ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської кваліфікаційної роботи

на ступінь вищої освіти бакалавр

на тему: **«РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ТРАНСКРИБУВАННЯ ТА СИНТЕЗУ МОВИ ЗА ДОПОМОГОЮ HTML, CSS, JS, PYTHON»**

Виконав: студент 4 курсу, групи ПД-44

спеціальності 121 Інженерія програмного забезпечення

(шифр і назва спеціальності)

Демиденко Н.О.

(прізвище та ініціали)

Керівник Негоденко О.В.

(прізвище та ініціали)

Рецензент

(прізвище та ініціали)

Нормоконтроль

(прізвище та ініціали)

Київ – 2021

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти - «Бакалавр»

Спеціальність -121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного
забезпечення
_____О.В. Негоденко
«____»_____2021 року

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

ДЕМИДЕНКУ НІКІТІ ОЛЕКСАНДРОВИЧУ

1.Тема роботи: «Розробка програмного забезпечення для транскрибування та синтезу мови за допомогою HTML, CSS, JS, Python»
Керівник роботи Негоденко Олена Василівна, доцент, кандидат технічних наук затверджені наказом вищого навчального закладу від — «12» березня 2021 року №65.

2. Строк подання студентом роботи 01.06.2021

3. Вхідні дані до роботи:

3.1. Середовище розробки PyCharm Community Edition 2020.3.3

3.2. Django Framework

3.3. GTTS-бібліотека

3.4. Науково-технічна література, пов’язана з синтезом і розпізнаванням мови

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити).

4.1. Аналіз предметної області

4.2. Дослідження засобів реалізації

4.3. Розробка функціоналу ресурсу

5. Перелік графічного матеріалу

5.1.1. Діаграма варіантів використання

5.1.2. Діаграма класів програмного продукту

5.1.3. Графічний інтерфейс користувача

5.1.4. Огляд можливостей розробленого веб-сервісу

6. Дата видачі завдання 19.04.2021

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	19.04-23.04	
2	Аналіз існуючих прототипів	24.04-26.04	
3	Дослідження програмних засобів	27.04-3.05	
4	Моделювання об’єкту проектування	4.05-7.05	
5	Розробка функціоналу сайту	8.05-17.05	
6	Вступ, висновки, реферат	18.05-20.05	
7	Розробка презентації застосунку	21.05-23.05	
8	Попередній захист роботи	15.05.21	

Студент

Керівник роботи

РЕФЕРАТ

Текстова частина бакалаврської роботи 65 с., 9 рис., 2 табл., 61 джерел.

Ключеві слова: PyCharm, Python, HTML, CSS, JS, Django, синтез мови, розпізнавання мови

Об'єкт дослідження – підвищення ефективності перетворення інформації для транскрибуванні та синтезу мови.

Предмет дослідження – методи та засоби забезпечення ефективного перетворення інформації для синтезу і розпінавання мови.

Мета роботи – підвищення ефективності транскрибування та синтезу мови шляхом розробки програмного забезпечення за допомогою HTML, CSS, JS та Python.

Методи дослідження – методи теорії інформації, методи структурного аналізу і проектування, методи розробки програмного забезпечення, методи тестування, валідації та верифікації програмного забезпечення.

У відповідності з поставленою метою для вирішення технічної проблеми в роботі вирішено такі завдання:

1. Проаналізувати предметну область;
2. Обрати засоби реалізації;
3. Розробити програмне забезпечення.

Використання даного програмного забезпечення дозволить будь-якій людині проводити транскрибування та синтез мови методом text-to-speech.

Галузь використання – ресурс може використовувати будь-яка людина, яка має необхідність в фукнціоналі, який надає розроблене програмне забезпечення.

ЗМІСТ

ВСТУП

Актуальність дослідження. Кожного дня автоматизація різноманітних процесів стає все більш актуальною тенденцією в сучасному світі. Кожний процес автоматизують і перетворюють на процеси, що вимагають менше часу або взагалі можуть обійтися без фізичного втручання людини.

На хвилі спрощення монотонних процесів з’явилися загальнодоступні алгоритми розпізнавання мови і її озвучування з тексту, які покликані спростити перетворення інформації з текстового формату у звук і навпаки.

Виходячи з актуальності описаного явища, метою даної роботи було обрано підвищення ефективності транскрибування та синтезу мови шляхом розробки програмного забезпечення за допомогою HTML, CSS, JS та Python.

Для виконання поставленої мети слід виконати наступні завдання:

1. Проаналізувати предметну область;
2. Обрати засоби реалізації;
3. Розробити програмне забезпечення.

Об'єкт дослідження – підвищення ефективності перетворення інформації для транскрибуванні та синтезу мови.

Предмет дослідження – методи та засоби забезпечення ефективного перетворення інформації для синтезу і розпінавання мови.

Методи дослідження – методи теорії інформації, методи структурного аналізу і проектування, методи розробки програмного забезпечення, методи тестування, валідації та верифікації програмного забезпечення.

Практичне значення одержаних результатів. В результаті роботи отримано ресурс, який допомагає транскрибувати та синтезувати усну мову з можливістю додавання будь-яких мов. Такий ресурс може стати невід’ємною частиною більшого проекту, наприклад комерційної мережі для спілкування між носіями різних мов.

Особистий внесок. Підібрано відповідні технології, які якнайкраще сприяють розкриттю повного функціоналу програмного забезпечення, створено модулі взаємодії між технологіями, власноруч написані всі сторінки і взаємодії між ними.

Результати роботи. Матеріали дипломного проекту можуть сприяти розвиту технологій транскрибування та синтезу мови, підвищенню ефективності існуючих систем подібного роду.

Апробація роботи. Сама робота та її результати були опубліковані на конференції “Науково-технічна конференція «Застосування програмного забезпечення в ІКТ»” від 10 лютого 2021 року. Демиденко Н.О. “ВЕБ-САЙТ ДЛЯ ТРАНСКРИБАЦІЇ ТА СИНТЕЗУ МОВИ” / Демиденко Н.О. // Застосування програмного забезпечення в ІКТ: Матеріали науково-технічної конференції «Застосування програмного забезпечення в ІКТ». Збірник тез. 10.02.2021, ДУТ, м. Київ — К.: ДУТ, 2021. — С. 58.
http://www.dut.edu.ua/uploads/n_9058_51926054.pdf

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Розпізнавання мови

Розпізнавання мови - це міждисциплінарна підполя інформатики та обчислювальної лінгвістики, яка розробляє методології та технології, що дозволяють розпізнавати та перекладати розмовну мову в текст комп’ютерами. Він також відомий як автоматичне розпізнавання мови (ASR), комп’ютерне розпізнавання мови або перетворення тексту в текст (STT). Він включає знання та дослідження в галузі інформатики, лінгвістики та обчислювальної техніки.

Деякі системи розпізнавання мовлення вимагають "навчання" (також зване "зарахуванням"), коли окремий мовець читає текст або ізольований словниковий запас у систему. Система аналізує конкретний голос людини і використовує його для точної настройки розпізнавання мови цієї людини, що призводить до підвищення точності. Системи, що не використовують навчання, називаються "незалежними від оратора" [1] системами. Системи, що використовують навчання, називаються «залежними від мовця».

Програми розпізнавання мови включають голосові користувальницькі інтерфейси, такі як голосовий набір (наприклад, "зателефонувати додому"), маршрутизацію викликів (наприклад, "Я хотів би зробити дзвінок для збору"), керування приладом domotic, пошук ключових слів (наприклад, знайти подкаст, де певні слова говорили), простий введення даних (наприклад, введення номера кредитної картки), підготовка структурованих документів (наприклад, радіологічний звіт), визначення характеристик динаміка, [2] обробка мови в текст (наприклад, текстові процесори або електронні листи), і літаки (зазвичай їх називають прямим голосовим введенням).

Термін розпізнавання голосу [3] [4] [5] або ідентифікація динаміка [6] [7] [8] стосується ідентифікації мовця, а не того, що вони говорять. Розпізнавання мовця може спростити завдання перекладу мовлення в системах, які були навчені на

голос конкретної людини, або його можна використовувати для автентифікації або перевірки особи спікера як частини процесу безпеки.

З технологічної точки зору, розпізнавання мови має давню історію з кількома хвилями основних нововведень. Зовсім недавно галузь отримала вигоду від успіхів у глибокому навчанні та обробці великих даних. Про досягнення свідчить не лише сплеск академічних робіт, опублікованих у цій галузі, але, що ще важливіше, прийняття у світовій галузі різноманітних методів глибокого навчання при розробці та впровадженні систем розпізнавання мови.

Сучасні системи розпізнавання мови загального призначення базуються на прихованих моделях Маркова. Це статистичні моделі, які виводять послідовність символів або величин. HMM використовуються для розпізнавання мови, оскільки мовний сигнал можна розглядати як кусково нерухомий сигнал або короткочасний стаціонарний сигнал. За короткий проміжок часу (наприклад, 10 мілісекунд) мова може бути апроксимована як нерухомий процес. Мова можна сприймати як модель Маркова для багатьох стохастичних цілей.

Ще одна причина, по якій HMM користуються популярністю, полягає в тому, що їх можна навчити автоматично, їх просто і зручно обчислювати. При розпізнаванні мови прихована модель Маркова виводила б послідовність n-вимірних реально значущих векторів (при цьому n було малим цілим числом, наприклад 10), виводячи один із них кожні 10 мілісекунд. Вектори складались би з цепстральних коефіцієнтів, які отримували, приймаючи перетворення Фур'є короткого часового вікна мови та декорелювавши спектр за допомогою косинусного перетворення, а потім беручи перший (найбільш значущий) коефіцієнт. Прихована модель Маркова, як правило, матиме в кожному стані статистичний розподіл, який є сумішшю діагональної коваріації Гауса, що дасть вірогідність для кожного спостережуваного вектора. Кожне слово, або (для більш загальних систем розпізнавання мови), кожна фонема матиме різний розподіл вихідних даних; прихована модель Маркова для послідовності слів або фонем створюється шляхом об'єднання окремих навчених прихованих моделей Маркова для окремих слів та фонем.

Вище описані основні елементи найпоширенішого підходу до розпізнавання мовлення, заснованого на HMM. Сучасні системи розпізнавання мовлення використовують різні комбінації низки стандартних прийомів для покращення результатів порівняно з основним підходом, описаним вище. Типовій системі великого словника потрібна контекстна залежність для фонем (тому фонеми з різним лівим та правим контекстом мають різну реалізацію, як стверджує HMM); він використовував би цепстральну нормалізацію для нормалізації для різних умов динаміка та запису; для подальшої нормалізації оратора він може використовувати нормалізацію довжини голосового тракту (VTLN) для нормалізації чоловічої та жіночої статі та лінійну регресію з максимальною вірогідністю (MLLR) для більш загальної адаптації оратора.

Характеристики мали б так звані коефіцієнти дельта-дельта-дельта для фіксації мовної динаміки і, крім того, могли б використовувати гетеросцедастичний лінійний дискримінантний аналіз (HLDA); або може пропустити коефіцієнти дельта-дельта-дельта та використовувати сплайсинг та проекцію на основі LDA, що супроводжується, можливо, гетеросцедастичним лінійним дискримінантним аналізом або глобальним напівзв’язаним перетворенням дисперсії (також відомим як лінійне перетворення з максимальною правдоподібністю або MLLT). У багатьох системах використовуються так звані дискримінаційні методи навчання, які відмовляються від чисто статистичного підходу до оцінки параметрів HMM, а замість цього оптимізують деякі пов'язані з класифікацією виміри даних навчання. Прикладами є максимальна взаємна інформація (MMI), мінімальна похибка класифікації (MCE) та мінімальна похибка телефону (MPE).

Декодування мови (термін того, що відбувається, коли система представлена з новим висловлюванням і повинна обчислити найбільш імовірне вихідне речення), мабуть, використовував би алгоритм Вітербі, щоб знайти найкращий шлях, і тут є вибір між динамічним створенням комбінація прихованої моделі Маркова, яка включає як інформацію про акустичну, так і

мовну модель, і заздалегідь статично поєднуючи її (підхід перетворювача кінцевих станів, або FST).

Можливим покращенням декодування є збереження набору хороших кандидатів замість того, щоб просто утримувати найкращого кандидата, а також використання кращої функції підрахунку балів (переоцінка) для оцінки цих хороших кандидатів, щоб ми могли вибрати найкращого відповідно до цього уточненого балу . Набір кандидатів може зберігатися або як список (підхід N-кращого списку), або як підмножина моделей (решітка). Переоцінка зазвичай проводиться шляхом намагання мінімізувати ризик Байєса [58] (або його наближення): Замість того, щоб приймати вихідне речення з максимальною ймовірністю, ми намагаємось взяти речення, яке мінімізує очікуваність даної функції збитків щодо всі можливі транскрипції (тобто, ми беремо речення, яке мінімізує середню відстань до інших можливих речень, зважених за їхньою передбачуваною ймовірністю).

Функція збитків - це, як правило, відстань Левенштейна, хоча це можуть бути різні відстані для конкретних завдань; безліч можливих транскрипцій, звичайно, обрізано, щоб зберегти придатність. Розроблено ефективні алгоритми для оцінки решіток, представлених як зважені перетворювачі кінцевих станів, із відстанями редагування, представлених як перетворювач кінцевих станів, що перевіряють певні припущення. [59]

1.2 Синтез мови

Синтез мови - це штучне виробництво людської мови. Комп’ютерна система, яка використовується для цієї мети, називається мовним комп’ютером або синтезатором мови, і може бути впроваджена в програмні чи апаратні продукти. Система перетворення тексту в мовлення (TTS) перетворює текст звичайної мови в мовлення; інші системи надають символічні мовні уявлення, такі як фонетичні транскрипції у мові. [1]

Синтезовану промову можна створити, об’єднавши фрагменти записаної мови, які зберігаються в базі даних. Системи відрізняються за розміром збережених мовних одиниць; система, що зберігає телефони або дифони, забезпечує найбільший діапазон виведення, але може мати недостатню чіткість. Для конкретних доменів використання зберігання цілих слів або речень забезпечує якісний вихід. Крім того, синтезатор може включати модель голосового тракту та інші характеристики голосу людини, щоб створити повністю "синтетичний" голосовий вихід. [2]

Якість синтезатора мови оцінюється за його подібністю до людського голосу та за його здатністю чітко розуміти. Зрозуміла програма перетворення тексту в мову дозволяє людям з вадами зору або з порушенням читання слухати написані слова на домашньому комп’ютері. Багато комп’ютерних операційних систем включають синтезатори мови з початку 1990-х.

Система перетворення тексту в мовлення (або "механізм") складається з двох частин: [3] інтерфейс та інтерфейс. У інтерфейсу є два основних завдання. По-перше, він перетворює необроблений текст, що містить такі символи, як цифри та скорочення, в еквівалент виписаних слів. Цей процес часто називають нормалізацією тексту, попередньою обробкою або токенізацією. Потім інтерфейс присвоює фонетичні транскрипції кожному слову, розділяє та позначає текст на просодичні одиниці, такі як фрази, речення та речення. Процес присвоєння фонетичних транскрипцій словам називається перетворенням тексту в фонему або перетворенням графеми в фонему. Фонетична транскрипція та інформація про просодію разом складають символічне мовне подання, яке виводиться на передній панелі. Бек-енд, який часто називають синтезатором, перетворює символічне мовне подання у звук.

У певних системах ця частина включає обчислення цільової просодії (контур висоти, тривалість фонеми) [4], яка потім накладається на вихідну мову.

Найважливішими якостями системи синтезу мовлення є природність та зрозумілість. [26] Натуральність описує, наскільки близький результат звучить як людська мова, тоді як зрозумілість - це легкість розуміння результату. Ідеальний

синтезатор мови є як природним, так і зрозумілим. Системи синтезу мовлення зазвичай намагаються максимізувати обидві характеристики.

Дві основні технології, що генерують синтетичні мовні форми, - це об’єднаний синтез та синтез формантів. Кожна технологія має сильні та слабкі сторони, і передбачуване використання системи синтезу, як правило, визначає, який підхід застосовується.

Конкатенативний синтез заснований на об'єднанні (або об'єднанні) сегментів записаного мовлення. Як правило, конкатенативний синтез виробляє найбільш природне звучання синтезованої мови. Однак відмінності між природними варіаціями мови та природою автоматизованих методів сегментування сигналів іноді призводять до чутних збоїв у виведенні. Існує три основних підтипи конкатенативного синтезу.

Синтез виділення одиниць використовує великі бази даних записаної мови. Під час створення бази даних кожне записане висловлення сегментується на деякі або всі наступні: окремі телефони, дзвінки, напівтелефони, склади, морфеми, слова, фрази та речення. Як правило, поділ на сегменти здійснюється за допомогою спеціально модифікованого розпізнавача мови, встановленого в режим "вимушеного вирівнювання" з деякою ручною корекцією згодом, з використанням візуальних зображень, таких як форма сигналу та спектрограма. [27] Потім створюється індекс одиниць у базі даних мовлення на основі параметрів сегментації та акустики, таких як основна частота (висота тону), тривалість, положення в складі та сусідні телефони. Під час виконання бажане цільове висловлювання створюється шляхом визначення найкращого ланцюжка одиниць-кандидатів з бази даних (вибір одиниць). Зазвичай цей процес досягається за допомогою спеціально зваженого дерева рішень.

