PawfectMatch
PROJECT PLAN

Version 1.2

14/10/2025

VERSION HISTORY

Version #	Implemented By	Revision Date	Approved By	Approval Date	Reason
1.0	Athena	24/09/2025	Nelly	24/09/2025	Initial draft of Project Plan
1.1	Nelly	5/10/2025	Chloie	6/10/2025	Added sections 7 to 10
1.2	Zi Jian	13/10/2025	Chong Yao	14/10/2025	Final Project Plan

TABLE OF CONTENTS

1 INTRODUCTION

1.1 PROJECT OVERVIEW

The PawfectMatch project is a web-based platform designed to connect pet owners with trusted pet service providers, such as groomers, pet walkers, pet sitters, and other pet care services. The platform enables pet owners to easily discover and book services, while also allowing them to leave reviews and rate the providers of booked services.

Service providers, on the other hand, can showcase their services, manage bookings, and build credibility with potential customers through customer feedback. Key features include search, booking, user authentication, and profile management. The platform will be developed using a React frontend, FastAPI backend, and PostgreSQL database.

1.2 PROJECT DESCRIPTION AND SCOPE

The PawfectMatch platform is aimed at addressing the growing demand for reliable and accessible pet care services in Singapore. By centralizing services and streamlining the booking process, the platform will simplify the experience for pet owners and enhance operational efficiency for service providers. The project will address common challenges faced by both pet owners and pet service providers:

- Pet owners often face difficulties finding reliable and convenient pet care services, and need a centralized platform to search, compare, book, and rate services offered by various providers
- Service providers lack a platform to manage bookings, process payments, and establish trust with customers through reviews

This project involves the development of the following core features:

- User authentication and onboarding for both pet owners and pet service providers with Google OAuth 2.0 integration
- Pet profile management allowing pet owners to create and update pet profiles
- Service listing allowing service providers to list their services, set availabilities, and offer pricing
- Booking system for pet owners to book and schedule services, with payments processed through PayPal
- Review system for pet owners to review and rate booked services

The project's initial scope includes core functionalities such as user authentication, pet profile management, service listings, a booking system, and a review system. Key third-party integrations will be incorporated for secure payment processing using PayPal and user authentication using Google OAuth 2.0.

The project scope does not include advanced features such as loyalty programs or advanced data analytics for service providers. These features may be considered in the future, but the initial development will focus on building and refining the core functionalities to ensure the platform is user-friendly and scalable.

2 PROJECT ORGANISATION

2.1 TEAM STRUCTURE

The PawfectMatch project will be driven by a cross-functional team composed of a project manager, developers, and quality assurance personnel. The team will collaborate using Agile practices, iterating over sprints to deliver the core features of the project.

The responsibilities of each role will be discussed in more detail in the next section. The following is a list of roles and the corresponding project team member(s):

- **Project Manager:** Zi Jian
- **Lead Developer:** Chloie
- **Frontend Developer:** Quang, Anthony
- **Backend Developer:** Chong Yao, Anthony
- **QA Manager:** Nelly
- **QA Engineer:** Athena
- **Release Engineer:** Chloie

2.2 ROLES AND RESPONSIBILITIES

Project Manager: Zi Jian

Oversees the overall progress of the PawfectMatch project. The main tasks include:

- Overseeing the project progress and ensuring that deliverables are met on time
- Approving and executing the project plan, ensuring alignment with the project scope, requirements, and schedule
- Assigning tasks to team members and ensuring the timely executing of tasks
- Managing and motivating the team to ensure a positive and productive environment
- Representing the team to stakeholders and ensuring clear communication with external parties

Lead Developer: Chloie

Oversees the overall architecture, and ensures scalability and maintainability of the system. This includes:

- Setting up the system architecture
- Leading the development of both frontend and backend
- Collaborating with other developers to integrate backend systems with frontend components
- Coordinating development efforts across the team to ensure timely completion of features and bug fixes

Frontend Developer: Quang, Anthony

Focuses on the frontend development of the platform. The responsibilities include:

- Designing and implementing UI components using React and TailwindCSS
- Integrating the frontend with backend APIs for dynamic data handling
- Ensuring the frontend is user-friendly, accessible, and performs well across all devices
- Writing automated test cases for frontend components and pages
- Testing and debugging frontend components to ensure reliability and quality

Backend Developer: Chong Yao, Anthony

Develops the backend API services and the database. This includes:

- Designing and implementing RESTful APIs using FastAPI for system functionalities
- Ensuring the database schema is updated and optimized for performance
- Implementing role-based access control to differentiate between pet-owners and service providers
- Handling server-side validation and error handling
- Collaborating with frontend developers to ensure smooth integration between frontend and backend services
- Writing automated unit tests for backend APIs/functions

QA Manager: Nelly

Ensures overall quality of the application through testing and validation. This includes:

- Defining the test strategy and plans for both manual and automated testing

- Ensuring test coverage is complete for all critical features
- Tracking bugs and ensuring that they are reported, prioritized and fixed
- Ensuring that QA standards are consistently followed throughout development

