ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розроблення веб-застосунку з використанням інтелектуальної моделі з можливістю голосового введення інформації»

на здобуття освітнього ступеня бакалавра
зі спеціальності 122 Комп'ютерні науки

освітньо-професійної програми Штучний інтелект

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Олег ЛУГОВИК

Виконав:
здобувач вищої освіти
група ШІД-41 Олег ЛУГОВИК

Керівник:
науковий ступінь, вчене звання Максим ФЕСЕНКО

Рецензент:
науковий ступінь, вчене звання _____

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Штучного інтелекту

Ступінь вищої освіти Бакалавр

Спеціальність 122 Комп'ютерні науки

Освітньо-професійна програма Штучний інтелект

ЗАТВЕРДЖУЮ
Завідувач кафедрою Штучного інтелекту

_____ Ольга ЗІНЧЕНКО
«_____» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Луговику Олегу Валерійовичу

1. Тема кваліфікаційної роботи: Розроблення веб-застосунку з використанням інтелектуальної моделі з можливістю голосового введення інформації

керівник кваліфікаційної роботи Максим ФЕСЕНКО к.т.н., доцент,

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» 02.2024р. № 36

2. Строк подання кваліфікаційної роботи «31» травня 2024р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
   Аналіз інтелектуальних систем
   Аналіз засобів розробки
   Розробка вимог до веб-застосунку
5. Перелік графічного матеріалу: *презентація*
   1. Вимоги до застосунку

2. Інтерфейс веб-застосунку
3. Голосове введення повідомлень
4. Інтеграція інтелектуальної моделі

6. Дата видачі завдання«27» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз сучасних інтелектуальних асистентів та їх можливостей	27.02-09.03.24	
2	Дослідження основних технологій веб-розробки	10.03-20.03.24	
3	Аналіз засобів розпізнавання природньої мови	21.03-28.03.24	
4	Розробка функціоналу веб-застосунку	29.03-03.05.24	
5	Інтеграція інтелектуальної моделі	03.05-13.05.24	
6	Тестування веб-застосунку	14.05-20.05.24	
7	Оформлення роботи: вступ, висновки, реферат	21.05-27.05.24	
8	Розробка демонстраційних матеріалів	28.05-31.05.24	

Здобувач вищої освіти _____ Олег ЛУГОВИК
(підпис) *)

Керівник
кваліфікаційної роботи _____ Максим ФЕСЕНКО
(підпис)

РЕФЕРАТ

Текстова частина бакалаврської: 58 стор., 9 рис., 22 джерел.

Мета роботи –створення веб-застосунку, який дозволяє користувачам отримувати потрібну інформацію або відповіді на запитання шляхом голосового введення повідомлень, для спрощення процесу знаходження інформації.

Об'єкт дослідження – процес знаходження інформації.

Предмет дослідження – веб-застосунок для пошуку інформації шляхом голосового введення повідомлень.

Короткий зміст роботи: У роботі проведено аналіз сучасних інтелектуальних асистентів, їх можливостей та інтеграції. Проаналізовано засоби для розпізнавання природньої мови для ефективної взаємодії з користувачами.

Розглянуто основні технології розробки веб-застосунку для знаходжння інформації з можливістю голосового введення повідомлень.

Використано засоби веб-розробки такі як JavaScript, CSS, HTML, а також фреймворк Django для розробки серверної частини застосунку. Розроблено аглоритм розпізнавання голосових повідомлень користувачів та передачі розпізнаного тексту на сервер.

КЛЮЧОВІ СЛОВА: PYTHON, DJANGO, PYCHARM, JAVASCRIPT, МОДЕЛЬ, API, ВЕБ-ЗАСТОСУНОК

ЗМІСТ

ВСТУП

У сучасному світі, завдяки появі інтелектуальних чат-ботів, таких як ChatGPT від OpenAI, Claude від Anthropic, Copilot від Microsoft та інших, проблема пошуку й отримання потрібної інформації досить швидко вирішується. Популярність таких систем досить висока і продовжує стрімко зростати, адже вони надають користувачам зручний спосіб отримувати відповіді на запитання в діалоговому режимі, замість того, щоб самостійно шукати інформацію в інтернеті.

До появи ChatGPT у листопаді 2022 року ми навіть не могли уявити, що таке інтелектуальний чат-бот і якого роду інформацію він нам надаватиме. Успіх ChatGPT став справжнім проривом у галузі штучного інтелекту та захопив увагу мільйонів людей по всьому світу. Вважається, що поява ChatGPT стала переломною точкою розвитку в цій галузі, адже після цього ряд інших великих світових компаній, таких як Microsoft, Google та інші, стали активно розробляти подібні продукти, які ми можемо побачити сьогодні.

Всі ці розумні чат-боти по-своєму унікальні в плані генерації відповідей – одна система генерує більш точну та правдиву інформацію, інша, навпаки, менш точну, але зміст у більшості випадків дуже схожий. Попри це, вони мають спільну рису – всі вони базуються на потужних мовних моделях, навчених на величезних обсягах тексту з інтернету.

Більшість з цих чат-ботів працюють лише в браузері та мають обмежений функціонал щодо введення – або тільки текст, або текст із вкладенням, наприклад, фото. У такому форматі вводити дані не завжди зручно, особливо для людей з певними фізичними вадами чи обмеженнями. Крім того, в деяких ситуаціях, наприклад, під час керування автомобілем або під час виконання певних завдань, коли обидві руки зайняті, набирати текст на клавіатурі стає незручно або навіть небезпечно.

Ціль проекту полягає в тому, щоб розширити взаємодію користувачів з розумним чат-ботом, а саме додати можливість надсилати повідомлення не

тільки в текстовому форматі, а й за допомогою голосу саме в браузері. Це відкриє нові можливості для зручного, швидкого та ефективного отримання потрібної інформації безпосередньо за допомогою голосового введення.

Для досягнення поставленої мети потрібно виконати наступні кроки:

1. Проаналізувати предметну область:
   - оглянути поняття інтелектуальної мовної моделі;
   - оглянути розумні чат-боти популярних компаній, які надають доступ через API до інтелектуальних мовних моделей.
2. Обрати засоби реалізації:
   - оглянути засоби розпізнавання природньої мови;
   - оглянути способи передачі тексту розпізнаваної мови зі сторони клієнта на сервер;
   - провести огляд мови програмування та інших технологій для реалізації візуальної частини сайту;
   - провести огляд мови програмування для реалізації серверної частини веб-застосунку
   - провести огляд фреймворку обраної мови програмування для реалізації серверної частини веб-застосунку;
   - провести огляд системи керування бази даних;
   - провести огляд системи керування версіями;
   - провести огляд середовища розробки.
3. Спроектувати та реалізувати веб-застосунок з використанням інтелектуальної моделі з можливістю голосового введення інформації:
   - побудувати діаграму діяльності;
   - провести огляд інтеграції інтелектуальної мовної моделі;
   - провести огляд основного механізму роботи;
   - провести тестування.

1 АНАЛІЗ ІНТЕЛЕКТУАЛЬНИХ СИСТЕМ І МОДЕЛЕЙ

1.1 Поняття інтелектуальної моделі

Інтелектуальна мовна модель або велика мовна модель(LLM-Large Language Model)- це тип програми штучного інтелекту, яка, окрім інших завдань, здатна розпізнавати та генерувати текст. Мовні моделі навчаються на величезних об'ємах даних та побудовані на основі машинного навчання, а саме на моделі трансформера. Модель трансформера- це набір нейронних мереж, кожна з якої складається з кодера і декодера з можливістю самоспостереження. Кодер і декодер отримують значення з послідовності тексту і розуміють співвідношення між словами та фразами.

Простіше кажучи, LLM — це комп’ютерна програма, яка накопичила достатньо шаблонів, щоб мати можливість розпізнавати та інтерпретувати людську мову чи інші типи складних даних.Багато мовних моделей навчаються на основі даних, зібраних з Інтернету, тисяч або мільйонів гігабайт тексту.Але якість вибірки впливає на те, як LLM вивчає природну мову, тому програмісти LLM можуть використовувати більш ретельно підібрані набори даних. LLM використовує тип машинного навчання, який називається глибоким навчанням, щоб зрозуміти, як символи, слова та речення працюють разом.Глибоке навчання передбачає аналіз неструктурованих даних, що в кінцевому підсумку дозволяє моделям глибокого навчання розпізнавати відмінності між фрагментами вмісту без втручання людини. Потім мовні моделі додатково навчаються за допомогою налаштування: вони точно налаштовуються або швидко адаптуються до конкретних завдань, які їм надають програмісти, наприклад інтерпретація запитання та створення відповідей або перекладу тексту з однієї мови на іншу.

Великі мовні моделі використовуються для виконання низки завдань. Одним з найпопулярніших завдань є розпізнання та генерація тексту, тобто коли користувач надає питання чи твердження, а мовна модель генерує текст у відповідь. Деякі мовні моделі можуть допомогти програмістам писати

програмний код, наприклад, написати функції за запитом програміста, або дописати, наданий програмістом, код.

Інтелектуальні мовні моделі можна використовувати в:

- обслуговуванні клієнтів;
- дослідженні ДНК;
- онлайн пошуку інформації;
- чат-ботах.

Прикладами реальних великих мовних моделей є ChatGPT, Claude, Bard, Liama та інші.

Ключовою особливістю великих мовних моделей є їх здатність відповідати на непередбачувані запити. Якщо брати за приклад звичайну комп'ютерну програму, то вона приймає команди лише в форматі, який заздалегіть прописаний. Відеогра має обмежений набір кнопок і функціонал який теж був заздалегіть прописаний за допомогою точних операторів if/then. Натомість LLM може реагувати на природню людську мову користувача та аналізувати дані щоб відповісти майже на будь-яке питання яке має сенс. Звичайна програма чи гра не дасть відповідь на запитання «Які найпопулярніші пісні 2021 року?», натомість мовна модель не тіки надасть список пісень, а й пояснить чому вони були найпопулярніші.

Проте є і деякі негативні моменти. Мовні моделі надають відповіді виходячи з тих даних, на яких вони були навчені, тобто вони можуть бути ненадійними. Якщо навчити мовну модель на даних, які були фальсивіковані або неправдиві, вона буде генеруваи відповіді тільки опираючись на ці дані, а отже відповідь моделі буде неправдивою і не буде відповідати дійсності. Також LLM інколи можуть надавати вигадану інформацію, навіть якщо були навчені на правдивій інформації. Це відбувається тоді, коли модель не може надати точну відповідь і генерує щось, що може бути схожим на правду.

З точки зору безпеки, програми, побудовані на основі LLM, також схильні до помилок як і будь-які інші комп'ютерні програми. Мовними моделями можна легко маніпулювати, надаючи їм дані, які є небезпечними або неетичними. Одна

з проблем безпеки є те, що користувачі можуть завантажувати власні конфіденційні дані щоб підвищити точність генерації відповідей для своїх цілей. Нажаль, на даних які надають користувачі, мовні моделі продовжують навчатися, тобто LLM не є безпечним сховищем для конфіденційних даних. Вони можуть розкривати конфіденційну інформацію на запити інших користувачів, що точно не є позитивним моментом.

На базовому рівні програми LLM побудовані на машинному навчанні.Машинне навчання є підмножиною штучного інтелекту і відноситься до практики передачі великих обсягів даних у програму, щоб навчити її визначати характеристики цих даних без втручання людини. LLM використовує тип машинного навчання, який називається глибоким навчанням.По суті, моделі глибокого навчання можуть самі навчитися розпізнавати відмінності без втручання людини, хоча часто необхідні деякі коригування з боку людини. Глибоке навчання використовує ймовірність для «навчання».Наприклад, у реченні «Запитайте Анастасію що зберігається в історії Індії.» найчастіше зустрічаються літери «а» та «і» — кожна літера по 6 разів.З цього модель глибокого навчання може зробити висновок, що ці символи є одними з найбільш імовірних для появи в українському тексті. Насправді модель глибокого навчання не може нічого зробити обмежуючись лише одним реченням. Але після аналізу мільярдів речень вона може навчитися достатньо, щоб передбачити, як правильно завершити неповне речення або навіть створити власні речення.

Щоб забезпечити цей тип глибокого навчання, LLM побудовано на нейронних мережах. Подібно до того, як людський мозок складається з нейронів, які з’єднуються та посилають один одному сигнали, штучні нейронні мережі (часто їх скорочено називають «нейронними мережами») складаються з мережевих вузлів.Вони складаються з ряду «шарів»: вхідного шару, вихідного шару та одного або кількох проміжних шарів.Рівні передають інформацію один одному, лише якщо їхній вихід перевищує певний поріг.

Специфічний тип нейронної мережі, що використовується для LLM, називається моделлю трансформером.Моделі трансформери здатні вивчати

контекст, що особливо важливо для людської мови, яка сильно залежить від контексту.Моделі трансформерів використовують математичну техніку під назвою «самоувага», щоб досліджувати тонкі способи зв’язку елементів ланцюга один з одним.З цієї причини вони розуміють контекст краще, ніж інші типи машинного навчання. Це дозволяє їм «зрозуміти», наприклад, як кінець речення з’єднується з початком і як речення в абзаці пов’язані одне з одним.Це дозволяє мовним моделям інтерпретувати людську мову, навіть якщо ця мова неоднозначна або погано визначена, скомпонована в комбінації, яких вони ніколи раніше не зустрічали. Певною мірою вони «розуміють» семантику, оскільки можуть пов’язувати слова та поняття на основі їхнього значення.

1.2 Чат-боти популярних компаній та їх API

ChatGPT- це чат-бот побудований на основі штучного інтелекту розроблений компанією OpenAI, здатний спілкуватися з людьми за допомогою природної мови. Цей інтелектуальний агент заснований на моделі глибокого навчання під назвою GPT (Generative Pre-training Transformer). ChatGPT вивчає велику кількість текстових даних і може відповідати на запитання, вирішувати завдання, давати поради тощо. Завдяки детальності та чіткості відповідей популярність чат-ботів зросла напрочуд швидко. Всього через 2 місяці після релізу кількість користувачів, які активно користувались сервісом, досягнула і перевищила відмітку в 100 мільйонів - такий короткий проміжок часу став рекордом серед кастомних програм.Після випуску оцінка OpenAI зросла до 29 мільярдів доларів. Після запуску ChatGPT на ринку почали з'являтись конкуруючі продукти, такі як Bard, LLaMA, Claude, Grok та інші.

