CLASS-XII PROJECT

ACADEMIC YEAR :2023-24

PROJECT REPORT ON

TOPIC: VOICE ASSISTANT – FRIDAY.

BOARD ROLL NO	**:**	_____
NAME	**:**	**KRISH BRAHMBHATT**
CLASS	**:**	**12**
SUBJECT	**:**	**COMPUTER SCIENCE**
SUB CODE	**:**	**083**
PROJECT GUIDE	**:**	**Mr. NARENDRA ALIANI**

GREEN VALLEY

SCHOOL FOR CHILDREN

Certificate

This is to certify that **(KRISH BRAHMBHATT)** Board Roll No: **(_____)** has successfully completed the Project Work entitled **(Voice Assistant)** in the subject **COMPUTER SCIENCE (083)** laid down in the regulations of CBSE for the purpose of Practical Examination in **Class XII** for the **Academic Session: 2023-24** to be held in **GREEN VALLY SCHOOL FOR CHILDREN, GANDHINAGAR** on **(_____)**.

Internal Examiner's Signature	**External Examiner's Signature**
Principal's Signature	**School's Stamp**

TABLE OF CONTENTS

ACKNOWLEDGEMENT

Apart from one's own effort, the success of any project depends largely on the encouragement and guidelines of many others. I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project.

I express a deep sense of gratitude to almighty God for giving me strength for the successful completion of the project.

I express my heartfelt gratitude to my parents for their constant encouragement while carrying out this project.

I gratefully acknowledge the contribution of the individuals who contributed to bringing this project up to this level, who continue to look after me despite my flaws.

I express my deep sense of gratitude to the luminary Principal, Green Vally School For Children, Gandhinagar who has been continuously motivating and extending their helping hand to us.

I am overwhelmed to express my thanks to The Administrative Officer for providing me an infrastructure and moral support while carrying out this project in the school.

My sincere thanks to **Mr. Narendra Aliani**, In-charge, A guide, mentor, and above all a friend, who critically reviewed my project and helped in solving each and every problem, that occurred during the implementation of the project

The guidance and support received from all the members who contributed to this project were vital for the success of the project. I am grateful for their constant support and help.

PREFACE OF PROJECT

I have selected the project titled **"Voice Assistant - Friday"** . This project aims to develop a sophisticated voice-activated assistant using natural language processing and artificial intelligence. "Friday" will serve as a virtual assistant, seamlessly executing user commands and providing hands-free control over various functionalities. The project will leverage MySQL and Python for database connectivity, creating a cutting-edge voice assistant to enhance user convenience and productivity.

Our motivation springs from the captivating depiction of AI in movies, where characters like Tony Stark seamlessly interact with highly intelligent and responsive virtual assistants. The allure lies in the idea of transforming this cinematic concept into a tangible and functional reality. By bringing the charm of cinematic AI into the hands of users, we aspire to provide a genuine and enjoyable voice-controlled assistant experience.

OBJECTIVES OF PROJECT

The objective of this project is to apply programming knowledge to a real-world situation/problem and understand how programming skills help in developing good software.

1. Demonstrate in-depth knowledge of computer science, as exemplified in the areas of systems, theory, and software development, and facilitate required writing and presentation skills.
2. Demonstrate ability to conduct research or analysis on any real-life application and understand what data will be input, how it will be stored, and how it will be processed into meaningful information or desired output using computer science skills.
3. Develop a Computerized and Automated working model of any real-world application to make it work faster and thereby save time and money for any organization.
4. Write programs utilizing modern software tools.
5. Apply object-oriented programming principles effectively when developing small to medium-sized projects.
6. Write effective procedural code to solve small to medium-sized problems.

PROJECT DESCRIPTION

Your Voice Assistant - Friday is designed to make your tasks easier through simple voice commands. Let's see what it can do:

1. **Website Navigation:**
    - Simply mention a website, and Friday will open it for you. For example, saying "Open YouTube" will launch the YouTube website.
2. **Music Playback:**
    - Ask Friday to play your favourite songs. Specify the song and where you want to listen, be it YouTube, Spotify, or local storage.
3. **Shutdown and Goodbye:**
    - When you're ready to leave, use phrases like "sleep," "quit," or "goodbye," and Friday will bid you farewell and gracefully shut down.
4. **Settings Configuration:**
    - Need to tweak settings? Just say "open settings" or similar phrases, and Friday will guide you through the configuration process.
5. **Time Check:**
    - Wondering about the time? A simple "What's the time?" and Friday will tell you the current time.
6. **To-Do List Management:**
    - Stay organized by creating, reading, updating, and deleting to-do list items. Simply tell Friday what you want to do, and it will assist you accordingly.

HARDWARE & SOFTWARE SPECIFICATIONS

HARDWARE SPECIFICATIONS:

Processor	:	Intel Core I3 and above
Hard Disk	:	500 GB
Ram	:	8 GB

SOFTWARE SPECIFICATIONS:

Operating System	:	Windows 11
Platform	:	Python IDLE 3.11
Database	:	MySQL
Programming Language	:	Python

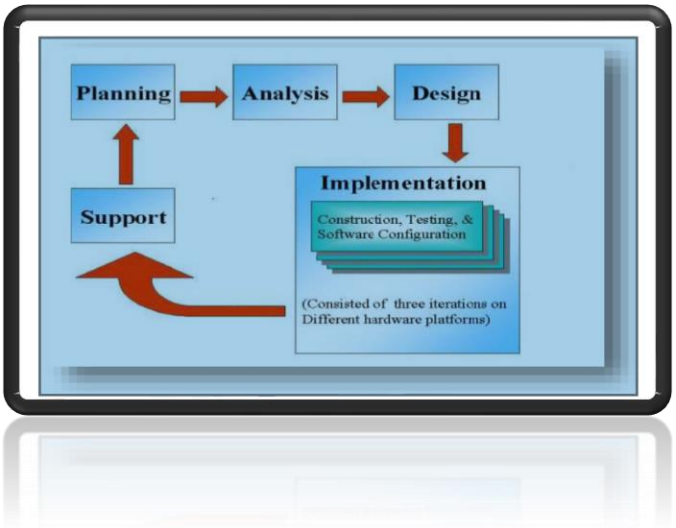
Note: For Python-MySQL connectivity, the following details have been used:-

