CP-III Project Report on

JARVIS – DESKTOP ASSISTANT

at

U. V. Patel College of Engineering

Internal Guide:

Prof. Manan D. Thakkar

Prepared By:

Karna Patel (21012021068)

Het Patel (21012021064)

Kandarp Pandya(21012021056)

B. Tech Semester-VII

(Information Technology)

Nov-Dec, 2024

Submitted to,

Department of Information Technology

U.V. Patel College of Engineering

Ganpat University, Ganpat Vidyanagar - 384 012

U.V. PATEL COLLEGE

OF

ENGINEERING

C E R T I F I C A T E

TO WHOM SO EVER IT MAY CONCERN

This is to certify that Mr. Karna SanjayKumar Patel student of **B.Tech. Semester-VII (Information Technology)** has completed his full semester Capstone Project - III work titled "**Jarvis - Desktop Assistant**" satisfactorily in partial fulfillment of the requirement of Bachelor of Technology degree of Information Technology of Ganpat University, Ganpat Vidyanagar in the year 2024-2025.

Prof. Manan D. Thakkar
College Project Guide

Dr. Devang S. Pandya
Head, Information Technology

U.V. PATEL COLLEGE

OF

ENGINEERING

C E R T I F I C A T E

TO WHOM SO EVER IT MAY CONCERN

This is to certify that Mr. Het Vasudevbhai Patel student of **B.Tech. SemesterVII (Information Technology)** has completed his full semester Capstone Project - III work titled "**Jarvis - Desktop Assistant**" satisfactorily in partial fulfillment of the requirement of Bachelor of Technology degree of Information Technology of Ganpat University, Ganpat Vidyanagar in the year 2024-2025.

Prof. Manan D. Thakkar
College Project Guide

Dr. Devang S. Pandya
Head, Information Technology

U.V. PATEL COLLEGE
OF
ENGINEERING

C E R T I F I C A T E

TO WHOM SO EVER IT MAY CONCERN

This is to certify that Mr. Kandarp Dushyantkumar Pandya student of **B.Tech. Semester-VII (Information Technology)** has completed his full semester Capstone Project - III work titled "**Jarvis - Desktop Assistant**" satisfactorily in partial fulfillment of the requirement of Bachelor of Technology degree of Information Technology of Ganpat University, Ganpat Vidyanagar in the year 2024-2025.

Prof. Manan D. Thakkar
College Project Guide

Dr. Devang S. Pandya
Head, Information Technology

ACKNOWLEDGEMENT

I express our sincere gratitude towards internal guide **Prof. Manan D. Thakkar** for his constant help, encouragement, suggestions and inspiration throughout the project work. Without his invaluable advice, suggestions and assistance it would not have been possible for me to complete this project work.

Thank you, sir

Abstract

The rapid advancements in artificial intelligence and natural language processing have made it possible to create intelligent desktop assistants capable of streamlining everyday tasks. This project, titled "Jarvis - Desktop Assistant," focuses on developing a robust and user-friendly virtual assistant designed to enhance productivity and simplify interactions with a personal computer.

The Jarvis Desktop Assistant is an advanced AI-powered assistant designed to enhance productivity and enable seamless smart home integration. Built using Python, the assistant leverages powerful voice recognition and natural language processing to perform various tasks such as searching for information, managing files, and interacting with online resources. The graphical user interface (GUI) is crafted using HTML, CSS, and JavaScript, ensuring a responsive and user-friendly experience. For backend operations, PHP is employed to manage server-side functionality and database interactions.

In addition to its desktop capabilities, Jarvis integrates smart home functionalities through Arduino Uno, enabling users to control devices like lights. With simple voice commands, users can turn lights on or off, enhancing convenience and energy efficiency. The system employs Python to communicate with the Arduino board via serial communication, making the integration smooth and efficient.

This project showcases the synergy of software development and hardware control, combining cutting-edge web technologies and IOT principles to create a versatile assistant that empowers users in both digital and physical environments.

Table of Contents

List of Figures

List of Tables

CHAPTER 1-INTRODUCTION

1.1. Project Overview

The goal of the Jarvis Desktop Assistant project is to develop an AI-powered virtual assistant that can interact with users via voice and text commands to perform various tasks, improve productivity, and provide information. The assistant is inspired by the fictional AI "Jarvis" from the Iron Man movies, known for its advanced capabilities and user-friendly interactions.

1.2. Background

The idea for a Jarvis Desktop Assistant comes from the AI named "J.A.R.V.I.S." in the Iron Man movies. In the films, J.A.R.V.I.S. is a smart assistant that helps Tony Stark with everything, from controlling his technology to providing information instantly. This cool concept inspired people to create a similar tool that could help us with everyday tasks.

1.3. Purpose of System

The Purpose System in Jarvis helps the assistant figure out what you want to do. When you ask it something, it tries to understand your goal so it can give you better help or do what you need more effectively.

1.3.1. Problem Statement

People often spend a lot of time doing repetitive tasks on their computers, like searching for information, managing schedules, or controlling devices. It can be tiring and time-consuming. The challenge is to create an intelligent assistant that can understand voice and text commands to help people with these tasks quickly and easily, making their daily routines more efficient and less stressful.

1.3.2 Project Aim

The aim of the Jarvis Desktop Assistant project is to create a smart, easy-to-use assistant that helps people manage their daily tasks, find information, and control their computer just by using voice or text commands, making life simpler and more efficient.

1.3.3 Project Objectives

To develop a smart desktop assistant that can perform a variety of tasks, including voice recognition, task automation, information retrieval, and interaction with different applications, mimicking the functionality of a virtual assistant like J.A.R.V.I.S. from the Iron Man movies.

1.4 Project Scope

"Project scope" refers to the range of tasks or features that the assistant is set up to handle for a specific project or goal. It defines what the assistant can and cannot do within that project, helping to keep things organized and focused. Essentially, it outlines the boundaries of what the assistant will help with and ensures it stays on track with the project's objectives.

1.5 Significance of the Project

- **Saves Time and Effort:**
  It handles repetitive tasks like organizing schedules, searching for information, and managing files, so users can focus on what really matters.
- **Easy to Use for Everyone:**
  With voice and text commands, it's simple and accessible, making it useful for people with different needs or preferences.
- **All-in-One Helper:**
  Instead of juggling multiple apps, the assistant brings everything together in one place, making things simpler and faster.
- **Shows the Power of AI:**
  This project is a great example of how artificial intelligence can make our daily routines better and more efficient.
- **Encourages Learning and Growth:**
  For developers and learners, it's a chance to understand and work with modern technologies like AI and voice recognition.
- **Keeps Up with Modern Trends:**
  Virtual assistants are becoming a big part of our lives, and this project helps show how they can be useful in managing tasks on a computer.

1.6 Benefits of the Jarvis Desktop Assistant

- **Personalized Assistance:**
  The assistant can be tailored to individual user needs, learning preferences and adapting to provide more relevant suggestions and solutions over time.
- **Increased Productivity:**
  By automating routine tasks, such as sending emails, managing files, or setting reminders, the assistant helps users focus on more critical and creative work.
- **Better Time Management:**
  With features like schedule organization, task reminders, and quick information retrieval, users can effectively plan and manage their time.
- **Enhanced User Experience:**
  The combination of voice and text interaction makes the assistant accessible and easy to use, catering to diverse users, including those with limited technical skills.
- **Integration with Multiple Applications:**
  The assistant can work seamlessly with various tools and applications, enabling smooth transitions between tasks without the need to switch between different programs manually.
- **Learning Opportunities for Developers:**
  The project offers a platform for developers and students to experiment with artificial intelligence, natural language processing, and task automation, fostering innovation and technical growth.

- **Encourages Technological Adoption:**
  By demonstrating the practical use of AI, the assistant helps promote the adoption of smart technologies in everyday life, making advanced tools more accessible to the general public.
- **Reduces Stress:**
  By simplifying complex workflows and handling repetitive tasks, the assistant can make work less stressful and more enjoyable, improving the overall user experience.

1.7 Structure of the Report

- The rest of the report is structured as follows
- Literature Review: Overview of existing technologies and solutions relevant to desktop assistants and IOT integration.
- System Design: Detailed architecture, modules, and design components of the system.
- Implementation: Description of the tools, technologies, and methodologies used in the project.
- Testing and Evaluation: Analysis of system performance, limitations, and improvements.
- Conclusion and Future Work: Summary of findings and potential future enhancements.

CHAPTER -2 LITERATURE SURVEY

Category	Existing Systems/Technologies	Features	Insights	Application in Jarvis
Virtual Assistants [1]	Siri, Alexa, Google Assistant	Voice recognition, contextual responses, smart home control, web search.	Emphasizes ease of use, third-party integration,and natural language understanding.	Implement `SpeechRecognition` and `NLTK` for voice commands and conversational responses.
	Cortana	File management, reminders, OS integration.	Strong in managing desktop functionalities.	AddOS-level integrations for file handling and application management.
GUI Development [2]	HTML, CSS, JavaScript	Create responsive, dynamic, and cross-platform interfaces.	Highly customizable and integrates with backend technologies.	Design a user-friendly dashboard for Jarvis using these technologies.
	Streamlit	Python-based web app development.	Simplifies GUI development for Python projects.	Create a web-based interface for Jarvis using Streamlit.
	PHP	Server-side scripting and dynamic content management.	Efficient for backend logic and database operations.	Manage user preferences and settings using PHP for server-side processing.
Smart Home Integration [3]	Arduino Uno	Control IoT devices (lights, fans) with sensors and relays.	Cost-effective and programmable for IoT applications.	Use Arduino to turn lights on/off with Python for serial communication.
	ESP8266, ESP32	Wireless modules for IoT control.	Adds Wi-Fi connectivity for remote operation.	Enable remote control of smart devices via web or mobile apps.
Core Functionalities [4]	SpeechRecognition (Python)	Converts speech to text.	Essential for voice-based interactions.	Implement voice recognition for receiving commands.
	NLTK/spaCy	Natural language understanding.	Improves conversational responses.	Process and interpret user queries more naturally.
	PyAutoGUI	Automates mouse and keyboard operations.	Useful for automating desktop tasks.	Add task automation like opening files, typing, or launching apps.
	pySerial	Serial communication with Arduino.	Simplifies integration with hardware for IoT control.	Use it to send commands to Arduino for smart home automation.
Existing Frameworks [5]	Jarvis by Microsoft	Enterprise-grade automation with modularity and scalability.	Demonstrates the importance of modular systems.	Design modular components for features like voice control and smart home integration.

	Mycroft AI (Open-source)	Customizable virtual assistant.	Highlights extensibility and user customization.	Allow users to add plugins and configure modules.

[Table 2.1 :Literature Survey]

The Literature Survey Table compares various virtual assistant systems, focusing on their features, functionalities, and technologies like Python, JavaScript, and Arduino. It highlights capabilities such as task management, smart home integration, and natural language processing, identifying gaps that the Jarvis Desktop Assistant aims to address, such as enhanced customization.

CHAPTER -3 FEASIBILITY ANALYSIS

1. Technical Feasibility : The Jarvis Desktop Assistant is about figuring out if it's possible to make the assistant do what you want with the current technology. It checks if the software and hardware can handle the tasks and features you're asking for. In simple terms, it's about making sure the tech can actually do what you need it to do.

2. Time Schedule Feasibility : Time Schedule Feasibility in the Jarvis desktop assistant means checking whether your planned schedule or tasks can be realistically completed within the time you have available. It's like asking, "Can I fit all my appointments and to-dos into my day without overloading myself?" Jarvis helps you figure this out by analyzing your calendar and deadlines, making sure your plans are doable.

3. Operational Feasibility : Figuring out if a plan or task is practical and manageable in your everyday work routine. It's like asking, "Can I actually carry out this task or project with the resources and systems I have?" Jarvis helps assess whether your plans can realistically fit into how you work and whether you have what you need to get them done smoothly.

4. Implementation feasibility : It is about checking if you can actually put your plan or idea into action. It's like asking, "Can I successfully start and complete this project with the tools and resources I have?" Jarvis helps you figure out if your plan can be carried out effectively and if everything needed is in place to make it happen.

5. Economic Feasibility : It is about figuring out if a project or plan is worth the cost. It's like asking, "Will the money and resources I spend on this be worth the benefits I get back?" Jarvis helps you evaluate whether the financial investment needed is reasonable compared to the expected results.

CHAPTER -4 SOFTWARE REQUIREMENT SPECIFICATIONS (SRS)

4.1 Functional Requirement

- **Sign in :**

1. Inputs:

- Username/Email Field: Text input for user credentials. Validates email format or username.
- Password Field: Password input to securely capture the user's password. Meets security criteria.
- Sign In Button: Submits the form.
- Forgot Password Link: Redirects to password recovery.
- Sign Up Link: Redirects to registration.
- Remember Me Checkbox: Option to stay signed in.

