**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ  
імені ІГОРЯ СІКОРСЬКОГО»**

**Факультет прикладної математики**

**Кафедра програмного забезпечення комп’ютерних систем**

**ЗВІТ**

**з диспципліни «Мультимедійні інтерфейси та 3D-візуалізація»**

**спеціальності 121 Інженерія програмного забезпечення**

**на тему: «Голосовий асистент та мультимедійний помічник для отримання послуг сервіс-центру»**

Виконав:

студент I курсу, групи КП-41мн

Хоменко Максим Вячеславович \_\_\_\_\_\_\_\_\_\_

До захисту допущено \_\_\_\_\_\_\_\_\_\_

Захищено з оцінкою \_\_\_\_\_\_\_\_\_\_

Київ – 2024 року

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

**Факультет прикладної математики**

**Кафедра програмного забезпечення комп’ютерних систем**

Рівень вищої освіти – другий (магістерський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-наукова програма «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

**ЗАВДАННЯ**

**на курсову роботу студенту**

Хоменку Максиму Вячеславовичу

1. Тема проєкту «Голосовий асистент та мультимедійний помічник для отримання послуг сервіс-центру», розроблення архітектури та логіки застосунку.
2. Термін подання студентом проєкту «26» грудня 2024 р.
3. Вихідні дані до проєкту: див. Технічне завдання.
4. Перелік обов’язкового графічного матеріалу:
   * структура програмного застосунка(креслення);
   * алгоритм взаємодії користувача з системою (креслення);
   * функціональність вебзастосунку (плакат).

Календарний план

|  |  |  |  |
| --- | --- | --- | --- |
| № з/п | Назва етапів виконання  курсового проєкту | Термін виконання  етапів проєкту | Примітка |
| 1. | Вивчення літератури за тематикою проєкту | 26.09.2024 |  |
| 2. | Розроблення та узгодження технічного завдання | 02.10.2024 |  |
| 3. | Розроблення структури програмного забезпечення | 03.10.2024 |  |
| 4. | Розроблення дизайну програмного забезпечення | 17.10.2024 |  |
| 5. | Розроблення програмного забезпечення | 22.11.2024 |  |
| 6. | Тестування програмного забезпечення | 28.11.2024 |  |
| 7. | Оформлення документації курсового проєкту | 12.12.2024 |  |

Студент Максим ХОМЕНКО

Керівник проєкту Євгенія СУЛЕМА

ДП.045440-01-90 ГОЛОСОВИЙ АСИСТЕНТ ТА МУЛЬТИМЕДІЙНИЙ ПОМІЧНИК ДЛЯ ОТРИМАННЯ ПОСЛУГ СЕРВІС-ЦЕНТРУ. Відомість проєкту

|  |  |  |  |
| --- | --- | --- | --- |
| Позначення | Найменування | Кіл-ть | Примітка |
|  | Документація проєкту |  |  |
|  |  |  |  |
| ДП.045440-02-91 | Інформаційна система | 5 |  |
|  | ідентифікації настрою |  |  |
|  | відгуку клієнта |  |  |
|  | про ресторан. Технічне |  |  |
|  | завдання |  |  |
| ДП.045440-03-81 | Інформаційна система | 55 |  |
|  | ідентифікації настрою |  |  |
|  | відгуку клієнта |  |  |
|  | про ресторан. |  |  |
|  | Пояснювальна записка |  |  |
| ДП.045440-04-51 | Інформаційна система | 4 |  |
|  | ідентифікації настрою |  |  |
|  | відгуку клієнта |  |  |
|  | про ресторан. |  |  |
|  | Керівництво користувача |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Факультет прикладної математики**

**Кафедра програмного забезпечення комп’ютерних систем**

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

\_\_\_\_\_\_\_\_\_\_ Євгенія СУЛЕМА

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_ 2024 р.

**ГОЛОСОВИЙ АСИСТЕНТ ТА МУЛЬТИМЕДІЙНИЙ ПОМІЧНИК ДЛЯ ОТРИМАННЯ ПОСЛУГ СЕРВІС-ЦЕНТРУ**

**Технічне завдання**

ДП.045440-02-91

|  |  |
| --- | --- |
| “ПОГОДЖЕНО”  Керівник проєкту:  \_\_\_\_\_\_\_\_\_\_ Євгенія СУЛЕМА  \_\_” \_\_\_\_\_\_\_\_\_\_\_\_ 2011 р. | Виконавець:  \_\_\_\_\_\_ Максим ХОМЕНКО |

2024

# **1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ**

**Назва розробки**: Голосовий асистент та мультимедійний помічник для отримання послуг сервіс-центру.

**Галузь застосування**: сервісні послуги.

# **2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ**

Підставою для розроблення є завдання на курсову роботу, затверджене кафедрою програмного забезпечення комп’ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського.

# **3. ПРИЗНАЧЕННЯ РОЗРОБКИ**

Розробка призначена для поліпшення інтеракції клієнтів з сервіс-центром через впровадження голосового асистента та мультимедійного помічника, які спрощують процес отримання інформації про продукти та послуги. Голосовий асистент дозволяє користувачам переглядати списки підтримуваних продуктів, отримувати детальну інформацію про окремі товари та їх користувацькі інструкції шляхом простих голосових команд. Також система надає можливість візуальної інтеракції завдяки тривимірній анімації помічника, яка реагує на команди користувача, тим самим підвищуючи залученість та зручність використання. Ця розробка сприяє ефективнішому та комфортнішому обслуговуванню клієнтів, забезпечуючи швидкий доступ до потрібної інформації та спрощення процесу подачі заяв до сервіс-центру, що, у свою чергу, підвищує задоволеність клієнтів та ефективність роботи сервіс-центру.

# **4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ**

Вебзастосунок повинен забезпечувати такі аспекти:

1. Голосовий асистент повинен бути інтегрований з веб-браузером, що дозволяє користувачам переглядати список підтримуваних продуктів;
2. Можливість переглядати деталі кожного продукту зі списку, включно з доступом до користувацьких інструкцій;
3. Відображення тривимірної анімованої моделі помічника під час використання голосових команд для кращої візуальної взаємодії;
4. Підтримка голосових команд для перегляду списку продуктів;
5. Виконання переходу на сторінку обраного продукту за повною назвою або псевдонімом зі списку;
6. Можливість активації команди «інструкція» на сторінці продукту для відображення користувацької інструкції;
7. Підтримка команди «створити заявку», яка відкриває форму для внесення заявки на обслуговування у сервіс-центрі;
8. Розробка програмного забезпечення на базі мов програмування C# і JavaScript з використанням фреймворків ASP.NET Core для забезпечення багатоплатформенності;
9. Система повинна підтримувати кросплатформенність, тобто працювати на різних операційних системах і пристроях без втрати функціональності;
10. Для кожного продукту в списку повинні бути передбачені щонайменше два псевдоніми для забезпечення гнучкості використання голосових команд;
11. Мультимедійна модель помічника повинна включати не менше двох видів анімації для покращення взаємодії з користувачем.

# **5. ВИМОГИ ДО ПРОЄКТНОЇ ДОКУМЕНТАЦІЇ**

У процесі виконання проєкту повинна бути розроблена наступна документація:

1) пояснювальна записка;

2) креслення:

* + «Стутктура програмного застосунка»;
  + «Алгоритм взаємодії користувача з системою».

# **6. ЕТАПИ ПРОЄКТУВАННЯ**

Вивчення літератури за тематикою проєкту 26.09.2024

Розроблення та узгодження технічного завдання 02.10.2024

Розроблення структури програмного забезпечення 03.10.2024

Розроблення дизайну програмного забезпечення 17.10.2024

Розроблення програмного забезпечення 22.11.2024

Тестування програмного забезпечення 28.11.2024

Оформлення технічної документації проєкту 12.12.2024

**Факультет прикладної математики**

**Кафедра програмного забезпечення комп’ютерних систем**

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

\_\_\_\_\_\_\_\_\_\_ Євгенія СУЛЕМА

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_ 2024 р.

