PROJECT REPORT ON

"Srishti Speech Recognition Internal System Host Technical Intelligence"

Submitted in the partial fulfillment of the requirement for the award of a degree of

Bachelor of Science in Computer Science (Honours)

SAMMILANI MAHAVIDYALAYA

Under the Guidance of

PROF. ARINDAM SAHA

Submitted By

MR. ARNAB MONDAL

(ROLL NO. 203513-21-0149, REG. NO. 513-1112-0886-20)

MR. ASHISH MANDI

(ROLL NO. 203513-21-0154, REG. NO. 513-1113-0887-20)

MR. TASIR AHHAMED LASKAR

(ROLL NO. 203513-21-0127, REG. NO. 513-1115-0263-20)

DEPARTMENT OF COMPUTER SCIENCE

SAMMILANI MAHAVIDYALYA

UNDER

UNIVERSITY OF CALCUTTA

THE UNIVERSITY OF CALCUTTA. KOLKATA, WEST BENGAL, INDIA

DECLARATION

I hereby declare that this project is based on my original work except for citations and quotations which have been duly acknowledged. I also describe that it has not been previously and concurrently submitted for any other degree or award at any university or any other institution.

Name of the Supervisor: **Mr. Arindam Saha**

Name of the College: **Sammilani Mahavidyalaya**

Signature:

MR. ARNAB MONDAL

ROLL NO. 203513-21-0149, REG. NO. 513-1112-0886-20

MR. ASHISH MANDI

ROLL NO. 203513-21-0154, REG. NO. 513-1113-0887-20

MR. TASIR AHHAMED LASKAR

ROLL NO. 203513-21-0127, REG. NO. 513-1115-0263-20

Sammilani Mahavidyalaya, Baghajatin.

University of Calcutta

SAMMILANI MAHAVIDYALAYA

SAMMILANI MAHAVIDYALAYA, BAGHAJATIN, KOLKATA

Department of Computer Science

CERTIFICATE OF APPROVAL

This is to certify that the dissertation is the record of the Final Year Project entitled **"Srishti Speech Recognition Internal System Host Technical Intelligence"** undergone at Sammilani Mahavidyalaya, Baghajatin carried out by Arnab Mondal(Roll No.: 203513-21-0149), Ashish Mandi(Roll No.: 203513-21-0154), Tasir Ahhamed Laskar(Roll No.: 203513-21-0127) of the Department of Computer Science, Sammilani Mahavidyalaya, Baghajatin for the partial fulfillment of the award of the degree of Bachelor of Science (Session 2021-23) by Sammilani Mahavidyalaya, University of Calcutta in the year 2023 under my supervision and guidance. To the best of my knowledge, the results embodied in this report, are original in nature and worthy of incorporation in the present version of the report for the B. Sc. program in Computer Science.

This report has not been submitted to any other university or institution for the award of any degree.

Guide / Supervisor

HEAD OF DEPARTMENT
COMPUTER SCIENCE
SAMMILANI
MAHAVIDYALAYA

MR. ARINDAM SAHA

ACKNOWLEDGMENT

First of all, we Arnab Mondal, Ashish Mandi, Tasir Ahhamed Laskar of B. Sc. Computer Science 6th Semester would like to express our profound sense of gratitude towards our Project Guide: **Mr. Arindam Saha** (**State Added College Teacher**), Department of Computer Science for his ability to guide, support, and encouragement during the course of our final year project: **"Srishti Speech Recognition Internal System Host Technical Intelligence".** This project work was undertaken in partial fulfillment of the requirement for the award of the degree of **Bachelor of Science** from **Sammilani Mahavidyalaya, Kolkata.**

We are deeply indebted to our project guide for giving us this opportunity to work on this project and for his kind help and support to develop an understanding of the subject and for making a clear knowledge by providing the necessary insight. His readiness for consultations at all times, his educative comments, his concern, his concrete support, and his assistance even with practical things have been invaluable.

Lastly, we would like to thank the entire faculty of Computer Science at our college for cultivating a healthy and creative environment to work on the project.

Date: _____

Place: _____

MR. ARNAB MONDAL

UNIVERSITY ROLL NO. 203513-21-0149

UNIVERSITY REG. NO. 513-1112-0886-20

Date: _____

Place: _____

MR. ASHISH MANDI

UNIVERSITY ROLL NO. 203513-21-0154

UNIVERSITY REG. NO. 513-1113-0887-20

Date: _____

Place: _____

MR. TASIR AHHAMED LASKAR

UNIVERSITY ROLL NO. 203513-21-0127

UNIVERSITY REG. NO. 513-1115-0263-20

TABLE OF CONTENT

ABSTRACT

This project is mainly concerned with the field of Artificial Intelligence Speech Recognition Systems and neural networking. By utilizing the Srishti AI Python library, developers can build intelligent applications that incorporate natural language processing, computer vision, and machine learning capabilities. The library simplifies the development process by providing pre-built modules and utilities, allowing developers to focus on implementing AI-specific functionality tailored to their application's requirements. This website also consists of hyperlinks such as Home, Login, About, Register, and Download. Common people can now get to know each and everything about the “SRISHTI” and can register themselves to be part of it. Our internal system hosts a range of powerful features designed to enhance your speech recognition experience. A different section is provided for contacting the organizers for any queries. With advanced technical intelligence, it offers seamless transcription capabilities across multiple languages and accents. The system's real-time feedback and intelligent speaker recognition make it a versatile solution for various applications.

Finally, this project was concluded with the scope for further work which can be done to achieve better results.

CHAPTER -1: INTRODUCTION

1.1 Introduction of SRISHTI

The "SRISHTI" interface design is an area that is giving detailed information about Speech Recognition System and its operation and hidden talent. As the world develops digitally and technically the organization of any institution becomes more reliant on their own software for furthering their mission. SRISHTI interface is feasible and crucial for usability. efficiency, correctness search tells you about how this interface will help to propagate neural networking, machine learning, artificial intelligence, Python modules, programs, and further search. Srishti AI is not just a speech recognition system; it's a productivity powerhouse. Whether you're a professional writer, a busy executive, or a student preparing for exams, Srishti AI can significantly boost your efficiency and effectiveness. Key features of Srishti AI: Highly accurate speech recognition, Support for multiple languages, Real-time transcription and translation, Customizable vocabulary and voice commands, Intelligent voice assistant integration, and Secure and private data handling. SRISHTI improves their knowledge. Therefore, it becomes a great platform for students in college, schools, and IT professors to learn and grow with fun. We are committed to continuously improving Srishti AI by leveraging the latest advancements in machine learning and artificial intelligence. This means you can expect regular updates and enhancements to further enhance your speech recognition experience.

1.2 Motivation

Speech recognition technology is already a part of our everyday lives like searching for songs clearing quires and doubt, but for now, it is still limited to relatively simple commands. As technology advances, researchers will be able to create more intelligent systems that understand conversational speech (remember the robot job interviewers?). One day, you will be able to talk to your computer the way you would talk to any human being, and it will be able to transmit reasoned responses back to you. All this will be made possible by signal processing technologies like **SRISHTI**. The number of specialists needed in this field is growing, and many companies are looking for talented people who want to be a part of it. Processing, interpreting, and understanding a speech signal is the key to many powerful new technologies and methods of communication. As per current upcoming needs, speech recognition technology will be a fast-growing (and world-changing) subset of signal processing for years to come.

1.3 Scope & Background of the Work

The proposed website of **SRISHTI** is a web application that gives opportunities to students. This system manages complete information in a single interface and dictation paired with transcriptions software means reduced transcription costs and a much easier workflow. Voice recognition software can be used by any industry or hospital. Have more time to focus on your patients and clients. Work from anywhere with a secure internet connection. The range of this application is very wide and diversified from industry to home applications. The intentions of the system are to reduce the complexity of specially-abled persons. Besides this, it is accessible to local people who find difficulties using smartphones, laptops, etc. In addition to being precise and accurate, speech recognition technology can detect accents and spell words accurately.

1.3.1 Background of The Work

A review of literature is the section of any research study which provides a critical view and detailed overview of all the magnitudes of the specific subject of study, which has already been exposed over different gaps. Computing powers and artificial intelligence are largely behind these spaces. With a massive amount of speech recognition has hit an inflection point where its capabilities are roughly on par with humans.

The first speech recognition systems were focused on numbers, not words in the late 1950 and 1960s. In 1952 Bell Laboratories designed the “Audrey” system which could recognize a single voice speaking digit allowed. In 2011, Apple launched ‘Siri’ which was similar to Google Voice Search, the early part of the decade saw an explosion of some other voice recognition apps and software. Today, some of the largest tech components are competing to herald the speech accuracy title.

This technology voice application is now relatively inexpensive and powerful, With advancements in AI and the increasing amounts of speech data that can be mined, it is very possible that voice becomes the next dominant interface.

1.4 OBJECTIVE

1. This system helps to know about the SRISHTI interface is feasible and crucial for usability. efficiency, correctness search tells me about how this interface will help to propagate neural networking, machine learning, artificial intelligence, and Python modules and Python programs and further search.

2. Decreased billable hours on searching.

3. Improved productivity and a more streamlined workflow for the entire team More work is done throughout the work week so working after hours and on weekends becomes a thing of the past

4. No more missing important events because you have to work

5. It provides support information for any queries in the mind of a beginner who has just started any new topic.

CHAPTER 2: WEB METHODOLOGY

2.1 Introduction of Web Methodology

Research indicates that many traditional AI development methodologies are based on outmoded concepts dating back to the 1970s. These methodologies are being utilized to develop websites and, not surprisingly, they are limited since they were never intended to be used for this purpose. Before moving on to put forward a methodology for speech recognition, it is worth considering traditional AI methodologies and their applicability to this process. Site Map 2.1 summarizes some of these methodologies, providing a brief explanation of their Text to Speech Recognition. When it comes to our interactions with machines, things have gotten a lot more complicated. We've gone from large mechanical buttons to touchscreens. However, hardware isn't the only thing that's changing. Throughout the history of computers, text has been the primary method of input. But thanks to developments in NLP and ML (Machine Learning), Data Science, and Artificial Intelligence we now have the means to use speech as a medium for interacting with our gadgets in the near future.

This project is an activity that helps us to improve our learning, planning, and critical thinking ability. Here in **SRISHTI** is an advanced speech recognition system that utilizes cutting-edge technology to provide accurate and intelligent speech recognition capabilities. Our internal system hosts a range of powerful features designed to enhance your speech recognition experience. The website consists of Home, About, Register, FAQ, Contact Us, and Download. Figure 2.3 It is the diagrammatic representation of the process after downloading the software from the website https://webpage-srishti.onrender.com. Users have to fill up the Registration Form and then proceed to the Face Recognition process by entering the strong security password and finally, the SRISHTI interface opens.

2.2 Site Map:

- Figure 2. Shows the basic structure of the website.

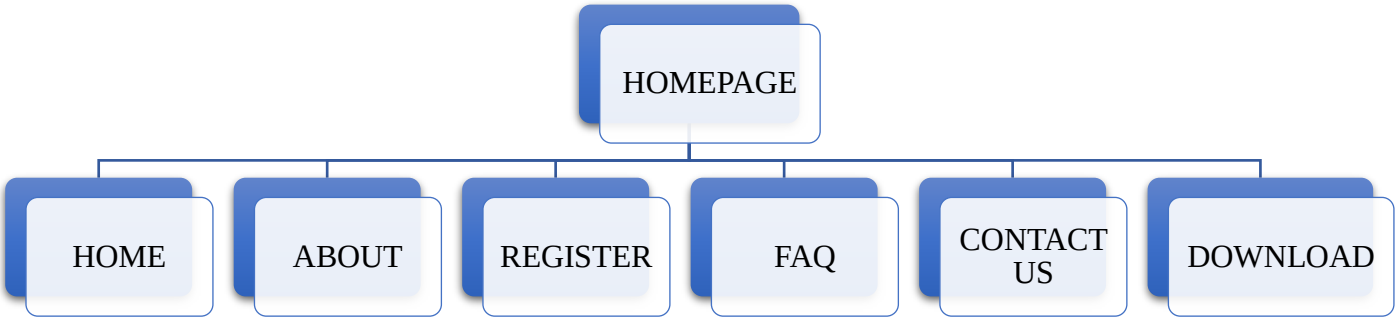


Figure 2.1 Site Map of the Homepage

2.3 Block Diagram:

Figure 2.3 shows the diagrammatical representation of the website which finds out the several relationships.

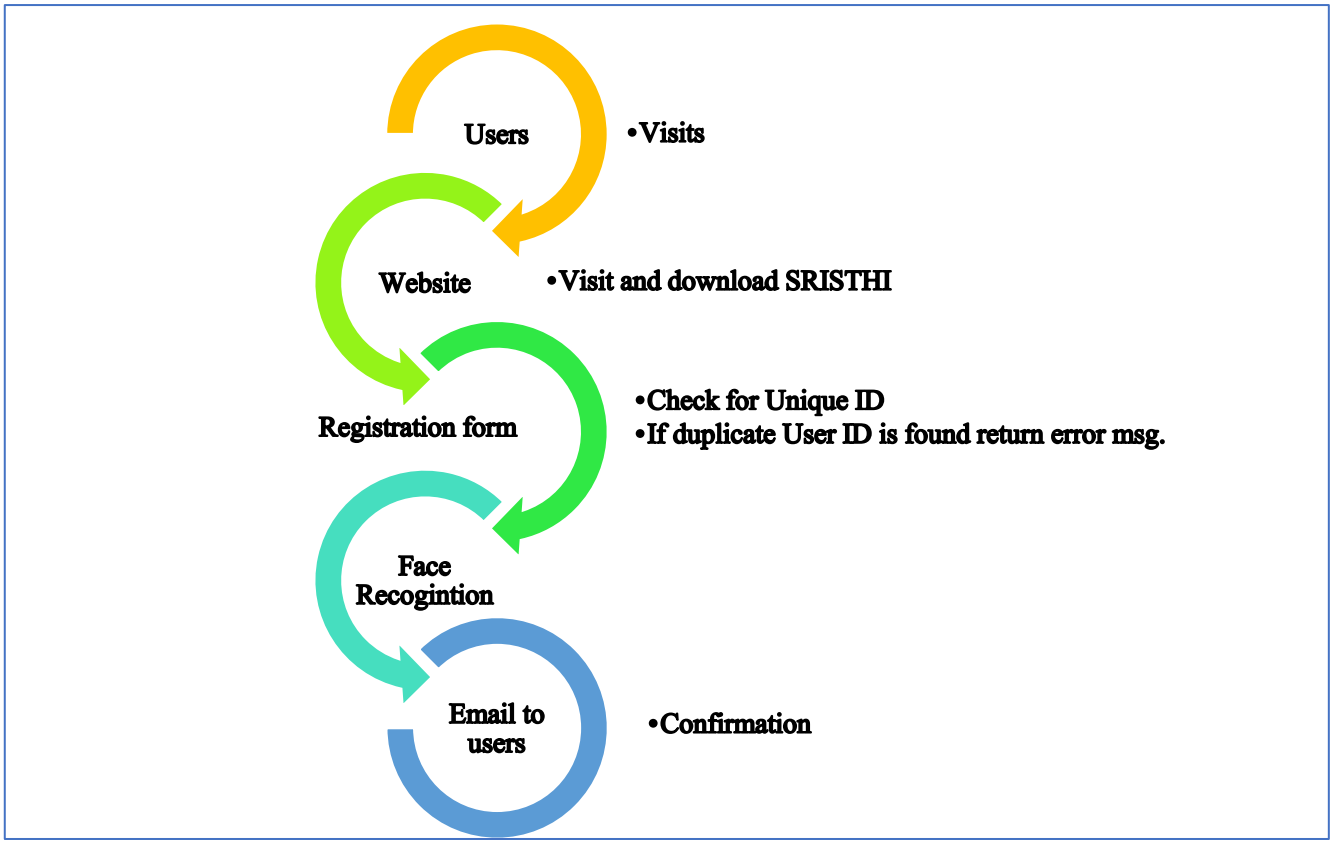


Figure 2.3 Block Diagram of SRISHTI Website Registration Process

2.4 Data Flow Diagram (DFD):

- The DFD also known as a bubble chart is a hierarchical graphical model of a system that can be used to represent a system in terms of the input data of the system, various processing carried out on the date, and the output data generated by the system.
- In the DFD terminology, each function is called a process or bubble that consumes some input data and produces some output data.
- DFD model represents the data flow aspects.
- It does not show the sequence of execution of the different functions and the conditions based on the function may or, may not be executed.

Figure 2.4 shows Level 0 DFD (Context Diagram) of SRISTHI.

- The top-level DFD
- This is the abstract(simplest) representation of the system (highest level).
- It represents the entire system (SRISHTI) as a single bubble.

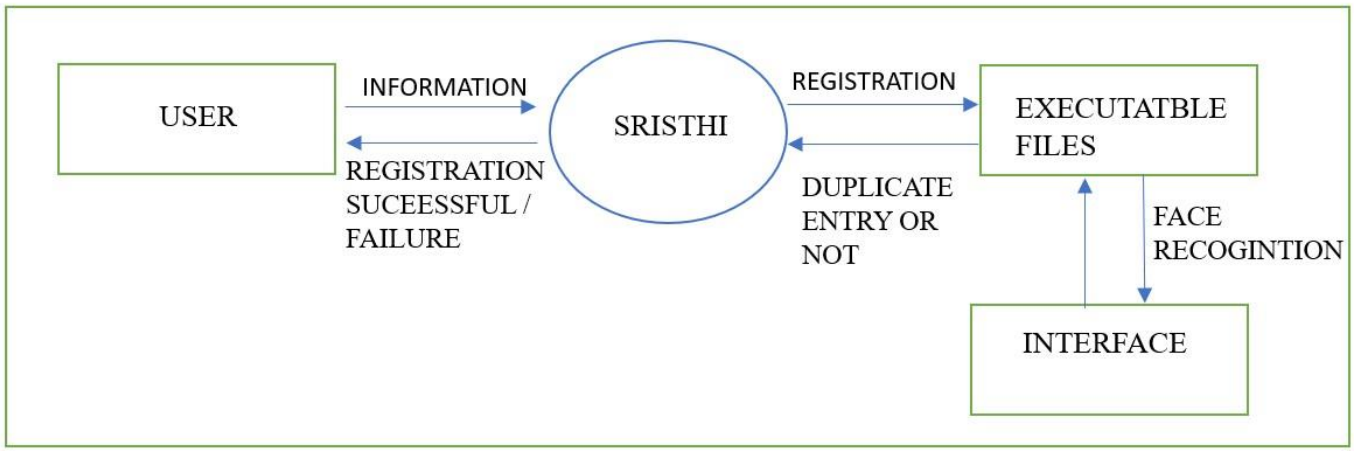


Figure 2.4: Level- 0 DFD

CHAPTER 3: IMPLEMENTATION

3.1 About SRISHTI

SRISHTI (Speech Recognition Internal System Host Technical Intelligence) is an advanced AI system designed to provide speech recognition and natural language understanding capabilities. It combines various technologies like audio preprocessing, speech recognition algorithms, and natural language processing (NLP) to enable seamless communication between users and the AI system. The primary goal of SRISHTI is to understand and interpret spoken language, allowing users to interact with the system using their voice. By capturing audio input from users, SRISHTI processes the speech through sophisticated algorithms that remove noise, enhance speech quality, and convert the audio into text using speech recognition techniques.

Once the audio is transformed into text, SRISHTI employs NLP algorithms to analyze and extract the intent and important keywords from the user's speech. This enables the system to understand the user's commands, queries, or requests. The extracted intent and keywords are then used to determine the type of task or action required. Based on the determined task, SRISHTI can perform various actions such as retrieving information from data sources or APIs, processing fetched data using algorithms, generating meaningful insights or responses, and executing appropriate actions based on user commands.

SRISHTI also incorporates error-handling mechanisms and clarification prompts to address any misunderstandings or uncertainties that may arise during the conversation. Additionally, the system can engage in voice-based interaction with users to gather additional information and refine its understanding.To improve its performance and adaptability, SRISHTI continuously updates its knowledge base or models based on user interactions and feedback. This ensures that the system learns from past interactions and can provide more accurate responses over time.

With its speech recognition and natural language understanding capabilities, SRISHTI offers a user-friendly and efficient means of interacting with AI systems. It finds applications in various domains, such as virtual assistants, voice-controlled systems, customer support, and more. The flexibility, accuracy, and adaptability of SRISHTI make it a valuable tool for enhancing human-AI interactions and enabling a seamless speech-driven user experience.

