Software Design Specification

Project Title :

Environment Detection System for Blind Users

Project Code :

EDBP-2025

Internal Advisor :

Prof. Muhammad Fahad

Project Manager :

Dr. Muhammad Ilyas

Project Team :

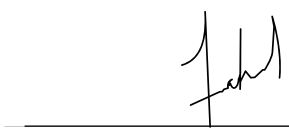
Ali Raza Ansari
BSCS51F22S008 (Team Lead)

Muhammad Abdullah Zammad
BSCS51F22S036

Muhammad Hassan Javed
BSCS51F22S040

Submission Date :

December 15, 2025



Project Manager's Signature

Document Information

Category	Information
Customer	Computer Science, University of Sargodha
Project	Environment Detection For Blind Person
Document	Software Design Specification
Document Version	1.0
Identifier	PGBHO1-2025-DS
Status	Draft
Arthor(s)	Ali Raza Ansari M. Abdullah Zammad M.Hassan Javed
Approver(s)	PM
Issue Date	Dec 17, 2025
Distribution	1. Advisor 2. PM 3. Project Office

Definition of Terms, Acronyms and Abbreviations

Terms	Description
UCD	Use Case diagrame
TTS	TextToSpeech
GUI	Graphical user interface

Table of Contents

1. Introduction

1.1 Purpose of the Document

This document provides a comprehensive design specification for the Environment detection for blind people mobile application. It is intended for the development team, project supervisors, and technical reviewers who will evaluate the system architecture and implementation approach. This document will serve as a blueprint for the development process, outlining the system architecture, design strategies, and technical considerations.The system design follows the Object Oriented Design (OOD) methodology.

1.2 Project Overview

The Environment Detection System for Blind Users is an AI-powered application designed to assist visually impaired individuals by detecting objects using a smartphone camera and providing real-time audio feedback. The system processes video frames in real time to identify obstacles in the user's surroundings and communicates the detected information through spoken output using a Text-to-Speech (TTS) engine. This enables users to navigate their environment safely and with increased confidence.

1.3 Scope

Included Features

- Real-time object detection using a smartphone camera
- Image preprocessing techniques such as resizing, normalization, and noise reduction
- Audio feedback generation using aText-to-Speech (TTS) engine
- Direction identification of detected objects (Left, Center, Right)
- Continuous real-time system operation
- A simple, accessible, and user-friendly mobile interface.

Excluded Features

- Distance estimation between the user and detected objects
- Object size measurement
- Navigation or path planning capabilities
- Integration with IoT devices or external hardware
- Facial recognition or optical character recognition (OCR) features.

Design Considerartion

2.1 Assumptions and Dependencies

Assumptions

1. Hardware and Device Requirements

- **Device Capabilities:** The mobile device has adequate processing power, memory, and camera resolution to support real-time image processing and AI inference.
- **Storage Space:** Sufficient storage is available to install the application, AI model, and required libraries.
- **Battery Availability:** The device has sufficient battery capacity to support continuous camera usage and real-time processing.

2. Software Platform

- **Operating System:** The system operates on a supported mobile operating system (Android) compatible with TensorFlow Lite, OpenCV, and the Text-to-Speech (TTS) engine.

3. Image Acquisition

- **Image Quality:** Captured images are clear, properly focused, and taken under adequate lighting conditions.
- **Camera Orientation:** The camera is correctly oriented toward the environment to ensure accurate object detection and spatial estimation.

4. User Interaction and Audio Feedback

- **User Interaction:** The user can perform basic mobile phone operations and launch the application independently.
- **Audio Output:** Headphones or earphones are used to receive clear real-time audio feedback.

5. Operational Environment

- **Environmental Conditions:** The system is intended for normal indoor and outdoor environments extreme weather or low-visibility conditions may affect performance.
- **Real-Time Processing:** The system assumes real-time object detection with minimal latency on the target device.

6. AI Model Scope

- **Object Categories:** The AI model is trained to detect a predefined set of common objects relevant to user navigation and safety.

Dependencies

- **Machine Learning Framework:** The system relies on TensorFlow Lite (TFLite) for on-device object detection and real-time model inference.
- I**mage Processing Libraries:** OpenCV is utilized for preprocessing captured images, including resizing, normalization, and enhancement, to ensure reliable model input.
- **Text-to-Speech (TTS) Engine:** The TTS engine is required to convert detected objects into real-time audio feedback, providing auditory guidance to the user.
- **Mobile Platform Services:** A supported mobile operating system (e.g., Android) is essential to run the application and access device hardware such as the camera and audio output.
- **Dataset Availability:** Access to labeled datasets containing relevant objects (e.g., cars, people, animals) is necessary for training, testing, and validating the AI model.

2.2 Risk Assessment and Volatile Areas

2.2.1 Technical Risks

- **Model Accuracy Variability:** Object detection may be affected by poor lighting, motion blur, or occluded objects, causing incorrect audio feedback.
  Mitigation: Implement image quality checks, confidence scores, and user guidelines for proper camera handling.
- **Device Performance Limitations:** Real-time processing may be slow on older devices or cause high battery/memory usage.
  Mitigation: Optimize AI model using quantization/compression and test on multiple devices.
- **Hardware Dependencies:** System relies on functional camera, headphones, sufficient memory, and battery.
  Mitigation: Define minimum hardware requirements and recommend supported devices.
- **Integration and Stability Issues:** Combining TFLite, OpenCV, and TTS may introduce bugs or crashes.
  Mitigation: Conduct thorough integration testing and modularize components to isolate errors.

