НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
„Київський політехнічний інститут імені Ігоря Сікорського”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

“ЗАТВЕРДЖЕНО”

Керівник роботи

_____ Євгенія СУЛЕМА

“___” _____ 2024 р.

ПЕРСОНАЛЬНИЙ АСИСТЕНТ ДЛЯ ПІДБОРУ ФІЛЬМІВ З ВИКОРИСТАННЯМ МУЛЬТИМЕДІЙНОГО ПОМІЧНИКА ТА ГОЛОСОВОГО АСИСТЕНТА. АРХІТЕКУТРА СИСТЕМИ ТА БЕКЕНД РОЗРОБКА

Пояснювальна записка

Виконавець:

_____ Роман ДОВЖЕНКО

2024

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

REST (скор. від англ. Representational State Transfer) – стиль архітектури для розроблення вебсервісів, що дозволяє передавати дані між клієнтом і сервером.

HTTP – HyperText Transfer Protocol (протокол передачі даних, що використовується в комп'ютерних мережах).

HTML – HyperText Markup Language (мова тегів, засобами якої здійснюється розмічання вебсторінок для мережі Інтернет).

Фреймворк – Framework (програмне середовище, яке спрощує та прискорює створення програмного забезпечення);.

API (скор. від англ. application programming interface) – прикладний програмний інтерфейс, який надає набір чітко визначних правил для взаємодії одних програм з іншими.

DOM – Document Object Model (стандартний спосіб представлення вебсторінок за допомогою набору об'єктів, що визначає набір інтерфейсів, незалежних від платформи і мови, який дозволяє програмам і сценаріям динамічно змінювати структуру, вміст і стиль документів).

AJAX – Asynchronous JavaScript And XML (технологія, яка дозволяє відправляти та отримувати дані з серверу без перезавантаження вебсторінки).

ВСТУП

На сьогоднішній день існує достатньо програмного забезпечення, яке застосовується в різних сферах життя. Очевидним є той факт, що розробники та дизайнери прагнуть створити такий застосунок, який би був найбільш зручним для використання кінцевими користувачами, тому що в світі загалом існує досить багато програмного забезпечення майже для всіх можливих задач, але користуються попитом лише ті програми, які виконують свої функціональні можливості за мінімальну кількість взаємодій, і причому коли ці взаємодії є інтуїтивними та зрозумілими. Тому в останні роки можна побачити стрімкий розвиток голосових асистентів, за допомогою яких можна швидко виконати будь-які дії з однієї «точки входу».

Даний проєкт присвячений розробленню персонального асистента для підбору фільмів, який об’єднує в собі мультимедійного помічника, функції голосового управління та систему персоналізованих рекомендацій. Однією з ключових складових цієї системи є модуль голосового інтерфейсу, який відповідає за розпізнавання голосових запитів, їх обробку та озвучування відповідей. Голосовий інтерфейс забезпечує користувачам можливість швидкого і зручного доступу до функцій пошуку та рекомендації фільмів, роблячи процес використання системи простішим та більш інтерактивним.

Метою даної роботи є розробка та реалізація інтеграційних рішень, які дозволять об'єднати різнорідні технології (Node.js, Angular, Google API, Llama3) в єдину систему для підбору фільмів за голосовими запитами користувачів. Основні задачі, які вирішуються в рамках цього завдання:

- вибір відповідних технологій та бібліотек для реалізації логіки програми;
- реалізація механізмів передачі даних: Визначення форматів обміну даними та розробка механізмів їх передачі між сервером і клієнтом.

- забезпечення синхронності роботи компонентів: Розробка механізмів синхронізації та координації роботи різних частин системи.

Розроблений модуль інтеграції компонентів програмного продукту має забезпечити інтуїтивно зрозумілий інтерфейс для користувача, швидку реакцію на голосові команди та персоналізовані рекомендації.