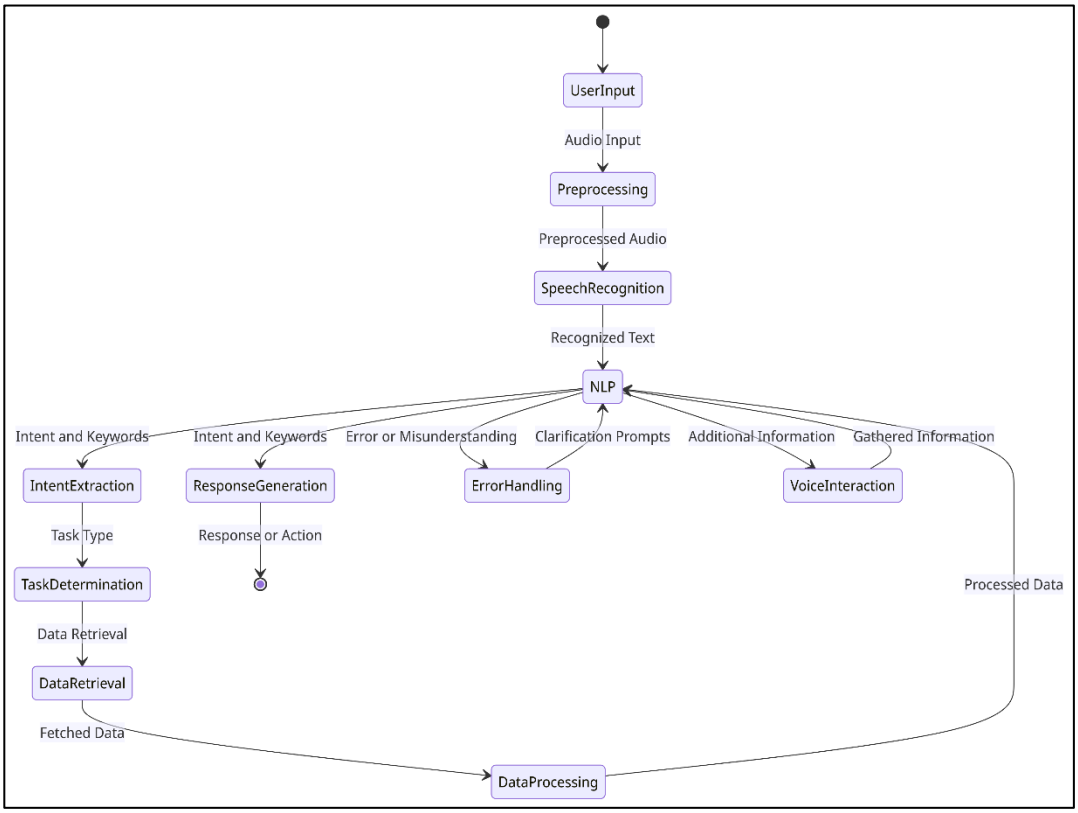
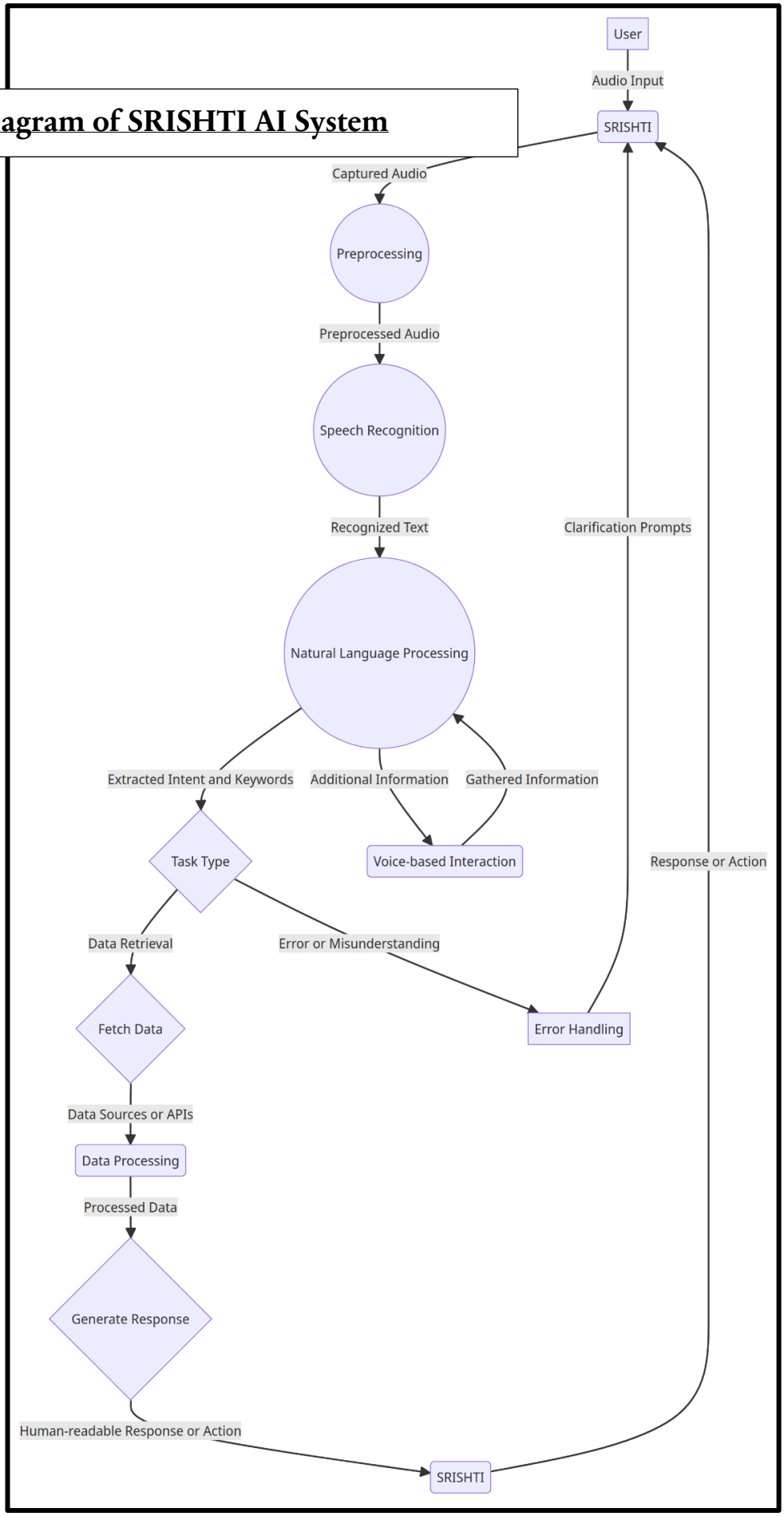
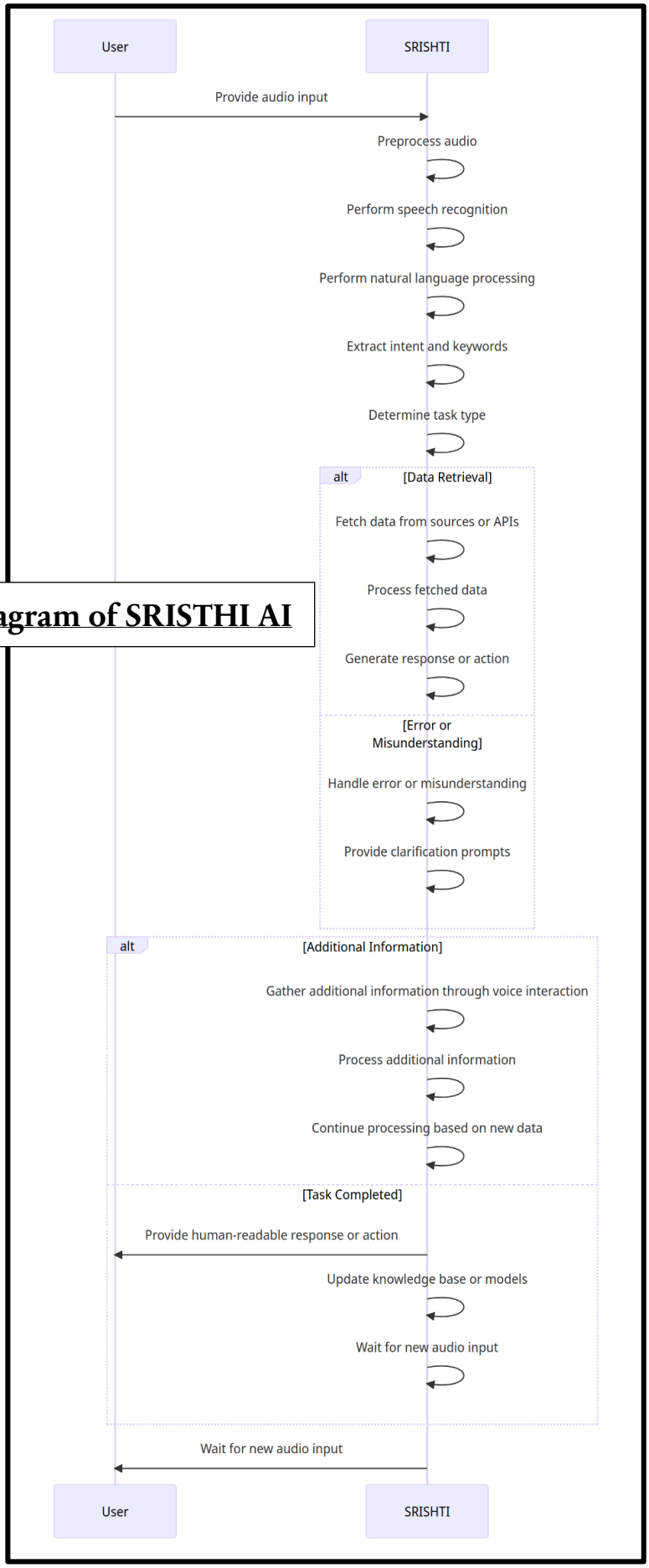


Figure 3.1.1: State diagram

3.2 Block Diagram of SRISHTI AI System



3.3 Sequence Diagram of SRISTHI AI



3.4 ALGORITHM FOR SRISHTI AI SYSTEM

1. Initialize the SRISHTI AI system.
2. Load necessary libraries and dependencies for speech recognition and natural language processing.
3. Set up the audio input device for capturing speech.
4. Continuously listen for user input by capturing audio.
5. Preprocess the captured audio to remove noise and enhance speech quality.
6. Convert the pre-processed audio into text using a speech recognition algorithm.
7. Perform natural language processing on the recognized text to extract the user's intent and important keywords.
8. Determine the type of task or query based on the user's intent and keywords.
9. Execute the appropriate action or provide a response based on the determined task type.
10. If the task requires retrieving information, connect to relevant data sources or APIs to fetch the required data.
11. Apply any necessary algorithms or techniques to process the fetched data and generate meaningful insights or responses.
12. Generate a human-readable response or perform the requested action based on the processed data.
13. If the task is completed, go back to step 4 to listen for new user input. Otherwise, continue to step 14.
14. If there is an error or the user input is not understood, provide appropriate error handling or clarification prompts.
15. Allow for voice-based interaction to gather additional information or clarify user intent, if necessary.
16. Update the system's knowledge base or models based on user interactions and feedback to improve future responses.
17. Continue to loop through steps 4 to 16, providing responses and interacting with the user as needed.
18. Terminate the SRISHTI AI system when the interaction with the user ends or when explicitly instructed to stop.

3.5 We use Python for developing our SRISHTI AI System

Solution: Python is commonly used for AI projects like SRISHTI due to several reasons:

- Simplicity and Readability: Python has a clean and easy-to-read syntax, making it beginner-friendly and highly readable. This makes it easier for developers to understand and maintain the codebase.

- Extensive Libraries and Frameworks: Python offers a wide range of libraries and frameworks specifically designed for AI and machine learning tasks. Libraries like TensorFlow, PyTorch, and scikit-learn provide powerful tools for building and training AI models.

- Vibrant Ecosystem: Python has a vibrant and active community that actively develops and maintains various AI-related libraries, tools, and resources. This means that developers have access to a wealth of resources, documentation, and community support.

- Interoperability: Python can easily integrate with other programming languages and technologies. This flexibility allows developers to leverage existing systems, libraries, and APIs, making it easier to connect SRISHTI with various data sources or APIs for fetching information.

- Data Processing Capabilities: Python offers robust data processing and manipulation capabilities through libraries like NumPy and pandas. These libraries provide efficient data structures and operations, which are crucial for tasks like preprocessing audio, and text, and working with structured data.

- Prototyping and Rapid Development: Python's simplicity and extensive libraries make it an ideal choice for prototyping and rapid development of AI systems like SRISHTI. It allows developers to quickly test and iterate on ideas, making the development process more efficient.

- Community and Industry Support: Python is widely adopted in both the AI research community and the industry. Many AI-related conferences, workshops, and competitions use Python as the primary language. This support and recognition make Python a natural choice for AI projects.

While Python is not the only language used for AI, its combination of simplicity, extensive libraries, and a strong community make it a popular and effective choice for building AI systems like SRISHTI.

3.6 We use Neural Network for developing this project

Solution: Using neural networks for the SRISHTI project offers several advantages:

- Pattern Recognition: Neural networks are well-suited for pattern recognition tasks, such as speech recognition and natural language understanding. They can learn complex patterns and relationships from large amounts of data, enabling accurate and robust recognition of speech patterns and semantic structures.

- Non-linear Mapping: Neural networks are capable of learning non-linear mappings between input (audio signals or textual data) and output (intent, keywords, or actions). This flexibility allows SRISHTI to capture the intricate relationships and nuances present in human speech and language.

- Feature Extraction: Neural networks can automatically learn and extract relevant features from the input data, reducing the need for manual feature engineering. This is particularly useful in speech recognition, where identifying relevant acoustic features from audio signals can be challenging.

- Adaptability: Neural networks can adapt and learn from new data, making them suitable for dynamic and evolving systems like SRISHTI. As the AI system interacts with users and receives feedback, the neural network models can be updated and fine-tuned to improve performance and adapt to changing user preferences and language patterns.

- Parallel Processing: Neural networks can take advantage of parallel processing capabilities offered by modern hardware, such as GPUs (Graphics Processing Units). This enables efficient training and inference, allowing SRISHTI to process audio inputs and generate responses in real time.

- Scalability: Neural networks can scale to handle large datasets and complex tasks. As SRISHTI grows and handles a higher volume of user interactions, neural networks can accommodate the increasing demands for processing power and data complexity.

- State-of-the-Art Performance: Neural networks, especially deep learning architectures, have achieved state-of-the-art performance in various AI tasks, including speech recognition and natural language processing. Leveraging neural networks in the SRISHTI project ensures that the AI system can provide accurate and reliable results to users.

It is worth noting that while neural networks offer significant benefits, their design, training, and optimization require expertise and computational resources. However, the advancements in deep learning frameworks and the availability of pre-trained models make it more accessible for developers to leverage neural networks in AI projects like SRISHTI.

CHAPTER 4: DESIGN OF THE PROJECT

4.1 Language Being Used for This Project

PYTHON

Python is a high-level, general-purpose, and very popular programming language. Python programming language (latest Python 3) is being used in web development, Machine Learning applications, along with all cutting-edge technology in Software Industry.

Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is a huge collection of standard libraries which can be used for the following:

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like OpenCV, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia
- Scientific computing
- Text processing and many more.

4.2 For the front-end and back-end, we use the following programming language.

SRISHTI: GUI (Graphics User Interface)

GUI stands for Graphical User Interface. It refers to an interface that allows one to interact with electronic devices like computers and tablets through graphic elements. It uses icons, menus, and other graphical representations to display information, as opposed to text-based commands. The graphic elements enable users to give commands to the computer and select functions by using a mouse or other input devices.

The programs which run under a GUI have a specific set of graphic elements so that after learning a specific interface, a user can use these programs without learning new commands

Basic Components of a GUI

- **Pointer:** It is a symbol that appears on the display screen. It can be moved to select commands and objects.
- **Pointing device:** It allows you to move the pointer and select objects on the screen, e.g., mouse or trackball.
- **Icons:** These refer to small images on the display screen that represent commands, files, windows, etc. Using a pointer and pointing device, you can execute these commands.
- **Desktop:** It is the display screen that contains the icons.

HTML

HTML or **Hypertext Markup Language** is the standard markup language to create web pages. HTML is written in the form of HTML elements consisting of tags enclosed in angle brackets (hike <html>), HTML describes the structure of a website semantically along with cues for presentation, making it a markup language rather than a programming language.

HTML elements form the building block of all websites. HTML allows images and objects to be embedded and can be used in interactive forms. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, quotes, and other items. It can embed scripts written in languages such as JavaScript which affect the behavior of an HTML webpage.

CASCADING STYLE SHEETS (CSS):

It is a style sheet language used for describing the look and formatting of a document written in markup language. CSS is a cornerstone specification of the web and almost all web pages use CSS style sheets to describe their presentation. CSS is designed primarily to enable the separation of document content from document presentation, including elements such as layout, colors, and fonts. It is recommended to use CS because the HTML attributes are deprecated. So, for making HTML pages compatible with future browsers, it is good to use CSS in HTML pages. Also, CSS provides more detailed attributes than plain HTML to define the look and feel of the website.

4.3 Database Being Used for This Project

JSON_DATABASE

JSON is a human-readable text format that facilitates data interchange between different programming languages.

Here are some quick points about JSON:

JSON is popular. JSON (JavaScript Object Notation) is quite possibly the most widely used data format for data interchange on the web. It has likely surpassed XML (which is used in AJAX applications) as the most common format used for asynchronous browser/server communication.

JSON is a human and machine-readable format. In other words, a JSON document is structured in a way that can easily be read by a computer program, while a human can usually quickly scan a JSON file and understand the data it contains.

JSON is based on a subset of JavaScript. JSON was inspired by the object literals of JavaScript (also known as ECMAScript). However, despite this, JSON is language-agnostic. It can facilitate data interchange between most, if not all programming languages. In fact, JSON uses common programming conventions, which makes it familiar to most programmers, regardless of their chosen language/s.

RethinkDB	Rethink DB is the first open-source, scalable JSON database built from the ground up for the real-time web. Rethink DB is designed specifically to push data to applications in real-time.

PostgreSQL

PostgreSQL, often referred to as Postgres, is an advanced open-source relational database management system (RDBMS). It is known for its robustness, scalability, and comprehensive feature set. PostgreSQL is designed to handle a wide range of workloads, from small personal projects to large-scale enterprise applications.

Here are some key features and characteristics of PostgreSQL:

- Relational database: PostgreSQL follows the relational model, allowing you to define tables, establish relationships between them using primary and foreign keys, and perform complex queries using SQL (Structured Query Language).

- Open-source: PostgreSQL is released under an open-source license, which means it is free to use, modify, and distribute. This fosters a strong community of developers and contributors who continuously improve and enhance the database.

- ACID compliance: PostgreSQL ensures ACID (Atomicity, Consistency, Isolation, Durability) compliance, providing transactional integrity and data consistency. This means that transactions are reliably processed and can be rolled back if needed, ensuring data integrity.

- Extensibility: PostgreSQL allows you to extend its functionality through the use of custom data types, operators, functions, and procedural languages. This extensibility enables developers to create complex data models and implement custom business logic within the database.

- Advanced features: PostgreSQL offers a wide range of advanced features, including support for JSON and JSONB data types, full-text search, geospatial data, concurrent indexing, and numerous indexing options for optimizing query performance. It also provides support for complex data types like arrays, stores (key-value pairs), and user-defined types.

- Scalability and replication: PostgreSQL supports various replication methods, including streaming replication, logical replication, and synchronous replication. These features allow you to distribute data across multiple servers, ensuring high availability and scalability.

- Security: PostgreSQL provides robust security features, including authentication methods, SSL encryption for secure connections, access control using roles and privileges, and support for row-level security, allowing fine-grained control over data access.

- Cross-platform: PostgreSQL is available for multiple operating systems, including Linux, Windows, macOS, and various Unix-like systems. This cross-platform compatibility makes it versatile and suitable for different deployment environments.

PostgreSQL is widely adopted and used by organizations of all sizes, ranging from startups to large enterprises. Its community-driven development, combined with its comprehensive feature set, makes it a popular choice for applications that require a reliable, scalable, and feature-rich relational database system.

4.4 Resources we use for this project

1.1.1 SOFTWARE USED

- Operating System: Windows
- Python PyCharm-Community
- Microsoft Word 2023
- Google Chrome and Brave

1.1.2 Hardware Used

- CPU: 11th Generation Intel® Core™ i5 Processors
- Ram: 8GB
- Hard Disk Memory: 1TB

CHAPTER 5: ANNEXURE

5.1 ANNEXURE I:

Python deploying code:

```
from flask import Flask, render_template, request, redirect, session, g, jsonify,
url_for, flash, get_flashed_messages
from github import Github
import psycopg2
import base64
from url import generate_hash, user_hash
from sende_mail_automation import send_mail, dev_mail, delete_mail
import json
from functools import wraps
import requests

app = Flask(__name__)
app.secret_key = 'your_secret_key'
com_key = "SRIshtiAAT2023"

# DATABASE_URL =
'postgres://srishti_database_ai_user:JaYaL1A92lAp0ikj0RxGjgKihQ3etVWj@dpg-
cic47a95rnuk9qb0sbc0-a/srishti_database_ai'
DATABASE_URL =
'postgres://srishti_database_ai_user:JaYaL1A92lAp0ikj0RxGjgKihQ3etVWj@dpg-
cic47a95rnuk9qb0sbc0-a.oregon-postgres.render.com/srishti_database_ai'

url = generate_hash()

# Connect to the PostgreSQL database
def get_db():
    db = getattr(g, '_database', None)
    if db is None:
        db = g._database = psycopg2.connect(DATABASE_URL)
    return db

# Create a table to store registered users if it doesn't exist
def create_table():
    conn = get_db()
    cursor = conn.cursor()
    cursor.execute('''CREATE TABLE IF NOT EXISTS dev_datas
                        (id SERIAL PRIMARY KEY,
                        hash_id_code TEXT,
                        username TEXT,
                        password TEXT,
                        email TEXT,
                        bio TEXT,
                        images BYTEA)''')
    cursor.execute('''CREATE TABLE IF NOT EXISTS person_database_sri
                       (id TEXT PRIMARY KEY,
                        name TEXT,
                        gender TEXT,
                        password TEXT,
                        email TEXT)''')
```

```
    cursor.execute('''CREATE TABLE IF NOT EXISTS API_KEY
                      (id SERIAL PRIMARY KEY,
                       api_key TEXT)''')
    cursor.execute('''CREATE TABLE IF NOT EXISTS test_user
                    (id SERIAL PRIMARY KEY,
                    name TEXT,
                    encoding BYTEA)''')
    conn.commit()

@app.teardown_appcontext
def close_connection(exception):
    db = getattr(g, '_database', None)
    if db is not None:
        db.close()

@app.route('/')
def index():
    conn = get_db()

    cursor1 = conn.cursor()
    cursor2 = conn.cursor()

    cursor1.execute('SELECT images FROM dev_datas')
    cursor2.execute('SELECT username FROM dev_datas')

    data = cursor1.fetchall()
    cursor1.close()
    users = cursor2.fetchall()
    cursor2.close()
    # print(users)
    image_user_mapping = {}  # Dictionary to store image-user mapping

    for i in range(len(users)):
        image_data = data[i][0]
        image_encoded = base64.b64encode(image_data).decode('utf-8')
        username = users[i][0]
        image_user_mapping[image_encoded] = username

    conn.close()

    return render_template('index.html', image_user_mapping=image_user_mapping,
url=url)

@app.route('/download')
def download():
    return redirect(f"/{url}/download/Srishti")

@app.route('/register')
def register():
    return redirect(f"/{url}/register/developer")

@app.route('/login')
def login():
    return redirect(f"/{url}/login/developer")

@app.route('/about')
```

```
def about():
    return redirect(f"/{url}/about/Srishti")

@app.route('/contact')
def contact():
    return redirect(f"/{url}/contact/us")

@app.route('/github')
def github():
    return redirect("https://github.com/ARNAB-BOTMAS/Srishti_project")


@app.route('/apis')
def apis_page():
    # Retrieve API keys from the API_KEY table
    try:
        conn = get_db()
        cursor = conn.cursor()
        cursor.execute('SELECT api_key FROM API_KEY')
        api_keys = [row[0] for row in cursor.fetchall()]
        cursor.close()
    except Exception as e:
        return f'Error retrieving API keys: {e}', 500

    # Create a dictionary containing the API keys
    api_keys_dict = {'api_keys': api_keys}

    # Convert the dictionary to JSON
    api_keys_json = json.dumps(api_keys_dict)

    # Set the response content type as JSON
    response = app.response_class(
        response=api_keys_json,
        status=200,
        mimetype='application/json'
    )

    return response


# Registration page
@app.route(f'/{url}/register/developer', methods=['GET', 'POST'])
def register_page():
    if request.method == 'POST':
        dev_key = request.form['dev_key']
        username = request.form['username']
        password = request.form['password']
        email = request.form['email']
        profile_picture = request.files['profile_picture']
        bio = request.form['bio']
        hash_id_code = user_hash(username, password)
        # print(hash_id_code)
        # print(profile_picture)  # Check if the object exists and contains data
        # print(profile_picture.filename)  # Check the filename to ensure it's not
empty
        if dev_key == com_key:
```

```
            if profile_picture.filename != '':
                try:
                    profile_picture_data = profile_picture.read()
                    # print(profile_picture_data)
                    conn = get_db()
                    cursor1 = conn.cursor()
                    cursor2 = conn.cursor()

                    cursor1.execute('INSERT INTO dev_datas (hash_id_code, username,
password, email, bio, images) VALUES (%s, %s, %s, %s, %s, %s)',
                                    (hash_id_code, username, password, email, bio,
psycopg2.Binary(profile_picture_data),))

                    cursor1.close()

                    cursor2.execute('INSERT INTO API_KEY (api_key) VALUES (%s)',
(hash_id_code,))
                    cursor2.close()
                    dev_mail(username, hash_id_code, email)
                    conn.commit()
                    return redirect('/login')
                except Exception as e:
                    return "Unable to try", 404
            else:
                return 'Image upload not successful', 404
        else:
            return redirect('/register')

    return render_template('register.html', url=url)


# Login page
@app.route(f'/{url}/login/developer', methods=['GET', 'POST'])
def login_page():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']

        # Check if the username and password match in the database
        conn = get_db()
        cursor = conn.cursor()
        cursor.execute('SELECT hash_id_code FROM dev_datas WHERE username=%s AND
password=%s',
                       (username, password))
        user = cursor.fetchone()
        hash_url = user[0]
        # print(user)
        if user:
            session['username'] = username
            return redirect(url_for('profile', hash_url=hash_url))

        return 'Invalid username or password'

    return render_template('login.html', url=url)

# Profile page
```

```
@app.route('/profile/<hash_url>')
def profile(hash_url):
    if 'username' in session:
        username = session['username']
        conn = get_db()
        cursor = conn.cursor()
        cursor1 = conn.cursor()
        cursor2 = conn.cursor()
        cursor3 = conn.cursor()
        cursor1.execute('SELECT email FROM dev_datas WHERE username=%s', (username,))
        email = cursor1.fetchone()[0]
        cursor1.close()
        cursor2.execute('SELECT images FROM dev_datas WHERE username=%s',
(username,))
        rows = cursor2.fetchall()
        valid_api_key = hash_url
        images = []
        for row in rows:
            image_data = base64.b64encode(row[0]).decode('utf-8')
            images.append(image_data)

        cursor.execute("SELECT * FROM person_database_sri")
        data = cursor.fetchall()
        user_database = []
        for row in data:
            user = {
                'id': row[0],
                'name': row[1],
                'gender': row[2],
                'password': row[3],
                'email': row[4]
            }
            user_database.append(user)
        cursor.close()
        cursor3.execute('SELECT * FROM test_user')
        face_data = cursor3.fetchall()
        face_id = []
        for row1 in face_data:
            face = {
                'id': row1[0],
                'face_id': row1[1],
            }
            face_id.append(face)
        cursor3.close()
    else:
        return redirect('/login')

    return render_template('profile.html', username=username, images=images,
email=email, user_database=user_database, valid_api_key=valid_api_key,
face_id=face_id)


# Logout
@app.route('/logout')
def logout():
    session.pop('username', None)
```

```
    return redirect('/')

@app.route(f'/{url}/download/Srishti')
def download_page():
    conn = get_db()
    cursor1 = conn.cursor()
    cursor2 = conn.cursor()
    cursor1.execute('SELECT images FROM dev_datas')
    cursor2.execute('SELECT username FROM dev_datas')
    data = cursor1.fetchall()
    cursor1.close()
    users = cursor2.fetchall()
    cursor2.close()

    image_user_mapping = {}  # Dictionary to store image-user mapping

    for i in range(len(users)):
        image_data = data[i][0]
        image_encoded = base64.b64encode(image_data).decode('utf-8')
        username = users[i][0]
        image_user_mapping[image_encoded] = username

    conn.close()

    return render_template('download.html', image_user_mapping=image_user_mapping,
url=url)
# Users data API

def authenticate_api_key(func):
    @wraps(func)
    def wrapper(*args, **kwargs):
        api_key = request.headers.get('X-API-Key')  # Assuming the API key is passed
in the request header

        # Retrieve API keys from the database or any other secure storage
        valid_api_keys = get_valid_api_keys()

        if api_key in valid_api_keys:
            return func(*args, **kwargs)  # Proceed with the API request
        else:
            return jsonify({'error': 'Invalid API key'}), 401  # Return error
response for unauthorized access

    return wrapper

@app.route('/users_data', methods=['GET', 'POST'])
@authenticate_api_key
def handle_users():
    conn = get_db()
    cursor = conn.cursor()

    if request.method == 'GET':
        cursor.execute("SELECT * FROM person_database_sri")
        rows = cursor.fetchall()

        users = []
```

```
        for row in rows:
            user = {
                'id': row[0],
                'name': row[1],
                'gender': row[2],
                'password': row[3],
                'email': row[4]
            }
            users.append(user)

        if len(users) > 0:
            return jsonify(users)
        else:
            return 'No users found', 404

    if request.method == 'POST':
        try:
            id = request.form['id']
            name = request.form['name']
            email = request.form['email']
            gender = request.form['gender']
            password = request.form['password']

            send = send_mail(id, name, gender, password, email)

            if send == True:
                sql = """INSERT INTO person_database_sri (id, name, email, gender,
password)
                         VALUES (%s, %s, %s, %s, %s)"""
                cursor.execute(sql, (id, name, email, gender, password))
                conn.commit()
                return "Database updated successfully", 200
            else:
                return "Fail to send mail"
        except Exception as e:
            return f"Failed to upload data: {str(e)}", 500

def get_valid_api_keys():
    try:
        conn = get_db()
        cursor = conn.cursor()
        cursor.execute('SELECT api_key FROM API_KEY')
        api_keys = [row[0] for row in cursor.fetchall()]
        cursor.close()
        return api_keys
    except Exception as e:
        return []

def delete_row(username):
    conn = get_db()
    cursor = conn.cursor()

    query = "DELETE FROM dev_datas WHERE username = %s"
    cursor.execute(query, (username,))

    conn.commit()
```

```
    cursor.close()
    conn.close()

@app.route('/data', methods=['GET'])
def get_data():
    try:
        with open('intents/intents.json') as file:
            data = json.load(file)
            return jsonify(data['intents'])
    except FileNotFoundError:
        return jsonify({'error': 'File not found'})

@app.route('/data', methods=['POST'])
def save_data():
    data = request.get_json()
    try:
        with open('your_file.json', 'w') as file:
            json.dump(data, file)
            return jsonify({'message': 'Data saved successfully'})
    except:
        return jsonify({'error': 'Failed to save data'})

@app.route('/delete', methods=['POST'])
def delete_user():
    if request.method == 'POST':
        user_id = request.form['user_id']
        hash_url = request.form['url']
        conn = get_db()
        cursor1 = conn.cursor()
        cursor2 = conn.cursor()
        cursor3 = conn.cursor()
        cursor3.execute("SELECT * FROM person_database_sri WHERE id = %s",
(user_id,))
        rows = cursor3.fetchall()
        users = []
        for row in rows:
            user = {
                'name': row[1],
                'email': row[4]
            }
            users.append(user)
        cursor3.close()
        if len(users) > 0:
            name = users[0]['name']
            mail = users[0]['email']
            cursor1.execute('DELETE FROM person_database_sri WHERE id = %s',
(user_id,))
            cursor1.close()
            cursor2.execute('DELETE FROM test_user WHERE name = %s', (user_id,))
            cursor2.close()
            delete_mail(name, mail)
            flash('Data deleted successfully', 'success')
            session['keep_flashed_messages'] = True
        else:
            flash('User not found', 'error')
```

```
        conn.commit()
        conn.close()

    return redirect(url_for('profile', hash_url=hash_url))


@app.context_processor
def inject_flashed_messages():
    # Retrieve the flashed messages and decide whether to keep them or not
    messages = get_flashed_messages()
    if session.get('keep_flashed_messages'):
        session.pop('keep_flashed_messages')
    else:
        messages = None
    return dict(messages=messages)

@app.route(f"/{url}/about/Srishti")
def about_page():
    conn = get_db()

    cursor1 = conn.cursor()
    cursor2 = conn.cursor()

    cursor1.execute('SELECT images FROM dev_datas')
    cursor2.execute('SELECT username, bio FROM dev_datas')  # Modify the query to
include the bio field

    data = cursor1.fetchall()
    cursor1.close()
    users = cursor2.fetchall()
    cursor2.close()

    image_user_mapping = {}  # Dictionary to store image-user mapping

    for i in range(len(users)):
        image_data = data[i][0]
        image_encoded = base64.b64encode(image_data).decode('utf-8')
        username = users[i][0]
        bio = users[i][1]  # Get the bio from the fetched data
        image_user_mapping[image_encoded] = {'username': username, 'bio': bio}  #
Store both username and bio in the dictionary

    conn.close()

    return render_template('about.html', image_user_mapping=image_user_mapping,
url=url)

@app.route(f'/{url}/contact/us')
def contact_page():
    conn = get_db()

    cursor1 = conn.cursor()
    cursor2 = conn.cursor()

    cursor1.execute('SELECT images FROM dev_datas')
    cursor2.execute('SELECT username FROM dev_datas')
```

```
    data = cursor1.fetchall()
    cursor1.close()
    users = cursor2.fetchall()
    cursor2.close()
    # print(users)
    image_user_mapping = {}  # Dictionary to store image-user mapping

    for i in range(len(users)):
        image_data = data[i][0]
        image_encoded = base64.b64encode(image_data).decode('utf-8')
        username = users[i][0]
        image_user_mapping[image_encoded] = username

    conn.close()

    return render_template('contact.html', image_user_mapping=image_user_mapping,
url=url)

# @app.route('/email_all/<username>', methods=['GET'])
# def email_all(username):
#     # delete_row(username)
#     return f"Deleted user with username: {username}"

if __name__ == '__main__':
    with app.app_context():
        create_table()
    app.run(debug=True)
```

