PetSoLive: A Specialized Platform for Animal Welfare

Furkan BULUT - 210316011, Furkan BAYTAK - 210316033, Furkan ÖZKAYA - 200316060
Group 28, Software Engineering Course, Fall 2024

***Abstract*—PetSoLive is a specialized social media platform designed to bridge the communication gap between animal owners and veterinarians. This document outlines the software's objectives, architecture, functional and non-functional requirements, and various scenarios it addresses. The goal is to enhance animal welfare through improved communication, adoption processes, and timely veterinary care. This paper presents a detailed description of PetSoLive's structure, models, requirements, and future evolution.**

***Index Terms*—PetSoLive, Animal Welfare, Veterinarian Support, Adoption Platform, Veterinary Communication, Emergency Assistance.**

I. Feasibility Study and Plan

A. Project Overview

PetSoLive is envisioned as a digital platform that bridges the gap between animal owners and veterinarians. The project aims to address the limitations of traditional communication methods and general social media platforms by providing tailored solutions for pet care, adoption processes, and veterinary assistance. Several similar projects, such as PetFinder and AdoptAPet, have successfully implemented specific features. For example:

- **PetFinder:** A platform for connecting potential adopters with pets in need, focusing on adoption listings and search filters. However, it lacks integrated features for real-time veterinary consultations or emergency care management, which PetSoLive aims to address comprehensively.
- **AdoptAPet:** Known for its wide reach and ease of use in connecting users with shelters. While it provides excellent adoption listings, its scope does not extend to community features or health reminders, which are key enhancements in PetSoLive.
- **BarkHappy:** Focused on fostering a social network for pet owners, it offers features like event organization and pet profiles but does not integrate veterinary support or emergency assistance tools.

By building on the strengths and addressing the limitations of these platforms, PetSoLive aims to offer a comprehensive solution that enhances adoption processes, provides real-time veterinary support, and fosters a vibrant community for pet owners and professionals.

B. Economic Feasibility

The project's financial plan accounts for:

- Development costs for a team proficient in .NET technologies.
- Licensing and operational costs for Windows IIS and Microsoft SQL Server.
- Marketing and outreach to ensure platform adoption by stakeholders.

Revenue will primarily be generated through subscription-based premium features such as veterinary consultation scheduling and priority adoption listings.

C. Operational Feasibility

PetSoLive is user-centric, offering:

- An intuitive interface that simplifies adoption processes and emergency responses.
- Customizable features to meet the unique needs of veterinarians and pet owners.

Feedback from key stakeholders will be integrated continuously to improve operational efficiency.

D. Schedule Feasibility

The project timeline is structured as follows:

- **Phase 1: Requirements Gathering (1 Month)** - Define functional and non-functional requirements.
- **Phase 2: System Design and Prototype Development (2 Months)** - Build a functional prototype using ASP.NET MVC.
- **Phase 3: Testing and Optimization (2 Months)** - Refine the platform based on feedback.
- **Phase 4: Deployment and Launch (1 Month)** - Launch with targeted marketing.

E. What is Kanban Methodology?

Kanban is a workflow management method that focuses on visualizing tasks, limiting work-in-progress, and enhancing overall productivity. Originating from the manufacturing industry, Kanban has proven effective in software development and other project environments due to its simplicity and adaptability.

1) Benefits of Kanban

- **Visual Workflow:** Tasks are organized on a board, providing clear visibility of progress.
- **Flexibility:** Teams can adapt priorities and manage tasks dynamically without disrupting workflow.
- **Improved Efficiency:** By limiting work-in-progress, teams focus on completing tasks before starting new ones.
- **Continuous Delivery:** Kanban supports incremental delivery, ensuring a steady flow of completed tasks.
- **Enhanced Collaboration:** Transparent workflows foster better communication and coordination among team members.

II. Glossary

This section defines the key terms used throughout the document, providing clarity for readers with varying levels of expertise.

- **Adopter:** An individual seeking to adopt a pet listed on the platform.
- **Administrator:** A platform moderator responsible for managing users, content, and ensuring adherence to guidelines.
- **API (Application Programming Interface):** A set of protocols and tools for building software applications and facilitating communication between components or services.
- **Authentication:** The process of verifying a user's identity to grant secure access to the platform.
- **Backend:** The server-side system responsible for data processing, logic, and database management.
- **Cloud Hosting:** The use of remote servers to store, manage, and process data, ensuring scalability and availability.
- **Dynamic Timeline:** A continuously updated feed displaying adoption stories, announcements, and other pet-related posts.
- **Encryption:** A security measure to protect sensitive data by converting it into an unreadable format unless decrypted.
- **Kanban Board:** A visual task management tool used to organize and track project workflows.
- **Lost Pet Alert:** A feature allowing users to post and share information about missing pets to solicit community assistance.
- **Notifications:** System-generated messages alerting users about events, updates, or reminders.
- **Pet Profile:** A digital record containing details about a pet, such as species, age, health status, and adoption status.
- **Real-Time Messaging:** A communication feature that enables instant exchanges of messages between users, such as pet owners and veterinarians.
- **Responsive Design:** A platform design approach ensuring usability and consistency across devices and screen sizes.
- **Role-Based Access Control (RBAC):** A security mechanism restricting access based on user roles (e.g., administrator, veterinarian, pet owner).
- **Use Case Diagram:** A visual representation illustrating the interactions between users and the system's functionalities.
- **Vaccination Calendar:** A tool for veterinarians to schedule and manage reminders for pet vaccinations.
- **Veterinarian:** A medical professional providing advice, care, and emergency assistance for pets through the platform.
- **UX (User Experience):** The overall quality and satisfaction users derive from interacting with the platform.

