МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп’ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

БАЩІЙ Василь Андрійович

# Алгоритм інтеграції систем розпізнавання мовлення в середовище для написання програмного коду / Algorithm for Integration of Speech Recognition Systems into the Environment for Writing Program Code

спеціальність: 122 - Комп'ютерні науки
освітньо-професійна програма - Комп’ютерні науки

Кваліфікаційна робота

Виконав студент групи
КНм-21
В.А. Бащій
_____

Науковий керівник:
д.т.н., доцент М.П. Комар
_____

Кваліфікаційну роботу
допущено до захисту:
«___» _____ 20___ р.
Завідувач кафедри
_____ М.П. Комар

ТЕРНОПІЛЬ - 2021

# ЗМІСТ

# ВСТУП

**Актуальність теми.** Уже достатньо багато часу проводяться пошуки і розробка ідеальної системи розпізнавання мовлення. На превеликий жаль розпізнавання українського мовлення не є так поширено, поки величезного прориву досягає розпізнавання англійського мовлення. Серед популярних і найбільш точних програм можна виділити старання Dragon speech recognition software, Text to Speech – IBM Watson, сервіс Azure під назвою Speech to text та багато інших, наприклад системи Apple – Siri, Amazon – Alexa та Google Assistnt. Щодо нашої рідної мови, можна сказати шо найбільшого успіху досягло програмне забезпечення Cybermova. Проект не зупиняється в розвитку і з успіхом продовжує удосконалювати свій сервіс у кращу сторону.

Не знаю чи залишились ще люди, які не знають про вклад перелічених програм вище. Кожного дня вони спрощують не тільки наші побутові потреби, а й проникли в освіту, медицину, розробку програмного забезпечення і звісно що зараз бізнес послуг не може обійтися без такого корисного і неймовірного інструменту в своїх руках.

Для наочного прикладу перечисллю деякі з поширених і популярних методів використання даної технології:

1) можливість керувати побутовою технікою. Це стосується не тільки таких гаджетів як смартфон, ноутбук, персональний комп’ютер або планшет. Уже не перший рік виробники кухонних приладів та телевізорів оснащують інтеграцією з голосовими асистентами. Можна сказати більше, значення «розумного будинку» також не стоїть на місці. Керувати ним можна не тільки через дистанційний пульт чи ваш смартфон під’єднаний до домашньої мережі, а й голосовими командами, які зможуть відкрити вам штори, включити любимого виконавця в музичному центрі і ще багато інших і цікавих команд лежать в вашому розпорядженні.

2) автовідповідачі мобільних служб. Голосові асистенти банку допоможуть вам дізнатися інформацію яка вас цікавить, потрібно лише дати запит і асистент розпізнає ваш запит і дасть вам результат в голосовому режимі. Не тільки банки можуть похвалитися такими технологіями, а й багато різних бізнесів, які займаються сферою послуг. Наприклад змінити тарифний план SIM картки або перевірити баланс вашого рахунку.

3) Програмні застосунки для розуміння мови людини. Це актуально для творчих людей, які займаються повсякденним записом своїй ідей для написання книги, та й безпосередньо самого процесу написання книги. Ще прикладом може бути детальна стенограма різних заходів. Для прикладу можна взяти роботу журналіста на конференціях. В результаті роботи можна отримати повний перелік доповідей без ручного нотування. Звісно що потрібно форматувати і коригувати цей текст, але це вже краще ніж писати все з чистого аркуша паперу [1].

Дана технологія має багато прихильників, та й компанії широко рекламують перелік можливостей, які підкорили багатьох людей своєю простотою в використанні і великою продуктивністю. Технологія досягла активного застосування в смартфонах, планшетах та інших засобах, які розуміють мову клієнта і можуть видавати результат запиту голосом. Думаю не в новинку просунутим користувачам користуватися голосовими запитами до пошукових систем замість ручного вводу. Звісно що прогресу немає меж і перетворення голосу в текст і навпаки тепер виконується в великих об’ємах. Використання допоміжного програмного забезпечення, розширення для браузерів, а також web-сервісів дозволяє максимально звільнити руки і менше псувати зір годинами дивлячись в екран. Люди які часто займаються набором текстів, такі як: копірайтери, письменники, лікарі та юристи високо цінять таку технологію в їх сфері [2].

Проблема програмного забезпечення в розпізнаванні голосу для написання коду закладається в застарілих або дорогих методах. Можна привести приклад на основі програми VoiceCode. Вона використовує Dragon Speech Recognition як модуль для розпізнавання, а решта – це оболонка для маніпулюванням вихідними даними. Безсумнівно, система Dragon Speech Recognition дуже розвинута в своїй стихії, вона набралась досвіду за достатньо великий проміжок часу, але ціна в розмірі двісті доларів за Home версію є достатньо великою як на мене.

Розроблюване програмне забезпечення буде актуальне для людей, яким діагностували синдром зап’ястного каналу. Це можна описати як біль, оніміння або поколювання в фалангах пальців рук та кисті. В детальному описі аналогів програм для написання коду голосом будуть представлені приклади програмістів, які отримали такий діагноз через свою постійну роботу за клавіатурою.

**Мета і завдання дослідження.** Метою кваліфікаційної роботи є підвищення ефективності написання програмного коду за допомогою алгоритму, який дозволить інтегрувати новітні системи розпізнавання мовлення в середовище програмування.

Для досягнення поставленої мети у роботі сформульовано наступні задачі:

- провести аналіз предметної області;
- провести порівняльний аналіз відомих рішень;
- зробити постановку задач дослідження;
- описати концептуальні основи інтеграції системи розпізнавання мовлення в середовище для написання програмного коду;
- дослідити та вибрати систему розпізнавання мовлення;
- дослідити алгоритм інтеграції системи розпізнавання мовлення Web Speech API в середовище для написання програмного коду;
- провести розробку архітектури інтегрованої системи;

– провести реалізацію алгоритму інтеграції системи розпізнавання мовлення в середовище для написання програмного коду;

– провести оцінку експериментальних досліджень та ефективності запропонованого алгоритму.

**Об’єкт дослідження** – процеси розпізнавання мовлення в автоматизованих системах розпізнавання мовлення.

**Предмет дослідження** – методи та засоби інтеграції автоматизованої систем розпізнавання мовлення Web Speech API.

**Методи дослідження.** Для вирішення поставлених задач в роботі використовувалися наступні методи глибоких нейронних мереж, автоматизованих розпізнавачів мовлення та методи побудови розширень для середовища програмування.

**Наукова новизна одержаних результатів.** Вдосконалено алгоритм інтеграції системи розпізнавання мовлення в середовище для написання програмного коду шляхом побудови розширення, який на відміну від інших дозволяє інтегрувати новітні автоматизовані системи розпізнавання мовлення не порушуючи існуючу архітектуру системи.

**Практичне значення отриманих результатів.** Розроблено архітектуру системи автоматичного розпізнавання мовлення в середовищі програмування. Дане розширення для середовища програмування дає можливість створювати файли формату *.html, *.js (в майбутньому можливо адаптувати систему для створення інших популярних форматів для програмування) та написання програмного коду за допомогою голосових зарезервованих команд. Також можливе інтегрування інших новітніх систем автоматичного розпізнавання мовлення без зміни архітектури системи.

**Публікації та апробація.** Результати кваліфікаційної роботи апробовані та опубліковані у матеріалах:

– III Міжнародної науково-практичної конференції «Topical issues of modern science, society and education», 3-5 жовтня 2021 р., Харків, Україна.

– IV Міжнародної науково-практичної конференції «Topical issues of modern science, society and education», 1-3 листопада 2021 р., Харків, Україна.

Кваліфікаційна робота складається із вступу, трьох розділів, висновків, списку використаних джерел та додатків.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ІНТЕГРАЦІЇ СИСТЕМИ РОЗПІЗНАВАННЯ МОВЛЕННЯ В СЕРЕДОВИЩЕ ДЛЯ НАПИСАННЯ ПРОГРАМНОГО КОДУ

## 1.1 Опис предметної області

Автоматичне розпізнавання мовлення – ціль останніх семи десятиліть. При перших комп'ютерах зв'язок між людиною і комп'ютером досягався механічними клавіатурами, а потім в допомогу прийшли і комп'ютерні мишки. Но останньою тенденцією, яка допомагає керувати комп'ютерними системами є розпізнавання мовлення і в подальшому перетворення цих команд в зрозумілу для комп'ютерних систем мову. Ця технологія стала широко розповсюджуватись майже у всіх сферах зв'язаних з інформаційними технологіями. Не так інтенсивно крокує розвиток, але дослідники багато вклали в можливість інтегрування в наш побут і продовжують це робити в даний момент часу. Ключовою проблемою залишається точність розпізнавання мовлення.

Точність порушується багатьма факторами, які нас оточують нас і переслідують кожну хвилину. Людина адаптувалася до більшості цих факторів і ігнорує їх в процесі розпізнавання тексту іншої людини. Комп'ютерній системі важко на перших часах розділити корисну інформацію від залишкової, тому її треба навчити це розрізняти і так само як людина ігнорувати цю інформацію або інтерпретувати нові діалектні слова. Серед основних факторів можна виділити шум, неоднотипне мовлення диктора, різні акценти і розставляння наголосів диктора, різні діалектичні слова або словосполучень, величина словникового запасу, продуктивність даної системи та багато інших.

До шуму або акустичного шуму можна віднести будь-які коливання частинок довкілля, які не дають корисної інформації. Усі вони заважають коректній роботі системі розпізнавання мовлення, так само коли ви розмовляєте по телефону з другом, а попри вас проїжджає автомобіль і привносить в ваш

діалог фізичне забруднення навколишнього середовище, таким чином також можна описати визначення шуму. До неоднотипного мовлення диктора можна віднести складність розпізнавання при ситуаціях, які викликали в диктора різні емоційні збудження. Що це означає? Наприклад розстроєні люди стають говорити тихо, можна сказати що говорять тільки в собі. Злість, яка може перерости в крик також важко буде вдаватися комп'ютерним системам в розпізнаванні мовлення. Не тільки емоції можуть стати бар'єром для розпізнавання, але й швидкість подачі інформації диктором комп'ютерній системі. Вони можуть так сказати «з'їдати» букви при вимовлені, а система отримає тільки інформацію у вигляді незрозумілих словоформ. Для цих проблеми є рішення, яке допоможе позбутися цих недоліків. Це навчання цих же систем за допомогою нейронних мереж, але зараз не про них.

Проблема розпізнавання акцентів і різного розставлення наголосів у словах є одною з важких завдань на даний момент для систем розпізнавання мовлення. Ця мовна варіативність породжує великі складнощі, які важко передбачити або спробувати навчити і адаптуватися. Різниця і проблема в акцентах в технічному розуміння полягає в тоні диктора, його наголосів та довжині, які важко передбачити та змоделювати. Не можна виключати такий параметр як схожість деяких видів акцентів, але є відмінності на фонологічному рівні. На даний момент є можливість вирішити цю проблему, Keqi Deng, Songjun Cao та Long Ma описали свій варіант вирішення: «… ми пропонуємо нову архітектуру для ідентифікації акценту. На відміну від x-vector, ми безпосередньо визначаємо акцент на основі кожного кадру, а не вектора на рівні пропозиції, а потім обчислюємо середнє значення всіх вихідних даних на рівні кадру як остаточне передбачення моделі. …» [3 - 5]. В сучасному світі саме англійська мова є однієї із самих поширених, бо нею користуються в міжнародних справах по бізнесу, різні організації, для прикладу Організація Об'єднаних Націй, Міжнародний Комітет Червоного Хреста, Організація Північноатлантичного договору,

Західноєвропейський союз, Організаація Об'єднаних Націй з питань освіти, науки і культури, а також багато інших. Учасники цих відносин розташовуються в різних куточках світу і їх рідна мова є різною. В кожного з цих учасників може бути свій акцент і розставлення наголосів у англійській мові. Саме англійська мова має велику кількість акцентів і саме її буде найскладніше адаптувати і навчити для автоматичного розпізнавання.

Ще одною проблемою яка була перелічена в списку вище - це діалекти. Розпізнавати, для прикладу, українську мову по сучасних словниках не має бути проблемою для систем автоматичного розпізнавання мовлення, але якщо відправитися від свого місця проживання в іншу частину України, то можна почути багато нових і інколи незрозумілих на перший погляд слів або словосполучень. В українській мові це ще називають наріччя. Пачева В.М. класифікує українські діалекти таким чином:

- північноукраїнський;
- південно-західний;
- південно-східний [7].

Зверніть увагу на рисунок 1.1 та на таблицю 1.1, які відображають розміщення різних діалектів на карті України.

Таблиця 1.1 – Розміщення різних діалектів на карті України [7]

| | Північноукраїнська (також в деяких областях Білорусі) | Південно-західна | Південно-східна (також в деяких областях Росії) |
|---|---|---|---|
| АР Крим | | | + |
| Вінницька | | + | |
| Волинська | + | + | |
| Дніпропетровська | | | + |

Продовження таблиці 1.1.

| | Північноукраїнська (також в деяких областях Білорусі) | Південно-західна | Південно-східна (також в деяких областях Росії) |
|---|---|---|---|
| Донецька | | | + |
| Житомирська | + | + | |
| Закарпатська | | + | |
| Запорізька | | | + |
| Івано-Фр. | | + | |
| Київська | + | | + |
| Кропивницька | | + | + |
| Луганська | | | + |
| Львівська | | + | |
| Миколаївська | | + | + |
| Одеська | | + | + |
| Полтавська | | | + |
| Рівненська | + | + | |
| Сумська | + | | + |
| Тернопільська | | + | |
| Харківська | | | + |
| Херсонська | | | + |
| Хмельницька | | + | |
| Черкаська | | + | + |
| Чернівецька | | + | |
| Чернігівська | + | | |

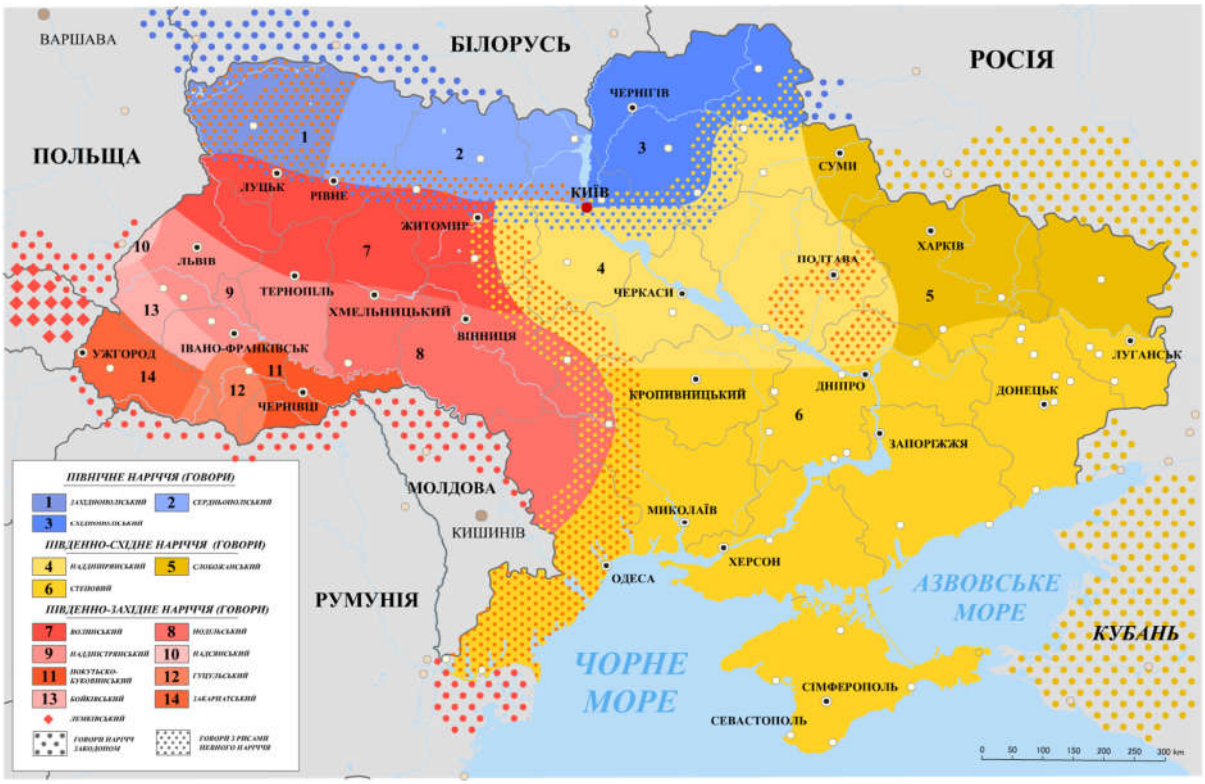


Рисунок 1.1 - Карта українських наріч і говорів (2005) [6]

Бачачи цей різновид наріч в українській мові, можна сказати, що навчити систему розпізнавання мовлення не є простою задачею, адже всі не використовуємо в побуті чисту українську мову для спілкування. Також не варто забувати ще й про так званий суржик, який оточує нас повсюди. Найбільше поширеним є українсько-російський суржик.

