Govt. Model Engineering College, Ernakulam
Department of Computer Engineering
B. Tech. Computer Science & Engineering

SAGE - AI Powered Smartglass

Software Requirements Specification (SRS)

Group B5
Project Members:

Ananya P (22CSB08 MDL22CS036)
M Gayathri (22CSB36 MDL22CS126)
Navneet Ranjish Pillai (22CSB41 MDL22CS138)
Nikhil M (22CSB44 MDL22CS144)

October 2025

Contents

1 Software Requirements Specification

1.1 Introduction

1.1.1 Purpose

The purpose of this document is to provide a detailed Software Requirements Specification (SRS) for the AI-Powered Smartglass (SAGE) Prototype. This document captures the functional and non-functional requirements of the system to guide the development team and communicate the project scope and objectives to all stakeholders.

1.1.2 Intended Audience and Reading Suggestions

This document is intended primarily for:

- Software developers responsible for the design and implementation of the smartglass system.
- Testers who will verify system functions against the requirements.
- Project managers and stakeholders interested in project scope and deliverables.
- Future maintenance teams who will update or enhance the system.

It is recommended that readers familiarize themselves with the overall system description in later sections for a comprehensive understanding.

1.1.3 Project Scope

The AI-Powered Smartglass (SAGE) Prototype is a wearable device integrating advanced AI-powered features such as voice-activated commands, heads-up display, object and face recognition, and translation capabilities. The system consists of a smartglass hardware platform, a companion mobile application, and cloud AI services. The solution targets users needing situational assistance and enhanced accessibility functionality. The project encompasses development of both hardware and software components, integration with AI APIs, and a user-friendly interface.

1.1.4 Overview of Developer's Responsibilities

The development team is responsible for:

- Designing and implementing the smartglass hardware and software.
- Developing the companion mobile application to interface with the smartglass.
- Integrating external AI services for natural language processing, vision, and translation.
- Ensuring secure and reliable communication between devices.
- Conducting system testing and documentation.
- Providing user documentation and usage guidelines.

.

1.2 Overall Description

1.2.1 Product Perspective

- Integrates wearable hardware (smartglass) with a companion mobile app.
- Utilizes cloud and edge AI services for real-time natural language and vision processing.
- Replaces traditional glass functions with intelligent features: HUD, voice commands, translation.
- Modular design facilitates easy software updates and hardware upgrades.
- Supports seamless communication over Wi-Fi hotspot created by the mobile device.

1.2.2 Product Functions

- Voice-activated wake word detection and command recognition.
- Real-time heads-up display of notifications, translations, and recognition results.
- Object and face recognition via camera feeds and ML models.
- Support for real-time multi-language translation of text and speech.
- Mobile app interface for user interaction and AI backend communication.
- Secure pairing and fallback network recovery modes.
- Optional features like gesture controls and call management.

1.2.3 User Classes and Characteristics

- **End Users**: Require hands-free assistance, including visually impaired and tech enthusiasts.
- **Developers**: Implement hardware and software, integrate APIs, and maintain the system.
- **Administrators**: Manage network setup, device pairing, and user security.
- **Testers**: Verify functionality and system performance via manual and automated tests.

1.2.4 Operating Environment

- Embedded Linux environment on Raspberry Pi in the glasses.
- Companion mobile application on Android/iOS.
- Wi-Fi hotspot network established by mobile device for communication.
- Cloud AI APIs accessible via Internet through mobile app.
- Hardware peripherals: camera, microphone, TFT display, buttons.

1.2.5 Design and Implementation Constraints

- Limited processing power and thermal budget on wearable hardware.
- Dependence on mobile device hotspot means network quality varies.
- Battery life constraints require efficient power management.
- API dependencies require stable external AI service availability.
- User interface must remain simple to accommodate diverse users.

1.2.6 User Documentation

- User guide explaining setup, usage of voice commands, and app features.
- Troubleshooting section for network and device issues.
- API documentation for developers detailing integration and extension.
- Online FAQs and tutorial videos hosted on project website.

1.2.7 General Constraints

- System performance affected by network latency and AI service response times.
- Physical comfort and weight limits restrict hardware component choices.
- Legal and privacy regulations govern data handling and facial recognition use.
- Real-time processing requires optimized code and multi-threading.

1.2.8 Assumptions and Dependencies

- Users own compatible smartphones with hotspot capability.
- Users have stable internet access for cloud API interactions.
- Third-party AI APIs remain operational and accessible.
- User environment supports voice and gesture interactions without excessive noise.
- All stakeholders will cooperate to integrate system properly with other services.