CSS Code:

[1] STYLE SCRIPT :

```
body {
    margin: 0;
    font-family: Arial, Helvetica, sans-serif;
}

a {
    text-decoration: none;
    color: inherit;
}

.btn-download{
    color: blue;
    animation: chenge 1s ease-in-out infinite alternate;
}

@keyframes chenge {
    from {
        color: #bd00ff;
        text-shadow: 0 0 10px #00b7ff, 0 0 20px #00b7ffbe, 0 0 30px #00b7ff7b;
    }
    to {
        color: #00b8ff;
        text-shadow: 0 0 20px #bd00ff, 0 0 30px #bb00ffa5, 0 0 40px #bb00ff65;
    }
}

.body{
    padding-left: 10px;
    padding-right: 10px;
}

/* .body_log {
    background-color: #f2f2f2;
    display: flex;
    justify-content: center;
    align-items: center;
    height: auto;
}
.container {
    background-color: #566573;
    padding: 50px;
    border-radius: 8px;
} */

.body_log {
    background-color: #f2f2f2;
    display: flex;
    /* padding: 50px; */
    padding-top: 60px;
    padding-bottom: 60px;
    justify-content: center;
```

```
    align-items: center;
    height: 50vh;
}
.container {
    background-color: #566573;
    margin: 50px;
    color: #f2f2f2;
    max-width: 400px;
    padding: 20px;
    border-radius: 8px;
}
@media screen and (max-width: 600px) {
    .container {
        margin: 50px;
        max-width: 60%;
    }
    /* .body_log{
        padding: 20px;
    } */
}

.log_in{
    border: none;
    max-width: 100%;
    width: 100%;
    max-height: 50px;
    height: 100%;
    padding: 10px;
    background-color: #04AA6D;
    color: #f2f2f2;

}

.log_in:hover{
    background-color: #ddd;
    color: black;
}

.reg_up{
    border: none;
    max-width: 100%;
    width: 100%;
    max-height: 50px;
    height: 100%;
    padding: 10px;
    background-color: #04AA6D;
    color: #f2f2f2;

}

.reg_up:hover{
    background-color: #ddd;
    color: black;
}

.topnav {
```

```
    overflow: hidden;
    background-color: #333;
}

.topnav a {
    float: left;
    display: block;
    color: #f2f2f2;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
    font-size: 17px;
}

.topnav a:hover {
    background-color: #ddd;
    color: black;
}

.topnav a.active {
    background-color: #04AA6D;
    color: white;
}

.topnav .icon {
    display: none;
}

@media screen and (max-width: 600px) {
    .topnav a:not(:first-child) {display: none;}
    .topnav a.icon {
        float: right;
        display: block;
    }
}

@media screen and (max-width: 600px) {
    .topnav.responsive {position: relative;}
    .topnav.responsive .icon {
        position: absolute;
        right: 0;
        top: 0;
    }
    .topnav.responsive a {
        float: none;
        display: block;
        text-align: left;
    }
}

.banner {
    background-image: url('https://images-wixmp-
ed30a86b8c4ca887773594c2.wixmp.com/f/7a5a3db7-fb07-4532-aa4c-93f5a5d5d651/d9l0muy-
fdf7c7ba-15e3-4ac4-9847-
0672b71190b0.gif?token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1cm46YXBwOjdlMGQxO
Dg5ODIyNjQzNzNhNWYwZDQxNWVhMGQyNmUwIiwiaXNzIjoidXJuOmFwcDo3ZTBkMTg4OTgyMjY0MzczYTVmMGQ0MT
```

```
VlYTBkMjZlMCIsIm9iaiI6W1t7InBhdGgiOiJcL2ZcLzdhNWEzZGI3LWZiMDctNDUzMi1hYTRjLTkzZjVhNWQ1ZDY
1MVwvZDlsMG11eS1mZGY3YzdiYS0xNWUzLTRhYzQtOTg0Ny0wNjcyYjcxMTkwYjAuZ2lmIn1dXSwiYXVkIjpbInVy
bjpzZXJ2aWNlOmZpbGUuZG93bmxvYWQiXX0.gkunQX3ImB0huji1sJ6H0bCxws9be-gfGs6M-7Twwx0');
    background-size: cover;
    background-position: center;
    max-height: auto;
    height: 200px;
    display: flex;
    text-align: center;
    align-items: center;
    justify-content: center;
    color: #f2f2f2;
}
.banner h1{
    font-size: 60px;
    color: #bd00ff;
    text-shadow: 0 0 10px #d600ff;
    animation: glow 1s ease-in-out infinite alternate;
}
@keyframes glow {
    from {
        color: #bd00ff;
        text-shadow: 0 0 10px #00b7ff, 0 0 20px #00b7ffbe, 0 0 30px #00b7ff7b;
    }
    to {
        color: #00b8ff;
        text-shadow: 0 0 20px #bd00ff, 0 0 30px #bb00ffa5, 0 0 40px #bb00ff65;
    }
}
/* .user-images{
    width: 200px;
    height: 200px;
    border-radius: 50%;
    overflow: hidden;
} */

.user-images {
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
    gap: 40px;
    padding: 20px;
}

.user-card {
    text-align: center;
}

.user-card img {
    border-radius: 50%;
    width: 80px;
    height: 80px;
}

@media only screen and (max-width: 768px) {
    .user-card img {
```

```
        width: 50px;
        height: 50px;
    }
}

@media (max-width: 768px) {
    .user-images table {
        display: block;
    }

    .user-images th {
        display: inline-block;
        vertical-align: top;
        width: 33.33%;
        padding: 10px;
    }
}
.footer{
    background-color: #333;
    color: #f2f2f2;
}

.footer p {
    padding: 20px;
    text-align: center;
    margin: 0;
}

.info_pic {
    width: 200px;
    height: 200px;
    border-radius: 50%;
    object-fit: cover;
    box-shadow: 0 0px 20px rgba(0, 0, 0, 0.4);
    border: 2px solid #fff;
}

@media (max-width: 768px) {
    .info_pic {
        width: 70px;
        height: 70px;
    }
    .info_container {
        flex-direction: column;
        align-items: center;
        text-align: center;
        padding: 10px;
    }
}

#bio {
    width: 100%;
    box-sizing: border-box;
  }

#dev_key {
```

```
    width: 100%;
    box-sizing: border-box;
}
#username {
    width: 100%;
    box-sizing: border-box;
}
#password {
    width: 100%;
    box-sizing: border-box;
}

#email {
    width: 100%;
    box-sizing: border-box;
}

  /* Styles for larger screens */
  @media only screen and (min-width: 768px) {
    #bio {
      max-width: 500px;
    }
    #dev_key {
        max-width: 295px;
      }
      #username {
        max-width: 315px;
      }
      #password {
        max-width: 318px;
      }
    #email {
        max-width: 350px;
    }
  }

  /* Styles for smaller screens */
  @media only screen and (max-width: 767px) {
    #bio {
      max-width: 100%;
    }
    #dev_key {
        max-width: 100%;
    }
    #username {
        max-width: 100%;
    }
    #password {
        max-width: 100%;
    }
    #email {
        max-width: 100%;
    }
  }
```

[2] PROFILE SCRIPT :

```
body {
    margin: 0;
    font-family: Arial, Helvetica, sans-serif;
}

a {
    text-decoration: none;
    color: inherit;
}

.body{
    padding-left: 10px;
    padding-right: 10px;
}

.pronav {
    overflow: hidden;
    position: relative;
    text-align: center;
    justify-content: center;
    color: aliceblue;
}

.pronav::before {
    content: "";
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background-image: url('https://images-wixmp-
ed30a86b8c4ca887773594c2.wixmp.com/f/7a5a3db7-fb07-4532-aa4c-93f5a5d5d651/dbff1rj-
c8fb290d-5a35-4137-b925-
759ea0e98f44.gif?token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1cm46YXBwOjdlMGQxO
Dg5ODIyNjQzNzNhNWYwZDQxNWVhMGQyNmUwIiwiaXNzIjoidXJuOmFwcDo3ZTBkMTg4OTgyMjY0MzczYTVmMGQ0MT
VlYTBkMjZlMCIsIm9iaiI6W1t7InBhdGgiOiJcL2ZcLzdhNWEzZGI3LWZiMDctNDUzMi1hYTRjLTkzZjVhNWQ1ZDY
1MVwvZGJmZjFyai1jOGZiMjkwZC01YTM1LTQxMzctYjkyNS03NTllYTBlOThmNDQuZ2lmIn1dXSwiYXVkIjpbInVy
bjpzZXJ2aWNlOmZpbGUuZG93bmxvYWQiXX0.Zk74FqcSimKjviOn9BE6AJBML75_bVcz0PzvFmwgm2Y');
    background-size: cover;
    filter: blur(8px); /* Adjust the blur intensity as needed */
    z-index: -1;
}

.probanner {
    background-image: url('https://images-wixmp-
ed30a86b8c4ca887773594c2.wixmp.com/f/7a5a3db7-fb07-4532-aa4c-93f5a5d5d651/dbff1rj-
c8fb290d-5a35-4137-b925-
759ea0e98f44.gif?token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1cm46YXBwOjdlMGQxO
Dg5ODIyNjQzNzNhNWYwZDQxNWVhMGQyNmUwIiwiaXNzIjoidXJuOmFwcDo3ZTBkMTg4OTgyMjY0MzczYTVmMGQ0MT
VlYTBkMjZlMCIsIm9iaiI6W1t7InBhdGgiOiJcL2ZcLzdhNWEzZGI3LWZiMDctNDUzMi1hYTRjLTkzZjVhNWQ1ZDY
1MVwvZGJmZjFyai1jOGZiMjkwZC01YTM1LTQxMzctYjkyNS03NTllYTBlOThmNDQuZ2lmIn1dXSwiYXVkIjpbInVy
bjpzZXJ2aWNlOmZpbGUuZG93bmxvYWQiXX0.Zk74FqcSimKjviOn9BE6AJBML75_bVcz0PzvFmwgm2Y');
    background-size: cover;
    background-position: center;
```

```
    display: flex;
    color: #f2f2f2;
    padding: 20px;
}

.propic {
    padding: 20px;
}

.profile_pic {
    width: 200px;
    height: auto;
    border-radius: 50%;
    object-fit: cover;
    box-shadow: 0 0px 20px rgba(0, 0, 0, 0.4);
    border: 2px solid #fff;
    animation: glow 1s ease-in-out infinite alternate;
}

@keyframes glow {
    from {
        box-shadow: 0 0 10px #00b7ff, 0 0 20px #00b7ffbe, 0 0 30px #00b7ff7b;
    }
    to {
        box-shadow: 0 0 20px #bd00ff, 0 0 30px #bb00ffa5, 0 0 40px #bb00ff65;
    }
}

.profile_text {
    background-color: #3333338c;
    width: 100%;
    padding: 25px;
    border-radius: 20px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
    backdrop-filter: blur(10px);
}

/* Responsive Styles */

@media screen and (max-width: 768px) {
    .pronav {
        text-align: left;
        justify-content: flex-start;
        padding: 10px;
    }

    .pronav h1 {
        margin: 0;
    }

    .probanner {
        flex-direction: column;
        align-items: center;
        padding: 10px;
    }
```

```
    .propic {
        padding: 10px;
    }

    .profile_text {
        padding: 15px;
    }
}

.user_database{
    margin: 20px;
    /* display: flex; */
    max-width: auto;
    background-color: #f2f2f2;
    padding: 20px;
    text-align: center;
    align-items: center;
    justify-content: center;
}

.user_database table{
    border: 2px solid #333;
    border-collapse: collapse;
    width: 100%;
}

.user_database table th{
    color: antiquewhite;
    background-color: #333;
}

.table-container {
    display: none;
}
.active {
    display: block;
}
table {
    width: 100%;
    border-collapse: collapse;
    margin-bottom: 20px;
}

table, th, td {
    border: 1px solid black;
    padding: 8px;
}

th {
    background-color: #f2f2f2;
    text-align: left;
}

@media screen and (max-width: 600px) {
    table {
        border: 0;
```

```
    }

    table thead {
        display: none;
    }

    table tr {
        margin-bottom: 10px;
        display: block;
        border-bottom: 2px solid #ddd;
    }

    table td, table th {
        display: block;
        text-align: left;
        font-size: 13px;
        border-bottom: 1px dotted #ccc;
    }

    table td::before, table th::before {
        content: attr(data-label);
        float: left;
        font-weight: bold;
        text-transform: uppercase;
    }
}

.api_tab{
    max-width: 80%;
    width: 20%;
}
```

Html Code:

[1] INDEX PAGE :

```
<!DOCTYPE html>
<html>
<head>
    <title>Welcome to Srishti</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.4/css/all.min.css">
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
    <link rel="icon" href="{{ url_for('static', filename='favicon.ico') }}"
type="image/x-icon">
    <link rel="shortcut icon" href="{{ url_for('static', filename='favicon.ico') }}"
type="image/x-icon">
</head>
<body>

<div class="topnav" id="myTopnav">
  <a href="/" class="active">Home</a>
  <a href="/login">Log in</a>
  <a href="/register">Register</a>
  <a href="/about">About</a>
  <a href="/download">Download</a>
  <a href="/contact">Contact Us</a>
  <a href="javascript:void(0);" class="icon" onclick="myFunction()">
    <i class="fa fa-bars"></i>
  </a>
</div>

<div class="banner">
    <h1 style="justify-content: center;">Welcome</h1>
</div>
<div class="body">
    <main>
      <div class="content">
          <h2>Srishti AI</h2>
          <p>Srishti AI is an advanced speech recognition system that utilizes cutting-
edge technology to provide accurate and intelligent speech recognition capabilities.
Our internal system hosts a range of powerful features designed to enhance your speech
recognition experience.</p>

          <h2>Key Features:</h2>
          <ul>
              <li>Highly accurate speech recognition</li>
              <li>Support for multiple languages</li>
              <li>Real-time transcription and translation</li>
              <li>Customizable vocabulary and voice commands</li>
              <li>Intelligent voice assistant integration</li>
              <li>Secure and private data handling</li>
          </ul>

          <h2>Download Srishti AI:</h2>
```

```
          <p>To experience the power of Srishti AI, download our speech recognition
system today. Click the button below to start the download.</p>
          <a
href="https://drive.google.com/uc?export=download&id=1qfE28dNIxw7uOo9sER2sHSPvILNyPEVw"
class="btn-download">Download Now</a>

          <div class="testimonial">
              <blockquote>"Srishti AI has revolutionized the way we interact with
speech recognition systems. Its accuracy and intelligence are unmatched, making it an
invaluable tool for our organization."</blockquote>
              <!-- <cite>John Smith, CEO of ABC Corporation</cite> -->
          </div>

          <h2>Enhance Your Productivity</h2>
          <p>Srishti AI is not just a speech recognition system; it's a productivity
powerhouse. Whether you're a professional writer, a busy executive, or a student
preparing for exams, Srishti AI can significantly boost your efficiency and
effectiveness.</p>

          <h2>Seamless Integration</h2>
          <p>Integrate Srishti AI seamlessly into your existing workflows. Our system
is compatible with popular productivity tools and platforms, allowing you to dictate
documents, compose emails, take voice notes, and more with ease.</p>

          <h2>Privacy and Security</h2>
          <p>At Srishti AI, we prioritize the privacy and security of your data. Our
system employs advanced encryption and data protection measures, ensuring that your
voice recordings and personal information are kept confidential.</p>

          <h2>Continuous Improvement</h2>
          <p>We are committed to continuously improving Srishti AI by leveraging the
latest advancements in machine learning and artificial intelligence. This means you can
expect regular updates and enhancements to further enhance your speech recognition
experience.</p>
      </div>
    </main>
</div>
<div class="footer">
    <h3 style="padding-left: 10px; padding-top: 10px;">This our Developer team</h3><br>
    <div class="user-images">
      {% for image, username in image_user_mapping.items() %}
      <div class="user-card">
          <img src="{{ image }}" alt="Image">
          <h4>
            {% if username == 'Arnab Mondal' %}
                Main Developer: {{ username }}
            {% else %}
                Developer: {{ username }}
            {% endif %}
          </h4>
      </div>
      {% endfor %}
    </div>
    <hr style="max-width: 90%; width: 100%;">
    <p>&copy; 2023 Srishti. All rights reserved.</p>
</div>
```

```
<script>
function myFunction() {
  var x = document.getElementById("myTopnav");
  if (x.className === "topnav") {
    x.className += " responsive";
  } else {
    x.className = "topnav";
  }
}
</script>

</body>
</html>
```

