# Smart Vision App

# Software Requirements and Design Specifications

**Muhammad Qasim**: (SP22-BCS-005)
**Muhammad Imran**: (SP22-BCS-044)

# Table of Content

# 1 Introduction

## 1.1 System Introduction

The **Smart Vision App** is an AI-powered mobile application designed to assist visually impaired individuals by helping them recognize objects, read printed text, and navigate their surroundings independently. The app operates through a voice-controlled interface, enabling users to interact with it hands-free.
Once activated, the application uses AI-based **object detection** to identify and announce surrounding items and an **OCR (Optical Character Recognition**) module to read printed or handwritten text aloud. To enhance usability in low-light environments, the app also features an automatic flashlight function. Developed using Flutter for a smooth user interface and Vosk for offline voice recognition, the Smart Vision App delivers a reliable and accessible experience tailored for users with visual impairments.

## 1.2 Background of the System

Various assistive technologies exist for visually impaired individuals, such as screen readers and smart glasses. However, these solutions are often expensive and depend on continuous internet connectivity. Applications like Seeing AI and Google Lookout offer similar capabilities but typically rely on cloud services, making them less effective in offline or resource-limited settings.
The Smart Vision App offers an affordable, offline alternative by performing AI-based object recognition and OCR entirely on-device. It uses lightweight, efficient models to deliver a fast and intuitive experience. Unlike existing tools, this app is optimized for mobile devices and runs independently of external services, making it suitable for a wide range of users and environments.

## 1.3 Objectives of the System

- Enable real-time object recognition through a mobile camera
- Provide text-to-speech functionality for printed text via OCR
- Support complete offline voice command navigation
- Detect low-light conditions and activate flashlight automatically
- Ensure accessibility and usability for visually impaired individuals

## 1.4 Significance of the System

The Smart Vision App aims to improve the quality of life for visually impaired individuals by offering them greater independence in performing everyday tasks. It reduces reliance on human assistance by providing essential features like object recognition and text reading through simple voice commands. Since it works entirely offline and on standard smartphones, the app is practical for use in homes, public places, educational settings, and emergency situations. Its accessibility and affordability make it a valuable contribution to assistive technology, especially in areas with limited resources.

# 2 Overall Description

## 2.1 Product Perspective

The Smart Vision App is a standalone mobile application designed for Android smartphones. It is developed using the Flutter framework and integrates AI capabilities for object detection and text recognition. The app does not rely on cloud-based services or external hardware, ensuring that it runs entirely offline on the user's device. By using pre-trained and lightweight AI models, the app provides real-time assistance to visually impaired individuals without requiring internet connectivity. It makes use of the device's built-in camera, microphone, and flashlight to perform its functions effectively.
Diagram:

- Mobile Device
- Flutter App
- SignUp/Login
- Voice Module
- Object Detection
- OCR
- Flashlight Control
- Authentication
- AI Models (TFLite / ML Kit)
- Firebase

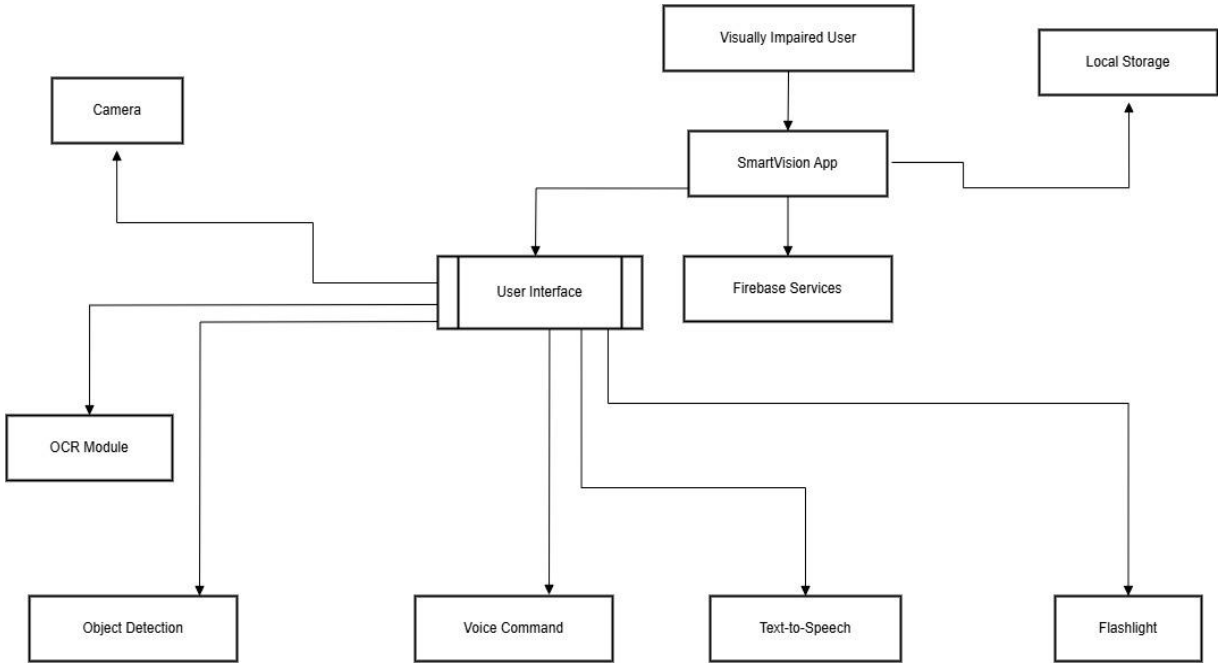


***Figure 2.1: System Design***

## 2.2 Product Scope

The primary goal of the Smart Vision App is to provide visually impaired users with tools to interact with their surroundings independently. The app allows users to:

- Recognize and identify nearby objects using the device's camera.
- Read printed text through OCR and convert it to speech.
- Use offline voice commands to control the app's core features.
- Automatically activate the flashlight in dark environments for improved detection.