2.2.2 Requirement-Related Risks

- **Scope Expansion:** Requests for additional object categories or features may arise during development.
  Mitigation: Maintain clear scope documentation and implement modular architecture for future updates.

- **User Safety Reliance:** Incorrect or delayed guidance may compromise safety.
  Mitigation: Include disclaimers emphasizing the system as an assistive tool, not a replacement for mobility aids.

- **User Training & Accessibility:** Users may face difficulty operating the app or understanding audio feedback.
  Mitigation: Provide clear instructions, simple interface, and user testing feedback integration.

2.2.3 Volatile Areas (Subject to Change)

- **Supported Object Categories:** Initial system covers common objects (cars, people, animals); additional categories may be added later.

- **AI Model Updates:** Model may require periodic retraining with new datasets to maintain accuracy.

- **Interface & Audio Prompts:** UI/UX and audio guidance may evolve based on usability testing.

- **Environmental Adaptability:** Performance in low-light, adverse weather, or crowded environments may require optimization.

2.2.4 Contingency Measures

- Adopt modular design to allow feature addition and model updates without major rewrites.
- Implement version control for AI models to allow rollback if updates reduce accuracy.
- Maintain clear documentation of hardware/software dependencies and fallback strategies.

3. System Architecture

This section presents a high-level architectural overview of the Environment Detection System for Blind Users. The proposed system adopts a modular and layered architecture, in which each component is responsible for a specific function, such as image acquisition, object detection, decision processing, and audio feedback. This design approach enhances maintainability, scalability, and performance while ensuring the system remains lightweight and suitable for smartphone deployment.

3.1 System-Level Architect

The system-level architecture decomposes the Environment Detection System into the following major subsystems:

3.1.1 User Interaction Subsystem

This System allows the user to interact between system and application, ensuring ease of use for visually impaired individuals.

- Maintain a user accessible interface
- Start/stop detection
- Control audio output setting

3.1.2 Image Acquisition Subsystem

This subsystem is responsible for capturing real-time visual data from the smart-phone camera. Continuous video input is obtained and divided into individual frames.These serve as the raw input for further processing.

Responsibilities:

- Access smartphone camera hardware
- Capture live video stream
- Extract frames at an appropriate frame rate.

3.1.3 Pre-Processing Subsystem

Before object detection, the captured frames are pre-processed to improve detection efficiency and reduce computational overhead.

Responsibilities:

- Resize frames to model-compatible dimensions
- Convert RGB images to grayscale or optimized formats
- Normalize noise and improve contrast if required

3.1.4 Object Detection Subsystem

This is the core component of the system. It uses a custom-trained deep learning model to detect and classify objects present in the environment in real time.

Responsibilities:

- Apply machine learning object detection model
- Identify objects such as people, vehicles, furniture, doors, etc.
- Generate confidence scores and bounding boxes
- Classify detected objects using trained datasets
- Classify the object direction

3.1.5 Decision & Interpretation Subsystem

This subsystem interprets detection results and decides which information should be conveyed to the user.It filters detections based on relevance and confidence to avoid overwhelming the user with excessive audio feedback.

Responsibilities:

- Filter low-confidence detections
- Prioritize critical obstacles
- Convert detection results into meaningful textual descriptions

