МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
МАРІУПОЛЬСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ЕКОНОМІКО-ПРАВОВИЙ ФАКУЛЬТЕТ
КАФЕДРА СИСТЕМНОГО АНАЛІЗУ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Допустити до захисту

Завідувач кафедри

_____ Мнацаканян М.С.

(підпис) *(ПІБ завідувача кафедри)*

«___»_____20__ р.

ВЕБ-ЗАСТОСУНОК НОТАТОК ІЗ МОДУЛЕМ РОЗПІЗНАВАННЯ ГОЛОСУ

Кваліфікаційна робота
здобувача вищої освіти
першого (бакалаврського) рівня
вищої освіти
освітньо-професійної програми
«Комп'ютерні науки»
(назва освітньо-професійної програми)

Кабанов Микита Андрійович
(прізвище, ім'я, по батькові здобувача вищої освіти)

Науковий керівник:
Козловський В.В. д.т.н., професор
(прізвище, ініціали, науковий ступінь, вчене звання)

Рецензент:

_____
(прізвище, ініціали, науковий ступінь, вчене звання, місце роботи)

Кваліфікаційна робота захищена
з оцінкою_____
Секретар ЕК_____
«____» _____20__ р.

Київ - 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
МАРІУПОЛЬСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ЕКОНОМІКО-ПРАВОВИЙ ФАКУЛЬТЕТ
КАФЕДРА СИСТЕМНОГО АНАЛІЗУ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Рівень вищої освіти *Бакалавр*
Шифр та назва спеціальності *122 «Комп'ютерні науки»*
Освітньо-професійна програма *«Комп'ютерні науки»*

ЗАТВЕРДЖУЮ
Завідувач кафедри _____к.т.н._____
(науковий ступінь, вчене звання)

_____ _____Мнацаканян М.С._____
(підпис) *(ПІБ завідувача кафедри)*

«__» _____ 20__ р.

ПЛАН ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Кабанов Микита Андрійович
(прізвище, ім'я, по батькові)

1. Тема роботи: «Веб-застосунок нотаток з розпізнаванням голосу»

керівник роботи: Козловський В.В., д.т.н., професор

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом Маріупольського державного університету від «___» _____20__ р. №__

2. Строк подання здобувачем роботи _____.

3. Вихідні дані до роботи (мета, об'єкт, предмет):

Об'єкт дослідження – веб-застосунок з функцією розпізнаванням голосу.

Предмет дослідження – веб-застосунок-нотатник з модулем розпізнаванням голосу розроблений на базі фреймворку ReactJS та модуля розпізнавання голосу TensorflowJS.

Мета кваліфікаційної роботи – розробка веб-застосунка-нотатника, з функцією розпізнавання голосу.

4. Зміст роботи

Розділ 1. Огляд та аналіз предметної області.

Розділ 2. Проектування додатка.

Розділ 3. Розробка та тестування додатка.

Розділ 4. Перелік обов’язкового ілюстративного матеріалу:

5. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Аналіз предметної області та огляд аналогів. Написання 1 розділу, представлення керівнику		
2.	Вибір та опис використаних технологій. Написання 2 розділу, представлення керівнику		
3.	Розробка додатка за допомогою обраного стеку технологій. Написання 3 розділу, представлення керівнику		
4.	Загальне редагування та друк пояснювальної записки		
5.	Проходження нормоконтролю, перепліт пояснювальної записки.		
6.	Розробка тексту доповіді. Оформлення графічного матеріалу для презентації		

Здобувач _____ Кабанов М.А.
(підпис) *(прізвище та ініціали)*

Науковий керівник роботи _____ Козловський В.В.
(підпис) *(прізвище та ініціали)*

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

SDK *(Software development kit)*	Набір для розробки програмного забезпечення
IDE *(Integrated development environment)*	Інтегроване середовище розробки
API (Application programming interface)	Програмний інтерфейс
MVP *(Model–view–presenter)* MVC *(Model–view–controller)* MVVM *(Model–view–viewmodel)*	Шаблони проектування мобільних додатків
MVP *(Minimal valuable product)*	Мінімально життєздатний продукт
XML *(Extensible Markup Language)*	Розширена мова розмітки
URL *(Uniform Resource Locator)*	Визначник місцезнаходження сайту в мережі Інтернет
LVCSR (*Large Vocabulary Continuous Speech Recognition*)	Словниковий запас бібліотеки ,яка розпізнає мовлення
SDK *(Software development kit)*	Набір для розробки програмного забезпечення
IDE *(Integrated development environment)*	Інтегроване середовище розробки
UML *(Unified Modeling Language)*	Універсальна мова моделювання

ВСТУП

З розвитком таких технологій, як доставка і запис інформації, яку людина потребує, зайшли далеко вперед. Стосовно комп'ютерів, то варто згадати такі носії інформації, як дискети – гнучкі диски з магнітними головками. Дискети слугували переносними носіями для зчитування та зберігання інформації. Дуже швидко їм на зміну прийшли компакт-диски, які могли підтримувати різні формати файлів, а отже можна було записати на диск як і музику, так і текстовий формат, так і відео-формати. Ще на користь дисків грав той факт, що вони були більш довговічні та були більш витривалі до фізичних впливів та пошкоджень.

На сьогодні було зроблено величезний прогрес в розвитку читання та запису інформації, а також її обробки та навіть розпізнавання.Зросла в рази швидкість передачі даних та надійність в плані їх зберігання та використання запису форматування зчитування. Явище нейромереж надало можливість розвитку штучного інтелекту та розвитку технологій загалом, адже з нейромережами прості життєві функції виконують апаратні пристрої.

За допомогою такого розвитку технологій в кишені людини тепер лежить одночасно кілька пристроїв, які раніше могли не поміститись навіть в окрему валізу. Навіть прості речі такі, як, запис нотаток – став набагато прогресивнішим та гнучкішим завдяки прогресу технологій та розвитку штучного інтелекту, а саме штучного інтелекту в галузі виявлення та обробки звукових імпульсів.

Актуальність теми кваліфікаційної роботи «Веб застосунок-нотаток з модулем розпізнавання голосу» пояснюється поширенням тенденції розвитку штучного інтелекту та розвитку голосових помічників для різних людських потреб, необхідінстю дослідження засобів розпізнавання та синтезу звукових сигналів, підвищення якості отриманих даних для подальшої роботи та досліджень.

Відповідно до поставленої мети роботи визначено основні **завдання дослідження**:

– провести аналіз наукової та методичної літератури;

– проаналізувати актуальні speech to text додатки які використовуються у веб просторі, їх можливості та особливості;

– провести огляд методологій розробки програмних застосунків, актуальних speech-to-text рішень, архітектурних рішень, SDK та технологій;

– розробити веб-застосунок для демонстрації роботи технології розпізнавання голосу та друку отриманої інформації в режимі реального часу ;

– описати принципи роботи веб додатку принципи роботи фреймворків та бібліотек які були використані , його коду та провести тестування.

Для досягнення поставленої мети й виконання завдань використано метод системно-структурного аналізу наукової літератури, який дав змогу показати особливості систем виявлення та розпізнавання звукових сигналів та розробки веб-застосунків на базі фреймворк-бібліотеки ReactJs та бібліотеки розпізнавання звукових сигналів TensorflowJS. Для формулювання і систематизації висновків використано методи аналізу, формалізації, абстрагування та узагальнення.

Наукова новизна роботи полягає у створенні веб додатку який може за допомогою голосу записати інформацію в реальному часі у вигляді тексту та зберегти її .

Додаток має перспективу та може бути використаний для персональних потреб користувача.Також цей додаток може бути використаний для подальших досліджень у галузі штучного інтелекту та розпізнавання та обробки голосових сигналів.

РОЗДІЛ 1
АНАЛІЗ МЕТОДІВ ПРОЗРОБКИ ВЕБ-ЗАСТОСУНКІВ З ФІНКЦІЄЮ РОЗПІЗНАВАННЯ ГОЛОСУ

1.1. Процес розвитку ВЕБ-технологій розпізнавання голосу

Першим офіційним прикладом сучасної технології розпізнавання мовлення стала система «Одрі», розроблена компанією Bell Laboratories у 1950-х роках. [1]

Система «Одрі», яка займала цілу кімнату, змогла розпізнати лише 9 цифр (цифри 1-9), сказаних її розробником, але зробила це з вражаючою точністю 90%.

Його метою було допомогти операторам платних послуг приймати більше телефонних дзвінків по дроту, але висока вартість і нездатність розпізнавати широкий спектр голосів зробили його непрактичним.

Для розробки наступного справжнього прогресу знадобилося ще 12 років. Під час прем'єри на Всесвітній виставці в 1962 році Shoebox від IBM зміг розпізнати та розрізнити 16 слів.

До цього моменту розпізнавання мовлення було трудомістким. Попередні системи були створені для розпізнавання та обробки бітів звуку (фонем). Інженери IBM запрограмували машини використовувати звук і висоту кожної фонеми як підказку, щоб визначити, яке слово було сказано.

На початку 1970-х років Міністерство оборони почало визнавати цінність технології розпізнавання мови. Здатність комп'ютера обробляти природну людську мову стала неоціненною в багатьох сферах військової та національної оборони.

Тож було інвестовано п'ять років у програму дослідження розуміння мовлення DARPA — одну з найбільших програм такого роду в історії розпізнавання мовлення.

Одним із найвидатніших винаходів цієї дослідницької програми була «Гарпія» — система, яка могла розпізнавати понад 1000 слів — словниковий запас середньостатистичної дитини.

Перший продукт розпізнавання мовлення для споживачів був запущений у 1990 році компанією Dragon під назвою DragonDictate. У 1996 році IBM представила перший продукт розпізнавання голосу, який міг розпізнавати безперервну мову. [2]

Після запуску смартфонів у другій половині 2000-х років Google запустив програму голосового пошуку для iPhone. Через три роки Apple представила Siri, який тепер є видатним помічником з розпізнавання голосу.

Протягом останнього десятиліття кілька інших технологічних лідерів також розробили більш складне програмне забезпечення для розпізнавання голосу, наприклад Echo від Amazon із Alexa та Cortana від Microsoft — обидва вони діють як персональні помічники, які відповідають на голосові команди.

На сьогоднішній день голосові помічники можуть підлаштовуватись під різні мови та вони можуть бути наявні не тільки в типових гаджетах таких як :смартфон або комп’ютер, а наприклад настільна лампа або музична колонка.

Окрему увагу було прикуто до браузерних додатків які можуть перетворювати голос в текст та було встановлено що механізм роботи працює по такому прнципу: Об’єкт чітко промовляє слова, а система перетворює в текст і записує.

Отриманий результат, швидше за все, потребуватиме редагування: розстановка розділових знаків, корекція та перевірка правильності написання складних слів. Для більш високої точності синтезованого мовлення рекомендовано вимовляти слова чітко та голосно та використовувати апаратне забезпечення високої якості.

1.2 Порядок функціонуваня ВЕБ-застосунків на основі технології розпізнавання голосу

Програмне забезпечення для розпізнавання голосу на комп'ютерах вимагає, щоб аналоговий звук перетворювався на цифрові сигнали, відомі як аналого-цифрове перетворення. Щоб комп'ютер міг розшифрувати сигнал, він повинен мати цифрову базу даних або словниковий запас слів або складів, а також швидкий засіб для порівняння цих даних із сигналами.На рис.1.1.зображено графік розпізнавання людської мови

Шаблони мовлення зберігаються на жорсткому диску та завантажуються в пам'ять під час запуску програми. Компаратор звіряє ці збережені шаблони з виходом аналого-цифрового перетворювача — ця дія називається розпізнаванням шаблонів. [2]

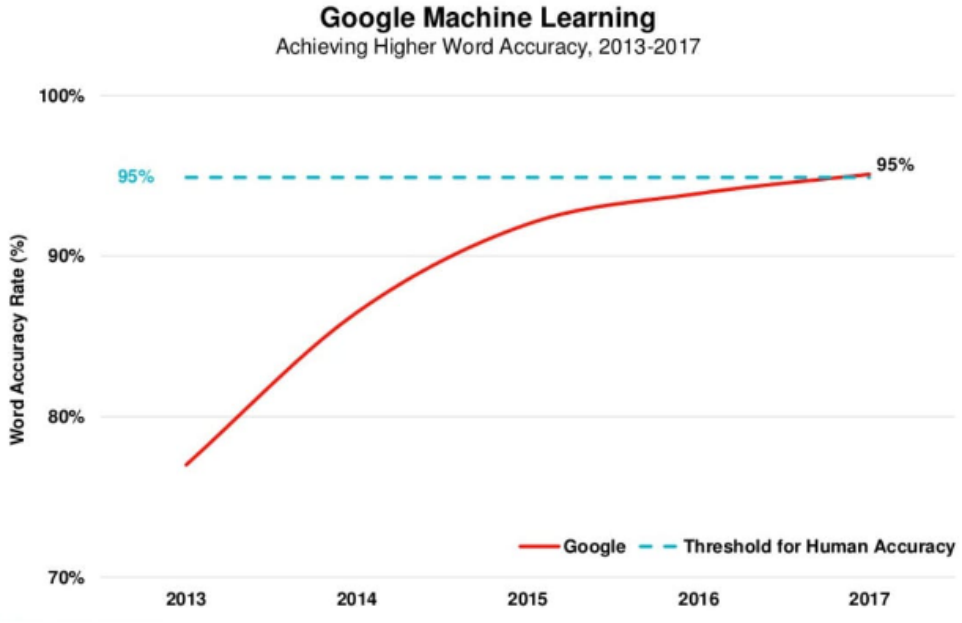


Рис. 1.1. Графік який показує точність розпізнавання людської мови від компанії Google

На практиці розмір ефективного словника програми розпізнавання голосу безпосередньо залежить від обсягу оперативної пам'яті комп'ютера, на якому вона встановлена. Програма розпізнавання голосу працює в рази швидше, якщо весь словниковий запас можна завантажити в оперативну пам’ять, порівняно з пошуком деяких збігів на жорсткому диску. Швидкість обробки також має вирішальне значення, оскільки вона впливає на те, наскільки швидко комп’ютер може шукати збіги в оперативній пам’яті.

У сучасних системах розпізнавання мовлення використовуються складні методи статистичного моделювання. Донедавна найпоширенішою була так звана прихована марківська модель, якої досить розпізнати лише частину фонем одного слова, інші вона підбирає за принципом ймовірності, тобто. вгадує слова, спираючись на бібліотеку шаблонів. [3]

На основі цієї моделі з'явилися більш сучасні системи, що базуються на рекурентній нейромережі. Вони також базуються на принципі ймовірності: якщо система не впізнає окремі слова через шум чи інші перешкоди, то вгадує їх, виходячи з контексту. Сучасні нейромережі добрі ще тим, що їх можна навчати.

Програмне забезпечення покладається на програму розпізнавання голосу, яка зчитує звукові сигнали за допомогою лінгвістичних алгоритмів. Потім він перетворює сигнали в текст за допомогою символів Unicode.

Це широкий огляд того, що робить програмне забезпечення для перетворення мови в текст.

Мова породжує вібрації. Програмне забезпечення вловлює ці вібрації та перекладає їх на мову, зрозумілу програмному забезпеченню та комп’ютерам, через аналоговий перетворювач. По суті, конвертер аналізує звукові хвилі в аудіофайлі та переглядає їх, щоб ідентифікувати звуки.