III. Requirements Engineering Steps

A. Requirements Engineering Process

The requirements engineering process for PetSoLive involves a systematic approach to identifying, analyzing, validating, and managing requirements. These iterative and interconnected activities ensure that the platform's design aligns with user needs and project goals. The following subsections outline each step in detail.

1) Requirements Elicitation

Requirements elicitation is the first step in understanding what users need from the system. For PetSoLive, the key activities include:

- Conducting interviews and surveys with pet owners, veterinarians, and adoption agencies to gather insights.
- Hosting workshops with stakeholders to brainstorm potential features and use cases.
- Analyzing existing platforms like PetFinder and AdoptAPet to identify strengths and weaknesses.
- Creating prototypes to visualize and refine user expectations.

2) Requirements Analysis

Requirements analysis involves evaluating and organizing the elicited information to identify conflicts, redundancies, and priorities. Activities in this phase include:

- Categorizing requirements into functional, non-functional, and domain-specific groups.
- Resolving conflicts between stakeholder demands through negotiation and prioritization.
- Using modeling techniques such as use case diagrams to clarify system behaviors.
- Checking for feasibility and technical constraints to ensure the viability of proposed requirements.

3) Requirements Validation

Requirements validation ensures that the documented requirements are complete, consistent, and aligned with user needs. Validation activities for PetSoLive include:

- Conducting review sessions with stakeholders to verify the accuracy of requirements.

- Using prototypes and mockups to test usability and functionality before development begins.
- Applying traceability techniques to ensure that each requirement links to a specific user need.
- Running pilot tests with a small group of users to gather feedback and identify gaps.

4) Requirements Management

Requirements management involves tracking and updating requirements as the project evolves. This phase is critical for maintaining alignment between the project scope and user needs. Key activities include:

- Version control to document changes and ensure traceability.
- Establishing a change request process to evaluate and incorporate new requirements.
- Regularly communicating with stakeholders to address emerging needs and expectations.
- Monitoring the implementation of requirements to ensure alignment with the original specifications.

B. Potential Problems and Solutions in Requirements Engineering Process

The requirements engineering process can encounter various challenges during elicitation, analysis, validation, and management. Identifying these potential problems and proposing solutions ensures the smooth progression of the project.

1) Problems in Requirements Elicitation

Problem: Miscommunication with stakeholders may lead to incomplete or ambiguous requirements. **Solution:**

- Conduct iterative interviews and surveys to ensure all user needs are captured.
- Use visual aids like prototypes and mockups to clarify expectations and align stakeholders.
- Include a diverse set of stakeholders in discussions to gather comprehensive feedback.

Problem: Stakeholders may have conflicting priorities or unrealistic expectations. **Solution:**

- Organize workshops to prioritize requirements collaboratively.
- Educate stakeholders about technical constraints to set realistic expectations.
- Use a structured prioritization framework, such as MoSCoW (Must Have, Should Have, Could Have, Won't Have), to address conflicts.

2) Problems in Requirements Analysis

Problem: Overlapping or redundant requirements can complicate the analysis phase. **Solution:**

- Regularly review and categorize requirements to eliminate redundancy.
- Use modeling techniques such as use case and class diagrams to highlight overlaps and refine requirements.

Problem: Feasibility issues may arise due to technical or resource constraints. **Solution:**

- Conduct a feasibility study early in the project to identify potential constraints.
- Involve technical experts in requirement reviews to assess implementation challenges.
- Adjust project scope based on resource availability and technical capabilities.

3) Problems in Requirements Validation

Problem: Validation sessions may fail to uncover inconsistencies or gaps in requirements. **Solution:**

- Use prototypes or mockups to simulate real-world scenarios and validate requirements.
- Involve a diverse group of users in validation sessions to gather varied perspectives.
- Employ automated tools for consistency checks and traceability.

4) Problems in Requirements Management

Problem: Changes in requirements may lead to scope creep and project delays. **Solution:**

- Establish a formal change management process to evaluate and approve changes.
- Regularly communicate with stakeholders to manage expectations and address emerging needs.
- Use version control systems to track and document changes systematically.

Problem: Requirements may become outdated due to evolving user needs or market conditions. **Solution:**

- Schedule periodic reviews to update requirements based on new information.
- Monitor industry trends and user feedback to anticipate and adapt to changes proactively.
- Maintain flexibility in the project plan to accommodate adjustments when necessary.

C. Functional Requirements

The functional requirements for PetSoLive define the core capabilities and services the system must deliver to its users. These requirements are integral to ensuring a seamless and engaging user experience. The key functionalities have been categorized into specific modules as described below.