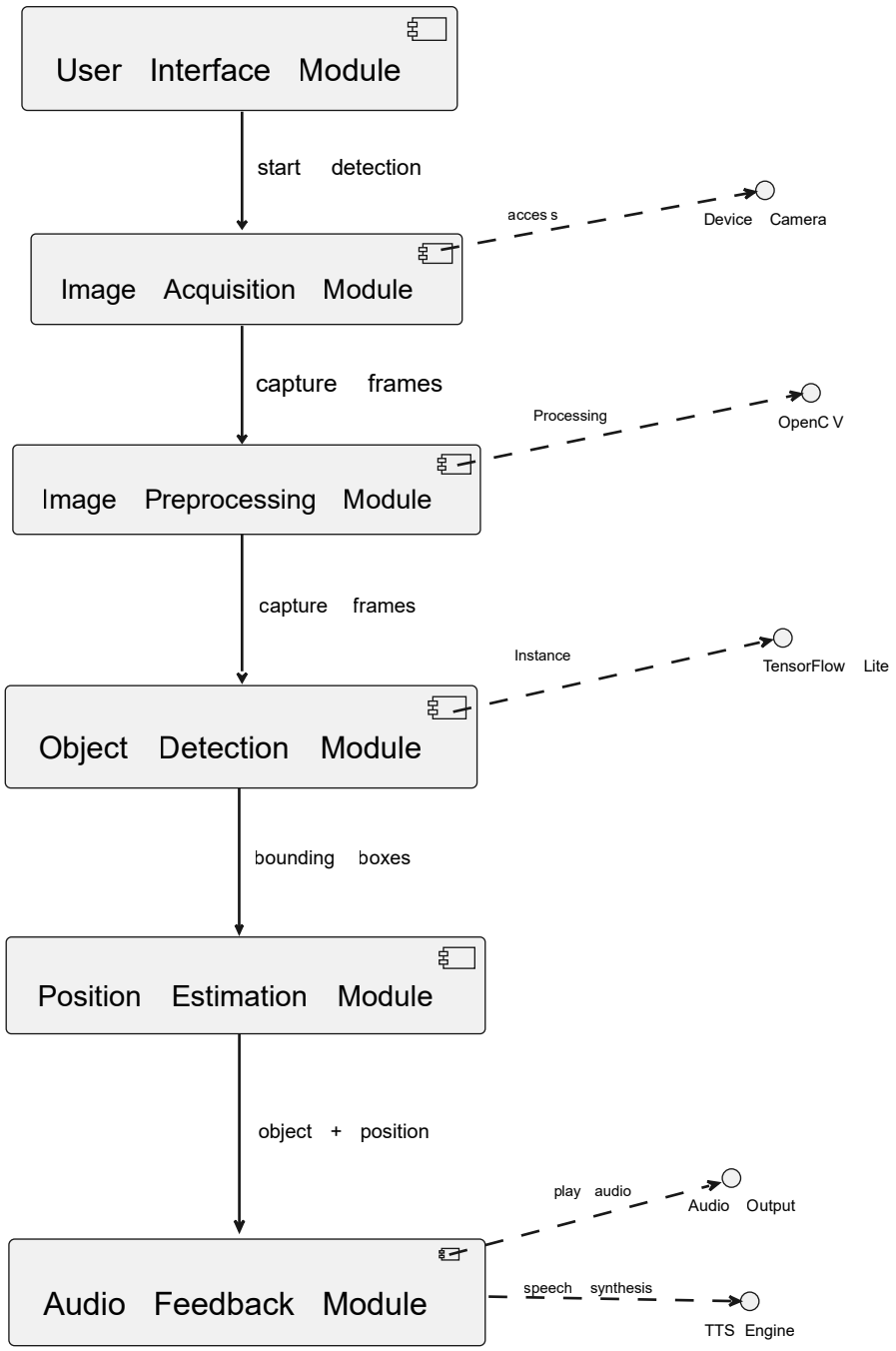
3.1.6 Audio Feedback Subsystem

This system converts textual information into audible instructions using a Text-to-Speech (TTS) engine, enabling blind users to receive environmental awareness through sound.

Responsibilities:

- Convert detected object labels into speech
- Deliver audio output via smartphone speaker or headphone
- Ensure clarity, minimal delay, and user friendly speech output

3.2 Sub-System / Component / Module Level Architectures



3.3 Sub-Component / Sub-Module Level Architecture (1…n)

3.3.1 User Interface Module

This is the human-facing layer. It shows what the camera sees, lets the user interact, and displays results and system state.

1. Camera View

- Displays live camera preview
- Gives real-time visual context to the user
- Does not process images, only renders frames

Typical responsibilities

- Start/stop preview
- Maintain aspect ratio
- Overlay support (bounding boxes)

2. Control Panel

- **User interaction hub**(Starts detection and controls behavior)

3. Result Overlay

- Draws bounding boxes
- labeled Object
- confidence scores
- Direction

4. Status Display

Communicates system state

- “Processing…”
- “Camera permission denied”
- “Model loading failed”

3.3.2 Image Acquisition Module

Gets raw image data from the device camera and prepares it for processing

1. Camera Handler

- Handles camera permissions
- Initializes camera hardware
- Manages lifecycle (open / close / pause)

Failure handled

- Permission denied
- Camera unavailable

2. Frame Capturer

Captures frames from camera stream

- Single frame capture
- Continuous (burst) capture
- Controls frame rate to save battery

3. Image Buffer

- Temporary memory holding captured frames
- Decouples camera speed from ML speed.

4. Format Converter

Converts camera output format YUV (common on Android) to RGB (ML-friendly)

3.3.3 Image Preprocessing Module

Transforms raw images into a model-ready tensor.

1. Image Resizer

Scales image to model input size(e.g 224x224, 300x300)

2. Normalizer

Adjusts pixel values (0–255 to 0–1or -1 → +1) this will improve model accuracy.

3. Color Converter

Ensures correct channel order (RGB to BGR) because Some models are trained in BGR (e.g., OpenCV-based)

4. Tensor Builder

Converts processed image into TensorFlow Lite tensor which is Core ML multi-array
This is The final bridge between image and ML model

3.3.4 Object Detection Module

Runs the ML model and extracts meaningful results from a image

1. Model Loader

Loads ML model into memory and Initializes inference enginefor detection

2. Inference Runner

This will Executes the model and Produces raw predictionsfrom the image.

3. **Result Parser**
   Converts raw outputs into:
   Bounding boxes
   Class labels
   Confidence scores
   Direction of object

4. **Confidence Filter**
   Removes detections below threshold (< 0.5) to reduce inaccuracy issue

3.3.5 Position Estimation Module

Converts detections into human-understandable spatial info.

1. **Coordinate Mapper**
   Converts image coordinates → screen coordinates and handles Aspect ratio differences Rotation and Mirroring

2. **Zone Calculator**
   This will Determines object location (Left, Center, Right)

3. **Direction Formatter**
   This will Produces natural phrases ("Person on the left", "Car in front", "Bottle on the right side")

3.3.6 Audio Feedback Module

Converts detection results into spoken feedback.

1. **Text Formatter**
   Generates speakable sentences and Deduplicates repeated detections (Avoid repeating "person ahead" every frame)

2. **TTS Engine**
   This will Converts text into speech that are easy to understandable for blind user

3. **Audio Player**
   This will Plays audio output automatically

4. **Queue Manager**
   This will ensures audio is spoken in order and Prevents overlapping speech.

3.3.7 Core Coordination

Controls the system and keeps it stable and efficient.

1. Pipeline Controller

Orchestrates the full flow (Capture → Preprocess → Detect → Estimate → Speak) and also controls timing Enables / disables modules.