Потім ці звуки розбиваються на мілісекунди, щоб їх можна було зіставити з базою даних фонем. Фонема представляє одну одиницю перцептивно відмінного звуку в певній мові. Потім математична модель порівнює фонеми зі словами, фразами чи реченнями. [4]

1.3 Аналіз алгоритмів ВЕБ-технологій розпізнавання голосу

1.3.1 Алгоритми динамічного програмування або DWT

При використанні методів та алгоритмів динамічного програмування відбувається контекстно залежна класифікація. При її реалізації з потоку промови виділяються окремі лексичні елементи - фонеми та алофони, які потім об'єднуються в склади та морфеми.

Динамічне викривлення часу – це алгоритм, який обчислює оптимальний шлях викривлення між двома даними зі звуком, щоб на виході були значення викривлення шляху та відстань між двома даними. Динамічний Шлях — це відстань між порівняльними двома графіками-шаблонами ,якщо траєкторія викривлення являється мінімальною, патерни можна вважати однаковими.

Різний час вирівнювання мовлення є основною проблемою для вимірювання відстані в розпізнаванні мовлення. Невеликі зсуви призводять до неправильної ідентифікації. DTW є ефективним методом визначення часу проблеми з вирівнюванням. Тому цей алгоритм є більш реалістичним для використання для вимірювання подібності шаблону (збіг патерна/шаблону). Оброблені дані завжди знаходяться в часовому поясі, тому вважається, що послідовність даних, які ми маємо, змінюється з часом.[6]

Приклад результату динамічного програмування зображено на рис.1.2.:

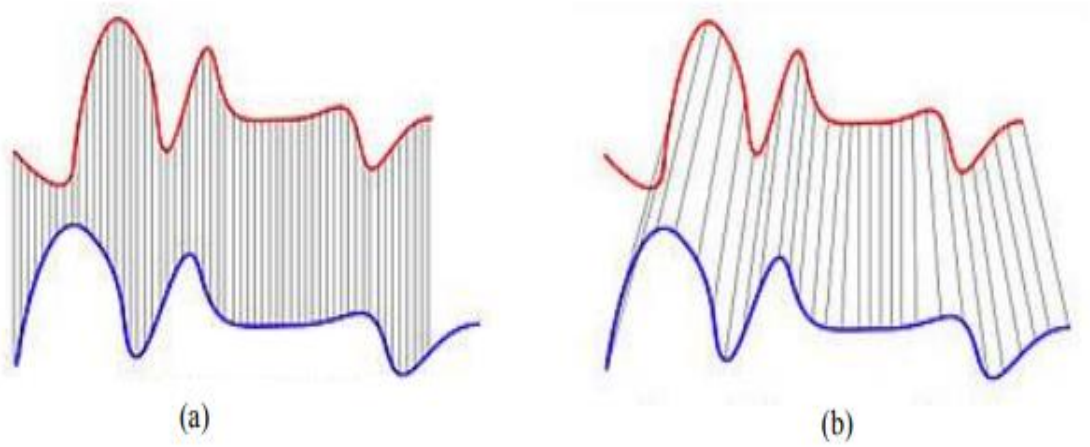


Рис. 1.2. Графічне представлення DWT

Де (a) –вихідна корекція голосової послідовності ,(b) – корекція за допомогою методів динамічного програмування

Алгоритм DTW призначений для вирівнювання двох векторних послідовностей поворотом осі часу кілька разів, доки не буде знайдено оптимальний збіг між двома послідовностями. Цей алгоритм працює як лінійне відображення осі для вирівнювання двох сигналів.

Після створення еталонних даних для навчальних даних потім виконуються тестові дані (тестування). Тест дані мають етапи, які майже такі ж, як процес у навчальних даних. У процесі цього випробування даних, користувач повертається до запису звуку у *.wav. Після отримання звукових характеристик від тестових даних, зіставлення буде виконано за допомогою методу DTW. Обчислюючи відстань від двох векторів між перевіреними даними та даними, які містяться в довідкових даних.

1.3.2 Приховані моделі Маркова

Приховані моделі Маркова (HMM), названі на честь російського математика Андрія Андрійовича Маркова, який розробив значну частину відповідної статистичної теорії, представлені та досліджені на початку 1970-х років. Вони вперше були використані для розпізнавання мови та успішно застосовані для аналізу біологічних послідовностей з кінця 1980-х років.

Нині вони розглядаються як специфічна форма динамічних байєсівських мереж, які базуються на теорії Байєса.

HMM — це статистичні моделі для захоплення прихованої інформації з спостережуваних послідовних символів (наприклад, нуклеотидної послідовності). Вони мають багато застосувань в аналізі послідовностей, зокрема для передбачення екзонів та інтронів у геномній ДНК, ідентифікації функціональних мотивів (доменів) у білках (профіль HMM), вирівнювання двох послідовностей (пара HMM).

У HMM передбачається, що система, що моделюється, є марковським процесом із невідомими параметрами, і завдання полягає у визначенні прихованих параметрів із спостережуваних параметрів. Хороший HMM точно моделює реальне джерело спостережуваних реальних даних і має можливість імітувати джерело.

Багато методів машинного навчання засновані на HMM, які успішно застосовуються для вирішення проблем, включаючи розпізнавання мовлення, оптичне розпізнавання символів, обчислювальну біологію, і вони стали фундаментальним інструментом у біоінформатиці: завдяки надійній статистичній основі, концептуальній простоті та пластичності вони адаптовані відповідати різноманітним проблемам класифікації. В обчислювальній біології прихована модель Маркова (HMM) — це статистичний підхід, який часто використовується для моделювання біологічних послідовностей.

Застосовуючи його, послідовність моделюється як результат дискретного стохастичного процесу, який проходить через ряд станів, які «приховані» від спостерігача. Кожен такий прихований стан випромінює символ, що представляє елементарну одиницю змодельованих даних, наприклад, у випадку послідовності білка, амінокислоти. У наступних розділах ми спочатку представимо концепцію прихованої марковської моделі як певного типу імовірнісної моделі в байєсівській системі; потім ми описуємо деякі важливі аспекти моделювання прихованих марковських

моделей для вирішення реальних проблем, приділяючи особливу увагу їх використанню в біологічному контексті. Щоб показати потенційні можливості цих статистичних підходів, ми представляємо стохастичне моделювання HMM, визначаючи спочатку архітектуру моделі, а потім алгоритми навчання та роботи. У цій роботі ми проілюструємо, як приклад, застосування в обчислювальній біології та біоінформатиці, і, зокрема, увага приділяється проблемі пошуку ділянок ДНК, які є метильованими або неметильованими (виявлення CpG-острівців).[7]

Моделі Маркова можуть бути використані в розпізнаванні голосу згідно з [8] Приховані моделі Маркова (HMM) забезпечують просту та ефективну структуру для моделювання спектральних векторних послідовностей, що змінюються в часі. Як наслідок, майже весь сучасний словниковий запас безперервної мови великої системи розпізнавання (LVCSR) засновані на HMM.

Тоді як основні принципи, що лежать в основі LVCSR на основі HMM, є досить прості, наближенні та спрощені припущення, задіяні у прямій реалізації цих принципів, будуть призводити до системи з низькою точністю та неприйнятною чутливістю до змін у робочому середовищі.

Таким чином, практичне застосування HMM у сучасних системах вимагає значного витонченість. Метою цього огляду є спочатку представити основну архітектуру систему LVCSR на основі HMM, а потім опишіть різні вдосконалення, необхідні для досягнення найсучаснішої продуктивності.

Ці удосконалення включають проекцію ознак, покращене моделювання коваріації, дискримінаційна оцінка параметрів, адаптація та нормалізація, Компенсація шумів і багатопрохідна система поєднання.

1.4 Огляд сучасних ВЕБ-застосунків на основі технології розпізнавання голосу

1.4.1. Google-документи

Сервіс Google Документи дозволяє переводити усне мовлення в записаний текст. Це вбудована функція із підтримкою різних мов.

Щоб активувати голосове введення, потрібно перейти до розділу «Інструменти» та натиснути «Голосове введення».[9]Інтерфейс додатку зображено на рис.1.3.

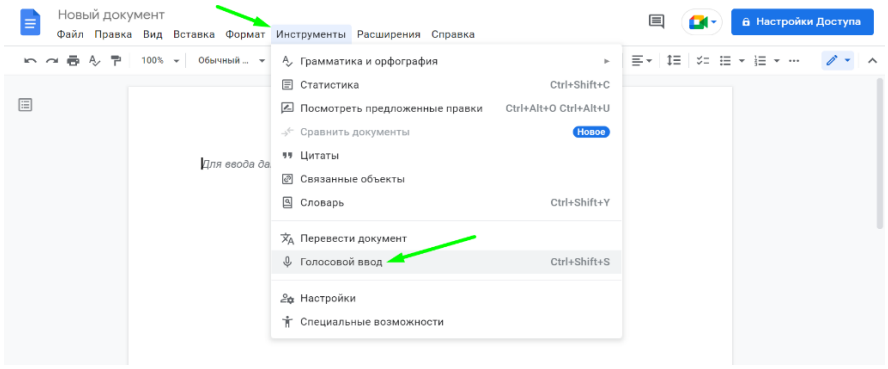


Рис. 1.3. Google-документи Як працювати з голосовим вводом

Після цього натиснути кнопку і говорити.Слова потрібно вимовляти повільно та чітко. Система вміє розпізнавати розділові знаки — просто потрібно говорити в потрібних місцях «Крапка», «Кома» і так далі. Англійською перелік голосових команд більший, повний список можна переглянути у Довідці.

Сервіс непогано конвертує голос у текст за умови чіткої та правильної вимови. Але коректура все одно може знадобитися — виправити регістр, перевірити розміщення розділових знаків і написання складних слів.

1.4.2. Speech to Text Bot

Онлайн-сервіс працює через браузер Chrome на робочому столі та деяких мобільних пристроях. Інтерфейс інтуїтивно зрозумілий: є вікно введення тексту, кнопка з мікрофоном для запуску запису та список команд, що підтримуються. Інтерфейс додатку зображено на рис.1.4.

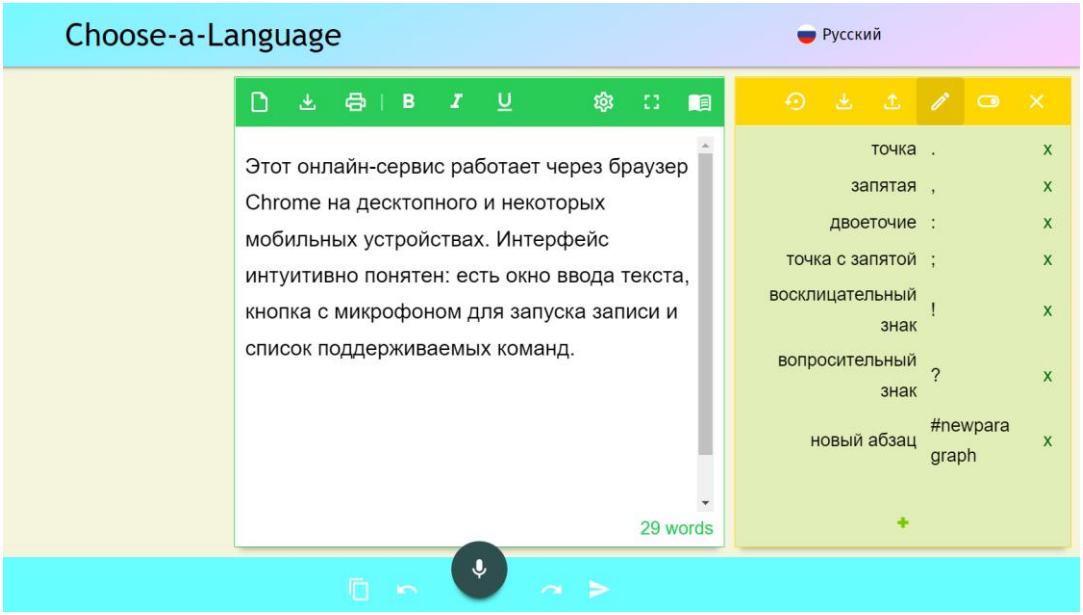


Рис. 1.4. Speech to Text Bot інтерфейс

Сервіс підтримує десятки різних мов. У налаштуваннях доступне форматування тексту: різні типи та розміри шрифту, написання речень із великої літери. Записаний текст можна редагувати, завантажувати, надсилати до друку, копіювати. Сервіс непогано перекладає мову в текст при надиктуванні, але не транскрибує аудіо- та відеофайли, навіть за їх гарної якості.

1.4.3. Speechpad

Speechpad - зручний онлайн-нотатник для мовного введення.Для диктування тексту доступно 15 різних мов. Доступно паралельне форматування тексту: заміна регістру, додавання знаків пунктуації та тегів. Запис мови включається та вимикається за потребою. Інтерфейс додатку зображено на рис.1.5.

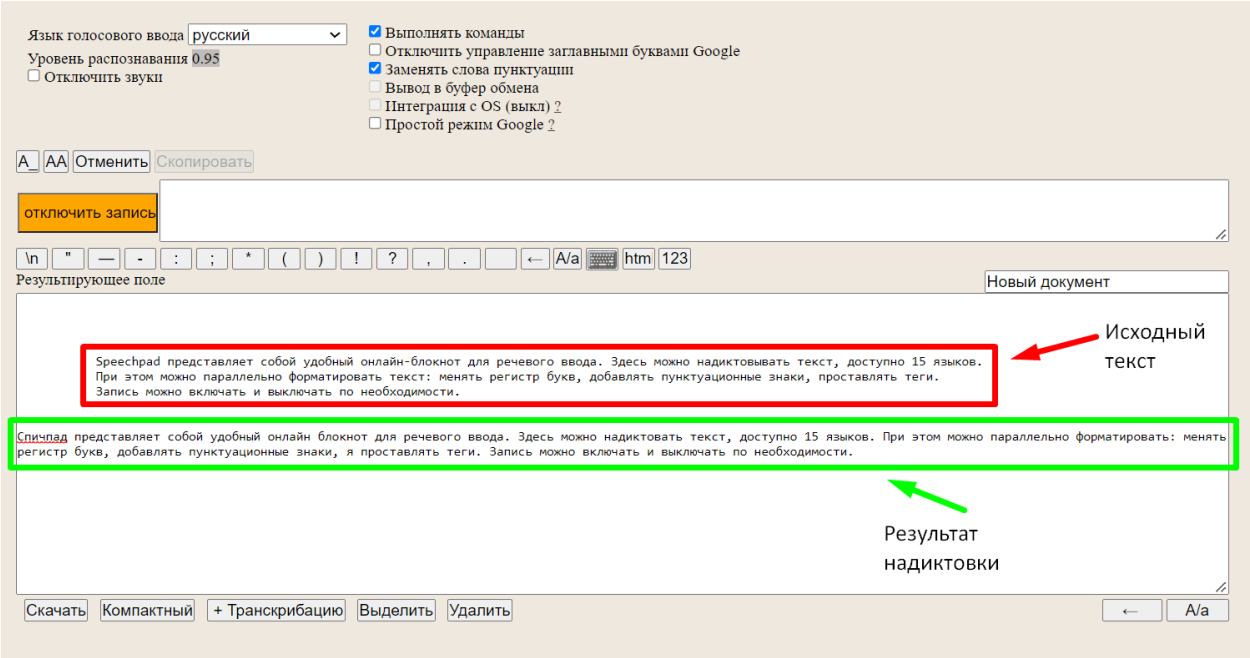


Рис. 1.5. Speechpad інтерфейс

Speechpad підтримує перетворення на текст аудіо- та відеозаписів. Для потрібно натиснути «+Транскрибацію» під полем введення. Після оновлення сторінки завантажити файл, вказати посилання або ID відео з YouTube. У разі потреби налаштувати параметри: якість і швидкість відтворення, вказівка тимчасових міток, захист від шумів. Після цього можна вмикати запис. Результат перетворення текстового формату з'явиться у вікні блокнота на цій же сторінці. Інтерфейс додатку зображено на рис.1.5.