2. Outputs:

- Error Messages:
- "Invalid username/email or password."
- "Please fill in all required fields."
- "Your account is locked due to multiple failed attempts."
- Success Message: "Welcome back, [User's Name]!"
- Redirection: To home screen/dashboard or error page based on
- authentication result. Loading Indicator: Shows when processing login.

3. Process

- Input Collection: Gather username/email and password.
- Validation:
  a) Client-side: Check format and completeness.
  b) Server-side: Authenticate credentials.
- Authentication: Verify credentials and account status.
- Session Management: Create and manage user session. Handle "Remember Me" functionality.
- Error Handling: Provide feedback and options for retry or recovery.
- Redirection: Redirect based on login success or failure.

Security Considerations:

- Encryption: Use HTTPS and secure password storage.
- Brute Force Protection: Implement account lockout or CAPTCHA.
- CAPTCHA: Optional for preventing automated logins.

User Experience:

- Usability: Clear, intuitive interface.
- Accessibility: Support for screen readers and keyboard navigation.

- **Sign-up:**

1. **Inputs**
    - Username: Text input, must be unique.
    - Email: Text input, must be a valid and unique email.
    - Password: Password input, must meet security criteria.
    - Confirm Password: Password input, must match the password.
    - Sign Up Button: Submits the form.
    - Terms and Conditions Checkbox: Must be checked to proceed.
    - Sign In Link**:** Redirects to sign-in page.

2.Output:

Error Messages:

- "Email or username is already in use."
- "Passwords do not match."
- "Please fill in all required fields."
- "You must agree to the Terms and Conditions."
- Success Message: "Account created successfully. Please check your email to verify your account."
- Redirection: To verification page or sign-in page.
- Loading Indicator: Shows processing status.

3. Process

- Input Collection: Gather username, email, password, and confirmation.
- Validation: Check format, uniqueness, and password strength.
- Account Creation: Create user account and send verification email if needed.
- Error Handling: Provide feedback on errors.
- Redirection: Direct to appropriate page based on success or failure.

Security Considerations:

- Encryption: Secure data transmission and password storage.
- Email Verification: Optional step for account activation.
- Brute Force Protection: Prevent abuse.

User Experience:

- Usability: Intuitive and easy-to-use interface.
- Accessibility: Ensure support for disabilities.

- **Subscription Management:**

1. Inputs:

User Inputs:

- Select subscription plan and duration
- Enter personal and payment information
- Provide login credentials

System Inputs:

- Subscription plans and prices
- Payment gateway details
- User account data

2. Outputs:

User Outputs:

- Subscription confirmation (details, next payment date)
- Subscription status (current plan, expiration date)
- Error messages (e.g., payment failure)

System Outputs:

- Billing records (transaction details, invoices)
- Subscription analytics (user metrics, revenue reports)

3. Process:

Subscription Selection:

- User selects a plan and provides payment info.
- System validates and processes payment.
- Confirmation sent to the user.

Subscription Management:

- User views and updates subscription details.
- System applies changes and updates billing info.

Renewal and Expiration:

- System tracks and notifies users about upcoming renewals.
- Handles automatic renewals or informs users of issues.

Error Handling:

- Provides error messages and logs issues for resolution.

- **Google Maps Directions:**

1. Inputs:

- User input specifying the start and destination locations (e.g., "Get directions from my current location to the Empire State Building")
- Optional parameters like waypoints, travel mode (driving, walking, biking, transit), or avoidances (e.g., tolls, highways)

2. Outputs:

- A list of step-by-step directions
- Estimated travel time and distance
- Map visualization with the route highlighted

3. Processes:

- Parse Input: Extract the starting point and destination from the user's request. Identify optional parameters if provided.
- Validate Locations: Ensure that the start and destination locations are valid and can be found on Google Maps.
- Request Directions: Use the Google Maps Directions API to fetch directions between the specified locations, including any optional parameters.
- Calculate ETA: Compute the estimated travel time and distance based on the directions received. o Format Directions: Generate a user-friendly list of turn-by-turn directions.
- Display Results: Present the directions, estimated travel time, and distance to the user. Optionally, show a map with the route highlighted.

- **Search for Wikipedia Articles:**

1. Inputs:

- User voice commands or text queries (e.g., "Search for information on the Great Wall of China")

2.Outputs:

- Summarized information from the Wikipedia article
- Links to the full Wikipedia article

3.Processes:

- Convert the user's voice command or text query into a search term.
- Send a search request to Wikipedia's API or web interface.
- Retrieve the summary and relevant details of the article.
- Present a summary to the user and provide a link to the full article.

4.2 Non-Functional Requirements:

Performance:

- Respond to user queries quickly and efficiently.
- Handle multiple concurrent tasks without significant performance degradation.

Reliability:

- Be stable and reliable, with minimal crashes or errors.
- Provide regular updates and bug fixes.

Security:

- Protect user data and privacy.
- Implement robust security measures to prevent unauthorized access.

Usability:

- Have a user-friendly and intuitive interface.
- Provide clear instructions and guidance.
- Be accessible to users with disabilities.

Scalability:

- Be able to handle increasing user loads and data volumes.

Compatibility:

- Be compatible with a wide range of hardware and software configurations.

CHAPTER -5 PROCESS MODEL

5.1. User Input (Interaction Layer)

- **Voice Input**: The user interacts with the assistant using voice commands. This can be via microphone input.
- **Text Input**: Alternatively, the user might type in text commands.
- **Sensors & Contextual Awareness**: Some advanced models include additional sensors (e.g., cameras, location, environmental context) to adjust the assistant's behavior.

5.2. Pre-Processing and Data Cleaning

- **Noise Reduction**: If the input is audio, the system first removes background noise or irrelevant sounds using noise-cancellation techniques.
- **Speech-to-Text (STT)**: For voice input, the assistant converts the speech to text for further analysis.
- **Tokenization**: For text input (whether typed or converted from speech), the text is tokenized into smaller units like words or phrases for easier processing.
- **Contextual Filtering**: The assistant tries to understand context, ensuring that the user's request is in line with the environment, time, or user's past interactions.

5.3. Intent Recognition (Natural Language Understanding)

- **Intent Detection**: The core of a Jarvis-like assistant is understanding the intent behind a user's request. For example, "What's the weather today?" would be recognized as an intent to retrieve weather information.
- **Entity Recognition**: Identifying important data in the input such as date, location, or objects mentioned. For instance, "Play music by The Beatles" would have "The Beatles" as the entity.
- **Context Awareness**: In advanced models, the assistant can infer additional context from previous interactions or environmental sensors.

5.4. Task Management and Processing

- **Action Dispatcher**: Once the intent is recognized, the system determines which action needs to be taken.
- **For example:**
    - If the intent is to "Set a reminder," the assistant interacts with a task management API.
    - If it's a weather inquiry, the assistant queries a weather API (e.g., OpenWeather, Google Weather).
    - If it's a smart home command, the assistant may communicate with IoT devices using protocols like MQTT or APIs.
- **Error Handling**: If the assistant doesn't understand a command or if something fails, an error-handling mechanism kicks in, providing feedback to the user.

5.5. Response Generation (Natural Language Generation)

- **Text Response Creation**: Once the task is completed, the assistant constructs a response. This can be a simple text reply (e.g., "The weather is sunny and 75°F").
- **Speech Synthesis**: If the assistant responds via voice, it uses a Text-to-Speech (TTS) engine to convert the generated text into a spoken format.
- **Personalization**: In advanced systems, the responses may be personalized based on user preferences, history, or settings.

5.6. Output Delivery

- **Audio Output**: If the user requested a voice response, the assistant reads out the response via a speaker.
- **Text Output**: For text responses, the information is displayed on a graphical user interface (GUI).
- **Action Execution**: For certain tasks, the assistant performs an action, such as controlling a device, launching an app, or sending an email.
- **Feedback Loop**: The assistant monitors user interactions and continually refines its responses and behavior based on feedback and usage patterns.

5.7. Learning and Adaptation (Machine Learning)

- **Continuous Learning**: The assistant learns from interactions and improves over time. This can include adapting to the user's voice, preferences, and frequently used commands.
- **Data Collection**: The system gathers user data (with consent) to improve its responses. This could include logs of past interactions, feedback, or preferences.
- **Reinforcement Learning**: Advanced assistants might incorporate reinforcement learning to optimize their responses and actions based on user satisfaction.

CHAPTER – 6 PROJECT PLAN

6.1. Project Initiation & Requirements Gathering

- **Objective:** Define the scope, features, and technologies for the assistant.
- **Tasks:**
  - Define project scope and goals: Establish the features and functionalities the assistant should have (e.g., voice commands, integration with smart devices, task automation).
  - Identify stakeholders and users: Understand who will use the assistant and what their needs are (e.g., end-users, developers, designers).
  - Determine technical requirements: Define system requirements, supported operating systems (Windows, MacOS, Linux), and hardware (microphone, speakers, etc.).
  - Research existing solutions: Review existing voice assistants (e.g., Google Assistant, Alexa, Siri) and open-source alternatives (e.g., Mycroft).
- **Deliverables:**
  - Project charter
  - Feature list and prioritization
  - Technical requirements document
  - Stakeholder analysis

6.2. System Design & Architecture

Objective: Design the overall system architecture and choose technologies.

Tasks:

- Define software architecture: Decide on the overall system structure (e.g., client-server, local execution, cloud integration).
- Select technologies and libraries: Choose appropriate technologies for speech recognition, NLP, TTS, APIs, and task automation. For example:
- Speech Recognition: Google Speech-to-Text, Mozilla DeepSpeech, or Microsoft's Speech SDK.
- NLP: Rasa, Dialogflow, or OpenAI GPT models for intent recognition.
- TTS (Text-to-Speech): Google Cloud TTS, Amazon Polly.
- Task Automation: IFTTT, Zapier, or direct API integrations.

Design UI/UX:

- Create wireframes for the desktop interface, ensuring ease of use.
- Define database schema (if needed): If the assistant needs to store user data or preferences, design a lightweight database or configuration file format (e.g., SQLite, JSON).

Deliverables:

- System architecture diagram
- Technology stack selection
- UI/UX wireframes

6.3. Development - Core Modules

- **Objective**: Build the foundational components of the assistant.

- **Tasks:**
    - Speech-to-Text (STT) module: Implement and test speech recognition to convert voice commands into text.
    - Natural Language Processing (NLP) module: Integrate an NLP engine for intent detection, entity recognition, and context understanding.
    - Task management module: Develop modules to handle specific tasks, such as:
        - Weather inquiries
        - Setting reminders or alarms
        - Music control
        - Email integration
        - Smart home control (optional, e.g., IoT device integration)
    - Text-to-Speech (TTS) module: Implement the TTS engine to convert text-based responses into voice.
    - Error handling and fallback: Design error handling mechanisms for when the assistant doesn't understand commands or fails to complete tasks.
- **Deliverables:**
    - Functional STT and TTS systems
    - Core NLP module integrated
    - Task management system with basic functionality (e.g., weather, reminders)
    - Error handling and fallback workflows

6.4. Development - Advanced Features & Integrations Objective:

- Add advanced functionalities and integrations.
- **Tasks:**
    - Advanced NLP and Machine Learning: Integrate more sophisticated NLP models (e.g., GPT or custom models) for context-aware, conversational abilities.
    - Personalization features: Enable the assistant to remember user preferences, history, and settings.
    - Third-party API integrations:
        - Integrate with calendars (Google Calendar, Outlook)
        - Email management (Gmail API, Outlook API)
        - Music services (Spotify, Apple Music, etc.)
        - IoT devices (e.g., smart lights, thermostats)
    - Multi-platform support: Ensure compatibility with Windows, MacOS, and Linux (or focus on one platform for initial release).
    - Voice activation and wake word detection: Implement a wake word (e.g., "Hey Jarvis") using libraries like Snowboy, Porcupine, or a custom wake word detection model.
- **Deliverables:**
    - Advanced NLP with context awareness
    - User-specific data (preferences, history) management system
    - Integration with third-party APIs (e.g., Spotify, email, IoT)
    - Wake word detection and multi-platform compatibility