1.3 External Interface Requirements

1.3.1 User Interfaces

- Mobile app supports voice commands, feature toggling, and status monitoring through touch and speech.
- Smartglass HUD displays navigation, translation, and recognition results with high-contrast, non-intrusive visuals.
- Tactile buttons on the glass enable actions like camera capture and voice assistant activation.
- Audio feedback complements visual cues to aid accessibility.

1.3.2 Hardware Interfaces

- Raspberry Pi GPIO and SPI connect cameras, displays, and buttons.
- USB/I2S microphones ensure clear voice input.
- Bone conduction speakers provide unobtrusive audio output.
- Power management circuitry supports rechargeable batteries.
- Sensors like accelerometers enable gesture and motion detection.

1.3.3 Software Interfaces

- REST APIs facilitate real-time data exchange between smartglass and mobile app.
- AI integrations include Google Vision, Gemini, and LibreTranslate APIs.
- HTTPS ensures encrypted and secure communications.
- Modular design allows easy addition of new AI services.
- On-device software handles voice activation and HUD display rendering.

1.3.4 Communications Interfaces

- Primary communication over Wi-Fi hotspot created by paired mobile device.
- Use of TCP/IP with HTTPS for secure and reliable data transmission.
- Automatic fallback to local access point mode on network failures.
- Bluetooth Low Energy used for pairing and device discovery.
- Communication protocols optimized for low latency and bandwidth efficiency.

1.4 Hardware and Software Requirements

1.4.1 Hardware Requirements

- Wearable computing platform: Raspberry Pi 4 or similar embedded Linux SBC with sufficient processing power and GPIO interfaces.
- High-resolution camera module for image capture and video streaming.
- TFT heads-up display (HUD) with reflection optics for overlay visuals.
- Microphone array capable of noise-cancelled voice input via USB or I2S.
- Bone conduction speakers for discreet audio output.
- Rechargeable Li-ion battery with power management circuitry, supporting at least 4-6 hours of continuous use.
- Sensors including accelerometer and gyroscope for gesture and motion detection.
- Physical control buttons for camera capture and voice assistant activation.
- Wireless communication hardware supporting Wi-Fi (for hotspot mode) and Bluetooth Low Energy (BLE) for device pairing.

1.4.2 Software Requirements

- Embedded Linux operating system (e.g., Raspberry Pi OS) with support for Python and C++ development.
- Voice activation and wake word detection software running locally.
- Software modules for image capture, streaming, and data transmission to companion mobile app.
- Companion mobile application implemented in Flutter for Android and iOS, managing AI service communication and user interface.
- Integration with cloud-based AI APIs (e.g., Gemini, Google Vision, LibreTranslate) for natural language processing, object detection, face recognition, and translation.
- Secure communication protocols (HTTPS, TLS) for data transmission.
- User interface software on smartglass for heads-up display rendering and audio feedback.
- Firmware for controlling sensors, managing power, and handling device startup and fallback modes.

1.5 Functional Requirements

1.5.1 Wake Word Detection and Voice Commands

- The system shall continuously listen for a predefined wake word ("Hey Glass") to activate voice command mode. (High priority)
- Upon detecting the wake word, the system shall capture and process voice commands for various functions including navigation, translation, object recognition, and system control. (High priority)

1.5.2 Heads-Up Display (HUD) Features

- The HUD shall display navigation cues, notifications, translated text, and recognition results in real-time with clear visibility. (High priority)
- The display shall support dynamic content updates with minimal latency. (High priority)
- The display brightness and contrast shall adjust automatically based on ambient lighting conditions. (Medium priority)

1.5.3 Image Capture and Streaming

- The system shall capture images or short video clips upon voice command or physical button press. (High priority)
- Captured data shall be streamed securely to the companion mobile app for processing or stored locally for batch processing. (High priority)

1.5.4 AI-Based Recognition and Translation

- The system shall perform object detection using AI models to identify and describe objects in the user's environment. (High priority)
- Facial recognition shall be available to identify known individuals when permitted by the user. (Medium priority)
- The system shall provide real-time text and speech translation between supported languages. (High priority)

1.5.5 Mobile Application Integration

- The mobile app shall manage AI service API calls, user preferences, and system updates. (High priority)
- It shall facilitate secure communication and data synchronization with the smartglass. (High priority)

1.5.6 Device Pairing and Network Management

- Initial setup shall support smartglass-hosted access point mode for seamless pairing with the mobile app. (High priority)
- The system shall save and securely handle user credentials and network configurations. (High priority)
- Automatic fallback to access point mode shall be available to recover connectivity issues. (Medium priority)

1.5.7 Additional Features

- Gesture control support for select commands. (Optional)
- Integration of phone call handling using the built-in microphone and speakers. (Optional)
- Persistent memory of recognized objects and context to assist the user over time. (Optional)

1.6 Nonfunctional Requirements

1.6.1 Performance Requirements

- Ensure latency for voice commands and AI responses remains below 2 seconds.
- Support continuous operations with safe thermal limits.
- Maximize battery life for 4–6 hours of active use.

