30.05.2025

TED UNIVERSITY

CMPE 491-0 SENIOR PROJECT

High-Level Design Report
VAVI – Voice Assistant for Visually Impaired

Team Members:

Ceyda Kuşçuoğlu(16348076072)

Kıvanç Terzioğlu(27233564574)

Berkay Kaan Karaca(68317070956)

Table Of Content:

1. Introduction

1.1 Purpose of the System

The purpose of the VAVI system is to enhance the daily life and mobility of visually impaired individuals by detecting people and obstacles through a smartphone camera, processing the images with a local server using deep learning, and providing intuitive auditory feedback to help users navigate their environment safely.

1.2 Design Goals

- Real-Time Feedback: Deliver minimal-latency feedback from image capture to audio output.
- High Detection Accuracy: Reliably detect people and potential obstacles in various environments.
- User-Centric Simplicity: Ensure the solution is easy to use, requiring minimal setup or interaction.
- Portability: Leverage the user's smartphone for both imaging and audio feedback, with no additional wearable hardware.
- Security and Privacy: Design for future encrypted data transfer and privacy controls.
- Expandability: Prepare for the addition of new features such as mapping, scene description, and offline/online fallback modes.
- Robustness: Plan for handling connectivity and hardware issues as the system evolves.

1.3 Definitions, Acronyms, and Abbreviations

- VAVI: Voice Assistant for Visually Impaired
- YOLO: You Only Look Once (object detection algorithm)
- Edge Device: In this prototype, the user's smartphone (camera + speaker)
- Server: Local computer processing images via Wi-Fi
- Feedback: Auditory output (directional speech and beeps)
- Prototype: Early working version under development
- Mapping: Planned feature to assist with spatial orientation

1.4 Overview

VAVI currently consists of a user's smartphone that captures video, sends frames over Wi-Fi to a local server, where a YOLO-based deep learning model detects people and obstacles. The server then returns auditory feedback (beeps or speech such as "Obstacle on your left") to the phone, which plays it through the phone's speaker. A dedicated mobile app is planned but not yet implemented. The project is in the prototype stage and rapidly evolving.

---

2. Current Software Architecture

Currently, the architecture uses a smartphone for image capture and playback. Images are sent over Wi-Fi to a local server running the YOLO model, which processes them and returns feedback. There is no mobile app yet; the communication uses simple network scripts. Audio feedback is delivered via the phone speaker. No user data is stored, and encryption is not yet implemented.

---

3. Proposed Software Architecture

3.1 Overview

The planned architecture maintains a client-server model: the phone (client) captures images, sends them to the local server (server), which analyzes and returns feedback. The system will soon include a dedicated mobile app for easier user interaction and settings management.

3.2 Subsystem Decomposition

- Client Subsystem (Smartphone):
    - Camera input
    - Data transmission to server
    - Audio playback
    - (Planned) Mobile application for user interface and settings
- Server Subsystem (Local PC):
    - Image reception
    - YOLO-based person and obstacle detection
    - Audio feedback generation (directional beeps or speech)
    - (Planned) Enhanced features: mapping, privacy controls
- Communication Subsystem:
    - Wi-Fi-based image and data transfer
    - (Planned) Encryption and offline fallback

3.3 Hardware/Software Mapping

- Hardware:
    - Smartphone (camera and speaker)
    - Local server/computer (runs YOLO model, processes images)
- Software:
    - Client: (For now) Python scripts; (Planned) Mobile app (Android/iOS)
    - Server: Python with YOLO (deep learning), socket/network scripts

UML Deployment Diagram

The architecture of the VAVI system is illustrated in the deployment diagram below. This diagram shows the interaction and responsibilities of each hardware and software component:

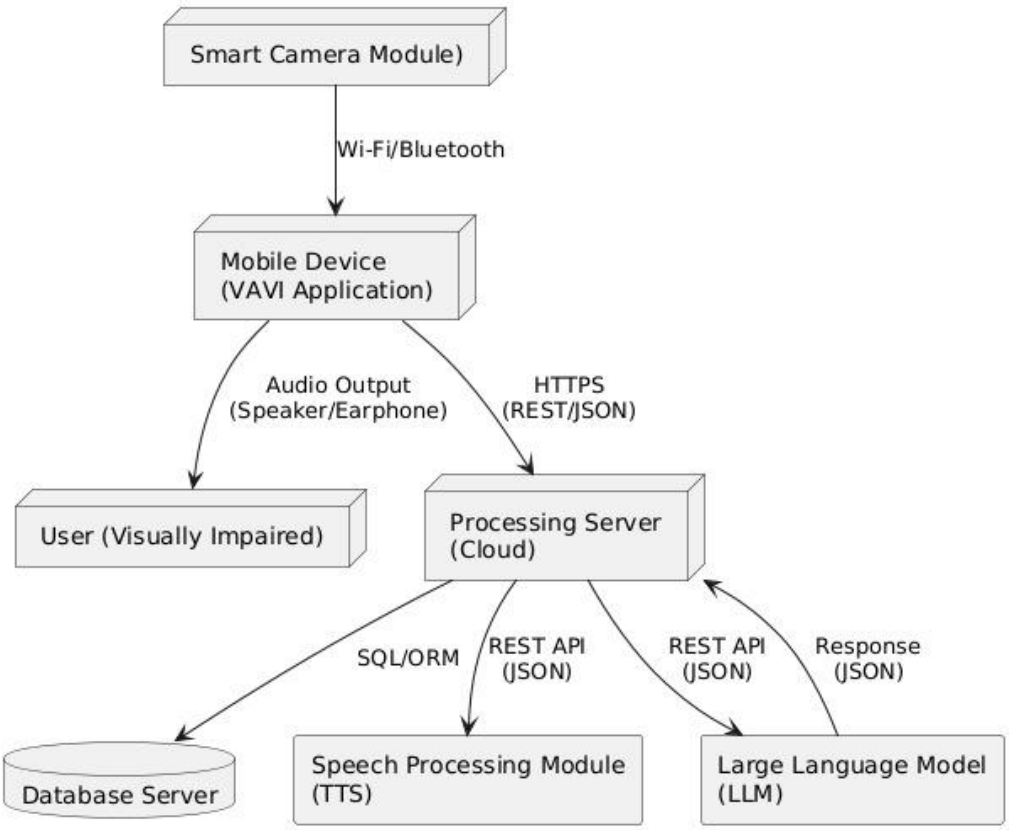

Figure 1. Hardware/Software Deployment Diagram for VAVI

Explanation:

- Smart Camera Module captures environmental images and transmits them to the Mobile Device via Wi-Fi or Bluetooth.
- The Mobile Device (VAVI Application) processes incoming data and delivers auditory feedback (via speaker/earphone) to the user. It also securely sends data to the Processing Server (Cloud) using HTTPS REST APIs.
- The Processing Server processes the incoming data, interacts with a Database Server for data management, uses a Speech Processing Module (TTS) for text-to-speech output, and can interface with a Large Language Model (LLM) for complex responses.

This structure ensures scalability, modularity, and enables integration with advanced features such as speech processing and large language models in future versions.

Activity Diagram

The following activity diagram illustrates the end-to-end process in the VAVI system, from user activation to audio feedback delivery. It demonstrates the collaboration between the user, smart camera module, mobile device, processing server, language model, and text-to-speech module.