Host- localhost, user- root, password- krish4926, database- friday_allusers

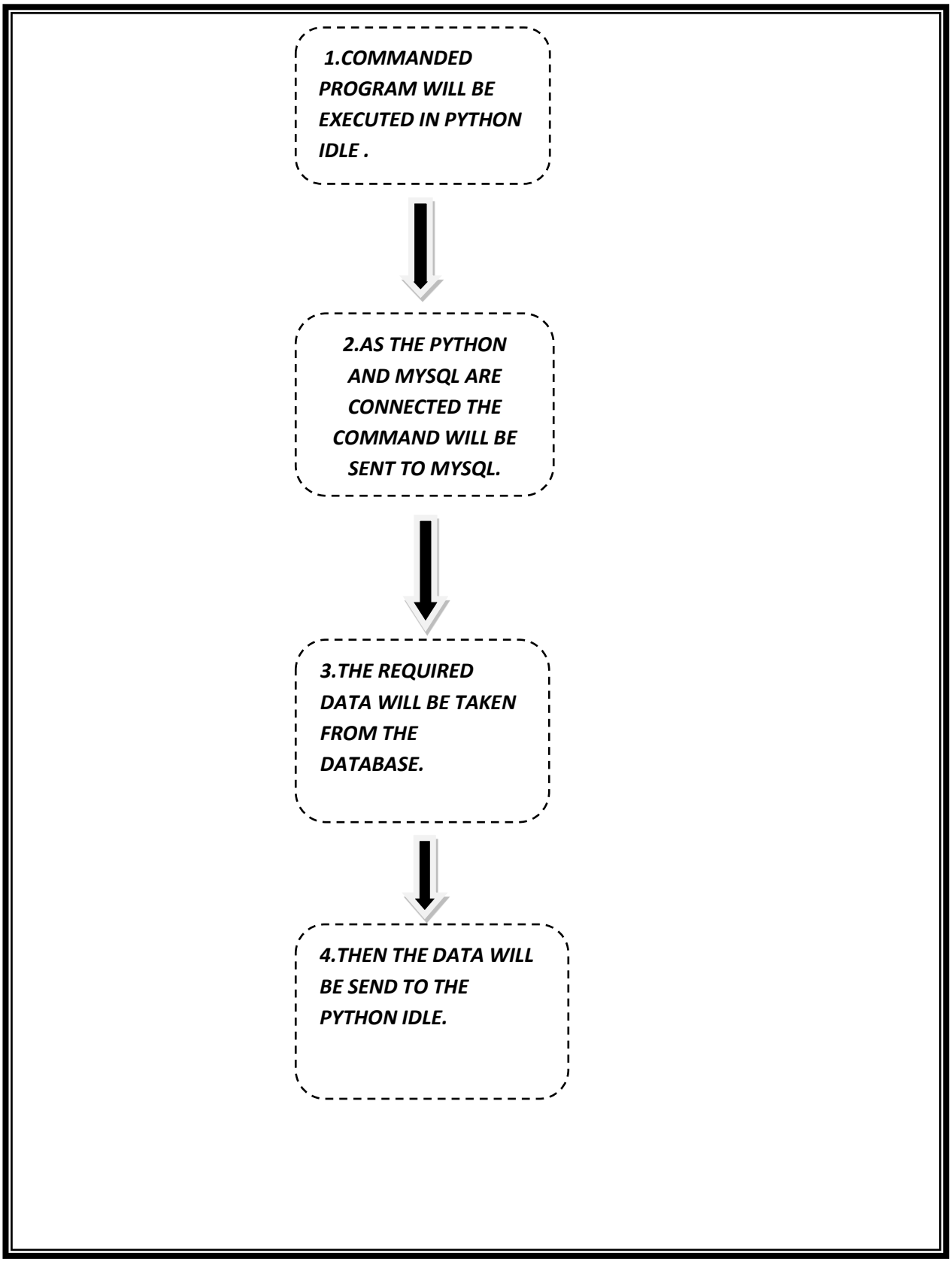
SYSTEM DEVELOPMENT LIFE CYCLE (SDLC)

The systems development life cycle is a project management technique that divides complex projects into smaller, more easily managed segments or phases. Segmenting projects allows managers to verify the successful completion of project phases before allocating resources to subsequent phases.

Software development projects typically include initiation, planning, design, development, testing, implementation, and maintenance phases. However, the phases may be divided differently depending on the organization involved. For example, initial project activities might be designated as request, requirements-definition, and planning phases, or initiation, concept-development, and planning phases. End users of the system under development should be involved in reviewing the output of each phase to ensure the system is being built to deliver the needed functionality.



FLOW CHART


1.COMMANDED PROGRAM WILL BE EXECUTED IN PYTHON IDLE .
2.AS THE PYTHON AND MYSQL ARE CONNECTED THE COMMAND WILL BE SENT TO MYSQL.
3.THE REQUIRED DATA WILL BE TAKEN FROM THE DATABASE.
4.THEN THE DATA WILL BE SEND TO THE PYTHON IDLE.

PYTHON MODULES USED

1. **speech_recognition (as sr):**
    - **Enables Friday to recognize and process speech input, facilitating seamless interaction with users.**
2. **pyttsx3:**
    - **Empowers Friday to convert text into speech, providing a natural and expressive voice output.**
3. **sys:**
    - **Offers system-specific parameters and functions, enhancing Friday's ability to interact with the underlying system.**
4. **datetime:**
    - **Provides functionality for working with dates and times, enabling Friday to incorporate time-related information in responses.**
5. **os:**
    - **Facilitates communication between Friday and the operating system, allowing execution of system-level commands and functions.**
6. **pywhatkit:**
    - **Allows Friday to interact with web services, enabling functionalities like playing music on YouTube.**
7. **random:**
    - **Enhances Friday's capabilities by introducing randomness, enabling varied and dynamic responses.**
8. **webbrowser:**
    - **Enables Friday to open web browsers, providing seamless access to online resources based on user commands.**
9. **mysql.connector (as sql):**
    - **Establishes connectivity between Friday and MySQL databases, allowing seamless interaction with stored data.**

10. time:

- **Provides functionality for working with time-related operations, enhancing Friday's ability to manage and display time-based information.**

11. smtplib:

- **Allows Friday to interact with email services, facilitating functionalities related to sending emails.**

12. PrettyTable:

- **Utilized for displaying data in a tabular format, enhancing the visual presentation of information in Friday's responses.**

These Python modules collectively empower Voice Assistant - Friday to perform a diverse range of tasks, ensuring a robust and interactive user experience.