Вибір одиниці забезпечує найбільшу природність, оскільки він застосовує лише невелику кількість цифрової обробки сигналу (DSP) до записаної мови. DSP часто робить записаний звук мовою менш природним, хоча деякі системи використовують невелику кількість обробки сигналу в точці конкатенації, щоб згладити форму сигналу. Результати найкращих систем вибору одиниць часто

неможливо відрізнити від справжніх людських голосів, особливо в контексті, на який була налаштована система TTS. Однак максимальна природність, як правило, вимагає, щоб бази даних мовлення з вибором одиниць були дуже великими, в деяких системах від гігабайт записаних даних, що представляє десятки годин мови. [28] Також відомо, що алгоритми вибору одиниць вибирають сегменти з місця, що призводить до менш ідеального синтезу (наприклад, незначні слова стають незрозумілими), навіть коли в базі даних існує кращий вибір. [29] Нещодавно дослідники запропонували різні автоматизовані методи виявлення неприродних сегментів у системах синтезу мовлення з виділенням одиниць. [30]

Синтез дифонів використовує мінімальну базу даних мови, що містить усі дифони (переходи звуку в звук), що відбуваються в мові. Кількість дифонів залежить від фонотактики мови: наприклад, іспанська має близько 800, а німецька - близько 2500. При синтезі дифонів у базі мовлення міститься лише один приклад кожного дифона. Під час виконання цільова просодія речення накладається на ці мінімальні одиниці за допомогою таких технологій цифрової обробки сигналів, як лінійне передбачувальне кодування, PSOLA [31] або MBROLA. [32] або новіші методи, такі як модифікація висоти тону у вихідній області за допомогою дискретного косинусного перетворення. [33] Синтез дифонів страждає від звукових збігів конкатенативного синтезу та роботизованого характеру синтезу формантів, і має лише деякі переваги будь-якого з підходів, крім малих розмірів. Таким чином, його використання в комерційних програмах зменшується [потрібне цитування], хоча воно продовжує використовуватися в дослідженнях, оскільки існує ряд вільно доступних програмних реалізацій. Раннім прикладом синтезу Діфона є навчальний робот, ліахім, який був винайдений Майклом Дж. Фріменом [34]. Леахім містив інформацію щодо навчальної програми класу та певну біографічну інформацію про 40 учнів, яких було запрограмовано викладати. [35] Його тестували в класі четвертого класу в Бронксі, штат Нью-Йорк [36] [37].

Специфічний для домену синтез поєднує попередньо записані слова та фрази для створення повних висловлювань. Він використовується в додатках, де різноманітність текстів, які система видаватиме, обмежена певним доменом,

наприклад, оголошення про розклад руху або звіти про погоду. [38] Ця технологія дуже проста у впровадженні і тривалий час використовується у комерційних цілях на таких пристроях, як годинник, що говорить, та калькулятори. Рівень природності цих систем може бути дуже високим, оскільки різноманітність типів речень обмежена, і вони тісно відповідають просодії та інтонації оригінальних записів.

Оскільки ці системи обмежені словами та фразами у своїх базах даних, вони не є загальним призначенням і можуть синтезувати лише ті комбінації слів і фраз, за допомогою яких вони були попередньо запрограмовані. Однак поєднання слів у природно розмовній мові все одно може спричинити проблеми, якщо не враховувати багато варіацій. Наприклад, у неротичних діалектах англійської мови "r" у таких словах, як "clear" / ˈklɪə /, як правило, вимовляється лише тоді, коли наступне слово має голосну як першу літеру (наприклад, "clear out" реалізується як / ˌklɪəɹˈʌʊt /). Так само у французькій мові багато кінцевих приголосних більше не мовчать, якщо за ними йде слово, що починається на голосну, ефект, який називається зв'язком. Це чергування неможливо відтворити за допомогою простої системи об'єднання слів, яка потребує додаткової складності, щоб бути контекстно-залежною.

Формантовий синтез не використовує зразки людської мови під час виконання. Натомість синтезований мовний результат створюється за допомогою адитивного синтезу та акустичної моделі (синтез фізичного моделювання). [39] Такі параметри, як основна частота, рівень голосу та рівень шуму змінюються з часом, щоб створити форму сигналу штучного мовлення. Цей метод іноді називають синтезом на основі правил; однак багато об'єднаних систем також мають компоненти, засновані на правилах. Багато систем, заснованих на технології синтезу формантів, генерують штучну, звучану роботом мову, яку ніколи не помилково сприймають як людську.

Однак максимальна природність не завжди є метою системи синтезу мовлення, і системи синтезу формантів мають переваги перед конкатенативними системами. Формована синтезована мова може бути надійно зрозумілою, навіть

на дуже високих швидкостях, уникаючи акустичних збоїв, які часто мучать конкатенативні системи. Високошвидкісна синтезована мова використовується для людей із вадами зору для швидкої навігації комп'ютерами за допомогою зчитувача з екрана. Формантові синтезатори зазвичай є меншими програмами, ніж об'єднані системи, оскільки вони не мають бази даних зразків мови. Тому їх можна використовувати у вбудованих системах, де пам'ять та потужність мікропроцесора особливо обмежені. Оскільки системи, засновані на форманті, мають повний контроль над усіма аспектами вихідної мови, можна вивести широкий спектр просодій та інтонацій, передаючи не просто запитання та висловлювання, а різноманітні емоції та тони голосу.

Приклади контролю в режимі реального часу, але дуже точного інтонаційного синтезу формантів, включають роботу, виконану наприкінці 1970-х років для іграшки Texas Instruments Speak & Spell, і на початку 1980-х років аркадні машини Sega [40] та в багатьох Atari, Inc. аркадні ігри [41] з використанням мікросхем LPC TMS5220. Створення належної інтонації для цих проектів було кропітким, і результати все ще мають відповідати інтерфейсам перетворення тексту в мовлення в реальному часі. [42]

Артикуляційний синтез відноситься до обчислювальних методів синтезу мовлення, заснованих на моделях голосового тракту людини та процесах артикуляції, що відбуваються там. Перший артикуляційний синтезатор, який регулярно використовується для лабораторних експериментів, був розроблений в лабораторіях Хаскінса в середині 1970-х років Філіпом Рубіном, Томом Баером та Полом Мермелштейном. Цей синтезатор, відомий як ASY, базувався на моделях голосових шляхів, розроблених у лабораторіях Белла в 1960-х та 1970-х роках Полом Мермельштейном, Сесілом Кокером та його колегами.

Донедавна моделі артикуляційного синтезу не вбудовувались у комерційні системи синтезу мовлення. Помітним винятком є система на базі NeXT, спочатку розроблена та продавана компанією Trillium Sound Research, відокремленою компанією Університету Калгарі, де проводилась значна частина оригінальних досліджень. Після загибелі різних втілень NeXT (розпочатої Стівом

Джобсом наприкінці 1980-х і об'єднаній з Apple Computer в 1997), програмне забезпечення Trillium було опубліковане під загальною публічною ліцензією GNU, робота продовжувалась як gnuspeech.

Система, вперше випущена на ринок в 1994 році, забезпечує повне перетворення тексту в мовлення на основі артикуляції, використовуючи хвилевід або аналог лінії передачі ротових та носових шляхів людини, контрольованих "особливою моделлю регіону" Карре.

Більш пізні синтезатори, розроблені Хорхе К. Лусеро та його колегами, включають моделі біомеханіки голосових складок, аеродинаміки глоталу та поширення акустичних хвиль у бронках, тракеї, носовій та ротовій порожнинах, і, таким чином, складають повні системи моделювання мовлення на основі фізики. [43] [44]

1.3 Розробка веб-сайтів

Веб-сайт - це сукупність веб-сторінок та пов’язаного вмісту, які ідентифікуються загальним доменним ім’ям та публікуються принаймні на одному веб-сервері. Помітними прикладами є wikipedia.org, google.com та amazon.com.

Усі загальнодоступні веб-сайти в сукупності складають Всесвітню павутину. Існують також приватні веб-сайти, доступ до яких доступний лише в приватній мережі, наприклад, внутрішній веб-сайт компанії для її співробітників.

Веб-сайти, як правило, присвячені певній темі або меті, такі як новини, освіта, комерція, розваги чи соціальні мережі. Гіперпосилання між веб-сторінками спрямовує навігацію по сайту, яке часто починається з домашньої сторінки.

Користувачі можуть отримати доступ до веб-сайтів на різних пристроях, включаючи настільні, ноутбуки, планшети та смартфони. Програма, що використовується на цих пристроях, називається веб-браузером.

Всесвітня павутина (WWW) була створена в 1990 році британським фізиком ЦЕРН Тімом Бернерсом-Лі. [1] 30 квітня 1993 р. ЦЕРН оголосив, що Всесвітня павутина буде безкоштовною для використання будь-ким [2].

До впровадження протоколу передачі гіпертексту (HTTP) для отримання окремих файлів із сервера використовувались інші протоколи, такі як протокол передачі файлів та протокол gopher. Ці протоколи пропонують просту структуру каталогів, в якій користувач переходить і де він обирає файли для завантаження. Документи найчастіше подавалися у вигляді текстових файлів без форматування або кодувались у форматах текстового процесора.

Веб-сайти можуть використовуватися по-різному: персональний веб-сайт, корпоративний веб-сайт компанії, урядовий веб-сайт, веб-сайт організації тощо. Веб-сайти можуть бути роботою приватної особи, бізнесу чи іншої організації і, як правило, присвячені конкретна тема або мета. Будь-який веб-сайт може містити гіперпосилання на будь-який інший веб-сайт, тому різниця між окремими сайтами, як сприймається користувачем, може бути розмитою.

Деякі веб-сайти вимагають реєстрації користувачів або передплати для доступу до вмісту. Приклади веб-сайтів, на які здійснюється підписка, включають багато бізнес-сайтів, веб-сайти новин, веб-сайти академічних журналів, ігрові веб-сайти, веб-сайти спільного використання файлів, дошки оголошень, веб-адреси електронної пошти, веб-сайти соціальних мереж, веб-сайти, що надають дані про фондовий ринок у режимі реального часу, а також веб-сайти, що надають різні інші послуги.

Хоча "веб-сайт" був оригінальним написанням (іноді з великої літери "Веб-сайт", оскільки "Веб" є власним іменником, коли йдеться про Всесвітню павутину), цей варіант став рідкісним, а "веб-сайт" став стандартним написанням . Усі основні керівництва стилем, такі як Чиказький посібник стилю [3] та AP Stylebook [4], відображають цю зміну.

Статичний веб-сайт - це веб-сторінки, які мають веб-сторінки, що зберігаються на сервері у форматі, який надсилається клієнтському веб-браузеру. В основному він кодується мовою розмітки гіпертексту (HTML); Каскадні таблиці

стилів (CSS) використовуються для контролю зовнішнього вигляду за базовим HTML. Зображення зазвичай використовуються для отримання бажаного вигляду та як частина основного змісту. Аудіо чи відео також можна вважати "статичним" вмістом, якщо він відтворюється автоматично або, як правило, є неінтерактивним. Цей тип веб-сайтів зазвичай відображає однакову інформацію для всіх відвідувачів. Подібно до роздачі друкованої брошури клієнтам або клієнтам, статичний веб-сайт, як правило, надає послідовну стандартну інформацію протягом тривалого періоду часу. Незважаючи на те, що власник веб-сайту може періодично робити оновлення, редагування тексту, фотографій та іншого вмісту здійснюється вручну, і може знадобитися базові навички дизайну веб-сайту та програмне забезпечення. Прості форми або маркетингові приклади веб-сайтів, такі як класичний веб-сайт, веб-сайт із п'ятьма сторінками або веб-сайт брошури, часто є статичними веб-сайтами, оскільки вони представляють користувачеві заздалегідь визначену, статичну інформацію. Це може включати інформацію про компанію та її продукти та послуги через текст, фотографії, анімацію, аудіо / відео та навігаційні меню.

Статичні веб-сайти все ще можуть використовувати серверні компоненти (SSI) як зручність редагування, наприклад, спільний доступ до загального рядка меню на багатьох сторінках.

Динамічний веб-сайт - це веб-сайт, який часто і автоматично змінюється або налаштовується. Динамічні сторінки на сервері генеруються "на льоту" за допомогою комп'ютерного коду, який виробляє HTML (CSS відповідає за зовнішній вигляд і, отже, статичні файли). Існує широкий спектр програмних систем, таких як CGI, Java-сервлети та Java Server Pages (JSP), Active Server Pages і ColdFusion (CFML), які доступні для створення динамічних веб-систем та динамічних веб-сайтів. Різні фреймворки веб-програм та системи веб-шаблонів доступні для загальновживаних мов програмування, таких як Perl, PHP, Python та Ruby, щоб полегшити та спростити створення складних динамічних веб-сайтів.

Сайт може відображати поточний стан діалогу між користувачами, відстежувати мінливу ситуацію або надавати інформацію певним чином

персоналізовану відповідно до вимог окремого користувача. Наприклад, коли запитується головна сторінка сайту новин, код, що працює на веб-сервері, може поєднувати збережені фрагменти HTML із новинами, отриманими з бази даних або іншого веб-сайту за допомогою RSS, щоб створити сторінку, що включає найсвіжішу інформацію.

Динамічні сайти можуть бути інтерактивними, використовуючи HTML-форми, зберігаючи та зчитуючи файли cookie браузера, або створюючи ряд сторінок, що відображають попередню історію кліків. Інший приклад динамічного вмісту - це коли роздрібний веб-сайт з базою даних медіа-продуктів дозволяє користувачеві вводити запит на пошук, наприклад за ключовим словом "Бітлз".

Динамічний HTML використовує код JavaScript, щоб вказувати веб-браузеру, як інтерактивно змінювати вміст сторінки. Одним із способів моделювання певного типу динамічного веб-сайту, уникаючи втрати продуктивності ініціювання динамічного механізму для кожного користувача або підключення, є періодична автоматична регенерація великої серії статичних сторінок.

Ранні веб-сайти мали лише текст, а незабаром і зображення. Потім плагіни веб-браузера використовувались для додавання аудіо, відео та інтерактивності (наприклад, для багатофункціональної програми Інтернету, яка відображає складність настільної програми, як текстовий процесор). Прикладами таких плагінів є Microsoft Silverlight, Adobe Flash, Adobe Shockwave та аплети, написані на Java. HTML 5 містить положення щодо аудіо та відео без плагінів. JavaScript також вбудований у більшість сучасних веб-браузерів і дозволяє творцям веб-сайтів надсилати код веб-браузеру, який вказує йому, як інтерактивно змінювати вміст сторінки та спілкуватися з веб-сервером, якщо це необхідно. Внутрішнє представлення вмісту браузера відомо як об’єктна модель документа (DOM), а техніка - динамічний HTML.

WebGL (Web Graphics Library) - це сучасний API JavaScript для візуалізації інтерактивної 3D-графіки без використання плагінів. Це дозволяє інтерактивний

контент, такий як 3D-анімація, візуалізація та відео-пояснення, представленим користувачам найбільш інтуїтивно зрозумілим способом. [5]

Тенденція 2010 року на веб-сайтах під назвою "адаптивний дизайн" забезпечила найкращий досвід перегляду, оскільки надає користувачам макет на основі пристрою. Ці веб-сайти змінюють свій макет відповідно до пристрою або мобільної платформи, що забезпечує багатий досвід користувачів. [6]

Веб-сайти можна розділити на дві великі категорії - статичні та інтерактивні. Інтерактивні сайти є частиною спільноти веб-сайтів Web 2.0 і дозволяють взаємодіяти між власником сайту та відвідувачами або користувачами сайту. Статичні сайти обслуговують або збирають інформацію, але не дозволяють взаємодіяти з аудиторією або користувачами безпосередньо. Деякі веб-сайти є інформаційними або виготовляються ентузіастами або для особистого користування чи розваги. Багато веб-сайтів мають на меті заробляти гроші, використовуючи одну або кілька бізнес-моделей, зокрема:

- Розміщення цікавого контенту та продаж контекстної реклами або шляхом прямих продажів, або через рекламну мережу.
- Електронна комерція: товари чи послуги купуються безпосередньо через веб-сайт
- Реклама товарів або послуг, доступних у цегельному та будівельному бізнесі
- Freemium: базовий вміст доступний безкоштовно, але преміум-вміст вимагає оплати (наприклад, веб-сайт WordPress, це платформа з відкритим кодом для створення блогу чи веб-сайту).

Деякі веб-сайти можуть бути включені в одну або кілька з цих категорій. Наприклад, веб-сайт бізнесу може рекламувати продукцію бізнесу, але також може розміщувати інформативні документи, такі як технічні документи. Існують також численні підкатегорії до перерахованих вище. Наприклад, порносайт - це певний тип веб-сайту електронної комерції або бізнес-сайту (тобто він намагається продати членство для доступу до свого сайту) або має можливості соціальних мереж. Фан-сайт може бути посвятою власника певній знаменитості.

Веб-сайти обмежені архітектурними обмеженнями (наприклад, обчислювальна потужність, присвячена веб-сайту).

Дуже великі веб-сайти, такі як Facebook, Yahoo!, Microsoft та Google, використовують багато серверів та обладнання для балансування навантажень, наприклад, комутатори служб вмісту Cisco, для розподілу навантажень відвідувачів на декілька комп'ютерів у різних місцях. На початок 2011 року Facebook використовував 9 центрів обробки даних із приблизно 63000 серверами.

У лютому 2009 року компанія Netcraft, компанія з моніторингу Інтернету, яка відслідковувала зростання Інтернету з 1995 р., Повідомила, що в 2009 р. Існувало 215 675 903 веб-сайтів з доменними іменами та вмістом на них, порівняно з лише 19 732 веб-сайтами в серпні 1995 р. [8] Після досягнення 1 мільярда веб-сайтів у вересні 2014 року, етап, підтверджений NetCraft в огляді веб-серверів за жовтень 2014 року, і те, що Інтернет-статистика стала першим, хто повідомив про це, про що свідчить цей твіт від самого винахідника Всесвітньої мережі, Тіма Бернерса Лі - кількість веб-сайтів у світі згодом зменшилась, повернувшись до рівня нижче 1 мільярда. Це пов'язано з щомісячними коливаннями кількості неактивних веб-сайтів. Кількість веб-сайтів до березня 2016 р. Продовжувала зростати до понад 1 млрд. І з тих пір продовжує зростати [9].

Веб-розробка - це робота, пов'язана з розробкою веб-сайту для Інтернету (Всесвітня павутина) або інтрамережі (приватна мережа). [1] Веб-розробка може варіюватися від розробки простої однієї статичної сторінки простого тексту до складних Інтернет-програм (Інтернет-додатків), електронного бізнесу та послуг соціальних мереж. Більш повний перелік завдань, до яких зазвичай відноситься веб-розробка, може включати веб-інженерію, веб-дизайн, розробку веб-вмісту, зв'язок з клієнтом, сценарії на стороні клієнта / сервера, налаштування безпеки веб-сервера та мережі та розвиток електронної комерції.