6.5. Testing & Debugging

- Objective: Ensure all features are working as expected and fix any issues.
  - **Tasks:**
    - Unit testing: Write and run unit tests for individual components, such as STT, NLP, TTS, and task management.
    - Integration testing: Test the integration of components (e.g., STT with NLP, NLP with task management).
    - User acceptance testing (UAT): Conduct user testing with a small group of users to ensure the assistant meets their needs and expectations.
    - Performance testing: Ensure the assistant performs well under various conditions (e.g., multiple commands, noisy environments).
    - Security testing: Test the system for vulnerabilities, especially in data handling and communication with third-party APIs. o Bug fixing and refinements: Address bugs, optimize performance, and refine the user experience.
- **Deliverables:**
    - Test cases and results
    - Bug reports and resolution
    - User feedback and improvements
    - Security assessment report

6.6. Deployment & Launch

- Objective: Deploy the assistant for end-users and ensure smooth operation.
- **Tasks:**
    - Prepare deployment environment: Set up servers, cloud infrastructure, or packaging for desktop application (e.g., for Windows or MacOS).
    - Deploy application: Publish the desktop application or make it available for download (e.g., via a GitHub release, or as an installer package).
    - Post-launch monitoring: Monitor the system for any errors or performance issues after deployment.
    - User documentation and tutorials: Create user manuals and documentation on how to install and use the assistant.
- **Deliverables:**
    - Deployed application or installer
    - User documentation (manual, FAQ, etc.)
    - Monitoring plan

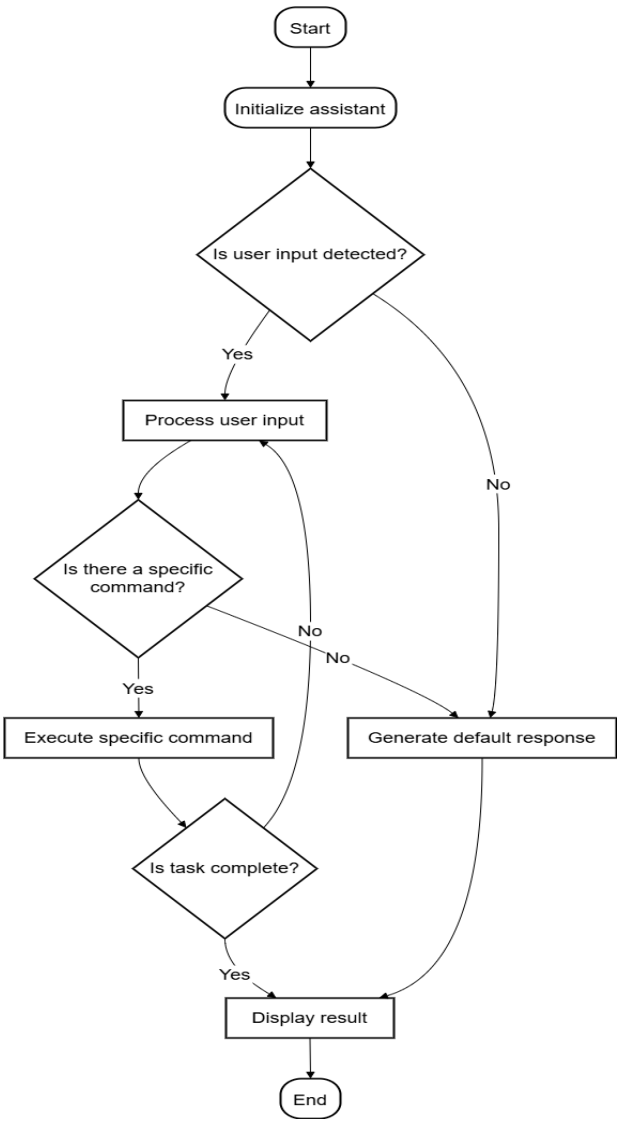
6.7. Maintenance & Updates

- Objective: Maintain the system, fix bugs, and add new features.
- **Tasks:**
    - Monitor performance: Regularly check for errors or performance degradation.
    - Update and upgrade: Implement improvements or add new features based on user feedback.
    - Patch security vulnerabilities: Regularly update the system to address any newly discovered security threats.

  - User support: Provide ongoing support for users, troubleshooting issues or handling feature requests.
- **Deliverables:**
  - Ongoing system updates and patches
  - User support system
  - Feedback loop for continuous improvement

CHAPTER 7- SYSTEM DESIGN

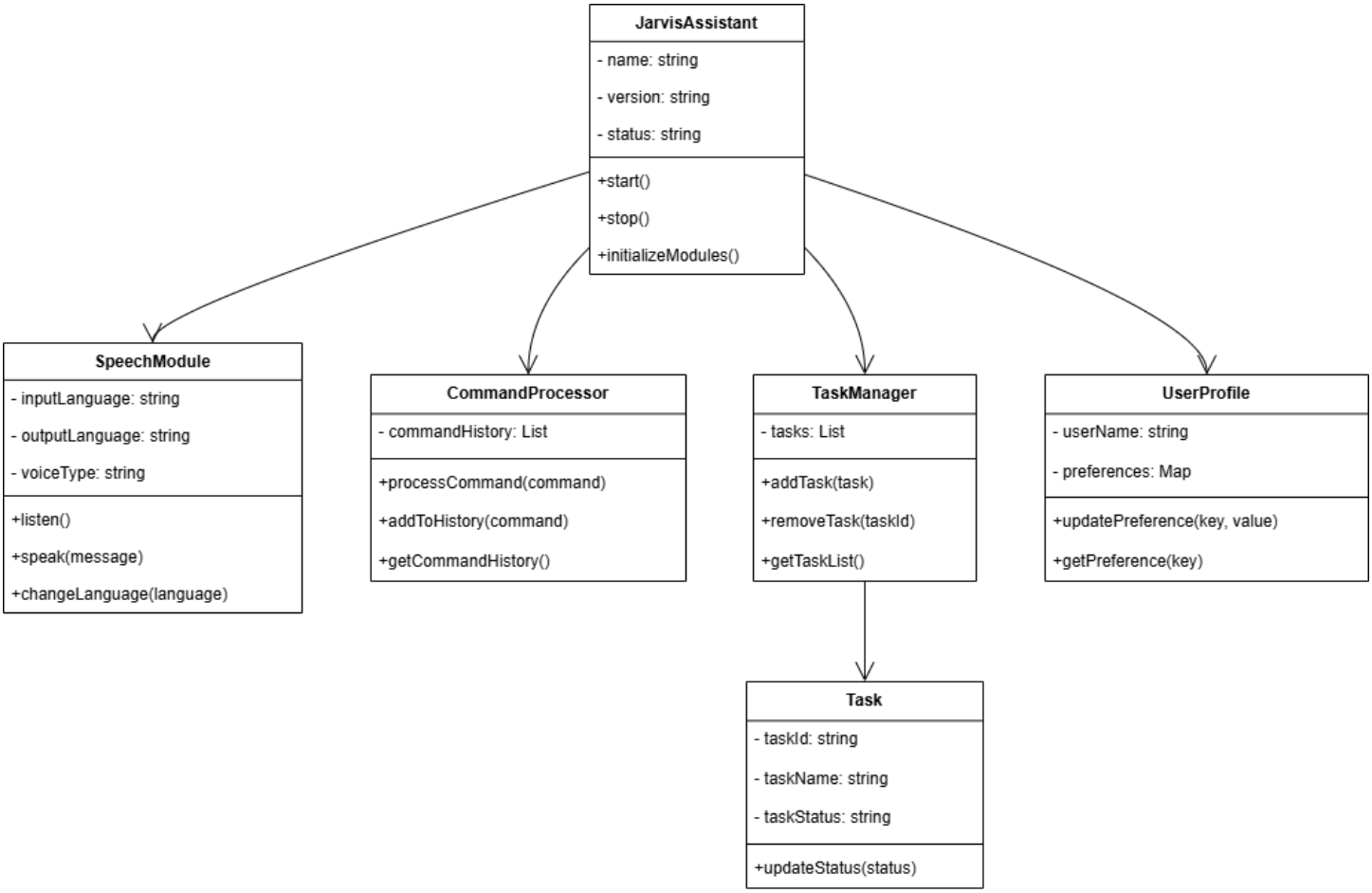
7.1 flowchart



[figure 7.1 : flow chart]

The diagram outlines the system's process, starting with user input via voice or GUI. It processes the input, executes desktop tasks or smart home commands via Arduino, and provides feedback to the user for seamless interaction.

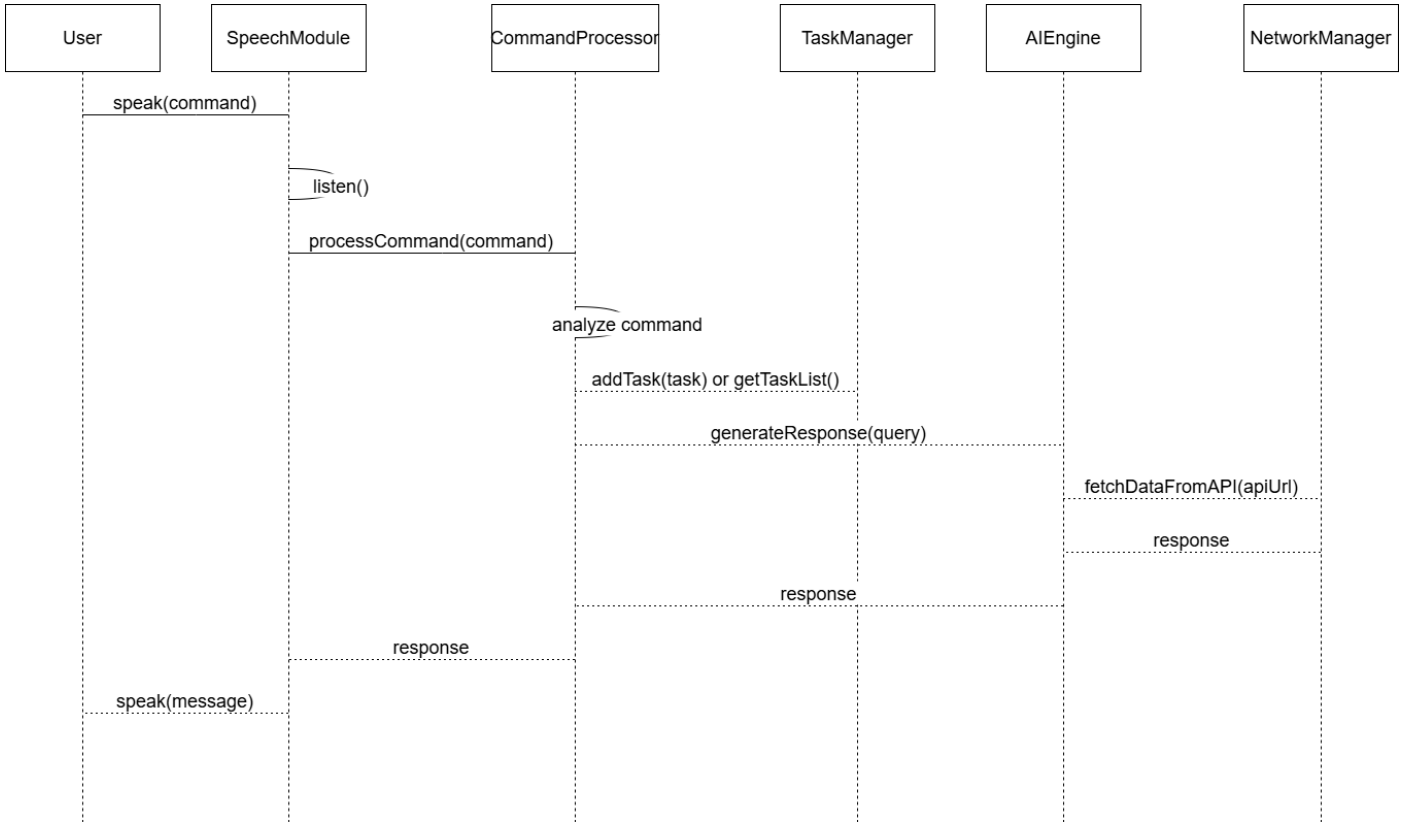
7.2 class diagram



[figure 7.2 : class diagram]

The class diagram illustrates the structure of the system, highlighting key classes like VoiceRecognition, TaskManager, and IOTController. It shows how these classes interact to process commands, manage tasks, and control smart home devices.

7.3 sequence diagram



[figure 7.3 : sequence diagram]

The sequence diagram depicts the interaction between the user, the system, and external components, showing the flow of commands and responses. It highlights how the system processes user inputs, executes tasks, and communicates with IoT devices for smart home control.