OpenAI – компанія, яка спеціалізується на розробці продуктів штучного інтелекту. Після запуску чат-боту ChatGPT, вона надає доступ через API до низки інтелектуальних моделей:

- GPT-4: велика мультимодальна модель, яка може приймати текстові повідомлення або фото і генерувати відповідь у вигляді тексту на запит

користувача. Модель цієї версії може вирішувати різні задачі з більшою точністю ніж моделі попередніх версій;

- GPT 3.5 Turbo: модель, яка здатна розпізнавати та генерувати текст природною мовою та може допомогти в написанні програмного коду;
- DALL-E: це система штучного інтелекту, яка може створювати образи та мистецтво з запиту, який був описаний природною мовою. Модель також підтримує редагування уже створеного зображення або створення подібних зображень, наданого користувачем;
- TTS- це модель, яка створена для синтезу мовлення, тобто генерації тексту в природну мову.

Claude – це чат-бот, який був створений компанією Anthropic, дуже подібний до ChatGPT. Він приймає як і текстові запити, так і файли docs і зображення.

Anthropic- це стартап, який був створений для роботи в сфері штучного інтелекту і заснований колишніми робітниками компанії OpenAI. Компанія розробляє системи штучного інтелекту і інтелектуальні мовні моделі.

Anthropic має 3 основні типи моделей, а саме:

- Haiku- найшвидша модель, яка може виконувати легкі дії. Модель створена для корпоративних програм і допоможе з таким завданням як підтримка клієнтів. Крім швидкості і доступності, Haiku має перевагу у надійності та безпеці корпоративного рівня. Проводяться ретельні тестування, щоб використання моделі було максимально безпечним;
- Sonet- модель, яка за словами Anthropic, є найкращим поєднанням швидкості і продуктивності для завдань різного рівня складності;
- Opus- найрозумніша модель компанії Anthropic. Вона може проводити складний аналіз, тривалі завдання з декількома кроками, математики вищого рівня та написання програмного коду.

Gemini- це теж чат-бот який працює на базі мультимодальної мовної моделі, розроблений компанією Google DeepMind. Компанія наголосила, що їх інтелектуальний агент може перевершити всім відомий ChatGPT, який був

розроблений компанією OpenAI. Був підкреслений також успіх програми AlphaGo від DeepMind. Програма відома перемогою у го, стародавньої китайської стратегічної настільної грі, над професійним гравцем го Лі Седолом. Зазначається що DeepMind у Gemini об'єднали потужність AlphaGo з іншими технологіями великих мовних моделей від Google.

Google DeepMind- це британська компанія, яка спеціалізується на розробці систем штучного інтелекту. Була заснована у 2010 році і мала назву DeepMind Technologies до 2014 року, після чого компанія Google придбала її. У DeepMind Technologies створили нейронну мережу, яка була здатна грати у комп'ютерні ігри як людина. Також компанія розробила Нейронну Машину Тюрінга, яка здатна отримувати доступ до зовнішньої пам'яті подібно до звичайної Машини Тюрінга. В кінці кінців це комп'ютер, який імітує короткочасну пам'ять людського мозку. Головною метою компанії DeepMind є «вирішення проблеми штучного інтелект» за допомогою поєднання «найкращих технік машинного навчання та системної нейробіології» для того, щоб розбробити потужні навчальні алгоритми для загального призначення.

Copilot- це ще один інструмент побудований на системах штучного інтелекту, який працює на GPT-4. Був розроблений корпорацією Microsoft разом з компанією OpenAI для операційних систем Windows 11 та Windows 10, мобільних операційних систем iOS і Android, браузера Microsoft Edje та сервісу Microsoft 365. Головним завданням Copilot є підвищення продуктивності, а саме автоматизація повсякденних задач.

LLaMA- велика мовна модель, яка була випущена компанією Meta AI у лютому 2023 року. Модель навчена на різних джерелах даних і підтримує 20 мов, включаючи латиницю та кирилицю. Особливістю LLaMA є те, що вона може працювати в автономному режимі на різних типах пристроїв, включаючи ноутбуки та смартфони, завдяки програмним інструментам, які дозволяють працювати на графічних процесорах споживаного класу. LLaMA має на меті демократизувати доступ до досліджень штучного інтелекту, оскільки потребує меншої обчислювальної потужності та ресурсів і не потрібує доступу до

Інтернету. Модель LLaMA з 13 мільярдами параметрів перевершує набагато більшу GPT-3 (175 мільярдів параметрів) у більшості тестів NLP. У той час, коли великі мовні моделі в більшості випадків доступні лише через API, Meta випустили у відкритий доступ ваги LlaMA за некомерційною ліцензією.

1.3 Висновки до розділу 1

Беручи до уваги вищесказане, можна зробити висновок що інтелектуальні мовні моделі - це передові програми штучного інтелекту, які можуть аналізувати великі обсяги даних і генерувати текст. Використовуючи глибоке навчання та нейронні мережі, особливо моделі Transformer, вони можуть ефективно розуміти контекст і відповідати на широкий спектр запитів користувачів. Такі моделі широко використовуються в роботі з клієнтами, дослідженнях ДНК, пошуку інформації в Інтернеті, чат-ботах тощо. Однак важливо враховувати потенційні ризики, пов’язані з надійністю даних і безпекою конфіденційної інформації.

Чат-боти, розроблені такими провідними компаніями, як OpenAI, Anthropic, Google DeepMind і Microsoft, виявилися дуже ефективними та популярними завдяки своїй здатності генерувати точні та детальні відповіді. ChatGPT від OpenAI надзвичайно популярний завдяки своїй здатності обробляти природну мову та надавати корисні відповіді. Інші компанії, такі як Anthropic від Claude і Google DeepMind від Gemini, також пропонують потужні рішення, які можна інтегрувати за допомогою API з різноманітними платформами та додатками і забезпечують інтелектуальну підтримку та автоматизацію завдань. Ці технології відкривають нові можливості для використання в різних галузях, де необхідно враховувати безпеку та етичні аспекти.

2 ЗАСОБИ РЕАЛІЗАЦІЇ ВЕБ-ЗАСТОСУНКУ ДЛЯ РОЗПІЗНАВАННЯ ПРИРОДНОЇ МОВИ

2.1 Засоби розпізнавання природньої мови в браузерах

Webkit Speech API- це експериментальна технологія, яка має на меті дозволити веб-розробникам надавати у браузері функції розпізнавання мовлення та для перетворення тексту в природню мову, які зазвичай недоступні при використанні стандартного прогрмного забезпечення для розпізнавання мовлення чи зчитування з екрану. Сам API не залежить від основної реалізації розпізнавання та синтезу мовлення і може бути реалізований як на сервері так і на стороні клієнта.API розроблено як для короткого(разового) голосового введення, так і для безперервного введення інформації. Web Speech API складається з двох основних частин- SpeechRecognition і SpeechSynthesis.

SpeechRecognition- частина API, яка дозволяє веб-додаткам розпізнавати мову користувачів з аудіо-сигналу, який надходить через мікрофон користувачів. Вона має різні методи для управління процесом розпізнавання, а саме:

- start() для запуску розпізнавання. Коли викликається метод start(), це означає що додаток готовий почати розпізнавання мовлення. Якщо звуковий сигнал передається в реальному часі через мікрофон користувача, виклик методу start() починає процес прослуховування та спроб розпізнавання голосу користувача;
- stop() для припинення процесу розпізнавання. Метод stop() вказує технології розпізнавання припинити прослуховування нового аудіо-потоку та спробувати розпізнати результат отриманого аудіо. Це може бути використано, наприклад, у веб-додатках, коли користувач керує моментом завершення свого висловлювання подібно до роботи рації. Коли користувач відпускає клавішу, яка відповідає за голосове введення, викликається метод stop() щоб припинити прослуховування, після чого технологія SpeechRecognition намагається розпізнати природню мову користувача на основі зібраного аудіо;

- abort() для припинення процесу прослуховування та розпізнавання без повернення будь-якої інформації крім тієї, що процес був завершений. Цей метод використовується в основному тоді, коли користувач допустив помилку при введенні інформації і хоче або просто зупинити прослуховування або почати процес заново.

SpeechSynthesis- ця частина API призначена для того, щоб надавати можливість веб-застосункам створювати синтез мовлення, тобто перетворювати текст в мову. Вона надає різні методи для створення та керування деякими аспектами процесу синтезу мовлення, а саме вибір голосу, швидкість та висота голосу.

SpeechSynthesis також має свої методи для керування синтезом мовлення. А саме такі:

- speak(utterance) для початку озвучування тексту, тобто говоріння. Коли цей метод викликається, текст, який повинен озвучуватись, додається в кінець черги, після чого починається процес озвучування тексту;
- cancel() видаляє з черги весь текст, який мав би бути озвученим. Якщо текст озвучується і метод був викликаний, вимова негайно зупиняється.
- pause() використовується для призупинення процесу говоріння. Коли процес озвучування тексту був запущений, метод pause() зупиняє озвучування тексту посеред процесу. Якщо його викликати знову нічого не станеться;
- resume() продовжує озвучування тексту, який був зупинений методом pause(). Коли користувач призупинив вимову тексту, метод resume() продовжить озвучення тексту рівно з того місця, де був зупинений процес озвучування методом pause();
- getVoices() поветає список доступних голосів для озвучування. Доступні голоси залежать від браузера користувача.

Переваги Webkit Speech API:

- підтримка мов- API підтримує розпізнавання багатьох мов;
- підтримка- Webkit Speech API підтримується у багатьох браузерах, які мають підтримку Webkit;

- конфіденційність- щоб API почав записувати голос користувача, спочатку користувач повинен дати дозвіл в браузері на використання сторінкою мікрофону для запису голосу;
- можливості розширення: API можна розширити за допомогою сторонніх бібліотек або сервісів, які надають додаткову функціональність, такі як розпізнавання різних спеціальних команд або підтримка додаткових мов;
- розвиток API: Web Speech API постійно розвивається і оновлюється разом з розвитком технологій розпізнавання мовлення та синтезу мови. Тому варто слідкувати за оновленнями та новими можливостями, які можуть з'явитися у майбутньому.

2.2 Механізми передачі розпізнаного тексту з клієнтської сторони на сервер

XMLHttpRequest- це вбудований інтерфейс у JavaScript, який надає клієнту функціональність для обміну даними між клієнтом і сервером. Цей API надає простий спосіб отримання даних по посиланню без необхідності перезавантажувати сторінку. Це дозволяє перезавантажувати тільки частину веб-сторінки не перериваючи користувача. XMLHttpRequest був спочатку розроблений компанією Microsoft, а потім вже був запозичений компаніями Mozile, Apple та Google. Не дивлячись на назву API, за допомогою нього можна надсилати не тільки дані типу XML, а й інших.

jQuery.ajax- це технологія, яка дозволяє взаємодіяти з сервером без необхідності перезавантажувати сторінку. За допомогою Ajax дані можна надсилати та отримувати асинхронно не чекаючи завершення запитів, за допомогою чого можна створювати динамічні веб-додатки.

Основними перевагами Ajax є:

- динамічне оновлення сторінки: за допомогою нього можна оновлювати частину сторінки без необхідності перезавантажувати всю сторінку. Це дозволяє швидше та зручніше взаємодіяти з користувачами;

- асинхронність: оскільки запити Ajax надсилаються асинхронно, користувач може продовжувати взаємодіяти зі сторінкою, поки запит виконується у фоновому режимі. Це покращує швидкість і реакцію веб-додатків;
- мінімальний перерозподіл ресурсів: запити Ajax зазвичай передають лише необхідні дані між клієнтом і сервером, що мінімізує обсяг передачі даних і зменшує навантаження на сервер.

Ajax використовується в багатьох веб-додатках, включаючи соціальні мережі, інтернет-магазини, пошукові системи тощо.Це важлива технологія для створення сучасних інтерактивних веб-додатків.

Fetch API- це інтерфейс для передачі та отримання ресурсів з серверу. За допомогою нього можна зручно взаємодіяти з серверами та отримувати файли без необхідності перезавантажувати сторінку. Fetch API є сучасним і потужним інструментом і має ряд переваг перед старими матодами, такими як XMLHttpRequest. API має один метод, в якому вказується шлях, куди потрібно надіслати дані, тип даних які хочемо надіслати на сервер, та самі дані- наприклад текст.

Переваги Fetch API перед jQuery.ajax:

- вбудована підтримка Promises: fetch API повертає promises, що робить обробку асинхронних запитів більш сучасною та зручною. Promises- це інформація про стан асинхронних запитів, є 3 стани- виконання (pending), успішно виконана (fulfilled) або відхилена (rejected). Це дозволяє використовувати методи then() і catch() для обробки відповідей або помилок, спрощуючи обробку асинхронних операцій;
- досить простий і зрозумілий синтаксис, що полегшує створення та виконання HTTP-запитів;
- підтримка потоків: Fetch API дозволяє обробляти великі обсяги даних за допомогою потоків;
- підтримка браузерів: Fetch API є частиному мови JavaScript та підтримується у багатьох браузерах без використання додаткових бібліотек, що значно полегшить процес розробки і підтримання коду веб-додатків.

Загалом Fetch API — це потужний і сучасний інструмент для створення HTTP-запитів у JavaScript, який забезпечує зручний синтаксис і розширену функціональність, не покладаючись на сторонні бібліотеки.

2.3 Мова програмування JavaScript

JavaScript – це динамічна об’єктно-орієнтована мова програмування, яка використовується для створення веб-додатків, а саме для їх візуальної частини. За допомогою JavaScript можна керувати браузером, змінювати структуру і зовнішній вигляд веб-сторінок та асинхронно обмінюватись даними з сервером. JavaScript відносять до прототипної(підмножина об’єктно-орієнтованої парадигми) скриптової мови програмування з типізацією, яка є динамічною. Окрім прототипування мова програмування може підтримувати інші парадигми, такі як імперативна та може частково підтримувати функціональну. Також JavaScript має відповідні архітектурні властивості, а саме динамічну і слабку типізацію, є автоматичне керування пам’яттю, що є важливою функцією, успадкування прототипів і функції як об’єкти класу.