QA Engineer: Athena

Focuses on executing test cases and verifying the platform's functionality. Responsibilities include:

- Writing and executing manual test cases for core features
- Reporting bugs and verifying bug fixes
- Collaborating with developers to ensure that UI/UX meets quality standards and is free from bugs

Release Engineer: Chloie

Manages the release process and ensures the platform is properly maintained. This includes:

- Overseeing the CI/CD pipeline, ensuring automated tests are executed without any errors
- Ensuring proper version control and rollback mechanisms are in place
- Maintaining development documentation
- Coordinating with developers to ensure that new features are properly integrated into the release cycle

2.3 TEAM COLLABORATION AND COMMUNICATION

The team will collaborate using the following tools:

- **GitHub:** Version control and code reviews
- **Jira:** Track project tasks, bugs, and progress
- **Google Drive:** Shared documentation and project files
- **Figma:** Design wireframes and UI prototypes

Weekly stand-up meetings will be held to track progress, identify blockers, and ensure clear communication across the team. Direct communication between project team members will be done through telegram, and the team will split up into subgroups as necessary to work more co-operatively on specific problems.

3 PROCESS DEFINITION

3.1 LIFECYCLE MODEL

The PawfectMatch project adopts an Agile Incremental Development Model. This approach emphasizes flexibility, iterative progress, and continuous integration of feedback throughout the development lifecycle. Given the project's estimated schedule, iterative sprints allow the team to progressively deliver functional requirements, and adjust to evolving designs or technical challenges. Each sprint includes activities such as design, implementation, testing, and review.

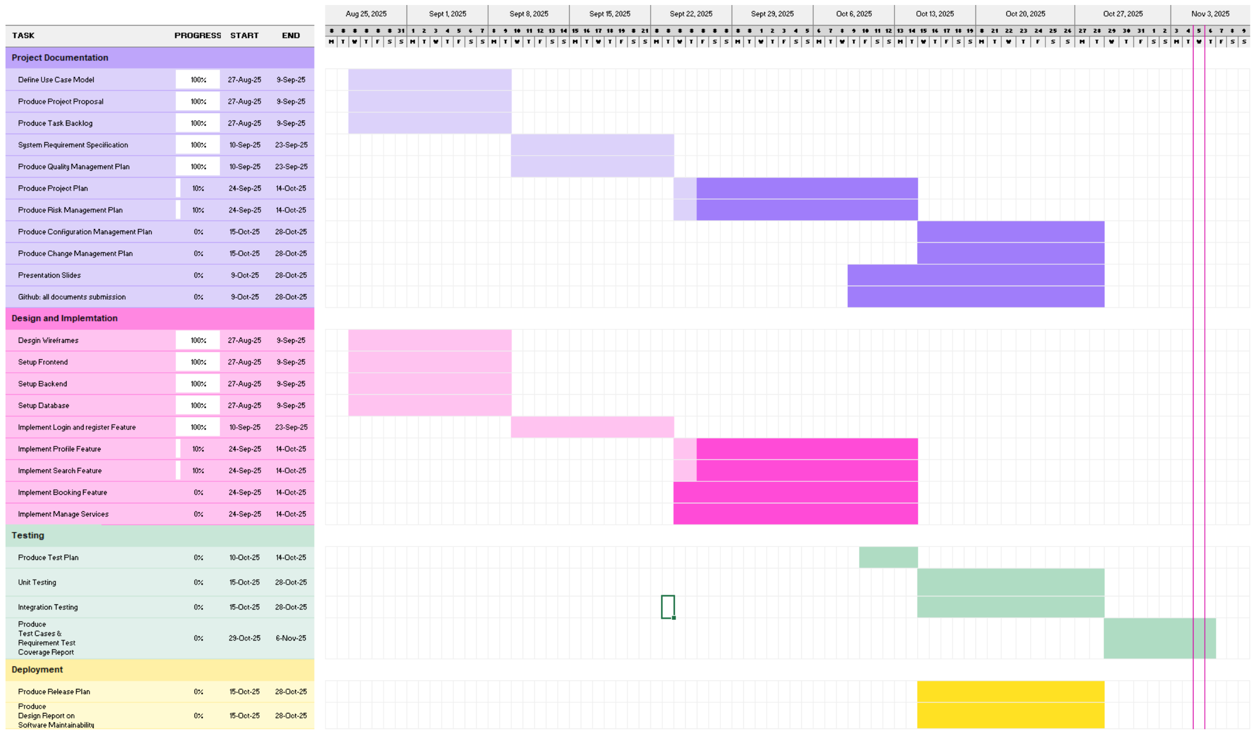
Unlike the traditional Waterfall model, the incremental approach ensures that each core functionality can be tested, reviewed, and refined independently before moving onto the next. The chosen model fits the project's scope, shorter timeline, and need for continuous collaboration between the project team's substeams.