1.6.2 Safety Requirements

- Compliance with wearability standards and electronic safety.
- Avoid overheating through thermal management and usage scheduling.

1.6.3 Security Requirements

- Secure authentication on mobile app.
- Encryption of network communications.
- Secure storage of user and device credentials.
- Privacy-aware handling of facial data and images.

1.6.4 Software Quality Attributes

- Modularity for maintainability and ease of upgrades.
- Scalability enabling addition of new AI features.
- Reliability with graceful error handling and recovery modes.
- Usability emphasizing simple user interfaces and minimal training.

2 Software Design Document (SDD)

2.1 Introduction

The Software Design Document (SDD) defines the transformation of requirements into a comprehensive system design for the S.A.G.E - Personalized AI-Powered Smartglass System. It establishes the foundation for implementation, testing, and maintenance, focusing on architecture, data design, module interactions, and core algorithms.

2.1.1 Purpose

The primary goals of this SDD are:

- To translate functional and non-functional requirements into modular system components.
- To ensure scalability, coherence, and maintainability of the S.A.G.E Smartglass System.
- To design collaborative modules for detecting objects, recognizing faces, translating languages, providing navigation, and delivering voice responses.

2.1.2 Overview

The system design emphasizes the following principles:

- **Scalability**: Supporting real-time vision processing for multiple use cases.
- **Flexibility**: Adapting to various environments and user needs.
- **Modularity**: Independent modules for vision, voice, translation, and navigation.
- **Maintainability**: Simplified debugging and iterative improvements.

2.2 System Architecture

The system features a modular architecture where each module specializes in a smartglass task: object detection, facial recognition, language translation, navigation, or voice assistance. Communication across modules ensures seamless integration and synchronized outputs.

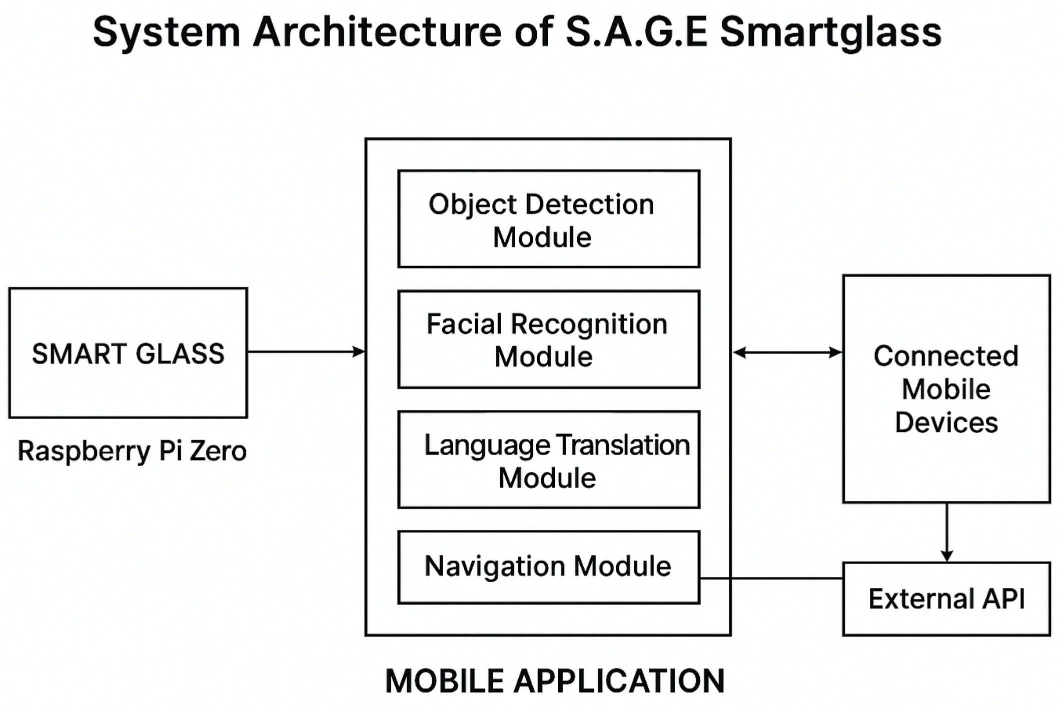

Figure 1: System Architecture Diagram

2.3 Use Case Diagram

The use case diagram illustrates the key interactions between the user and the SAGE Smartglass system. It highlights how users engage with various functional modules—such as object detection, facial recognition, language translation, navigation, and voice assistance—to perform tasks efficiently. Each use case represents a real-world scenario, ensuring the system aligns with user needs.