DATABASE DESIGN

```
mysql> show tables;
+---------------------------+
| Tables_in_friday_allusers |
+---------------------------+
| friday_config             |
| krish4926_history         |
| testuser1_history         |
| todos                     |
| usage_reg                 |
| users                     |
| websites                  |
+---------------------------+
```

```
mysql> SELECT * FROM users;
+----------+-----------------------------+-----------------+----------+------------+-------+
| username | email_id                    | name            | password | mobile_no  | role  |
+----------+-----------------------------+-----------------+----------+------------+-------+
| Krish4926 | krishbrahmbhatt4926@gmail.com | Krish Brahmbhatt | krish4926 | 8780930655 | admin |
| testuser1 | testuser1@gmail.com          | TestUser1       | testuser1 | 1234567890 | guest |
| testuser2 | testuser2@gmail.com          | TestUser2       | testuser2 | 0987654321 | guest |
+----------+-----------------------------+-----------------+----------+------------+-------+
```

```
mysql> SELECT * FROM usage_reg;
+----+-----------+------------------+---------------------+
| id | username  | name             | log_time            |
+----+-----------+------------------+---------------------+
|  1 | testuser1 | TestUser1        | 2023-12-26 04:11:24 |
|  2 | testuser2 | TestUser2        | 2023-12-26 05:19:24 |
| 13 | testuser1 | TestUser1        | 2023-12-26 06:37:44 |
| 15 | Krish4926 | Krish Brahmbhatt | 2023-12-26 06:46:11 |
| 16 | testuser1 | TestUser1        | 2023-12-26 12:22:27 |
+----+-----------+------------------+---------------------+
```

```
mysql> SELECT * FROM krish4926_history;
+----+---------------------+--------------------------+---------------------+
| id | user_input          | friday_output            | timestamp           |
+----+---------------------+--------------------------+---------------------+
|  1 | open youtube for me | Opening websites: youtube | 2023-12-26 12:17:15 |
+----+---------------------+--------------------------+---------------------+
```

```
mysql> SELECT * FROM friday_config;
+----+--------------------+--------------------------+-----------------+----------------+
| id | default_speech_rate | default_email            | default_password | default_volume |
+----+--------------------+--------------------------+-----------------+----------------+
|  1 |                130 | fridayaibykrish@gmail.com | fridayaibykrish  |              1 |
+----+--------------------+--------------------------+-----------------+----------------+
```

```
mysql> SELECT * FROM websites;
+-----+--------------------+-------------------------------------+
| id  | name               | address                             |
+-----+--------------------+-------------------------------------+
|   1 | Google             | https://www.google.com              |
|   2 | YouTube            | https://www.youtube.com             |
|   3 | Facebook           | https://www.facebook.com            |
|   4 | WhatsApp           | https://www.whatsapp.com            |
|   5 | Amazon             | https://www.amazon.in               |
|   6 | Flipkart           | https://www.flipkart.com            |
|   7 | Instagram          | https://www.instagram.com           |
|   8 | Twitter            | https://www.twitter.com             |
|   9 | LinkedIn           | https://www.linkedin.com            |
|  10 | Yahoo              | https://in.yahoo.com                |
|  11 | Netflix            | https://www.netflix.com             |
|  12 | Hotstar            | https://www.hotstar.com             |
|  13 | Paytm              | https://www.paytm.com               |
|  14 | Snapdeal           | https://www.snapdeal.com            |
|  15 | Indiatimes         | https://www.indiatimes.com          |
|  16 | IRCTC              | https://www.irctc.co.in             |
|  17 | Zomato             | https://www.zomato.com              |
|  18 | Swiggy             | https://www.swiggy.com              |
|  19 | OLX                | https://www.olx.in                  |
|  20 | Quora              | https://www.quora.com               |
|  21 | BookMyShow         | https://www.bookmyshow.com          |
|  22 | Cricbuzz           | https://www.cricbuzz.com            |
|  23 | JioSaavn           | https://www.jiosaavn.com            |
|  24 | MakeMyTrip         | https://www.makemytrip.com          |
|  25 | Myntra             | https://www.myntra.com              |
|  26 | Ajio               | https://www.ajio.com                |
|  27 | Hindustan Times    | https://www.hindustantimes.com      |
|  28 | NDTV               | https://www.ndtv.com                |
|  29 | ICICI Bank         | https://www.icicibank.com           |
|  30 | State Bank of India | https://www.sbi.co.in              |
|  31 | HDFC Bank          | https://www.hdfcbank.com            |
|  32 | Axis Bank          | https://www.axisbank.com            |
|  33 | Cricinfo           | https://www.espncricinfo.com        |
|  34 | Gaana              | https://www.gaana.com               |
|  35 | BigBasket          | https://www.bigbasket.com           |
|  36 | Rediff             | https://www.rediff.com              |
|  37 | Economic Times     | https://economictimes.indiatimes.com |
|  38 | Jabong             | https://www.jabong.com              |
|  39 | Shopclues          | https://www.shopclues.com           |
|  40 | Lenskart           | https://www.lenskart.com            |
|  41 | Zee News           | https://zeenews.india.com           |
|  42 | Yatra              | https://www.yatra.com               |
|  43 | India Today        | https://www.indiatoday.in           |
|  44 | Firstpost          | https://www.firstpost.com           |
|  45 | Moneycontrol       | https://www.moneycontrol.com        |
|  46 | Inshorts           | https://www.inshorts.com            |
|  47 | Cleartrip          | https://www.cleartrip.com           |
|  48 | Justdial           | https://www.justdial.com            |
|  49 | Magicbricks        | https://www.magicbricks.com         |
|  50 | Naukri.com         | https://www.naukri.com              |
|  51 | Shine.com          | https://www.shine.com               |
```