The lifecycle stages are summarized as follows:

1. Project Proposal/Requirements
2. Software Quality Design
3. Implementation Phase 1
4. Implementation Phase 2
5. Project Configuration
6. Software Testing

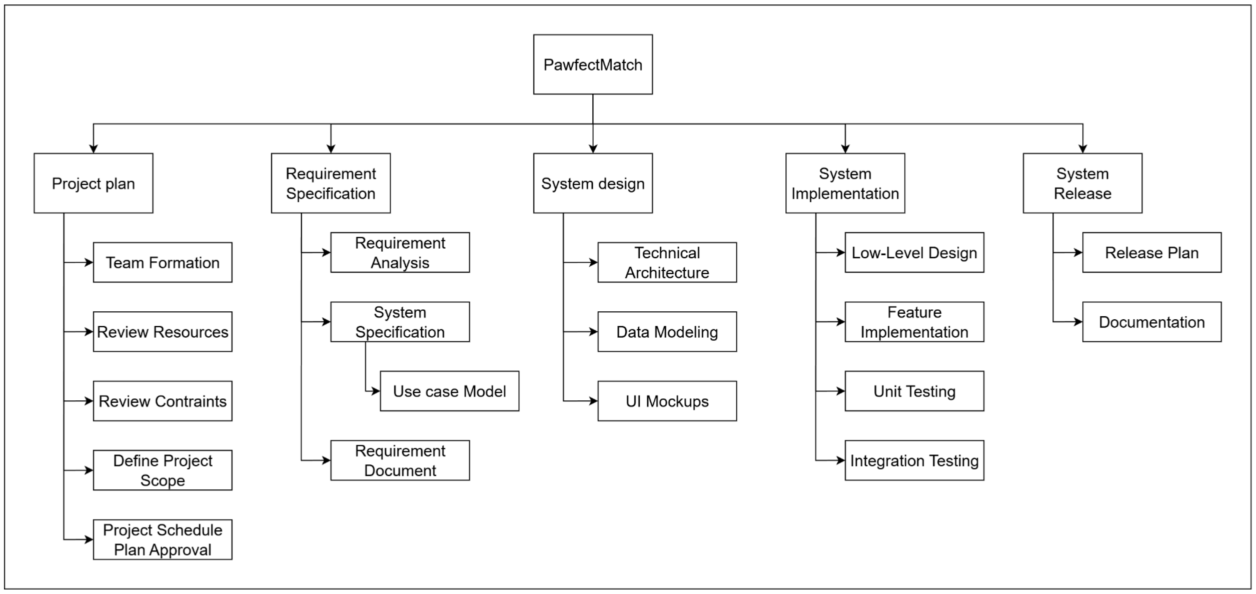
4 SCHEDULE

4.1 ACTIVITY DEPENDENCIES AND SCHEDULE



TASK	PROGRESS	START	END
Project Documentation			
Define Use Case Model	100%	27-Aug-25	9-Sep-25
Produce Project Proposal	100%	27-Aug-25	9-Sep-25
Produce Task Backlog	100%	27-Aug-25	9-Sep-25
System Requirement Specification	100%	10-Sep-25	23-Sep-25
Produce Quality Management Plan	100%	10-Sep-25	23-Sep-25
Produce Project Plan	10%	24-Sep-25	14-Oct-25
Produce Risk Management Plan	10%	24-Sep-25	14-Oct-25
Produce Configuration Management Plan	0%	15-Oct-25	28-Oct-25
Produce Change Management Plan	0%	15-Oct-25	28-Oct-25
Presentation Slides	0%	9-Oct-25	28-Oct-25
Github: all documents submission	0%	9-Oct-25	28-Oct-25
Design and Implemtation			
Desgin Wireframes	100%	27-Aug-25	9-Sep-25
Setup Frontend	100%	27-Aug-25	9-Sep-25
Setup Backend	100%	27-Aug-25	9-Sep-25
Setup Database	100%	27-Aug-25	9-Sep-25
Implement Login and register Feature	100%	10-Sep-25	23-Sep-25
Implement Profile Feature	10%	24-Sep-25	14-Oct-25
Implement Search Feature	10%	24-Sep-25	14-Oct-25
Implement Booking Feature	0%	24-Sep-25	14-Oct-25
Implement Manage Services	0%	24-Sep-25	14-Oct-25
Testing			
Produce Test Plan	0%	10-Oct-25	14-Oct-25
Unit Testing	0%	15-Oct-25	28-Oct-25
Integration Testing	0%	15-Oct-25	28-Oct-25
Produce Test Cases & Requirement Test Coverage Report	0%	29-Oct-25	6-Nov-25
Deployment			
Produce Release Plan	0%	15-Oct-25	28-Oct-25
Produce Design Report on Software Maintainability	0%	15-Oct-25	28-Oct-25

4.2 WORK BREAKDOWN STRUCTURE



4.3 WORK PACKAGES

The entire project work is broken down by the important phases of the software development life cycle. They include the following:

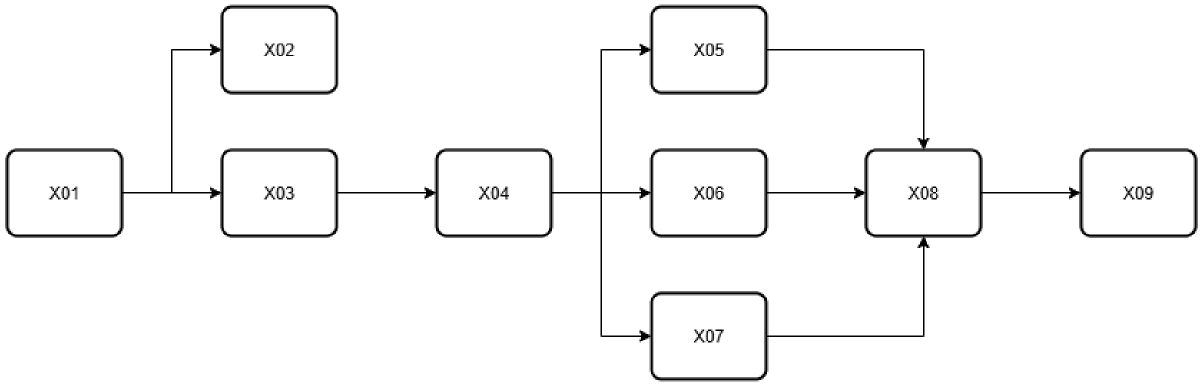
1. Project Proposal
2. Project Plan
3. Requirement Elicitation
4. System Requirement Specification
5. Technical Architecture
6. Data Modelling
7. User Interface Design
8. Coding and Unit Testing
9. Integration and Quality Assurance

4.4 ACTIVITY DEPENDENCIES

The following table describes the dependencies of the deliverable work packages:

Work Package #	Work Package Description	Duration	Dependencies
X01	Project Proposal	7 days	--
X02	Project Plan	7 days	X01
X03	Requirement Elicitation	4 days	X01
X04	System Requirement Specification	14 days	X03
X05	User Interface	7 days	X04
X06	Technical Architecture	7 days	X04
X07	Data Modeling	7 days	X04
X08	Coding & Unit Testing	30 days	X05,X06,X07
X09	Integration & System Testing	14 days	X08

The following Activity Diagram describe the above in more graphical detail:



4.5 WORK PACKAGE DETAILS

Work Packages are listed below. A team member, indicated in bold, has been assigned as primarily responsible for each work package and will coordinate that package.

Work Package	X01 - Project Proposal
Assigned to	Athena, Nelly, Chloie, Zi Jian
Effort	7 Days
Start Date	27 August 2025
Purpose	To outline the problem, objectives, and planned solution, providing a clear framework for evaluation and approval.
Inputs	None
Activities	This work package includes providing a brief overview of the project, its objectives, and a set of proposed project deliverables throughout the development of the software cycle. The people responsible for this work package will also be transcribing ideas brought up in the group meeting discussion into a formal report
Outputs	A written document of Project Proposal

Work Package	X02 - Project Plan

Assigned to	Athena, Nelly, Chloie, Zi Jian
Effort	7 Days
Start Date	24 September 2025
Purpose	To determine an introductory overview of the project, to be refined in later work packages.
Inputs	None
Activities	This work package provides an overview of software estimations for better resource and task allocation planning.
Outputs	A written document of Project Plan

Work Package	X03 - Requirement Elicitation
Assigned to	Anthony, Zi Jian
Effort	4 Days
Start Date	10 September 2025
Purpose	To identify, gather, and understand stakeholders' needs and expectations to ensure the system or product meets its intended goals.
Inputs	Customer's Requirements
Activities	The project manager will hold a meeting with the client to determine the requirements of the system.
Outputs	A written document of Requirement Elicitation

Work Package	X04 - System Requirement Specification
Assigned to	Anthony, Zi Jian
Effort	14 Days
Start Date	10 September 2025

Purpose	To provide a clear, detailed description of a system's requirements, ensuring all stakeholders share a common understanding before development begins.
Inputs	X03 - Requirements Elicitation
Activities	This work package will include analysing the requirements and feedback from package X03 to produce the System Requirement Specification
Outputs	A written document of System Requirement Specification

Work Package	X05 - Technical Architecture
Assigned to	Anthony, Chloie, Chong yao, Quang, Zi Jian
Effort	7 Days
Start Date	17 September 2025
Purpose	To define the system's structure, components, and interactions, providing a blueprint that guides development, ensures scalability, and aligns technology with business goals.
Inputs	X04 - System Requirement Specification
Activities	High level design entails defining the architecture of the software system and identifying the various components and how they are inter-related to and interactive with each other. The developers will decide on the software infrastructure to use such as language used to implement the software and tools/platform to be used. The developers and Project manager will discuss and produce a System Architecture Diagram to visualise the relationship between the components of the software.
Outputs	A written document of System Architecture Diagram

