorFlow має вбудовану підтримку для глибинного навчання, що дозволяє створювати та тренувати нейронні мережі з великою кількістю шарів. Він надає широкий набір високорівневих API для спрощення процесу розробки моделей глибинного навчання;
- модульність та розширюваність: TensorFlow надає багато модулів та інструментів для зручного створення та роботи з моделями. Він підтримує різні шари, оптимізатори, функції втрат, метрики оцінки та інше. Крім того, TensorFlow має відкрите API, яке дозволяє розширювати його функціональність

та використовувати спеціалізовані бібліотеки для різних завдань;

- переносимість: TensorFlow підтримує різні платформи, включаючи CPU, GPU, TPU (Tensor Processing Unit) та хмарні середовища. Це дозволяє використовувати TensorFlow на різних пристроях та масштабувати його для великих обчислювальних завдань;
- велика спільнота: TensorFlow має велику активну спільноту розробників, яка надає підтримку, документацію, приклади та розширення. Це сприяє швидкому розповсюдженню знань та допомагає вирішувати проблеми під час розробки та використання TensorFlow.

TensorFlow є однією з найпопулярніших бібліотек для глибинного навчання та чисельних обчислень у сфері штучного інтелекту та машинного навчання. Він надає потужні інструменти для розробки та тренування моделей, які можна використовувати у широкому спектрі завдань обробки даних.

Rider - це інтегроване середовище розробки (IDE) для мови програмування C# та інших мов на платформі .NET. Воно розроблене компанією JetBrains і підтримується для роботи на різних операційних системах, включаючи Windows, macOS і Linux [14].

Основні особливості Rider:

- кросплатформеність: Rider підтримує розробку на різних операційних системах, що дозволяє розробникам працювати в зручному середовищі незалежно від платформи, на якій вони працюють. Ви можете розробляти програми на C# і .NET як на Windows, так і на macOS або Linux;
- повна підтримка C# і .NET: Rider надає повний набір інструментів та функцій для розробки на мові C# та платформі .NET. Це включає автодоповнення коду, перехід по визначенню, рефакторинги, налагодження, аналіз коду, підтримку тестування та багато іншого;
- інтеграція з іншими інструментами: Rider інтегрується з різними

інструментами розробки, такими як Git, SVN, Mercurial, TFS, Docker, SQL, HTTP-клієнти та багато інших. Ви можете легко працювати зі своїми улюбленими інструментами прямо з інтерфейсу Rider;

- візуальний редактор XAML: Rider надає потужний візуальний редактор для розробки інтерфейсів на мові XAML. Ви можете швидко створювати, редагувати та переглядати XAML-файли для розробки веб-сайтів, десктопних та мобільних додатків;
- підтримка різних мов програмування: Окрім розробки на C# і .NET, Rider підтримує інші мови програмування, такі як JavaScript, TypeScript, HTML, CSS, XML, JSON і багато інших. Ви можете комбінувати різні мови в одному проєкті і зручно розробляти на них.

Rider є потужним інструментом для розробки на мові C# та платформі .NET. Він надає багато корисних функцій, які полегшують роботу розробників і дозволяють швидко розробляти високоякісні програми на .NET [15].

PyCharm - це інтегроване середовище розробки (IDE), розроблене компанією JetBrains, спеціально для розробки програм на мові програмування Python. Воно надає розширені можливості для розробки, налагодження і керування проєктами на Python.

Основні особливості PyCharm:

- розширені можливості редактора коду: PyCharm має потужний редактор коду з підсвічуванням синтаксису, автодоповненням, рефакторингом, швидкими переходами та багатьма іншими функціями, які полегшують написання коду на Python;
- універсальність: PyCharm підтримує розробку на різних платформах, включаючи Windows, macOS і Linux. Ви можете використовувати його незалежно від операційної системи, на якій ви працюєте;
- управління проєктами: PyCharm надає зручні інструменти для

управління проєктами на Python. Ви можете створювати, керувати та відлажувати свої проєкти з використанням вбудованих інструментів, таких як управління віртуальними середовищами, управління залежностями та інше;

- відладка та профілювання: PyCharm надає розширені можливості для відлагодження коду Python. Ви можете встановлювати точки зупину, відстежувати значення змінних, виконувати крок за кроком і багато іншого. Крім того, PyCharm має вбудовані інструменти профілювання, які допомагають знайти та оптимізувати швидкодіючі ділянки коду;
- підтримка інших технологій: PyCharm не обмежується лише розробкою на Python. Він підтримує інші мови програмування, такі як JavaScript, HTML, CSS, SQL та інші. Ви можете розробляти багатомовні проєкти та використовувати різні технології в одному середовищі.

PyCharm є популярним вибором серед розробників Python завдяки своїм потужним функціям, зручному інтерфейсу та підтримці різних платформ. Він допомагає розробникам прискорити робочий процес та створити високоякісний код на мові Python [16].

3.2 Моделювання та аналіз програмного забезпечення

3.2.1 Моделювання та аналіз модулю навчання моделі автоматичного розпізнавання мови

Автоматичне розпізнавання мови (ASR) є галуззю комп'ютерної лінгвістики, що спрямована на перетворення голосових сигналів у текстовий формат. Під час останніх десятиліть розвитку технологій ASR досягнуто значних успіхів, особливо завдяки глибокому навчанню і нейронним мережам.

Людське вухо розрізняє частоти не лінійно. Мел-фреквенційні кепстральні коефіцієнти (MFCC) намагається відобразити сприйняття частоти людиною у спектральних коефіцієнтах. Для отримання MFCC аудіосигнал поділяється на

короткі фрагменти, до яких застосовується перетворення Фур'є, а потім перетворюється до мел-шкали. Для вирахування MFCC використано бібліотеку librosa, після отримання спектральних коефіцієнтів застосовано техніку "data augmentation" для збільшення розмаїття даних та підвищення здатності моделі до узагальнення.

Для покращення роботи моделей глибокого навчання на реальних даних, звичайно використовують техніку аугментації. Це зокрема важливо для ASR, оскільки реальне середовище може включати шуми, різну швидкість мовлення тощо. Використано випадкові модифікації звукового сигналу: додавання шуму, зміну висоти тона та швидкості. Це допомагає моделі стати менш чутливою до змін в умовах введення.

Ефективна підготовка даних є ключовим аспектом для будь-якої задачі машинного навчання. Створено функцію завантаження даних, яка обходить каталоги, знаходить аудіофайли та відповідні їм транскрипції, після чого конвертує аудіо в MFCC.

Нейронні мережі, зокрема рекурентні нейронні мережі (RNN), є відомими своєю ефективністю у задачах розпізнавання послідовностей, до яких належить ASR. У нашому підході використано RNN, а також можемо розглядати застосування більш складних структур, таких як LSTM або GRU.

Для ефективного навчання моделі потрібен великий набір даних, який відображає реальний світ. Для досягнення цієї мети використано відкритий датасет, що надає близько 1000 годин аудіозаписів тексту англійською мовою, та файли з транскрипцією у текстовому форматі.

Функціонал що забезпечує цей програмний модуль:

- зчитування даних для тренування моделі;
- аугментація вхідних даних, для підвищення ефективності

тренування;

- тренування моделі;
- валідація моделі за допомогою тестових даних.

3.2.2 Моделювання та аналіз модулю користувацьких налаштувань

Функціонал що забезпечує цей програмний модуль:

- зчитування даних користувацьких голосових команд;
- додавання користувацьких голосових команд;
- видалення користувацьких голосових команд;
- редагування користувацьких голосових команд.

Для імплементації програмного модулю створено перелік (enum) типів стандартних операцій виконуваних операційною системою, інтерфейс та класи команд, що змістить інформацію про операцію, що виконується та додаткові дані, необхідні для виконання того, чи іншого типу операції. Також, створено клас для що надає функціонал для зчитування та отримання інформації про користувацькі налаштування (клас-рідер), та клас для додавання, видалення та редагування голосових команд користувача (клас-процесор).

3.2.3 Моделювання та аналіз модулю обробки голосових команд

Функціонал що забезпечує цей програмний модуль:

- перетворення голосових команд у текст;
- перетворення тексту команд у токени;
- створення команди на основі токенів отриманих з тексту голосових команд.

Для реалізації програмного блоку сформовано набір класів. Клас що взаємодіє з моделлю машинного навчання та перетворює вхідні голосові команди у текст. Клас, парсер, що порівнює команду з наявними у користувацьких

налаштуваннях, та за відсутності, формує клас-команду за допомогою токенізованих елементів тексту голосової команди.

3.2.4 Моделювання та аналіз модулю виконання команд

Функціонал що забезпечує цей програмний модуль:

- виконання голосових команд користувача.

На основі класів попередніх модулів та класу обробнику команд, цей модуль забезпечує виконання операцій, які викликає користувач за допомогою голосових команд.

3.2.5 Моделювання та аналіз модулю локалізацій

Функціонал що забезпечує цей програмний модуль:

- додавання користувацьких локалізацій;
- вибір необхідної користувачу локалізації.

На основі класів попередніх модулів та класу обробнику команд, цей модуль забезпечує можливість користувачу власноруч додавати необхідні локалізації, що знаходяться у відкритому доступі та обирати необхідну для роботи.

3.3 Архітектура програмного забезпечення

3.3.1 Моделювання та аналіз модулю навчання моделі автоматичного розпізнавання мови

Таблиця 3.1 - Метод Train

Назва	Train
Вхідні дані	This.Model, X_train, y_train

Продовження таблиці 3.1

Вихідні параметри	This.Model
Опис роботи	Виклик методу для навчання моделі розпізнавання мовлення, вхідні данні це сиквенційна бідирекційна модель з даними для навчання. У результаті ми отримуємо навчену модель.

3.3.2 Моделювання та аналіз модулю користувацьких налаштувань

Таблиця 3.2 - Метод GetUserCommandsAsync

Назва	GetUserCommandsAsync
Вхідні дані	Відстутні
Вихідні параметри	Task<IEnumerable<UserCommandDto>>
Опис роботи	Відбувається виклик методу GetUserCommandsAsync класу-рідера, який повертає колекцію сутностей типу UserCommand, що після цього мапляться у сутності типу UserCommandDto

Таблиця 3.3 - Метод GetUserCommandByKeyAsync

Назва	GetUserCommandByKeyAsync
Вхідні дані	String
Вихідні параметри	Task<UserCommandDto>
Опис роботи	Відбувається виклик методу GetUserCommandByKeyAsync класу- рідера, який повертає сутність типу UserCommand, що мапиться у UserCommandDto

Таблиця 3.4 – Метод UpdateUserCommandAsync

Назва	UpdateUserCommandAsync
Вхідні дані	UserCommandDto
Вихідні параметри	Task
Опис роботи	Відбувається виклик методу UpdateUserCommandAsync класу-процесора, у який передається об’єкти типу UserCommandDto з вхідних даних, який парситься у UserCommand та оновлює дані користувацьких голосових команд відповідно до ідентифікатора вхідних даних.