1. АРХІТЕКТУРА ВЗАЄМОДІЇ КОМПОНЕНТІВ

1.1 . Вибір клієнт-серверної моделі для проєкту

Архітектура програмного продукту – це сукупність принципів, що описують, яким чином система повинна бути розділена на модулі і як ці компоненти мають бути пов’язані між собою. У сучасному світі розвитку ІТ існує велика кількість типів архітектури ПЗ, які широко застосовуються при створенні систем. При виборі архітектури важливо звертати увагу на декілька факторів, наприклад:

1) функціональні та нефункціональні вимоги до системи;

2) терміни розроблення ПЗ;

3) масштабованість, що дозволятиме додавати нові функції.

Для реалізації голосового асистента обрано клієнт-серверну архітектуру, в основі якої лежить розподілення системи на дві частини: сервер і клієнт. Сервер відповідає за постачання ресурсу, а клієнт – за запит його. Комунікація між цими частинами відбувається за допомогою протоколу запит-відповідь. Насамперед клієнт формує запит з певними даними і надсилає це повідомлення серверу. Сервер отримує запит, обробляє дані всередині нього, формує відповідь з успішного результату або помилки і надсилає клієнту. Нарешті клієнт отримує повідомлення-відповідь і обробляє його.

Серед особливостей клієнт-серверної архітектури важливо виділити наступне:

1) розділення функціональності між серверною і клієнтською частинами, завдяки чому жоден з компонентів не є перевантаженим непотрібними даними або складними обчисленнями;

2) централізований доступ до даних;

3) адаптивність – можливість збільшення кількості клієнтів та серверів;

4) захищеність даних;

5) простота у роботі з файлами завдяки тому, що вони зберігаються на одному сервері;

6) відсутність залежності від фреймворків;

7) простота у подальшому тестуванні;

8) зручність використання.

1.2 . Архітектура розроблюваної системи

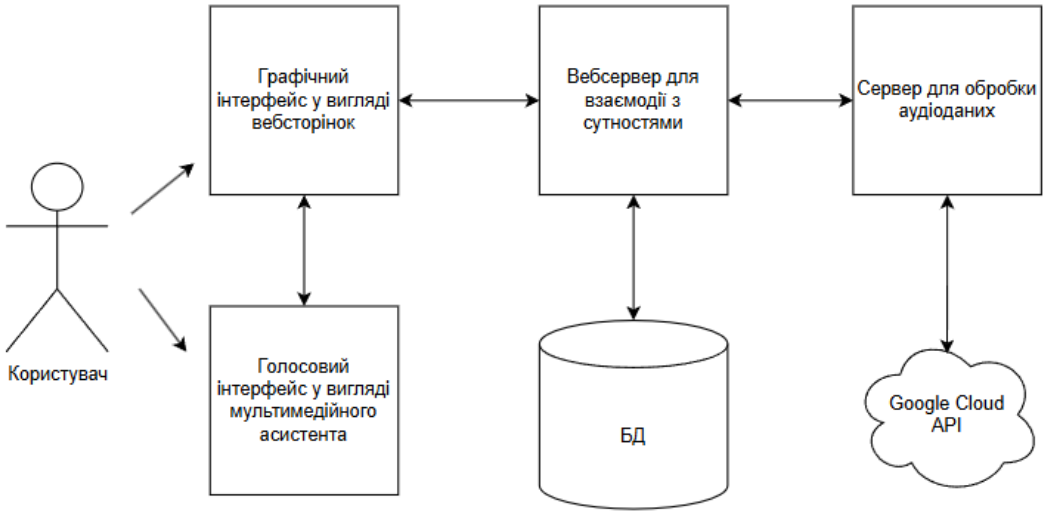

Рис. 1.1. Архітектура розроблюваної системи

Серверна частина системи складається з двох основних модулів:

1. Веб-сервер: Цей модуль відповідає за всі операції з базою даних, такі як реєстрація, авторизація користувачів, створення, редагування та видалення даних про фільми. Він також виконує логіку бізнес-процесів, наприклад, обробку запитів на пошук фільмів.
2. Сервер обробки аудіо: Цей модуль спеціалізується на роботі зі звуковими даними. Він перетворює голосові команди користувачів на

текстовий формат, щоб система могла їх зрозуміти, а також синтезує текстові відповіді у звукові файли для відтворення.