Розмір словника також має ключову роль для розпізнавання мовлення диктора. Розмір словникового запасу Fendji, Jean Louis K.E. класифікують таким чином: « …

- малий словниковий запас: від 1 до 100 слів або речень.
- середній словниковий запас: від 101 до 1000 слів або речень.
- великий словниковий запас: від 1001 до 10 000 слів або речень.
- дуже великий словниковий запас: понад 10 000 слів або речень.

…»[8]. Є ще інша класифікація запропонована Whittaker and Woodland, але про неї розмови не буде.

Продуктивність даних систем можна описати як порівняння фактичного результату і очікуваного, а якщо коротко, то це просто точність співпадіння. Ще до продуктивності відноситься швидкість розпізнавання. Це є ключовим фактор для бізнесу. Якщо людина спробує зробити запит через їх систему, а швидкість опрацювання тексту буде втричі довшим за довжину самого запиту, тоді клієнт цього продукту просто відмовиться від даного програмного забезпечення і буде шукати альтернативи в конкурентів, а цього ніяким чином не можна допустити. Яким чином визначити точність? Точність визначається за допомогою коефіцієнта помилок слів або Word Error Rate (далі WER). Швидкість розпізнавання мовлення досліджується через коефіцієнт реального часу. Також можна вказати інші варіанти відображення точності, такі як коефіцієнт одного слова або Single Word Error Rate (далі SWER) та коефіцієнт успіху команди або Command Success Rate (далі CSR).

Яким чином система автоматичного розпізнавання мовлення працює? Якщо коротко, тоді для початку на потрібний звук. Людина починає вимовляти якісь слова в мікрофон, які мають бути перетворені в текст. Для розуміння комп’ютером цієї вимови потрібно ці звуки оцифрувати. Ці звуки потім перетворюються в акустичні вектори за допомогою акустичної моделі. В подальшому система проводить дослідження цієї послідовності векторів і порівнює її з уже записаними в мовній моделі. Система аналізує найбільшу відповідність сказаному і видає результат.

Якщо вдаватися в деталі, то ця система має таку архітектуру, яка описана Fendji, Jean Louis K.E. і вміщає в собі п’ять частин : «…

- вилучення ознак.
- акустична модель.
- мовна модель.

- модель вимови.
- декодер …»[8].

Останніми роками технологія розпізнавання мовлення получає великого розголосу Цей метод часто використовується фірмами та фізичними особами підприємцями, тому що вона приносить багато переваг.

«Привіт, Siri» або «Alexa» - розпізнавання голосу, яке зазвичай називають технологією розпізнавання голосу, у нашому світі не є чимось новими. Це відноситься до технології, яка може перетворювати розмовну мову в машиночитану форму. Тепер є можливість підключитися до свого пристрою, щоб він виконувати команди, як у науково-фантастичному серіалі.

Більша частина технологій розпізнавання мовлення володіють точністю в 95%. Немає сумнівів, що в 2022 році голосовий пошук буде мати 50% від усіх запитів до пошукових і інших систем. Окрім персональних мобільних пристроїв, розпізнавання голосу також забезпечує міцну основу для бізнесу, особливо для покращення обслуговування клієнтів та боротьби з кіберзлочинністю.

Інновації голосового розпізнавання також увійшла у сферу охорони здоров'я. Фахівці протягом тривалого часу переписують свої клінічні записи, використовуючи цю інновацію. Оскільки додаткові досягнення, такі як штучний інтелект, машинне навчання та обробка характерних діалектів, входять в індустрію охорони здоров'я, інновації у сфері розпізнавання мови можуть кардинально змінити спосіб спілкування спеціалістів, медичних працівників та пацієнтів.

Тож які переваги інновації з розпізнавання голосу? Чому нам потрібні комп'ютери, щоб розшифрувати наш діалект, коли письмо, як правило, швидше? Численні комп'ютерні програми, які стають все більш поширеними, мають голосове введення. Вот перелік переваг інноваційного розпізнавання голосу та його роль в житті людей:

- вища продуктивність. Велика частина людей розмовляє набагато швидше ніж пише, програма розпізнавання голосу може без зусиль змінити характерний діалект на зміст, не затримуючись. Багато людей використовують його через швидкість.

- інновація під назвою «вільні руки». Коли ви за кермом або маєте справу, яка не дає можливості користуватися руками, голосові помічники з інтегрованою функцією розпізнавання голосу прийдуть на допомогу. Можливість підключення до Apple Siri або Google Maps, щоб направити вас туди, де ви хотіли б бути, зводячи до мінімуму ймовірність загубитися або зупинятися.

- дає можливість роботам розмовляти. Можливо ви не знали, але спілкування з роботами – звична справа. У деяких сервісах, роботи витісняють людей у спілкуванні та взаємодії з клієнтами. Можна проілюструвати уже актуальний випадок, коли компанії зараз тестує роботів і комп'ютерну програму для інтерв'ю. Оскільки зустріч має бути комунікативною, робот повинен зрозуміти те, що говорить інтерв'юйований. Без такого програмного забезпечення яке містить алгоритми розпізнавання мовлення це б не було можливим.

- Комп'ютеризоване адміністрування обладнання. Комп'ютеризовані індивідуальні співробітники, такі як Alexa і Google Home, вимагають від груп людей усного спілкування з ними. Вони є чудовим описом того, як машинне навчання та комп'ютери можуть з часом покращити розуміння вашого голосу. Як би там не було, для цього потрібна технологія розпізнавання голосу через обробку сигналу.

- Вирішення проблеми людей, які мають проблеми із слухом та зором. Багато осіб з даними обмеженими можливостями використовують екранні читачі та гаджети для запису слів диктора. Для людей із вадами слуху і поганого зору інтерпретація звуку в зміст може врятувати життя.

Тенденція щодо розпізнавання мовлення в даний час є частиною нашого повсякденного життя, незважаючи на те, що наразі вони обмежуються нескладними функціями. У міру розвитку інновації аналітики зможуть створювати набагато складніші процедури для розпізнавання розмовного діалекту. Одного дня ви можливо зможете спілкуватися разом зі своїм комп'ютером як з живою особою, яка реагуватиме виваженими відповідями. Технологія в обробці сигналів дасть можливість все це принести в нас світ. Попит на професіоналів у цій галузі продовжує невпинно іти вгору і численні організації шукають досвідчених людей, щоб підключити їх до своїх проектів. Велика кількість сучасних новацій та стратегії комунікації залежать від обробки, перекладу та розуміння сигналів голосу. Відповідно до існуючих шаблонів, розпізнавання мовлення буде продовжувати розвиток як підгрупа розпізнавання сигналів, яка швидко буде крокувати вперед протягом тривалого часу[9 - 10].

## 1.2 Порівняльний аналіз відомих рішень

На сьогоднішній день написання коду за допомогою голосу через спеціальне програмне забезпечення не є такою поширеною практикою в сфері розробки програмного забезпечення. Все через те, що саме розпізнавання мовлення не є таким швидким в порівнянні з ручним написання коду. На мою думку це програмне забезпечення знадобиться для людей, які мають травми або хвороби зв'язані з їх інструментами – руками. Саме такий чоловік получив травму через свою постійну роботу за комп'ютером. Його звуть Мет Вітхофф, він працював розробником у фірмі Quora. У нього діагностували травму - синдром зап'ястного каналу кисті рук і тому він перестав кодувати. «Я залишив роботу інженера-програміста в Quora, тому що не міг більше працювати», - сказав він. «Потрібно було вибрати іншу професію, яка не вимагала стільки друкування, або знайти якесь рішення». Він і його знайомий Томі Маквільям спробували

скористатися штучним інтелектом для розробки програмного забезпечення, яке дасть можливість кодувати за допомогою голосу. Таким чином був прокладений перший крок до програми під назвою Serenada. Для розвитку проекту Serenade отримала інвестиції в розмірі 2.1 мільйона доларів[11]. Так які особливості має ця програма?

Насамперед це безкоштовне програмне забезпечення, яке є мультиплатформним. Тобто воно може співпрацювати з такими операційними системами як Windows, Linux та macOS. Коли програмне забезпення вставиться, вас перенаправлять до підтримуваних інтегрованих середовищ, які встановлені на комп'ютері, а підтримує вона Visual Studio Code, Chrome та Hyper. Там ви повинні встановити плагіни від Serenade для правильної роботи Serenade і вашого середовища програмування. Весь код який набирається за допомогою мовлення обробляється на хмарному сервісі, хоча є можливість робити запити до програми і локально. Це допоможе зберегти всі ваші дані на вашому комп'ютері, тобто обробка буде відбуватися без посередників.

Використання цього програмного забезпечення простіше нікуди. Вікно програми може бути закріпленим над всіма іншими вікнами робочого столу для комфортного використання. Сам інтерфейс програми складається з індикатора, який відображає режим прослуховування. Активація прослуховування команд відбувається через натискання комбінації клавіш Alt+Space або натиснути лівою клавішею миші по кнопці з назвою Listening. Під нам знаходиться список розпізнаних системою команд. Їх може бути багато, якщо система не визначилась остаточно. В цьому випадку вам потрібно визначитися із переліком результатів і сказати номер варіанту який вам підходить. Для цього необхідно озвучити цей номер, в іншому випадку сказати «Undo» і спробувати сказати команду ще раз. В стані по замовчуванню стоїть перший варіант розпізнання. Інтерфейс системи розпізнавання голосу та роботи з кодом написаним на JavaScript представлений

на рисунку 1.2. Також на ньому відображено приклад використання форми утворення команди “add” та ” end of file ”.

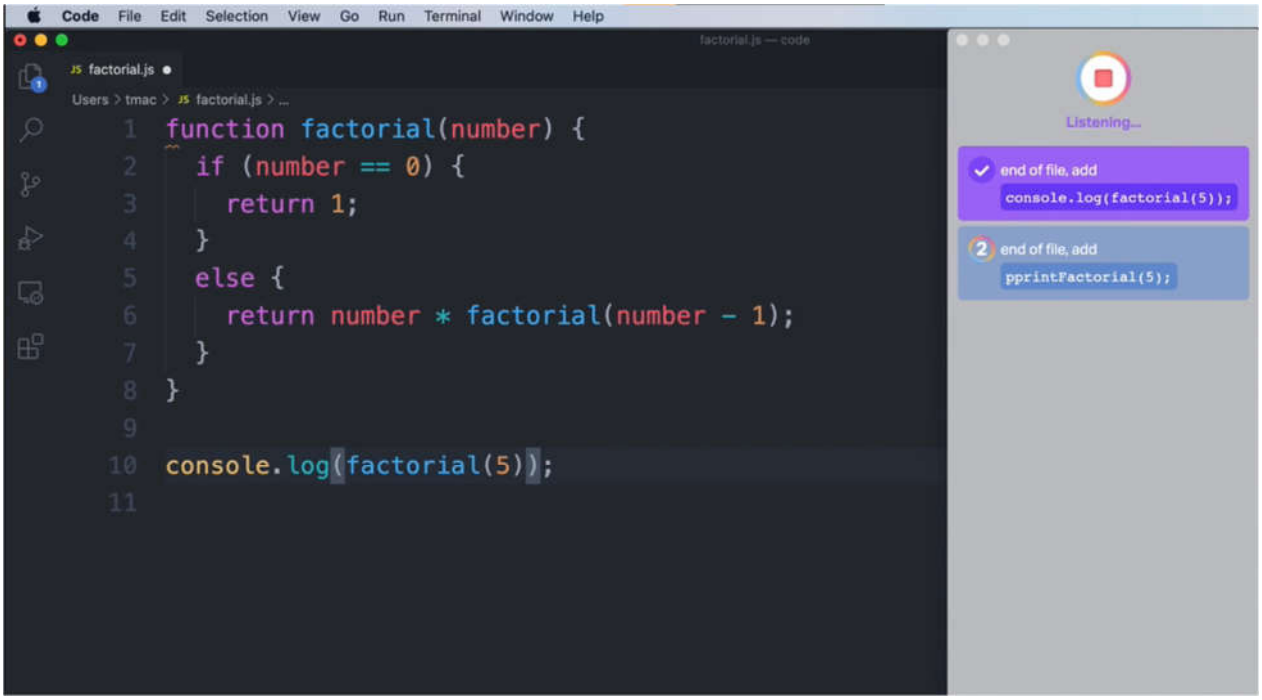


Рисунок 1.2 – Приклад інтерфейсу і запиту: «End of file, add print factorial of five»

Майже всі команди до системи мають схожі форм утворення – це «action» та «selector». Ця форма комбінації дає велику кількість варіантів. Якщо по простому, то «action» - це те, що потрібно зробити з кодом, а «selector» - це код, який частина коду, над якою будуть проходити маніпуляції. Базові дії – це «add» - щоб добавити щось в код, «change» - змінити щось в коді та «delete» - видалити щось з вашого коду. Селекторами можуть бути такі слова як «line», «word», «class», «function» та «return value». Приклади комбінації:

- go to word Tests.
- add return true.

- change good to nice.
- delete line ten [12].

На рисунку 1.3 відображено написання коду HTML з допомогою програми Serenade. Також відображено похибку в розпізнаванні, яка показана через велику варіативність результатів.

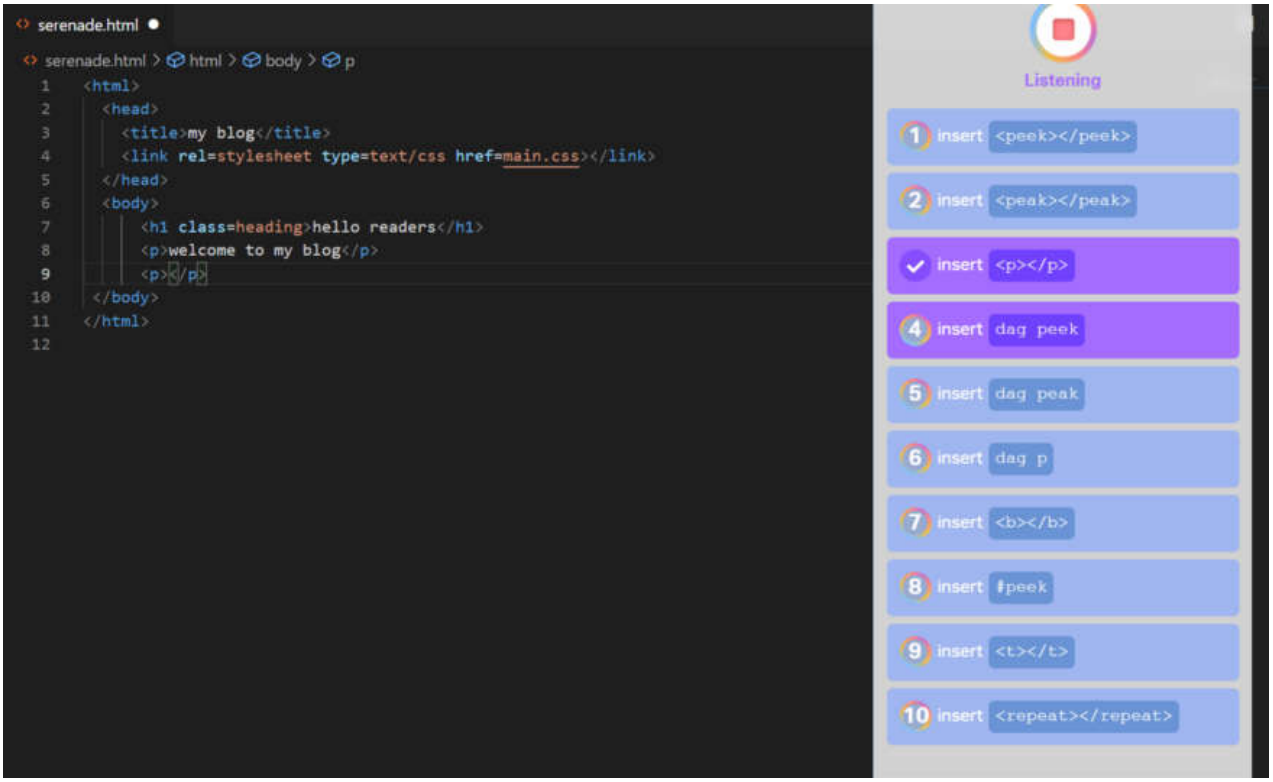


Рисунок 1.3 – Приклад роботи програми з кодом HTML[13]

Така сама проблема сталася із Беном. Програміст, якому діагностували синдром зап'ястного каналу кистей рук через надмірне використання клавіатури. З його слів, йому було навіть важко тримати вилку в руках, а за програмування можна було забути. Різні засоби типу ергономічних клавіатур або м’які клавіатури iPad не давали потрібного результату. Початок проекту почався від використання Dragon Dictate. Це програмне забезпечення для розпізнавання мовлення. Це була розробка Dragon System для Microsoft Windows. На заміну

старій версій прийшла новіша версія – Dragon NaturallySpeaking для Windows. На даний момент цю фірму придбала інша небезіменна компанія Nuance Communications. Їх продукт для розпізнавання мовлення є простим в налаштуванні та розташовується у хмарі. Його легко імплементувати в уже існуючу сферу, також є можливість оптимізувати систему для всіх перелічених особливостей замовлення[14]. Інтерфейс Dragon Dictation відображений на Рисунку 1.4.