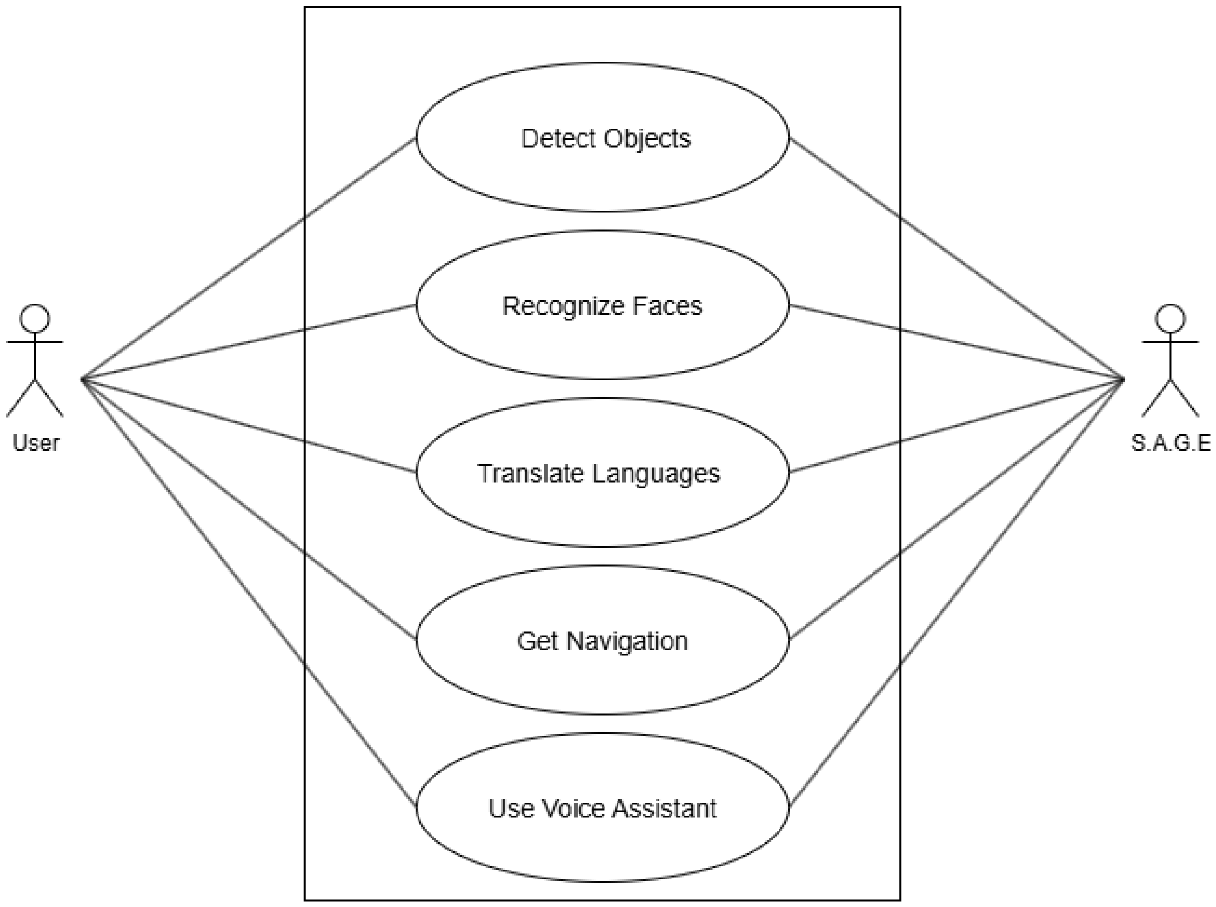

Figure 2: System Architecture Diagram

2.4 Module Descriptions

The S.A.G.E system consists of the following primary modules:

2.4.1 Object Detection Module

- **Input:** Real-time visual feed from the smartglass camera (Raspberry Pi Zero).
- **Computation:** YOLO-based object detection on a mobile application.
- **Output:** Detected objects are marked and displayed on the HUD. Text-to-Speech (TTS) provides audio feedback.

2.4.2 Facial Recognition Module

- **Input:** Live facial images from smartglass camera.
- **Processing:** Feature vectors are stored securely and compared for recognition.
- **Output:** Identified person's name announced through voice.

2.4.3 Language Translation Module

- **Input:** Visual/text content captured from environment (e.g. signboards).
- **Processing:** Gemini API for translation.
- **Output:** Translated content delivered as voice output through smartglass speaker.