Клієнтська частина системи представлена у вигляді односторінкового веб-сайту. Вона складається з двох основних блоків:

1. Графічний інтерфейс: Цей блок відповідає за візуальну частину сайту, тобто за те, що бачить користувач на екрані. Він відображає різні сторінки, елементи управління та дані про книги.
2. Голосовий інтерфейс: Цей блок забезпечує інтерактивну взаємодію користувача з системою за допомогою голосу. Він дозволяє користувачеві виконувати різні дії, такі як пошук фільмів, перехід між сторінками сайту, не використовуючи клавіатуру та мишу.

Таким чином, система працює наступним чином: Користувач взаємодіє з голосовим інтерфейсом, вимовляючи свої запити. Голосовий інтерфейс перетворює голосові команди в текстовий формат і передає їх на сервер обробки аудіо. Сервер обробляє запит і надсилає його веб-серверу. Веб-сервер виконує необхідні операції з базою даних і формує відповідь, яку потім передає назад на клієнтську частину. Клієнтська частина синтезує текстову відповідь у звуковий файл і відтворює його для користувача, а також оновлює графічний інтерфейс відповідно до результатів пошуку або інших дій.

2. ОБГРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РОЗРОБЛЕННЯ

2.1 Обґрунтування засобів розробки для серверної частини

На даному етапі потрібно сформулювати основні вимоги до засобів реалізації програмного продукту, провести структурований аналіз обраних засобів, навести список особливостей кожного з інструментів та скласти таблиці для їх порівняння за сформованими критеріями.

Як було сказано раніше, система голосового асистенту поділена на дві основні частини: клієнтську та серверну. Таке розділення забезпечить високу ефективність роботи застосунку, оскільки даний підхід значно зменшить завантаженість клієнтської частини. Крім того, це підвищує безпеку системи, оскільки сервер може виконувати авторизацію та перевірку прав доступу користувачів до конфіденційної інформації, що, у свою чергу, зменшує ризик її витоку.

Мова програмування, за допомогою якої реалізовано серверну частину застосунку, повинна задовольняти наступні висунуті до неї вимоги:

- підтримка асинхронного програмування;
- наявність модульності;
- великий вибір модулів та бібліотек;
- зручність управління залежностями;
- легкість розширювання;
- можливість використовувати один і той самий код для розроблення серверної та клієнтської частини;
- наявність великої кількості документації та ресурсів для вивчення.

JavaScript – об’єктно-орієнтована динамічна мова програмування. Найчастіше вона асоціюється з розробленням клієнтської частини, оскільки дозволяє створювати інтерактивні користувацькі інтерфейси застосунків та динамічні вебсторінки. До використання JavaScript вебсторінки були статичні і для того, щоб оновити вигляд сторінки, користувачеві потрібно було виконати запит на отримання нової версії з серверу та перезавантажити браузер. Сучасні вебсторінки, використовуючи підхід AJAX, отримують актуальну інформацію з серверу без мануального оновлення браузеру. Станом на початок 2023 року, більше 97% вебсайтів створені за допомогою JavaScript.

Проте дана мова програмування нерідко зустрічається і як засіб реалізації серверної частини застосунків. Середовище виконання Node.js розширило можливості JavaScript і перетворило її з технології для створення лише клієнтської частини застосунків на мову програмування загального вжитку. Завдяки платформі Node.js код JavaScript може використовуватись для створення як клієнтської, так і серверної частин програми, що значно полегшує та пришвидшує процес розроблення системи.

Серед переваг JavaScript виділено наступні:

1) можливість її використання для розроблення повного стеку;

2) наявність фреймворку для побудови вебсерверу;

3) підтримка асинхронного програмування, завдяки чому сервер матиме змогу обробляти декілька запитів одночасно, не очікуючи завершення попереднього;

4) висока швидкість виконання – оскільки компіляція та виконання JavaScript-коду відбувається на сервері, запуск програми займатиме значно менше часу, ніж якби його потрібно було завантажувати та інтерпретувати у браузері;

5) підтримка модульної організації, яка полегшує розуміння, зберігання та повторне використання написаного коду;

6) обробка винятків;

7) наявність збирача сміття.