Серед веб-спеціалістів "веб-розробка" зазвичай відноситься до основних недизайнерських аспектів побудови веб-сайтів: написання розмітки та кодування. [2] Веб-розробка може використовувати системи управління вмістом (CMS), щоб

зробити зміст вмістом простішим та доступнішим з базовими технічними навичками.

Для великих організацій та бізнесу групи веб-розробників можуть складатися з сотень людей (веб-розробників) і дотримуватися стандартних методів, таких як Agile, під час розробки веб-сайтів. Менші організації можуть вимагати лише одного постійного або підрядного розробника, або вторинне призначення на відповідні посади, такі як графічний дизайнер або технік інформаційних систем. Веб-розробка може бути спільним зусиллям між департаментами, а не областю призначеного департаменту. Існує три різновиди спеціалізації веб-розробників: розробник з інтерфейсу, розробник з інтерфейсу та розробник із повним стеком. Інтернетні розробники несуть відповідальність за поведінку та візуальні ефекти, які працюють у браузері користувача, тоді як інтерфейсні розробники мають справу з серверами.

З моменту комерціалізації Інтернету веб-розробка стала зростаючою галуззю. Зростання цієї галузі відбувається за рахунок підприємств, які бажають використовувати свій веб-сайт для реклами та продажу товарів та послуг споживачам. [3]

Існує багато інструментів з відкритим кодом для веб-розробки, таких як BerkeleyDB, GlassFish, стек LAMP (Linux, Apache, MySQL, PHP) та Perl / Plack. Це дозволило звести витрати на навчання веб-розробці до мінімуму. Ще одним фактором, що сприяє зростанню галузі, стало зростання простого у використанні програмного забезпечення для веб-розробки WYSIWYG, такого як Adobe Dreamweaver, BlueGriffon та Microsoft Visual Studio. Для використання такого програмного забезпечення все ще необхідні знання мови розмітки HyperText (HTML) або мов програмування, але основи можна швидко вивчити та впровадити.

Постійно зростаючий набір інструментів та технологій допоміг розробникам створювати більш динамічні та інтерактивні веб-сайти. Крім того, веб-розробники тепер допомагають надавати програми як веб-служби, які традиційно були доступні лише як програми на настільному комп’ютері. Це дало

багато можливостей для децентралізації розповсюдження інформації та засобів масової інформації. Приклади можна побачити із зростанням хмарних сервісів, таких як Adobe Creative Cloud, Dropbox та Google Drive. Ці веб-служби дозволяють користувачам взаємодіяти з програмами з багатьох місць, замість того, щоб бути прив’язаними до певної робочої станції для свого середовища програм.

Прикладами кардинальних перетворень у спілкуванні та комерції на чолі з веб-розробкою є електронна комерція. Інтернет-сайти аукціонів, такі як eBay, змінили спосіб споживачів знаходити та купувати товари та послуги. Інтернет-магазини, такі як Amazon.com та Buy.com (серед багатьох інших), змінили досвід покупок та торгівлі для багатьох споживачів.

Розробка веб-сайтів також вплинула на особисті мережі та маркетинг. Веб-сайти вже не є просто інструментами для роботи чи комерції, а ширше служать для спілкування та соціальних мереж. Такі веб-сайти, як Facebook та Twitter, надають користувачам платформу для спілкування, а організації - більш особистий та інтерактивний спосіб залучення громадськості.

Веб-розробка враховує багато міркувань безпеки, таких як перевірка помилок при введенні даних через форми, фільтрація вихідних даних та шифрування. Шкідливі практики, такі як введення SQL, можуть виконуватися користувачами з ненавмисними намірами, але лише з примітивними знаннями про веб-розробку в цілому. Сценарії можна використовувати для використання веб-сайтів, надаючи несанкціонований доступ зловмисним користувачам, які намагаються збирати таку інформацію, як адреси електронної пошти, паролі та захищений вміст, наприклад номери кредитних карток.

Дещо з цього залежить від серверного середовища, на якому запущена мова сценаріїв, наприклад ASP, JSP, PHP, Python, Perl або Ruby, і тому не обов'язково підтримувати самого розробника. Однак рекомендується жорстке тестування веб-додатків перед публічним випуском, щоб запобігти подібним подвигам. Якщо на веб-сайті є якась контактна форма, вона повинна включати в неї поле captcha, яке не дозволяє комп’ютерним програмам автоматично заповнювати форми, а також розсилати пошту.

Захист веб-сервера від вторгнень часто називають зміцненням порту сервера. Багато технологій застосовуються для захисту інформації в Інтернеті, коли вона передається з одного місця в інше. Наприклад, сертифікати TLS (або "сертифікати SSL") видаються органами сертифікації для запобігання шахрайству в Інтернеті. Багато розробників часто використовують різні форми шифрування при передачі та зберіганні конфіденційної інформації. Базове розуміння питань безпеки інформаційних технологій часто є частиною знань веб-розробника.

Оскільки нові веб-програми виявляють нові діри в безпеці навіть після тестування та запуску, оновлення виправлень безпеки часто трапляються для широко використовуваних програм. Часто робота веб-розробників - постійно оновлювати програми, коли випускаються виправлення безпеки та виявляються нові проблеми безпеки.

2. ВИБІР І ОБГРУНТУВАННЯ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1 Мова програмування Python

Python - інтерпретована мова програмування високого рівня та загального призначення. Філософія дизайну Python наголошує на читабельності коду завдяки помітному використанню значних відступів. Його мовні конструкції та об’єктно-орієнтований підхід мають на меті допомогти програмістам написати чіткий логічний код для малих та великих проектів.

Python динамічно набирається та збирається сміття. Він підтримує декілька парадигм програмування, включаючи структуроване (зокрема, процедурне), об'єктно-орієнтоване та функціональне програмування. Python часто описують як мову, що включає батареї, завдяки своїй повній стандартній бібліотеці.

Гідо ван Россум почав працювати над Python наприкінці 1980-х років, як наступник мови програмування ABC, і вперше випустив її в 1991 році під назвою Python 0.9.0. [32] Python 2.0 був випущений в 2000 році і представив нові функції, такі як розуміння списків та система збору сміття з використанням підрахунку посилань, і був припинений з версією 2.7.18 у 2020 році [33]. Python 3.0 був випущений в 2008 році і був серйозним переглядом мови, яка не є повністю сумісною із зворотною суттю, і більша частина коду Python 2 не працює незміненою на Python 3.

Python стабільно входить до числа найпопулярніших мов програмування.

Python був задуманий наприкінці 1980-х Гвідо ван Россумом з Centrum Wiskunde & Informatica (CWI) в Нідерландах як спадкоємець мови програмування ABC, натхненної SETL, здатною обробляти винятки та взаємодіяти з Операційна система Amoeba. Його реалізація розпочалась у грудні 1989 р. Ван Россум взяв на себе виключну відповідальність за проект, як провідний розробник, до 12 липня 2018 року, коли він оголосив про свої "постійні канікули", виконуючи обов'язки як "Доброзичливий диктатор за життя" Пітона, титул, який йому надала спільнота Пітона, щоб відобразити його тривалий час. строкове зобов'язання як головний

керівник проекту. Зараз він ділиться своїм керівництвом як член наглядової ради з п'яти осіб. У січні 2019 року активні розробники ядра Python обрали Бретта Кеннона, Ніка Коглана, Барі Варшаву, Керол Віллінг та Ван Россума членами керівної ради з п’яти членів. З тих пір Гвідо ван Россум відкликав свою кандидатуру на посаду Керівної ради 2020 року.

Python 2.0 був випущений 16 жовтня 2000 року з багатьма основними новими можливостями, включаючи збирач сміття, що виявляє цикл, та підтримку Unicode.

Python 3.0 був випущений 3 грудня 2008 року. Це був серйозний перегляд мови, який не є повністю сумісним із зворотною стороною. Багато його основних функцій були передані до версій Python 2.6.x та 2.7.x. Випуски Python 3 включають утиліту 2to3, яка автоматизує (принаймні частково) переклад коду Python 2 на Python 3.

Дата закінчення терміну служби Python 2.7 спочатку була встановлена на 2015 рік, а потім перенесена на 2020 рік через занепокоєння тим, що велика кількість існуючого коду не може бути легко перенесена на Python 3. Більше виправлень безпеки та інших удосконалень для нього випускати не буде. У кінці життя Python 2 підтримується лише Python 3.6.x і пізніші версії.

Python 3.9.2 та 3.8.8 були прискорені, оскільки всі версії Python мали проблеми із безпекою, що призвело до можливого віддаленого виконання коду та отруєння веб-кешу.

Python - мова програмування з багатьма парадигмами. Об'єктно-орієнтоване програмування та структуроване програмування повністю підтримуються, і багато його функцій підтримують функціональне програмування та аспектно-орієнтоване програмування (в тому числі метапрограмуванням та метаоб'єктами (магічні методи)). Багато інших парадигм підтримуються за допомогою розширень, включаючи дизайн за контрактом та логічне програмування.

Python використовує динамічне введення тексту та комбінацію підрахунку посилань та збирач сміття, що визначає цикл, для управління пам’яттю. Він

також має динамічну роздільну здатність імен (пізнє прив'язування), яка зв'язує імена методів та змінних під час виконання програми.

Дизайн Python пропонує певну підтримку функціонального програмування за традицією Lisp. Він має функції фільтрування, картографування та зменшення; перелічити розуміння, словники, набори та вирази генераторів. Стандартна бібліотека має два модулі (itertools та functools), що реалізують функціональні інструменти, запозичені у Haskell та Standard ML.

Основна філософія мови узагальнена в документі Zen of Python (PEP 20), який включає такі афоризми, як:

- Красиве - краще, ніж потворне.
- Явне краще, ніж неявне.
- Просте - краще, ніж складне.
- Складний - це краще, ніж складний.
- Підрахунок читабельності.

Замість того, щоб усі його функціональні можливості були вбудовані в його ядро, Python був розроблений таким чином, щоб бути дуже розширюваним (з модулями). Ця компактна модульність зробила його особливо популярним як засіб додавання програмованих інтерфейсів до існуючих додатків. Бачення Ван Россума щодо малої основної мови з великою стандартною бібліотекою та легко розширюваним перекладачем випливало з його розчарувань у ABC, який підтримував протилежний підхід. Python прагне до більш простого, менш захараченого синтаксису та граматики, одночасно надаючи розробникам можливість вибору методології кодування. На відміну від девізу Perl "існує більше ніж один спосіб зробити це", Python охоплює "повинен бути один - і бажано лише один - очевидний спосіб зробити це" філософія дизайну. [69] Алекс Мартеллі, співробітник Фонду програмного забезпечення Python і автор книги про Python, пише, що "Описувати щось як" розумне "не вважається компліментом у культурі Python" .

Розробники Python прагнуть уникнути передчасної оптимізації та відкидають виправлення для некритичних частин еталонної реалізації CPython,

які пропонують незначне збільшення швидкості ціною ясності. Коли швидкість важлива, програміст Python може переміщати критично важливі для часу функції до модулів розширення, написаних мовами, такими як C, або використовувати PyPy, своєчасний компілятор. Також доступний Cython, який перекладає скрипт Python на C і здійснює прямі виклики API рівня C в інтерпретатор Python.

Важливою метою розробників Python є підтримка його задоволення від використання. Це відображається в назві мови - данина поваги британській гумористичній групі Монті Пайтон - і в іноді грайливих підходах до навчальних посібників та довідкових матеріалів, таких як приклади, що стосуються спаму та яєць (із відомого етюду Монті Пайтона) стандартних foo and bar.

Поширеним неологізмом у спільноті Python є пітонічний, який може мати широкий діапазон значень, пов’язаних зі стилем програми. Сказати, що код пітонічний, означає сказати, що він добре використовує ідіоми Python, що він природний або демонструє вільну мову, що відповідає мінімалістичній філософії Python та наголосу на читабельності. На відміну від цього, код, який важко зрозуміти або читається як груба транскрипція з іншої мови програмування, називається непітонічним.

Користувачів та шанувальників Python, особливо тих, хто вважається обізнаним чи досвідченим, часто називають Pythonistas. Python має бути легкочитабельною мовою. Його форматування візуально не захаращене, і в ньому часто використовуються англійські ключові слова, де інші мови використовують розділові знаки. На відміну від багатьох інших мов, він не використовує фігурні дужки для розмежування блоків, а крапка з комою після операторів дозволена, але рідко, якщо взагалі використовується. Він має менше синтаксичних винятків та особливих випадків, ніж C або Pascal.

Python використовує пробіли, а не фігурні дужки або ключові слова, для відмежування блоків. Збільшення відступу відбувається після певних тверджень; зменшення відступу означає кінець поточного блоку. Таким чином, візуальна структура програми точно відображає семантичну структуру програми. Цю функцію іноді називають правилом "поза стороною", яке мають деякі інші мови,

але в більшості мов відступ не має семантичного значення. Рекомендований розмір відступу - чотири пробіли.

Оператори Python включають (серед іншого):

- Оператор присвоєння, використовуючи один знак рівності =.
- Оператор if, який умовно виконує блок коду, разом з else та elif (скорочення else-if).
- Оператор for, який переглядає ітерабельний об’єкт, захоплюючи кожен елемент до локальної змінної для використання вкладеним блоком.
- Оператор while, який виконує блок коду, доки його умова відповідає істині.
- Оператор try, що дозволяє вилученням, що містяться у вкладеному блоці коду, ловити та обробляти за винятком речень; він також гарантує, що код очищення в блоці нарешті завжди буде виконуватися незалежно від того, як блок виходить.
- Оператор raise, що використовується для отримання вказаного винятку або повторного підняття спійманого винятку.
- Оператор class, який виконує блок коду та приєднує його локальний простір імен до класу для використання в об'єктно-орієнтованому програмуванні.
- Оператор def, що визначає функцію або метод.
- Оператор with з Python 2.5, випущений у вересні 2006 р. [82], який охоплює блок коду в контекстному менеджері (наприклад, отримання блокування перед запуском блоку коду та звільнення блокування після цього, або відкриття файлу, а потім закриваючи його), дозволяючи поведінку, схожу на отримання ресурсів - це ініціалізація (RAII), і замінює загальну ідіому try / нарешті.
- Оператор break, виходить із циклу.
- Оператор continue, пропускає цю ітерацію і продовжує наступний пункт.

- Оператор del видаляє змінну, що означає, що посилання з імені на значення видаляється, і спроба використання цієї змінної спричинить помилку. Видалену змінну можна перепризначити.
- Оператор pass, яка виконує функцію NOP. Це синтаксично потрібно для створення порожнього блоку коду.
- Оператор assert, який використовується під час налагодження для перевірки умов, які мають застосовуватися.
- Оператор yield, який повертає значення з функції генератора. З Python 2.5, yield також є оператором. Ця форма використовується для реалізації спільних програм.
- Оператор return, який використовується для повернення значення з функції.
- Оператор import, який використовується для імпорту модулів, функції чи змінні яких можна використовувати в поточній програмі. Є три способи використання імпорту: імпорт <ім'я модуля> [як <псевдонім>] або з <ім'я модуля> імпорт * або з <ім'я модуля> імпорт <визначення 1> [як <псевдонім 1>], <визначення 2> [як <псевдонім 2>], ...

Оператор присвоєння (=) діє шляхом прив'язки імені як посилання на окремий динамічно розподілений об'єкт. Потім змінні можуть бути відскочені в будь-який час до будь-якого об’єкта. У Python ім'я змінної є загальним власником посилання і не має фіксованого типу даних, пов’язаного з ним. Однак у даний момент часу змінна буде посилатися на якийсь об'єкт, який матиме тип. Це називається динамічним набором тексту і протиставляється статично набраним мовам програмування, де кожна змінна може містити лише значення певного типу.

Python не підтримує оптимізацію хвостових викликів або першокласні продовження, і, за словами Гвідо ван Россума, він ніколи не буде. Однак краща підтримка подібних до корутинів функціональних можливостей надається в 2.5, розширюючи генератори Python. До 2.5 генератори були ледачими ітераторами; інформація передавалась в односпрямованому напрямку з генератора. З Python 2.5

можна передавати інформацію назад у функцію генератора, а з Python 3.3 інформацію можна передавати через кілька рівнів стека.

Деякі вирази Python подібні до таких, що зустрічаються в таких мовах, як C та Java, а деякі - ні:

- Додавання, віднімання та множення однакові, але поведінка ділення відрізняється. У Python існує два типи поділів. Вони є розподілом підлоги (або цілочисельним поділом) // та плаваючою комою / поділом. Python також використовує оператор ** для піднесення до степені.
- З Python 3.5 був представлений новий оператор @ infix. Він призначений для використання бібліотеками, такими як NumPy, для множення матриць.
- З Python 3.8 був введений синтаксис: =, який називається «моржовим оператором». Він присвоює значення змінним як частину більшого виразу. [91]
- У Python == порівнює за значенням проти Java, яка порівнює числові значення за значенням, а об'єкти за посиланням. (Порівняння значень в Java на об'єктах можна виконувати методом equals ().) Оператор Python is is може використовуватися для порівняння ідентифікацій об'єктів (порівняння за посиланням). У Python порівняння можуть бути ланцюговими, наприклад a <= b <= c.
- Python використовує слова та, або, не для своїх булевих операторів, а не для символічного &&, ||,! використовується в Java та C.
- Python має тип виразу, що називається розумінням списку, а також більш загальний вираз, який називається генераторським виразом.
- Анонімні функції реалізовані за допомогою лямбда-виразів; однак вони обмежені тим, що тіло може бути лише одним виразом.
- Умовні вирази в Python пишуться як x, якщо c else y (відрізняється за порядком операндів від оператора c? X: y, загального для багатьох інших мов).

