PawfectMatch
RELEASE PLAN

Version 1.2

25/10/2025

APPROVALS

Submitting Organization's Approving Authority:

Signature	Printed Name	Date	Phone Number
	Chan Zi Jian	25/10/25	8255 2668

Project Manager

VERSION HISTORY

Version #	Implemented By	Revision Date	Approved By	Approval Date	Reason
1.0	Chloie	21/10/2025	Zi Jian	25/10/2025	Release Plan Template
1.1	Chloie	23/10/2025	Zi Jian	25/10/2025	Release Plan Draft
1.2	Chloie	25/10/2025	Zi Jian	25/10/2025	Final Release Plan

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

The Release Plan describes the planned deployment and incremental release strategy for the *PawfectMatch* system. The purpose of this document is to define the release strategy, schedule, impacts, and communication process governing the transition of *PawfectMatch* from its development phase to fully deployed versions. It outlines the sequence of planned releases, the functionality delivered at each stage, and the management activities required to ensure stable and verifiable deployments. This document serves as the guiding reference for the project team, teaching staff, and assessors to understand how each release contributes to the system's incremental evolution.

The scope of this Release Plan includes all releases from the baseline Core System (v1.0.0) to the final enhancement release Optimization and Quality of Life (v1.3.0). It focuses on system functionality, technical dependencies, deployment procedures, stakeholder communication, and risk mitigation related to each release. Activities described within this plan were developed collaboratively by the project team and are aligned with the academic project milestones and deliverable schedule.

The intended audience for this document includes the project supervisor (lab teaching assistant) overseeing the SC3040 project, as well as the project team members responsible for software deployment and documentation.

From a security and privacy perspective, the plan acknowledges that *PawfectMatch* operates in an academic environment using a sandbox deployment model. All external integrations, including Google OAuth 2.0 Sandbox, are implemented within controlled test environments to ensure user data protection and compliance with institutional data policies. No sensitive or production-level personal data is stored or processed in any phase of deployment.

2 REFERENCED DOCUMENTS

Document Name	Document Number	Issuance Date
System Requirements Specifications	SRS v1.0	23/09/25
Quality Plan	QP v1.0	22/09/25
Project Plan	PP v1.2	14/10/25
Risk Management Plan	RMP v1.2	12/10/25

Table 1. Referenced Documents

3 OVERVIEW

PawfectMatch is a full-stack web application designed to connect pet owners with service providers such as pet sitters, groomers, and trainers through a unified online platform. The system enables pet owners to create accounts, manage pet profiles, search for available services, make bookings, and complete payments with a secure and user-friendly environment. For service providers, the platform allows profile management, service listings, and review tracking to support transparent and trustworthy interactions between users.

PawfectMatch follows a MVC web architecture consisting of a React/TypeScript frontend, a FastAPI/Python backend, and a PostgreSQL relational database.

- **Frontend (Client):** Provides the user interface for pet owners and service providers to interact with the platform via web browsers
- **Backend (Application):** Implements business logic, authentication, and API endpoints, handling requests between the frontend and database
- **Database (Data):** Stores structured information about users, pets, services, bookings, and reviews in relational tables

The system interacts with several external services, including:

- **Google OAuth 2.0** for user authentication
- **Google Maps API** for location-based service discovery

These integrations extend the platform's capabilities while maintaining separation from core functionalities through well-defined interface contracts and dependency isolation.

The updated use case diagram in figure 1 illustrates how *PawfectMatch* interacts with its main users and external services. It contains the **Reviews** and **Location** modules that are to be developed in the enhancement phase, along with their integration into the existing architecture.

