Одеський національний університет імені І. І. Мечникова

(повне найменування закладу вищої освіти)

Факультет математики, фізики та інформаційних технологій

(повне найменування факультету)

Кафедра алгебри, геометрії та диференціальних рівнянь

(повна назва кафедри)

Кваліфікаційна робота

на здобуття ступеня вищої освіти «магістр»

«Розробка інтерактивного двобічного аудіо-відео сурдоперекладача»

(тема кваліфікаційної роботи українською мовою)

«Development the interactive system of two side AV sign-language interpreter»

(тема кваліфікаційної роботи англійською мовою)

Виконав: здобувач денної форми навчання

Спеціальності: 123 – Комп'ютерна інженерія

(код, назва спеціальності)

Освітня програма _____

(назва)

Іванов Олег Русланович

(прізвище, ім'я, по-батькові здобувача)

Керівник: д. ф.-м. н., професор Варбанець С.П. _____

(науковий ступінь, вчене звання,прізвище, ініціали) (підпис)

Рецензент: кандидат технічних наук, доц. Якімова Н.А

(науковий ступінь, вчене звання,прізвище, ініціали)

Рекомендовано до захисту:

Протокол засідання кафедри

№ ___ від _____. _____. 20___ р.

Завідувач(ка) кафедри

_____ __Євтухов В.М.__

(підпис) (прізвище,ім'я)

Захищено на засіданні ЕК № _____

протокол № __від ____._____.20___ р.

Оцінка_____/_____/_____

(за національною шкалою/шкалою ECTS/ бали)

Голова ЕК

_____ _____

(підпис) (прізвище, ім'я)

Одеса 2023

АНОТАЦІЯ

Дипломна робота присвячена розробці та впровадженню інтерактивного двобічного аудіо-відео сурдоперекладача для людей з вадами слуху, використовуючи сучасні веб-технології. Робота включає створення веб-сайту за допомогою HTML, CSS та JavaScript, а також використання популярних бібліотек та фреймворків, зокрема React та Next.js. Для підвищення ефективності та надання користувачам більших можливостей, використовується SCSS як препроцесор стилів.

Однією з ключових особливостей розробленого сурдоперекладача є використання технології розпізнавання жестів через Mediapipe gesture recognition, що дозволяє користувачам взаємодіяти з системою за допомогою жестів рук. Крім того, для розпізнавання мови використовується пакет react-speech-recognition, що робить можливим автоматичне розпізнавання та інтерпретацію мовлення людей з вадами слуху.

Ця дипломна робота не лише демонструє технічну експертизу в області веб-розробки, але й ставить перед собою завдання полегшення комунікації для осіб із вадами слуху, забезпечуючи їм зручний та доступний інструмент для взаємодії та спілкування.

ЗМІСТ

ВСТУП

В сучасному світі, де технології стрімко розвиваються, важливо враховувати потреби всіх верств населення, забезпечуючи їм рівний доступ до інформації та можливостей. Однією з важливих сфер, яка вимагає особливої уваги, є полегшення комунікації для осіб із вадами слуху. У цьому контексті маю честь представити результати дослідження та розробки - інтерактивного двобічного аудіо-відео сурдоперекладача.

Мій проект є відповіддю на актуальну проблему взаєморозуміння та взаємодії між людьми, де одна зі сторін стикається із вадами слуху. Метою даного дослідження та розробки є створення інноваційного рішення, що дозволить особам із вадами слуху не лише ефективно спілкуватися, але й відчувати себе повноцінними учасниками сучасного інформаційного суспільства.

За останні роки інтернет та цифрові технології значно змінили спосіб, яким ми взаємодіємо та спілкуємося. Однак для осіб із вадами слуху доступ до цих технологій може бути ускладненим. У багатьох випадках існуючі засоби комунікації не повністю враховують їхні потреби.

Метою моєї дипломної роботи є розробка та впровадження інтерактивного сурдоперекладача, який забезпечить двосторонню аудіо-відео комунікацію між особами із вадами слуху та їхніми співрозмовниками. Для досягнення цієї мети, я використовуємо широкий спектр сучасних технологій та інструментів розробки веб-додатків.

У процесі розробки веб-сайту, який використовується як основа для мойого сурдоперекладача, були задіяні HTML, CSS та JavaScript. Фронтенд розробка виконана за допомогою React та Next.js, що дозволило забезпечити ефективний та ресурсозберігаючий інтерфейс. Використання препроцесора стилів SCSS спростило управління та розвиток структури стилів додатку.

Для розпізнавання жестів використовується технологія Mediapipe gesture recognition, що надає можливість взаємодії користувачів із

сурдитністю через рухи рук. Пакет react-speech-recognition використовується для реалізації автоматичного розпізнавання та інтерпретації мовлення, що дозволяє користувачам виражати свої думки та ідеї без перешкод.

У подальших розділах роботи детально розглядаються теоретичні аспекти вад слуху, архітектура розробленого сурдоперекладача, реалізація його функціональності, тестування та оцінка продуктивності, а також висновки та подальші напрями розвитку проекту.

Моя робота спрямована на створення інноваційного та ефективного інструменту, що полегшить життя та покращить комунікацію для людей з вадами слуху, внесучи свій внесок у розвиток інклюзивного суспільства.

1 АКТУАЛЬНІСТЬ ПРОБЛЕМИ ІНКЛЮЗІЇ ЛЮДЕЙ З ВАДАМИ СЛУХУ У СУСПІЛЬСТВО

У контексті сучасного розвитку суспільства, де технології визначають новий стандарт комунікації, актуальність проблеми осіб із вадами слуху стає одразу більш очевидним.

У сучасному світі, де стежимо за тенденціями до інклюзивності та рівних можливостей, соціальна ізоляція осіб із вадами слуху лише поглиблюється через обмежений доступ до традиційних форм спілкування. Традиційні засоби комунікації, такі як мова тіла чи читання по губам часто виявляються неефективними або недостатньо зручними.

Понад те, що доступ до освіти є ключовим аспектом для інтеграції в суспільство, для осіб із вадами слуху цей шлях стає суттєво ускладненим. Відсутність адаптованих інструментів для ефективної комунікації може обмежувати їх професійний розвиток та доступ до робочих можливостей.

Зокрема, відзначається важливість інтерактивних засобів комунікації для участі в культурних і громадських подіях. Інклюзія в культурні аспекти життя стає недостатньою без належного забезпечення можливостей взаєморозуміння.

Розвиток інтернет-технологій несе в собі як нові можливості, так і нові виклики. Особи із вадами слуху також мають право на використання новітніх технологій та веб-сервісів. Забезпечення їм доступу до цифрових інновацій стає необхідністю для їх повноцінної участі у сучасному інформаційному суспільстві.

Ефективне передавання невербальних сигналів є ключовим аспектом комунікації. Застосування технологій розпізнавання жестів та мови надає можливість ефективно передавати і отримувати невербальну інформацію, забезпечуючи повноцінність спілкування.

2 МЕТА ТА ЗАВДАННЯ ДОСЛІДЖЕННЯ

Метою даного дослідження є розробка та впровадження інтерактивного двобічного аудіо-відео сурдоперекладача, спрямованого на поліпшення комунікації між особами із вадами слуху та їх оточенням. Основна мета полягає в створенні технологічного рішення, яке не лише компенсує втрату слуху, але й надає повноцінний та ефективний канал спілкування, адаптований до індивідуальних потреб користувачів.

2.1 Завдання дослідження

До основних завдань цього проекту можна віднести:

a) Аналіз Існуючих Рішень:

1) Провести детальний аналіз існуючих технологій та інструментів для сприяння комунікації осіб із вадами слуху.

2) Визначити переваги та недоліки існуючих рішень для вдосконалення та врахування їх у подальшій розробці.

b) Розробка Інтерактивного Сурдоперекладача:

1) Розробити архітектуру та функціональність інтерактивного двобічного аудіо-відео сурдоперекладача.

2) Використати HTML, CSS, JS, React, Next.js та SCSS для розробки веб-інтерфейсу засобу комунікації.

c) Інтеграція Розпізнавання Жестів:

1) Обрати та інтегрувати технологію розпізнавання жестів, зокрема Mediapipe gesture recognition, для полегшення невербальної комунікації.