SOURCE CODE (PYTHON)

```python
def activeuser():
    cursor_func.execute('''
    SELECT username
    FROM usage_reg
    ORDER BY log_time DESC
    LIMIT 1
    ''')
    latest_username_data = cursor_func.fetchone()
    loginusername = latest_username_data[0]
    return loginusername

def activeuname():
    cursor_func.execute('''
    SELECT name
    FROM usage_reg
    ORDER BY log_time DESC
    LIMIT 1
    ''')
    latest_uname_data = cursor_func.fetchone()
    loginusern = latest_uname_data[0]
    return loginusern

current_username = activeuser()
current_uname = activeuname()

#Default settings----------

def current_defaults() :
    cursor_func.execute('''
        SELECT * FROM friday_config
        ORDER BY id DESC
        LIMIT 1
    ''')
    latest_config = cursor_func.fetchone()
    default_id, default_speech_rate, default_email, default_password, default_volume =
latest_config
    return default_speech_rate, default_email, default_password, default_volume

default_rate, default_sendersemail, default_senderspassword, default_volume = current_defaults()

def slowprint(text):
    for char in text:
        print(char, end='', flush=True)
        time.sleep(0.1)
    print()

def clear():
    os.system('cls')


def mail(to, content) :
    server = smtplib.SMTP("smtp.gmail.com", 587)
    server.ehlo()
    server.starttls()
    server.login(default_sendersemail, default_senderspassword)
    server.sendmail(default_sendersemail, default_senderspassword, to, content)
    server.close()
        time.sleep(0.1)
    print()
def clear():
    os.system('cls')
```

```python
con_auth = sql.connect(host='localhost', user='root', password='krish4926', database='friday_allusers')
cursor_auth = con_auth.cursor()
def signup():
    speak("Please fill in the details. Type 'exit' at any time to quit.")
    while True:
        username = input("Enter your username: ")
        if username.lower() == 'exit':
            sys.exit("User opted to exit the signup process.")
        email_id = input("Enter your email: ")
        cursor_auth.execute('''
            SELECT 1 FROM users WHERE username = %s OR email_id = %s
        ''', (username, email_id))
        duplicate_entry = cursor_auth.fetchone()
        if duplicate_entry:
            print("Error: Username or email already exists. Please choose a different one.")
            continue
        uname = input("Enter your full name: ")
        while True:
            password = input("Enter your password: ")
            repeat_password = input("Repeat your password: ")
            if password == repeat_password:
                break
            else:
                print("Passwords do not match. Please try again.")
        mobile_no = input("Enter your mobile number: ")
        log_time = time.strftime('%Y-%m-%d %H:%M:%S')
        cursor_auth.execute('''
            INSERT INTO users (username, email_id, name, password, mobile_no)
            VALUES (%s, %s, %s, %s, %s)
        ''', (username, email_id, uname, password, mobile_no))
        cursor_auth.execute('''
            INSERT INTO usage_reg (username, name, log_time)
            VALUES (%s, %s, %s)
        ''', (username, uname, log_time))
        con_auth.commit()
        speak("You have been signed up to your account")
        time.sleep(1)
        break

def login():
    login_successful = True
    while login_successful:
        login_input = input("Enter your username/email/mobile_no (type 'exit' to quit): ")
        if login_input.lower() == 'exit':
            sys.exit("User opted to exit the login process.")
        password_input = input("Enter your password: ")
        cursor_auth.execute('''
            SELECT * FROM users
            WHERE username = %s OR email_id = %s OR mobile_no = %s
        ''', (login_input, login_input, login_input))
        user_data = cursor_auth.fetchone()
        if user_data and user_data[3] == password_input:
            speak("Login successful!")
            log_time = time.strftime('%Y-%m-%d %H:%M:%S')  # Define log_time
            cursor_auth.execute('''
                INSERT INTO usage_reg (username, name, log_time)
                VALUES (%s, %s, %s)
            ''', (user_data[0], user_data[2], log_time))
            con_auth.commit()
            time.sleep(2)
            clear()
            login_successful = False
            print("You are logged in.")
            return
        else:
            speak("Login failed. Invalid username/email/mobile_no or password.")
            time.sleep(1)

def UserAuth():
    speak("Would you like to Signup or Login as an Existing User?")
    authdata = listen()
    if any(keyword in authdata.lower() for keyword in ["login", "old user", "not a new user", "not new user",
"signin", "sign in", "log in"]):
        login()
    elif any(keyword in authdata.lower() for keyword in ["signup", "new user", "sign me up", "logup", "sign up",
"log up"]):
        signup()

def activeuser():
    cursor_auth.execute('''
    SELECT username
    FROM usage_reg
    ORDER BY log_time DESC
    LIMIT 1
    ''')
    latest_username_data = cursor_auth.fetchone()
    loginusername = latest_username_data[0]
    return loginusername

def activeuname():
    cursor_auth.execute('''
    SELECT name
    FROM usage_reg
    ORDER BY log_time DESC
    LIMIT 1
    ''')
    latest_uname_data = cursor_auth.fetchone()
    loginusern = latest_uname_data[0]
    return loginusern
```

```python
con_setting = sql.connect(host='localhost', user='root', password='krish4926', database='friday_allusers')
cursor_setting = con_setting.cursor()
current_user= activeuser() #for username
uname = activeuname() #for name
def settings(login_user):
    settingsloop = True

    while settingsloop:
        speak("What would you like to access?")
        print("""
            1) Admin Controls
            2) Friday Settings
            3) Users History
            4) Account Centre
            5) Exit
        """)
        query1 = listen()
        if any(keyword in query1.lower() for keyword in ["nothing", "exit", "close", "5", "five"]):
            settingsloop = False
        elif any(keyword in query1.lower() for keyword in ["1", "one", "admin", "admin controls", "admin settings", "admin setting", "admin control"]):
            admin_controls(login_user)  # Pass login_user to admin_controls
        elif any(keyword in query1.lower() for keyword in ["2", "two", "setting", "settings"]):
            friday_settings()
        elif any(keyword in query1.lower() for keyword in ["3", "three", "history"]):
            users_history()
        elif any(keyword in query1.lower() for keyword in ["4", "four", "profile","account","center","centre"]):
            profile()