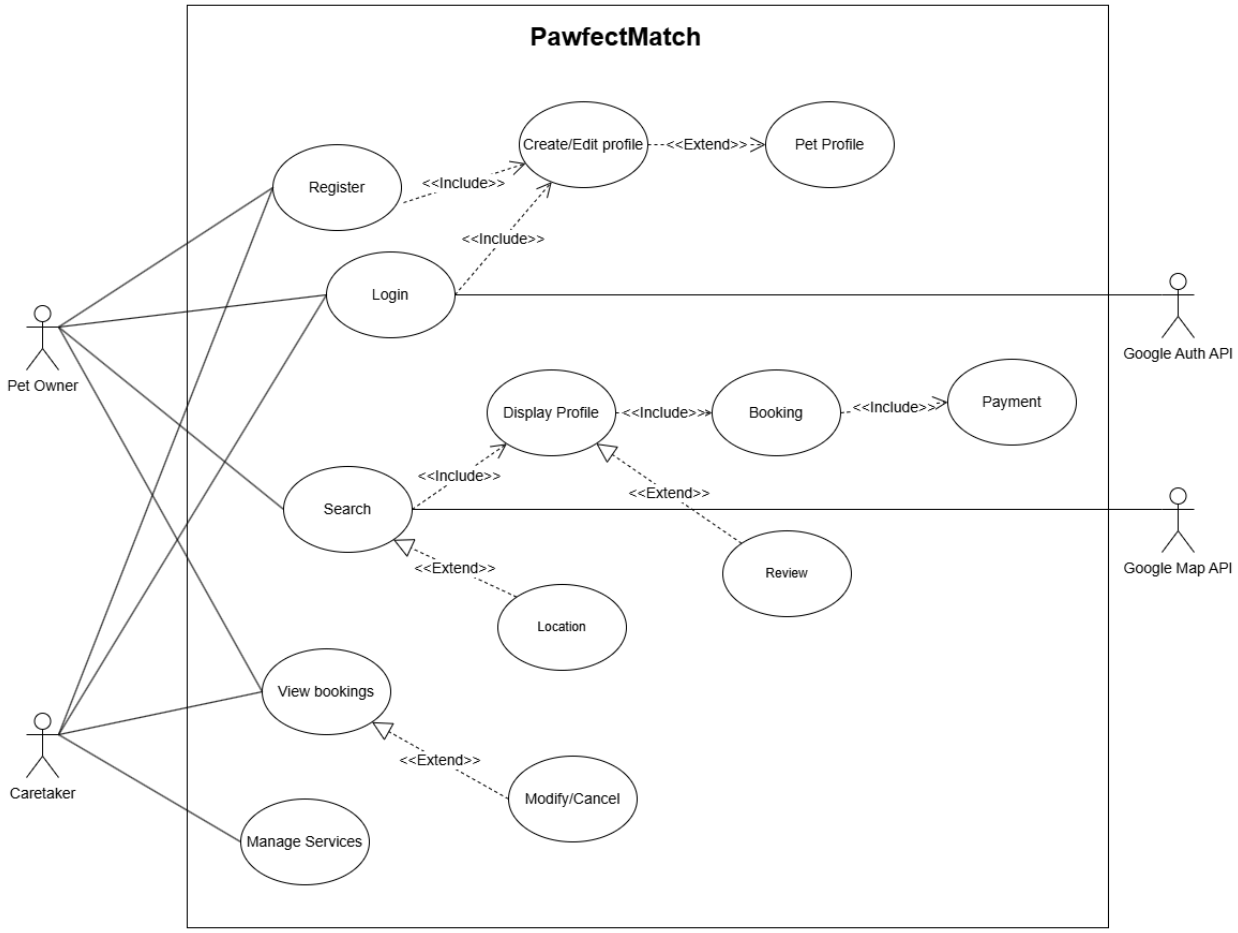

Figure 1. PawfectMatch Use Case Diagram

4 ASSUMPTIONS, CONSTRAINTS, RISKS

This section outlines the key factors influencing the project's release and outcome, which are explained in detail in the following sub-sections.

4.1 ASSUMPTIONS

The following assumptions describe the key conditions that must be met for the successful release and subsequent operation of the *PawfectMatch* system.