2) Забезпечити оптимальну взаємодію між жестовим розпізнаванням та іншими функціями сурдоперекладача.

d) Розпізнавання Мови за Допомогою React-Speech-Recognition:

1) Використати пакет react-speech-recognition для імплементації розпізнавання мови та перетворення її на текст.

2) Забезпечити точність та швидкість розпізнавання мови для ефективного спілкування.

e) Вивчення Потенціалу Інтеграції з Іншими Платформами:

1) Дослідити можливості інтеграції інтерактивного сурдоперекладача з іншими платформами та сервісами.

2) Врахувати вимоги до масштабованості та сумісності із загальноприйнятими стандартами.

f) Створення Документації та Посібника Користувача:

1) Розробити докладну документацію, яка включатиме технічні деталі та інструкції щодо використання інтерактивного сурдоперекладача.

2) Забезпечити легкий доступ до інформації для користувачів і технічних спеціалістів.

3 ОГЛЯД ІСНУЮЧИХ ЗАСОБІВ КОМУНІКАЦІЇ ДЛЯ ЛЮДЕЙ ІЗ ВАДАМИ СЛУХУ

У світі швидкого технологічного розвитку роль інноваційних інструментів у поліпшенні комунікації осіб із вадами слуху стає все більш визначальною. З метою надання повноцінного доступу до інформації та сприяння ефективному взаєморозумінню, було розроблено та впроваджено різноманітні програми та веб-сервіси.

Для аналізу було обрано пять інноваційних рішень для комунікації осіб із вадами слуху. Кожен із цих інструментів виступає у ролі важливого посередника, спрямованого на розвиток інклюзивного середовища та забезпечення комфортного спілкування для кожного користувача. Нижче розписано їхні основні характеристики, переваги та недоліки для зрозуміння їхнього внеску в поліпшення якості життя людей із вадами слуху.

3.1 Google Live Transcribe

Google Live Transcribe - це інноваційний сервіс, розроблений компанією Google, який дозволяє автоматично перекладати мовлення на текст у режимі реального часу. Це значно полегшує комунікацію для осіб із вадами слуху, надаючи їм можливість читати те, що вони чують. Сервіс використовує передові технології мовного розпізнавання, щоб точно відтворювати вимовлене слово чи речення у текстовій формі.

Характеристики:

Google Live Transcribe використовує передові технології мовного розпізнавання, щоб миттєво перекладати вимовлене слово чи фразу в текст. Це забезпечує негайний доступ до інформації для осіб із вадами слуху.

Система використовує високопродуктивні алгоритми розпізнавання мови, що дозволяє точно відтворювати вимовлене слово або речення у вигляді тексту з високою якістю.

Плюси:

Google Live Transcribe демонструє високу ефективність у ситуаціях, де інші сервіси можуть зіткнутися з труднощами через велику відстань або високий рівень шуму.

Сервіс миттєво реагує на мовлення та забезпечує високу точність у відтворенні тексту, що дозволяє користувачам отримувати швидкий і точний результат.

Мінуси:

У ситуаціях з низьким рівнем звукового сигналу або непридатною якістю мікрофона, Google Live Transcribe може втрачати точність розпізнавання, що може вплинути на здатність користувача зрозуміти вміст тексту.

3.2 Ava

Ava – це інноваційний інструмент, розроблений для полегшення спілкування осіб із вадами слуху. Сервіс спеціалізується на груповому чаті, що дозволяє користувачам одночасно спілкуватися з декількома учасниками та отримувати переклад мовлення на текст для поліпшення комунікації.

Характеристики:

Ava пропонує зручний та ефективний груповий чат, що дозволяє взаємодіяти з різними учасниками одночасно.

Сервіс використовує технології мовного розпізнавання для точного відтворення вимовленого у вигляді тексту.

Плюси:

Ava підтримує активну взаємодію у групі, що дозволяє користувачам не лише спілкуватися, але й взаємодіяти між собою, роблячи комунікацію більш динамічною.

Ava вражає зручним та інтуїтивно зрозумілим інтерфейсом, а також надає можливість зберігання історії чатів для подальшого використання.

Мінуси:

Хоча Ava надає безкоштовний доступ до основних функцій, деякі розширені можливості доступні лише у преміум-версії, що може вплинути на повний функціонал для окремих користувачів.

3.3 Microsoft Translator

Microsoft Translator - це засіб для мовного перекладу, розроблений Microsoft, який дозволяє користувачам ефективно спілкуватися на різних мовах. Сервіс використовується для перекладу мовлення у текстовий формат та навпаки, сприяючи міжнародній комунікації.

Характеристики:

Microsoft Translator володіє широким спектром мов, що дозволяє користувачам спілкуватися та отримувати переклади на багатьох мовах світу.

Сервіс надає можливість відтворення перекладеного тексту в аудіоформаті, що полегшує сприйняття перекладу для користувачів.

Плюси:

Microsoft Translator пропонує багатомовні можливості, що робить його ідеальним інструментом для міжнародних комунікацій та спілкування.

Сервіс має високу ступінь інтеграції з іншими продуктами Microsoft, що дозволяє користувачам зручно використовувати його в різних програмах та сервісах.

Мінуси:

Оскільки Microsoft Translator працює в режимі онлайн, точність перекладів може залежати від якості інтернет-з'єднання, що може бути обмеженням в умовах низької швидкості чи відсутності з'єднання.

3.4 SignLanguage Interpreter

SignLanguage Interpreter є інноваційним додатком, спрямованою на полегшення спілкування для осіб із вадами слуху через переклад мовлення на жестову мову. Вона використовує відеорозпізнавання для інтерпретації жестів, забезпечуючи більш ефективний та інклюзивний засіб комунікації.

Характеристики:

SignLanguage Interpreter спеціалізується на інтерпретації мовлення на жестову мову, забезпечуючи ефективний засіб спілкування для осіб із обмеженим слухом.

Сервіс використовує передові технології відеорозпізнавання для точної інтерпретації жестів, забезпечуючи найкращу можливу якість перекладу.

Плюси:

SignLanguage Interpreter враховує особливості жестової мови, роблячи її ідеальним інструментом для комунікації для групи людей із сурдитністю.

Аплікація пропонує зручний та інтуїтивно зрозумілий інтерфейс, що спрощує введення та інтерпретацію жестів для користувачів.

Мінуси:

Деякі користувачі можуть потребувати тренування для оптимального використання сервісу, оскільки точність розпізнавання може покращитися з практикою та звичкою.

3.5 Be My Eyes

Be My Eyes є інноваційною платформою, призначеною для сприяння взаємодії між волонтерами та користувачами із вадами зору чи слуху. Основним функціоналом є реальний час відеокомунікації, що дозволяє волонтерам надавати допомогу в режимі онлайн.

Характеристики:

Be My Eyes створений для того, щоб об'єднувати волонтерів із людьми, які мають обмеження в зорі чи слуху, забезпечуючи підтримку та допомогу в реальному часі.

Основна функція сервісу - відеокомунікація в реальному часі, що дозволяє волонтерам бачити те, що бачить користувач, та надавати конкретну допомогу.

Плюси:

Be My Eyes розрахований на підтримку осіб із різними обмеженнями в сприйнятті, що робить його універсальним та важливим інструментом для інклюзивного спілкування.

Платформа забезпечує простий та ефективний спосіб взаємодії, дозволяючи користувачам отримувати допомогу в режимі реального часу від волонтерів з усього світу.

Мінуси:

Ефективність Be My Eyes залежить від наявності волонтерів, їх готовності та доступності, що може становити виклик в деяких ситуаціях, особливо в регіонах з обмеженим доступом до Інтернету або низьким обсягом користувачів.

4 ЖЕСТОВА МОВА В КОНТЕКСТІ РОЗРОБКИ ПРОЕКТУ З РОЗПІЗНАВАННЯ ЖЕСТІВ

Жестова мова, яка є важливим засобом комунікації для людей з вадами слуху, представляє собою комплексну систему вираження думок та ідей за допомогою рухів рук та інших частин тіла. При розробці проекту з розпізнавання жестів, важливо розглядати різні аспекти жестової мови, щоб забезпечити точність та ефективність системи.