2.4.4 Navigation Module

- **Input:** GPS data and map requests from mobile app.
- **Processing:** Google Maps API integration for route mapping.
- **Output:** Route, turns, proximity alerts on HUD and voice guidance for navigation.

2.4.5 Voice Assistant Module

- **Input:** Voice commands from microphone.
- **Processing:** Wake phrase detection and command interpretation using Gemini API.
- **Output:** Conversational responses and feedback via speaker.

2.5 Activity Diagram

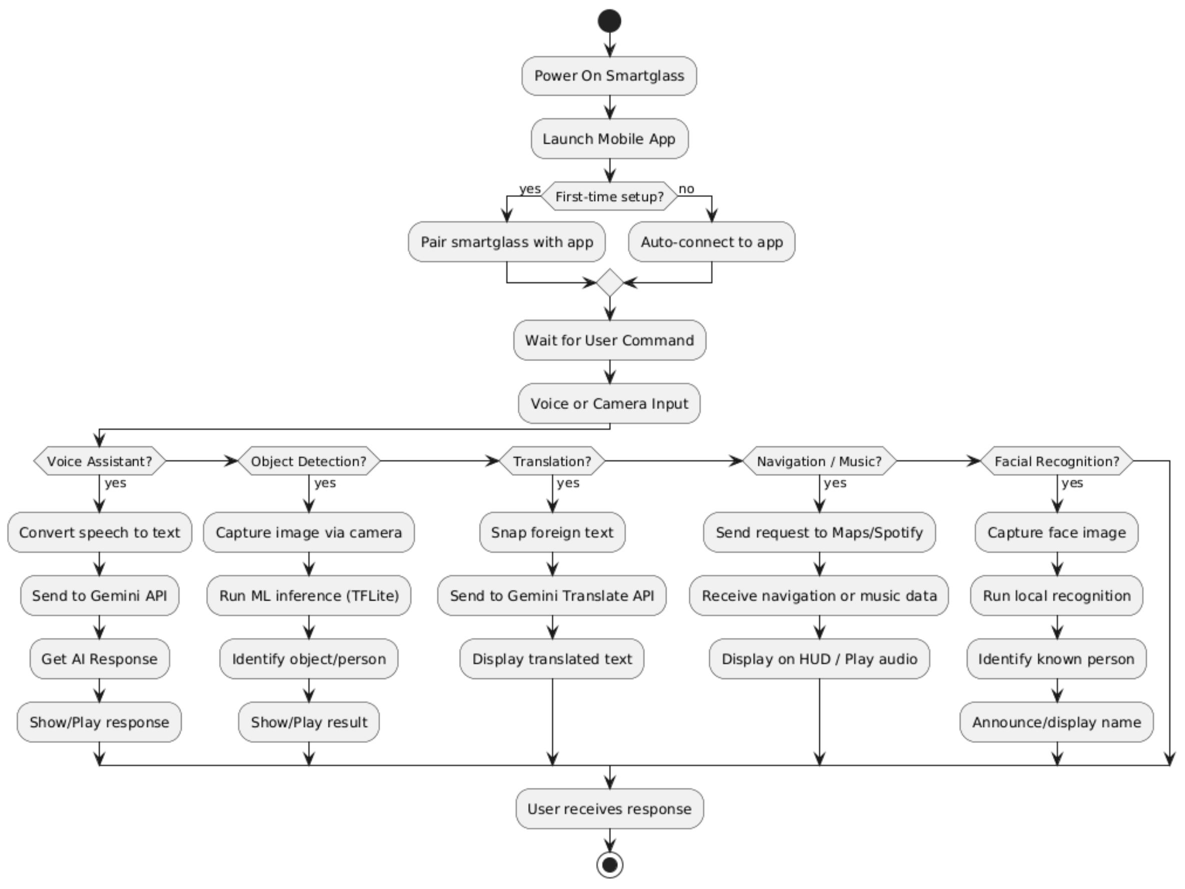

Figure 3: System Architecture Diagram

2.6 Data Flow Diagrams

2.6.1 Level 0

Provides a high-level overview of the Smartglass System. Shows user interaction for object detection, navigation, translation, and voice assistance.