Capabilities and Scope

- **Core functionality** of PawfectMatch, as defined in the Software Requirements Specification (SRS), are assumed to be **fully developed, tested, and verified** prior to the commencement of the deployment phase, such that the release is based on a stable and functional version of the system
- All necessary **third-party licenses, frameworks, and tools** required for the production environment are assumed to be **secured, configured, and available** before the deployment phase begins
- **No major change in the requirements** of the customer is assumed after the baseline release plan has been approved

External Circumstances and Commitments

- **Approved budget, schedule, and resource allocation** for the final stages of development, testing, and deployment are assumed to **remain stable and unchanged**, with no significant increase from the required budget calculated
- NTU's institutional **IT infrastructure** is assumed to be **available and sufficient** to meet the performance and scalability requirements of the PawfectMatch system upon release

Dependencies and Participation

- **Team members** are assumed to be able to **complete all tasks** assigned to them and to be **available** throughout the deployment period
- Key **stakeholders** are assumed to provide **timely feedback and official sign-off** during User Acceptance Testing and to communicate effectively with developers to ensure the product meets customer requirements
- **No critical external dependencies** that would prevent the system from going live are assumed

- Required **APIs or data sources** from other systems are assumed to be **stable, well-documented, and accessible** for integration prior to release
- **Release schedule** is assumed to have been **set to avoid conflicts** with significant NTU business cycles or institutional events that could limit essential stakeholder participation or impact system resource availability

4.2 CONSTRAINTS

The release of *PawfectMatch* is subject to the following constraints, which may limit the ability to deploy the system in accordance with this release plan.

Schedule and Budget

- **Fixed End Date:** Release/deployment timeline is immovable due to the academic timetable and the fixed submission and demo deadlines. Any slippage reduces time for UAT, bug fixing, and documentation
- **Limited Resource Budget:** Project is restricted to university-provided resources and free/education tier tools. Premium hosting, domains, or paid API tiers cannot be added mid-release without prior approval

External Services and APIs

- **Rate Limits:** Integration with third-party services like Google OAuth 2.0 is constrained by their free-tier limits and quotas. Spiked in sign-ups, logins, or test transactions during the release window may result in temporary outages
- **Service Stability and Versioning:** Outages or API changes during the release window are outside the project team's control and will require unplanned rework

Security, Compliance and Data

- **Data Handling:** Use of real personal or sensitive production data is prohibited, test data must be fabricated or anonymized, which constrains the realism of performance testing and complex edge-case validation
- **Payment Handling:** Storage of sensitive payment data is not allowed, production payment flows must remain tokenized or in sandbox for demonstration

Team Capacity and Skills

- **Finite Availability:** Project team consists of seven students who have concurrent coursework commitments which limits the productive hours per week and the ability to perform parallel work and to provide rapid incident response

- **Lack of Skill Depth:** Project is constrained by the current skillset of the team, tasks outside of the team's skillset must be tightly scoped or avoided

4.3 RISKS

The following risks may adversely affect the release readiness during the *PawfectMatch* deployment. Each risk is monitored in the Risk Register and managed in accordance with the Project Management Plan.

Risk ID/Title	Impact	Mitigation	Fallback
Business			
R05 Google OAuth Policy Change	Users are unable to sign in	Monitor provider notices, and implement a secondary email/password authentication method	Temporarily switch all the users to the email/password login, and queue the OAuth fix for the next release cycle
Technical			
R07 Deployment Environment Failure	Application is unavailable during the scheduled maintenance window	Rehearse the full deployment on a staging environment, and keep known-good stable backups	Roll back to the last tagged stable build, and retry the deployment within the allotted window
R08 Critical Software Bugs	Reduces stakeholder trust and negatively affects academic assessment	Enforce mandatory peer reviews and CI test gates before merging	Execute a hotfix policy to immediately roll back to the stable tag, and notify stakeholders of the incident
R09 Dependency Issues	Build/CI failures causing delayed or broken release candidates	Pin all dependency versions, test all upgrades on a separate branch before merging	Re-pin all packages to the last known good versions and defer all non-essential upgrades to the next cycle