CHAPTER 8-Prototype

8.1 Home Page:

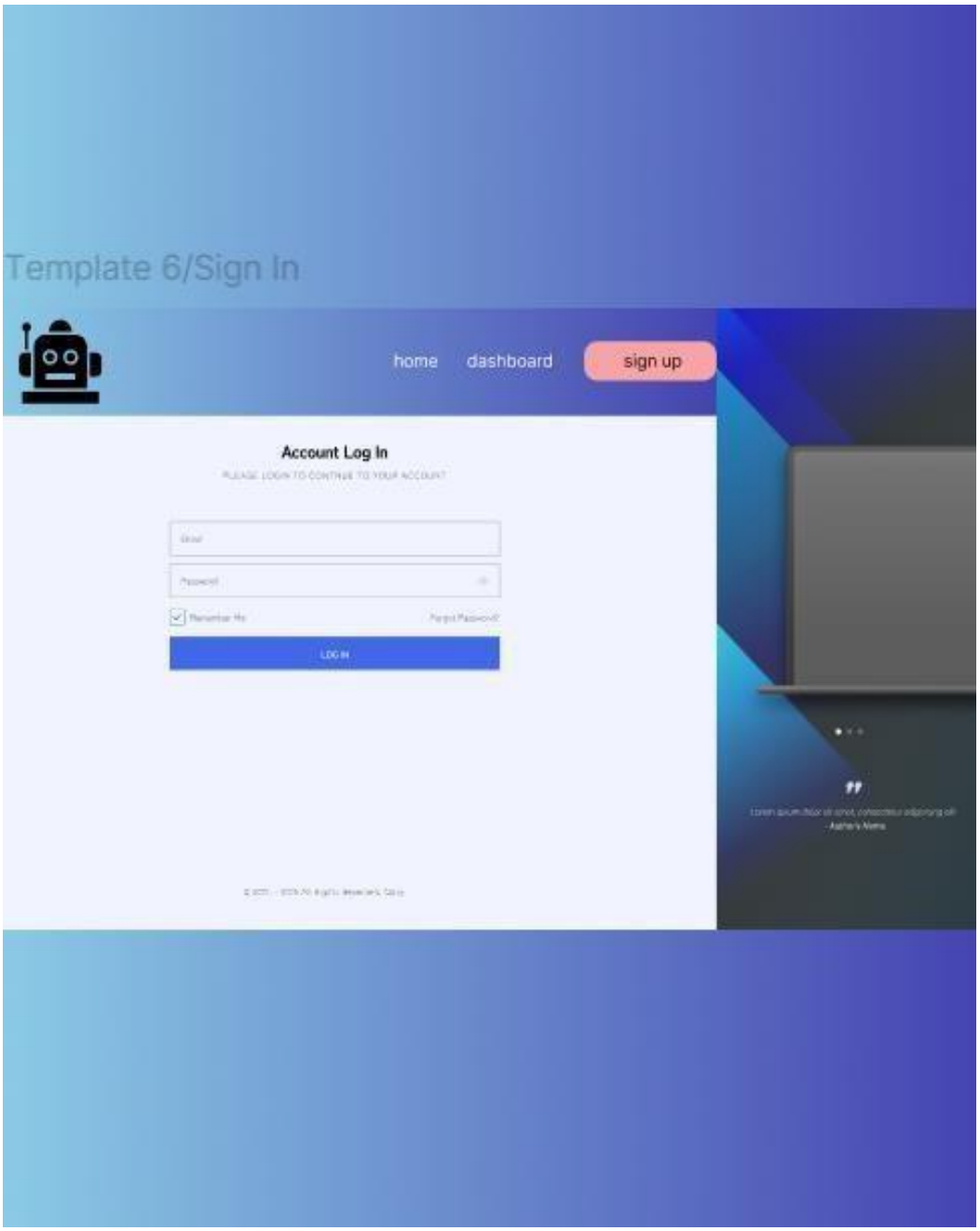
[figure 8.1 : Home Page]

The image shows a webpage for the Jarvis Desktop Assistant. It includes a welcome message with a brief introduction to Jarvis, highlighting features like Conversation History, Pinning Conversations, and Customizable Advice. There are also sections encouraging users to sign up and experience the power of AI assistance, with a futuristic interface design featuring blue and purple gradients and a tech-inspired background.

[figure 8.2 : Home Page]

The image shows the Jarvis Desktop Assistant dashboard. It greets the user, "Hey KD, Welcome to your dashboard!" and displays options like starting a new chat. The interface features a futuristic design, with a message input section at the bottom, indicating a "future message." The layout includes a navigation menu on the left for selecting chat categories, and a tech-themed background with an "ONLINE" status indicator.

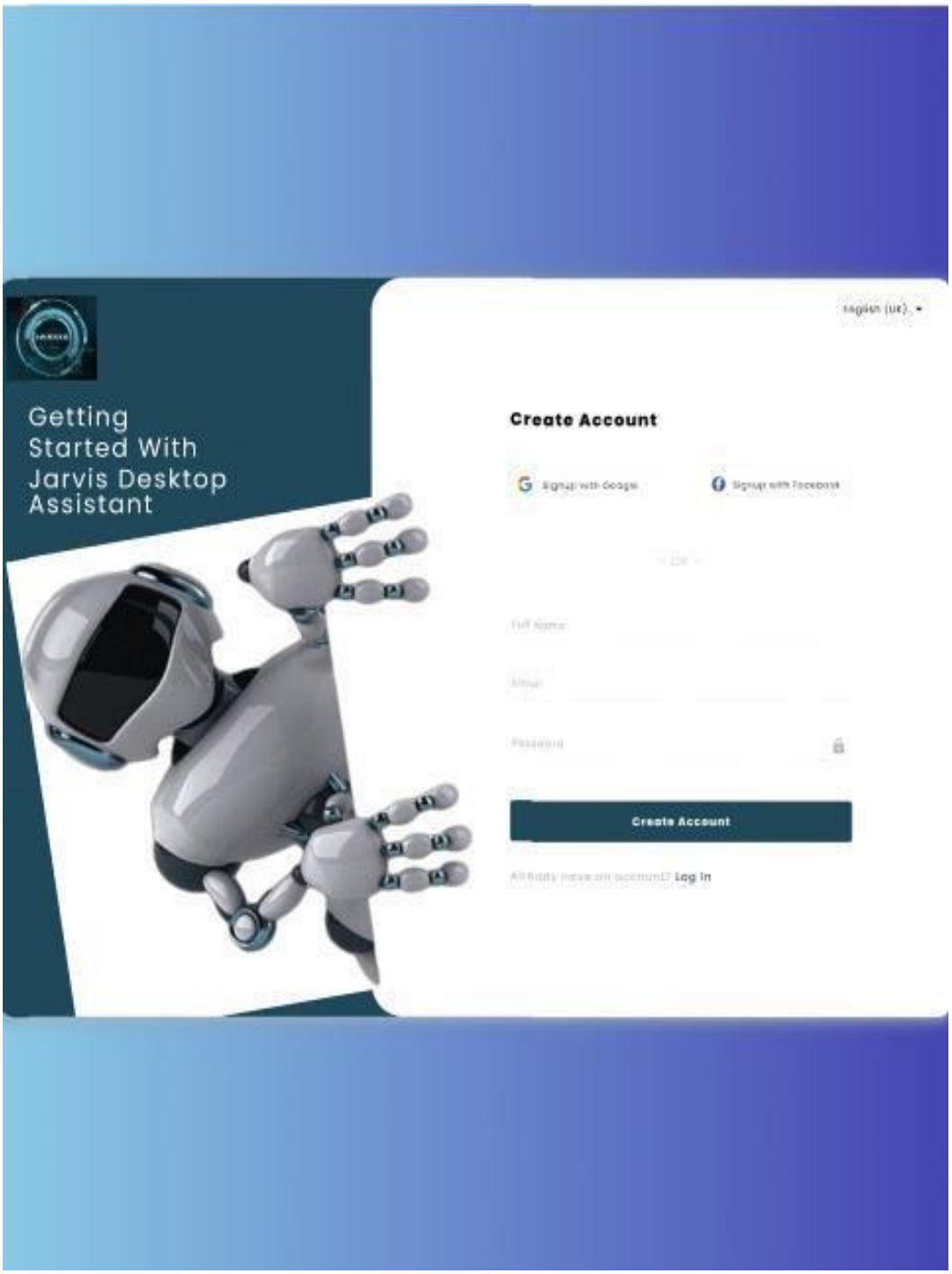
8.2 Login:



[figure 8.3 : Login]

The image shows the Jarvis Desktop Assistant login page. It features fields for the user to enter their email and password, along with options to reset the password or keep the user logged in. There is a blue "Log In" button, and a motivational quote on the right side of the page. The background features a gradient design with a modern, sleek look.

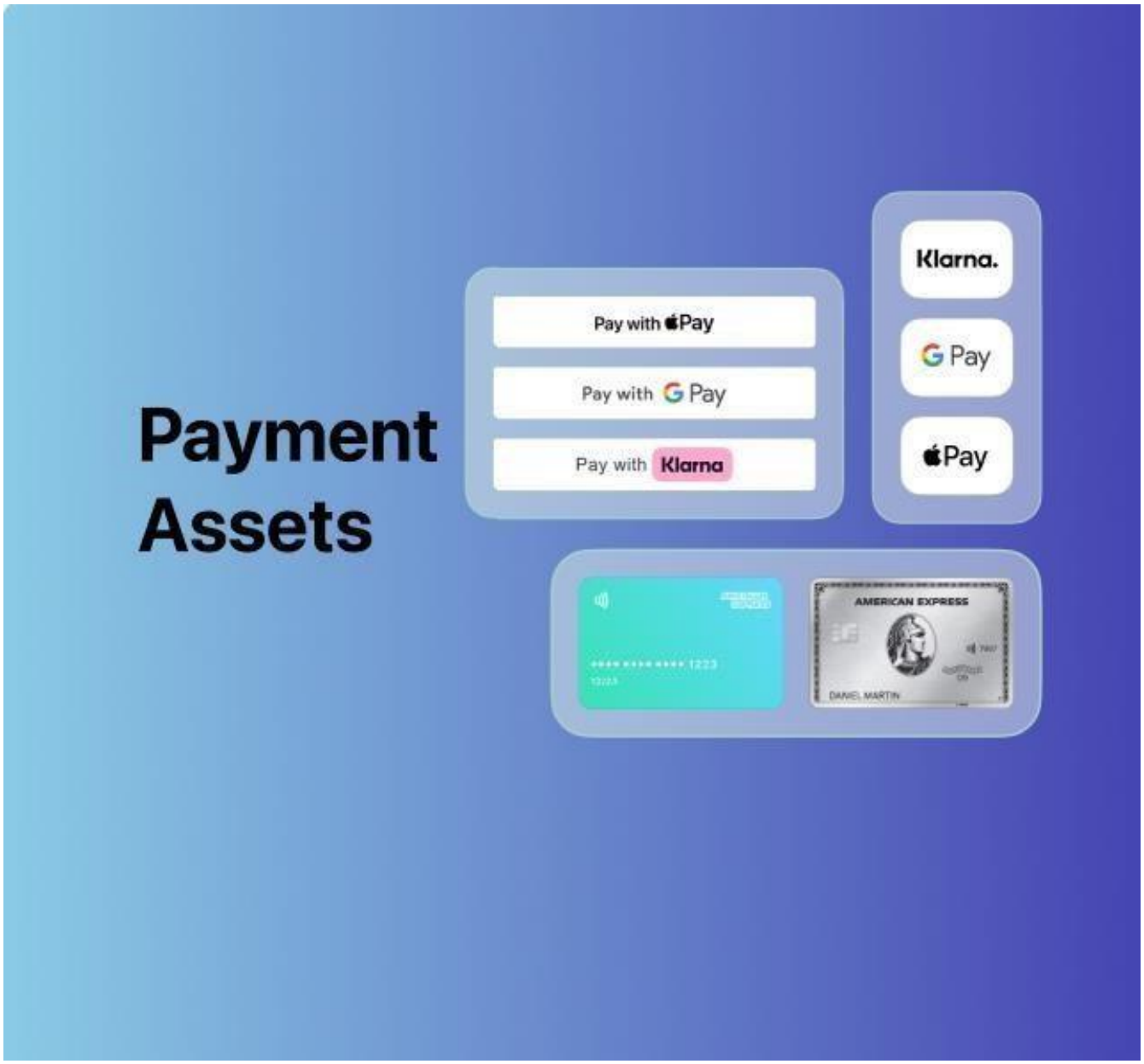
8.3 Sign-Up:



[figure 8.4 : Sign-Up]

