PawfectMatch
TEST PLAN

Version 1.2

3/11/2025

VERSION HISTORY

Version #	Implemented By	Revision Date	Approved By	Approval Date	Reason
1.0	Chloie	1/11/25	Zi Jian	4/11/25	Complete Test Plan Section 2 - 8
1.1	Nelly	3/11/25	Zi Jian	4/11/25	Complete Test Plan Section 9 - 12 Drafted Section 15
1.2	Zi Jian	3/11/25	Zi Jian	4/11/25	Complete Test Plan Section 13 - 14

TABLE OF CONTENTS

1 INTRODUCTION....5
2 TEST ITEMS....6
2.1 CONFIGURATION ITEMS....6
2.2 MODULES AND FEATURES....6
2.3 INTERFACES....7
2.4 TEST ITEM TRACEABILITY....8
3 SOFTWARE RISK ISSUES....9
3.1 RISK MANAGEMENT PLAN RISKS....9
3.2 ADDITIONAL RISKS....10
4 FEATURES TO BE TESTED....11
4.1 CORE FEATURES....11
4.2 ENHANCEMENT FEATURES....12
5 FEATURES NOT TO BE TESTED....13
6 APPROACH (STRATEGY)....14
6.1 TESTING LEVELS....14
6.2 TESTING METHODS AND TECHNIQUES....14
6.3 TOOLS AND CONFIGURATION MANAGEMENT....15
6.4 REGRESSION TESTING APPROACH....15
6.5 METRICS AND REPORTING....16
6.6 HANDLING AMBIGUOUS REQUIREMENTS....16
6.7 ORGANIZATIONAL PROCESSES AND REVIEWS....16
6.8 SPECIAL TESTING CONSIDERATIONS....16
6.9 AUTOMATED TESTING SUPPORT....17
7 ITEM PASS/FAIL CRITERIA....18
7.1 INDIVIDUAL TEST CASE CRITERIA....18
7.2 DEFECT SEVERITY AND ACCEPTANCE THRESHOLDS....18
7.3 OVERALL COMPLETION CRITERIA....19
8 SUSPENSION CRITERIA AND RESUMPTION REQUIREMENTS....20
8.1 SUSPENSION CRITERIA....20
8.2 RESUMPTION REQUIREMENTS....20
9 TEST DELIVERABLES....22
10 ENVIRONMENTAL NEEDS....23
11 RESPONSIBILITIES....24
12 STAFFING AND TRAINING NEEDS....25
13 SCHEDULE....26
14 RISKS AND CONTINGENCIES....27

15 APPROVALS......28
REFERENCES......29

1 INTRODUCTION

This Test Plan outlines the overall testing objectives, scope and approach for verifying and validating the PawfectMatch App. This document will cover all major parts of the testing process including:

- The items to be tested, such as the interfaces and features.
- The risk and challenges that may affect testing and how they will be managed.
- The features to be included or excluded in testing, along with its priority.
- The testing approach and strategy, including the details of the different test levels: Unit, integration, system and User Acceptance, followed by testing techniques and tools.
- The pass/fail criteria for evaluation, test deliverables and overall test schedule.

2 TEST ITEMS

The items to be tested in this plan are the core modules, submodules, and interfaces that comprise the *PawfectMatch* system. Each item corresponds to a Configuration Item (CI) and release component identified in the Configuration Management Plan (CMP) and Release Plan.

Testing will verify that each module performs as defined in the Software Requirements Specification (SRS), meets the functional and non-functional requirements, and integrates correctly with dependent components and external services.

2.1 CONFIGURATION ITEMS

The *PawfectMatch* system is a 3-tier application. Testing will validate all three integrated layers as a single entity. The CIs under test are as follows.

CI Tier	Identifier	Description
Frontend	v1.0.0	React + TypeScript presentation layer, including all components, pages and UI logic
Backend	v1.0.0	FastAPI + Python application layer, including all business logic, API endpoints, and database connections
Database	v1.0.0	PostgreSQL database schema, including all tables, relationships, and Alembic migration history

2.2 MODULES AND FEATURES

The following table details the specific functional modules to be tested, mapped to the release in which they are introduced.