Work Package	X06 - Data Modelling

Assigned to	Anthony, Chloie, Chong yao, Quang
Effort	7 days
Start Date	17 September 2025
Purpose	To visually represent and organize data structures and relationships, ensuring clarity, consistency, and efficiency in how data is stored, accessed, and managed.
Inputs	X04 - System Requirement Specification
Activities	The development team will perform analysis on data flow and entity relationships. Considerations for data types and tables that are required to represent stored data in the system will be decided.
Outputs	A written document of Data Modelling

Work Package	X07 - User Interface Design
Assigned to	Anthony, Chloie, Quang
Effort	7 Days
Start Date	17 September 2025
Purpose	To create a visually clear, intuitive, and user-friendly layout that enhances usability, accessibility, and overall user experience when interacting with a system.
Inputs	X04 - System Requirement Specification
Activities	This work package includes analysing the requirements and translating it into the User Interface design. The UI mockup will be produced.
Outputs	UI Mockup and Prototype

Work Package	X08 - Coding and Unit Testing
Assigned to	Anthony,Chloie, Chong Yao, Quang, Athena, Nelly

Effort	30 Days
Start Date	10 September 2025(experimental), 24 September (Finalised)
Purpose	To implement the system's functionality in code and verify each component individually, ensuring correctness, reliability, and early detection of defects.
Inputs	X05 - Technical Architecture, X06 - Data Modelling, X07 - User Interface Design
Activities	This work package includes implementing all the features according to the software specification. The QA team will also produce a Test Plan and perform unit testing to ensure all functions are working. Unit Test will also be done after each feature implementation by developers and be included in the test plan at a later stage.
Outputs	Source code and Header files, Test Plan

Work Package	X09 - Integration & System Testing
Assigned to	Athena, Nelly, Anthony, Chloie, Chong Yao, Quang,
Effort	14 Days
Start Date	15 October 2025
Purpose	To implement the system's functionality in code and verify each component individually, ensuring correctness, reliability, and early detection of defects.
Inputs	X08 - Coding and Unit Testing
Activities	The development and QA team will simulate how a user might interact with the system and will also examine issues such as system performance and integrity. The metrics used during the performance and integrity test will then be referenced to further improve the system
Outputs	A written document of the Test Report and Quality Assurance Report.

5 PROJECT ESTIMATES

5.1 CODE SIZE ESTIMATION USING FUNCTION POINTS

5.1.1 Unadjusted Function Points

The measure of unadjusted function points is based on IFPUG component types, namely External Inputs, External Outputs, External Inquiries, Internal Logic Files and External Interface Files. The File Type References (FTR) or Record Element Type (RET), and Data Element Type (DET) is counted and used to derive the complexity of each function, before the standard IFPUG weights are applied to obtain its Function Points (FP).

The key modules of the PawfectMatch project include Login, Register, Profile Creation, Manage Booking, Payment, and Search.

External Inputs (EI)

Function	FTR	DET	Complexity	FP
Login	1	2	Low	3
Google OAuth Login	2	3	Low	3
Register	2	3	Low	3
Onboarding (Pet Owner)	2	12	Average	4
Onboarding (Service Provider)	2	10	Average	4
Edit User Profile	1	7	Low	3
Add Pet	2	6	Average	4
Edit Pet	1	7	Low	3
Delete Pet	2	1	Low	3
Add Service	2	3	Low	3
Edit Service	1	4	Low	3
Delete Service	2	1	Low	3
Create Booking	3	3	Average	4
Edit Booking	1	3	Low	3

Delete Booking	3	1	Average	4
Submit Review	3	4	Average	4
Submit Payment	4	3	Average	4
EI Total				**58**

External Outputs (EO)

Function	FTR	DET	Complexity	FP
Booking Summary	3	~10	Average	5
Dashboard	4	~14	High	7
Review Confirmation	2	~6	Average	5
Error/Validation Messages	0	3	Low	4
EO Total				**21**

External Inquiries (EQ)

Function	FTR	DET	Complexity	FP
View Profile	2	10	Average	4
View Pets	2	7	Average	4
View Services	2	6	Average	4
View Bookings (Service Provider)	3	13	Average	4
View Bookings (Pet Owner)	3	13	Average	4
View Reviews	2	5	Low	3
Search/Filter Services	4	10	High	6
View Service Listing	2	6	Average	4
EQ Total				**33**

Internal Logic Files (ILF)

Function	RET	DET	Complexity	FP
Authentication	2	4	Low	7
User	1	9	Low	7
Pet	1	8	Low	7
Service	2	6	Low	7
Booking	2	13	Low	7
Payment	1	9	Low	7
Review	1	6	Low	7
Location	1	2	Low	7
ILF Total				**56**

External Interface Files (EIF)

Function	RET	DET	Complexity	FP
Google OAuth 2.0	1	6	Low	5
PayPal	1	8	Low	5
Google Maps/Location	1	10	Average	7
EIF Total				**17**

Summary

IFPUG weights: EI 3/4/6, EO 4/5/7, EQ 3/4/6, ILF 7/10/15, EIF 5/7/10

Component	Low	Average	High	Subtotal (FP)
External Inputs	10 × 3 = 30	7 × 4 = 28	0 × 6 = 0	58
External Outputs	1 × 4 = 4	2 × 5 = 10	1 × 7 = 7	21
External Inquiries	1 × 3 = 3	6 × 4 = 24	1 × 6 = 6	33
Internal Logic Files	8 × 7 = 56	0 × 10 = 0	0 × 15 = 0	56

