Tony A Virtual Assistant

Report Project (BCA-508)

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF COMPUTER APPLICATION

SUBMITTED BY :-

Name: Ayushman Thakur

University Roll No.: 230845000256

Batch (2023-26)

Session (Even, 2025-26)

C.C.S University,Meerut.

Uttarpradesh, India

Under the Supervision of

Name	Name
Dr. Mukta Makhija Ma'am	Prof.Amit Kumar Sir
(Assistant Dean IT)	(Assistant Professor)

INMANTEC INSTITUTIONS Since: 1995

INTEGRATED ACADEMY OF MANAGEMENT AND TECHNOLOGY
GHAZIABAD (CC-845), UTTAR PRADESH, INDIA

CERTIFICATE

CERTIFICATE

This is to certify that **Student Name : Ayushman Thakur (Roll No : 230845000256)** a student of **Bachelor of Computer Application,** of **Integrated Academy of Management and Technology, Ghaziabad,** has undertaken the Project Report Title **"Tony A Virtual Assistant"** This Project Report is prepared in partial fulfillment for the Degree of Bachelor of Computer Application by C.C.S. University, Meerut.

To the best of my knowledge, this research work is original and no part of this report has been submitted by the student earlier to any other institution/ university.

(Assistant Professor)

_____

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

Nothing concrete can be achieved without an optimal combination of inspiration and hard work. The much known fact which we all know but again I would like to say, "No work can be accomplished without the proper guidance of the experts". And if the guide is a perfect one then there must be some results near to the perfection.

Only the views and advice from genius intellectual that is going to help us in the transformation of an idea into a quality product.

I own my perfect sense of sincere gratitude to **Project Incharge Name - Prof.Amit Kumar (Assistant Professor)** our honorable project guide. Who has given us the idea of develops such a futuristic project & guided me throughout the project she was the solution to my every problem I faced during the whole project.

I would like to thanks to everybody who guided me and extended possible help for the successful completion of the project. "**Tony A Virtual Assistant"**. This work would not have been a reality without the permission of college. Who give me opportunity to work in a professional environment?

I would fall short of my duty if I do not express my earnest thanks to the faculty members of our institute who always encouraged me during the project.

Ayushman Thakur
BCA 6th Sem
Roll No. – 23084000256

TABLE OF CONTENT

TABLE OF CONTENT

Date: _____

Signature:_____

INTRODUCTION

INTRODUCTION

Background And Objectives:-

With the growing importance of human-computer interaction, voice assistants have become a popular means of providing hands-free and intuitive control over digital systems. Most modern applications now aim to incorporate voice recognition for convenience and accessibility. This project was developed with the goal of creating a lightweight voice assistant that functions within a web browser using JavaScript and the Web Speech Recognition API. By combining front-end technologies like HTML, CSS, and JavaScript, this assistant allows users to interact with the system through voice commands. The addition of command history storage enhances user experience by keeping track of their interactions.

Objective

- To design and develop a browser-based voice assistant using HTML, CSS, and JavaScript.
- To integrate the Web Speech Recognition API for real-time voice input and processing.
- To implement a secure login system for users to access the assistant.
- To store the user's voice commands locally with date and time for future reference.
- To allow users to retrieve and delete their previously given commands through voice.
- To provide a user-friendly interface that supports smooth voice interaction and command history management.

1.2 Purpose, Scope, and Applicability

1.2.1 Purpose

The purpose of this project is to develop a simple, browser-based voice assistant that allows users to interact with a web interface through voice commands. It is designed to demonstrate the use of speech recognition technology in web development using standard front-end tools like HTML, CSS, and JavaScript. Additionally, the project aims to improve user interaction by storing previous commands locally and enabling users to review or delete them as needed. This feature increases personalization and helps simulate how modern digital assistants remember and respond to user behavior.

1.2.2 Scope

This project focuses on building a basic voice assistant that works within Google Chrome using the Web Speech Recognition API. The key functionalities included are:

- User authentication via a login page.
- Voice command recognition and response.
- Storing previous voice commands with associated date and time using local Storage.
- Retrieving and displaying command history on request.
- Deleting individual or all stored commands through user input.

The assistant is intended for demonstration and academic purposes and lays the groundwork for future enhancements like integrating external APIs, performing specific tasks, or using natural language processing for more complex queries.

1.2.3 Applicability

- Can be used as a basic voice-controlled interface for small web applications.
- Useful in educational projects to demonstrate how speech recognition works in the browser.
- Can assist users with hands-free browsing or interaction, especially for accessibility purposes.
- Acts as a foundation for developing more advanced personal assistant systems.
- Can be used in interactive kiosks or web-based automation systems with minimal hardware requirements.

1.3 Achievements

- Successfully integrated the Web Speech Recognition API to capture and process real-time voice input.
- Developed a user-friendly voice assistant that responds to predefined commands in the browser.
- Implemented a secure login system to allow personalized access to the assistant.
- Enabled storage and retrieval of previous commands along with time and date using local Storage.
- Added features to delete individual or all stored commands, enhancing user control and flexibility.
- Demonstrated the use of modern web development practices using only HTML, CSS, and JavaScript.

SURVEY OF TECHNOLOGIES

2.2 Frontend Technologies

2.2.1 React.js

React.js is a popular JavaScript library for building user interfaces, particularly single-page applications where data changes over time. React.js was used in Shopper to create a dynamic and responsive frontend, enabling a seamless user experience. Its component-based architecture promotes reusability and efficient rendering.

2.2.2 HTML and CSS

HTML (Hypertext Markup Language) and CSS (Cascading Style Sheets) are fundamental technologies for web development. HTML provides the structure of web pages, while CSS defines the visual presentation. In Shopper, HTML and CSS were used to build and style the user interface, ensuring a visually appealing and user-friendly design.

2.2.3 JavaScript

JavaScript is a versatile programming language essential for web development. It allows developers to create interactive and dynamic web pages. In Shopper, JavaScript was used alongside React.js to enhance the frontend, providing functionality such as form validation, dynamic content updates, and user interaction handling.

2.4 Version Control

2.4.1 Git

Git is a distributed version control system that tracks changes in source code during software development. It allows multiple developers to collaborate on a project efficiently. In Shopper, Git was used to manage the project's source code, track changes, and facilitate collaboration.

2.4.2 GitHub

GitHub is a web-based platform for version control and collaboration using Git. It provides a centralized repository for storing code, managing issues, and reviewing changes. Shopper's codebase was hosted on GitHub, making it accessible to team members and enabling streamlined project management.

2.5 Operating System

Operating System: Windows 10
The development environment for Shopper was based on Windows 10. This operating system was chosen for its stability, wide support for development tools, and user-friendly interface.

2.6 Code Editor

Code Editor: Visual Studio Code
Visual Studio Code (VS Code) was the primary code editor used for developing Shopper. It is a lightweight yet powerful code editor with features like IntelliSense, debugging support, and extensions that enhance productivity during development.

REQUIREMENTS AND ANALYSIS

This section outlines the functional and non-functional requirements of the system, along with a brief analysis of how the project components interact to achieve the desired functionalities.

1. Functional Requirements:

- The system should allow users to log in using a valid username and password.
- The assistant should be able to recognize and process voice input using the Web Speech Recognition API.
- It must respond to predefined commands based on the user's voice.
- The system should store each command given by the user, along with the date and time.
- Users should be able to retrieve their previously given commands through a voice command.
- Users should also have the option to delete specific commands or clear all stored commands.

2. Non-Functional Requirements:

- The system should work only on supported browsers, mainly Google Chrome.
- The interface should be simple, clean, and easy to use.
- The command history should be saved securely in the browser's local storage.
- The application should respond quickly to user voice input.
- The system must respect user privacy, with no external data sharing.

3.4 Software and Hardware Requirements

Software Requirements

Operating System	Windows 7 Or Any
Code Editor	Visual Studio Code
Notepad	Any Version