1) Main Page Functionalities

The main page acts as the central hub for user interactions and information dissemination. The functionalities include:

- Displaying a dynamic social media timeline featuring adoption stories, veterinary tips, and lost pet alerts.
- Enabling click-through functionality for detailed post views.

- Providing a dedicated section for expert veterinary advice to enhance knowledge sharing.
- Allowing users to share posts within the platform, fostering community engagement.
- Supporting user interactions such as commenting and offering advice on shared posts.
- Displaying upcoming pet-related events such as shows and seminars with options to express interest and set reminders.
- Facilitating discussions on various pet-related topics through topic-specific threads.

2) Assistance Announcement Functionalities

The assistance announcement feature enables users to create and share urgent requests for help related to their pets. The functionalities include:

- Providing an option to create detailed assistance announcements with essential details such as the pet's name, species, age, and health issues.
- Allowing users to highlight the urgency of the situation using specific tags (e.g., ”Emergency”).
- Publishing announcements prominently on the platform to attract attention and support from the community.

3) Interaction on Assistance Announcements

Interaction on assistance announcements aims to connect users and veterinarians effectively. The functionalities include:

- Allowing users to view and respond to assistance announcements with supportive messages or advice.
- Enabling veterinarians to respond directly to emergency cases and gather more information about the pet's condition.
- Facilitating direct communication between users and veterinarians using integrated messaging tools.
- Supporting the sharing of relevant resources such as clinic locations and veterinarian recommendations.

D. Non-Functional Requirements

The non-functional requirements ensure the platform's reliability, security, and usability, forming the foundation for a robust system. These are critical to providing a consistent and secure user experience.

1) Performance

- Ensure smooth scrolling and quick loading times on the main page to enhance user experience.
- Efficiently handle increased traffic during emergency situations without performance degradation.

2) Reliability

- Maintain 24/7 platform availability to ensure continuous service for all users.
- Ensure data accuracy to prevent misinformation in adoption listings, veterinary advice, and emergency posts.
- Implement regular data backups to minimize the risk of data loss during unforeseen system failures.
- Provide robust data recovery mechanisms to restore operations promptly in case of hardware or software malfunctions.

3) Security

Security is crucial to protect sensitive user data and maintain trust within the community. The following measures will be adopted:

- Encrypt sensitive user data, such as login credentials and veterinary records, using advanced encryption standards (AES).
- Implement secure login mechanisms, including two-factor authentication (2FA), to prevent unauthorized access.
- Regularly perform security audits and vulnerability assessments to identify and mitigate potential risks.
- Comply with industry standards and regulations like GDPR to ensure lawful data handling practices.

4) Usability

To maximize user engagement and satisfaction, PetSoLive will focus on the following usability aspects:

- Design an intuitive user interface that simplifies navigation and supports clear workflows for common tasks.
- Provide comprehensive guidance and tooltips for creating posts, managing adoption requests, and seeking veterinary advice.
- Optimize the platform for use across various devices, including smartphones, tablets, and desktops.
- Support multiple languages to cater to a diverse user base and improve accessibility.

5) Accessibility

Ensuring accessibility for all users, including those with disabilities, is a key priority:

- Implement compatibility with assistive technologies, such as screen readers and voice commands, to support visually impaired users.
- Ensure a responsive design that adapts seamlessly to different screen sizes and resolutions.
- Optimize the platform for low-bandwidth environments to accommodate users with limited internet access.
- Adhere to Web Content Accessibility Guidelines (WCAG) to promote inclusivity and equal access.

IV. User Requirements

User requirements represent the needs and expectations of the end users interacting with the PetSoLive platform. By understanding these requirements, we ensure the system delivers maximum value to its users. The primary user groups for PetSoLive and their specific requirements are detailed below:

A. Pet Owners

- Ability to create and manage assistance announcements for emergencies.
- Access to a dynamic timeline showcasing adoption stories, veterinary advice, and pet-related events.
- Tools to post and manage lost pet alerts with geolocation tagging.
- Notifications and reminders for important pet health tasks, such as vaccinations.
- Secure communication channels with veterinarians and other community members.

B. Veterinarians

- Ability to respond to assistance announcements and provide medical advice in emergencies.
- Tools to manage profiles, display expertise, and interact with pet owners.
- A dedicated section to publish articles or advice columns to educate the community.
- Access to patient history shared by pet owners for better context during interactions.

C. Adoption Agencies

- Ability to list adoptable pets with comprehensive profiles (age, breed, health conditions, etc.).
- Tools for communication and follow-up with prospective adopters.
- Visibility on upcoming adoption events and the ability to promote listings on the platform.

D. Platform Administrators

- Management tools to oversee platform activities and ensure compliance with community guidelines.
- Access to reports and analytics for user engagement and performance metrics.
- Tools for monitoring and resolving disputes or issues raised by users.

V. System Requirements

System requirements translate the user needs into technical specifications that guide the development and implementation of the PetSoLive platform. These requirements are classified into functional and non-functional categories to ensure a structured and actionable approach.