Module	Description	Functionalities	Release
User Management	Manages user registration, authentication, onboarding, and profile data	Email + Password Login Google OAuth Login Pet Owner Onboarding Service Provider Onboarding View User Profile Edit User Profile	R1
Pet Management	Handles pet information linked to user accounts	View Pet Profile Create Pet Profile Edit Pet Profile	R1

		Delete Pet Profile	
Service Management	Enables service providers to create and manage offered services	View Offered Service Create Offered Service Edit Offered Service Delete Offered Service	R1
Booking Management	Controls creation and lifecycle of bookings between owners and providers	View Booking History View Booking Details Create Booking Approve Booking Decline Booking Confirm Booking Cancel Booking	R1
Search	Provides searchable and filterable access to available pet services	View Service Listings Keyword Search Category Filter Price Filter Location Filter	R1
Payment	Integrates PayPal sandbox for secure payments	Submit Payment Transaction Validation	R1
Review	Allows pet owners to provide post-service feedback	Submit Review Edit Review Delete Review View Average Service Rating	R2
Location	Provides map-based discovery of nearby services	Google Maps Integration Distance Filtering Address Validation	R3
Enhancements	Adds polish and operational reliability features	Booking Email Notifications UI/UX Improvements Accessibility compliance	R4

2.3 INTERFACES

Testing will also validate the critical interfaces between the Configuration Items.

Interface	Description	Test Objective

Frontend ↔ Backend	REST API calls between the React (client) and FastAPI (server)	Verify correct data exchange (JSON), HTTP status codes, session management, and error handling
Backend ↔ Database	ORM layer (SQLAlchemy) communication with the PostgreSQL database	Ensure all CRUD operations are accurate, data integrity is maintained, and schema migrations are correct
Backend ↔ External Services	API calls to Google OAuth 2.0, PayPal Sandbox, and Google Maps API	Validate authentication, token handling, payment processing, and error handling for external errors

2.4 TEST ITEM TRACEABILITY

All test items are fully traceable. Each module in section 2.2 maps directly to one or more functional requirements specified in the SRS. All components in sections 2.1 and 2.3 map directly to CIs defined in the CMP.

Testing will confirm that:

- All components integrate without regression
- Data flow between interfaces remains consistent and secure
- Final product baseline functions as a single, complete system

3 SOFTWARE RISK ISSUES

The testing of the *PawfectMatch* system may be affected by a number of technical and operational risks. The primary risks are derived directly from the project's formal Risk Management Plan (RMP) to ensure complete traceability. Additional risks specific to the testing process itself are also identified.

3.1 RISK MANAGEMENT PLAN RISKS

ID	Risk	Testing Impact	Mitigation
R07	Deployment Environment Failure	May block end-to-end and UAT testing or result in inconsistent results between builds	Conduct early environment verification Maintain a backup local deployment Create rollback scripts before test cycles
R08	Critical Software Bugs	High disruption to testing schedule and data validity May require retesting of multiple modules	Perform unit and integration testing prior to each merge Log defects in GitHub issues Prioritize fixes before regression testing
R11	API Integration Failure	Fails integration tests and prevents validation of inter-module functionality	Validate schema contracts Conduct early integration tests on all interfaces
R12	Database Schema Mismatch	CRUD-related test cases may fail due to structural differences or missing fields	Synchronize migrations using Alembic Confirm database version tags before executing tests
R16	Lack of Requirement Validation	Leads to incorrect or incomplete test coverage and false-positive results	Verify acceptance criteria during meetings Derive all test cases directly from finalized SRS
R24	Missed Lab Submission Deadlines	Reduces available time for regression or non-critical feature testing	Prioritize critical modules Defer low-risk cases to contingency testing window
R25	Scope Creep	Additional unplanned test cases increase regression	Enforce the Change Request process

		workload and delay test closure	Rebaseline the Test Plan and schedule only after CCB approval

3.2 ADDITIONAL RISKS

The following risks are not part of the formal Risk Management Plan but have been identified as specific to the testing effort. They are derived from the QA team's past experience and reflect test execution realities.

ID	Risk	Testing Impact	Mitigation
T01	Regression Defects from Phased Integration	Critical paths may fail after new module integrations, leading to retesting overhead	Maintain a Regression Test Suite covering all core modules Re-run regression cases after every deployment or fix Tag stable builds after each release cycle
T02	Data Integrity and State Inconsistency	Database and UI data may not match, causing false-positive results or invalid workflows	Conduct gray-box testing by validating backend database states after each test action Verify data relationships using SQL queries and logs
T03	Insufficient or Unrealistic Test Data	Test results may vary between environments, making defect replication difficult	Standardize environment setup via Docker or configuration templates Perform environment verification before each test cycle Document all configuration changes in GitHub

4 FEATURES TO BE TESTED

Each major feature to be tested is assigned a test priority based on its criticality to the application's success, its technical complexity, and the risks identified.