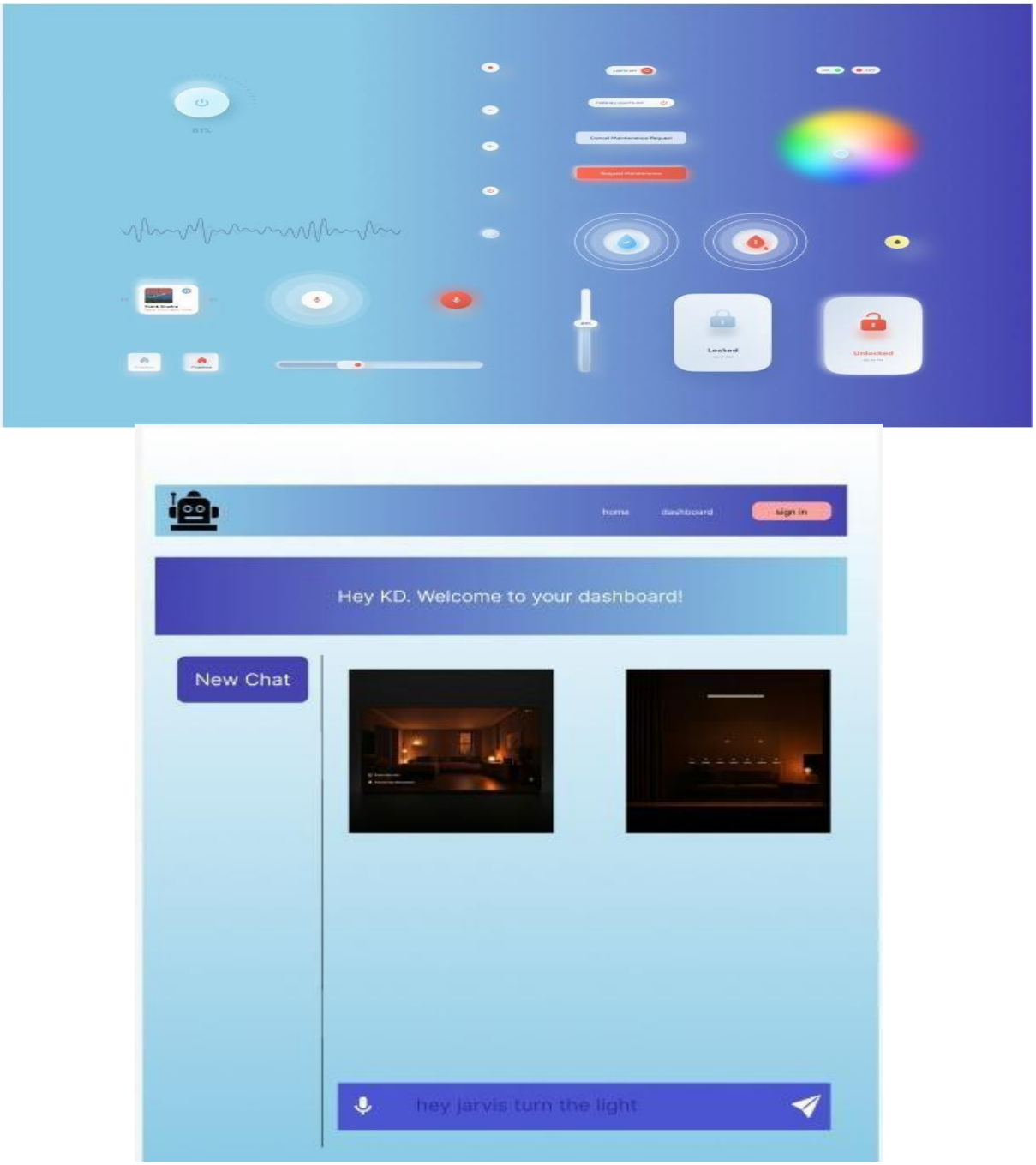
This image appears to be a graphical user interface (GUI) for setting up and using a "Jarvis Desktop Assistant". The GUI includes a robotic or AI-like character in the top left corner, along with text and input fields to "Create Account" for the Jarvis Desktop Assistant. The overall design and visual elements suggest this is a user interface for a digital assistant or virtual assistant software.

8.4 Payment Method:



[figure 8.5 : Payment Method]

This image appears to be an illustration of various payment options and assets. The main text states "Payment Assets", and it shows different payment methods such as Apple Pay, Google Pay, and Klarna. The image also includes a credit card icon, suggesting these payment methods can be linked to or used with a credit card. The overall design and layout suggest this is an informational or promotional visual related to the different payment options and assets available to users.



[figure 8.6 : Smart Home]

This image appears to be a user interface for a smart home assistant, possibly part of your project. The dashboard welcomes the user "K.D." and displays two images, likely representing different rooms or areas in the home. Below, there is a voice input field labeled "hey Jarvis, turn the light" and a send button, suggesting voice control capabilities for home automation features like lighting. The overall design has a clean, modern aesthetic with a blue and purple color scheme. This interface seems designed to provide an intuitive and integrated control system for managing various aspects of a connected home environment.

CHAPTER 9 – IMPLEMENTATION CODE

```
document.addEventListener('DOMContentLoaded', () => {
    const inputField = document.getElementById('assistant-input');
    const outputDiv = document.getElementById('assistant-output');
    const askButton = document.getElementById('button-ask');
    const recordButton = document.getElementById('button-record');

    const userName = "Boss";
    let isSpeaking = false;
    let micActive = false;
    let serialPort = null;

    const predefinedCommands = {
        'open youtube': () => openSite('https://www.youtube.com', 'Opening YouTube...'),
        'open github': () => openSite('https://www.github.com', 'Opening GitHub...'),
        'open facebook': () => openSite('https://www.facebook.com', 'Opening Facebook...'),
        'open twitter': () => openSite('https://www.twitter.com', 'Opening Twitter...'),
        'open linkedin': () => openSite('https://www.linkedin.com', 'Opening LinkedIn...'),
        'open instagram': () => openSite('https://www.instagram.com', 'Opening Instagram...'),
        'play music': () =>
openSite('https://music.youtube.com/watch?v=XO8wew38VM8&list=OLAK5uy_liFpkvwf-
4WbtSaTpkYnt98R0u_Wsu04k', 'Playing Music Online...'),
        'what is the time': () => respondWithTime(),
        'what is the date': () => respondWithDate(),
        'what is my battery level': () => checkBatteryLevel(),
        'what is the weather': () => fetchWeatherData(),
        'calculate': (expression) => calculateExpression(expression),
        'tell me about': (topic) => searchWikipedia(topic),
        'play song': () => initiateMusicSearch(),
        'shutdown device': () => shutdownDevice(),
        'restart device': () => restartDevice(),
        'sleep device': () => sleepDevice(),
```

```
    'search youtube': (query) => searchYouTube(query),
    'find on google': (query) => searchGoogle(query),
    'turn on bulb': () => controlBulb('ON'),
    'turn off bulb': () => controlBulb('OFF')
  };


// New Arduino Bulb Control Function
async function controlBulb(state) {
  if (!('serial' in navigator)) {
    displayAndSpeakResponse("Web Serial API not supported.");
    return;
  }

  try {
    // Request port with more specific filters
    const port = await navigator.serial.requestPort({
      filters: [
        { usbVendorId: 0x2341 },  // Arduino generic vendor ID
        { usbProductId: 0x0043 }  // Uno specific product ID
      ]
    });

    // Open port with extended configuration
    await port.open({
      baudRate: 9600,
      dataBits: 8,
      stopBits: 1,
      parity: 'none'
    });

    const writer = port.writable.getWriter();
```

```
        const encoder = new TextEncoder();
        const data = encoder.encode(state + '\n');

        await writer.write(data);
        writer.releaseLock();

        displayAndSpeakResponse(Bulb turned ${state.toLowerCase()});

        // Optional: Close port after command
        await port.close();
    } catch (error) {
        console.error('Detailed Bulb Control Error:', error);
        displayAndSpeakResponse("Failed to control bulb. Check Arduino connection.");
    }
}


 // Wikipedia Search Function
 function searchWikipedia(topic) {
    const searchUrl = https://en.wikipedia.org/w/index.php?search=${encodeURIComponent(topic)};
    displayAndSpeakResponse(Searching Wikipedia for information about ${topic}...);
    setTimeout(() => {
        window.open(searchUrl, '_blank');
    }, 2000);
}

// Music Search Function
function initiateMusicSearch() {
    displayAndSpeakResponse("Would you like to search for a specific song or play a random playlist?");
    // This could be expanded with more interactive logic
    setTimeout(() => {
        const searchUrl = 'https://music.youtube.com/';
```

```
        window.open(searchUrl, '_blank');
    }, 2000);
}

// Device Control Functions
function shutdownDevice() {
    displayAndSpeakResponse("Initiating device shutdown. Please save all your work.");
    // Note: Actual shutdown requires system-level permissions
    alert("Device shutdown command triggered. This is a simulated response.");
}

function restartDevice() {
    displayAndSpeakResponse("Restarting device. Please wait a moment.");
    // Note: Actual restart requires system-level permissions
    alert("Device restart command triggered. This is a simulated response.");
}

function sleepDevice() {
    displayAndSpeakResponse("Putting the device to sleep mode.");
    // Note: Actual sleep mode requires system-level permissions
    alert("Device sleep command triggered. This is a simulated response.");
}

// YouTube Search Function
function searchYouTube(query) {
    const searchUrl = https://www.youtube.com/results?search_query=${encodeURIComponent(query)};
    displayAndSpeakResponse(Searching YouTube for ${query}...);
    setTimeout(() => {
        window.open(searchUrl, '_blank');
    }, 2000);
}
```

```
// Google Search Function
function searchGoogle(query) {
    const searchUrl = https://www.google.com/search?q=${encodeURIComponent(query)};
    displayAndSpeakResponse(Searching Google for ${query}...);
    setTimeout(() => {
        window.open(searchUrl, '_blank');
    }, 2000);
}

  function openSite(url, responseMessage) {
      displayAndSpeakResponse(responseMessage);
      setTimeout(() => {
          window.open(url, '_blank');
      }, 2000); // Wait for the speech synthesis to complete before redirection.
  }

  function respondWithTime() {
      const currentTime = new Date().toLocaleTimeString();
      displayAndSpeakResponse(The current time is ${currentTime});
  }

  function respondWithDate() {
      const currentDate = new Date().toLocaleDateString();
      displayAndSpeakResponse(Today's date is ${currentDate});
  }

  async function checkBatteryLevel() {
      try {
          const battery = await navigator.getBattery();
          const level = battery.level * 100;
          displayAndSpeakResponse(Your battery level is ${level.toFixed(2)}%.);
      } catch (error) {
```

