TED UNIVERSITY
Faculty of Engineering
Department of Computer Engineering

Cmpe 492 – VAVI Test Plan Report

Team Members:
Ceyda Kuşçuoğlu - 16348076072
Kıvanç Terzioğlu - 27233564574
Berkay Kaan Karaca - 68317070956

1. Introduction

This Test Plan outlines the testing strategy for VAVI – Voice Assistant for Visually Impaired, a Flutter-based mobile indoor navigation and object detection system developed for visually impaired users. The system integrates Wi-Fi fingerprinting, IMU-based motion estimation, YOLO-based object detection, directional audio feedback, and a FastAPI backend to provide accurate real-time navigation and obstacle awareness inside A Block of the campus.

The objective of this document is to define the testing scope, approach, responsibilities, environment setup, risks, and the detailed test cases necessary to validate the functional correctness, performance, usability, and reliability of VAVI.

2. Scope of Testing

2.1 In-Scope Features

The following features will be included in the testing activities:

- Wi-Fi fingerprint scanning
- IMU sensor reading (accelerometer + gyroscope)
- Camera-based object detection using YOLO (TFLite)
- Directional audio feedback (stereo beep intensity + panning)
- Navigation engine & pathfinding (Dijkstra/A*)
- Accessibility-focused user interface
- REST API communication (Python FastAPI)
- Fusion pipeline (Wi-Fi + IMU + camera integration)

2.2 Out of Scope

- Testing in buildings other than A Block
- iOS platform testing
- Long-duration stress testing (≥12 hours)
- Multi-user concurrent server load testing

3. Test Objectives

The goals of the testing activities include:

- Ensuring indoor navigation error remains within ±2 meters.
- Validating real-time YOLO object detection accuracy and speed.
- Confirming correct integration of Wi-Fi, IMU, and fusion model outputs.
- Testing the reliability of server communication and local fallback systems.
- Evaluating the clarity, timing, and accuracy of audio feedback.
- Ensuring the system is safe and intuitive for visually impaired users.

4. Testing Methodology

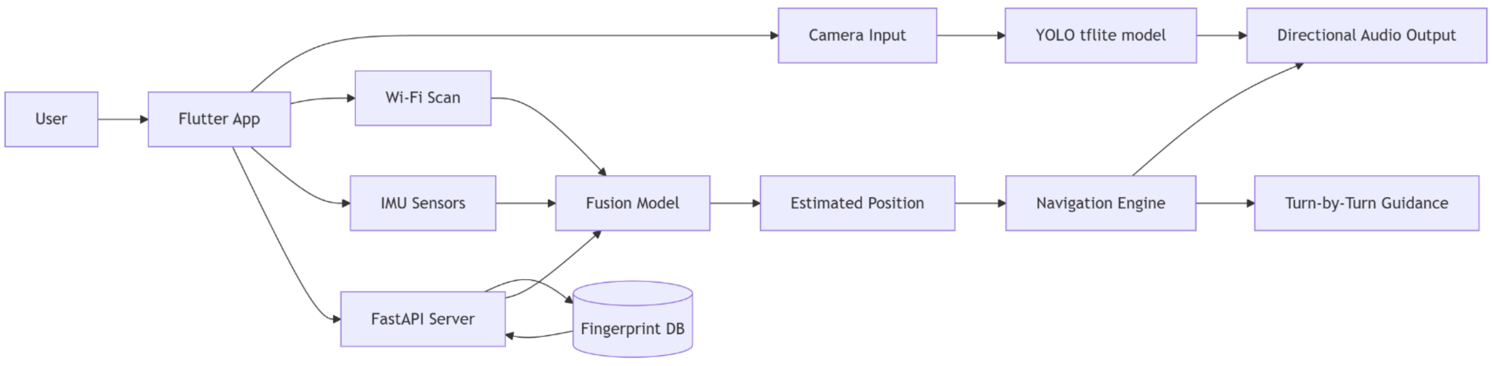

Figure 1 Data Flow Diagram

4.1 Unit Testing

Performed on isolated components such as:

- Wi-Fi parsing & normalization
- IMU step and orientation algorithms
- Camera → YOLO inference layer
- Navigation graph utilities
- API JSON formatting
- Audio mapping logic

Tools: flutter_test, pytest, Mockito, FastAPI TestClient.

4.2 Integration Testing

Integration pairs include:

- Wi-Fi Scanner → Localization Inference
- IMU → Fusion Engine
- YOLO → Audio Module
- Flutter App → FastAPI Backend
- Fusion Engine → Pathfinding → UI Guidance

Goal: verify data flow consistency and absence of module-level incompatibilities.