4.1 Види Жестової Мови

Жестова мова є унікальним способом спілкування, використовуваним людьми з вадами слуху по всьому світу. Різні культури та регіони розвинули свої власні системи жестів, які відрізняються за граматикою, лексикою та синтаксисом. Важливо розглядати цю різноманітність при розробці проекту з розпізнавання жестів.

Американська жестова мова (ASL): Створена в США, ASL є однією з найпоширеніших форм жестової мови. Вона використовує власну граматику та лексику, яка відрізняється від мови говоріння.

Британська жестова мова (BSL): Використовується в Великобританії та володіє своєю системою жестів і синтаксисом. Вона відрізняється від ASL та інших форм жестової мови.

Міжнародна жестова мова (ISL): Призначена для міжнародного спілкування, ISL створена для полегшення взаєморозуміння між людьми з різних країн.

Мануальна алфавітна жестова мова: Використовує окремі жести для кожної літери алфавіту та дозволяє формувати слова.

Французька жестова мова (LSF): Застосовується в Франції і в інших франкомовних регіонах зі своєю власною лексикою та жестовими виразами.

Китайська жестова мова (CSL): Розроблена для спілкування глухих в Китаї та інших китайських спільнотах.

Жестова мова для дітей з аутизмом: Спеціально створена для полегшення спілкування та розуміння дітей з аутизмом.

Ця різноманітність видів жестової мови відображає культурні відмінності та нюанси спілкування, які необхідно враховувати при створенні систем розпізнавання жестів.

Особливості Жестової Мови:

Жестова мова має свої унікальні особливості, які важливо враховувати при розробці системи розпізнавання. Просторова граматика використовується для передачі часу та місця. Експресивність дозволяє виражати емоції за допомогою рухів. Фасіальні вирази грають важливу роль у вираженні суті повідомлення.

Труднощі при Розробці Проекту:

Розпізнавання контексту є однією з ключових труднощів, оскільки однакові жести можуть мати різні значення в різних ситуаціях. Варіабельність жестів між різними мовами та індивідуальні стилі користувачів можуть впливати на точність системи. Адаптація до фізичних обмежень та врахування рухливості користувача також представляють виклики.

5 ВИБІР ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ВЕБ-ДОДАТКУ

Розробка веб-додатку для інтерактивного двобічного аудіо-відео сурдоперекладача — завдання, яке потребує уважного підходу до вибору технологій. У цьому розділі розглянемо ключові аспекти та обґрунтуємо вибір конкретних технічних інструментів.

5.1 Frontend (Клієнтська частина)

React та Next.js обрані для клієнтської частини з метою створення високоефективного та швидкого інтерактивного додатку. Ця комбінація технологій забезпечить не тільки ефективність та швидкодію додатку, але й спростить його розробку та підтримку завдяки високій рівні організації та широкому спектру можливостей React та Next.js.

5.1.1 React в клієнтській частині проекту

React - це бібліотека для створення інтерфейсів, і вона вибрана для клієнтської частини проекту з ряду вагомих причин.

Переваги React в проекті:

a) Компонентна архітектура: React базується на компонентній архітектурі, що спрощує розробку та управління кодом. Кожен елемент інтерфейсу представлений як окремий компонент, що робить код більш читабельним та легко розширюваним.

b) Декларативний підхід: React використовує декларативний стиль програмування, де ви описуєте, як повинен виглядати інтерфейс у різних станах. Це спрощує процес відлагодження та підтримки коду.

c) Віртуальний DOM: Використання віртуального DOM дозволяє ефективно взаємодіяти з реальним DOM, мінімізуючи зайві операції

оновлення. Це призводить до покращення продуктивності та швидкодії додатку.

d) Реактивні компоненти: React дозволяє створювати реактивні компоненти, які автоматично оновлюються при зміні стану або властивостей. Це робить додаток більш інтерактивним та відповідальним.

e) Розширюваність та спільнота: React має велику та активну спільноту, що забезпечує доступ до багатьох сторонніх бібліотек та інструментів. Це сприяє легкій розширюваності проекту та використанню найновіших розробок.

f) Оптимізація швидкодії: Завдяки віртуальному DOM, можливостям оптимізації коду та широким можливостям управління станом, React забезпечує швидкі та ефективні додатки.

g) Широке використання у індустрії: React вже успішно використовується в багатьох великих проектах, що свідчить про його надійність та ефективність.

У контексті проекту React грає ключову роль у створенні інтерактивного та ефективного інтерфейсу. Він дозволяє ефективно організувати код, забезпечуючи легкість розробки та підтримки. Компонентна архітектура дозволяє розділити інтерфейс на невеликі, самостійні частини, що полегшує розширення та управління. Його декларативний підхід дозволяє чітко визначити, як має виглядати інтерфейс, забезпечуючи читабельність та підтримку коду. Віртуальний DOM та реактивні компоненти сприяють швидкодії та ефективності додатку. Також, завдяки широкій спільноті та активному використанню в індустрії, React є високоякісним вибором для реалізації клієнтської частини проекту.

5.1.2 Next.js в клієнтській частині проекту

Next.js є фреймворком для React і вибраним інструментом для створення клієнтської частини проекту. Цей вибір зумовлений кількома

ключовими перевагами та ролями, які Next.js відіграє в контексті розробки інтерактивного двобічного аудіо-відео сурдоперекладача.

Переваги Next.js в проекті:

a) Серверний рендеринг: Next.js підтримує серверний рендеринг, що дозволяє генерувати HTML на сервері та відсилати його на клієнт, що сприяє поліпшенню швидкодії та індексації веб-сторінок.

b) Віртуалізація та ліниве завантаження: Next.js використовує віртуалізацію та ліниве завантаження, що дозволяє оптимізувати завантаження компонентів та покращити час завантаження сторінок.

c) SEO-оптимізація: Завдяки серверному рендерингу та генерації статичних сторінок, Next.js сприяє поліпшенню SEO та забезпеченню високого рівня індексації додатку.

d) Підтримка TypeScript: Next.js підтримує TypeScript, що дозволяє забезпечити типізацію та підвищити робочий процес розробників.

e) Зручний робочий процес: Next.js надає зручний робочий процес з автоматичною конфігурацією та широким спектром інструментів.

Ролі Next.js в проекті:

a) Поліпшення швидкодії: Завдяки серверному рендерингу та віртуалізації, Next.js сприяє створенню високоефективного та швидкого інтерфейсу.

b) Оптимізація SEO: Генерація статичних сторінок та серверний рендеринг поліпшують індексацію додатку та його видимість у пошукових системах.

c) Ліниве завантаження: Механізм лінивого завантаження дозволяє оптимізувати завантаження компонентів, підвищуючи ефективність сторінок.

d) Підтримка TypeScript: Next.js дозволяє використовувати TypeScript, що полегшує роботу з кодом та допомагає уникнути потенційних помилок.

e) Зручний робочий процес: Надійний робочий процес Next.js забезпечує зручність розробки та конфігурацію.

Next.js у проекті використовується для забезпечення високої швидкодії, оптимізації для пошукових систем, полегшення роботи з TypeScript та покращення загального робочого процесу розробки. Його функціональність, як серверного рендерингу, так і віртуалізації, грає ключову роль у створенні ефективного та динамічного інтерфейсу для аудіо-відео сурдоперекладача.

5.1.3 HTML/CSS/JS, TypeScript та Sass в клієнтській частині проекту

HTML, CSS, та JavaScript (включаючи TypeScript) є основними технологіями для створення інтерфейсу клієнтської частини проекту. Додатково, використання Sass як препроцесора для CSS дозволяє ефективно організовувати та підтримувати стилі проекту.