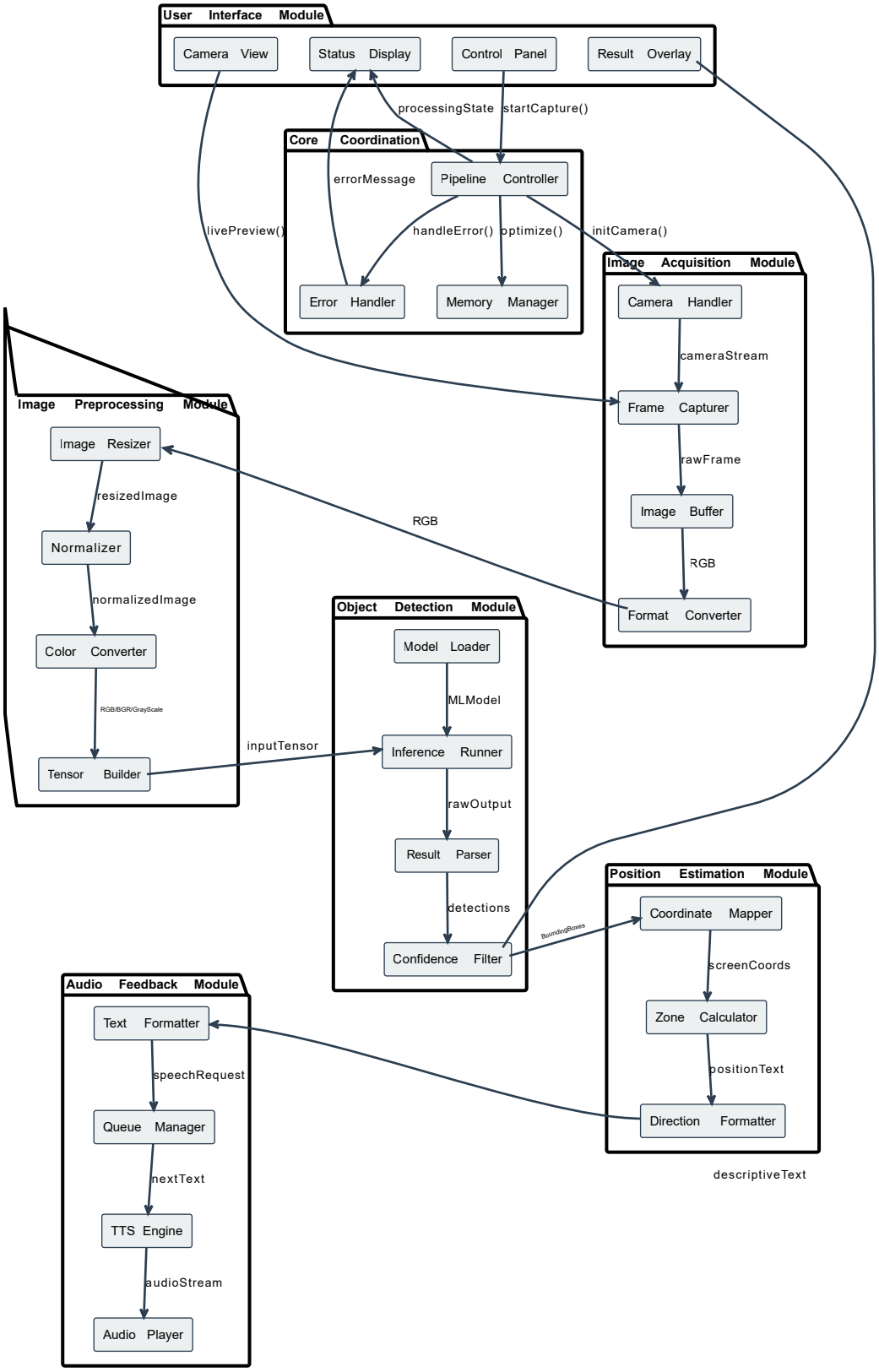
2. Error Handler

Centralized error management (e.g Camera failure, Model load error, Inference crash)

3. Memory Manager

This will Frees unused buffers and keep space for next frame (image)

DIiagrame detailed sub-component



4. Design Strategies

The design strategies for the Environment Detection System for Blind Users aim for real- time performance, usability for visually impaired users, scalability, and maintainability while working within the limits of mobile hardware. These strategies directly impact the system's overall structure, component interaction, and design framework. The system uses a modular, layered, and object-oriented design approach. This allows for independent development, testing and improvement of components like image acquisition, object detection, decision-making, and audio feedback. All choices focus on the main goals of accessibility, low latency, portability, and cost-effectiveness..

4.1 Strategy 1…n

Strategy 1: Modular and Layered Architectural Design

The proposed system uses a modular and layered architectural approach. Its overall functionality is split into distinct modules: Image Acquisition, Preprocessing, Object Detection, Decision and Interpretation, Audio Feedback, and User Interaction. Each module handles a specific task and interacts with others through clearly defined interfaces.

Description:

The system assesses detected objects based on confidence levels and relevance before giving audio feedback. Only important and critical information is shared with the user.

Reasoning:

Too many audio notifications can overwhelm users and diminish system effectiveness. Filtering ensures users receive clear and practical information, especially about immediate obstacles.

Trade-offs:

Some low-priority or non-essential objects may not be announced, and confidence thresholds need careful adjustment.