## 2.3 Product Functionality

- Capture images from the mobile device camera
- Perform object detection using AI and announce results through TTS
- Read and extract text from images using OCR
- Provide voice command control to access features without touch input
- Auto-detect low-light conditions and turn on flashlight
- Save user preferences and interaction logs using SQLite and Firebase

- Provide offline functionality for core features like OCR, object detection, and voice command processing

## 2.3.1 Functional Requirement

- **Authentication & User Access**
  - The app shall allow users to securely log in and register using email and password via Firebase.
  - The app shall store and manage user profiles with preferences.
- **Camera & Image Capture**
  - The app shall use the phone camera to capture images in real time.
  - It shall stream live camera input for object detection.
- **Object Detection Module**
  - The system shall detect and identify objects in camera view using TensorFlow Lite.
  - The system shall announce detected objects using audio (Text-to-Speech).
- **OCR (Optical Character Recognition) Module**
  - The app shall extract text from printed material using Google ML Kit.
  - The extracted text shall be converted to audio and read aloud to the user.
- **Voice Command Module**
  - The system shall listen for user voice commands offline using Vosk.
  - Based on spoken commands, the system shall trigger actions (e.g., "Read Text", "Detect Object", "Close the app").
- **Flashlight Control Module**
  - The system shall detect ambient light levels.
  - If low light is detected, the flashlight shall automatically turn on.
- **Offline AI & TTS Integration**
  - The app shall work offline using on-device AI models and voice engines.
  - TTS (Text-to-Speech) will provide auditory feedback for all functions.
- **Logging & Storage Module**
  - The app shall store user interaction logs and extracted content to Firebase/SQLite.

- Logs will include timestamps, detected objects, and OCR text.

- **Accessibility & UI**
  - The app shall offer a simple, audio-guided interface suitable for blind users.
  - All buttons shall have large touch areas and speak aloud their function.

## 2.4 Users and Characteristics

- Primary users: Visually impaired individuals
- Limited or no screen interaction
- Rely on voice and audio feedback
- Prefer simple, accessible, and fast UI
- Secondary users: Developers, testers, and caregivers

## 2.5 Operating Environment

- Android mobile device (minimum Android 10)
- Internet required for Firebase sync
- Offline features (OCR, Object Detection, Voice Commands) must run locally
- Tools: Flutter SDK, TensorFlow Lite, ML Kit, Firebase

# 3 Specific Requirements

## 3.1 Functional Requirements

The system shall provide the following functional requirements:

- **FR1:** The system shall recognize and announce nearby objects using the device's camera and an object detection model.
- **FR2:** The system shall extract and read printed text using OCR technology and convert it to speech using TTS.
- **FR3:** The system shall accept voice commands for tasks such as:
  - "Identify object"
  - "Read text"
  - "Close the app"
- **FR4:** The system shall turn on the device flashlight automatically in low-light environments.

- **FR5:** The system shall provide haptic feedback after completing object or text recognition tasks.
- **FR6:** The system shall operate entirely offline without needing internet access.
- **FR7:** The system shall display a basic UI allowing manual selection of core features (identify object, read text).
- **FR8:** The system shall give audible feedback using text-to-speech for all recognized content and system actions.

## 3.2 Behaviour Requirements

**Actors:**

- User (Visually Impaired)

**Use Cases:**

- Detect Object
- Read Text
- Speak Output
- Voice Command
- Turn On Flashlight

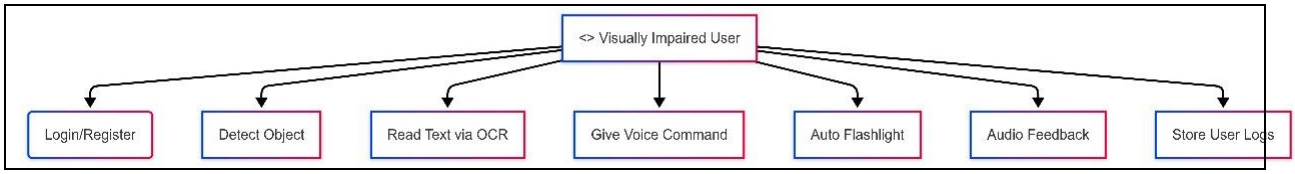


***Figure 3.1: Use Case Diagram***

## 3.3 External Interface Requirements

### 3.3.1 User Interfaces

- Home screen with large icons and text-to-speech guidance
- Settings screen to select voice language/speed
- Voice feedback for every action

### 3.3.2 Hardware Interfaces

- Rear camera for capturing images
- Light sensor for ambient light detection
- Microphone for voice commands

### 3.3.3 Software Interfaces

- Android SDK for system access
- Firebase SDK for auth and database
- TFLite models for object detection
- Google ML Kit for OCR
- Vosk for offline speech-to-text

### 3.3.4 Communications Interfaces

- HTTPs used for Firebase sync
- Data encrypted using Firebase security rules
- Voice commands processed offline (no cloud dependency)

# 4 Other Non-functional Requirements

## 4.1 Performance Requirements

- Object detection response time < 2 seconds
- OCR and speech output within 3 seconds
- Voice command recognition < 1.5 seconds
- Total app size < 80 MB

## 4.2 Safety and Security Requirements

- Encrypted communication with Firebase
- No sensitive data stored locally without encryption
- Users can delete their data anytime
- Offline fallback if Firebase is unreachable

- App asks for only required permissions

## 4.3 Software Quality Attributes

### 4.3.1 Reliability

- Runs without crashes on 90% of tested Android devices

### 4.3.2 Usability

- UI designed for blind users (audio-first)
- No visual-only interface components

### 4.3.3 Maintainability

- Modular architecture with separate logic for OCR, AI, Voice, and Auth
- Code documented with inline comments

### 4.3.4 Portability

- Flutter framework allows future iOS version
- Model fooooioooiiiooooooo99rmats (TFLite) supported across platforms