**ГОЛОСОВИЙ АСИСТЕНТ ТА МУЛЬТИМЕДІЙНИЙ ПОМІЧНИК ДЛЯ ОТРИМАННЯ ПОСЛУГ СЕРВІС-ЦЕНТРУ**

**Пояснювальна записка**

ДП.045440-03-81

|  |  |
| --- | --- |
| “ПОГОДЖЕНО”  Керівник проєкту:  \_\_\_\_\_\_\_\_\_\_ Євгенія СУЛЕМА | Виконавець:  \_\_\_\_\_\_\_ Максим ХОМЕНКО |

2024

ЗМІСТ

[СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ 4](#_Toc167553318)

[ВСТУП 8](#_Toc167553319)

[1. АНАЛІЗ ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ 9](#_Toc167553320)

[1.1. Аналіз предметної області 9](#_Toc167553321)

[1.2. Аналіз існуючих аналогів застосунків 10](#_Toc167553322)

[1.3. Вимоги до програмного забезпечення 13](#_Toc167553323)

[1.4. Висновки 14](#_Toc167553324)

[2. ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ 16](#_Toc167553325)

[2.1. Серверна частина застосунку 16](#_Toc167553326)

[2.2. Клієнтська частина застосунку 23](#_Toc167553327)

[2.3. Висновки 28](#_Toc167553328)

[3. СТРУКТУРНО-АЛГОРИТМІЧНА ОРГАНІЗАЦІЯ РОЗРОБЛЕНОГО ВЕБЗАСТОСУНКУ 30](#_Toc167553329)

[3.1. Архітектура вебзастосунку 30](#_Toc167553330)

[3.2. Структура та алгоритми предметної галузі 31](#_Toc167553331)

[3.3. Опис функціональних вимог 33](#_Toc167553332)

[3.4. Висновки 34](#_Toc167553333)

[4. ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 37](#_Toc167553334)

[4.1. Особливості реалізації клієнтської частини вебзастосунку 37](#_Toc167553335)

[4.2. Особливості реалізації серверної частини вебзастосунку 38](#_Toc167553336)

[4.3. Тестування розробленого вебзастосунку 39](#_Toc167553337)

[4.4. Напрямки подальшого розвитку 42](#_Toc167553338)

[4.5. Висновки 42](#_Toc167553339)

[ВИСНОВКИ 44](#_Toc167553340)

[СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ 47](#_Toc167553341)

[ДОДАТКИ 49](#_Toc167553342)

# **СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ**

*Java –* є високорівневою, об’єктно-орієнтованою мовою програмування, яка була розроблена компанією Sun Microsystems (зараз належить Oracle). Вона була випущена у 1995 році та стала однією з найпопулярніших мов програмування.

*JSON –* JavaScript Object Notation (формат запису об’єктів JavaScript).

*СКБД –* (Система Керування Базами Даних), набір взаємопов’язаних даних (база даних) і програм для доступу до цих даних. Надає можливості створення, збереження, оновлення та пошуку інформації в базах даних з контролем доступу до даних.

*REST*– Representational State Transfer (передача стану подання).

*Авторизація –* надання певній особі або групі осіб прав на виконання певних дій; а також процес перевірки (підтвердження) даних прав при спробі виконання цих дій.

*Аутентифікація* – процедура встановлення належності користувачеві інформації в системі через перевірку пред’явленого ним ідентифікатора.

*SQL* – Structured Query Language (мова структурованих запитів), декларативна мова програмування для взаємодії користувача з базами даних, що застосовується для формування запитів, оновлення і керування реляційними БД, створення схеми бази даних та її модифікації, системи контролю за доступом до бази даних.

*Endpoint* – (кінцева точка) відноситься до конкретного маршруту або URL в вебсервісі, куди можна відправити HTTP-запити (наприклад, GET, POST, PUT, DELETE) для виконання певних операцій.

*CRUD* – (create, read, update, delete), операції над записами в  базі даних – створення, читання, оновлення та видалення.

*API* – (інтерфейс програмування застосунків), це набір правил та протоколів для побудови та взаємодії програмного забезпечення.

*Angular* – це популярний фреймворк для розробки вебсервісів з використанням мови програмування TypeScript.

*TypeScript* – це мова програмування, розроблена компанією Microsoft, яка є розширенням мови JavaScript.

*Модульне тестування* – це процес тестування окремих компонентів програми (наприклад, методів класу) для перевірки того, чи вони виконують очікувану поведінку.

*Ручне тестування* – це процес виконання тестового сценарію вручну, без використання автоматизованого тестового скрипта або інструменту.

*Інтеграційне тестування* – це процес тестування взаємодії між різними компонентами системи.

*ООП* – (Об’єктно-Орієнтоване Програмування), це модель програмування, в центрі якої лежить поняття найпростіших одиниць: об’єкту та класу, а також принципи, яким мають слідувати ознаки цих одиниць – принципам абстракції та інкапсуляції, наслідуванню і поліморфізму.

*Архітектура N-Tier* – це проєктний план, який використовується при розробці програмного забезпечення, де різні частини програми розділені на шари, або «яруси», верстви. Кожен рівень відповідає за певний тип завдань; один рівень може відповідати за інтерфейс користувача, інший – за дані, а третій – за логіку та обчислення; поділ робить систему більш організованою, легшою в управлінні та підвищує її здатність до змін і зростання.

*DDD* – (Domain-Driven Design), це підхід у розробці програмного забезпечення, який акцентує увагу на складності, яка виникає через потреби домену, або сфери знань, в якій буде використовуватися розроблювана система; DDD пропонує створення програмного забезпечення, виходячи з моделювання домену, що передбачає тісну співпрацю між фахівцями в даній предметній області та розробниками. Важливим аспектом є розбиття домену на окремі частини, кожна з яких має свою модель і логіку; такий підхід допомагає управляти складністю і створювати високофункціональне та легко адаптоване до змін програмне забезпечення.

*TDD* – (Test-Driven Development) це методологія розробки програмного забезпечення, яка вимагає писати тести до того, як буде написаний сам код програми. Цей підхід заснований на коротких циклах розробки, які називаються «Red-Green-Refactor», це означає, що спочатку пишеться тест, який не проходить (Red), потім пишеться мінімально необхідний код, щоб тест пройшов (Green), і нарешті код підлягає рефакторингу для підвищення якості та читабельності (Refactor).

*GRASP* – це набір принципів або вказівок щодо розподілу відповідальності за класами та об’єктами в об’єктно-орієнтованому проєктуванні для покращення якості програмного забезпечення. Основна мета цих шаблонів полягає в тому, щоб дизайн програмного забезпечення був надійним, придатним для обслуговування та масштабованим. ҐРАСП надає розробникам програмного забезпечення спосіб вирішення загальних проблем об’єктно-орієнтованого проєктування.

*Clean Architecture* – це архітектурний шаблон проєктування, який організовує проєктування програмного забезпечення на рівні зі строгими правилами залежностей, гарантуючи, що залежності перетікають всередину до сценаріїв використання та основної бізнес-логіки; він наголошує на відокремленні проблем, спрямованих на те, щоб зробити програмне забезпечення простішим в управлінні, розробці та тестуванні; ізолюючи програмні елементи від зовнішніх впливів та інфраструктури, покращує підтримуваність, гнучкість та масштабованість застосунків.

*LINQ* – (Language Integrated Query) – це функція в C#, яка дозволяє розробникам писати запити для доступу та маніпулювання даними з різних джерел, таких як бази даних, масиви та XML-документи, безпосередньо в мові C#; він надає спосіб писати чистий і лаконічний код для фільтрації, сортування та виконання інших операцій з даними, бездоганно поєднуючись із синтаксисом C#; Ця інтеграція допомагає розробникам писати більш читабельний і підтримуваний код у порівнянні з традиційними мовами запитів, такими як SQL.

*Сервер* – це комп’ютер або система, що надає послуги, ресурси або функціональні можливості іншим комп’ютерам або програмам, відомим як клієнти.

*Клієнт* – це комп’ютер або програма, яка отримує доступ до послуг або ресурсів, наданих сервером. У контексті мережі, клієнтська програма зазвичай звертається до сервера для отримання інформації, надсилання запитів, виконання операцій або отримання результатів обробки.*Java –* є високорівневою, об’єктно-орієнтованою мовою програмування, яка була розроблена компанією Sun Microsystems (зараз належить Oracle). Вона була випущена у 1995 році та стала однією з найпопулярніших мов програмування.

# **ВСТУП**

Розробка призначена для поліпшення інтеракції клієнтів з сервіс-центром через впровадження голосового асистента та мультимедійного помічника, які спрощують процес отримання інформації про продукти та послуги. Голосовий асистент дозволяє користувачам переглядати списки підтримуваних продуктів, отримувати детальну інформацію про окремі товари та їх користувацькі інструкції шляхом простих голосових команд.