- Python розрізняє списки та кортежі. Списки записуються як [1, 2, 3], змінюються та не можуть використовуватися як ключі словників (ключі словника повинні бути незмінними в Python). Кортежі записуються як (1, 2, 3), є незмінними і, отже, можуть використовуватися як ключі словників, за умови, що всі елементи кортежу є незмінними. Оператор + може бути використаний для об'єднання двох кортежів, що безпосередньо не змінює їх вміст, а створює новий кортеж, що містить елементи обох передбачених кортежів. Таким чином, враховуючи змінну t, спочатку рівну (1, 2, 3), при виконанні t = t + (4, 5) спочатку обчислюється t + (4, 5), що дає (1, 2, 3, 4, 5) , який потім призначається назад до t, тим самим ефективно "модифікуючи вміст" t, одночасно відповідаючи незмінній природі об'єктів кортежу. Дужки не є обов’язковими для кортежів у однозначному контексті.
- Python має розпаковування послідовностей, при якому кілька виразів, кожен з яких обчислює будь-що, до чого можна присвоїти (змінну, властивість для запису тощо), пов’язані ідентично тому, що формує кортежні літерали, і, як ціле, розміщуються на ліва сторона знака рівності у твердженні про присвоєння. Оператор очікує ітерабельного об'єкта праворуч від знака рівності, який видає таку ж кількість значень, що і надані вираз для запису, коли його перебирають і буде перебирати його, присвоюючи кожне із створених значень відповідному виразу зліва .
- Python має оператор "формат рядка"%. Це функціонує аналогічно рядкам формату printf на C, наприклад "спам =% s яєць =% d"% ("бла", 2) оцінюється як "спам = бла-яйця = 2". У Python 3 та 2.6+ це було доповнено методом format () класу str, наприклад "спам = {0} яйця = {1}". формат ("бла", 2). Python 3.6 додав "f-рядки": blah = "blah"; яйця = 2; f'spam = {blah} яйця = {яйця} '.
- Рядки в Python можна об'єднати, "додавши" їх (той самий оператор, що і для додавання цілих чи значень з плаваючою точкою). Наприклад

"спам" + "яйця" повертає "спамегги". Навіть якщо ваші рядки містять числа, вони все одно додаються як рядки, а не цілі числа. Наприклад "2" + "2" повертає "22".

- Python має різні типи рядкових літералів:
- Рядки, розділені одинарними або подвійними лапками. На відміну від оболонок Unix, мови Perl та Perl, що впливають на Perl, одинарні лапки та подвійні лапки функціонують однаково. Обидва типи рядків використовують зворотну скісну риску (\) як символ виходу. Інтерполяція рядків стала доступною в Python 3.6 як "відформатовані рядкові літерали".
- Рядки з подвійними лапками, які починаються та закінчуються серією з трьох одинарних або подвійних лапок. Вони можуть охоплювати кілька рядків і функціонувати, як тут документи в оболонках, Perl і Ruby.
- Сирі різновиди рядків, що позначаються префіксом рядкового літералу перед r. Послідовності втечі не інтерпретуються; отже, необроблені рядки корисні там, де буквенні зворотні слеші є загальними, такі як регулярні вирази та шляхи у стилі Windows. Порівняйте "@ -citation" у C #.
- Python має індекси масивів та вирази нарізування масивів у списках, що позначаються як [ключ], [старт: зупинка] або [старт: зупинка: крок]. Індекси базуються на нулі, а негативні індекси відносно кінця. Зрізи беруть елементи від початкового індексу до, але не включаючи, індексу зупинки. Третій параметр зрізу, який називається кроком або кроком, дозволяє пропускати та обертати елементи. Індекси зрізів можуть бути опущені, наприклад, [:] повертає копію всього списку. Кожен елемент зрізу є неглибокою копією.

У Python чітко дотримується різниці між виразами та висловлюваннями, на відміну від таких мов, як Common Lisp, Scheme або Ruby. Це призводить до дублювання деяких функціональних можливостей.

Методи на об'єктах - це функції, приєднані до класу об'єкта; синтаксис instance.method (аргумент) - це для звичайних методів та функцій синтаксичний цукор для Class.method (екземпляр, аргумент). Методи Python мають явний параметр self для доступу до даних екземпляра, на відміну від неявного self (або цього) в деяких інших об’єктно-орієнтованих мовах програмування (наприклад, C ++, Java, Objective-C або Ruby).

Python використовує набір качок і має набрані об'єкти, але нетипізовані імена змінних. Обмеження типу не перевіряються під час компіляції; швидше, операції з об'єктом можуть зазнати невдачі, що означає, що даний об'єкт не є відповідним типом. Не дивлячись на те, що Python набирається динамічно, забороняє операції, які не є чітко визначеними (наприклад, додавання числа до рядка), а не намагається їх тихо зрозуміти.

Python дозволяє програмістам визначати власні типи за допомогою класів, які найчастіше використовуються для об'єктно-орієнтованого програмування. Нові екземпляри класів будуються за допомогою виклику класу (наприклад, SpamClass () або EggsClass ()), а класи є екземплярами типу метакласу (сам екземпляр самого себе), що дозволяє метапрограмувати та відображати.

До версії 3.0 Python мав два типи класів: старий і новий. Синтаксис обох стилів однаковий, різниця полягає в тому, чи об’єкт класу успадковується, прямо чи опосередковано (усі класи нового стилю успадковуються від об’єкта і є екземплярами типу). У версіях Python 2, починаючи з Python 2.2, можна використовувати обидва типи класів. Класи старого стилю були ліквідовані в Python 3.0.

Довгостроковий план полягає в підтримці поступового набору тексту, а з Python 3.5 синтаксис мови дозволяє вказувати статичні типи, але вони не перевіряються в реалізації за замовчуванням, CPython. Експериментальна додаткова перевірка статичного типу з ім’ям mypy підтримує перевірку типу під час компіляції.

CPython - це посилальна реалізація Python. Він написаний на мові C, відповідаючи стандарту C89 з декількома вибраними функціями C99 (з випуском

пізніших версій C він вважається застарілим; CPython включає власні розширення C, але розширення сторонніх розробників не обмежуються старими C версії, наприклад, можуть бути реалізовані з C11 або C ++). Він компілює програми Python у проміжний байт-код, який потім виконується його віртуальною машиною. CPython поширюється з великою стандартною бібліотекою, написаною на суміші C та рідного Python. Він доступний для багатьох платформ, включаючи Windows (починаючи з Python 3.9, інсталятор Python навмисно не вдається встановити в Windows 7 і 8; Windows XP підтримувався до Python 3.5) та більшості сучасних Unix-подібних систем, включаючи macOS (та Apple M1 Macs, починаючи з Python 3.9.1, з експериментальним інсталятором) та неофіційну підтримку, наприклад VMS. Переносимість платформи була одним із її перших пріоритетів, протягом періодів Python 1 та 2 підтримувались навіть OS / 2 та Solaris; підтримка відтоді відмовилася для багатьох платформ.

Розробка Python здійснюється в основному за допомогою процесу Програми вдосконалення Python (PEP), основного механізму пропонування основних нових можливостей, збору вкладу спільноти з питань та документування рішень щодо проектування Python. Стиль кодування Python викладений у PEP 8. Видатні PEP переглядаються та коментуються спільнотою Python та керівною радою.

Покращення мови відповідає розробці посилальної реалізації CPython. Список розсилки python-dev є основним форумом для розвитку мови. Конкретні проблеми обговорюються у програмі відстеження помилок Roundup, розміщеній на bugs.python.org. Спочатку розробка здійснювалась у власному сховищі вихідного коду під управлінням Mercurial, поки Python не перейшов на GitHub у січні 2017 року.

Публічні випуски CPython бувають трьох типів, що відрізняються тим, яка частина номера версії збільшується:

Несумісні із зворотною мовою версії, де, як очікується, зламається код і його потрібно перенести вручну. Перша частина номера версії збільшується. Ці випуски трапляються рідко - версія 3.0 вийшла через 8 років після 2.0.

Основні або "функціональні" випуски відбувалися приблизно кожні 18 місяців, але з прийняттям щорічної каденції випуску, починаючи з Python 3.9, очікується, що це відбуватиметься раз на рік. Вони значною мірою сумісні, але вводять нові функції. Друга частина номера версії збільшується. Кожна основна версія підтримується виправленнями протягом декількох років після її випуску.

Випуски виправлень, які не вводять нових функцій, трапляються приблизно кожні 3 місяці і робляться, коли з останнього випуску виправлена достатня кількість помилок. Уразливості безпеки також виправлені у цих випусках. Третя і остання частина номера версії збільшується.

Багато альфа-, бета-версій та кандидатів на випуск також випускаються у вигляді попереднього перегляду та тестування перед остаточними випусками. Хоча існує приблизний графік кожного випуску, вони часто затримуються, якщо код не готовий. Команда розробників Python контролює стан коду, запускаючи великий пакет модульних тестів під час розробки.

Основною академічною конференцією з Python є PyCon. Існують також спеціальні програми наставництва Python, такі як Pyladies.

Pythons 3.10 припиняє wstr (буде видалено в Python 3.12; це означає, що розширення Python потрібно буде змінити до того часу), а також планує додати відповідність шаблону до мови.

Використання

З 2003р. Python стабільно входить до десятки найпопулярніших мов програмування в Індексі програмного забезпечення TIOBE, де, станом на лютий 2021 р., Це третя за популярністю мова (після Java та C). Він був обраний мовою програмування року (за "найвищий ріст рейтингу за рік") у 2007, 2010, 2018 та 2020 роках (єдина мова, яка зробила це чотири рази).

Емпіричне дослідження показало, що мови сценаріїв, такі як Python, є більш продуктивними, ніж звичайні мови, такі як C та Java, для проблем програмування, що включають маніпулювання рядками та пошук у словнику, і встановило, що споживання пам'яті часто "краще, ніж Java, а не набагато гірше, ніж C або C ++ ".

До великих організацій, які використовують Python, належать Wikipedia, Google, Yahoo!, CERN, NASA, Facebook, Amazon, Instagram, Spotify та деякі менші організації, такі як ILM та ITA. Сайт соціальних новин Reddit був написаний переважно на Python.

Python може служити мовою сценаріїв для веб-додатків, наприклад, через mod_wsgi для веб-сервера Apache. Завдяки інтерфейсу шлюзу веб-сервера для полегшення роботи цих програм розроблено стандартний API. Веб-фреймворки, такі як Django, Pylons, Pyramid, TurboGears, web2py, Tornado, Flask, Bottle і Zope, підтримують розробників у розробці та обслуговуванні складних додатків. Pyjs та IronPython можуть бути використані для розробки клієнтської частини програм на базі Ajax. SQLAlchemy може використовуватися як перетворювач даних у реляційну базу даних. Twisted - це фреймворк для програмування зв’язку між комп’ютерами, який використовується (наприклад) Dropbox.

Такі бібліотеки, як NumPy, SciPy та Matplotlib, дозволяють ефективно використовувати Python у наукових обчисленнях, а спеціалізовані бібліотеки, такі як Biopython та Astropy, забезпечують функціональність, що залежить від домену. SageMath - математичне програмне забезпечення з інтерфейсом ноутбука, програмоване на Python: його бібліотека охоплює багато аспектів математики, включаючи алгебру, комбінаторику, числову математику, теорію чисел і числення. OpenCV має палітурні прив'язки з багатим набором функцій для комп'ютерного зору та обробки зображень.

Python зазвичай використовується в проектах штучного інтелекту та машинному навчанні за допомогою таких бібліотек, як TensorFlow, Keras, Pytorch та Scikit-learn. Як мова сценаріїв з модульною архітектурою, простим синтаксисом та інструментами обробки багатого тексту, Python часто використовується для обробки природної мови. Python успішно вбудований у багато програмних продуктів як мову сценаріїв, включаючи програмне забезпечення методом скінченних елементів, таке як Abaqus, 3D-параметричний моделер, такий як FreeCAD, пакети 3D-анімації, такі як 3ds Max, Blender, Cinema 4D, Lightwave, Houdini, Maya, modo , MotionBuilder, Softimage, композитор

візуальних ефектів Nuke, програми для 2D-зображень, такі як GIMP, Inkscape, Scribus і Paint Shop Pro, та музичні нотатні програми, такі як автори та капели. Налагоджувач GNU використовує Python як гарний принтер для показу складних структур, таких як контейнери C ++.

Багато операційних систем включають Python як стандартний компонент. Він постачається з більшістю дистрибутивів Linux, AmigaOS 4 (з використанням Python 2.7), FreeBSD (як пакет), NetBSD, OpenBSD (як пакет) та macOS і може використовуватися з командного рядка (терміналу). Багато дистрибутивів Linux використовують інсталятори, написані на Python: Ubuntu використовує інсталятор Ubiquity, тоді як Red Hat Linux і Fedora використовують інсталятор Anaconda. Gentoo Linux використовує Python у своїй системі управління пакунками Portage. Python широко використовується в галузі інформаційної безпеки, в тому числі в розробці експлойтів.

Більшість програм Sugar для одного ноутбука на дитину XO, які зараз розробляються в Sugar Labs, написані на Python. Проект одноплатної комп'ютерної програми Raspberry Pi прийняв Python як основну мову програмування користувачів.

LibreOffice включає Python і має намір замінити Java на Python. Його постачальник сценаріїв Python є основною функцією з версії 4.0 від 7 лютого 2013 року.

Виходячи зі сфери використання і величезного функціоналу мови Python, який був описаний вище, можна зробити висновок про його мультифунціональність. Саме через це дана мова програмування була обрана для виконання даної роботи.

2.2 Фреймворк Django

Django - це безкоштовний веб-фреймворк на основі Python, що відповідає архітектурному зразку model-template-views (MTV). [9] [10] Він підтримується

Django Software Foundation (DSF), американською незалежною організацією, створеною як некомерційна організація 501 (c) (3).

Основна мета Django - полегшити створення складних веб-сайтів, керованих базами даних. Структура підкреслює багаторазовість та "підключеність" компонентів, менше коду, низьку зв'язок, швидкий розвиток та принцип "не повторюватися". [11] Python використовується всюди, навіть для налаштувань, файлів та моделей даних. Django також надає додатковий адміністративний інтерфейс створення, читання, оновлення та видалення, який генерується динамічно за допомогою самоаналізу та налаштовується за допомогою моделей адміністратора.

Деякі добре відомі сайти, які використовують Django, включають PBS, [12] Instagram, [13] Mozilla, [14] The Washington Times, [15] Disqus, [16] Bitbucket, [17] і Nextdoor. [18]

Python широко використовується у розробці різноманітних чат-ботів - розважальних, ділових, сервісних. Особливого значення розробка ботів займає в останні роки: чат-боти допомагають зменшити тиск на спеціалістів, розширити функціонал банків, кафе та інших установ. Для прикладу приведу всесвітньовідомий месенджер “Телеграм” - за допомогою саме Python всередині цієї потужньої екосистеми користувачі та розробники інтегрували тисячі ботів для покращення і без того дуже зручного месенджера. Донедавна саме за допомогою чат-ботів було можливо проводити транзакції у месенджері.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Діаграма варіантів використання

Діаграма використання найпростіша - це представлення взаємодії користувача із системою, яка показує взаємозв'язок між користувачем та різними випадками використання, в яких користувач бере участь. Діаграма випадків використання може ідентифікувати різні типи користувачів системи та різні випадки використання, і часто вона супроводжується також іншими типами діаграм. Варіанти використання представлені кругами або еліпсами.

Незважаючи на те, що сам випадок використання може детально вивчити кожну можливість, діаграма прикладів використання може допомогти забезпечити огляд системи на більш високому рівні. Раніше вже було сказано, що "схеми використання - це принципи вашої системи".

Через їх спрощений характер, схеми використання можуть бути хорошим інструментом комунікації для зацікавлених сторін. Креслення намагаються імітувати реальний світ і дають зацікавленій стороні уявлення про те, як буде розроблена система. Сіау та Лі провели дослідження, щоб визначити, чи взагалі існувала дійсна ситуація для схем використання або вони були непотрібними. Було виявлено, що діаграми випадків використання передають намір системи більш спрощеним чином зацікавленим сторонам і що вони "інтерпретуються більш повно, ніж діаграми класів".

Метою діаграми використання є відображення динамічного аспекту системи. Додаткові схеми та документація можуть бути використані для забезпечення повного функціонального та технічного уявлення про систему. Вони забезпечують спрощене та графічне представлення того, що система насправді повинна робити.

Елементи:

- рамки системи (англ. system border) - прямоугольник із назвою у верхніх частинах та еліпсами (прецедентами) всередині. Часто може бути опущено без корисної інформації про полезну інформацію,
- актор (англ. актор) - стилізований людський персонаж, обозначаючий набір ролей користувача (розуміється в широкому змісті: людина, внешняя сутність, клас, інша система), взаємодіючого з деякою сутністю (системною, підсистемою, класом). Актори не можуть бути пов'язані між собою з іншим (за вимкнення відносин щодо обробки / дослідження),
- прецедент - еліпс із надписом, що обозначає виконувану систематичну дію (може включати можливі варіанти), що призводить до спостерігаємих акторами результатів. Надпис може бути ім'ям або описом (з точки зору актора) того, "що" робить система (а не "як"). Ім прецедента зв’язано з неперервним (атомарним) сценарієм - конкретною послідовністю дій, ілюстріруючою повіддю. Під час сценарію актори обмінюються із систематичними повідомленнями. Сценарій може бути приведений на діаграмі прецедентів у відео UML-коментарі. З одним прецедентом може бути пов'язано кілька різних сценаріїв

На рисунку 3.1 зображено діаграму варіантів використання, яка описує можливі дії користувача в системі. Діаграма зображае лише можливі дії користувачів, без зв’язки з бекендом.