4.3 System Testing

Executed in A Block, covering:

- End-to-end navigation sessions
- Real-world obstacle detection with movement
- Stress scenarios: weak Wi-Fi, crowded corridor, fast motion
- Real-time audio feedback synchronization

4.4 Performance Testing

Performance metrics include:

Component	Target Requirement
YOLO inference latency	<120 ms
Camera preview	≥ 15 FPS
Wi-Fi scan duration	< 300 ms
Fusion update rate	≥ 3 Hz
Navigation accuracy	±2 m
Server response time	< 200 ms

4.5 User Acceptance Testing (UAT)

Participants: Normal users (initial testing phase)

Success criteria:

- User reaches selected destination with ±2m accuracy
- User correctly perceives audio-based warnings
- Navigation feels natural and safe

4.6 Beta Testing

Conducted in A Block for 1 week.

Focus:

- Long-term stability
- Logging edge cases
- Error handling and recovery
- Real environmental variability (crowds, movement, Wi-Fi drops)

5. Test Environment

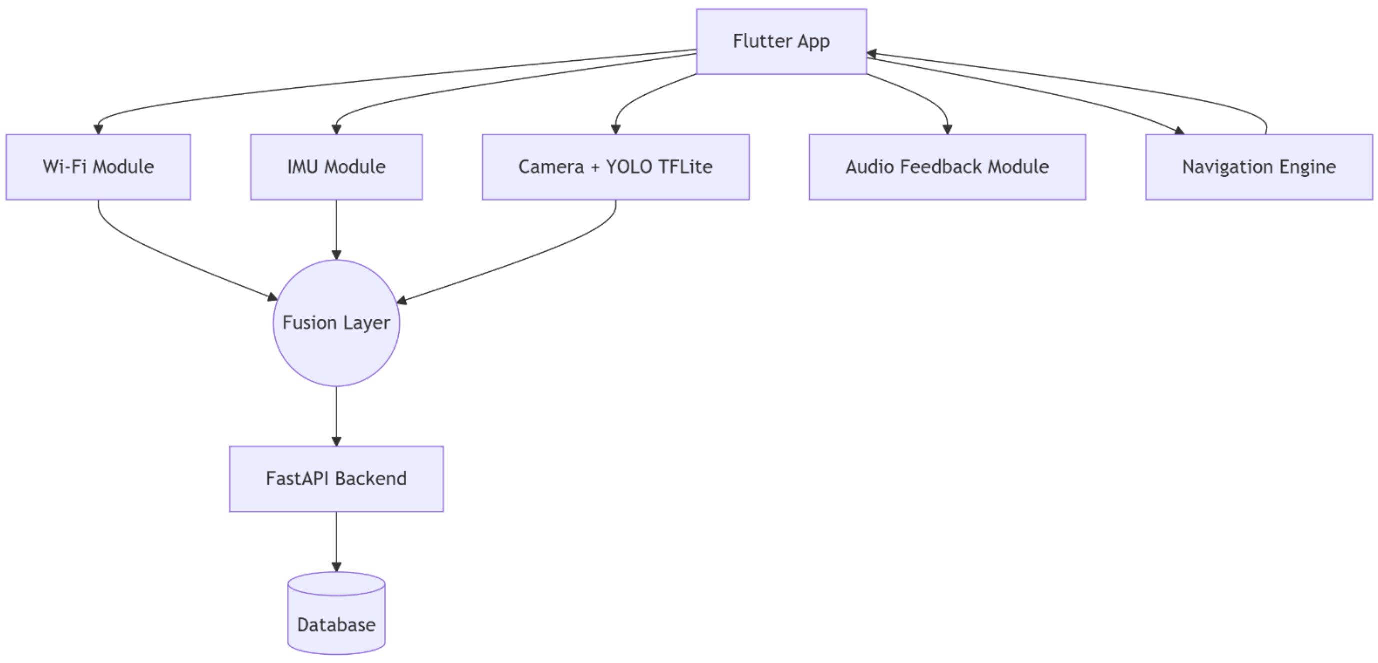

Figure 2 Architecture Diagram

5.1 Hardware

- Google Pixel 6 Pro (Android 16)
- Google Pixel 8 (Android 15)

5.2 Software

- Flutter SDK
- YOLOv8n / YOLOv5n TFLite model
- FastAPI backend
- PostgreSQL / JSON fingerprints
- Testing tools: Insomnia, Postman, pytest, Flutter Test

6. Roles and Responsibilities

Team Member	Role	Responsibilities
Ceyda Kuşçuoğlu	Localization Lead	Wi-Fi fingerprinting, IMU fusion, navigation accuracy
Kıvanç Terzioğlu	Backend Lead	FastAPI endpoints, fusion algorithms
Berkay Kaan Karaca	Mobile Lead	Flutter app, camera + audio modules

7. Risk Analysis

Risk	Likelihood	Impact	Mitigation
Wi-Fi signal instability	High	High	Weighted fusion, smoothing algorithms
IMU drift	Medium	Medium	Kalman tuning, periodic resets
Slow YOLO inference	Medium	High	Use smaller models, optimize preprocessing
Incorrect path edges	Low	High	Manual verification of node-edge maps
Wrong navigation (safety risk)	Low	Very High	Emergency warnings, fallback logic