Розробка голосового асистента та мультимедійного помічника для сервіс-центру направлена на поліпшення інтерактивності та доступності послуг для клієнтів. Завдяки використанню передових технологій у сфері штучного інтелекту та обробки природної мови, цей інструмент дозволяє користувачам взаємодіяти з сервісом за допомогою голосу, значно спрощуючи процес отримання послуг та забезпечуючи високий рівень задоволеності.

Таким чином, запропонований голосовий асистент і мультимедійний помічник мають стати вагомим внеском у сферу сервісних послуг, покращуючи взаємодію з клієнтами та сприяючи розвитку індустрії в цілому.

# **АНАЛІЗ ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ**

## **Аналіз предметної області**

Традиційний підхід до забезпечення та підтримки сервісних послуг зазвичай включає в себе необхідність особистої взаємодії клієнта з сервісними центрами, що може бути часомістким і не завжди ефективним. Виклики, з якими зіштовхуються клієнти, часто включають складність у знаходженні необхідної інформації, затримки у відповідях та обмежену доступність допомоги під час неробочих годин. Впровадження голосового асистента та мультимедійного помічника може кардинально змінити цю динаміку, надаючи миттєвий доступ до інформації та підтримку 24/7, що робить процес взаємодії з сервісним центром більш гнучким і клієнтоорієнтованим.

## **Аналіз існуючих аналогів застосунків**

### ***«Amazon Alexa»***

«Amazon Alexa» – є широко відомим голосовим асистентом, який надає можливість управління різними пристроями, доступом до інформації та підтримки користувачів через голосові команди. Однак, його основна спрямованість на загальні функції і інтеграцію з різноманітними платформами може не завжди відповідати специфічним вимогам сервісних центрів у плані спеціалізованої інформації про продукти та їх обслуговування.

### ***«Google Assistant»***

«Google Assistant» також пропонує широкий спектр можливостей для голосового управління і доступу до інформації. Проте, як і Alexa, цей асистент оптимізований більше для загального використання, і може мати обмеження, коли справа доходить до інтеграції з конкретними базами даних або сервісними платформами, що вимагають глибокої спеціалізації.

### ***«Siri»***

Хоча «Siri» від Apple є популярним вибором для користувачів iOS, її функціональність у сфері сервісних послуг все ще залишається обмеженою, особливо в частині інтеграції з корпоративними системами і сервісними центрами, що робить її менш придатною для специфічних потреб нашого проекту.

## **Вимоги до програмного забезпечення**

Основні вимоги до нашого програмного продукту включають:

### ***Вимоги до функціональності***

Голосовий асистент має надавати точну і актуальну інформацію про продукти та сервіси, з можливістю подання заяв на обслуговування.

### ***Вимоги до інтерфейсу***

Програма повинна легко інтегруватися з існуючими базами даних сервіс-центру, забезпечуючи актуальність і повноту інформації.

### ***Вимоги до безпеки***

Високий рівень захисту користувацьких даних, що передаються та зберігаються в системі.

### ***Апаратні вимоги***

Забезпечення безперервної роботи сервісу 24/7 без значних затримок у відповідях асистента.

## **Висновки**

Аналіз існуючих голосових асистентів показує, що хоча вони пропонують багато універсальних функцій, їх застосування в специфічних областях, таких як обслуговування в сервісних центрах, може бути обмеженим через відсутність спеціалізованої функціональності та необхідної інтеграції. Таким чином, розробка спеціалізованого голосового асистента, який задовольняє конкретні потреби клієнтів та сервісних центрів, є актуальною та може значно підвищити ефективність взаємодії з клієнтами, їх задоволеність та лояльність до сервісу.

# **ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ**

## **Серверна частина застосунку**

Розроблюване програмне забезпечення є вебзастосунком. Предметна галузь означує невеликий вміст бізнесової логіки: понять, значень, правил та подій. Задоволення її функціональних вимог не потребує організації комплексних процедур, що містять операції відображення даних, валідацію, виконання корисної для предметної галузі логіки, CRUD у базі даних, авторизацію та аутентифікацію, логування та відслідковування швидкодії. Відтак, ціннісними орієнтирами у виборі платформи розробки, вебфреймворку та мови програмування є строга типізація мови та високий рівень сумісності з ООП, наявність широкого стандартного розробницького набору. Крім цього, цінністю може бути лаконічний синтаксис, компільованість, підтримка асинхронної моделі програмування, швидкодія та оптимальне використання ресурсів, доступність сучасних середовищ розробки для платформи, наявність інструментів автоматичного генерування коду.

### ***Мова програмування С#***

Мова програмування C# є сильно типізованою, статично типізованою і компільованою мовою, створеною Microsoft як частина .NET ініціативи. Вона поєднує елементи мов C++ і Java, забезпечуючи розробникам потужні засоби для створення застосунків різної складності. C# підтримує об’єктно-орієнтоване програмування і містить особливості, такі як наслідування, поліморфізм, і інкапсуляцію. Також мова пропонує можливості функціонального програмування, такі як використання лямбда-функцій і LINQ. C# має строгу перевірку типів і автоматичне управління пам’яттю за допомогою збирача сміття, що спрощує управління ресурсами і знижує ризик витоків пам’яті [5].

*Переваги:*

* Строга статична типізація дозволяє виявляти помилки ще під час компіляції, замість часу виконання, що підвищує надійність програм.
* Зрозумілий синтаксис, наближений до природньої мови, схожий на C++ і Java, робить C# привабливим вибором для розробників з досвідом у цих мовах.
* Підтримка сучасних парадигм програмування, таких як асинхронне програмування та паралельні обчислення, дозволяє створювати високопродуктивні застосунки.
* Переносимість коду між різними платформами з допомогою .NET Core, що розширює використання C# за межі Windows.

*Недоліки:*

* Вважається складним для освоєння новачками.
* Потреба в збирачі сміття може призводити до непередбачуваної затримки в програмах, де важлива висока продуктивність та мінімізація затримок.

### ***Платформа .NET 8***

Платформа .NET 8 є продовженням еволюції широко відомої та використовуваної платформи .NET, розробленої Microsoft для створення і запуску різних типів програмного забезпечення, включаючи веб, мобільні та настільні застосунки. Вона об’єднує різні мови програмування, бібліотеки і API в єдиному інтегрованому середовищі розробки, що дозволяє розробникам використовувати одну технологічну основу для будь-якої платформи або пристрою. .NET 8 продовжує підтримувати кросплатформність, дозволяючи запускати програми на Windows, Linux і macOS. Особливу увагу в новій версії приділено підвищенню продуктивності, покращенню безпеки і зменшенню використання пам’яті.

*Переваги:*

* Висока продуктивність та оптимізація: .NET 8 містить численні покращення у продуктивності, що дозволяють програмам виконуватися швидше і ефективніше.
* Кросплатформність: розробники можуть створювати застосунки, які працюють на різних операційних системах із мінімальними змінами до коду.
* Інтегрована підтримка контейнеризації: .NET 8 спрощує використання контейнерів, що є важливим для сучасних хмарних застосунків та мікросервісної архітектури.
* Широкий набір бібліотек: розширений набір бібліотек та готового, протестованого, підтримуваного спільнотою коду спрощує розробку, забезпечуючи готові рішення для звичайних завдань.

*Недоліки:*

* Залежність від Microsoft: хоча .NET є відкритим і крос-платформенним, основна розробка і напрямок платформи контролюються Microsoft, що може бути недоліком для деяких організацій.
* Високі вимоги до ресурсів: для повноцінної роботи .NET 8 можуть знадобитися потужні системи, особливо при розробленні великих застосунків.
* Комплексність міграції для старих проєктів: міграція існуючих проєктів на нову версію платформи може бути складною і ресурсномісткою.

### ***Мова програмування JavaScript***

Мова програмування JavaScript є високорівневою, інтерпретованою, з гнучким синтаксисом та динамічною типізацією. Спочатку створена для додавання інтерактивності вебсторінок, JavaScript швидко розвинулася в універсальну мову, яка зараз використовується для розроблення як клієнтських, так і серверних застосунків. Мова має багату екосистему сторонніх бібліотек і фреймворків, таких як React, Angular, і Node.js, що дозволяє створювати складні вебзастосунки, серверні застосунки, мобільні застосунки, а також інтерактивні ігри і системи реального часу. Як мова з динамічною типізацією, JavaScript дозволяє змінним змінювати свій тип у процесі виконання програми, що надає гнучкість, але також може призвести до складнощів з відлагодженням помилок.