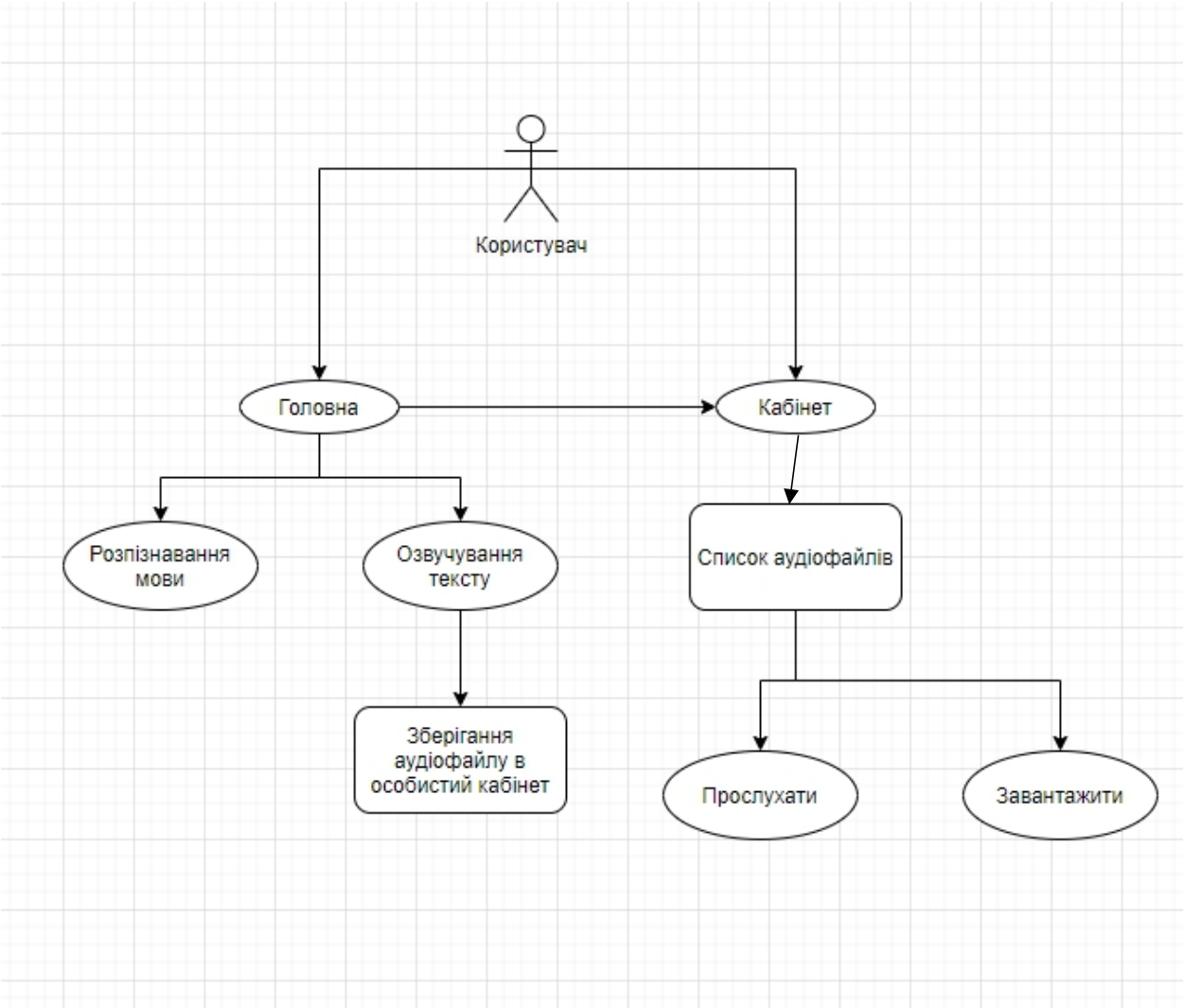


Рисунок 3.1 — Діаграма варіантів використання

3.2 Діаграма класів програмного продукту

У програмній інженерії діаграма класів в Уніфікованій мові моделювання (UML) - це тип статичної структурної діаграми, що описує структуру системи, показуючи класи системи, їх атрибути, операції (або методи) та взаємозв'язки між об'єктами.

Діаграма класів є основним будівельним елементом об’єктно-орієнтованого моделювання. Він використовується для загального концептуального моделювання структури програми та для детального моделювання переведення моделей у програмовий код. Діаграми класів також можуть бути використані для

моделювання даних. Класи на діаграмі класів представляють як основні елементи, взаємодії в програмі, так і класи, що програмуються.

На схемі класи представлені вікнами, які містять три відділення:

- У верхньому відділенні міститься назва класу. Надруковано жирним шрифтом і відцентровано, а перша літера написана великими літерами.
- Середній відсік містить атрибути класу. Вони вирівняні за лівим краєм, а перша буква мала.
- У нижньому відділенні містяться операції, які може виконувати клас. Вони також вирівняні за лівим краєм, а перша буква - мала.

При проектуванні системи ряд класів ідентифікується та згруповується у схему класів, яка допомагає визначити статичні відносини між ними. При детальному моделюванні класи концептуального проекту часто поділяються на ряд підкласів.

Залежність - це семантичний зв’язок між залежними та незалежними елементами моделі. Він існує між двома елементами, якщо зміни у визначенні одного елемента (сервера або цілі) можуть спричинити зміни для іншого (клієнта або джерела). Ця асоціація є односпрямованою. Залежність відображається у вигляді штрихової лінії з відкритою стрілкою, яка вказує від клієнта до постачальника.

Для подальшого опису поведінки систем ці діаграми класів можуть бути доповнені діаграмою стану або машиною стану UML.

Асоціація представляє родину посилань. Двійкова асоціація (з двома кінцями) зазвичай представляється у вигляді рядка.

Існує чотири різні типи асоціацій: двонаправлена, односпрямована, агрегаційна (включає агрегацію композиції) та рефлексивна. Двонаправлені та односпрямовані асоціації є найбільш поширеними.

Наприклад, клас польоту асоціюється з класом літака двонаправлено. Асоціація представляє статичне відношення, яке ділиться між об'єктами двох класів.

Агрегація є варіантом взаємозв'язку "має"; агрегація є більш конкретною, ніж асоціація. Це асоціація, яка представляє частково цілі або часткові стосунки. Як показано на зображенні, професор "має" клас для викладання. Як тип асоціації, агрегація може бути названа та мати ті самі прикраси, що і асоціація. Однак агрегація не може включати більше двох класів; це має бути бінарна асоціація. Крім того, навряд чи існує різниця між агрегаціями та асоціаціями під час реалізації, і діаграма може взагалі пропустити відносини агрегування. [7]

Агрегація може відбуватися, коли клас є колекцією або контейнером інших класів, але вміщені класи не мають сильної залежності життєвого циклу від контейнера. Вміст контейнера все ще існує, коли контейнер знищений.

В UML він графічно представлений у вигляді порожнистої форми ромба на вміщуючому класі одним рядком, що зв’язує його із вміщеним класом. Сукупність - це семантично розширений об’єкт, який у багатьох операціях трактується як одиниця, хоча фізично він складається з декількох менших об’єктів.

Приклад: Бібліотека та студенти. Тут студент може існувати без бібліотеки, зв'язок між студентом і бібліотекою є агрегацією.

Це вказує на те, що один із двох пов’язаних класів (підклас) вважається спеціалізованою формою іншого (супер тип), а суперклас - узагальненням підкласу. На практиці це означає, що будь-який екземпляр підтипу є також екземпляром суперкласу. Зразкове дерево узагальнень цієї форми зустрічається в біологічній класифікації: людина - це підклас маймуни, який є підкласом ссавців тощо. Зв’язок найлегше зрозуміти за допомогою фрази „А - це В” (людина - це ссавець, ссавець - тварина).

Графічне представлення UML узагальнення - це форма порожнистого трикутника на кінці суперкласу рядка (або дерева рядків), що зв’язує його з одним або кількома підтипами.

Відносини узагальнення також відомі як спадщина або відносини "є".

Суперклас (базовий клас) у відносинах узагальнення також відомий як "батьківський", суперклас, базовий клас або базовий тип.

Підтип у відносинах спеціалізації також відомий як "дочірній", підклас, похідний клас, похідний тип, клас успадкування або тип успадкування.

A - це тип B

Наприклад, "дуб - це тип дерева", "автомобіль - це тип транспортного засобу"

Узагальнення може бути показано лише на діаграмах класів та на діаграмах використання.

При моделюванні UML взаємозв'язок реалізації - це взаємозв'язок між двома елементами моделі, в яких один елемент моделі (клієнт) реалізує (реалізує або виконує) поведінку, яку вказує інший елемент моделі (постачальник).

Графічне представлення UML реалізації - це порожниста форма трикутника на кінці інтерфейсу штрихової лінії (або дерева рядків), яка з'єднує її з одним або кількома реалізаторами. Проста головка стрілки використовується на кінці інтерфейсу штрихової лінії, що з'єднує її з користувачами. У діаграмах компонентів використовується графічна умова «м'яч і сокет» (реалізатори виставляють кульку або льодяник, тоді як користувачі показують сокет). Реалізації можна показати лише на діаграмах класів або компонентів. Реалізація - це взаємозв'язок між класами, інтерфейсами, компонентами та пакетами, що з'єднує елемент клієнта з елементом постачальника. Зв'язок реалізації між класами / компонентами та інтерфейсами показує, що клас / компонент реалізує операції, пропоновані інтерфейсом.

Залежність - це слабша форма зв'язку, яка вказує на те, що один клас залежить від іншого, оскільки він використовує його в певний момент часу. Один клас залежить від іншого, якщо незалежний клас є змінною параметра або локальною змінною методу залежного класу. Це відрізняється від асоціації, де атрибут залежного класу є екземпляром незалежного класу. Іноді відносини між двома класами дуже слабкі. Вони взагалі не реалізовані зі змінними-членами.

Представлення UML асоціації - це лінія, що з'єднує два пов'язані класи. На кожному кінці рядка є додаткові позначення. Наприклад, ми можемо вказати, використовуючи наконечник стрілки, що загострений кінець видно з хвоста стрілки. Ми можемо вказати власність шляхом розміщення кульки, ролі, яку

відіграють елементи цього кінця, вказавши ім'я ролі та множинність екземплярів цієї сутності (діапазон кількості об'єктів, які беруть участь в асоціації з точки зору іншого кінця).

Класи сутності моделюють довгоживучу інформацію, якою обробляє система, а іноді і поведінку, пов'язану з цією інформацією. Їх не слід ідентифікувати як таблиці баз даних чи інших сховищ даних.

Вони намальовані як кола з короткою лінією, прикріпленою до нижньої частини кола.

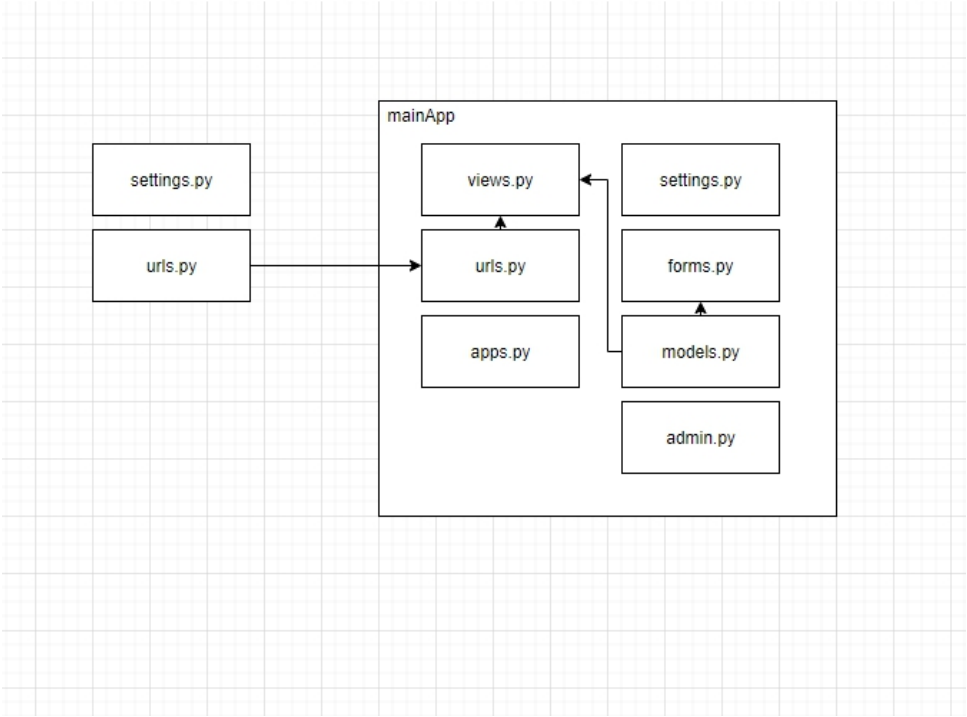


Рисунок 3.2 — Діаграма класів

3.3 Графічний інтерфейс користувача

Робота користувача з ресурсом починається з авторизації (рис. 3.3) і реєстрації (рис. 3.4).

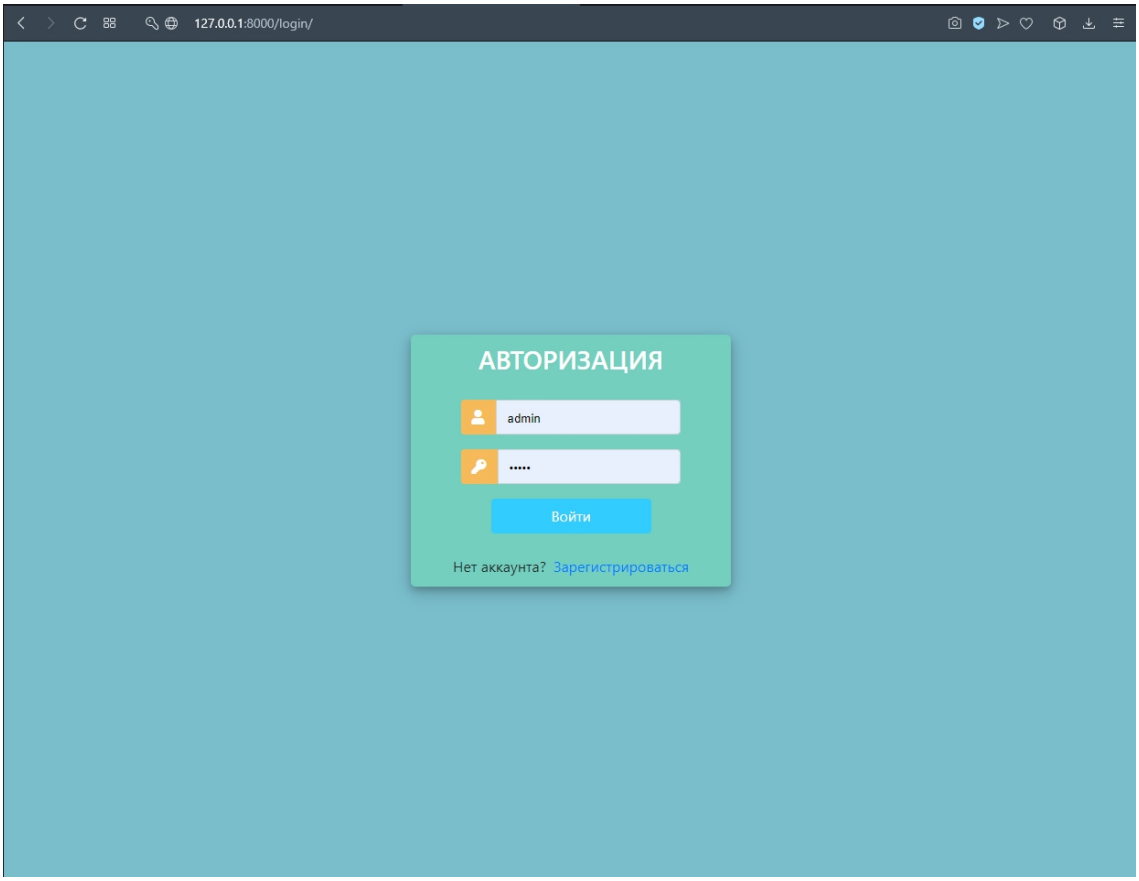


Рисунок 3.3 — Форма авторизації

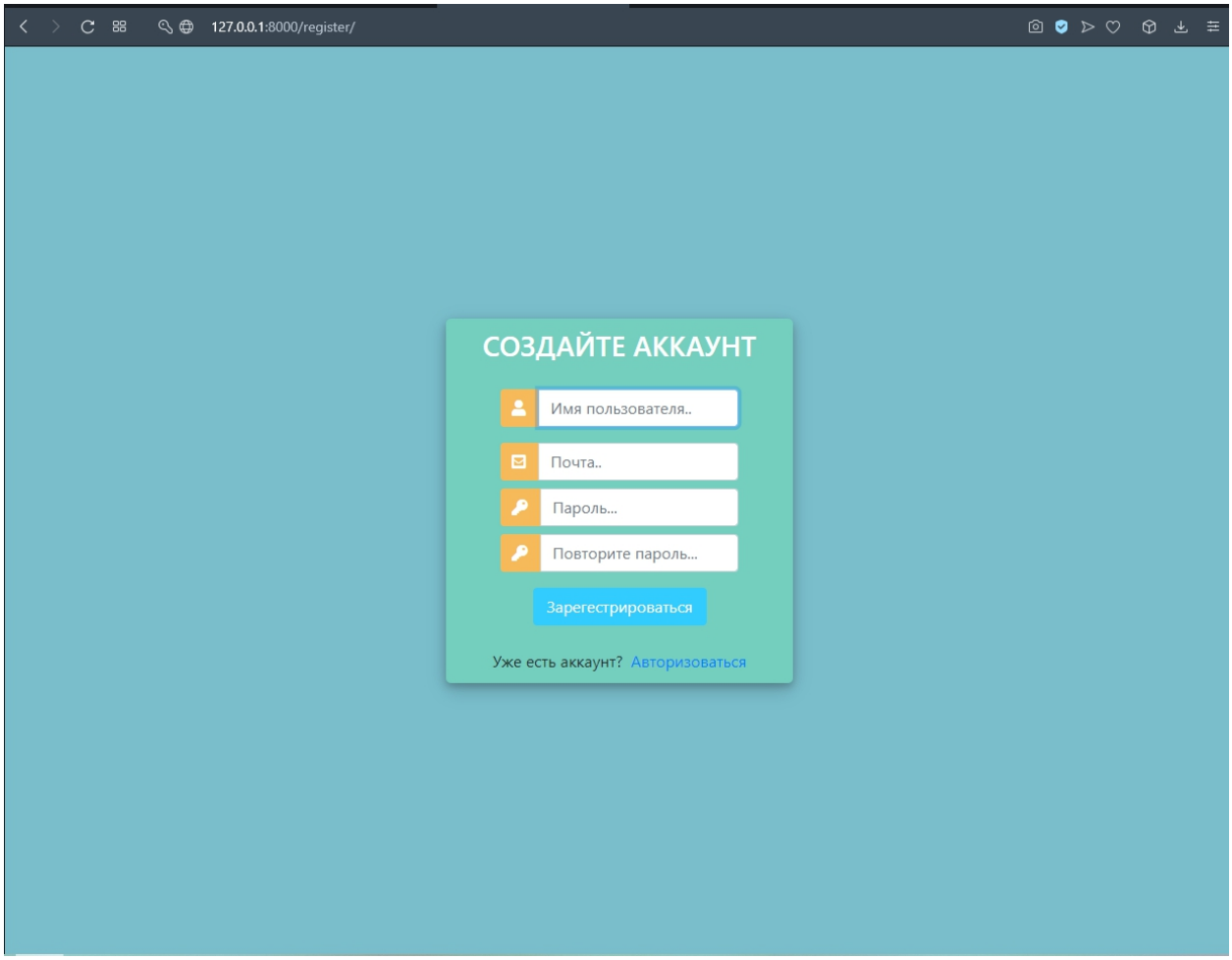


Рисунок 3.4 — Форма реєстрації

Після авторизації користувач потрапляє на головну сторінку (рис. 3.5), де має можливість скористатися розпізнаванням мови (рис. 3.6) або озвучуванням мови (рис. 3.7)

