**SRI SRI UNIVERSITY(SSU)**

**Hackathon – Low-Level Design**

**on**

**AI- POWERED VOICE ASSISTANT WITH CUSTOM SILLS**

Team members:

Shreya Adya

Kirtidev Parida

Hara Soumya Parida

Shubhankar Pradhan

Apparna Singh

**Table of Contents**

1. Introduction...............................................................................................................6

1.1. Scope of the document..........................................................................................6

1.2. Intended audience.................................................................................................6

1.3. System overview..................................................................................................6

2. Low Level System Design.........................................................................................7

1.1. Sequence Diagram.............................................................................................7

1.2. Navigation Flow/UI Implementation.................................................................7

1.3. Screen Validations, Defaults and Attributes......................................................7

1.4. Client-Side Validation Implementation.............................................................8

1.5. Server-Side Validation Implementation............................................................8

1.6. Components Design Implementation................................................................8

1.7. Configurations/Settings.....................................................................................9

1.8. Interfaces to other components..........................................................................9

3. Data Design................................................................................................................10

1.1. List of Key Schemas/Tables in database.......................................................... 10

1.2. Details of access levels on key tables in scope................................................. 10

1.3. Key design considerations in data design......................................................... 10

4. Details of other frameworks being used.....................................................................11

1.1. Session Management......................................................................................... 11

1.2. Caching.............................................................................................................. 11

5. Unit Testing...............................................................................................................12

6. Key notes.................................................................................................................. 13

7. Reference.................................................................................................................. 13

INTRODUCTION

**Scope of The Document:**

This document provides the low-level design (LLD) for building a Python-based voice assistant capable of understanding and executing complex tasks such as managing calendars, sending emails, or controlling IoT devices. The assistant will be trained using AI frameworks like TensorFlow for natural language understanding (NLU) and integrated with a MERN-based web interface.

To enhance user interaction, the voice assistant will be seamlessly integrated with a MERN (MongoDB, Express.js, React, Node.js) web interface. This integration will provide a robust and user-friendly platform, allowing users to interact with the assistant through both voice and text commands. The MERN stack will facilitate real-time data processing and storage, ensuring that user requests are handled efficiently and promptly.

The assistant’s functionality will extend to managing calendars, where it can schedule, modify, and remind users of their appointments. For email management, the assistant will be capable of composing, sending, and organizing emails based on user instructions. Additionally, the integration with IoT devices will enable users to control various smart home appliances, such as lights, thermostats, and security systems, through simple voice commands.

In summary, this document outlines the detailed design of a Python-based voice assistant that leverages TensorFlow for NLU and integrates with a MERN-based web interface. The assistant is designed to perform a variety of complex tasks, providing users with a powerful and intuitive tool for managing their daily activities and controlling IoT devices. Through advanced AI training and seamless integration, the assistant aims to deliver a highly responsive and secure user experience.

**Intended audience:**

The intended audience for this project are:

1. **Individual Consumers:**
   * **Smart Home Enthusiasts:** People who use smart home devices like lights, thermostats, and security systems can benefit from voice assistants to control these devices hands-free.
   * **Busy Professionals:** Individuals who need help managing their schedules, sending emails, and setting reminders can find voice assistants extremely useful.
   * **Elderly and Disabled Users:** Voice assistants can provide significant assistance to those with mobility issues or visual impairments, making technology more accessible.

2. **Businesses**:

* + [**Customer Service:** Companies can use voice assistants to handle customer inquiries, provide information, and even process orders, enhancing customer experience and reducing the workload on human agents](https://www.infobip.com/blog/ai-voice-assistants).
  + [**Retail and E-commerce:** Businesses in these sectors can leverage voice assistants to help customers with product searches, recommendations, and purchases, streamlining the shopping experience](https://link.springer.com/article/10.1007/s11002-020-09537-5).
  + **Healthcare**: Voice assistants can assist in scheduling appointments, providing medication reminders, and offering health-related information, improving patient care and operational efficiency.

**3**. **Educational Institutions**:

* + **Students and Educators**: Voice assistants can help students with research, reminders for assignments, and even language learning. Educators can use them to manage schedules and provide quick access to information.

**4.Hospitality and Travel**:

* [**Hotels and Airlines:** These businesses can use voice assistants to enhance guest experiences by providing information about services, booking assistance, and local recommendations](https://wearebrain.com/blog/ai-smart-assistants-customer-service/).