Мова програмування JavaScript використовується для таких цілей:

- написання скриптів для створення інтерактивних веб-сторінок;
- створення односторінковий і прогресивних веб-додатків за допомогою бібліотеки React та фреймворків AngularJS та Vue.js;
- серверне програмування за допомогою платформи Node.js;
- створення мобільних додатків за допомогою ферймворка React Native та платформи Cordova.

Незважаючи схожість назв Java та JavaScript, ці дві мови програмування є абсолютно різними, хоча й мають схожість в стандартних бібліотеках та умовами іменування, мають схожий синтаксис, який був успадкований від мови «C», але з різною семантикою.

Поява мови JavaScript є досить цікавою.У 1995 році компанія Netscape вирішила інтегрувати мову програмування Scheme або схожу мову програмування в браузер Netscape.Для цього вони вирішили залучити Брендана

Айха, американського розробника з досвідом системного програмування.Щоб прискорити процес розробки, Netscape також почав співпрацювати з Sun Microsystems.

З часом концепція мови, яку розробляли, розширилася до можливості її використання безпосередньо в HTML-коді сторінок. Мета полягала в тому, щоб створити мову для зв’язування різних частин веб-сайту, таких як зображення, аплети Java та об’єктна модель документа (DOM). Це повинно бути корисно веб-дизайнерам і людям, які не мають глибоких знань програмування. Спочатку нова мова називалася Mocha, але в перших двох бета-версіях браузера Netscape 2.0 назва була змінена на LiveScript.У зв'язку з популярністю бренду Java, LiveScript пізніше був перейменований в JavaScript, і під цією назвою вже була випущена третя бета-версія браузера Netscape 2.0. Для цього компанія отримала відповідну ліцензію від Sun Microsystems, власника бренду Java.

У 1992 році Nombas розробив Cmm (C-мінус-мінус), мову програмування сценаріїв, яку можна було вбудовувати у веб-сторінки. Пізніше це було перейменовано на ScriptEase. Існує помилкова думка, що JavaScript був створений під впливом CMM. Насправді Брендан Айх нічого не знав про Cmm до написання LiveScript. Після цього Nombas припинив розробку Cmm, почав використовувати JavaScript і приєднався до групи стандартів JavaScript. Таким чином дві компанії, а саме Netscape та Sun Microsystems, працюючи разом створили мову програмування яку можна було використовувати в HTML, що значно спростило взаємодію з веб-сторінками і зробило їх більш інтерактивними та динамічними.

У листопаді 1996 року Netscape оголосила, що подала JavaScript до ECMA на розгляд як галузевий стандарт. Це призвело до створення стандартизованої мови під назвою ECMAScript. У червні 1997 року ECMA випустила першу редакцію специфікації ECMAScript під номером ECMA-262. Наступного року, у червні 1998 року, було опубліковано друге видання з деякими змінами, внесеними для узгодження специфікації зі стандартом ISO/IEC 16262. Третє видання вийшло в грудні 1999 року. Версія 4 стандарту ECMAScript так і не була

завершена, а четверта версія так і не була опублікована. Проте 5-е видання вийшло в грудні 2009 року. Шоста версія вийшла в червні 2015 року. Потім комітет ECMAScript вирішив перейти до щорічних оновлень, і нова версія отримала назву ES2015. Він містить багато інноваційних функцій, таких як об'єкти Promise для зручного асинхронного виконання коду, структурування призначень, функції стрілок, функції генератора, рядки шаблонів і оператори оголошення змінних let і const. Версія ES2016 була випущена в червні 2016 року. Ця інноваційна функція включає в себе оператор потужності ** і метод Array.prototype.includes, щоб перевірити, чи певний елемент включено в масив. Версія ES2017, випущена в червні 2017 року, додала підтримку асинхронних функцій, можливість використовувати «висячі» коми в параметрах функції, об'єкти Atomics і кілька нових методів для роботи з рядками. ES2018 було випущено в червні 2018 року.

Це включає в себе асинхронну ітерацію, оператори поширення об'єктів і масивів, нові функції регулярного виразу та метод Promise.prototype.finally для обробки виконання Promise. Версія ES2019, випущена в червні 2019 року, представляє нові символьні типи даних, нові методи роботи з рядками та масивами, а також можливість перетворювати об'єкти в масиви та навпаки за допомогою Entries. Наразі це версія ES2020, випущена в червні 2020 року в якій було додано новий тип даних BigInt, оператор під назвою ?? для перевірки null і undefined, можливість використання необов'язкових значень в об'єктах, динамічний імпорт, об'єкт globalThis, метод String.prototype.matchAll для пошуку у рядках за допомогою регулярних виразів та метод Promise.allSettled для обробки виконання всіх Promise.

Зараз JavaScript є однією з найпопулярніших мов програмування в Інтернеті. Оскільки програма була спрямована на програмістів-любителів, більшість професійних програмістів скептично ставилися до неї на початку її створення. Однак із появою технології AJAX все змінилося, і JavaScript став центром уваги професійної спільноти. Подальші зміни до мови ES6+ додали багато важливих функцій, необхідних для ефективного програмування. Ці зміни

допомогли розробити та покращити багато способів використання JavaScript, наприклад тестування та налагодження. Було створено численні бібліотеки та фреймворки, що значно розширило використання JavaScript за межами браузера. У рейтингу найпопулярніших мов програмування серед IT-спеціалістів IEEE Spectrum 2023 для 59 мов JavaScript займає першу п’ятірку, поступаючись Python, Java, C++ і C.

2.3 Веб-технології HTML CSS

HTML (HyperText Markup Language) — це стандартизована мова розмітки документів, яка використовується для відображення веб-сторінок у браузерах. Браузер отримує HTML-документ із сервера через протоколи HTTP або HTTPS або відкриває його з локального жорсткого диска, інтерпретує код і перетворює його на візуальний інтерфейс, який користувач бачить на екрані.

Елементи HTML є будівельними блоками веб-сторінок. HTML дозволяє вставляти зображення та інші об’єкти (наприклад, інтерактивні форми) у веб-сторінки. HTML дозволяє створювати структуровані документи, використовуючи різні елементи для позначення заголовків, абзаців, списків, посилань, цитат в лапках та інших компонентів тексту. Елементи HTML розділені тегами, вкладеними в кутові дужки. Деякі теги, наприклад <img /> або <input />,, додають вміст безпосередньо на сторінку, тоді як інші, наприклад <p> оточують текст і надають семантичну інформацію. Теги також можуть містити інші теги як вкладені елементи. Браузери самі не відображають теги, але використовують їх для розуміння структури та вмісту сторінки.

HTML також дозволяє вбудовувати програми, написані за допомогою скриптових мов, такими як JavaScript, які впливають на поведінку і вміст веб-сторінок. CSS (каскадні таблиці стилів) використовується для визначення зовнішнього вигляду та компонування вмісту. З 1997 року Всесвітній веб-консорціум (W3C), який підтримує стандарти для HTML і CSS, сприяє використанню CSS замість використання елементів презентації HTML.

HTML застосовується щоб:

- створювати структуровані документи шляхом позначення різних елементів тексту, таких як заголовки, абзаци, списки, таблиці, цитати та інше;
- отримувати інформацію з Інтернету через гіперпосилання;
- створювати інтерактивні форми;
- вбудовувти в текст зображення, звуки, відео та інші об'єкти.

У 1980 році фізик Тім Бернерс-Лі, який тоді працював у CERN, запропонував систему INQUIRE і розробив прототип для полегшення обміну документами між дослідниками CERN.

У 1989 році Бернерс-Лі запропонував розробку системи гіпертекстових документів на основі Інтернету. До кінця 1990 року він розробив HTML і створив браузер та серверне програмне забезпечення для цієї системи. Того ж року Бернерс-Лі та Роберт Кейлот, інженер інформаційних систем CERN, подали заявку на фінансування проекту, але CERN офіційно не прийняв.

Наприкінці 1991 року Тім Бернерс-Лі опублікував перший публічний опис мови розмітки HTML, відомого як документ тегів HTML. Цей документ описує 18 елементів оригінальної, відносно простої схеми розмітки HTML, більшість з яких тісно пов’язані з внутрішнім форматом документа SGML CERN. Одинадцять із цих елементів досі використовуються в HTML4.

Бернерс-Лі вважав HTML похідною від SGML. У середині 1993 року Internet Task Force (IETF) офіційно визначила його та опублікувала першу специфікацію HTML під назвою «Мова розмітки гіпертексту (HTML)» Тіма Бернерса-Лі та Дена Коннолі. Ця специфікація містила визначення типів документів, які чітко описували граматику HTML.

Хоча проект було закрито через шість місяців, використання тегу add-image у браузері NCSA Mosaic було помітним і відображало підхід IETF до створення стандартів на основі успішних прототипів. У 1993 році HTML+ (Hypertext Markup Format), ще один Інтернет-проект Дейва Рагетта, запропонував стандартизувати нові функції, такі як таблиці та інтерактивні форми.

На початку 1994 року, коли проекти HTML і HTML+ були завершені, IETF створив робочу групу HTML. У 1995 році ця група завершила документ «HTML 2.0», який був опублікований як RFC 1866. Це була перша специфікація, яка слугувала базовим стандартом для подальшого вдосконалення HTML. У версії 2.0 чітко прописані відмінності від попередніх проектів.

Хронологія версій HTML:

- HTML (3 листопада 1992 р.): була першою версією HTML, яка зосередилася на вмісті простого тексту;
- HTML (30 квітня 1993 р.): додано текстові атрибути для виділення курсивом і жирним текстом і зображеннями;
- HTML+ (листопад 1993): запропоновані доповнення, які були включені до пізніших версій HTML, але не визначені явно як HTML+;
- HTML 2.0 (листопад 1995): версія , визначена RFC 1866, яка підтримує форми. Зараз цей стандарт є «історичним», а попередні версії вважаються застарілими;
- HTML 3.0: ця версія вже була застарілою після випуску Netscape Navigator версії 3, тому вона ніколи не прижилася;
- HTML 3.2 (14 січня 1997): додає нові функції, такі як таблиці, обтікання текстом навколо зображень та інтеграція аплетів;
- HTML 4.0 (18 грудня 1997): додано таблиці стилів, сценарії та рамки. Їх розділили на Strict (сувора відповідність стандарту), Frameset (з підтримкою фреймів) і Transitional. Переглянуту версію цього стандарту було опубліковано 24 квітня 1998 року;
- HTML 4.01 (24 грудня 1999): замінює HTML 4.0 і містить багато незначних виправлень;
- HTML 5 (робоча чернетка, 5 квітня 2008 р.): HTML 5 містить нові словники на основі HTML 4.01 і XHTML 1.0. Специфікацію DOM, пов’язану з HTML, також було переглянуто та розширено;

- XHTML 1.0 (26 січня 2000): представлення стандарту HTML 4.01 за допомогою XML. Оновлена версія цього стандарту була опублікована 1 серпня 2002 року;
- XHTML 1.1 (31 травня 2001 р.): визначає сувору версію XHTML після модуляції, виключаючи можливості Frameset і Transitional, представлені в HTML 4;
- XHTML 2.0 (розробка припинена в 2010 році): версія, яка вже не базується на HTML 4.01, додано деякі нові теги і завершено розділення між представленням та вмістом.

Щоб покращити взаємодію з SGML, кожна похідна мова, включаючи HTML, повинна визначати власну кодову таблицю для кожного документа. Ця таблиця містить репертуар (перелік різних символів) і позицію символу (числове посилання на символ у репертуарі). Кожен HTML-документ складається з набору символів із цього репертуару.

HTML використовує найповнішу кодову таблицю універсального набору символів (UCS). Однак цього недостатньо для належного відображення HTML-документів у браузері. Браузери повинні знати конкретну кодову таблицю для кожного документа, а автори завжди повинні вказувати кодову таблицю в мета-елементах за допомогою параметра charset. За замовчуванням використовується кодова таблиця ISO-8859-1, відома як Latin-1.

CSS (Cascading Style Sheets) — це спеціальна мова стилів, яка описує зовнішній вигляд веб-сайту. Веб-сторінки, написані мовами розмітки, такими як HTML і XHTML, використовують CSS для візуального дизайну. CSS є однією з ключових технологій Всесвітньої павутини разом із HTML і JavaScript. CSS найчастіше використовується для форматування документів HTML і XHTML, але його також можна використовувати з іншими документами XML.

Специфікації CSS розробляються та підтримуються Консорціумом Всесвітньої павутини (W3C). CSS має різні рівні та профілі, які розширюють його функціональність. Кожен новий рівень CSS, як-от CSS1, CSS2 і CSS3, базується на попередніх рівнях CSS, додаючи нову функціональність або

покращуючи існуючу. Профілі CSS створюються для певних типів пристроїв або інтерфейсів, наприклад принтерів або мобільних пристроїв. Каскадні таблиці стилів замінили табличний макет веб-сторінок і запропонували головну перевагу відокремлення вмісту сторінки від візуального представлення. Це значно спрощує процес розробки та обслуговування веб-сайту, забезпечуючи чітке розмежування між даними та їх дизайном.

Той самий документ HTML або XML може виглядати по-різному залежно від застосованих стилів CSS. Стилі відображення сторінки можуть надходити з різних джерел:

1. Стилі автора (інформацію надає автор сторінки):

- зовнішні таблиці стилів (таблиці стилів): окремі файли з розширенням .css, включені у файл документа;
- внутрішні таблиці стилів : інтегровані безпосередньо в документ або його частини;
- стилі окремих елементів: застосовуються безпосередньо до окремих елементів HTML за допомогою атрибута style.

2. Стилі користувача:

- локальні файли CSS: визначені користувачем файли для використання на всіх веб-сторінках. Вказується в налаштуваннях браузера (наприклад, Opera).