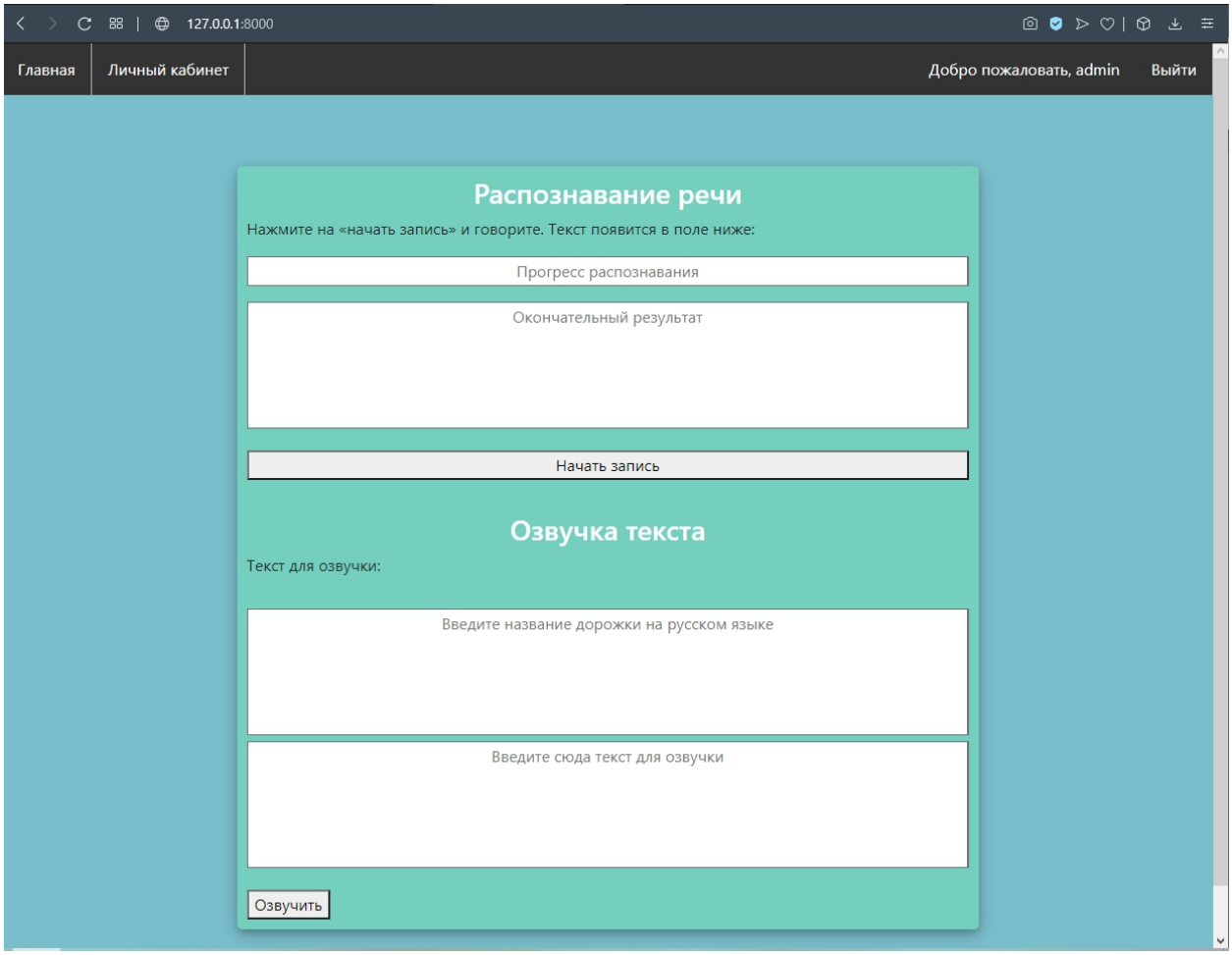


Рисунок 3.5 — Головна сторінка

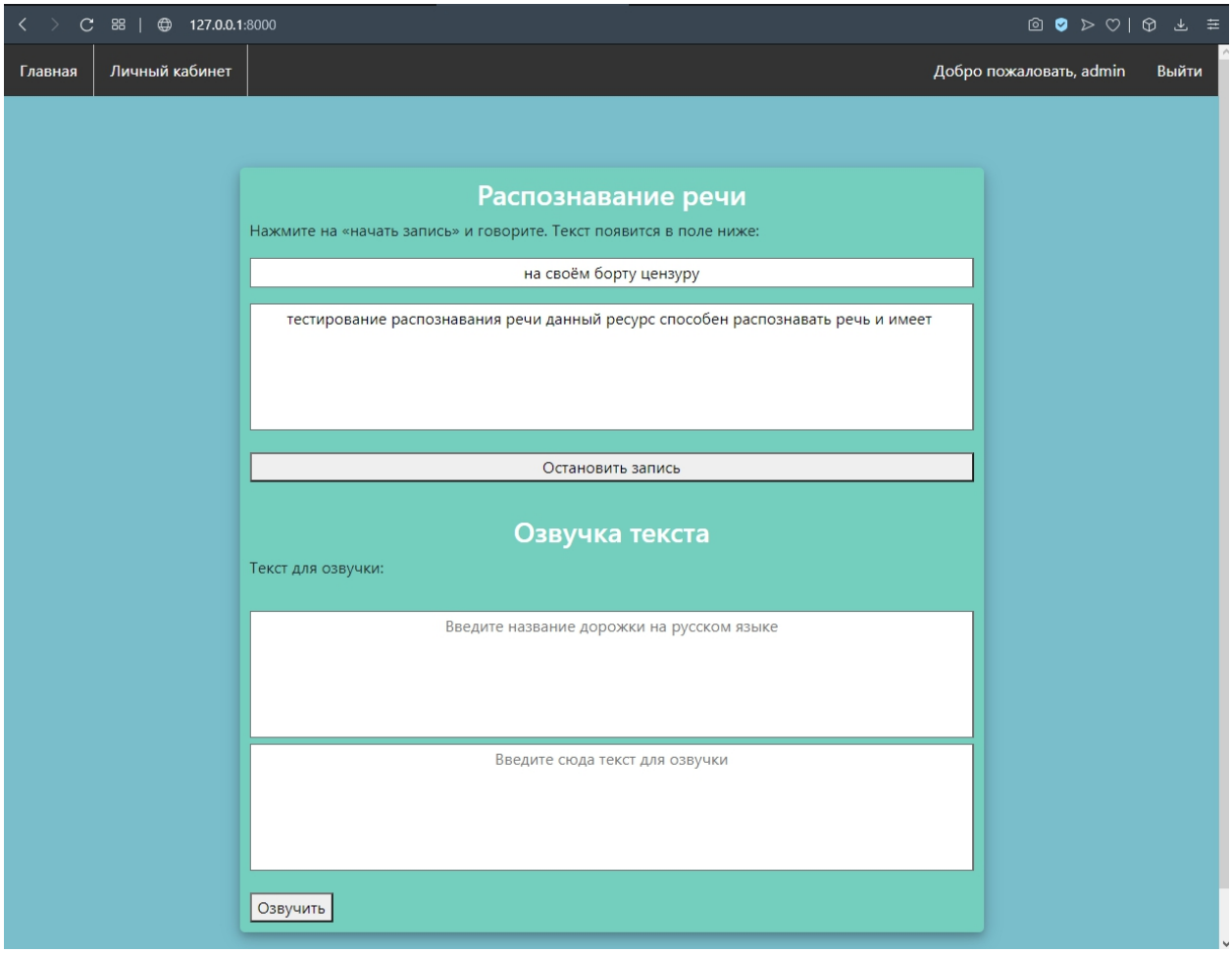


Рисунок 3.6 — Процес розпізнавання мови

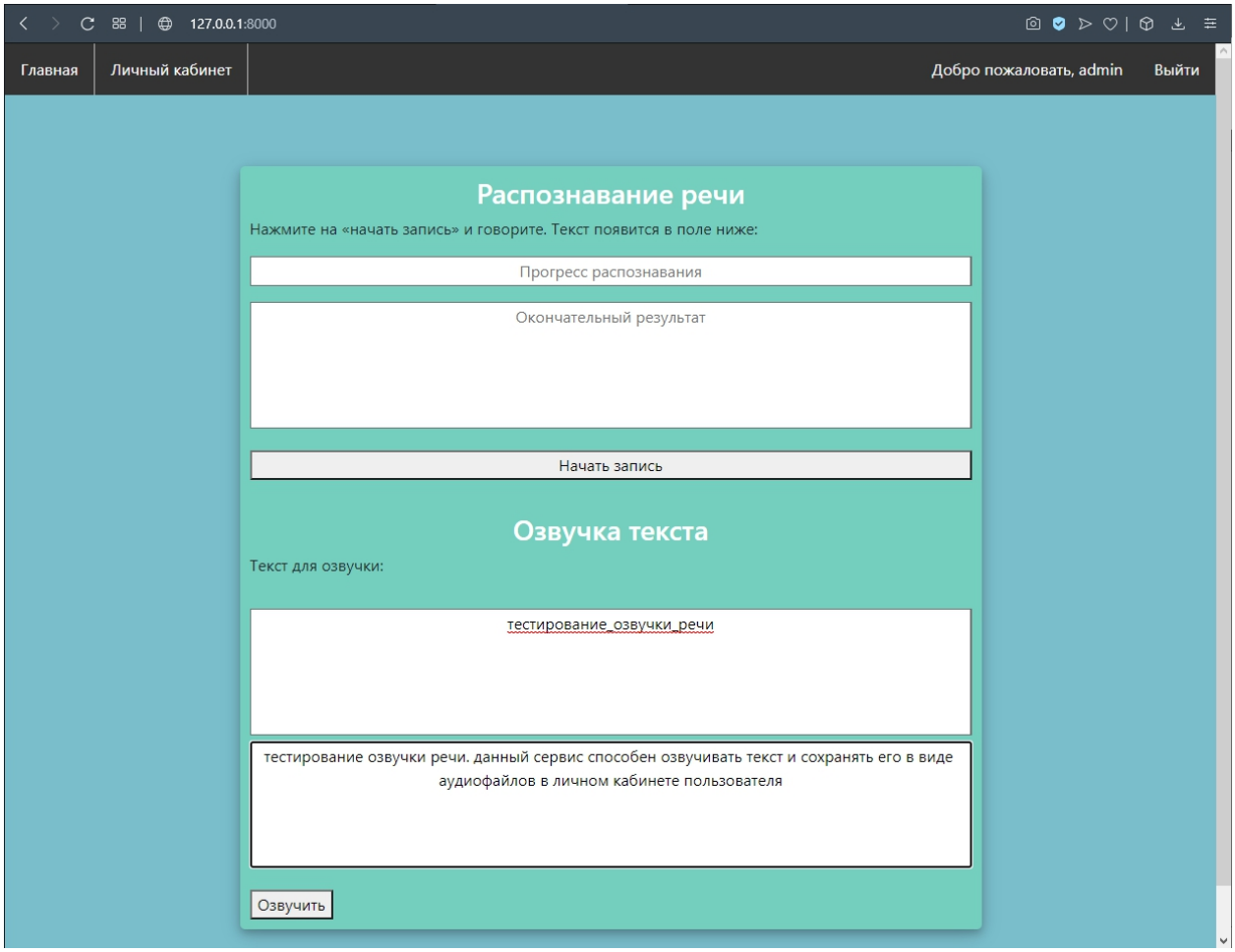


Рисунок 3.7 — Озвучування тексту

Для перегляду озвучених текстів користувачеві слід перейти в особистий кабінет (рис. 3.8), де він може перейти до прослуховування і скачування аудіофайлів (рис. 3.9).

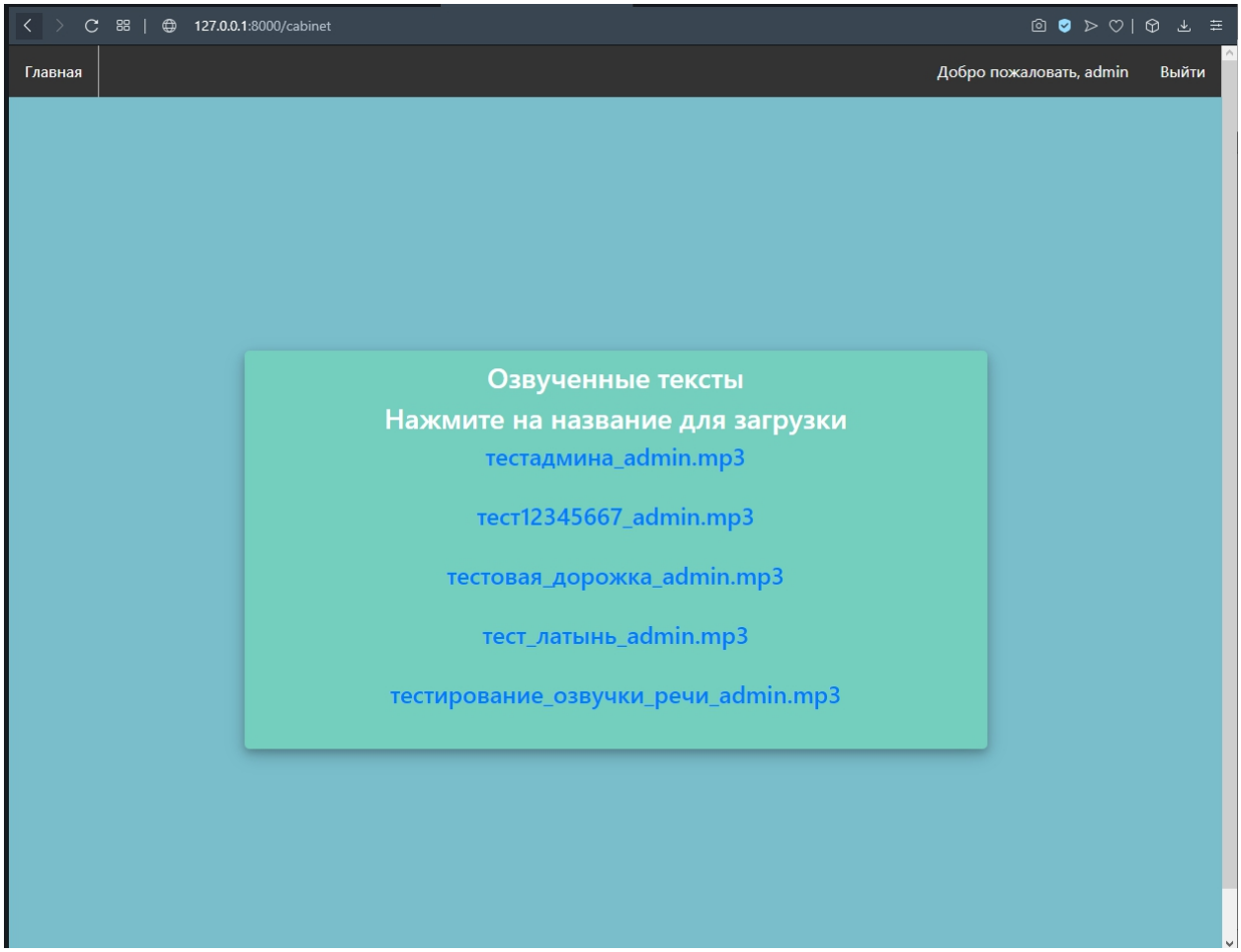


Рисунок 3.8 — Особистий кабінет

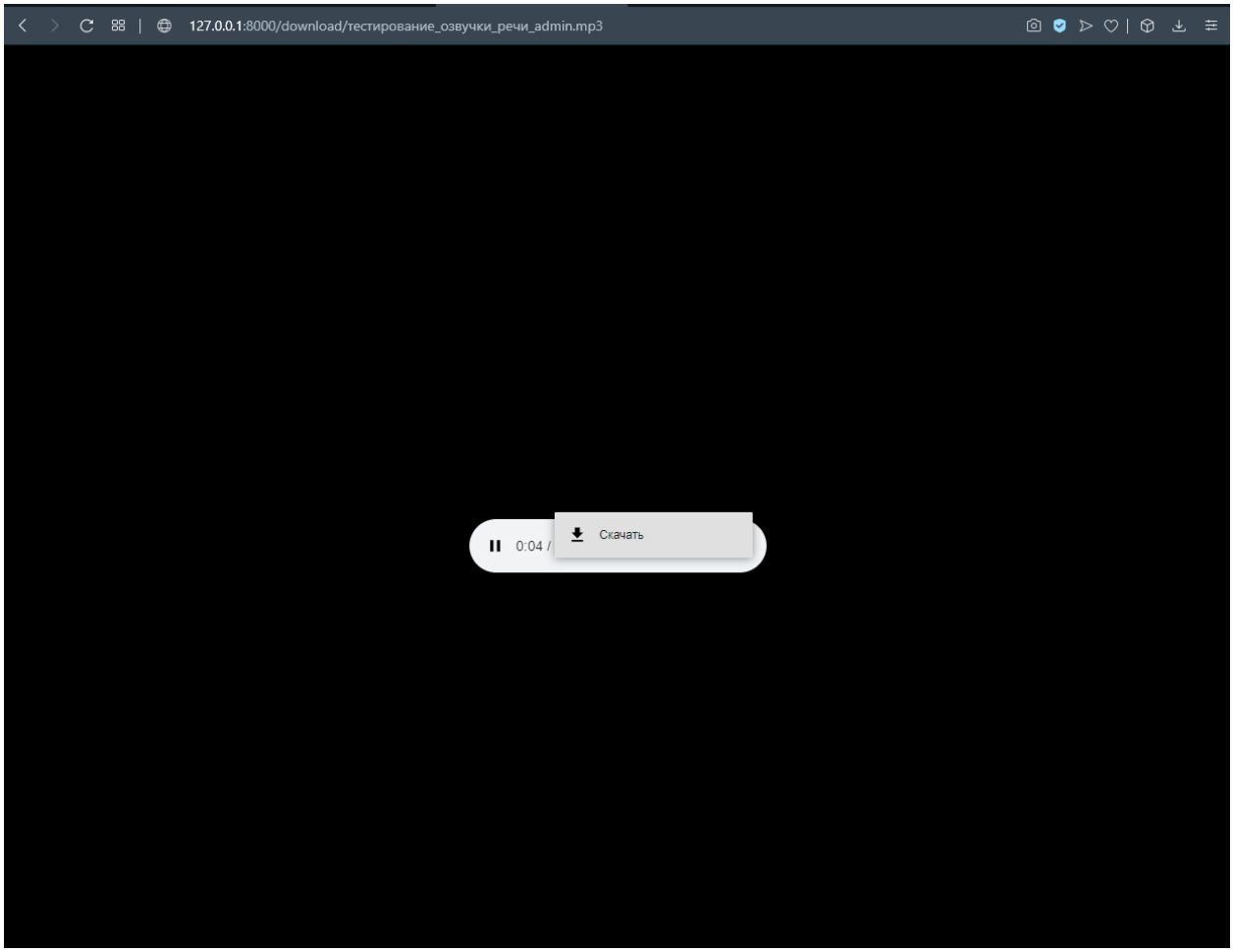


Рисунок 3.9 — Вікно прослуховування і завантаження

3.4 Тестування додатку

Для перевірки коректного виконання додатком функцій, закладених в технічному завданні. Були складені тестові набори, показані в таблицях 3.1 – 3.2.