*Переваги:*

* Універсальність: JavaScript може використовуватися для розроблення клієнтської та серверної частин, а також для застосунків мобільних, настільних.
* Велика екосистема: наявність величезної кількості фреймворків і бібліотек спрощує вирішення більшості задач розробки.
* Висока швидкість виконання: сучасні JavaScript двигуни, такі як V8 (використовується в браузері Chrome і Node.js), забезпечують високу продуктивність виконання скриптів.
* Спільнота: Масивна глобальна спільнота розробників і багато ресурсів для навчання та підтримки.

*Недоліки:*

* Динамічна типізація: хоча це надає гнучкість, динамічна типізація може призвести до складнощів з відлагодженням і потенційно прихованих помилок під час виконання.
* Несумісність між браузерами: різниці у реалізації JavaScript рушіїв можуть призвести до того, що один і той же код працює по-різному в різних браузерах.
* Надмірна складність: велика кількість доступних інструментів і бібліотек може ускладнити вибір правильних рішень і підходів до розробки.
* Зловживання з боку розробників; легкість використання і доступність JavaScript може призводити до написання неоптимізованого або небезпечного коду.

### ***Платформа Node.js***

Node.js є середовищем виконання JavaScript на стороні сервера, що дозволяє розробникам використовувати JavaScript для створення швидких і масштабованих мережевих застосунків. Вона використовує асинхронне, подієво-орієнтоване програмування та неблокуючий ввід-вивід, що робить її ідеальною для вебзастосунків з високою продуктивністю, таких як потокова передача даних та реальний час. Node.js має багату екосистему з тисячами вільних модулів, доступних через npm (Node Package Manager), що є найбільшою екосистемою відкритого програмного забезпечення у світі [6].

*Переваги:*

* Швидкість виконання: Node.js використовує рушій V8 від компанії Google, що перетворює JavaScript безпосередньо в машинний код, дозволяючи швидке виконання коду.
* Асинхронна архітектура: асинхронний, неблокуючий ввід та вивід робить Node.js здатним обробляти велику кількість з’єднань одночасно, що знижує затримки і підвищує продуктивність.
* Одна мова на всіх етапах розробки: використання JavaScript як на клієнтській частині, так і на серверній спрощує розробку і покращує взаємодію між командами.
* Сильна спільнота та велика кількість модулів: наявність npm забезпечує доступ до великої кількості пакетів, що робить Node.js гнучким і забезпечує швидке вирішення багатьох типових задач.

*Недоліки:*

* Проблеми з масштабуванням: подієво-орієнтований підхід може стати складним при великих і складних застосунках.
* Нестабільність деяких пакетів npm: через велику кількість доступних пакетів деякі з них можуть бути погано підтримуваними або застарілими.
* Зворотні виклики: використання асинхронного коду може призвести до «пекла зворотніх викликів».
* Потреба у знанні кращих практик: ефективне використання Node.js вимагає глибоких знань у кращих практиках розробки та архітектури.

### ***Фреймворк ASP.NET CORE***

ASP.NET Core – це високопродуктивний фреймворк з відкритим вихідним кодом для створення сучасних хмарних застосунків, підключених до Інтернету [7]. Розроблений корпорацією Microsoft, він дозволяє створювати монолітні вебзастосунки, вебсервіси та технології реального часу, такі як вебсокети. ASP.NET Core – це повна модернізація свого попередника, ASP.NET, з акцентом на модульність і кросплатформеність, що дозволяє розробникам запускати застосунки на Windows, Linux і macOS. Він підтримує архітектурний шаблон MVC, а також може використовуватися в більш простій моделі «запит-відповідь»:

*Переваги:*

* Кросплатформенна підтримка: ASP.NET Core можна розробляти і запускати на Windows, Linux і macOS, що забезпечує велику гнучкість для розробників.
* Висока продуктивність: це один з найшвидших фреймворків для вебзастосунків, згідно з результатами тестування TechEmpower [8].
* Вбудована ін’єкція залежностей: ASP.NET Core містить вбудований фреймворк ін’єкції залежностей, який сприяє створенню більш чистого та модульного коду.
* Розширюваність і модульність: завдяки модульному конвеєру HTTP-запитів, він легко налаштовується під потреби будь-якої програми.
* Підтримка сучасних фреймворків сторони клієнта: він добре працює з популярними фреймворками, такими як Angular, React і Vue.js, полегшуючи розробку багатих інтерактивних інтерфейсів користувача.

*Недоліки:*

* Складність: комплексний характер фреймворку та широта його можливостей можуть бути непосильними для нових розробників; вивчення всіх функцій та найкращих практик ASP.NET Core може зайняти значний час та зусилля.
* Вимагає постійного оновлення: Microsoft часто оновлює ASP.NET Core, що може вносити суттєві зміни, а також вимагати від розробників постійного оновлення знань та застосунків, щоб залишатися в курсі подій.
* Ресурсоємність: хоча ASP.NET Core оптимізовано для продуктивності, застосунки можуть ставати ресурсоємними при великих навантаженнях, особливо якщо вони не оптимізовані належним чином.
* Комплексність розгортання: незважаючи на те, що ASP.NET Core є кросплатформним, розгортання та підтримка застосунків на різних операційних системах може вимагати додаткової конфігурації та налаштування, особливо в складних середовищах.

### ***ML.NET***

ML.NET – це відкритий та кросплатформений фреймворк машинного навчання, розроблений корпорацією Microsoft для розробників, які працюють у середовищі .NET. ML.NET дозволяє створювати, тренувати та впроваджувати моделі машинного навчання без необхідності вивчення спеціалізованих мов програмування або використання зовнішніх бібліотек. Завдяки інтеграції з екосистемою .NET, ML.NET забезпечує зручність використання для розробників, які вже знайомі з цим середовищем, дозволяючи ефективно інтегрувати машинне навчання в існуючі застосунки:

*Переваги:*

* Інтеграція з .NET: ML.NET безшовно інтегрується з іншими технологіями .NET, такими як ASP.NET Core, що спрощує процес розробки та впровадження моделей машинного навчання в існуючі веб-застосунки та сервіси.
* Відкритий вихідний код та безкоштовність: Як проект з відкритим вихідним кодом, ML.NET є доступним для всіх розробників безкоштовно, що сприяє широкому використанню та внеску спільноти у його розвиток.
* Підтримка різноманітних сценаріїв машинного навчання: ML.NET підтримує різні типи задач, включаючи класифікацію, регресію, кластеризацію, рекомендаційні системи та аналіз тексту, що робить його універсальним інструментом для різних проектів.
* Автоматизоване машинне навчання (AutoML): ML.NET включає функціонал AutoML, який автоматизує процес вибору та налаштування моделей, що дозволяє швидко створювати високоякісні моделі без глибоких знань у сфері машинного навчання.
* Підтримка багатьох платформ: Завдяки кросплатформенній природі, ML.NET можна використовувати на різних операційних системах, включаючи Windows, Linux та macOS.
* Документація та спільнота: Широка документація, приклади коду та активна спільнота розробників сприяють швидкому навчанні та вирішенню проблем, що виникають під час роботи з ML.NET

*Недоліки:*

* Обмежена підтримка складних моделей: Хоча ML.NET підходить для багатьох задач машинного навчання, він може бути менш ефективним для дуже складних або специфічних моделей порівняно з іншими фреймворками, такими як TensorFlow чи PyTorch.
* Продуктивність: Для великих обсягів даних або вимогливих до продуктивності моделей ML.NET може бути менш оптимізованим, що може впливати на швидкість тренування та прогнозування.
* Менша кількість готових моделей: На відміну від деяких інших платформ машинного навчання, ML.NET має меншу кількість попередньо натренованих моделей, що може вимагати більше часу на розробку власних рішень.
* Крива навчання: Незважаючи на інтеграцію з .NET, розробники, які не мають досвіду в машинному навчанні, можуть зіткнутися з труднощами при створенні та налаштуванні моделей.
* Обмежені можливості для глибокого навчання: Хоча ML.NET підтримує деякі аспекти глибокого навчання, його можливості в цій сфері є обмеженими порівняно з спеціалізованими фреймворками, такими як Keras або PyTorch.