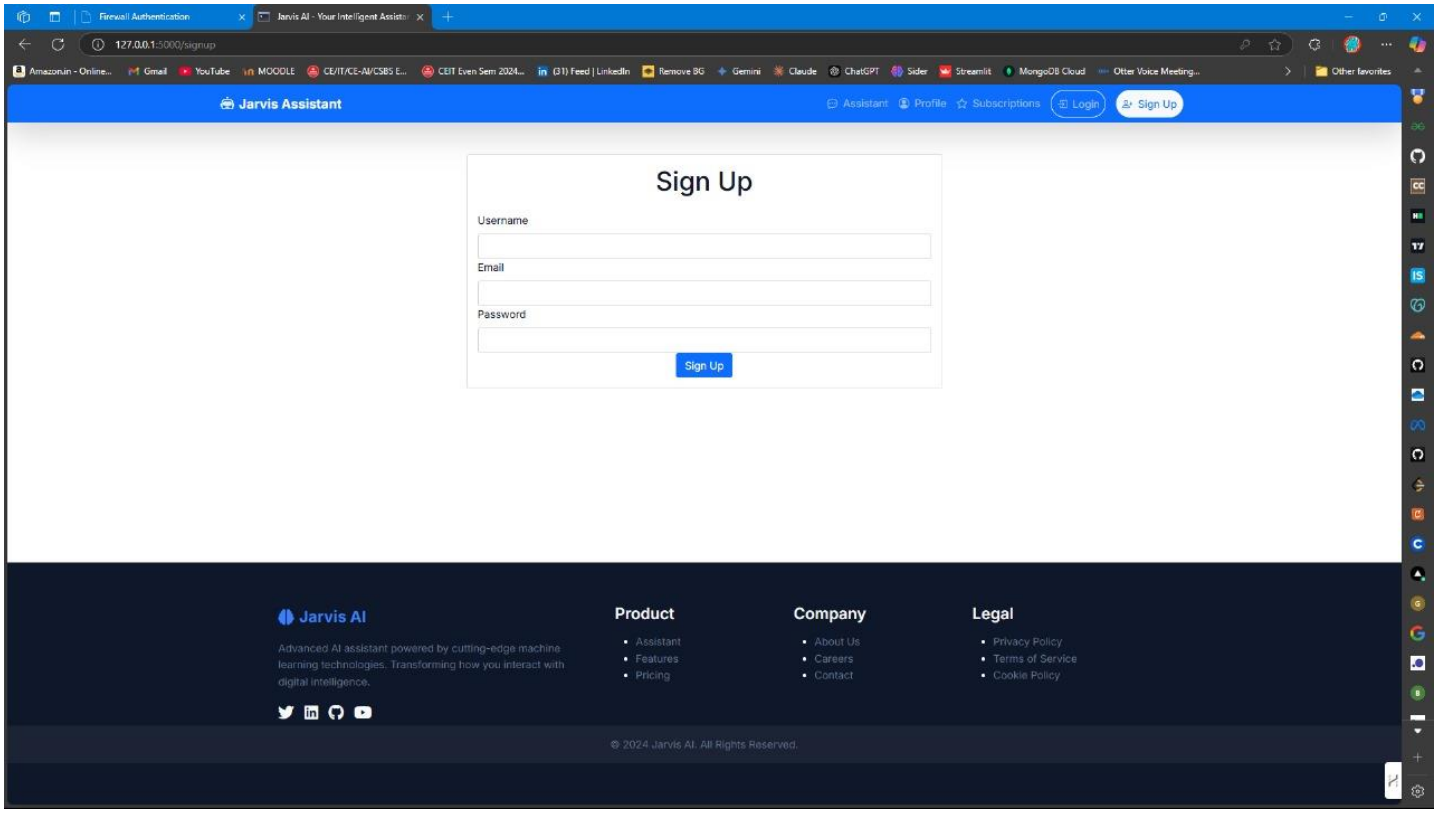
```
        displayAndSpeakResponse("Sorry, I couldn't retrieve the battery level.");
    }
}

async function fetchWeatherData() {
    try {
        const response = await fetch('https://api.openweathermap.org/data/2.5/weather?q=India&appid=3e89f882c90a1d99b70491775297aa67&units=metric');
        const weatherData = await response.json();
        const { name, weather, main } = weatherData;
        const weatherDescription = weather[0].description;
        const temperature = main.temp;
        displayAndSpeakResponse(The weather in ${name} is ${weatherDescription} with a temperature of ${temperature}°C.);
    } catch (error) {
        displayAndSpeakResponse("Sorry, I couldn't fetch the weather information.");
    }
}

function calculateExpression(expression) {
    try {
        const result = eval(expression);
        displayAndSpeakResponse(The result is ${result});
    } catch {
        displayAndSpeakResponse("Sorry, I couldn't perform the calculation.");
    }
}

function greetUser(name) {
    const greeting = Hello ${name}, welcome back! How can I assist you today?;
    displayAndSpeakResponse(greeting);
}
```

```
function displayAndSpeakResponse(response) {
    outputDiv.innerHTML = response;
    speak(response);
}

function speak(message) {
    isSpeaking = true;
    const speech = new SpeechSynthesisUtterance(message);
    speech.lang = 'en-US';
    speech.onend = () => {
        isSpeaking = false; // Allow interactions after speaking ends.
    };
    speechSynthesis.speak(speech);
}

// Enhanced Voice Recognition
function startVoiceRecognition() {
 if ('webkitSpeechRecognition' in window) {
      const recognition = new webkitSpeechRecognition();
      recognition.continuous = false;
      recognition.interimResults = false;
      recognition.lang = 'en-US';

      recognition.onstart = () => {
          micActive = true;
          console.log('Voice recognition started.');
      };

      recognition.onresult = async (event) => {
          if (isSpeaking) {
              recognition.stop();
```

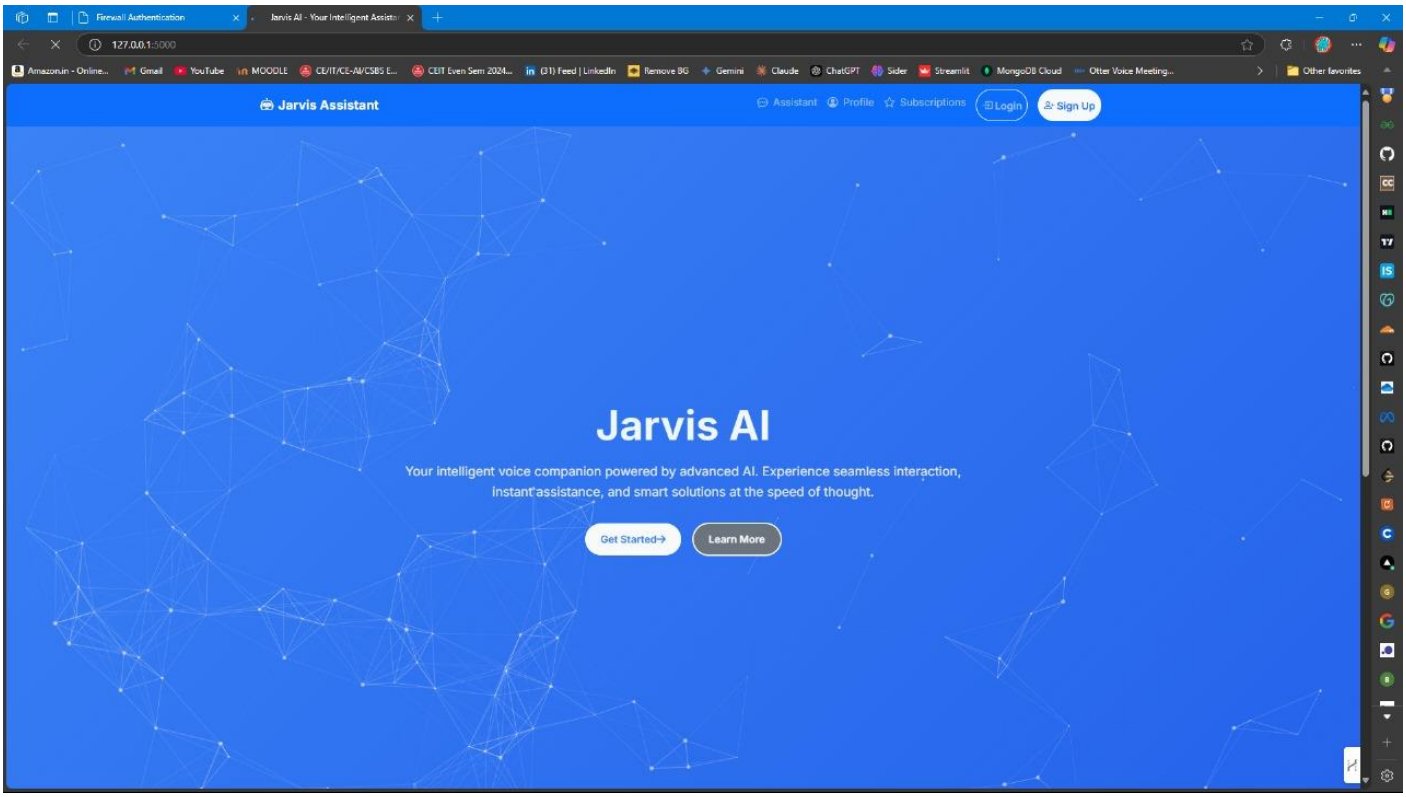
```
            return;
        }
        const userInput = event.results[0][0].transcript.trim().toLowerCase();
        await processUserInput(userInput);
    };

    recognition.onerror = (event) => {
        displayAndSpeakResponse("Sorry, there was an error with voice recognition.");
        console.error("Voice recognition error:", event.error);
    };

    recognition.onend = () => {
        micActive = false;
        console.log("Voice recognition ended.");
    };

    recognition.start();
  } else {
    displayAndSpeakResponse("Sorry, your browser does not support voice recognition.");
  }
}


// Enhanced processUserInput to handle async calls
async function processUserInput(input) {
  if (isSpeaking) return;

  for (const command in predefinedCommands) {
    if (input.includes(command)) {
      const extractedTopic = input.replace(command, '').trim();

      if (['tell me about', 'search youtube', 'find on google', 'calculate'].includes(command)) {
```

```
            predefinedCommands[command](extractedTopic);
        } else {
            predefinedCommands[command]();
        }
        return;
    }
}

displayAndSpeakResponse(I'm searching Google for: ${input});
setTimeout(() => {
    window.open(https://www.google.com/search?q=${encodeURIComponent(input)}, '_blank');
}, 2000);
}

// Existing event listeners and initial setup
greetUser(userName);

askButton.addEventListener('click', () => {
    const userInput = inputField.value.trim();
    processUserInput(userInput);
    inputField.value = '';
});
recordButton.addEventListener('click', () => {
    if (isSpeaking) {
        displayAndSpeakResponse("Please wait until I finish speaking.");
        return;
    }
    startVoiceRecognition();
});

});
```