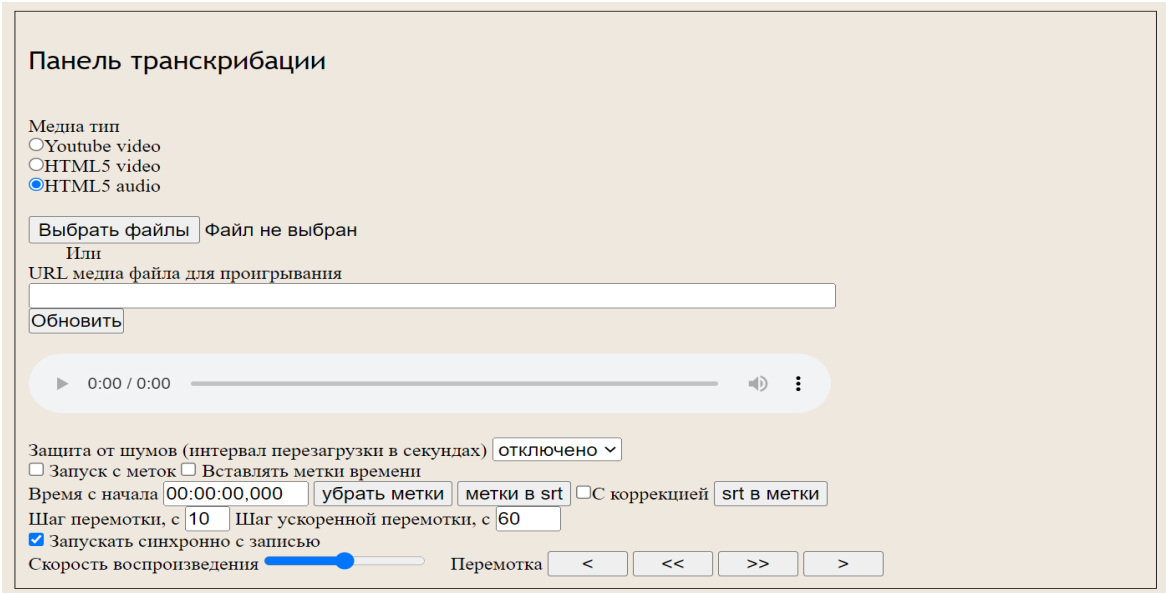


Рис. 1.5. Speechpad Транскрибація

Можна встановити розширення, щоб використовувати голосове введення в будь-якому текстовому полі браузера. Також є модуль інтеграції з Windows, Mac чи Linux.

1.4.4. Dictation

Індійський сервіс Dictation підтримує понад 100 мов, включаючи російську. Принцип роботи схожий на Google Документи, але швидкість розпізнавання вище. Під час диктування можна використовувати команди «Новий рядок» та «Новий абзац». Вказування розділових знаків враховується не завжди, але їх можна проставити вручну при редакції отриманого тексту. Інтерфейс додатку зображено на рис.1.6.

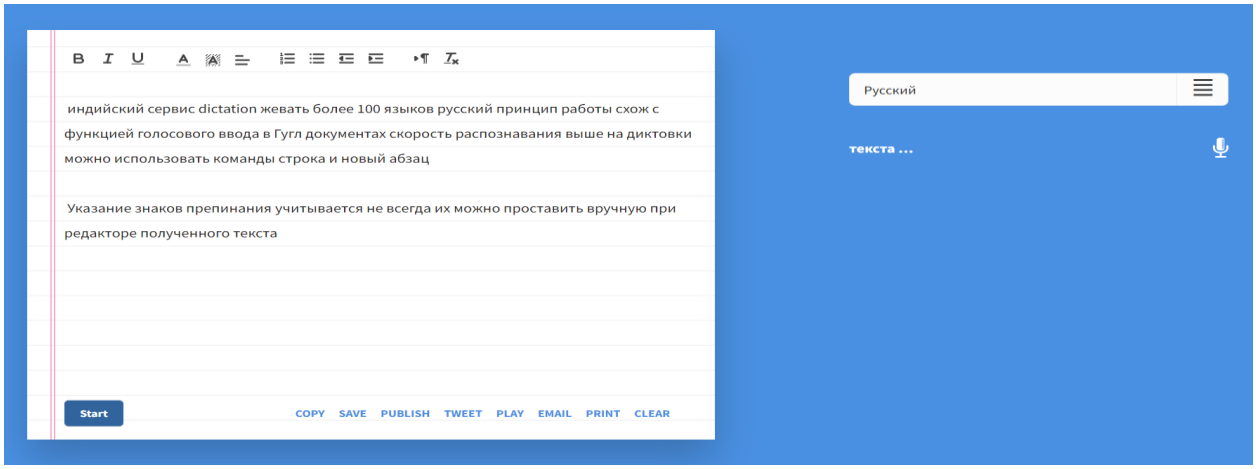


Рис. 1.6. Інтерфейс Dictation

Результат можна відформатувати і відредагувати,скопіювати,зберегти, опублікувати, твітнути, надіслати по email або роздрукувати. Якість розпізнавання в Dictation дозволяє транскрибувати аудіо- та відеофайли. Для цього потрібно включити їхнє відтворення поряд з мікрофоном. Готовий текст вимагатиме редакції.

Також розглянуто додатки, які доступні лише як мобільні версії на Android чи IOS.

1.4.5. Google Keep

Дозволяє диктувати нотатки голосом. Сервіс перетворює мову на текст, який за необхідності можна відредагувати. Створені нотатки синхронізуються на різних пристроях одного облікового запису. Їх можна відкрити на телефоні або комп'ютері, через програму або веб-версію, у Google Документах або Gmail. Інтерфейс додатку зображено на рис.1.7.

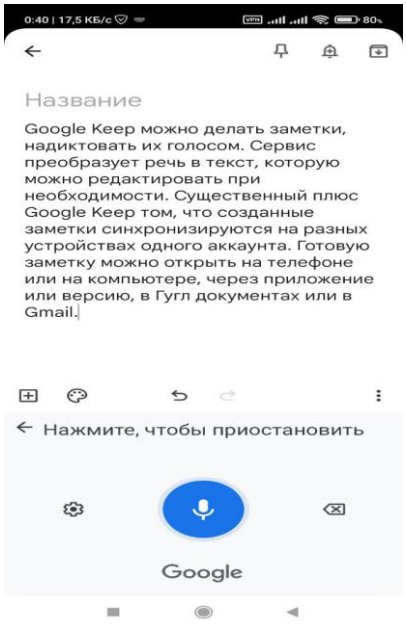


Рис. 1.7. Інтерфейс Google Keep

1.4.6 Dictation IOS

Плюс цієї програми для iOS — без обмежень за часом диктування. Dictation підтримує 40 мов, а надиктований текст можна швидко перекласти іншою мовою. Інтерфейс додатку зображено на рис.1.8.

Рис. 1.8. Інтерфейс Dictation

У Dictation можна швидко писати нотатки для соцмереж Також програма дозволяє транскрибувати аудіофайли. Всі записи синхронізуються на різних пристроях, коли увімкнено iCloud. Надиктованими текстами можна ділитися: відправляти до месенджерів або по email.

1.4.7. Speechnotes Android

Програма Speechnotes працює на основі розпізнавання мови Google. Для початку запису потрібно по кнопці мікрофона і почати говорити. Деякі знаки пунктуації можна озвучувати голосом, іншим доступна вбудована клавіатура, якою можна користуватися безпосередньо в процесі надиктовки.

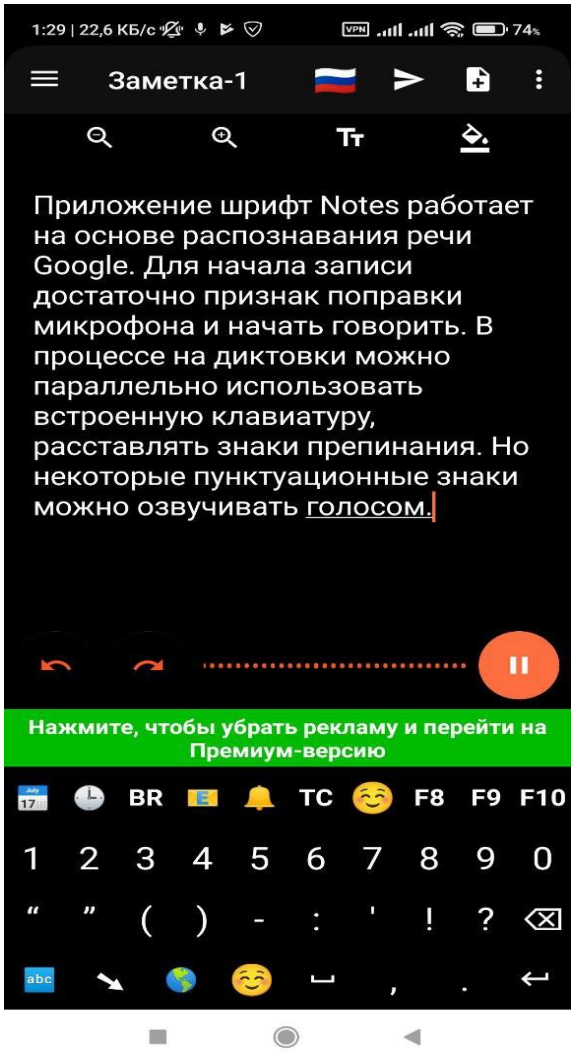


Рис. 1.9. Запис тексту голосом у мобільному додатку

Результат надиктування у Speechnotes вимагає зовсім незначної редакції

Готовий текст можна відредагувати, зберегти, надіслати, роздрукувати. У преміум-версії (від 1,5 $) доступне створення клавіш для вставки фраз, що використовуються найбільше.

1.5 Висновки до розділу

У першому розділі було розглянуто загальну формацію та розвиток технологічного прогресу людства а також було розглянуто принципи роботи розпізнавання голосу алгоритми завдяки яким можливо розробити

продукт який буде розпізнавати мовлення та розглянуто історичні прототипи пристроїв розпізнавання голосу.

Також було розглянуто сучасні аналоги додатків розпізнавання голосу та конвертації розмовних елементів в текст, розглянуто плюси та мінуси представлених аналогів та їх інтерфейси. Підсумовуючи дані тезиси можна сказати про те, що створення додатку для розпізнання голосу та конвертація його до тексту потребує великих знань та ресурсів і чим більшу точність текстового результату потрібно досягти, тим більшу бібліотеку шаблонів нам потрібно мати, а також тим більш навченою має бути нейронна мережа.

Одна з наявних проблем концепції розглянутої в дипломі , що текст після перетворення ще вимагає ручної корекції та правок знаків пунктуації. Технологіям і далі є куди розвиватися та удосконалюватися.

РОЗДІЛ 2

ПРОЕКТУВАННЯ ДОДАТКА

2.1 Огляд та аналіз середовища розробки Visual Studio Code

Visual Studio Code — це безкоштовний, легкий, але потужний редактор вихідного коду для робочого столу та веб-версії для Windows, macOS, Linux і Raspberry Pi. Він має вбудовану підтримку JavaScript, TypeScript і Node.js, а також багату екосистему розширень для інших мов програмування (таких як C++, C#, Java, Python, PHP і Go), середовища виконання (наприклад, .NET і Unity), середовища (такі як Docker і Kubernetes) і хмари (такі як Amazon Web Services, Microsoft Azure і Google Cloud Platform). [10] Рисунок 2.1. Показано логотип Visual Studio Code.

Рис. 2.1. Логотип Visual Studio Code

Visual Studio Code має мільйони активних користувачів, і не лише в Microsoft. Наприклад, багато користувачів VS Code є розробниками в Google або Facebook.

Розробникам подобається простота використання Visual Studio Code як редактора, а також його здатність перевіряти синтаксис, завершувати код, рефакторювати код, налагоджувати та передавати в репозиторій. Розробникам хмар і контейнерів подобаються можливості віддаленого керування VS Code і його явна підтримка основних хмар, Docker і Kubernetes. Розробники, які працюють у командах, люблять інтеграцію Git від VS Code. Немає необхідності встановлювати Visual Studio Code на власний комп'ютер. Є можливість переглянути vscode.dev або github.dev для стабільних збірок або insiders.vscode.dev для останніх щоденних збірок. Ви також можете відкривати файли та папки на своєму комп'ютері чи віддаленому сховищі в спрощеній версії Visual Studio Code. Це веб-редактор, який не може запускати код або мовні сервери, хоча він може запускати багато розширень і налаштувань VS Code. Для додаткових функцій вам знадобиться GitHub Codespace, який може запускати та налагоджувати код і використовувати мовний сервер, якщо ваша організація використовує план GitHub Team або GitHub Enterprise Cloud.

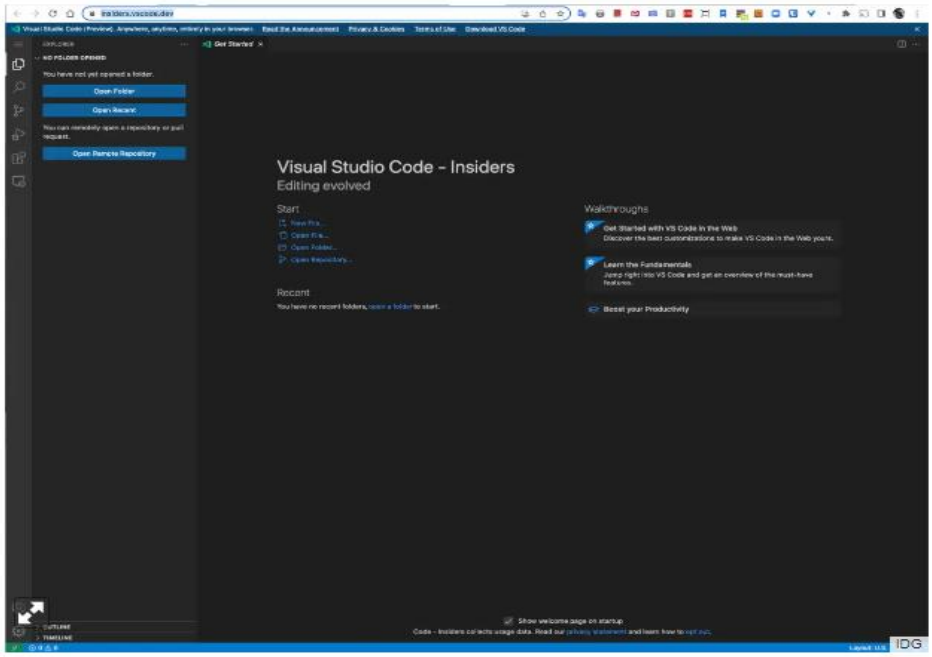


Рис. 2.2. Початок роботи з Visual Studio Code

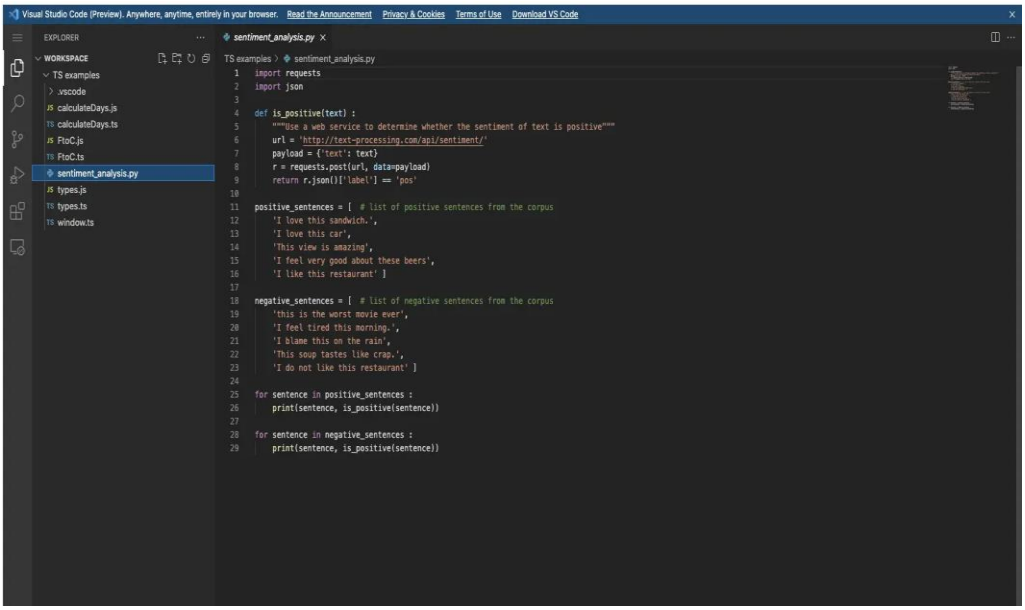


Рис. 2.3. Робота Visual Studio Code Python

Таким чином, Visual Studio Code — це швидкий безкоштовний редактор програмування, який підтримує більшість, якщо не весь життєвий цикл розробки програмного забезпечення. VS Code має десятки тисяч плагінів і підтримує сотні мов програмування.На рис.2.3.показано вікно роботи Visual Studio Code з мовою програмування Python. Це один із найкращих редакторів коду.