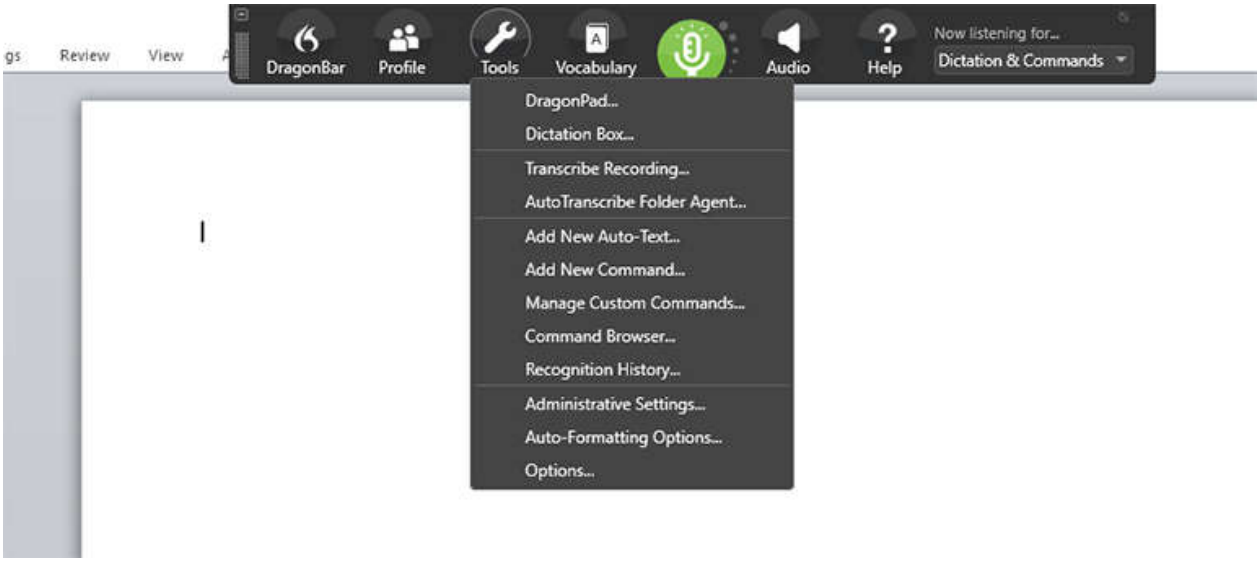


Рисунок 1.4 – Інтерфейс панелі керування Dragon Dictation[15]

Сама програма Dragon Dictate могла тільки розпізнавати мовлення і переводити його в текст. Цього замало щоб відновити можливість писати програми. Для цього Бен спробував налаштувати все під себе. Створював користувацькі команди, які були необхідні для комфортного користування комп’ютером. Розробка відбувалася протягом трьох років, які важко вдавалися враховуючи хворобу. В результаті вкладеного часу в Бена вийшов програмний продукт, який давав можливість людині в режимі реального часу керувати

комп’ютером. Можливостей є достатньо, наприклад створення, написання і відправка електронного листа, переключання вікон програм або навіть навігація по Photoshop ну і звісно головною метою розробки була можливість написання програмного коду в середовищі для програмування. Бен також придбав інфрачервону камеру для керуванням курсором по робочому столі комп’ютера. Цей комплект називається SmartNav 4:EG, він спеціально був розроблений для людей, які мають синдром зап'ястного каналу кистей рук. Для найточнішого керуванню курсором людина одягала на голову кепку із спеціальним датчиком на щитку головного убору. Приклад відображено на рисунку 1.5. Все що потрібно, так це ледь повертати голову для керування курсором. Інфрачервона камера реагує на рух кепки з спеціальним давачем, а в результаті спеціальне програмне забезпечення реєструє найменші зміни параметрів і заставляє рухатися курсор по робочому столі комп’ютера [16].


Рисунок 1.5 - SmartNav 4:EG [17]

Ця фантастична комбінація VoiceCode на базі Dragon Dictate та SmartNav дає безмежні можливості для людей з синдром зап'ястного каналу

кистей рук. Встановити це програмне забезпечення можна як плагін для інтегрованого середовища програмування Visual Studio Code [14].

Ще одним аналогом систем для написання коду голосом є Talon. Ця система дає можливість людям програмувати та повністю володіти персональним комп'ютером, у яких є обмеження або хвороби зв'язані з руками. Також в програмі є можливість підключити системи по відстеженню руху очей Tobii 4C, Tobii 5. Все це працює на основі інтегрованих в Talon алгоритмів. Вона підтримує багато популярних операційних систем, а саме macOS High Sierra(10.13) та звісно і нові версії, Linux/X11 (зокрема дистрибутив Ubuntu версії 18.04) та сама відома система Windows версії 8 та 10. Доповнити систему Talon додатковою функціональністю наразі є можливим за допомогою API сценаріїв, вони використовують мову програмування Python та мовою для сценаріїв TalonScript. Дана API дає користувачу потужний інструмент, який моде детально налаштувати систему Talon під ваші потреби. Наприклад реєстрація голосової активності, погляд користувача в якусь точку або певні шуми. Весь перелік цих активностей можна зарезервувати для якоїсь реакції Talon. Це може бути як відкриття конкретної програми або імітування натискання комбінації клавіш.

На даний момент доступна версія 0.2.3 від 30 вересня 2021 року. В ній було виправлено деякі помилки з відображенням Dragon Speech Recognition в меню. На системі під управлінням Linux була виправлена проблема з неправильним центруванням imGUI на декількох моніторах. Це свідчить про те, що проект росте і процвітає, на відмінну від інших аналогів, які перестали підтримуватися розробниками. Розпізнавання мовлення за допомогою Talon може відбуватися на різних двигунах. До їх переліку входять як повністю безплатні варіанти, так і платні версії. Перший з них носить назву «w2l gen2». Має сумісність з такими операційними системами як Windows, macOS та Linux. Використання даного двигуна в кооперації з Talon є повністю безкоштовним. Другий – «w2l conformer», можна сказати що один з найкращих варіантів для нових користувачів. Він

гарантує високу точність і швидкість розпізнавання як команд, так і написання диктанту. Доступна мала затримка для користувачів, які користуються beta версією Talon, досягається це за рахунок постійної оптимізації продуктивності. Його відносять також до безплатних двигунів. Останнім варіантом є Dragon Speech Recognition. Він також сильний в розпізнаванні команд і написання диктантів, хоча є нюанси, які розробники не можуть налагодити. Слова з вікіпедії на рахунок використання Dragon: «…Професійна версія рекомендована замість домашньої (домашня версія не має командного режиму). …»[18 - 20].

Ще існують менш відомі продукти, які знаходяться в репозиторіях GitHub. Це є Aenea та Caste. Вони служать як оболонками, бо основну роботу виконує той самий двигун Dragon Speech Recognition від Nuance Communications.

## 1.3 Постановка задачі дослідження

Поставлену задачу по дослідженню потрібно розділити на окремі пункти для кращої візуалізації та коригуванню плану. Які вимоги можуть бути у розробника програмного забезпечення, який буде користуватися даною системою? Насамперед – це безпека. Система має виконуватися локально, щоб усі голосові команди та код безпечно зберігалися в локальному середовищі. Завдяки розпізнаванню мовлення на пристрої голосові команди повинні виконуватися ще швидше через відсутність посередників. Команди ніколи не повинні залишати пристрій, тому не прийдеться турбуватися про брандмауери чи VPN. Має вийти ідеальне рішення для проектів із закритим кодом. Але це реалізувати достатньо важко, потрібно мати високопродуктивний пристрій та гарно навчену систему. В нашому випадку – це буде система навчена за допомогою глибинних нейронних мереж. Багато компаній використовують цей підхід в своїх проектам по розпізнаванню голосу. Переважно це все системи, які виконують аналіз віддалено на своїх серверах.

За безпекою повинна слідувати точність, без точного і швидкого реагування на команди ця система не дасть відповідного враження користувачу. За точність відповідати повинна інтегрована API. В нашому випадку – це Web Speech API – Speech Recognition. Це безплатний сервіс з відкритим вихідним кодом. Від нього не потрібно чекати чудес, але для алгоритму інтегрування він зможе дати цікавий результат, який задовільнить потреби. Потрібно розпізнавати лише команди – короткі і стислі запити, які будуть інтерпретуватися системою в потрібний для нас результат.

Ще однією вимогою є велика варіативність команд, які повинні бути присутніми в базі системи. Наприклад команди з роботою над файловою системою:

– створення;
– читання;
– видалення;
– редагування.

Створення має бути непростим. Має бути заложений функціонал для створення різного формату, а для задач програмування їх є велика кількість. Серед них можна виділити одні з поширених, а саме:

– *.htnl – формат для написання коду на мові гіпертекстової розмітки (HyperText Markup Language).

– *.css – формат для написання коду по стилізації веб-сторінки. Мова називається CSS або каскадні таблиці стилів (Cascading Style Sheets).

– *.js – формат для написання коду на мові програмування JavaScript. Проста об’єктно-орієнтована мова програмування. На ньому переважно пишуть front-end функціонал, реакції на дії користувача та анімаційну складову.

– *.php – формат для написання коду на мові програмування PHP або Hypertext Preprocessor (гіпертекстовий процесор). Пешою назвою було Personal

Home Page, тому і получилась абревіатура PHP. Його переважно використовують для веб-розробки, а приймає участь в опрацюванні форм на веб-сторінці, дії з базами даних – CRUD. Ця абревіатура розшифровується як Create, Read, Update та Delete. Аналоги запитів до бази даних – це Insert, Select, Update та Delete. Ще однієї з сильних сторін, так це створення і керування сесіями і куками (cookies). Також можна завантажувати і працювати з файлами.

– *.py – формат для написання коду на мові Python. Це скриптова мова може використовуватися для написання комп'ютерних і мобільних програм. Ще для написання веб-додатків та використання в машинному навчанні. Поширено використовують у своїх проектах такі гіганти як Amazon, а найбільше в Amazon Web Service, Spotify, Youtube та Instagram.

Читання як видалення має бути само собою, для перегляду і видалення залишків коду обов'язково необхідні ці функції. Редагування – це основна ціль цієї системи.

Уже при написанні коду важливим буде вставляти в файл готові шаблони мови. Для HTML – це стандартний набір з тегів <html>, <head>, <meta>, <title>, <style> та <body>. <html> - це вмістилище, в якому знаходяться всі дані веб-сторінки. <head> - може вміщає в собі багато інших тегів: <link>, <script>, <title>, <style> та <meta>. Він допомагає браузеру в роботі з інформацією, наприклад метадані, які вміщають в собі опис сайту і ключові слова для кращого ідентифікування сайта в мережі Інтернет. <title> - це заголовок документа. Він знаходиться в назві вкладки браузера. Наприклад пошукова система Google.com має заголовок Google. <style> - тут може знаходитися код стилів для сайту або CSS. Його не рекомендують писати в файлі *.html, кращим варіантом буде написати посилання на файл стилів, а сам код розмістити у файл *.css у кореневу папку або навіть створити ще одну папку з назвою "css" та помістити файл туди. <script> - це контейнер переважно використовують для написання в ньому JavaScript коду. Рекомендації для <style> також варто застосувати і тут,

залишивши в файлі *.html тільки посилання на JavaScript код. <body> або «тіло» html файлу. Все шо ви хочете показати користувачу слід писати в ньому. В ньому можна розміщати текст, фотографії, інші теги та навіть JavaScript код. Приклад такого шаблону відображено на рисунку 1.6.


Рисунок 1.6 – Вбудований шаблон коду html в Visual Studio Code

Враховуючи що це має бути розширення для інтегрованого середовища розробки Visual Studio Code, в майбутньому його можна буде розширити для інших платформ і інтегрувати його в комфортне для вас середовище. Для прикладу:

- Eclipse;
- ActivateState Komodo;
- IntelliJ IDEA;
- MyEclipse;
- Oracle JDeveloper;
- NetBeans;
- Codenvy;
- Microsoft Visual Studio та інші.

Тільки ця система може мати і деякі складнощі з помилками розпізнавання, які важко буде вирішити за короткий час. Щоб вирішити цю проблему необхідно накопичувати велику базу даних щоб побороти тільки деякі невідповідності в отриманих результатах. Вот перелік основних факторів, які можуть кардинально вплинути на фактичний результат:

- фоновий шум;
- акценти;
- рідкісні слова.

Найсучаснішим, швидким і найбільш розвиненим методом для розпізнавання мовлення вибрано алгоритми глибинної нейронної мережі. Саме його масово використовуються для розпізнавання зображень, обробки природної мови та автоматичного розпізнавання мовлення. Онищенко, К. Г. та інші дають таке визначення глибинним нейронним мережам: «… - галузі машинного навчання, що ґрунтується на наборі алгоритмів, які намагаються моделювати високорівневі абстракції в даних, застосовуючи глибинний граф із декількома обробними шарами, що побудовано з кількох лінійних або нелінійних перетворень …» [21].

Оболонкою для цієї системи буде Artyom.js. Саме вона вміщає в собі webkitSpeechRecognition. webkitSpeechRecognition – це можливість використовувати Web Speech API для браузерів та настільних комп’ютерів, а ще можна інтегрувати і в мобільні телефони (можна інтегрувати в мобільні системи тільки з версії 33).

Перелік всіх доступних браузерів можна глянути в таблиці 1.2.

Таблиця 1.2 – Підтримка Web Speech API браузерами [22]

| | Браузери | Web Speech API |
|---|---|---|
| Desktop | Chrome | Підтримується з версії № 33 (Apache/Nginx etc). |
| | Edge | Підтримується з версії № 79 (Apache/Nginx etc). |
| | Firefox | Не підтримується даним браузером. |
| | Internet Explorer | Не підтримується даним браузером. |
| | Opera | Не підтримується даним браузером. |
| | Safari | Підтримується з версії № 14.1 |
| Mobile | WebView Android | Підтримується з версії № 4.4.3 (для роботи потрібний веб-сервер) |
| | Chrome Android | Підтримується з версії № 33 (для роботи потрібний веб-сервер) |
| | Firefox for Android | Не підтримується даним браузером. |
| | Opera Android | Не підтримується даним браузером. |
| | Safari on iOS | Підтримується з версії № 14.5 |
| | Samsung Internet | Підтримується з версії № 2.0 (для роботи потрібний веб-сервер) |

В підсумку можна виділити основні переваги та недоліки цієї ідеї. До переваг можна віднести:

– розпізнавання багатьох мов світу (гінді, французька, німецька, японська, нідерландська, іспанська, польська, кантонська, мандаринстка, індонезійська, російська, англійська американська і британська англійська, італійська та португальська). Web Speech API працює на основі Google Cloud Speech-To-Text, там перелік мов набагато більший, але API від Mozilla та оболонка від Artyom.js дають свої обмеження;

– відкритий вихідний код для можливості добавляти свої команди. Це є великою перевагою, якщо ви бажаєте максимально налаштувати систему для свого комфортного використання;

– на відміну від інших аналогів (VoiceCode, Talon та Serenade) в майбутньому є можливість легко інтегрувати новітні системи розпізнавання мовлення через прикладний програмний інтерфейс;

– можливість створювати файли в програмному середовищі спеціальних форматів (*.js, *.html і так далі). В аналогах можна працювати тільки у відкритому файлі.

До негативних сторін можна віднести:

– достатньо сирий продукт, який тільки ілюструє можливий потенціал системи;

– відносно низька точність розпізнавання мовлення;

– відсутній комфортний інтерфейс;

– важкий в розуміння для новачків.

Отже, метою кваліфікаційної роботи є підвищення ефективності написання програмного коду за допомогою алгоритму, який дозволить інтегрувати новітні системи розпізнавання мовлення в середовище програмування.

## Висновки до розділу 1

1. Проведено дослідження області розпізнавання мовлення для написання програмного коду. Описано застосування даної технології в побуті, сфері охорони здоров’я, бізнесі та інших галузях. Відображено вирішення проблем через використання даної інновації. Показано важкість навчання таких систем через глибокі нейронні мережі, що вимагають великі об’єми наборів аудіо даних та потужні електронно-обчислювальні машини.

2. Запропоновано вдосконалити алгоритм інтеграції системи автоматичного розпізнавання мовлення в середовище написання програмного коду через метод пошуку недоліків. Виконано порівняльний аналіз одних із самих популярних аналогів систем для написання програмного коду голосом. Інтеграція інноваційних систем розпізнавання мовлення та універсального середовища, яке має можливість до адаптації під нові вимоги. Для прикладу – багатомовність. Детально описано переваги та недоліки вибраної системи в порівнянні з аналогами.

3. Універсальний алгоритм для інтеграції системи автоматичного розпізнавання мовлення в середовище написання програмного коду дасть можливість створювати файли з доступними форматами. Також спроможність голосом вводити код у створені файли.