Таблиця 3.5 - Метод AddUserCommandAsync

Назва	AddUserCommandAsync
Вхідні дані	UserCommandDto
Вихідні параметри	Task
Опис роботи	Відбувається виклик методу AddUserCommandAsync класу-процесора, у який передається об’єкти типу UserCommandDto з вхідних даних, який парситься у UserCommand та додає нові дані користувацьких голосових команд.

Таблиця 3.6 - Метод DeleteUserCommandAsync

Назва	DeleteUserCommandAsync
Вхідні дані	String
Вихідні параметри	Task

Продовження таблиці 3.6

Опис роботи	Відбувається виклик методу DeleteUserCommandAsync класу-процесора, у який передається ідентифікатор типу String з вхідних даних, який видаляє дані користувацьких голосових команд відповідно до ідентифікатора вхідних даних.

3.3.3 Моделювання та аналіз модулю обробки голосових команд

Таблиця 3.7 - Метод AudioToTextAsync

Назва	AudioToTextAsync
Вхідні дані	audio
Вихідні параметри	string
Опис роботи	Вхідним параметром виступає аудіо інформація з мікрофону користувача, що перетворюється у librosa.feature.mfcc, що далі передається натренованій моделі для передбачення, тобто визначення текстової інформації аудіо файлу, та повертає результуюче значення.

Таблиця 3.8 - Метод TokenizeTextAsync

Назва	TokenizeTextAsync
Вхідні дані	String
Вихідні параметри	IEnumerable<Token>

Продовження таблиці 3.8

Опис роботи	Текстові вхідні дані передаються в обробник, який формує з них набір токенів, відповідно до отриманих даних та повертає результуючу колекцію цих токенів.

Таблиця 3.9 - Метод SearcForCommandAsync

Назва	SearcForCommand
Вхідні дані	String
Вихідні параметри	UserCommandDto
Опис роботи	Текстові вхідні дані можуть бути ключем користувацької голосової команди, тож передаються в обробник, що викликаж метод GetUserCommandByKeyAsync для пошуку співпадінь, у разі відсутності команди, буде викликано TokenizeTextAsync, а потім відбудеться виклик методу GenerateCommandAsync, що створить команду на основі токенів вхідного рядка.

Таблиця 3.10 - Метод GenerateCommandAsync

Назва	GenerateCommandAsync
Вхідні дані	IEnumerable<Token>
Вихідні параметри	UserCommandDto
Опис роботи	Метод приймає колекцію об’єктів типу Token, на основі яких відбудеться формування команди, що повертається з методу.

3.3.4 Моделювання та аналіз модулю виконання команд

Таблиця 3.11 - Метод RunCommandAsync

Назва	RunCommandAsync
Вхідні дані	UserCommandDto
Вихідні параметри	Task
Опис роботи	Метод приймає команду як вхідні дані, та на основі її параметрів формує команду-запит, що буде виконуватись у як консольна команда операційної системи.

3.3.5 Моделювання та аналіз модулю локалізацій

Таблиця 3.12 - Метод AddLocalizationAsync

Назва	AddLocalizationAsync
Вхідні дані	string[]
Вихідні параметри	Task
Опис роботи	Метод приймає набір шляхів до файлів локалізації як вхідні дані, та на основі параметрів додає локалізації до системи.

Таблиця 3.13 - Метод ChangeLocalizationAsync

Назва	ChangeLocalizationAsync
Вхідні дані	Int
Вихідні параметри	Task

Продовження таблиці 3.13

Опис роботи	Метод приймає ідентифікатор локалізації, та встановлює цю локалізацію, як оброблювану системою.

3.4 Конструювання програмного забезпечення

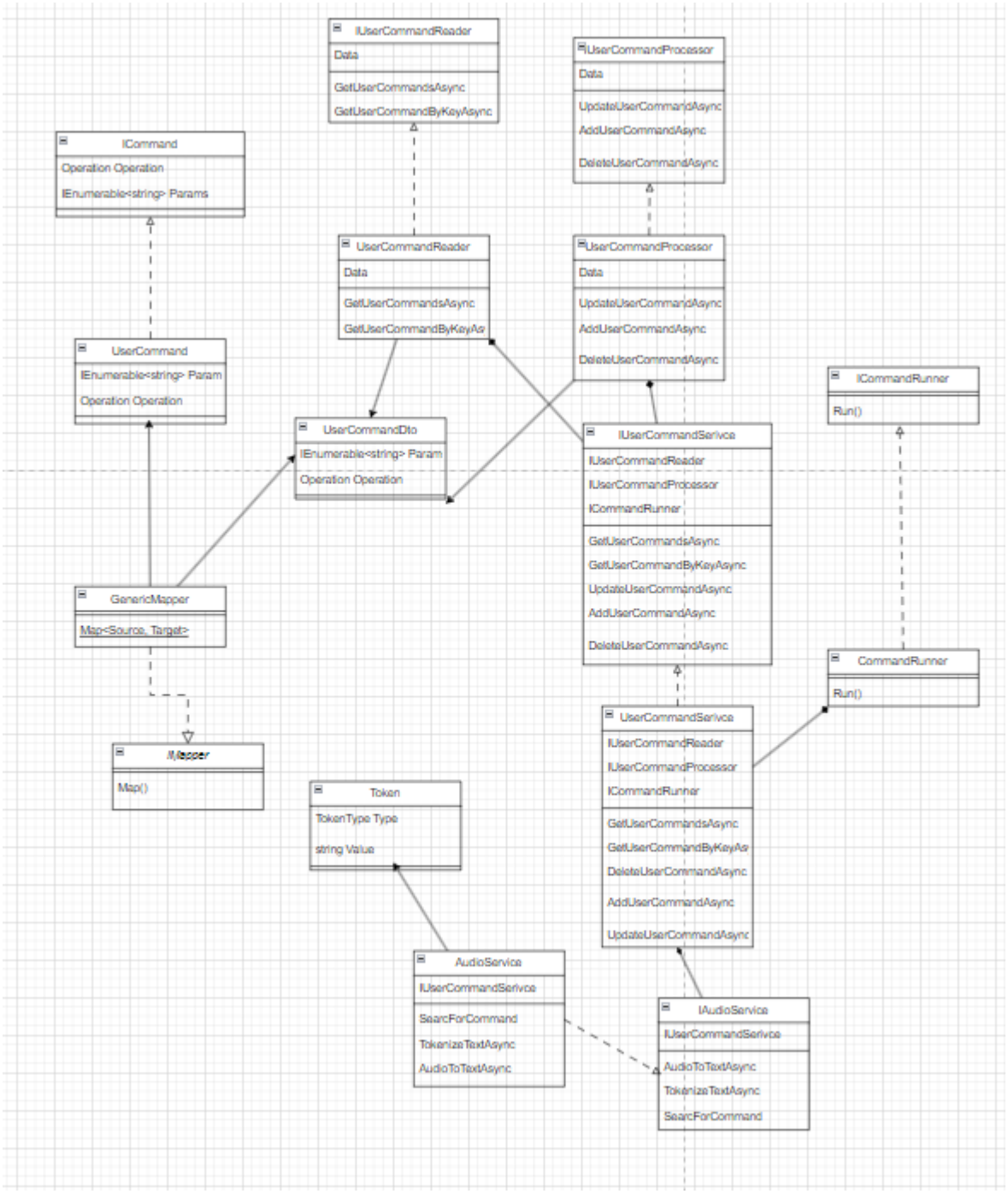


Рисунок 3.1 – Діаграма класів

На основі наданої діаграми класів можна отримати перелік основних класів та їх функціоналу, а також зв’язки між ними. Інтерфейс IMapper та клас GenericMapper описують функціонал для перетворення сутностей у Dto та навпаки. Клас Token відповідає за опис параметричної одиниці у користувацькій команді. ICommand та UserCommand описують сутність користувацької команди, що зберігається у налаштуваннях та використовується Runner’ом, IAudioService та AudioService описують логіку взаємодії голосових команд з системою та їх обробку та перетворення у командні сутності. UserCommandService описує логіку взаємодії користувача з користувацькими налаштуваннями та діями з ними, такими як: зчитування, додавання, видалення та редагування.

Висновки до розділу

В даному розділі проведено моделювання та аналіз компонентів проєкту. Презентовано архітектуру та наведено діаграми класів з поясненням їх компонентів.

4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Аналіз якості ПЗ

Програмне забезпечення містить критичні модулі обробки інформації і потребує детального тестування, щоб гарантувати, що додаток не вийде з ладу в майбутньому. Для тестування було обрано автоматизований підхід та використано фреймворк nUnit.

У ході розробки програмного забезпечення використано методологію TDD (Test-Driven Development). Це означає, що специфікація модуля визначається у вигляді тестових кейсів, в яких написані модульні та інтеграційні тести для підтримки всього процесу розробки.

Юніт-тестування - це тип тестування, який фокусується на найменших компонентах (класах і методах), які можуть бути протестовані ізольовано один від одного. Цей тип тестування повинен бути незалежним від середовища, в якому він виконується, простим в обслуговуванні, зрозумілим і надійним.

Інтеграційне тестування має на меті перевірити взаємодію між різними модулями програми, які вже були протестовані на етапі модульного тестування.

Тести, описані в цьому розділі, називаються тестуванням білого ящика, оскільки вони виконуються з повним доступом до вихідного коду і можливістю його модифікації. Таку модель тестування можна починати на ранній стадії розробки, не чекаючи реалізації користувацького інтерфейсу, а завдяки повному доступу до кода програми можна проводити більш кваліфіковані тести, охоплюючи більшу кількість сценаріїв роботи програми.

Не менш важливим аспектом тестування серверів є тестування стабільності (також відоме як тестування стійкості). Це робиться для того, щоб переконатися, що всі можливі помилки у використанні програми повністю усунені для підтримки стабільної та безперебійної роботи.

4.2 Опис процесів тестування

Виходячи з методології Test-Driven Development, у таблиці 2.1 представлено складений перелік тест-кейсів, які стали основою для написання подальших тестів функціоналу.