2.2. Аналіз іструментів розробки HTML

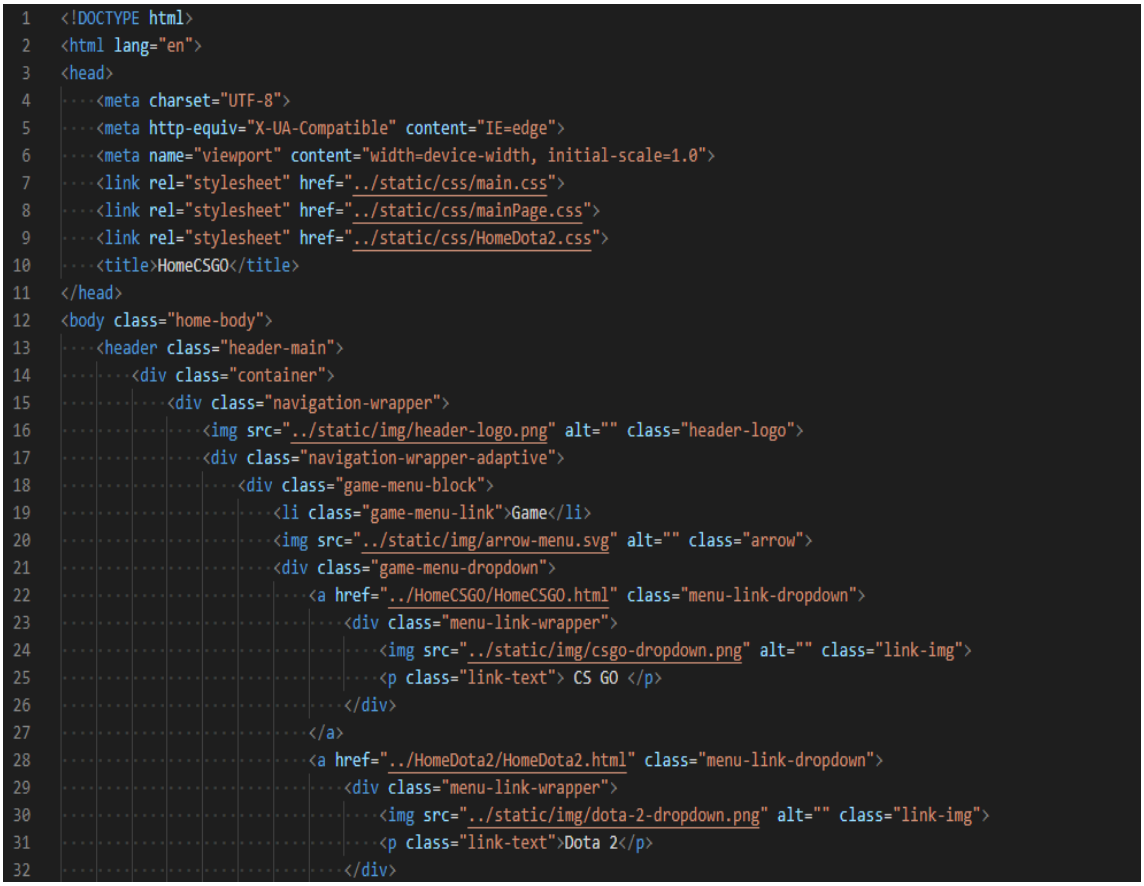
2.2.1 HTML

HTML — це стандартизована мова розмітки гіпертекстових документів, яка використовується для перегляду веб-сторінок у браузері. Веб-браузер отримує HTML-документ із сервера або відкриває його з локального диска за допомогою протоколу HTTP/HTTPS, а потім інтерпретує код в інтерфейс, який буде відображатися на екрані монітора. На рис.2.4.показано логотип мови HTML.

Рис. 2.4. Логотип HTML

Елементи HTML є будівельними блоками сторінок HTML. Використовуючи HTML, ви можете вставляти різні дизайни, зображення та інші об’єкти, наприклад інтерактивні веб-форми, на сторінки, що відображаються. HTML надає інструменти для створення заголовків, абзаців, списків, посилань, цитат та інших елементів. Елементи HTML розділені тегами, написаними за допомогою кутових дужок. Інші теги, такі як <p>, обтікають і обрамляють текст усередині себе та можуть містити інші теги як дочірні елементи. Браузери не відображають теги HTML, але використовують їх для інтерпретації вмісту сторінки.

Мова XHTML — це суворіша версія HTML, яка дотримується синтаксису XML і є розширенням мови XML у сфері гіпертекстової розмітки. HTML може вбудовувати програмний код JavaScript для керування поведінкою та вмістом веб-сторінок. Крім того, CSS, що міститься в HTML, описує зовнішній вигляд і макет сторінки.



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="../static/css/main.css">
    <link rel="stylesheet" href="../static/css/mainPage.css">
    <link rel="stylesheet" href="../static/css/HomeDota2.css">
    <title>HomeCSGO</title>
</head>
<body class="home-body">
    <header class="header-main">
        <div class="container">
            <div class="navigation-wrapper">
                <img src="../static/img/header-logo.png" alt="" class="header-logo">
                <div class="navigation-wrapper-adaptive">
                    <div class="game-menu-block">
                        <li class="game-menu-link">Game</li>
                        <img src="../static/img/arrow-menu.svg" alt="" class="arrow">
                        <div class="game-menu-dropdown">
                            <a href="../HomeCSGO/HomeCSGO.html" class="menu-link-dropdown">
                                <div class="menu-link-wrapper">
                                    <img src="../static/img/csgo-dropdown.png" alt="" class="link-img">
                                    <p class="link-text"> CS GO </p>
                                </div>
                            </a>
                            <a href="../HomeDota2/HomeDota2.html" class="menu-link-dropdown">
                                <div class="menu-link-wrapper">
                                    <img src="../static/img/dota-2-dropdown.png" alt="" class="link-img">
                                    <p class="link-text">Dota 2</p>
                                </div>
```

Рис 2.5. Приклад розмітки

2.2.2 CSS та SCSS

Творці веб-сторінок використовують CSS для визначення кольорів, шрифтів, стилів, компонування окремих блоків та інших аспектів представлення цих веб-сторінок.

Основною метою розробки CSS було поєднання опису логічної структури веб-сторінки (створеної за допомогою HTML або іншої мови розмітки) з описом зовнішнього вигляду веб-сторінки (зараз з використанням офіційної мови CSS). Такий поділ може підвищити доступність документів, забезпечити більшу гнучкість і контроль над їх поданням, а також зменшити складність і дублювання структурованого вмісту. [12]

Крім того, CSS дозволяє візуалізувати один і той самий документ у різних стилях або результатах, наприклад, на екрані, у режимі друку, читати

вголос (через спеціальний мовний браузер чи програму зчитування з екрана) або під час відображення на пристрої за допомогою шрифту Брайля.

На рис. 2.6. зображено логотип CSS.

Рис. 2.6. Логотип CSS

Нові функції CSS розробляються або призначені для робочих груп CSS, деякі як спеціальні інструменти, зацікавлені в будь-якій можливості, іноді як веб-дизайнери та розробники як розробники, деякі з них працюють у групах. Нові функції CSS створюються робочими групами CSS, які розробляють або визначають - іноді тому, що окремий браузер зацікавлений у функції, іноді тому, що веб-дизайнери та розробники запитують функцію, а іноді тому, що робоча група сама визначає вимогу. CSS постійно розвивається, і постійно з’являються нові функції. Однак головне в CSS полягає в тому, що всі наполегливо працюють над тим, щоб нічого не змінювати так, щоб старі сайти вийшли з ладу. Веб-сайт, створений у 2000 році з використанням обмеженого CSS, доступного на той час, має бути доступним у браузері сьогодні.На рис 2.7. зображено приклад CSS властивостей.

```css
.main {
  display: grid;
  grid-template-columns: repeat(4,1fr);
  column-gap: 30px;
  row-gap: 70px;
  max-width: 890px;
  margin: 30px auto 0;
}


.main__block {
  display: flex;
  justify-content: center;
  flex-direction: column;
  align-items: center;
  padding: 10px 10px 16px 10px;
  border-radius: 10px;
  gap: 8px;
  transition: 0.3s ease-in-out;
}

.main__block:hover {
    transform: scale(1.1);
    background: #eee;
}
.main__block:hover .main__card-text__subinfo {
    display: block;
}

.main__block:hover .main__photo-wrapper__image {
    animation-name: image;
    animation-duration: 1s;
    animation-delay: 1s;
}

.main__photo-wrapper {
  position: relative;
}

.main__photo-wrapper__image {
  max-width: 200px;
  max-height: 200px;
  border-radius: 50%;
}
```

Рис. 2.7. приклад CSS властивостей

Sass — це мета-мова на основі CSS, яка використовується для того, щоб зробити опис стилю документа більш чітким і впорядкованим; він потужніший порівняно з простим CSS. Sass надає CSS більш доступний і елегантний синтаксис, а також різні корисні інструменти для створення таблиць стилів, які легко підтримувати в майбутньому. [12] Отже, хоча звичайний CSS ще не підтримує змінні, міксини (блоки стилів, які можна використовувати кілька разів) та інші переваги, Sass надає синтаксис для підтримки всього цього, а не лише доповнюючи надздібності, які звичайні CSS має функцію.На рис 2.8. зображено логотип SASS препроцесора.

Потім він інтерпретує, або компілює, цей синтаксис, у звичайні CSS файли, за допомогою командного рядка або простого плагіна для веб-фреймворку.

Рис. 2.8. приклад CSS властивостей

Використання препроцесінга SASS покращує оптимізацію коду та швидкість обробки коду та завантаження веб сторінки Покращення відбувається за рахунок Sass можливостей таких як:

– **Змінні:** запис нерезервованих ключів в іменовані комірки які можливо буде використовувати в процесі написання коду

Приклад використання на рис 2.9.