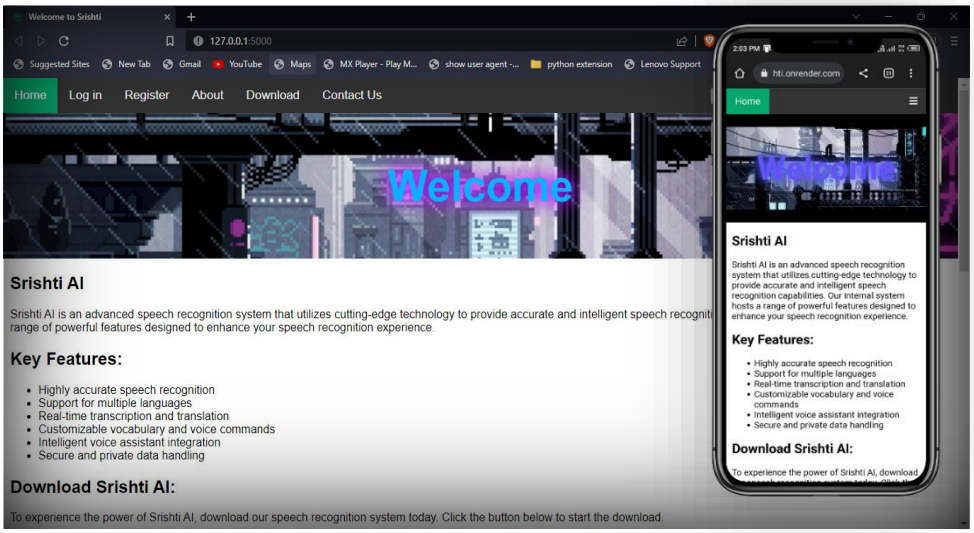


Figure 5.1.1: Home Page

[2] ABOUT PAGE :

```
<!DOCTYPE html>
<html>
<head>
    <title>About</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.4/css/all.min.css">
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
    <link rel="icon" href="{{ url_for('static', filename='favicon.ico') }}"
type="image/x-icon">
    <link rel="shortcut icon" href="{{ url_for('static', filename='favicon.ico') }}"
type="image/x-icon">
</head>
<body>

  <div class="topnav" id="myTopnav">
    <a href="/" class="active">Home</a>
    <a href="/login">Log in</a>
```

```
    <a href="/register">Register</a>
    <a href="/about">About</a>
    <a href="/download">Download</a>
    <a href="/contact">Contact Us</a>
    <a href="javascript:void(0);" class="icon" onclick="myFunction()">
      <i class="fa fa-bars"></i>
    </a>
  </div>
<div class="banner">
    <h1 style="justify-content: center">About Srishti</h1>
</div>
<div class="body">
    <h3 style="padding-left: 10px; padding-top: 10px;">Developer team</h3><br>
    {% for image, user_info in image_user_mapping.items() %}
    <div style="display: flex; padding: 20px;" class="info_container">
        <div>
            <img src="{{ image }}" alt="Image"
class="info_pic">
        </div>
        <div style="padding: 20px;">
            <h4>
                {% if user_info.username == 'Arnab Mondal' %}
                    Main Developer: {{ user_info.username }}
                {% else %}
                    Developer: {{ user_info.username }}
                {% endif %}
            </h4>
            <p><i>{{ user_info.bio }}</i></p>  <!-- Access the bio data -->
        </div>
    </div>
    {% endfor %}
</div>

<div class="footer">
  <hr style="max-width: 90%; width: 100%;">
  <p>&copy; 2023 Srishti. All rights reserved.</p>
</div>

<script>
function myFunction() {
  var x = document.getElementById("myTopnav");
  if (x.className === "topnav") {
    x.className += " responsive";
  } else {
    x.className = "topnav";
  }
}
</script>

</body>
</html>
```

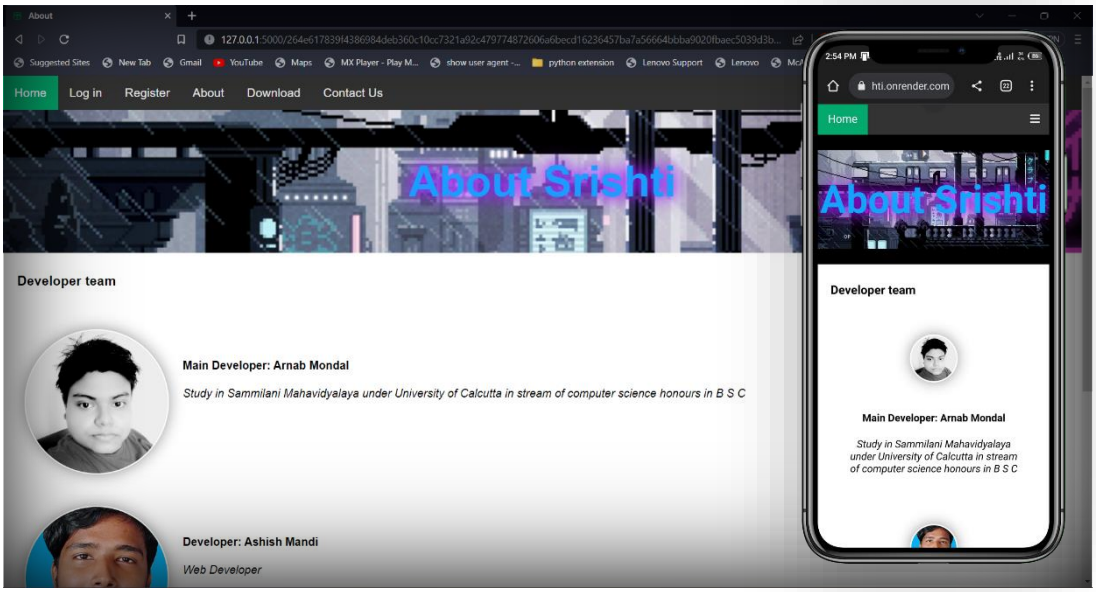


Figure 5.1.2: About Page

[3] DOWNLOAD PAGE :

```
<!DOCTYPE html>
<html>
<head>
    <title>Download</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.4/css/all.min.css">
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
    <link rel="icon" href="{{ url_for('static', filename='favicon.ico') }}"
type="image/x-icon">
    <link rel="shortcut icon" href="{{ url_for('static', filename='favicon.ico') }}"
type="image/x-icon">
</head>
<body>

  <div class="topnav" id="myTopnav">
    <a href="/" class="active">Home</a>
    <a href="/login">Log in</a>
    <a href="/register">Register</a>
    <a href="/about">About</a>
    <a href="/download">Download</a>
    <a href="/contact">Contact Us</a>
    <a href="javascript:void(0);" class="icon" onclick="myFunction()">
      <i class="fa fa-bars"></i>
    </a>
  </div>
<div class="banner">
    <h1 style="justify-content: center">Download Srishti</h1>
</div>
<div class="body">
    <!-- Feature Section -->
```

```
    <section>
        <h2>Key Features</h2>
        <ul>
            <li>Accurate and efficient speech-to-text conversion</li>
            <li>Support for multiple languages and accents</li>
            <li>Real-time transcription with instant feedback</li>
            <li>Intelligent speaker recognition</li>
            <li>Advanced technical intelligence for improved accuracy</li>
        </ul>
    </section>

    <!-- Download Section -->
    <section>
        <h2>Get Started with Srishti AI Speech Recognition</h2>
        <p>Download our software and start experiencing the power of advanced speech
recognition.</p>
        <a
href="https://drive.google.com/uc?export=download&id=1qfE28dNIxw7uOo9sER2sHSPvILNyPEVw"
target="_blank" class="btn-download">Download Now</a>
    </section>

    <!-- About Section -->
    <section>
        <h2>About Srishti AI Speech Recognition</h2>
        <p>Srishti AI Speech Recognition is a state-of-the-art speech recognition
system designed to provide accurate and efficient speech-to-text conversion. With
advanced technical intelligence, it offers seamless transcription capabilities across
multiple languages and accents. The system's real-time feedback and intelligent speaker
recognition make it a versatile solution for various applications.</p>
    </section>
</div>

<div class="footer">
  <h3 style="padding-left: 10px; padding-top: 10px;">This our Developer team</h3><br>
  <div class="user-images">
    {% for image, username in image_user_mapping.items() %}
    <div class="user-card">
        <img src="{{ image }}" alt="Image">
        <h4>
          {% if username == 'Arnab Mondal' %}
              Main Developer: {{ username }}
          {% else %}
              Developer: {{ username }}
          {% endif %}
        </h4>
    </div>
    {% endfor %}
  </div>
  <hr style="max-width: 90%; width: 100%;">
  <p>&copy; 2023 Srishti. All rights reserved.</p>
</div>

<script>
function myFunction() {
  var x = document.getElementById("myTopnav");
  if (x.className === "topnav") {
```

```
    x.className += " responsive";
  } else {
    x.className = "topnav";
  }
}
</script>

</body>
</html>
```

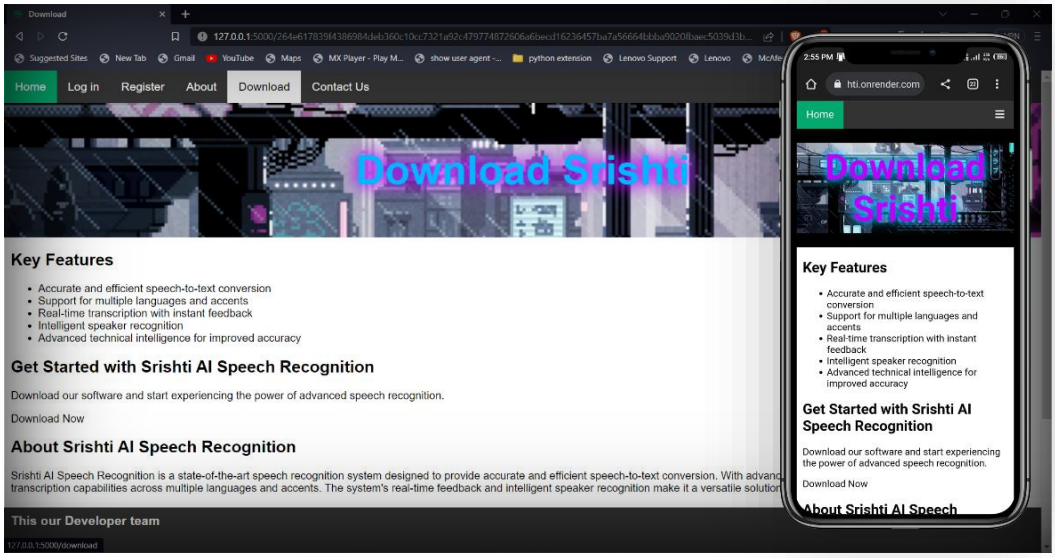


Figure 5.1.3: Download Page

[4] CONTACT PAGE :

```
<!DOCTYPE html>
<html>
<head>
    <title>Contact Us</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.4/css/all.min.css">
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
    <link rel="icon" href="{{ url_for('static', filename='favicon.ico') }}"
type="image/x-icon">
    <link rel="shortcut icon" href="{{ url_for('static', filename='favicon.ico') }}"
type="image/x-icon">
<style>
  .contact-details {
    display: flex;
    flex-wrap: wrap;
    justify-content: space-between;
}

.location, .email, .image-gif {
    flex-basis: 100%;
    margin-bottom: 20px;
```

```
}

@media (min-width: 768px) {
  .location, .email, .image-gif {
    flex-basis: calc(50% - 20px);
  }
}

@media (min-width: 992px) {
  .location, .email, .image-gif {
    flex-basis: calc(33.33% - 20px);
  }
}

.location, .email {
    margin-bottom: 20px;
    display: flex;
}

.location p, .email p {
    margin-right: 10px;
}

.location img,
.email img {
    width: 50px;
    height: 50px;
}

.image-gif {
    background-image:
url("https://cdna.artstation.com/p/assets/images/images/028/102/058/original/pixel-
jeff-matrix-s.gif?1593487263");
    width: 100%;
    max-width: 500px;
    height: 200px;
    background-size: cover;
    background-position: center;
    display: flex;
    color: #f2f2f2;
    border-radius: 30px;
    box-shadow: 0 0px 20px rgba(0, 0, 0, 0.4);
    align-items: center;
    text-align: center;
    justify-content: center;
}

.text-animation {
    animation: textAnimation 5s ease-in-out infinite;
}

@keyframes textAnimation {
    0% {
        opacity: 0;
        transform: scale(0);
    }
```

```
    50% {
        opacity: 1;
        transform: scale(1.2);
    }
    100% {
        opacity: 0;
        transform: scale(0);
    }
}

</style>
</head>
<body>

  <div class="topnav" id="myTopnav">
    <a href="/" class="active">Home</a>
    <a href="/login">Log in</a>
    <a href="/register">Register</a>
    <a href="/about">About</a>
    <a href="/download">Download</a>
    <a href="/contact">Contact Us</a>
    <a href="javascript:void(0);" class="icon" onclick="myFunction()">
      <i class="fa fa-bars"></i>
    </a>
  </div>

<div class="banner">
    <h1 style="justify-content: center;">Contact Us</h1>
</div>
<div class="body">
    <main>
      <div class="content">
        <div class="contact-info">
          <h2>Talk to us</h2>
          <p>We would love to hear from you! Please don't hesitate to reach out to us
with any questions, comments, or concerns you may have</p>
        </div>
        <div class="contact-details">
          <div class="location">
            <a href="https://goo.gl/maps/p7biHrvaFpTo3Mky9"><img
src="https://i.pinimg.com/originals/20/24/22/2024229a3dde09deb53b13e901bc9b65.png"
alt="pin"></a>
            <p>
              <b>Sammilani Mahavidyalaya</b><br>
              Baghajatin Eastern Metropolitan Bypass,<br>1925, Chak Garia, Baghajatin
Colony,<br>Kolkata-700094<br>West Bengal
            </p>
          </div>
          <div class="email">
            <img
src="https://i.pinimg.com/originals/c6/19/20/c61920ea7ddcecd752ff8874c58c2627.png"
alt="mail">
            <p>
              <b>Contact Us</b><br>
              <b>Email: </b>srishti.ai.arnab.ashish.tasir@gmail.com
            </p>
```

```
          </div>
          <div class="image-gif">
            <div style="background-color: #3333335e; width: 100%; height: 100%;
backdrop-filter: blur(5px); border-radius: 30px; align-items: center; justify-
content: center; text-align: center; display: flex;">
              <div class="text-animation">Welcome to our website</div>
            </div>
          </div>
        </div>
      </div>
    </main>
</div>
<div class="footer">
    <h3 style="padding-left: 10px; padding-top: 10px;">This our Developer
team</h3><br>
    <div class="user-images">
      {% for image, username in image_user_mapping.items() %}
      <div class="user-card">
          <img src="{{ image }}" alt="Image">
          <h4>
            {% if username == 'Arnab Mondal' %}
                Main Developer: {{ username }}
            {% else %}
                Developer: {{ username }}
            {% endif %}
          </h4>
      </div>
      {% endfor %}
    </div>
    <hr style="max-width: 90%; width: 100%;">
    <p>&copy; 2023 Srishti. All rights reserved.</p>
</div>

<script>
function myFunction() {
  var x = document.getElementById("myTopnav");
  if (x.className === "topnav") {
    x.className += " responsive";
  } else {
    x.className = "topnav";
  }
}

var textElement = document.querySelector(".text-animation");
var texts = ["We are happy to help you", "Don't hesitate contact us", "We answer your
question or your queries", "Anytime anywhere", "We're glad to answer your question"];
// Array of text options

function changeText() {
    var randomIndex = Math.floor(Math.random() * texts.length);
    var newText = texts[randomIndex];
    textElement.textContent = newText;
}

// Call the changeText function every 4 seconds
setInterval(changeText, 4000);
```

```
</script>

</body>
</html>
```

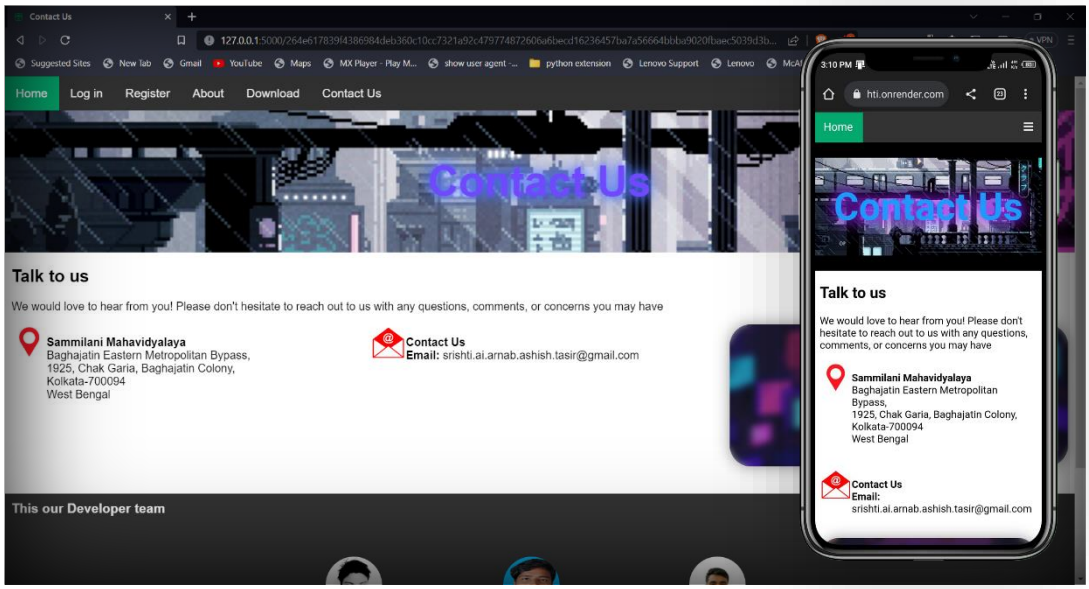


Figure 5.1.4: Contact Page

[5] REGISTER PAGE :

```
<!DOCTYPE html>
<html>
<head>
    <title>Developer registration page</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.4/css/all.min.css">
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
    <link rel="icon" href="{{ url_for('static', filename='favicon.ico') }}"
type="image/x-icon">
    <link rel="shortcut icon" href="{{ url_for('static', filename='favicon.ico') }}"
type="image/x-icon">
</head>
<body>

  <div class="topnav" id="myTopnav">
    <a href="/" class="active">Home</a>
    <a href="/login">Log in</a>
    <a href="/register">Register</a>
    <a href="/about">About</a>
    <a href="/download">Download</a>
    <a href="/contact">Contact Us</a>
    <a href="javascript:void(0);" class="icon" onclick="myFunction()">
      <i class="fa fa-bars"></i>
    </a>
  </div>
```

```
<div class="banner">
    <h1 style="justify-content: center">Developer Registration</h1>
</div>
<br>
<div class="body_log">
    <div class="container">
        <form method="POST" enctype="multipart/form-data">
            <label for="dev_key"><b>Company ID:</b></label>
            <input type="text" id="dev_key" name="dev_key" required><br><br>

            <label for="username">Username:</label>
            <input type="text" id="username" name="username" required><br><br>

            <label for="password">Password:</label>
            <input type="password" id="password" name="password" required><br><br>

            <label for="email">Email:</label>
            <input type="email" id="email" name="email" required><br><br>

            <label for="profile_picture">Profile Picture:</label>
            <input type="file" id="profile_picture" name="profile_picture"
required><br><br>

            <label for="bio">Bio:</label>
            <textarea id="bio" name="bio" rows="4" cols="50"></textarea>

            <input type="submit" value="Register" class="reg_up">
        </form>
    </div>
</div>

<div class="footer">
    <h1 style="text-align: center; padding-top: 10px;">
        This is the page of the developer to log in. This is only for developers, not
for any other users.
    </h1>
    <hr style="max-width: 90%; width: 100%;">
    <p>&copy; 2023 Srishti. All rights reserved.</p>
</div>

<script>
function myFunction() {
  var x = document.getElementById("myTopnav");
  if (x.className === "topnav") {
    x.className += " responsive";
  } else {
    x.className = "topnav";
  }
}
</script>

</body>
</html>
```

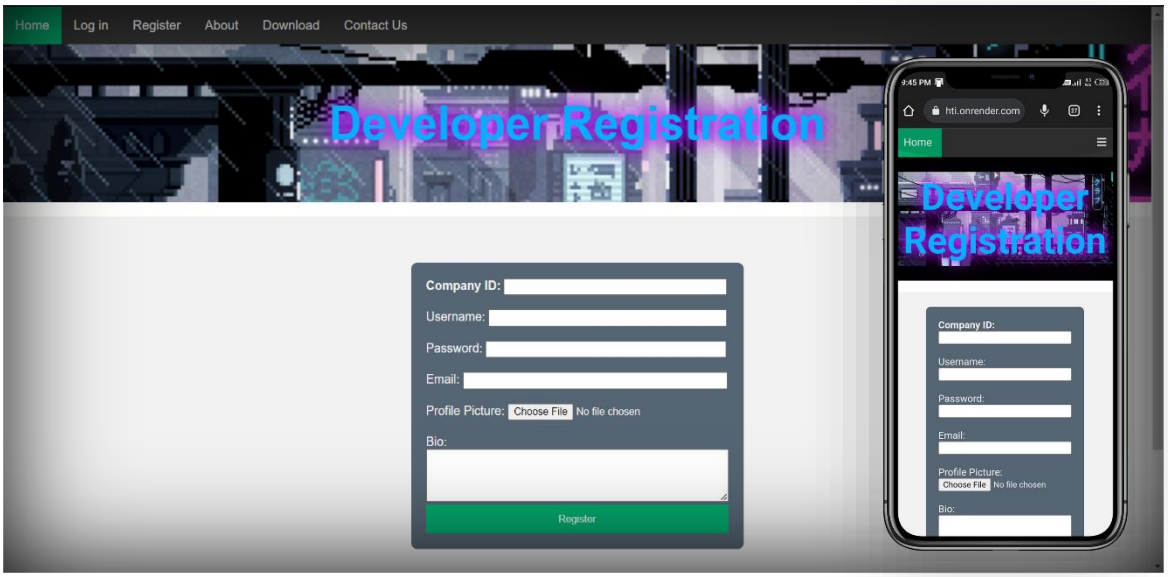


Figure 5.1.5: Registration Page

[6] LOGIN PAGE:

```
<!DOCTYPE html>
<html>
<head>
    <title>Developer login page</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.4/css/all.min.css">
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
    <link rel="icon" href="{{ url_for('static', filename='favicon.ico') }}"
type="image/x-icon">
    <link rel="shortcut icon" href="{{ url_for('static', filename='favicon.ico') }}"
type="image/x-icon">
</head>
<body>

  <div class="topnav" id="myTopnav">
    <a href="/" class="active">Home</a>
    <a href="/login">Log in</a>
    <a href="/register">Register</a>
    <a href="/about">About</a>
    <a href="/download">Download</a>
    <a href="/contact">Contact Us</a>
    <a href="javascript:void(0);" class="icon" onclick="myFunction()">
      <i class="fa fa-bars"></i>
    </a>
  </div>

<div class="banner">
    <h1 style="justify-content: center">Developer login</h1>
</div>
<br>
<div class="body_log">
```

```
    <div class="container">
        <form method="POST">
            <label for="username">Username:</label>
            <input type="text" id="username" name="username" required><br><br>

            <label for="password">Password:</label>
            <input type="password" id="password" name="password" required><br><br>

            <input type="submit" value="Login" class="log_in">
        </form>
    </div>
</div>

<div class="footer">
    <h1 style="text-align: center; padding-top: 10px;">
        This is the page of developer to login this is only for developer not for any
other user
    </h1>
    <hr style="max-width: 90%; width: 100%;">
    <p>&copy; 2023 Srishti. All rights reserved.</p>
</div>

<script>
function myFunction() {
  var x = document.getElementById("myTopnav");
  if (x.className === "topnav") {
    x.className += " responsive";
  } else {
    x.className = "topnav";
  }
}
</script>

</body>
</html>
```