- **High (H):** Critical-path functionality where failure is catastrophic, blocks other features, or represents a major business risk
- **Medium (M):** Essential supporting functionality where failure is a severe defect but may not block the entire application
- **Low (L):** Non-critical features or Quality of Life enhancements where failure is an inconvenience but does not impact core functionality

4.1 CORE FEATURES

Feature	Description	Priority	Justification
User Registration	Users can create an account using email + password	M	Registration is essential for gaining new first-time users. However, it is unnecessary for existing users and used infrequently compared to other features.
User Authentication	Users can login via email + password Users can log in via Google OAuth	H	Authentication is critical for security and access control. Failure here prevents all users from entering the system.
Onboarding	On first login, users are guided through onboarding to complete their profile Pet owners add pet profiles Service providers create service listings	H	Onboarding is mandatory for both user roles to activate key workflows. Without it, neither pet owners nor service providers can use core functionalities.
Profile Management	Users can view and update their personal details, contact info, and more	M	Impacts data accuracy but does not directly affect core system functionality. Failures are inconvenient but not system breaking.

Pet Management	Pet owners can add, edit, or delete pets and record details such as name, breed, or notes	M	Important for booking accuracy but the system remains functional even if this feature experiences issues.
Service Management	Service providers can list, update, or delete offered services and set pricing	H	Core function for service providers. Any failure prevents new services from being displayed to users.
Search and Filter Services	Users can browse or search all available services and apply filters	H	Central to the user experience. Failure prevents users from discovering or accessing services, making the platform unusable.
Booking Management	Pet owners can create, or cancel bookings Service providers can accept and manage bookings	H	Core transactional flow of the system. Failures directly affect platform usability for both user roles.
Payment	Users can complete payments securely using PayPal sandbox integration	H	High-risk feature involving financial transactions. Errors could lead to data loss, failed payments, or mistrust in the platform.

4.2 ENHANCEMENT FEATURES

Feature	Description	Priority	Justification
Review and Rating	Pet owners can submit, edit, or delete reviews and rate service providers after a completed booking	M	Builds platform credibility and trust. However, its absence does not prevent primary operations such as booking or payment.
Location	Integrates Google Maps API to display service providers on a map and enable distance-based	M	Improves discovery convenience but is not critical to system functionalities. Failure

	filtering of the search results		affects usability, not system integrity.
Notifications and Email Confirmations	Sends automated confirmation and update emails for bookings, payments, and cancellations	L	Provides reassurance and clarity to users but does not hinder the ability to complete transactions. Failures mainly affect communication quality.
UI/QoL Improvements	Includes visual refinements, accessibility updates, and usability enhancements such as responsive design and improved navigation	L	Enhances the user experience but does not affect underlying functionality or data integrity.
System Performance Optimization	Backend and database improvements to provide users reduced loading times and improve responsiveness under higher loads	L	Important for efficiency and perceived quality. However, the system remains fully functional even without performance optimization.

5 FEATURES NOT TO BE TESTED

All features implemented and accessible in the *PawfectMatch* system are required to be tested as part of this release. No user-facing functions are excluded from the testing scope.

Testing will cover every feature available in the deployed application, including all core and enhancement functionalities identified in section 4. This ensures complete end-to-end verification of user workflows across both the pet owner and service provider roles.

6 APPROACH (STRATEGY)

The overall testing strategy for *PawfectMatch* ensures that all user-accessible features and functions are verified against the system's functional and non-functional requirements.

Testing will be conducted using a multi-level, incremental approach. Each level of testing aims to progressively validate the correctness, integration, and usability of the system from both the developer and user perspective.

6.1 TESTING LEVELS

Test	Description	Purpose	Done by
Unit Testing	Conducted using FastAPI's built-in test framework and React's Jest utilities to confirm that individual functions and components behave correctly	Verify individual functions or components within the frontend and backend	Developers
Integration Testing	Focuses on data flow between system components and external APIs Mock testing is used to simulate API responses	Ensure correct interaction between modules	Developers QA Team
System Testing	Executed on the main branch after each merge to confirm all workflows operate correctly end-to-end	Validate the entire application as a complete system against SRS defined requirements	QA Team
User Acceptance Testing	Team members simulate real user behavior to confirm usability, reliability, and completeness before release	Verify that the system meets user expectations and business goals	QA Manager/ Engineer Project Team

6.2 TESTING METHODS AND TECHNIQUES

Testing primarily employs black-box testing techniques, focusing on verifying that system inputs produce correct outputs without reference to internal code logic.