# 2 ІНТЕГРАЦІЯ СИСТЕМИ РОЗПІЗНАВАННЯ МОВЛЕННЯ В СЕРЕДОВИЩЕ ДЛЯ НАПИСАННЯ ПРОГРАМНОГО КОДУ

## 2.1 Концептуальні основи інтеграції систем розпізнавання мовлення в середовище для написання програмного коду

Для розуміння тексту написаного нижче необхідно всі концептуальні основи описати і відобразити прикладами для кращого розуміння фундаменту алгоритму інтегрування. Інтегрування алгоритму системи розпізнавання мовлення в середовище розробки дозволяється лише через розширення. Що таке розширення, які є їх види і для чого вони існують?

Кожен, хто працював у будь-якому інтегрованому середовищі розробки Visual Studio, у певний момент працював із розширенням або Extensions. Розширення — це надбудови до Visual Studio, які дозволяють розробникам налаштовувати та покращувати роботу Visual Studio, додаючи нові функції та інтегруючи наявні інструменти. Основна мета розширення — підвищити продуктивність і задовольнити робочий процес розробників.

Однією з найбільш вагомих причин використання Microsoft Visual Studio як інтегрованого середовища розробки або редактора коду є те, що він має величезну бібліотеку розширень, які можна завантажити, щоб додати функції та функціональні можливості до того існуючого функціоналу. Ці розширення, доступні через Visual Studio Marketplace, як безкоштовними, так і платними версіями, багато з яких пропонують безкоштовну пробну версію [23 - 24]. Також ці всі розширення поділяються на різні категорії, а саме:

- Data Science;
- налагоджувачі;
- навчання;
- пакети розширень;

- форматери;
- keymaps;
- мовні пакети;
- linters;
- машинне навчання;
- блокноти;
- мови програмування;
- SCM провайдери;
- snippets;
- тестування;
- теми;
- візуалізація.

Кожен з цих категорії вміщає в собі величезний потенціал, який залишається розкрити розробнику програмного забезпечення або іншому спеціалісту в сфері інформаційних технологій. Наприклад одна з категорій під назвою Linters вміщає в собі розширення, які відповідають за пошук і відображення помилок при написанні коду. Це слово пішло від науковця під іменем Картіс Джонсон. Він писав програму для подолання проблем при переміщення Unix на 32-розрядну систему. Програма мала назву Lint, а саме слово lint перекладалось як шарпина – це поширена назва волокон тканинного матеріалу з вовни або чогось іншого, які скупчуються в кульки. Слово Linter – це ткацький верстат для роботи з вовною. Можна провести аналогію, що розширення Linter, як ткацький верстат, який розміщає кожний кусок коду на своє місце [25].

В магазині Visual Studio Marketplace можна знайти розширення під назвою ESLint. ESLint — це одне з розширень VS Code для JavaScript. Воно дозволяє аналізувати код JavaScript на наявність помилок і виправляти їх. Розширення

вказує на помилки, завдяки чому їх легко знайти. Це також допомагає виявити попередження. Крім того, ESLint також може автоматично форматувати ваш код JavaScript, щоб у вас була однакова архітектура написання коду в команді розробників. Інсталяція проводиться через Node Packet Manager (менеджер пакунків для JavaScript ) за допомогою команди в терміналі: npm install -g eslint. Знайдені помилки можна виправити через комбінацію клавіш в Visual Studio Code Ctrl+P або інша, залежно як налаштоване середовище. Помилки при компіляції будуть відображатися після запуску проекту через команду в термінали: npm run. Якщо будуть які помилки, ESLint одразу вас сповістить в термінали[26]. Приклад можна глянути на рисунку 2.1.

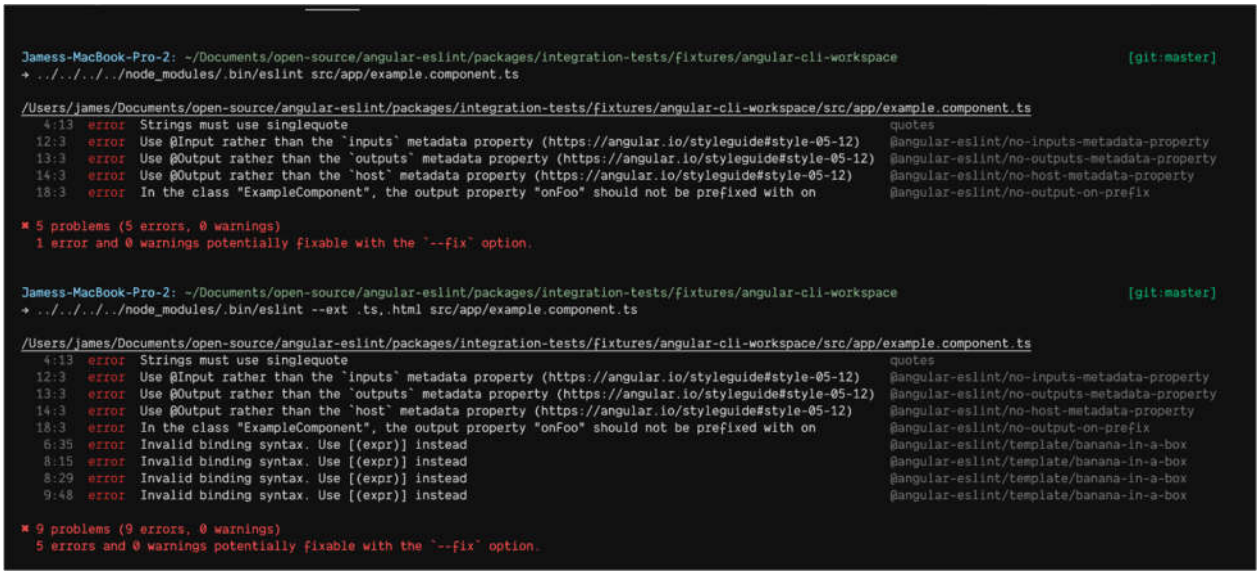


Рисунок 2.1 – Приклад запуску програми і відповідь ESLint [27]

Це розширення можна різним чином налаштовувати під свої потреби. Для цього у кореневій папці вашого проекту знаходиться файл під назвою .eslintr.* або поле eslintConfig в тому ж кореневому шляху в файлі package.json. Вигляд такого файлу можна глянути на рисунку 2.2.

Ось деякі з параметрів, які ви можете налаштувати в ESLint:

- середовища – для яких середовищ призначений ваш сценарій. Кожне середовище несе з собою певний набір попередньо визначених глобальних змінних.
- глобальні значення – додаткові глобальні змінні, до яких ваш скрипт отримує доступ під час виконання.
- правила – які правила ввімкнено і на якому рівні помилки.
- плагіни – які сторонні плагіни визначають додаткові правила, середовища, конфігурації тощо для використання ESLint [26].

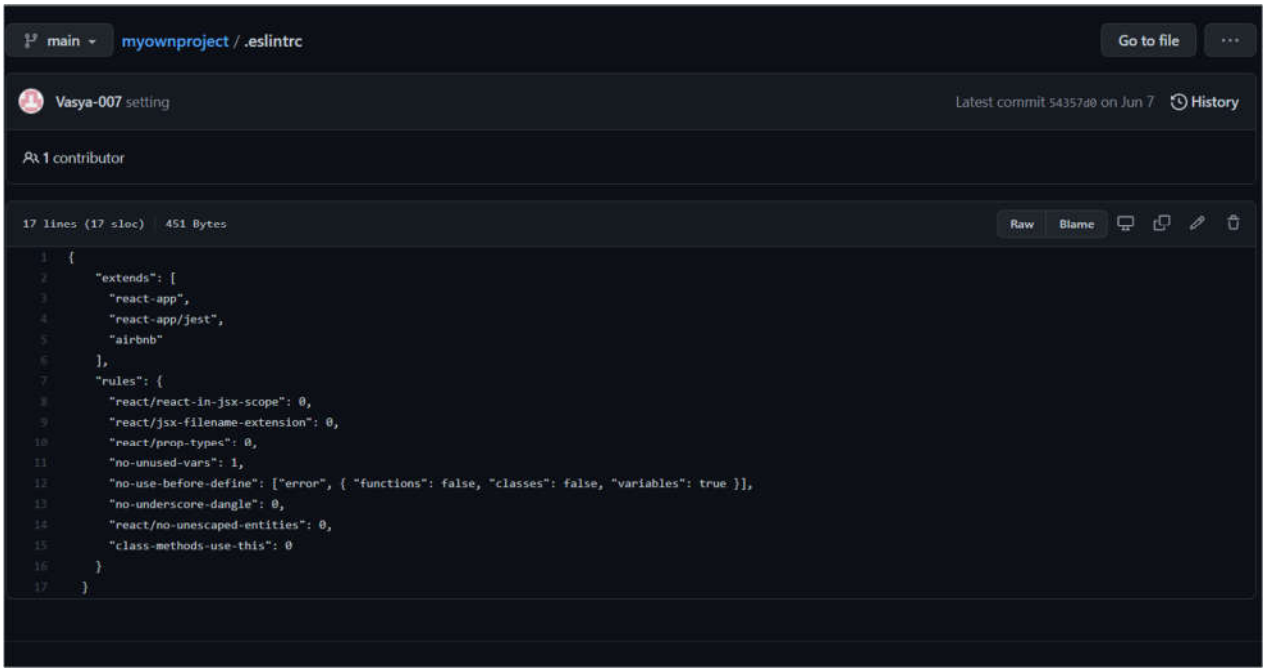


Рисунок 2.2 – Приклад структури файлу .eslintrc мого колишнього проекту

З розширення тепер все зрозуміло – це інструменти для збільшення продуктивності і водночас спрощення роботи для розробника. Далі потрібно розібратися з інструментами, які знадобляться при виконанні інтегрування. Що вони собою являють і на яких етапах вони будуть потрібні?

В першу чергу знадобиться сам редактор коду – Visual Studio Code. Саме для нього відбудеться розробка розширення – він буде фундаментом цього проекту. На його основі потрібно в подальшому встановити спеціалізовані інструменти.

Перший з них - це розширення з назвою Yaoman. Це спеціальний інструмент для запускання уже готових шаблонів програм або їх уривків. Це автоматизований процес, який уже має набір інструментів та програмних каркасів, які спрощують роботу і збільшують продуктивність розробників програмного забезпечення [28]. Він допоможе встановити всі залежності і шаблон для написання розширення.

Для розробки знадобиться також Node.js та встановлений Git. Що це таке і навіщо вони знадобляться?

Node.js — це середовище виконання JavaScript, яка побудоване на двигуні V8 Chrome. Він дозволяє запускати код за межами веб-браузера. Часто його використовують для розробки додатків, веб-додатків та написання сценаріїв на стороні сервера через JavaScript. Великою перевагою Node.js є менеджер пакетів Node або просто NPM. Підключення бібліотек в проект через NPM набагато спрощує роботу і економить величезну кількість часу [29].

Git є однією з відомих і знайомих програм в галузі інформаційних технологій. Це є система контролю версій, що широко використовується по всьому світі. Проект на етапі заснування в 2005 році мав відкритий код, його творцем є розробник операційної системи Linux – Лінус Торвальд. В даний момент проект активно підтримується і невпинно розвивається. Велика армія розробників програмного забезпечення покадається на керування версіями Git. В ньому є можливість розробити проекти комерційних і некомерційних типів. Основна мета – це забезпечення продуктивності, безпеки (вміст файлів, коміти і каталоги захищені криптографічним алгоритмом SHA1) та гнучкості, а це означає підтримку кількох видів нелінійних робочих процесів розробки [30].

І останнім фрагментом для збору цього пазлу залишається оболонка Artyom.js. Вона буде серцем проекту, за його допомогою можна розпізнавати голос і інтерпретувати в запрограмовану реакцію, наприклад слово «create html» створить пустий файл з розширенням *.html.

## 2.2 Дослідження та вибір системи розпізнавання мовлення

В даний період стрімкого розвитку технології по автоматизованому розпізнаванню мовлення багато компаній і зацікавлених осіб почали масово розробляти свої продукти для ринку. Це такі гіганти як Google, IBM, Microsoft, Facebook, Amazon та не треба забувати про некомерційну організацію Mozilla Foundation. Так само є і некомерційні організації, які також внесли свою проекти в маси. Майже всі з них мають свій персональний проект. Деякі з цих компаній діляться відкритим кодом своїх проектів, хоча переважно це системи для навчання своєї власної бази на основі записаних голосів. В них уже вбудований алгоритм, наприклад нейронна мережа, яка вчиться розпізнавати голос. Для цього їй потрібний словник фонем, акустична модель та мовна модель, яка пробує скласти набір слів у граматично правильне речення.

З популярних можна виділити такі системи як Watson Speech to Text(від International Business Machines Corporation), Google Cloud Speech-To-Text (від Google LLC), Alexa Skills (від Amazon ), сервіс Azure Speech to text (від Microsoft Corporation), wav2vec-U та Flashlight's ASR(колишній wav2letter++ від Facebook AI Research), DeepSpeech (від Mozilla Foundation), Dragon NaturallySpeaking (від Nuance Communications). Ще можна згадати про такі старі проекти, які підтримуються і досі або зовсім закрились:

- Kaldi ASR;
- CMUSphinx;

– SpeechBrain;

– Web Speech API;

– OpenSeq2Seq та інші.

Kaldi — це програмне забезпечення для розпізнавання мовлення з відкритим вихідним кодом, яке вільно доступне під ліцензією Apache. В Університеті Джона Хопкінса розробка почалася на семінарі у 2009 році під назвою «Низькі витрати на розробку, якісне розпізнавання мовлення для нових мов».

14 травня 2011 року був випущений код для Kaldi після кількох років роботи над проектом. Швидко Kaldi завоювала репутацію завдяки простоті в роботі. Він написаний на C++ і призначений для використання в основному для досліджень акустичного моделювання.

Особливості:

– підтримує повні коваріаційні структури разом із модулями Гаусової суміші разом із діагоналлю;

– утримує MMI та посилений MMU;

– інтеграція на рівні коду з перетворювачами кінцевого стану накопичується на основі набору інструментів Openfst;

– володіє інструментами для зміни LM в стандартному форматі ARPA на FST;

– користується підтримкою загальної лінійної алгебри разом із матричною бібліотекою, яка обгортає стандартні підпрограми базової лінійної алгебри та підпрограми пакету лінійної алгебри;

– розширювана конструкція з дискримінаційним навчанням у просторі;

– пропонує повні засоби та глибокі нейронні мережі;

– використовуйте лінійну регресію максимальної правдоподібності для підтримки адаптації простору моделі та використовуйте лінійну регресію

максимальної правдоподібності простору ознак для підтримки адаптації простору ознак [31 - 32].

Коротка форма CMUSphinx – Sphinx. Це незалежний від мовця розпізнавач безперервного мовлення з великим словником, який випущений під ліцензією BSD. Це група систем розпізнавання мовлення, розроблена Університетом Карнегі-Меллона.

Всі пакетів можна знайти з відкритим вихідним кодом та використовувати безплатно дане програмному забезпечення для розпізнаванні мовлення. Кожен з них призначений для різних типів завдань і додатків.

– Pocketsphinx - легкий механізм розпізнавання мовлення, написаний на C. Він спеціально розроблений для портативних і мобільних пристроїв;

– Sphinx base - містить необхідні бібліотеки, які спільно використовують тренажер CMU Sphinx, декодери Sphinx, деякі поширені утиліти, які впливають на акустичні функції та аудіофайли;

– Sphinx4 - незалежна від диктора, найсучасніша система безперервного розпізнавання мовлення, написана мовою програмування Java;

– Sphinxtrain - тренер акустичної моделі з відкритим кодом Університету Карнегі-Меллона.

Особливості набору інструментів:

– для платформ з низьким рівнем ресурсів розроблені інструменти CMUSphinx. Ефективне розпізнавання мови можливе завдяки сучасним алгоритмам розпізнавання мовлення;

– має гнучкий дизайн, зосереджений на реалістичній розробці додатків, а не на дослідженнях;

– безліч інструментів, призначених для цілей пов’язаних із розпізнаванням мовлення, як-от визначення ключових слів, оцінка вимови та вирівнювання;

– використання різних мов, як-от мандарин, голландська, німецька, російська, англійська та французька. Присутня можливість створювати нові моделі для інших мов [31, 33].

Якщо використовувати прості слова, то ви можете за допомогою інструмента Dragon NaturallySpeaking просо говорити і ваші слова з'являться на екрані. Комп’ютер підкорятиметься вашим командам. Програмне забезпечення Dragon NaturallySpeaking надасть вам відповідні рішення для ваших власних потреб.