```

```
def admin_controls(login_user):
    cursor_setting.execute('''
        SELECT role FROM users
        WHERE username = %s
    ''', (login_user,))
    user_role = cursor_setting.fetchone()

    if user_role and user_role[0] == 'admin':
        speak("Welcome Admin!")
        speak("Performing admin actions...")

        while True:
            speak("Admin Controls opened! ")
            print("""
                Admin Controls:
                1) View User List
                2) Grant Admin Access
                3) Revoke Admin Access
                4) View All User History
                5) Exit Admin Controls
            """)
            query2 = listen()

            if any(keyword in query2.lower() for keyword in ["1", "one", "user list", "list", "first"]):
                view_user_list()
            elif any(keyword in query2.lower() for keyword in ["2", "two", "grant", "allow", "second", "give"]):
                grant_admin_access()
            elif any(keyword in query2.lower() for keyword in ["3", "three", "revoke", "remove", "third"]):
                revoke_admin_access()
            elif any(keyword in query2.lower() for keyword in ["4", "four", "view all", "user history", "all users", "fourth"]):
                view_all_user_history()
            elif any(keyword in query2.lower() for keyword in ["5", "five", "exit", "close", "bye", "fifth"]):
                speak("Exiting Admin Controls.")
                break
            else:
                speak("Invalid choice. Please enter a valid option.")

    else:
        speak("You do not have permission to access admin controls. Please log in as an admin.")
        admin_login_successful = False
        while not admin_login_successful:
            admin_username = input("Enter admin username: ")
            admin_password = input("Enter admin password: ")
            cursor_setting.execute('''
                SELECT 1 FROM users
                WHERE username = %s AND password = %s AND role = 'admin'
            ''', (admin_username, admin_password))
            is_admin = cursor_setting.fetchone()

            if is_admin:
                cursor_setting.execute('''
                        SELECT * FROM users
                        WHERE username = %s
                    ''', (admin_username,))
                user_data = cursor_setting.fetchone()
                print(user_data[2],user_data[0])
                log_time = time.strftime('%Y-%m-%d %H:%M:%S')
                cursor_setting.execute('''
                        INSERT INTO usage_reg (id, username, name, log_time)
                        VALUES (NULL, %s, %s, %s)
                    ''', (user_data[0], user_data[2], log_time))
                con_setting.commit()
                speak("Admin login successful!")
                login_user = admin_username
                admin_login_successful = True
                print("Performing admin actions...")
                while True:
                    speak("Admin Controls opened! ")
                    print("""
                        Admin Controls:
                        1) View User List
                        2) Grant Admin Access
                        3) Revoke Admin Access
                        4) View All User History
                        5) Exit Admin Controls
                    """)
                    query2 = listen()

                    if any(keyword in query2.lower() for keyword in ["1", "one", "user list", "list", "first"]):
                        view_user_list()
                    elif any(keyword in query2.lower() for keyword in ["2", "two", "grant", "allow", "second", "give"]):
                        grant_admin_access()
                    elif any(keyword in query2.lower() for keyword in ["3", "three", "revoke", "remove", "third"]):
                        revoke_admin_access()
                    elif any(keyword in query2.lower() for keyword in ["4", "four", "view all", "user history", "all users", "fourth"]):
                        view_all_user_history()
                    elif any(keyword in query2.lower() for keyword in ["5", "five", "exit", "close", "bye", "fifth"]):
                        speak("Exiting Admin Controls.")
                        break
                    else:
                        speak("Invalid choice. Please enter a valid option.")
            else:
                speak("Invalid admin credentials. What else can I do for you?")
                break
```

```python
def view_user_list():
    speak("Displaying the User List.")
    cursor_setting.execute('''
        SELECT * FROM users
    ''')
    user_list_data = cursor_setting.fetchall()
    if not user_list_data:
        speak("No users found.")
    else:
        table = PrettyTable()
        table.field_names = ["Username", "Email ID", "Name", "Password", "Mobile Number", "Role"]
        for user_data in user_list_data:
            table.add_row([user_data[0], user_data[1], user_data[2], user_data[3], user_data[4],
user_data[5]])
        print(table)
    speak("End of User List.")

def display_users_table():
    cursor_setting.execute('''
        SELECT name, username, role FROM users
    ''')
    user_list_data = cursor_setting.fetchall()
    table = PrettyTable()
    table.field_names = ["Name", "Username", "Role"]
    for user_data in user_list_data:
        table.add_row([user_data[0], user_data[1], user_data[2]])
    print(table)

def grant_admin_access():
    display_users_table()
    speak("Enter the username or name to grant admin access: ")
    username_to_grant = input("Username/Name: ")
    cursor_setting.execute('''
        SELECT name, username, role FROM users
        WHERE name = %s OR username = %s
    ''', (username_to_grant, username_to_grant))
    user_data = cursor_setting.fetchone()

    if user_data:
        name, username, role = user_data
        if role == 'admin':
            print(f"{name} ({username}) already has admin access.")
        else:
            cursor_setting.execute('''
                UPDATE users
                SET role = 'admin'
                WHERE username = %s
            ''', (username,))
            con_setting.commit()
            print(f"Admin access granted to {name} ({username}).")
    else:
        print(f"No user found with the name/username: {username_to_grant}.")

def revoke_admin_access():
    display_users_table()
    speak("Enter the username or name to revoke admin access: ")
    username_to_revoke = input("Username/Name: ")
    cursor_setting.execute('''
        SELECT name, username, role FROM users
        WHERE name = %s OR username = %s
    ''', (username_to_revoke, username_to_revoke))
    user_data = cursor_setting.fetchone()

    if user_data:
        name, username, role = user_data
        if role == 'guest':
            print(f"{name} ({username}) already doesn't have admin access.")
        else:
            cursor_setting.execute('''
                UPDATE users
                SET role = 'guest'
                WHERE username = %s
            ''', (username,))
            con_setting.commit()
            print(f"Admin access revoked from {name} ({username}).")
    else:
        print(f"No user found with the name/username: {username_to_revoke}.")

def history(username, user_input, friday_output):
    table_name = f"{username}_history"
    cursor_setting.execute(f'''
        CREATE TABLE IF NOT EXISTS {table_name} (
            id INT AUTO_INCREMENT PRIMARY KEY,
            user_input TEXT,
            friday_output TEXT,
            timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP
        )
    ''')
    cursor_setting.execute(f'''
        INSERT INTO {table_name} (user_input, friday_output)
        VALUES (%s, %s)
    ''', (user_input, friday_output))
    con_setting.commit()
```

```python
def friday_settings():
    print("Friday Settings")
    while True:
        print("""
            1) Change Speech Rate
            2) Change Default Email Address (Used for Emailing Services)
            3) Adjust Volume
            4) Back to Main Menu
        """)
        query2 = listen()
        if any(keyword in query2.lower() for keyword in ["nothing", "exit", "close", "4", "four"]):
            print("Exiting Friday Settings.")
            break
        elif any(keyword in query2.lower() for keyword in ["1", "one", "speech rate", "change speech rate"]):
            change_speech_rate()
        elif any(keyword in query2.lower() for keyword in ["2", "two", "default email", "change email"]):
            change_default_email()
        elif any(keyword in query2.lower() for keyword in ["3", "three", "volume", "adjust volume"]):
            adjust_volume()
        else:
            print("Invalid choice. Please enter a valid option.")
def change_default_email():
    speak("Enter your new default email address: ")
    new_default_email = input("Email: ")

    speak("Enter your new default password: ")
    new_default_password = input("Password: ")

    speak(f"Confirm changes: New email: {new_default_email}, New password: {new_default_password}. Is that correct?")
    confirmation = listen()
    if any(keyword in confirmation for keyword in ["yes", "confirm"]):
        cursor_func.execute('''
            UPDATE friday_config
            SET default_email = %s, default_password = %s
            ORDER BY id DESC
            LIMIT 1
        ''', (new_default_email, new_default_password))
        con_func.commit()

        speak("Changes confirmed. Default email and password updated successfully.")
    else:
        speak("Changes not confirmed. Default email and password remain unchanged.")
def change_speech_rate():
    speak("What is the new rate? ")
    print("""Default: 130
High: 150
Slow: 100""")
    user_input = listen().lower()
    rate_mapping = {"default": "130", "fast": "150", "slow": "100"}
    if user_input.isdigit():
        new_rate = user_input
    elif user_input in rate_mapping:
        new_rate = rate_mapping[user_input]
    else:
        speak("Invalid input. Please provide a numeric value or choose from the allowed terms.")
        return
    cursor_func.execute('''
        UPDATE friday_config
        SET default_speech_rate = %s
        ORDER BY id DESC
        LIMIT 1
    ''', (new_rate,))
    con_func.commit()
    speak(f"Speech rate updated to {new_rate}.")
def adjust_volume():
    speak("What is the new volume level? Please say a number between 0 and 100.")
    new_volume = listen()
    try:
        new_volume = int(new_volume)
        if 0 <= new_volume <= 100:
            new_volume_range = new_volume/100
            cursor_func.execute('''
                UPDATE friday_config
                SET default_volume = %s
                ORDER BY id DESC
                LIMIT 1
            ''', (new_volume_range,))
            con_func.commit()
            speak(f"Volume level updated to {new_volume}.")
        else:
            speak("Please provide a valid volume level between 0 and 100.")
    except ValueError:
        speak("Sorry, I couldn't understand the volume level. Please provide a valid number.")