Additional testing methods include:

- Boundary Value and Equivalence Partitioning for input validation
- Error Guessing to explore likely user errors and exception scenarios
- Regression Testing after each release to ensure new enhancements do not break existing workflows
- Exploratory Testing to identify usability or behavioral issues not covered in formal test cases

6.3 TOOLS AND CONFIGURATION MANAGEMENT

Category	Tool	Purpose
Test Management	Test Execution Report	Record and track execution of test cases and results
Defect Tracking	GitHub Issues/JIRA	Log and manage test failures or bugs
API Testing	Postman	Validate API endpoints and responses
Database Validation	pgAdmin	Confirm backend data consistency and integrity after test actions
Browser Platform	Chrome, Firefox, and Safari	Supported browsers for UI and system testing
Version Control and Configuration	GitHub	Manage environment consistency and deployment tags

All configurations comply with the Configuration Management Plan to maintain version consistency across environments.

6.4 REGRESSION TESTING APPROACH

Regression testing will be conducted after each release cycle and before subsequent deployments. High risk modules receive priority during regression testing.

If critical defects are discovered, a full regression of related modules is performed. If minor defects are found, targeted regression testing is executed, and a cumulative regression suite is maintained throughout all releases.

6.5 METRICS AND REPORTING

Metric	Purpose	Collection Level
Test Case Execution Rate	Ratio of executed to total planned test cases	System/UAT
Defect Density	Number of defects per module or release	Integration/System
Defect Resolution Time	Time between defect identification and closure	System/Regression
Pass Rate	Percentage of successfully passed test cases	All levels
Requirements Coverage	Percentage of tested VS total SRS requirements	System

Metrics will be reviewed weekly by the QA Manager and summarized in the Test Execution Report.

6.6 HANDLING AMBIGUOUS REQUIREMENTS

Any requirement that cannot be clearly tested will be raised to the Project Manager (PM) and QA Manager for clarification. Ambiguities will also be documented in the Test Summary Report if unresolved prior to testing.

6.7 ORGANIZATIONAL PROCESSES AND REVIEWS

- **Test Reviews:** Occurs during weekly team meetings, and led by the QA Manager
- **Communication Channels:** JIRA and GitHub for task tracking with Telegram for coordination
- **Version Control:** All test artifacts are versioned and stored in the project repository
- **Status Updates:** QA Manager reports progress and defect summaries to the Project Manager weekly

6.8 SPECIAL TESTING CONSIDERATIONS

- All core and enhancement features will be tested as integrated workflows
- Testing will be performed in a standard local environment

- Mobile or device-specific testing is out of scope
- No specialized training is required to execute the tests

6.9 AUTOMATED TESTING SUPPORT

Automated testing complements manual testing to maintain continuous system reliability. Automation is integrated into the development workflow and CI/CD pipeline as follows.

Automated Test	**Scope**	**Purpose**
Backend Unit Tests	Implemented using pytest	Validate core business logic
End-to-end (E2E) Tests	Main flows for each page developed, executed via Playwright	Simulate full user interactions
CI/CD Integration	GitHub Actions pipeline executes all automated tests on every Pull Request created	Ensures only stable, verified builds are merged into the main branch

7 ITEM PASS/FAIL CRITERIA

This section defines the conditions under which individual test cases, features, and the overall *PawfectMatch* system are considered to have passed, failed, or are deferred for later evaluation. The criteria ensures consistent evaluation of testing outcomes and defines what constitutes satisfactory completion of this Test Plan.

7.1 INDIVIDUAL TEST CASE CRITERIA

Status	**Definition**
Pass	The actual result matches the expected result for all test steps with no major or minor deviations observed
Fail	The actual result does not match the expected result, or the system exhibits incorrect behavior, data loss, or errors
Blocked	The test could not be executed due to unresolved dependencies, environment issues, or other defects
Deferred	The test is postponed due to dependency on a future feature, version, or non-critical defect

All failed and blocked test cases will be logged in our Issue Tracking System (ITS), GitHub Issues or JIRA. It will reference the relevant Test Case ID, SRS requirement, and defect severity.