R11 API Integration Failure	Broken login, booking or other critical service flows are visible to users	Utilize contract tests, mocked services, and pre-release end-to-end tests	Switch affected API calls to mock services, set the UI to a read-only state
R12 Database Schema Mismatch	Data errors or empty listings shown to users, causing confusions for users	Use versioned migrations, and backward compatible changes	Pause database writes, run corrective migration script or restore from a snapshot
R14 Data Privacy Breach	Restricted features, reputational risk, and a lowered academic assessment	Enforce least-privilege access controls, and encrypt sensitive data	Disable affected endpoints, regenerate all keys and credentials, and document the incident and the fix
Organizational			
R19 Member Unavailable	Ownership gaps and slower incident response during the critical release phase	Assign secondary owners to critical modules	Reassign tickets to secondary owners, and use pair programming to manage tasks
Project Management			
R24 Missed Lab Submission Deadlines	Demo quality is reduced, and incomplete features are shown to stakeholders, risking a lower grade	Set internal cut-off dates at least one week earlier than the official deadline and tighten scope to “must have” features	Move non-critical features to future releases

Table 2. Release Risks

5 RELEASE APPROACH

5.1 RATIONALE

The release approach for *PawfectMatch* is designed as a **Phased Release Model**. The core rationale for this decision is to prioritize stability and de-risk the end-user experience. This strategy dictates that Phase 1, the core functionalities of the application will be fully developed according to System Requirements Specification, and internally verified first before deployment. This establishes a stable production baseline.

Only after this foundational phase is proven will subsequent phases commence, delivering non-critical, value-adding modules in sequential, manageable rollouts. This approach avoids user churn on foundational flows, reduces public-facing risks, and allows the team to validate deployment and operations on a stable system.

This strategy was directly shaped and validated by the following key deliverables and considerations:

- **System Requirements Specification:** Defines the Minimum Viable Product (MVP) baseline, allowing for a clean separation between Phase 1 and all subsequent phases, which will be additive and not introduce breaking changes to core user flows
- **Technical Design:** The main technology stack (React/TypeScript, FastAPI/Python and MySQL) plus integrations (Google OAuth) favor a modular architecture, which makes a phased model that separates core services from optional modules the most logical choice, minimizing cross-component risk during rollouts
- **Risk Management Plan:** Each release incorporates mitigation measures such as feature flags, rollback commits, reversible migrations, and staged testing to handle the risks detailed in the Risk Management Plan effectively
- **Lessons Learned:** Integration during earlier sprints revealed that large simultaneous changes increased the likelihood of defects, so, smaller, post-core releases are preferred to simplify testing, version control, and rollback procedures

Furthermore, the assumptions, constraints, and risks identified in Section 4 were pivotal in selecting this phased model.

- **Constraints:** Hard constraints such as immovable academic submission dates and finite team capacity demands a conservative approach of stabilizing before enhancing to ensure a high-quality core product is deliverable by the deadline
- **Risks:** The chosen approach directly mitigates high-priority risks. By first developing the core MVP in Phase 1, the risks of scope creep and complex schema errors were addressed. By isolating new features into their own phases,

risks can be managed using feature flags and reversible migrations, without jeopardizing the main application's stability

Overall, *PawfectMatch* adopts a phased, enhancement-based release approach that begins only after the main system has been stabilized. This method balances reliability and incremental progress by isolating new modules into small testable releases, enabling the team to maintain system reliability, manage risks, and deliver functional improvements in controlled, low-impact stages.