Роль та переваги використання HTML/CSS/JS, TypeScript та Sass в проекті:

a) Стандартні технології інтернету: HTML, CSS та JavaScript є стандартними технологіями для розробки веб-додатків, забезпечуючи високий рівень сумісності та доступності.

b) Динамічний інтерфейс за допомогою JavaScript: JavaScript використовується для створення динамічного та інтерактивного інтерфейсу, що забезпечує користувачеві зручність у використанні.

c) Типізація та ефективність з TypeScript: TypeScript, який є розширенням JavaScript, дозволяє використовувати систему типізації, що полегшує виявлення помилок та поліпшує робочий процес.

d) Зручна робота зі стилями за допомогою Sass: Sass, який є препроцесором для CSS, надає можливість використовувати змінні, вкладеність та інші функції, полегшуючи роботу зі стилями та роблячи код більш організованим.

e) Підтримка різних екосистем: Використання HTML/CSS/JS та TypeScript забезпечує сумісність та інтеграцію з різними бібліотеками, фреймворками та інструментами.

f) Легка підтримка та розширюваність: Стандартні технології дозволяють легко підтримувати та розширювати проект, що важливо для подальшого розвитку.

g) Ефективне управління стилями за допомогою Sass: Sass надає додаткові можливості для ефективного управління стилями, такі як змінні, міксини та імпорт, що спрощує розробку та підтримку стилів.

HTML, CSS, та JavaScript, доповнені TypeScript та Sass, грають ключову роль у створенні зручного, ефективного та стильного інтерфейсу. Забезпечуючи стандарти веб-розробки та використовуючи зручні інструменти для стилізації та динамічного інтерфейсу, ці технології сприяють створенню повноцінного клієнтського додатку для аудіо-відео сурдоперекладача.

5.2 Mediapipe Gesture Recognition (Розпізнавання жестів)

Mediapipe Gesture Recognition є потужним інструментом для розпізнавання жестів, що використовується в клієнтській частині проекту. Ця бібліотека дозволяє виявляти та інтерпретувати жести, здійснюючи високоточне взаємодію користувача з системою.

Принцип Роботи та Технічні Деталі:

Розпізнавання ключових точок:

Медіаорган розпізнавання жестів використовує алгоритм роозпізнавання ключових точок, що дозволяє точно визначити положення руки та її частин.

Моделі машинного навчання:

Бібліотека використовує попередньо навчені моделі машинного навчання для розпізнавання різних жестів. Ці моделі дозволяють ефективно класифікувати та інтерпретувати рухи користувача.

Розпізнавання жестів:

За допомогою детектованих ключових точок та моделей машинного навчання, Mediapipe Gesture Recognition розпізнає різні жести, такі як махання рукою, показування певних знаків тощо.

Переваги в Проекті:

Висока Точність:

Бібліотека гарантує високу точність розпізнавання жестів, що є критичним для забезпечення точності та ефективності взаємодії з користувачем.

Швидкість Обробки:

Медіаорган Gesture Recognition пропонує швидку обробку в реальному часі, що робить його ідеальним для інтерактивних застосувань.

Гнучкість у Використанні:

Завдяки своїм функціональностям та конфігураційним опціям, ця бібліотека може бути гнучкою у використанні для різних жестів та взаємодій.

Інтеграція з React та Next.js:

Легко інтегрується з існуючим проектом на React та Next.js, сприяючи зручному використанню у веб-додатку.

Роль у Проекті:

Mediapipe Gesture Recognition використовується для забезпечення інтерактивного та динамічного управління функціоналом додатку. Розпізнавання жестів дозволяє користувачам взаємодіяти з аудіо-відео сурдоперекладачем шляхом простих та природних рухів, що полегшує використання та робить додаток більш доступним для людей із вадами слуху.

Навчання та Підтримка:

Mediapipe Gesture Recognition може бути налаштований для розпізнавання специфічних жестів, що вимагає від розробників ознайомлення з документацією та можливість підгонки параметрів для досягнення оптимальних результатів у конкретному використанні. Бібліотека також

надає документацію та спільноту для підтримки розробників у процесі інтеграції та використання.

5.3 React Speech Recognition (Розпізнавання мови)

React-Speech-Recognition є інструментом, який додає можливість розпізнавання мовлення до веб-додатку на базі React. Цей пакет важливий для забезпечення комунікації з користувачем за допомогою голосових команд та введення тексту через мовлення.

Принцип Роботи та Технічні Деталі:

Активація за Допомогою Компонентів React:

React-Speech-Recognition дозволяє активувати розпізнавання мовлення за допомогою компонентів React, що спрощує інтеграцію в існуючий код.

Використання API Браузера:

Пакет використовує функціональність розпізнавання мовлення, яка доступна в браузерах, тобто Speech Recognition API.

Обробка Результатів:

Результати розпізнавання передаються в компонент React для подальшої обробки та використання в логіці додатку.

Переваги в Проекті:

Зручність введення:

Забезпечує користувачам зручний спосіб введення тексту та взаємодії з додатком за допомогою мовлення.

Підтримка Голосових Команд:

Реалізує підтримку голосових команд, що розширює функціональність додатку та полегшує взаємодію.

Інтерактивність та Доступність:

Робить додаток більш інтерактивним та доступним для користувачів, особливо тих, у кого є обмеження у введенні тексту за допомогою клавіатури.

Легка Інтеграція з React та Next.js:

Легко інтегрується з існуючими технологіями, такими як React та Next.js, що полегшує роботу розробників.

Роль у Проекті:

React-Speech-Recognition відповідає за забезпечення можливості введення тексту через мовлення та інтеграцію голосових команд. Це розширює способи взаємодії з додатком та забезпечує більш гнучке та доступне введення для користувачів.

Навчання та Підтримка:

Використання React-Speech-Recognition вимагає розуміння його API та інтеграції з компонентами React. Документація та спільнота розробників можуть служити джерелами підтримки та відповідей на можливі питання.

6 РОЗРОБКА ІНТЕРФЕЙСУ КОРИСТУВАЧА

В сучасному світі розробка інтерфейсу користувача стала ключовою складовою будь-якого веб-додатку чи сайту. Швидкий та ефективний інтерфейс забезпечує зручність взаємодії, завдяки чому користувачі можуть насолоджуватися позитивним досвідом використання продукту.

Розглянемо розробку інтерфейсу користувача з використанням низки потужних інструментів та технологій. В основі моєї роботи лежить використання HTML, CSS та JavaScript, щоб створювати основний скелет сторінок та надавати їм зовнішній вигляд.

React та Next.js виступають ключовими бібліотеками, які дозволяють нам створювати компонентну структуру та оптимізувати завантаження сторінок. Реактивність та можливість використання серверного рендерингу в Next.js дозволяють забезпечити швидкий та ефективний рендеринг веб-сторінок.

6.1 Розробка хедера сайту

Код структури хедера виглядає таким чином:

```
     <div className={styles.header}>
           <div className={clsx(styles.header__content,
"content")}>
             <div className={styles.header__logoBlock}>
               <Link href="/" onClick={() =>
dispatch(setBurgerState(false))}>
                 <Image
                   className={styles.logo}
                   src="/logo.png"
                   alt="LogoImage"
                   width={50}
                   height={50}
                 />
               </Link>
               <Link href="/" onClick={() =>
dispatch(setBurgerState(false))}>
                 <p>
```

```
                Mio<span>Gesto</span>
              </p>
            </Link>
          </div>
          <nav className={styles.header__navigation}>
            <ul>
              {navItems.map((item) => (
                <li key={item}>
                  <Link href={`/${item.toLowerCase()}`}>
                    <p>{item}</p>
                  </Link>
                </li>
              ))}
            </ul>
            <div
              className={clsx(styles.header__burger, {
                [styles.active]: burgerState,
              })}
              onClick={() =>
dispatch(setBurgerState(!burgerState))}
            >
              <span></span>
            </div>
          </nav>
          <NavigationPanel />
        </div>
      </div>
```

Розглянемо кожен елемент окремо та пояснимо їх функції та призначення:

`<div className={styles.header}>`: Це зовнішній контейнер, який представляє собою верхню частину сторінки (header). Клас styles.header вказує на використання стилів, які визначені в іншому файлі стилів (за допомогою CSS-модулів).

`<div className={clsx(styles.header__content, "content")}>`: Це контейнер для вмісту заголовка. Клас styles.header__content вказує на стилі, а content є, ще одним класом, який також визначає стилі для цього контейнера. clsx використовується для злиття різних класів, переданих як аргументи.

`<div className={styles.header__logoBlock}>`: Це контейнер для логотипу та текстового напису. Включає в себе два дочірні елементи: логотип (зображення) та текстовий напис.