7.2 DEFECT SEVERITY AND ACCEPTANCE THRESHOLDS

Severity	**Definition**	**Acceptance Criteria**
Critical	Causes complete system failure, data corruption, or prevents users from completing core workflows	Must be resolved before release
Major	Significantly impairs a major feature or module but does not cause total failure	Must be resolved before release
Moderate	Affects secondary functionalities or workflows but has a workaround	May be accepted if mitigation is documented and approved
Minor	Minor UI inconsistencies, non-functional errors, or low visibility defects	May remain unresolved if non-disruptive and documented in the final report

7.3 OVERALL COMPLETION CRITERIA

Testing for *PawfectMatch* will be considered complete when all of the following conditions have been met:

1. **Test Execution Completion**
   - 100% of test cases have been executed
   - All high priority and high risk modules have achieved full pass status
2. **Defect Resolution**
   - 95% or more of total defects have been resolved and verified
   - All critical and major defects are fixed and retested successfully
   - Any remaining minor defects are documented with acceptable workarounds that are approved by the QA Manager and Project Manager
3. **Regression Testing**
   - Full regression suite executed after every major code change or deployment
   - No new critical or high severity defects introduced by fixes
4. **Coverage and Traceability**
   - 100% of SRS requirements have corresponding test cases
   - All test results are traceable to requirements and versioned test artifacts in GitHub
5. **Stability of Build**
   - The final build passes all smoke and sanity tests
   - The system remains stable under typical user operations without data loss or crashes
6. **Documentation and Reporting**
   - Test Execution Report and defect log are complete, reviewed, and approved
   - Final test summary report submitted and approved by QA Manager and Project Manager

8 SUSPENSION CRITERIA AND RESUMPTION REQUIREMENTS

Testing may need to be suspended when conditions prevent valid or productive test execution. The following sections define the criteria under which testing will be paused and the requirements that must be met before testing can resume.

These rules apply across all testing levels for the *PawfectMatch* system.

8.1 SUSPENSION CRITERIA

Testing will be suspended under any of the following conditions:

1. **Critical Build Failure**
   - The deployed build cannot be launched or accessed due to fatal server or frontend errors
   - Application crashes prevent navigation to core workflows
2. **Blocking Defects Identified**
   - One or more critical or major defects block the execution of dependent test cases
3. **Environment/Data Unavailability**
   - The test database, API endpoints, or external dependencies are unavailable or unstable
   - Test data corruption occurs, leading to invalid results
4. **Configuration/Deployment Errors**
   - The system under test is not synchronized with the correct database schema or configuration version
   - CI/CD deployment or version mismatch prevents accurate testing

If testing is suspended, the QA Manager will immediately log the incident in GitHub Issues, classify it as a blocker, and notify the Project Manager.

8.2 RESUMPTION REQUIREMENTS

Testing will resume only when the following conditions are met:

1. **Blocking Defects Resolved**

   - All blocking critical or major defects have been fixed, verified, and retested
   - Regression testing confirms that the fix did not introduce new issues

2. **Environment Restored**
   - The application, database, and external APIs are accessible and functioning normally
   - Configuration and version alignment verified via GitHub version tags and migration scripts
3. **Validated Build Available**
   - A new, stable build has passed initial smoke and sanity tests
   - QA Manager confirms readiness to proceed with system level test execution
4. **Formal Approval**
   - QA Manager documents the suspension resolution and formally approves test resumption

9 TEST DELIVERABLES

The following test artifacts will be produced and maintained for *PawfectMatch*. These deliverables support requirement coverage and release readiness.

Deliverables	**Description**
Test Plan	Overall test strategy, objectives, schedules, environments and responsibilities across unit, integration, system and UAT.
Test Cases	Detailed scenarios derived from the Software Requirement Specifications (SRS) including inputs, expected results and pass/fail criteria
Test Coverage Report	Maps each test case to its corresponding requirement to ensure all functionalities have been tested and verified for completeness
Test Data Files	Mock datasets created to simulate realistic user and service data (e.g. sample accounts, pets, bookings and payment) for test execution
Test Summary Report	Provides an overview of overall test results, coverage metrics and system readiness for release, summarizing key findings and outcomes

10 ENVIRONMENTAL NEEDS

The *PawfectMatch* testing environment mirrors the project's development setup to ensure consistency and accurate test results. All tools, software and services are configured according to the Configuration Management Plan

Category	Description
Hardware	Standard developer laptops running Windows 10/11, macOS or Linux with minimum of 8 GB RAM and stable internet connectivity. No dedicated servers are required; local environments replicate deployment conditions
Software	Node.js + npm (React + TypeScript frontend), Python 3.11 + FastAPI (backend_ and PostgreSQL (database). Supporting tools include Git (for version control), Visual Studio Code, Postman and pgAdmin
APIs	Google OAuth 2.0 for authentication, Google Maps API for location and distance filtering
Network	Stable internet connectivity for accessing external APIs and GitHub repository. CI/CD operations are managed through GitHub Actions within a secure local environment.