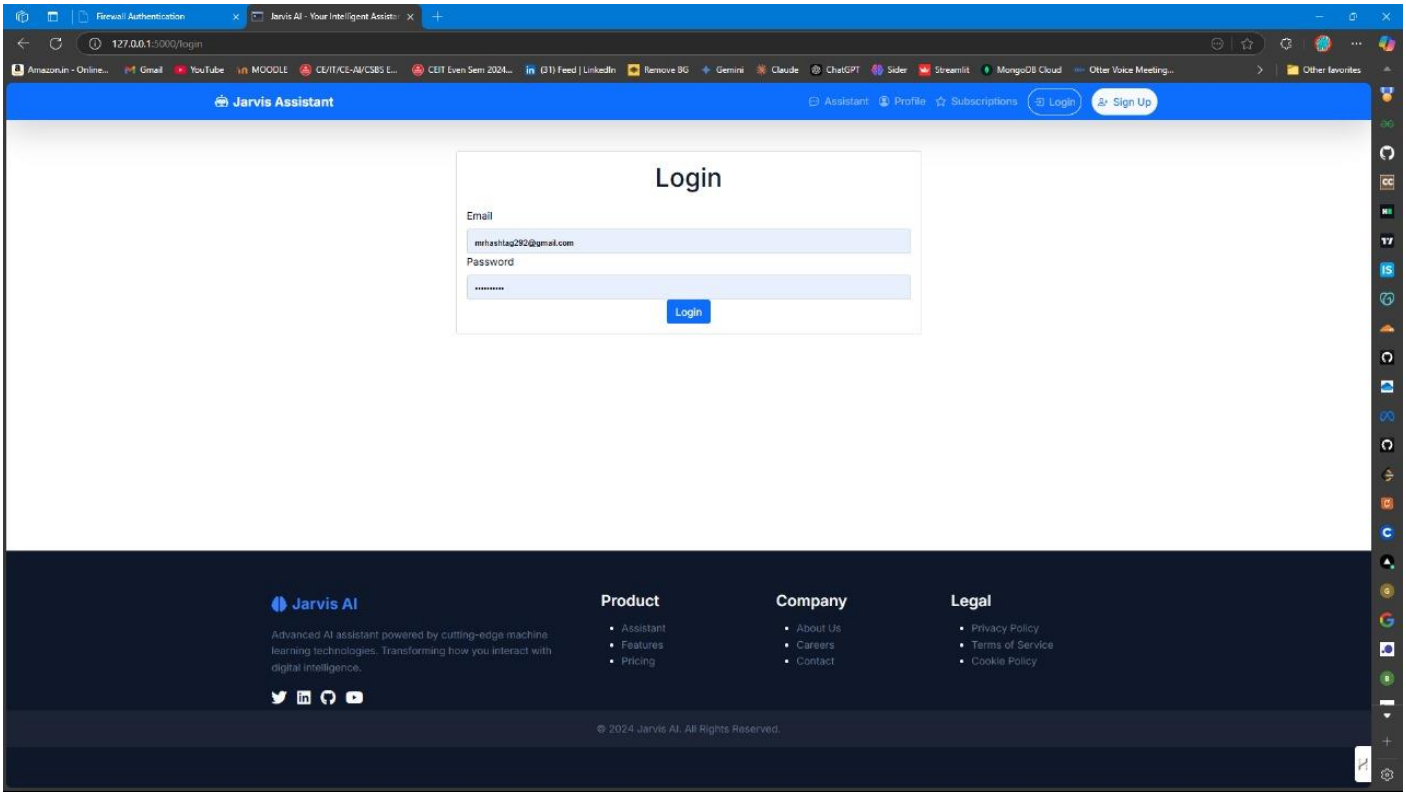
OutPut :-



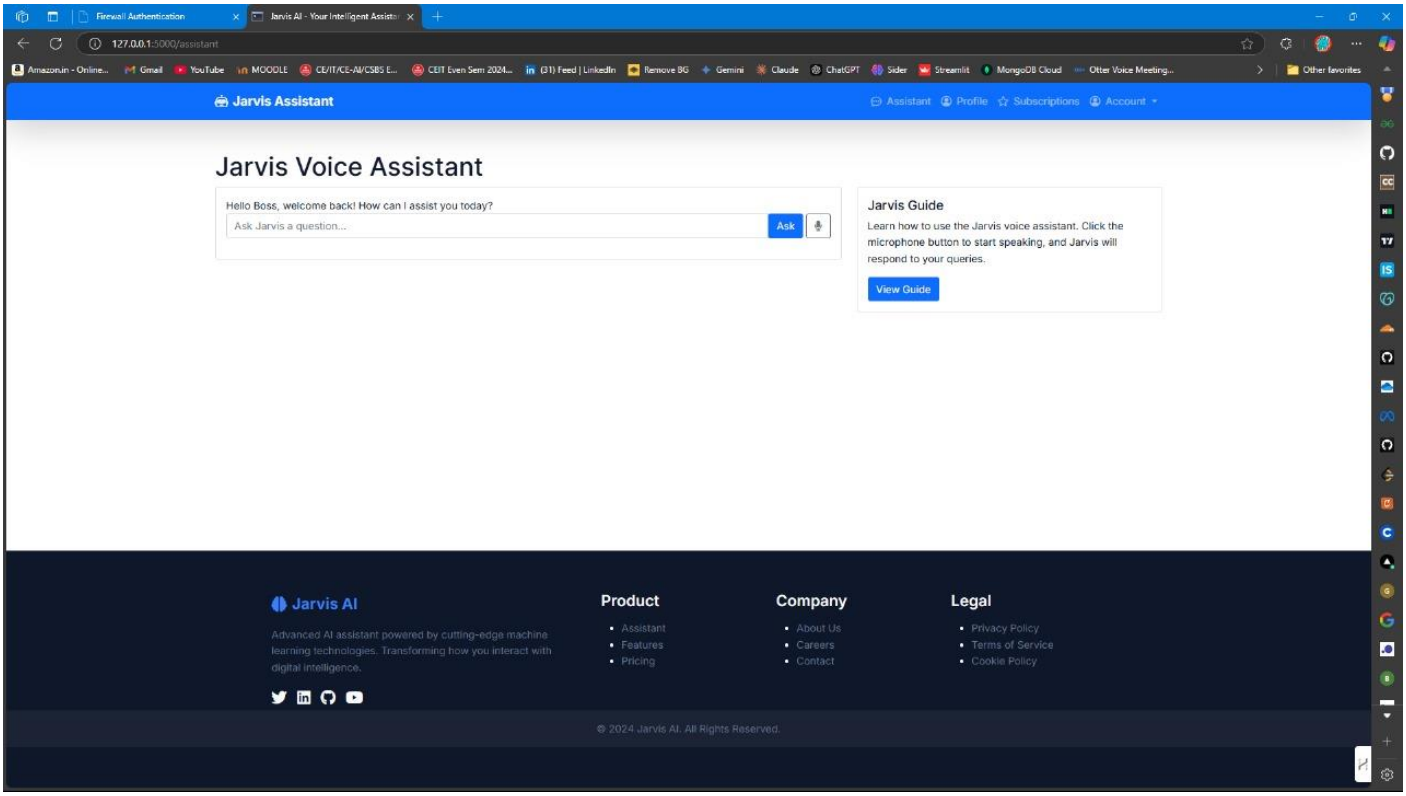
[Image 9.1:- Sign Up]



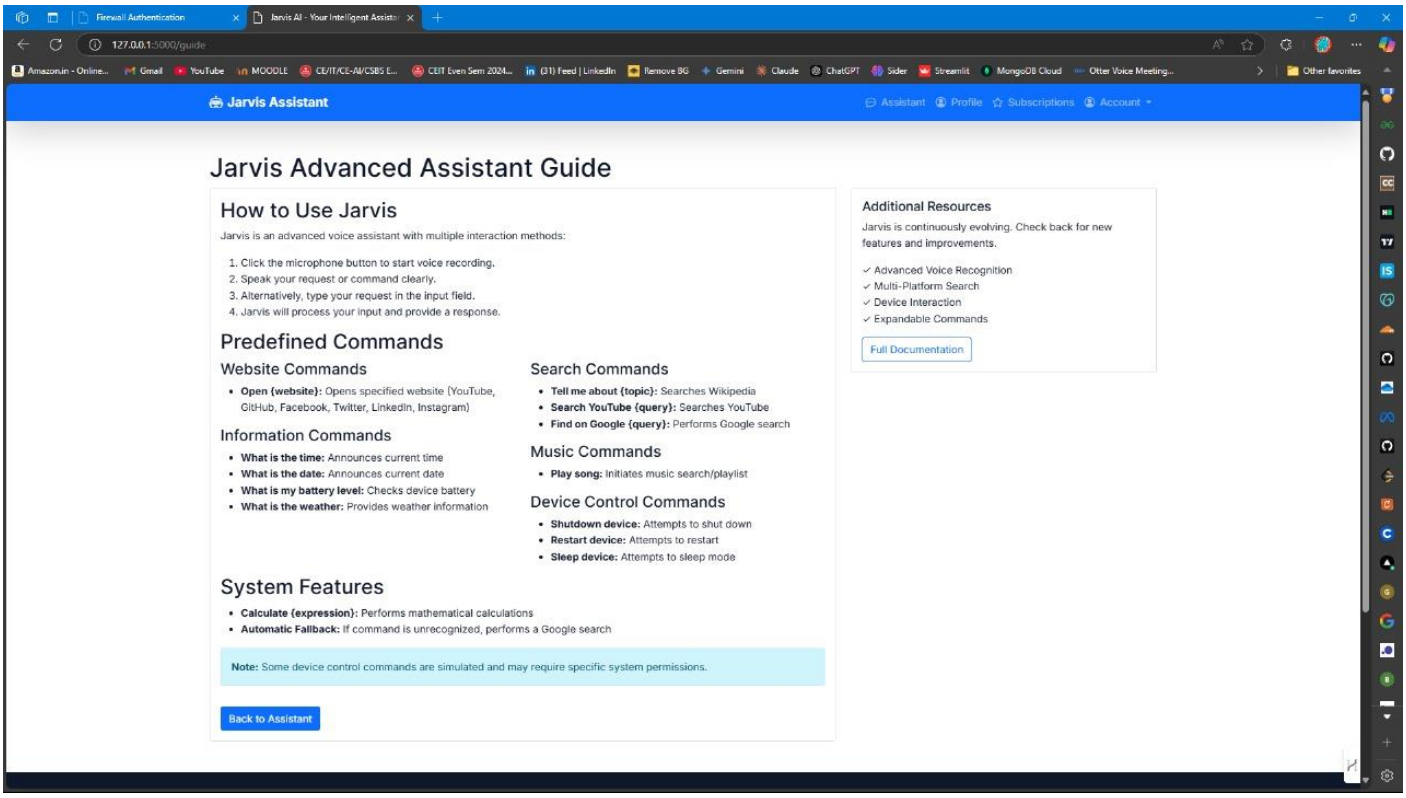
[Image 9.2:- Home Page]



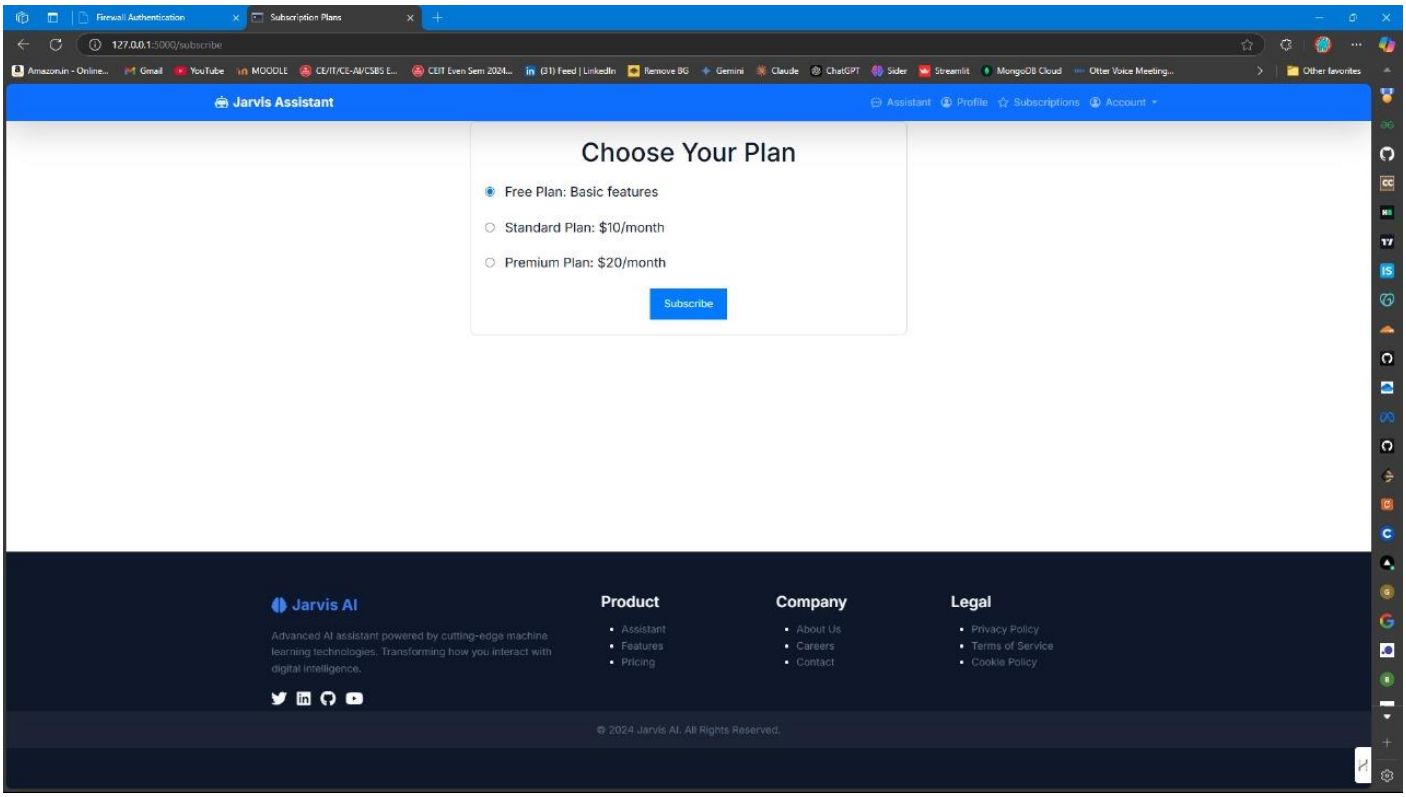
[Image 9.3:- Login]