Незважаючи на переваги JavaScript, наявні і недоліки даної мови програмування, наприклад:

1) збільшення розміру та часу розгортання програми через використання фреймворків та бібліотек;

2) динамічна типізація – хоча іноді це пришвидшує процес написання коду, оскільки розробникам не потрібно обов’язково вказувати тип оголошених

змінних, це може призвести і до негативних наслідків під час виконання програми;

3) можливе виникнення труднощів при налаштуванні платформи Node.js для виконання на сервері програми, написаної мовою JavaScript.

Підсумовуючи, можна сказати, JavaScript задовольнила усі поставлені вимоги до засобу реалізації серверної частини програми. Вона має не лише фреймворк для побудови вебсерверу, але й великий вибір інших модулів та бібліотек, що дозволяють розробникам використовувати вбудовані можливості мови і економити час на створення подібних функцій. Що стосується наявності документації, то на просторах Інтернету можна знайти велику кількість ресурсів для вивчення JavaScript, що є досить важливою перевагою.

2.2 . Обґрунтування засобів розробки для клієнтської частини

Найпопулярнішими технологіями для розроблення клієнтської частини застосунків є React.js, Angular та Vue.js. В нашому проєкті будемо використовувати саме Angular.

Технологія для реалізації клієнтської частини повинна задовольняти наступні вимоги:

1) підтримка компонентного підходу до архітектури;

2) висока швидкість оновлення стану сторінки;

3) лаконічність коду;

4) великий вибір бібліотек;

5) висока продуктивність.

Angular – фреймворк з відкритим кодом, що використовується для створення односторінкових або багатосторінкових застосунків, які визначаються швидкістю та продуктивністю роботи.

Як і бібліотека React.js, Angular підтримує компонентний підхід до архітектури, дозволяючи повторно використовувати написаний код у різних місцях системи. Таким чином, розробники витрачають менше часу та зусиль на створення системи, оскільки вони маніпулюють не окремими рядками коду, а цілими компонентами або його частинами, що можуть легко масштабуватись у майбутньому завдяки їхній автономності. Крім економії часу, такий підхід допомагає інженерам ПЗ підтримувати порядок у коді, розбиваючи його на компоненти, що відповідають за різні функціональності.

Говорячи про рівень продуктивності застосунків, що використовують Angular у якості технології для розроблення клієнтської частини, важливо зазначити, що він є досить високим. Оскільки браузер не розуміє компонентів та HTML-шаблонів, з яких складається застосунок Angular, він вимагає виконання компіляції. Швидка візуалізація у браузері досягається завдяки AOT Compilation – компіляції коду Angular у JavaScript на етапі збірки програми ще до запуску застосунку у браузері. Таким чином, процес завантаження і запуску проходитиме значно швидше, що матиме позитивний вплив на продуктивність роботи програми.

Іншою важливою особливістю Angular є те, що даний фреймворк може взаємодіяти з необмеженою кількістю інших інструментів. Проте при підключенні інших бібліотек важливо перевіряти їх на сумісність, оскільки через неправильне їхнє налаштування можуть виникнути проблеми.

Отже, серед особливостей фреймворку Angular можна виділити високу продуктивність, використання декларативної парадигми програмування, підтримку компонентного підходу до архітектури та необмежену кількість

бібліотек для взаємодії та даний фреймворк чудово підійде для розробки нашого асистенту.

2.3 . Взаємодія з Google API для збирання даних про фільми

Інтеграція з Google API є важливим компонентом для забезпечення ефективного функціонування програмного продукту у сфері збору та обробки даних про фільми. Google API, зокрема Google Custom Search API, дозволяє здійснювати пошук релевантної інформації у веб-просторі, використовуючи структуровані запити. Застосування цієї технології дозволяє нашому продукту автоматизувати процес збору даних про фільми, що є критично важливим для коректної роботи системи рекомендацій.