`<Link href="/" onClick={()=> dispatch(setBurgerState(false))}>`: Це компонент з бібліотеки Next.js, який надає навігаційні посилання в програмі. href="/" вказує на адресу, на яку перейде користувач при кліці на посилання. `onClick={() => dispatch(setBurgerState(false))}` означає, що при кліці на посилання буде викликана функція setBurgerState з параметром false, яка відповідає за зміну стану бургер-меню.

`<Image className={styles.logo} src="/logo.png" alt="LogoImage" width={50} height={50} />`: Це компонент для відображення зображень. Зображення логотипу завантажується з файлу "/logo.png". Ширина та висота також вказані.

`<Link href="/" onClick={() => dispatch(setBurgerState(false))}>`: Аналогічно попередньому, це посилання на головну сторінку.

`<p>Mio<span>Gesto</span></p>`: Заголовок, який складається з двох частин - "Mio" та "Gesto". "Gesto" обгорнуто тегом , щоб застосувати до нього окремі стилі.

`<nav className={styles.header__navigation}>`: Це контейнер для навігаційного меню, яке містить список посилань та бургер-меню.

`<ul>`: Невупорядкований список для розміщення посилань.

`{navItems.map((item) => (</li>))}`: Це використання JavaScript-функції map для створення елементів списку на основі масиву navItems. Кожен елемент масиву представляє одне посилання в навігаційному меню.

`<li key={item}>`: Елемент списку, де key вказує на унікальний ідентифікатор для елемента.

`<Link href={/${item.toLowerCase()}}>`: Посилання на сторінку, яка визначається змінною item та конвертується в нижній регістр.

`<p>{item}</p>`: Текстовий вміст посилання.

`<div className={clsx(styles.header__burger, {[styles.active]: burgerState,})} onClick={() => dispatch(setBurgerState(!burgerState))}>`: Це бургер-меню, яке з'являється при зменшенні ширини екрану. Клас styles.header__burger визначає стилі для бургер-меню, а клас styles.active додається в залежності від стану burgerState. Клік на бургер-меню викликає функцію, яка змінює стан burgerState.

`<NavigationPanel />`: Це компонент <NavigationPanel />, містить додатковий вміст або функціональність для навігації, а саме відповідає за відображення пунктів меню у шапці сайту.

6.2 Розробка блока користувача у кімнаті

Код структури блока користувача у розмовній кімнаті виглядає так:

```
<div className={styles.user}>
      <div className={styles.user__block}>
        <div
          className={clsx(styles.user__avatar, {
            [styles.blockoff]: webcamState,
          })}
        >
          <Image
            src="/avatar.jpg"
            alt="User Avatar"
            height={120}
            width={100}
            priority={true}
          />
        </div>

        <section
          className={clsx(styles.videoSection, {
            [styles.invisible]: gestureRecIsLoad,
            [styles.blockoff]: !webcamState,
          })}
        >
          <div className={styles.videoView}>
```

```
                <div className={styles.mediaBlock} style={{
position: "relative" }}>
                  <video
                    className={styles.videoBlock}
                    id="webcam"
                    autoPlay
                    playsInline
                    ref={video}
                  ></video>
                  <canvas
                    className={styles.output_canvas}
                    ref={canvasElement}
                    id="output_canvas"
                    style={{ position: "absolute", left: "0px",
top: "0px" }}
                  ></canvas>
                  <p
                    ref={gestureOutput}
                    className={styles.output}
                    style={{
                      backgroundColor: "white",
                      color: "black",
                      position: "absolute",
                      left: "0px",
                      top: "0px",
                    }}
                  >
                    Hi
                  </p>
                </div>
              </div>
            </section>

            <div className={styles.user__bottomPanel}>
              <div
                className={styles.user__nickname}
              >{`broke.oli: ${transcript}`}</div>

              <div className={styles.user__bottomButtons}>
                <div
                  className={clsx(styles.user__micImageBlock, {
                    [styles.micoff]: !micState,
                  })}
                >
                  <Image
                    className={styles.user__micro}
```

```
                    onClick={() => setMicState((mic) => !mic)}
                    src="/microphone.svg"
                    alt="Microphone"
                    width={50}
                    height={50}
                  />
                </div>

                <button
                  ref={enableWebcamButton}
                  id="webcamButton"
                  className={clsx(styles.user__camImageBlock, {
                    [styles.camoff]: !webcamState,
                  })}
                  onClick={enableCam}
                >
                  <Image
                    className={styles.user__camera}
                    src="/camera.svg"
                    alt="Camera"
                    width={50}
                    height={50}
                  />
                </button>

                <div>
                  <p>Microphone: {listening ? "on" : "off"}</p>
                  <button onClick={() =>
onStartListen()}>Start</button>
                  <button onClick={() =>
onStopListen()}>Stop</button>
                  <button
onClick={resetTranscript}>Reset</button>
                </div>
            </div>
          </div>
      </div>
      </div>
```

Розглянемо кожен елемент окремо та пояснимо їх функції та призначення:

`<div className={styles.user}>`: Це зовнішній контейнер, представляючий блок для інформації про користувача та його взаємодії з додатком.

`<div className={styles.user__block}>`: Це контейнер, який об'єднує різні частини інтерфейсу користувача.

`<div className={styles.user__avatar}>`: Блок для відображення аватару користувача. Клас styles.blockoff використовується для приховування блоку у випадку, якщо webcamState є істинним.

`<section className={styles.videoSection}>`: Секція для відображення відеопотоку та жестів.

`<div className={styles.videoView}>`: Блок для відображення відеопотоку та результатів розпізнавання жестів.

`<div className={styles.mediaBlock} style={{ position: "relative" }}>`: Блок, який містить відео, канвас для відображення жестів та текстовий вивід.

`<video className={styles.videoBlock} id="webcam" autoPlay playsInline ref={video}></video>`: Елемент <video> для відтворення відеопотоку з веб-камери. ref={video} використовується для посилання на елемент в React.

`<canvas className={styles.output_canvas} ref={canvasElement} id="output_canvas" style={{ position: "absolute", left: "0px", top: "0px" }}></canvas>`: Канвас для відображення жестів на відео.

`<p ref={gestureOutput} className={styles.output} style={{ backgroundColor: "white", color: "black", position: "absolute", left: "0px", top: "0px" }}></p>`: Текстовий вивід результатів розпізнавання жестів.

`<div className={styles.user__bottomPanel}>`: Нижній блок інтерфейсу користувача.

`<div className={styles.user__nickname}>`: Блок для відображення ніку користувача та транскрипції звукового введення.

`<div className={styles.user__bottomButtons}>`: Блок з кнопками та іншими елементами.

`<div className={styles.user__micImageBlock}>`: Блок для відображення іконки мікрофона.

`<Image className={styles.user__micro} onClick={() => setMicState((mic) => !mic)} src="/microphone.svg" alt="Microphone" width={50} height={50}></Image>`: Зображення мікрофона, на яке можна клікнути для управління станом мікрофона.

`<button ref={enableWebcamButton} id="webcamButton" className={styles.user__camImageBlock} onClick={enableCam}>`: Кнопка для управління веб-камерою.

`<Image className={styles.user__camera} src="/camera.svg" alt="Camera" width={50} height={50}></Image>`: Зображення камери.

`<div>`: Блок із текстовим та кнопковим виведенням стану мікрофона та кнопками для управління аудіо-розпізнаванням.

`<p>Microphone: {listening ? "on" : "off"}</p>`: Текстовий вивід стану мікрофона.

`<button onClick={() => onStartListen()}>Start</button>`: Кнопка для початку аудіо-розпізнавання.

`<button onClick={() => onStopListen()}>Stop</button>`: Кнопка для зупинки аудіо-розпізнавання.

`<button onClick={resetTranscript}>Reset</button>`: Кнопка для скидання транскрипції аудіо-введення.

Цей код використовується для створення інтерфейсу, який включає в себе відображення веб-камери, розпізнавання жестів та аудіо-розпізнавання.