Таблиця 4.1 – Перелік тест-кейсів

Назва	Мета
Тест-кейс 1	Перевірити коректність роботи модулю тренування моделі нейронної мережі
Тест-кейс 2	Перевірити коректність роботи методу GetUserCommandsAsync
Тест-кейс 3	Перевірити коректність роботи методу GetUserCommandByKeyAsync
Тест-кейс 4	Перевірити коректність роботи методу UpdateUserCommandAsync
Тест-кейс 5	Перевірити коректність роботи методу AddUserCommandAsync
Тест-кейс 6	Перевірити коректність роботи методу DeleteUserCommandAsync
Тест-кейс 7	Перевірити коректність роботи методу AudioToTextAsync
Тест-кейс 8	Перевірити коректність роботи методу TokenizeTextAsync
Тест-кейс 9	Перевірити коректність роботи методу SearcForCommand
Тест-кейс 10	Перевірити коректність роботи методу GenerateCommandAsync

Продовження таблиці 4.1

Тест-кейс 11	Перевірити коректність роботи методу RunCommandAsync
Тест-кейс 12	Перевірити коректність роботи методу AddLocalizationAsync
Тест-кейс 13	Перевірити коректність роботи мапера CommandMapper
Тест-кейс 14	Перевірити коректність роботи методу ChangeLocalization

4.3 Опис контрольного прикладу

В даному підрозділі наведено більш детально описані тести, що були проведені та їх результати. Вони представлені у таблицях 2.2-2.21.

Таблиця 4.2 – Тест-кейс 1

Мета	Перевірити коректність роботи модулю тренування моделі нейронної мережі
Вхідні дані	Файли для навчання моделі нейронної мережі
Кроки виконання	– До методу Train передати файли для навчання моделі нейронної мережі – Перевірити чи Word Error Rate (WER) не перевищує 7%
Очікуваний результат	Значення Word Error Rate менше рівне 7%
Фактичний результат	Значення Word Error Rate дорівнює 6.3%

Таблиця 4.3 – Тест-кейс 2

Мета	Перевірити коректність роботи методу GetUserCommandsAsync
Вхідні дані	Екземпляр класу UserCommandService
Кроки виконання	– Викликати метод GetUserCommandsAsync – Звірити отримані на вихідні данні з очікуваним значенням результату
Очікуваний результат	Отримана з методу колекція користувацьких налаштувань повинна бути рівна очікуваним значенням тест-кейсу
Фактичний результат	Отримана з методу колекція користувацьких налаштувань рівна очікуваним значенням тест-кейсу

Таблиця 4.4 – Тест-кейс 3

Мета	Перевірити коректність роботи методу GetUserCommandByKeyAsync
Вхідні дані	Екземпляр класу UserCommandService, ідентифікатор Id
Кроки виконання	– До методу GetUserCommandByKeyAsync передати Id – Звірити отриманий на виході об’єкт з очікуваним об’єктом UserCommandDto
Очікуваний результат	Отриманий з методу UserCommandDto має бути рівний очікуваному UserCommandDto;
Фактичний результат	Отриманий з методу UserCommandDto рівний очікуваному UserCommandDto;

Таблиця 4.5 – Тест-кейс 4

Мета	Перевірити коректність роботи методу UpdateUserCommandAsync
Вхідні дані	Екземпляр класу UserCommandService, об’єкт типу UserCommandDto
Кроки виконання	– До методу UpdateUserCommandAsync передати UserCommandDto – Перевірити оновлення даних
Очікуваний результат	Дані успішно оновлено
Фактичний результат	Дані успішно оновлено

Таблиця 4.6 – Тест-кейс 5

Мета	Перевірити коректність роботи методу AddUserCommandAsync
Вхідні дані	Екземпляр класу UserCommandService, об’єкт типу UserCommandDto
Кроки виконання	– До методу AddUserCommandAsync передати UserCommandDto – Перевірити додавання даних
Очікуваний результат	Дані успішно додано
Фактичний результат	Дані успішно додано

Таблиця 4.7 – Тест-кейс 6

Мета	Перевірити коректність роботи методу DeleteUserCommandAsync
Вхідні дані	Екземпляр класу UserCommandService, ідентифікатор Id
Кроки виконання	– До методу UpdateUserCommandAsync передати UserCommandDto – Перевірити видалення даних
Очікуваний результат	Дані успішно видалено
Фактичний результат	Дані успішно видалено

Таблиця 4.8 – Тест-кейс 7

Мета	Перевірити коректність роботи методу AudioToTextAsync
Вхідні дані	Екземпляр класу AudioService, аудіо дані голосової команди Data
Кроки виконання	– До методу AudioToTextAsync передати Data – Перевірити відповідність вихідних даних методу з очікуваним результатом типу string
Очікуваний результат	Отриманий з методу результат має бути рівний очікуваному результату
Фактичний результат	Отриманий з методу результат рівний очікуваному результату

Таблиця 4.9 – Тест-кейс 8

Мета	Перевірити коректність роботи методу TokenizeTextAsync
Вхідні дані	Екземпляр класу AudioService, дані Data типу string
Кроки виконання	– До методу TokenizeTextAsync передати вхідні дані Data – Звірити отриманий на виході набір токенів з очікуваним результатом
Очікуваний результат	Отриманий з методу набір токенів має бути рівний очікуваному набору токенів
Фактичний результат	Отриманий з методу набір токенів рівний очікуваному набору токенів

Таблиця 4.10 – Тест-кейс 9

Мета	Перевірити коректність роботи методу SearcForCommand
Вхідні дані	Екземпляр класу AudioService, дані Data типу string
Кроки виконання	– До методу SearcForCommand передати Data – Перевірити наявність коректного результату виклику та обробки методу на існуючу команду – До методу SearcForCommand передати Data – Перевірити наявність коректного результату виклику та обробки методу на неіснуючу команду
Очікуваний результат	Отриманий з методу UserCommandDto має бути рівний очікуваному UserCommandDto; Отриманий результат має бути рівний null

Продовження таблиці 4.10

Фактичний результат	Отриманий з методу UserCommandDto рівний очікуваному UserCommandDto; Отриманий результат рівний null

Таблиця 4.11 – Тест-кейс 10

Мета	Перевірити коректність роботи методу GenerateCommandAsync
Вхідні дані	Екземпляр класу AudioService, колекція токенів Tokens
Кроки виконання	– До методу GenerateCommandAsync передати Tokens – Звірити отриманий на виході об’єкт з об’єктом очікуваним результатом Command
Очікуваний результат	Отриманий з методу Command має бути рівний очікуваному Command
Фактичний результат	Отриманий з методу Command рівний очікуваному Command

Таблиця 4.12 – Тест-кейс 11

Мета	Перевірити коректність роботи методу RunCommandAsync
Вхідні дані	Екземпляр класу UserCommandProcessor, команда UserCommandDto
Кроки виконання	– До методу RunCommandAsync передати UserCommandDto – Перевірити результат виконання команли

Продовження таблиці 4.12

Очікуваний результат	Команду успішно виконано
Фактичний результат	Команду успішно виконано

Таблиця 4.13 – Тест-кейс 12

Мета	Перевірити коректність роботи методу AddLocalizationAsync
Вхідні дані	Екземпляр класу LocalizationService, файли локалізації
Кроки виконання	– До методу LocalizationService передати файли локалізації – Перевірити наявність можливості використання нової локалізації
Очікуваний результат	Локалізацію успішно додано
Фактичний результат	Локалізацію успішно додано

Таблиця 4.14 – Тест-кейс 13

Мета	Перевірити коректність роботи мапера CommandMapper
Вхідні дані	UserCommandDto і об’єкт класу UserCommand
Кроки виконання	– До методу Map передати UserCommandDto – Звірити отриманий на виході об’єкт з об’єктом UserCommand

Продовження таблиці 4.14

Очікуваний результат	Отриманий з методу UserCommand має бути рівний очікуваному UserCommand; отриманий з методу UserCommandDto має бути рівний UserCommandDto
Фактичний результат	Отриманий з методу UserCommand рівний очікуваному UserCommand; отриманий з методу UserCommandDto рівний UserCommandDto

Таблиця 4.15 – Тест-кейс 14

Мета	Перевірити коректність роботи методу ChangeLocalization
Вхідні дані	Екземпляр класу LocalizationService, ідентифікатор локалізації Id
Кроки виконання	– До методу LocalizationService передати Id – Перевірити використовувану локалізацію
Очікуваний результат	Локалізацію успішно змінено
Фактичний результат	Локалізацію успішно змінено

Висновки до розділу

В даному розділі проведено тестування розрозбленого програмного забезпечення. Наведено перелік тест-кейсів та їх детальний опис разом із результатами проведеного тестування. Створене програмне забезпечення розроблено якісним та відмовостійким, а також повністю задовільняє визначені функціональні вимоги.

5 МАРКЕТИНГОВИЙ АНАЛІЗ СТАРТАП-ПРОЄКТУ

5.1 Опис ідеї проєкту

Таблиця 5.1 — Опис ідеї стартап-проєкту

Зміст ідеї	Напрямки застосування	Вигоди для користучача
Розробка програмного забезпечення для налаштування та виконання голосових команд у операційній системі Windows	Пересічні користувачі	Використання операційної систеими за допомогою голосового управління буде зручнішим
	ІТ-спільнота	

Таблиця 5.2 — Опис ідеї стартап-проєкту

№	Техніко-економічні характери-тики ідеї	Продукція конкурентів				W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проєкт	Bixby	Siri	Cortana			
1	Можливість налаштування голосових команд	+	+	+	-			+
2	Можливість вибору локалізації	+	+	+	+			+

Продовження таблиці 5.2

3	Можливість додавання користувацьких локалізацій	+	-	-	-	+		
4	Незалежність від підключення до мережі інтернет	+	-	-	-			+
5	Наявність аудіовідповіді від застосунку	+	+	+	-		+	
6	Інтеграція зі сторонніми сервісами	-	+	+	-		+	

Після аналізу технічних та економічних факторів, встановлено міцну основу для розвитку конкурентоспроможності нашого стартапу. Виявлено що наш продукт має ряд переваг у своїй категорії, включаючи ті, які вже мають наші конкуренти, а також унікальні. Водночас, функції, які не включено, здебільшого не є критично важливими або їх значення залишається невизначеним.

5.2 Технологічний аудит ідеї проєкту

Таблиця 5.3 — Технологічна здійсненність ідеї проєкту

№	Ідея проєкту	Технології її реалізації	Наявність технологій	Доступність технологій

Продовження таблиці 5.3

1	Створення моделі нейронної мережі для автоматичного розпізнавання мовлення	Python, Tensorflow, Librosa, Common Voice	Наявна	Доступна
2	Розробка програмного забезпечення для налаштування та виконання голосових команд	.Net, WPF,TensorFlow, Entitiy Framework	Наявна	Доступна

5.3 Аналіз ринкових можливостей запуску стартап-проєкту

Таблиця 5.4 — Попередня характеристика потенційного ринку

№	Показники стану ринку	Характеристика
1	Кількість головних гравців, од	3
2	Загальний обсяг продаж, грн./ум.од	Невідома
3	Динаміка ринку	Зростає
4	Наявність обмежень для входу	Відсутні
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6	Середня норма рентабельності в галузі або по ринку, %	Невідомо

Висновок: враховуючи кількість головних гравців по ринку, зростаючу динаміку ринку, невелику кількість конкурентів та середню норму

рентабельності можна зробити висновок, що на даний момент, ринок для входження стартап-продукту є привабливим.