```

```python
def users_history():
    print("Users History")
    table_name = f"{current_user}_history"
    cursor_setting.execute(f"SELECT * FROM {table_name}")
    history_data = cursor_setting.fetchall()
    table = PrettyTable()
    table.field_names = ["ID", "User Input", "Friday Output", "Timestamp"]
    for entry in history_data:
        table.add_row(entry)
    print("Displaying Users History:")
    print(table)
    print("""
        1) Delete All History
        2) Delete History in a Range
        3) Delete Single History by ID
        4) Back to Main Menu
    """)

    action_choice = listen()
    print("Action Choice:", action_choice)  # Add this line


    if any(keyword in action_choice.lower() for keyword in ["nothing", "exit", "close", "4", "four"]):
        print("Exiting Users History.")
    elif any(keyword in action_choice.lower() for keyword in ["1", "delete all", "delete all history"]):
        delete_all_history()
    elif any(keyword in action_choice.lower() for keyword in ["2", "delete range", "delete history range"]):
        delete_history_range()
    elif any(keyword in action_choice.lower() for keyword in ["3", "delete single", "delete history
single"]):delete_single_history()
    else:
        print("Invalid choice. Please enter a valid option.")
    con_setting.commit()


def delete_all_history():
    table_name = f"{current_user}_history"
    cursor_setting.execute(f"DELETE FROM {table_name}")
    con_setting.commit()

def delete_history_range():
    table_name = f"{current_user}_history"
    speak("Enter the range of history entries to delete (start ID and end ID):")
    start_id = input("Start ID: ")
    end_id = input("End ID: ")
    cursor_setting.execute(f"DELETE FROM {table_name} WHERE id BETWEEN %s AND %s", (start_id, end_id))
    con_setting.commit()
    speak(f"History entries from ID {start_id} to {end_id} deleted successfully.")

def delete_single_history():
    table_name = f"{current_user}_history"
    speak("Enter the ID of the history entry to delete:")
    entry_id = input("Entry ID: ")
    cursor_setting.execute(f"DELETE FROM {table_name} WHERE id = %s", (entry_id,))
    con_setting.commit()
    speak(f"History entry with ID {entry_id} deleted successfully.")


```

```
def view_profile():
    table_name = "users"
    cursor_setting.execute(f"SELECT * FROM {table_name} WHERE username = %s", (current_user,))
    user_data = cursor_setting.fetchone()

    if user_data:
        table = PrettyTable()
        table.field_names = ["Username", "Email", "Name", "Mobile Number", "Role"]
        table.add_row(user_data[:-1])  # Exclude the "Role" field from user_data
        print("Your Profile:")
        print(table)
    else:
        print("User not found. Please check your credentials.")

def update_profile():
    speak("What would you like to change?")
    print("1) Name\n2) Email\n3) Mobile Number\n4) Password\n5) Back to Main Menu")

    choice = listen()

    if any(keyword in choice.lower() for keyword in ["1", "name"]):
        new_name = input("Enter your new name: ")
        cursor_setting.execute(f"UPDATE users SET name = %s WHERE username = %s", (new_name, current_user))
        speak("Name updated successfully.")
    elif any(keyword in choice.lower() for keyword in ["2", "email"]):
        new_email = input("Enter your new email: ")
        cursor_setting.execute(f"SELECT * FROM users WHERE email_id = %s", (new_email,))
        existing_user = cursor_setting.fetchone()
        if existing_user:
            speak("Email already exists. Please choose a different email.")
        else:
            cursor_setting.execute(f"UPDATE users SET email_id = %s WHERE username = %s", (new_email,
current_user))
            speak("Email updated successfully.")
    elif any(keyword in choice.lower() for keyword in ["3", "mobile"]):
        new_mobile = input("Enter your new mobile number: ")
        cursor_setting.execute(f"SELECT * FROM users WHERE mobile_no = %s", (new_mobile,))
        existing_user = cursor_setting.fetchone()
        if existing_user:
            speak("Mobile number already exists. Please choose a different number.")
        else:
            cursor_setting.execute(f"UPDATE users SET mobile_no = %s WHERE username = %s", (new_mobile,
current_user))
            speak("Mobile number updated successfully.")
    elif any(keyword in choice.lower() for keyword in ["4", "password"]):
        new_password = input("Enter your new password: ")
        cursor_setting.execute(f"UPDATE users SET password = %s WHERE username = %s", (new_password,
current_user))
        speak("Password updated successfully.")
    elif any(keyword in choice.lower() for keyword in ["5", "back"]):
        speak("Returning to Main Menu.")
    else:
        speak("Invalid choice. Please enter a valid option.")

def delete_account():
    speak("Are you sure you want to delete your account? (confirm/no): ")
    confirm = listen()

    if any(keyword in confirm.lower() for keyword in ["yes", "confirm"]):
        cursor_setting.execute("DELETE FROM users WHERE username = %s", (current_user,))
        con_setting.commit()
        speak("Account deleted successfully. Goodbye!")
        exit()
    else:
        speak("Account deletion canceled.")

def profile():
    speak("Profile Options:")
    print("1) View Profile\n2) Update Profile\n3) Delete Account\n4) Back to Main Menu")
    choice = listen()
    if any(keyword in choice.lower() for keyword in ["1", "view"]):
        view_profile()
    elif any(keyword in choice.lower() for keyword in ["2", "update"]):
        update_profile()
    elif any(keyword in choice.lower() for keyword in ["3", "delete"]):
        delete_account()
    elif any(keyword in choice.lower() for keyword in ["4", "back"]):
        speak("Returning to Main Menu.")
    else:
        speak("Invalid choice. Please enter a valid option.")
```

```python
con_auth = sql.connect(host='localhost', user='root', password='krish4926', database='friday_allusers')
cursor_auth = con_auth.cursor()
def signup():
    speak("Please fill in the details. Type 'exit' at any time to quit.")
    while True:
        username = input("Enter your username: ")
        if username.lower() == 'exit':
            sys.exit("User opted to exit the signup process.")
        email_id = input("Enter your email: ")
        cursor_auth.execute('''
            SELECT 1 FROM users WHERE username = %s OR email_id = %s
        ''', (username, email_id))
        duplicate_entry = cursor_auth.fetchone()
        if duplicate_entry:
            print("Error: Username or email already exists. Please choose a different one.")
            continue
        uname = input("Enter your full name: ")
        while True:
            password = input("Enter your password: ")
            repeat_password = input("Repeat your password: ")
            if password == repeat_password:
                break
            else:
                print("Passwords do not match. Please try again.")
        mobile_no = input("Enter your mobile number: ")
        log_time = time.strftime('%Y-%m-%d %H:%M:%S')
        cursor_auth.execute('''
            INSERT INTO users (username, email_id, name, password, mobile_no)
            VALUES (%s, %s, %s, %s, %s)
        ''', (username, email_id, uname, password, mobile_no))
        cursor_auth.execute('''
            INSERT INTO usage_reg (username, name, log_time)
            VALUES (%s, %s, %s)
        ''', (username, uname, log_time))
        con_auth.commit()
        speak("You have been signed up to your account")
        time.sleep(1)
        break

def login():
    login_successful = True
    while login_successful:
        login_input = input("Enter your username/email/mobile_no (type 'exit' to quit): ")
        if login_input.lower() == 'exit':
            sys.exit("User opted to exit the login process.")
        password_input = input("Enter your password: ")
        cursor_auth.execute('''
            SELECT * FROM users
            WHERE username = %s OR email_id = %s OR mobile_no = %s
        ''', (login_input, login_input, login_input))
        user_data = cursor_auth.fetchone()
        if user_data and user_data[3] == password_input:
            speak("Login successful!")
            log_time = time.strftime('%Y-%m-%d %H:%M:%S')  # Define log_time
            cursor_auth.execute('''
                INSERT INTO usage_reg (username, name, log_time)
                VALUES (%s, %s, %s)
            ''', (user_data[0], user_data[2], log_time))
            con_auth.commit()
            time.sleep(2)
            clear()
            login_successful = False
            print("You are logged in.")
            return
        else:
            speak("Login failed. Invalid username/email/mobile_no or password.")
            time.sleep(1)

def UserAuth():
    speak("Would you like to Signup or Login as an Existing User?")
    authdata = listen()
    if any(keyword in authdata.lower() for keyword in ["login", "old user", "not a new user", "not new user", "signin", "sign in", "log in"]):
        login()
    elif any(keyword in authdata.lower() for keyword in ["signup", "new user", "sign me up", "logup", "sign up", "log up"]):
        signup()

def activeuser():
    cursor_auth.execute('''
    SELECT username
    FROM usage_reg
    ORDER BY log_time DESC
    LIMIT 1
    ''')
    latest_username_data = cursor_auth.fetchone()
    loginusername = latest_username_data[0]
    return loginusername

def activeuname():
    cursor_auth.execute('''
    SELECT name
    FROM usage_reg
    ORDER BY log_time DESC
    LIMIT 1
    ''')
    latest_uname_data = cursor_auth.fetchone()
    loginusern = latest_uname_data[0]
    return loginusern
```

```
con_friday = sql.connect(host='localhost', user='root', password='krish4926', database='friday_allusers')
cursor_friday = con_friday.cursor()
cursor_friday.execute("SELECT name, address FROM websites")
websites = {name.lower(): address for name, address in cursor_friday.fetchall()}
current_user= activeuser()
uname = activeuname()
UserAuth()
while True:
    print_hi_art()
    Greet()
    query = listen()
    matches = [website_name for website_name in websites if website_name in query.lower()]
    if matches:
        x = f"Opening websites: {', '.join(matches)}"
        print(x)
        history(current_user, query, x)
        for match in matches:
            webbrowser.open(websites[match])
    elif any(keyword in query for keyword in ["song", "music"]):
        speak("Which song would you like to listen")
        songquery = listen().lower()
        speak("From where would you like to listen? Youtube, Spotify, Or local storage")
        songlocationquery = listen().lower()
        y = f"Playing Song: {songquery}"
        history(current_user, query, y)
        if any(keyword in songlocationquery for keyword in ["youtube", "you tube", "utube", "u tube"]):
            speak(f"Playing {songquery} on youtube:")
            pywhatkit.playonyt(songquery)
        elif "spotify" in songlocationquery:
            speak(f"Playing {songquery} on Spotify:")
            webbrowser.open(f"https://open.spotify.com/search/{songquery}")

        elif any(keyword in songlocationquery for keyword in ["locals", "local", "pc", "laptop", "folder"]):
            speak("Please type in the path for your music file")
        else :
            print("Unable to find your song")

    elif any(keyword in query for keyword in ["sleep", "quit", "bye", "turn off", "goodbye", "turnoff", "good
bye", "bi", "bye", "log off", "logoff", "shutdown", "shut down", "exit"]):
        speak("goodbye sir")
        exit()

    elif any(keyword in query for keyword in ["open settings", "settings", "open setting", "setting", "change
setting", "change settings", "configure", "configurations"]):
        settings()
        print("setings done")
        z = "Settings changed succesfully"
        history(current_user, query,z)

    elif "time" in query:
        current_time = datetime.datetime.now().strftime("%I:%M %p")
        a = f"Sir, the time is {current_time}."
        history(current_user, query, a)
        speak(a)
    elif "create to do" in query:
        create_todo()
        history(current_user, query, "To-do created.")

    elif "read to do" in query:
        read_todos()

    elif "update to do" in query:
        update_todo()

    elif "delete to do" in query:
        delete_todo()
```