```
src > styles > utils > variables.scss > ...
1   $clockSize:400px;
2   $smallRoundSize:30px;
3   $arrowRadius:5px;
4   $arrowWidth:10px;
5
```

Рис. 2.9. Використання змінних

– **Міксіни:** окремо винесені блоки властивостей, які можна перевикористати в розмітці в різних блоках, попередньо заімпортувавши їх. Можуть приймати змінні, як параметр.

Приклад використання міксінів на рис 2.10. які являються шаблонами на різні об’єкти

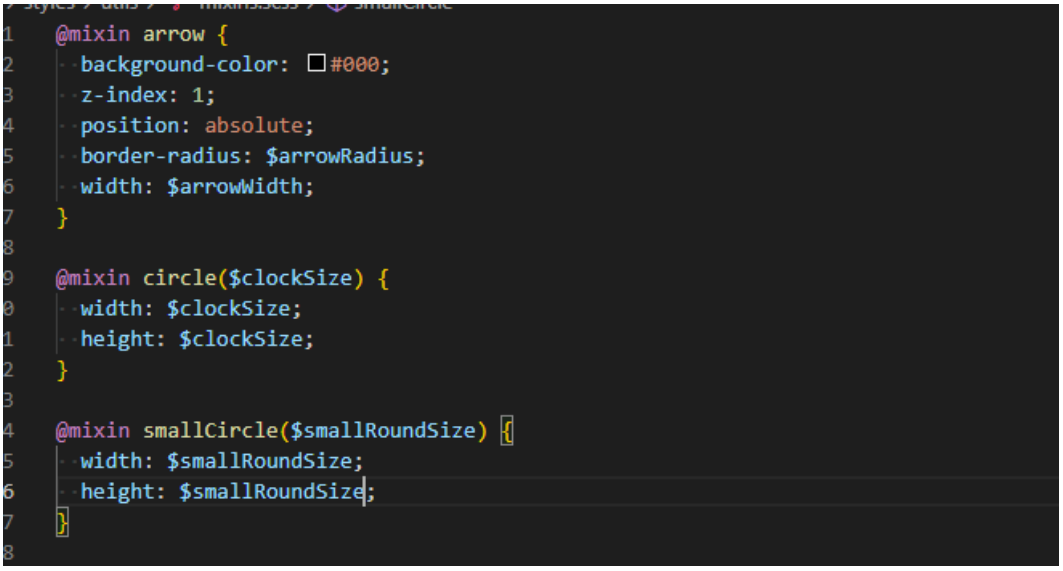


Рис. 2.10. Використання міксінів

Перший блок окрім розміру описує також колір та позицію відносно осі z стрілок зверстаного годинника. Наступні міксини працюють зі змінними-параметрами, які описують відповідно розміри годинника та розміри деталі годинника.

– **Цикли:** дозволяють застосувати певний набір функцій та властивостей для однотипних блоків коду цикли в препроцесорі є двох видів:

1. For
2. For each

На рис 2.11. показано практичне застосування циклу for each

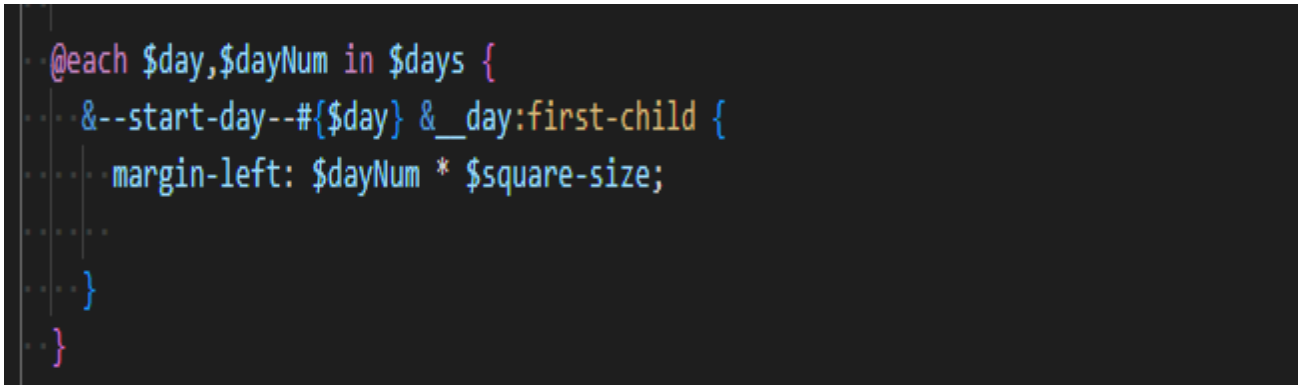


Рис. 2.11. Використання for each

В даному прикладі показано програмування віджету календар.

Цикл допомагає виставити коректний відступ між клітинками у перший ряд календаря. На рис 2.12. показано практичне застосування циклу for.

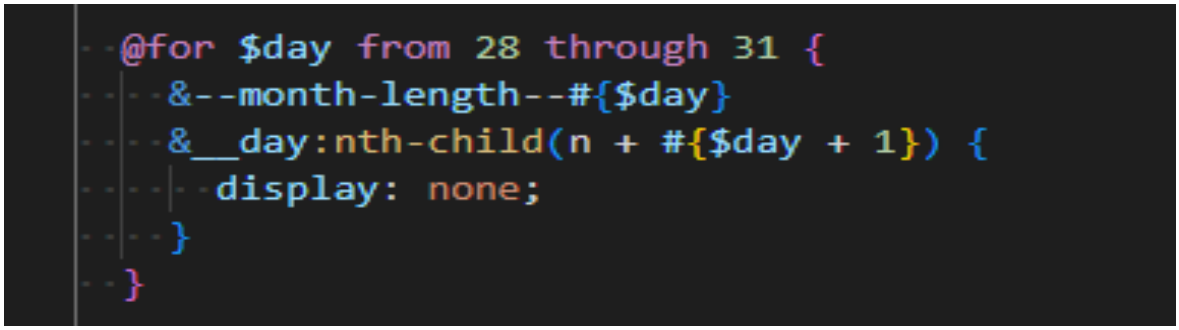


Рис. 2.12. Використання for

В даному випадку цикл програмує різницю днів в календарі,адже місяці в календарі можуть мати від 28 до 31 дня відповідно.

Вложеність стилів

Вложеність стилів допомагає вибудувати правильну ієрархію при вкладеності та стилізації блоків та уникати конфлітку неймінгів при написанні коду стилів в різних блоках застосунку.

Крім того, вкладеність блоків підвищує читабельність через застосування на класах спеціальних символів та амперсандів.

На рис.2.13. показано приклад вкладеності :

Рис. 2.13. Використання вложення

2.2.3 Javascript

JavaScript є об'єктно-орієнтованою мовою, але прототипування, що використовується в мові, [13] зумовлює відмінності в роботі з об'єктами в порівнянні з традиційними клас-орієнтованими мовами. Крім того, JavaScript має ряд властивостей, властивих функціональним мовам – функції як об'єкти першого класу, об'єкти як списки, каринг, анонімні функції, замикання– що надає мові додаткової гнучкості. [14]

Незважаючи на схожий із Сі синтаксис, JavaScript в порівнянні з мовою Сі має корінні відмінності:

- об'єкти з можливістю інтроспекції;
- функції як об'єкти першого класу;
- автоматичне наведення типів;
- автоматичне складання сміття;
- анонімні функції.

У мові відсутні такі корисні речі[14], як:

– стандартна бібліотека: зокрема, відсутня інтерфейс програмування програм по роботі з файловою системою, управління потоками введення-виведення, базових типів для бінарних даних;
– стандартні інтерфейси до веб-серверів та баз даних;
– система управління пакетами,яка б відстежувала залежності та автоматично встановлювала їх. [13]

На сьогоднішній день підтримку JavaScript забезпечують сучасні версії всіх браузерів, що найчастіше використовуються.На рис.2.14 зображено логотип Javascript. В Internet Explorer, Opera, Mozilla Firefox, Safari, Google Chrome є повна підтримка третьої редакції ECMA-262.

При цьому в Mozilla Firefox зроблено спробу підтримки четвертої редакції специфікації, а першим браузером, в якому з'явилася неповна підтримка специфікації 3.1, з'явився Internet Explorer 8[14].

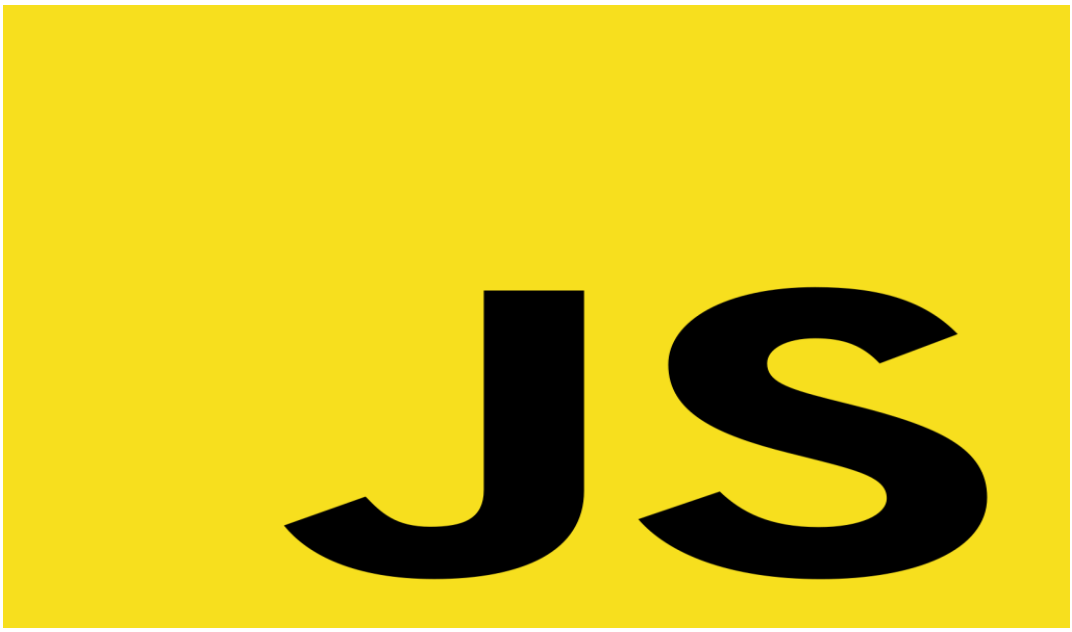


Рис 2.14. Логотип Javascript

Допущені розробниками популярних браузерів помилки у реалізації специфікації, зазвичай, незначні.[14] На листопад 2009 року об'єктна модель документа має більш обмежену підтримку.[13]

На думку творця мови, підтримка в Internet Explorer компанією Microsoft одного з існуючих та застосовуваних в інших браузерах швидких

движків JavaScript здатне призвести до появи додатків, що працюють з тривимірною графікою, написаних на JavaScript 3D-ігор, використання JavaScript у завданнях, в яких раніше застосовувалася технологія Adobe Flash.[14]

2.2.4 Reactjs

Фреймворк React.js — це фреймворк і бібліотека JavaScript із відкритим кодом, розроблена Facebook. Він використовується для швидкого й ефективного створення інтерактивних інтерфейсів користувача та веб-додатків із значно меншою кількістю коду, ніж із ванільним JavaScript. [15]

У React відбувається розробка , створюючи повторно використовувані компоненти, які можна розглядати як незалежні блоки. Ці компоненти є окремими частинами кінцевого інтерфейсу, які, будучи зібраними, утворюють весь інтерфейс користувача програми.На рис.2.12. зображено логотип ReactJS

Основна роль React у додатку полягає в тому, щоб обробляти рівень перегляду цього додатка так само, як V у шаблоні «model-view-controller» (MVC), забезпечуючи найкраще та найефективніше виконання візуалізації. Замість того, щоб мати справу з усім інтерфейсом користувача як єдиним блоком, React.js заохочує розробників розділяти ці складні інтерфейси користувача на окремі багаторазові компоненти, які утворюють будівельні блоки цілого інтерфейсу користувача. При цьому фреймворк ReactJS поєднує швидкість і ефективність JavaScript з більш ефективним методом маніпулювання DOM для швидшого відтворення веб-сторінок і створення високодинамічних і адаптивних веб-додатків.

В звичайному варіанті запит на веб-сторінку, ввівши її URL-адресу у веб-переглядач. Потім ваш веб-переглядач надсилає запит на цю веб-сторінку, яку він відображає. Якщо опрацьовується посилання на веб-сторінці, щоб перейти на іншу сторінку веб-сайту, на сервер надсилається новий запит для отримання цієї нової сторінки.

Шаблон завантаження вперед і назад між вашим браузером (клієнтом) і сервером продовжується для кожної нової сторінки чи ресурсу, до якого ви намагаєтеся отримати доступ на веб-сайті. Цей типовий підхід до завантаження веб-сайтів працює чудово, але розгляньте веб-сайт, який дуже керується даними. Попереднє завантаження повної веб-сторінки було б зайвим і створювало б погану взаємодію з користувачем.

Крім того, коли дані змінюються в традиційній програмі JavaScript, для відображення цих змін потрібно ручне маніпулювання DOM.Повинно бути визначити, які дані змінилися, і оновити DOM, щоб відобразити ці зміни, що призведе до повного перезавантаження сторінки.

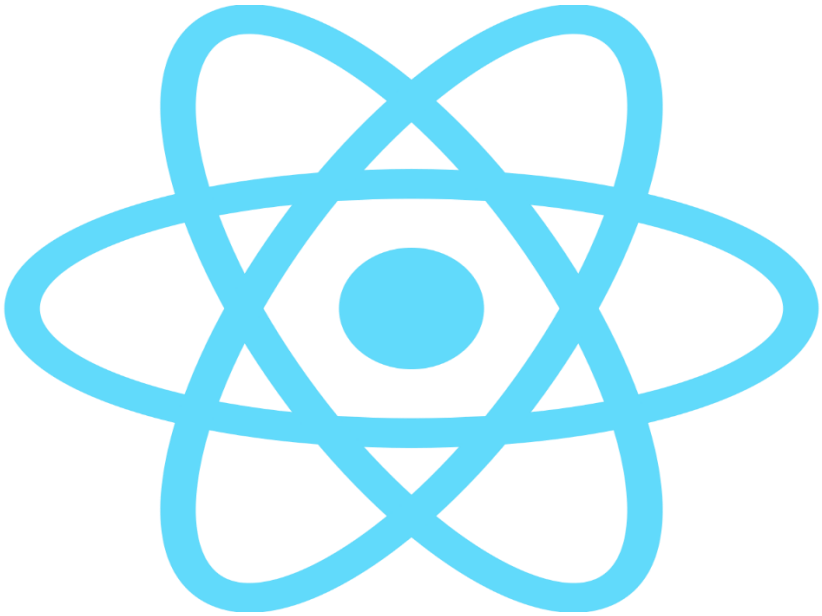


Рис. 2.12. Логотип ReactJS

React використовує інший підхід, дозволяючи вам створювати так звані односторінкові програми (SPA). Односторінкова програма завантажує лише один документ HTML за першим запитом.

Потім він використовує JavaScript для оновлення певного розділу, вмісту або основного вмісту веб-сторінки, який потрібно оновити. Цей шаблон відомий як маршрутизація на стороні клієнта, оскільки клієнту не потрібно перезавантажувати всю веб-сторінку, щоб отримати нову сторінку кожного разу, коли користувач робить новий запит. Натомість React

перехоплює запит і отримує та змінює лише ті частини, які потрібно змінити, не запускаючи повне перезавантаження сторінки. Такий підхід забезпечує кращу продуктивність і більш динамічний досвід користувача. React покладається на віртуальний DOM, який є копією справжнього DOM. Щоразу, коли стан даних змінюється, віртуальний DOM React негайно перезавантажується, щоб відобразити цю нову зміну.

Після цього React порівнює віртуальний DOM із справжнім DOM, щоб побачити, що змінилося, як показано на малюнку 2.13. Відображає об'єктну модель документа в ReactJS. Віртуальний DOM (VDOM) — це концепція програмування, де ідеалізоване або «віртуальне» представлення інтерфейсу користувача зберігається в пам'яті та синхронізується з «реальним» DOM за допомогою бібліотеки, такої як ReactDOM. Цей процес називається узгодженням.

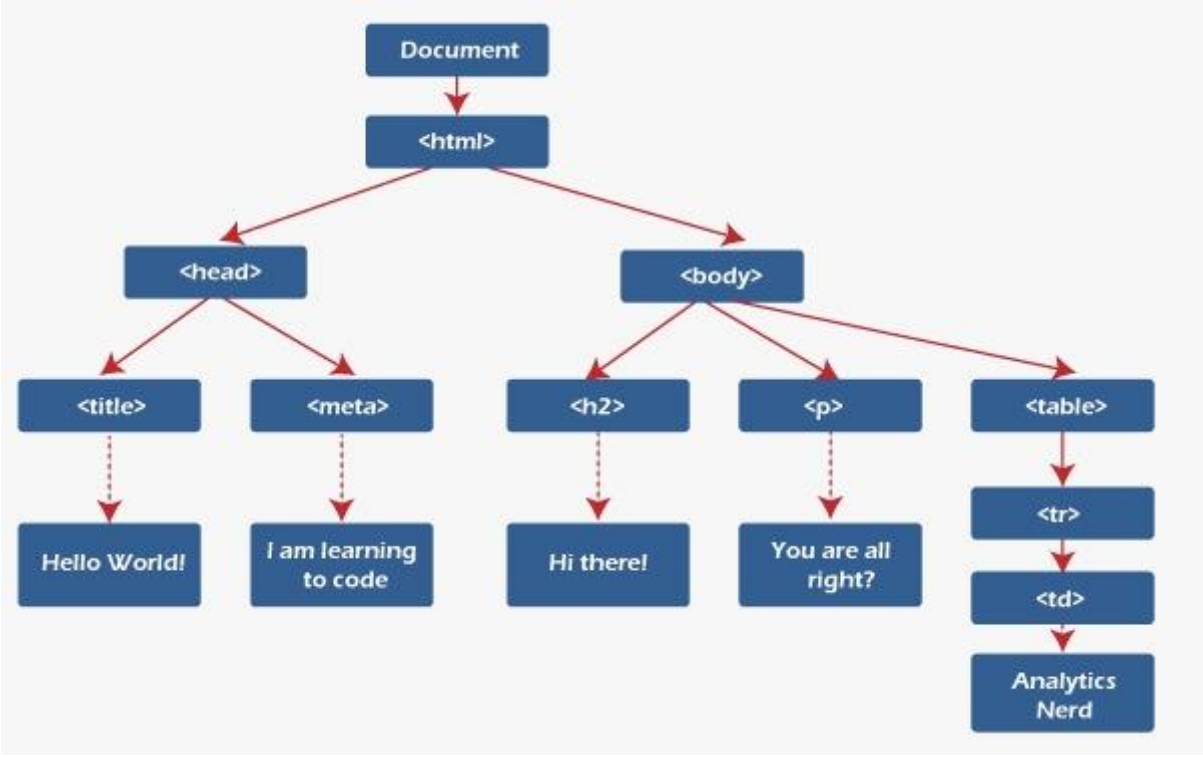


Рис. 2.13. Приклад роботи віртуальної об'єктної моделі документа в ReactJS

Такий підхід і робить API React декларативним: ви вказуєте, в якому стані повинен знаходитися інтерфейс користувача, а React домагається, щоб

DOM відповідав цьому стану. Це абстрагує маніпуляції з атрибутами, обробку подій та ручне оновлення DOM, які інакше довелося б використовувати для розробки програми.[17] На рис.2.14. зображено приклад роботи ReactJS.