Особливості:

– це програмне забезпечення є точним на 99% і втричі швидше, ніж набір тексту;

– це програмне забезпечення дозволяє особам створювати та обмінюватися високоякісною документацією та допомагає спрощувати складні робочі процеси. Можна стати більш продуктивними, використовуючи це програмне забезпечення;

– допомагає в кількох щоденних справах, таких як надсилання електронних листів, веб-серфінг, диктування домашніх завдань;

– працівники та малі підприємства можуть створювати та транскрибувати документи через Dragon Professional;

– синхронізація можлива з Dragon Anywhere;

– завдяки швидкості та точності він керує комп’ютером за допомогою голосу;

– ну і цікавим для нас, можливість використати це дане програмно забезпечення, як інструмент для написання коду. Можна глянути на приклад VoiceCode та Talon [31, 34].

Ще однією системою, а точніше прикладним програмним інтерфейсом є Web Speech API. Цей інтерфейс широко поширений і точно відомий багатьом

через такі додатки як Duoling, Google Translate та сама пошукова система Google. Що він вміє та чим славиться:

«…

– розпізнавання мови браузером;
– послуга перекладу;
– синтез мовлення/розповідь;
– постійне слухання;
– голосовий помічник (див. нижче);
– голосовий пошук. …» [35].

Він підтримує тільки деякі браузери, зверніть увагу на Таблицю 1.2. Як працює ця система та яку має архітектуру? Отримані аудіо дані будуть відправлятися на сервери Cloud Google Speech-to-Text і там оброблятися. На даний момент саме Google є головним гравцем в цій галузі та має можливість розпізнавати більше як ста мов.

Перелік основних доступних мов для розпізнавання мовлення:

– турецька;
– українська;
– шведська;
– тамільська (Індія);
– іспанська;
– словенська;
– німецька;
– португальська;
– польська;
– норвезька;
– корейська;
– японська;

– італійська і багато інших.

Перед тим як відправляти ці дані на сервери Google, Mozilla спершу маршрутизує їх через проксі-сервер нашого власного сервера (рисунок 2.3) щоб позбавити їх від інформації про користувача. «…Це має на меті зробити недоцільним для Google пов'язувати такі запити з обліковим записом користувача лише на основі даних, які Mozilla надає у запитах на транскрипцію. …»[35]. Mozilla не дає можливості Google зберігати голосові запити, в результаті, запити користувача через браузер не будуть зберігатися та їх не можна буде використовувати для оформлення профілю користувача на невизначений термін. Наразі ліцензія на Google Cloud STT контролюється і підтримується командою Mozilla Cloud Ops відповідно до загального контракту з Google Cloud.

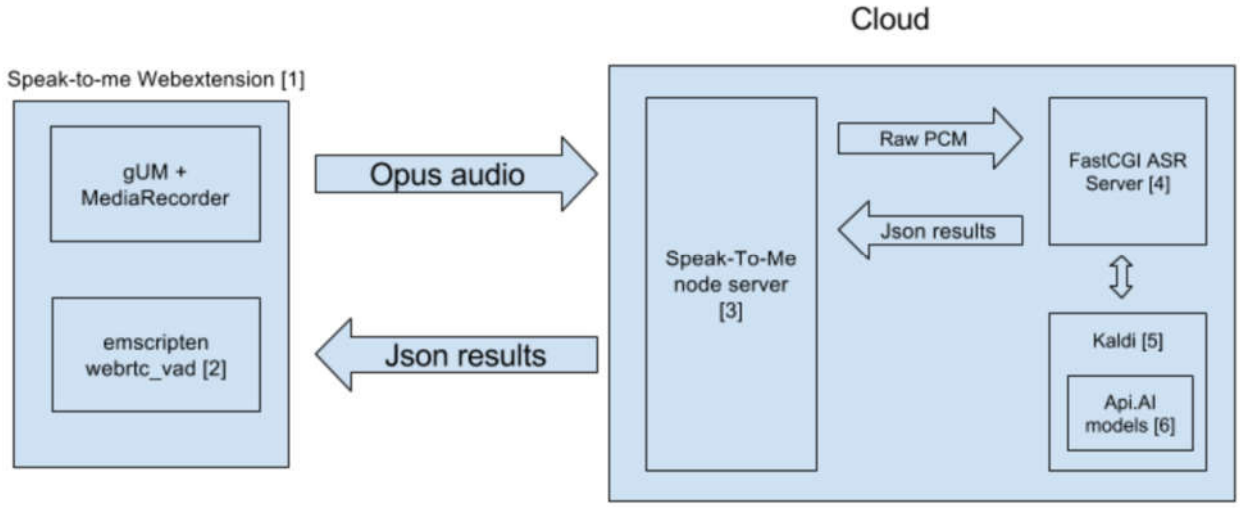


Рисунок 2.3 – Архітектура проксі сервера [36]

Mozilla на даний момент також може користуватися даними наборами даних для навчання свого API, але аудіо колекцію вимкнено за замовчування. В подальшому користувачу будуть давати вибір, чи бажає він ділитися цими даними з Mozilla.

Яким чином дані переходять між користувачем, серверами Mozilla та Google Cloud? Всім back-end-ом керує Mozilla Cloudops та команда, яка

відповідає за роботу цих служб. На рисунку 2.4 відображається архітектура роботи Web Speech API.

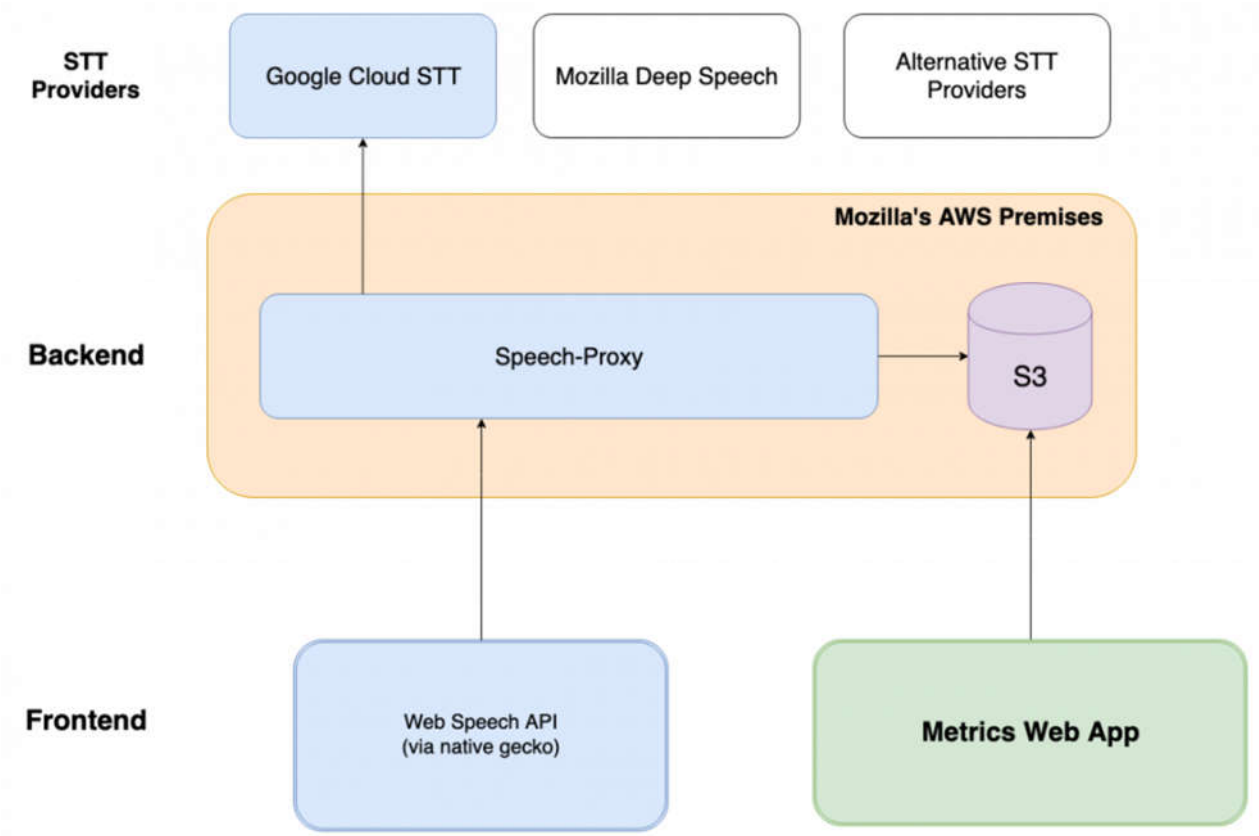


Рисунок 2.4 – Архітектура Web Speech API [37]

Відповідно до мізерних потреб, які диктує розробка, ця система буде гармонувати ідеально. Для розпізнавання буде достатньо одної англійської мови, на плечі якої ляже вантаж розпізнавання всіх вхідних команд [35].

## 2.3 Алгоритм інтеграції системи розпізнавання мовлення Web Speech API в середовище для написання програмного коду

Процес інтеграції системи розпізнавання мовлення в розширення на основі Visual Studio Code та Web Speech API в результаті дасть корисний інструмент, який можна буде налаштовувати для потреб. Від звичайного вводу тексту для верстання веб сторінок до виконання складних процедур. Яким чином буде відбуватися цей алгоритм?

Самого початку необхідною ланкою буде встановлення Visual Studio Code, саме для нього буде відбуватися розробка розширення. VS Code можна завантажити з їх офіційного сайту: https://code.visualstudio.com. Завантажити необхідно версію для Linux Ubuntu, на цій операційні системі буде відбуватися весь процес розробки. Ще одним швидким методом являється встановлення через термінал операційної системи Ubuntu, але для цього перед виконанням цієї команди потрібно встановити сам snap (рисунок 2.5):

```
$ sudo apt update
$ sudo apt install snapd
$ snap install --classic code
```

Рисунок 2.5 – Встановлення snap та VS Code[38]

Ще є один довший метод, який представлений на рисунку 2.6. Кожну команду необхідно виконувати по черзі. Тільки якщо wget не встановлений на операційні системі, його також прийдеться встановити:

```
$ sudo apt update
$ sudo apt-get install wget
$ sudo apt install software-properties-common apt-transport-https wget
$ wget -q https://packages.microsoft.com/keys/microsoft.asc -O- | sudo apt-key add –
$ sudo add-apt-repository "deb [arch=amd64] https://packages.microsoft.com/repos/vscode stable main"
$ sudo apt install code
$ sudo apt update
$ sudo apt upgrade
```

Рисунок 2.6 – Перелік команд для встановлення Visual Studio Code[39]

Наступним кроком буде встановлення залежностей – це розширення, які мають в своїй бібліотеці перелік необхідних методів або функцій. Першою залежністю буде Git – система управління версіями з розподіленою архітектурою. Вона знадобиться для вивантаження проекту на GitHub. Немає нічого простішого, ніж встановлення Git (рисунок 2.7). Після успішної інсталяції потрібно перевірити коректність встановлення. Якщо результатом буде: «git version» і номер версії, то встановлення пройшло успішно.

```
$ sudo apt install git-all
$ git --version
```

Рисунок 2.7 – Команди для встановлення і перевірки Git

Тепер необхідно провести налаштування git. Для цього потрібно зареєструвати обліковий запис на GitHub. Далі в командному рядку потрібно ініціалізувати щойно створеного користувача через команди на рисунку 2.8.

```
$ git config --global user.name "User_name"
$ git config --global user.email Useremail@gmail.com
```

Рисунок 2.8 – Команди для ініціалізації користувача

Ще одною необхідною ланкою є встановлення Node.js. Там є пакетний менеджер, який допоможе встановити інші залежності. Встановлення проходить через такий перелік команд як: sudo apt update, sudo apt upgrade та sudo apt install node.js. Для перевірки потрібно в термінали написати: nodejs -v. Тепер встановлення самого пакетного менеджера: sudo apt install npm.

Наступним кроком буде побудова розширення, а почнемо з створення пустого проекту через Visual Studio Code та набирання команд в терміналі середовища:

– npm install -g yo – встановлення Yeoman для встановлення шаблону розширення;

– npm install -g yo generator-code – створення самого шаблону розширення.

Тепер потрібно встановити оболонку проекту. Ця бібліотека розміщується на GitHub. Для її інтегрування в проект є декілька методів. Можна просто в терміналі VS Code написати: npm install artyom.js або знайти її на GitHub і завантажити архів з розширенням *.zip та розпакувати в проекті.

В готовому середовищі пишеться код для реагування на голосові команди, в нього входить створення екземпляра класу Artyom, налаштування мови, та код, який буде відповідати за реакцію на розпізнане слово. Для реакції команд на голос потрібний сервер, який буде опрацьовувати ці команди, без нього наприклад функція для створення файлу не буде працювати.

Для написання сервера знадобиться створити ще один проект і встановити для нього програмний каркас для Node.js – Express. Він буде отримувати і відправляти команди через POST та GET запити. З основного проекту буде відправлятися POST запит і уже сервер буде його отримувати і обробляти. Встановлення проходить через виконання команди: npm install express –save.

Висновки до розділу 2

1. Проведено детальний опис концептуальних основ інтегрування системи розпізнавання мовлення. Описано всі види та основну ідею розширень для середовища написання програмного коду. Представлено актуальні та наглядні приклади підняття продуктивності через використання даної функції. Відображено перелік невід'ємних фрагментів, які входять в кроки інтеграції алгоритму.

2. Досліджено різні варіанти систем автоматичного розпізнавання мовлення. Проведено порівняльний аналіз характеристик даних систем за допомогою детального опису всього функціоналу. Визначено переваги та недоліки поширених систем розпізнавання мовлення, та вибір кращого варіанту із списку запропонованих для інтегрування в розроблювану систему.

3. Вдосконалено алгоритм інтеграції системи розпізнавання мовлення в середовище для написання програмного коду шляхом побудови розширення, який на відміну від інших дозволяє інтегрувати новітні автоматизовані системи розпізнавання мовлення не порушуючи існуючу архітектуру системи. Даний алгоритм розроблено на основі вибраного прикладного програмного інтерфейсу Web Speech від Mozilla Corporation.

# 3 РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ АЛГОРИТМУ ІНТЕГРАЦІЇ СИСТЕМИ РОЗПІЗНАВАННЯ МОВЛЕННЯ В СЕРЕДОВИЩЕ ДЛЯ НАПИСАННЯ ПРОГРАМНОГО КОДУ

## 3.1 Архітектура інтегрованої системи

Архітектура розробленого проекту є достатньо складною через використання посереднього API від Mozilla та оригінального двигуна по розпізнаванню голосу від Google Cloud. Для початку потрібно відобразити повністю самодостатню систему, яка включає багато етапів. Вона складається з публічної сторони, тобто Front-end, сторони серверів – Back-end та провайдера послуг у лиці Google Cloud Speech-to-Text. Для повної картини використаємо архітектуру Web Speech API та інтегруємо нашу частину (рисунок 3.1).

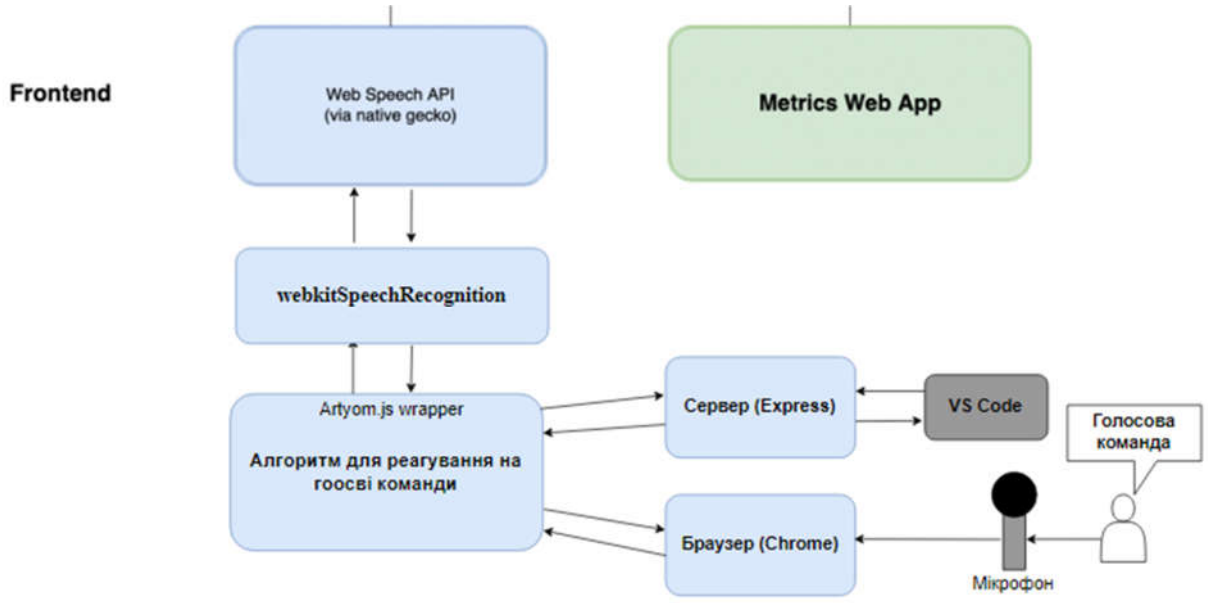


Рисунок 3.1 – Архітектура Front-end (та частина Back-end у вигляді сервера на Express)

Говорити окремо про Back-end та Front-end неможливо, бо це один цілий організм, який ніяким чином працювати без якоїсь частинки не буде, тому для

опису роботи архітектури необхідно почати з самого початку, а саме з вимовлення диктором команди. Для зручності архітектура буде розділена на дві частини (рисунок 3.1 та рисунок 3.2).