3. Стиль переглядача (браузер):

- стиль браузера за замовчуванням: стиль елемента за замовчуванням, визначений браузером. Використовується, коли інформація про стиль елемента відсутня або неповна.

Стандарти CSS визначають порядок і область застосування стилю, тобто, у якому порядку та до яких елементів застосовуються стилі. Ця послідовність називається принципом каскаду. Лише інформація про стиль, яка змінилася або не визначена в більш загальному стилі, потім застосовується до елемента. Це дозволяє уникнути зайвого коду та дає змогу створювати гнучкі та ефективні стилі веб-сайту.

Таблиці стилів мають наступні переваги:

- ви можете включити інформацію про стилі для всього або частини вашого сайту у файл CSS, дозволяючи швидко змінювати дизайн і зовнішній вигляд ваших сторінок;
- більше інформації про стилі для різних типів користувачів: більші розміри шрифту для користувачів із вадами зору, стилі для друку сторінок, стилі для мобільних пристроїв тощо;
- сторінки менші та більш структуровані, оскільки інформація про стиль відокремлена від тексту, існують спеціальні правила застосування, і сторінка створюється з урахуванням цих правил;
- прискорює завантаження сторінки, зменшуючи обсяг переданої інформації та навантаження на сервери та канали передачі. Це досягається за рахунок того, що сучасні браузери кешують (запам'ятовують) інформацію про стилі та роблять її доступною для всіх сторінок замість того, щоб завантажувати її для кожної окремої сторінки.

CSS має відносно простий синтаксис і використовує кілька англійських слів для позначення різних компонентів стилю. Стиль складається зі списку правил. Кожне правило містить один або кілька селекторів і блок оголошення. Блок визначення вкладено у фігурні дужки та складається зі списку властивостей та їхніх значень.

Властивості у блоці вказуються наступним чином:

- назва властивості: вказує, яка властивість елемента змінюється (колір, розмір шрифту тощо);
- двокрапка (:): розділяє назву властивості та її значення.
- значення: вказує на значення, призначене властивості (наприклад, 'red', '16px');
- крапка з комою (;): використовується для розділення окремих пар властивість-значення в блоці визначення.

Таблиці стилів існували в тій чи іншій формі з моменту появи SGML у 1970-х роках. Однак каскадні таблиці стилів (CSS) створені для полегшення додавання стилів до ваших веб-сторінок.

У міру розвитку HTML з’явилися різні механізми додавання стилів до елементів сторінки. Хоча ці механізми дали веб-дизайнерам більше можливостей для розробки веб-сайтів, вони також ускладнили написання та зміну HTML-коду. Крім того, різні браузери можуть відображати сторінки по-різному, що ускладнює підтримку узгодженого стилю та надаючи користувачам більше контролю над тим, як виглядає їхній вміст.

Дев'ять різних версій таблиць стилів були запропоновані для розгляду W3C. Після обговорення у спеціальному списку розсилки було обрано дві основи сучасного CSS: Cascading HTML Style Sheets (CHSS) і Stream-based Style Sheet Proposal (SSP). У жовтні 1994 року поточний технічний директор Opera Software Хокон Вільям Лі запропонував CHSS, який схожий на сучасний CSS. Берт Бос працював над браузером Argo, який використовує власний варіант таблиці стилів SSP. Лі та Бос почали працювати разом над створенням стандарту CSS ("H" було вилучено з назви, оскільки таблиці стилів можна застосовувати не лише до HTML).

На відміну від попередніх таблиць стилів, таких як DSSSL і FOSI, CSS дозволяв застосовувати декілька таблиць стилів до одного документа. Це створило каскадну систему (наприклад, браузер Opera), де як дизайнери веб-сайтів, так і користувачі можуть контролювати використання стилів.

Дев'ять різних версій таблиць стилів були запропоновані для розгляду W3C. Після обговорення у спеціальному списку розсилки було обрано дві основи сучасного CSS: Cascading HTML Style Sheets (CHSS) і Stream-based Style Sheet Proposal (SSP). У жовтні 1994 року поточний технічний директор Opera Software Хокон Вільям Лі запропонував CHSS, який схожий на сучасний CSS. Берт Бос працював над браузером Argo, який використовує власний варіант таблиці стилів SSP. Лі та Бос почали працювати разом над створенням стандарту CSS ("H" було

вилучено з назви, оскільки таблиці стилів можна застосовувати не лише до HTML).

На відміну від попередніх таблиць стилів, таких як DSSSL і FOSI, CSS дозволяв застосовувати декілька таблиць стилів до одного документа. Це створило каскадну систему , що дозволяло як дизайнерам веб-сайтів, так і користувачам контролювати використання стилів(наприклад, у браузері Opera). Пропозиції щодо стандарту CSS обговорювалися на конференціях 1994 та 1995 років. У 1994 році Консорціум Всесвітньої павутини (W3C) був заснований спеціально для розробки CSS. Робочу групу очолював Стівен Пембертон, а старшими технічними експертами були Хокон Вільям Лі та Барт Бос. Рекомендації CSS рівня 1 були випущені в грудні 1996 року. CSS Level 2 (Cascading Style Sheets Level 2 Revision 2 або CSS 2.2) був випущений у квітні 2016 року та все ще перебуває в розробці.

2.4 Мова програмування Python

Python (загально вимовляється як «Python»; назва походить від британського шоу Monty Python) — це інтерпретована об’єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією. Створено Гвідо Ван Россумом у 1990 році. Високорівневі структури даних, динамічна семантика та динамічне зв’язування Python роблять його придатним для швидкої розробки програм та інтеграції існуючих компонентів. Мова підтримує модулі та пакети, сприяючи модульності та повторному використанню коду. Інтерпретатор Python і стандартна бібліотека доступні у скомпільованій формі на всіх основних платформах. Python підтримує кілька парадигм програмування: об’єктно-орієнтоване, процедурне, аспектно-орієнтоване та функціональне програмування.

Основними перевагами мови Python є:

- чистий синтаксис (блоки коду розділені відступами);
- можливість переносити програми (типова властивість для більшості інтерпретованих мов);

- великий набір корисних модулів у стандартному дистрибутиві, включаючи модулі для створення графічних інтерфейсів;
- можливість працювати в діалоговому режимі (корисно для експериментів і вирішення простих задач);
- просте, але потужне середовище розробки IDLE, написане мовою Python.
- зручність у вирішенні математичних задач (обробка комплексних чисел, підтримка операцій над цілими числами довільного розміру, використання як потужного калькулятора в діалоговому режимі);
- відкритий код (можливість редагування іншими користувачами).

Python має ефективні високорівневі структури даних і простий, але потужний підхід до об'єктно-орієнтованого програмування. Елегантний синтаксис, динамічна обробка типів і інтерпретованість мови, роблять Python ідеальним для створення сценаріїв і швидкої розробки додатків у багатьох галузях і на більшості платформ.

Інтерпретатор мови і обширна стандартна бібліотека (як вихідні, так і бінарні дистрибутиви для всіх основних операційних систем) доступні на веб-сайті Python і можуть вільно поширюватися. Цей сайт також містить дистрибутив, численні модулі, програми, утиліти та додаткову документацію.

Інтерпретатор Python можна розширити функціями та типами даних, написаними на C або C++. Python також є хорошою мовою для розширення прикладних програм, які потребують додаткового налаштування.

Розробка мови Python почалася наприкінці 1980-х років Гвідо ван Россумом, співробітником інституту CWI в Нідерландах. Він створив розширювану мову скриптів, необхідну для розподіленої операційної системи Amoeba. Гвідо почав розробляти Python у вільний час, використовуючи знання, отримані під час попередньої роботи над мовою ABC. У лютому 1991 року він опублікував вихідний код у групі новин alt.sources. Це дозволило Python поширитися в Інтернеті та залучити інших програмістів. Відтоді Python став повністю об'єктно-орієнтованою мовою. Він використовує багато можливостей

мов C, C++, Modula-3 та Icon, а також деякі функції функціонального програмування Lisp.

Назва цієї мови не має нічого спільного з рептиліями. Гвідо назвав його на честь популярного британського комедійного шоу «Повітряний цирк Монті Пайтона». Однак ця назва часто асоціюється зі змією, яка з'являється на піктограмах файлів KDE, Windows і логотипі офіційного веб-сайту.

Велика та дружня спільнота користувачів і дизайнерська інтуїція Гвідо є одними з ключових факторів успіху Python. Розробка мови відбувається через процес створення, обговорення, вибору та реалізації документа PEP (Python Enhancement Proposal), який є пропозицією щодо розробки Python. 3 грудня 2008 року після тривалого тестування була випущена перша версія Python 3000 (або Python 3.0, також відома як Py3k). Python 3000 виправляє багато архітектурних помилок і забезпечує максимальну (але не повну) сумісність зі старими версіями. Наразі підтримується версія Python 3.

За даними IEEE Spectrum, Python став найпопулярнішою мовою програмування серед IT-фахівців у 2023 році. У рейтинг 2023 року увійшли 59 мов програмування.

Python підтримує динамічну типізацію. Тобто тип змінної визначається лише під час виконання. Основні типи Python підтримують цілі та комплексні числа довільної довжини. Мова також має потужні бібліотеки для маніпулювання рядками, включаючи обробку тексту, закодованого Unicode. Python підтримує різноманітні колекції, зокрема: Приклади: кортежі, списки (масиви), словники (асоціативні масиви), а з версії 2.4 додано підтримку множин. Система класів Python підтримує множинне успадкування та метапрограмування. Усі типи даних, включаючи базові типи, входять в систему класів, тому є можливість також успадковувати їх, якщо потрібно.

Розробники мови Python дотримуються певної філософії програмування, відомої як «Python Zen». Текст цієї філософії можна отримати, виконавши команду import this в інтерпретаторі Python (лише один раз за сеанс). Автором цієї філософії є Тім Пітерс.

Текст філософії:

- красиве краще, ніж потворне;
- явне краще, ніж неявне;
- просте краще, ніж складне;
- краще складне, ніж плутанина;
- плоский краще, ніж вкладений;
- розріджений краще, ніж щільний;
- читабельність важлива;
- особливі випадки недостатньо особливі, щоб порушувати правила;
- практичність важливіша за досконалість;
- помилки ніколи не пропускаються;
- якщо їх приховування не прописано явно;
- коли ви стикаєтеся з невизначеністю, краще утримайтеся від спокуси здогадуватися;
- має бути один, і бажано лише один, очевидний спосіб зробити це;
- якщо ви не голландець, ви напевно і не зможете побачити його відразу;
- краще зараз, ніж ніколи;
- хоча ніколи, часто краще, ніж зараз;
- якщо реалізацію важко пояснити, то задумка погана;
- задумка може бути хорошою, якщо реалізацію легко пояснити;
- простори імен — це чудово, давайте створимо їх більше.

Програмне забезпечення Python (додатки або бібліотеки) організовано в модулі, які можна збирати в пакунки. Модулі можна розміщувати як у каталогах, так і в архівах ZIP. Модулі діляться на два типи: ті, що написані на чистому Python, і модулі розширення (модулі розширення), написані на інших мовах програмування. Наприклад, стандартна бібліотека має «чистий» модуль Pickle і його аналог на C: cPickle. Модулі створюються як окремі файли, а пакети – як окремі каталоги. Модулі підключаються до програми через оператори імпорту. Після імпортування модуль представляється іншим об’єктом, який забезпечує

доступ до простору імен модуля. Під час роботи програми ви можете перезавантажувати модулі за допомогою функції reload().

2.5 Фреймворк Django

Django — це високорівневий фреймворк Python з відкритим кодом для створення веб-додатків. Він був названий на честь джазового музиканта Джанго Рейнхардта, слідуючи музичним смакам одного з його розробників. Веб-сайт створений за допомогою Django складається з однієї або кількох модульних частин. Це одна з ключових архітектурних особливостей, яка відрізняє його від інших фреймворків, таких як Ruby on Rails.

Архітектура Django схожа на шаблон MVC (Model-View-Controller), але є деякі відмінності в термінології. У Django «контролер» у MVC називається «view», а «view» у MVC називається «шаблоном». Тому архітектура Django називається MTV (Model-Template-View).

Django було розроблено для керування новинними сайтами та мало великий вплив на їхню архітектуру. Містить інструменти для швидкої розробки корисних веб-сайтів. Наприклад, Django має вбудований модуль керування вмістом, тому розробникам не потрібно писати контролери чи сторінки для адміністративних частин своїх сайтів. Цей модуль можна інтегрувати в будь-який сайт Django і дозволяє керувати кількома сайтами на одному сервері. Модуль адміністрування дозволяє створювати, редагувати та видаляти об’єкти контенту, фіксувати всі дії, а також надає інтерфейс для керування користувачами та групами з призначеними дозволами.

Дистрибутив Django включає програми для коментарі, синдикація RSS і Atom, «статичні сторінки» (якими можна керувати, не створюючи контролерів або переглядів) і перенаправлення URL-адрес. Django спочатку було створено для керування сайтами новин LJWorld.com, lawrence.com і KUsports.com для The World Company (Лоуренс, Канзас, США). З моменту випуску програмного забезпечення з відкритим кодом він стало популярною платформою для багатьох систем у всьому світі.

Django включає такі можливості:

- ORM (Object-Relational Mapping), який забезпечує API для доступу до бази даних з підтримкою транзакцій;
- вбудований інтерфейс адміністратора, з перекладами на більшість мов;
- диспетчер URL на основі регулярних виразів;
- розширювана система шаблонів з тегами та наслідуванням;
- система кешування;
- інтернаціоналізація;
- Архітектура підключаємих застосунків, які можна встановлювати на будь-які Django-сайти;
- generic views — шаблони функцій контролерів;
- авторизація та аутентифікація, з підтримкою зовнішніх модулів аутентифікації: LDAP, OpenID та інші;
- система фільтрів (middleware) для створення додаткових обробників запитів, включаючи кешування, стиснення, нормалізацію URL і підтримку анонімних сесій;
- бібліотека для роботи з формами, яка дозволяє створювати форми на основі моделей бази даних;
- вбудована автоматична документація по тегам шаблонів та моделям даних, доступна через адміністративний інтерфейс.