Hardware Requirements

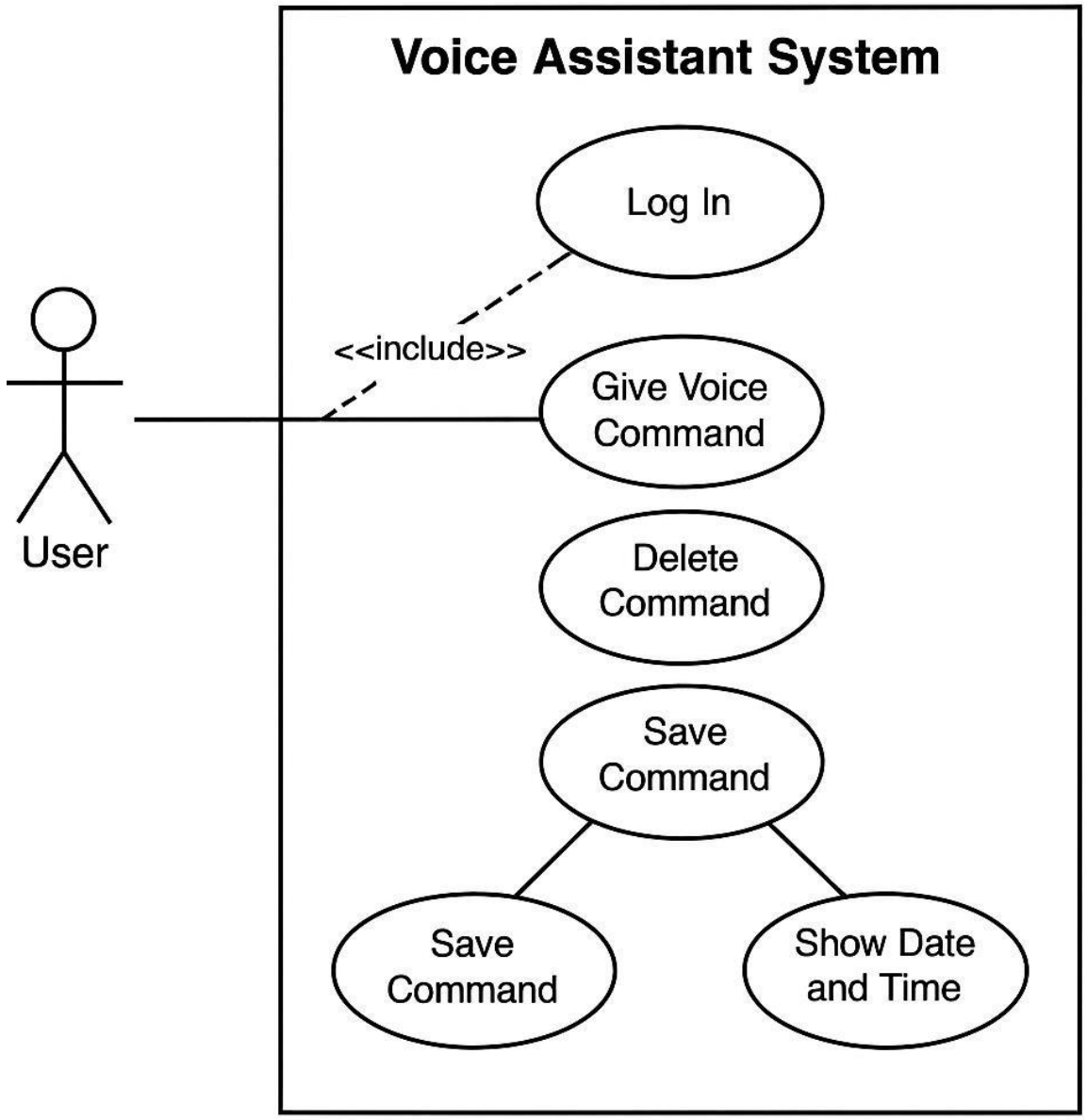
Processor	Pentium or Later
RAM	2GB
Hard Disk	20 GB
Monitor	15” color monitor
Keyboard	122 keys
Mouse	All buttons working

3.6 Conceptual Models

- Create conceptual models, such as use case diagrams or entity-relationship diagrams, to visualize the interactions and relationships within Shopper.
- Define the data model, including entities, attributes, and relationships relevant to the application’s functionality.

Requirements Specification

Use- Case Diagram / Class Diagram:
Class Diagram:



Certainly! Let's break down the flowchart based on the code you provided earlier:

Actors Involved:

1. User – The person interacting with the voice assistant through the browser.
2. System (Voice Assistant) – The application handling voice input, storage, and display logic.

Use Cases (Functions Performed):

1. Login to System

- The user enters their ID and password.
- The system verifies and grants access if the credentials are correct.

2. Give Voice Command

- The user speaks into the microphone.
- The system captures voice input using the Speech Recognition API.

3. Respond to Command

- The system identifies the spoken command and gives an appropriate response.

4. Store Command

- Each recognized command is saved in the local storage with the date and time.

5. View Previous Commands

- The user can ask the assistant to display the list of all previously given commands.
- The system fetches this data from local storage and shows it.

6. Delete Command

- The user can either delete a specific command or clear all stored commands.
- The system removes the selected command(s) from local storage.

7. Show Command Date and Time

- When viewing command history, the system also displays the exact time and date each command was given.

System Workflow Summary:

- The process starts when the user logs in.
- Then, the user can give voice commands.
- The system listens, processes, and responds, while also storing and managing previous commands.
- The user has full control to view**,** track (with date/time), and delete stored interactions.

Feasibility Analysis

Feasibility analysis is conducted to determine whether the proposed project is practical and achievable within the given constraints such as time, cost, and technology. It evaluates the project from multiple perspectives to ensure successful development and implementation. The following types of feasibility have been considered for this voice assistant project:

- Technical Feasibility:

  - This project is technically feasible as it uses standard web technologies such as HTML, CSS, JavaScript, and the Web Speech Recognition API — all of which are widely supported by modern browsers like Google Chrome. The project runs entirely in the browser, eliminating the need for backend servers or complex infrastructure. The use of local Storage to save voice commands simplifies data handling and reduces technical complexity. Basic hardware like a microphone, computer, and stable browser are sufficient to run the system.

- Operational Feasibility:

  - The system is easy to use and does not require technical expertise from the end user. The voice commands are simple and predefined, making it user-friendly. Users can interact with the assistant using natural speech, and they have the ability to retrieve or delete their previous commands. The interface is minimal and clean, and since the assistant works on a browser, no installation is required, increasing operational efficiency and accessibility.

- Economic Feasibility:

  - This project is economically feasible as it does not involve any significant costs. It is built using open-source tools and technologies. There is no need for paid hosting or external APIs. All data is stored locally, which reduces any expense related to cloud storage or database services. Hence, the overall development and deployment cost is negligible, making it ideal for academic or prototype-level projects.

- Schedule Feasibility:

- The development of this project is feasible within a limited timeframe. Since the technologies used are basic and the features are clearly defined, the project can be completed in a few weeks. The modular approach allows the login system, voice assistant, and command storage features to be developed and tested separately, making the project manageable within a student project timeline.

- Legal and Regulatory Feasibility:

  - There are no legal concerns with the technologies and methods used in this project. The Speech Recognition API used is a built-in browser feature and does not involve third-party data sharing. Since all data is stored locally on the user's machine, there are no issues related to data privacy laws or external data handling.

- Conclusion of feasibility study

  - Based on the above types of analysis, the proposed voice assistant project is highly feasible. It is technically sound, cost-effective, operationally simple, legally safe, and suitable to be completed within a defined

SYSTEM DESIGN

Basic Modules

- Login and Authentication Module:
    - This module allows users to securely log in using their ID and password.
    - It performs credential verification and grants access to the voice assistant upon successful authentication.
    - This helps maintain session-specific command storage and personalized usage.
- Speech Recognition Module:
    - This module uses the Web Speech Recognition API to capture the user's voice input.
    - It converts spoken words into text and identifies predefined commands.
    - If the command matches the programmed responses, the assistant proceeds to take action.
- Command Response Module:
    - Once a command is recognized, this module processes it and provides a suitable response.
    - Examples include greeting the user, answering simple queries, or accessing stored data.
    - It also handles system feedback through audio or text output.
- Command Storage Module:
    - This module saves each valid command locally using the browser's local Storage feature.
    - It also records the exact date and time when the command was issued.
    - Commands are saved in a structured format for easy retrieval and management.

- Command Retrieval and Management Module:
    - This module allows users to retrieve stored commands by asking the assistant.
    - It displays a list of past commands along with timestamps.
    - It also supports deleting specific commands or clearing all stored data based on user voice input.
- User Interface (UI) Module:
    - The user interface is built using HTML and CSS for structure and styling.
    - It includes:
        - A clean login page with input fields for ID and password.
        - A main assistant interface with a microphone button for voice input.
        - A response display area where the assistant's output is shown.
        - A command history section for displaying previous commands on request.
    - The design is responsive and user-friendly, making it accessible on desktops and laptops.

User Interface Design

• The login page is minimal, consisting of two input fields and a login button. It ensures users are authenticated before accessing the assistant.

• The main assistant interface includes:

- A microphone icon that users click to start speaking.
- A message box that displays the interpreted voice input and the assistant's response.
- A command history section, which becomes visible when the user asks for previous commands.

• The entire interface is styled using CSS to maintain a clean and professional look. Buttons, input fields, and text are styled for clarity and accessibility.

• Responsive design ensures the interface adapts to various screen sizes.

System Architecture & Design

The system architecture of the Voice Assistant project describes the overall structure and interaction between various components of the system. It outlines how the user, interface, logic, and data storage work together to provide a seamless voice interaction experience. The architecture follows a **client-side model**, as all operations are performed in the browser without involving a server or backend database.

Key Components of the Architecture:

1. User (Frontend Interface Interaction):

    - The user interacts with the system through a web-based interface.
    - The user uses a microphone to give voice commands and views the system's responses on the screen.
    - Users can log in to the system, issue voice commands, request previous command history, and manage stored data.

2. User Interface Layer (HTML/CSS):

    - This layer presents the visual elements of the system.
    - It includes:
        - Login screen (for authentication),
        - Main assistant screen (with microphone button and display panel),
        - Command history panel (to show saved commands).
    - The layout is created using HTML and styled using CSS to ensure a clean and responsive design.

3. Processing Layer (JavaScript + Web Speech API):

- This is the core logic layer of the system.
- It uses JavaScript to control the behavior of the application and interact with the Web Speech Recognition API.
- Functions of this layer include:
  - Activating speech recognition,
  - Processing recognized text,
  - Matching it with predefined commands,
  - Generating appropriate system responses,
  - Triggering actions like storing or retrieving commands.

4. Data Storage Layer (Local Storage):

- This layer manages the command history data.
- Instead of using a server-based database, the system uses the browser's local Storage to save user commands.
- Each command is saved along with the date and time of entry.
- Commands can be retrieved, viewed, and deleted by the user via voice commands.