### ***TensorFlow***

TensorFlow – це відкритий фреймворк машинного навчання, розроблений компанією Google. Він широко використовується для створення, тренування та впровадження моделей глибокого навчання (deep learning). TensorFlow підтримує як низькорівневі, так і високорівневі API, що дозволяє розробникам ефективно працювати з різними аспектами машинного навчання, від досліджень до виробництва:

*Переваги:*

* Гнучкість та масштабованість: TensorFlow дозволяє створювати складні моделі машинного навчання, які можуть бути масштабовані від локальних систем до великих кластерів обчислювальних ресурсів.
* Підтримка глибокого навчання: TensorFlow спеціалізується на глибокому навчанні та забезпечує потужні інструменти для створення нейронних мереж різної складності.
* Широка екосистема: Включає різноманітні бібліотеки та інструменти, такі як TensorBoard для візуалізації, TensorFlow Lite для мобільних пристроїв та TensorFlow Extended (TFX) для виробничих середовищ.
* Підтримка GPU та TPU: Оптимізований для роботи з графічними процесорами (GPU) та тензорними процесорними одиницями (TPU), що значно прискорює процес тренування моделей.
* Активна спільнота та документація: Широка підтримка спільноти розробників, регулярні оновлення та детальна документація сприяють швидкому вирішенню проблем та впровадженню нових функцій.

*Недоліки:*

* Складність для початківців: TensorFlow має високу криву навчання через свою складність та багатофункціональність, що може бути складним для новачків у сфері машинного навчання.
* Високі вимоги до ресурсів: Тренування великих моделей потребує значних обчислювальних ресурсів, що може бути недоступним для малих проектів або окремих розробників.
* Великі розміри бібліотек: TensorFlow може бути громіздким у порівнянні з іншими фреймворками, що ускладнює його використання в обмежених середовищах.
* Менш зручний API для деяких задач: Хоча TensorFlow підтримує високорівневі API, деякі розробники вважають їх менш інтуїтивно зрозумілими порівняно з альтернативами, такими як PyTorch.

### ***PyTorch***

PyTorch – це відкритий фреймворк машинного навчання, розроблений компанією Facebook's AI Research lab (FAIR). Він став популярним завдяки своїй простоті, гнучкості та підтримці динамічного обчислювального графа, що робить його ідеальним для дослідницьких цілей та розробки прототипів:

*Переваги:*

* Динамічний обчислювальний граф: PyTorch використовує динамічний граф, що дозволяє змінювати архітектуру моделі на льоту, що особливо корисно для досліджень та експериментів.
* Простий та інтуїтивно зрозумілий синтаксис: PyTorch має чистий та зрозумілий API, що робить його легким для вивчення та використання, особливо для тих, хто знайомий з Python.
* Широка підтримка спільноти: Активна спільнота розробників сприяє швидкому розвитку фреймворку, а також забезпечує доступ до численних бібліотек та ресурсів.
* Інтеграція з Python: Глибока інтеграція з Python дозволяє використовувати всі можливості екосистеми Python, включаючи бібліотеки для наукових обчислень та обробки даних.
* Підтримка GPU: Оптимізований для роботи з GPU, що значно прискорює процес тренування моделей.

*Недоліки:*

* Менш оптимізований для виробництва: Хоча PyTorch активно розвивається у напрямку виробничого використання, він все ще може бути менш оптимізованим порівняно з TensorFlow для деяких завдань.
* Відсутність деяких функцій із коробки: Для деяких спеціалізованих задач може знадобитися додаткова розробка або використання зовнішніх бібліотек.
* Менша кількість готових моделей порівняно з TensorFlow: Хоча кількість доступних моделей постійно зростає, TensorFlow все ще має більшу бібліотеку попередньо натренованих моделей.

### ***scikit-learn***

scikit-learn – це відкритий фреймворк машинного навчання для мови програмування Python. Він побудований на основі бібліотек NumPy, SciPy та matplotlib і призначений для розробки простих та ефективних інструментів для аналізу даних і машинного навчання:

*Переваги:*

* Простота використання: scikit-learn має інтуїтивно зрозумілий API, що робить його легким для вивчення та використання, навіть для початківців у машинному навчанні.
* Широкий спектр алгоритмів: Підтримує різноманітні алгоритми машинного навчання, включаючи класифікацію, регресію, кластеризацію, зменшення розмірності та обробку даних.
* Висока інтеграція з екосистемою Python: Глибока інтеграція з іншими бібліотеками Python, такими як pandas для обробки даних та matplotlib для візуалізації, дозволяє ефективно виконувати комплексні аналізи.
* Оптимізована продуктивність: scikit-learn оптимізований для роботи з великими наборами даних, забезпечуючи швидке виконання алгоритмів.
* Документація та приклади: Відмінна документація та численні приклади коду допомагають швидко освоїти фреймворк та застосовувати його для різних задач.

*Недоліки:*

* Обмежена підтримка глибокого навчання: scikit-learn не призначений для розробки складних моделей глибокого навчання, таких як нейронні мережі, що обмежує його використання у деяких проектах.
* Менш підходить для великих даних: Хоча scikit-learn може працювати з великими наборами даних, він може бути менш ефективним порівняно з іншими фреймворками, такими як TensorFlow чи PyTorch, для дуже великих обсягів даних.
* Відсутність підтримки GPU: scikit-learn не оптимізований для роботи з GPU, що може уповільнити процес тренування моделей на великих даних.
* Менша гнучкість для налаштування моделей: Порівняно з TensorFlow чи PyTorch, scikit-learn має менше можливостей для тонкого налаштування моделей та створення складних архітектур.

## **Клієнтська частина застосунку**

### ***React***

Популярна бібліотека JavaScript для створення користувацьких інтерфейсів, особливо для вебзастосунків. Він розроблений і підтримується компанією Meta і має велику спільноту розробників [9].

Основні компоненти React:

* Components: React побудований навколо концепції компонентів. Кожен компонент має власну логіку та керує власним вимальовуванням, і його можна повторно використовувати у всій програмі, що допомагає підтримувати більші програми з великою кількістю рухомих частин.
* JSX: React використовує JSX, розширення синтаксису для JavaScript. JSX дозволяє писати HTML-структури в тому ж файлі, що і JavaScript-код, що робить структуру компонентів зрозумілою і лаконічною.
* Virtual DOM: React створює віртуальний DOM в пам’яті. Замість того, щоб маніпулювати DOM браузера безпосередньо, React створює віртуальне DOM-представлення інтерфейсу, яке потім відрізняється від попереднього стану та оновлює справжній DOM лише змінами.
* Потік даних: React призначений для односпрямованого потоку даних. Це робить логіку застосунку більш передбачуваною та зрозумілою. Це також полегшує налагодження.
* Hooks: представлені у версії 16.8, хуки – це функції, які дозволяють вам «підключатися» до стану React та функцій життєвого циклу з функціональних компонентів. Хуки не працюють всередині класів – вони дозволяють використовувати React без класів.
* Context API: це забезпечує спосіб передавати дані через дерево компонентів без необхідності передавати пропси вручну на кожному рівні.

*Переваги:*

* Ефективність: React використовує віртуальний DOM для мінімізації прямого маніпулювання DOM, що може бути вузьким місцем продуктивності у високонавантажених застосунках.
* Багаторазові компоненти: компоненти можуть бути розроблені як окремі блоки, які керують власним станом і елементами керування. Таке багаторазове використання полегшує розробку та підтримку великих вебзастосунках.
* Потужна підтримка спільноти: наявність сильної підтримки Meta і величезної спільноти розробників означає, що доступно багато ресурсів, інструментів і плагінів, які допоможуть вирішити проблеми та покращити процес розробки.
* Гнучкість: React можна використовувати як основу при розробці односторінкових або мобільних застосунків. Він також сумісний з іншими фреймворками.
* Покращена підтримуваність: компонентна архітектура React допомагає легше підтримувати кодову базу в міру масштабування проєктів.