A. Functional System Requirements

- Implement a dynamic timeline to display adoption stories, veterinary advice, and lost pet alerts.
- Provide a form-based interface for creating assistance announcements, including fields for pet details and urgency levels.
- Enable real-time notifications for emergency cases to ensure timely responses.
- Develop secure messaging systems for communication between users and veterinarians.
- Include advanced search and filtering options for adoption listings.
- Integrate a calendar system to manage pet health reminders and event notifications.
- Allow administrators to monitor and moderate platform content efficiently.

B. Non-Functional System Requirements

- Ensure the platform handles up to 10,000 concurrent users without performance degradation.
- Maintain system uptime of 99.9% to ensure consistent availability.
- Encrypt sensitive user data and communications to prevent unauthorized access.
- Optimize the platform for various devices, including desktops, tablets, and smartphones.
- Provide multilingual support to cater to a diverse user base.
- Adhere to accessibility standards such as WCAG for inclusive user experiences.

VI. System Stakeholders

Identifying the stakeholders of the PetSoLive platform is essential to ensure that the system meets the needs of all parties involved. Stakeholders include individuals and entities that directly or indirectly interact with or are impacted by the platform. The key stakeholder groups are outlined below:

A. Primary Stakeholders

- **Pet Owners:** The primary users of the platform who seek adoption services, veterinary advice, and emergency assistance.
- **Veterinarians:** Professionals providing medical advice, emergency support, and general pet care tips to the community.
- **Adoption Agencies:** Organizations managing adoption processes and using the platform to list adoptable pets and interact with prospective adopters.

B. Secondary Stakeholders

- **Platform Administrators:** Responsible for overseeing the platform's functionality, ensuring compliance with guidelines, and resolving user disputes.
- **Developers and Maintenance Teams:** Tasked with building, updating, and maintaining the platform's technical infrastructure.

C. Tertiary Stakeholders

- **Animal Welfare Organizations:** Use the platform to raise awareness about animal welfare issues and organize community events.
- **Event Organizers:** Promote and manage pet-related events, such as adoption drives and pet shows, through the platform.
- **Advertisers and Sponsors:** Collaborate with the platform to reach a targeted audience for pet-related products and services.

D. External Stakeholders

- **Regulatory Authorities:** Oversee compliance with local and international laws, such as data protection regulations (e.g., GDPR).
- **Internet Service Providers (ISPs):** Ensure reliable internet connectivity for the platform's operations.

VII. Completeness and Consistency of Requirements

The completeness and consistency of requirements are essential for the success of any software project. Incomplete or inconsistent requirements can lead to misunderstandings, development delays, and an overall failure to meet user expectations. This section discusses how PetSoLive's requirements are ensured to be both complete and consistent.

A. Completeness of Requirements

Completeness ensures that all necessary requirements have been identified and documented. For PetSoLive, the completeness of requirements is achieved by:

- Conducting comprehensive user interviews and surveys to gather all potential use cases.
- Reviewing similar platforms such as PetFinder and AdoptAPet to ensure no major functionality is overlooked.
- Establishing clear goals, such as ease of use, timely emergency responses, and community engagement, as guiding principles for the platform's design and implementation.
- Verifying that all user stories and scenarios cover the core functionalities, including adoption management, emergency announcements, and event promotion.

B. Consistency of Requirements

Consistency ensures that there are no conflicting or ambiguous requirements within the system documentation. For PetSoLive, the following measures are taken to maintain consistency:

- Utilizing structured specifications and templates to document requirements uniformly.
- Performing cross-reviews among stakeholders to identify and resolve conflicts early.
- Ensuring alignment between functional and non-functional requirements, such as matching system performance goals with user expectations for responsiveness.
- Regularly updating requirements documentation to reflect changes in user needs or project scope.

C. Verification and Validation

To further ensure completeness and consistency:

- Requirements are validated through prototyping and user feedback sessions.
- Consistency checks are performed using tools and techniques like traceability matrices and requirement comparison models.
- Measurable goals (e.g., a timeline refresh rate of under 2 seconds) are established to provide objective benchmarks for non-functional requirements.

VIII. Usability Requirements

Usability requirements focus on ensuring that the PetSoLive platform is intuitive, user-friendly, and accessible to a diverse audience. These requirements aim to minimize user errors, reduce the learning curve, and optimize the overall user experience.

A. Goals for Usability

The primary usability goals for the PetSoLive platform are:

- **Ease of Use:** The system should be easy to navigate for all user groups, including pet owners, veterinarians, and adoption agencies.
- **Error Reduction:** The platform should be designed to minimize user errors through clear instructions, validations, and intuitive workflows.
- **Accessibility:** Support for different devices, screen sizes, and user capabilities to ensure inclusivity.

B. Testable Usability Requirements

To objectively verify usability, the following measurable criteria are defined:

- **Training Time:** Users should be able to effectively use all major system functions after a maximum of 4 hours of training.
- **Error Rate:** After training, experienced users should make no more than two errors per hour of system use.
- **Response Time:** User actions should yield visible results within 2 seconds on average, enhancing the perception of system responsiveness.
- **Help Features:** The system should include context-sensitive help tools and FAQs accessible from any page.

C. Metrics for Usability Evaluation

The usability of PetSoLive will be evaluated based on the following metrics:

- **Ease of Use:** Training time and the number of help frames required to understand the platform.
- **Error Reduction:** The percentage reduction in errors over time as users become familiar with the system.
- **User Satisfaction:** Collected via surveys and feedback forms with a target satisfaction score of 90% or higher.