5.2 RELEASE STRATEGY

The *PawfectMatch* project adopts a Phased Function Release strategy. This approach involves segmenting the delivery of the system based on completed features, where each release represents a self-contained module. This strategy was chosen over a Phased User Base Rollout as it is more appropriate for an academic project that does not have a live, established user base to segment. This incremental model allows for controlled integration, rigorous testing of each new feature set, and stable, demonstrable progress aligned with academic milestones.

A critical component of this strategy is rigorous build and version control, a role managed by the release engineer. The release engineer is responsible for ensuring the correct combination of all software components and their dependencies for any given release. Semantic Versioning is used to manage this complexity.

- **Major Release (e.g. v1.x.x → v2.x.x):** Used for incompatible, breaking API changes
- **Minor Release (e.g. v1.0.x → v1.1.x):** Used for adding new, backward compatible functionality
- **Patch (e.g. v1.0.0 → v1.0.1):** Used for backward compatible bug fixes

This versioning ensures that every release is a well-defined, auditable package. This process also ensures the correct data is used by packaging all necessary database migration scripts with each release, guaranteeing that the database schema evolves correctly and in sync with the application code.

Release Process

To execute this strategy, the team follows a standardized release process built on Continuous Integration (CI) and version tagging. This pipeline ensures that every release is built in a consistent, automated, and traceable manner.

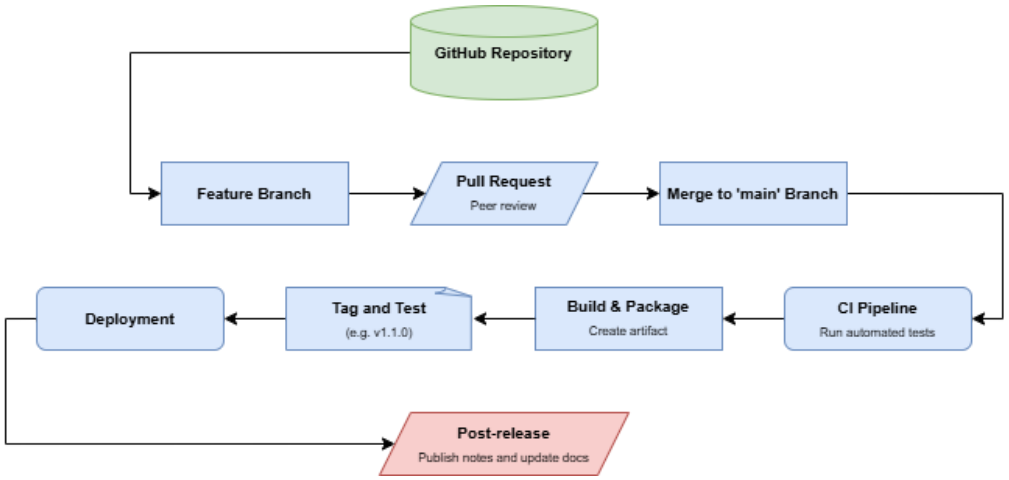

Figure 2. PawfectMatch Release Process

The process follows these key phases:

1. **Source Control (GitHub):** Developers commit code to feature branches and all code is peer-reviewed via Pull Requests (PRs) before being merged into the main branch

2. **Continuous Integration (CI):** Every PR merge to main automatically triggers a CI pipeline that runs all automated unit and integration tests to ensure the new code does not break existing functionality

3. **Build and Package:** Once CI passes, a build is created and packaged into a deployable artifact, which includes the compiled frontend assets, the backend API, and any new database migration scripts

4. **Tag and Test**: The team creates a new version tag for the build (e.g., v1.1.0) in GitHub and this tagged build is then sent for final internal testing and quality assurance

5. **Deployment:** After testing and quality assurance sign-off, the exact same tagged build is deployed

6. **Post-release:** The release is finalized by publishing formal Release Notes and creating a rollback anchor

Quality Standard (CRISP)

To govern the quality and readiness of each build that passes through this process, the team has adopted the CRISP requirements framework as its quality standard. Before any build is approved for deployment, it must be verified against the following five criteria.