Процес взаємодії з Google API можна розбити на кілька основних етапів:

1. Ініціалізація API-запиту:
   Google Custom Search API використовується для отримання структурованих даних про фільми. Спершу слід налаштувати обліковий запис Google Cloud і отримати ключ доступу до API.
2. Створення запиту:
   Запит формується у вигляді HTTP GET-запиту, в якому передаються наступні параметри:
   - q — текстовий рядок із пошуковим запитом (наприклад, назва фільму чи жанр).
   - key — ключ доступу до Google API.
   - cx — ідентифікатор Custom Search Engine (CSE), налаштованого для пошуку фільмів.

3. Обробка відповіді:

   Відповідь API надається у форматі JSON, що містить релевантні результати пошуку, включаючи:

   - заголовки фільмів;
   - описи та короткі синопсиси;
   - посилання на веб-ресурси;
   - зображення обкладинок.

4. Інтеграція з серверною логікою:

   Взаємодія з Google API відбувається через сервер, написаний на Node.js. Доступ до API виконується за допомогою стандартної бібліотеки fetch або пакету axios.

5. Формування результатів для кінцевого користувача:

   Отримані дані обробляються на сервері, фільтруються, структуруються і передаються у відповідь клієнту у вигляді JSON-об'єкта. У подальшому ці результати можуть бути використані для формування рекомендацій.

Взаємодія з Google API для збору даних про фільми дозволяє створити ефективну та масштабовану систему, що інтегрується із серверною логікою на Node.js. Реалізація використовує сучасні методи обробки HTTP-запитів та працює із JSON-відповідями. Застосування Google API забезпечує високу точність та актуальність отримуваної інформації, що є важливою умовою для систем рекомендацій.

3. ІНТЕГРАЦІЯ МУЛЬТИМЕДІЙНИХ КОМПОНЕНТІВ

3.1 . Зв’язок між мультимедійним помічником і логікою системи

Зв'язок між мультимедійним помічником та логікою системи в проєкті базується на інтеграції анімацій, що розроблені у графічному середовищі Blender,

з динамічним вебінтерфейсом, де помічник реагує на введені користувачем запити. Ця взаємодія забезпечується поєднанням JavaScript-функціоналу, REST API для логіки обробки запитів і відповідей, а також завантаженням анімованих відеофрагментів.

Інтерфейс користувача побудований на HTML, CSS та JavaScript. Реалізована інтерактивна область для текстового введення, виводу повідомлень і керування мультимедійним помічником.

Використовуються мультимедійні моделі, створені у Blender та збережені у форматі MP4 (з можливістю використання MKV).

Відео включають анімації різних станів помічника:

- animation_hi.mp4 — вітання;
- animation_looking.mp4 — обробка запиту;
- animation_confused.mp4 — запис голосу або нерозуміння;
- animation_standby.mp4 — режим очікування.

Логіка системи реалізована на Node.js (на сервері) з використанням API для обробки даних.

Запит передається через REST API на адресу http://localhost:3000/ask. Логіка сервера обробляє повідомлення користувача та повертає згенеровану відповідь.

Основні функції:

- switchVideo() — змінює анімацію помічника.
- sendMessage() — запускає анімацію animation_looking.mp4 перед відправкою запиту, після чого виконується переключення на animation_standby.mp4.
- startRecording() та stopRecording() — керують анімацією під час голосового введення.

Функція switchVideo() виконує завантаження різних станів анімації залежно від подій у системі:

- animation_looking.mp4 — коли система обробляє запит.
- animation_confused.mp4 — під час запису голосу.
- animation_standby.mp4 — коли помічник чекає на наступний запит.

Blender використовується для створення анімованих персонажів помічника. Формат відео: файли у форматі MP4 (з можливістю конвертації у MKV за потреби). Анімації забезпечують візуальний фідбек: перехід між різними станами дозволяє користувачу бачити реакцію помічника у реальному часі.

3.2 . Тестування розроблених компонентів

Тестування системи відбулося шляхом мануального тестування.