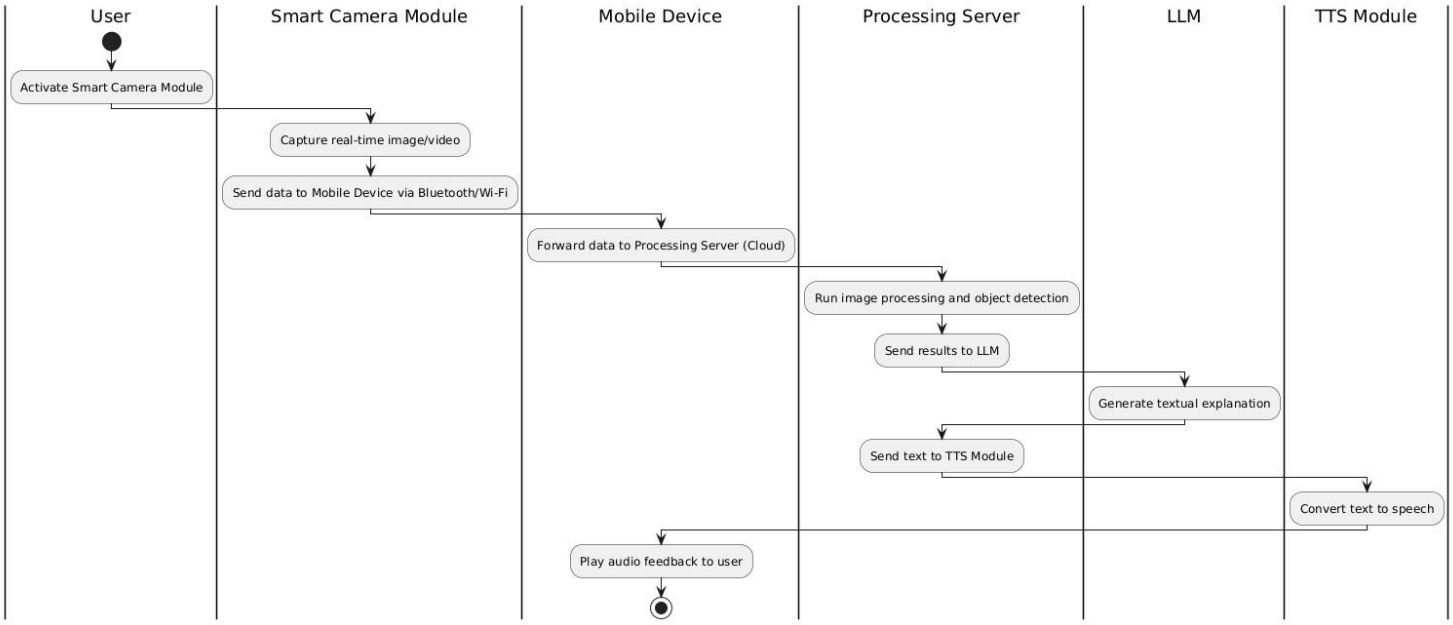

Figure 2. Activity Diagram for VAVI System Flow

Explanation:

- The user activates the smart camera module.
- The camera captures real-time images or video and transmits them to the mobile device via Bluetooth or Wi-Fi.
- The mobile device forwards the data to the processing server (cloud).
- The processing server performs image analysis and object detection, then sends results (if needed) to the LLM for generating a textual explanation.
- The generated explanation is sent to the TTS module to convert text to speech.
- The mobile device plays the resulting audio feedback to the user.

This activity diagram provides a clear overview of how VAVI processes information and interacts with each component to assist visually impaired users.

Component Diagram

The following component diagram illustrates the main software modules and their interactions within the VAVI system. It visualizes how data flows between the client side (user interface, image capture, communication modules), the cloud server (processing, object detection, and connectors for AI services), and the external services (database, TTS, LLM).