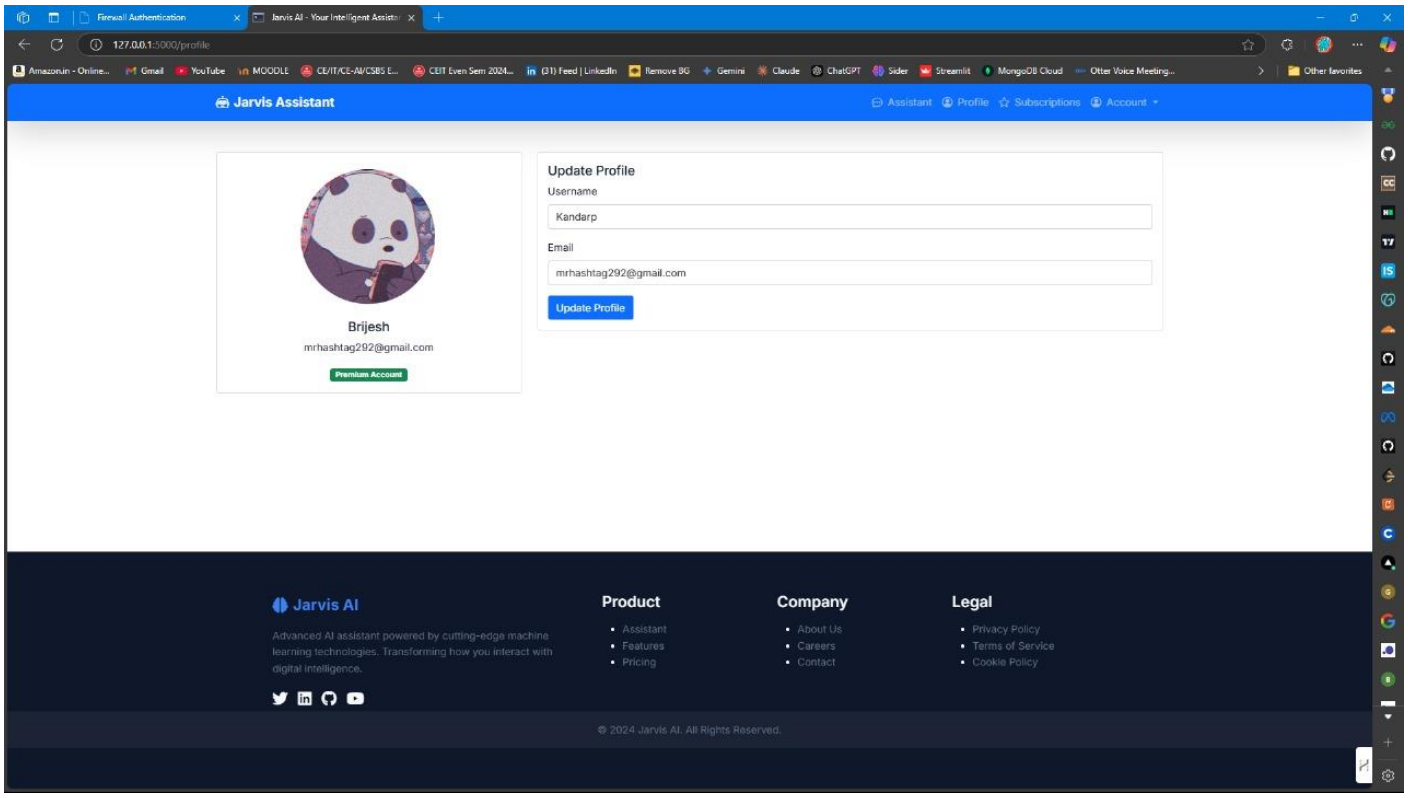
[Image 9.4:- Jarvis-Desktop Assistant]



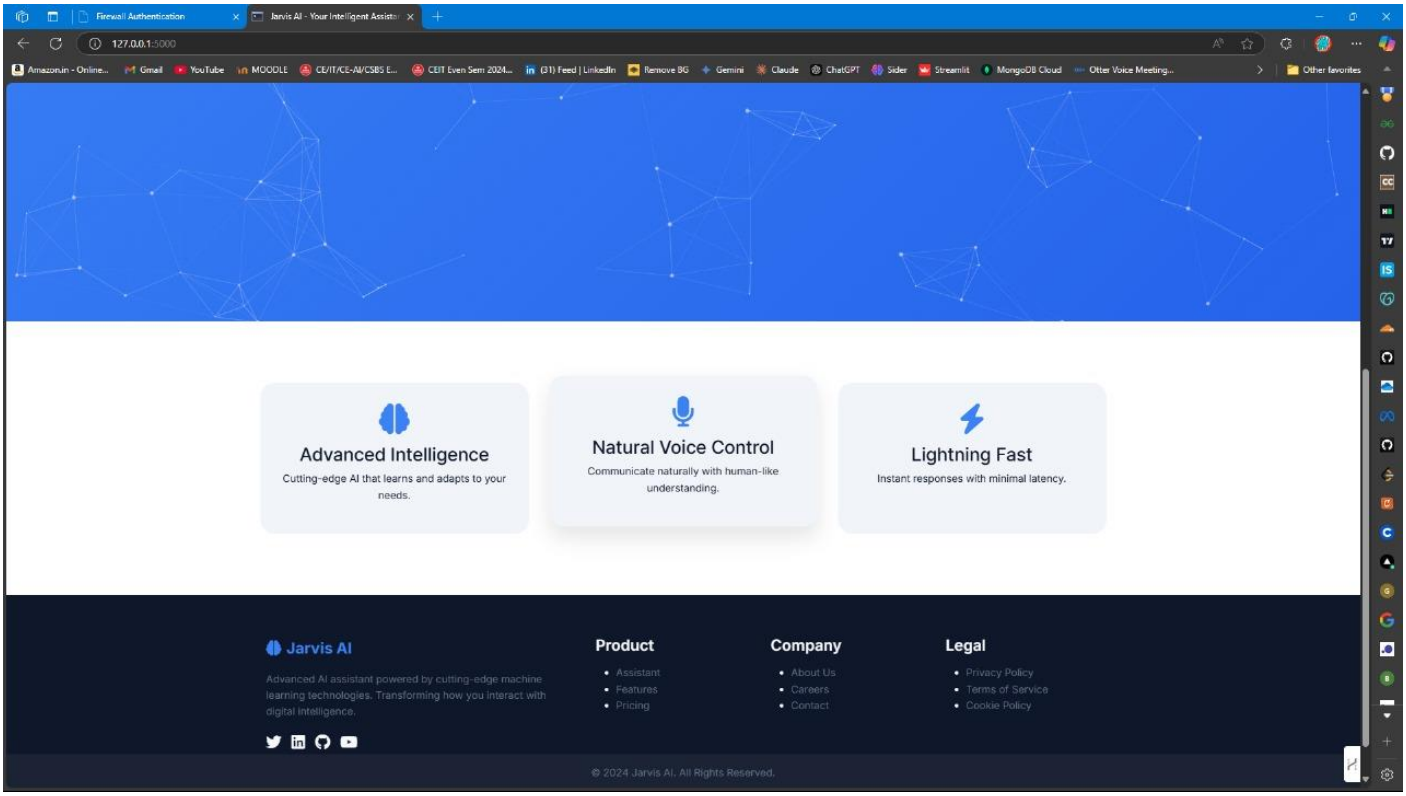
[Image 9.5:-Assistant Guide]



[Image 9.6:-Payment Method]



[Image 9.7:-Update Profile]



[Image 9.8:-About Jarvis AI]

CHAPTER 10 – TEST CASE STUDIES

Test Case ID	Feature	Scenario	Input Type	Expected Result	Comments
TC-001	Login	Successful login	Written	User is logged in successfully and redirected to the dashboard.	
TC-002	Login	Invalid credentials	Written	Display error message: 'Invalid username or password.'	
TC-003	Sign Up	Successful sign-up	Written	User account is created, and a confirmation email is sent.	
TC-004	Sign Up	Invalid input (e.g., missing email)	Written	Display error message: 'Please provide a valid email address.'	
TC-005	Open YouTube	Open YouTube with voice order	Voice	YouTube is opened in the default browser.	
TC-006	Open YouTube	Open YouTube with written order	Written	YouTube is opened in the default browser.	
TC-007	Directions	Provide directions to a specific location (voice)	Voice	Google Maps opens with the directions displayed for the specified location.	
TC-008	Directions	Provide directions to a specific location (written)	Written	Google Maps opens with the directions displayed for the specified location.	
TC-009	Play Music	Play online music (voice)	Voice	Default music streaming platform opens and plays the requested song/playlist.	
TC-010	Play Music	Play online music (written)	Written	Default music streaming platform opens and plays the requested song/playlist.	

TC-011	Check Time	Ask for the current time (voice)	Voice	The assistant announces the current time.	
TC-012	Check Time	Ask for the current time (written)	Written	The assistant displays the current time.	
TC-013	Weather Update	Ask about the weather (voice)	Voice	The assistant announces the current weather condition for the specified location.	
TC-014	Weather Update	Ask about the weather (written)	Written	The assistant displays the current weather condition for the specified location.	
TC-015	Battery Check	Check battery percentage and plug-in status (voice)	Voice	The assistant announces the current battery percentage and whether the system is plugged in.	
TC-016	Battery Check	Check battery percentage and plug-in status (written)	Written	The assistant displays the current battery percentage and whether the system is plugged in.	
TC-017	Smart Home Control	Turn lights on/off (voice)	Voice	The lights in the specified room turn on or off.	Requires smart home integration.
TC-018	Smart Home Control	Turn lights on/off (written)	Written	The lights in the specified room turn on or off.	Requires smart home integration.
TC-019	Shut Down System	Shut down system (voice)	Voice	The system shuts down with a confirmation message.	
TC-020	Shut Down System	Shut down system (written)	Written	The system shuts down with a confirmation message.	
TC-021	Payment	Successful payment process	Written	Payment is processed, and confirmation is received.	
TC-022	Payment	Failed payment (e.g.,	Written	Display error message: 'Payment failed	

		insufficient funds)		due to insufficient funds.'	

[Table 10.1 : Test Case Studies]

This table contains a detailed test case table for a desktop assistant application. It covers key features like login, sign-up, YouTube access, directions, music playback, time/weather queries, battery status, smart home control, system shutdown, and payment. Both voice and written input scenarios are included, ensuring comprehensive functionality validation.

CHAPTER 11 - CONCLUSION & FUTURE WORK

Conclusion:

Jarvis is a helpful computer assistant that can understand and respond to your commands. It can do many things like helping you with your work, finding information, and controlling your smart devices.

Future Work:

We can make Jarvis even better by:

- **Teaching it more:** We can teach Jarvis about specific topics like law, medicine, or technology so it can help you with those things.
- **Adding more ways to talk to it:** We can let Jarvis understand not just your words but also your gestures and facial expressions.
- **Making it more helpful:** We can make Jarvis predict what you need before you ask and offer suggestions.
- **Keeping it safe:** We need to make sure Jarvis is used in a way that is fair and doesn't hurt anyone.
- **Using new technology:** We can try using new technology like augmented reality and blockchain to make Jarvis even more useful.

CHAPTER-12 REFERENCES

1. Meuser, T., Lovén, L., Bhuyan, M., Patil, S.G., Dustdar, S., Aral, A., Bayhan, S., Becker, C., De Lara, E., Ding, A.Y. and Edinger, J., 2024. Revisiting edge ai: Opportunities and challenges. IEEE Internet Computing, 28(4), pp.49-59.

2. MISHRA, A.A., AI-based Conversational Agents: A Scoping Review from Technologies to Future Directions.

3. Meshram, S., Naik, N., Megha, V.R., More, T. and Kharche, S., 2021, June. Conversational AI: chatbots. In *2021 International Conference on Intelligent Technologies (CONIT)* (pp. 1-6). IEEE.

4. Bengesi, S., El-Sayed, H., Sarker, M.K., Houkpati, Y., Irungu, J. and Oladunni, T., 2024. Advancements in Generative AI: A Comprehensive Review of GANs, GPT, Autoencoders, Diffusion Model, and Transformers. *IEEE Access*.

5. Mahmood, A., Wang, J., Yao, B., Wang, D. and Huang, C.M., 2023. LLM-Powered Conversational Voice Assistants: Interaction Patterns, Opportunities, Challenges, and Design Guidelines. *arXiv preprint arXiv:2309.13879*.

6. Mahmood, A., Wang, J., Yao, B., Wang, D. and Huang, C.M., 2023. LLM-Powered Conversational Voice Assistants: Interaction Patterns, Opportunities, Challenges, and Design Guidelines. *arXiv preprint arXiv:2309.13879*.

CHAPTER-13. ABOUT COLLEGE

U.V. Patel College of Engineering Ganpat University

Ganpat University-U. V. Patel College of Engineering (GUNI-UVPCE) is situated in Ganpat Vidyanagar campus. It was established in September 1997 with the aim of providing educational opportunities to students from It is one of the constituent colleges of Ganpat University various strata of society. It was armed with the vision of educating and training young talented students of Gujarat in the field of Engineering and Technology so that they could meet the demands of Industries in Gujarat and across the globe. The College is named after Shri Ugarchandbhai Varanasibhai Patel, a leading industrialist of Gujarat, for his generous support. It is a self-financed institute approved by All India Council for Technical Education (AICTE), New Delhi and the Commissionerate of Technical Education, Government of Gujarat. The College is spread over 25 acres of land and is a part of Ganpat Vidyanagar Campus. It has six ultra- modern buildings of architectural splendor, class rooms, tutorial rooms, seminar halls, offices, drawing hall, workshop, library, well equipped departmental laboratories, and several computer laboratories with internet connectivity through 1 Gbps Fiber link, satellite link education center with two-way audio and one-way video link. The superior infrastructure of the Institute is conducive for learning, research, and training. The Institute offers various undergraduate programs, postgraduate programs, and Ph.D. programs. Our dedicated efforts are directed towards leading our student community to the acme of technical excellence so that they can meet the requirements of the industry, the nation and the world at large. We aim to create a generation of students that possess technical expertise and are adept at utilizing the technical 'knowhows' in the service of mankind. We strive towards these Aims and Objectives:

- To offer guidance, motivation, and inspiration to the students for well-rounded development of their personality.
- To impart technical and need-based education by conducting elaborated training programs.
- To shape and mold the personality of the future generation. To construct fertile ground for adapting to dire challenges.