### 4.3.5 Adaptability

- Language and TTS engine easily changeable from settings

# 5 Design Description

## 5.1 Composite Viewpoint

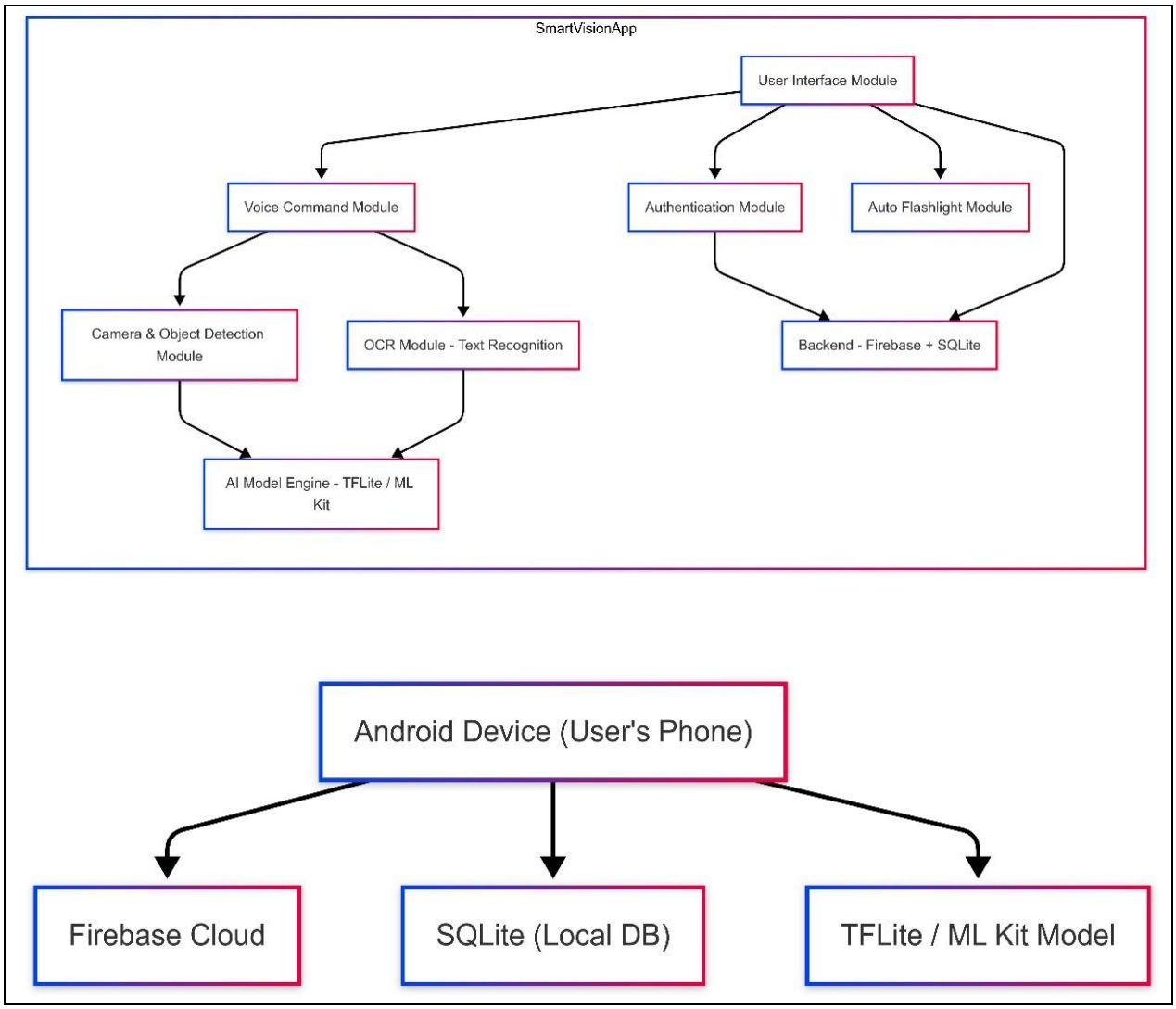


*Figure 5.1: Composite View*

## 5.2 Logical Viewpoint

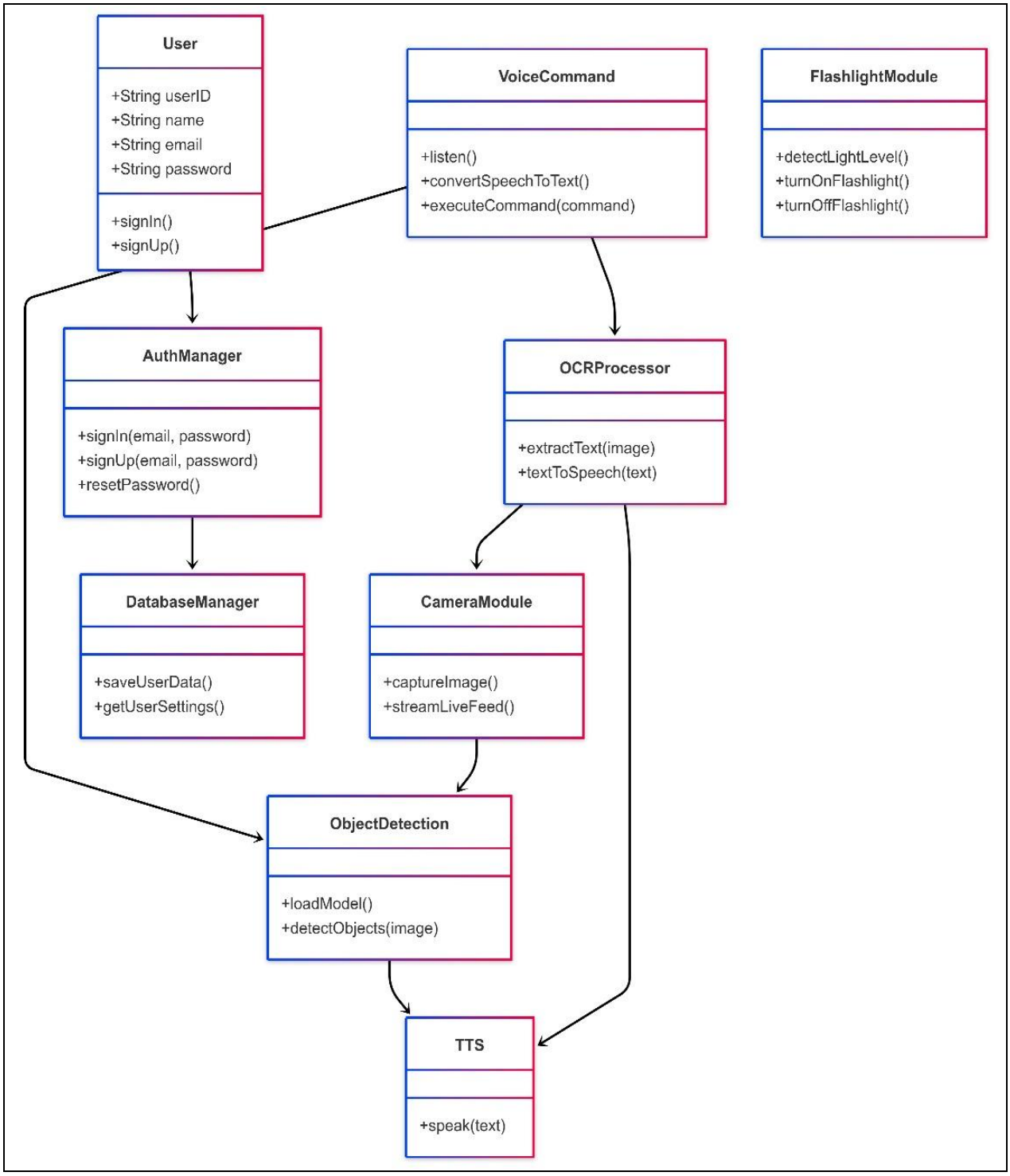


*Figure 5.2: Logical Viewpoint*

## 5.3 Information Viewpoint

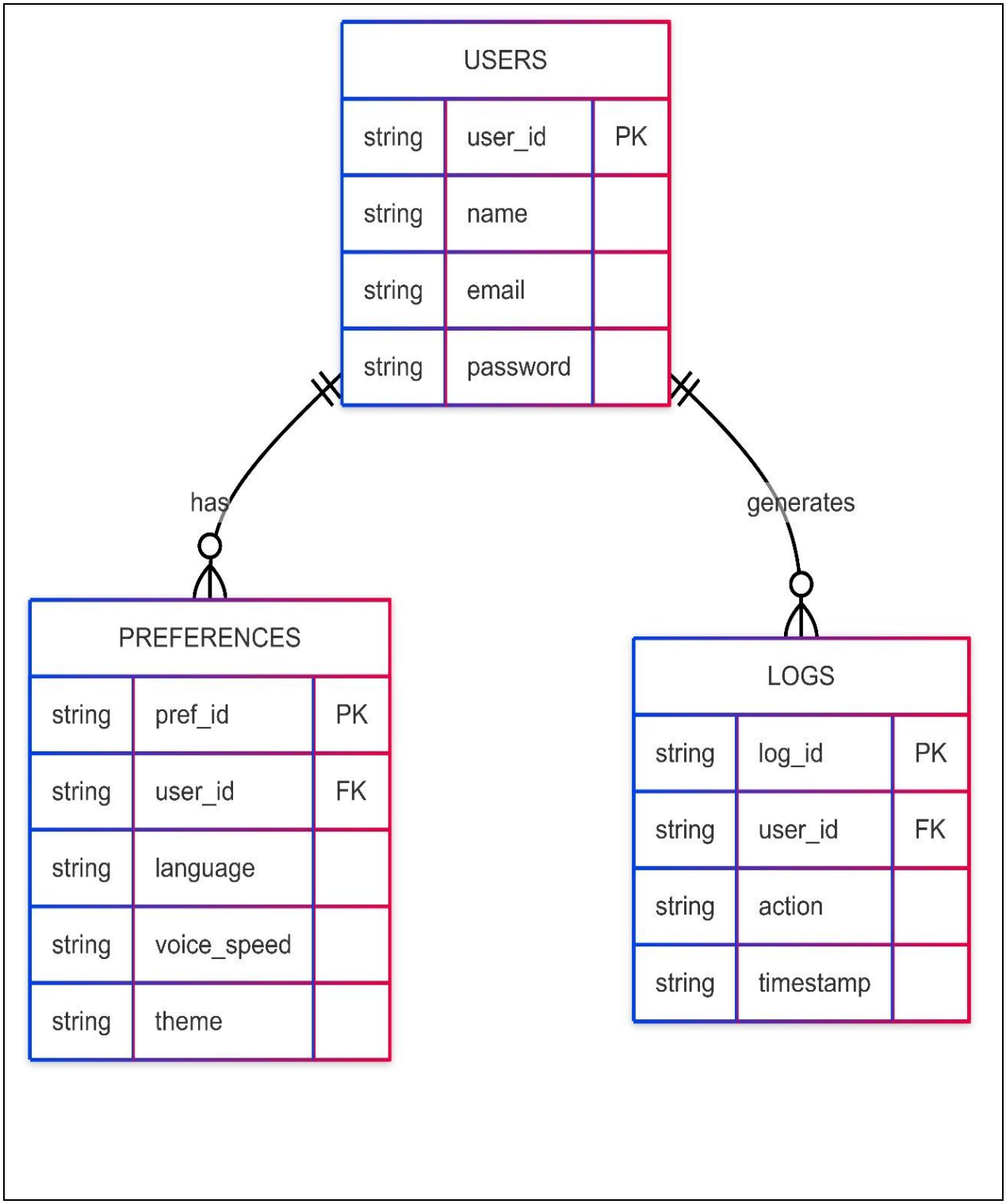


*Figure 5.3: Informative View*

## 5.4 Interaction Viewpoint

**Sequence Diagram for Login/Signup Module**

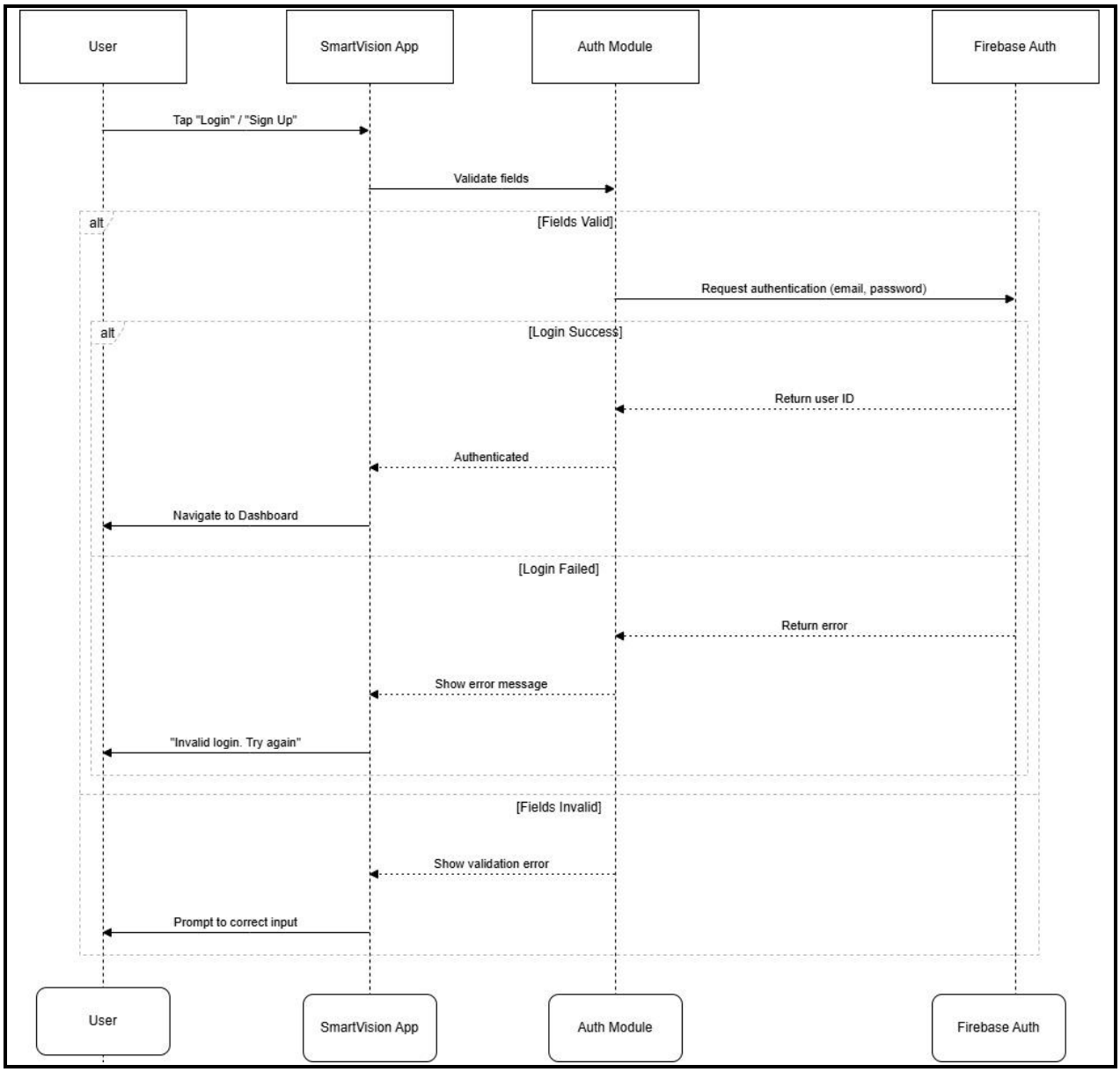


*Figure 5.4: Sequence Diagram of Login/Signup Module*