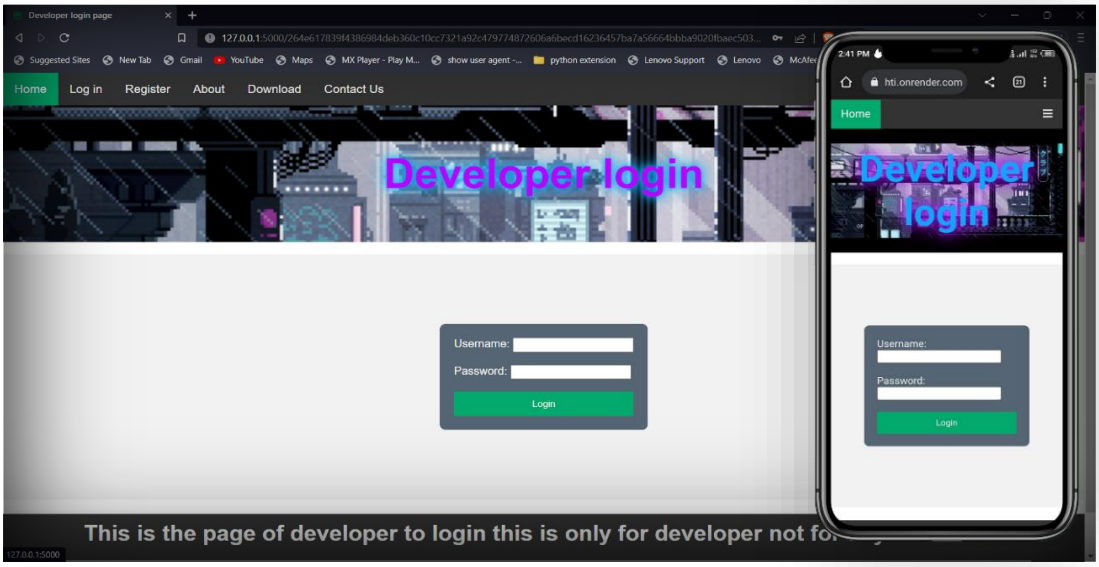


Figure 5.1.6: Login Page

[7] PROFILE PAGE:

```
<!DOCTYPE html>
<html>
<head>
    <title>Profile</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.4/css/all.min.css">
    <link rel="stylesheet" href="{{ url_for('static', filename='profile.css') }}">
    <link rel="icon" href="{{ url_for('static', filename='favicon.ico') }}"
type="image/x-icon">
    <link rel="shortcut icon" href="{{ url_for('static', filename='favicon.ico') }}"
type="image/x-icon">
</head>
<body bgcolor="#333">
    <div class="pronav" id="mypronav">
        <h1>Developer</h1>
    </div>
    <div class="probanner">
        <div class="propic">
            {% for image in images %}
                <img src="{{ image }}" class="profile_pic">
            {% endfor %}
        </div>
        <div class="profile_text">
            <h3>Welcome, Developer {{ username }}!</h3>
            <p>Email: {{ email }}</p>
            <p>Company ID: SRIshtiAAT2023</p>
            <label>
                API_KEY<input type="text" id="text-input"
placeholder="*****************************************" value="{{ valid_api_key
}}" readonly class="api_tab">
            </label>
            <button onclick="copyText()">Copy</button><br><br>
            <a href="/logout"><button style="max-width: 150px; width: 100%; min-
width: 80px;">Log Out</button></a><br><br>
            <button onclick="showTable('table1')" style="max-width: 73px; width:
100%; min-width: 50px">Table 1</button>
            <button onclick="showTable('table2')" style="max-width: 73px; width:
100%; min-width: 50px;">Table 2</button>
        </div>
    </div>
        <div class="user_database">
            {% with messages = get_flashed_messages() %}
                {% if messages %}
                    <div class="flash-messages">
                        {% for message in messages %}
                            <div class="flash-message">{{ message }}</div>
                        {% endfor %}
                    </div>
                {% endif %}
            {% endwith %}
            <div id="table1" class="table-container active">
                <h2>Active user</h2>
                <table>
```

```
                    <tr>
                        <th>ID</th>
                        <th>Name</th>
                        <th>Gender</th>
                        <th>Email</th>
                        <th>Password</th>
                        <th>Action</th>
                    </tr>
                    {% for row in user_database %}
                        <tr>
                            <td>{{ row['id'] }}</td>
                            <td>{{ row['name'] }}</td>
                            <td>{{ row['gender'] }}</td>
                            <td>{{ row['email'] }}</td>
                            <td>{{ row['password'] }}</td>
                            <td>
                                <form method="POST" action="/delete">
                                    <input type="hidden" name="_method"
value="DELETE">
                                    <input type="hidden" name="user_id" value="{{
row['id'] }}">
                                    <input type="hidden" name="url" value="{{
valid_api_key }}">
                                    <input type="submit" value="DELETE" style="width:
100%;">
                                </form>
                            </td>
                        </tr>
                    {% endfor %}
                </table>
            </div>

            <div id="table2" class="table-container">
                <h2>Active Face ID</h2>
                <table>
                    <tr>
                        <th>Face_ID</th>
                    </tr>
                    {% set seen_ids = [] %}
                    {% for row in face_id %}
                        {% if row['face_id'] not in seen_ids %}
                            {% set _ = seen_ids.append(row['face_id']) %}
                            <tr>
                                <td>{{ row['face_id'] }}</td>
                            </tr>
                        {% endif %}
                    {% endfor %}
                </table>
            </div>
        </div>
        <script>
            function showTable(tableId) {
                // Hide all table containers
                var containers = document.getElementsByClassName('table-container');
                for (var i = 0; i < containers.length; i++) {
                    containers[i].classList.remove('active');
```

```
            }

            // Show the selected table container
            var tableContainer = document.getElementById(tableId);
            tableContainer.classList.add('active');
        }

        function copyText() {
            /* Get the input field and its value */
            var inputField = document.getElementById("text-input");
            var textToCopy = inputField.value;

            /* Create a temporary input element to hold the text */
            var tempInput = document.createElement("input");
            tempInput.type = "text";
            tempInput.value = textToCopy;
            document.body.appendChild(tempInput);

            /* Copy the text from the input element */
            tempInput.select();
            document.execCommand("copy");

            /* Remove the temporary input element */
            document.body.removeChild(tempInput);

            /* Clear the input field */
            inputField.value = "";

            /* Alert the user that the text has been copied */
            alert("Text copied: " + textToCopy);
        }
    </script>
</body>
</html>
```

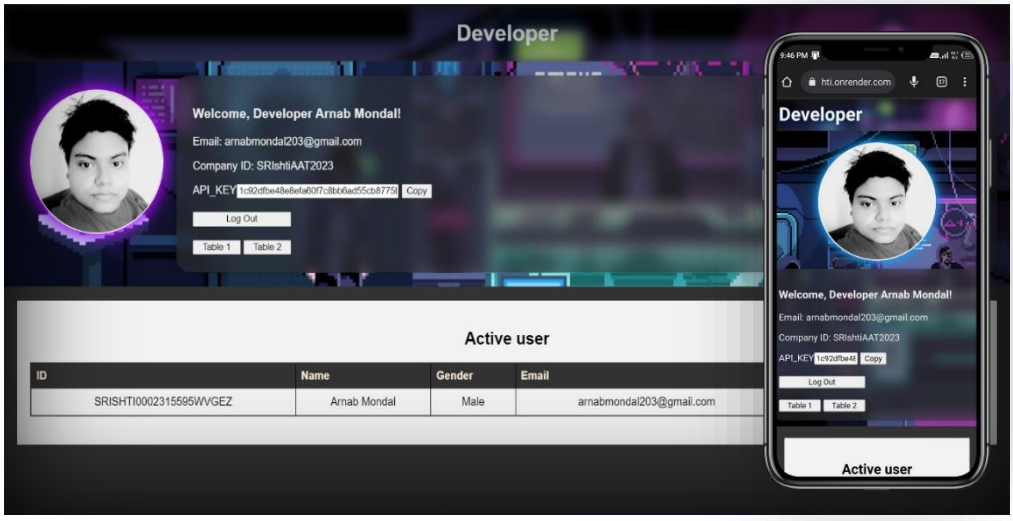


Figure 5.1.7: Profile Page

5.2 ANNEXURE II:

Project SRISHTI Code:

[1] SRIshti.py:

```
import SRI
import threading

if __name__ == "__main__":
    threading.Thread(target=SRI.dashboard).start()
```

[2] SRI.py:

```
import time
import threading
import tkinter as tk
from tkinter import ttk
from PIL import Image, ImageTk
from database_json_ROK import login, sign_up_data
import gui_test
from sound import error, accpet
from test import online

dashboard_called = False

def dashboard():
    online("Welcome to dashboard")
    def on_enter1(event):
        log_btn.config(bg='#7D3C98', fg="white")

    def on_leave1(event):
        log_btn.config(bg='#BB8FCE', fg="black")

    def on_enter2(event):
        sign_btn.config(bg='#7D3C98', fg="white")

    def on_leave2(event):
        sign_btn.config(bg='#BB8FCE', fg="black")

    def close_window():
        window.destroy()

    window = tk.Tk()
    window.title("Dashboard")
    window.resizable(False, False)
    image1 = Image.open("gui/log.png")
    resize_image1 = image1.resize((500, 500))
    img1 = ImageTk.PhotoImage(resize_image1)
    label1 = tk.Label(window, bg="#F2F4F4", image=img1)
    label1.pack(side=tk.LEFT, pady=20, padx=20)
    label1.image = img1

    frame = tk.Frame(window, bg="#AED6F1", width=500, height=500)
    frame.pack(side=tk.RIGHT, padx=20, pady=20, expand=True)
```

```
    heading = tk.Label(frame, text="🤖SRIshti", fg="#A569BD", bg="#AED6F1",
font=("Microsoft YaHei UI Light", 30, "bold"))
    heading.place(x=150, y=10)

    image2 = Image.open("gui/ai.png")
    resize_image2 = image2.resize((350, 350))
    img2 = ImageTk.PhotoImage(resize_image2)
    label2 = tk.Label(frame, bg="#AED6F1")
    label2.place(x=150, y=100, width=200, height=250)
    label2.image = img2

    log_btn = tk.Button(frame, text="Log in", bg="#BB8FCE", bd=0, command=lambda:
log_in(window))
    log_btn.place(x=140, y=400, width=100, height=50)
    log_btn.bind('<Enter>', on_enter1)
    log_btn.bind('<Leave>', on_leave1)

    sign_btn = tk.Button(frame, text="Sign Up", bg="#BB8FCE", bd=0, command=lambda:
sign_up(window))
    sign_btn.place(x=270, y=400, width=100, height=50)
    sign_btn.bind('<Enter>', on_enter2)
    sign_btn.bind('<Leave>', on_leave2)

    label1.config(image=img1)
    label2.config(image=img2)

    window.iconbitmap("gui/logo.ico")
    window.protocol("WM_DELETE_WINDOW", close_window)
    window.mainloop()

def log_in(window):
    online("open login window")
    window.destroy()
    root = tk.Tk()
    root.geometry("600x600")
    root.resizable(False, False)
    frame = tk.Label(root, bg="#CCD1D1")
    frame.pack(side=tk.TOP, expand=True, fill="both", padx=20, pady=20)
    image3 = Image.open("gui/ai.png")
    resize_image3 = image3.resize((300, 300))
    img3 = ImageTk.PhotoImage(resize_image3)
    top_img = tk.Label(frame, image=img3, width=160, height=200, bg="#CCD1D1")
    top_img.pack(side=tk.TOP, pady=20, padx=10)
    top_img.image = img3

    data = tk.Label(frame, bg="#CCD1D1", width=50)
    data.pack(side=tk.TOP, expand=True, fill="both")

    def countdown():
        def sorry_log(root):
            online("Sorry for Eruption. I think this is the fault of the Internet on
the camera. please try to log in again.")
            close_log(root)

        def close_log(root):
```

```
            root.destroy()
            dashboard()

        def on_enter_ok(event):
            okey.config(bg='#1E8449', fg="white")

        def on_leave_ok(event):
            okey.config(bg='#A9DFBF', fg="black")

        def on_enter_cancel(event):
            cancel.config(bg='#1E8449', fg="white")

        def on_leave_cancel(event):
            cancel.config(bg='#A9DFBF', fg="black")

        def on_enter_go_back(event):
            can_btn.config(bg='#B03A2E', fg="white")

        def on_leave_go_back(event):
            can_btn.config(bg='#F1948A', fg="black")

        i = 0
        data.config(text="At first I start to scan your face\nand get your Face ID",
                    font=('Microsoft Yahei UI Light', 15, "bold"), fg="#34495E")
        online("At first I start to scan your face, and get your Face ID")
        data.config(text="make sure the camera is connected\nAnd the Internet is OK
")
        online("make sure the camera is connected, And the Internet is OK")
        data.config(text="you have total 5 second to do that")
        online("you have total 5 second to do that")
        data.config(text="After 5 second\nThe password section will appear\nand you
need to provide the password\nfor your user account")
        online("After 5 second, The password section will appear, and you need to
provide the password, for your user account")
        while i < 5:
            time.sleep(0.5)
            i += 1
            data.config(text=f"{i}")
            online(i)
        data.config(text="Your face is scanning now!!!")
        online("Your face is scanning now")
        time.sleep(2)
        pack = login()
        time.sleep(2)
        lab_msg = tk.Frame(data, bg="#CCD1D1")
        lab_msg.pack(side=tk.TOP, fill="both")
        print(pack)
        if pack == None or pack == False:
            data.config(text="(ʊ˳ ʊ)", font=("Arial", 80, "bold"), fg="#CB4335")
            error()
            time.sleep(2)

            not_found = tk.Label(lab_msg)
            not_found.pack(side=tk.TOP, fill="both")
            not_found.config(text="Face ID not found", font=("Arial", 20))
            confirm_layer = tk.Frame(lab_msg, width=20)
```

```
            confirm_layer.pack(side=tk.TOP)

            can_btn = tk.Button(confirm_layer, bd=0, text="Go Back", bg="#F1948A",
width=10,
                                command=lambda: close_log(root))
            can_btn.pack(side=tk.TOP, pady=10, padx=10)
            can_btn.bind("<Enter>", on_enter_go_back)
            can_btn.bind("<Leave>", on_leave_go_back)

            data.config(text="You need to go back and log in again!!\nOr if you not
sign up in SIGN UP",
                        font=("Arial", 10, "bold"), fg="#B03A2E")
            online("Face ID not found")
            online("You need to go back and log in again, Or if you not sign up in
SIGN UP")
        else:
            userid = pack[0]
            name = pack[1]
            gender = pack[2]
            password = pack[3]
            if gender == "Male":
                add = "Sir"
            elif gender == "Female":
                add = "Ma'am"
            else:
                add = "them"
            data.config(text="(◍•ᴗ•◍)", font=("Arial", 80, "bold"), fg="#27AE60")
            accpet()
            time.sleep(2)
            data.config(text=f"Is that You, {add}?", font=("Arial", 10, "bold"),
fg="#27AE60")
            online(f"Is that You, {add}?")

            user_id = tk.Label(lab_msg)
            user_id.pack(side=tk.TOP, fill="both")
            user_id.config(text=f"Face ID: {userid}", font=("Arial", 15))

            name_id = tk.Label(lab_msg)
            name_id.pack(side=tk.TOP, fill="both")
            name_id.config(text=f"Name: {name}", font=("Arial", 12))

            confirm_layer = tk.Frame(lab_msg)
            confirm_layer.pack(side=tk.TOP)

            okey = tk.Button(confirm_layer, width=10, text="Yes", bg="#A9DFBF", bd=0,
                             command=lambda: log_in_ai(root, name, gender, password))
            okey.pack(side=tk.LEFT, pady=10, padx=10)
            okey.bind('<Enter>', on_enter_ok)
            okey.bind('<Leave>', on_leave_ok)

            cancel = tk.Button(confirm_layer, width=10, text="No", bg="#A9DFBF",
bd=0,
                               command=lambda: sorry_log(root))
            cancel.pack(side=tk.RIGHT, pady=10, padx=10)
            cancel.bind('<Enter>', on_enter_cancel)
            cancel.bind('<Leave>', on_leave_cancel)
```

```
        def log_in_ai(root, name, gender, password):

            def on_pasd_enter(e):
                pasd.delete(0, 'end')

            def on_pasd_leave(e):
                name = pasd.get()
                if name == '':
                    pasd.insert(0, 'Password')

            lab_msg.destroy()
            confirm_layer.destroy()
            data.config(text="Confirm your password")
            online("Confirm your password")
            pasd = tk.Entry(data, font=('Microsoft Yahei UI Light', 15, "bold"))
            pasd.pack(side=tk.TOP, fill="both")
            pasd.insert(0, "Password")
            pasd.bind("<FocusIn>", on_pasd_enter)
            pasd.bind("<FocusOut>", on_pasd_leave)
            check_btn = tk.Button(data, text="Confirm", command=lambda:
check_btn_sri(root, name, gender, password))
            check_btn.pack(side=tk.TOP)

            def check_btn_sri(root, name, gender, password):
                psd = pasd.get()
                if psd == password:
                    root.destroy()
                    gui_test.gui(name, gender)
                else:
                    data.config(text="Wrong password", fg="red")
                    online("Wrong password")

        top_img.config(image=img3)

    threading.Thread(target=countdown).start()
    root.mainloop()

def sign_up(window):
    online("open sign up window")
    window.destroy()

    def close_log(root):
        root.destroy()
        dashboard()

    def on_enter_ok(event):
        submit_btn.config(bg='#1E8449', fg="white")

    def on_leave_ok(event):
        submit_btn.config(bg='#A9DFBF', fg="black")

    def on_enter_cancel(event):
        cancel_btn.config(bg='#1E8449', fg="white")

    def on_leave_cancel(event):
```

```
        cancel_btn.config(bg='#A9DFBF', fg="black")

    def on_user_enter(e):
        user_name.config(fg="black")
        user_name.delete(0, 'end')

    def on_user_leave(e):
        name = user_name.get()
        if name == '':
            user_name.insert(0, 'USERNAME')

    def on_pass_enter(e):
        password_ent.config(fg="black")
        password_ent.delete(0, 'end')

    def on_pass_leave(e):
        name = password_ent.get()
        if name == '':
            password_ent.insert(0, 'PASSWORD')

    def on_email_enter(e):
        mail_ent.config(fg="black")
        mail_ent.delete(0, 'end')

    def on_email_leave(e):
        name = mail_ent.get()
        if name == '':
            mail_ent.insert(0, 'EMAIL')

    def on_selection(event):
        gender_combobox.config(foreground="black")
        selected_item = gender_combobox.get()
        print("Selected gender:", selected_item)

    root = tk.Tk()
    root.geometry("600x600")
    root.resizable(False, False)
    frame = tk.Label(root, bg="#CCD1D1")
    frame.pack(side=tk.TOP, expand=True, fill="both", padx=20, pady=20)
    image3 = Image.open("gui/ai.png")
    resize_image3 = image3.resize((300, 300))
    img3 = ImageTk.PhotoImage(resize_image3)
    top_img = tk.Label(frame, image=img3, width=160, height=200, bg="#CCD1D1")
    top_img.pack(side=tk.TOP, pady=20, padx=10)
    top_img.image = img3

    center_label = tk.Label(frame, bg="#CCD1D1")
    center_label.pack(side=tk.TOP, expand=True, fill="both")

    user_name = tk.Entry(center_label, font=('Microsoft Yahei UI Light', 10, "bold"),
bd=0, fg="#AAB7B8")
    user_name.pack(side=tk.TOP, pady=10, padx=10, fill="both")
    user_name.insert(0, "USERNAME")
    user_name.bind("<FocusIn>", on_user_enter)
    user_name.bind("<FocusOut>", on_user_leave)
```

```
    password_ent = tk.Entry(center_label, font=('Microsoft Yahei UI Light', 10,
"bold"), bd=0, fg="#AAB7B8")
    password_ent.pack(side=tk.TOP, pady=10, padx=10, fill="both")
    password_ent.insert(0, "PASSWORD")
    password_ent.bind("<FocusIn>", on_pass_enter)
    password_ent.bind("<FocusOut>", on_pass_leave)

    gender_combobox = ttk.Combobox(center_label, values=["Male", "Female", "Other"],
                                   font=('Microsoft Yahei UI Light', 10, "bold"),
foreground="#AAB7B8")
    gender_combobox.pack(side=tk.TOP, pady=10, padx=10, fill="both")
    gender_combobox.set("GENDER")
    gender_combobox.bind("<<ComboboxSelected>>", on_selection)

    mail_ent = tk.Entry(center_label, font=('Microsoft Yahei UI Light', 10, "bold"),
bd=0, fg="#AAB7B8")
    mail_ent.pack(side=tk.TOP, pady=10, padx=10, fill="both")
    mail_ent.insert(0, "EMAIL")
    mail_ent.bind("<FocusIn>", on_email_enter)
    mail_ent.bind("<FocusOut>", on_email_leave)

    bottom_label = tk.Label(center_label, bg="#CCD1D1", height=20)
    bottom_label.pack(side=tk.TOP)

    submit_btn = tk.Button(bottom_label, text="Submit", font=('Microsoft Yahei UI
Light', 10, "bold"), bd=0,
                           bg='#A9DFBF', fg="black", command=lambda: submit_sign())
    submit_btn.pack(side=tk.LEFT, padx=10, pady=10)
    submit_btn.bind("<Enter>", on_enter_ok)
    submit_btn.bind("<Leave>", on_leave_ok)

    def submit_sign():
        username = user_name.get()
        password = password_ent.get()
        gender = gender_combobox.get()
        mail = mail_ent.get()

        if username == "USERNAME":
            user_name.config(fg="red")

        elif password == "PASSWORD":
            password_ent.config(fg="red")

        elif gender == "GENDER":
            gender_combobox.config(foreground="red")

        elif mail == "EMAIL":
            mail_ent.config(fg="red")

        else:
            sign_up_data(username, gender, password, mail, root)

    cancel_btn = tk.Button(bottom_label, text="Cancel", font=('Microsoft Yahei UI
Light', 10, "bold"), bd=0,
                           bg='#A9DFBF', fg="black", command=lambda: close_log(root))
    cancel_btn.pack(side=tk.RIGHT, padx=10, pady=10)
```

```
    cancel_btn.bind("<Enter>", on_enter_cancel)
    cancel_btn.bind("<Leave>", on_leave_cancel)

    top_img.config(image=img3)
    root.mainloop()

# if __name__ == "__main__":
#     threading.Thread(target=dashboard).start()
```

[3] database_json_ROK.py:

```
import threading
import time
import tkinter as tk
import requests
from face_recog import register_face, recognize_face
import random
import string

url = 'https://webpage-srishti.onrender.com/users_data'
valid_api_key = '1c92dfbe48e8efa60f7c8bb6ad55cb8775b267ebdbc1b4e2f5973f2d5a0062a5'
headers = {
    'X-API-Key': valid_api_key
}

def sign_up_data(name, gender, password, mail, root):
    def generate_random_code(length):
        digits = ''.join(random.choices(string.digits, k=length - 5))
        characters = ''.join(random.choices(string.ascii_uppercase, k=5))
        code = "SRISHTI00023" + digits + characters
        return code
    data = requests.get(url, headers=headers)
    print(data.status_code)
    if data.status_code == 200:
        datas = data.json()
        ids = []
        for person in datas:
            id = person['id']
            ids.append(id)
        while True:
            random_number = generate_random_code(10)
            if random_number not in ids:
                new_id = random_number
                break
            else:
                continue
        face = recognize_face()
        print(face)
        if face is None:
            register_face(new_id)
            print(gender)
            print(mail)
            print(password)
            data = {
                "id": new_id,
                "name": name,
```

```
                "gender": gender,
                "password": password,
                "email": mail
            }
            res = requests.post(url, headers=headers, data=data)
            print(res.status_code)

            if res:
                print("Database add successfully")
                root.destroy()
                open_sign(new_id, name, password, gender, mail)
            else:
                # Request failed
                print('Fail to add database, Error Code:')

    else:
        root.destroy()
        close_sign()

def open_sign(new_id, name, password, gender, mail):
    def go_back():
        root.destroy()
        import SRI
        SRI.dashboard()
    root = tk.Tk()
    root.title("Thank you for signing")
    root.geometry("1000x400")
    root.resizable(False, False)
    frame = tk.Frame(root)
    frame.pack(side=tk.TOP, expand=True, fill="both", pady=25, padx=25)
    center = tk.Label(frame)
    center.pack(side=tk.TOP, expand=True, fill="both")
    center.config(text=f"Your Face ID: {new_id}", font=("Microsoft YaHei UI Light",
30, "bold"))
    center_name = tk.Label(frame)
    center_name.pack(side=tk.TOP)
    center_name.config(text=f"Name: {name}", font=("Microsoft YaHei UI Light", 15,
"bold"))
    center_thank = tk.Label(frame)
    center_thank.pack(side=tk.TOP)
    center_thank.config(text="Thank You", font=("Microsoft YaHei UI Light", 15,
"bold"))
    back_btn = tk.Button(frame, text="Go Back", command=lambda: go_back())
    back_btn.pack(side=tk.TOP)
    root.mainloop()

def close_sign():
    def go_back():
        root.destroy()
        import SRI
        SRI.dashboard()
    root = tk.Tk()
    root.title("You are Already signing up")
    root.geometry("1000x400")
    root.resizable(False, False)
    frame = tk.Frame(root)
```

```
    frame.pack(side=tk.TOP, expand=True, fill="both", padx=25, pady=25)
    center = tk.Label(frame)
    center.pack(side=tk.TOP, expand=True, fill="both")
    center.config(text="You are already sign up", font=("Microsoft YaHei UI Light",
30, "bold"))
    back_btn = tk.Button(root, text="Go Back", command=go_back)
    back_btn.pack(side=tk.TOP)
    root.mainloop()
def login():
    face = recognize_face()
    print(face)
    try:
        if face is None:
            pack = None
            return pack
        else:
            data = requests.get(url, headers=headers)
            datas = data.json()
            for person in datas:
                id = person['id']
                if id == face:
                    name = person['name']
                    gender = person['gender']
                    password = person['password']
            # print(name)
            pack = f"{face},{name},{gender},{password}"
            # print(pack)
            pack = pack.split(",")
            return pack
    except Exception as e:
        return False
```