6.3 Розробка чата для кімнати користувачів

```
<div className={styles.chat}>
      <ul className={styles.chat__messageList}>
        {chatItems.map((item, index) => (
```

```
              <li key={index} className={styles.chat__message}>
                <Image
                  className={styles.chat__avatar}
                  alt="Chat Avatar"
                  src="/avatar.jpg"
                  width={50}
                  height={67}
                />
                <div className={styles.chat__userMessage}>
                  <p
className={styles.chat__nickname}>broke.oli</p>
                  <p className={styles.chat__text}>{item}</p>
                </div>
              </li>
            ))}
          </ul>
          <div className={styles.chat__input}>
            <input ref={inputRef} type="text" placeholder="Input
your message..." />
            <button onClick={() => sendButtonClick()}
type="submit">
              <Image
                src="/send-message.png"
                alt="Send Message"
                width={20}
                height={20}
              />
            </button>
          </div>
        </div>
```

Розглянемо кожен елемент окремо та пояснимо їх функції та призначення:

`<div className={styles.chat}>`: Зовнішній контейнер для компоненту чату.

`<ul className={styles.chat__messageList}>`: Невпорядкований список для відображення повідомлень чату.

`{chatItems.map((item, index) => (</li>))}`: Використання методу map для ітерації через елементи масиву chatItems та створення елементів списку для кожного повідомлення.

`<li key={index} className={styles.chat__message}>`: Кожне повідомлення є окремим елементом списку. Ключ key={index} вказує на унікальний ідентифікатор для React, щоб ефективно оновлювати DOM при зміні.

`<Image className={styles.chat__avatar} alt="Chat Avatar" src="/avatar.jpg" width={50} height={67}>`: Зображення аватару користувача, яке відображається поруч з кожним повідомленням.

`<div className={styles.chat__userMessage}>`: Контейнер для інформації про користувача та тексту повідомлення.

`<p className={styles.chat__nickname}>broke.oli</p>`: Нік користувача, який відображається поруч з аватаром.

`<p className={styles.chat__text}>{item}</p>`: Текстовий контент повідомлення, де item - це текстовий зміст конкретного повідомлення.

`<div className={styles.chat__input}>`: Контейнер для введення нового повідомлення.

`<input ref={inputRef} type="text" placeholder="Input your message..."></input>`: Поле введення для введення тексту нового повідомлення. ref={inputRef} використовується для отримання посилання на елемент в React.

`<button onClick={() => sendButtonClick()} type="submit"></button>`: Кнопка для відправлення нового повідомлення. При кліці викликається функція sendButtonClick().

`<Image src="/send-message.png" alt="Send Message" width={20} height={20}></Image>`: Зображення, що відображає кнопку відправлення повідомлення.

Цей код використовується для створення компонента чату, який включає в себе відображення списку повідомлень та можливість введення нових повідомлень.

7 РЕАЛІЗАЦІЯ ФУНКЦІОНАЛЬНОСТІ СУРДОПЕРЕКЛАДАЧА

У цьому розділі розглянемо процес розробки та реалізації ключової функціональності сурдоперекладача. Описані нижче етапи дозволять краще розуміти, як розроблений мною інтерактивний інструмент дозволить ефективно спілкуватися особам із вадами слуху.

7.1 Розпізнавання жестів з використанням Mediapipe Gesture Recognition

Бібліотека Mediapipe Gesture Recognition є потужним інструментом для визначення та розпізнавання жестів, що базується на вхідних даних від веб-камери чи іншого джерела відео. Розроблена командою Google, вона використовує глибокі нейронні мережі для точного аналізу рухів та конфігурації рук користувача.

Ця технологія дозволяє визначати не лише прості жести, такі як махання рукою чи замах, але й складні комбінації рухів, забезпечуючи високоточність та надійність в розпізнаванні.

Інтеграція в Проект:

Підключення та налаштування бібліотеки Mediapipe Gesture Recognition відбувається на етапі розробки. Починаємо з імпорту необхідних модулів та класів, які надає бібліотека. Наступним кроком є налаштування параметрів для забезпечення взаємодії із веб-камерою та отримання відеопотоку.

Для інтеграції з веб-додатком важливо визначити, які саме жести або комбінації жестів будуть розпізнаватися та які функції вони викличуть. Це визначається за допомогою програмного коду, який визначає поведінку системи при розпізнаванні конкретного жесту.

Всі ці кроки спрямовані на те, щоб гарантувати ефективну інтеграцію та використання бібліотеки Mediapipe Gesture Recognition у вашому проекті,

надаючи зручний та надійний інтерфейс для розпізнавання жестів користувача.

7.1.1 Реалізація алгоритмів для точного визначення жестів

На першому етапі обробки відбувається передача вхідних даних, які надходять з веб-камери, в бібліотеку Mediapipe Gesture Recognition. Ці дані включають в себе відеопотік, який відображає рухи та конфігурацію рук користувача. Завдяки використанню алгоритмів глибокого навчання, бібліотека здатна точно аналізувати ці дані та виявляти ключові особливості рухів.

На другому етапі розробляються алгоритми, які дозволяють системі визначати конкретні жести, що виконуються користувачем. Ці алгоритми спроектовані для класифікації різних типів рухів рук, таких як махання, замах, закручування тощо. Вони забезпечують точне визначення жестів та їх інтерпретацію як команд для системи.

Один із ключових елементів цього етапу - розуміння контексту використання конкретного жесту. Наприклад, один і той же рух може мати різне значення залежно від ситуації. Алгоритми повинні враховувати ці варіації для точного та надійного розпізнавання жестів у реальному часі.

Всі ці етапи спрямовані на те, щоб забезпечити високоточне та швидке визначення жестів користувача, роблячи систему інтерактивною та легкою у використанні.

7.1.2 Інтеграція жестового управління у веб-додаток

Код реалізації розпізнавання жестів з веб-камери користувача має такий вигляд:

```
import {
  GestureRecognizer,
```

```
  FilesetResolver,
  DrawingUtils,
} from "@mediapipe/tasks-vision";
let gestureRecognizer: GestureRecognizer;
  let runningMode = "VIDEO";
  let enableWebcamButton: HTMLButtonElement;
  let webcamRunning: Boolean = false;
  const videoHeight = "100%";
  const videoWidth = "100%";
  let gestureRecIsLoad = false;
  const canvasElement = React.useRef(null);
  let canvasCtx;
```

GestureRecognizer, FilesetResolver і DrawingUtils імпортуються з бібліотеки @mediapipe/tasks-vision.

Визначаються змінні, такі як gestureRecognizer, runningMode, enableWebcamButton, і інші, які будуть використовуватися в подальших частинах коду.

```
React.useEffect(() => {
    const createGestureRecognizer = async () => {
      const vision = await FilesetResolver.forVisionTasks(
        "https://cdn.jsdelivr.net/npm/@mediapipe/tasks-
vision@0.10.3/wasm"
      );
      gestureRecognizer = await
GestureRecognizer.createFromOptions(vision, {
        baseOptions: {
          modelAssetPath:
            "https://storage.googleapis.com/mediapipe-
models/gesture_recognizer/gesture_recognizer/float16/1/gesture_r
ecognizer.task",
          delegate: "GPU",
        },
        runningMode: runningMode,
      });
      gestureRecIsLoad = true;
    };

    createGestureRecognizer();

    return () => {
      gestureRecognizer.close();
```

```
    };
  }, []);
```

useEffect викликає функцію createGestureRecognizer для ініціалізації GestureRecognizer.

FilesetResolver.forVisionTasks використовується для завантаження задач видіння.

Створюється екземпляр GestureRecognizer з вказаною моделлю та параметрами.

```
  React.useEffect(() => {
    if (canvasElement.current) {
      canvasCtx = canvasElement.current.getContext("2d");
      console.log("canvasCtx", canvasCtx);
    }
  }, [canvasElement]);
```

useEffect слідкує за змінами посилання на канвас та отримує посилання на контекст канвасу.

```
const video = React.useRef(null);

  const gestureOutput = React.useRef(null);

  enableWebcamButton = React.useRef(null);
```

Використовуються useRef для створення посилань на елементи відео, вихід жестів та кнопку включення веб-камери.

```
  function enableCam(event) {
    setWebcamState((webcam) => !webcam);

    if (!gestureRecognizer) {
      alert("Please wait for gestureRecognizer to load");
      return;
    }

    if (webcamRunning === true) {
      webcamRunning = false;
    } else {
      webcamRunning = true;
```

```
}
```

Змінна webcamRunning служить для відстеження стану включення/вимкнення веб-камери.