Таблиця 5.5 — Характеристика потенційних клієнтів стартап-проєкту

№	Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці цільових груп клієнтів	Вимоги споживачів до товару
1	Програмне забезпечення для налаштування та виконання голосових команд у операційній системі Windows	Пересічні користувачі	Відсутні	Зручність, точність, швидкість, відмовостійкість
2		IT-спільнота	Відсутні	

Таблиця 5.6 — Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Конкуренти	Наявність конкурентів котрі надають схожі рішення	Розробка унікальних характеристик товару; Надання ліцензій на обслуговування
2	Кошти на розробку та підтримку продукту	Закінчення грошей та недостатнє фінансування	Залучення додаткових інвесторів, мотивація роботи на перспективу; Ітеративна розробка продукту задля покрокового виведення продукту на ринок та отримання відповіді користувачів

Таблиця 5.7 — Фактори можливостей

№	Фактор	Зміст можливості	Можлива реакція компанії
1	Новий продукт	Вихід на ринок, Зменшення монополії, Надання нових рішень у сфері	Розробка нової функціональності; Вихід нової продукції на ринок; Надання різноманітних типів ліцензій в залежності від потреб користувача \ замовника.
2	Вихід аналогу	Надати продукт з певними характеристиками та можливостями що відсутні у компаній конкурентів	Аналіз ринку та користувачів задля задоволення їх потреб та надання функціональності у найкоротші строки за ціну, котра є дешевшою ніж у продуктів-замінників.
3	Зворотній зв’язок від користувачів	Можливість отримання необхідної інформації для вдосконалення продукту	Наявність вхідних даних та реакція на них з боку команди розробників задля задоволення потреб та бажань кінцевих користувачів системи кешування даних.
4	Грошова винагорода за рекламу	При достатньому попиту на систему кешування даних можлива комерціалізація продукту на основі реклами задля отримання грошової винагороди для подальшого розвиту продукту та оплати заробітної плати працівникам	Точкова комерціалізація продукту; Введення реклами; Ведення додаткових коштів у проєкт задля його подальшого розвитку.

Таблиця 5.8 — Ступеневий аналіз конкуренції на ринку

№	Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1	Тип конкуренції: монополістична	Товар від кожної компанії на ринку, являється недосконалим замінником товару, реалізованого іншими фірмами; На ринку є умови для входу та виходу; Ціна корелює між суперниками;	Розробка продукту з характеристиками, які покривають сфери вживання що не покривають інші товари-замінники; Кореляція цін у відповідності до товарів замінників; Різни типи ліцензій.
2	Рівень конкурентної боротьби: світовий	Всі продукти замінники розроблялись інтернаціональними командами з різних куточків світу, продукти не належать до певної держави, а належать команді розробників	Вихід на ринок збуту продукту з клієнто-необхідною функціональністю; Налагодження маркетингу на основних Інтернет ресурсах задля охоплення великої кількості потенційних користувачів; Надання бета-версій продукту.
3	Галузева ознака: внутрішньогалузева	Даний тип продукту діє в одній галузі економіки, компанії виробляють і реалізують однакові товари, що задовольняють одну й ту саму потребу	Надання зручного, інтуїтивно зрозумілого інтерфейсу; Підтримка всім відомих методів взаємодії з середовищем розробки; Наявність документації та он-лайн підтримки.

Продовження таблиці 5.8

4	Конкуренція за видами товарів: товарно-видова	Дана конкуренція – конкуренція між товарами одного виду.	Впровадження функціональності яка відсутня у товарів-замінників; Спрощення інтерфейсів; Надання підтримки.
5	Характер конкурентних переваг: цінова та не цінова	Цінові переваги – точкова комерціалізація; Не цінова – надання функціональності, що відсутня у товарах-замінниках.	Надання платних ліцензій лише на критично важливу функціональність для клієнта з певним строком підтримки, що зазначена у відповідній ліцензії; Впровадження унікальної функціональності.
6	За інтенсивністю: марочна	Наявність унікального знаку що відрізняє даний продукт від продуктів-замінників	Впровадження власної назви та власного знаку.

Таблиця 5.9 — Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Cortana	Siri, Bixby	Набір функціоналу	IT компанії	Відсутні
Висновки	Існують	Дані відсутні	Чим більше функціоналу пропонує постачальник, тим краще рішення	Клієнти хочуть отримати максимально зручне та ефективне рішення	Відсутні

Проаналізувавши можливості роботи на ринку з огляду на конкурентну ситуацію можна зробити висновок: оскільки кожний з існуючих продуктів не впливає у великій мірі на поточну ситуацію на ринку в цілому, кожний з існуючих продуктів має свою специфічну сферу використання та свої позитивні та негативні сторони щодо рішення певних типів задач, то робота та вихід на даний ринок є можливою і реалізованою задачею.

Для виходу на ринок продукт повинен мати функціонал що відсутній у продуктів-аналогів, повинен задовольняти потреби користувачів, мати необхідний та достатній функціонал з конфігурування, підтримку зі сторони розробників та можливість розробки спеціального функціоналу за відповідною ліцензією.

Таблиця 5.10 — Обґрунтування факторів конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування
1	Швидкість та ефективність обробки	Програмне забезпечення забезпечує швидку обробку користувацьких запитів, що підвищує ефективність його використання
2	Простота використання	Інтуїтивно зрозумілий інтерфейс спрощує навчання та використання
3	Залежність від підключення до мережі Інтернет	Програмне забезпечення не втрачає доступ до функціоналу за відсутності підключення до мережі Інтернет
4	Інтеграція з зовнішніми додатками	Вбудовані засоби для інтеграції з іншими популярними сервісами, дозволяючи користувачам легко об'єднувати різні технологічні рішення.
5	Можливість персоналізації програмного забезпечення	Програмне забезпечення надає інструментарій для внесення змін користувачем, для його зручнішого використання.

Продовження таблиці 5.10

6	Підтримка спільноти та оновлення	Активна спільнота та регулярні оновлення забезпечують тривалу актуальність інструменту

Таблиця 5.11 — Порівняльний аналіз сильних та слабких сторін стартап-проєкту

№	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з запропонованим						
			-3	-2	-1	0	+1	+2	+3
1	Швидкість та ефективність обробки	14					+		
2	Простота використання	17			+				
3	Залежність від підключення до мережі Інтернет	19	+						
4	Інтеграція з зовнішніми додатками	5							+
5	Можливість персоналізації програмного забезпечення	13			+				
6	Підтримка спільноти та оновлення	7					+		

Таблиця 5.12 — SWOT аналіз стартап-проєкту

Сильні сторони (S): – Актуальність технології для сучасного ринку. – Важливість такого типу програмного забезпечення для людей з обмеженими можливостями – Можливість додавання користувацьких локалізацій – Низька залежність від підключення до мережі інтернет	Слабкі сторони (W): – Обмеження ресурсів на ранньому етапі розробки. – Низький рівень інтеграції з зовнішніми застосунками та сервісами. – Платформозалежність.
Можливості (O): – Розширення ринку за рахунок виходу на інші платформи – Партнерство з іншими розробниками програмного забезпечення. – Розширення інтеграції за сторонніми сервісами та програмним забезпеченням – Розширення списку доступних вбудованих локалізацій	Загрози (T): – Стрімке оновлення технологій та поява нових конкурентів. – Відсутність оновлень відкритих даних для навчання моделей автоматичного розпізнавання мовлення – Ризики, пов'язані з захистом інтелектуальної власності в рамках розробки ПЗ.

Таблиця 5.13 — Альтернативи ринкового впровадження стартап-проєкту

№	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Безкоштовне надання певного функціоналу у користування споживачам на обмежений термін	Головний ресурс – люди, даний ресурс - наявний	2-3 місяці

Продовження таблиці 5.13

2	Реклама	Залучення власних коштів для реклами товару	1-2 місяці
3	Написання статей та опис товару на відомих ресурсах	Головний ресурс – час, даний ресурс - наявний	2-3 тижні
4	Презентація товару на хакатонах й інших IT заходах	Ресурс – час та гроші для участі, наявні	1-3 місяці

5.4 Розроблення ринкової стратегії проєкту

Для розробки ринкової стратегії, потрібно визначити охоплення ринку за допомогою визначення груп потенційних споживачів. В таблиці 5.15 оглянути вибір цільових груп потенційних користувачів.

Відповідно до проведеного аналізу можна зробити висновок, що підходящою цільовою групою для розповсюдження даного програмного продукту є пересічні користувачі, та IT компанії в цілому, які використовують ОС Windows. Відповідно до стратегії охоплення ринку збуту товару обрано стратегію масового маркетингу, оскільки для користувачів надається стандартизований продукт з можливістю розширення функціональності за домовленістю (відповідно до ліцензії).

Таблиця 5.14 — Вибір цільових груп потенційних споживачів

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Пересічні користувачі	Висока готовність	Середній попит	Низька конкуренція	Легко
2	ІТ спільнота	Висока готовність	Середній попит	Низька конкуренція	Легко
Які цільові групи обрано: Пересічні користувачі, ІТ спільнота					

Таблиця 5.15 — Визначення базової стратегії розвитку

Обрана альтернатива розвитку проєкту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Надання функціональності що відсутня у товарів-замінників, підтримка клієнтів	Проведення реклами, освітлення унікальної функціональності через інтернет ресурси та інші канали, контакт напряму з споживачами; формування лояльності і прихильності споживачів	Зниження ступеню замінності товару; Прихильність клієнтів; Відмітні властивості товару; Відмітні характеристики товару;	Стратегія диференціації

Таблиця 5.16 — Визначення базової стратегії конкурентної поведінки

Чи є проєкт «першопрохідцем» на ринку	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, які?	Стратегія конкурентної поведінки
Ні, оскільки є товари-замінники, але дані товари замінники не мають деякого необхідного функціоналу	Так, ціль компанії знайти нових споживачів та, частково, забрати існуючих у конкурентів задля задоволення потреб останніх	Компанія частково копіює характеристики товару конкурента, основна ціль компанії розробка нового унікального функціоналу, з підтримкою основного функціоналу конкурентів	Стратегія заняття конкурентної ніші

Таблиця 5.17 — Визначення стратегії позиціонування

№	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проєкту	Вибір асоціацій, які мають сформувати комплексну позицію власного проєкту
1	Гнучкість у налаштуванні для задоволення специфічних потреб користувачів	Надання інтегрованих рішень для підвищення ефективності використання та гнучкості налаштувань.	Інтуїтивність у використанні.	Універсальність та гнучкість.