**Sequence Diagram for Voice Module**

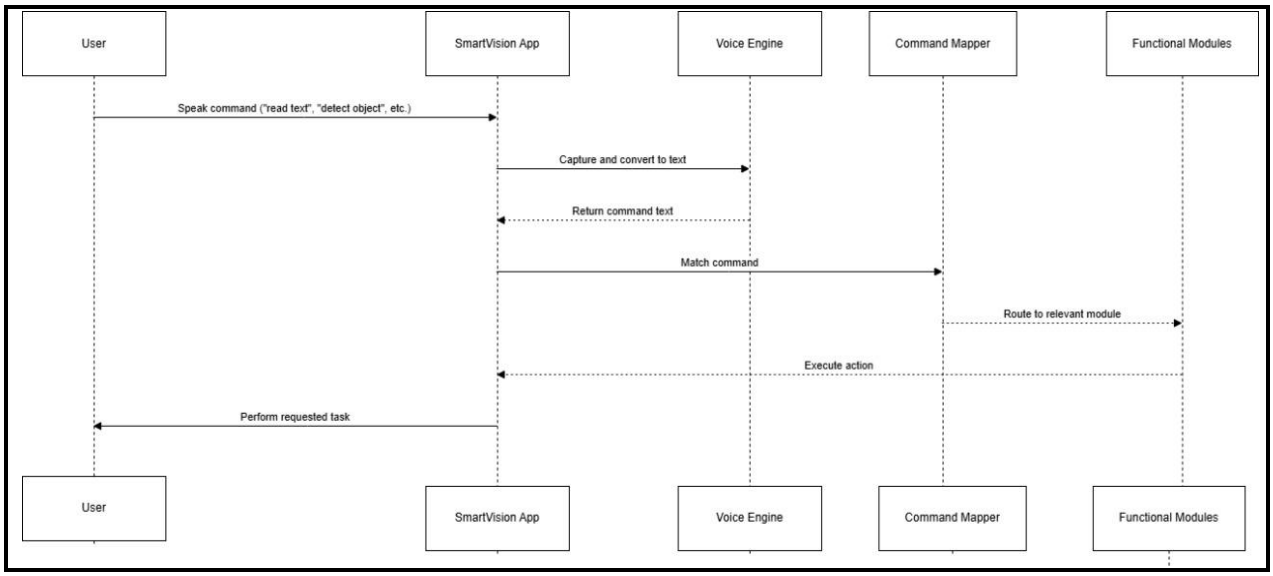


*Figure 5.5: Sequence Diagram of Voice Commands Module*

## Sequence Diagram for Object Detection Module

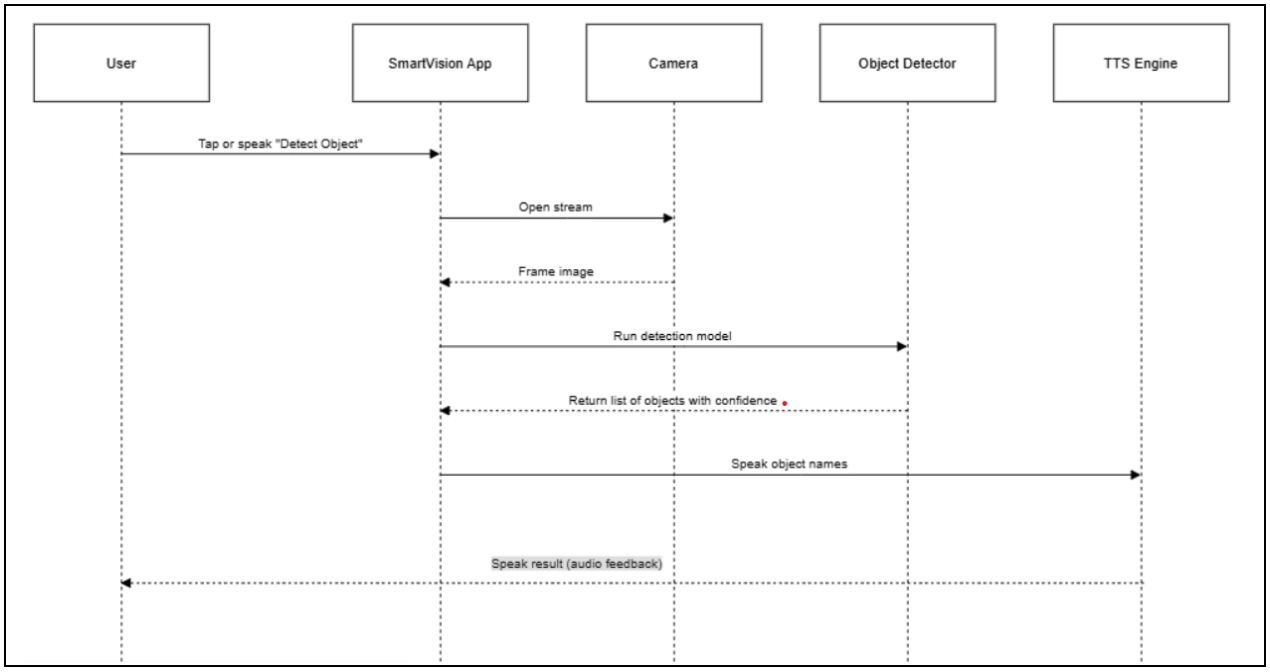


*Figure 5.6: Sequence Diagram of Object Detection Module*

**Sequence Diagram for OCR Module**

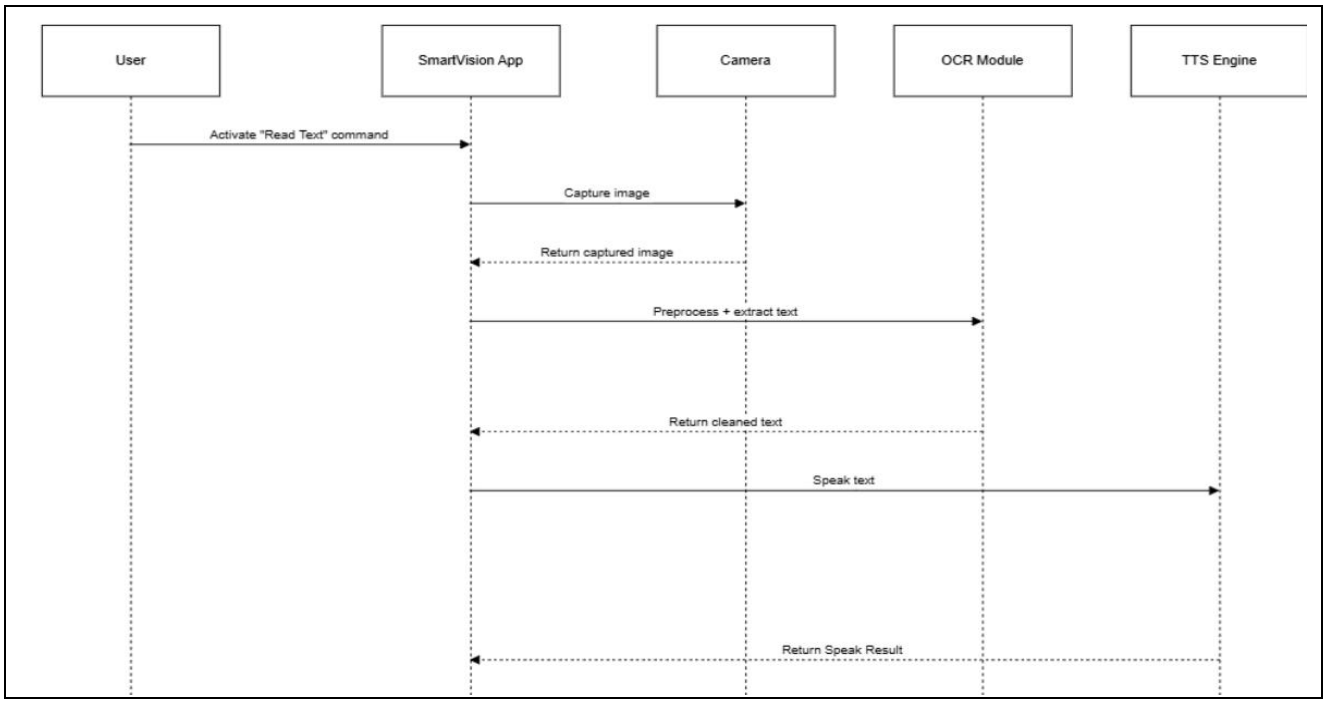


*Figure 5.7: Sequence Diagram of OCR Module*

**Sequence Diagram for Flashlight Module**

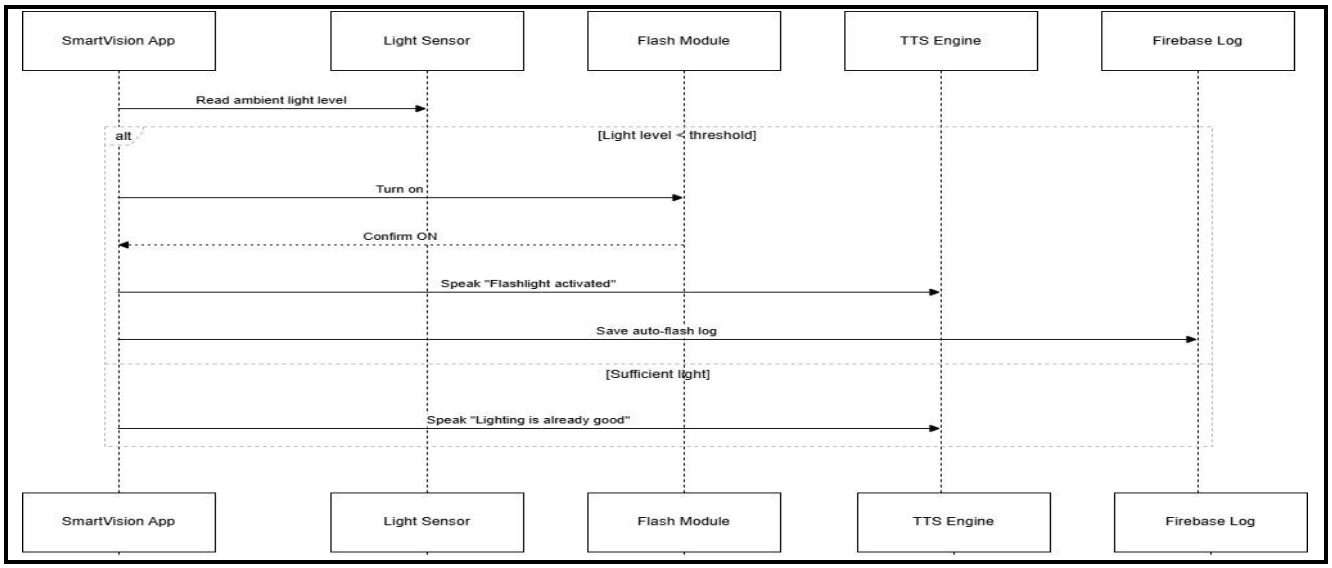


*Figure 5.8: Sequence Diagram of Flashlight Module*