Компоненти фреймворку Django між собою слабо пов'язані, через що можна легко замінювати будь-яку частину на іншу. Наприклад, замість вбудованої системи шаблонів можна використовувати Mako або Jinja.

2.6 Система керування базами даних SQLite

SQLite — це легка система керування реляційною базою даних, реалізована як бібліотека, яка підтримує багато стандартів SQL-92. Вихідний код для SQLite знаходиться у відкритому доступі і може використовуватися для будь-яких цілей без обмежень і безкоштовно. Фінансову підтримку розробникам

надаватиме консорціум, до якого входять такі компанії, як Adobe, Oracle, Mozilla, Nokia, Bentley і Bloomberg. З 2018 року Бібліотека Конгресу рекомендує SQLite для зберігання структурованих даних разом із форматами JSON і CSV. У 2005 році цей проект отримав нагороду Google-O'Reilly Open Source Awards.

SQLite відрізняється від традиційних систем керування реляційними базами даних тим, що не використовує модель клієнт/сервер. Натомість механізм SQLite є частиною програми, яка використовує його як вбудовану бібліотеку. Цей підхід дозволяє вам взаємодіяти з базою даних через виклики API, зменшуючи накладні витрати та час відповіді, а також спрощуючи структуру вашої програми. SQLite зберігає всю базу даних, включаючи визначення, таблиці, індекси та дані, в одному стандартному файлі на комп’ютері, де запущена програма. Простіше реалізувати блокування всього файлу бази даних на початку транзакції. Зокрема, функціональність ACID досягається за допомогою журналу транзакцій.

SQLite дозволяє кільком процесам або потокам легко читати дані з однієї бази даних одночасно. Запис до бази даних можливий, лише якщо на цей момент не обробляються інші запити. Якщо база даних зайнята, спроба запису або закінчиться з кодом помилки, або автоматично повториться протягом певного періоду часу. Пакет SQLite містить клієнтську частину у вигляді виконуваного файлу SQLite3, який демонструє функціональність бібліотеки. Цей клієнт працює з командного рядка і дозволяє виконувати звичайні операції з базою даних. Архітектура SQLite дозволяє використовувати його як у вбудованих системах, так і в потужних машинах з великими обсягами даних.

Сама бібліотека SQLite написана на мові програмування C. Існує також реалізація JavaScript цієї бібліотеки під назвою sql.js. Це дозволяє працювати з файлами бази даних безпосередньо у вашому браузері. Багато інших мов програмування розробили механізми для підключення до баз даних і керування ними за допомогою бібліотеки SQLite. До них відносяться такі мови, як C++, Java, Python, Perl, PHP, Ruby, Haskell, Scheme, Smalltalk Lua та інші. Інструменти

для роботи з Tcl включені в стандартний пакет SQLite. Повний список доступних інструментів можна знайти на сторінці проекту.

2.7 Система управління версіями Git

Git — розподілена система керування версіями файлів і спільної роботи. Цей проект був створений Лінусом Торвальдсом для управління розробкою ядра Linux, і зараз його очолює Джуніо Сі. Git вважається однією з найефективніших, надійних і потужних систем контролю версій, що забезпечує гнучкі засоби нелінійної розробки на основі розгалуження та злиття гілок. Криптографічні методи використовуються для забезпечення історичної цілісності та захисту від ретроактивних змін. Крім того, цифрові підписи розробників можна прив'язувати до тегів і комітів.

Приклади проектів, які використовують Git, включають ядро Linux, Android, LibreOffice, Cairo, GNU Core Utilities, Mesa 3D, Wine, багато проектів X.org, XMMS2, GStreamer, Debian, DragonFly BSD, Perl, Eclipse, приклади включають GNOME, KDE та Qt, Ruby on Rails, PostgreSQL, VideoLAN, PHP, One Laptop Per Child (OLPC), ABIS Koha, GNU LilyPond, ELinks і деякі дистрибутиви GNU/Linux.

Ця програма є безкоштовною та випущена під ліцензією GNU GPL версії 2. Система Git розроблена як набір програм, спеціально розроблених для використання в сценаріях, що полегшує створення спеціалізованих систем контролю версій та інтерфейсів користувача на основі Git. Наприклад, Cogito є інтерфейсом для сховищ Git, а StGit використовує Git для керування колекціями патчів. Git має багато інтерфейсів користувача. gitk і git-gui поширюються разом із самим Git.

Віддалений доступ до сховищ Git здійснюється через службу Git, SSH або HTTP-сервер. Служба TCP git-daemon включена в дистрибутив Git і, разом із SSH, є найпопулярнішим і надійнішим методом доступу. Хоча метод доступу HTTP має багато обмежень, він дуже поширений у контрольованих мережах, оскільки дозволяє використовувати існуючі конфігурації мережевого фільтра.

На відміну від Subversion та подібних систем, Git не зберігає інформацію у вигляді списку змін (патчів) до файлів. Натомість Git зберігає дані як серію зліпків. Щоразу, коли ви фіксуєте поточну версію свого проекту, Git зберігає знімок стану всіх файлів вашого проекту. Якщо файл не змінився, Git підтримує посилання на попередньо збережений файл. Отже, Git діє як свого роду файлова система з інструментами на її основі. Git зберігає такі дані, як розмір, час створення та час останньої модифікації для кожного відстежуваного файлу. Ця інформація зберігається у файлі індексу у папці .git. Уся база даних Git також зберігається в папці під назвою .git.

У Git файли можуть перебувати в одному з трьох станів:

- зафіксовано (файл уже збережено в локальній базі даних);
- змінено (файл змінено, але зміни ще не зафіксовано);
- підготовлено (файл змінено та позначено для фіксації).

Більшість операцій Git користувач може виконувати у своїй власній локальній файловій системі без підключення до Інтернету. Уся історія змін зберігатиметься локально та завантажуватиметься у віддалений репозиторій лише за потреби. На відміну від Subversion, де користувач може лише редагувати файли без Інтернету і не зберігати зміни в базі даних (оскільки вона не підключена до репозиторію), Git виконує кожну фіксацію спочатку локально, а потім у віддаленому репозиторії. Система керування версіями зберігає все в базі даних у хешах файлів. Хеші файлів обчислюються за допомогою функції SHA-1. Перед збереженням кожного файлу Git обчислює хеш файлу SHA-1, і отриманий хеш стає індексом файлу в Git. Хеші дозволяють Git легко відстежувати зміни у файлах.

Відгалуження або система гілок — це процес відгалуження від основної головної лінії розробки. Git дозволяє створювати кілька гілок і перемикатися між ними. Це особливо корисно для співпраці розробників, оскільки дозволяє працювати над різними функціями незалежно, не впливаючи на основну гілку. За замовчуванням Git створює головну гілку під назвою "master". Гілка в Git — це вказівник на один коміт. Кожного разу, коли відбувається нове комітування,

гілка автоматично перемикається на новий коміт. Гілка представлена як простий файл, що містить 40-символьну контрольну суму SHA-1 коміту. Створення нової гілки в Git відбувається дуже швидко, оскільки для запису у файл потрібно лише 41 байт (40 символів + символ нового рядка).

Git підтримує два способи інтеграції змін між гілками: merge, тобто злиття та rebase — перебазування. Основна відмінність між ними полягає в тому, як обробляється історія комітів.

Merge об’єднує зміни з двох гілок в одну гілку. Після злиття всі коміти з обох гілок зберігаються під час розробки, тому користувач може точно бачити, як зміни були інтегровані. Як правило, merge створює додаткову фіксацію зливання, яка об’єднує два набори змін. Це корисно для отримання повної історії еволюції гілки, але може ускладнити читання історії.

Rebase на відміну від злиття, змінює походження гілки, переміщуючи її до нової початкової точки. Rebase бере всі коміти з поточної гілки, зберігає їх як патчі, переміщує базову гілку до нової точки та застосовує ці патчі як нові коміти. Завдяки цьому створюється лінійна історія що полегшує її читання. Однак перебазування перезаписує вашу історію комітів, що може спричинити проблеми, якщо інші розробники вже працюють над гілкою, яку ви перебазуєте.

2.8 Середовище розробки PyCharm

PyCharm — це інтегроване середовище розробки (IDE) для мови програмування Python, розроблене чеською компанією JetBrains на основі IntelliJ IDEA. PyCharm надає розробникам потужні інструменти для розробки, аналізу, тестування та налагодження коду Python, значно підвищуючи продуктивність і ефективність процесу розробки.

Основні можливості PyCharm:

- аналіз коду: середовище розробки автоматично перевіряє код на наявність помилок, підсвічує їх та пропонує варіанти виправлення;

- графічний відлагоджувач коду: PyCharm має зручний графічний інтерфейс для відлагодження коду, надає можливість встановлювати точки призупинення- breakpoints, відстежує змінні та їх використання в реальному часі;
- інструменти для тестування: середовище має вбудовану підтримку юніт-тестів, бібліотеки для тестування такі як pytest та unittest, можливість інтерактивного запуску і перегляду результатів тестування;
- веброзробка з фреймворком Django: PyCharm має вбудовані спеціалізовані інструменти для роботи з Django, генерує код для моделей, форм, шаблонів та для URL-адрес, наявна інтеграція з адмін-панеллю Django і підтримка функцій ORM.

2.9 Висновки до розділу 2

Завдяки поєднанню Web Speech API, JavaScript, HTML5, CSS, Python, Django, SQLite3, Git і PyCharm, створення веб-застосунку з використанням інтелектуальної моделі з можливістю голосового введення інформації забезпечує відповідну функціональність, продуктивність, масштабованість і ефективність процесу розробки.

Web Speech API у поєднанні з JavaScript і HTML5 надає можливість розпізнавати голосові команди та перетворювати мовлення в текст безпосередньо в браузері. JavaScript відповідає за обробку подій, взаємодію з DOM і асинхронний обмін даними з сервером. HTML5 надає веб-сайту структуру та семантику. Для візуального представлення доречно використовувати CSS для візуального представлення вашої програми. CSS дозволяє відокремити дизайн від структури та вмісту, що робить його більш гнучким. Для серверної частини доцільно вибрати Python Python — це потужна та гнучка мова програмування зі зрозумілим синтаксисом та величезним набором бібліотек і фреймворків. Фреймворк Django, написаний на Python, забезпечує швидку розробку веб-додатків завдяки своїй архітектурі, яка включає ORM, панель адміністратора, маршрутизацію URL-адрес, систему шаблонів тощо. Для зберігання даних можна використовувати SQLite3, інтегровану

реляційну СУБД, яка спрощує розробку. Система контролю версій Git забезпечує ефективне відстеження змін коду, співпрацю та можливість повернутися до попередніх версій. Інтегроване середовище розробки PyCharm надає зручні інструменти для написання, налагодження та тестування коду Python, а також інтеграції з Git.

3 РЕАЛІЗАЦІЯ ВЕБ-ЗАСТОСУНКУ З ВИКОРИСТАННЯМ ІНТЕЛЕКТУАЛЬНОЇ МОДЕЛІ З МОЖЛИВІСТЮ ГОЛОСОВОГО ВВЕДЕННЯ ІНФОРМАЦІЇ

3.1 Діаграма діяльності

Діаграми діяльності - це інструмент в області моделювання систем і процесів. Вони використовуються для візуалізації послідовностей дій і потоків даних. Основою таких діаграм є граф діяльності, який представляє послідовність дій, які виконуються в певному порядку. Кожна дія може мати власні вхідні та вихідні сигнали, які визначають, які дані вона отримує та які дані повертає. Виконання дій може контролюватися різними логічними умовами, які контролюють порядок виконання. Діаграми діяльності можна використовувати для моделювання різноманітних процесів: від бізнес-процесів до програмних алгоритмів. Вони дозволяють легко виразити складні зв’язки між діями та наказами на виконання. На практиці діаграми діяльності використовуються для системного аналізу та проектування, документації процесу, розробки програмного забезпечення тощо. Це важливий інструмент для розуміння складних процесів і систем і керування ними.

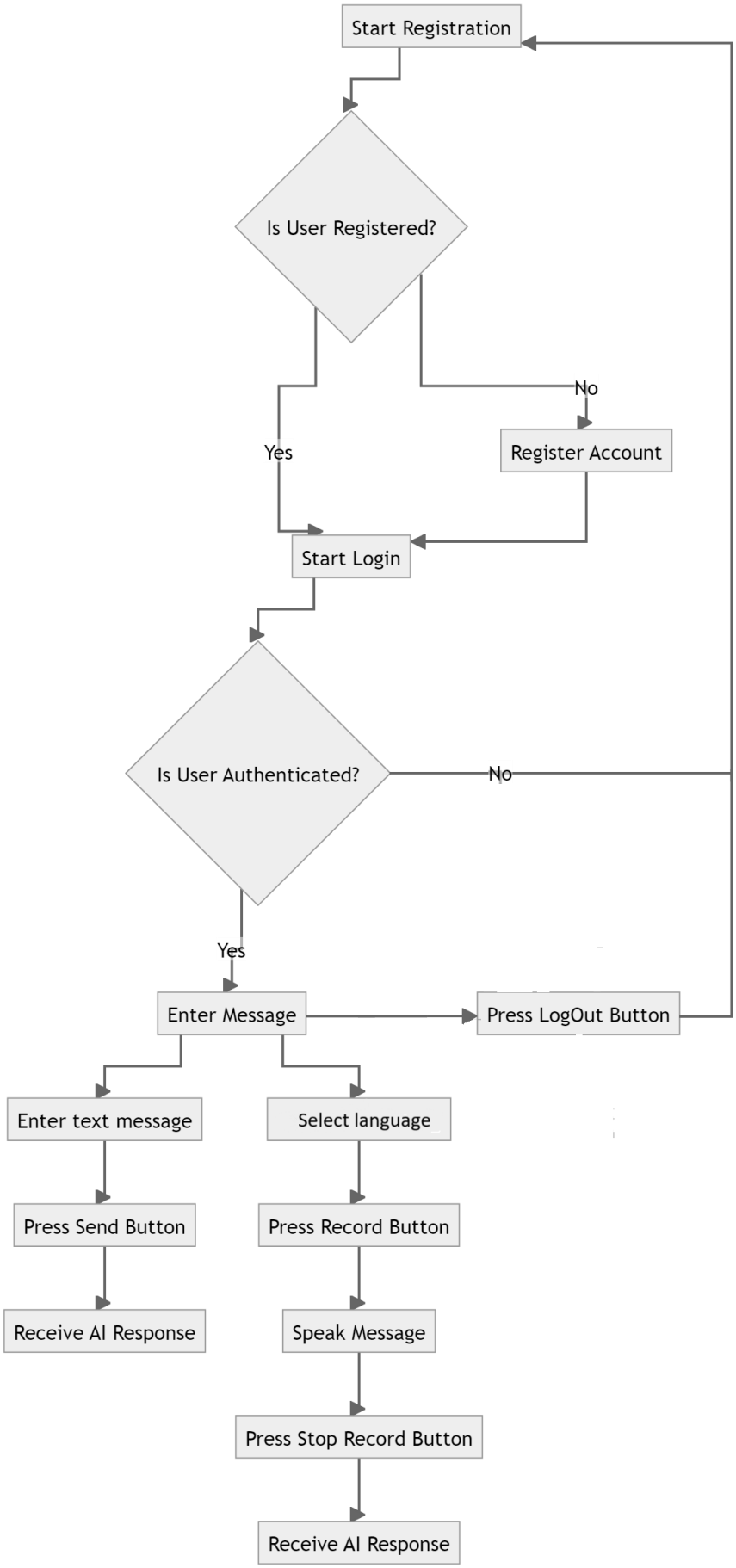

Рис. 3.1 Діаграма діяльності

3.2 Інтеграція інтелектуальної моделі

Для інтеграції інтелектуальної моделі в проект, який був написаний на мові програмування Python, потрібно встановити бібліотеку Anthropic, використивуючи команду pip install anthropic, оскільки буде використовуватись модель від компанії Anthropic.

