Software Project Management Plan (SPMP)

Position Digital Health Inc.
Project: Personal Food Log App
Course: SEG 4105 – Software Project Management
Team #18
Date: November 05, 2025

Signatures

Project Sponsor
Signature: _____ Date: _____

Executive Sponsor
Signature: _____ Date:

Project Manager
Signature: _____ Date:

Change History

Version	Date	Author(s)	Description
1.0	Nov 5, 2025	Team #18	Initial SPMP per IEEE 1058
1.1	Nov 15, 2025	Pradyu Vasudev + Luke DuSautoy	Added traceability, metrics, and Gantt overview

Contents

Signatures 2
Change History 3
1. Overview 6
1.1 Project Summary 6
1.1.1 Purpose and Scope 6
1.1.2 Assumptions and Constraints 6
1.1.3 Deliverable Traceability Matrix 6
2 References 7
3. Definitions, Acronyms, and Abbreviations 7
4. Project Organization 7
4.1 External Interfaces 7
4.2 Internal Structure 8
4.3 Stakeholders and Roles 8
5. Managerial Process Plans 9
5.1 Start-Up Plan 9
5.1.1 Estimation Methods 9
5.1.2 Staffing Plan 10
5.1.3 Resource Acquisition Plan 10
5.1.4 Training Plan 11
5.2 Work Plan 11
5.2.1 Work Activities 5.2.2 Schedule Allocation 11
5.2.2 Schedule Allocation 13
5.2.3 Resource Allocation 14
5.2.4 Budget Allocation 14
5.3 Control Plan 14
5.3.1 Requirements control plan 14
5.3.2 Schedule control plan 15
5.3.3 Budget control plan 15
5.3.4 Quality control plan 15
5.3.5 Reporting plan 15
5.3.6 Metrics collection plan 15
5.4 Risk Management plan 16
5.5 Closeout plan 17

6 Technical control plans 17
6.1 Process model 17
6.2 Methods Tools and Techniques 17
6.3 Infrastructure plan 18
6.4 Product acceptance plan 18
7. Technical and Supporting Process Plans 19
7.1 Configuration Management 19
7.2 Verification and Validation plan 19
7.3 Documentation plan 20
7.4 Quality assurance plan 20
7.5 Reviews and Audits 20
7.6 Problem resolution plan 20

1. Overview

1.1 Project Summary

1.1.1 Purpose and Scope

This SPMP defines how Team #18 will plan, execute, monitor, and close the *Personal Food Log App* project. It establishes responsibilities, schedules, risks, and control mechanisms consistent with IEEE 1058 and SEG 4105 requirements.

The project delivers a proof-of-concept mobile application that enables users to log daily food intake through smartphone images. Photos are uploaded to AWS, processed by a machine-learning stub that identifies visible ingredients and returns estimated calories. Advanced nutrition analytics, social features, and external-platform integration are excluded.

1.1.2 Assumptions and Constraints

- Ingredient-level detection and calorie calc drive design; PoC uses placeholders, but architecture supports higher accuracy later.
- Visible ingredients are assumed to extend uniformly through the dish.
- Auto-calibration uses image-based methods; no external objects are allowed.
- Non-visible items (salt/oil/seasoning) may be undetectable, acceptable limitation.
- Public food datasets allowed; incremental learning expected later—PoC exposes data/labeling hooks.
- Must use AWS and design for scalability.
- Mobile platforms: Android and iOS.
- Feedback available only during the two scheduled weeks; PM is sole communicator with customer.
- Use non-PII images or consented anonymized data (privacy awareness).
- Smartphones with functional cameras are available for testing.
- Stable network access is available during testing and demo sessions.
- Must use AWS infrastructure, no external calibration objects.
- Must support both Android and iOS for the app.
- Privacy norms (e.g., PIPEDA) apply to any real-user data.