- **Complete:** The build must contain all intended features, code, documentation, and configuration for that specific version
- **Repeatable:** The build process must be automated and documented, capable of producing the exact same artifact from the same source code every time
- **Informative:** The release is packaged with complete release notes, a version number, and a changelog mapping features to SRS requirements
- **Schedulable:** The release process is manageable and can be planned to align with the project schedule
- **Portable:** The build artifact is self-contained and can be deployed to any environment without modification

5.2.1 Release Content

The *PawfectMatch* system's main functionalities — User registration and authentication, service listing, booking management, payment processing, and user profiles — form the baseline for the initial release from which all subsequent releases are derived.

Future releases focus on extending functionality, enhancing user experience, and integrating supplementary features while maintaining system stability and backward compatibility.

Release 1 (R1): Core System (v1.0.0)

This will be a feature-complete version of the application that implements all core requirements defined in the SRS. It will be the initial deployment and serve as the stable, verified foundation and rollback anchor of all future releases.

The functionality of R1 encapsulates the entire set of core functional requirements defined in the SRS:

- **SRS 7.1:** User Management (Registration, Login, User Profile)
- **SRS 7.2:** Pet Management (Create/Edit Pet Profile)
- **SRS 7.3:** Service Discovery (Search, Display Profile)
- **SRS 7.4:** Booking Management (View/Modify/Cancel Booking)
- **SRS 7.5:** Payment and Service Management (Payment, Manage Services)

The rationale for this initial release is to launch a complete, stable, and fully functional core product. Bundling all core requirements into the v1.0.0 release ensures the foundational user journey exists from the start. This provides a stable and verified

baseline, which is a necessary prerequisite for all future enhancement waves, and prioritizes a comprehensive and reliable initial user experience.

Release 2 (R2): Reviews Module (v1.1.0)

This will be the first enhancement release for *PawfectMatch*. It introduces the Reviews and Ratings feature, a critical component for building user trust.

This release will deliver the following functionality:

- Enables registered pet owners to submit, edit, and delete reviews (1 - 5 star rating and description) for completed services
- Displays aggregated average ratings and all written feedback on service provider profiles and service-specific pages

The reviews feature is the most critical enhancement feature because it directly depends on a stable user and booking modules, and provides immediate, high value to users.

Release 3 (R3): Location and Geo-Search Module (v1.2.0)

This release introduces location-based services and map integration, fulfilling the vision of the Search functionality with location search filters.

The functionality of this release is focused on enhancing service discovery:

- Integrates Google Maps API for displaying service provider locations
- Enables distance-based filtering of search results, which enhances the basic search functionality defined in SRS 7.3.1

This module is explicitly sequenced after R2 because it introduces a significant external dependency (Google Maps API) with associated quota, performance, and billing risks. This sequencing isolated that risk, allowing the core app and reviews enhancement to be stable before tackling this complex integration.

Release 4 (R4): Optimization and Quality of Life (v1.3.0)

This is a consolidated quality and performance release intended to prepare the application for final submission. It does not introduce new headline features but instead focuses on polish, performance, and responding to feedback based on the system's operational requirements.

The release will implement the following:

- Performance improvements to key database queries
- UI/UX refinements and accessibility compliance

- Email notifications for booking confirmations

This final release is scheduled last to allow the team to gather feedback on the already released enhancement modules. This ensures that performance optimizations and final polish are applied to the entire application, resulting in a higher quality and stable product.

To maintain focus and adhere to the project's academic deadlines, the following features are explicitly out of scope for the releases defined in this plan: in-app real-time chat, loyalty programs, and advanced analytics dashboards for providers. These items may be considered for future iterations beyond the v1.3.0 release.

5.2.2 Release Schedule

The *PawfectMatch* release schedule follows an incremental rollout model where each release builds upon a stable, verified baseline. This schedule ensures that the core system is fully deployed and validated before any enhancement modules are introduced. Each enhancement release is time-boxed to allow sufficient testing and feedback before proceeding to the next phase.