Для тестування було написано та протестовані такі варіанти:

Етап 1: Текстовий ввід запиту

1. Користувач вводить текст у поле та натискає кнопку Send.
2. Запускається функція sendMessage(), яка виконує наступні кроки:
    - Додає повідомлення користувача до чату через функцію appendMessage().
    - Завантажує відео animation_looking.mp4 (стан обробки запиту).
    - Відправляє запит на сервер через fetch().
3. Сервер обробляє запит і повертає відповідь у форматі JSON:

    { "reply": "Текст відповіді від помічника" }
4. Клієнтська частина отримує відповідь і:
    - Завантажує animation_standby.mp4 (режим очікування).
    - Виводить повідомлення через appendMessage().

- Додає кнопку для озвучення відповіді (через SpeechSynthesis API).

Етап 2: Голосовий ввід

1. Користувач натискає мікрофон: Викликається функція startRecording(), яка: запускає MediaRecorder API для запису голосу та завантажує анімацію animation_confused.mp4.
2. Завершення запису:
    - Викликається функція stopRecording().
    - Голосовий файл відправляється на сервер за допомогою FormData та POST запиту:
    - Сервер розпізнає текст з голосу та повертає результат.
3. Додаткові дії:
    - Отриманий текст обробляється аналогічно текстовому вводу:
        - Відправляється на сервер через /ask.
        - Відповідь додається до чату та озвучується.

3.3 . Рекомендації щодо подальшого вдосконалення

В процесі роботи над інтерактивною системою пошуку фільмів з використанням мультимедійного помічника та голосового асистента було виявлено ряд напрямів та функцій, які можуть покращити розроблене ПЗ, а саме:

- Оптимізація мультимедійного помічника: Використати більш ефективні алгоритми стиснення (наприклад, H.265), щоб зменшити навантаження на клієнтську частину та прискорити завантаження анімацій.
- Запровадити обробку відео та запитів у хмарному середовищі (наприклад, AWS або Google Cloud), щоб уникнути перевантаження локального серверу.

- Реакція на емоції користувача: Додати алгоритми розпізнавання тону голосу або міміки через вебкамеру (з використанням TensorFlow.js).

ВИСНОВКИ

У процесі розробки було успішно реалізовано інтеграцію компонентів програмного продукту, що забезпечує злагоджену взаємодію між клієнтською частиною, серверною логікою та мультимедійним помічником. Створений зв’язок дозволяє мультимедійному помічнику динамічно реагувати на запити користувачів завдяки зміні анімацій, обробці текстових і голосових запитів, а також генерації відповідей через серверну систему.

Для мультимедійного помічника використано високоякісні анімації, створені у Blender, які інтегруються через JavaScript і фреймворк Angular. Серверна частина, написана на Node.js, забезпечує обробку запитів користувачів, включаючи голосові та текстові, і генерує відповіді через інтеграцію з мовною моделлю Llama3. Це дозволяє створити інтерактивний та зручний досвід для користувача.

Важливим досягненням є реалізація механізмів автоматичної зміни анімацій залежно від стану системи. Впроваджена архітектура забезпечує масштабованість та можливість подальшого розширення функціональних можливостей помічника.

Успішна інтеграція усіх компонентів програмного продукту створює основу для надання користувачам сучасного, інтуїтивного та високоефективного інструменту для взаємодії з мультимедійним асистентом. Подальші вдосконалення можуть ще більше розширити функціонал системи та підвищити її продуктивність.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Millman, K.J. Python for Scientists and Engineers [Text] / K.J. Millman, M. Aivazis // Computing in Science & Engineering. — 2011. — Vol. 13, № 2. — P. 9 – 12.
2. The difference between Virtual DOM and DOM [Електронний ресурс]. — Режим доступу: https://reactkungfu.com/2015/10/the-differencebetween-virtual-dom-and-dom/
3. TypeScript Programming. [Електронний ресурс]. — Режим доступу: https://code.visualstudio.com/docs/languages/typescript
4. Client-Server Architecture. Advantages and Disadvantages of the Network Computing Model [Електронний ресурс]. — Режим доступу: 80 https://kitrum.com/blog/client-server-architecture-advantages-anddisadvantages/
5. Ahead-of-time (AOT) compilation [Електронний ресурс]. — Режим доступу: https://angular.io/guide/aot-compiler.
6. Moment.js [Електронний ресурс]. — Режим доступу: https://momentjs.com/.