1.1.3 Deliverable Traceability Matrix

Milestone	Deliverable	Primary Owner	Evaluation Criterion
M1	Charter	Pradyu	Customer approval
M2	UX Flows & Backlog	Luke	Feedback Meeting 1
M3	Mobile Capture Upload	Samuel	Functional Demo
M4	ML Stub	Wilt	Feedback Meeting 2

Milestone	Deliverable	Primary Owner	Evaluation Criterion
M5	End-to-End Demo	Samuel	Stable PoC
M6	Project Plan + PoC	Pradyu	Customer review
M7	Presentation	Luke	Sponsor evaluation
M8	Final Stakeholder approval	Luke	Sponsor evaluation
M9	Post-Performance Analysis	Pradyu	Final submission

2 References

- SEG 4105 Course Outline and Deliverable Schedule
- *Project Charter v1.0* (26 Sep 2025)
- IEEE Std 1058-1998 — Software Project Management Plans
- AWS Free-Tier Documentation

3. Definitions, Acronyms, and Abbreviations

AI – Artificial Intelligence.
AWS – Amazon Web Services.
PoC – Proof of Concept (course deliverable).
VBM – Visual-Based Measurement systems for food logging.
User – End user of the mobile application (patients, athletes, fitness users).
Clinician – Dietician, doctor, or coach who may later review user data (primarily future scope).
Ingredient – A visually distinguishable food component (e.g., lettuce, tomato, rice).
CCB – Change Control Board (PM + Customer).

4. Project Organization

4.1 External Interfaces

User Interface (Mobile App)

- **Home Screen:** Quick actions for logging meals, viewing today's summary, history, and profile.
- **Camera Screen:** Live camera view with standard controls and image preview.
- **Results Screen:** Shows captured image and detected ingredients with estimated portions and calories.
- **History Screen:** Displays past meals by date with navigation to details.
- **Settings Screen:** Allows editing user info and dietary preferences.

Hardware Interfaces

- Uses the device camera via native APIs.
- Uses device storage for temporary image caching and offline uploads.

Software & Communication Interfaces

- **Backend API (HTTPS):** Endpoints for uploading meals and retrieving meal data (JSON-based).

- **AWS Services:** S3 for image storage, Lambda/containers for processing, a database for metadata, and optional future authentication via Cognito.

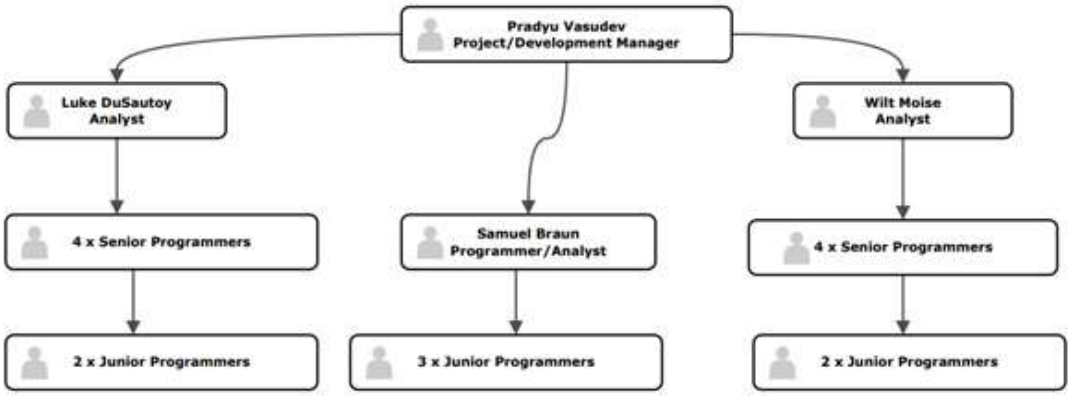
4.2 Internal Structure

Structure: Functional structure, People are divided based on their specialities.

Teams: 3 separate teams, UI/UX, ML, and Cloud

Change Control Board (CCB): PM + Customer. Sponsor consulted as needed.

Decision Rights: Component owners → PM tie-break → CCB for scope/time issues.



4.3 Stakeholders and Roles

Role	Name	Responsibilities
Sponsor	Shervin Shirmohammadi	Oversight, approval of milestones
Customer	Tushar Bhatia	Feedback
Project Manager	Pradyu Vasudev	Planning, risk management, AWS infrastructure
Analyst & UX Lead	Luke DuSautoy	Requirements, wireframes, competitor benchmarking
Analyst & ML Lead	Wilt Moise	Data pipeline stub and calibration mock
Mobile Lead	Samuel Braun	Mobile capture, API integration, CI builds

5. Managerial Process Plans

5.1 Start-Up Plan

Charter approval.

Git repository and AWS set up.

Low-fidelity wireframes and API contract ready.

POC demonstration, and stakeholder approval to move forward.