[4] face_recog.py:

```
import cv2
import numpy as np
import face_recognition
import psycopg2

# Connect to the PostgreSQL database
DATABASE_URL =
'postgres://srishti_database_ai_user:JaYaL1A92lAp0ikj0RxGjgKihQ3etVWj@dpg-
cic47a95rnuk9qb0sbc0-a.oregon-postgres.render.com/srishti_database_ai'

# Create a table to store face data if it doesn't exist
create_table_query = '''
    CREATE TABLE IF NOT EXISTS test_user (
        id SERIAL PRIMARY KEY,
        name TEXT,
        encoding BYTEA
    )
'''

# Register a face and save the data to the database
def register_face(name):
    # Open the webcam
```

```
    video_capture = cv2.VideoCapture(0)

    # Initialize a counter to keep track of the number of images captured
    counter = 0

    # Connect to the database
    conn = psycopg2.connect(DATABASE_URL)
    cursor = conn.cursor()

    # Execute the create table query
    cursor.execute(create_table_query)

    # Keep capturing images until 50 images are captured or 'q' key is pressed
    while counter < 50:
        # Capture a frame from the webcam
        ret, frame = video_capture.read()

        # Convert the image to RGB format
        rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

        # Find the face location in the image
        face_locations = face_recognition.face_locations(rgb_frame)

        # If there is a face in the image, save it to the database
        if len(face_locations) > 0:
            # Encode the face image
            face_encoding = face_recognition.face_encodings(rgb_frame,
face_locations)[0]

            # Save the face data to the database
            cursor.execute('INSERT INTO test_user (name, encoding) VALUES (%s, %s)',
(name, face_encoding.tobytes()))

            # Increment the counter
            counter += 1

        # Display the image with the face location marked
        for (top, right, bottom, left) in face_locations:
            cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)
        cv2.imshow("Face Registration", frame)

        # Wait for 'q' key to be pressed to exit
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    # Commit the changes to the database
    conn.commit()

    # Release the webcam
    video_capture.release()
    cv2.destroyAllWindows()

    # Close the cursor and connection
    cursor.close()
    conn.close()
```

```
# Recognize faces from the database
def recognize_face():
    # Open the webcam
    video_capture = cv2.VideoCapture(0)
    count = 0

    # Connect to the database
    conn = psycopg2.connect(DATABASE_URL)
    cursor = conn.cursor()

    # Execute the create table query
    cursor.execute(create_table_query)

    # Loop over each frame from the webcam
    while True:
        # Capture a frame from the webcam
        ret, frame = video_capture.read()

        # Convert the image to RGB format
        rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

        # Find the face locations in the image
        face_locations = face_recognition.face_locations(rgb_frame)

        # Encode the faces in the image
        face_encodings = face_recognition.face_encodings(rgb_frame, face_locations)

        # Loop over each face in the image
        for face_encoding, face_location in zip(face_encodings, face_locations):
            # Query the database to find matching faces
            cursor.execute('SELECT name, encoding FROM test_user')
            rows = cursor.fetchall()

            # Compare the face encoding to the known face encodings
            for name, encoding_blob in rows:
                # Convert the encoding from blob to numpy array
                face_encoding_db = np.frombuffer(encoding_blob, dtype=np.float64)

                # Compare the face encoding with the encoding from the database
                matches = face_recognition.compare_faces([face_encoding_db],
face_encoding)

                # If the face is recognized, show the name and break from the loop
                if matches[0]:
                    return name
                else:
                    return None
            break
        break

    # Release the webcam and close the window
    video_capture.release()
    cv2.destroyAllWindows()

# Register a face and save the data to the database
# register_face("JAT")
```

```
# Recognize faces from the database
# recognize_face()
```

[5] gui_test.py:

```
import tkinter as tk
import threading
from pippa import command

def close_program(root):
    root.quit()

def gui(name, gender):
    root = tk.Tk()
    root.title("SRIshti AI For college Project- ONLINE")
    root.configure(bg="#85929E")
    root.protocol("WM_DELETE_WINDOW", close_program(root))
    head_lable = tk.Label(root, text="🤖SRIshti", font=("Arial", 120, "bold"),
background="#85929E", fg="#117A65")
    head_lable.pack(side=tk.TOP, fill="both")
    centerFrame = tk.Label(root, bd=0)
    centerFrame.pack(fill='both', expand=True)
    scrollbar = tk.Scrollbar(centerFrame)
    scrollbar.pack(side='right', fill='y')
    textarea = tk.Text(centerFrame, font=('Arial', 15, 'bold'), height=10,
yscrollcommand=scrollbar.set,
                       wrap='word', background="#283747", fg="white", pady=4, padx=4,
bd=0)
    textarea.pack(side=tk.LEFT, fill="both", expand=True)
    scrollbar.config(command=textarea.yview)
    bottom_lable = tk.Label(root, text="welcome", font=("Arial", 15, "bold"),
background="white", fg="black")
    bottom_lable.pack(side=tk.BOTTOM, fill="both")
    threading.Thread(target=command, args=(root, head_lable, textarea, bottom_lable,
name, gender)).start()
    root.mainloop()

# gui("Arnab", "Male")
```

[6] pippa.py:

```
import sys
import torch
import random
import requests
from Brain import NeuralNet
from test import takeCommand, speak, online, wish_us
from NeuralNetwork import bag_of_word, tokenize
from task import NonInputExecution, InputExecution, exit_open

exit_thread = False

def ai_model():
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
    url = 'https://webpage-srishti.onrender.com/data'
```

```
    response = requests.get(url)
    intents = response.json()

    FILE = 'saved_model/TrainData.pth'
    data = torch.load(FILE)

    input_size = data['input_size']
    hidden_size = data['hidden_size']
    output_size = data["output_size"]
    all_words = data["all_words"]
    tags = data["tags"]
    model_state = data["model_state"]

    model = NeuralNet(input_size, hidden_size, output_size).to(device)
    model.load_state_dict(model_state)
    model.eval()

    return model, intents, all_words, tags, device

def load_model(text, model, intents, all_words, tags, device, root, head_lable,
textarea, bottom_lable, name):
    query = str(text)
    if text is not None:
        text = tokenize(text)
        X = bag_of_word(text, all_words)
        X = X.reshape(1, X.shape[0])
        X = torch.from_numpy(X).to(device)

        output = model(X)

        _, predicted = torch.max(output, dim=1)

        tag = tags[predicted.item()]

        probs = torch.softmax(output, dim=1)
        prob = probs[0][predicted.item()]
        if prob.item() > 0.9:
            for intent in intents:
                if tag == intent['tag']:
                    print(tag)
                    reply = random.choice(intent['responses'])
                    print(reply)
                    if reply is not None:
                        if "none" in reply:
                            speak("Sorry I don't understand what you're saying.",
head_lable, textarea, bottom_lable)
                            speak("say that again please", head_lable, textarea,
bottom_lable)
                        elif "time" in reply:
                            NonInputExecution(reply, head_lable, textarea,
bottom_lable)
                        elif "date" in reply:
                            NonInputExecution(reply, head_lable, textarea,
bottom_lable)
                        elif "face" in reply:
```

```
                            NonInputExecution(reply, head_lable, textarea,
bottom_lable)
                        elif "register" in reply:
                            NonInputExecution(reply, head_lable, textarea,
bottom_lable)
                        elif "location" in reply:
                            NonInputExecution(reply, head_lable, textarea,
bottom_lable)
                        elif "weather" in reply:
                            NonInputExecution(reply, head_lable, textarea,
bottom_lable)
                        elif "exit" in reply:
                            exit_open(root, head_lable, textarea, bottom_lable)
                        elif "wikipedia" in reply:
                            InputExecution(reply, query, head_lable, textarea,
bottom_lable)
                        elif "search" in reply:
                            InputExecution(reply, query, head_lable, textarea,
bottom_lable)
                        elif "yes" in tag:
                            speak(reply, head_lable, textarea, bottom_lable)
                        elif "no" in tag:
                            speak(reply, head_lable, textarea, bottom_lable)
                        elif "youtube" in reply:
                            InputExecution(reply, query, head_lable, textarea,
bottom_lable)
                        elif "chat" in reply:
                            InputExecution(reply, query, head_lable, textarea,
bottom_lable)
                        elif "code" in reply:
                            InputExecution(reply, query, head_lable, textarea,
bottom_lable)
                        elif "news" in reply:
                            NonInputExecution(reply, head_lable, textarea,
bottom_lable)
                        elif "log out" in reply:
                            speak(f"Okey {name}, log out", head_lable, textarea,
bottom_lable)
                            speak("SRIshti offline", head_lable, textarea,
bottom_lable)
                            speak("Just close it it automatic logout", head_lable,
textarea, bottom_lable)
                            root.destroy()
                            sys.exit()
                        else:
                            speak(reply, head_lable, textarea, bottom_lable)


def command(root, head_lable, textarea, bottom_lable, name, gender):
    model, intents, all_words, tags, device = ai_model()
    online("srishti online")
    wish_us(gender, head_lable, textarea, bottom_lable)
    while True:
        text = takeCommand(head_lable, textarea, bottom_lable)
        load_model(text, model, intents, all_words, tags, device, root, head_lable,
textarea, bottom_lable, name)
```

[7] task.py:

```
import time
from database_json_ROK import login
import openai
import os
import pywhatkit
import wikipedia
import requests
import webbrowser as web
from test import takeCommand, speak
from prediction import hotword
import datetime
from plyer import notification
from news import news

#############TASK#############
#############Non Input Functions############
def exit_open(root, head_lable, textarea, bottom_lable):
    hour = int(datetime.datetime.now().hour)
    if hour >= 17:
        speak("Okay Sir, Goodnight", head_lable, textarea, bottom_lable)
        speak("Lets talk next day", head_lable, textarea, bottom_lable)
    else:
        speak("Okay Sir", head_lable, textarea, bottom_lable)
        speak("Have a nice day", head_lable, textarea, bottom_lable)
    root.protocol("WM_DELETE_WINDOW", root.withdraw())
    notification.notify(
        title="Srishti",
        message="To wake up me you need to call my name",
        app_icon="srishti.ico",
        timeout=5
    )
    time.sleep(2)
    while True:
        hot = hotword()
        if hot == True:
            break
    # while True:
    #     key = hotword()
    #     if key == "wake up srishti" or key == "weke up drishti":
    #         break
    root.protocol("WM_DELETE_WINDOW", root.deiconify())
    time.sleep(1)
    speak("Welcome back sir", head_lable, textarea, bottom_lable)
    speak("how may I help you", head_lable, textarea, bottom_lable)


def Time(head_lable, textarea, bottom_lable):
    time = datetime.datetime.now().strftime('%H:%M')
    speak(f"The Current time is {time}", head_lable, textarea, bottom_lable)

def Date(head_lable, textarea, bottom_lable):
    date = datetime.date.today()
    day = date.strftime("%A")
    speak(f"The Current date is {date}", head_lable, textarea, bottom_lable)
```

```
    speak(f"Today is {day}", head_lable, textarea, bottom_lable)

def WhoAmI(head_lable, textarea, bottom_lable):
    try:
        name = login()
        speak(f"I think you are {name}", head_lable, textarea, bottom_lable)
    except Exception as e:
        print(e)

def location(head_lable, textarea, bottom_lable):
    r = requests.get("https://get.geojs.io/")
    ip_request = requests.get("https://get.geojs.io/v1/ip.json")
    ipAdd = ip_request.json()['ip']

    url = "https://get.geojs.io/v1/ip/geo/" + ipAdd + ".json"
    geo_request = requests.get(url)
    geo_data = geo_request.json()
    speak(f"I think we are in {geo_data['country']}'s city {geo_data['city']},
{geo_data['region']} region the timezone {geo_data['timezone']}", head_lable,
textarea, bottom_lable)
    speak(f"latitude :{geo_data['latitude']}", head_lable, textarea, bottom_lable)
    speak(f"longitude :{geo_data['longitude']}", head_lable, textarea, bottom_lable)
    speak("anythink else, you want to know?", head_lable, textarea, bottom_lable)

def weather(head_lable, textarea, bottom_lable):
    r = requests.get("https://get.geojs.io/")
    ip_request = requests.get("https://get.geojs.io/v1/ip.json")
    ipAdd = ip_request.json()['ip']
    url = "https://get.geojs.io/v1/ip/geo/" + ipAdd + ".json"
    geo_request = requests.get(url)
    geo_data = geo_request.json()
    city = geo_data['city']
    key = "8b9272a4f84f4c63906130146232105"
    BASE_URL =
f"https://api.weatherapi.com/v1/current.json?key={key}&q={city}&aqi=yes"
    w_url = BASE_URL
    w_request = requests.get(w_url)
    w_data = w_request.json()
    speak(f"The current weather in Fahrenheit {w_data['current']['temp_f']} and
Celsius is {w_data['current']['temp_c']}", head_lable, textarea, bottom_lable)
    speak(f"humidity is {w_data['current']['humidity']}, wind speed is
{w_data['current']['wind_kph']}kilometer per Hours", head_lable, textarea,
bottom_lable)
    speak(f"pressure {w_data['current']['pressure_mb']}", head_lable, textarea,
bottom_lable)
    speak("Anythink else sir", head_lable, textarea, bottom_lable)

def news_report(head_lable, textarea, bottom_lable):
    news_data = news()
    speak("Today's top five news are", head_lable, textarea, bottom_lable)
    for key, value in list(news_data.items())[:5]:
        speak(f"{key}. {value['title'][0]}", head_lable, textarea, bottom_lable)


def newUser(head_lable, textarea, bottom_lable):
```

```
    speak("To login new user you need to log out then log in or create a new user",
head_lable, textarea, bottom_lable)

##############Input Function##############
def Wikipedia(query, head_lable, textarea, bottom_lable):
    try:
        query = str(query).replace("who is", "").replace("about", "").replace("tell
me about", "").replace("what is", "").replace("wikipedia", "")
        len_query = len(query.replace(" ", ""))
        if len_query > 0:
            result = wikipedia.summary(query)
        else:
            speak("Tell me your query in wikipedia", head_lable, textarea,
bottom_lable)
            question = takeCommand()
            result = wikipedia.summary(question)
        speak(result, head_lable, textarea, bottom_lable)

    except Exception as e:
        speak('Wikipedia occurs an error', head_lable, textarea, bottom_lable)

def google(query, head_lable, textarea, bottom_lable):
    query = str(query).replace("google", "").replace("search", "").replace("about",
"")
    len_query = len(query.replace(" ", ""))
    if len_query > 0:
        pywhatkit.search(query)
        speak("Here is search result", head_lable, textarea, bottom_lable)
    else:
        speak("what should I search?", head_lable, textarea, bottom_lable)
        question = takeCommand(head_lable, textarea, bottom_lable)
        speak("Here is search result", head_lable, textarea, bottom_lable)
        pywhatkit.search(question)

def youtube(query, head_lable, textarea, bottom_lable):
    ans = str(query)
    query = str(query).replace("youtube", "").replace("play video",
"").replace("video about", "").replace("latest video", "").replace("music",
"").replace("videos", "").replace("play songs", "")
    len_query = len(query.replace(" ", ""))
    if len_query > 0:
        if 'videos' in ans:
            result = f"https://www.youtube.com/results?search_query={query}"
            speak("Here is search result", head_lable, textarea, bottom_lable)
            web.open(result)
        else:
            speak("Here is search result", head_lable, textarea, bottom_lable)
            pywhatkit.playonyt(query)
    else:
        speak("Tell me the video name do you want to search", head_lable, textarea,
bottom_lable)
        quest = takeCommand(head_lable, textarea, bottom_lable)
        speak("Here is search result", head_lable, textarea, bottom_lable)
        quest = str(quest)
        if 'random' in quest:
```

```
            quest = quest.replace("random", "").replace("play", "").replace("video",
"")
            pywhatkit.playonyt(quest)
        elif "videos" in ans:
            quest = quest.replace("videos", "")
            result = f"https://www.youtube.com/results?search_query={quest}"
            web.open(result)
        elif "play songs" in ans:
            quest = quest.replace("play songs", "")
            pywhatkit.playonyt(quest)
        else:
            pywhatkit.playonyt(quest)

def open_ai(query, head_lable, textarea, bottom_lable):
    API_KEY = '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'
    prompt = query
    os.environ['OPENAI_Key'] = API_KEY
    openai.api_key = os.environ['OPENAI_Key']
    response = openai.Completion.create(engine='text-davinci-003', prompt=prompt,
max_tokens=50, temperature=0.7)
    ans = (response['choices'][0]['text'])
    speak(ans, head_lable, textarea, bottom_lable)

def write_program(query, head_lable, textarea, bottom_lable):
    API_KEY = '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'
    query = query.replace('write a program', '').replace('write a code',
'').replace('program', '').replace('write a', '')
    print(query)
    print(len(query))
    if len(query) == 0:
        speak("What programme should I write and what type of program should i
write?", head_lable, textarea, bottom_lable)
        query = takeCommand(head_lable, textarea, bottom_lable)
        query = str(query)
    else:
        query = str(query)
    prompt = query
    os.environ['OPENAI_Key'] = API_KEY
    openai.api_key = os.environ['OPENAI_Key']
    response = openai.Completion.create(engine='text-davinci-003', prompt=prompt,
max_tokens=50, temperature=0.7)
    ans = (response['choices'][0]['text'])
    # speak(ans, head_lable, textarea, bottom_lable)
    speak('tell me the program name that you want to save....', head_lable, textarea,
bottom_lable)
    name = takeCommand(head_lable, textarea, bottom_lable)
    if 'python' in query:
        exten = 'py'
    elif 'java' in query:
        if 'public class' in ans:
            ans = ans.replace('public class', 'class')
        exten = 'java'
    elif 'c' in query:
        exten = 'c'
    else:
        exten = 'txt'
```

```
    file = open(f'C:\\Users\\arnab\\OneDrive\\Documents\\SRISHTI MAKE
CODE\\{name}.{exten}', 'w')
    file.write(ans)
    speak(f'The path where the program save ->
C:\\Users\\arnab\\OneDrive\\Documents\\SRISHTI MAKE CODE\\{name}.{exten}',
head_lable, textarea, bottom_lable)

def NonInputExecution(query, head_lable, textarea, bottom_lable):

    query = str(query)

    if "time" in query:
        Time(head_lable, textarea, bottom_lable)
    elif "date" in query:
        Date(head_lable, textarea, bottom_lable)
    elif "face" in query:
        WhoAmI(head_lable, textarea, bottom_lable)
    elif "register" in query:
        newUser(head_lable, textarea, bottom_lable)
    elif "location" in query:
        location(head_lable, textarea, bottom_lable)
    elif "weather" in query:
        weather(head_lable, textarea, bottom_lable)
    elif "news" in query:
        news_report(head_lable, textarea, bottom_lable)


def InputExecution(tag, query, head_lable, textarea, bottom_lable):
    query = str(query)

    if 'wikipedia' in tag:
        Wikipedia(query, head_lable, textarea, bottom_lable)

    elif 'search' in tag:
        google(query, head_lable, textarea, bottom_lable)

    elif 'youtube' in tag:
        youtube(query, head_lable, textarea, bottom_lable)

    elif 'chat' in tag:
        open_ai(query, head_lable, textarea, bottom_lable)

    elif 'code' in tag:
        write_program(query, head_lable, textarea, bottom_lable)
```

There are some files like test.py, Brain.py, NeuralNetwork.py, PreparingData.py, PreprocessingData.py etc. Too train model we make training_nurans.py to Neurons of the brain of our AI, training.py it uses train our AI to understand a specific voice that make him wake up every time we call it. If you want to check out those code, go to this website https://webpage-srishti.onrender.com/github or scan this QR Code too redirect to our GitHub account.

Figure 5.2.1: Download Project code

CHAPTER -6: How normal users can access our project.

Normal people can access our project by downloading from the following website are described below:

- Website Link: https://webpage-srishti.onrender.com.

Scan the QR code:

System Requirement for SRISHTI AI

Before you start SRISHTI AI free download make sure your pc meets the minimum system requirements: -

- Memory (RAM): 4 GB of RAM required
- Hard Disk Space: 64 GB of free space required
- Processor: Intel Dual Core or higher processor

CHAPTER -7: RESULT AND DISCUSSION

7.1 Result and Discussion

Figure 7.1 Describes the first section of the home page of the website where we can get the “Register” page as a hyperlink in the “Download” button.

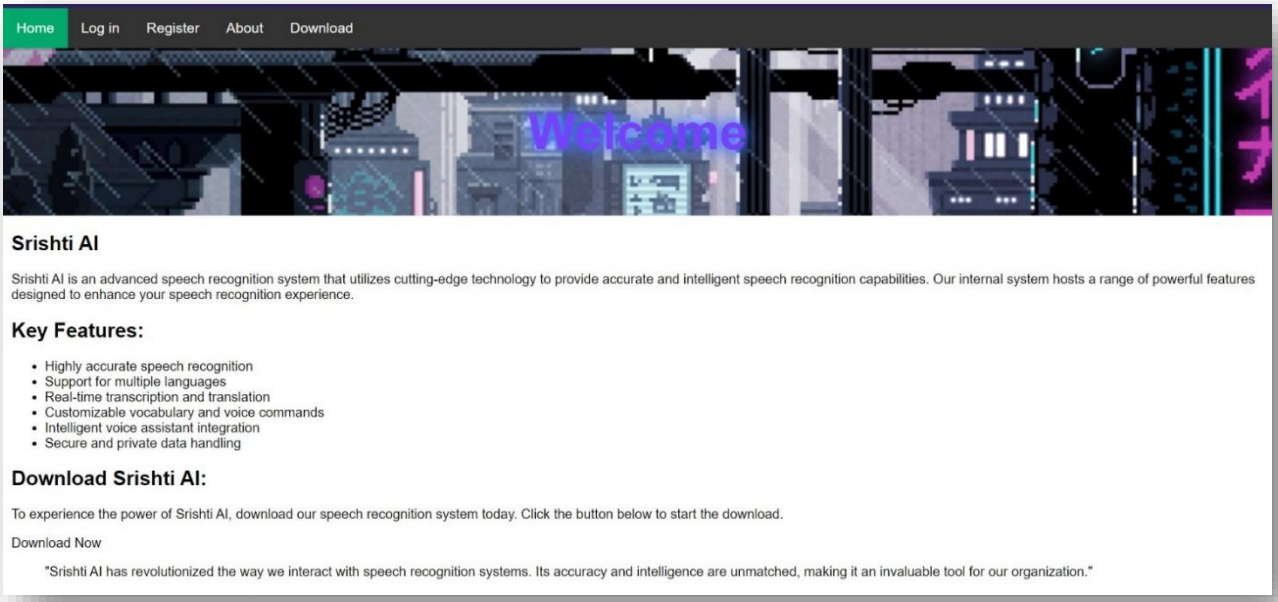


Figure 7.1

Figure 7.2 describes the second section of the homepage of the website where we can get about “Privacy and Security”, and Continuous Improvement, and the last section of the homepage of the website where we can see the footer of the website where terms and conditions are available.

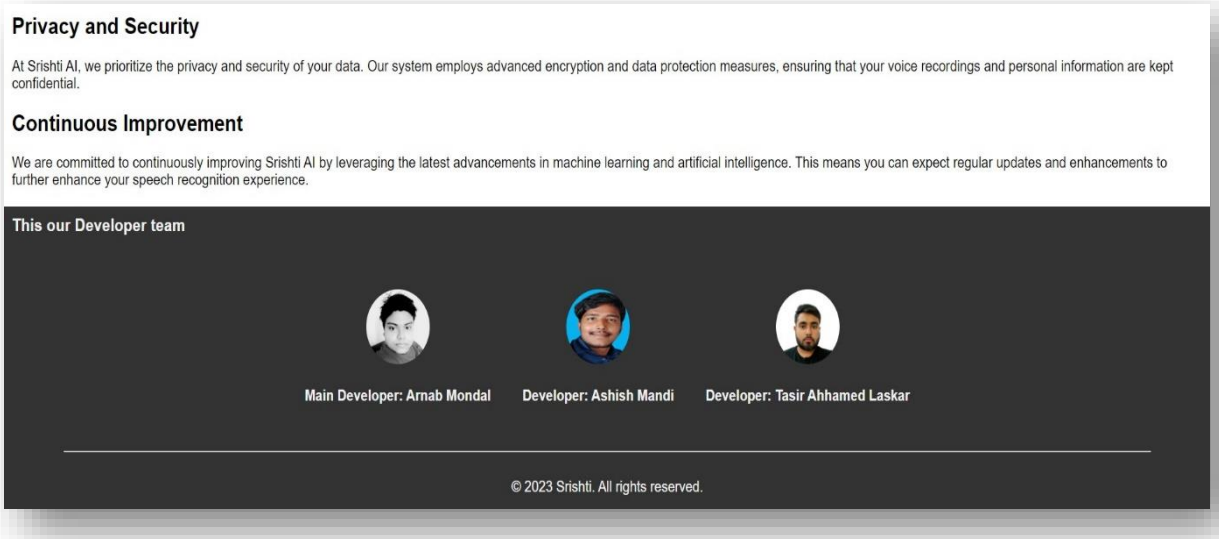


Figure 7.3 Describes the looks of the “Developer Log In” page of the website. This page is only for the developer’s login not for any other user.