Бібліотека Anthropic Python забезпечує зручний спосіб взаємодії з Anthropic REST API безпосередньо з програм, написаних на Python версії 3.7 або новішої. Ця бібліотека містить визначення типів для всіх параметрів запиту та полів відповіді, що забезпечує надійність і простоту використання API. Крім того, на основі бібліотеки httpx передбачено параметри як синхронного, так і асинхронного клієнта, що дозволяє вибрати найбільш зручний спосіб інтеграції в існуючі програми або системи.

Бібліотека дозволяє легко надсилати запити до Anthropic REST API, обробляти відповіді та інтегрувати отримані дані у програми Python. Визначення типів гарантує правильне форматування запитів і обробку відповідей, зменшуючи ймовірність помилок. Синхронні клієнти підходять для простих сценаріїв використання, коли запити API виконуються послідовно, тоді як асинхронні клієнти можуть виконувати запити паралельно, що може допомогти підвищити продуктивність для більш складних програм.

Після встановлення бібліотеки в середовище Python, потрібно ініціалізувати її, ініціалізувати API-ключ моделі та підключити його в проекті(лістинг 3.1).

Лістинг 3.1 Імпорт необхідної бібліотеки та підключення API ключа

```
import anthropic

API_key = "Ваш API ключ"

client = anthropic.Anthropic(

    api_key=API_key,

)
```

Після цих кроків вже є можливість ініціалізувати саму функції з інтелектуальної моделлю та налаштувати її, що і наведено у лістингу 3.2

Лістинг 3.2 Функція для обробки повідомлення та надання відповіді моделі

```
def handle_message(message_text):
    text = message_text
    message = client.messages.create(
        model="claude-3-opus-20240229",
        max_tokens=1000,
        temperature=0.0,
        system="You are assistant. Your name is VoiceGPT. You can understand the voice",
        messages=[
            {"role": "user", "content": text}
        ]
    )
    response_texts = [content.text for content in message.content if hasattr(content, 'text')]
    reply = '\n'.join(response_texts)
    return reply
```

Після цих кроків можна переходити до розробки частини, де сервер буде отримувати повідомлення від користувача та надавати відповідь на його повідомлення, яке буде згенероване саме функцією, в якій було ініціалізовано та налаштовано інтелектуальну мовну модель.

3.3 Основний механізм роботи

Коли користувач потрапляє на веб-сайт, він повинен зареєструватись або авторизуватись. Після авторизації він потрапляє на основну сторінку, де може ввести повідомлення текстом або обере мову і буде використовувати голосове введення.

Для використання голосового введення користувачу потрібно обрати мову, натиснути на кнопку запису. Відповідний програмний код який відповідає за цей процес, наведено в лістингу 3.3.

Лістинг 3.3 Частина коду, яка відповідає за початок запису голосу та його розпізнавання

```
// Set the onClick property of the start button
document.querySelector("#startRecording").onclick = () => {

    final_transcript = [];
    // Start the Speech Recognition
    speechRecognition.start();
};
```

Після вибору мови та натиснення кнопки запису користувач має сказати повідомлення голосом, на яке хоче отримати відповідь. Частина коду, яка відповідає за цей процес наведений у лістингах 3.4 та 3.5.

Лістинг 3.4 Частина коду, яка відповідає за припинення запису голосу та його розпізнавання

```
// Set the onClick property of the stop button
    document.querySelector("#stopRecording").onclick = () => {
        // Stop the Speech Recognition
        speechRecognition.stop();
```

```
        document.querySelector("#status_waiting").textContent =
"Downloading...";
```

Лістинг 3.5 Частина коду, яка відповідає за формування розпізнаного тексту

```
speechRecognition.onresult = (event) => {
        interim_transcript = "";
        // Loop through the results from the speech recognition object.
        for (let i = event.resultIndex; i < event.results.length; ++i) {
          // If the result item is Final, add it to Final Transcript, Else add it to
Interim transcript
          if (event.results[i].isFinal) {
            final_transcript += event.results[i][0].transcript;
          } else {
            interim_transcript += event.results[i][0].transcript;
          }
        }
      };
```

Після зупинки запису голосу і розпізнавання природньої мови користувача, розпізнаний текст відправляється на сервер з клієнтської сторони для подальшої його обробки. Процес відправлення відображено в лістингу 3.6

Лістинг 3.6 Частина коду, яка відповідає за відправку розпізнаного тексту у форматі JSON на сервер

```
// Get the final_transcript and interim_transcript and send them to the server
const textToPost = final_transcript;
const interimTextToPost = interim_transcript;
```

```
const csrfToken =
document.querySelector('input[name="csrfmiddlewaretoken"]').value;
        console.log("Final transcript:", final_transcript);
        console.log("Interim transcript:", interim_transcript);
        console.log("Sending data to server:", textToPost);
        fetch('/', {
                    method: 'POST',
                    headers: {
                       'Content-Type': 'application/json',
                       'X-CSRFToken': csrfToken,
                    },
                    body: JSON.stringify({ text: textToPost, interimText:
interimTextToPost}),
                }).then(response => {
                    document.querySelector("#status_waiting").textContent = "    ";
                if (response.ok) {
                    window.location.reload();
                } else {
                    console.error('Помилка відправки запиту на сервер');
                }
            })
            };
        } else {
            console.log("Speech Recognition Not Available");
```

```
}
```

Після відправлення розпізнаного тексту на сервер, його потрібно видобути з JSON файлу, цей процес наведено у лістингу 3.7.

Лістинг 3.7 Частина коду, за допомогою якого сервер отримує розпізнаний текст користувача

```
if request.method == 'POST':
    user = request.user
    content_type = request.META.get('CONTENT_TYPE', '')
    # Voice message
    if content_type == 'application/json':
        data = json.loads(request.body)
        # getting text from JSON
        received_text = data.get('text', '')
```

Після отримання та видобування тексту, його потрібно перевірити на валідність, тобто чи текст не є пустим і чи є там хоча б одне слово і передати текст моделі. Цей процес наведено у лістингу 3.8

Лістинг 3.8 Частина коду, яка відповідає за перевірку валідності отриманого тексту, передачі його моделі та збереження його у базі даних

```
if len(received_text) > 0:
    print(received_text)
    user_message = Message(user=user, user_message=received_text)
    user_message.save()
```

Після передачі моделі тексту, потрібно отримати її відповідь після чого зберегти її в базі даних. Це наведено у лістингу 3.9.

Лістинг 3.9 Частина коду, яка відповідає за отримання відповіді від моделі та збереження її у базі даних

```
# Getting answer from model
response = handle_message(received_text)
response_message = Message(user=user, model_response=response)
response_message.save()
```

Після цих кроків потрібно вивести збережені повідомлення з бази даних на веб-сайт, тобто на клієнтську сторону. Код, який відповідає за це наведено у лістингу 3.10.

Лістинг 3.10 Частина коду, яка відповідає за передачу з серверної частини веб-застосунку повідомлень користувача та моделі, які були збережені в процесі обробки

```
messages = Message.objects.filter(user=user).order_by('-timestamp')
return render(request, 'chat.html', {'form': form, 'messages': messages,
'generated_response': response})
```

Після цього потрібно прописати код сторінки, щоб користувач міг візуально побачити всі повідомлення свої та відповіді на них від моделі (лістинг 3.11).

Лістинг 3.11 Код HTML який відповідає за відображення повідомлень, які були надіслані з серверу

```
<div id="chat-container">
    <div id="chat-messages">
        {% for message in messages reversed %}
            {% if message.user_message %}
                <div class="message user-message">{{ message.user_message }}</div>
            {% endif %}
            {% if message.model_response %}
```

```
                <div class="message model-message">{{ message.model_response
}}</div>
            {% endif %}
        {% endfor %}
    </div>
</div>
```

3.4 Тестування веб-застосунку

Тестування призначене для того, щоб повністю перевірити функціонал програмного забезпечення.

Почнемо з самого початку. Коли користувач відкриває веб-сайт, він повинен або зареєструватись, або авторизуватись, якщо вже має зареєстрований акаунт. Я, як новий користувач, почну з реєстрації, після чого авторизуюсь щоб отримати доступ до головної сторінки сайту (рис. 3.2-3.4).

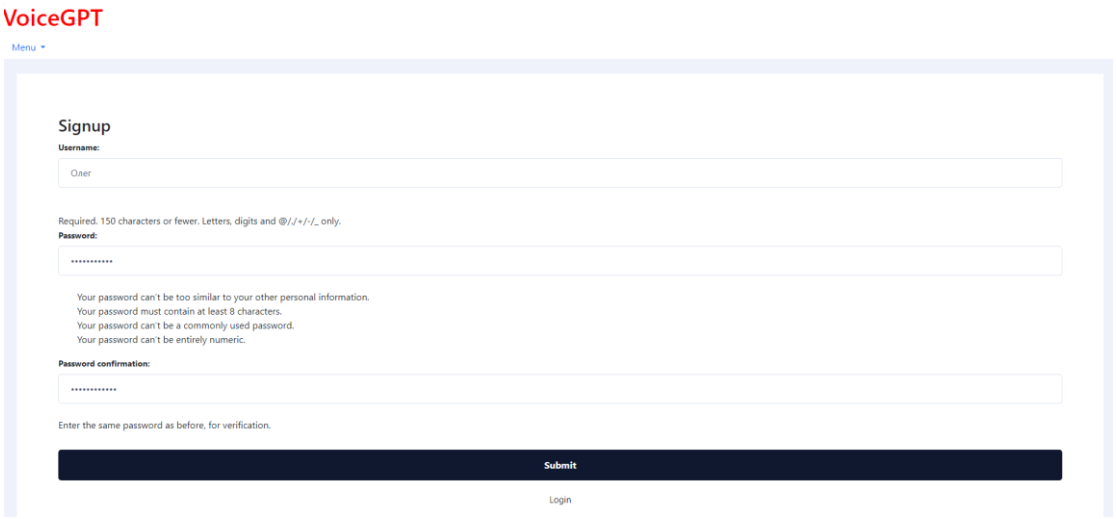

Рис. 3.2 Сторінка реєстрації

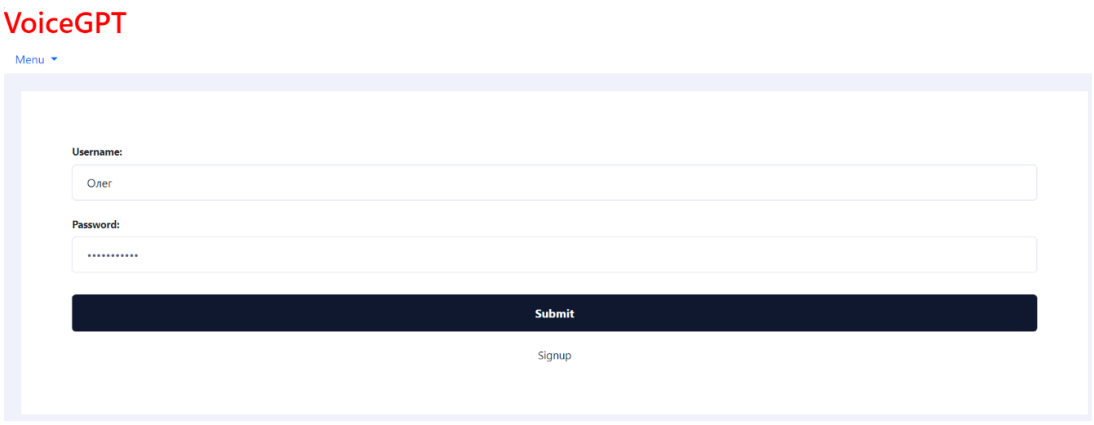

Рис. 3.3 Сторінка авторизації

Рис. 3.4 Головна сторінка веб-застосунку

Оскільки я ввійшов як новий користувач, то історії повідомлень ніякої нема, спробуємо написати «Привіт»(Рис. 3.5).