5.1.1 Estimation Methods

Two complementary effort-estimation techniques were used to ensure that schedule and workload projections were realistic, unbiased, and cross-validated:

Blitz Method Estimation

The team performed a rapid, collaborative estimation session where the entire WBS was reviewed as a group. Each member independently proposed rough effort ranges for each task (in hours), followed by a group discussion to quickly converge on a consensus estimate.
Blitz Estimation was chosen because:

- It accelerates initial forecasting in early-phase planning.
- It exposes hidden tasks and missing dependencies through fast group feedback.
- It suits projects with small teams and compressed timelines, such as SEG4105. Blitz produced the first pass estimate, which served as the baseline for review.

Wideband Delphi Estimation

After Blitz results were established, a three-round Wideband Delphi cycle refined the numbers. Each member estimated task effort anonymously, after which:

1. Individual estimates were collected and averaged.
2. The range and variance were analyzed.
3. Outliers triggered a discussion moderated by the Project Manager.
4. A second and third round were conducted until convergence (variance < 20%).

Wideband Delphi was chosen because it:

- Reduces bias from dominant personalities.
- Forces consideration of uncertainty and hidden risk.
- Produces statistically stable estimates despite a small team size.

Resulting Estimations

The final effort values used in the WBS, and schedule represent the **converged Wideband Delphi output**, cross-validated against the Blitz session for consistency. These estimates were then converted to calendar time using an availability constraint of **around 40 hours/week per employee** and applied against the schedule in and milestone deadlines defined in Section 3.3.

5.1.2 Staffing Plan

The project requires a small, cross-functional team capable of building a mobile proof-of-concept with an AWS-backed image-processing pipeline. Staffing aligns to the project phases: Inception → Construction → Transition → Close-Out.

Total Staff: 18 developers
Average Availability: ~40 hours/week (per the SPMP assumption)

Roles emphasize shared responsibilities but maintain clear ownership for deliverables and approvals.

5.1.3 Resource Acquisition Plan

Software Tools

- Flutter SDK, Dart, Python – Installed by each developer.
- GitHub Repo – Created during project setup.
- Figma – UX design tool (free tier).
- VS Code/Android Studio/Xcode – Developer-installed.

Acquisition Method:
Team installs and configures tools individually following PM's setup guide.

Cloud Resources

- S3 bucket (test + prod)
- Lambda function for ML Stub
- IAM roles & permissions
- Optional Cognito user pool (future)

Acquisition Method:
The PM provisions all AWS components using free-tier eligible resources. Access keys are shared securely via GitHub Secrets for CI.

Hardware Resources

- Personal laptops (BYOD)
- At least one Android device
- At least one iOS device or simulator

Acquisition Method:
Devices provided by team members; simulators used when physical devices unavailable.

5.1.4 Training Plan

Internal 15-minute micro-sessions on AWS setup, Flutter build, and Git workflow; how-to notes in repo.

Training is two-tiered:

1. **Onboarding of Junior Developers** – Pair programming sessions and 15-minute "micro-labs" on Flutter and GitHub workflows led by senior members.
2. **Cross-Team Machine Learning Upskilling** – All members complete the Coursera "Intro to TensorFlow" module and study dataset annotation using Kaggle Food-101.
   Each training activity is logged, and completion is reviewed during weekly stand-ups.
   The intent is to ensure shared competency in both mobile and ML domains before PoC integration.

5.2 Work Plan

5.2.1 Work Activities 5.2.2 Schedule Allocation

Initiation Phase

This phase focuses on establishing the project's scope, boundaries, initial architecture, and detailed requirements.

ID	Stories	Owner	Deliverable	Due
S1	**Project Charter & Setup**	**Pradyu**	**Approved Charter**	**Sep 30**
	Description: Formal documentation recognizing project existence, defining goals, scope, objectives, quality level, and validating business justification.			
S2	**Requirements & UX Flows**	**Luke**	**Wireframes + Backlog**	**Nov 10**
	Description: Defining user needs, system constraints, and interface visualization. Deliverables include wireframes (design element) and a backlog (collection of user stories/requirements).			
S3	**Final Plan & PoC**	**Samuel**	**IEEE-1058 SPMP + PoC**	**Nov 20**
	S3.1	SPMP Documentation		IEEE-1058 SPMP
	Description: Finalized Software Project Management Plan (SPMP) documentation.			
	5.2.1.3.2	Proof of Concept Delivery		PoC

	Description: Delivering the final Proof of Concept software.			
S4	**Demo Presentation**	**Wilt**	**Slides + Live Demo**	**Nov 28**
	S4.1	Demo Preparation		Slides
	S4.2	Live Demonstration		Live Demo

Core Development and System Assembly (Construction Phase)

This phase involves achieving useful, working versions of the product, often referred to as alpha or beta versions.