Голосова команда буде подаватися на вбудований або дискретний мікрофон пристрою. Весь перелік голосової активності приймає на себе веб-браузер, у нашому випадку – браузер Google Chrome. Для коректної роботи необхідно дати певні дозволи браузеру через налаштування. До цих дозволів відноситься:

– дозвіл на мікрофон;
– дозвіл на виконання JavaScript;
– дозвіл на використання динаміків.

Голосова активність відображається в консолі встановленого браузера. Там відображаються слова, які система змогла розпізнати та назви команд які будуть виконані, якщо система знайшла зарезервоване слово.

Якщо забігати вперед, то всі розпізнані слова попадають на Front-end частину системи. До Front-end частини відноситься обгортка Artyom.js, та так звані зарезервовані слова, при розпізнаванні яких, система буде давати якусь реакцію. Це похоже на умовний оператор “if else” або навіть switch. Умовний оператор “if else” має тільки одну умову, після якої виконується тіло, а якщо умова не виконається, тоді командування переходить на оператора else. В switch структура більше схожа на алгоритм по знаходженню співпадаючого слова. В операторі switch є одна змінна в умові, яка проходить по всім випадкам або case. Якщо змінна співпала припустимо з третім випадком, тоді виконається його тіло. Так само працює і розпізнавання команд в алгоритмі, розпізнане слово проходить по всім випадкам, якщо є співпадіння, тоді дане тіло умови буде виконуватися.

Насправді запис голосу спочатку дійсно проходить через Artyom.js, але попадає одразу на API. Тільки дане API має дві функціональності для роботи з

голосом – це розпізнавання голосу, за нього відповідає інтерфейс SpeechRecognition та префікса webspeechrecognition для комп'ютерних версій (є ще можливість виклакати його для мобільних додатків через прапор media.webspeech.recognition.enable) та інтерфейса для синтезу мовлення SpeechSynthesis (дозволяє штучному диктору прочитати даний вами текст).

Для проекту цікавим буде тільки інтерфейс SpeechRecognition та префікс webspeechrecognition, хоча для розвитку, удосконалення системи та надання нових цікавих функцій завжди можна буде дописати можливість диктування вашого тексту, тільки це буде більше пасувати для голосового асистента, який зможе відповідати на ваші команди голосом ніж для системи написання коду.

Переданий голос на Web Speech API переходить на частину Back-end. Які кроки там виконуються, щоб записаний звук повернувся розпізнаним текстом (рисунок 3.2)?

Типовий запит виконується за наведеними нижче кроками:

– код Web Speech API у браузері відповідає за запит користувача на дозвіл запису з мікрофона, визначення закінчення розмови та передачу даних на проксі-сервер. Є чотири заголовки, які користувач може використовувати для зміни поведінки проксі-сервера:

1) Accept-Language-STT: визначає мову, яка має бути декодована службою Google Cloud Speech-to-Text;

2) Store-Sample: визначає, чи дозволяє користувач зберігати зразок аудіо на серверах Mozilla для подальшого використання (наприклад для навчання моделей для їх власної системи розпізнавання Deep Speech);

3) Store-Transcription: визначає, чи дозволяє користувач зберігати транскрипцію на серверах Mozilla для подальшого використання (також для навчання їх власної системи);

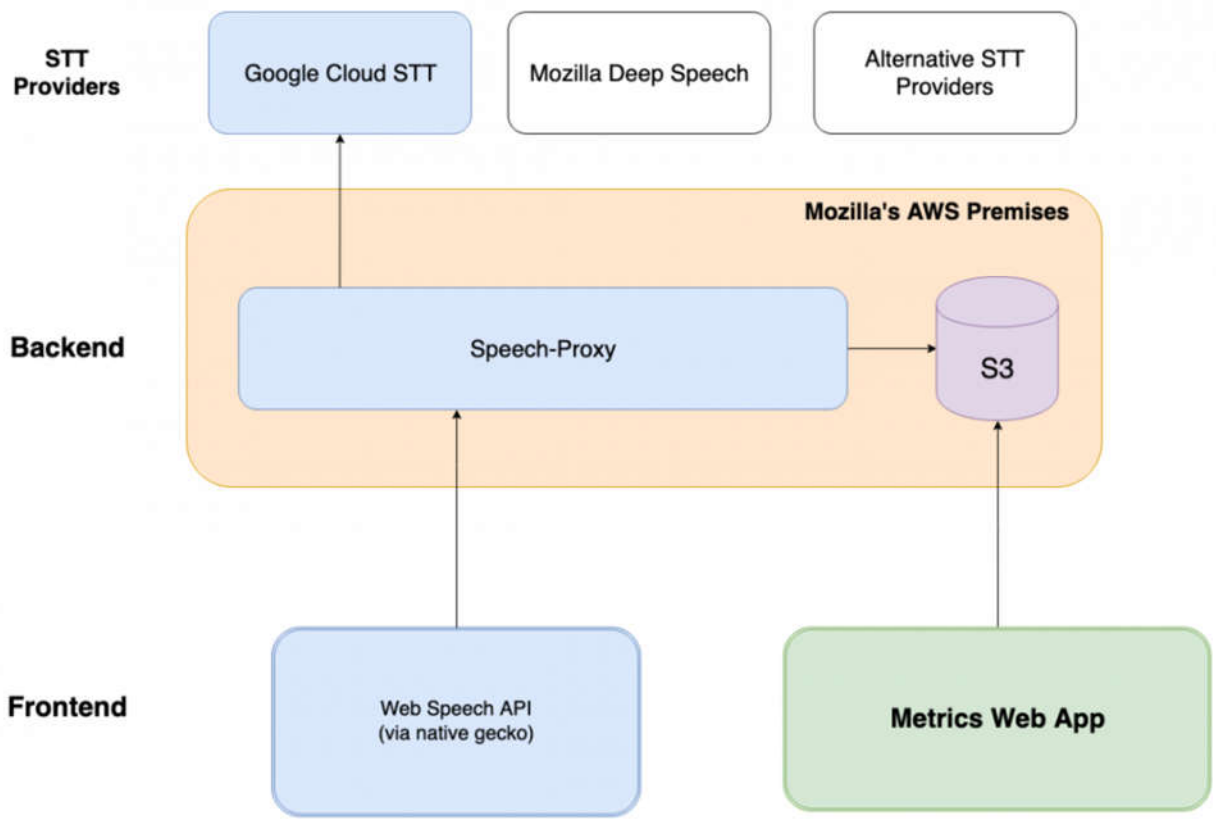


Рисунок 3.2 – Архітектура Back-end та посередника між Front та Back-end у вигляді API Web Speech [37]

4) Product-Tag: визначає, який продукт використовує API. Це може бути для голосового заповнення, для Firefox Reality, та для Web Speech API та багато інших.

– коли проксі-сервер отримує запит із зразком аудіо, він шукає встановлені заголовки. Не зберігається нічого, крім того, що було запитано користувачем, а також позначка часу та користувацький агент;

– потім проксі-сервер шукає формат файлу та декодує його в необроблений *.pcm.

– наступний крок – це передача запиту до постачальників Speech-to-Text (постачальників, тому що є можливість використовувати не лише Google Cloud), встановленого у файлі конфігурації проксі, що містить лише аудіофайл;

– після того, як постачальник Speech-to-Text повертає запит, що містить транскрипцію та оцінку точності, він пересилається клієнту [35].

Використання проксі-сервера не є простою забаганкою Mozilla, в ньому прихований певний зміст та принципи Mozilla. Існують як технічні, так і практичні причини інтегрування проксі-сервера в цю систему. Mozilla хотіли володіти гнучкістю системи для перенаправлення запитів користувачів на різні служби Speech-to-Text без зміни коду клієнта, а також мати єдиний протокол, який буде використовуватися в усіх проектах Mozilla. Найбільш вигідною причиною було збереження анонімності користувачів, коли є потреба у використанні стороннього постачальника. У цьому випадку запити до провайдера здійснюються з сервера Mozilla і для отримання транскрипції подається лише зразок аудіо. Переваги маршрутизації даних через проксі-сервер:

– перед тим як надсилати дані будь-якому сторонньому постачальнику Speech-to-Text, Mozilla видаляє ідентифікаційну інформацію користувача та робить запит до постачальника анонімним;

– коли системі потрібно використовувати сторонні або платні послуги Speech-to-Text, не потрібно надсилати ключ служби разом із кодом клієнта;

– централізація та передача запитів через проксі-сервери зменшує ймовірність зловживань з боку клієнтів і дозволяє впроваджувати такі механізми, як обмеження, чорні списки тощо;

– можливість перемикатися між службами Speech-to-Text в режимі реального часу і перенаправляти запит на будь-яку послугу, яку вибере Mozilla, не змінюючи жодного коду в клієнті;

– можливість підтримувати послуги Speech-to-Text як локально, так і поза межами, не маючи жодної додаткової логіки в клієнті;

– можливість централізувати всі запити, що надходять від різних продуктів, в єдину кінцеву точку мовлення, що полегшує вимірювання механізмів як кількісних, так і якісних.

– можливість підтримувати різні аудіоформати не додаючи клієнтам додаткової логіки. Це все не залежить від формату підтримуваного постачальником Speech-to-Text, наприклад додавання стиснення або потокової передачі між клієнтом і проксі-сервером;

– якщо користувачі бажають зробити свій внесок у Mozilla і дозволити зберігати свої набори даних, то це є можливим, включаючи відмову надсилання їх третім сторонам [35].

Після повернення транскрипції назад до клієнта, вона надходить в оболонку Artyom.js, де шукає відповідні зарезервовані слова, які відповідають за виконання команд. Якщо слово успішно знайшло співпадіння, тоді за ним слідує код, який перенаправить запит на сервер (знайти на рисунку 3.1). Сервер виконає відповідні інструкції і результат відображатиметься в середовищі VS Code.

В результаті архітектура повноцінної системи має такий вигляд (рисунок 3.3).

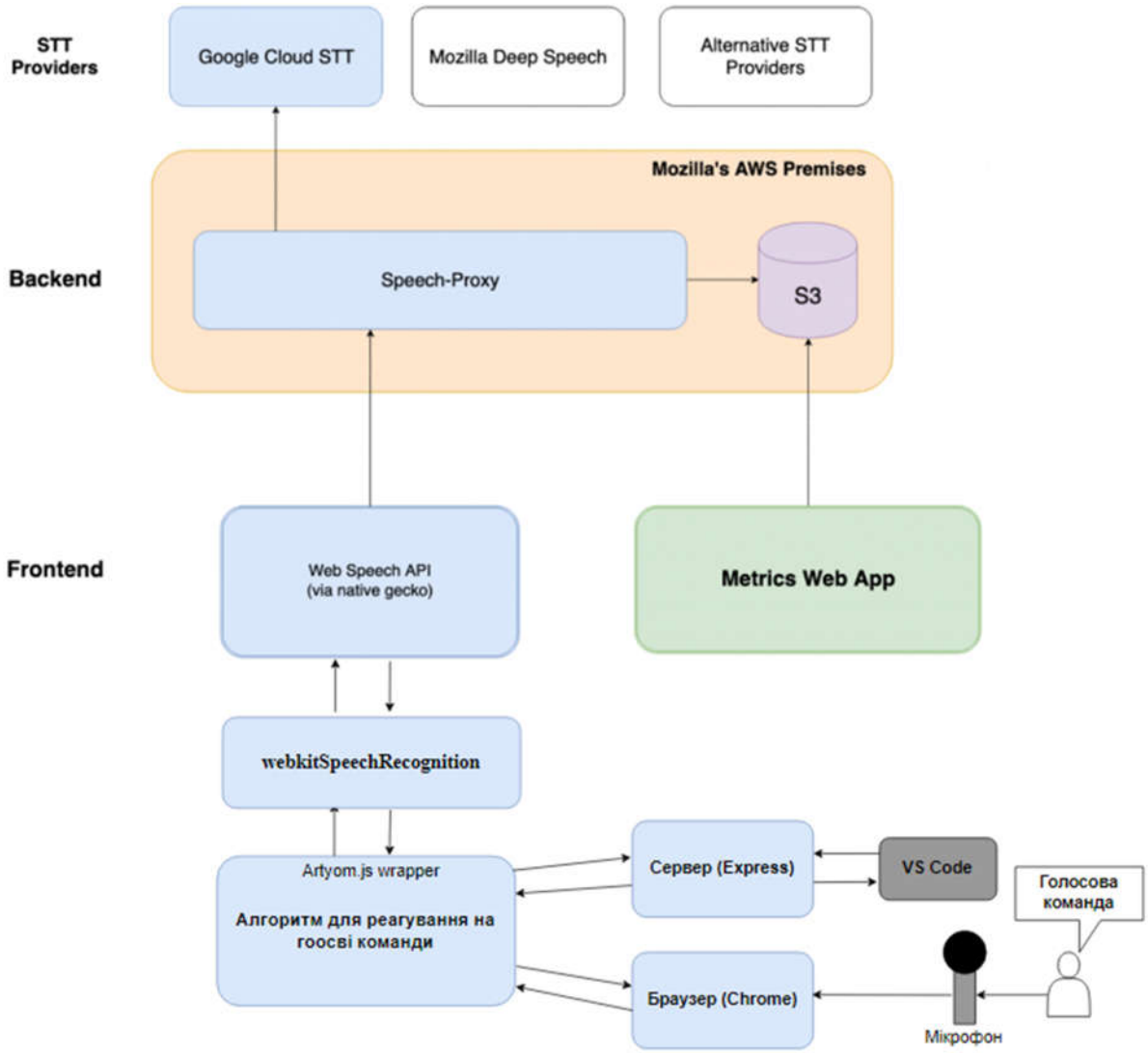


Рисунок 3.3 – Цілісна архітектура системи

3.2 Реалізація алгоритму інтеграції системи розпізнавання мовлення в середовище для написання програмного коду

Код починається з базового файлу – index.html. В ньому знаходить базова HTML розмітка (рисунок 3.4):

– заголовок з текстом “test”;

– тег “html” з атрибутом lang = “en” для коректного відображення англійської мови та спеціальних символів;

– тес “script” в якому знаходиться посилання на файл JavaScript, який потрібно підключити до цього файлу. В нашому прикладі, це файл від бібліотеки Artyom.js;

– тег “link” в якому знаходиться посилання на файл з стилями style.css;

index.html > ...

```
1   <!DOCTYPE html>
2   <html lang="en" >
3     <head>
4       <script src='node_modules/artyom.js/build/artyom.window.min.js'>
5     </script>
6     <meta charset="UTF-8">
7     <title>test</title>
8   <link rel="stylesheet" href="style.css">
9
10  </head>
11  <body>
12  <h1>Artyom</h1>
13  <p id="recognised"></p>
14  <script type="module" src="bundle.js"></script>
15
16  </body>
17  </html>
18
```

Рисунок 3.4 – Вміст файлу index.html

– тег “p” із атрибутом “id”. Цей ідентифікатор вказаний для того, щоб можна на нього посилатися через і мати можливість керувати ним.

– ще один повтор тегу “script”. На цей раз в ньому знаходиться код з командами і ініціалізація запуску системи розпізнавання мовлення.

В проекті присутній файл з переліком залежностей(див. Рисунок 3.5), які встановлені в проекті. Цей файл важливо тримати в парі з .gitignore. Якщо буде

бажання опублікувати цей проект на GitHub, то така можливість є через виконання переліку команд в терміналі середовища Visual Studio:

– git add . - Після того, як був встановлений Git, VS Code реєструє всі зміни в коді. Для вибору на відправлення цих змін на репозиторій GitHub і потрібно писати цю команду. “.” в кінці команди означає вибір всіх файлів для надсилання.

– git status — відбражає статус відібраних файлів;

– git commit -m “My first commit” — назва коміта, ця назва буде застосована для всіх файлів, які були змінені і позначені в команді add.

– git push — остаточне публікування коду.

Отож, щоб не публікувати великі об’єми встановлених бібліотек, назви папок і файлів записують в файл .gitignore. Таким чином перелік папок з цього файлу не будуть публікуватися в репозиторії GitHub для економії місця. Якщо вам потрібно скачати цей проект, то його необхідно просто клонувати відповідною командою (git clone посилання на репозиторій) і після клонування написати команду npm -i, яка самостійно встановить всі залежності посилаючись на файл package.json.


Рисунок 3.5 – Перелік залежностей

Файл bundle.js, який уже з’являвся в index.html. Саме він відповідає за початкові налаштування роботи системи розпізнавання. На Рисунку 3.6

відображено файл script.js, це не є помилкою. Все через програмні проблеми з імпортуванням класів і використання бібліотеки fs для роботою над файлами. Для вирішення цієї проблеми буде досягнуто вирішення за допомогою встановлення бібліотеки browserify-fs. Вона дає можливість побороти цю проблему через babel. Babel — це транспайлер, який переписує код нового формату JS на пізній. В результаті код на рисунку 3.6 перетворюється на код на рисунку 3.7.