IX. User Stories, Scenarios and Task Cards

User stories and scenarios provide a practical way to understand how the system will be used in real-life situations. For PetSoLive, they help relate the system's features to the needs of different stakeholders while providing actionable insights for development.

A. Scenario 1: Create Assistance Announcement

Scenario Description: Alice notices her pet is experiencing a medical emergency and wants to seek help through PetSoLive. She logs into the platform, navigates to the ”Assistance Announcement” section, fills in the details about her pet's health condition, urgency level, and adds a photo. Once she submits the announcement, veterinarians are notified, and the post becomes visible on her timeline.

Task Cards:

- **Task Card 1: Implement Assistance Announcement Submission**
  - Develop a form for users to input details such as pet name, condition, and urgency level.
  - Integrate the form with the backend to store announcements in the database.
  - Allow users to upload a photo and attach it to the announcement.
  - Provide a ”Submit” button to post the announcement.
- **Task Card 2: Emergency Notification to Veterinarians**
  - Implement a notification system to alert veterinarians of emergency announcements.
  - Ensure notifications include critical details such as pet condition and contact information.
- **Task Card 3: Display Assistance Announcements**
  - Create a timeline module that prominently displays assistance announcements.
  - Ensure announcements include tags like ”Emergency” for quick visibility.

B. Scenario 2: Post Lost Pet Alert

Scenario Description: Alex realizes his pet has gone missing and wants to post an alert on PetSoLive. He logs into the platform, navigates to the ”Lost Pet Alert” section, fills in the pet's name, last seen location, and uploads a photo. Once posted, local users receive notifications about the lost pet, and Alex monitors his notifications for tips from the community.

Task Cards:

- **Task Card 1: Implement Lost Pet Alert Submission**
  - Develop a form for users to input pet details like name and last seen location.
  - Enable photo uploads for lost pet alerts.
  - Integrate the form with the backend to save the alert in the database.
- **Task Card 2: Notify Local Users**
  - Design a notification system to alert local users about lost pets.
  - Ensure notifications include the pet's photo, name, and location.
- **Task Card 3: Community Interaction with Alerts**
  - Enable community members to comment on or share the lost pet alert.
  - Provide a ”Tip Submission” feature to report sightings.

C. Scenario 3: Manage Vaccination Calendar

Scenario Description: A veterinarian logs into PetSoLive and accesses the ”Vaccination Calendar” feature. They add a new vaccine reminder for a pet with a specific due date. The system automatically sends reminders to the pet owner about the upcoming vaccination.

Task Cards:

- **Task Card 1: Develop Vaccination Calendar Module**
  - Create a calendar interface for veterinarians to view and manage vaccine schedules.
  - Enable adding, editing, and deleting vaccine reminders.
- **Task Card 2: Vaccination Reminder Notifications**
  - Implement a notification system to alert pet owners about upcoming vaccinations.
  - Ensure reminders are sent a configurable number of days before the due date.

D. Scenario 4: Browse Timeline and Interact with Posts

Scenario Description: Alice logs into PetSoLive and navigates to her timeline. She browses posts, including adoption stories, assistance announcements, and lost pet alerts. She interacts with posts by liking, commenting, and sharing, and the system updates interactions in real-time.

Task Cards:

- **Task Card 1: Develop Timeline Browsing Feature**
  - Create a timeline interface to display posts dynamically.
  - Implement filtering options for categories like ”Adoption” or ”Assistance.”
- **Task Card 2: Enable Post Interactions**
  - Add ”Like,” ”Comment,” and ”Share” buttons for each post.
  - Ensure interactions are stored in the database and reflected in real-time.
- **Task Card 3: Notify Post Owners**
  - Implement a notification system to inform post owners about interactions.
  - Include details of the interaction (e.g., ”Alice liked your post”).

X. Structured Specification

Structured specifications provide a standardized format for documenting system requirements, ensuring clarity, consistency, and comprehensiveness. For PetSoLive, this approach ensures that developers and stakeholders have a clear understanding of system functionalities. Below, we outline structured specifications for key requirements in the system.

A. Assistance Announcement Submission

Function: Create and submit an assistance announcement. **Description:** Allows users to report emergency situations involving their pets by filling out relevant details and attaching a photo. The announcement will notify veterinarians and appear on the timeline.

Inputs:

- Pet Name (String)
- Pet Condition (Text)
- Urgency Level (Dropdown: Low, Medium, High)
- Photo (Image File)

Source: User-provided data via the form interface.

Outputs:

- Assistance announcement saved in the database.
- Notifications sent to veterinarians.

Destination:

- Timeline displaying assistance announcement.
- Veterinarian notification system.

Action: On form submission, the data is validated and stored in the database. Notifications are triggered to alert veterinarians.

Pre-condition: User must be authenticated.

Post-condition: Assistance announcements appear on the timeline, and veterinarians are notified.

B. Lost Pet Alert Submission

Function: Post an alert for a lost pet. **Description:** Enables users to create alerts for missing pets, providing information such as last seen location and a photo. Alerts notify users in the local community.