Тестові набори включають в себе перевірку на кроссбраузерность Веб Програми. Веб-додаток повинен коректно відображатися в відомих браузерах: Safari, Opera, Mozilla Firefox.

Таблиця 3.1 – Тестові набори для перевірки функціональності

сторінка	браузер	відображення
index.html	Safari	Коректне без змін
	Opera	Коректне без змін
	Mozilla Firefox	Коректне без змін
cabinet.html	Safari	Коректне без змін
	Opera	Коректне без змін
	Mozilla Firefox	Коректне без змін
login.html	Safari	Коректне без змін
	Opera	Коректне без змін
	Mozilla Firefox	Коректне без змін
register.html	Safari	Коректне без змін
	Opera	Коректне без змін
	Mozilla Firefox	Коректне без змін

Під час розробки системи і вирішення поставлених завдань, були розроблені такі тестовані функції:

1. функцій видалення інформації адміністратором;
2. функції реєстрації користувача;
3. функції розпізнавання мови;

4. функції синтезу мови;
5. функції відображення сбереження аудіофайлу;
6. функції відображення списку аудіфайлів користувча;
7. функція авторизації в системі;
8. функція виходу з системи.

На базі проведеного тестування можна зробити висновок, що основні функції інформаційної системи працюють чітко і без неточностей - результат нижче.

Таблиця 3.2 – Таблиця тестованих функцій видалення / редагування / створення інформації адміністратором.

	№ Назва тесту	Тестований результат	Очікуваний результат	Статус
1	Тестування Функції транскрибування мови	Вивід мови в форматі тексту	Вивід мови в форматі тексту	Виконано
2	Тестування функції синтезу мови	Успішне збереження файлу в директорії	Успішне збереження файлу в директорії	Виконано
3	Тестування функції виводу списку аудіофайлів користувача	Успішне виведенян файлів, які належать даному користувачеві	Успішне виведенян файлів, які належать даному користувачеві	Виконано

ВИСНОВКИ

Робота присвячена розробці системи транскрибування і синтезу мови засобами HTML, CSS, JS та Python.

Проведено дослідження, котрі обгрунтовують актуальність роботи та наукову новизну. А саме, рівень зацікавленності сучасного суспільства в технологіях транскрибування та синтезу мови та їх необхідність на сучасному ринку програмного забезпечення.

Було проаналізовано предметну область. Проведений аналіз понять синтезу та розпізнавання мови, розробки веб сайтів. Отримані знання були використані для розробки власного алгоритму.

Описано програмні засоби, котрі було застосовано для розробки програмного забезпечення. Визначено, що оптимальним рішенням є колаборація із стандартного веб-стеку для фронтенду HTML, CSS, JS та мови програмування Python у зв'язці з фреймворком Django для серверної частини.

Підібрано відповідні технології, які якнайкраще сприяють розкриттю повного функціоналу програмного забезпечення, створено модулі взаємодії між технологіями, власноруч написані всі сторінки і взаємодії між ними.

Повністю описаний процес розробки програмного забезпечення за домогою UML-діаграм та скріншотів розробленго графічного інтерфейсу користувача.

Проведено тестування за допомогою ручних процедур перевірки (тестування) програмного забезпечення, а саме тестових наборів, і показано, що дане ПЗ виконує всі функції, задля виконання яких воно було розроблене.

Застосунок буде ефективним для користувачів, які мають необхідність працювати з великими обсягами текстових даних, які необхідно перевести в формат звуку, або навпаки, для користувачів, які мають необхідні в переведенні усної мови в текстовий формат.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. "Speaker Independent Connected Speech Recognition- Fifth Generation Computer Corporation". Fifthgen.com. Archived from the original on 11 November 2013. Retrieved 15 June 2013.
2. P. Nguyen (2010). "Automatic classification of speaker characteristics". International Conference on Communications and Electronics 2010. pp. 147–152. doi:10.1109/ICCE.2010.5670700. ISBN 978-1-4244-7055-6. S2CID 13482115.
3. "British English definition of voice recognition". Macmillan Publishers Limited. Archived from the original on 16 September 2011. Retrieved 21 February 2012.
4. "voice recognition, definition of". WebFinance, Inc. Archived from the original on 3 December 2011. Retrieved 21 February 2012.
5. "The Mailbag LG #114". Linuxgazette.net. Archived from the original on 19 February 2013. Retrieved 15 June 2013.
6. Sarangi, Susanta; Sahidullah, Md; Saha, Goutam (September 2020). "Optimization of data-driven filterbank for automatic speaker verification". Digital Signal Processing. 104: 102795. arXiv:2007.10729. doi:10.1016/j.dsp.2020.102795. S2CID 220665533.
7. Reynolds, Douglas; Rose, Richard (January 1995). "Robust text-independent speaker identification using Gaussian mixture speaker models" (PDF). IEEE Transactions on Speech and Audio Processing. 3 (1): 72–83. doi:10.1109/89.365379. ISSN 1063-6676. OCLC 26108901. Archived (PDF) from the original on 8 March 2014. Retrieved 21 February 2014.
8. "Speaker Identification (WhisperID)". Microsoft Research. Microsoft. Archived from the original on 25 February 2014. Retrieved 21 February 2014. When you speak to someone, they don't just recognize what you say: they recognize who you are. WhisperID will let computers do that, too, figuring out who you are by the way you sound.
9. "Obituaries: Stephen Balashek". The Star-Ledger. 22 July 2012.
10. "IBM-Shoebox-front.jpg". androidauthority.net. Retrieved 4 April 2019.

11. Juang, B. H.; Rabiner, Lawrence R. "Automatic speech recognition–a brief history of the technology development" (PDF): 6. Archived (PDF) from the original on 17 August 2014. Retrieved 17 January 2015.
12. Melanie Pinola (2 November 2011). "Speech Recognition Through the Decades: How We Ended Up With Siri". PC World. Retrieved 22 October 2018.
13. Gray, Robert M. (2010). "A History of Realtime Digital Speech on Packet Networks: Part II of Linear Predictive Coding and the Internet Protocol" (PDF). Found. Trends Signal Process. 3 (4): 203–303. doi:10.1561/2000000036. ISSN 1932-8346.
14. John R. Pierce (1969). "Whither speech recognition?". Journal of the Acoustical Society of America. 46 (48): 1049–1051. Bibcode:1969ASAJ...46.1049P. doi:10.1121/1.1911801.
15. Benesty, Jacob; Sondhi, M. M.; Huang, Yiteng (2008). Springer Handbook of Speech Processing. Springer Science & Business Media. ISBN 978-3540491255.
16. John Makhoul. "ISCA Medalist: For leadership and extensive contributions to speech and language processing". Archived from the original on 24 January 2018. Retrieved 23 January 2018.
17. Blechman, R. O.; Blechman, Nicholas (23 June 2008). "Hello, Hal". The New Yorker. Archived from the original on 20 January 2015. Retrieved 17 January 2015.
18. Klatt, Dennis H. (1977). "Review of the ARPA speech understanding project". The Journal of the Acoustical Society of America. 62 (6): 1345–1366. Bibcode:1977ASAJ...62.1345K. doi:10.1121/1.381666.
19. Rabiner (1984). "The Acoustics, Speech, and Signal Processing Society. A Historical Perspective" (PDF). Archived (PDF) from the original on 9 August 2017. Retrieved 23 January 2018.
20. "First-Hand:The Hidden Markov Model – Engineering and Technology History Wiki". ethw.org. Archived from the original on 3 April 2018. Retrieved 1 May 2018.

21. "James Baker interview". Archived from the original on 28 August 2017. Retrieved 9 February 2017.
22. "Pioneering Speech Recognition". 7 March 2012. Archived from the original on 19 February 2015. Retrieved 18 January 2015.
23. Xuedong Huang; James Baker; Raj Reddy. "A Historical Perspective of Speech Recognition". Communications of the ACM. Archived from the original on 20 January 2015. Retrieved 20 January 2015.
24. Juang, B. H.; Rabiner, Lawrence R. "Automatic speech recognition–a brief history of the technology development" (PDF): 10. Archived (PDF) from the original on 17 August 2014. Retrieved 17 January 2015.
25. "History of Speech Recognition". Dragon Medical Transcription. Archived from the original on 13 August 2015. Retrieved 17 January 2015.
26. Kevin McKean (8 April 1980). "When Cole talks, computers listen". Sarasota Journal. AP. Retrieved 23 November 2015.
27. Melanie Pinola (2 November 2011). "Speech Recognition Through the Decades: How We Ended Up With Siri". PC World. Archived from the original on 13 January 2017. Retrieved 28 July 2017.
28. "Ray Kurzweil biography". KurzweilAINetwork. Archived from the original on 5 February 2014. Retrieved 25 September 2014.
29. Juang, B.H.; Rabiner, Lawrence. "Automatic Speech Recognition – A Brief History of the Technology Development" (PDF). Archived (PDF) from the original on 9 August 2017. Retrieved 28 July 2017.
30. "Nuance Exec on iPhone 4S, Siri, and the Future of Speech". Tech.pinions. 10 October 2011. Archived from the original on 19 November 2011. Retrieved 23 November 2011.
31. "Switchboard-1 Release 2". Archived from the original on 11 July 2017. Retrieved 26 July 2017.
32. Jason Kincaid. "The Power of Voice: A Conversation With The Head Of Google's Speech Technology". Tech Crunch. Archived from the original on 21 July 2015. Retrieved 21 July 2015.

33. Froomkin, Dan (5 May 2015). "THE COMPUTERS ARE LISTENING". The Intercept. Archived from the original on 27 June 2015. Retrieved 20 June 2015.
34. Herve Bourlard and Nelson Morgan, Connectionist Speech Recognition: A Hybrid Approach, The Kluwer International Series in Engineering and Computer Science; v. 247, Boston: Kluwer Academic Publishers, 1994.
35. Sepp Hochreiter; J. Schmidhuber (1997). "Long Short-Term Memory". Neural Computation. 9 (8): 1735–1780. doi:10.1162/neco.1997.9.8.1735. PMID 9377276. S2CID 1915014.
36. Schmidhuber, Jürgen (2015). "Deep learning in neural networks: An overview". Neural Networks. 61: 85–117. arXiv:1404.7828. doi:10.1016/j.neunet.2014.09.003. PMID 25462637. S2CID 11715509.
37. Alex Graves, Santiago Fernandez, Faustino Gomez, and Jürgen Schmidhuber (2006). Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural nets. Proceedings of ICML'06, pp. 369–376.
38. Santiago Fernandez, Alex Graves, and Jürgen Schmidhuber (2007). An application of recurrent neural networks to discriminative keyword spotting. Proceedings of ICANN (2), pp. 220–229.
39. Haşim Sak, Andrew Senior, Kanishka Rao, Françoise Beaufays and Johan Schalkwyk (September 2015): "Google voice search: faster and more accurate." Archived 9 March 2016 at the Wayback Machine
40. "Li Deng". Li Deng Site.
41. NIPS Workshop: Deep Learning for Speech Recognition and Related Applications, Whistler, BC, Canada, Dec. 2009 (Organizers: Li Deng, Geoff Hinton, D. Yu).
42. Hinton, Geoffrey; Deng, Li; Yu, Dong; Dahl, George; Mohamed, Abdel-Rahman; Jaitly, Navdeep; Senior, Andrew; Vanhoucke, Vincent; Nguyen, Patrick; Sainath, Tara; Kingsbury, Brian (2012). "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The shared views of four research groups". IEEE Signal Processing Magazine. 29 (6): 82–97. Bibcode:2012ISPM...29...82H. doi:10.1109/MSP.2012.2205597. S2CID 206485943.

43. Deng, L.; Hinton, G.; Kingsbury, B. (2013). "New types of deep neural network learning for speech recognition and related applications: An overview". 2013 IEEE International Conference on Acoustics, Speech and Signal Processing: New types of deep neural network learning for speech recognition and related applications: An overview. p. 8599. doi:10.1109/ICASSP.2013.6639344. ISBN 978-1-4799-0356-6. S2CID 13953660.
44. Markoff, John (23 November 2012). "Scientists See Promise in Deep-Learning Programs". New York Times. Archived from the original on 30 November 2012. Retrieved 20 January 2015.
45. Morgan, Bourlard, Renals, Cohen, Franco (1993) "Hybrid neural network/hidden Markov model systems for continuous speech recognition. ICASSP/IJPRAI"
46. T. Robinson (1992). "A real-time recurrent error propagation network word recognition system". [Proceedings] ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing. pp. 617–620 vol.1. doi:10.1109/ICASSP.1992.225833. ISBN 0-7803-0532-9. S2CID 62446313.
47. Waibel, Hanazawa, Hinton, Shikano, Lang. (1989) "Phoneme recognition using time-delay neural networks. IEEE Transactions on Acoustics, Speech, and Signal Processing."
48. Baker, J.; Li Deng; Glass, J.; Khudanpur, S.; Chin-Hui Lee; Morgan, N.; O'Shaughnessy, D. (2009). "Developments and Directions in Speech Recognition and Understanding, Part 1". IEEE Signal Processing Magazine. 26 (3): 75–80. Bibcode:2009ISPM...26...75B. doi:10.1109/MSP.2009.932166. S2CID 357467.
49. Sepp Hochreiter (1991), Untersuchungen zu dynamischen neuronalen Netzen Archived 6 March 2015 at the Wayback Machine, Diploma thesis. Institut f. Informatik, Technische Univ. Munich. Advisor: J. Schmidhuber.
50. Bengio, Y. (1991). Artificial Neural Networks and their Application to Speech/Sequence Recognition (Ph.D.). McGill University.
51. Deng, L.; Hassanein, K.; Elmasry, M. (1994). "Analysis of the correlation structure for a neural predictive model with application to speech recognition". Neural Networks. 7 (2): 331–339. doi:10.1016/0893-6080(94)90027-2.

52. Keynote talk: Recent Developments in Deep Neural Networks. ICASSP, 2013 (by Geoff Hinton).
53. Keynote talk: "Achievements and Challenges of Deep Learning: From Speech Analysis and Recognition To Language and Multimodal Processing," Interspeech, September 2014 (by Li Deng).
54. "Improvements in voice recognition software increase". TechRepublic.com. 27 August 2002. Maners said IBM has worked on advancing speech recognition ... or on the floor of a noisy trade show.
55. "Voice Recognition To Ease Travel Bookings: Business Travel News". BusinessTravelNews.com. 3 March 1997. The earliest applications of speech recognition software were dictation ... Four months ago, IBM introduced a 'continual dictation product' designed to ... debuted at the National Business Travel Association trade show in 1994.
56. Ellis Booker (14 March 1994). "Voice recognition enters the mainstream". Computerworld. p. 45. Just a few years ago, speech recognition was limited to ...
57. "Microsoft researchers achieve new conversational speech recognition milestone". 21 August 2017.
58. Goel, Vaibhava; Byrne, William J. (2000). "Minimum Bayes-risk automatic speech recognition". Computer Speech & Language. 14 (2): 115–135. doi:10.1006/csla.2000.0138. Archived from the original on 25 July 2011. Retrieved 28 March 2011.
59. Mohri, M. (2002). "Edit-Distance of Weighted Automata: General Definitions and Algorithms" (PDF). International Journal of Foundations of Computer Science. 14 (6): 957–982. doi:10.1142/S0129054103002114. Archived (PDF) from the original on 18 March 2012. Retrieved 28 March 2011.
60. Waibel, A.; Hanazawa, T.; Hinton, G.; Shikano, K.; Lang, K. J. (1989). "Phoneme recognition using time-delay neural networks". IEEE Transactions on Acoustics, Speech, and Signal Processing. 37 (3): 328–339. doi:10.1109/29.21701. hdl:10338.dmlcz/135496.