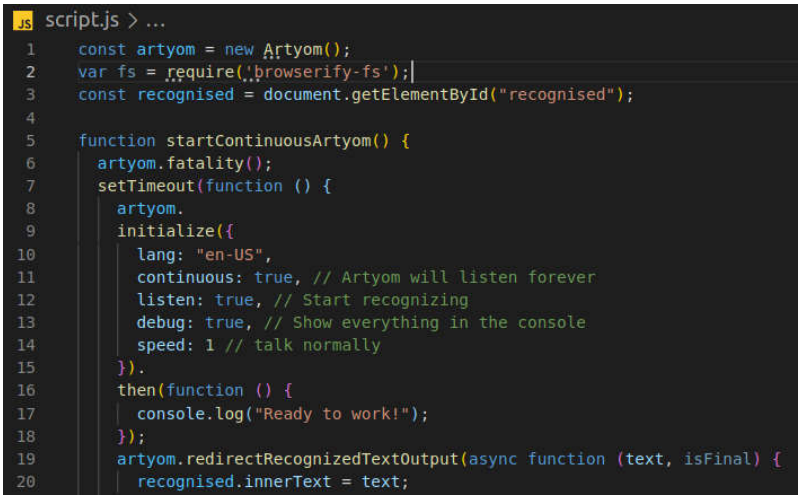


Рисунок .3.6 – Код налаштування роботи оболонки Artyom.js


Рисунок 3.7 – Транскопільований код файлу script.js

Показати весь перелік команд файлу bundle,js неможливо, тому що він сягає неймовірного розміру. Кількість стрічок коду рівна числу 23519. Транскомпіляція файлу виконується командою: browserify script.js -o bundle.js.

Файл bundle.js можна проігнорувати, можна поки повернутися до змісту файла script.js. В ньому ми створюємо новий екземпляр класу Artyom і отримуємо всі його публічні методи. Наступні стрічки коду відповідають за початкові налаштування. Для прикладу, яка мова розпізнавання, час прослуховування команд, відображення всіх подій в консолі і так далі. На наступному Рисунку 3.8 знаходиться продовження коду. В ньому знаходиться масив слів, на які буде реагувати система. Код реакції на команду знаходиться в тілі анонімної і асинхронної функції.

```
var commands = [
  {
      indexes: ["script"],
      action:  function(){
        artyom.redirectRecognizedTextOutput(async function (text, isFinal) {
          await fetch("http://localhost:3234", {
            method: 'POST',
            headers: {
              'Content-Type': 'application/json'
            },
            body:  JSON.stringify({ 'voiceText':text  })
          });
        })

      }},
  {
      indexes: ["create"],
      action: function(){
        artyom.redirectRecognizedTextOutput(async function (text, isFinal) {
          await fetch("http://localhost:3234", {
            method: 'POST',
            headers: {
              'Content-Type': 'application/json'
            },
            body:  JSON.stringify({ 'voiceText':text  })
          });
        })
      }
  },
  {
```

Рисунок 3.8 – Масив команд

Наступними будуть файли сервера написаного на програмному каркасі Express. В цьому окремому проекті також знаходиться файл з залежностями. Там

встановлений програмний каркас Express, Body-parser та бібліотека file-saver для різного роду маніпулюванням файлами. Останнім головним файлом буде index.js. В ньому знаходяться основні налаштування і команди для реагування на запити від Front-end (рисунок 3.9).

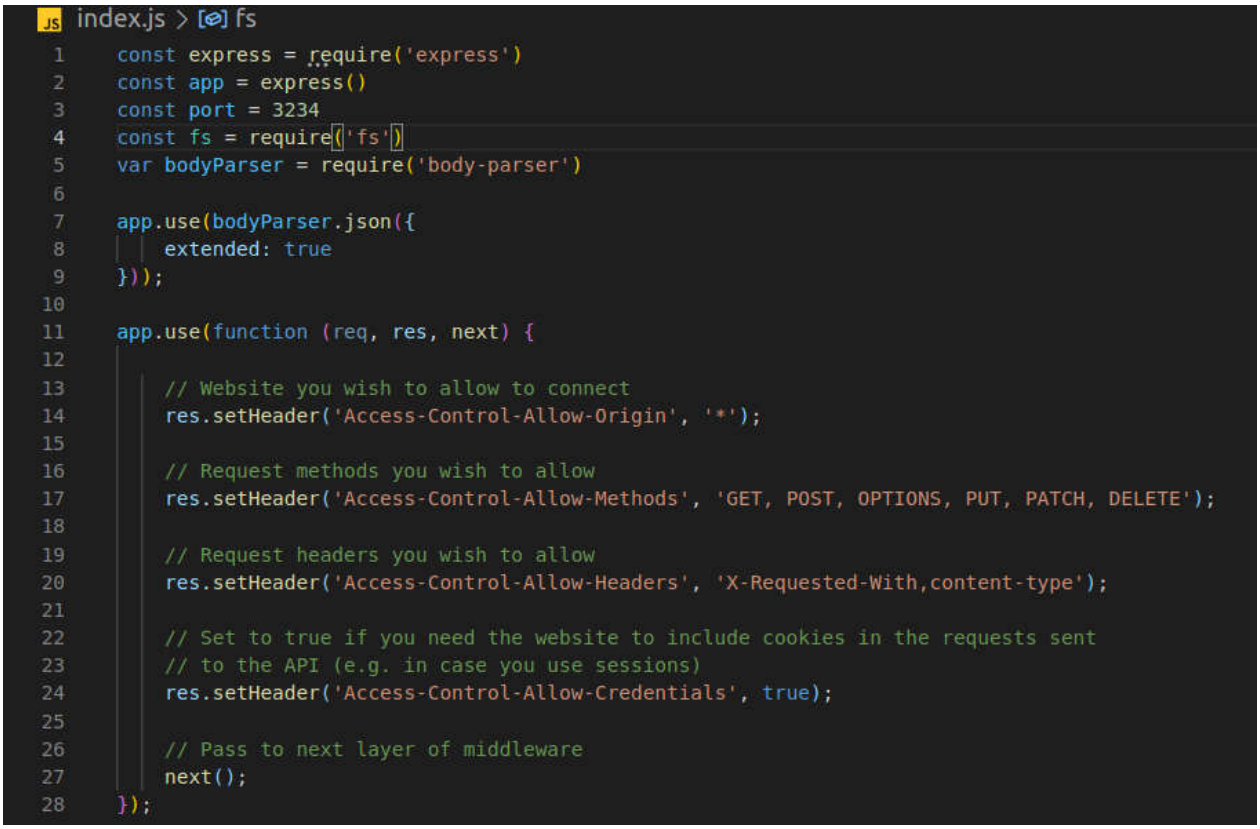


Рисунок 3.9 – Список налаштувань для сервера

До базових налаштувань сервера відносяться:

– номер порта;
– дозвіл на використання cookie;
– дозвіл на виконання методів;
– дозвіл ресурсу, які знаходяться на домені відмінного від основного.

Без написання цих заголовків, звернення до сервера буде проблематичним. Відправляти запити буде неможливо і в результаті будуть відображатися

помилки, які стосуються відомої проблеми під назвою CORS. Точніше це не проблема, а спеціальна політика, яка забороняє виконувати запити між різними джерелами.

В наступній частині файлу знаходяться стрічки коду, які відповідають за створення файлів і наповненням їх кодом. Якщо прийшов запит від кореневого домену, тоді буде виковуватися перелік команд відображених на рисунку 3.10.

```
app.post('/', (req, res) => {

  if (req.body.voiceText == 'hyper') {
    fs.open('testFile.html', 'w', (err) => {
      if(err) throw err;
      console.log('File created');
  });

  }
  if (req.body.voiceText == 'create') {
    fs.writeFile('testFile.html', "<!DOCTYPE html>
    <html lang='en'> <head> <meta charset='UTF-8'>
    <title>Document</title> </head> <body> </body></
    html>", err => {
      console.log(req.body.voiceText);
        if (err) {
          console.error(err)
          return
        }
      })
  }

  if (req.body.voiceText == 'script') {
    fs.open('javascript.js', 'w', (err) => {
      if(err) throw err;
      console.log('File created');
    })
  }


})

app.listen(port, () => {
  console.log(`Example app listening at http://
  localhost:${port}`)
})
```

Рисунок 3. 10 – Команди для створення і наповнення файлів

В тілі запиту, який приходить від Front-end знаходиться змінна voiceText в якій знаходиться розпізнане слово. Якщо воно збігається з умовою, тоді виконується тіло умовоного оператора. Наприклад при отриманні запиту з словом “script” — алгоритм створить файл javascript.js. Для запуску сервера потрібно ввести команду в терміналі: node index.js.

## 3.3 Експериментальні дослідження

Прийшов час протестувати проект. Для цього ми будемо порівнювати фактичний і очікуваний результат. Спочатку поглянемо на деревоподібну структуру проекту (рисунок 3.11). В ній немає бути файлів з розширенням *.html та файлу testFile.js.

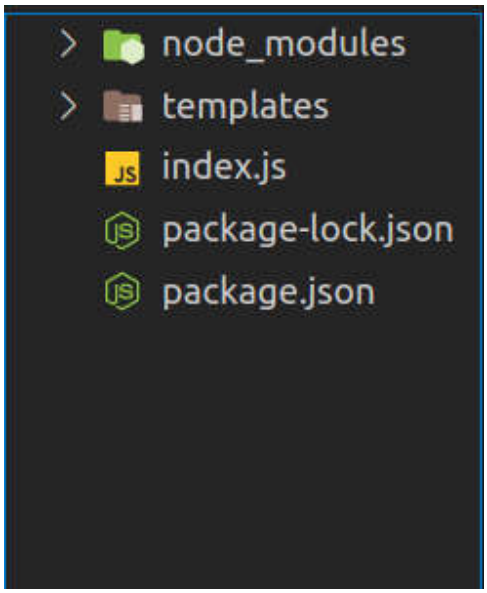


Рисунок 3.11 – Деревоподібна структура проекту

Пересвідчившись, що проект пустий, можна починати тестування системи. Тестувати ми будемо на локальній машині за допомогою встановленого сервера

Apache та Node.js. Для початку нам потрібно в браузері дати дозвіл на виконання сценаріїв та дозвіл на запис звуку з мікрофона (рисунок 3.12).

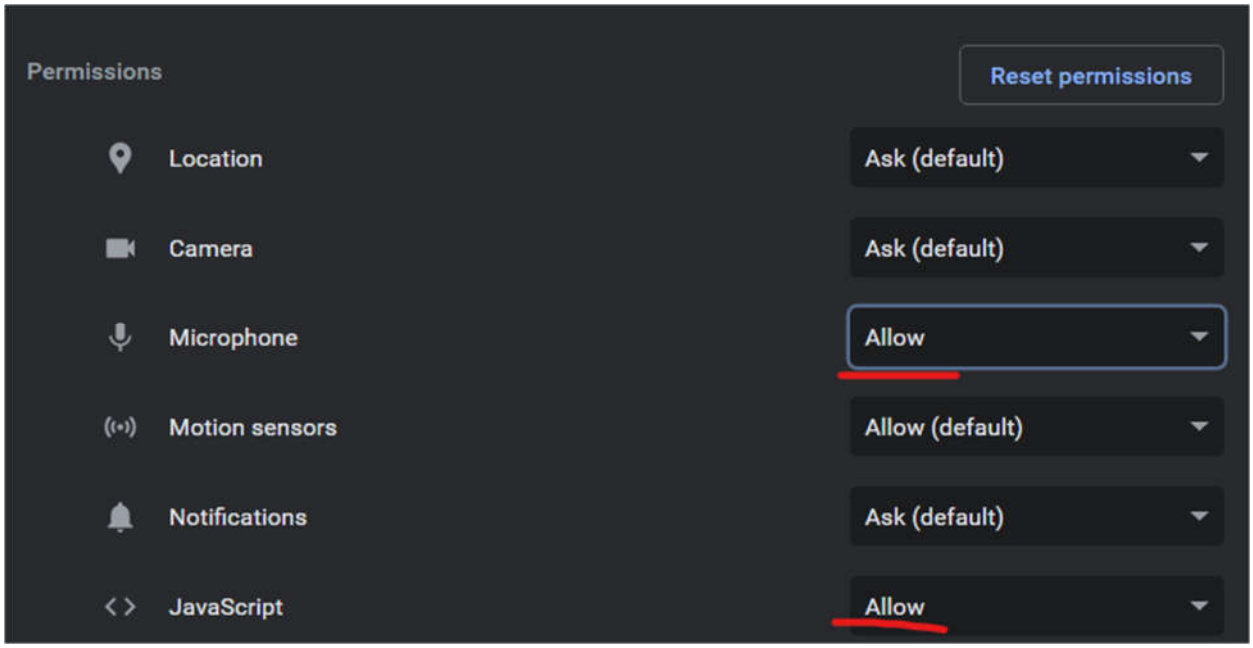


Рисунок 3.12 – Необхідні дозволи для роботи системи

Наступним етапом буде запуск Front-end та Back-end частини. Ще одною основною вимогою буде підключення до мережі Інтернет, сервіс розпізнавання голосу працює на серверах Google. Запустити сервер можна за допомогою введеної команди в терміналі середовища кореневої папки ($ node script.js). Локальний сервер запущений і чекає запитів. Щоб запустити Front-end потрібно написати в адресній стрічці браузера домен локального сервера: http://localhost або 127.0.0.1. Все залежить від налаштувань сервера Apache. Тепер можна вимовляти команди і спостерігати за результатами. Для початку спробуємо створити файл javascript.js. Це має бути виконано через голосову команду “script”. Результат відображено на рисунку 3.13.

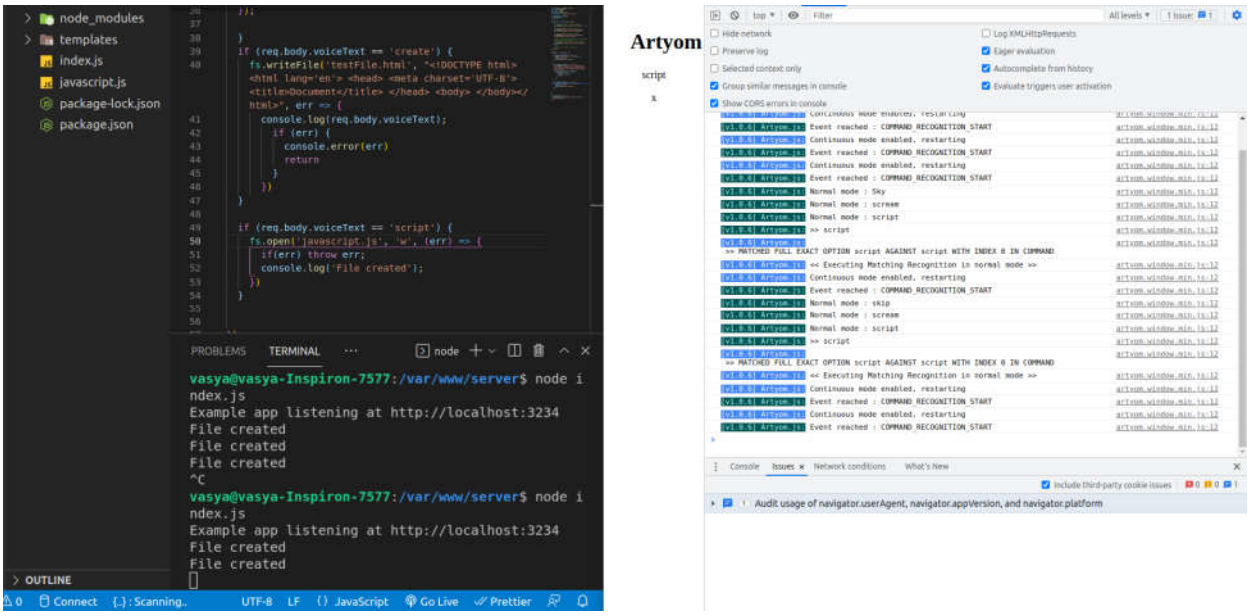


Рисунок 3.13 – Результат голосової команди «script»

Згідно рисунка ми бачимо не найкращу точність, але програмі таки вдалось створити файл. З консолі браузера видно що слово було розпізнане і воно є зарезервованим. В терміналі VS Code також є звіт, який показує успішне створення файлу. Також цьому свідчить присутність в кореневій теці проекту файл з назвою javascript.js.