Відповідно до проведеного аналізу можна зробити висновок, що стартап-компанія вибирає як базову стратегію розвитку – стратегію диференціації, як базову стратегію конкурентної поведінки – стратегію заняття конкурентної ніші.

5.5 Розроблення маркетингової програми стартап-проєкту

Першим кроком до розроблення маркетингової програми стартап-проєкту є формування маркетингової концепції товару, який отримує споживач. Для цього у таблиці 5.18 підсумуємо результати попереднього аналізу конкурентоспроможності товару

Таблиця 5.18 — Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Прискорення роботи з ОС	Зниження часу виконання маніпуляцій з ОС	Тісна інтеграція з операційною системою
2	Інтуїтивний інструмент для голосового керування	Покращення досвіду користування ОС за допомогою голосового управління	Гнучкість налаштування користувацьких команд та можливість додавання мовних пакетів

Таблиця 5.19 — Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові
1. Товар за задумом	Програмне забезпечення для налаштування та виконання голосових команд в операційній системі Windows

Продовження таблиці 5.19

2. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх/Тл/Е/Ор
	Якість	Нм	Тл
	Зрозумілість	М	Е
	Швидкість роботи	Нм	Тх
	Універсальність	М	Тх
	Спосіб взаємодії: десктопний додаток		
	Якість: ISO/IEC 25010 25000:2014		
3. Товар із підкріпленням	Ознайомлення клієнта з усіма сутностями та функціоналу проєкта для зрозумілого користування		

Таблиця 5.20 — Визначення меж встановлення ціни

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
0$	0$	Пересічні користувачі від 1000$ за рік	Не визначена
		ІТ спільнота від 1000$ за рік	

Таблиця 5.21 — Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Задоволення потреб людей, що користуються голосовим управлінням	Надати можливість встановити інструмент	Від виробника до споживача	Прямий канал збуту

Таблиця 5.22 — Концепція маркетингових комунікацій

№	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Пересічний користувач, який хоче спростити та пришвидшити роботу з операційною системою за рахунок використання голосового управління	Інтернет, електронна пошта, мессенджери, живе спілкування	Простота використання, доступність та зручність як головні властивості продукту.	Підкреслити простоту та доступність продукту та переваги, що він надає при роботі з системою	Створення образу інструменту, який є необхідним помічником у повсякденній роботі будь-якої людини, з акцентом на зручність та простоту.

Як результат було створено ринкову (маркетингову) програму, що включає в себе визначення ключових переваг концепції потенційного товару, опис моделі товару, визначення меж встановлення ціни, формування системи збуту та концепцію маркетингових комунікацій.

Висновки до розділу

В п’ятому розділі описано стратегії та підходи з розроблення стартап-проєкту, визначено наявність попиту, динаміку та рентабельність роботи ринку, як висновок було вказано що існує можливість ринкової комерціалізації проєкту. Розглянувши потенційні групи клієнтів, бар’єри входження, стан конкуренції та конкурентоспроможність проєкту було встановлено що проєкт є перспективним. Розглянуто та вибрано альтернативу впровадження стартап-проєкту та доведено доцільність подальшої імплементації проєкту.

ВИСНОВКИ

Магістерська десертація присвячена розробці програмного забезпечення для налаштування та виконання користувацьких голосових команд в операційній системі Windows.

Основною метою роботи є підвищення варіативності використання програмного забезпечення для виконнання голосових команд в операційній системі Windows з можливістю гнучкого налаштування користувацьких команд.

У процесі виконання досліджень предметної області проведено аналіз аналогічного та схожого програмного забезпечення, визначено їх ключові переваги та недоліки, наприклад, висока залежність аналогів від наявності підключення до мережі Інтернет та відсутність можливості гнучкого налаштування голосових команд, створених користувачем.

У десертації описано процес розробки десктопного програмного забезпечення мовою програмування C# для налаштування та виконання користувацьких голосових команд в операційній системі Windows. Запропоноване рішення повністю задовільняє вимоги визначені у ході виконання досліджень. У результаті розробки програмного забезпечення також було створено модель нейронної мережі для автоматичного розпізнавання мовленя, яку було використано у модулі обробки користувацьких команд.

Проведено ретельне модульне, інтеграційне та системне тестування компонентів створеного програмного забезпечення, а також бета-тестування з результатами опитування користувачів.

Проведено: маркетинговий аналіз стартап-проєктів, опис проєктної ідеї, технічний аудит проєктних ідей та аналіз ринкових можливостей для запуску стартап-проєктів, розробку ринкової стратегії проєкту та маркетингової програми стартап-проєкту.

Наукова новизна: вдосконалення існуючих підходів до створення голосових асистентів та розробка відповідного програмного забезпечення.

Результатом цієї роботи є розробка програмного забезпечення для налаштування та виконнання голосових команд в операційній системі Windows на основі розпізнавання мовлення, яке може бути використане пересічними користувачами операційних систем для зручної експлуатації під час роботи з персональним комп’ютером.

Результати роботи пройшли апробацію на науково-практичній конференції «SoftTech-2023» та опубліковані у матеріалах конференції.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Alexa Voice Service [Електронний ресурс] – Режим доступу: https://developer.amazon.com/en-US/docs/alexa/alexa-voice-service/api-overview.html

2) Siri [Електронний ресурс] – Режим доступу: https://www.apple.com/siri/

3) BixBy [Електронний ресурс] – Режим доступу: https://www.samsung.com/global/galaxy/apps/bixby/

4) Russell, Stuart J.; Norvig, Peter (2003), Artificial Intelligence: A Modern Approach (2nd ed.), Upper Saddle River, New Jersey: Prentice Hall, ISBN 0-13-790395-2.

5) Poole, David; Mackworth, Alan; Goebel, Randy (1998). Computational Intelligence: A Logical Approach. New York: Oxford University Press. ISBN 978-0-19-510270-3. Archived from the original on 26 July 2020. Retrieved 22 August 2020.

6) Pirani, Giancarlo, ed.: Advanced algorithms and architectures for speech understanding. Vol. 1. Springer Science & Business Media, 2013. ISBN 978-3-642-84341-9

7) Lawrence R. Rabiner, Ronald W. Schafer: Digital Processing of Speech Signals, 1978, ISBN 0-13-213603-1.

8) Lawrence R. Rabiner, Biing-Hwang Juang Juang: Fundamentals of Speech Recognition, 1993, ISBN 0-13-015157-2.

9) Common Voice [Електронний ресурс] – Режим доступу: https://commonvoice.mozilla.org/en/about

10) Common Voice Datasets [Електронний ресурс] – Режим доступу: https://commonvoice.mozilla.org/en/datasets

11) .NET документація [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/ru-ru/dotnet/

12) What is Windows Presentation Platform [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/overview/?view=netdesktop-8.0

13) Python [Електронний ресурс] – Режим доступу: https://docs.python.org/

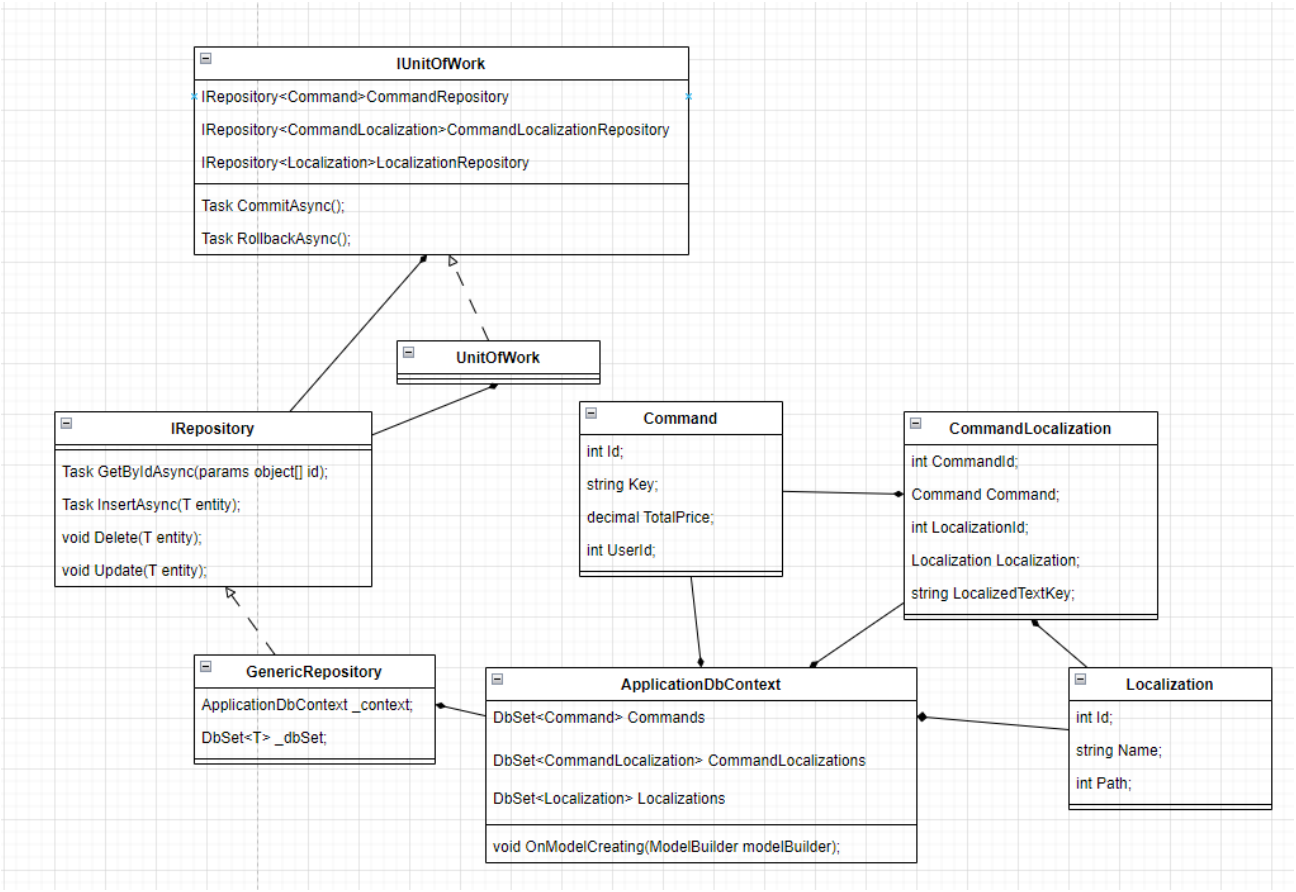
14) Tensorflow [Електронний ресурс] – Режим доступу: https://www.tensorflow.org/