ID	Stories	Owner	Deliverable	Due
S5	**Mobile Capture & Upload**	**Samuel**	**Android/iOS Upload Demo**	**Nov 15**
	Description: Implementation of the mobile application functionality to capture and upload data.			
S6	**ML Pipeline Stub**	**Wilt**	**Server Stub Return Data**	**Nov** 18
	Description: Implementation of the machine learning backend component, resulting in a server stub that returns data.			
S7	**Integration & Testing**	**Pradyu**	**Stable Demo Build**	**Nov 25**
	Description: Combining individual modules (Mobile, ML Stub) and ensuring they interact correctly (integration testing). Goal is to achieve adequate quality.			
S8	**AI training and fine tuning**	**Samuel**	**AI model**	**Dec 15**
	Description: Fine tune AI model to achieve at least 94% accuracy against real world test data.			
S9	**Completed UI interface**	**Luke**	**Flutter App**	**Jan 20**
	Description: Fine tune AI model to achieve at least 94% accuracy against real world test data.			

Project Finalization and Deployment (Transition Phase)

This phase ensures stakeholder concurrence of completeness and achieves the final product baseline and begins the process of deploying the application.

ID	Stories	Owner	Deliverable	Due
S10	**Final Stakeholder Approval**	**Luke**	**Working Application**	**Jan 50**
	S10.1	Non-deployed app		All documentation,

				Final app prototype
	Description: Finalized Software Project Management Plan (SPMP) documentation.			
	5.2.3.1.2	Deployment plan		Detailed plan for app release
	Description: Documentation detailing the release of the application.			
S11	**Deployment**	**Samuel**	**User Facing Application**	**Feb 15**
	S11.1	List application in stores		Approval

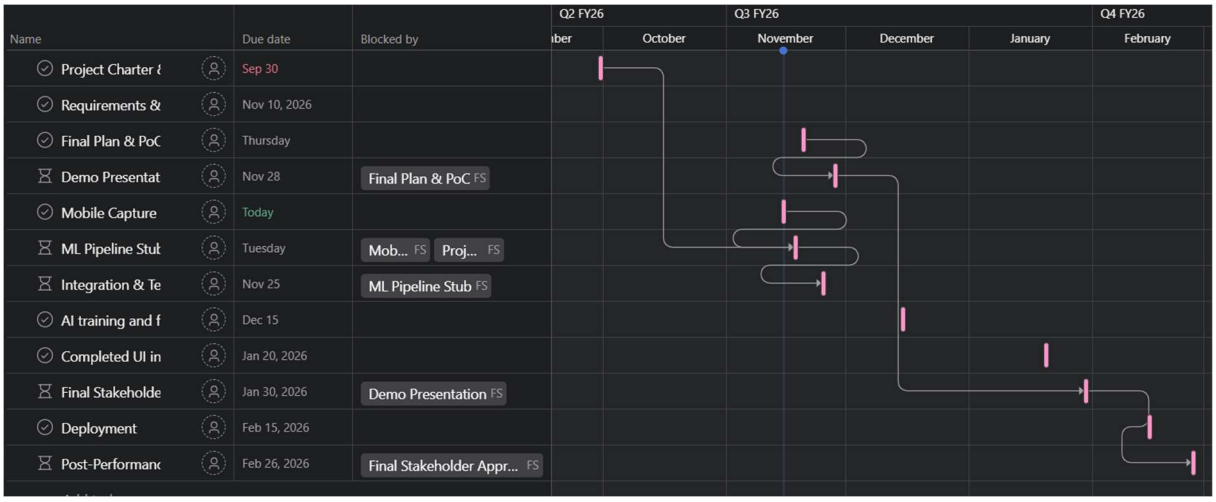
Project Closure (Post-Performance Analysis)

This activity occurs after the project is functionally complete, supporting Continuous Process Improvement.

ID	Stories	Owner	Deliverable	Due
S12	**Post-Performance Analysis**	**Pradyu + Luke**	**Performance Report**	**Feb 26**
	Description: Measuring and analyzing process metrics, collecting data on lessons learned, and providing strategies for improving the process for future projects.			