External Interface Files	2 × 5 = 10	1 × 7 = 7	0 × 10 = 0	17
Total UFP				**185**

5.1.2 Adjusted Function Points

Influence Factors	Score	Details
Data Communications	3	Web-based system relies on continuous internet communication between client and backend
Distributed Processing	3	Frontend (React) and backend (FastAPI) run as separate distributed components
Performance	3	Response time or throughput is critical during all business hours. No special design for CPU utilization was required. Processing deadline requirements with interfacing systems are constraining
Heavily Used Configuration	2	No enterprise-level load expected but some security or timing considerations are included
Transaction Rate	3	Frequent small transactions
Online Data Entry	5	All core functions are real-time and interactive
End-User Efficiency	3	Designed for ease-of-use through simple forms and intuitive UI
Online Update	3	Online update of major internal logical files is included
Complex Processing	2	Contains business rules but no heavy computation required
Reusability	3	Modular React components and shared API services
Installation Ease	2	Simple container-based deployment (Docker)
Operational Ease	3	Automated startup, backup and monitoring to ensure stable operations
Multiple Sites	1	Primarily a single-site deployment

Facilitate Change	4	Codebase uses configuration files, REST APIs and modular design to support updates
Total Degree of Influence (DI)		**40**
Value Adjustment Factor (VAF)		DI × 0.01 + 0.65 = 40 × 0.01 + 0.65 = **1.05**
Adjusted Function Points (AFP)		UFP × VAF = 185 × 1.05 = **194.25**

Scoring (0 – 5)
0 = No influence
1 = Insignificant influence
2 = Moderate influence
3 = Average influence
4 = Significant influence
5 = Strong influence

5.1.3 Lines of Code

Lines of Code (LOC) estimation provides a quantitative basis for effort and schedule planning, and is derived by converting the Adjusted Function Points (AFP) into source lines of code using established industry conversion ratios.

The average LOC per Function Point (FP) used are based on empirical benchmarks reported by Caper Jones. According to the statistics, typical conversion values are: JavaScript/TypeScript ≈ 47–55 LOC/FP and Python ≈ 35–45 LOC/FP. Therefore, 50 LOC/FP for the TypeScript frontend and 40 LOC/FP for the Python backend were selected.

The project is expected to be user-interface-intensive, so the team assumes 55% of the total functionality will be allocated to the frontend and 45% to the backend.

Component	FP	LOC/FP	Estimated LOC
Frontend	106.8375	50	5341.875
Backend	87.4125	40	3496.5
Total Estimated LOC			**≈ 8839 LOC**

5.2 EFFORTS, DURATION AND TEAM SIZE ESTIMATION

To estimate the effort and duration required for the project, we use function points as the basis to calculate Effort, Duration, Team Size and finally the schedule. The estimates are expanded to account for project management and extra contingency time to obtain the total average effort estimates. From these averages, the duration of each work package in working days is estimated based on the following calculations.

Effort estimation in PawfectMatch follows a size-based approach, where the total estimated lines of code is converted into effort using an average production rate. The estimates are expanded to account for project management and extra contingency time to obtain the total average effort estimates.

For this project:

- Total Estimated Lines of Code: 8839 LOC
- Average Production Rate: 39 LOC per person-day (PD)
- Working Schedule: 5 days per week, 8 hours per day

Effort Estimation

The total development effort (person-days) is computed as:

$$Effort\ (PD) = \frac{Size\ (LOC)}{Production\ Rate\ (LOC/PD)} = \frac{8839\ LOC}{39\ LOC/PD} = 226.64\ PD$$

To account for project management, documentation, and contingency activities, an additional 15% buffer is added:

$$Total\ Effort\ (PD) = 226.64 \times 1.15 = 260.63\ PD$$

The total estimated effort is approximately 261 person-days.

Duration Estimation

The project duration (working days) can be estimated as:

$$Duration\ (Days) = 3 \times (Total\ Effort)^{1/3} = 3 \times (260.63)^{1/3} = 19.2\ days$$

$$Duration\ (Weeks) = \frac{19.2}{5} = 3.8\ weeks$$

The estimated development schedule for PawfectMatch is approximately 3.8 working weeks.

Team Size Estimation

The average team size required is computed as:

$$Team\ Size\ =\ \frac{Total\ Effort\ (PD)}{Duration\ (Days)}\ =\ \frac{260.63}{19.2}\ =\ 13.57\ persons$$

Rounded to the nearest whole number, an average of 14 developers would be required if all work were done concurrently over a 3.8-week period. However, in the PawfectMatch project team, there are only 7 members, so, the actual project duration is extended proportionally to reflect realistic weekly effort availability (e.g. 10 weeks).