15) Rider [Електронний ресурс] – Режим доступу: https://www.jetbrains.com/rider/

16) Pycharm [Електронний ресурс] – Режим доступу: https://www.jetbrains.com/pycharm/

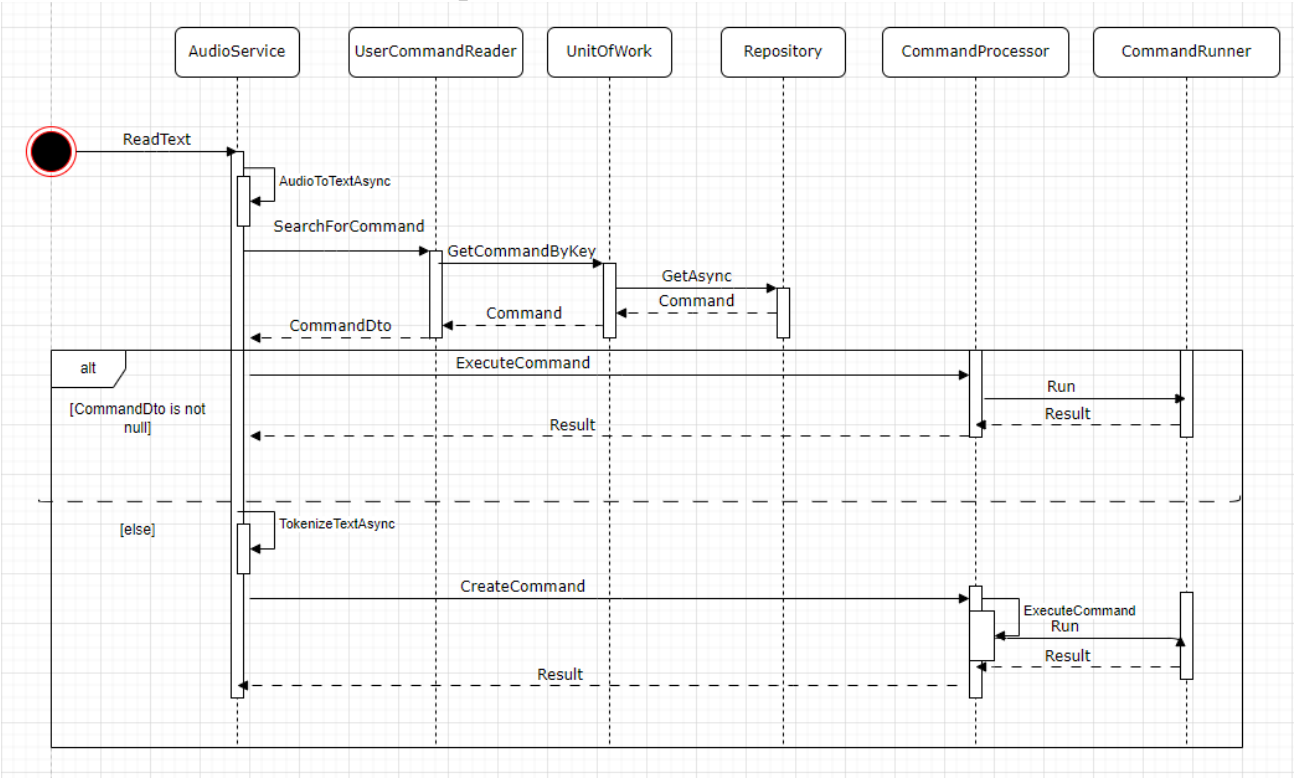
ДОДАТОК А

Діаграма шару доступу даних



ДОДАТОК Б

Діаграма послідовності системи



ДОДАТОК В

Лістинг програмного коду

Файл Train.py

```
from collections import OrderedDict
import librosa
import numpy as np
import torch
from torch.utils.data import Dataset, DataLoader
import os
from torch.nn.utils.rnn import pad_sequence



def process_transcriptions(filename, vocab):
    transcriptions = []
    names = []
    with open(filename, 'r') as file:
        for line in file:

            parts = line.split()
            audio_name, transcription = parts[0], ' '.join(parts[1:])

            sequence = text_to_sequence(transcription, vocab)

            transcriptions.append(sequence)
            names.append(audio_name)

    return transcriptions, names
```

```
def collate_fn(batch):

    audios, transcripts = zip(*batch)

    audios = [torch.squeeze(audio, 0) for audio in audios]
    audios_padded = pad_sequence(audios, batch_first=True)

    transcripts = [torch.tensor(transcript) for transcript in transcripts]
    transcripts_padded = pad_sequence(transcripts, batch_first=True)

    return audios_padded, transcripts_padded



def load_librispeech_data(root_dir, vocab):
    data = []

    for subdir, _, files in os.walk(root_dir):
        for file in files:
            if file.endswith('.txt'):

                transcriptions, names = process_transcriptions(os.path.join(subdir, file),
vocab)

                for i in range(len(transcriptions)):
                    audio_file = os.path.join(subdir, f"{names[i]}.flac")
```

```
                data.append((audio_file, transcriptions[i]))
    return data


vocab = {
    'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6, 'H': 7, 'I': 8,
    'J': 9, 'K': 10, 'L': 11, 'M': 12, 'N': 13, 'O': 14, 'P': 15, 'Q': 16,
    'R': 17, 'S': 18, 'T': 19, 'U': 20, 'V': 21, 'W': 22, 'X': 23, 'Y': 24,
    'Z': 25, ' ': 26, "'": 27
}


def text_to_sequence(text, vocab):
    return [vocab[char] for char in text]


def preprocess_audio(audio_path):
    y, sr = librosa.load(audio_path, sr=16000)

    y = y / np.max(np.abs(y))

    speed_rate = np.random.uniform(0.9, 1.1)
    y = librosa.effects.time_stretch(y, rate=speed_rate)

    mfccs = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=13, hop_length=320, n_fft=640)
    return mfccs.T
```

```
class LibriSpeechDataset(Dataset):
    def __init__(self, root_dir, vocab):
        self.data = load_librispeech_data(root_dir, vocab)
        self.vocab = vocab

    def __len__(self):
        return len(self.data)

    def __getitem__(self, idx):
        audio_file, transcript_int = self.data[idx]
        mfccs = preprocess_audio(audio_file)
        print("mfccs shape:", mfccs.shape)
        return torch.tensor(mfccs), transcript_int



class DeepSpeech2(torch.nn.Module):
    def __init__(self, input_size, hidden_size, output_size, num_layers=5):
        super(DeepSpeech2, self).__init__()

        rnn_type = torch.nn.LSTM
        self.rnns = []
        rnn_input_size = input_size
        for l in range(num_layers):
            self.rnns.append(
                (f"rnn_{l}", rnn_type(input_size=rnn_input_size, hidden_size=hidden_size,
bidirectional=True))
```

```
            )
            rnn_input_size = hidden_size * 2
        self.rnns = torch.nn.Sequential(OrderedDict(self.rnns))
        self.fc = torch.nn.Linear(rnn_input_size, output_size)

    def forward(self, x):
        for _, layer in self.rnns.named_children():
            x, _ = layer(x)
        x = self.fc(x)
        return x


train_dataset = LibriSpeechDataset("C:\\Users\\sobol\\Downloads\\train-clean-1000", vocab)
val_dataset = LibriSpeechDataset("C:\\Users\\sobol\\Downloads\\dev-clean", vocab)
train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True, collate_fn=collate_fn)
val_loader = DataLoader(val_dataset, batch_size=32, collate_fn=collate_fn)
model = DeepSpeech2(input_size=13, hidden_size=512, output_size=29)
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
criterion = torch.nn.CTCLoss(blank=28, zero_infinity=True)
CLIP_GRAD = 3.0


def evaluate(model, val_loader, criterion):
    model.eval()
    total_loss = 0.0
```

```
    with torch.no_grad():
        for batch_idx, (data, target) in enumerate(val_loader):
            output = model(data)

            target_ints = [text_to_sequence(text, vocab) for text in target]

            target_ints = [int_elem for sublist in target_ints for int_elem in sublist]
            target_ints = torch.tensor(target_ints, dtype=torch.int)



            input_lengths = torch.full(size=(len(data),), fill_value=output.size(1),
dtype=torch.long)
            target_lengths = torch.tensor([len(text) for text in target], dtype=torch.long)
            loss = criterion(output, target_ints, input_lengths, target_lengths)
            total_loss += loss.item()
    return total_loss / len(val_loader)


print("start training")
for epoch in range():
    model.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        print(f"Starting batch {batch_idx + 1} for Epoch {epoch + 1}")
        optimizer.zero_grad()
        data = data.permute(1, 0, 2)
        output = model(data)

        target_ints = [text_to_sequence(text, vocab) for text in target]
```

```
        target_ints = [int_elem for sublist in target_ints for int_elem in sublist]
        target_ints = torch.tensor(target_ints, dtype=torch.int)

        input_lengths = torch.full(size=(len(data),), fill_value=output.size(1), dtype=torch.long)
        target_lengths = torch.tensor([len(text) for text in target], dtype=torch.long)

        loss = criterion(output, target_ints, input_lengths, target_lengths)

        loss.backward()
        torch.nn.utils.clip_grad_norm_(model.parameters(), CLIP_GRAD)  # Gradient clipping
        optimizer.step()
        val_loss = evaluate(model, val_loader, criterion)
        print(f"Epoch {epoch + 1}, Validation Loss: {val_loss:.4f}")

print("end training")
```

Файл ApplicationDbContext.cs

```
using System.Runtime.Intrinsics.X86;
using Desertation.Core.Entities;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.DataEncryption;
using Microsoft.EntityFrameworkCore.DataEncryption.Providers;
using Microsoft.EntityFrameworkCore.Storage;
```

```
namespace Desertation.Core.Context;

public sealed class ApplicationDbContext : IdentityDbContext<User>

{

    private readonly IEncryptionProvider _encryptionProvider;

    private readonly AesKeyInfo _encryptionKey =
AesProvider.GenerateKey(AesKeySize.AES128Bits);

    private readonly AesKeyInfo _encryptionIv =
AesProvider.GenerateKey(AesKeySize.AES128Bits);

    public DbSet<Command> Commands { get; set; }

    public DbSet<Localization > Localizations { get; set; }

    public DbSet<CommandLocalization > CommandLocalizations { get; set; }


    protected override void OnModelCreating(ModelBuilder modelBuilder)

    {

        modelBuilder.UseEncryption(_encryptionProvider);

        base.OnModelCreating(modelBuilder);

    }


    public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
: base(options)

    {

        Database.EnsureCreated();

    }

}
```