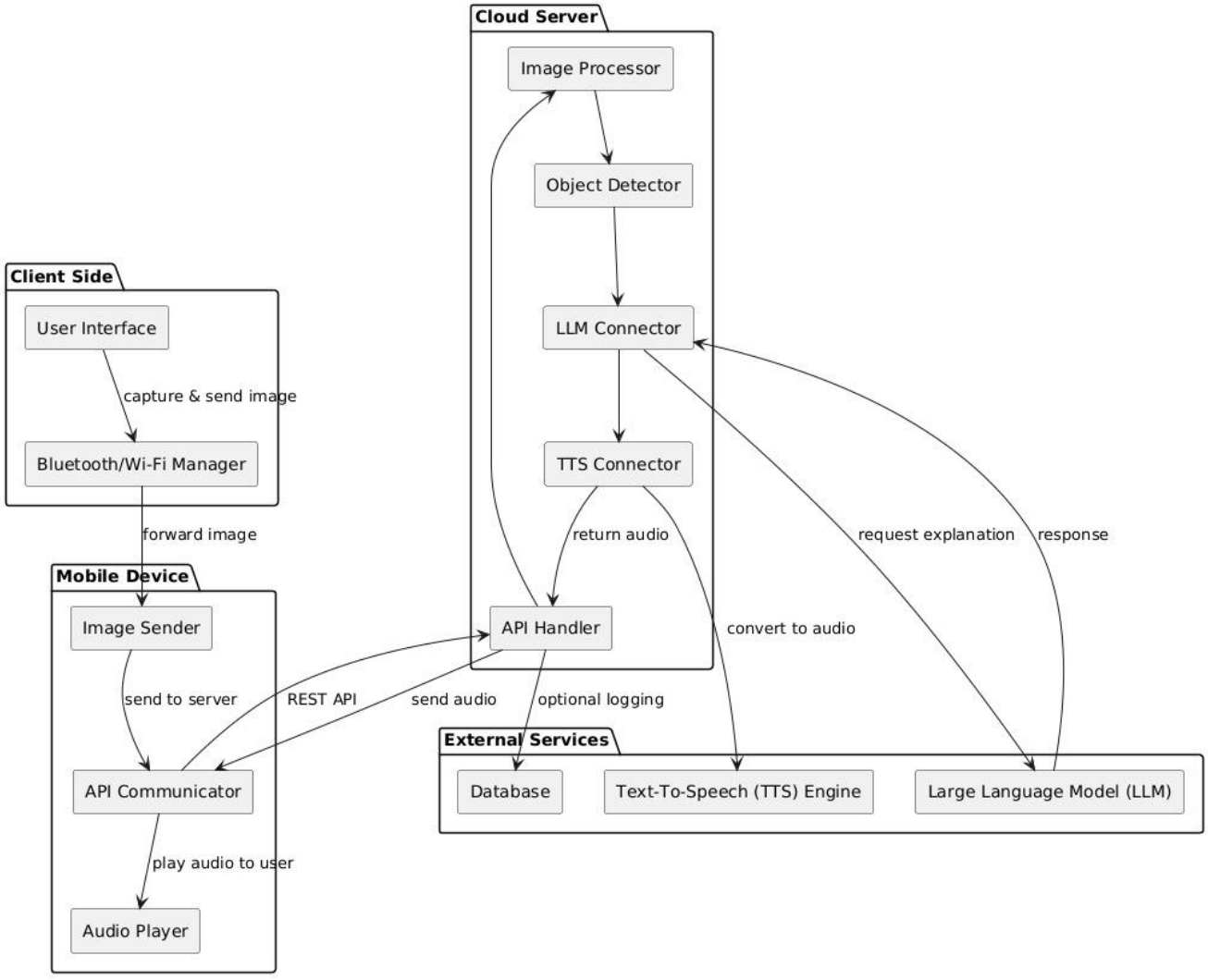

Figure 3. Component Diagram for VAVI System

Explanation:

- Client Side: Consists of the user interface and Bluetooth/Wi-Fi manager responsible for capturing and sending images.
- Mobile Device: Sends images to the server and plays audio feedback to the user.
- Cloud Server: Runs core processing modules including image processing, object detection, LLM and TTS connectors, and an API handler.
- External Services: Include the database, TTS engine, and large language model, all of which are accessed by the cloud server for various tasks.

Subsystem Diagram

The subsystem diagram below illustrates the major high-level components of the VAVI system and the main data/control flows between them.