OUTPUT-SCREENS & SAMPLE OUTPUTS

VOICE ASSISTANT FRIDAY

1)login screen

```
Would you like to Signup or Login as an Existing User?
Listening...
You said: login
Enter your username/email/mobile_no (type 'exit' to quit): Krish4926
Enter your password: krish4926
Login successful!
You are logged in.
```

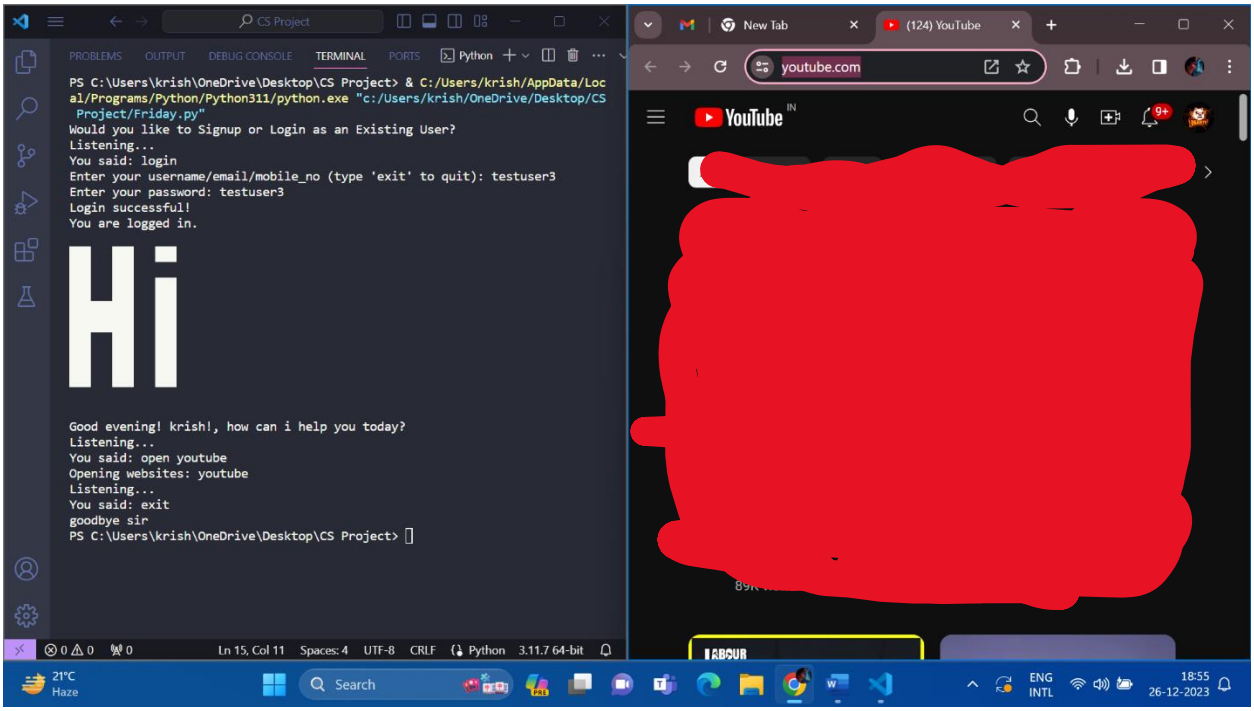

```
Good evening! krish!, how can i help you today?
Listening...
```