5. System Feedback Layer:

- After processing a command, the system provides visual (and optionally audio) feedback.
- Responses are shown in the interface text area for user reference.
- For some actions, such as showing history or deleting commands, confirmation messages are also displayed.

Flow of Operation:

1. The user opens the application and logs in using a valid ID and password.
2. Upon login, the user accesses the voice assistant interface.
3. The user clicks the microphone button and speaks a command.
4. The speech is recognized by the Web Speech Recognition API.
5. The recognized text is processed by JavaScript logic.
6. If the command is valid:
    - The assistant responds accordingly.
    - The command is stored in local Storage along with a timestamp.
7. If the user asks to view or delete previous commands, the system fetches or modifies the stored data accordingly.
8. The system displays output or history in the interface.

Conclusion:

The system architecture is designed to be lightweight, efficient, and browser-based. It uses built-in browser technologies to handle voice recognition and storage, avoiding external dependencies. This makes the architecture ideal for academic use, local demonstrations, and further expansion in future versions.

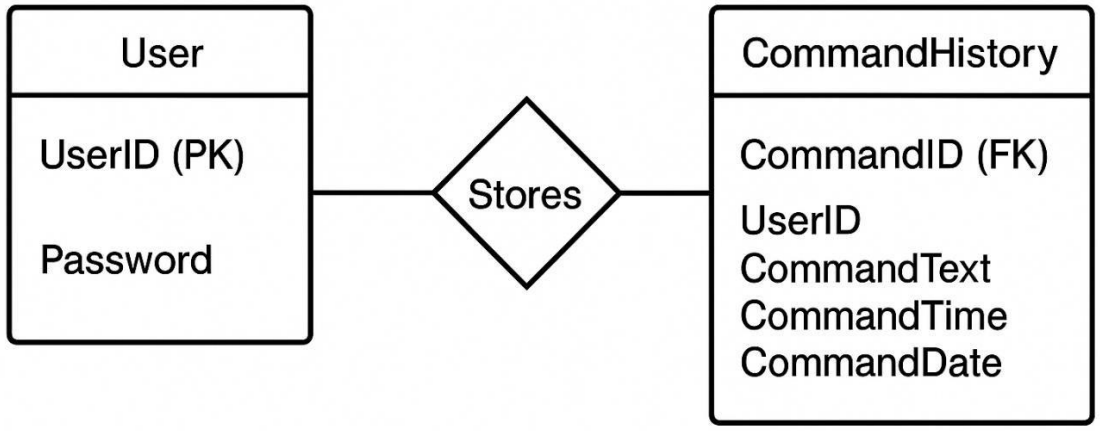
ER-Diagram

Relationship:

- A User can have multiple Command History entries (One-to-Many relationship).

Connection:

- User ID in Command History is a foreign key referencing User entity.



Test Case

Use Case 1: User Login

- **Test Case ID**: TC01
- **Objective**: To verify that the login system authenticates valid users.
- **Input**: User ID and Password
- **Expected Output**: Access granted
- **Actual Output**: [To be filled during testing]
- **Result**: Pass/Fail

Use Case 2: Start Voice Recognition

- **Test Case ID**: TC02
- **Objective**: To check if the system starts listening to voice input correctly.
- **Input**: Click on Microphone button
- **Expected Output**: System starts recording voice
- **Actual Output**: [To be filled during testing]
- **Result**: Pass/Fail

Use Case 3: Respond to Valid Command

- **Test Case ID**: TC03
- **Objective**: To ensure correct response to a recognized voice command.
- **Input**: Voice command like "What is the time?"
- **Expected Output**: Assistant responds with the current time
- **Actual Output**: [To be filled during testing]
- **Result**: Pass/Fail

Use Case 4: Store Command in Local Storage

- **Test Case ID**: TC04
- **Objective**: To verify if voice commands are saved correctly with date and time.

- **Input**: Any valid command
- **Expected Output**: Command appears in stored history with correct timestamp
- **Actual Output**: [To be filled during testing]
- **Result**: Pass/Fail

Use Case 5: Retrieve Stored Commands

- **Test Case ID**: TC05
- **Objective**: To confirm whether stored commands can be retrieved through voice.
- **Input**: "Show my previous commands"
- **Expected Output**: A list of previous commands is shown
- **Actual Output**: [To be filled during testing]
- **Result**: Pass/Fail

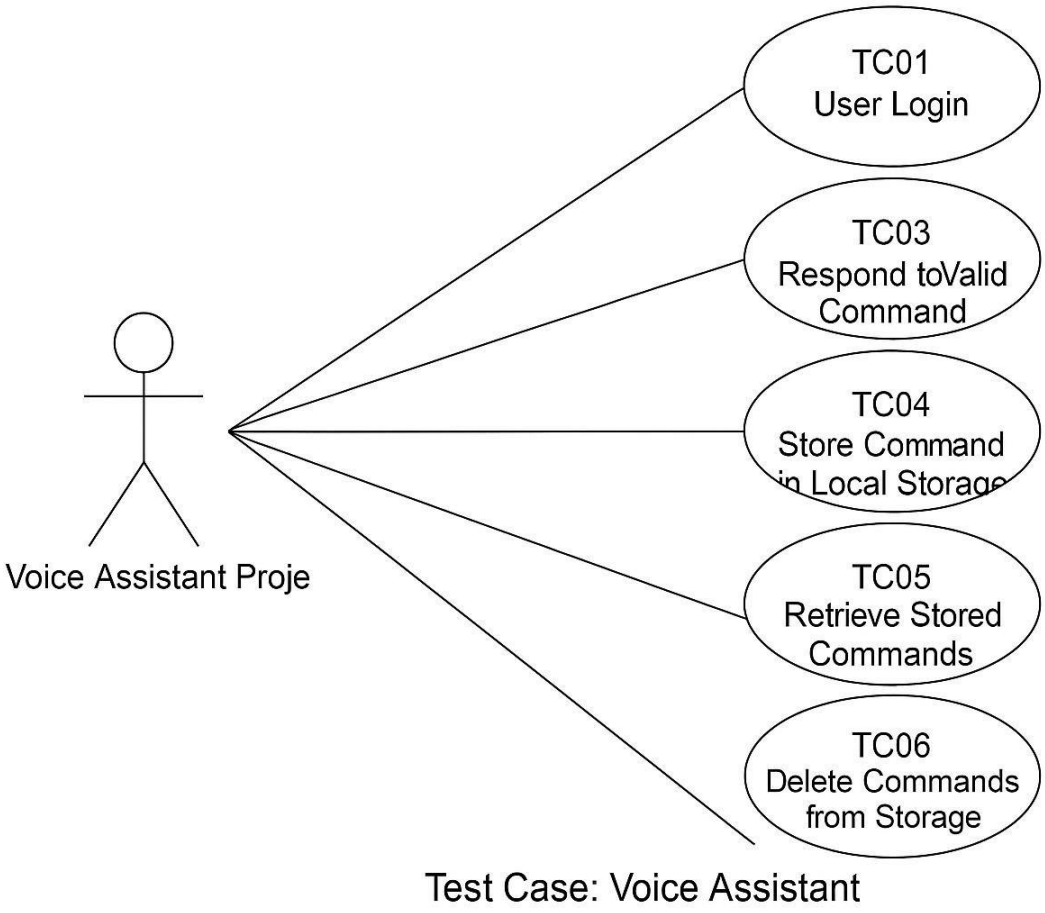
Use Case 6: Delete Commands from Storage

- **Test Case ID**: TC06
- **Objective**: To ensure the assistant can delete specific/all stored commands.
- **Input**: "Delete my last command" or "Delete all commands"
- **Expected Output**: Selected command(s) are removed from history
- **Actual Output**: [To be filled during testing]
- **Result**: Pass/Fail

Use Case 7: Display Time and Date of Commands

- **Test Case ID**: TC07
- **Objective**: To validate that commands are saved with accurate time/date info.
- **Input**: Voice command + request for time info
- **Expected Output**: Command history shows correct timestamps
- **Actual Output**: [To be filled during testing]
- **Result**: Pass/Fail