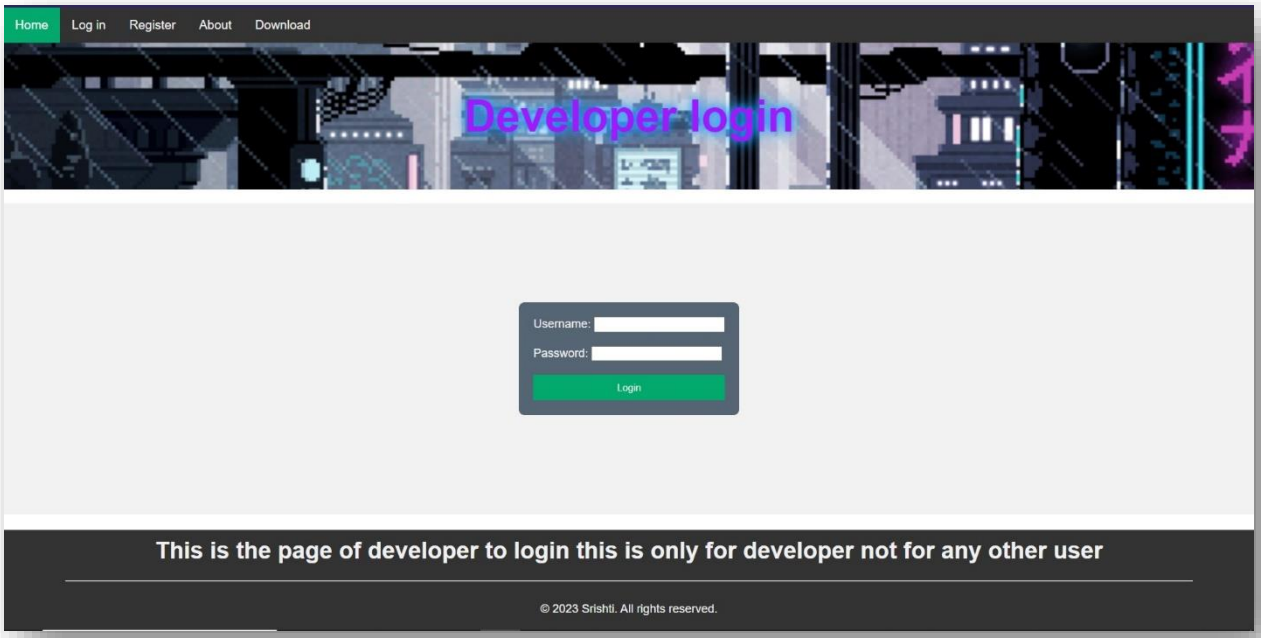


Figure 7.3

Figure 7.4 Describes the looks of the “Developer Register” page of the website.

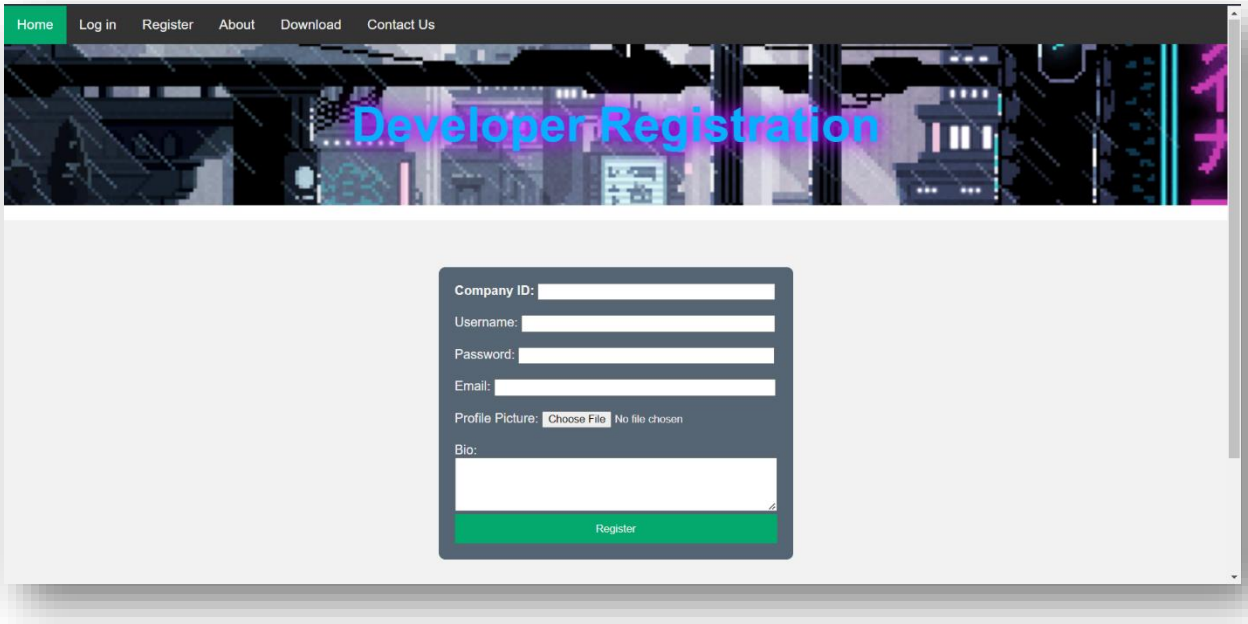


Figure 7.4

Figure 7.5 Describes the looks of the “About” page of the website. This page is described the developer of this software.

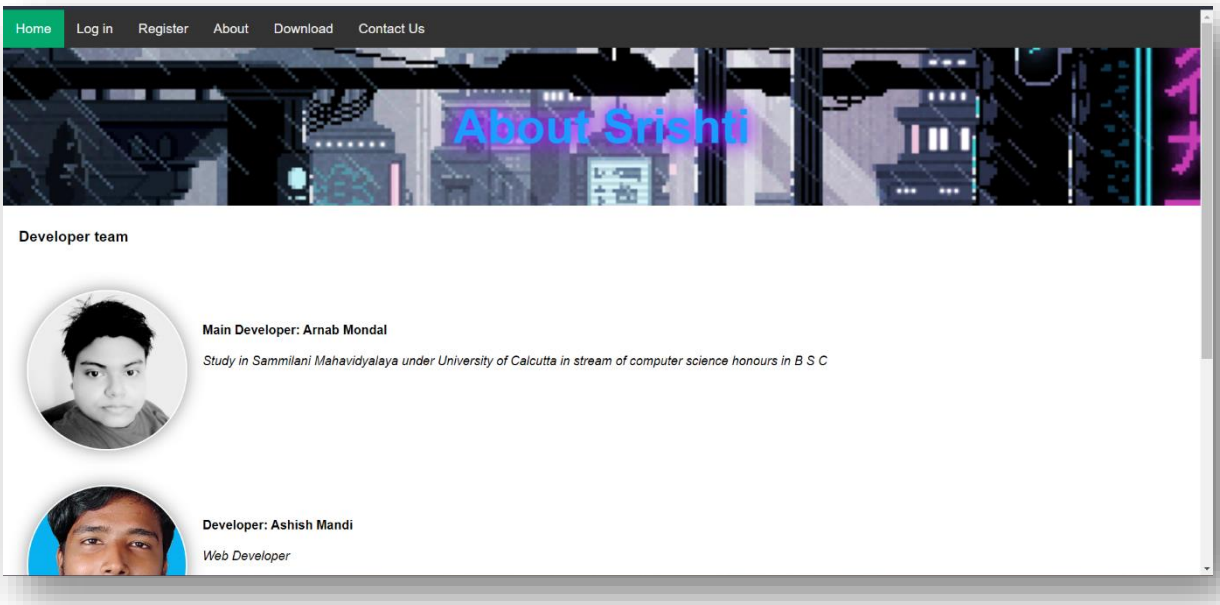


Figure 7.5

Figure 7.6 Describes the looks of the “Download Page” page of the website.

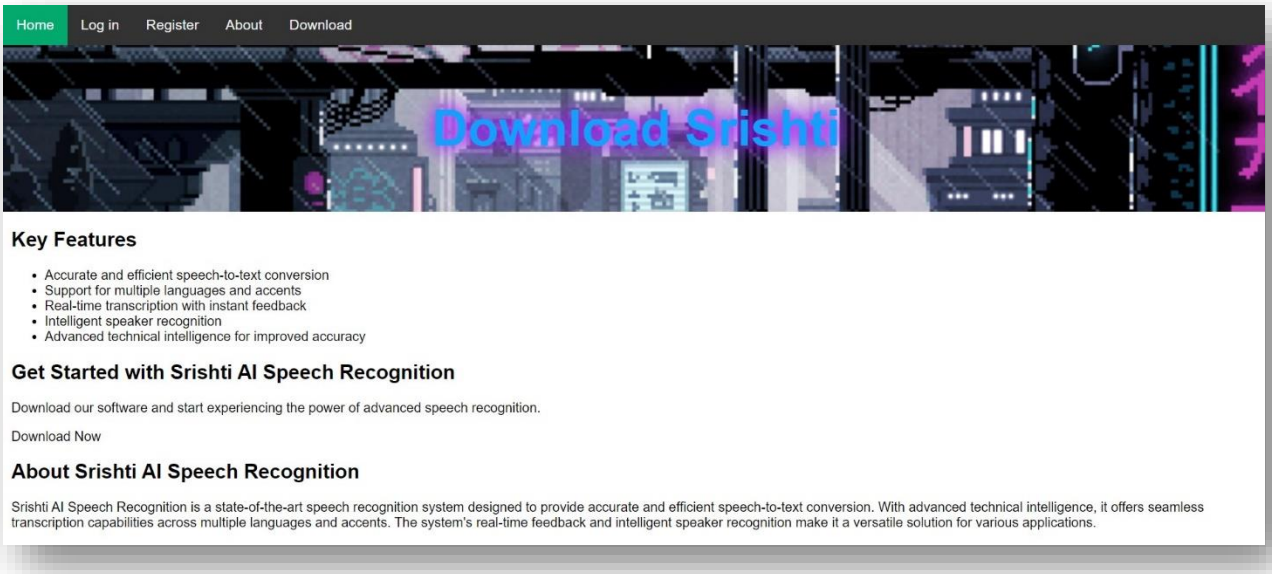


Figure 7.6

Figure 7.8 Describes the looks of the Email received after submitting the Developer Registration Form" page of the website.

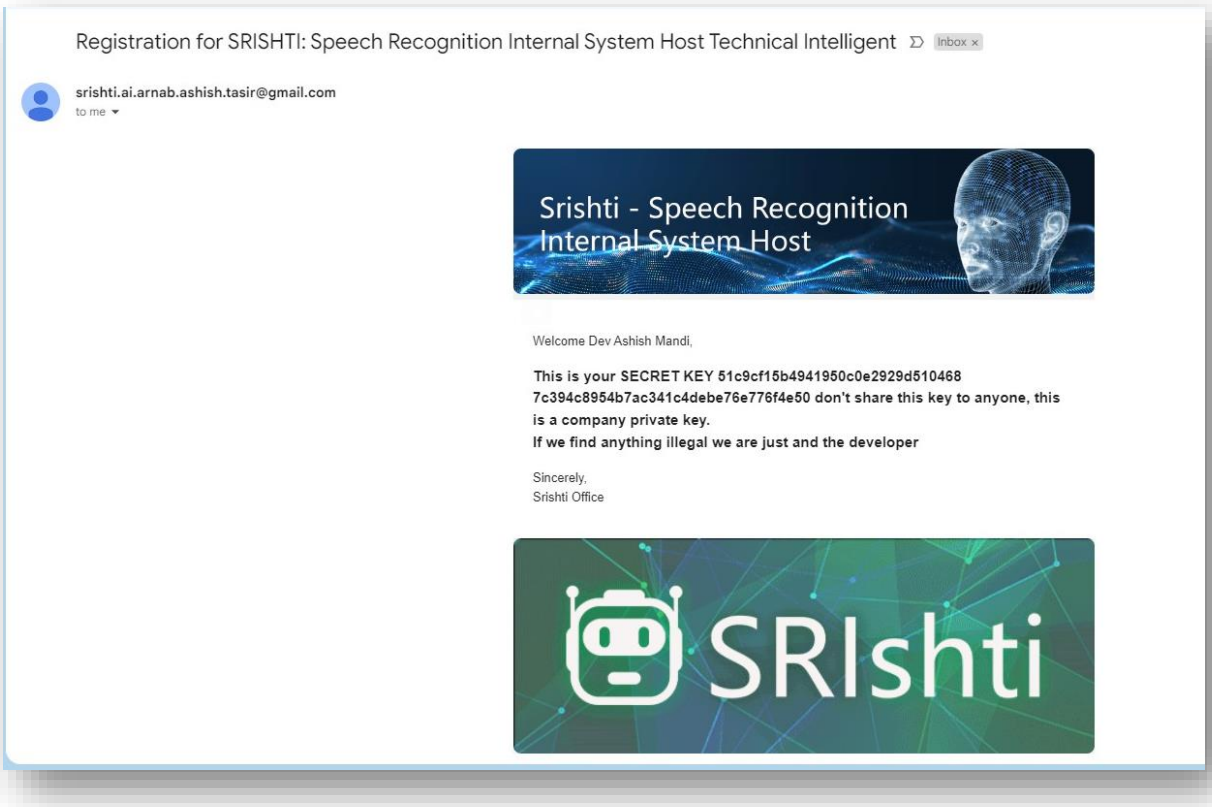


Figure 7.8

Figure 7.9 After the software download the interface look like this.

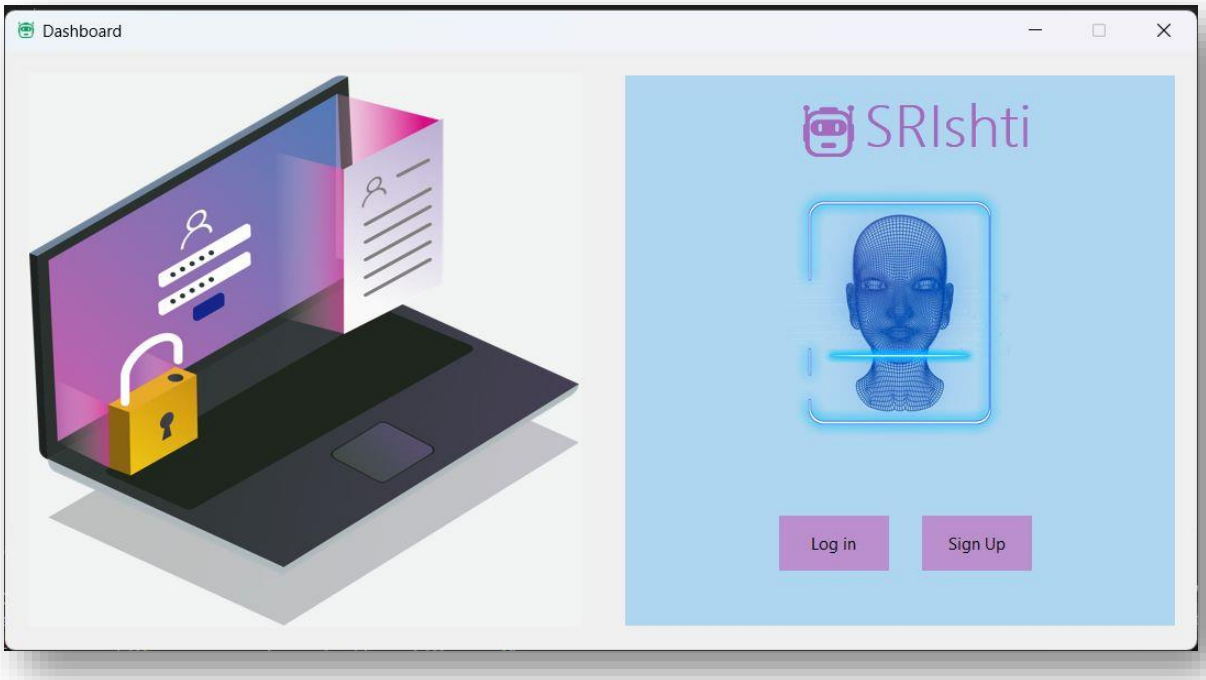


Figure 7.9

Figure 7.10 Describes the Sign-Up Page.

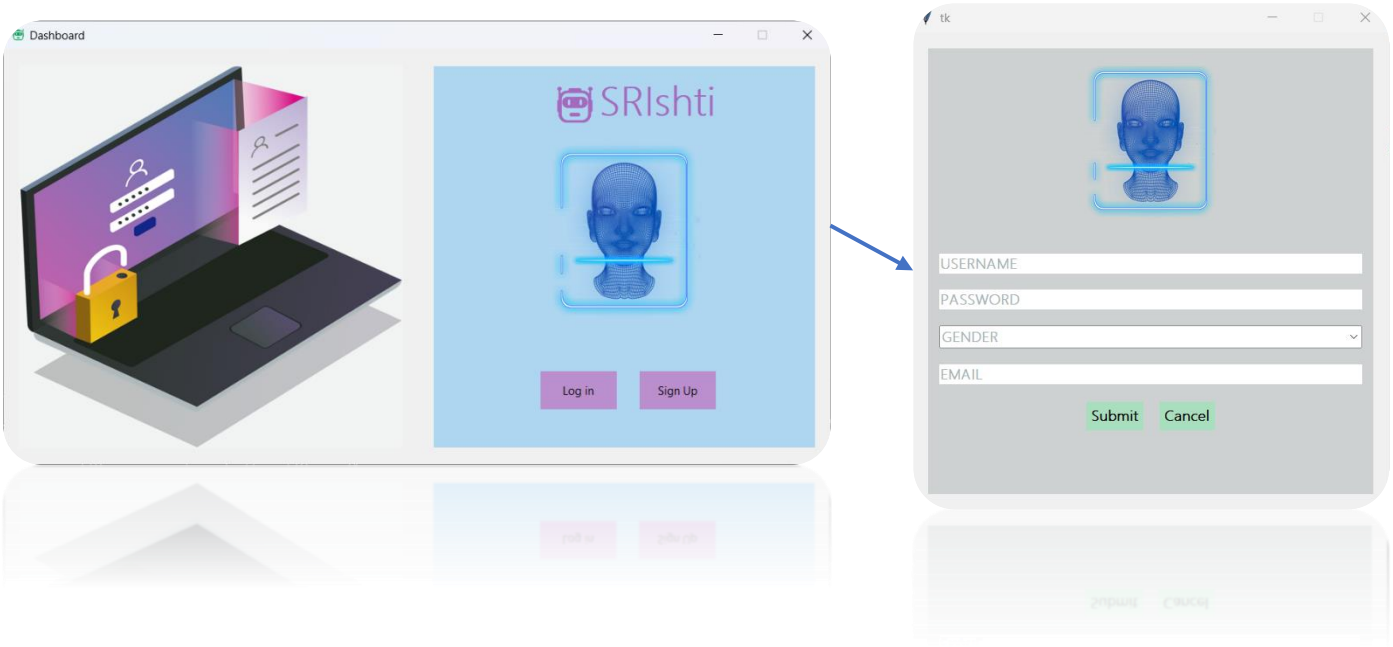


Figure 7.10 Describes the looks of the “Sign Up” page of the software where we can get the “Dashboard” page as a hyperlink in the “Sign Up” button.

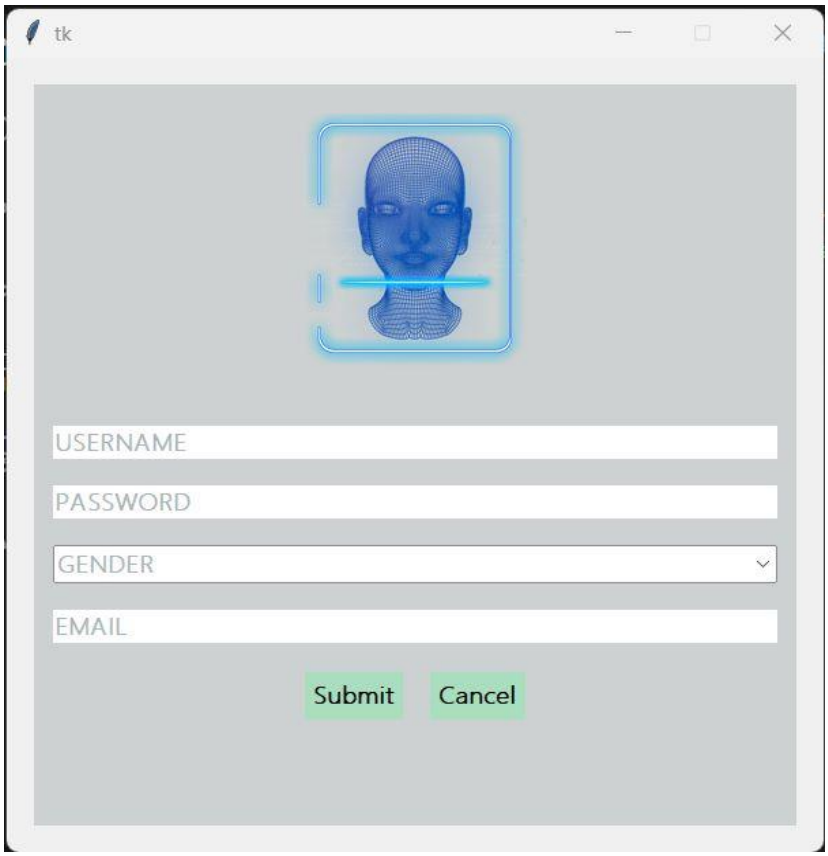


Figure 7.11

Figure 7.12, 7.13 At first, the required information is filled up by the user then, the software starts to scan your face and get your Face ID.

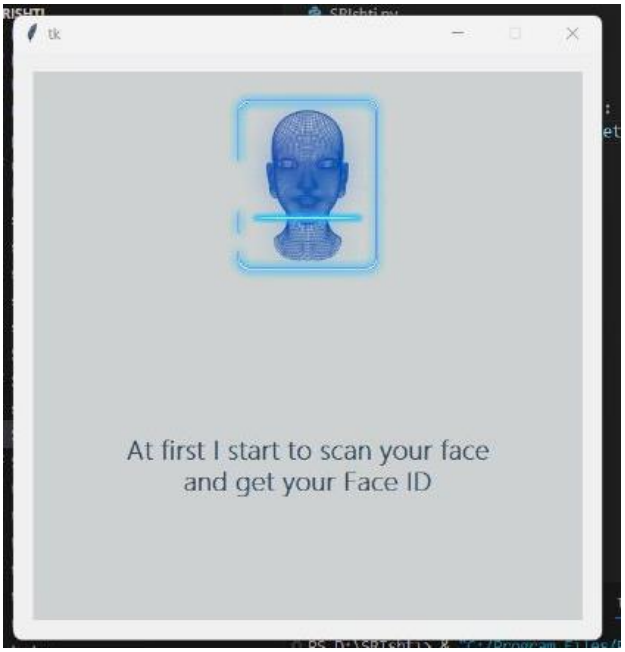


Figure 7.12

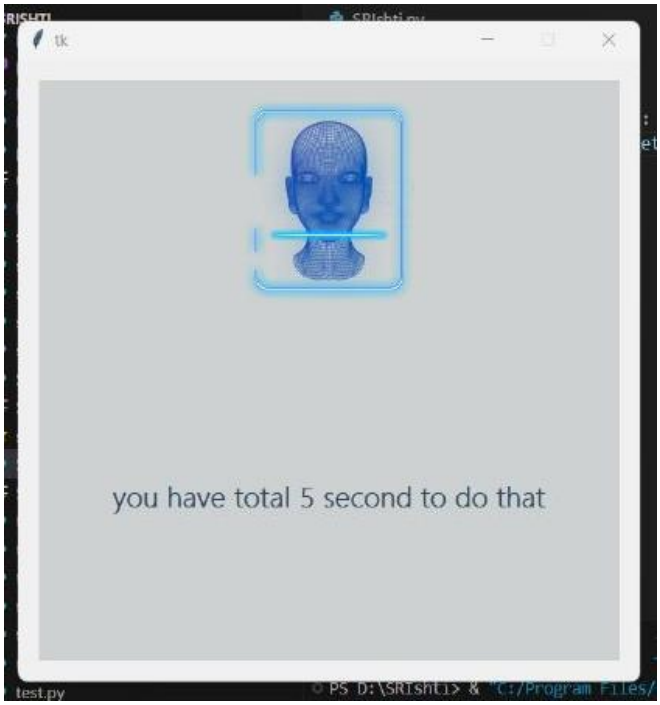


Figure 7.13

Figure 7.14 Describes the face recognition process and two-way verification method for ensuring the integrity and safety of the account user.