2)signup screen

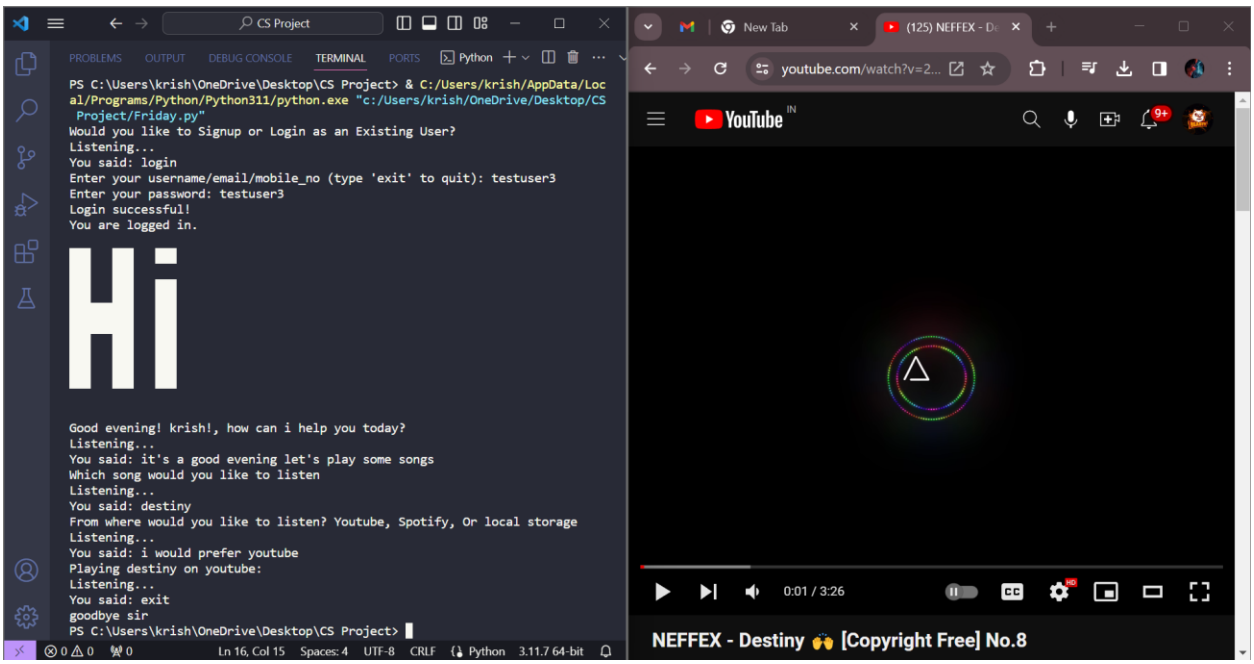
```
Listening...
You said: sign up
Please fill in the details. Type 'exit' at any time to quit.
Enter your username: testuser3
Enter your email: testuser3@gmail.com
Enter your full name: TestUser3
Enter your password: testuser3
Repeat your password: testuser3
Enter your mobile number: 1212121212
You have been signed up to your account
```


```
Good evening! krish!, how can i help you today?
Listening...
You said: exit
goodbye sir
```

3)OPEN WEBSITES



4)PLAY MUSIC



3)CREATE TO-DO LIST

```
Listening...
You said: create to do list
What task would you like to add to your to-do list?
Listening...
You said: coffee break
To-do added: coffee break
Listening...
You said: show to do list
To-do testuser3: write chemistry journal (Not Done), Created on: 2023-12-26 19:15:12
To-do testuser3: complete cs project (Not Done), Created on: 2023-12-26 19:16:54
To-do testuser3: complete physics homework (Not Done), Created on: 2023-12-26 19:19:18
To-do testuser3: coffee time (Not Done), Created on: 2023-12-26 19:33:04
To-do testuser3: coffee break (Not Done), Created on: 2023-12-26 19:34:52
Listening...
You said: update to do
To-do testuser3: write chemistry journal (Not Done), Created on: 2023-12-26 19:15:12
To-do testuser3: complete cs project (Not Done), Created on: 2023-12-26 19:16:54
To-do testuser3: complete physics homework (Not Done), Created on: 2023-12-26 19:19:18
To-do testuser3: coffee time (Not Done), Created on: 2023-12-26 19:33:04
To-do testuser3: coffee break (Not Done), Created on: 2023-12-26 19:34:52
Which to-do would you like to update? Please provide the to-do number.
```

3)TIME

```
Listening...
You said: what is the time
Sir, the time is 07:44 PM.
Listening...
You said: can you tell me the time now
Sir, the time is 07:44 PM.
```

SCOPE OF THE PROJECT

Scope of Voice Assistant - Friday:

The project has ample room for improvement and expansion. Key areas for future development include:

1. **Tkinter GUI Integration:**
    - Elevate user interaction by integrating Tkinter module functions, introducing a Graphical User Interface (GUI) for a more user-friendly experience when handling voice assistant functionalities.
2. **Enhanced Validation Checks:**
    - Strengthen system robustness by implementing additional validation checks, reducing errors, and enhancing overall reliability in voice command processing.
3. **Modification Module:**
    - Introduce a "Modification" module, enabling users to efficiently modify details and settings, ensuring the voice assistant remains flexible and adaptive to user preferences.

These enhancements align with the project's goal of continual improvement, providing Voice Assistant - Friday with advanced features and a dynamic user experience.

BIBLIOGRAPHY

COMPUTER SCIENCE IN PYTHON BY :–

SUMITA ARORA, Dhanpatrai Publication
PREETI ARORA, Sultan Chand Publication

Websites:

- www.icbse.com
- www.cbse.nic.in
- https://www.w3schools.com/python/
- https://www.tutorialspoint.com/python/index.htm