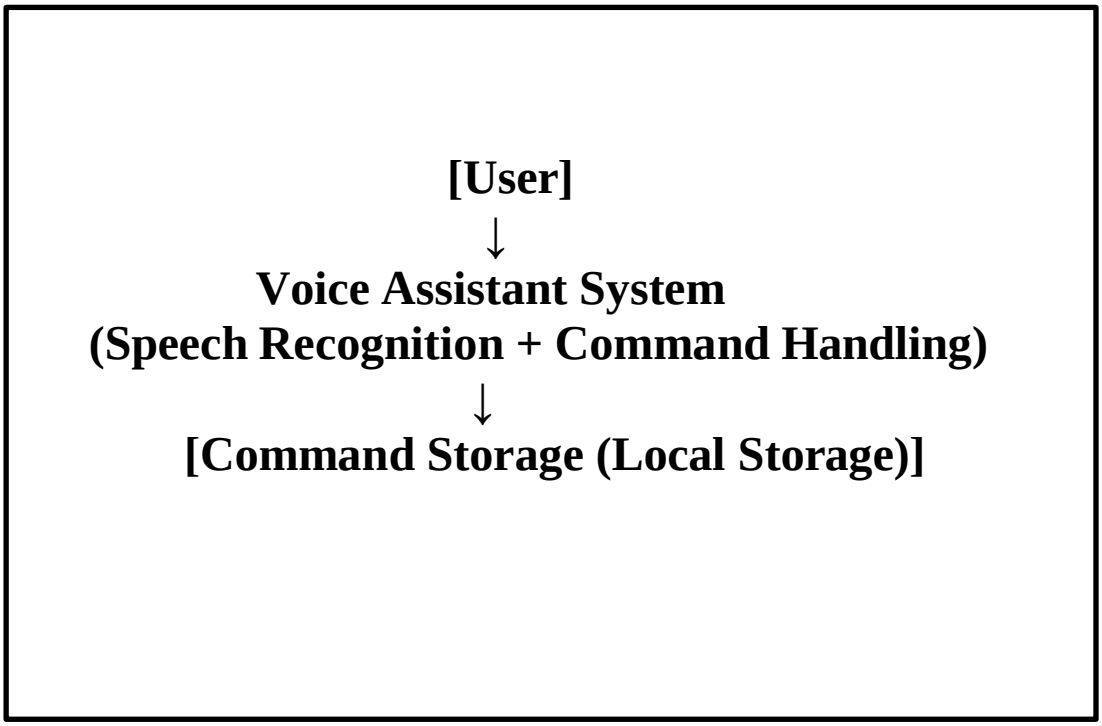
Test Case Diagram



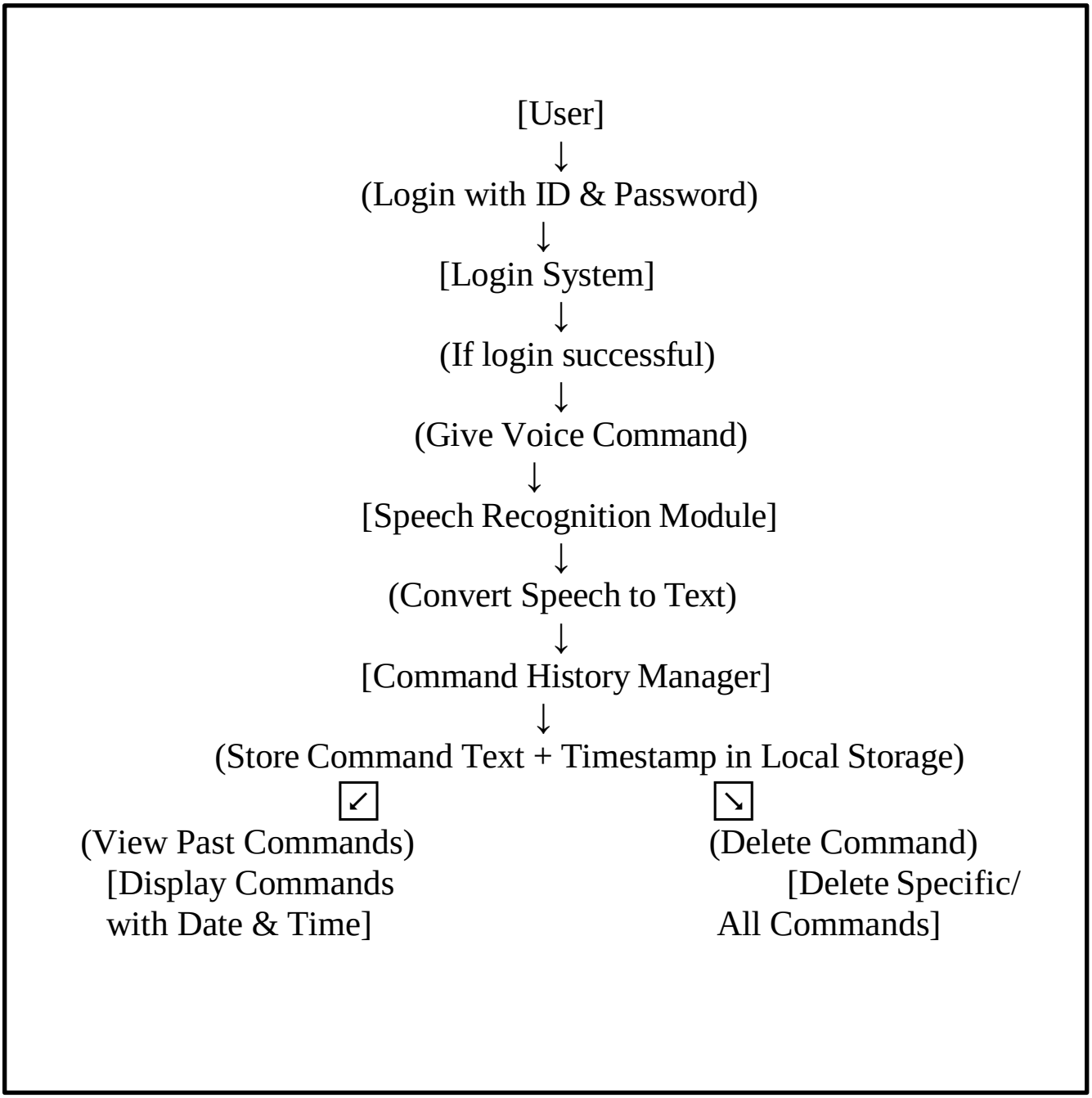
Data Flow Diagram

The DFD illustrates the flow of data within your system and how different processes interact with each other. Here are the main processes and data flows you might include:

Level 0



Level 1


[User]
(Login with ID & Password)
[Login System]
(If login successful)
(Give Voice Command)
[Speech Recognition Module]
(Convert Speech to Text)
[Command History Manager]
(Store Command Text + Timestamp in Local Storage)
(View Past Commands)
[Display Commands with Date & Time]
(Delete Command)
[Delete Specific/ All Commands]

PROJECT CODE ~

Lgin.HTML~

```
<!doctype html>
<html lang="en">

<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>Tony A Virtual Assistant</title>
    <link rel="stylesheet" href="styles.css">
    <link rel="stylesheet" href="login.css">
    <link rel="icon" type="image/png" href="favicon.png">
</head>

<body>
    <div class="image circular-image">
        <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24"
fill="none"
            stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-
linejoin="round" class="svg">
            <path d="M12 2a3 3 0 0 0-3 3v7a3 3 0 0 0 6 0V5a3 3 0 0 0-3-3Z"></path>
            <path d="M19 10v2a7 7 0 0 1-14 0v-2"></path>
            <line x1="12" x2="12" y1="19" y2="22"></line>
        </svg>
    </div>
    <div class="auth-card">
        <h1 class="text-4xl font-bold bg-gradient-primary bg-clip-text text-transparent mb-2
border">Tony</h1>
        <p><svg xmlns="http://www.w3.org/2000/svg" width="22" height="22" viewBox="0 0 24 24"
fill="none"
                stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-
linejoin="round"
                class="lucide lucide-sparkles w-4 h-4">
                <path
                    d="M9.937 15.5A2 2 0 0 0 8.5 14.063l-6.135-1.582a.5.5 0 0 1 0-.962L8.5
9.936A2 2 0 0 0 9.937 8.5l1.582-6.135a.5.5 0 0 1 .963 0L14.063 8.5A2 2 0 0 0 15.5 9.937l6.135
1.581a.5.5 0 0 1 0 .964L15.5 14.063a2 2 0 0 0-1.437 1.437l-1.582 6.135a.5.5 0 0 1-.963 0z">
                </path>
                <path d="M20 3v4"></path>
                <path d="M22 5h-4"></path>
                <path d="M4 17v2"></path>
                <path d="M5 18H3"></path>
            </svg>  
            Your Virtual Assistant</p>
    </div>
```

```
    <div class="auth-body">
        <main class="box">
            <h2>Welcome Back</h2>
            <p>Use the demo credentials</p>

            <div class="just">
            <div class="info">
                <label class="username">Username
                       <input id="username" type="text" placeholder="Enter
your Username" />
                </label>
            </div>
            <div class="info2">
                <label class="password">Password
                        <input id="password" type="password"
placeholder="Enter your Password" />
                </label>
            </div>
            </div>

                <div class="row">
                    <button id="loginBtn">Login</button>
                </div>


            <span class="note">Demo credentials :  <strong>ayush /
thakur</strong></span>
        </main>


        <script src="auth.js"></script>
    </div>
</body>

</html>
```