Файл IRepository.cs

```
using System.Linq.Expressions;

namespace Desertation.Core.Interfaces;

public interface IRepository<T> where T : class
{
    Task<T> GetByIdAsync(params object[] id);

    Task<IEnumerable<T>> GetAllAsync(Expression<Func<T, bool>> filter = null,
        Func<IQueryable<T>, IOrderedQueryable<T>> orderBy = null, string[]
includeProperties = null,
        int? pageNumber = null, int? pageSize = null);

    Task InsertAsync(T entity);

    void Delete(T entity);
    void Update(T entity);
}
```

Файл IUnitOfWork.cs

```
using Desertation.Core.Entities;

namespace Desertation.Core.Interfaces;

public interface IUnitOfWork
{
    IRepository< Command > CommandRepository { get; }
```

```
    IRepository< Localization > LocalizationRepository { get; }
    IRepository< CommandLocalization> CommandLocalizationRepository { get; }
    Task CommitAsync();
    Task RollbackAsync();
}
```

Файл GenericRepository.cs

```
using System.Linq.Expressions;
using Desertation.Core.Context;
using Desertation.Core.Interfaces;
using Microsoft.EntityFrameworkCore;

namespace Desertation.Core.Repositories;

public class GenericRepository<T> : IRepository<T> where T : class
{
    private ApplicationDbContext _context;
    protected DbSet<T> _dbSet;

    public GenericRepository(ApplicationDbContext context)
    {
        _context = context;
        _dbSet = context.Set<T>();
    }

    public async Task<T> GetByIdAsync(params object[] id)
    {
        return await _dbSet.FindAsync(id);
```

```
    }

    public async Task<IEnumerable<T>> GetAllAsync(Expression<Func<T, bool>>
filter = null,
        Func<IQueryable<T>, IOrderedQueryable<T>> orderBy = null,
        string[] includeProperties = null,
        int? pageNumber = null, int? pageSize = null)
    {
        IQueryable<T> query = _dbSet;

        if (filter != null)
        {
            query = query.Where(filter);
        }

        if (includeProperties != null)
        {
            foreach (var property in includeProperties)
            {
                query = query.Include(property);
            }
        }

        if (orderBy != null)
        {
            query = orderBy(query);
        }
```

```
        if (pageNumber.HasValue && pageSize.HasValue)
        {
            query = query.Skip(pageNumber.Value *
pageSize.Value).Take(pageSize.Value);
        }

        return await query.ToListAsync();
    }

    public async Task InsertAsync(T entity)
    {
        await _dbSet.AddAsync(entity);
    }

    public void Delete(T entity)
    {
        if (_context.Entry(entity).State == EntityState.Detached)
        {
            _dbSet.Attach(entity);
        }

        _dbSet.Remove(entity);
    }

    public void Update(T entity)
    {
```

```
        _dbSet.Attach(entity);
        _context.Entry(entity).State = EntityState.Modified;
    }
}
```

Файл UnitOfWork.cs

```
using Desertation.Core.Context;
using Desertation.Core.Entities;
using Desertation.Core.Interfaces;
using Desertation.Core.Repositories;

namespace Desertation.Core.UnitOfWork;

public class UnitOfWork : IUnitOfWork
{
    private ApplicationDbContext _context;

    private IRepository< Command > _commandRepository;
    public IRepository< Command > CommandRepository => _ commandRepository
??= new GenericRepository< Command >(_context);

    private IRepository<Localization> _localizationRepository;
    public IRepository< Localization > LocalizationRepository => _
localizationRepository ??= new GenericRepository< Localization >(_context);

    private IRepository<CommandLocalization> _commandLocalizationRepository;
    public IRepository< CommandLocalization > CommandLocalizationRepository =>
_commandLocalizationRepository ??= new
GenericRepository<CommandLocalization>(_context);
```

```
public UnitOfWork(ApplicationDbContext context)
{
    _context = context;
}



public async Task CommitAsync()
{
    await _context.SaveChangesAsync();
}

public async Task RollbackAsync()
{
    await _context.DisposeAsync();
}

private bool disposed = false;
public void Dispose()
{
    Dispose(true);
    GC.SuppressFinalize(this);

}

public void Dispose(bool disposing)
{
    if (!disposed)
```

```
        {
            if (disposing)
            {
                _context.Dispose();
            }
        }
        disposed = true;
    }
}
```

Файл Command.cs

```
Desertation.Core.Entities;

public class Command
{
    public int Id { get; set; }
    public string TextKey { get; set; }
    public string CommandOptions { get; set; }
    public bool IsChangable { get; set; }
}
```

Файл Localization.cs

```
Desertation.Core.Entities;

public class Localization
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string LocalizationPath{ get; set; }
```

```
    public bool IsPreset{ get; set; }
}
```

Файл CommandLocalization.cs

```
Desertation.Core.Entities;
public class CommandLocalization
{
    public int CommandId { get; set; }
    public int LocalizationId { get; set; }
    public string LocalizedTextKey {get; set;}
}
```

Файл ISettingsProcessor.cs

```
namespace Desertation.BL.Settings;

public interface ISettingsProcessor
{
    Task<bool> AddSetting(VoiceCommand command);
    Task<bool> EditSetting(string textKey, VoiceCommand command);
    Task<bool> DeleteSetting(string textKey);
}
```

Файл ISettingsReader.cs

```
using System.Collections.ObjectModel;
using Desertation.BL.Command;

namespace Desertation.BL.Settings;
```

```
interface ISettingsReader
{
   Task<ObservableCollection<VoiceCommand>> GetAllSettings();
   Task<VoiceCommand> GetSetting(string textKey);
}
```

Файл SettingsProcessor.cs

```
using Desertation.BL.Command;
using Newtonsoft.Json;

namespace Desertation.BL.Settings;

public class SettingsProcessor : ISettingsProcessor
{
   public SettingsProcessor()
   {
      this.voiceCommands = new List<VoiceCommand>();
      LoadVoiceCommands();
   }
   private void LoadVoiceCommands()
   {
      if (File.Exists(filePath))
      {
         string json = File.ReadAllText(filePath);
         voiceCommands =
JsonConvert.DeserializeObject<List<VoiceCommand>>(json) ?? new
List<VoiceCommand>();
      }
```

```
    }


    public async Task<bool> AddSetting(VoiceCommand command)
    {
        if (voiceCommands.Any(vc => vc.TextKey.Equals(command.TextKey,
StringComparison.OrdinalIgnoreCase)))
        {
            return false;
        }
        voiceCommands.Add(command);
        SaveVoiceCommands();
        return true;
    }
    private void SaveVoiceCommands()
    {
        string json = JsonConvert.SerializeObject(voiceCommands,
Formatting.Indented);
        File.WriteAllText(filePath, json);
    }


    public async Task<bool> EditSetting(string textKey, VoiceCommand command)
    {
        var existingCommand = voiceCommands.FirstOrDefault(vc =>
vc.TextKey.Equals(textKey, StringComparison.OrdinalIgnoreCase));
        if (existingCommand == null)
        {
            return false; // Command with the original TextKey not found
        }
```

```
        // Check if the updated TextKey conflicts with another command's TextKey (if it's being changed)
        if (!textKey.Equals(command.TextKey, StringComparison.OrdinalIgnoreCase) &&
            voiceCommands.Any(vc => vc.TextKey.Equals(command.TextKey, StringComparison.OrdinalIgnoreCase)))
        {
            return false; // Updated TextKey already exists
        }

        // Update the command properties
        existingCommand.TextKey = command.TextKey;
        existingCommand.Operation = command.Operation;
        existingCommand.CommandOptions = command.CommandOptions;
        existingCommand.IsChangeable = command.IsChangeable;

        SaveVoiceCommands();
        return true;
    }

    public async Task<bool> DeleteSetting(string textKey)
    {
        var commandToDelete = voiceCommands.FirstOrDefault(vc => vc.TextKey.Equals(textKey, StringComparison.OrdinalIgnoreCase));
        if (commandToDelete == null)
        {
            return false; // Command not found
```

```
        }

        voiceCommands.Remove(commandToDelete);
        SaveVoiceCommands();
        return true;
    }
}
```

Файл SettingsReader.cs

```
using System.Collections.ObjectModel;
using Desertation.BL.Command;
using Newtonsoft.Json;

namespace Desertation.BL.Settings;

public class SettingsReader : ISettingsReader
{
    public async Task<ObservableCollection<VoiceCommand>> GetAllSettings()
    {
        if (File.Exists(filePath))
        {
            string json = File.ReadAllText(filePath);
            return
JsonConvert.DeserializeObject<ObservableCollection<VoiceCommand>>(json) ??
new ObservableCollection<VoiceCommand>();
        }
        return  new ObservableCollection<VoiceCommand>();
    }
```

```
    public async Task<VoiceCommand> GetSetting(string textKey)
    {
        if (File.Exists(filePath))
        {
            string json = File.ReadAllText(filePath);
            var commands =
JsonConvert.DeserializeObject<List<VoiceCommand>>(json) ?? new
List<VoiceCommand>();
            return commands.FirstOrDefault(_ => _.TextKey == textKey);
        }
        return null;
    }
}
```

Файл KeyWords.cs

```
namespace Desertation.BL.Models;


public enum KeyWords
{
    Open,
    Close,
    Copy,
    Delete,
    Settings,
    TurnOn,
    TurnOff,
    Sleep,
```

```
    Type,
    Mute,
    Unmute
}
```

Файл IVoiceCommand.cs

```
using Desertation.BL.Models;

namespace Desertation.BL.Command;

public interface IVoiceCommand
{
    string TextKey { get; set; }
    public KeyWords Operation { get; set; }
    public string CommandOptions { get; set; }
    bool IsChangeable { get; set; }
}
```

Файл UserCommandDto.cs

```
using Desertation.BL.Models;

namespace Desertation.BL.Command;

public class UserCommandDto: IVoiceCommand
{
    public string TextKey { get; set; }
    public KeyWords Operation { get; set; }
    public string CommandOptions { get; set; }
```

```
    public bool IsChangeable { get; set; }
}
```

Файл ICommandProcessor.cs

```
using Desertation.BL.Command;

namespace Desertation.BL.CommandProcessor;

interface ICommandProcessor
{
    Task RunCommand(IVoiceCommand command);
}
```

Файл CommandProcessor.cs

```
using Desertation.BL.Command;

namespace Desertation.BL.CommandProcessor;

public class CommandProcessor : ICommandProcessor
{
    private readonly ExecutorHelper ExecutorHelper{ get; set; }

    public CommandProcessor()
    {
        ExecutorHelper = new ExecutorHelper();
    }

    public Task RunCommand(IVoiceCommand command)
```

```
    {
        var executor = ExecutorHelper.GetExecutor(command.Operation);

        executor.Execute(command.CommandOptions);
    }
}
```