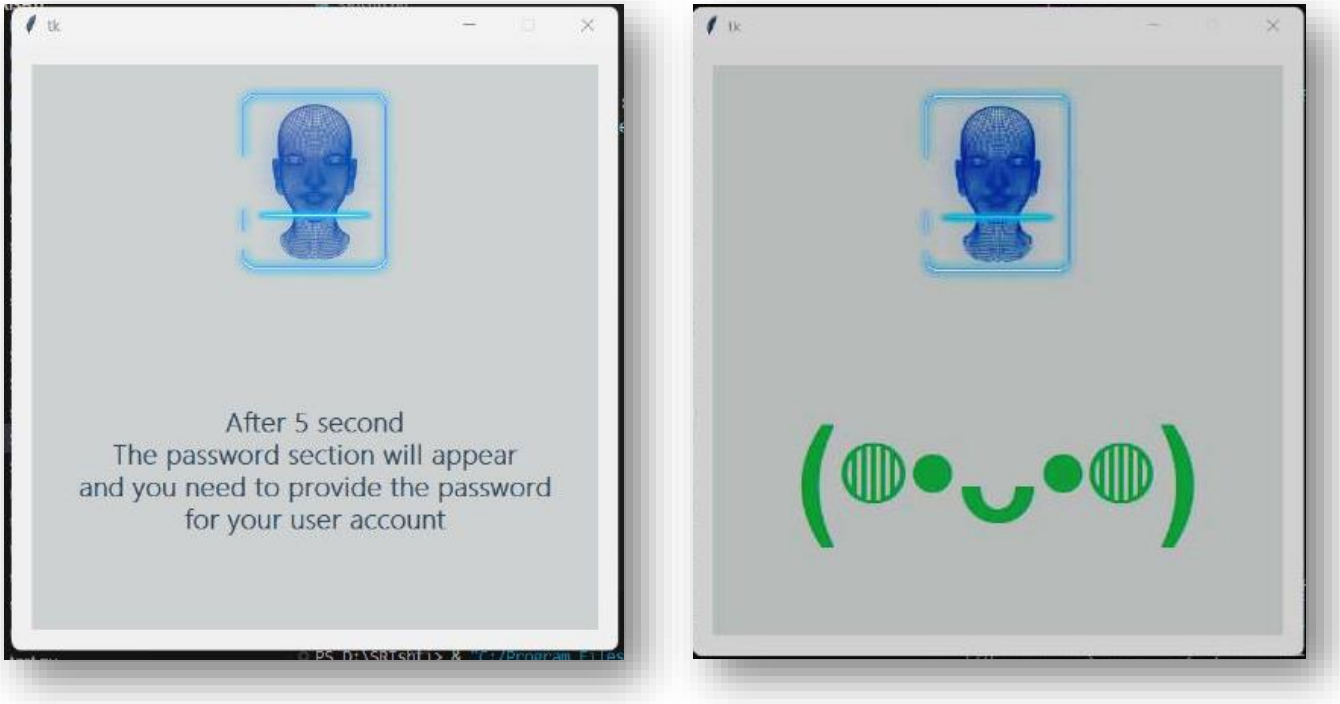


Figure 7.14

Figure 7.15 Describes the face recognition process and two-way verification method for ensuring the integrity and safety of the account user. And provide Face ID and User Name.

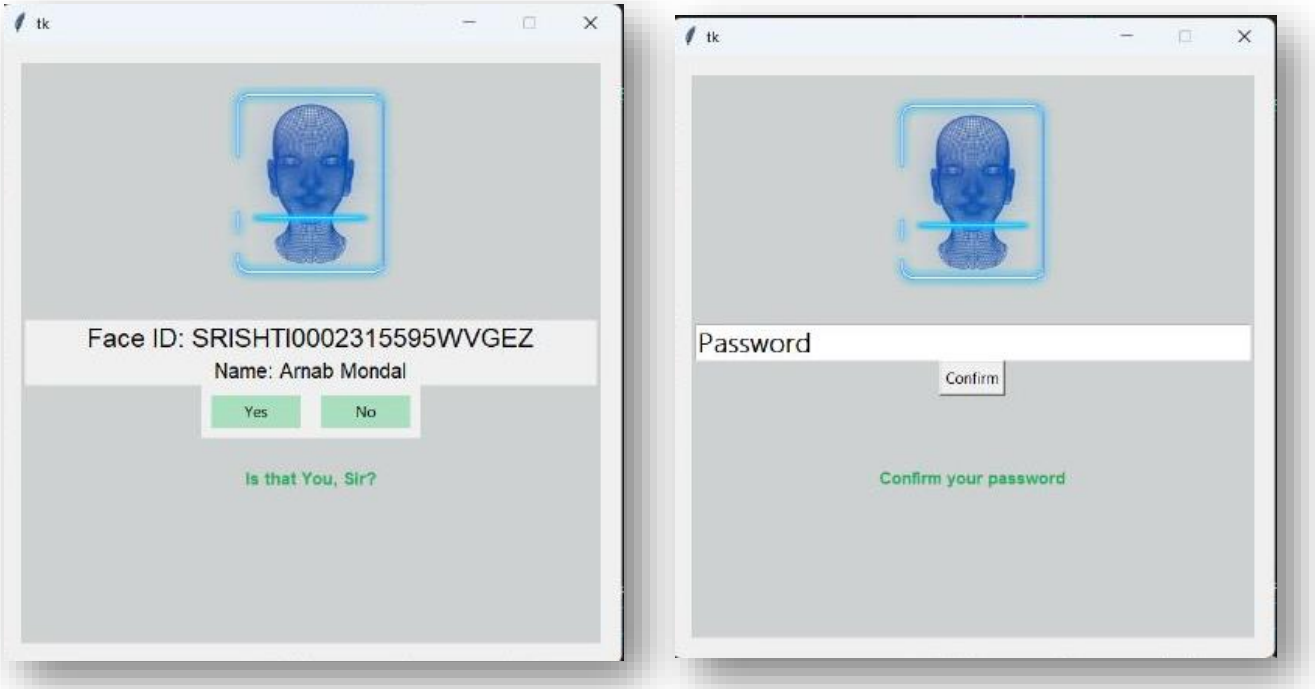
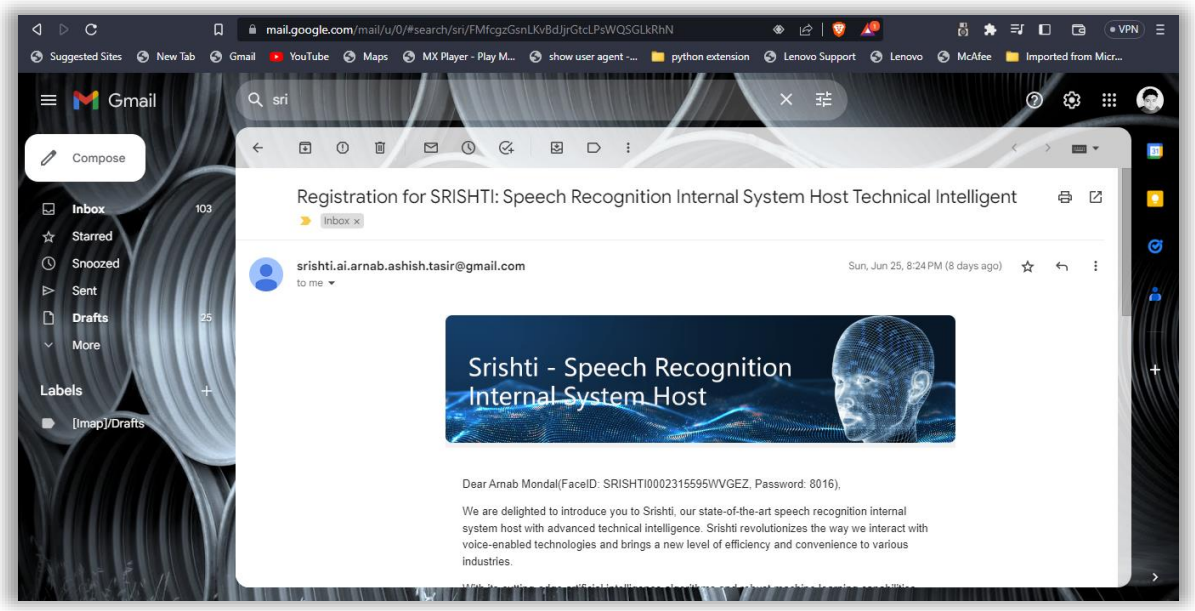


Figure 7.15

Figure 7.16 Describes the starting section of SRISHTI speech recognition where anyone can search for any queries, doubts, YouTube, and many more things.

FIGURE: CONFIRMATION OF EMAIL



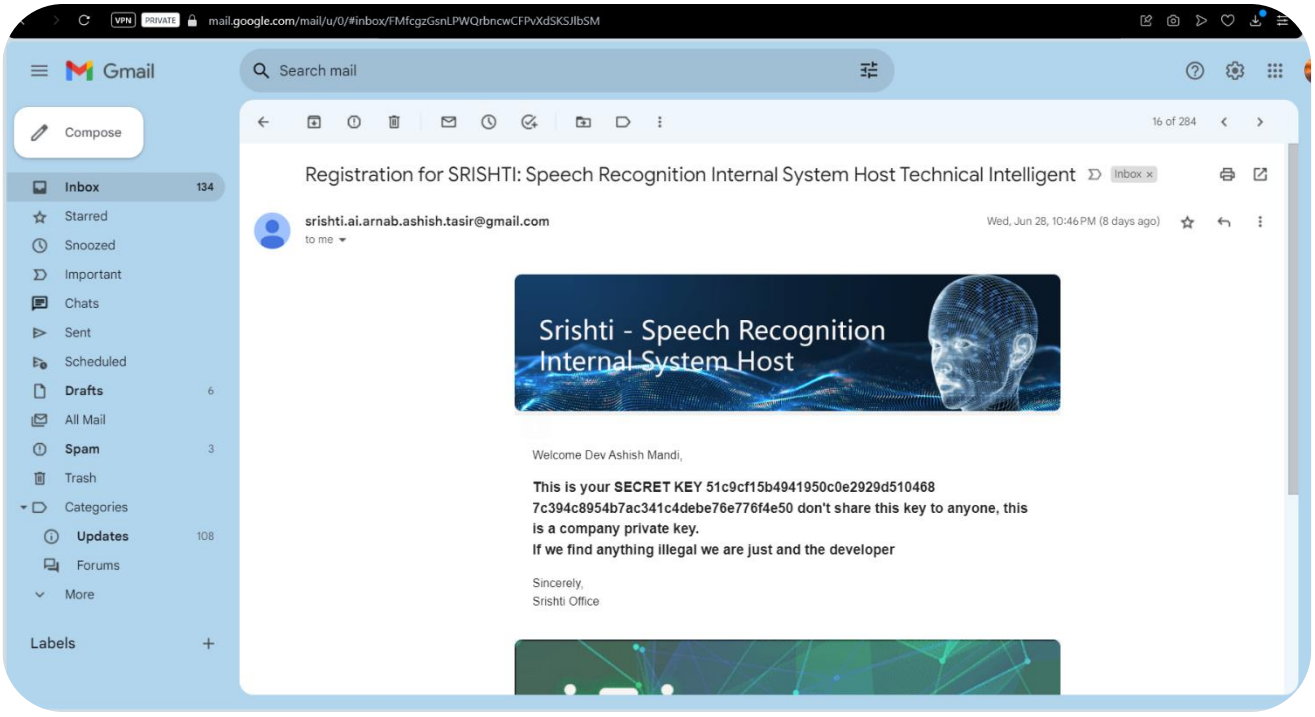


FIGURE 7.17 THE SOFTWARE THE WORKING GUI IS: LOGIN PAGE

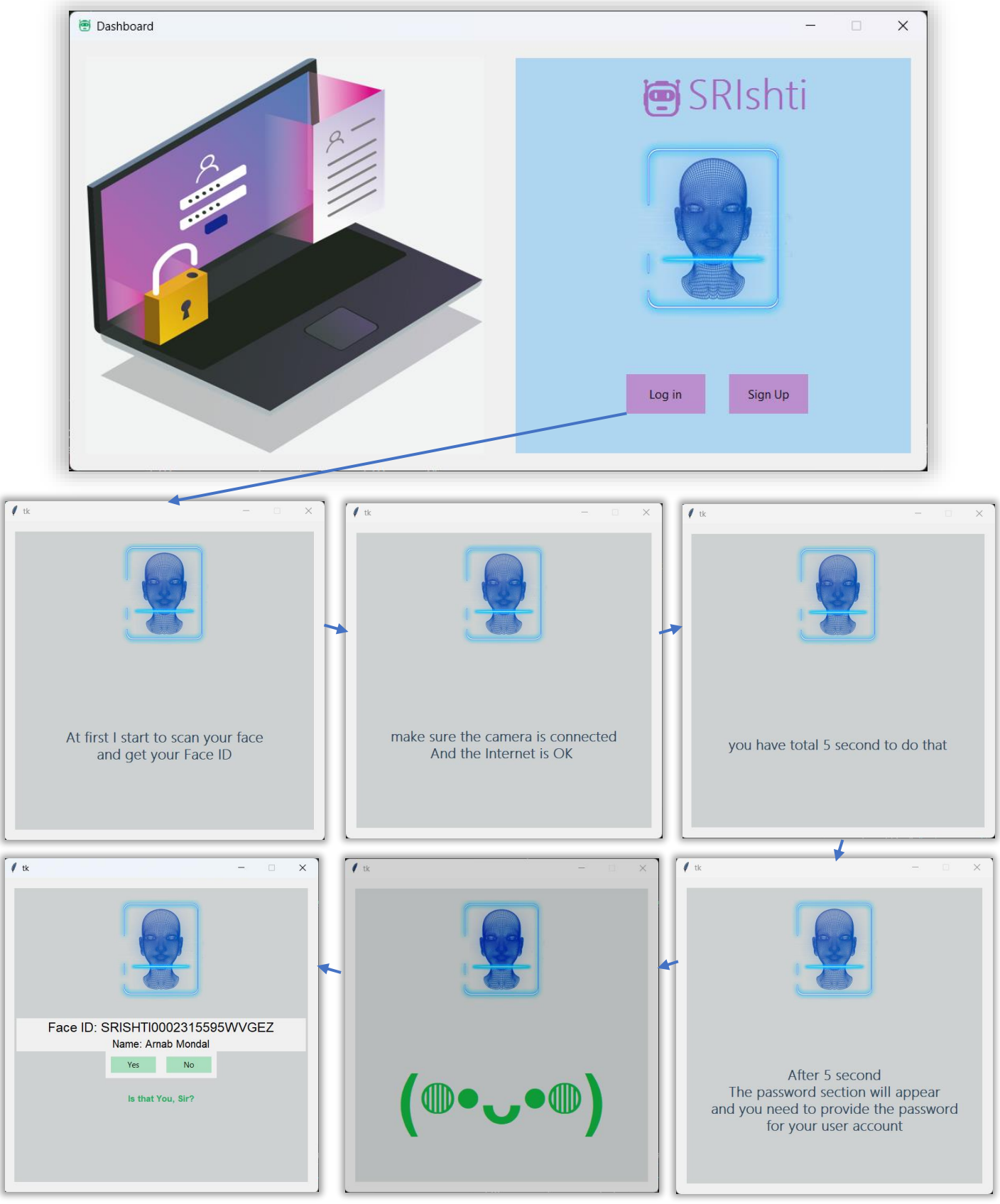


Figure 7.18 Describes the face recognition process and two-way verification method And provide your Face ID and User Name and after face recognition then by after typing your strong password automatically move to the SRISHTI search page.

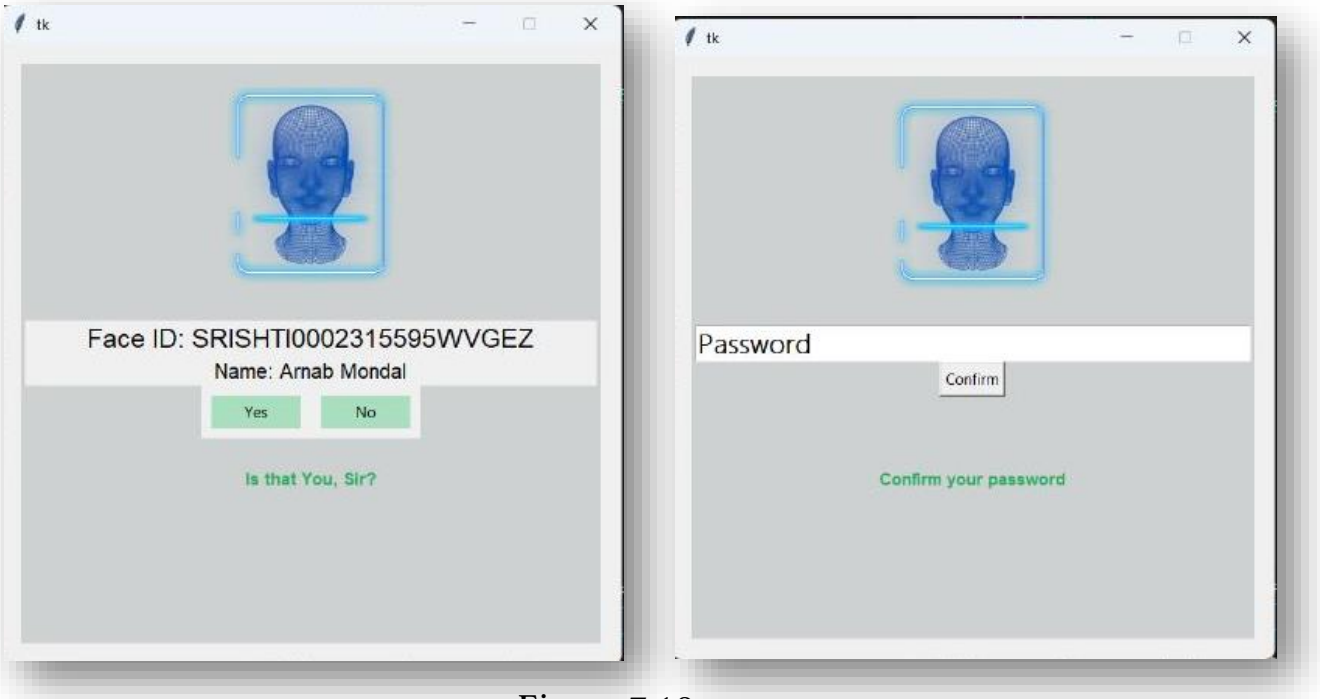


Figure 7.18

Figure 7.19 Describes the starting section of SRISHTI speech recognition where anyone can search for any queries, doubts, YouTube, and many more things.

Figure 7.19

7.2 Limitation of This Project

The limitation of this project is listed below in the following points ways manner:

1. If there is no camera and no microphone in the system then the software is not working properly.
2. If the user's voice is not clear they may be software not work properly.
3. Good internet connection is required for using the software properly.

7.3 Future Work

Srishti AI offers impressive speech recognition capabilities that can greatly enhance the way we interact with technology. The system demonstrates a high level of accuracy in understanding and interpreting spoken language, enabling users to effortlessly give voice commands, dictate text, or transcribe audio recordings. This functionality is particularly useful in situations where manual input is inconvenient or not possible. In terms of performance, Srishti AI generally delivers satisfactory results. Its speech recognition accuracy is commendable, and the system demonstrates a good understanding of context and intent. However, like any AI system, it may occasionally encounter challenges with accurately interpreting certain accents, dialects, or speech patterns. Continuous researches and developments are crucial to refine and improve the system's performance over time. One aspect that could be further explored is the integration of Srishti AI with other technologies or platforms. Seamless integration with voice-enabled devices, mobile applications, or customer service systems would enhance the overall user experience and extend the system's capabilities. Considering its strengths and potential areas for improvement, Srishti AI is a promising AI system with practical applications in various industries. It's advanced speech recognition and internal system hosting capabilities make it a valuable tool for businesses and developers alike. With ongoing development and refinement, Srishti AI has the potential to become even more accurate and versatile, contributing to the advancement of speech-enabled technologies.

In the future, Srishti AI, with the integration of Adriano, holds immense potential to redefine the way we interact with technology. Adriano, an advanced conversational AI, brings a human-like touch to the Srishti AI system, enhancing its capabilities and making it even more versatile and intuitive.

With Adriano's integration, Srishti AI will be able to engage in natural and lifelike conversations, effectively understanding and responding to user queries and requests. Adriano's advanced natural language processing abilities will enable the system to grasp complex nuances, context, and emotions, leading to more meaningful interactions. The addition of Adriano to Srishti AI will significantly benefit various industries and applications. In customer service, it can offer personalized and

empathetic support, resolving issues and inquiries with a human-like touch. In education, it can act as an interactive tutor, adapting to individual learning styles and providing tailored explanations and guidance.

Furthermore, the integration of Adriano will open doors for improved speech recognition and synthesis capabilities. With enhanced speech recognition, the system will be more accurate in transcribing and understanding spoken language. Additionally, it will be capable of generating speech that closely resembles human voices, making interactions with the system more natural and immersive. As technology continues to evolve, it is crucial to ensure responsible development and address ethical considerations. This includes maintaining transparency, respecting user privacy, and ensuring proper handling of sensitive information.

In conclusion, with the integration of Adriano, the future of Srishti AI looks promising. The combined power of advanced conversational AI and robust speech recognition will revolutionize the way we interact with intelligent systems. By offering lifelike conversations and human-like responses, Srishti AI with Adriano has the potential to reshape customer service, education, and various other sectors, bringing us closer to a seamless integration of AI into our daily lives.

7.4: Conclusion

Srishti AI (Speech Recognition Internal System Host Technical Intelligent) represents an impressive technological advancement in the field of artificial intelligence. Its primary focus is on speech recognition and internal system hosting, making it a powerful tool for various applications. Srishti AI's ability to accurately understand and process spoken language opens up possibilities for seamless communication and interaction with machines. Whether it's voice commands, dictation, or transcription, Srishti AI aims to provide efficient and reliable speech recognition capabilities. Additionally, Srishti AI's internal system hosting features enable it to serve as a robust platform for hosting and managing various AI applications and services. It provides the necessary infrastructures and resources to support the deployment and operation of intelligent systems. However, it's important to note that Srishti AI's effectiveness and performance depend on several factors, including data quality, training, and continuous improvement. Ongoing research and development are crucial to refine and enhance its speech recognition accuracy and system hosting capabilities.

In conclusion, Srishti AI holds great potential for revolutionizing speech recognition and internal system hosting. With its advanced technology, it offers opportunities for more natural and intuitive human-machine interaction. However, further advancements and refinements are necessary to ensure optimal performance and address any limitations that may arise.

7.5 References

1. Openai: https://openai.com/blog/chatgpt
2. Render: https://render.com/
3. GitHub: https://github.com/
4. Google: https://www.google.com/
5. News API: https://newsapi.org/
6. Weather API: https://www.weatherapi.com/
7. Cloudinary: https://cloudinary.com/
8. Books:
    a. Artificial Neural Networks, B. Yegnarayana, Prentice Hall of India.
    b. Artificial Intelligence with Python: A Comprehensive Guide to Building Intelligent Apps for Python Beginners and Developers
    c. Artificial Intelligence Programming with Python by Perry Xiao

Scan this QR Code access our website