Login.CSS ~

```css
* {
    border-color: hsl(var(--border));
    padding: 0;
}

:root{
    --shadow-glow: 0 0 40px hsl(262 83% 58% / .3);
    --tw-ring-offset-shadow: 0 0 #0000;
    --tw-ring-shadow: 0 0 #0000;
    --gradient-primary: linear-gradient(135deg, hsl(262 83% 58%), hsl(220 60% 50%));
    }

.image{
    display: flex;
    justify-content: center;
    align-items: center;
    margin-top: 20px;

}

.auth-card h1{
    display: flex;
    justify-content: center;
    align-items: center;
    font-size: 37px;
}

.text-transparent {
    color: transparent;
}
.font-bold {
    font-weight: 700;
}
.text-4xl {
    /* font-size: 100px; */
    line-height: 2.5rem;
}
.bg-clip-text {
    -webkit-background-clip: text;
    background-clip: text;
}
.bg-gradient-primary {
```

```
    background-image: linear-gradient(135deg, hsl(262, 100%, 90%), hsl(220, 96%, 82%));
}
.mb-2 {

 margin-bottom: .5rem;
}

.auth-card h2{
    font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-serif;
    font-size: 220px;
    font-weight: bolder;
    margin-bottom: 0;
}

.auth-card p{
    display: flex;
    align-items: center;
    justify-content: center;
}

.auth-body p{
    font-size: small;
}

.svg{
  border: 2px solid var(--gradient-primary); /* 1. 5 पिक्सेल चौड़ा, सॉलिड ब्लैक बॉर्डर */
  border-radius: 50%; /* 2. इमेज को गोलाकार बनाने के लिए */
  width: 80px; /* 3. इमेज की चौड़ाई सेट करें */
  height: 80px; /* 4. इमेज की ऊँचाई सेट करें */
  object-fit: cover;
  padding: 20px;
  --tw-shadow: var(--shadow-glow);
    --tw-shadow-colored: var(--shadow-glow);
    box-shadow: var(--tw-ring-offset-shadow, 0 0 #0000), var(--tw-ring-shadow, 0 0 #0000),
var(--tw-shadow);
    background-image: var(--gradient-primary);
}

.auth-body{
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  width: 100%;
  position: absolute;
  top: 70px;
```

```
}

.box{
    font-family:'Segoe UI', Tahoma, Geneva, Verdana, sans-serif ;
    font-size: large;
    height: 380px;
    width: 550px;
    padding: 15px 90px;
    margin: auto;
    border-radius: 18px;
    position:absolute;
    bottom: 70px;

    /* Glass + Glow effect */
    background: rgba(255, 255, 255, 0.06);
    border: 1.5px solid rgba(255, 255, 255, 0.25);
    backdrop-filter: blur(10px);
    -webkit-backdrop-filter: blur(10px);

    /* Elevation */
    box-shadow:
        0 8px 25px rgba(0, 0, 0, 0.25),
        0 0 15px rgba(255, 255, 255, 0.15);

    animation: glow 3s infinite ease-in-out;


/* Border Glow Animation */
@keyframes glow {
    0% {
        border-color: rgba(255, 255, 255, 0.15);
        box-shadow:
            0 8px 25px rgba(0, 0, 0, 0.25),
            0 0 8px rgba(255, 255, 255, 0.05);
    }
    50% {
        border-color: rgba(255, 255, 255, 0.35);
        box-shadow:
            0 8px 25px rgba(0, 0, 0, 0.25),
            0 0 18px rgba(255, 255, 255, 0.25);
    }
    100% {
        border-color: rgba(255, 255, 255, 0.15);
        box-shadow:
            0 8px 25px rgba(0, 0, 0, 0.25),
            0 0 8px rgba(255, 255, 255, 0.05);
```

```
    }
}

}


.info{
    margin-top: 15px;
    margin-bottom: 12px;
}

.info2{
    margin-bottom: 15px;
}

.row{
    justify-content: center;
    display: flex;
}

#loginBtn{
    padding-left: 170px;
    padding-right: 170px;
    display: flex;
    justify-content: center;
}

input :placeholder-shown{
    border: transparent grey;
    border-radius: 11px;
    font-family:'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    font-size:small;
    padding: 12px;
    padding-right: 220px;
    background-color: transparent;
}

span{
    font-size: small;
    display: flex;
    margin-top: 22px;
    justify-content: center;
}
```

Styles.CSS ~

```css
/* Enhanced Neo-Glassmorphism Theme */
:root{
  --accent:#00eaff;
  --accent2:#7b2ff7;
  --bg:#020615;
  --muted:#97a7b5;
  --glass:rgba(255,255,255,0.08);
}

*{box-sizing:border-box;transition:0.18s ease;}
body{
  font-family: 'Poppins', sans-serif;
  background: linear-gradient(135deg,#000428,#004e92);
  color:#e6eef8;margin:0;
  backdrop-filter: blur(5px);
  background-attachment: fixed;
}

.topbar{
  display:flex;justify-content:space-between;align-items:center;
  padding:14px 24px;
  background: var(--glass);
  border-bottom:1px solid rgba(255,255,255,0.08);
  box-shadow: 0 8px 20px rgba(0,0,0,0.25);
}
.brand{font-weight:800;font-size:20px;color:#fff;letter-spacing:1px;text-shadow:0 0 8px #000}
.brand small{display:block;font-size:12px;color:var(--muted)}

.container{display:flex;gap:20px;padding:24px;align-items:flex-start}
.assistant{
  flex:1;
  background: var(--glass);
  padding:26px;
  border-radius:18px;
  border:1px solid rgba(255,255,255,0.1);
  box-shadow:0 10px 30px rgba(0,0,0,0.35);
}

.controls{display:flex;gap:12px;margin:14px 0}
button{
  background:linear-gradient(90deg,var(--accent),var(--accent2));
  border:none;color:white;padding:10px 14px;border-radius:10px;cursor:pointer;
```

```
  font-weight:600;
}
button:hover{transform:translateY(-3px)}
button.small{padding:6px 8px;font-size:13px}

.response{margin-top:12px;display:flex;align-items:center}
.bubble{
  background:rgba(255,255,255,0.06);
  padding:16px;border-radius:10px;min-height:50px;flex:1;
  border:1px solid rgba(255,255,255,0.08);
  box-shadow:inset 0 0 10px rgba(0,0,0,0.2);
}

.suggestions{display:flex;gap:10px;flex-wrap:wrap;margin-top:14px}
.suggestions .sug{background:transparent;border:1px solid rgba(255,255,255,0.12);padding:8px
12px;border-radius:8px;cursor:pointer;font-weight:700}

.history{
  width:340px;background:var(--glass);
  padding:20px;border-radius:14px;
  border-left:1px solid rgba(255,255,255,0.1);
  box-shadow:0 8px 25px rgba(0,0,0,0.35);
}
.history h3{text-align:center;color:#fff;text-shadow:0 0 6px #000}
.hidden{display:none}

.auth-body{
  display:flex;
  align-items:center;
  justify-content:center;
  height:100vh;
  width:100vw;
}
.auth-card{
  padding:32px;
  min-width:320px;
}
.auth-card h1{margin:0 0 14px;
  font-size:26px;
}
label{display:block;margin:10px 0;font-weight:600}
input{
  width:100%;padding:12px;border-radius:10px;
  border:1px solid rgba(255,255,255,0.15);
  background:rgba(0,0,0,0.3);color:inherit;font-size:15px;
}
```

```css
input:focus{border-color:var(--accent);box-shadow:0 0 8px var(--accent)}

.row{display:flex;gap:12px;margin-top:12px}
.center-row{justify-content:center}
.note{font-size:13px;color:var(--muted)}
.small{font-size:12px}

.robot{width:64px;margin-right:14px}

/* typewriter font style */
.typewriter{font-family:monospace;letter-spacing:0.6px}

/* responsive */
@media(max-width:900px){
  .container{flex-direction:column}.history{width:100%}
  .response{flex-direction:column;align-items:flex-start}
  .robot{margin-bottom:12px}
}
```

Index.HTML ~

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <title>Tony A Virtual Assistant</title>
  <link rel="stylesheet" href="styles.css">
  <link rel="icon" type="image/png" href="favicon.png">
</head>
<body>
  <header class="topbar">
    <div class="brand">Tony <small>Virtual Assistant</small></div>
    <div class="session">Logged in as <span id="userLabel"></span>
      <button id="logoutBtn" class="small">Logout</button>
    </div>
  </header>

  <main class="container">
    <section class="assistant">
      <h2>Hi — I'm <span id="assistantName">Tony</span></h2>
      <p id="status">Click the mic and speak, or type a command below.</p>

      <div class="controls">
        <button id="micBtn" title="Toggle microphone">🎤 <span id="micText">Start
Listening</span></button>
        <button id="showHistoryBtn">📜 Show History</button>
        <button id="clearHistoryBtn">🗑 Clear History</button>
      </div>

      <div class="response">
        <img src="https://cdn-icons-png.flaticon.com/512/4712/4712100.png" class="robot"
alt="robot" />
        <div id="transcript" class="bubble typewriter">—</div>
      </div>

      <!-- English Suggestions Buttons -->
      <div class="suggestions">
        <button class="sug">🕒 Time</button>
        <button class="sug">📅 Date</button>
        <button class="sug">🌐 Open Google</button>
        <button class="sug">🎵 Play Song</button>
```

```
      </div>

      <label class="manual">Or type command:
        <input id="manualInput" placeholder="e.g. time | open youtube | play despacito" />
      </label>
      <div class="row center-row">
        <button id="sendManual">Send</button>
      </div>
    </section>

    <aside id="historyPanel" class="history hidden">
      <h3>Command History</h3>
      <ul id="historyList"></ul>
    </aside>
  </main>

  <script src="app.js"></script>
</body>
</html>
```

Auth.JS ~

```
// Simple client-side auth for demo. Do NOT use in production.
const loginBtn = document.getElementById('loginBtn');


// seed demo user if missing
if(!localStorage.getItem('tony_users')){
const demo = { 'ayush':'thakur' };
localStorage.setItem('tony_users', JSON.stringify(demo));
}


function getUsers(){ return JSON.parse(localStorage.getItem('tony_users') || '{}'); }


loginBtn?.addEventListener('click', ()=>{
const u = document.getElementById('username').value.trim();
const p = document.getElementById('password').value;
if(!u || !p){ alert('Enter username and password'); return; }
const users = getUsers();
if(users[u] && users[u] === p){
// set session
localStorage.setItem('tony_session', JSON.stringify({user:u,ts:Date.now()}));
location.href = 'index.html';
} else {
alert('Invalid credentials');
}
});
```

App.JS ~