Inputs:

- Pet Name (String)
- Last Seen Location (Text/Map Coordinates)
- Photo (Image File)

Source: User interface form.

Outputs:

- Lost pet alert stored in the database.
- Notifications sent to local users.

Destination:

- Timeline showing the lost pet alert.
- Local user notification system.

Action: Upon submission, the alert is validated, stored in the database, and broadcast to nearby users.

Pre-condition: User must be logged in and provide a valid last seen location.

Post-condition: The alert is visible on the timeline, and notifications are sent to local users.

C. Vaccination Reminder Notification

Function: Notify pet owners about upcoming vaccination deadlines. **Description:** The system sends reminders to pet owners about vaccinations due for their pets based on schedules set by veterinarians.

Inputs:

- Vaccine Type (String)
- Due Date (Date)
- Owner Contact Details (String)

Source: Veterinarian input in the vaccination calendar.

Outputs:

- Reminder notification delivered to pet owners.

Destination: Pet owner notification system (via app notifications or email).

Action: The system checks the database for upcoming vaccination dates and sends reminders a configurable number of days before the deadline.

Pre-condition: Vaccination details must be entered and approved by a veterinarian.

Post-condition: Pet owners receive vaccination reminders, and vaccination history is updated upon confirmation.

D. Timeline Browsing and Post Interactions

Function: Allow users to browse the timeline and interact with posts. **Description:** Users can scroll through their timeline to view posts, including adoption stories, assistance announcements, and lost pet alerts. Posts can be liked, commented on, or shared.

Inputs:

- Timeline Filters (Dropdown: Adoption, Assistance, Lost Pets)
- Interaction Options (Buttons: Like, Comment, Share)

Source: User interface.

Outputs:

- User interactions (likes, comments, shares) stored in the database.
- Notifications sent to post owners.

Destination:

- Updated timeline reflecting interactions.
- Notification system for post owners.

Action: Interactions are recorded in the database and notifications are triggered for post owners.

Pre-condition: Users must be authenticated to interact with posts.

Post-condition: Timeline displays updated interaction data, and post owners are notified.

E. Adoption Process Management

Function: Manage adoption listings and inquiries. **Description:** The system enables users and agencies to post, browse, and interact with adoption listings, facilitating pet adoption.

Inputs:

- Pet Profile Details (e.g., name, age, breed, health status).
- User Interaction Options (Buttons: Contact Agency, Show Interest).

Source: User or agency-provided data.

Outputs:

- Adoption listings displayed on the timeline.
- Notifications sent to adoption agencies for user interactions.

Destination:

- Adoption timeline for users.
- Notification system for agencies.

Action: Listings are created and displayed on the timeline, and user interactions are logged and communicated to agencies.

Pre-condition: User or agency must have an authenticated account.

Post-condition: Listings are visible, and interactions are processed in real-time.

F. Notification Management System

Function: Manage and deliver notifications for system events. **Description:** The system generates and sends notifications for activities such as assistance announcements, lost pet alerts, vaccination reminders, and adoption inquiries.

Inputs:

- Notification Type (e.g., Emergency, Interaction, Reminder).
- Recipient Details (User ID or Group ID).

Source: System event triggers.

Outputs:

- Notifications sent to user devices (Push or Email).
- Notification logs updated in the database.

Destination:

- User notification interface.
- Notification database for logging.

Action: Notifications are generated based on system events and dispatched to users.

Pre-condition: Triggering event must occur within the system (e.g., form submission, interaction).

Post-condition: Notifications are received by users and logged for traceability.

XI. Requirements Checking

Requirements checking ensures that all requirements meet the project's goals and adhere to critical evaluation criteria such as validity, consistency, completeness, realism, and verifiability. The process is applied systematically to identify and address any gaps or conflicts within the requirements. Below, the steps taken to apply this methodology to the PetSoLive platform's requirements are outlined.

A. Application of Requirements Checking

Each requirement within the platform's structured specification has been evaluated against the five key criteria of requirements checking. The findings and implementation plan are detailed as follows:

1) Validity

Validity ensures that the requirements align with the users' needs and the overall objectives of the PetSoLive platform. The process for validating requirements includes:

- Mapping each requirement to a specific user need or business goal. For example, the ”Lost Pet Alert Submission” requirement directly addresses the need for pet owners to locate their missing pets through community engagement.
- Reviewing stakeholder feedback to confirm that the requirement supports the intended functionality. For instance, emergency announcements are validated as essential features by veterinarians and pet owners during stakeholder workshops.

2) Consistency

Consistency ensures that there are no contradictions or overlaps among requirements. To achieve consistency:

- Each requirement was cross-referenced with related requirements and system modules. For example, the ”Notification Management System” was verified to integrate seamlessly with the ”Lost Pet Alert” and ”Vaccination Reminder” functionalities.
- Conflicting requirements, if identified, were resolved through stakeholder discussions. For instance, ensuring that ”Urgency Tags” for announcements are uniformly applied across all relevant features.

3) Completeness

Completeness guarantees that all required functionalities are fully specified and accounted for. This was achieved through:

- Verifying that all key features, such as ”Lost Pet Alerts,” ”Adoption Listings,” and ”Vaccination Reminders,” have detailed input, output, and process definitions within the structured specification.
- Identifying and addressing potential gaps. For example, requiring a mandatory ”Photo Upload” field in the ”Lost Pet Alert Submission” form was added to enhance completeness.