```
const recognizeCommands = async () => {
    console.log("Listening for commands");
    // start model and listen for command
    model.listen(
      (result) => {
        // print result
        console.log(result.spectrogram);
      },
      { includeSpectrogram: true, probabilityThreshold: 0.9 }
    );
    // set timeout after which the stops listening
    setTimeout(() => model.stopListening(), 10e3);
  };
```

Рис. 2.14. Приклад роботи ReactJS з конструкцією speechToText

Оскільки «Віртуальний DOM» — це шаблон, а не конкретна технологія, цей термін іноді відноситься до різних понять. У світі React «віртуальний DOM» часто асоціюється з елементами React, оскільки вони є об’єктами, які представляють інтерфейс користувача. Однак React також використовує внутрішні об’єкти, які називаються волокнами, для зберігання додаткової інформації про дерево компонентів. Їх також можна вважати частиною реалізації «віртуального DOM» React.

2.2.5 TensorFlowJs

TensorFlow.JS — бібліотека з відкритим вихідним кодом, який можна використовувати для визначення, навчання та запуску моделей машинного навчання повністю в браузері, використовуючи Javascript і високорівневий API.

ML робота в браузері означає, що з точки зору користувача немає необхідності встановлювати будь-які бібліотеки або драйвера. Просто відкрийте веб-сторінку і ваша програма готова до запуску. Крім того, все готове до роботи із прискоренням на GPU. TensorFlow.js автоматично підтримує WebGL і непомітно прискорює код, коли GPU доступний. Користувачі також можуть відкрити веб-сторінку з мобільного пристрою, і в цьому випадку ваша модель може скористатися даними датчиків, наприклад, від гіроскопа або акселерометра. Нарешті всі дані залишаються на стороні клієнта, роблячи TensorFlow.js придатним для виведення з низькою затримкою, а також для додатків, що зберігають конфіденційність.

На TensorFlow.js, є можливість розробляти:.

1. Імпортувати наявну попередньо підготовлену модель.

– Якщо є існуюча модель на TensorFlow або Keras, яку ви раніше навчали в офлайн режимі, ви можете перетворити її в TensorFlow.js формат, і завантажити в браузер.

2. Повторно навчити імпортовану модель.

– Можете використовувати transfer learning для доповнення існуючої моделі, навченої в offline режимі, використовуючи невелику кількість даних, зібраних у браузері, використовуючи метод, який називається Image Retraining. Це один із способів швидкої підготовки точної моделі з використанням лише невеликої кількості даних.

3. Створити модель прямо у браузері.

– TensorFlow.js використовується для створення, навчання та запуску моделей повністю у браузері з використанням Javascript та високорівневого API.[16]

2.2.6 AssemblyAI

AssemblyAI – це компанія зі штучного інтелекту, яка створює платформу API для транскрипції та розуміння аудіоданих. Платформа

автоматично перетворює аудіо- та відеофайли та живі аудіопотоки в текст. Користувачі можуть робити більше за допомогою аудіорозвідки, як-от узагальнення, модерація вмісту, визначення тем тощо.[18]

2.3 Аналіз клієнт-серверної архітектури

Архітектура клієнт-сервер — це архітектура комп'ютерної мережі, в якій багато клієнтів (віддалених процесорів) запитують і отримують послуги від центрального сервера (хосту). Клієнтський комп'ютер забезпечує інтерфейс, який дозволяє користувачеві комп'ютера запитувати послуги з сервера та відображати результати, які повертає сервер.

Сервер чекає запити від клієнтів і потім відповідає на них. В ідеалі сервер представляє клієнту стандартизований прозорий інтерфейс, щоб клієнту не потрібно було знати подробиці системи, що надає послуги (тобто апаратне та програмне забезпечення). Клієнти зазвичай розташовані на робочих станціях або ПК, тоді як сервери розташовані в інших місцях мережі, зазвичай на більш потужних машинах.

Ця модель обчислень особливо ефективна, коли клієнт і сервер регулярно виконують різні завдання. Наприклад, під час обробки лікарняних даних клієнтський комп'ютер може запускати програму для введення інформації про пацієнта, а серверний комп'ютер запускає іншу програму для керування базою даних, яка постійно зберігає інформацію. Багато клієнтів можуть одночасно отримувати доступ до інформації сервера, тоді як клієнтські комп'ютери можуть виконувати інші завдання, наприклад надсилати електронну пошту.

Оскільки клієнтські та серверні комп'ютери вважаються окремими пристроями, модель клієнт-сервер значно відрізняється від старої моделі мейнфрейму, у якій централізований мейнфрейм виконував виконання для пов'язаних із ним «тупих» терміналів. Для всіх завдань ці термінали

спілкуються лише з центральний мейнфрейм. [17]На рис.2.13. зображено схему клієнт-серверної архітектури.

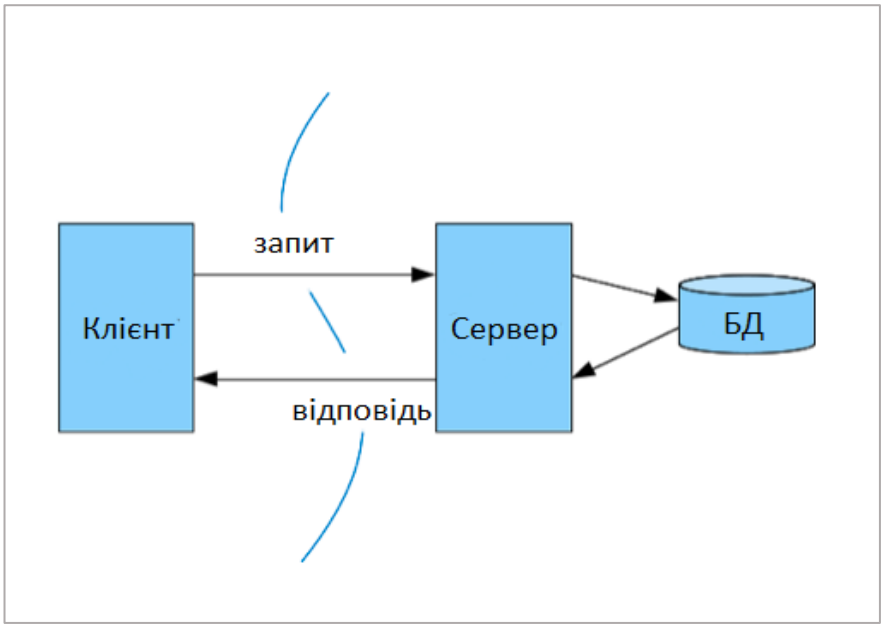


Рис. 2.13. Клієнт-серверна архітектура

Переваги клієнт-серверної архітектури:

1. Керувати легко – керувати файлами досить легко, оскільки всі вони зберігаються на одному сервері. Мережі клієнт-сервер мають найкраще керування для відстеження та пошуку записів необхідних файлів.
2. Легкий доступ – клієнти можуть входити в систему незалежно від свого місцезнаходження чи вибраної платформи, що дозволяє кожному співробітнику отримувати доступ до своєї корпоративної інформації без використання режиму терміналу чи процесора.
3. Сервери масштабовані – клієнт-серверна мережа має високу масштабованість. Щоразу, коли користувач потребує, він може додати більше ресурсів, таких як клієнти та сервери, таким чином збільшуючи розмір сервера без будь-яких перерв. Через те, що сервер є централізованим, дозвіл на доступ до мережевих ресурсів не є проблемою, тому для конфігурацій потрібно дуже мало співробітників, навіть якщо розмір збільшується.

4. Централізоване керування – мережі клієнт-сервер мають перевагу централізованого керування, оскільки вся інформація зберігається в одному місці. Оскільки адміністратор мережі має повний контроль над керуванням і адмініструванням, це особливо корисно. У результаті будь-які проблеми, які виникають у всій мережі, можна вирішувати з одного централізованого місця. У результаті оновлювати дані та ресурси стало набагато легше.
5. Безпека. Завдяки централізованій архітектурі клієнт-серверні мережі добре захищають дані. Одну резервну копію можна використати для відновлення всіх файлів у разі втрати даних, наприклад, нав'язування облікових даних, таких як ім'я користувача та паролі. Інший метод забезпечення контролю доступу полягає в нав'язуванні облікових даних, таких як ім'я користувача та пароль.

Недоліки клієнт-серверної архітектури:

1. Менш надійний – через централізовану природу клієнт-серверних мереж у разі збою або перешкод на головному сервері робота всієї мережі буде перервана. Таким чином, клієнт-серверні мережі менш надійні.
2. Потрібне регулярне технічне обслуговування – сервери працюватимуть у мережі безперервно. Це означає, що їх потрібно належним чином обслуговувати. Якщо є якісь проблеми, їх потрібно негайно усувати. Менеджер мережі повинен бути призначений для обслуговування сервера.
3. Вимагає витрат – у мережі клієнт-сервер вартість налаштування та обслуговування сервера зазвичай дуже висока, як і мережеві операції. Оскільки мережі потужні, їх придбання може бути дорогим. Таким чином, не всі користувачі зможуть ними скористатися.
4. Перевантаженість мережі. Якщо надто багато клієнтів звертаються до одного сервера, це може призвести до збоїв або уповільнення

з’єднання. Перевантажений сервер створює багато проблем при доступі до інформації.

2.4 Обгрунтування архітектури та аналіз архітектурних рішень

Архітектура програмного забезпечення є структурою програми, яка включає програмні компоненти, видимі зовні властивості цих компонентів, а також відносини між ними. Вона є основою продукту та вважається найбільш складною частиною розробки ПЗ. Щоб покращити читабельність коду, зробити його більш зрозумілим та забезпечити можливість тестування необхідно використовувати шаблони проектування додатків та архітектурних рішень.

Коли люди в галузі програмного забезпечення говорять про «архітектуру», вони мають на увазі туманно визначене поняття найважливіших аспектів внутрішнього дизайну програмної системи. Хороша архітектура важлива, інакше додавати нові можливості в майбутньому стане повільніше та дорожче.

Багато хто у світі програмного забезпечення обережно ставиться до терміну «архітектура», оскільки він часто передбачає відокремлення від програмування та нездорову дозу помпезності. Але хороша архітектура – це те, що підтримує власну еволюцію та глибоко переплетене з програмуванням.Основна дилемма пов’язана з архітектурою це те, як виглядає гарна архітектура, як команди можуть її створювати та як найкраще розвивати архітектурне мислення в наших організаціях, що займаються розробкою.

Архітектура є складною темою для клієнтів і користувачів програмних продуктів, оскільки це не те, що вони сприймають відразу. Але погана архітектура є основним фактором зростання крафту – елементів програмного

забезпечення, які заважають розробникам зрозуміти програмне забезпечення. Програмне забезпечення, яке містить багато поганого, набагато важче модифікувати, що призводить до того, що функції надходять повільніше та з більшою кількістю дефектів.

Ця ситуація суперечить звичайному досвіду.Привита концепція, що «якісне» є дорожчим. Для деяких аспектів програмного забезпечення, таких як взаємодія з користувачем, це може бути правдою. Але коли мова заходить про архітектуру та інші аспекти внутрішньої якості, цей зв’язок змінюється. Висока внутрішня якість призводить до швидшої доставки нових функцій, оскільки менше заважає.

2.5 Вибір шаблону проектування

Не менш важливою частиною розробки програмного продукту є шаблон проектування, а саме його вибір. Патерни проектування значно зменшують кількість помилок та економлять час, витрачений на розробку. Готові уніфіковані рішення полегшують комунікацію між розробниками та дозволяють посилатися на відомі конструкції.

Є три найпопулярніших патерни розробки програмних -додатків, а саме: MVP, MVVM, MVC.

1. Model View Controller (MVC).

Шаблон проектування MVC розділяє програму на три основні аспекти: Модель, Представлення та Контролер (рис.2.14).

Модель (Model) - це дані, які необхідні для відображення у преставленні. Модель являє собою набір класів, що описує бізнес-логіку (бізнес-модель і модель даних). Вона також визначає бізнес-правила для даних, а саме як їх можна змінити та як ними керувати.

Представлення (View) включає компоненти інтерфейсу користувача, такі як XML (файли з таким розширенням вдносяться до Android-додатків), HTML, WPF форма, Windows Form. Цей аспект відображає дані, отримані від

Контролера як результат. У шаблоні MVC Представлення відстежує будь-які зміни стану моделі та відображає оновлену модель. Модель і Представлення взаємодіють один з одним за допомогою Контролера.

Контролер відповідає за обробку вхідних запитів. Він обробляє дані користувача через Модель і передає результати назад до Представлення. Зазвичай він діє як посередник між Представленням і Моделлю. В архітектурі Android-додатків до нього відносять класи Activity та Fragment.

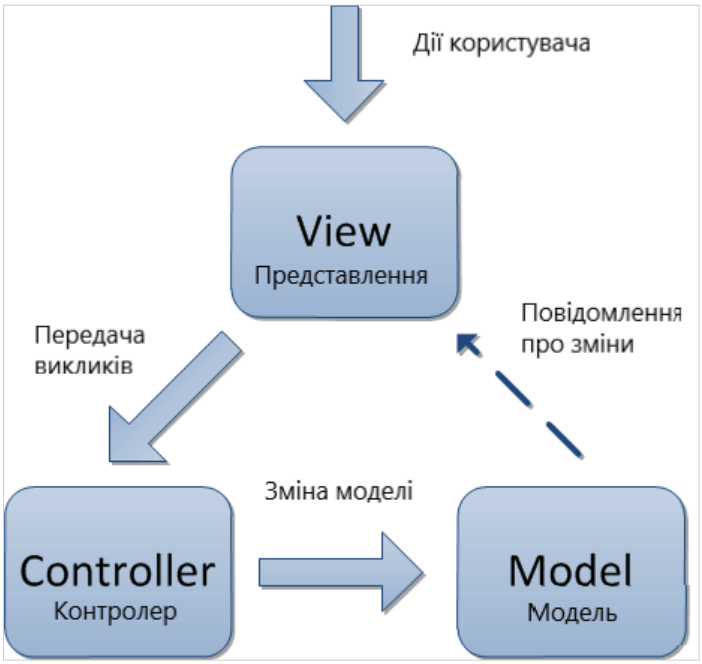


Рис. 2.14. Схема MVC шаблону

Model View Presenter (MVP)

Представник моделі (MVP) Шаблон базується на шаблоні MVC, де контролер замінено презентатором (рис. 2.15). Модель — це набір класів, які описують бізнес-логіку та дані. Представлення (View) — це компонент, який безпосередньо взаємодіє з користувачем, наприклад XML, Activity, Fragment. Він не містить жодної реалізованої логіки.

Презентатор отримує введені користувачем дані через View, потім використовує Model для обробки даних користувача та передачі результату назад у View. Делегати спілкуються з делегатами через інтерфейси. Цей інтерфейс визначено в класі Representative, і він передає необхідні дані в

клас. Activity/Fragment або будь-який інший компонент View реалізують цей інтерфейс і відображають дані в бажаний спосіб.

У шаблоні MVP можна легко розділити рівні інтерфейсу користувача, бізнес-логіки та моделі. У результаті розробники мають можливість писати чітко структурований код, який легко тестувати. Тому в роботі використовується цей шаблон.

Рис. 2.15. Схема MVP шаблону

Model View View-model (MVVM)

Шаблон MVVM підтримує двостороннє прив’язування даних між View і View-Model. Це дозволяє автоматично розповсюджувати зміни всередині стану View-Model до View. Як правило, View-Model використовує observer pattern для інформування про зміни у View-Model до Моделі (рис.2.15).

View-Model відповідає за розкриття методів, команд та інших властивостей, які допомагають підтримувати стан представлення, керувати моделлю та ініціювати події в самому представленні. View має посилання на View-Model, але View-Model не має інформації про View. Між View і View-

Model існує зв'язок багато-до-одного, що означає, що багато Views можна зіставити з однією моделлю View. Він повністю не залежить від Views.

Двостороннє зв’язування даних між View і View-Model гарантує, що моделі та властивості в моделі представлення синхронізуються з представленням. Шаблон проектування MVVM добре підходить для програм, які потребують підтримки двонаправленого зв'язування даних.

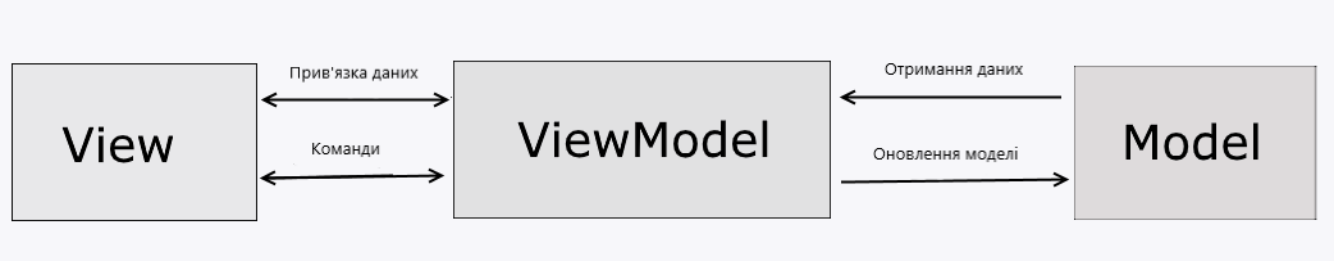


Рис. 2.16. Схема MVVM шаблону

2.6 Формування діаграми варіантів

В Уніфікованій мові моделювання діаграма варіантів використання може узагальнити деталі користувачів вашої системи (також відомих як актори) та їх взаємодію з системою. Щоб створити його, ви будете використовувати набір спеціалізованих символів і з’єднувачів. Ефективна діаграма варіантів використання може допомогти вашій команді обговорити та представити:

- Сценарії, у яких ваша система або програма взаємодіє з людьми, організаціями або зовнішніми системами
- Цілі, яких ваша система або програма допомагає цим об’єктам (відомим як актори) досягти
- Область вашої системи

Діаграма варіантів використання не містить багато деталей — наприклад, не очікуйте, що вона моделюватиме порядок виконання кроків. Натомість правильна діаграма варіантів використання відображає загальнорівневий огляд взаємозв’язків між варіантами використання, акторами та системами. Експерти рекомендують використовувати діаграми

варіантів використання, щоб доповнити більш описовий текстовий варіант використання.[20]

Діаграмма варіантів використання зображена на рис.2.17.

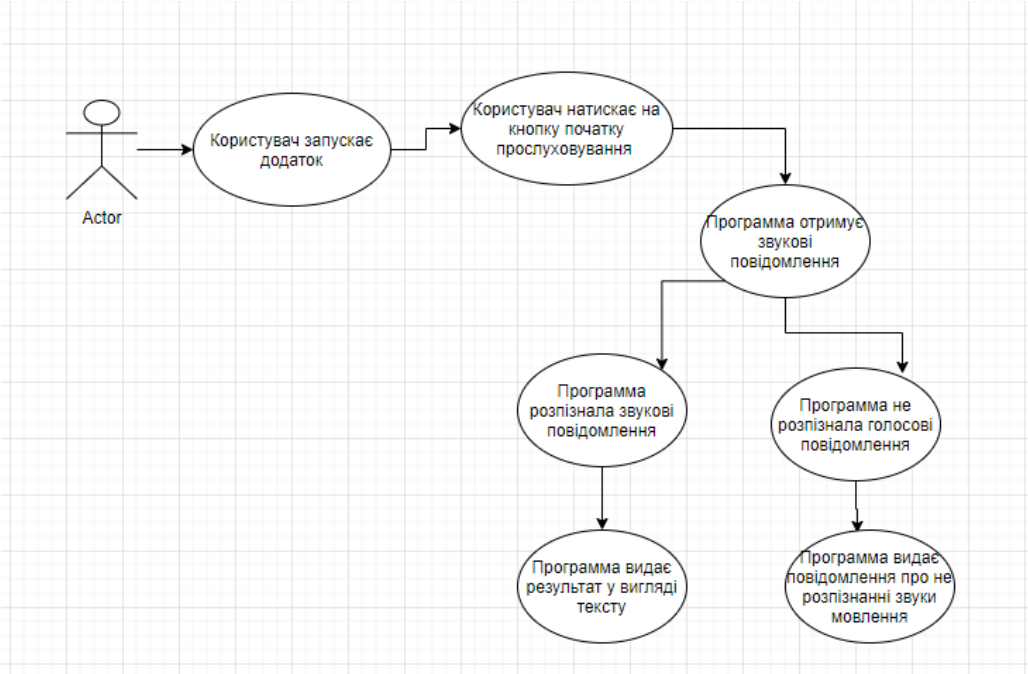


Рис. 2.17. Діаграмма варіантів використання

2.7. Формування бізнес вимог

Веб-застосунок нотаток з модулем розпізнавання голосу відповідати наступним функціональним вимогам:

– додаток повинен мати зручне відображення фраз які розпізнаються ;

– додаток повинен мати кнопку для того щоб після натиску почати розпізнавання мовлення

У результаті аналізу функціональних вимог та огляду аналогів, сформовано наступні нефункціональні вимоги:

– додаток повинен мати зручний та мінімалістичний інтерфейс;

– додаток повинен бути розроблений на мові програмування Javascript та використовувати бібліотеку ReactJS для оптимізації роботи застосунку та TensorFlowJS для розпізнавання голосу ;

– серверна частина додатка повинна бути розробленою на мові програмування JavaScript,а саме на платформі Node.js;

– додаток повинен працювати на WindowsOC;

– додаток повинен працювати на популярних комп’ютерних браузерах;

– комп’ютер повинен мати мікрофон для можливості користування додатком;

– комп’ютер повинен мати оптимальний доступ до Інтернету;

3.3. Висновки до розділу

У другому розділі було описано функціональні та нефункціональні вимоги до веб-додатку з модулем розпізнавання голосу .

Розглянуто використані шаблони проектування додатків та архітектурних рішень. Розглянуто наступні шаблони проектування: MVC, MVP та MVVC. За рахунок значної кількості переваг було обрано та використано шаблон проектування MVP.

Веб-додаток було розроблено з використанням сучасного стеку технологій, який використовується при створенні клієнт-серверних веб додатків. До нього відносяться HTML, CSS/SASS, Javascript, ReactJS, TensorflowJS AssemblyAI.

За допомогою розлянутих у другому розділі шаблонів проектування, архітектурних рішень, технологій та було розроблено стабільний веб додаток, що легкий та корисний у використанні.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНОЇ ЧАСТИНИ ВЕБ-ДОДАТКУ

3.1 Розробка ВЕБ-додатку

Розробка додатку починається з роботи з терміналом та командним рядком в середовищі розробки Visual Studio Code. Для відкриття терміналу потрібно зайти в середовище та натиснути комбінацію CTRL + `. На рис.3.1. зображено термінал та команда, яку буде виконано на старті для завантаження оболонки ReactJS за замовчуванням для подальшої розробки додатку.

Рис 3.1. Початок розробки

На рис 3.2. зображено процес установки модулів ReactJS

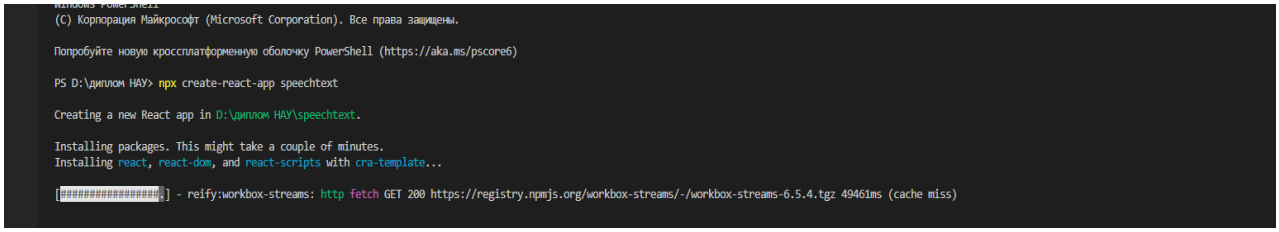


Рис 3.2.Установка модулів

Для запуску проекту потрібно виконати команду npm start.На рис 3.3. зображено стартове вікно додатку.Прототип додатку запущений на локальному сервері, який має порт localhost 3000.

На рис 3.3. зображено установку модуля TensorFlowJS для розпізнавання голосу:

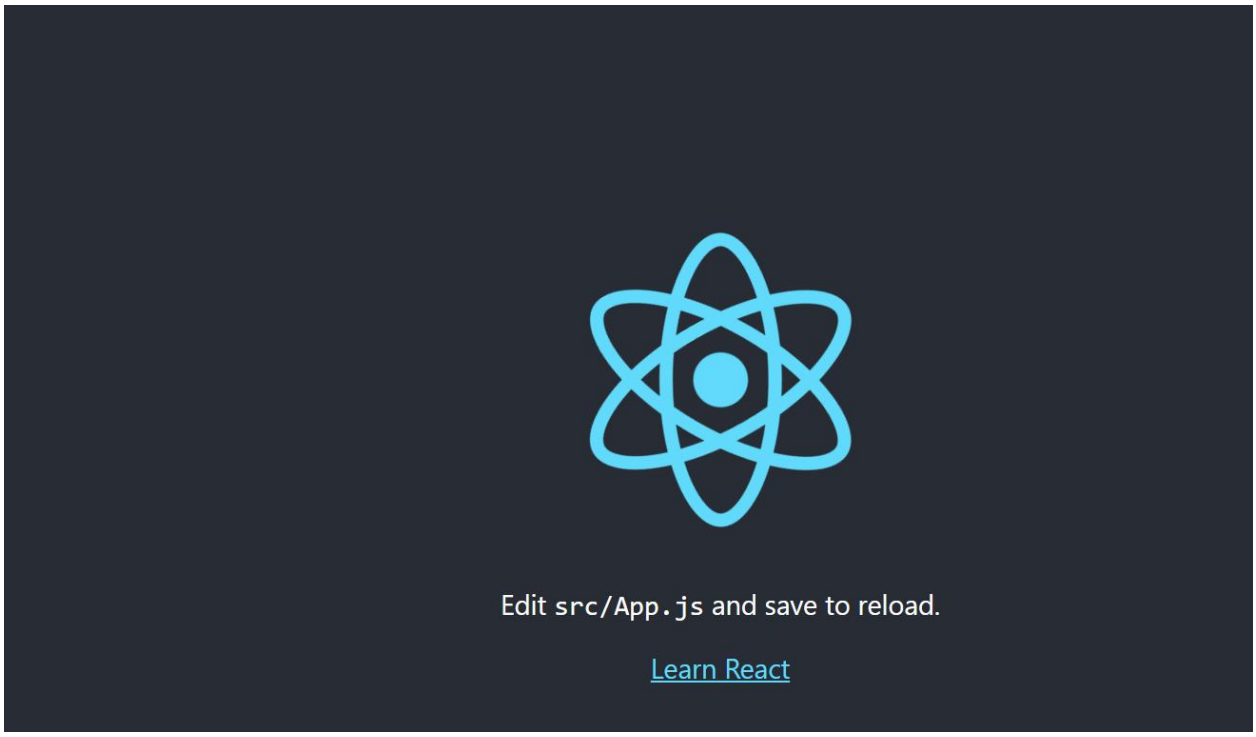


Рис 3.3.Початкове вікно додатку

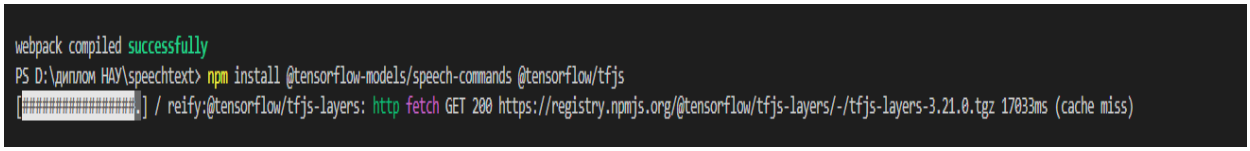


Рис 3.4.Установка модулів

Для реалізації концепції розпізнавання голосових сигналів у фреймворку ReactJs існує поняття хуки станів та хуки ефектів В кейсі додатку було використано три хуки стану які опрацьовують поведінку додатку при роботі На рис 3.5. зображено синтаксис хуків які були використані в програмі.