Якщо веб-камера включена (webcamRunning === true), то вона вимикається (становиться false), і навпаки.

```
const constraints = {
  video: true,
};
```

Визначає об'єкт constraints, який містить параметри для отримання потоку відео. У цьому випадку вказано, що потрібен доступ до відеокамери.

```
navigator.mediaDevices.getUserMedia(constraints).then(function
(stream) {
          video.current.srcObject = stream;
          video.current.addEventListener("loadeddata",
predictWebcam);
        });
      }
```

Використовує navigator.mediaDevices.getUserMedia для отримання доступу до відеокамери і створення потоку stream.

`video.current.srcObject = stream;` встановлює отриманий потік як джерело для елементу відео (video.current).

`video.current.addEventListener("loadeddata", predictWebcam);` додає слухача подій, який викликає функцію predictWebcam, коли дані від відеопотоку готові до використання.

Функція enableCam призначена для включення та вимкнення веб-камери, а також ініціалізації потоку відео та налаштування подій для подальшого визначення жестів.

Розглянемо цю частину коду поетапно:

`setWebcamState((webcam) => !webcam);`: Використовує функцію стану setWebcamState, яка оновлює стан webcamState, змінюючи його на протилежне значення.

Це використовується для відслідковування та оновлення стану включення/вимкнення веб-камери.

`if (!gestureRecognizer) { alert("Please wait for gestureRecognizer to load"); return; }`: Перевіряє, чи gestureRecognizer ініціалізований. Якщо ні, виводить повідомлення про те, що користувач повинен зачекати на завантаження gestureRecognizer, та припиняє виконання функції.

```
let lastVideoTime = -1;
let results = undefined;
async function predictWebcam() {
  const webcamElement = document.getElementById("webcam");
```

Отримує елемент веб-камери за його ідентифікатором з DOM.

```
if (runningMode === "IMAGE") {
  runningMode = "VIDEO";
  await gestureRecognizer.setOptions({ runningMode:
"VIDEO" });
}
```

Перевіряє, чи поточний режим роботи - "IMAGE" (зображення). Якщо так, змінює режим на "VIDEO" і оновлює опції gestureRecognizer для визначення режиму "VIDEO".

```
let nowInMs = Date.now();
if (video.current.currentTime !== lastVideoTime) {
  lastVideoTime = video.current.currentTime;
  results =
gestureRecognizer.recognizeForVideo(video.current, nowInMs);
}
```

Визначає поточний час та перевіряє, чи відбулися зміни в часі відео. Якщо так, викликає recognizeForVideo, щоб отримати результати розпізнавання для поточного кадру відео.

```
canvasCtx.save();
canvasCtx.clearRect(
  0,
  0,
  canvasElement.current.width,
  canvasElement.current.height
);
```

Зберігає стан канвасу та очищає його, готуючи його для відображення нових результатів.

```
const drawingUtils = new DrawingUtils(canvasCtx);

canvasElement.current.style.height = videoHeight;
webcamElement.style.height = videoHeight;
canvasElement.current.style.width = videoWidth;
webcamElement.style.width = videoWidth;

if (results.landmarks) {
  for (const landmarks of results.landmarks) {
    drawingUtils.drawConnectors(
      landmarks,
      GestureRecognizer.HAND_CONNECTIONS,
      {
        color: "#00FF00",
        lineWidth: 5,
      }
    );
    drawingUtils.drawLandmarks(landmarks, {
      color: "#FF0000",
      lineWidth: 2,
    });
  }
}
```

Якщо є результати знайдених ключових точок (landmarks), то вони відображаються на канвасі за допомогою drawConnectors та drawLandmarks з об'єктом drawingUtils.

```
          canvasCtx.restore();
          if (results.gestures.length > 0) {
            gestureOutput.current.style.display = "block";
            gestureOutput.current.style.width = videoWidth;
            const categoryName =
results.gestures[0][0].categoryName;
            const categoryScore = parseFloat(
              results.gestures[0][0].score * 100
            ).toFixed(2);
            const handedness =
results.handednesses[0][0].displayName;
            gestureOutput.current.innerText = `GestureRecognizer:
${categoryName}\n Confidence: ${categoryScore} %\n Handedness:
${handedness}`;
          } else {
            gestureOutput.current.style.display = "none";
          }
```

Якщо існують результати визначення жестів, вони виводяться на екран у вигляді тексту з використанням innerText. Якщо результатів немає, відображення тексту приховується.

```
          if (webcamRunning === true) {
            window.requestAnimationFrame(predictWebcam);
          }
        }
```

Якщо веб-камера включена, то викликає predictWebcam у такт з кадрами відео за допомогою window.requestAnimationFrame, забезпечуючи рекурсивний виклик для подальшого визначення жестів.

7.2 Розпізнавання Мовлення за допомогою React Speech Recognition

Для реалізації функціоналу розпізнавання мовлення в проекті використовується пакет react-speech-recognition. Цей пакет дозволяє легко інтегрувати розпізнавання мовлення в React-додаток, забезпечуючи зручний та ефективний інтерфейс для взаємодії з користувачем.

7.2.1 Навчання додатку розуміти та інтерпретувати мовлення користувача в реальному часі

Пакет react-speech-recognition грає ключову роль у навчанні додатку розпізнавати та ефективно інтерпретувати мовлення користувача в режимі реального часу. Розглянемо цей процес більш детально:

a) Аналіз Звукових Хвиль:

Починаємо з отримання звукового сигналу від мікрофона користувача.

react-speech-recognition використовує алгоритми для аналізу звукових хвиль та виділення характеристик мовлення.

b) Ідентифікація Мовлення:

Проходить етап ідентифікації мовлення, під час якого визначається, що отримані звукові дані є голосовим сигналом.

c) Перетворення у Текст:

За допомогою вбудованих алгоритмів або облачних сервісів, звуковий сигнал перетворюється в текстовий вигляд.

Текстова інтерпретація може включати в себе визначення конкретних слів, речень або навіть команд, вимовлених користувачем.

d) Адаптація та Оптимізація:

Система може вдосконалювати свою точність через навчання на прикладах, зокрема через корекцію розпізнаних слів або фраз користувача.

Оптимізація включає в себе аналіз контексту, адаптацію до особливостей мовлення конкретного користувача та урахування варіативності мовлення.

e) Синхронізація з Додатком:

Отриманий текстовий результат інтегрується у веб-додаток, забезпечуючи його доступність для подальшого використання та відображення.

Цей процес гарантує, що додаток може інтерпретувати мовлення користувача миттєво та з високою точністю, створюючи ефективний та доступний інтерфейс для осіб із сурдитністю.

7.2.2 Інтеграція функціоналу розпізнавання мовлення у інтерфейс сурдоперекладача

```
import SpeechRecognition, {
  useSpeechRecognition,
} from "react-speech-recognition";
```

Цей рядок імпортує основний об'єкт SpeechRecognition та функції з бібліотеки react-speech-recognition.

```
const dispatch = useDispatch();
  const [micState, setMicState] = React.useState(false);
```

Створюється об'єкт dispatch за допомогою Redux useDispatch для взаємодії зі стором Redux.

Встановлюється стан micState та функція його оновлення setMicState за допомогою React Hook useState. Цей стан використовується для відстеження стану мікрофона (включений або вимкнений).

```
const {
    transcript,
    listening,
    resetTranscript,
    browserSupportsSpeechRecognition,
    browserSupportsContinuousListening,
  } = useSpeechRecognition();
```

Використання useSpeechRecognition для отримання об'єкта з різними властивостями та функціями для керування розпізнаванням мовлення.

```
      if (!browserSupportsSpeechRecognition) {
        return <span>Browser doesn't support speech
recognition.</span>;
      }
```

Перевірка, чи браузер підтримує функціонал розпізнавання мовлення. Якщо ні, повертається повідомлення про непідтримку.

```
      const onStartListen = () => {
        if (micState) {
          if (browserSupportsContinuousListening) {
            SpeechRecognition.startListening({ continuous: true
});
          } else {
            SpeechRecognition.startListening();
          }
        } else {
          console.log("Ваш микрофон выключен");
        }
      };
```

Функція onStartListen викликається при спробі розпочати слухання мовлення.