4) Realism

Realism evaluates whether the requirements can be feasibly implemented within the project’s budget, timeline, and technical constraints. This was ensured by:

- Conducting technical feasibility studies for key functionalities such as ”Geo-Location Based Notifications” to ensure they are achievable with existing technology stacks.
- Assessing resource availability and adjusting scope where necessary. For example, prioritizing core functionalities like ”Emergency Assistance Announcements” in the initial release.

5) Verifiability

Verifiability ensures that each requirement can be objectively tested and validated. To meet this criterion:

- Test cases were developed for all major functionalities. For instance, a test scenario for the ”Lost Pet Alert” requirement involves verifying that alerts reach users within the specified geographic area.
- Establishing measurable benchmarks for system performance, such as ensuring notification delivery within 2 seconds of event triggering.

B. Planned Implementation of Requirements Checking

To ensure ongoing compliance with the requirements checking process, the following practices will be integrated into the project lifecycle:

- **Regular Reviews:** Periodic stakeholder reviews will validate the relevance and accuracy of requirements as the project evolves.
- **Traceability Matrices:** Each requirement will be linked to its corresponding user story, system module, and test case to ensure traceability.
- **Continuous Validation:** Functional prototypes and user feedback sessions will be used to validate requirements in real-world scenarios.
- **Change Management:** A formal change management process will address updates to requirements, ensuring that changes remain consistent, complete, and realistic.

By applying these practices, the PetSoLive project ensures that all requirements are accurate, feasible, and aligned with the platform’s goals, providing a robust foundation for development.

XII. System Boundaries

The following section outlines the system boundaries for PetSoLive, detailing the functionalities included within the system and those that fall outside its scope. This structured approach ensures clarity on what the system is designed to handle directly and what relies on external services or actors.

A. Inside the System Boundary

The PetSoLive platform provides the following core functionalities within its system boundary:

a) User Management

- User registration, authentication, and profile management.
- Maintaining and managing pet profiles with details such as species, age, and health history.

b) Social Media Features

- **Dynamic Timeline:** Displays adoption stories, veterinary tips, lost pet alerts, and other pet-related posts.
- **Post Management:** Users can create, like, share, and bookmark posts. Posts can include photos and other media.
- **Post Filtering:** Users can filter timeline content based on specific criteria.

c) Assistance Announcements

- Allowing pet owners to create Assistance Announcements with details like urgency and pet health conditions.
- Highlighting emergency posts with tags.
- Notifying veterinarians about emergency cases.

d) Adoption and Lost Pet Management

- Providing tools for posting and managing Adoption Listings and Lost Pet Alerts.
- Sending notifications to local users about lost pets or adoption opportunities.
- Allowing communication between pet owners, adopters, and community members.

e) Vaccination Management

- Veterinarians can create and update Vaccination Calendars.
- Sending automated reminders to pet owners about upcoming vaccines.
- Allowing pet owners to schedule appointments through the platform.

f) Notifications and Reminders

- Sending notifications for events, assistance announcements, and vaccination reminders.
- Ensuring that critical updates are delivered to relevant users.

g) Real-Time Communication

- Facilitating real-time messaging between pet owners and veterinarians or other community members.

h) File Storage and Management

- Handling user-uploaded photos and media for posts, assistance announcements, and adoption profiles.
- Windows IIS manages file uploads and retrieval securely and efficiently.

i) System Administration

- Monitoring platform usage and resolving technical issues.
- Moderating user content to ensure compliance with platform guidelines.
- Generating analytics and reports to improve platform performance.

1) Outside the System Boundary

The following functionalities are outside the scope of the PetSoLive system but may rely on external services or actors:

a) Veterinary Diagnostics and Treatments

- The platform does not perform medical diagnostics or treatments. It only facilitates communication between pet owners and veterinarians.

b) Payment Systems

- Adoption fees or payments for veterinary services are handled through external payment gateways.

c) Third-Party Services

- Notifications (email, SMS, push) and mapping services are integrated via third-party APIs.

d) Data Hosting

- Data hosting, backups, and disaster recovery are handled by external cloud services.

e) Event Organization

- While the platform supports event announcements, it does not manage or organize events directly.

2) External Actors

a) Pet Owners

- Use the platform to manage posts, receive notifications, and communicate with veterinarians and the community.

b) Veterinarians

- Access assistance announcements, manage vaccination calendars, and provide medical advice.

c) Adoption Agencies

- Manage adoption listings and communicate with potential adopters.

d) Community Members

- Interact with posts, participate in discussions, and provide support for lost pet alerts.

e) System Administrators

- Manage user accounts, monitor system health, and maintain platform operations.

f) Third-Party Services

- Provide external support for notifications, mapping, and payment processing.

3) Boundary Justification

- **Inside the Boundary:** All functionalities directly implemented and controlled by the PetSoLive system, including core features like timeline management, assistance announcements, and file storage.
- **Outside the Boundary:** Services and activities that require external actors or third-party integration, such as payment processing and medical diagnostics, are excluded from the system boundary.