Файл CommandType.cs

```
namespace Desertation.BL.Models;

public static class CommandType
{
    public static Dictionary<KeyWords, Type> OperationTypes = new
Dictionary<KeyWords, Type>()
    {
        new KeyValuePair<KeyWords, Type>(KeyWords.Open, OpenExecutor),
        new KeyValuePair<KeyWords, Type>(KeyWords.Close, CloseExecutor),
        new KeyValuePair<KeyWords, Type>(KeyWords.Delete, DeleteExecutor),
        new KeyValuePair<KeyWords, Type>(KeyWords.Settings, SettingsExecutor),
        new KeyValuePair<KeyWords, Type>(KeyWords.TurnOn, TurnOnExecutor),
        new KeyValuePair<KeyWords, Type>(KeyWords.TurnOff, TurnOffExecutor),
        new KeyValuePair<KeyWords, Type>(KeyWords.Sleep, SleepExecutor),
        new KeyValuePair<KeyWords, Type>(KeyWords.Type, TypeExecutor),
        new KeyValuePair<KeyWords, Type>(KeyWords.Mute, MuteExecutor),
        new KeyValuePair<KeyWords, Type>(KeyWords.Unmute, UnmuteExecutor),
    };
}
```

Файл CommandProcessor.cs

```
using Desertation.BL.Command;

namespace Desertation.BL.CommandProcessor;

public class CommandProcessor : ICommandProcessor
{
    private readonly ExecutorHelper ExecutorHelper{ get; set; }

    public CommandProcessor()
    {
        ExecutorHelper = new ExecutorHelper();
    }

    public Task RunCommand(IVoiceCommand command)
    {
        dynamic executor = ExecutorHelper.GetExecutor(command.Operation);

        executor?.Execute(command.CommandOptions);
    }
}
```

Файл IExecutor.cs

```
public interface IExecutor
{
    public Task Execute(string[] parameters);
}
```

Файл OpenExecutor.cs

```
using System.Diagnostics;

public class OpenExecutor
{
    public async Task Execute(string[] parameters)
    {
        ProcessStartInfo startInfo = new ProcessStartInfo
        {
            FileName = parameters[0],
            UseShellExecute = true
        };
        Process.Start(startInfo);
    }
}
```

Файл MuteExecutor.cs

```
using NAudio.CoreAudioApi;

public class MuteExecutor
{
    public async Task Execute(string[] parameters)
    {
        using (var enumerator = new MMDeviceEnumerator())
        {
            using (var device = enumerator.GetDefaultAudioEndpoint(DataFlow.Render,
Role.Multimedia))
            {
                device.AudioEndpointVolume.Mute = true;
```

```
            }
        }
    }
}
```

Файл UnmuteExecutor

```
using NAudio.CoreAudioApi;

public class UnmuteExecutor
{
    public async Task Execute(string[] parameters)
    {
        using (var enumerator = new MMDeviceEnumerator())
        {
            using (var device = enumerator.GetDefaultAudioEndpoint(DataFlow.Render,
Role.Multimedia))
            {
                device.AudioEndpointVolume.Mute = false;
            }
        }
    }
}
```

Файл SettingsExecutor.cs

```
public class SettingsExecutor
{
    public async Task Execute(string[] parameters)
    {
```

```
        try
        {
            ProcessStartInfo startInfo = new ProcessStartInfo
            {
                FileName = "ms-settings:",
                UseShellExecute = true
            };

            Process.Start(startInfo);
        }
        catch (Exception ex)
        {
            Console.WriteLine("An error occurred: " + ex.Message);
        }
    }
}
```

Файл TypeExecutor.cs

```
public class TypeExecutor
{
[DllImport("user32.dll", SetLastError = true)]
static extern uint SendInput(uint nInputs, INPUT[] pInputs, int cbSize);

[DllImport("user32.dll")]
static extern short VkKeyScan(char ch);

const int INPUT_KEYBOARD = 1;
const uint KEYEVENTF_KEYUP = 0x0002;
```

```
struct INPUT
{
    public int type;
    public InputUnion u;
}

[StructLayout(LayoutKind.Explicit)]
struct InputUnion
{
    [FieldOffset(0)]
    public KEYBDINPUT ki;
}

struct KEYBDINPUT
{
    public ushort wVk;
    public ushort wScan;
    public uint dwFlags;
    public uint time;
    public IntPtr dwExtraInfo;
}
    static void SimulateKeyStroke(char c)
    {
        short key = VkKeyScan(c);
        ushort vk = (ushort)(key & 0xff);
        bool shift = (key & 0x100) > 0;
```

```
    if (shift)
    {
        PressKey(0x10, true);

    PressKey(vk, true);
    PressKey(vk, false);

    if (shift)
    {
        PressKey(0x10, false);
    }
}
static void PressKey(ushort key, bool down)
{
    INPUT input = new INPUT
    {
        type = INPUT_KEYBOARD,
        u = new InputUnion
        {
            ki = new KEYBDINPUT
            {
                wVk = key,
                dwFlags = down ? 0 : KEYEVENTF_KEYUP
            }
        }
    };
```

```
        SendInput(1, new INPUT[] { input }, Marshal.SizeOf(typeof(INPUT)));
    }
    public async Task Execute(string[] parameters)
    {
        foreach (var str in parameters)
        {
            foreach (char c in str)
            {
                SimulateKeyStroke(c);
            }
        }
    }
}
```

Файл MainWindow.cs

```
using System.Collections.ObjectModel;
using System.Windows;
using Desertation.BL.Command;
using Desertation.BL.Settings;

namespace Desertation.App;

/// <summary>
/// Interaction logic for MainWindow.xaml
/// </summary>
public partial class MainWindow : Window
{
    public ObservableCollection<VoiceCommand> Settings { get; set; }
```

```
public SettingsReader Reader { get; set; }

public MainWindow()
{
    InitializeComponent();
    Reader = new SettingsReader();
    Settings = new ObservableCollection<VoiceCommand>();
    LoadSettings();
    lvSettings.ItemsSource = Settings;
}
private async Task LoadSettings()
{
    Settings = await Reader.GetAllSettings();
}

private void AddButton_Click(object sender, RoutedEventArgs e)
{
    var addNewSettingWindow = new AddNewSettingWindow();
    addNewSettingWindow.ShowDialog();
    LoadSettings();
}

private void EditButton_Click(object sender, RoutedEventArgs e)
{
    if (lvSettings.SelectedItem is VoiceCommand selectedSetting)
    {
        var editSettingWindow = new EditSettingWindow(selectedSetting.TextKey);
```

```
            editSettingWindow.ShowDialog();
            LoadSettings();
        }
    }

    private void DeleteButton_Click(object sender, RoutedEventArgs e)
    {
        if (lvSettings.SelectedItem is VoiceCommand selectedSetting)
        {
            Settings.Remove(selectedSetting);
        }
    }
}
```

Файл EditSettingWindow.cs

```
using System.Windows;
using Desertation.BL.Command;
using Desertation.BL.Models;
using Desertation.BL.Settings;

namespace Desertation.App;

public partial class EditSettingWindow : Window
{
    public KeyWords SelectedOperation { get; set; }
    public SettingsProcessor SettingsProcessor { get; set; }
    public SettingsReader SettingsReader { get; set; }
    public EditSettingWindow(string textKey)
```

```
    {
        InitializeComponent();
        SettingsProcessor = new SettingsProcessor();
        SettingsReader = new SettingsReader();
        cmbOperation.ItemsSource = Enum.GetValues(typeof(KeyWords));
        LoadCommand(textKey);
    }

    private async void LoadCommand(string textKey)
    {
        var command = await SettingsReader.GetSetting(textKey);
        txtTextKey.Text = command.TextKey;
        cbIsChangeable.IsChecked = command.IsChangeable;
        cmbOperation.SelectedItem = command.Operation;
    }
    private async void SaveChangesButton_Click(object sender, RoutedEventArgs e)
    {
        var voiceCommand = new VoiceCommand()
        {
            TextKey = txtTextKey.Text,
            Operation = SelectedOperation,
            CommandOptions = "",
            IsChangeable = (bool)cbIsChangeable.IsChecked!
        };
        var result = await SettingsProcessor.EditSetting(txtTextKey.Text,
voiceCommand);
```

```
        MessageBox.Show(result ? "Success" : "Failed to edit setting!", "Alert",
MessageBoxButton.OK, MessageBoxImage.Warning);
        Close();
    }

    private void CancelButton_Click(object sender, RoutedEventArgs e)
    {
        this.Close();
    }
}
```

Файл AddNewSettingWindow.cs

```
using System.Windows;
using Desertation.BL.Command;
using Desertation.BL.Models;
using Desertation.BL.Settings;

namespace Desertation.App;

public partial class AddNewSettingWindow : Window
{
    public KeyWords SelectedOperation { get; set; }
    public SettingsProcessor SettingsProcessor { get; set; }
    public AddNewSettingWindow()
    {
        InitializeComponent();
        SettingsProcessor = new SettingsProcessor();
        cmbOperation.ItemsSource = Enum.GetValues(typeof(KeyWords));
```

```
    }

    private async void SaveButton_Click(object sender, RoutedEventArgs e)
    {
        var voiceCommand = new VoiceCommand()
        {
            TextKey = txtTextKey.Text,
            Operation = SelectedOperation,
            CommandOptions = "",
            IsChangeable = (bool)cbIsChangeable.IsChecked!
        };
        var result = await SettingsProcessor.AddSetting(voiceCommand);

        MessageBox.Show(result ? "Success" : "Failed to create setting!", "Alert",
MessageBoxButton.OK, MessageBoxImage.Warning);
        Close();
    }

    private void CancelButton_Click(object sender, RoutedEventArgs e)
    {
        Close();
    }
}
```

ДОДАТОК Г

Ім'я користувача:
Лісовиченко Олег Іванович

ID перевірки:
1016065545

Дата перевірки:
16.01.2024 09:07:55 EET

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
16.01.2024 09:09:25 EET

ID користувача:
76913

Назва документа: **ІП-22мп_Соболевський_ПЗ**

Кількість сторінок: **57** Кількість слів: **9382** Кількість символів: **75215** Розмір файлу: **4.02 MB** ID файлу: **1015770514**

10.2%
Схожість

Найбільша схожість: **2.86%** з Інтернет-джерелом (https://ela.kpi.ua/bitstream/123456789/53534/1/Smuchok_Sobolevskyi..

6.11% Джерела з Інтернету | 86 ..... Сторінка 59

10.1% Джерела з Бібліотеки | 386 ..... Сторінка 60

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи | 4