Release	Version	Key Deliverables	Milestone Target	Dependencies/Notes
R1: Core System	v1.0.0	Full implementation of all core features (User, Pet, Service, Booking, Payment)	Week 8	Establishes the verified baseline, and milestone aligns with the major project lab demo milestone
R2: Reviews Module	v1.1.0	Review and rating system (View, Create, Aggregate)	Week 10	R1 baseline must be stable and deployed successfully
R3: Location and Geo-Search Module	v1.2.0	Location services integration (Google Maps API, Distance-based search filters)	Week 11	R2 release validation must be complete, and external API keys must be generated
R4: Optimization and Quality of Life	v1.3.0	System performance improvements, UI/UX polish, accessibility compliance, email notifications	Week 12	All feature-releases must be completed, and milestone aligns with final project submission deadline

Table 3. Release Schedule

Each milestone target in the schedule above represents the end of a full development and deployment cycle, which includes:

- Development completion and code merge to main branch
- Successful completion of all integration and regression testing
- Deployment to the production environment
- Tagging the release in the source control repository

This structured release timeline ensures that *PawfectMatch* transitions smoothly from its initial deployment to an enhanced, stable, and polished final version, all within the academic project timeframe.

5.2.3 Release Impacts

Each release in this plan incrementally enhances system functionality and the end-user experience, building upon the stable, backward-compatible core. The following details the specific system and user impacts for each release, along with the objectives and benefits that release provides.

Release 1: Core System (v1.0.0)

- **Objective:** Deliver a stable, feature complete, and demonstrable core product that aligns with the project's main academic timeline
- **System Impacts:** Establishes the entire foundational architecture, deploys the production database schema and core API routes required to fulfill all requirements, and establishes the CI/CD pipeline and serves as the v1.0.0 tagged anchor for all future rollbacks
- **User Impacts:** Complete end-to-end journey for both pet owners and service providers, with all primary functions made available for the first time
- **Benefits:** Provide a robust foundation for all subsequent enhancement releases

Release 2: Reviews Module (v1.1.0)

- **Objective:** Enhance platform's utility and credibility by introducing a user-driven feedback mechanism
- **System Impacts:** Introduce a schema extension by adding new database tables for Reviews, linked via foreign keys to the Booking and Service tables, and extends the API with new endpoints for review submission and retrieval

- **User Impacts:** Enriches the user experience by enabling a critical feedback loop where pet owners gain the ability to provide feedback, and all users benefit from the added transparency of visible ratings
- **Benefits:** Helps build user trust and engagement, and demonstrates the architecture's ability to easily extend with new, non-core features without destabilizing the core

Release 3: Location and Geo-Search Module (v1.2.0)

- **Objective:** Enhance the core service discovery value proposition and validate the system's capability to integrate and manage critical third-party APIs
- **System Impacts:** Introduce a significant external system dependency, Google Maps API, which expands the Service data model to include geolocation attributes and introduces more complex geospatial query logic to the backend and search interface
- **User Impacts:** Improves the utility of service discovery, where users can now move from the originally implemented search filters to a location-aware search, improving the usability of finding nearby providers
- **Benefits:** Increases platform's utility and user retention by making the service search process significantly faster and more relevant for end-users

Release 4: Optimization and Quality of Life (v1.3.0)

- **Objective:** Deliver high quality, polished, and production ready application suitable for final project submission
- **System Impacts:** Implements an asynchronous email notification service for booking confirmations
- **User Impacts:** Smoother and faster user experience where users receive timely, out-of-app confirmations and benefit from improved page load times
- **Benefits:** Improves overall user satisfaction and trust in the platform by providing a professional, responsive, and reliable experience

5.2.4 Release Notification

As *PawfectMatch* is developed and deployed within an academic environment, the release notification process is focused on ensuring clear communication, coordination, and documentation among the key stakeholders. This process is managed by the project team and provides timely updates to the Project Supervisor (Lab Teaching Assistant) and project team.