ДОДАТОК А

Розробка основних механік

Першочерговою механікою роботи системи є механіка авторизації. Вона реалізована за допомогою форм Django (лістинг 3.1)

Лістинг 3.1.

Авторизація користувача

```
<!DOCTYPE html>
<html>

<head>
    <title>Авторизация</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO" crossorigin="anonymous">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
    <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.6.1/css/all.css" integrity="sha384-gfdkjb5BdAXd+lj+gudLWI+BXq4IuLW5IT+brZEZsLFm++aCMlF1V92rMkPaX4PP" crossorigin="anonymous">
</head>
<body>
    <div class="container h-100">
        <div class="d-flex justify-content-center h-100">
            <div class="user_card">
                <div class="d-flex justify-content-center">
```

```
                    <h3 id="form-title">АВТОРИЗАЦИЯ</h3>
                </div>
                <div class="d-flex justify-content-center form_container">
                    <form method="POST" action="">
            {% csrf_token %}
                        <div class="input-group mb-3">
                            <div class="input-group-append">
                                <span class="input-group-text"><i class="fas fa-user"></i></span>
                            </div>
                            <input type="text" name="username" placeholder="Логин..." class="form-control">
                        </div>
                        <div class="input-group mb-2">
                            <div class="input-group-append">
                                <span class="input-group-text"><i class="fas fa-key"></i></span>
                            </div>
                                <input type="password" name="password" placeholder="Пароль..." class="form-control" >
                        </div>

                            <div class="d-flex justify-content-center mt-3 login_container">
                                <input class="btn login_btn" type="submit" value="Войти">
                            </div>
                    </form>
```

```
                        </div>

            {% for message in messages %}
              <p id="message">{{message}}</p>
            {% endfor %}

                        <div class="mt-4">
                            <div class="d-flex justify-content-center links">
                                Нет аккаунта? <a href="{% url 'register' %}" class="ml-2">Зарегистрироваться</a>
                            </div>

                        </div>
                    </div>
                </div>
            </div>
        </body>

        </html>
        def loginPage(request):
            if request.method == 'POST':
                username = request.POST.get('username')
                password = request.POST.get('password')
                user = authenticate(request, username=username, password=password)

                if user is not None:
                    login(request, user)
                    return redirect('index')
                else:
                    messages.info(request, 'Имя пользователя или пароль введены неверно')
```

```
    context = {}
    return render(request, 'login.html', context)
```

Окрім авторизації подібного роду ресурсу необхідна механіка реєстрації нових користувачів, яка відображена в лістингу 3.2

Лістинг 3.2

Реєстрація користувача

```
<body>
        <div class="container h-100">
                <div class="d-flex justify-content-center h-100">
                        <div class="user_card">
                                <div class="d-flex justify-content-center">
                                        <h3 id="form-title">СОЗДАЙТЕ АККАУНТ</h3>
                                </div>
                                <div class="d-flex justify-content-center form_container">

                                        <form method="POST" action="">
                        {% csrf_token %}
                                                <div class="input-group mb-3">
                                                        <div class="input-group-append">
                                                                <span class="input-group-text"><i
class="fas fa-user"></i></span>
                                                        </div>
                                {{form.username}}
                                                </div>
                                                <div class="input-group mb-2">
                                                        <div class="input-group-append">
                                                                <span class="input-group-text"><i
class="fas fa-envelope-square"></i></span>
                                                        </div>
```

```
                    {{form.email}}
                </div>
                <div class="input-group mb-2">
                    <div class="input-group-append">
                        <span class="input-group-text"><i
class="fas fa-key"></i></span>
                    </div>
                    {{form.password1}}
                </div>
                <div class="input-group mb-2">
                    <div class="input-group-append">
                        <span class="input-group-text"><i
class="fas fa-key"></i></span>
                    </div>
                    {{form.password2}}
                </div>

                <div class="d-flex justify-content-center mt-3
login_container">
                    <input class="btn login_btn" type="submit"
value="Зарегестрироваться">
                </div>
            </form>
        </div>
            {{form.errors}}
        <div class="mt-4">
            <div class="d-flex justify-content-center links">
                Уже есть аккаунт? <a href="{% url 'login' %}"
class="ml-2">Авторизоваться</a>
            </div>
```

```
                    </div>
                </div>
            </div>
        </div>
        <script>

            var form_fields = document.getElementsByTagName('input')
            form_fields[1].placeholder='Имя пользователя..';
            form_fields[2].placeholder='Почта..';
            form_fields[3].placeholder='Пароль...';
            form_fields[4].placeholder='Повторите пароль...';


            for (var field in form_fields){
                form_fields[field].className += ' form-control'
            }
        </script>
</body>
</html>
def regist(request):
    form = CreateUserForm
    if request.method == "POST":
        form = CreateUserForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('login')
    context = {'form': form}
    return render(request, 'register.html', context)
class CreateUserForm(UserCreationForm):
    class Meta:
```

```
        model = User
        fields = ['username', 'email', 'password1', 'password2']
```

Окрім того в системі присутня база даних, яка відображається в проекті за рахунок побудови і реєстрації моделей сутностей (лістинг 3.3)

Лістинг 3.3.

Побудова і реєстрація моделей

```
from django.db import models
from django.contrib.auth.models import User
from django.core.files.storage import FileSystemStorage
from django.core.files import File

from audiohelper import settings


class text_to_speech_Data(models.Model):
    audio_name = models.TextField('Название дорожки')
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    audio_file = models.FileField(blank=True, null=True)

    def __str__(self):
        return self.audio_name

    def save_file(self):
        self.audio_file.save(self.audio_name, self.audio_file)

    class Meta:
        verbose_name = 'Аудиофайл'
        verbose_name_plural = 'Аудиофайлы'
```

```
from django.contrib import admin
from .models import text_to_speech_Data
admin.site.register(text_to_speech_Data)
```

Головна сторінка представляє собою реалізацію алгоритмів розпізнавання мови і озвучування тексту зі збереженням його в файли (лістинг 3.4)

Лістинг 3.4.

Розпізнавання мови і озвучування тексту

```
        <body>
<ul>
  <li><a href="{% url 'index' %}">Главная</a></li>
    <li><a href="{% url 'cabinet' %}">Личный кабинет</a></li>
    <li  style="float:right" st><a href="{% url 'logout' %}">Выйти</a></li>
  <li class="usrName">Добро пожаловать, {{request.user}}</li>

</ul>

  <div class="container h-100">
                <div class="d-flex justify-content-center h-100">
                        <div class="user_card">
                                <div class="d-flex justify-content-center">


                                        <h3 class="form-title">Распознавание речи</h3>


                                </div>
```

```
            <p>Нажмите на «начать запись» и говорите. Текст появится в поле ниже:</p>
<p>
  <input id="interim" placeholder="Прогресс распознавания">
</p>
<p>
  <textarea id="message" placeholder="Окончательный результат"></textarea>
</p>
<button id="recognize">Начать запись</button>
<hr>

                <div class="d-flex justify-content-center">


                                        <h3 class="form-title">Озвучка текста</h3>

                                    </div>

<p>Текст для озвучки:</p>
<p>
      <form method="POST" action="">
  <textarea id = "name"  name = "name" placeholder="Введите название дорожки на русском языке"></textarea>
  <textarea id="text" name="text" placeholder="Введите сюда текст для озвучки"></textarea>
</p>

                    {% csrf_token %}
                  <button id="speak" method = "POST">Озвучить</button>
                </form>
```

```
<script type="text/javascript">
        function isMobile() {
  return /iPhone|iPad|iPod|Android/i.test(navigator.userAgent);
}

const SpeechRecognition = window.SpeechRecognition || window.webkitSpeechRecognition || window.mozSpeechRecognition || window.msSpeechRecognition;

class Recognizer {
  constructor() {
    this.recognition = new SpeechRecognition();
    this.recognition.lang = "ru-RU";
    if (!isMobile()) {
      this.recognition.continuous = true;
      this.recognition.interimResults = true;
    }
    this.isRecognizing = false;
    this.transcript = "";
  }

  start(handler) {
    this.transcript = "";
    this.recognition.onresult = (event) => {
      this.onResult(event, handler);
    };
    this.recognition.start();
```

```
    this.isRecognizing = true;
    console.log("Started recognition");
  }

  stop() {
    this.recognition.abort();
    this.isRecognizing = false;
    console.log("Stopped recognition");
  }

  onResult(event, handler) {
    var interim_transcript = "";
    for (var i = event.resultIndex; i < event.results.length; ++i) {
      var result = event.results[i];
      if (result.isFinal) {
        this.transcript += result[0].transcript;
      } else {
        interim_transcript += result[0].transcript;
      }
    }
    console.log(interim_transcript);
    handler(interim_transcript);
  }
}

const btnRecognize = document.querySelector("#recognize");
const txtInterim = document.querySelector("#interim");
const txtMessage = document.querySelector("#message");
const recognizer = new Recognizer();
```

```
function showText(text) {
  txtInterim.value = text;
  txtMessage.value = recognizer.transcript;
}

function start() {
  txtInterim.value = "";
  txtMessage.value = "";
  recognizer.start(showText);
  btnRecognize.innerHTML = "Остановить запись";
}

function stop() {
  recognizer.stop();
  btnRecognize.innerHTML = "Начать запись";
}

btnRecognize.addEventListener("click", () => {
  if (!recognizer.isRecognizing) {
    start();
  } else {
    stop();
  }
});
</script>



          </div>
        </div>
  </div>
```

```
  </body>
def index(request):
    if request.method == 'POST':
        text = request.POST.get('text')
        name = request.POST.get('name')
        myobj = gTTS(text=text, lang='ru', slow=False)

        addAudioToDB(myobj, name, request)
        return redirect('index')
    if request.user.is_authenticated:
        return render(request, 'index.html')
    else:
        return redirect('login')
def addAudioToDB (obj, name, req):
    f_name = name + "_" +  req.user.username + ".mp3"
    full_fn = settings.MEDIA_ROOT +  "/" + name + "_" +  req.user.username + ".mp3"
    audio = text_to_speech_Data(audio_name=f_name, user=req.user)
    audio.save()
    obj.save(full_fn)
```

На сторінці кабінету користувачеві дається можливість прослухати озвучені тексти і завантажити їх (лістинг 3.5)

Лістинг 3.5.

Алгоритми особистого кабінету

```
<body>

<ul>
  <li><a href="{% url 'index' %}">Главная</a></li>
    <li  style="float:right" st><a href="{% url 'logout' %}">Выйти</a></li>
  <li class="usrName">Добро пожаловать, {{request.user}}</li>
```

```
</ul>

  <div class="container h-100">
                <div class="d-flex justify-content-center h-100">
                    <div class="user_card">
                <div class="d-flex justify-content-center">
                                    <h3 class="form-title">Озвученные тексты</h3>

                </div>
               <div class="d-flex justify-content-center">
                                    <h3 class="form-title">Нажмите на название для загрузки</h3>

                </div>

               {% if audio_list %}

    {% for a in audio_list %}

      <h4 align="center"> <a href="{% url 'download' a.audio_name %}">{{a.audio_name}}</a></h4><br>

    {% endfor %}

  {% endif %}
```

```
            </div>
        </div>
    </div>
    </body>
def cabinetPage(request):
    all_audio = text_to_speech_Data.objects.filter(user = request.user)
    return render(request, 'cabinetPage.html', {'audio_list':all_audio})
def download_file(request, fileName):
    # fill these variables with real values
    fl_path = settings.MEDIA_ROOT + "/"
    filename = fileName
    file = fl_path + filename
    fl = open(file, 'rb')
    mime_type, _ = mimetypes.guess_type(file)
    response = HttpResponse(fl, content_type=mime_type)
    response['Content-Disposition'] = "attachment; filename=%s" % filename
    return response
```

ДОДАТОК Б

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

“Розробка програмного забезпечення для транскрибування та синтезу мови за допомогою HTML, CSS, JS, Python.”

Виконав студент 4 курсу
групи ПД-44
Демиденко Нікіта Олександрович
Керівник роботи
Негоденко Олена Василівна

Київ – 2021

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

У сучасному світі надважливим є пошук рішень для спрощення та автоматизації усіх процесів. Актуальність роботи у розробці рішення для пришвидшення та спрощення роботи з мовою та текстом, процесу транскрибування та синтезу мови.

- **Мета роботи** - підвищення ефективності транскрибування та синтезу мови шляхом розробки програмного забезпечення за допомогою HTML, CSS, JS та Python
- **Об'єкт дослідження** - підвищення ефективності перетворення інформації для транскрибуванні та синтезу мови
- **Предмет дослідження** - методи та засоби забезпечення ефективного перетворення інформації для синтезу і розпінавання мови

1

ПОРІВНЯННЯ АНАЛОГІВ ЗАСТОСУНКУ

НАЗВА ЗАСТОСУНКУ	ОФОРМЛЕННЯ	МОЖЛИВОСТІ	МОВИ ТА Ч/Ж ГОЛОСІВ	СИНТЕЗ МОВИ ТА ТАНСКРИБУВАННЯ	ЛІМІТ СИМВОЛІВ
МІЙ ЗАСТОСУНОК	Інтуїтивно зрозуміле, просте та лаконічне, російською мовою	Реєстрація та кабінет, збереження аудіо, можливість дати назву доріжці, авторозпізнавання мови	Жіночий голос - Рос, Англ, та інші іноземні мови	Обидві можливості	Немає ліміту + динамічне оновлення тексту при надиктовуванні

1

НАЗВА ЗАСТОСУНКУ	ОФОРМЛЕННЯ	МОЖЛИВОСТІ	МОВИ ТА Ч/Ж ГОЛОСІВ	СИНТЕЗ МОВИ ТА ТАНСКРИБУВАННЯ	ЛІМІТ СИМВОЛІВ
cybermova.com (http://78.47.9.109/tts/)	Голий HTML	Тільки озвучування	ЖІН. - УКР, РОС	Синтез мови	300
http://www.fromtexttospeech.com/	Приємне оформлення	Вибір швидкості мови, голосу	3 Жін, 2 Чол - 7 мов без Укр	Синтез мови	50 000

ПРОШУ ЗВЕРНУТИ УВАГУ! Я ОБРАВ ДЛЯ ПОРІВНЯННЯ СЕРВІСИ УКРАЇНСЬКОГО ВИРОБНИЦТВА ТА БЛИЗЬКОГО ПО ВАРТОСТІ СПЕКТРУ

1

АНАЛОГИ

Мій додаток краще за аналоги тим що він простий у використанні, підтримує динамічну зміну тексту у надиктовуванні, стабільний, не має ліміту на символи, дозволяє зареєструватися та зберігати аудіофайли на локальній базі даних, має лаконічний дизайн, є безкоштовним для користувача та має потенціал, можливість розширитись у функціоналі та бути застосованим у екосистемах компаній та інших сервісів.

Мій застосунок гірше ніж інші тим що у ньому нема підтримки української мови, він поки що активується на локальному пристрою, а не розміщений на сайті через свою специфіку.Також, немає можливості обрати швидкість та голос для озвучування тексту.

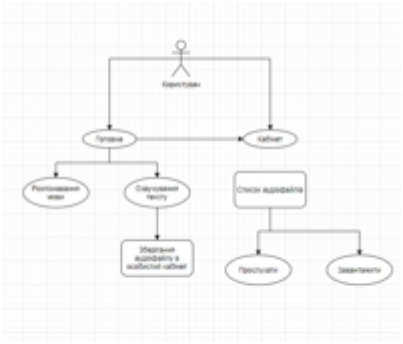
ТЕХНІЧНІ ЗАВДАННЯ

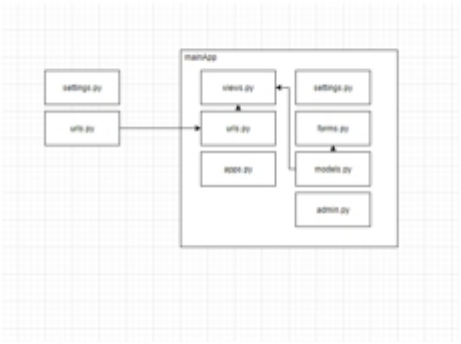
- *Проаналізувати предметну область*
- *Обрати засоби реалізації*
- *Розробити програмне забезпечення*

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

- *HTML (англ., “Hyper Text Markup Language”) - мова розмітки гіпертексту*
- *CSS (англ., “Cascading Style Sheets”) - каскадні таблиці стилів*
- *JavaScript - мова програмування*
- *Python + фреймворк Django - мова програмування*

МЕТОДИ ТА КЛАСИ ПРОГРАМИ





1

1

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

Сама робота та її результати були опубліковані на конференції “Науково-технічна конференція «Застосування програмного забезпечення в ІКТ»” від 10 лютого 2021 року. Демиденко Н.О. “ВЕБ-САЙТ ДЛЯ ТРАНСКРИБАЦІЇ ТА СИНТЕЗУ МОВИ” / Демиденко Н.О. // Застосування програмного забезпечення в ІКТ: Матеріали науково-технічної конференції «Застосування програмного забезпечення в ІКТ». Збірник тез. 10.02.2021, ДУТ, м. Київ — К.: ДУТ, 2021. — С. 58. http://www.dut.edu.ua/uploads/n_9058_51926054.pdf

1

ВИСНОВКИ

1. Проведено дослідження, котрі обгрунтовують актуальність роботи та наукову новизну. Проведений аналіз понять синтезу та розпізнавання мови, розробки веб сайтів.
2. Визначено, що оптимальним рішенням є колаборація із стандартного веб-стеку для фронтенду HTML, CSS, JS та мови програмування Python у зв'язці з фреймворком Django для серверної частини.
3. Розроблений застосунок буде ефективним для користувачів, які мають необхідність працювати з великими обсягами текстових даних, які необхідно перевести в формат звуку, або навпаки, для користувачів, які мають необхідні в переведенні усної мови в текстовий формат.

Перспективи подальших досліджень - можливість покращити сервіс, інтегрувати його у сторонні сервіси та компанії.

ДЯКУЮ ЗА УВАГУ!