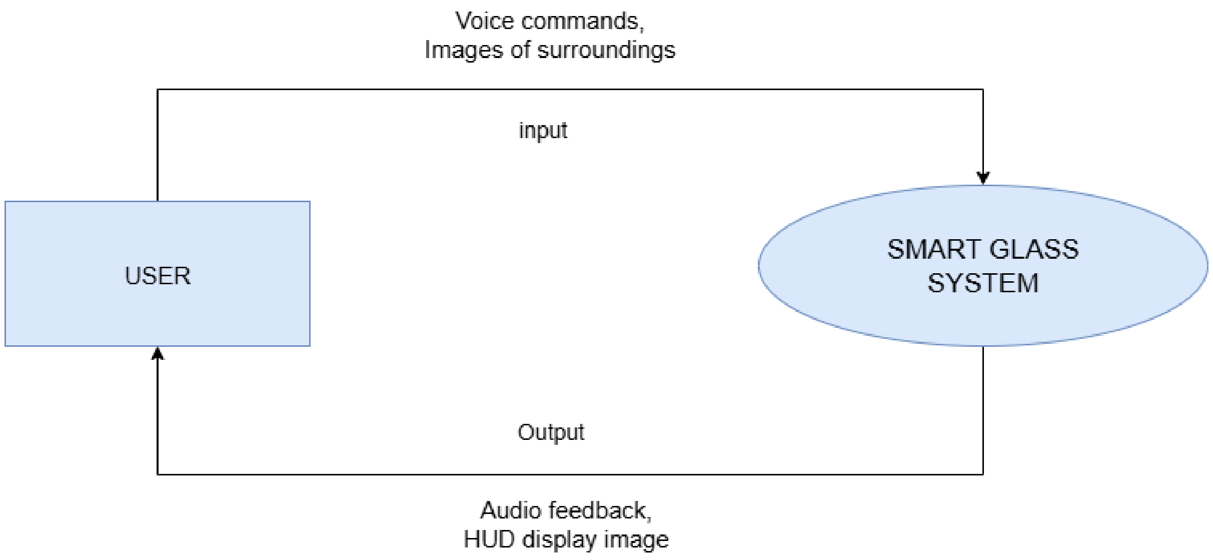

Figure 4: DFD: Level 0

2.6.2 Level 1

Breaks down system workflow into input collection, processing by individual modules, and output delivery.