8. Test Schedule

Week	Activity
1–2	Unit Testing
3	Integration Testing
4	System Testing
5	Performance Testing
6	Fixes + Regression Testing
7	UAT
8	Beta Testing
9	Submission

9.Test Cases

🟦 WI-FI MODULE

Test ID	Summary	Expected Result
TC-01	Wi-Fi scan returns results	Valid RSSI list
TC-02	RSSI normalization	Proper normalized output
TC-03	Missing AP handling	App continues normally
TC-04	Corrupted BSSID	Ignored safely
TC-05	Weak signal test	No crash, accepted values
TC-06	Consecutive scan consistency	Similar RSSI outputs
TC-07	Scan time <300ms	Passed
TC-08	No Wi-Fi fallback	IMU-only mode enabled

🟩 IMU MODULE

Test ID	Summary	Expected Result
TC-09	Step detection accuracy	Error <5%
TC-10	Orientation drift	Stable heading
TC-11	Gyro noise filtering	Smooth values
TC-12	Acceleration spike handling	Filtered correctly
TC-13	Fusion frequency	≥3Hz updates
TC-14	IMU freeze recovery	System resumes

🟥 CAMERA + YOLO

Test ID	Summary	Expected Result
TC-15	YOLO model loads	Success
TC-16	Latency <120ms	Passed
TC-17	Detect person left	Correct
TC-18	Detect person right	Correct
TC-19	Detect center object	Correct
TC-20	No object → No audio	Passed
TC-21	Multi-object detection	Highest priority selected
TC-22	Low-light performance	Detects within limits
TC-23	Motion blur handling	Partial detection still works
TC-24	Wrong class prevention	No false warning

🟧 AUDIO MODULE

Test ID	Summary	Expected Result
TC-25	Stereo mapping	Correct side
TC-26	Volume scaling	Correct by distance
TC-27	Beep frequency scaling	Correct
TC-28	Headphone mode	Works
TC-29	No false alerts	Passed

🟪 NAVIGATION

Test ID	Summary	Expected Result
TC-30	Shortest path	Correct
TC-31	Invalid node handling	Error shown
TC-32	Multi-corridor path	Correct
TC-33	Deviate from route	Recalculated
TC-34	JSON loading	No errors
TC-35	Navigation accuracy	±2m

🟫 SERVER

Test ID	Summary	Expected Result
TC-36	FastAPI returns response	Valid JSON
TC-37	Timeout handling	Retry
TC-38	Fusion accuracy	≤2m error

🟩 UI / ACCESSIBILITY

Test ID	Summary	Expected Result
TC-39	Accessibility toggle	Works
TC-40	Camera permission denied	Error message
TC-41	Wi-Fi permission denied	Error message
TC-42	Crash-free 10 min run	Passed

🟥 UAT

Test ID	Summary	Expected Result
TC-43	UAT navigation	±2m accuracy
TC-44	UAT audio understanding	User passes

🟦 PERFORMANCE

Test ID	Summary	Expected Result
TC-45	CPU usage	<60%
TC-46	RAM usage	<1GB
TC-47	5 min walk test	Stable
TC-48	Peak YOLO stress	No crash

10. Control Procedures

- Version control via GitHub
- Issues tracked using GitHub Projects
- Regression tests after every bug fix
- API logs stored through FastAPI middleware
- Automatic crash logging in Flutter

11. Approval

Name	**Role**
Ceyda Kuşçuoğlu	Localization Lead
Kıvanç Terzioğlu	Backend Lead
Berkay Kaan Karaca	Mobile Lead

12. References

[1] IEEE Standard for Software and System Test Documentation, IEEE Std 829-2008, IEEE Computer Society, 2008.

[2] ISO/IEC/IEEE 29119-1:2013 — Software and Systems Engineering — Software Testing.

[3] Myers, G. J., Sandler, C., & Badgett, T. (2011). *The Art of Software Testing* (3rd ed.). Wiley.

[4] Beizer, B. (1990). *Software Testing Techniques* (2nd ed.). Van Nostrand Reinhold.

[5] Kaner, C., Bach, J., & Pettichord, B. (2002). *Lessons Learned in Software Testing*. Wiley.

[6] Google Flutter Documentation – Testing.
https://docs.flutter.dev/testing

[7] Python FastAPI Official Documentation.
https://fastapi.tiangolo.com/

[8] Ultralytics YOLO Documentation (Object Detection).
https://docs.ultralytics.com/

[9] Android Sensor API Documentation (IMU Sensors).
https://developer.android.com/guide/topics/sensors

[10] Kalman Filtering Tutorial for Sensor Fusion.
https://www.kalmanfilter.net/