Verification and Validation Plan

1. Introduction

The purpose of this Verification and Validation (V&V) Plan is to ensure that the system is built correctly (verification) and that the correct system is built (validation). Verification consists of static activities such as reviews and inspections to ensure phase deliverables meet their requirements. Validation consists of dynamic testing to confirm that the implemented system behaves correctly and satisfies user needs.

2. Verification Plan (Static Testing)

2.1 Verification Methods

Document Reviews (Static Testing)

Reviews will be conducted for the SPMP, SRS, architecture diagrams, and other design artifacts. These reviews confirm correctness, completeness, consistency, and alignment with project requirements. Reviews are held before exiting each development phase.

Code Reviews

All pull requests will undergo peer review. Reviewers verify logic, coding standards, and ensure adequate test coverage. Complexity of modules is assessed, and high-complexity modules are flagged for refactoring.

Unit Testing

Unit tests will be implemented for the ML stub, API handlers, mobile UI components, and utility modules. Tests cover statements, decisions, and conditions to ensure correctness of internal logic.

Integration Testing

Integration tests validate interactions among modules, for example the sequence Capture → Upload → ML Stub → Result → Logging. These tests ensure proper communication between the mobile app, backend API, and database.

Build Verification

A continuous integration pipeline using Git Actions will run build checks, unit tests, and integration tests on each PR. Regression tests will run automatically when merging into the main branch.

2.2 Verification Responsibilities

Project Manager: Oversees all verification activities and documentation reviews.

AWS Lead: Responsible for backend unit tests and related integration checks.

ML Lead: Validates ML stub functionality and API behavior.

Analyst/UX Lead: Ensures requirements traceability and UI flow consistency.

All Team Members: Participate as reviewers during formal inspections and walkthroughs.

2.3 Schedule of Reviews (Static Testing)

Requirements Phase: Review SRS and use cases in Week 2.

Architecture Phase: Review system design diagrams and architecture documentation in Week 4.

Detailed Design Phase: Review API specifications, data models, and UI flows in Week 6.

Implementation Phase: Code reviews occur continuously during development.

Pre-Demonstration: Conduct final system review and dry run of the demo in Week 10.

3. Validation Plan (Dynamic Testing)

3.1 Validation Methods

Customer Demonstration

A complete end-to-end demonstration of the mobile capture, image upload, ML stub inference, nutrition extraction, and trend reporting. The customer evaluates whether the system meets the intended objectives.

Acceptance Tests

A set of 10 curated food images will be used to evaluate core functionality. Acceptance tests will be performed by the customer or designated reviewer, not the developers.

Scenario Walkthroughs

Structured walkthroughs demonstrating onboarding, daily meal logging, user corrections, and viewing of weekly nutritional trends.

Usability Testing

A usability review will ensure the interface is intuitive, efficient, and consistent. The analyst evaluates clarity, navigation, and user experience concerns.

4. Validation Criteria

Functional Criteria

* The app must capture images reliably.

* Image uploads must complete within 3 seconds in the demonstration environment.

* The ML stub must return structured JSON that includes ingredient names and placeholder calorie values.

* The mobile app must display results correctly without formatting issues.

Non-Functional Criteria

* The app must not crash during the demonstration.

* UI flows must be consistent and easy to understand.

* The system must reliably process at least five sequential uploads without slowdown or errors.

5. Dynamic Testing Types Included

Unit Testing: Verifies correctness of individual modules in isolation.

Integration Testing: Ensures modules interact correctly across the data pipeline.

System Testing: Tests the full system in a controlled environment.

Regression Testing: Re-tests prior features after bug fixes or changes.

Stress Testing: Tests sequential uploads and network delays to assess robustness.

Acceptance Testing: Customer-driven confirmation of readiness for use.

Usability Testing: Evaluates the clarity and ease of use of the interface.

6. Test Cases (Examples)

Test Case 01: Basic Upload

Input: Valid JPG image of a single food item

Expected Outcome: Upload completes in under 3 seconds; stub returns valid JSON; app displays one food result.

Test Case 02: Multi-Item Plate

Input: Image showing three food items

Expected Outcome: App displays three items with calorie values and handles multiple detections properly.

Test Case 03: Low Connectivity

Input: Upload attempted with simulated weak network

Expected Outcome: App displays an informative error or retry message, without crashing.

Test Case 04: User Correction

Input: User edits the detected ingredient name

Expected Outcome: App updates the entry and logs the correction successfully.

Test Case 05: Weekly Trends

Input: Dataset with seven logged meals

Expected Outcome: App produces a weekly summary without rendering errors.

7. Traceability

All requirements in the SRS will map to specific verification activities (such as document reviews or code inspections) and to specific validation activities (such as test cases or acceptance tests). This ensures full requirement coverage and clear traceability from requirement to validation evidence.

8. Conclusion

This V&V Plan integrates static verification activities and dynamic validation activities to ensure that defects are identified early and that the complete system behaves correctly and meets customer expectations. The plan ensures comprehensive coverage of both the correctness of the implementation and the suitability of the final system for its intended use.