Impact Areas:

- Information handling
- User experience
- Operational safety

Strategy 2: Smartphone-Based and Hardware-Independent Design

Description:

The system is designed to work entirely on a standard smartphone. It uses the built-in camera, processing unit, and audio output. No extra hardware or IoT sensors are needed for the system to operate.

Reasoning:

This choice cuts overall system costs and makes it more portable and easier to set up. It also fits within the project scope and helps visually impaired users who already have smartphones.

Trade-offs:

The performance of the system depends on the smartphone's hardware. Without extra sensors, there is limited access to additional environmental data.

Impact Areas:

- System reuse across devices
- Scalable deployment
- Cost-effectiveness

Strategy 3: Real-Time and Low-Latency Processing Design

Description:

To achieve real-time performance, the system uses a lightweight object detection model, optimized frame processing techniques, and minimal preprocessing operations.

Reasoning:

Timely detection and feedback are crucial for the safety of visually impaired users.YOLO(CV model) strikes a good balance between detection accuracy and processing speed, which makes it a good fit for mobile real-time applications.

Trade-offs:

In challenging environmental conditions, detection accuracy may drop slightly, and continuous processing might increase the load on the device’s CPU or GPU.

Impact Areas:

- Concurrent execution
- Performance optimization
- User safety

Strategy 4 : Selective and Intelligent Decision Filtering

Description:

The system assesses detected objects based on confidence levels and relevance before giving audio feedback. Only important and critical information is shared with the user.

Reasoning:

Too many audio notifications can overwhelm users and diminish system effectiveness.Filtering ensures users receive clear and practical information, especially about immediate obstacles.

Trade-offs:

Some low-priority or non-essential objects may not be announced, and confidence thres-holds need careful adjustment.

Strategy 5 : Offline-Capable Audio Feedback Mechanism (Optional)

Description:

Text-to-Speech functionality uses offline-capable engines to maintain system operation no matter the network availability.

Reasoning:

Relying on internet connectivity can lower system reliability, especially outdoors. Offline TTS reduces delays and ensures consistent audio feedback.

Trade-offs:

Offline speech engines might provide slightly lower voice quality and fewer language options compared to cloud-based services.

Impact Areas:

- Data handling
- System reliability
- Real-time response

Strategy 6: Minimal Data Storage and Privacy-Focused Design

Description:

The system processes visual data in real time and avoids storing images or videos unless necessary for testing or debugging. Most data stays in temporary memory.

Reasoning:

This method protects user privacy, reduces storage needs, and follows ethical software design practices, especially for assistive technologies.

Trade-offs:

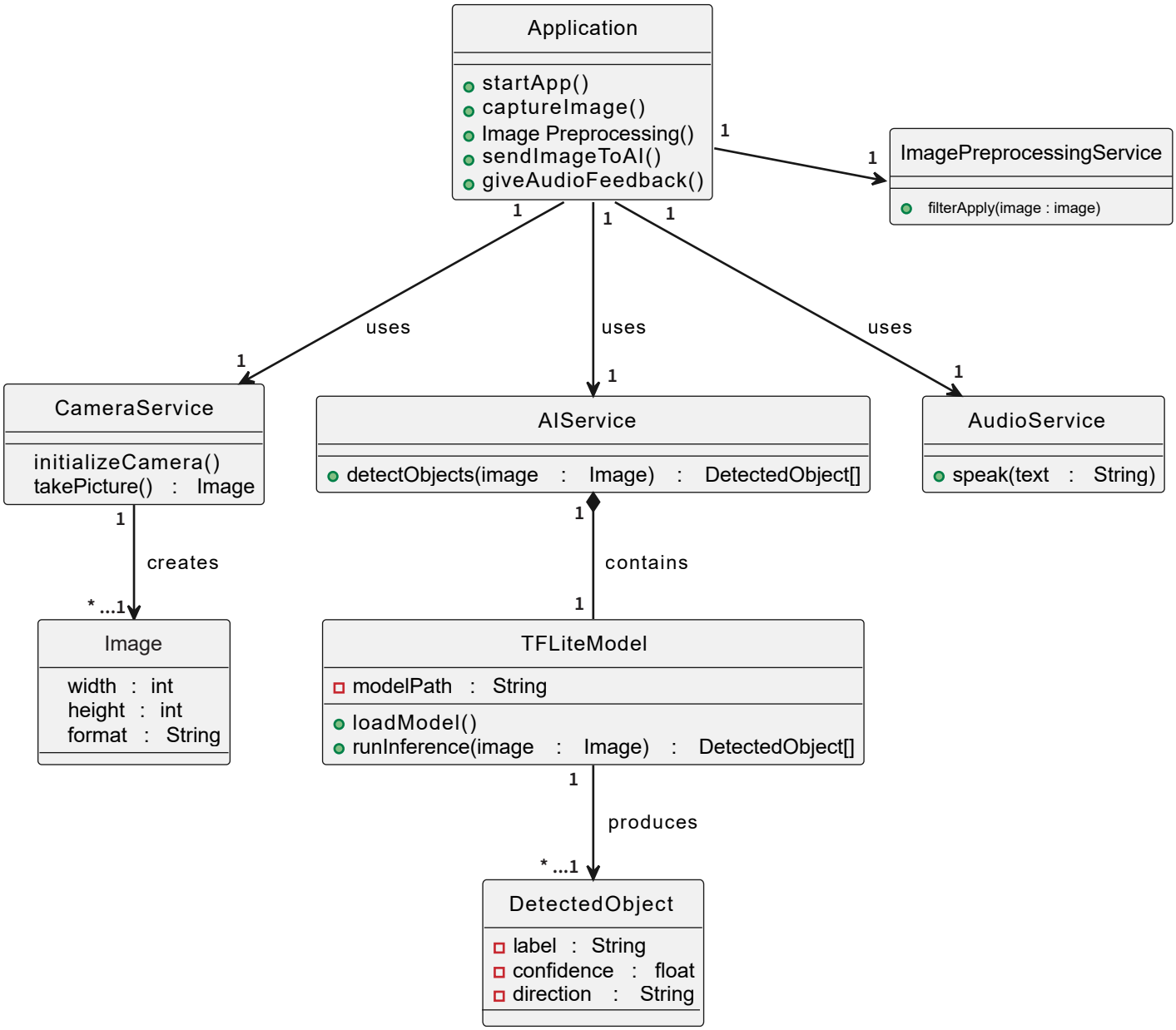
Limited historical data is available for analysis, and debugging might need manual logging.