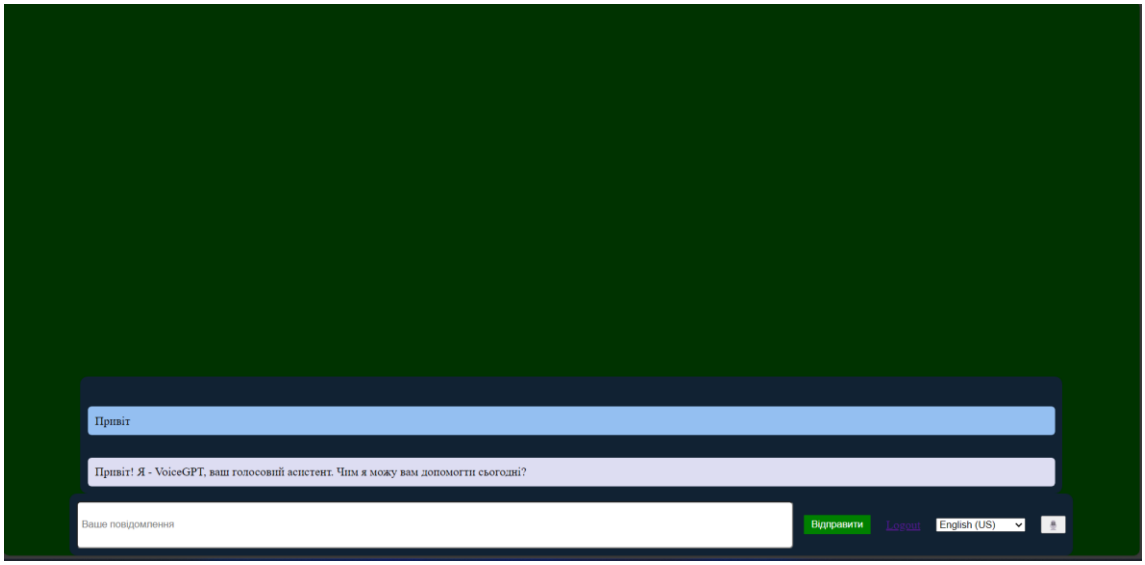

Рис. 3.5 Сторінка веб-сайту з отриманим повідомленням від моделі

Як бачимо, текстове введення працює. Спробуємо використати голосове введення повідомлень. Спробуємо сказати «Як твої справи?» (Рис. 3.6-3.7).

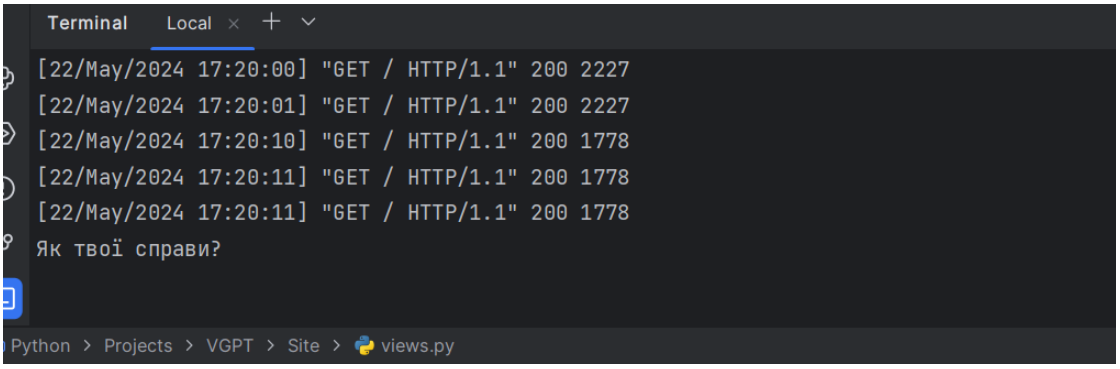

Рис. 3.6 Розпізнаний текст вивівся в консолі сервера, який був розпізнаний та надісланий з клієнтської частини

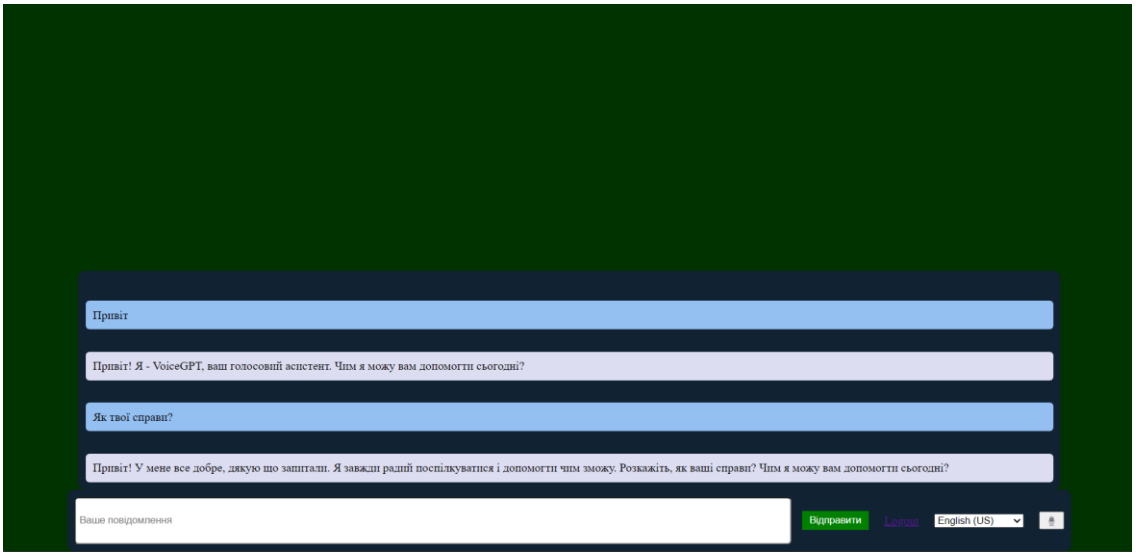

Рис. 3.7 Отримане повідомлення від моделі у відповідь на повідомлення користувача, яке було надіслане за допомогою голосового введення

Спробуємо тепер щось сказати англійською мовою, наприклад «Can you help me?» (Рис. 3.8-3.9).

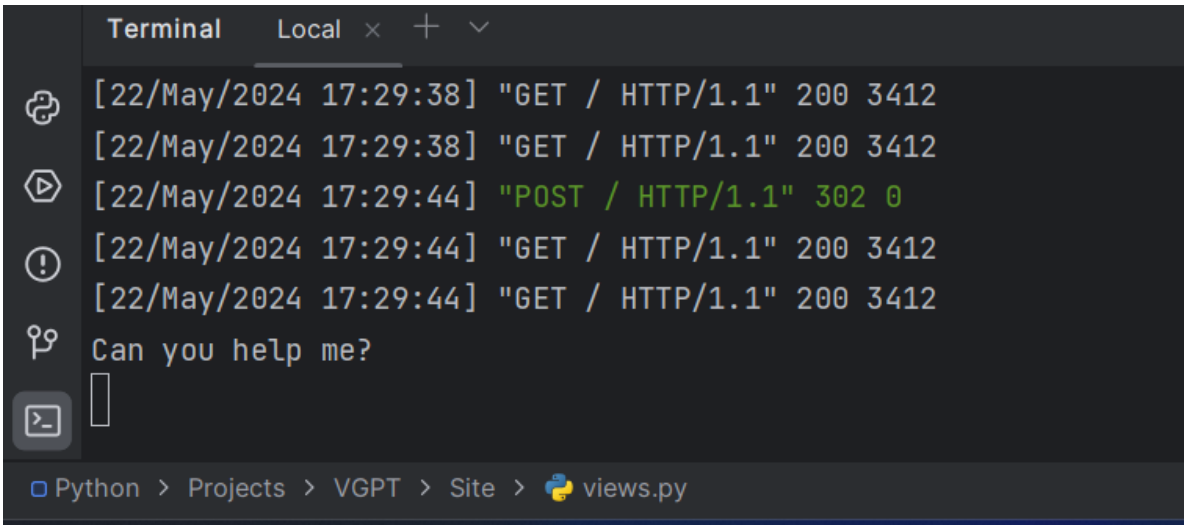

Рис. 3.8 Розпізнаний текст в консолі сервера

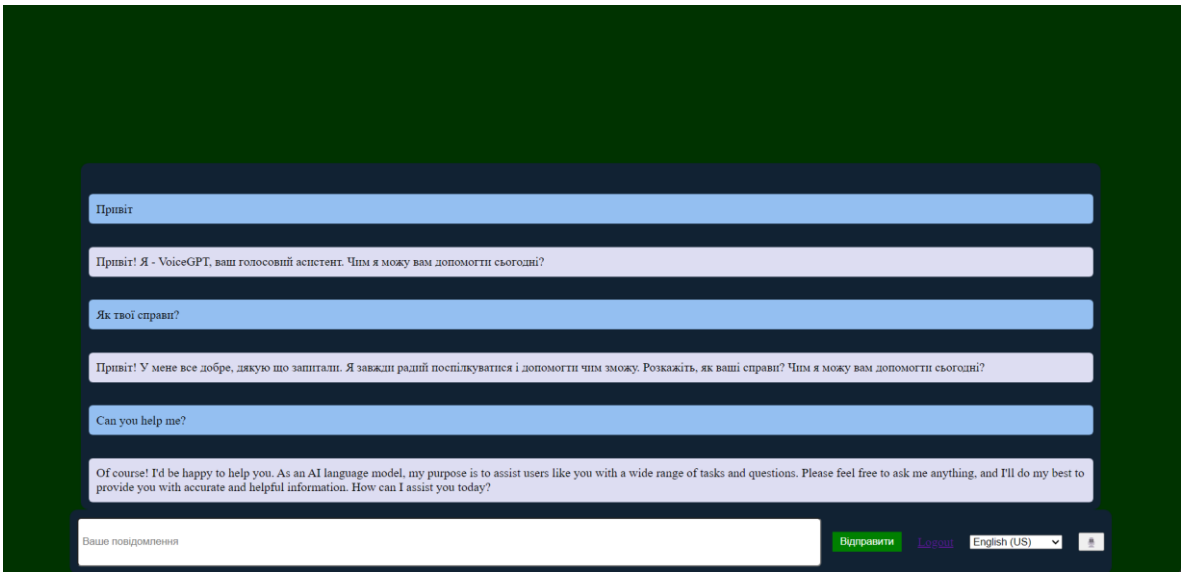

Рис. 3.9 Відповідь моделі на повідомлення користувача, яке було сказано англійською мовою за допомогою голосового введення інформації

3.5 Висновки до розділу 3

Отже, основним інструментом для моделювання дій було використано діаграму діяльності, яка дозволяє візуалізувати основні процеси. Інтелектуальну мовну модель було використано від компанії Anthropic, а саме claude-3-opus-20240229, яка була інтегрована за допомогою бібліотеки Anthropic REST API у проект та була основним засобом для генерації відповідей на повідомлення користувача.

Користувач, потрпивши на веб-сайт, має зареєструватись або авторизуватись, після чого має доступ до головної сторінки для введення

повідомлень. Для голосового введення повідомлень передбачено вибір мови. Вибір повстає між англійською та українською мовами, після чого користувач починає запис і проговорює своє повідомлення, яке в цей момент в реальному часі поступово розпізнається. Після цього розпізнаний текст відправляється на сервер для подальшої обробки. Сервер, отримавши повідомлення від користувача, передає його моделі, яка і генерує відповідь на повідомлення користувача. Всі повідомлення користувача і відповіді моделі зберігаються в базі даних, з якої на веб-сайті і відображаються. Тестування показало, що голосове та текстове введення працює належним чином. При відправленні повідомлень українською, користувач отримуватиме відповідь українською, а при використанні англійської, відповідь буде генеруватись англійською.

ВИСНОВКИ

У рамках цього проекту був розроблений та впроваджений веб-застосунок з використанням інтелектуальної моделі та можливістю голосового введення інформації. Ідея проекту полягала у створенні зручного та універсального засобу взаємодії з сучасними системами штучного інтелекту, який дозволяє користувачам отримувати потрібну інформацію чи відповіді на запитання не лише шляхом текстового вводу, а й за допомогою голосу.

На початковому етапі було проаналізовано предметну область, а саме: вивчено поняття інтелектуальних мовних моделей та проведено огляд популярних розумних чат-ботів. Наступним кроком стало обрання засобів реалізації проекту: огляд існуючих засобів розпізнавання природної мови, способів передачі тексту з клієнтської частини на сервер, технологій для створення веб-інтерфейсу та серверної частини застосунку.

Під час проектування було розроблено діаграму діяльності, що демонструє основний механізм роботи веб-застосунку, а також проведено огляд методів інтеграції обраної інтелектуальної моделі. На етапі реалізації створено повноцінний веб-застосунок із системою реєстрації та аутентифікації користувачів, а також головною сторінкою, де користувач може вводити повідомлення голосом англійською або українською мовами. Також присутній текстовий ввід для написання повідомлень з клавіатури.

Розроблений веб-застосунок успішно пройшов тестування і повністю задовольняє поставлені вимоги. Він готовий до впровадження та використання в реальних умовах.

Завдяки виконанню цього проекту було отримано цінні знання та практичний досвід у сферах розробки веб-застосунків, інтеграції інтелектуальних моделей та систем розпізнавання голосу, а також проектування зручних та універсальних користувацьких інтерфейсів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Keselj V. Speech and Language Processing (second edition) Daniel Jurafsky and James H. Martin (Stanford University and University of Colorado at Boulder) Pearson Prentice Hall, 2009, xxxi+988 pp; hardbound, ISBN 978-0-13-187321-6, $115.00. Computational Linguistics. 2009. Т. 35, № 3. С. 463–466. URL: https://doi.org/10.1162/coli.b09-001 (дата звернення: 27.02.2024).

2. Can large language models pass official high-grade exams of the European Society of Neuroradiology courses? A direct comparison between OpenAI chatGPT 3.5, OpenAI GPT4 and Google Bard / G. D'Anna та ін. Neuroradiology. 2024. URL: https://doi.org/10.1007/s00234-024-03371-6 (дата звернення: 28.02.2024).

3. Bengio Y., Courville A., Goodfellow I. Deep Learning. MIT Press, 2016. 800 с..

4. Heaton J. Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning. Genetic Programming and Evolvable Machines. 2017. Т. 19, № 1-2. С. 305–307. URL: https://doi.org/10.1007/s10710-017-9314-z (дата звернення: 28.02.2024).