5.2.2 Schedule Allocation

Gantt Chart



Schedule Dependencies

Story 3 (Mobile Capture) → Story 4 (ML Pipeline Stub) → Story 5 (Integration & Testing).

Charter approval (M1) → Pipeline Stub (M4).

PoC stability (M6) → Presentation (M7) → Final Approval (M8) → PPA (M9)

5.2.3 Resource Allocation

The project uses a functional structure with three specialized teams—UI/UX, ML, and Cloud—each responsible for installing their own software tools such as Flutter, Dart, Python, Figma, and IDEs following the project manager's setup guide. GitHub serves as the shared code repository for all teams. The project manager provisions all AWS cloud resources, including S3 buckets, a Lambda ML stub, and IAM roles, with access securely shared through GitHub Secrets. Hardware resources are provided through a bring-your-own-device approach, with team members using personal laptops and at least one shared Android and iOS device **per team, if required use** simulator**s** for testing.

5.2.4 Budget Allocation

Human: 18 employees, 40 h/week each.

Technical: AWS, GitHub, Development Devices.

The estimated total project budget is **$246500 CAD equivalent**, based on nominal rates and not billed costs.

Category	Description	Estimated Cost (CAD)
Development Salaries	18 developers × 40 h/week × 10 weeks × $30/h	$216,000 (in-kind)
Software Licenses	IDEs (VS Code free), design tools (Figma license), Git Hub	$500
Cloud Rental	AWS (S3, Lambda, Cognito)	$10,000
Hardware	Personal laptops and smartphones for testing	$20,000
Total		**$246,500 in effort value**

5.3 Control Plan

5.3.1 Requirements control plan

The project manager will maintain a centralized requirements document, and any changes must be submitted through a formal change request, reviewed by the PM and relevant team leads (UI/UX, ML, Cloud), and approved before being added to the project scope. Requirements will be version-controlled in the project repository, and updates will be

communicated during weekly team meetings to ensure all teams stay aligned and no unauthorized changes are made.

5.3.2 Schedule control plan

The project manager will maintain the project schedule using a shared tracking tool, updating task statuses weekly based on input from the UI/UX, ML, and Cloud team leads. Any delays, risks, or needed adjustments must be reported immediately to the PM, who will assess impacts and decide whether to reassign resources, modify deadlines, or escalate changes through the formal change control process. Progress reviews will occur in weekly stand-ups, ensuring that deviations are identified early and the schedule remains realistic and achievable.

5.3.3 Budget control plan

The project manager will monitor expenses weekly, focusing on cloud resource usage, software subscriptions, and any unexpected costs. The main budget oversight will involve AWS usage to prevent accidental overages. Any cost impacts or potential overruns must be reported immediately to the PM, who will evaluate the issue and approve or deny additional spending through the formal change request process.

5.3.4 Quality control plan

Each team is responsible for reviewing its own work—UI/UX validates designs and usability, the ML team tests model functionality and outputs, and the Cloud team verifies integrations and performance. The project manager will enforce code reviews, require peer testing before merging to the main branch, and ensure that all features pass functional tests on both Android and iOS devices. Any defects discovered during reviews or testing must be logged, assigned, and resolved before deployment.

5.3.5 Reporting plan

Progress Reviews: Weekly stand-ups + GitHub burndown.

Status Reports: Weekly summary to Customer

Change Control: Request → Impact (24 h) → CCB Decision (48 h).

5.3.6 Metrics collection plan

Metrics:

- Schedule variance ≤ ±10 %
- Defect density < 0.3/LOC (qualitative check)
- Demo readiness index ≥ 90 % of planned features operational.

Schedule variance is calculated based on the following Milestones.