Impact Areas:

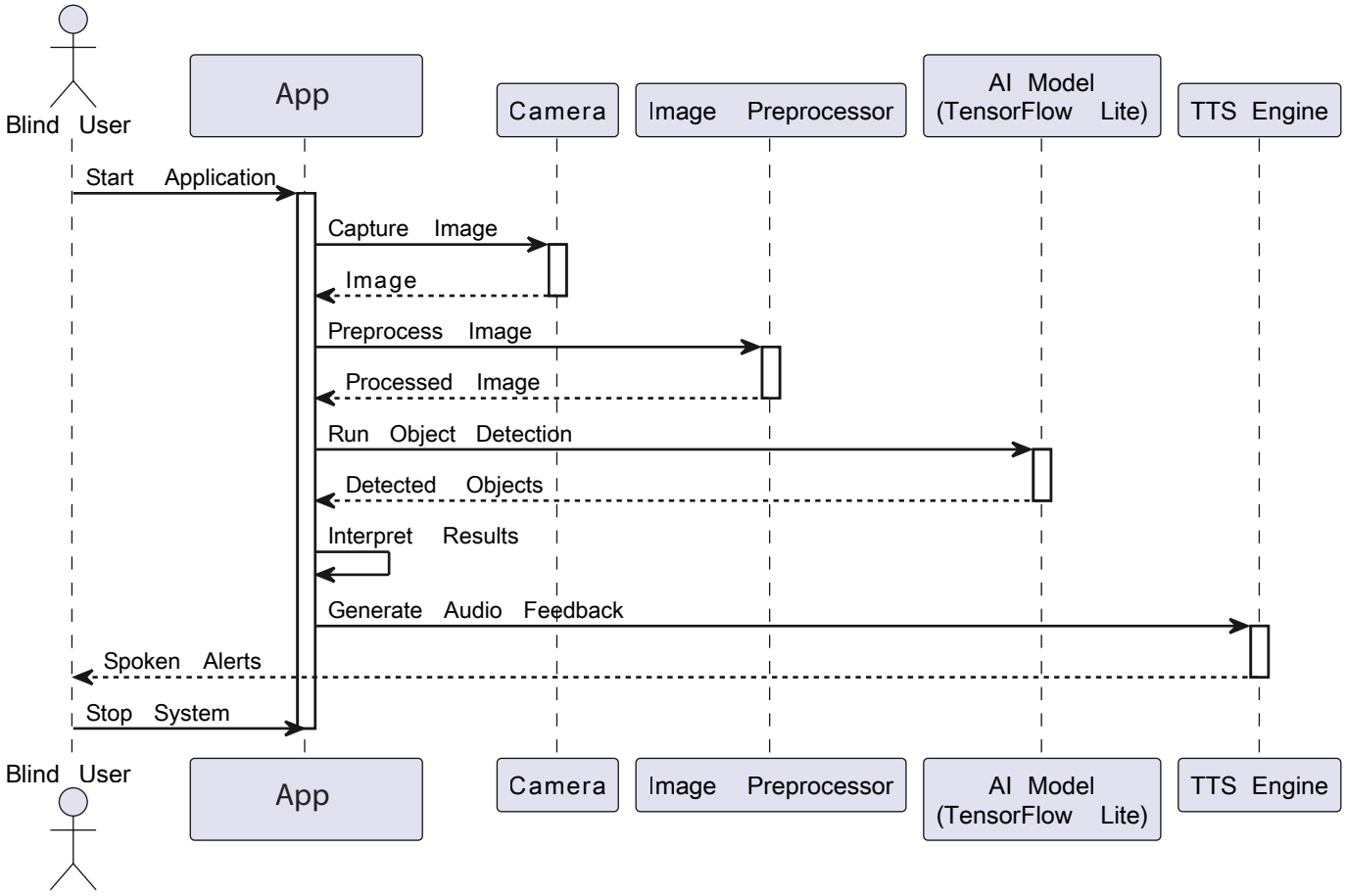
- Data management
- Privacy and security
- Performance efficiency