Перевіряє стан micState, якщо він включений, то викликає SpeechRecognition.startListening() для розпочатку слухання мовлення. Якщо підтримується режим безперервного слухання (browserSupportsContinuousListening), встановлюється параметр continuous: true.

```
      const onStopListen = () => {
        dispatch(addChatItem(transcript));
        resetTranscript();
        SpeechRecognition.stopListening();
      };
```

Функція onStopListen викликається при спробі зупинити слухання мовлення.

Додає текст розпізнаного мовлення до чату через Redux за допомогою dispatch.

Очищає текст розпізнаного мовлення за допомогою resetTranscript.

Викликає SpeechRecognition.stopListening() для зупинки слухання.

7.3 Реалізація спілкування в чаті на сайті

Розглянемо фрагмент коду який є частиною реалізації стану та логіки для чату в моєму React Redux додатку. У React Redux, стан зберігається в спільному об'єкті, відомому як "стор", який може бути розбитий на окремі частини, відомі як "слайси". Кожен слайс відповідає за конкретний аспект стану та його логіку.

В даному випадку, використовуючи @reduxjs/toolkit, я створюю слайс для чату, який містить інформацію про повідомлення та необхідні функції для їхньої модифікації. Розглянемо кожну частину коду:

```
    import { PayloadAction, createSlice } from
"@reduxjs/toolkit"
    import { ChatState } from "./types";
```

Імпортуємо необхідні функції та об'єкти з @reduxjs/toolkit для роботи зі стором та діями.

```
    const initialState: ChatState = {
        chatItems: ["Hey There!", "I'm Oli", "How are u?"]
    }
```

Встановлюємо початковий стан для слайсу, який містить масив chatItems із початковими повідомленнями.

```
    const chatSlice = createSlice({
        name: "chat",
        initialState,
        reducers:{
```

```
        addChatItem(state, action: PayloadAction<string>){
            state.chatItems.push(action.payload);
        }
    }
})
```

Використовуємо createSlice для створення слайсу "chat". Вказуємо ім'я слайсу, початковий стан та функції зменшення стану, зокрема addChatItem, яка додає нове повідомлення до чату.

```
export const {addChatItem} = chatSlice.actions;
```

Експортуємо дії зі слайсу, щоб їх можна було використовувати в компонентах React.

```
export default chatSlice.reducer;
```

Експортуємо редуктор, який буде обробляти ці дії та змінювати стан слайсу.

Використовуючи цей слайс, ми можете легко додавати та управляти повідомленнями чату в моєму React Redux додатку. Це важлива частина для організації та управління станом додатку.

8 ПОДАЛЬШІ НАПРЯМИ РОЗВИТКУ СУРДОПЕРЕКЛАДАЧА

Інтерактивний сурдоперекладач, як новаторський продукт, завжди прагне до вдосконалення та розвитку. Для досягнення цієї мети рекомендується звернутися до наступних аспектів:

a) Оптимізація Розпізнавання Жестів:

Проведення додаткових досліджень та розробка нових методів для поліпшення точності та реакції системи на жести користувачів. Точніше та ефективніше розпізнавання жестів робить взаємодію із сурдитними користувачами більш природною та інтуїтивно зрозумілою.

b) Розширення Жестового Словника:

Зростання функціоналу за допомогою додавання нових жестів та їх інтерпретації. Це дозволить розширити спектр команд, які може виконувати інтерактивний сурдоперекладач, що робить його більш адаптованим до індивідуальних потреб користувачів.

c) Вдосконалення Мовленнєвого Розпізнавання:

Постійне оновлення моделей розпізнавання мовлення для підтримки нових мов та покращення точності ідентифікації слів. Інтеграція альтернативних пакетів для визначення емоцій та інтонацій у мовленні.

Інтерактивний сурдоперекладач має потенціал для застосування у різних сферах. Розглянемо перспективи його використання:

a) Освітній Сектор:

В освітньому секторі інтерактивний сурдоперекладач може бути використаний для навчання жестової мови та полегшення взаєморозуміння між викладачами та сурдитними студентами. Розробка спеціальних курсів та інтерактивних матеріалів може покращити навчання та комунікацію.

b) Медичний Сектор:

У медичних закладах інтерактивний сурдоперекладач може служити важливим інструментом для поліпшення комунікації між медичним

персоналом та пацієнтами із сурдитністю. Можливість докладного обговорення симптомів та планів лікування сприяє вдосконаленню медичного обслуговування.

c) Бізнес та Комерція:

В сфері бізнесу та комерції інтерактивний сурдоперекладач може бути використаний для поліпшення комунікації на робочому місці та під час презентацій. Взаємодія з клієнтами та партнерами стає більш ефективною та інклюзивною.

d) Технологічний Прогрес:

У контексті технологічного прогресу інтерактивний сурдоперекладач може бути інтегрований з розумними пристроями, що дозволяє взаємодіяти з технологією за допомогою жестів та мовлення. Розвиток штучного інтелекту та використання продукту для інших аспектів життя покращує його універсальність.

ВИСНОВКИ

У рамках даної дипломної роботи було проведено розробку інноваційного проекту – інтерактивного двобічного аудіо-відео сурдоперекладача. Даний продукт є результатом використання широкого спектру сучасних технологій та інструментів, що включають у себе HTML, CSS, JS, React, Next.js, SCSS, Mediapipe Gesture Recognition та react-speech-recognition.

Розроблений веб-сайт визначається не лише ефективним та естетичним інтерфейсом, але й передовими можливостями в сфері взаємодії з користувачем, особливо для осіб із вадами слуху. Використання технологій React та Next.js забезпечує швидке завантаження та відзивчивість веб-додатку, тоді як SCSS дозволяє створювати динамічний та сучасний дизайн.

Важливою складовою функціоналу сурдоперекладача є розпізнавання жестів за допомогою Mediapipe Gesture Recognition та розпізнавання мовлення з використанням пакету react-speech-recognition. Ці технології додають інтерактивність та доступність до проекту для різних категорій користувачів.

Отже, реалізація інтерактивного сурдоперекладача відкриває нові можливості для поліпшення комунікації та взаєморозуміння людей із вадами слуху. Проект не лише демонструє технічну компетентність у розробці веб-додатків, але й має потенціал покращити якість життя осіб із сурдитністю, надаючи їм нові інструменти для взаємодії та спілкування.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Хайкин, С. Нейронные сети: полный курс / С. Хайкин. - М.: Диалектика, 2019. - 1104 с.
2. Редько, В.Г. Эволюция, нейронные сети, интеллект: Модели и концепции эволюционной кибернетики / В.Г. Редько. - М.: Ленанд, 2019. - 224 с.
3. Гудфеллоу І., Бенджіо Й., Курвіль А. Глибоке навчання / І. Гудфеллоу, Й. Бенджіо, А. Курвіль. – М.: ДМК Пресс, 2017. – 652 с.
4. Жесты и мимика. Психология невербального общения / Под ред. В.А. Лабунской. – Ростов н/Д.: Феникс, 2009. – 344 с.
5. Распознавание и компьютерное зрение. Современные алгоритмы / Под ред. С.С. Кугушева. – М.: КУДИЦ-ПРЕСС, 2020. – 688 с.
6. Бэнкс А., Порселло Э. React и Redux. Функциональная веб-разработка / А. Бэнкс, Э. Порселло. – СПб.: Питер, 2019. – 464 с.
7. Адамс К. Основы Next.js / К. Адамс. – М.: Вильямс, 2020. – 256 с.
8. React (JavaScript-бібліотека) // Wikipedia. – 2022. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/React_(JavaScript-бібліотека).
9. Next.js // Wikipedia. – 2022. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Next.js.
10. Mediapipe // Wikipedia. – 2022. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Mediapipe.
11. react-speech-recognition // GitHub. – 2022. – Режим доступу до ресурсу: https://github.com/JamesBrill/react-speech-recognition.