ID	Milestones	Owner	Deliverable	Due

M1	Project Charter & Setup	Pradyu	Approved Charter	Sep 30
M2	Requirements & UX Flows	Luke	Wireframes + Backlog	Oct 6
M3	Mobile Capture & Upload	Samuel	Android/iOS Upload	Nov 10
M4	ML Pipeline Stub	Wilt	Server Stub Return Data	Nov 15
M5	Integration & Testing	All	Stable Demo Build	Nov 20
M6	Final Plan & PoC	All	IEEE-1058 SPMP + PoC	Nov 20
M7	Demo Presentation	All	Slides + Live Demo	Nov 28
M8	Final Stakeholder Approval	Luke	Working Application, Deployment plan	Jan 30
M9	Post-Performance Analysis	Pradyu + Luke	Final Report	Feb 26

5.4 Risk Management plan

ID	Risk	Cat.	P	I	Score	Owner	Mitigation	Contingency
R1	Scope creep from late TA requests	Scope	3	5	**15 (High)**	PM	Enforce CCB; freeze scope by M5	Defer new requests to post-demo
R2	Developer unavailable (illness, overload)	Resource	2	5	**10 (Med-High)**	PM	Cross-train; shared "how-to" docs	Reassign tasks; drop non-critical features
R3	Mobile build breaks across Android/iOS	Tech	3	3	**9 (Med)**	Mobile Lead	Pin SDKs; CI build tests	Use simplified fallback UI
R4	AWS Free Tier throttling or quota issues	Infra	2	3	**6 (Low-Med)**	PM	Local mocks; reduce payload size	Use cached responses for demo
R5	Auto-calibration produces unrealistic outputs	Algorithm	3	4	**12 (High)**	ML Lead	Baseline calibration method early	Apply fixed scale factor

ID	Risk	Cat.	P	I	Score	Owner	Mitigation	Contingency
R6	Integration delays jeopardize PoC stability	Schedule	3	5	**15 (High)**	All	Weekly integration checkpoints	Cut scope to core flow only
R7	Demo-day cloud or device failure	External	1	5	**5 (Med)**	Mobile Lead	Offline demo mode; backup device	Pre-recorded demo video

5.5 Closeout plan

Deploy approved application.
Submit PPA.
Conduct team retrospective and archive GitHub repo (tag v1.0).
Deliver demo video and final documentation bundle for continuous support.

6 Technical control plans

6.1 Process model

Hybrid Agile incremental model: each 1 iteration has analysis → prototype → release → feedback.

6.2 Methods Tools and Techniques

Project communication follows a structured, transparent pattern:

- **Weekly Team Syncs** – 30 min meetings to review progress, blockers, and upcoming tasks.
- **Customer Status Reports** – Concise summaries submitted via Brightspace every Friday.
- **Issue Tracking** – All tasks, bugs, and change requests logged and discussed in GitHub Issues.
- **Emergency Channels** – Slack DMs or email (PM → Customer only) for time-critical issues.
- **Documentation Updates** – Major decisions recorded in Decision_Log.md within the repo.
  The Project Manager is the sole liaison to the Customer; all other communication occurs internally.

Change control follows a structured workflow:

- **Submit Change Request (CR)** in GitHub Issues
- **Impact Analysis** (PM, within 24 hours)
- **CCB Decision** (PM + Customer, within 48 hours)
- **Baseline Update** (if approved)

- **Documentation Update** of affected artifacts

6.3 Infrastructure plan

The system will be built with a cloud-first architecture using AWS services to ensure scalability. The mobile application, developed in Flutter with an Android-first approach, will handle image capture, upload, and meal logging. Users will authenticate through AWS Cognito, which supports SSO via Google and Apple, providing secure identity management. Uploaded images will be stored in Amazon S3 using pre-signed URLs generated by a backend hosted on AWS Lambda behind API Gateway, enabling stateless, serverless APIs for image inference requests, meal CRUD operations, user management, and feedback collection. AWS Lambda ensures scalability with automatic resource allocation, while API Gateway enforces authentication, throttling, and request routing.

For machine learning, AWS SageMaker will host the food recognition models, including multi-label classifiers and segmentation modules for volume estimation. SageMaker endpoints will return predicted ingredients, portion sizes, and nutritional information, which the backend will pass to the mobile app. User corrections and low-confidence predictions will feed into a feedback and retraining pipeline orchestrated by AWS Step Functions, with datasets stored in S3 and metadata in Amazon DynamoDB. SageMaker Model Monitor will track data drift, accuracy degradation, and underrepresented classes, triggering automated retraining workflows to update model weights in production.

The data storage layer leverages S3 for raw images, thumbnails, and training datasets, and DynamoDB for structured data, including user profiles, meal logs, prediction metadata, feedback, and the nutrition database. Security is enforced using IAM roles with least privilege, server-side encryption for S3 and DynamoDB, and network isolation where necessary via VPCs for Lambda or SageMaker endpoints. Logging and monitoring are centralized in CloudWatch, with alerts routed through Amazon SNS for incidents such as model drift, API errors, or storage thresholds.