5. **Automotive Industry**:

* **Car Manufacturers:** Integrating voice assistants into vehicles can provide drivers with hands-free control over navigation, music, and communication, enhancing safety and convenience.

**System overview:**

The system consists of a Python-based backend voice assistant powered by TensorFlow for NLU and a MERN-based web interface. The backend will handle voice inputs, process the NLU tasks, and trigger the appropriate action. The MERN web interface will allow users to customize the assistant, add new skills, and interact with the assistant through a browser interface.

2.Low Level System Design

**2.1** **Sequence Diagram**

**2.2 Navigation Flow/UI Implementation**

The MERN front-end will consist of various sections, such as:

Home screen: displays available skills, voice input functionality, and system status.

Settings page: allows users to customize the assistant, manage API keys, and add new skills.

Logs: display command history and interaction results.

Skill Management: allows adding new actions via a form-driven UI.

The UI will be implemented using React components, providing modular and responsive design.

**2.3 Screen Validations, Defaults, and Attributes**

The voice input button should have validations to ensure the microphone is active.

Settings page: API key inputs must validate format (e.g., email APIs, calendar APIs).

Skill Management: Input fields for new skills should validate task name, description, and

Python code (if applicable).

**2.4. Client-Side Validation Implementation**

Client-side validation will include: Ensuring voice input is properly captured through the browser.

Form validation in the Settings and Skill Management pages using JavaScript/React form validators.

**2.5. Server-Side Validation Implementation**

Server-side validation will focus on:

Validating NLU model inputs (ensuring proper tokenization and input formatting).

Validating API responses (e.g., ensuring calendar events are formatted correctly).

Ensuring new skills added by the user are syntactically correct before deployment.

**2.6. Components Design Implementation**

Frontend (React):

Components for voice input, skill management, and user settings.

Each component should follow a modular design to ensure reusability.

Backend (Python):

A Flask or FastAPI service to receive and process voice/text commands.

TensorFlow NLU component to interpret and process commands.

APIs to control external services like calendars, email systems, and IoT devices.

**2.7. Configurations/Settings**

Configuration files will store API keys, server addresses, and other critical information.

A .env file will be used for managing environment variables (API keys, database connection strings, etc.).

Users will configure their own third-party integrations (like Gmail or Google Calendar) through the

web interface.

**2.8. Interfaces to Other Components**

Web interface (MERN) will communicate with the backend via RESTful APIs.

Python backend will interact with external services (calendars, emails, IoT devices) through API calls and

return responses to the web interface.

3.DATA DESIGN

**3.1 Database**

Skills: Stores custom user-defined skills (task name, description, action script).

Command Logs: Tracks user commands and the system's responses.

Preferences: Stores user preferences for various settings (e.g., language, task priority)

**3.2 Key design considerations in data design**

The database should ensure scalability and handle numerous custom skills from multiple users.

Logs should be time-stamped and archived periodically for performance reasons.

Relations between user data and skills should be optimized for fast querying and loading in the web interface.

4. DETAILS OF OTHER FRAMEWORKS BEING USED

**4.1. Session Management**

Sessions will be managed using JWT (JSON Web Token) for securing user interactions across the web interface.

JWT tokens will be used to authenticate users for API requests between the MERN front-end and Python backend.

**4.2. Caching**

Caching will be implemented using Redis to store frequent requests (such as NLU model responses for similar tasks).

Caching will be applied to improve the performance of repetitive tasks (e.g., fetching calendar data).

5. UNITT TESTING

**Unit testing will cover:**

Testing the NLU component to ensure correct interpretation of commands.

Testing the API integrations for emails, calendars, and IoT devices.

Testing the MERN interface components (React) to ensure proper validation and user experience.

Testing frameworks:

Python's unittest or pytest for backend testing.

Jest and Enzyme for frontend component testing in React.

**6.KEYNOTE**

To the future of intelligent interaction with our AI Voice Assistant, powered by Python. This cutting-edge model leverages advanced natural language processing and machine learning techniques to deliver a seamless and intuitive user experience. Designed to understand and respond to a wide range of commands, our voice assistant can perform tasks from setting reminders and answering queries to controlling smart home devices and providing real-time information. With its robust architecture and continuous learning capabilities, this AI model not only adapts to individual user preferences but also improves over time, ensuring a personalized and efficient assistant that evolves with your needs. Join us as we explore the limitless possibilities of voice-driven technology and redefine the way we interact with the digital world.

**7.REFERENCE**