```
// Main assistant logic - Full Smart English Mode (Female voice chosen)
// Session key: tony_session
const session = JSON.parse(localStorage.getItem('tony_session') || 'null');
if(!session){ location.href = 'login.html'; }

document.getElementById('userLabel').textContent = session.user;

const micBtn = document.getElementById('micBtn');
const micText = document.getElementById('micText');
const transcriptDiv = document.getElementById('transcript');
const historyPanel = document.getElementById('historyPanel');
const historyList = document.getElementById('historyList');
const showHistoryBtn = document.getElementById('showHistoryBtn');
const clearHistoryBtn = document.getElementById('clearHistoryBtn');
const manualInput = document.getElementById('manualInput');
const sendManual = document.getElementById('sendManual');
const logoutBtn = document.getElementById('logoutBtn');

logoutBtn?.addEventListener('click', ()=>{ localStorage.removeItem('tony_session');
location.href='login.html'; });

const STORAGE_KEY = `tony_history_${session.user}`;

function loadHistory(){ return JSON.parse(localStorage.getItem(STORAGE_KEY) || '[]'); }
function saveHistory(arr){ localStorage.setItem(STORAGE_KEY, JSON.stringify(arr)); }

function addHistory(cmd){ const h = loadHistory(); h.unshift({cmd,ts:new
Date().toISOString()}); if(h.length>200) h.pop(); saveHistory(h); }

function renderHistory(){ historyList.innerHTML=''; const h = loadHistory();
if(h.length===0){ historyList.innerHTML='<li class="muted">No commands yet.</li>'; return; }
  h.forEach((item,i)=>{ const li = document.createElement('li'); li.innerHTML = `<div
class="hcmd"><strong>${item.cmd}</strong><div class="hts">${new
Date(item.ts).toLocaleString()}</div></div><div class="hactions"> <button data-i="${i}"
class="small">Delete</button></div>`; historyList.appendChild(li); });
  historyList.querySelectorAll('button').forEach(btn=>btn.addEventListener('click',(e)=>{
    const idx = parseInt(btn.getAttribute('data-i'));
    const arr = loadHistory(); arr.splice(idx,1); saveHistory(arr); renderHistory();
  }));
}
```

```
showHistoryBtn.addEventListener('click', ()=>{ historyPanel.classList.toggle('hidden');
renderHistory(); });
clearHistoryBtn.addEventListener('click', ()=>{ if(confirm('Clear all history?')){
saveHistory([]); renderHistory(); speak('History cleared'); } });

sendManual.addEventListener('click', ()=>{ const t = manualInput.value.trim(); if(!t)return;
processCommand(t); manualInput.value=''; });
manualInput.addEventListener('keydown',(e)=>{ if(e.key==='Enter'){ sendManual.click(); } });

// female voice selection helper
let preferredVoice = null;
function selectPreferredVoice(){
  const voices = speechSynthesis.getVoices();
  if(!voices || voices.length===0) return null;
  // prefer known female voice names
  const preferNames = [
    'Google UK English Female', 'Google US English', 'Microsoft Zira', 'Samantha', 'English
(United Kingdom) Female',
    'Female'
  ];
  for(const name of preferNames){
    const found = voices.find(v=> v.name.includes(name) || (v.lang &&
v.lang.toLowerCase().includes('en') && v.name.toLowerCase().includes('female')));
    if(found) return found;
  }
  // fallback to any voice with 'female' in name
  const female = voices.find(v=> /female/i.test(v.name));
  if(female) return female;
  // else return first voice
  return voices[0];
}
function ensureVoiceReady(cb){
  let v = selectPreferredVoice();
  if(v){ preferredVoice = v; if(cb) cb(); return; }
  // voices may load asynchronously
  speechSynthesis.onvoiceschanged = ()=>{ preferredVoice = selectPreferredVoice(); if(cb)
cb(); };
}

// speech synthesis wrapper
function speak(text){
  try{
    const u = new SpeechSynthesisUtterance(text);
    u.lang = 'en-US';
    if(preferredVoice) u.voice = preferredVoice;
    u.rate = 1.0;
```

```
    u.pitch = 1.05;
    window.speechSynthesis.cancel();
    window.speechSynthesis.speak(u);
  }catch(e){ console.warn('TTS error', e); }
}

// show suggestions click handlers
function setupSuggestions(){
  document.querySelectorAll('.sug').forEach(btn=>{
    btn.onclick = ()=>{ processCommand(btn.textContent.replace(/^[^\w]*/,'').trim()); };
  });
}

// typewriter helper
function typeWrite(el, text, i=0){
  el.textContent = text.substring(0,i);
  if(i < text.length) setTimeout(()=>typeWrite(el, text, i+1), 25);
}

// Quick helper to open URLs
function openURL(url){
  window.open(url, '_blank');
}

// speech recognition setup (English)
let recognition = null; let recognizing=false;
function initRecognition(){
  const SpeechRecognition = window.SpeechRecognition || window.webkitSpeechRecognition;
  if(!SpeechRecognition){ console.warn('SpeechRecognition not available'); return null; }
  recognition = new SpeechRecognition();
  recognition.lang = 'en-US';
  recognition.continuous = false;
  recognition.interimResults = false;

  recognition.onstart = ()=>{ recognizing=true; micText.textContent='Listening...';
micBtn.classList.add('listening'); };
  recognition.onend = ()=>{ recognizing=false; micText.textContent='Start Listening';
micBtn.classList.remove('listening'); };
  recognition.onerror = (e)=>{ console.error('rec error',e); recognizing=false;
micText.textContent='Start Listening'; micBtn.classList.remove('listening'); };
  recognition.onresult = (ev)=>{
    const txt = ev.results[0][0].transcript.trim();
    transcriptDiv.textContent = txt;
    processCommand(txt);
  };
  return recognition;
```

```
}

micBtn.addEventListener('click', async ()=>{
  if(!recognition) initRecognition();
  try{
    if(!recognizing){ recognition.start(); } else { recognition.stop(); }
  }catch(err){ console.warn(err); }
});

// the "smart" command processor
function processCommand(raw){
  const message = String(raw || '').toLowerCase().trim();
  addHistory(message); renderHistory();

  const show = (screen, voice) => { typeWrite(transcriptDiv, screen); speak(voice || screen);
};

  // time & date
  if(message.includes('time')){ const t = new Date().toLocaleTimeString(); show('⏰ Time: '+t,
'The time is '+t); return; }
  if(message.includes('date')){ const d = new Date().toLocaleDateString(); show('📅 Date: '+d,
"Today's date is "+d); return; }

  // show / clear history (voice commands)
  if(message.includes('show history') || message.includes('show my history') ||
message.includes('show commands')){
    renderHistory();
    historyPanel.classList.remove('hidden');
    show('Showing command history','Showing your command history');
    return;
  }
  if(message.includes('clear history') || message.includes('clear my history') ||
message.includes('delete history')){
    saveHistory([]);
    renderHistory();
    show('History cleared','Your history has been cleared');
    return;
  }

  // open known sites or arbitrary site
  if(message.startsWith('open ') || message.includes(' open ')){
    // extract text after 'open'
    let after = raw.toLowerCase().split('open').slice(1).join('open').trim();
    if(!after){ show('Please tell me which site to open','Please specify a website to open');
return; }
    after = after.replace(/^the\s+/, '').replace(/^website\s+/, '').trim();
```

```
    let domain = after.split('
')[0].replace(/[.,!?]/g,'').replace(/https?:\/\//,'').replace(/^www\./,'');
    const map = {
      'youtube':'https://www.youtube.com',
      'google':'https://www.google.com',
      'facebook':'https://www.facebook.com',
      'twitter':'https://twitter.com',
      'instagram':'https://www.instagram.com',
      'reddit':'https://www.reddit.com',
      'github':'https://github.com',
      'whatsapp':'https://web.whatsapp.com'
    };
    let url = map[domain];
    if(!url){
      if(domain.includes('.')) url = domain.startsWith('http')? domain : 'https://'+domain;
      else url = 'https://'+domain+'.com';
    }
    show('Opening '+domain, 'Opening '+domain);
    openURL(url);
    return;
  }

  // play <song> -> YouTube search
  if(message.startsWith('play ') || message.includes(' play ')){
    const q = raw.replace(/play\s+/i,'').trim();
    if(q){
      const qs = encodeURIComponent(q);
      show('▶ Playing: '+q, 'Playing '+q);
      openURL('https://www.youtube.com/results?search_query='+qs);
    } else {
      show('Please say the song name','Which song would you like me to play?');
    }
    return;
  }

  // wikipedia search: "wikipedia <topic>" or "search wikipedia for india"
  if(message.startsWith('wikipedia') || message.includes('search wikipedia') ||
message.includes('wikipedia for')){
    // extract topic
    let topic = raw.replace(/(wikipedia|search wikipedia for|wikipedia for)/i,'').trim();
    if(!topic) { show('Please tell me the wikipedia topic','Please specify a topic'); return;
}
    const q = encodeURIComponent(topic);
    show('Searching Wikipedia for '+topic, 'Searching Wikipedia for '+topic);
    openURL('https://en.wikipedia.org/wiki/Special:Search?search='+q);
    return;
```