For development and deployment, the infrastructure will support multiple environments, including local development, a development AWS sandbox, staging, and production. CI/CD pipelines will be implemented using AWS-native to automate builds, tests, and deployments for mobile, backend, and ML models. This setup ensures a reproducible, secure, and maintainable infrastructure that supports continuous improvement of the system, with clear paths for scaling to additional platforms like iOS or web in the future.

6.4 Product acceptance plan

The system will be accepted once it passes functional, performance, and usability tests. Functional acceptance includes user authentication via Cognito, image uploads to S3, accurate food recognition and volume estimation via SageMaker, and correct meal logging in DynamoDB. Performance targets include ≥94% recognition accuracy and reliable API responses under expected load. Usability is validated through intuitive meal logging, editing, and trend visualization. All critical defects must be resolved, security standards met, and monitoring via CloudWatch and SNS operational before acceptance.

7. Technical and Supporting Process Plans

7.1 Configuration Management

Repo structure: app/, backend/, docs/.

Change tracking via GitHub issues and tags (v1-M3, v1-M5…).

Artifacts stored in /docs.

The following configuration items form the controlled baseline for this project:

- **Documents:** Project Charter, SPMP, Requirements Specification, Software Architecture Document, Risk Log, API Contract, PPA
- **Code Artifacts:** Flutter mobile app (Android/iOS), backend Lambda functions, data models
- **Media:** Wireframes, mockups, demo video
- **Infrastructure:** AWS configuration (S3 buckets, Lambda handlers, IAM roles)
- **Supporting Files:** Test images, labeling notes, how-to guides, meeting minutes

7.2 Verification and Validation plan

Verification will be performed by testing each system function against expected inputs and outputs, including image uploads, ML inference, meal logging, and feedback capture. Automated and manual tests will ensure API endpoints, database operations, and mobile workflows operate correctly under normal and edge-case conditions. Validation will be conducted through end-to-end demonstrations of the complete system, showing image recognition, nutrition calculation, user corrections, and trend reporting. Customer review and approval of these demonstrations will confirm that all objectives are fully met.

Verification Methods

- Document Reviews: Inspection of SPMP, requirements, and architecture for completeness
- Code Reviews: Peer review of all PRs before merge
- Unit Testing: Manual unit tests for ML stub, API handlers, and UI components
- Integration Testing: End-to-end pipeline tests (Capture → Upload → Stub → Result)
- Build Verification: CI ensures mobile and backend builds succeed with latest commits

Verification Responsibilities

- PM: Documentation verification
- Mobile Lead: Mobile unit tests
- ML Lead: Stub verification
- Analyst/UX Lead: Requirements traceability and UI flow consistency

Validation Methods

- Customer Demo: Validation of PoC behavior
- Acceptance Tests: Using 10 curated test images across different food types

- Scenario Walkthrough: Full flow demonstration in the presentation
- Usability Review: Analyst evaluates clarity of UI flows

Validation Criteria

- App captures images without failure
- Upload to AWS completes under 3 seconds (demo environment)
- Stub returns structured JSON with at least ingredient names + placeholder calories
- Mobile app displays results without formatting errors

7.3 Documentation plan

The project's documentation will be maintained in a centralized repository, covering system architecture, API specifications, data models, and ML model behavior. User-facing documentation will include onboarding instructions, feature descriptions, and troubleshooting guidance for the mobile app. Operational documentation will describe monitoring, alerting, and maintenance workflows for AWS services.

7.4 Quality assurance plan

Quality assurance will include automated and manual testing for all core features, with every deliverable undergoing peer review by a non-author and meeting defined acceptance criteria per milestone. All external interface deliverables will require customer approval before advancing to the next project phase, ensuring quality and alignment with expectations.

7.5 Reviews and Audits

All project deliverables will undergo formal peer reviews and periodic internal audits to ensure compliance with technical standards, documentation quality, and security requirements.

7.6 Problem resolution plan

All issues will be logged and categorized by severity, then assigned to the appropriate team member for analysis. The resolution process will follow these steps: identify and reproduce the problem, determine the root cause, implement and document the fix retest to verify resolution, and obtain customer confirmation when applicable. Critical issues will follow an accelerated escalation path to ensure timely remediation.