*Недоліки:*

* Крива навчання: хоча сам React не складний для розуміння, екосистема навколо нього, включаючи Redux, Webpack, Babel та інші, може бути приголомшливою для початківців.
* Швидкий темп розробки: середовище навколо React змінюється досить швидко, що може бути палицею з двома кінцями. Постійні оновлення та вдосконалення корисні, але можуть ускладнити встигання за останніми змінами.
* JavaScript XML (JSX): JSX – це синтаксис, який поєднує HTML з JavaScript, що може збивати з пантелику нових розробників і є відхиленням від традиційного JavaScript.
* Накладні витрати: створення проєкту лише за допомогою React часто недостатньо для складних застосунків. Можливо, вам доведеться включити кілька інших бібліотек та інструментів, щоб отримати повне робоче середовище, що може збільшити складність і час розробки.
* Проблеми пошукової оптимізації: односторінкові застосунки, створені за допомогою React, можуть зіткнутися з проблемами пошукової оптимізації, оскільки контент завантажується динамічно.

### ***Angular***

Angular – це платформа і фреймворк для створення односторінкових клієнтських застосунків з використанням HTML і мови TypeScript. Розроблений і підтримуваний Google, Angular надає надійний набір функцій для розробки сучасних вебзастосунків [10].

Складові Angular:

* TypeScript: Angular використовує TypeScript, надмножину JavaScript, яка додає статичну типізацію та такі функції, як інтерфейси, декоратори та суворі перевірки null.
* Компоненти та директиви: компоненти керують частиною екрану і складаються з шаблону HTML, стилів CSS і класу TypeScript. Директиви використовуються для розширення HTML шляхом додавання нетипової поведінки.
* Модулі: модулі організовують застосунок у цілісні блоки функціональності. Кожен застосунок Angular має як мінімум один модуль – кореневий модуль.
* Dependency Injection (DI): Angular має вбудовану систему ін’єкції залежностей, яка допомагає створювати об’єкти в різних частинах програми.
* Двостороння прив’язка даних: Angular забезпечує двостороннє зв’язування даних між представленням і моделлю, забезпечуючи синхронізацію в реальному часі.
* Маршрутизатор: маршрутизатор Angular забезпечує навігацію з одного подання в інше, коли користувачі виконують завдання програми. Він підтримує відкладене завантаження модулів для кращої продуктивності.
* Форми: Angular має два різні підходи до створення форм: реактивні форми та форми, керовані шаблонами, що дає розробникам гнучкість у роботі з даними форм.
* RxJS: Angular використовує RxJS для реактивного програмування, полегшуючи обробку асинхронних потоків даних.
* CLI: Angular CLI (Command Line Interface) є потужним інструментом для ефективної розробки та підтримки застосунків Angular.
* Тестування: Angular надає утиліти для тестування, такі як «Karma» та «Jasmine» для модульного тестування, а також «Protractor» для наскрізного тестування.
* Інтернаціоналізації (i18n): вбудована підтримка декількох мов, що полегшує інтернаціоналізацію застосунків.

*Переваги:*

* Комплексність: містить широкий спектр інструментів і функцій із коробки.
* Модульність: модульна архітектура робить його масштабованим і підтримуваним.
* Можливість супроводу: потужна типізація та компонентний дизайн покращують якість коду.
* Підтримка спільноти та корпорацій: потужна підтримка з боку Google та великої спільноти.
* Інструменти: відмінні інструменти для розробки, тестування та налагодження.
* Інтеграція з .NET: бездоганно працює з серверними службами .NET, розширюючи можливості розробки повного стека за допомогою надійних, масштабованих рішень.
* Двостороння прив’язка даних: автоматично синхронізує модель і представлення.
* Оптимізація продуктивності: такі функції, як AoT-компіляція і модель «tree shaking», мінімізують розмір програми та підвищують продуктивність.
* Послідовність: Angular забезпечує сувору структуру та шаблон у всій програмі, що покращує ремонтопридатність.
* Корпоративна підтримка: Angular підтримується Google і використовується багатьма великими корпораціями, такими як Microsoft і Adobe.

*Недоліки:*

* Складність: крута крива навчання завдяки своєму всебічному характеру.
* Детальність: вимагає більше шаблонного коду, ніж деякі інші фреймворки.
* Продуктивність: може бути повільнішою порівняно з легшими фреймворками для простих програм.
* Важкий: більший розмір кодової бази може призвести до більших початкових навантажень.
* Імперативний у використанні: наказує, як структурувати та розробляти застосунок.

### ***Azure AI***

Azure AI від Microsoft є інтегрованим набором облачних служб, які забезпечують розширені можливості для створення, тренування, і впровадження ШІ моделей.

Складові Angular:

* Інтеграція з хмарними сервісами: Azure AI добре інтегрується з іншими хмарними сервісами Azure, забезпечуючи гнучкість та масштабування.
* Безпека та відповідність нормам: Azure забезпечує високий рівень безпеки даних, включаючи дотримання регулятивних вимог.
* Різноманітність ШІ-інструментів: Платформа пропонує широкий спектр інструментів для машинного навчання, когнітивних служб (включаючи зір, мову, і розуміння мови), і автоматичного рішення завдань..

*Переваги:*

* Інтеграція з хмарними сервісами: Azure AI добре інтегрується з іншими хмарними сервісами Azure, забезпечуючи гнучкість та масштабування.
* Безпека та відповідність нормам: Azure забезпечує високий рівень безпеки даних, включаючи дотримання регулятивних вимог.
* Різноманітність ШІ-інструментів: Платформа пропонує широкий спектр інструментів для машинного навчання, когнітивних служб (включаючи зір, мову, і розуміння мови), і автоматичного рішення завдань.

*Недоліки:*

* Вартість: Використання хмарних ШІ служб може бути відносно дорогим, особливо при великому обсязі даних і високих вимогах до обчислювальних ресурсів.
* Залежність від платформи: Захисний ефект хмарної платформи може бути проблемою для організацій, які мають жорсткі політики щодо зберігання даних.

## **Висновки**

Після детального розгляду доступних технологій для серверної та клієнтської частин веб-застосунку, критично важливо врахувати фактори, які впливають на реалізацію бізнес-логіки, стабільність, масштабованість та зручність обслуговування системи. Виходячи з вимог проекту, який орієнтований на забезпечення високого рівня інтерактивності та доступності сервісних послуг через голосовий асистент та мультимедійного помічника, обрані технології ASP.NET Core, Angular та Azure або Python представляють собою оптимальний вибір.

Серверна частина застосунку на базі ASP.NET Core дозволяє ефективно реалізувати бізнес-логіку, забезпечуючи чистоту коду, модульність та легкість внесення змін. ASP.NET Core, як високопродуктивний фреймворк, надає потужні засоби для розробки вебсервісів, які легко масштабуються і оптимізовані для хмарних рішень.

Клієнтська сторона, реалізована за допомогою Angular, є виправданим вибором з огляду на його модульність та глибоку інтеграцію з TypeScript, який додає строгу типізацію, зменшуючи кількість помилок на етапі компіляції і покращуючи читабельність коду. Angular забезпечує сильну структуру застосунку, що дозволяє ефективно управляти станом застосунку та взаємодією з користувачем через розширене управління залежностями та легке тестування компонентів.

Використання Azure або Python для реалізації засобів штучного інтелекту дає змогу інтегрувати розширені алгоритми обробки природної мови та інші ШІ-можливості, які підвищують функціональність системи. Azure AI надає потужні облачні інструменти для масштабованості та ефективного управління ресурсами, тоді як Python забезпечує гнучкість у розробці алгоритмів машинного навчання з великою кількістю доступних бібліотек.

Таким чином, комбінація Angular, ASP.NET Core, та Azure або Python дозволяє створити потужну, надійну та гнучку інформаційну систему, яка забезпечує ефективність, високу продуктивність і зручність у взаємодії з користувачами, що є ключовими аспектами для успішного впровадження та довготривалої підтримки веб-застосунків у сфері сервісних послуг.

# **СТРУКТУРНО-АЛГОРИТМІЧНА ОРГАНІЗАЦІЯ РОЗРОБЛЕНОГО ВЕБЗАСТОСУНКУ**

## **Архітектура вебзастосунку**

Система розділена за принципом «тонкого клієнту» на дві частини: клієнтська та серверна. Клієнтська частина: отримує дані від користувача, проводить попередню валідацію форми, проводить запити та отримує відповіді, у разі помилки видає повідомлення про помилку. Серверна частина: приймає запити від клієнта, валідує дані запиту, виконує аналіз за допомогою сервісу розпізнавання побудованого на основі штучного інтеллекту.

Структура застосунку зображено на рис. 3.1.