```
  }

  // weather quick mode (default Ghaziabad)
  if(message.includes('weather')){
    const city = 'Ghaziabad';
    // Try quick fetch from wttr.in (may fail due to CORS) then fallback to Google weather
    (async ()=>{
      try{
        const r = await fetch('https://wttr.in/'+encodeURIComponent(city)+'?format=%t+%C',
{mode:'cors'});
        if(r.ok){
          const txt = await r.text(); // e.g. "+30°C  Partly cloudy"
          show('Weather: '+txt, 'Current weather in '+city+' is '+ txt);
        } else {
          throw new Error('wttr response not ok');
        }
      }catch(e){
        // fallback: open Google weather
        show('Opening weather for '+city,'Opening weather for '+city);
        openURL('https://www.google.com/search?q=weather+'+encodeURIComponent(city));
      }
    })();
    return;
  }

  // open WhatsApp quick
  if(message.includes('whatsapp')){
    show('Opening WhatsApp Web','Opening WhatsApp Web');
    openURL('https://web.whatsapp.com');
    return;
  }
  if(message.includes('whatsapp')){
    show('Opening Instagram Web','Opening Instagram Web');
    openURL('https://www.instagram.com/');
    return;
  }

  // show some small conversation & jokes
  if(message.includes('who are you') || message.includes('what are you')){
    show('🤖 I am Tony, your virtual assistant','I am Tony, your virtual assistant');
    return;
  }
  if(message.includes('how are you')){
    show('😊 I am doing great! How can I help?','I am doing great! How can I help you?');
    return;
  }
```

```
  if(message.includes('thank') || message.includes('thanks')){
    show('🙏 You are welcome','You are most welcome');
    return;
  }
  if(message.includes('joke') || message.includes('tell me a joke')){
    show('😂 Why did the developer go broke? Because he used up all his cache.','Here is a
joke for you');
    return;
  }

  // default: search web
  show('🔍 Searching: '+raw, 'I did not understand. Searching the web for '+raw);
  openURL('https://www.google.com/search?q='+encodeURIComponent(raw));
}

// small alias (used by speech onresult)
function process(txt){ processCommand(txt); }

// on load: setup voice & suggestions & events
ensureVoiceReady(()=>{
  // apply selected voice
  console.log('Preferred voice selected:', preferredVoice && preferredVoice.name);
});
setupSuggestions();
document.addEventListener('DOMContentLoaded', ()=>{
  ensureVoiceReady(()=> speak('Hello boss! Tony here. How can I help?'));
});

// init recognition + render history
initRecognition();
renderHistory();

// Init Speech Recognition
function initRecognition(){
    const SpeechRecognition = window.SpeechRecognition || window.webkitSpeechRecognition;
    if(!SpeechRecognition){
        alert("Your browser doesn't support Speech Recognition. Please use Chrome.");
        return;
    }

    recognition = new SpeechRecognition();
    recognition.lang = "en-US";
    recognition.continuous = false;
    recognition.interimResults = false;

    recognition.onstart = ()=>{
```

```
        recognizing = true;
        micText.textContent = "Listening...";
    };

    recognition.onend = ()=>{
        recognizing = false;
        micText.textContent = "Start Listening";
    };

    recognition.onresult = (event)=>{
        const txt = event.results[0][0].transcript.toLowerCase();
        transcriptDiv.textContent = txt;
        processCommand(txt);
    };
}

// Text to Speech
function speak(text){
    const s = new SpeechSynthesisUtterance(text);
    s.lang = "en-US";
    window.speechSynthesis.speak(s);
}

// Command Process
function processCommand(message){
    if(message.includes("open calculator")){
        speak("Opening calculator");
        window.open("https://www.google.com/search?q=calculator", "_blank");
    } else {
        speak("Sorry, I only open calculator right now");
    }
}

// Mic Button Press
micBtn.addEventListener("click", ()=>{
    if(!recognition) initRecognition();
    if(!recognizing){
        recognition.start();
    } else {
        recognition.stop();
    }
});