Total Person-Hours

Assuming one working day is 8 hours:

$$Total\ Person\ Hours\ (PH)\ =\ 260.63\ \times\ 8\ =\ 2085\ hours$$

The total estimated development workload is ≈ 2085 person-hours.

5.2.1 Distribution of Effort

The total estimated effort of 261 person-days is distributed across the major project phases in proportion to the relative workload and complexity of each activity in the PawfectMatch development lifecycle.

Work Package/ Phase	Description	Effort (%)	Effort (PD)
Project Planning and Management	Preparation of project charter, scheduling, coordination of team tasks, and stakeholder communication	10%	26.1
Requirements Analysis	Collection of functional and non-functional requirements, drafting of user stories, and validation with stakeholders	10%	26.1
System and Architecture Design	Design of overall architecture, database schema, and API specifications	15%	39.15
Development	Development of React (TypeScript) frontend components and FastAPI (Python) backend services, including integration with PayPal and Google OAuth	35%	91.35

Testing and Quality Assurance	Unit, integration, and acceptance testing, defect fixing, and performance validation	15%	39.15
Deployment and Maintenance	Final deployment, configuration, documentation updates, and post-release maintenance support	5%	13.05
Contingency and Refinement Buffer	Allowance for risk mitigation, change requests, and unforeseen work	10%	26.1
Total		**100%**	**≈ 261 PD**

5.3 COST ESTIMATES

The cost estimation for PawfectMatch covers all tangible and human resources required for the successful completion of the project. The total cost comprises hardware, software, personnel, and miscellaneous operating expenses.

Hardware Costs

Item	Specifications	Quantity	Unit Cost	Total
Developer Laptops	Dell 16 Laptop - AMD Ryzen 7 8-core 5.1 GHz CPU, 16 GB RAM, 1 TB SSD	7	1300	9100.00
Total Hardware Cost				**9100.00**

Software Costs

Category	Tool/Platform	License Type	Cost
Development	Visual Studio Code, FastAPI, React, Node.js, Python	Open Source	0.00
Design	Figma, Draw.io	Free	0.00
Version Control	GitHib, Jira	Free	0.00
Documentation	Google Sheets/Docs	Free	0.00
Total Software Cost			**0.00**

Personnel Costs

Role	No. of Staff	Average Cost per Person	Total
Project Manager	1	15,000.00	15,000.00
Team Members	6	13,500.00	81,000.00
Total Personnel Cost			**96,000.00**

Total Cost = $105,100.00

6 PRODUCT CHECKLIST

Deliverables	Actual Dateline
Project Proposal	09 September 25 (Lab 2)
Use Case Model	09 September 25 (Lab 2)
System Requirement Specification	23 September 25 (Lab 3)
Quality Plan	23 September 25 (Lab 3)
Project Plan	14 October 25 (Lab 4)
Risk Management	14 October 25 (Lab 4)
Prototype Demo, Codebase, Documents, slides	14 October 25 (Lab 4)
Design Report on Software Maintainability	28 October 2025 (Lab 5)
Configuration Management Plan	28 October 2025 (Lab 5)
Change Management Plan	28 October 2025 (Lab 5)
Release Plan	28 October 2025 (Lab 5)
Presentation Slides	11 November 2025 (final submission)
Test Plan	11 November 2025 (final submission)
Test Cases and Requirement Test Coverage Report	11 November 2025 (final submission)
Github:All documents checked in	11 November 2025 (final submission)

7 BEST PRACTICE CHECKLIST

No.	Practice	
1	**Verify requirements.** Pay close attention to requirements and confirm that they are complete, clear and consistent. The SRS and Use Case Model together must form a full functional specification that accurately represents user and system behavior.	☐
2	**Keep it simple.** Design should remain simple and modular to manage complexity. Minimize dependencies between modules (frontend, backend, and database) and reduce unnecessary coordination by maintaining clear JIRA task assignments.	☐
3	**Focus on essentials.** Implement only critical features that directly support system objectives and avoid excessive or “nice-to-have” functions that add unnecessary scope.	☐
4	**Require visibility.** Maintain transparency of progress through up-to-date sprint boards, GitHub pull requests for peer review, and regular sprint reviews for design and implementation validation.	☐
5	**Control change.** Manage all documents, designs, and source files with version numbers, revision dates, and change logs. Revisions must be reviewed by the Project Manager, Release Engineer and Quality Assurance Manager before merging into the main branch.	☐
6	**Manage configuration.** Use GitHub as the central configuration management system to retain full version history of code and documents. Regularly update and clean repositories to stay consistent with the current project baseline.	☐
7	**Don’t underestimate.** Obtain realistic estimates for time, effort, integration, testing, and documentation. Track actual effort in JIRA and review discrepancies during sprint retrospectives to improve forecasting accuracy.	☐
8	**Review code.** Conduct mandatory peer code reviews before merging into the main branch to detect defects early and maintain code quality, readability, and consistency.	☐
9	**Test thoroughly.** Apply both black-box and white-box testing at unit, integration, and system levels to verify that each component and the overall platform meet all functional and non-functional requirements.	☐

8 RISK MANAGEMENT

Besides the general risk management activities described in the **Risk Management Plan**, the following key risks have been identified for the *PawfectMatch* project. These risks are continuously monitored throughout the project lifecycle, with mitigation and contingency actions taken as necessary.