5. Goldberg Y. Neural Network Methods for Natural Language Processing. Synthesis Lectures on Human Language Technologies. 2017. Т. 10, № 1. С. 1–309. URL: https://doi.org/10.2200/s00762ed1v01y201703hlt037 (дата звернення: 02.03.2024).

6. Language Models are Few-shot Multilingual Learners / G. I. Winata та ін. Proceedings of the 1st Workshop on Multilingual Representation Learning, м. Punta Cana, Dominican Republic. Stroudsburg, PA, USA, 2021. URL: https://doi.org/10.18653/v1/2021.mrl-1.1 (дата звернення: 02.03.2024).

7. ChatGPT: Optimizing Language Models for Dialogue. AutoGPT Official. URL: https://autogpt.net/chatgpt-optimizing-language-models-for-dialogue/ (дата звернення: 02.03.2024).

8. Trajectory-BERT: Pre-training and fine-tuning bidirectional transformers for crowd trajectory enhancement / L. Li та ін. Computer Animation and Virtual Worlds. 2023. URL: https://doi.org/10.1002/cav.2190 (дата звернення: 03.03.2024).

9. Singh S. BERT Explained: State-of-the-art language model for NLP. Labellerr. URL: https://www.labellerr.com/blog/bert-explained-state-of-the-art-language-model-for-nlp/ (дата звернення: 03.03.2024).

10. Efficient Estimation of Nepali Word Representations in Vector Space / J. Bhatta та ін. Journal of Innovations in Engineering Education. 2020. Т. 3, № 1. С. 71–77. URL: https://doi.org/10.3126/jiee.v3i1.34327 (дата звернення: 03.03.2024).

11. Systems C. o. N. I. P. Advances in neural information processing systems. Cambridge, MA : MIT Press, 2007. 1668 с.

12. Foundations of Statistical Natural Language Processing. Cambridge, Mass : MIT Press.

13. Mastering the game of Go with deep neural networks and tree search / D. Silver та ін. Nature. 2016. Т. 529, № 7587. С. 484–489. URL: https://doi.org/10.1038/nature16961 (дата звернення: 03.03.2024).

14. Widom J. D., Ullman J. D. A First Course in Database Systems (2nd Edition). Prentice Hall, 2001. 528 с.

15. Feldman R., Sanger J. Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data. Cambridge University Press, 2002.

16. Freeman E. T., Robson E. Head First JavaScript Programming. O'Reilly Media, Incorporated, 2014.

17. Crockford D. JavaScript: The good parts. Sebastopol, CA : O'Reilly, 2008. 153 с.

18. Nixon R. Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5. 4-те вид. O'Reilly Media, 2014. 812 с.

19. Duckett J. HTML & CSS: Design and build websites. 2014. 490 с.

20. Musciano C., Kennedy B. HTML & XHTML : The Definitive Guide. O'Reilly, 2000. 672 с.

21. Smith A. Python Programming: An Introduction to Computer Science. Independently Published, 2018.

22. Django documentation. Django Project. URL: https://docs.djangoproject.com/en/5.0/ (дата звернення: 17.03.2024).

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробленнявеб-застосункуз використанням інтелектуальноїмоделіз можливістюголосовоговведення інформації»

Викона в: здобувач вищої освіти гр. ШІД-41
Олег ЛУГОВИК
Керівник: кандидат технічних наук, доцент
Максим ФЕСЕНКО

2024 рік

***Мета роботи*:** створення веб-застосунку, що дозволяє користувачам отримувати потрібну інформацію або відповіді на запитання шляхом голосового введення повідомлень, для спрощення процесу знаходження інформації.

***Об'єкт дослідження*:** процес знаходження інформації.

***Предмет дослідження*:** веб-застосунокдля пошуку інформації шляхом голосового введення повідомлень.

Основні завдання кваліфікаційної роботи:

1. *Проаналізувати інтелектуальні системи та моделі, які використовуються для створення веб-застосунків.*
2. *Розглянути засоби розробки для веб-застосунку.*
3. *Розглянути способи інтеграції інтелектуальної моделі у веб-застосунок.*
4. *Реалізувати голосове введення інформації у веб-застосунок.*

ВИМОГИ ДО ЗАСТОСУНКУ

1. *Можливість реєстрації користувачами.*
2. *Можливість текстового вводу повідомлень.*
3. *Можливість голосового вводу повідомлень.*
4. *Можливість обирати мову для голосового вводу.*
5. *Можливість перегляду історії чату.*

ДІАГРАМА АКТИВНОСТЕЙ

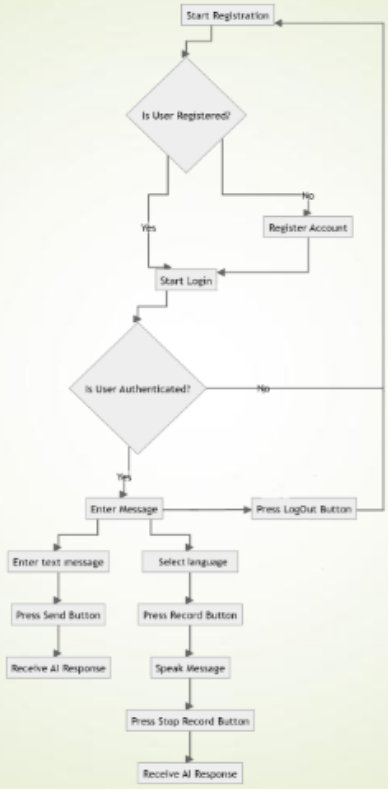

Рис. 1 – Діаграма діяльності

ЗАСІБ ДЛЯ РОЗПІЗНАВАННЯ ПРИРОДНОЇ МОВИ У БРАУЗЕРІ

Web Speech API – це технологія, за допомогою якої реалізовуються функції розпізнавання та синтез мовлення.
API має дві частини - Speech recognitioni Speech synthesis.

Speech recognitionвикористовується для розпізнавання природної мови у браузерах без потреби обробки на сервері.

Основні методи- onstart, onresult, onerror, onend

Speech synthesis використовується для синтезу мовлення та дозволяє обирати голос, налаштовувати швидкість та тон голосу.

ЗАСОБИ ДЛЯ РЕАЛІЗАЦІЇ ВЕБ-ЗАСТОСУНКУ

ІНТЕРФЕЙС ВЕБ-ЗАСТОСУНКУ

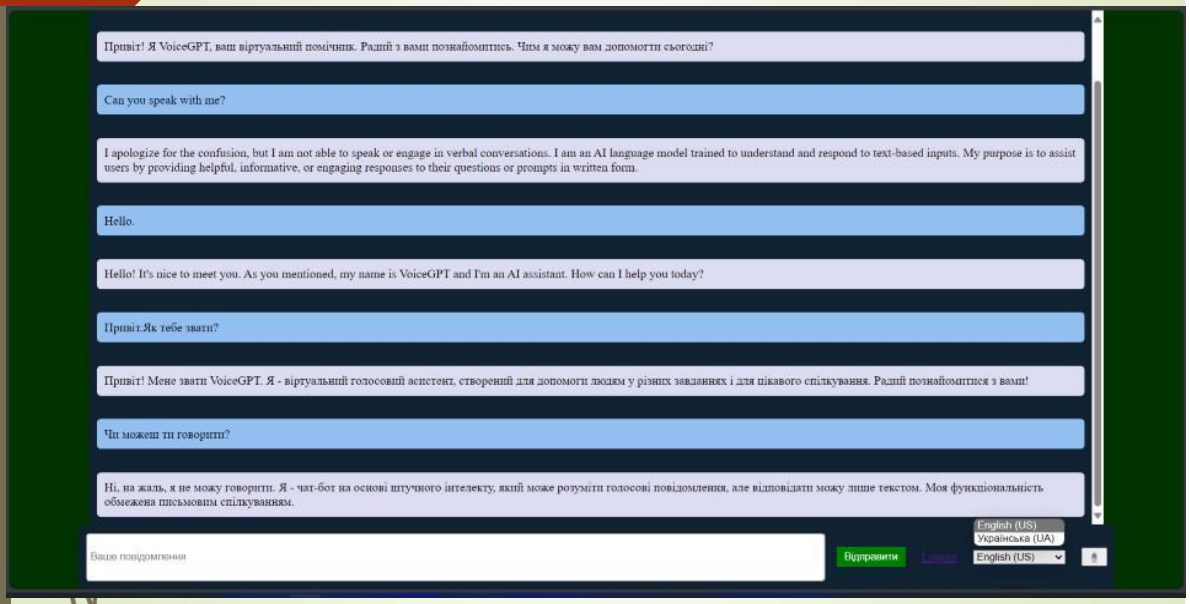

Рис. 2 – Візуальна частина веб-застосунку, а саме інтерфейс користувача

ІНТЕРФЕЙС ВЕБ -ЗАСТОСУНКУ

Рис. 3 – Текстове введення та відповідь моделі

ГОЛОСОВЕ ВВЕДЕННЯ ПОВІДОМЛЕНЬ

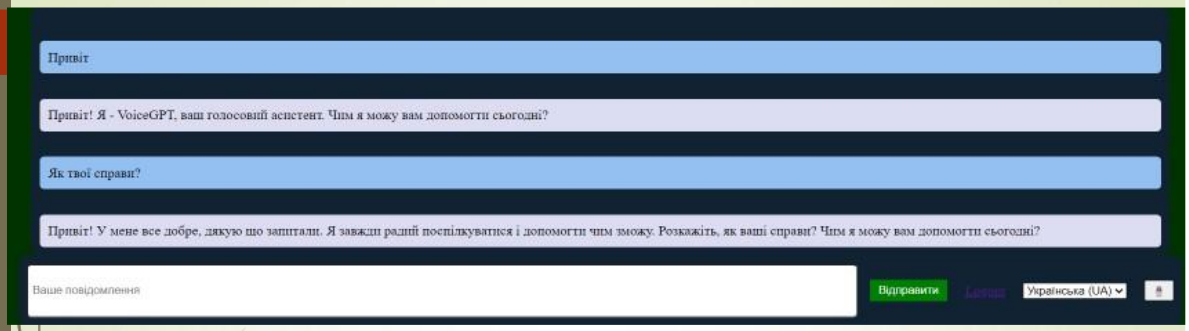

Рис. 4 – Відповідь моделі на повідомлення користувача, яке було введено за допомогою голосового введення

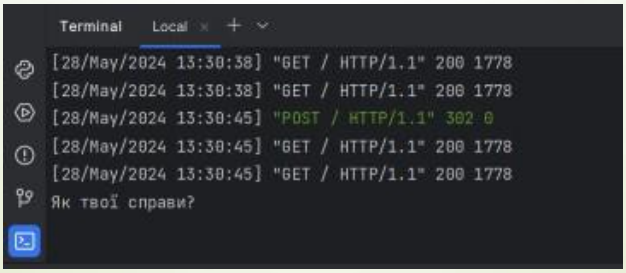

Рис. 5 – Розпізнаний текст з консолі серверу

ІНТЕГРАЦІЯ ІНТЕЛЕКТУАЛЬНОЇ МОДЕЛІ

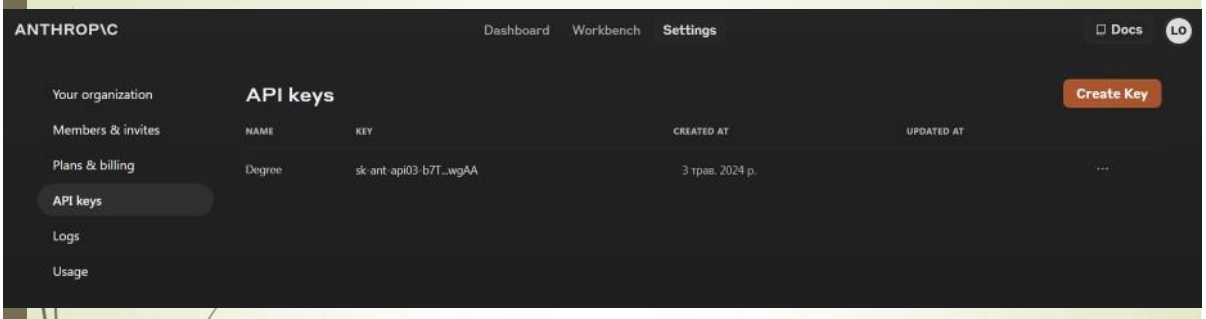

Рис. 6 – Отримання ключа API на сайті компанії Anthropic

```
13
14  # Key API Anthropic
15  API_key = "sk-ant-api03-b7ThhLupQ5Z-Z3P
16
17  client = anthropic.Anthropic(
18      api_key=API_key,
19  )
20
```

Рис. 7 – Прив’язка веб-застосунку до моделі за допомогою ключа API

ІНТЕГРАЦІЯ ІНТЕЛЕКТУАЛЬНОЇ МОДЕЛІ

```
2 usages  Oleg Lugovik
def handle_message(message_text):
    text = message_text
    message = client.messages.create(
        model="claude-3-opus-20240229",
        max_tokens=1000,
        temperature=0.0,
        system="You are assistant. Your name is VoiceGPT. You can understand the voice",
        messages=[
            {"role": "user", "content": text}
        ]
    )
    response_texts = [content.text for content in message.content if hasattr(content, 'text')]
    reply = '\n'.join(response_texts)
    return reply
```

Рис. 8 – Функція ініціалізації інтелектуальної моделі в програмному коді

ВИСНОВКИ

1. *У кваліфікаційній роботі було проаналізовано інтелектуальні системи та моделі.*
2. *Було проаналізовано засіб розпізнавання природної мови у браузері.*
3. *Оглянуто засоби розробки клієнтської частини веб-застосунку.*
4. *Оглянуто засоби розробки серверної частини веб-застосунку.*
5. *Було інтегровано інтелектуальну мовну модель.*
6. *Було розроблено веб-застосунок з використанням інтелектуальної моделі з можливістю голосового введення інформації.*

Дякую за увагу!