**Sequence Diagram for feedback Module**

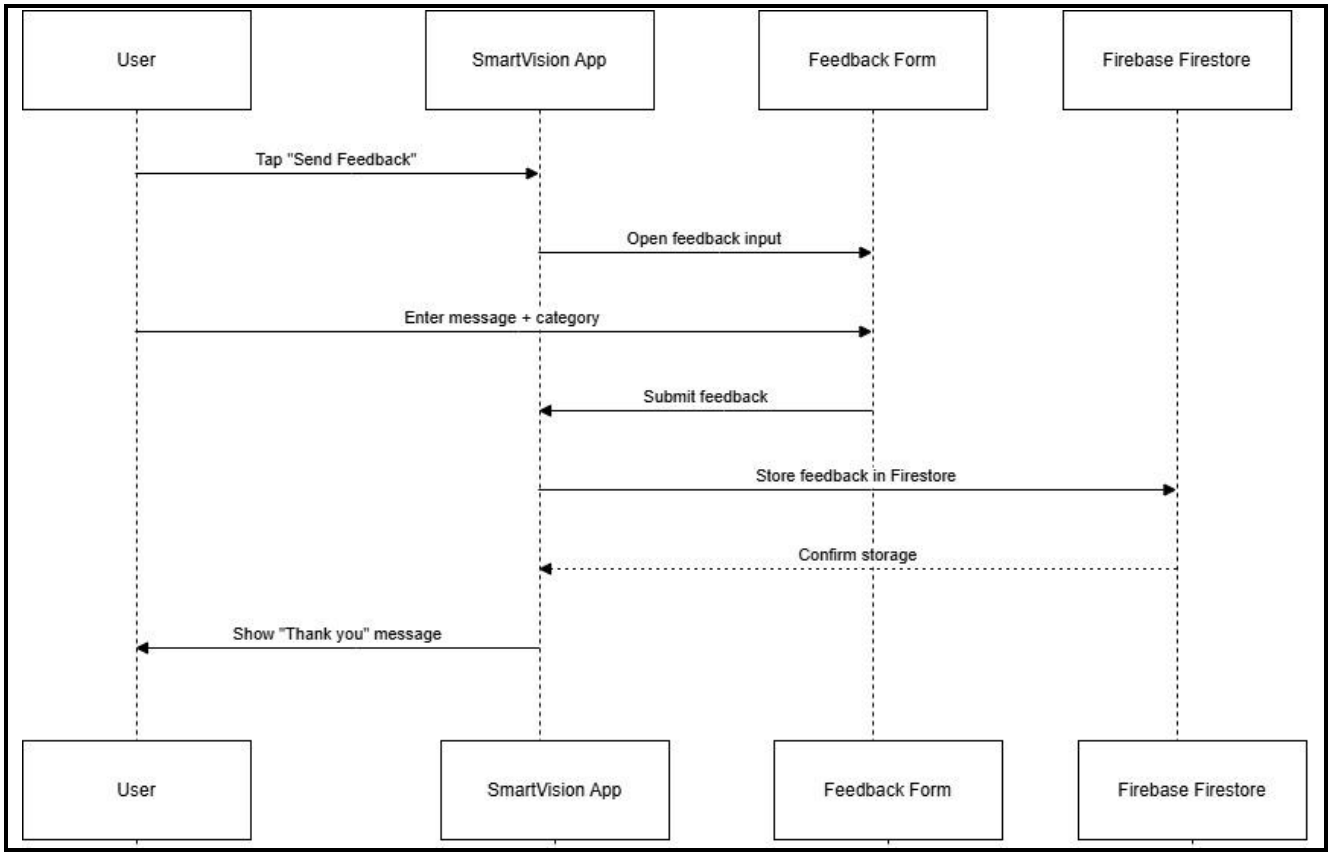


*Figure 5.9: Sequence Diagram of Feedback Module*

## 5.5 State Dynamics Viewpoint

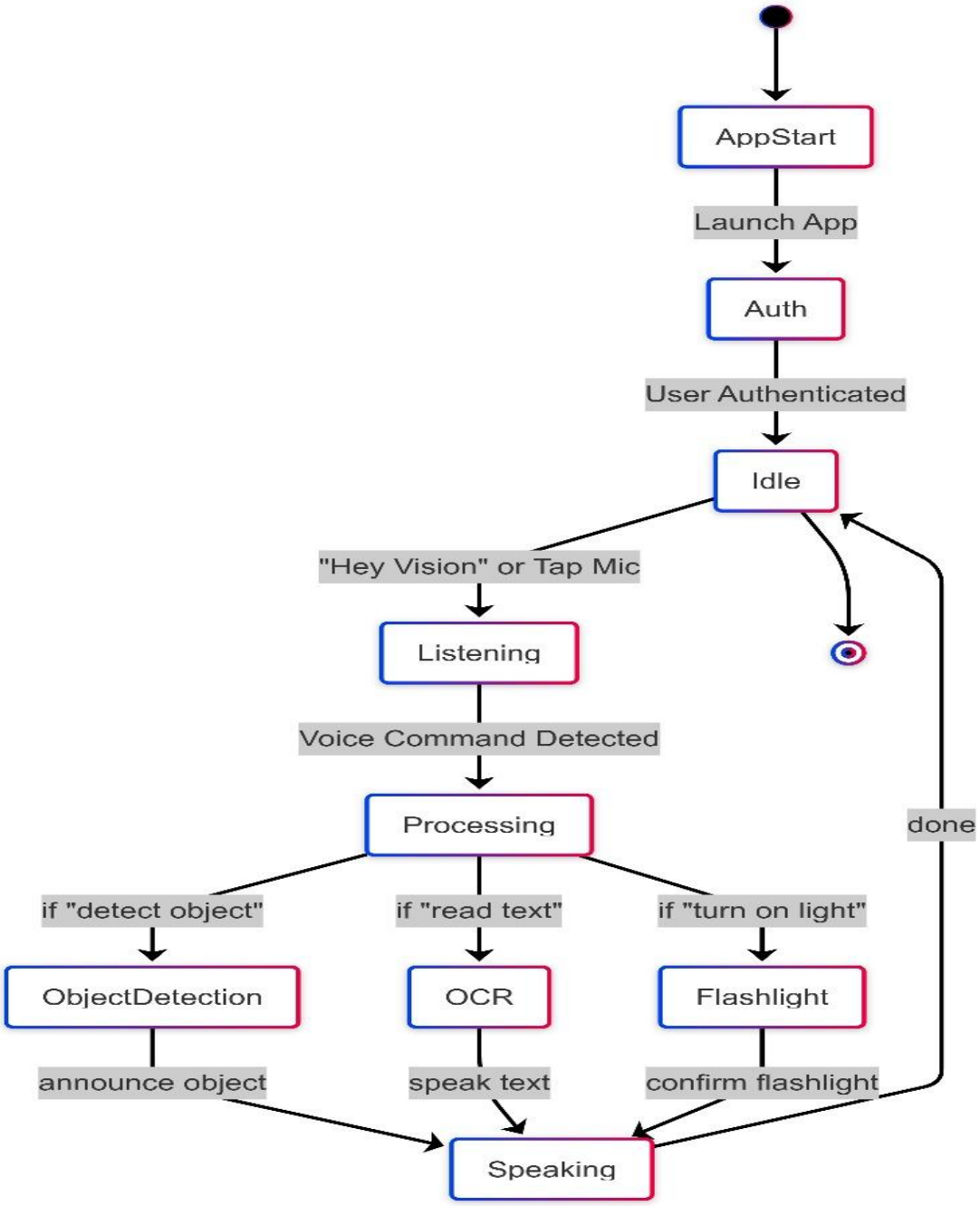


*Figure 5.10: State Dynamic View*

## 5.6Algorithm Viewpoint

### 5.6.1 Pseudocode 1: Object Detection Flow

```
FUNCTION detectObjectsFromCamera():
   INIT camera
   LOAD TFLite object detection model
   WHILE app is running:
      CAPTURE frame from camera
      IF frame is not empty:
         PROCESS frame using TFLite model
         DETECT objects and their positions