Requirement changes during development

Impact Severity:	High
Probability:	25%
Impacts:	Changes introduced mid-development could require redesign or re-implementation of core modules such as booking or payment. This may delay subsequent tasks and increase testing workload.
Risk Reduction:	Validate requirements early and record formal change approvals before implementation.

Delay in requirement or deliverable completion

Impact Severity:	High
Probability:	20%
Impacts:	Late completion of major documents or features would shift the overall schedule and reduce testing time.
Risk Reduction:	Closely track progress using JIRA and conduct weekly sprint reviews to ensure tasks stay on schedule.

Underestimation of system complexity

Impact Severity:	Moderate
Probability:	30%
Impacts:	Additional effort may be needed to handle technical dependencies such as database relationships and API integration, potentially affecting deadlines.
Risk Reduction:	Re-evaluate estimates after each sprint and reassign resources where required.

Team member unavailability

Impact Severity:	Major
Probability:	10%
Impacts:	Absences due to illness or academic workload may cause delays or loss of task ownership.
Risk Reduction:	Maintain up-to-date documentation and cross-train members on key components to ensure continuity.

Coordination or communication issues within team

Impact Severity:	Moderate
Probability:	40%
Impacts:	Unclear communication could result in duplicated work, missed updates or inconsistent implementation across modules.
Risk Reduction:	Adhere to communication practices defined in the Project Management section, including weekly meetings and use of Telegram and JIRA for task tracking.

External dependency failure (e.g. Google OAuth 2 or hosting services)

Impact Severity:	Major
Probability:	10%
Impacts:	System access or authentication may fail temporarily, affecting testing or deployment.
Risk Reduction:	Implement local fallback authentication and maintain backup deployment options to minimize downtime.

9 QUALITY ASSURANCE

Quality assurance in the PawfectMatch project is achieved through strict adherence to the standards, review processes, and testing methodologies defined in the **Software Quality Assurance (SQA) Plan**.

The project team adopts a structured approach to quality management that integrates both preventive and corrective actions throughout the software development lifecycle. Specific test procedures and documentation details shall be provided in the **Test Plan** and **Test Case Report.**

The team will employ two key testing methodologies:

- **Unit Testing**
  Each system component is independently tested to verify correctness, reliability, and compliance with its design specification. Developers are responsible for creating and executing unit tests within the development environment before code is merged into the main branch.

- **System/End-to-end Testing**
  System-level tests are performed to confirm that the platform operates as a complete and coherent solution. These tests verify the accuracy of data flow, API communication, and interaction between frontend and backend components.

These methodologies will be used to validate two critical aspects of *PawfectMatch:*

- **System Functionality:** Confirm that all modules work together seamlessly to meet functional requirements defined in the Software Requirements Specification.

- **User Experience and Reliability:** Ensures that the platform performs consistently and intuitively under various operating conditions, including both typical and boundary scenarios.

The team will employ realistic test data that reflects actual usage scenarios, including service listing, bookings and payment flows. Extreme or boundary cases (such as invalid logins or unavailable booking slots) will also be used to verify that the system performs reliably under exceptional conditions.

10 MONITORING AND CONTROL

To ensure the *PawfectMatch* project stays on track and meets its objectives, several monitoring and control measures will be implemented throughout the development process.

Progress tracking and resource measurement: The team will monitor progress using JIRA, where each sprint’s tasks are logged, assigned and tracked. Comparing planned versus actual completion rates provides a clear view of progress and allows the team to identify any schedule slippage early.

Risk identification and management: Potential risks are recorded in the **Risk Management Plan**, together with mitigation and contingency measures. Regular reviews will be carried out by the Project Manager and QA Manager to ensure that risks are being managed effectively and that new ones are identified promptly.

Weekly progress review: Weekly sprint meetings will be conducted to review the overall status of design, development and testing activities. These meetings provide visibility into team performance, highlighting any issues affecting deliverables and allow corrective actions to be taken immediately.

Timeline and task breakdown: The Gantt chart defines the main milestones and deadlines for the project. Each major task has been broken down into smaller, measurable activities to make tracking more accurate and workload distribution more balanced. This structure supports detailed progress monitoring throughout each sprint cycle.

APPENDIX A: REFERENCES

Jones, C. (2017) *Applied Software Measurement: Global Analysis of Productivity and Quality (3rd ed.)*. McGraw-Hill Education. Retrieved 13 October, 2025 from https://www.accessengineeringlibrary.com/binary/mheaeworks/829ef30c60b20d92/96af26c048a067d5d3bec53ac2b2c7ddc143c27fbf02f78bc72c6894d93f431a/book-summary.pdf

International Function Points User Group (n.d.) *Function Point Analysis (FPA): Consistent and Stable Software Size*. IFPUG. Retrieved 13 October, 2025 from https://ifpug.org/ifpug-standards/fpa