```
const [model,setModel] = useState(null);
const [action,setAction] = useState(null);
const [label,setLabels] = useState(null);
```

Рис 3.5.Синтаксис хуків

Хук model завантажує початкову модель :набір деяких даних, які додаток може розпізнати, коли користувач спробує говорити в мікрофон.

При бажанні ці дані можна міняти на свій розсуд. Початкова модель – завантажується у вигляді об’єктоподібного масиву, який має також команди, що реагують на нерозпізнані слова та шум на фоні при розмові.

На рис 3.5. зображено первинний набір даних у вигляді масиву.

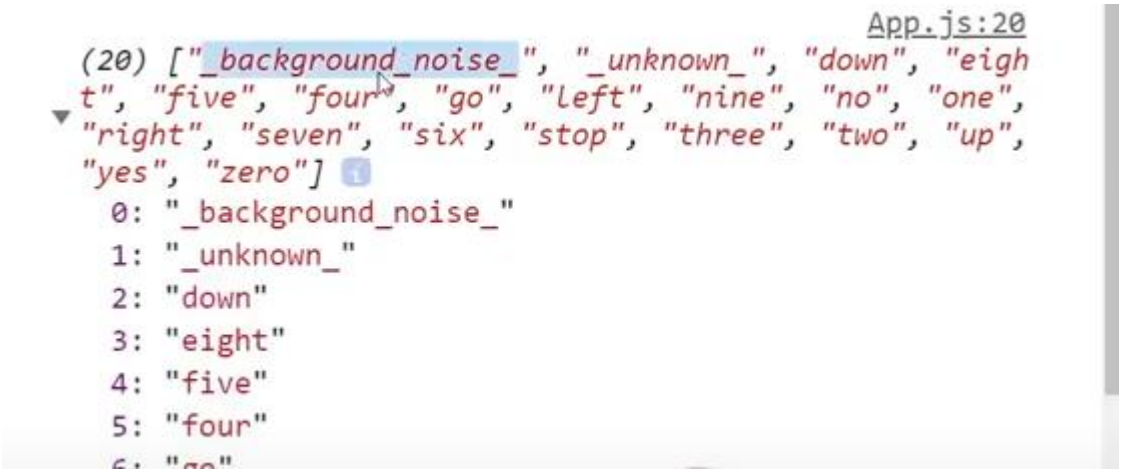


Рис 3.6.масив набору даних

При початку роботи слова, які потрапляють в програму представлені не у вигляді слів у людському розумінні, а у вигляді цифрової спектрограмми, яка потребує функціональної обробки для подальшої роботи та, наприклад виводу інформації на екран. На рис. 3.7. показано вхідні дані у вигляді числового масиву спектрограми.

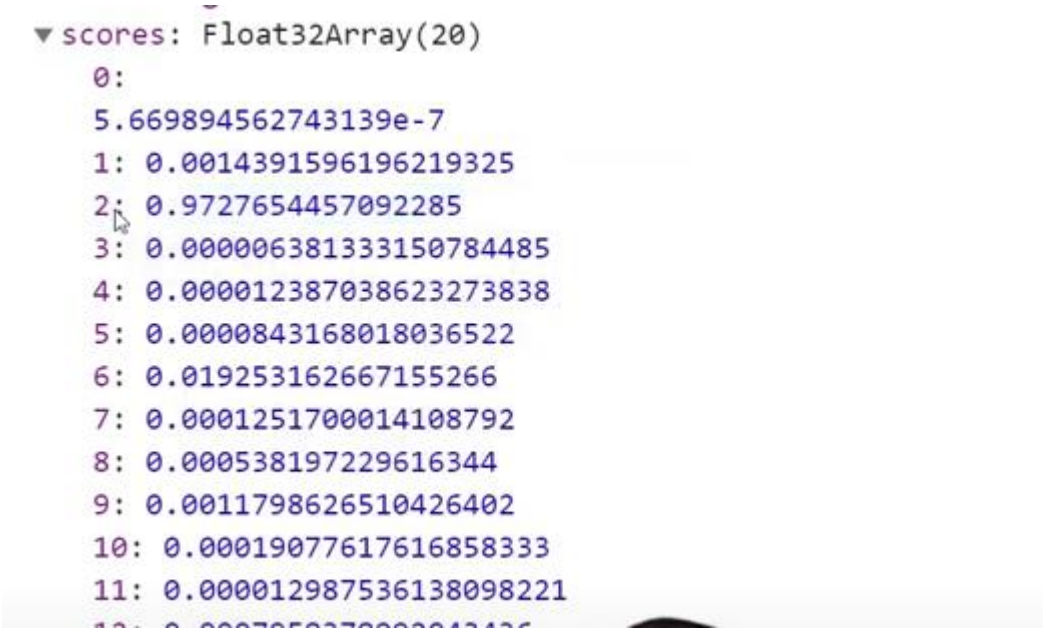


Рис 3.7.масив-спектрограмма

Для того щоб підвищити вірогідність того, що слова які проговорить користувач, будуть опрацьовані, була написана функція, яка вибирає з масиву вловлених чисел спектрограмми найбільше число, що має на увазі сигнал вищої якості. На рис 3.8. зображено функції обробки.