11 RESPONSIBILITIES

Testing responsibilities are distributed among the project team members as outlined below. Each role contributes to ensuring that all test activities are properly planned, executed and reported to maintain software quality and release readiness.

Role	Responsibilities
Project Manager	Provides overall direction and approval for testing activities. Ensures testing aligns with project objectives, reviews major test deliverables and approves final reports before release. Facilitates communication between QA team and other stakeholders.
QA Manager	Overseas test planning and execution. Defines the testing strategy, monitors progress, reviews reports and ensures compliance with quality standards. Approves the final Test Summary Report and sign-off for release readiness.
QA Engineer	Coordinates and executes system regression and UAT testing. Prepares test data, executes test cases, records results and reports defects to developers. Supports the QA Manager in maintaining traceability and coverage.
Release Engineer	Ensures build integrity and version control consistency. Verifies that tested builds are properly tagged, packaged and deployed according to Release Plan. Coordinates with QA and developers during pre- and post-release testing.
Developers	Conduct unit and integration testing of assigned modules. Resolve defects identified during testing, verify fixes and ensure regression stability before merging into the main branch. Handle both frontend and backend testing responsibilities as required.

12 STAFFING AND TRAINING NEEDS

The existing project team will perform all testing activities. No additional staff or external testers are required.

All team members are familiar with the *PawfectMatch* system architecture, modules and testing tools due to their involvement in system development.

Area	**Description**
Staffing	The QA Manager, QA Engineer, Release Engineer and two Developers form the core testing team. The Release Engineer supports build verification, deployment validation and version control consistency. Additional team members may assist with test case preparation and execution under QA supervision.
Training	No formal training required. All testers are familiar with the system's modules, workflows and tools such as GitHub, Postman, pgAdmin. Automated testing using Playwright has already been integrated into the workflow.
Tool Setup	All testing environments and tools have been configured according to the Configuration Management Plan. Each tester has access to the GitHub repository, environment variables and database instances required for testing.

13 SCHEDULE

The Testing phase for PawfectMatch development will take place once the major features are implemented. The Table below is a summary of the schedule for the testing phase.

Task	Estimated Time Taken	Start Date
Produce Test plan	5 days	10 October 2025
Unit Test	14 days	15 October 2025
Integration Test	14 days	15 October 2025
Produce Test coverage report	7 days	29 October 2025

14 RISKS AND CONTINGENCIES

The risk and responses to mitigate them are specified in the table.

Risk	Contingency
Lack of manpower	In the event where QA personnel are unavailable or requires additional manpower, Team members from development and management will step in to assist QA with the testing.
Lack of training for testers	The QA Manager and Lead Developer will ensure that adequate training and clear documentation will be provided to all testers to perform effective testing.
Time constraint	In the event where the testing phase is shortened due to the delay in the earlier phases, The QA team will reprioritize the test cases as deemed appropriate.

15 APPROVALS

Signature:	[illegible]	Date:	3/11/2025
Print Name:	Chan Zi Jian		
Title	Mr.		
Role	Project Manager		

Signature:	nelly.	Date:	3/11/2025
Print Name:	Nelly Nurelda Binte Zulkiflee		
Title	Ms.		
Role	QA Manager		

Signature:	[illegible]	Date:	1/11/2025
Print Name:	Chloie Tan Yue Yun		
Title	Ms.		
Role	Release Engineer		

Signature:	[illegible]	Date:	3/11/2025
Print Name:	Athena Choo		
Title	Ms.		
Role	QA Engineer		

REFERENCES

Referenced Document	Link
Project Plan	https://github.com/softwarelab3/3040-TEL2-KK/blob/main/Documentation/Project%20Plan.pdf
Configuration Management Plan	https://github.com/softwarelab3/3040-TEL2-KK/blob/main/Documentation/Configuration%20Management%20Plan.pdf
Release Plan	https://github.com/softwarelab3/3040-TEL2-KK/blob/main/Documentation/Release%20Plan.pdf