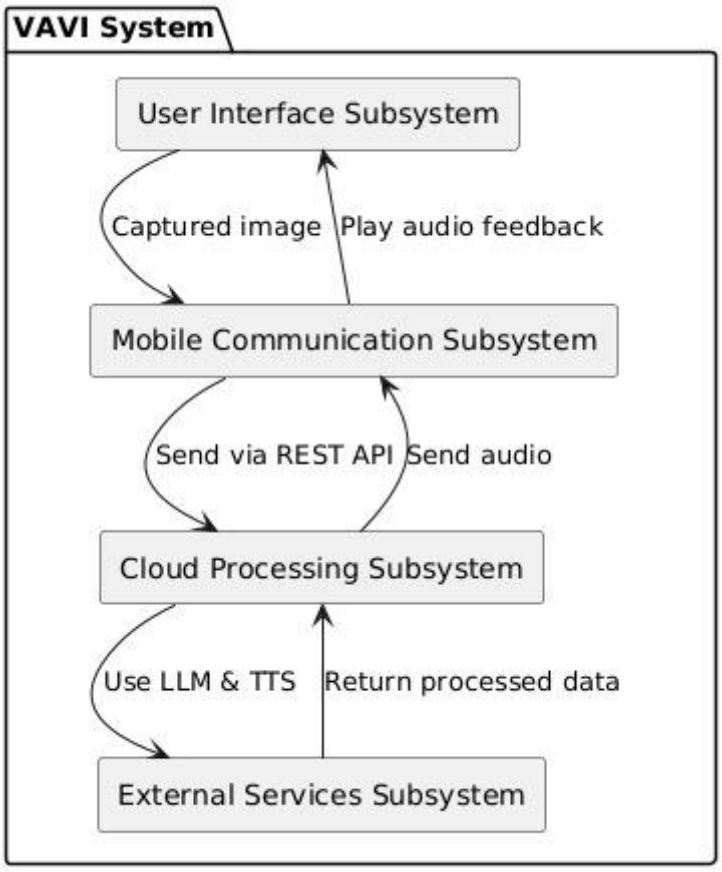

Figure 4. Subsystem Diagram for VAVI System

Explanation:

- The User Interface Subsystem manages image capture and audio feedback to the user.
- The Mobile Communication Subsystem transmits images and receives audio via REST APIs.
- The Cloud Processing Subsystem is responsible for running advanced AI models (LLM, TTS) and returning processed results.
- The External Services Subsystem provides specialized resources, such as language models and speech engines, for the cloud processing unit.

This structure demonstrates the modular and extensible design of the VAVI system, supporting easy integration of additional services or functionalities.

3.4 Persistent Data Management

- No user data (images, audio, logs) is stored at this stage.
- All processing is done in real time; privacy by design.
- (Planned) In future, options for secure, consent-based diagnostics or performance logs.

3.5 Access Control and Security

- Currently, there is no encryption or authentication; direct local Wi-Fi communication is used.
- (Planned) Future versions will implement encrypted data transfer (e.g., HTTPS) and device authentication.
- No personally identifiable information is collected or stored.

3.6 Global Software Control

- Event-driven operation:
  - Phone captures and sends images → Server processes → Sends back audio feedback
  - Feedback is immediately played via phone speaker
- Planned mobile app will provide better user flow and error notifications.

3.7 Boundary Conditions

- Connectivity Loss: System currently becomes non-functional; fallback/offline mode is planned.
- Low Battery: Not yet implemented, but future versions will include battery and hardware alerts.
- Camera Blocked/Unavailable: Planned feature to detect and notify the user.
- Server Overload/Error: Minimal error handling; planned for robust queuing and user alerts.

---

4. Subsystem Services

Subsystem	Services Provided
Client/Smartphone	Image capture, transmission, audio playback
Server/Local PC	Image processing (YOLO), person/obstacle detection, feedback generation
Communication	Wi-Fi data transfer (currently unencrypted)

5. Glossary

- YOLO: Deep learning-based, real-time object detection algorithm.
- Edge Device: User's smartphone acting as camera and audio device.
- Feedback: Auditory information indicating obstacle position (e.g., "left", "right", beep frequency).
- Prototype: First working version, not feature-complete.
- Mapping: (Planned) Advanced feature for spatial orientation assistance.
- Encryption: Securing data transfer to prevent interception (planned).
- Offline Mode: Functionality when no network is available (planned).

6. References

1. Bruegge, B., & Dutoit, A. H. (2004). *Object-Oriented Software Engineering, Using UML, Patterns, and Java* (2nd ed.). Prentice-Hall.
2. Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. arXiv:1804.02767.
3. Python official documentation.
4. PyTorch/TensorFlow documentation (YOLO ve deep learning).
5. Android Developers