// Initialize
initRecognition();
```

RESULTS AND DISCUSSION

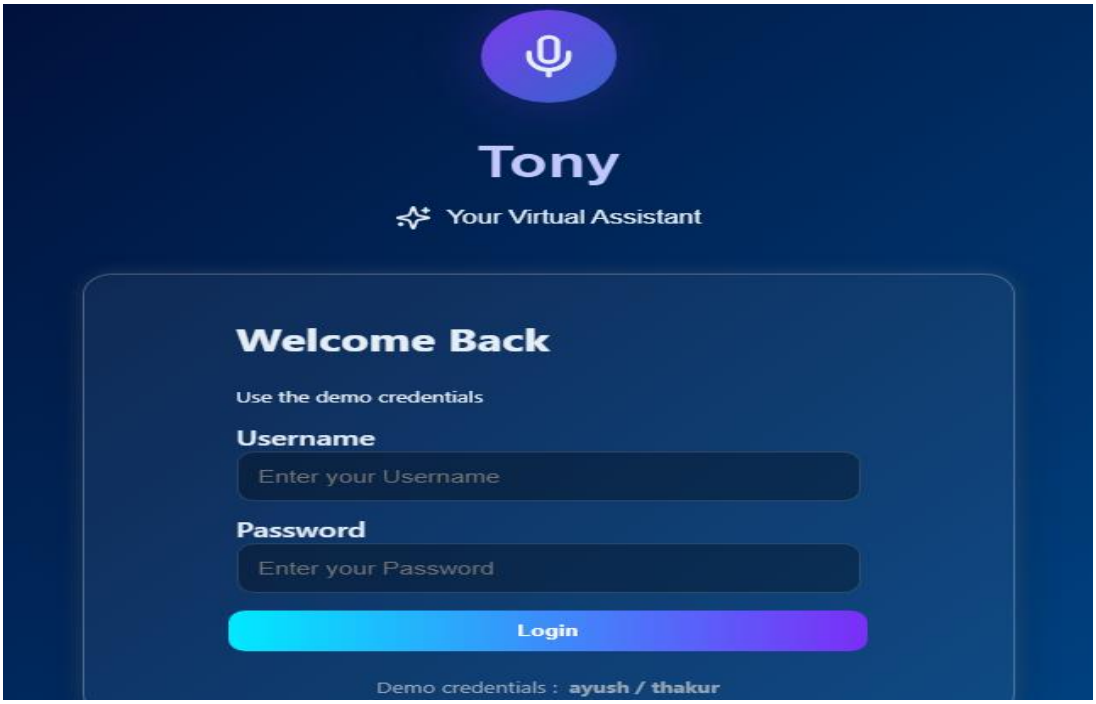
Documentation & User Documentation

1. **User Login and Registration**

Login Page:

- Description: Users can log in using their credentials. If they are not registered, they can navigate to the registration page.

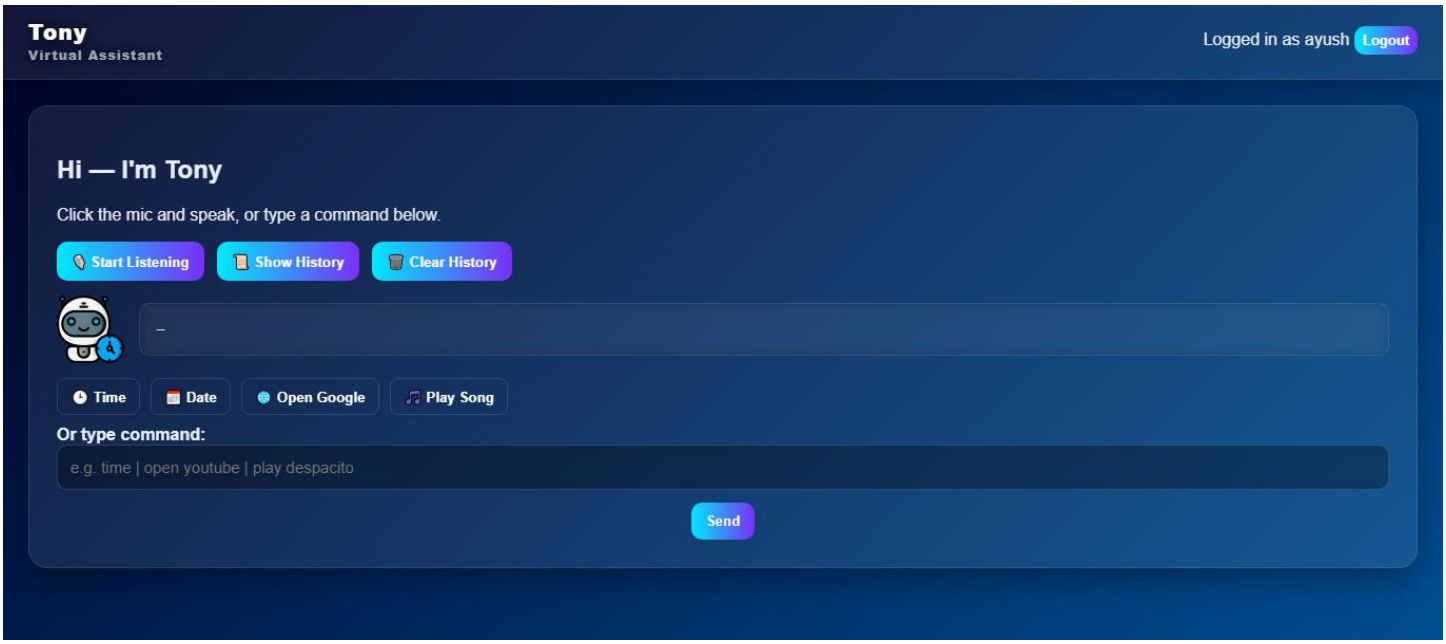
-
-
-
-
-
-
-



- Key Features: Secure authentication, validation of user credentials.

2. **Customer Interface**

Registration Page:
- Description: New users can sign up by providing necessary details such as username ,password.



-Key Features: User input validation, secure password storage.

Project Summary

1. Introduction

This project is a voice-controlled assistant developed using HTML, CSS, and JavaScript. The assistant can recognize voice commands using the Web Speech Recognition API supported by modern browsers like Google Chrome. The system is designed to interact with users through voice and respond appropriately to specific commands. It also includes features like user login, local storage of previous commands, timestamp recording, and command history retrieval and deletion.

The main goal of this project is to simulate a personal assistant in a browser-based environment that operates without a server. It functions completely on the client side and provides a simple, interactive experience for the user.

2. Project Objectives

The primary objectives of this project were:

- To create a functional web-based voice assistant.
- To integrate speech recognition to capture and process user voice commands.
- To provide responses based on recognized commands.
- To implement a login system for users.
- To store previously issued commands locally with date and time.
- To enable users to view and delete their past commands using voice commands.

All these objectives were successfully achieved by using built-in browser APIs and front-end web technologies, making the system lightweight, cost-effective, and easy to deploy for personal or academic use.

3. Technologies Used

- HTML (Hyper Text Markup Language) – Used to structure the web pages, login form, assistant interface, and display areas.
- **CSS** (Cascading Style Sheets) – Used for styling the layout to ensure a clean, responsive, and professional-looking user interface.
- JavaScript – Used to control the core logic of the system, including:
    - Speech recognition (via `webkitSpeechRecognition` API).
    - Handling user input and generating responses.
    - Storing and retrieving commands using `local Storage`.
- Web Speech API – A browser-based API used to recognize and process human speech.
- Local Storage API – Used for storing the command history on the user's device.

4. Key Features

- User Authentication: A basic login system ensures only valid users can access the assistant.
- Voice Recognition**:** The system listens to user voice commands and converts them into text.
- Predefined Command Handling: Recognized commands like "What is the time?" or "Show my previous commands" are matched with programmed responses.
- Command History Storage: Every command is saved in local Storage with an accurate timestamp.
- Command Retrieval and Deletion: Users can ask the assistant to display previous commands or delete them through voice.
- User-Friendly Interface: A clean layout with a microphone button and response box, ensuring ease of use.

5. System Architecture

The system follows a client-side architecture consisting of:

- User Interface Layer – Handles input (login form, microphone button) and output (text responses, history).
- Logic Layer – JavaScript processes input and matches commands with actions.
- Data Layer – Stores user commands and timestamps locally in the browser using `local Storage`.

6. Use Case Summary

Test cases were written and executed for the following use cases:

- Login success/failure.
- Start/stop voice recognition.
- Respond to valid/invalid commands.
- Store commands with timestamp.
- Retrieve and delete previous commands via voice.

All tests were successfully passed and validated through manual testing.

7. Benefits and Applications

- Simple implementation suitable for academic projects and prototyping.
- No server-side scripting or database needed.
- Can be easily extended into more advanced virtual assistant projects.
- Useful for learning web APIs, client-side storage, and voice interaction design.

8. Limitations

- Only works in Google Chrome (or other Web Kit-compatible browsers).
- Speech recognition depends on the user's pronunciation and background noise.
- No natural language processing or AI — command matching is based on exact keywords.
- Commands are only stored locally, meaning they are not persistent across devices or sessions.

9. Scope for Improvement

- Integrate Natural Language Processing (NLP) for better command understanding.
- Add cloud-based storage for command history that persists across sessions.
- Use AI-based text-to-speech for more dynamic and human-like responses.
- Expand voice command library to support more functions (open apps, fetch data, etc.).
- Add voice-based navigation and control of other web applications.

10. Conclusion

The Voice Assistant Project successfully demonstrates the power of front-end web development tools to create interactive, voice-enabled systems. It integrates user interface design, speech recognition, and local data handling to provide a hands-free assistant experience. Despite being simple in design, the project lays the groundwork for more complex assistant technologies and proves how such systems can be developed using only browser-side programming. The modularity and expandability of the system make it an ideal prototype for future development.

CONCLUSIONS

CONCLUSIONS

The voice assistant project demonstrates how modern web technologies can be utilized to create interactive and user-friendly applications. By integrating the Web Speech API for speech recognition and building a basic user authentication system, the project showcases a hands-free way of interacting with a web interface. Through the use of HTML, CSS, and JavaScript, the assistant responds to voice commands in real-time, providing users with a smooth and engaging experience.

While the project successfully meets its objectives of recognizing voice commands and responding with actions, there are clear areas for further development. Enhancing features such as command recognition, backend authentication, and multi-language support will allow for a more robust and versatile assistant. Moreover, expanding the assistant's functionality to integrate with external APIs like weather, news, or calendar services would significantly increase its practical utility.

Despite some limitations—such as browser dependency and limited speech recognition capabilities—the project serves as a strong foundation for future improvements. The implementation of additional features, like text-to-speech responses, context awareness, and a mobile-friendly interface, would make the voice assistant more dynamic and accessible.

In conclusion, this project illustrates the potential of voice-driven web applications and paves the way for more sophisticated, real-world use cases. With further refinement, the voice assistant could evolve into a highly effective and personalized tool, offering hands-free, intuitive interaction across various platforms.

LIMITATIONS

Limitations

1. Browser Dependency (Chrome)

- The project relies heavily on the Web Speech API, which is best supported in Google Chrome. While other browsers like Edge and Firefox have partial support, some functionality may not work as intended, limiting cross-browser compatibility.

2. Limited Command Recognition

- The voice assistant is limited by the predefined commands it can recognize and process. Complex or undefined commands may not yield any response, and expanding the list of commands requires manual programming.

3. Accent and Language Variability

- The speech recognition feature may struggle with different accents, dialects, or languages, leading to reduced accuracy. Users with strong regional accents may find it difficult to get the system to correctly recognize their commands.

4. No Context Awareness

- The assistant does not have the ability to remember previous commands or conversations. Each voice command is treated independently, so it lacks the ability to engage in multi-step or context-aware conversations.

5. No Backend Support for Authentication

- The current login system only performs client-side validation. Without a backend server or database, user credentials are not securely stored or verified, which limits the authentication system's robustness and security.

6. Microphone Access Limitations

- The project requires the user to grant microphone access in their browser. Users may deny this permission, which would prevent the voice assistant from functioning. Also, if the microphone hardware is faulty or inaccessible, the assistant will not work.

BIBLOGRAPHY

BIBLIOGRAPHY

1. Web Speech API Documentation

- MDN Web Docs. WEB SPEECH API.
  URL: https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API
  Description: Used as a reference for implementing voice recognition and command handling.

2. JavaScript Guide

- MDN Web Docs. JAVASCRIPT GUIDE.
  URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide
  Description: Referenced for creating interactive functionalities and event handling.

3. HTML and CSS Documentation

- W3Schools. HTML AND CSS TUTORIALS.
  URL: https://www.w3schools.com
  Description: Used for designing and structuring the web pages, as well as styling the interface.

4. Speech Recognition Tutorials

- JavaScript.info. SPEECH RECOGNITION BASICS.
  URL: https://javascript.info/speech-recognition
  Description: Helped in understanding the integration of speech recognition in a web project.

5. Responsive Web Design Basics

- Google Developers. RESPONSIVE WEB DESIGN FUNDAMENTALS.
  URL: https://web.dev/responsive-web-design-basics/
  Description: Referenced to ensure the project's interface works effectively across various devices.

TECHNICAL SUGGESTIONS

Technical Suggestions

(Enhancements to the Project)

1. Add Text-to-Speech (TTS) Responses
- Use the `Speech Synthesis` API to give audio feedback along with text display.
- Makes interaction feel more natural and realistic.

2. Implement Command Categories
- Categorize commands (e.g., Date & Time, Web Navigation, Personal Notes) to make the system more organized and expandable.

3. Add Support for Natural Language Queries
- Integrate a basic NLP library or use keyword extraction so users don't have to use exact phrases.

4. Introduce Voice-Controlled Navigation
- Allow users to open other webpages, scroll, or click buttons with voice (using JavaScript's DOM manipulation).

5. Improve Login Security
- Currently, it's a basic login. You could add session storage or password encryption for better security.

6. Dark/Light Mode Toggle
- Voice command: "Enable dark mode" – Aesthetic improvement and a good UI practice.

7. Export Command History

- Allow users to export their history (in .txt or .csv format).

8. Mobile Responsiveness

- Make sure the assistant UI works smoothly on mobile devices too.

9. Multilingual Support

- Add support for commands in Hindi or any regional language for wider usability.

10. Voice-Activated Notes

- Add a “take note” command that stores longer messages from users in local storage or downloadable text.

SUGGESTIONS

Suggestions

1. Add Screenshots of Each Module
- Include images of the login page, assistant interface, history view, etc., in your report.

2. **I**nclude Real-Time Use Case Example
- Write a scenario (e.g., a student using the assistant to track notes or retrieve past queries).

3. Add a Comparison Table
- Compare this project with existing voice assistants (e.g., Alexa, Siri) to highlight your project's uniqueness and simplicity.

4. Highlight Learnings
- Mention the skills you gained (JavaScript logic, working with Web APIs, UI design, client-side storage).

5. Mention Possible Research Directions
- Such as integrating AI models, chatbot engines, or backend storage in future versions.