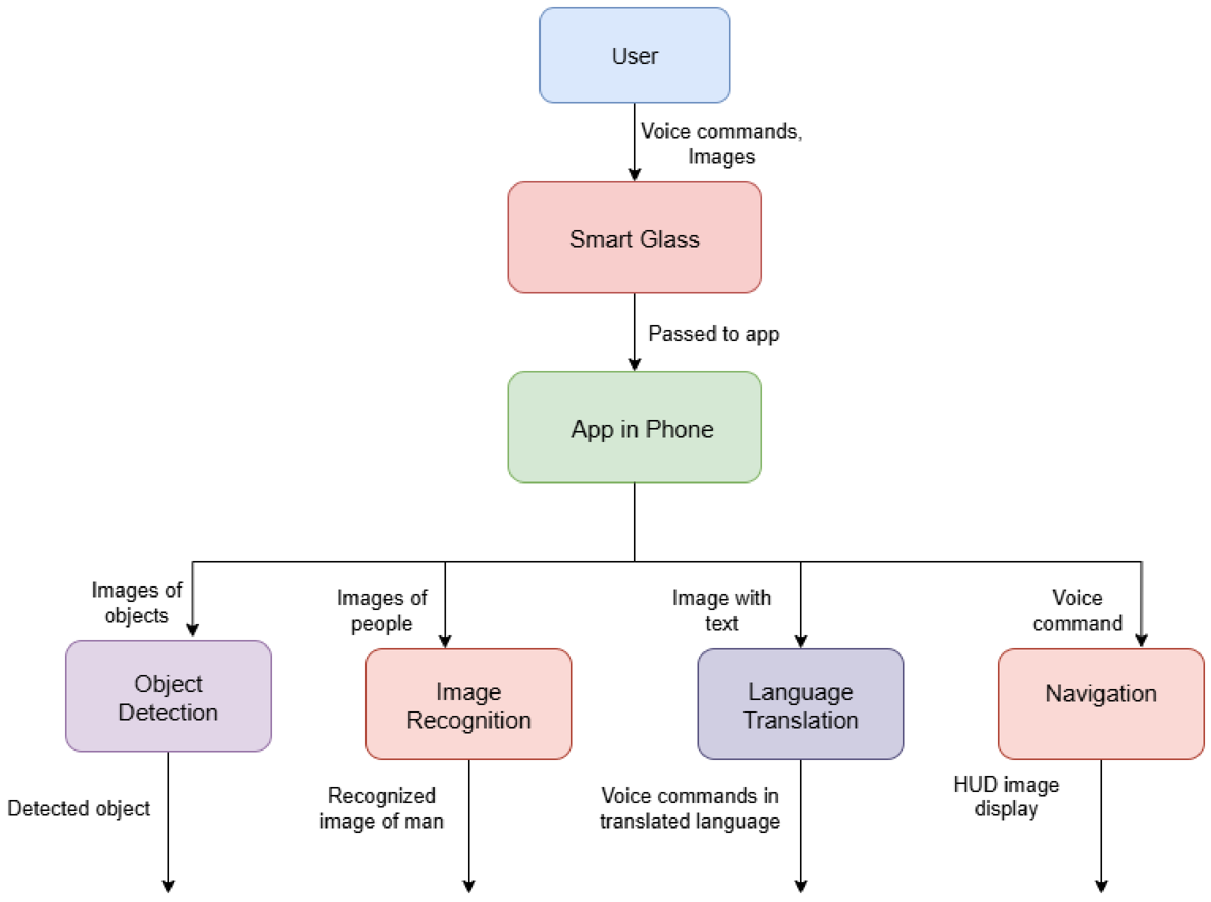

Figure 5: DFD: Level 1

2.6.3 Level 2

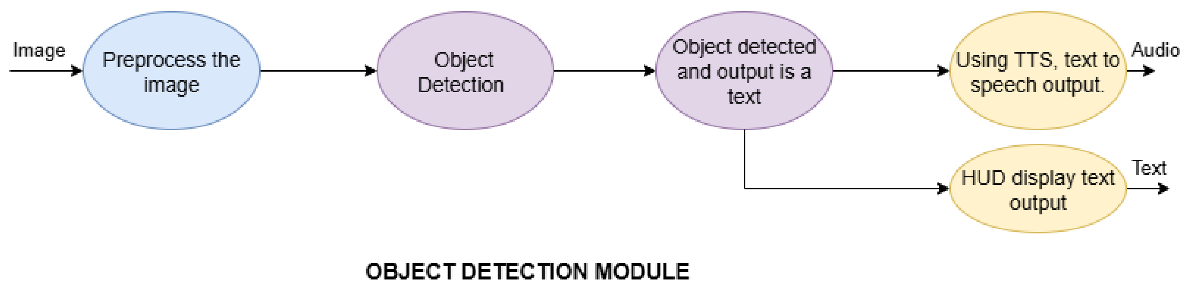

Figure 6: DFD: Level 2.1

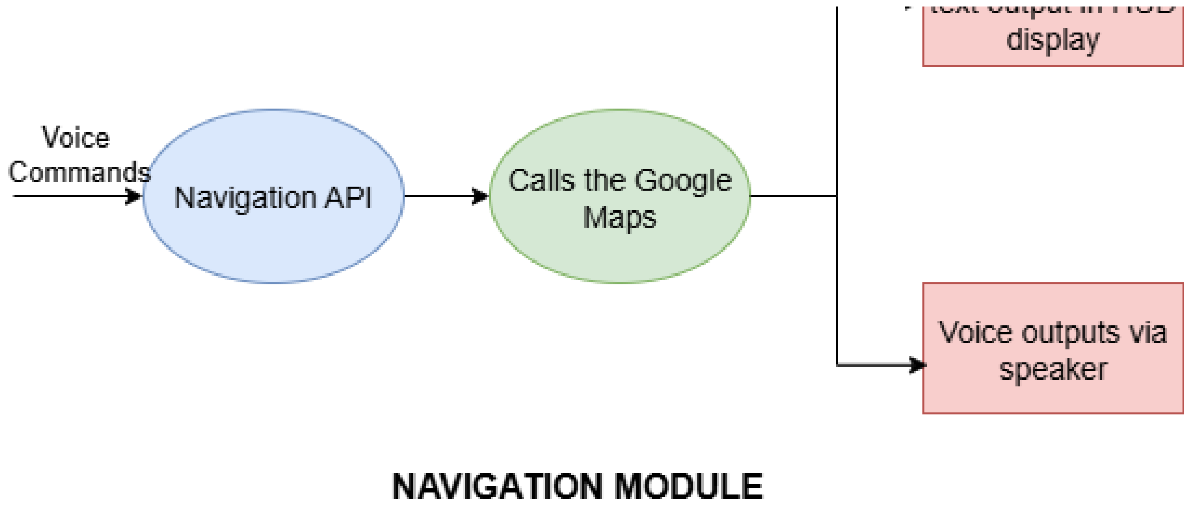

Figure 7: DFD: Level 2.2

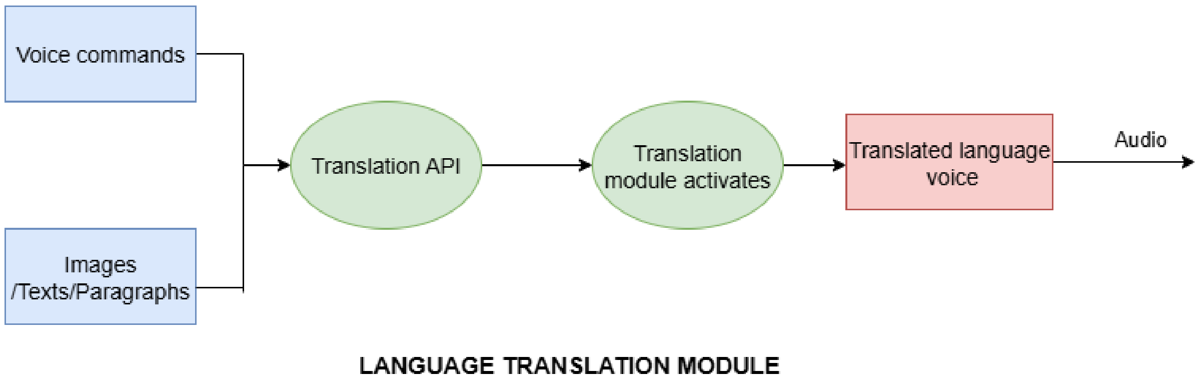
Voice commands
Images /Texts/Paragraphs
Translation API
Translation module activates
Translated language voice
Audio
LANGUAGE TRANSLATION MODULE

Figure 8: DFD: Level 2.3

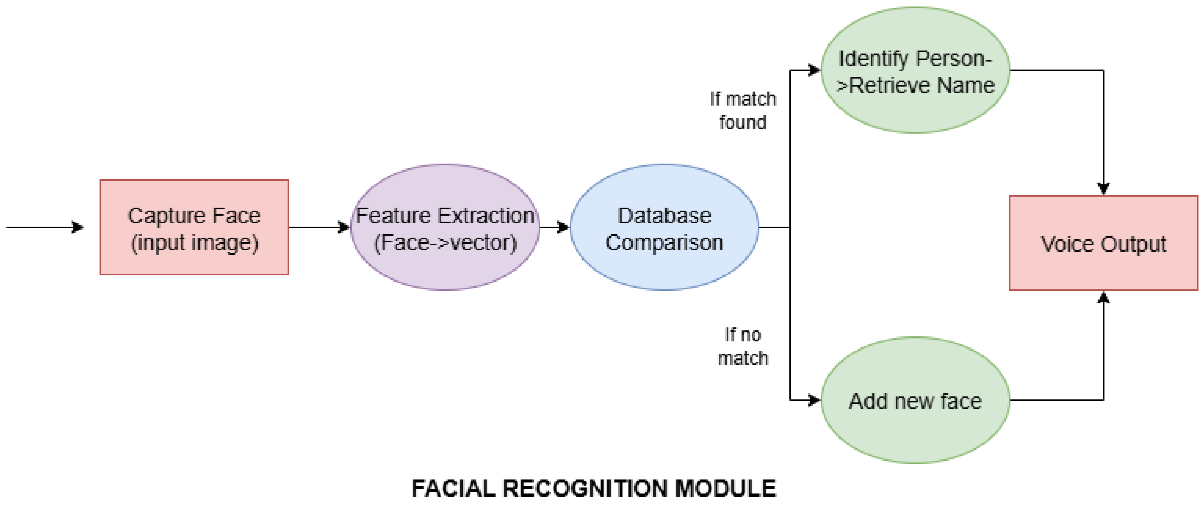
Capture Face (input image)
Feature Extraction (Face->vector)
Database Comparison
If match found
Identify Person->Retrieve Name
If no match
Add new face
Voice Output
FACIAL RECOGNITION MODULE

Figure 9: DFD: Level 2.4

2.7 Database Design

The database follows a normalized relational architecture:

- **Users**: Authentication details and preferences.
- **Detection Logs**: Records of detected objects/faces.
- **Translation Sessions**: Stores input and translated output.
- **Navigation History**: Saved routes and travel logs.

Foreign keys ensure data integrity across entities.

References

[1] Maria Pia Fanti, Beatrice Di Pierro, "A Connectivity Platform for Intermodal Transportation and Logistics Systems," 2018.

[2] Dr. Gyanesh Kumar Sinha, "Study of Indian Logistics Industry in Changing Global Scenario," February 2016.

[3] Gheorghe Minculete, Polixenia Olar, "Logistics Integration of 'Hub and Spoke' Model," 2014.

[4] Nader Ghaffari-Nasab, Mehdi Ghazanfari, Ebrahim Teimoury, "Hub-and-spoke logistics network design for third party logistics service providers," January 2015.

[5] Google Developers, "Google Vision API Documentation," 2025.

[6] Google, "Gemini AI API Technical Overview," 2025.

[7] LibreTranslate, "LibreTranslate API Documentation," 2025.

[8] Qualcomm, "How AI and Smart Glasses Give You a New Perspective on Real Life," 2025.

[9] ABI Research, "Optimizing User Experience for Smart Glasses," March 2020.