```
// 3. Listen for Actions
function argMax(arr){
  return arr.map((x,i)=>[x,i]).reduce((r,a)=>(a[0] > r[0] ? a:r))[1];
}
```

Рис 3.8.функція обробки

3.2 Тестування ВЕБ-додатку

3.2.1 Статичне тестування

Статичне тестування (static testing) — напрямок тестування, під час якого програмній код не буде запускатися. В цьому випадку тестування може бути і ручним і автоматизованим.

Статичне тестування починається з перших етапів циклу розробки ПЗ і є частиною процесу верифікації. Для цього типу тестування в окремих випадках не потрібен комп'ютер. Наприклад, тестування вимог, програмного коду та документації проекту не потребує запуску програмного коду та самої програми. Статичне тестування може бути автоматизовано, як і динамічне. Наприклад, для перевірки синтаксису програмного коду можна використовувати різні автоматичні засоби.

Такий тип тестування дуже важливий у розробці, бо він не потребує готового програмного продукту, і дає змогу початі тестування з єтапу написання вимог. Якщо спеціаліст ретельно підійде до цього тестування, то можна уникнути багатьох можливих помилок у логіці роботи програми, ще до її створення, і це збереже багато часу на їх виправлення, коли програма вже буде готова. У більшості випадків, такий тип контролю якості передбачає застосування різного роду перевірок і спостережень, базуючись на власному досвіді, пошук порушень стандартів програмування і т. д.

Приблизний тест-план для статичного тестування виглядає так:

— вимоги до проекту;

— методи та типи тестування, які будуть застосовуватися;

— технічний аналіз проекту та документації;

— список інструментів, необхідних для розробки;

— обов'язки та ролі учасників проекту;

— розклад і дедлайни кожного етапу розробки.

3.2.2 Динамічне тестування

Динамічне тестування (dynamic testing) – напрямок тестування, який навідміну від статичного, передбачає запуск програмного коду, завдяки чому аналізується поведінка програми під час її роботи.

Для проведення динамічного тестування програмний код повинен бути написаний, скомпільований та запущений. Також при динамічному тестуванні може досліджуватися поведінка зовнішніх параметрів роботи програми, таких як завантаження процесора, використання додатком пам'яті, час відгуку програми на дії користувача і т.д. Також динамічне тестування включає в себе різні підвиди, кожен з яких залежить від:

1) доступу до коду:

- метод білого ящика (white-box testing) — у тестувальника є доступ до коду;

- метод чорного ящика (black-box testing) — у тестувальника нема доступу до коду;

- метод сірого ящика (gray-box testing) — є доступ лише до частини коду.

Переваги та недоліки цих методів наведені в таблиці 3.1.

2) рівня тестування (модульне, інтеграційне, системне, приймальне тестування).

Таблиця 3.1 — Переваги та недоліки методів чорного, сірого, білого ящиків

Метод	**Переваги**	**Недоліки**
Білого ящика	— Показує приховані проблеми. — Полегшує діагностику проблеми — Допускає тестування на ранніх стадіях.	— Необхідний кваліфікований персонал — Складно імітувати роботу користувача.
Чорного ящика	— Нема необхідності знань з програмування. — Тест-кейси виконуються з точки зору кінцевого користувача. — Тест-кейси можна розробляти одразу, як тільки з'явилися вимоги.	— Можлива надлишковість деяких тест-кейсів. — Складно розробляти тест-кейси для незнайомого, «концептуального» додатку. — Складніше прогнозувати об'єм робот.

Сірого ящика	Об'єднує в собі переваги і недоліки методів білого и чорного ящика.

3.3.3 Рівні та види функціонального тестування

Тестування проводиться протягом усього життєвого циклу розробки програмного забезпечення на різни рівнях, етапах та подальшого його супроводу та технічної підтримки.

Рівень тестування залеже від того, над чим проводятся тести: над окремим модулем програми, групою модулів або цілої системи. Тестування на всіх рівнях необхідне для успішної реалізації продукту, та забезпечення якості на етапі здачі продукту замовнику.

Функціональне тестування це основний вид тестування, направлений на перевірку відповідності програми функціональним вимогам. Також системне тестування підтверджує, чи володіє продукт усім функціоналом, який вимагає замовник.

3.3. Висновки до розділу

В даному розділі було описано алгоритм роботи та розроблено мобільний веб –додаток з модулем розпізнавання голосу, серверну частину додатку та тестування веб додатку.

Для розробки користувацького інтерфейсу було використано SAAS препроцесінг для який надав змогу зробити гнучку стилізацію користувацього інтерфейсу та оптимізувати стилі з точки зору їх написання.Препроцесор був інтегрований всередині фреймворку ReactJS який в свою чергу завдяки віртуальному DOM покращує оптимізацію додатку в усіх модулях коду.

Отримання даних додатком здійснюється за допомогою REST API серверної частини. Це дало можливість реалізації моделі клієнт-серверного додатку, гнучкості та незалежності системи від зовнішніх систем.

Модуль розпізнавання голосу був розроблений за допомогою бібліотеки TensorFlowJS та зовнішнього API під назвою Assembly AI які дають можливості обробки голосових сигналів та розширює ширину точність та швидкість обробки звуку

Тестування додатку за допомогою Unit- та UI-тестів та ручними тестами дало змогу виявити помилки в коді та перевіряти правильність роботи додатку під час впровадження нового функціоналу та зробити відповідні висновки для оптимізації та покращення роботи коду та програми.

ВИСНОВКИ

В епоху тотального розвитку цифрових технологій не залигаються без уваги додатки, які покращують людське життя та економлять людський час та енергію та допомагають контролювати кожну дрібницю за людської потреби.

До таких розробок можна віднести додатки типу нотатників, адже вони допомагають людям тримати під контролем всі справи, які вони запланували, а можливість голосового вводу додає зручності та комфорту де б ви не були і не знаходились.

Під час аналізу та огляду додатків-аналогів зроблено висновки про те, що нотатники мають великий спектр додаткових можливостей та високу точність та сприйняття звукових сигналів, а до спектру можливостей можна віднести конвертацію субтитрів з відео або аудіо або відео цілком.Також в аналогах по бажанню присутній переклад та адаптація багатьма мовами Кількість мов залежить від додатку.

Проаналізовано сучасні додатки, які мають можливості конвертувати звукові сигнали в текст.Було розглянуто основні теоритечні концепції обробки звукових сигналів,завдяки яким відбувається конвертація голосу в текст а також було розглянуто додаткові можливості аналогів такі як: транскрибуція та переклад тексту на різні мови.

При аналізі було виявлено переваги, недоліки та особливості кожного сервісу. Це дало змогу створити перелік вимог до функціоналу, який повинен бути доступним в додатку для забезпечення конкурентоспроможності.Було відмічено те, що для точнішого та ширшого конвертування голосу потрібні ширші можливості API, якщо додаток використовує API технології.Розширення можливостей, скоріше всього, не являється безкоштовим.

У випадку не використання API та розробки аналогічної концепції додаток потребуватиме великих технічних ресурсів для тренування

нейромережі, яка в свою чергу являється основним стовпом для розпізнавання та обробки мовлення та звукових сигналів. Тренування займає багато часу в залежності від об’єму та ширини отриманої інформації, вимог до точності опрацювання отриманих даних, а також швидкість обробки інформації та кількість функцій в додатку.

Проаналізовано наукову та методичну літературу. Це допомогло поглибити знання з проектування та розробки веб-додатків, мови розмітки веб-сторінок HTML, каскадних таблиць стилів (CSS) та обробку цих таблиць за допомогою препроцесора SASS, мови програмування Javascript, та прогресивного веб-фреймворку ReactJS та набуття та поглиблення знань з технологій штучного інтелекту а саме розпізнавання та обробка звукових сигналів за допомогою модуля TensorFlowJS та API сервісу AssemblyAI.

Під час написання кваліфікаційної роботи досліджено сучасні шаблони проектування, архітектурні рішення написання додатків, шаблони проектування додатків, та стек технологій сучасних веб-додатків.

Для розробки мобільного додатка використано шаблон проектування MVP та здійснено порівняння з іншими шаблонами проектування веб-додатків. Використання даних шаблонів та архітектурних рішень дало можливість написати легко тестований та чітко структуризований код.

Дослідження сучасного стеку технологій дало можливість визначитись з переліком технологій, необхідних для розробки веб-додатку. До цього переліку відносяться:HTML, CSS, SASS, ReactJS, Javascript та TensorFlowJS. Використання даних технологій допомогло розробити легко тестований та легко розширюваний веб-додаток.

Протестовано веб-додаток. Усі знайдені в процесі розробки помилки виправлено. Після завершення процесу розробки додаток успішно пройшов усі тести на коректну працездатність основного функціоналу.

Всі поставленні завдання виконано. Результатом виконання кваліфікаційної роботи є веб-додаток нотаток з модулем розпізнавання голосу. Даний додаток відповідає описаним та загальним вимогам сучасних веб додатків, повністю протестований.

Додаток має перспективу дослідженння з точки зору технологій штучного інтелекту та має перспективу поширюватися на ринку IT-технологій за рахунок зручності, доступності та важливості. Даний додаток є MVP, що означає його майбутнє вдосконалення новими функціями для повноцінного виходу на ринок.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Історія розпізнавання голосу: [Електроний ресурс]:[Веб-сайт].-[Електронні дані].-Режим доступу: https://summalinguae.com/language-technology/speech-recognition-software-history-future/
2) Історія розпізнавання голосу та принцип роботи: [Електроний ресурс]:[Веб-сайт].- [Електронні дані].-Режим доступу: https://www.techtarget.com/searchcustomerexperience/definition/voice-recognition-speaker-recognition
3) Розпізнавання голосу Як це працює : [Електроний ресурс]:[Веб-сайт].-[Електронні дані].-Режим доступу: https://bezopasnik.info/%D1%80%D0%B0%D1%81%D0%BF%D0%BE%D0%B7%D0%BD%D0%B0%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5-%D1%80%D0%B5%D1%87%D0%B8-%D0%BA%D0%B0%D0%BA-%D1%8D%D1%82%D0%BE-%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%D0%B5%D1%82/
4) Розпізнавання голосу та перетворення в текст Як це працює : [Електроний ресурс]:[Веб-сайт].- [Електронні дані].-Режим доступу: https://www.techslang.com/how-does-speech-to-text-software-work/
5) Приклад голосового помічника : [Електроний ресурс]:[Веб-сайт].-[Електронні дані].-Режим доступу: https://www.unisender.com/ru/blog/idei/servisy-dlya-perevoda-golosa-v-tekst/#add-comment
6) Алгоритм розпізнавання голосу : [Електроний ресурс]:[Веб-сайт].-[Електронні дані].-Режим доступу: https://iopscience.iop.org/article/10.1088/1742-6596/1366/1/012091
7) Алгоритм розпізнавання голосу : [Електроний ресурс]:[Веб-сайт].-[Електронні дані].-Режим доступу: https://www.sciencedirect.com/topics/medicine-and-dentistry/hidden-markov-model

8) Алгоритм розпізнавання голосу : [Електроний ресурс]:[Веб-сайт].- [Електронні дані].-Режим доступу: https://mi.eng.cam.ac.uk/~mjfg/mjfg_NOW.pdf
9) Приклад голосового помічника : [Електроний ресурс]:[Веб-сайт].- [Електронні дані].-Режим доступу: https://www.unisender.com/ru/blog/idei/servisy-dlya-perevoda-golosa-v-tekst/#add-comment
10) Розпізнавання голосу та перетворення в текст Як це працює : [Електроний ресурс]:[Веб-сайт].- [Електронні дані].-Режим доступу: https://www.techslang.com/how-does-speech-to-text-software-work/
11) Середовище розробки Visual Studio Code : Електроний ресурс]:[Веб-сайт].- [Електронні дані].-Режим доступу: https://www.infoworld.com/article/3666488/what-is-visual-studio-code-microsofts-extensible-code-editor.html
12) CSS/Sass : Електроний ресурс]:[Веб-сайт].- [Електронні дані].-Режим доступу: https://frontender.info/why-sass/
13) Флеваран, Девід. JavaScrirt: кишеньковий довідник, 3-тє вид. - М., 2013. - 320 с. - ISBN 978-5-8459-1830-7
14) Javascript : Електроний ресурс]:[Книга в електронному форматі].- [Електронні дані].-Режим доступу: https://github.com/getify/You-Dont-Know-JS
15) ReactJs : Електроний ресурс]:[Веб-сайт].- [Електронні дані].-Режим доступу: https://blog.hubspot.com/website/react-js
16) TensorFlowJs : Електроний ресурс]:[Веб-сайт].- [Електронні дані].- Режим доступу: https://neurohive.io/ru/tutorial/obzor-tensorflow-js-mashinnoe-obuchenie-na-javascript/
17) ReactJs : Електроний ресурс]:[Веб-сайт].- [Електронні дані].-Режим доступу: https://artjoker.ua/ru/blog/chto-mozhno-sdelat-na-react-js/
18) AssemblyAI : Електроний ресурс]:[Веб-сайт].- [Електронні дані].-Режим доступу: https://www.assemblyai.com/blog/react-text-to-speech-simplified/

19) Клієнт-сервер архітектура: Електроний ресурс]:[Веб-сайт].-[Електронні дані].-Режим доступу: https://www.interviewbit.com/blog/client-server-architecture/
20) Діаграмма варіантів використання: Електроний ресурс]:[Веб-сайт].-[Електронні дані].-Режим доступу: https://www.lucidchart.com/pages/uml-use-case-diagram

ДОДАТКИ