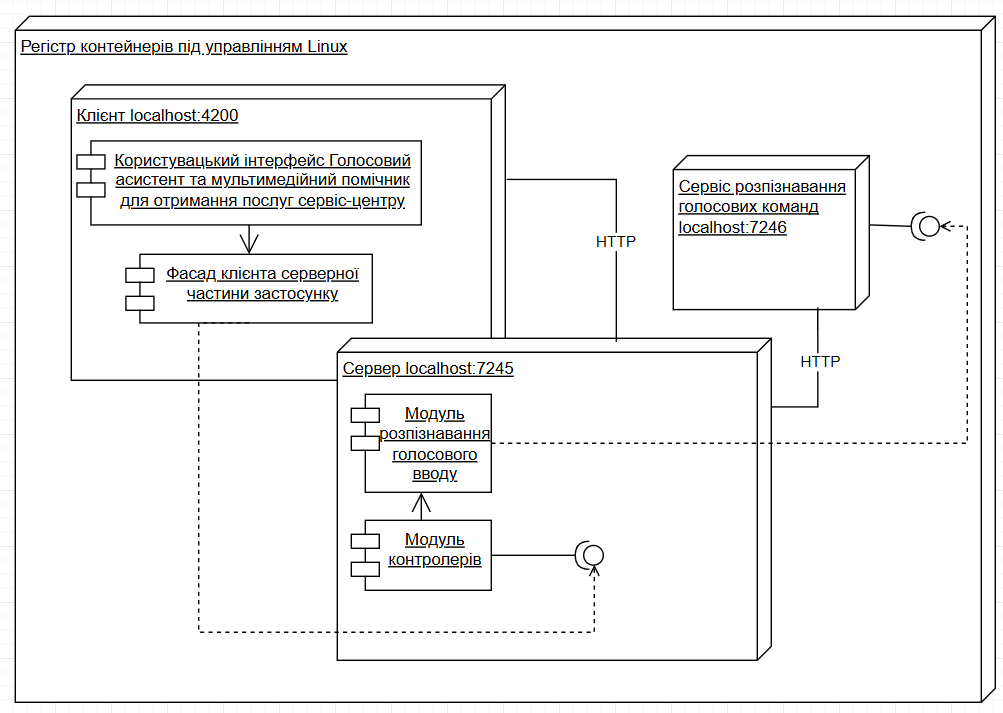


Рис. 3.1. Стутктура програмного застосунка

## **Структура та алгоритми предметної галузі**

Функціональність голосового асистента базується на інтеграції механізмів розпізнавання мови та інтерпретації запитів для виконання необхідних дій. Коли користувач вводить голосовий запит за допомогою браузера, система активує механізм захоплення голосових даних, які передаються на спеціалізовану службу для обробки. На цьому етапі використовується алгоритм розпізнавання мовлення, який перетворює звукові дані у текстовий формат. Цей текст аналізується для виявлення ключових слів, структур та інструкцій, які потім трансформуються у відповідний запит до внутрішньої системи.

Система обробляє запит, визначаючи його категорію та призначення, після чого виконує відповідну функціональність: відображення списку продуктів, пошук конкретного товару чи виконання інших дій, таких як відкриття форми для подачі заяви. Результати обробки передаються назад користувачеві через візуальний або голосовий інтерфейс. Такий підхід дозволяє забезпечити зручність і швидкість взаємодії, мінімізуючи затримки та забезпечуючи точність виконання команд. Система є гнучкою, адаптивною та інтегрованою, що забезпечує її ефективність у роботі з клієнтськими запитами у сфері сервісних послуг.

## **Опис функціональних вимог**

Вебзастосунок, розроблений для поліпшення взаємодії клієнтів з сервіс-центром через впровадження голосового асистента, забезпечує широкий спектр функціональностей, що орієнтовані на автоматизацію та зручність користувачів. Система дозволяє користувачам вводити голосові команди через інтерфейс браузера, які потім передаються на сервер для обробки. Основні функціональні вимоги до вебзастосунку включають:

* Вебзастосунок активно захоплює голос через мікрофон користувача та відправляє аудіодані на сервер для подальшої обробки.
* Служба розпізнавання перетворює голосовий ввід у текст, який аналізується для здійснення подальших команд.
* Система аналізує текстові дані на наявність команд та виконує необхідну функціональність, таку як пошук інформації про продукт, відображення інструкцій або створення сервісних заявок.
* Користувач отримує відповідь від системи у вигляді тексту або голосу, залежно від виконаної дії.
* Система перевіряє вводи на наявність помилок або невизначених команд і повідомляє користувача про необхідність корекції.
* Вебзастосунок розроблений з урахуванням сучасних вимог до дизайну інтерфейсів, що забезпечує зручність та інтуїтивність використання.
* Всі персональні дані та вводи користувачів захищені згідно з вимогами конфіденційності і безпеки даних.
* Архітектура вебзастосунку дозволяє обробляти велику кількість запитів одночасно, забезпечуючи стабільність і надійність при збільшенні кількості користувачів.

## **Висновки**

Розроблений вебзастосунок, який впроваджує голосового асистента для покращення взаємодії клієнтів з сервіс-центром, є втіленням сучасної інформаційної системи, організованої за принципом "тонкого клієнта". Клієнтська частина відповідає за прийом голосового вводу від користувачів, попередню валідацію та перетворення голосу в текст, а також за надсилання оброблених запитів до серверної частини і відображення отриманих результатів. Це забезпечує інтуїтивно зрозумілий інтерфейс з повідомленнями про помилки чи підтвердження виконаних дій. Серверна частина застосунку здійснює додаткову валідацію даних, обробляє голосові запити користувачів, виконує необхідні дії відповідно до визначених команд і відправляє результати назад у клієнтську частину. Вона використовує передові технології для аналізу мовлення та інтеграції з внутрішніми системами сервісу, забезпечуючи швидку та точну відповідь на запити користувачів. Система спроєктована таким чином, що може ефективно масштабуватися для обробки великої кількості одночасних запитів, забезпечуючи високу продуктивність і надійність без зниження якості обслуговування. Крім того, вона відповідає сучасним стандартам безпеки даних, гарантуючи захист і конфіденційність інформації користувачів. Завдяки цій структурі та використанню сучасних технологій, вебзастосунок не тільки ефективно вирішує поставлені задачі, але й забезпечує зручність, швидкість обслуговування та можливість динамічної адаптації до змінних потреб сервісного центру. Це робить систему цінним інструментом для покращення якості обслуговування та підвищення задоволеності клієнтів.

# **ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

## **Особливості реалізації клієнтської частини вебзастосунку**

Клієнтська частина вебзастосунку розроблена з використанням фреймворку Angular, що забезпечує зручний і інтуїтивно зрозумілий інтерфейс. Завдяки цьому користувачі можуть легко вводити голосові команди, які перетворюються на текст для подальшої обробки. Ці команди формуються в HTTP-запити, які надсилаються на сервер через протокол HTTP, використовуючи асинхронні методи та реактивне програмування за допомогою бібліотеки RxJS. Такий підхід дозволяє не тільки ефективно обмінюватися даними з сервером, але й забезпечує швидку відповідь системи на дії користувача, що є особливо важливим при роботі з голосовими командами.

Компоненти Angular на клієнтській стороні займаються відображенням інтерфейсу у веб-браузері, забезпечуючи інтерактивність та візуальну привабливість вебзастосунку. Вони також виконують валідацію вводу, перш ніж надсилати дані на сервер, і обробляють відповіді від сервера для відображення результатів користувачу. Використання реактивного програмування допомагає управляти потоками даних та змінами стану в додатку, що робить користувацький інтерфейс динамічним і здатним швидко реагувати на зміни.

## **Особливості реалізації серверної частини вебзастосунку**

Серверна частина вебзастосунку, створена за допомогою ASP.NET Core Web API, є відповідальною за обробку вхідних запитів від клієнтської частини, які передаються через HTTP. Сервер здійснює валідацію та обробку отриманих даних, а також інтегрується з зовнішніми службами для розпізнавання голосу і обробки команд. Після обробки команд сервер формує відповіді, які надсилаються назад до клієнтської частини, де вони представляються користувачу у зрозумілому і корисному форматі.