         FOR each detected object:
            GET object label (e.g., "bottle", "chair", "phone")
            ANNOUNCE label using Text-to-Speech
         IF no objects detected:
            ANNOUNCE "No object detected"
      WAIT for short delay (to avoid flooding)
   END WHILE
END FUNCTION
```

### 5.6.2 Pseudocode 2: Read Text from Image Flow

```
FUNCTION startRealTimeTextRecognition():

   INIT camera
   LOAD on-device OCR engine (e.g., Google ML Kit)

   WHILE camera is streaming:
      CAPTURE current frame
      IF frame is valid:
         EXTRACT text using OCR engine
         IF text is detected AND text is not repeated:
            FORMAT the extracted text
            READ text aloud using Text-to-Speech
            STORE the last read text to avoid repetition
      WAIT for short delay (e.g., 500 milliseconds)

   END WHILE

END FUNCTION
```

### 5.6.3 Pseudocode 3: Voice Command Handler

```
FUNCTION voiceCommandHandler():
```

```
   INITIALIZE voice recognition engine (e.g., Vosk)
   LOAD predefined commands:
      - "identify object"
      - "read text"
      - "close the app"

   START listening for voice input

   WHILE app is running:
      WAIT for user to speak
      PROCESS audio input
      CONVERT speech to text using voice recognition

      IF recognizedText == "identify object":
         CALL detectObjectsFromCamera()

      ELSE IF recognizedText == "read text":
         CALL readTextFromImage() OR startRealTimeTextRecognition()

      ELSE IF recognizedText == "close the app":
         CALL closeApp()

      ELSE IF recognizedText == "turn on flashlight":
         CALL turnOnFlashlight()

      ELSE:
         ANNOUNCE "Command not recognized"
   END WHILE
END FUNCTION
```