Наступним тестом перевіримо створення і запис у файл testFile.html. Для створення файлу з розширенням *.html необхідно озвучити команду “hyper”. Після розпізнавання голосу має виконатися код, який за допомогою метода open створить файл testFile.html. Думаю одразу можна протестувати наповнення файлу html базовим шаблоном. Тепер ще потрібно озвучити команду “create”, яка запише в щойно створений файл частину коду. Результат можна оглянути на рисунку 3.14. Він вказує на те, що всі команди було успішно виконано. Звіт з консольної панелі браузера говорить про співпадіння слів з зарезервованими. Також можна поглянути на звіт в терміналі середовища VS Code, який каже що виконання команд було успішним (я не виправив звіти в коді і тому дві різні

команди відображають однаковий результат). І в кінці можна пересвідчитись в деревоподібній структурі проекту, яка поповнилась новим файлом testFile.html. Відкривши цей файл можна звернути увагу на те, що шаблон html було успішно в нього записано(див. рисунок 3.14).

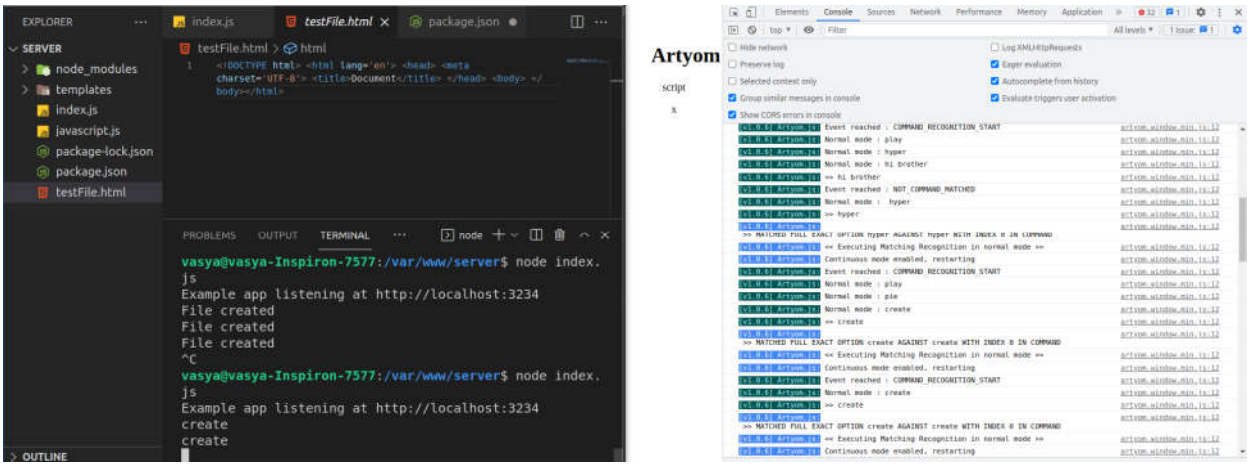


Рисунок 3.14 – Успішне виконання голосових команд “create” та “hyper”

Також можна додати за недоліки аналогів, які спростовує дана система. Багатомовність даної системи підтримується за допомогою Web Speech API та явному вказуванню в коді (рисунок 3.15).

```
function Artyom() {
    this.ArtyomCommands = [];
    this.ArtyomVoicesIdentifiers = {
        // German
        "de-DE": ["Google Deutsch", "de-DE", "de_DE"],
        // Spanish
        "es-ES": ["Google español", "es-ES", "es_ES", "es-MX", "es_MX"],
        // Italian
        "it-IT": ["Google italiano", "it-IT", "it_IT"],
        // Japanese
        "jp-JP": ["Google 日本人", "ja-JP", "ja_JP"],
        // English USA
        "en-US": ["Google US English", "en-US", "en_US"],
        // English UK
        "en-GB": ["Google UK English Male", "Google UK English Female", "en-GB", "en_GB"],
        // Brazilian Portuguese
        "pt-BR": ["Google português do Brasil", "pt-PT", "pt-BR", "pt_PT", "pt_BR"],
        // Portugal Portuguese
        // Note: in desktop, there's no voice for portugal Portuguese
        "pt-PT": ["Google português do Brasil", "pt-PT", "pt_PT"],
        // Russian
        "ru-RU": ["Google русский", "ru-RU", "ru_RU"],
        // Dutch (holland)
        "nl-NL": ["Google Nederlands", "nl-NL", "nl_NL"],
        // French
        "fr-FR": ["Google français", "fr-FR", "fr_FR"],
        // Polish
        "pl-PL": ["Google polski", "pl-PL", "pl_PL"],
        // Indonesian
        "id-ID": ["Google Bahasa Indonesia", "id-ID", "id_ID"],
        // Hindi
        "hi-IN": ["Google हिन्दी", "hi-IN", "hi_IN"],
        // Mandarin Chinese
        "zh-CN": ["Google 普通话 (中国大陆) ", "zh-CN", "zh_CN"],
```

Рисунок 3.15 – Фрагмент коду з відображенням доступних мов для розпізнавання

Це велика перевага яка розкриває багато можливостей для Front-end розробників. Створення файлів також було досягнута успішно. Інтегрування

інших відомих систем для розпізнавання через прикладний програмний інтерфейс, всі функції і методи виробники цих програм публікують в документації до свого програмного продукту. Можливість додавати свої команди досягається через публічність проекту. Його можна вивантажити на платформу GitHub і використовувати як бібліотеку. Саму бібліотеку можна буде модифікувати за власним бажанням.

Висновки до розділу 3

1. Проведено детальний опис архітектури розроблювального програмного продукту для автоматичного розпізнавання мовлення на основі прикладного програмного інтерфейсу Web Speech та оболонки Artyom.js. Відображено всі кроки проходження наборів даних від диктора розмови і запису голоса диктора до виконання команд в програмному середовищі.

2. На основі розробленої архітектури та за допомогою спеціалізованих інструментів було розроблено програмний продукт для розпізнавання мовлення. Послідовно описано алгоритм виконання програми з демонстрацією коду через рисунки. Проведено експериментальні дослідження на розробленій системі. Відображено результати даних досліджень

3. Розроблено програмний продукт, який виконує заплановані вимоги. За допомогою голосу та розробленого програмного забезпечення було створено файли з розширенням *.js та *.html. Також було введено програмний код за допомогою голосової команди в створений файл *.html.

# ВИСНОВКИ

В результаті написання кваліфікаційної роботи вирішено актуальну задачу інтеграції систем розпізнавання мовлення в середовище для написання програмного коду.

Отримано такі основні результати:

1. Проведено дослідження області розпізнавання мовлення для написання програмного коду. Описано застосування даної технології в побуті, сфері охорони здоров’я, бізнесі та інших галузях. Відображено вирішення проблем через використання даної інновації. Показано важкість навчання таких систем через глибокі нейронні мережі, що вимагають великі об’єми наборів аудіо даних та потужні електронно-обчислювальні машини.

2. Запропоновано вдосконалити алгоритм інтеграції системи автоматичного розпізнавання мовлення в середовище написання програмного коду через метод пошуку недоліків. Виконано порівняльний аналіз одних із самих популярних аналогів систем для написання програмного коду голосом. Інтеграція інноваційних систем розпізнавання мовлення та універсального середовища, яке має можливість до адаптації під нові вимоги. Для прикладу – багатомовність. Детально описано переваги та недоліки вибраної системи в порівнянні з аналогами.

3. Універсальний алгоритм для інтеграції системи автоматичного розпізнавання мовлення в середовище написання програмного коду дасть можливість створювати файли з доступними форматами. Також спроможність голосом вводити код у створені файли.

4. Проведено детальний опис концептуальних основ інтегрування системи розпізнавання мовлення. Описано всі види та основну ідею розширень для середовища написання програмного коду. Представлено актуальні та наглядні приклади підняття продуктивності через використання даної функції.

Відображено перелік невід'ємних фрагментів, які входять в кроки інтеграції алгоритму.

5. Досліджено різні варіанти систем автоматичного розпізнавання мовлення. Проведено порівняльний аналіз характеристик даних систем за допомогою детального опису всього функціоналу. Визначено переваги та недоліки поширених систем розпізнавання мовлення, та вибір кращого варіанту із списку запропонованих для інтегрування в розроблювану систему.

6. Вдосконалено алгоритм інтеграції системи розпізнавання мовлення в середовище для написання програмного коду шляхом побудови розширення, який на відміну від інших дозволяє інтегрувати новітні автоматизовані системи розпізнавання мовлення не порушуючи існуючу архітектуру системи. Даний алгоритм розроблено на основі вибраного прикладного програмного інтерфейсу Web Speech від Mozilla Corporation.

7. Проведено детальний опис архітектури розроблювального програмного продукту для автоматичного розпізнавання мовлення на основі прикладного програмного інтерфейсу Web Speech та оболонки Artyom.js. Відображено всі кроки проходження наборів даних від диктора розмови і запису голоса диктора до виконання команд в програмному середовищі.

8. На основі розробленої архітектури та за допомогою спеціалізованих інструментів було розроблено програмний продукт для розпізнавання мовлення. Послідовно описано алгоритм виконання програми з демонстрацією коду через рисунки. Проведено експериментальні дослідження на розробленій системі. Відображено результати даних досліджень

9. Розроблено програмний продукт, який виконує заплановані вимоги. За допомогою голосу та розробленого програмного забезпечення було створено файли з розширенням *.js та *.html. Також було введено програмний код за допомогою голосової команди в створений файл *.html.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ткаченко В. Застосування систем розпізнавання мовлення. URL: https://www.victka.net/node/36.

2. Кращі програми для голосового набору тексту. URI: http://smartandyoung.com.ua/krashhi-programi-dlja-golosovogo-naboru-tekstu.

3. Saksamudre, Suman K., P. P. Shrishrimal, and R. R. Deshmukh. "A review on different approaches for speech recognition system." *International Journal of Computer Applications* 115.22 (2015). URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.695.7267&rep=rep1&type=pdf.

4. Washani, Nitin, and Sandeep Sharma. "Speech recognition system: A review." *International Journal of Computer Applications* 115.18 (2015). URL: https://www.researchgate.net/profile/Sandeep-Sharma-53/publication/276136263_Speech_Recognition_System_A_Review/links/5e411df292851c7f7f2bdee5/Speech-Recognition-System-A-Review.pdf.

5. Deng, Keqi, Songjun Cao, and Long Ma. "Improving Accent Identification and Accented Speech Recognition Under a Framework of Self-supervised Learning." arXiv preprint arXiv:2109.07349 (2021). URL: https://arxiv.org/abs/2109.07349.

6. Alex K - Енциклопедія «Українська мова». — Київ, 2005., CC BY 3.0. Зображення. URL: https://commons.wikimedia.org/w/index.php?curid=6071411.

7. Пачева, В. М. (2021) Діалекти української мови. ФО-П Однорог Т.В., Мелітополь. ISBN 978-617-7823-42-0. URL: http://eprints.mdpu.org.ua/id/eprint/12148/.

8. Fendji, Jean Louis KE, et al. "Automatic Speech Recognition using limited vocabulary: A survey." arXiv preprint arXiv:2108.10254 (2021). URL: https://arxiv.org/abs/2108.10254. URL: https://arxiv.org/abs/2108.10254.

9. What Are the Benefits of Speech Recognition Technology? URI: https://signalprocessingsociety.org/publications-resources/blog/what-are-benefits-speech-recognition-technology.

10. What Are The Benefits of Speech Recognition Technology? (25 Jul 2021) URI: https://onpassive.com/blog/what-are-the-benefits-of-speech-recognition-technology/.

11. Ron Miller. Serenade snags $2.1M seed round to turn speech into code[Електронний ресурс] / Ron Miller // Techcrunch. URL: https://techcrunch.com/2020/11/23/serenade-snags-2-1m-seed-round-to-turn-speech-into-code/.

12. Serenade Labs Inc. Serenade. URL: http://serenade.ai/docs.

13. Image with interface Serenade. URL: https://miro.medium.com/max/3000/1*LSWwVr25fJiWbwIowZwumA.png. 18

14. VoiceCode. URL: https://www.voicecode.io/.

15. Dragon Dictacion. Image. URL: https://www.ahead.ie/userfiles/images/AT%20Hive/AT%20Hive%20-%20screenshots/dragon-15-bar.jpg.

16. NaturalPoint, Inc. SmartNav. URL: https://www.naturalpoint.com/smartnav/.

17. NaturalPoint, Inc. Image. https://www.naturalpoint.com/smartnav/images/homefeature.jpg.

18. Berge, Gaute Andreas. Leveraging Language Tooling for Better Voice Coding: Implementing program awareness and structural editing for Talon. MS thesis. 2021. URL: https://www.duo.uio.no/handle/10852/86944.

19. Talon. Powerful hands-free input replacement. URL: https://talonvoice.com/docs/.

20. Talon Wiki. URL: https://talon.wiki/getting_started/#configure-a-speech-recognition-engine.

21. Онищенко К. Г. Аналіз методів обробки природної мови / К. Г. Онищенко, Я. Данієль, Р. Каменєв // Інформаційні системи та технології : матеріали 9-ї Міжнар. наук.-техн. конф., 17-20 листопада 2020 р. – Харків : Друкарня Мадрид, 2020. – С. 186–190. URL: https://openarchive.nure.ua/handle/document/16164.

22. MDN Web Docs. Using the Web Speech API. URL: https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API/Using_the_Web_Speech_API.

23. James Payne. Top Extensions and Add-ons for Visual Studio. June 13, 2021. URL: https://www.codeguru.com/dotnet/top-extensions-and-add-ons-for-visual-studio/.

24. James Payne. Visual Code Extensions for Web Developers. July 1, 2021. URL: https://www.codeguru.com/csharp/visual-code-extensions-for-web-developers/.

25. Wikipedia. Lint. URL: https://en.wikipedia.org/wiki/Lint_(software).

26. ESLint documentations. Configuring ESLint. URL: https://eslint.org/docs/user-guide/configuring/.

27. Image from GitHub. Not showing all errors when using ESLint processor on a file #922. URL: https://user-images.githubusercontent.com/900523/76773809-f8803e00-679a-11ea-8494-ac1fe1af5c81.png. 23

28. Yaoman. URL: https://yeoman.io/.

29. What is Node.JS and what is it used for? URI: https://www.tremplin-numerique.org/en/quest-ce-que-node-js-et-a-quoi-sert-il-cloudsavvy-it.

30. What is Git? What benefits does Git offer? URI: https://guide.quickscrum.com/git-guide.

31. Zachary Lukasiewicz. What are the top ten speech recognition APIs? URL: https://www.quora.com/What-are-the-top-ten-speech-recognition-APIs.

32. Kaldi ASR documentation. URL: http://kaldi-asr.org/doc/.

33. CMUSphinx documentation. URL: https://cmusphinx.github.io/wiki/.

34. Dragon speech recognition. Helping students reach their full potential. URL: https://www.nuance.com/content/dam/nuance/en_us/collateral/dragon/whitepaper/wp-helping-students-reach-their-full-potential-en-us.pdf.

35. Mozilla wiki. Web Speech API - Speech Recognition. URL: https://wiki.mozilla.org/Web_Speech_API_-_Speech_Recognition#WebSpeech_API_-_Speech_Recognition.

36. GitHub. Mozilla Speech-proxy. Image. URL: https://github.com/mozilla/speech-proxy/raw/master/docs/images/servers.png.

37. Mozilla wiki. Image. URL: https://wiki.mozilla.org/images/0/0c/Wsa_architecture.png.

38. Snapcraft. Installing snap on Debian. URL: https://snapcraft.io/docs/installing-snap-on-debian.

39. How to Install Visual Studio Code on Ubuntu 20.04. May 1, 2020. URL: https://linuxize.com/post/how-to-install-visual-studio-code-on-ubuntu-20-04/.

40. Загальні рекомендації з підготовки, оформлення, захисту та оцінювання випускних кваліфікаційних робіт здобувачів вищої освіти першого «бакалаврського» і другого «магістерського» рівнів / За ред. доц. М.І. Шинкарика. Тернопіль: ТНЕУ, 2018. 67 с.

41. Комар М.П., Саченко А.О., Васильків Н.М. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп'ютерні науки» спеціальності 122 «Комп'ютерні науки» за другим (магістерським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2021. 32 с.

42. Бащій В.А. Найшвидша модель розпізнавання мовлення на основі нейронної мережі. The 4th International scientific and practical conference "Topical issues of modern science, society and education" (November 1-3, 2021) SPC "Sciconf.com.ua", Kharkiv, Ukraine. 2021. 1402 p. ISBN 978-966-8219-85-6. 286-290 с. URL: https://sci-conf.com.ua/iv-mezhdunarodnaya-nauchno-prakticheskaya-

konferentsiya-topical-issues-of-modern-science-society-and-education-1-3-noyabrya-2021-goda-harkov-ukraina-arhiv/.

43. Бащій В.А. Аналіз методів автоматичного перетворення мовленнєвого сигналу на текст. The 3rd International scientific and practical conference “Topical issues of modern science, society and education” (October 3-5, 2021) SPC ”Sciconf.com.ua”, Kharkiv, Ukraine. 2021. 1096 p. ISBN 978-966-8219-85-6. URL: https://sci-conf.com.ua/iii-mezhdunarodnaya-nauchno-prakticheskaya-konferentsiya-topical-issues-of-modern-science-society-and-education-3-5-oktyabrya-2021-goda-harkov-ukraina-arhiv/.