Detailed System Design

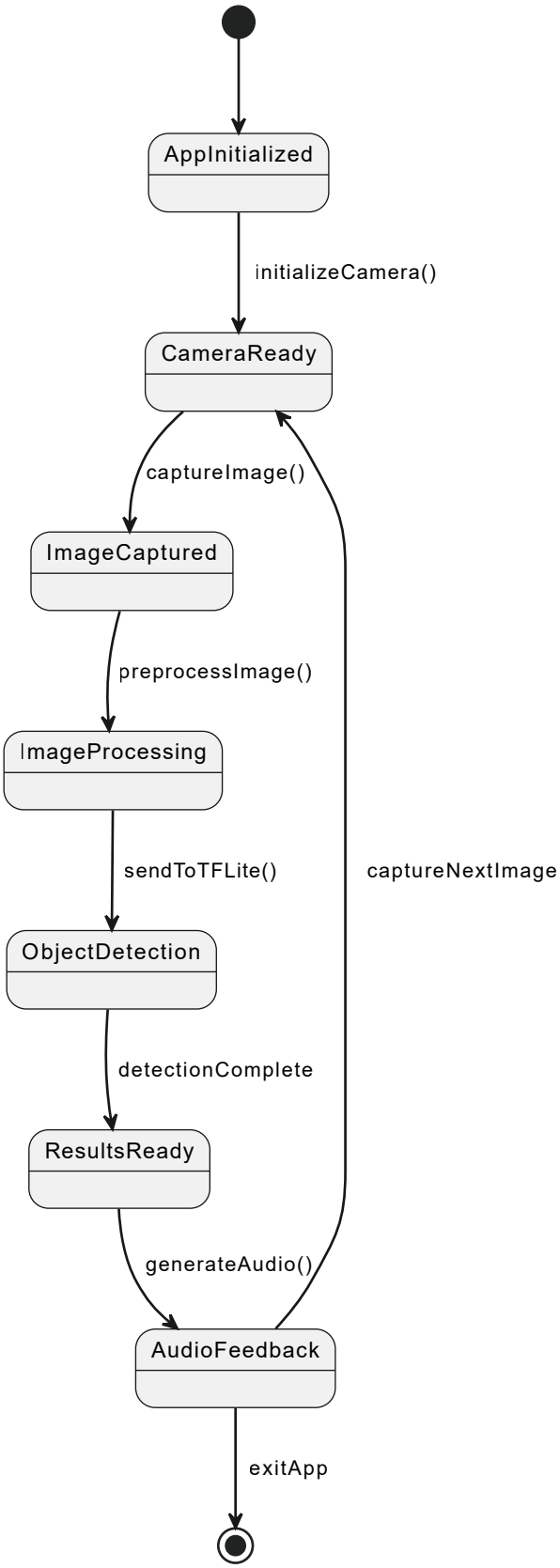
Class Diagrame



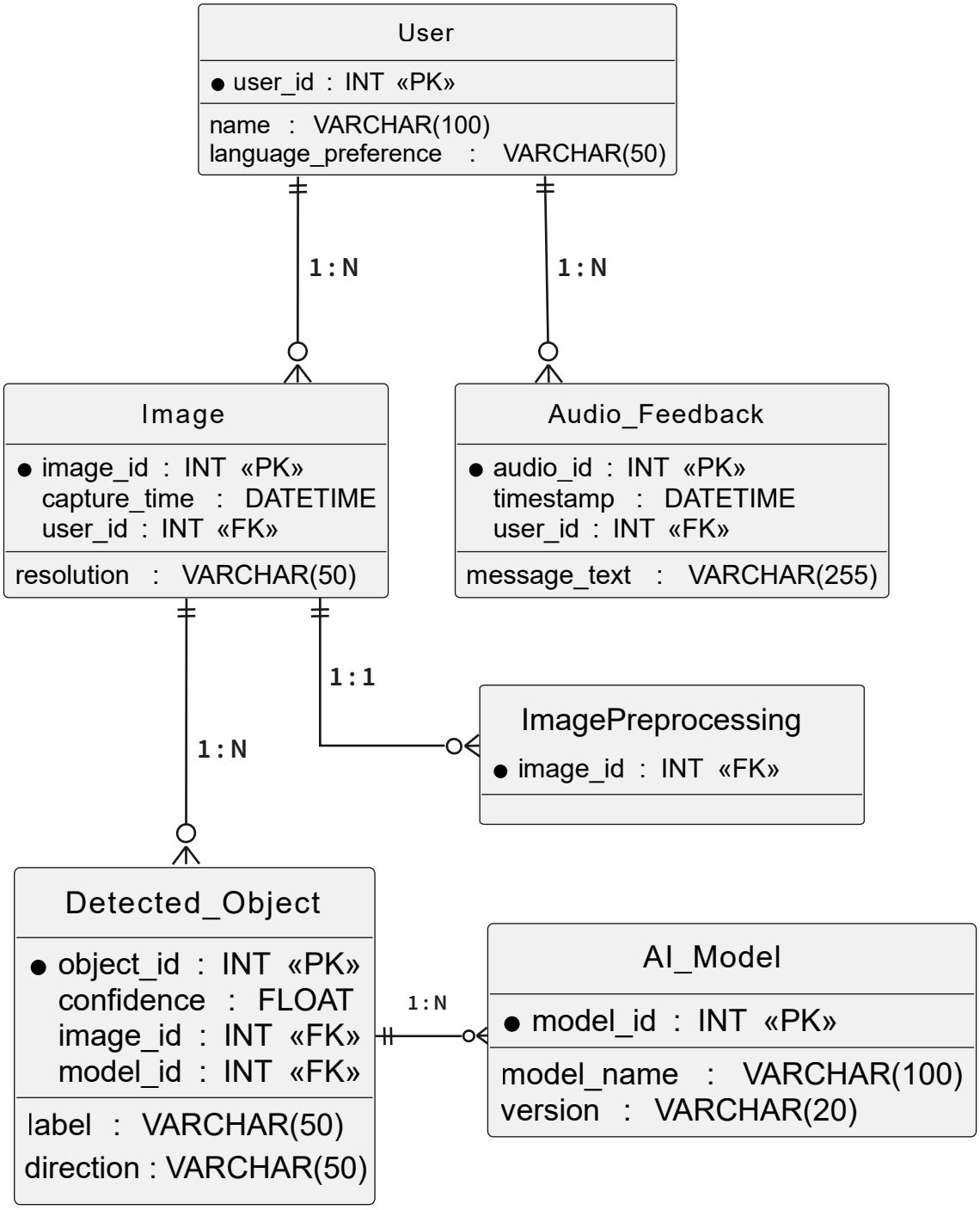
Sequence diagram



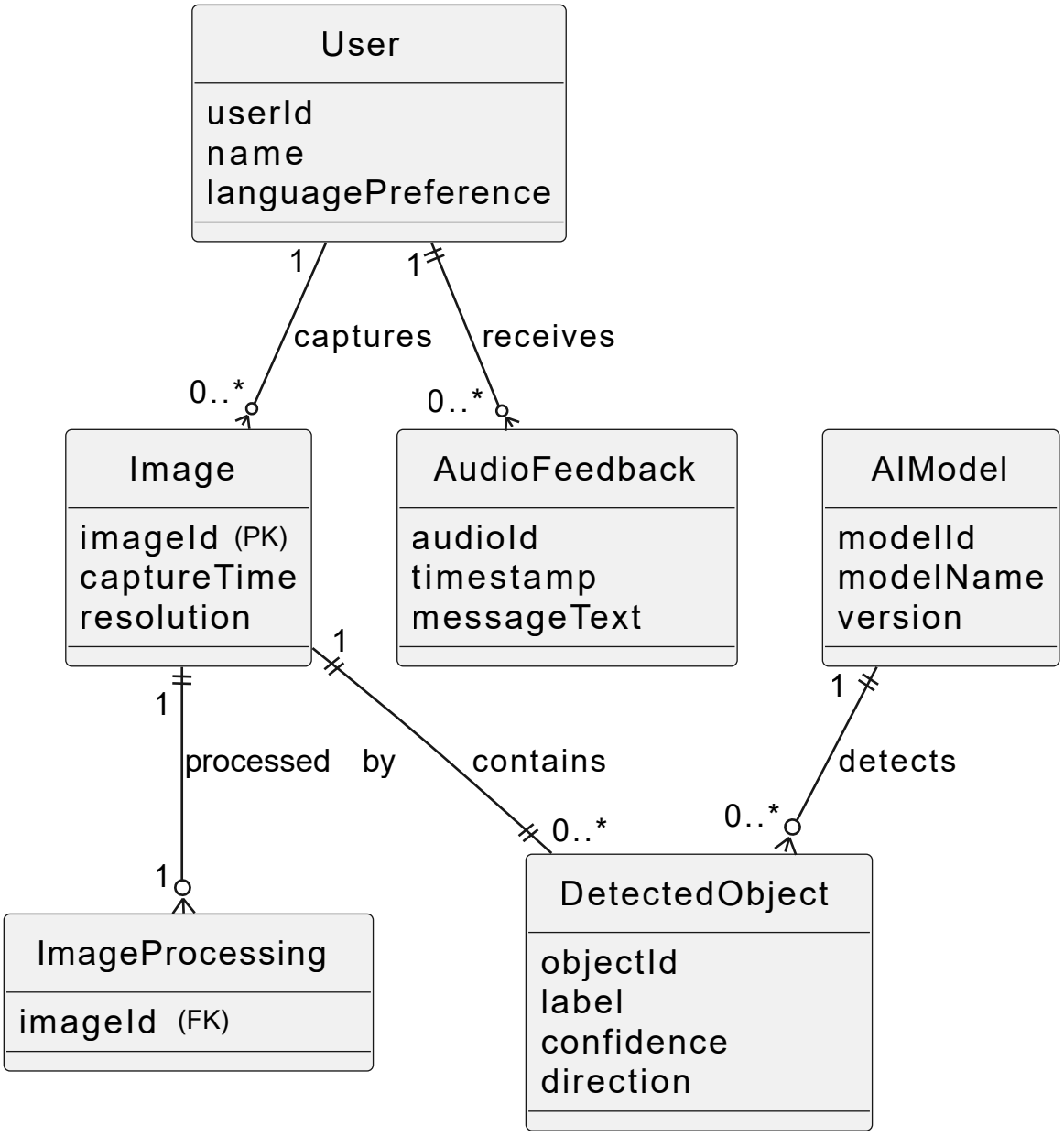
State Transition diagram



Physical data Model



Logical Model(E/R Model)



Detailed GUI

GUI features:

- Display a live camera preview
- Detected objects using bounding boxes
- Shows the detected object name(label) clearly
- Dispalys confidence percentage
- Provides feeback without user interaction

5. References

Ref. No	Document Title	Date of release	Doc Source
PGBH01 2025-Proposal	Proposal	Oct 20, 2025	https://github.com/ali-raza-91/FYP/tree/main/CP-1/Project_Docs/Proposal
PGBH01 2025-SRS	Software Requirement Specification	Oct 20, 2025	https://github.com/ali-raza-91/FYP/tree/main/CP-1/Project_Docs/SRS
PGBH01 2025-[1[	google lookout	Mar 13, 2019	https://support.google.com/accessibility/android/answer/9031274?hl=en
PGBH01 2025-[2]	Be my eyes	Oct, 2017	https://www.bemyeyes.com/download-app/