## **Висновки**

В результаті аналізу можливостей реалізації клієнтської та серверної частин вебзастосунку можна зробити висновок про злагоджену роботу цієї системи. Комбінація Angular на клієнтській стороні та ASP.NET Core Web API на серверній стороні дозволяє створити ефективний, надійний та легко масштабований вебзастосунок. Ця технологічна інфраструктура забезпечує високу швидкість обробки запитів, що є критично важливим для систем, що працюють з голосовими вводами та командами. Використання сучасних підходів до розробки також сприяє зручності та безпеці користувачів, що підвищує задоволеність від користування системою і може значно покращити взаємодію між користувачами та сервісним центром.

# **ВИСНОВКИ**

У ході розробки вебзастосунку для покращення взаємодії клієнтів з сервіс-центром через голосовий асистент та мультимедійний помічник було створено сучасну інформаційну систему, яка відповідає сучасним вимогам функціональності, продуктивності та зручності використання. Система побудована за принципом "тонкого клієнта", з чітким розподілом обов'язків між клієнтською та серверною частинами. Клієнтська частина, реалізована на базі фреймворку Angular, забезпечує легкий та інтуїтивно зрозумілий інтерфейс для вводу голосових команд, їх валідацію та передачу до серверної частини, а також відображення результатів обробки. Застосування бібліотеки RxJS дозволяє реалізувати реактивний підхід до обробки даних, що підвищує швидкість реакції інтерфейсу та надійність клієнтської частини.

Серверна частина, розроблена з використанням ASP.NET Core, займається обробкою вхідних HTTP-запитів, валідацією отриманих даних, взаємодією зі службою розпізнавання голосу та обробкою команд, і надсилає результати назад клієнту. Архітектура серверної частини спроектована так, щоб забезпечити модульність, масштабованість і простоту у подальшому розвитку та підтримці системи. Функціональні вимоги до системи, такі як прийом та аналіз голосових команд, швидка обробка даних, захист і конфіденційність користувацької інформації, були успішно реалізовані. Розроблені алгоритми та обрані технології дозволяють системі ефективно виконувати свої функції, бути гнучкою до змінних вимог бізнесу і забезпечувати свою актуальність на довгий термін.

Загалом, розроблений вебзастосунок є високоефективним інструментом для взаємодії клієнтів із сервіс-центром, сприяючи поліпшенню якості обслуговування і підвищенню задоволеності клієнтів. Обрана архітектура та технологічні рішення забезпечують високу продуктивність, гнучкість і легкість у використанні, роблячи систему конкурентоспроможною і перспективною для широкого впровадження.

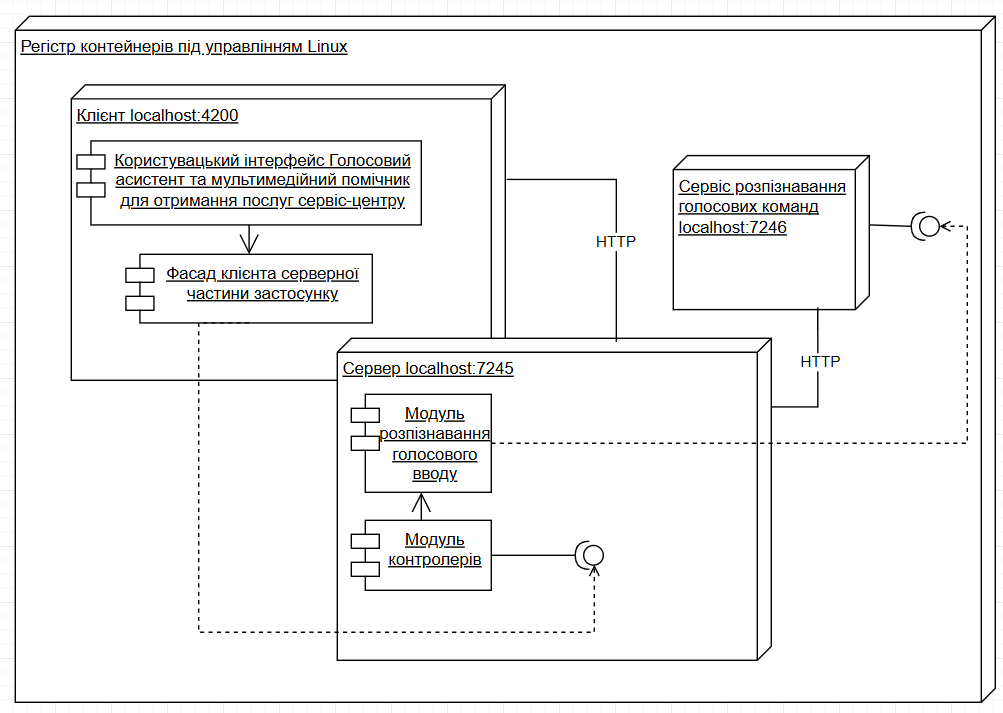
# **СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ**

1. Amazon Alexa. [Електронний ресурс]. — Режим доступу: https://alexa.amazon.com/ — (12.12.2024)
2. Google Assistant [Електронний ресурс]. — Режим доступу: https://assistant.google.com/ — (12.12.2024).
3. Siri [Електронний ресурс]. — Режим доступу: https://www.apple.com/siri/ — (12.12.2024).
4. YouTube [Електронний ресурс]. Режим доступу: https://www.youtube.com/ — (12.12.2024).
5. C# documentation [Електронний ресурс]. — Режим доступу: https://learn.microsoft.com/en-us/dotnet/csharp/ — (12.12.2024).
6. NodeJs guide [Електронний ресурс]. — Режим доступу: https://nodejs.org/en/docs/guides — (02.03.2024).
7. ASP.NET CORE documentation [Електронний ресурс]. — Режим доступу: https://dotnet.microsoft.com/en-us/apps/aspnet — (05.03.2024).
8. Насколько быстр ASP.NET Core? [Електронний ресурс]. — Режим доступу: https://habr.com/ru/articles/701352/ — (23.03.2024).
9. React documentation [Електронний ресурс]. — Режим доступу: https://react.dev/ — (12.12.2024).
10. Angular documentation [Електронний ресурс]. — Режим доступу: https://angular.io/docs — (12.12.2024).

# **ДОДАТКИ**

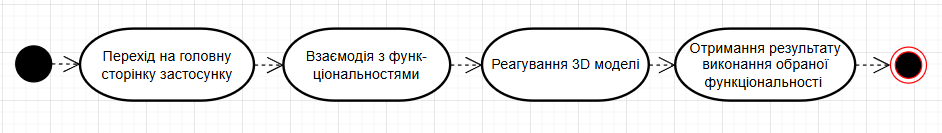
# **Додаток 1**

# **Копії графічних матеріалів**

****

ДП.045440-06-99

Голосовий асистент та мультимедійний помічник для отримання послуг сервіс-центру. Стутктура програмного застосунка. Діаграма компонентів

****

ДП.045440-07-99

Голосовий асистент та мультимедійний помічник для отримання послуг сервіс-центру. Схема алгоритму взаємодії користувача з системою. Діаграма активності

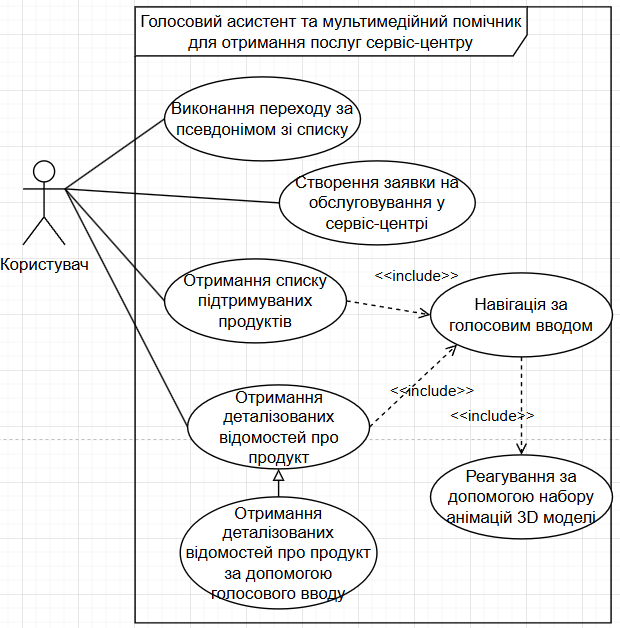
****

Схема функціональності інформаційної системи.

Хоменко М.В., група КПмн-41