The primary communication channels for release-specific information are as follows.

- **Formal Notifications (Email):** Used for official pre-release and post-release summaries sent to both the Project Supervisor and the development team
- **Technical Coordination (Telegram and JIRA):** Used for real-time communication within the project team during deployment
- **Formal Documentation (GitHub):** Serves as the permanent record, with all releases tagged and accompanied by detailed release notes

Notification Timeline

The notification process is structured around three key notifications for each planned release.

1. **Pre-release Notification:** Formal email is sent to the Project Supervisor and development team approximately **three days prior** to a release to confirm the target release date and provide a high-level summary of the new functionalities. This notification will also confirm that all internal testing has been completed and the release is on track, providing an opportunity for final stakeholder feedback
2. **Day-of-release Coordination:** On the day of deployment, communication is handled internally within the project team via telegram. The team announces the start of the deployment, any brief service interruptions for testing, and the final confirmation of a successful rollout. This is a technical, real-time coordination channel and does not involve external stakeholders
3. **Post-release Confirmation:** Final confirmation email is sent to the Project Supervisor and development team within 24 hours of a successful release to officially announce that the new version of the application is live. It includes a link to the corresponding GitHub Release Notes, which detail the new features, any minor bug fixes, and the specific requirements that have been fulfilled. This notification also confirms that the release is stable and ready for review or demonstration

GitHub Release Notes

To ensure consistency and maintain a clear, professional project history, each release posted to the project's GitHub repository will follow a standardized format. This note serves as the formal "Statement of Work Completed" for the release.

The content of each release note will include:

- **Release Version and Date:** Official version tag and date of deployment
- **Summary:** One or two sentence overview of the release's main purpose

- **New Features:** List of new functionalities introduced in the release
- **Bug Fixes and Improvements:** List of non-critical issues or performance enhancements addressed
- **Requirements Fulfilled:** Mapping of completed features to corresponding JIRA ticket IDs
- **Known Issues:** Transparent list of any minor, non-blocking bugs that were identified and deferred to a future release
- **Full Changelog:** Link to the GitHub commit log for the release

6 GLOSSARY

Application Programming Interface (API)

A set of protocols and definitions that allow the frontend and backend systems in PawfectMatch to communicate. Examples include internal APIs for user management and external APIs such as Google OAuth.

Continuous Integration / Continuous Deployment (CI/CD)

An automated pipeline that runs tests, builds, and deploys the PawfectMatch system upon changes to the source code. Used to ensure consistent and error-free releases.

Deployment

The process of transferring a tested version of PawfectMatch from the development environment to the staging or production environment for use by end users.

Feature Flag

A development mechanism that allows certain features to be enabled or disabled without modifying the underlying codebase. Used in PawfectMatch to manage incremental feature rollouts.

GitHub Repository

A cloud-based version control repository used by the development team to store, track, and manage the source code of PawfectMatch.

Project Supervisor

The lab teaching assistant overseeing the project's progress and approving release documentation and deliverables.

Release

A specific version of PawfectMatch that introduces new functionality or improvements. Each release is identified by a semantic version tag (e.g., v1.0.0, v1.1.0).

Release Plan

A formal project document describing the release strategy, schedule, impacts, communication process, and related activities for PawfectMatch.

User Acceptance Testing (UAT)

The final testing phase conducted by the development team and supervisor to ensure that a release meets the functional requirements and operates as intended.

7 ACRONYMS

Acronym	Literal Translation
API	Application Programming Interface
CI	Continuous Integration
CD	Continuous Deployment
NTU	Nanyang Technological University
SDLC	Software Development Life Cycle
SRS	Software Requirements Specification
UAT	User Acceptance Testing
UI/UX	User Interface / User Experience