XIII. System Design

This section describes the system design of the PetSoLive platform, including behavioral and structural models along with associated diagrams for better visualization.

A. Use Case Diagram

The use case diagram below provides an overview of the interactions between users, veterinarians, and the system, showcasing the primary functionalities offered by the platform.

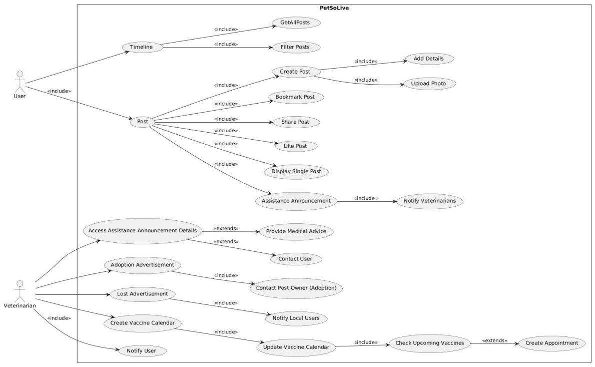


Fig. 1. Use Case Diagram for PetSoLive

B. State Diagram

The state diagram illustrates the user flow within the PetSoLive platform, starting from opening the homepage to logging out.

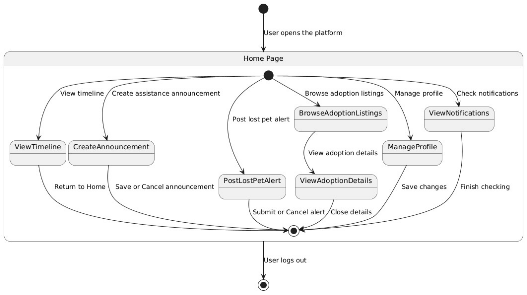


Fig. 2. State Diagram for PetSoLive

C. Sequence Diagrams

This subsection provides sequence diagrams for key functionalities within the PetSoLive platform.

1) Missing Advertisement Process

The sequence diagram below demonstrates the process of creating and resolving a missing advertisement for a lost pet.

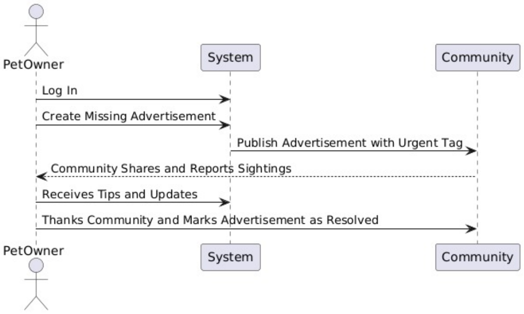

PetOwner
System
Community
Log In
Create Missing Advertisement
Publish Advertisement with Urgent Tag
Community Shares and Reports Sightings
Receives Tips and Updates
Thanks Community and Marks Advertisement as Resolved
PetOwner
System
Community

Fig. 3. Sequence Diagram for Missing Advertisement Process

2) Vaccine Reminder Process

This sequence diagram outlines how the system notifies pet owners about upcoming vaccinations and updates vaccination records.

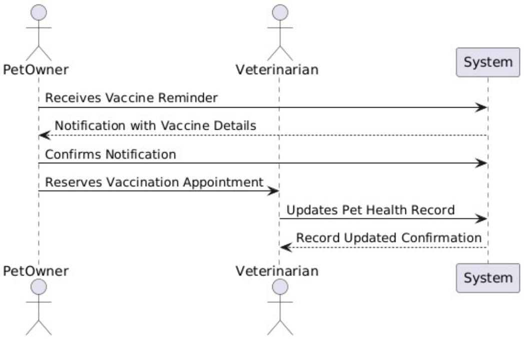

PetOwner
Veterinarian
System
Receives Vaccine Reminder
Notification with Vaccine Details
Confirms Notification
Reserves Vaccination Appointment
Updates Pet Health Record
Record Updated Confirmation
PetOwner
Veterinarian
System

Fig. 4. Sequence Diagram for Vaccine Reminder Process

3) Adoption Process

The following sequence diagram explains the interaction flow during the adoption process, from browsing listings to finalizing adoption.

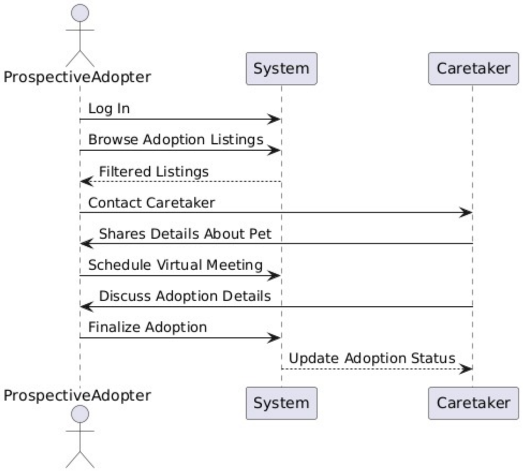


Fig. 5. Sequence Diagram for Adoption Process

4) Timeline Interaction

This diagram showcases how users interact with the timeline, including viewing posts, interacting with content, and setting event reminders.

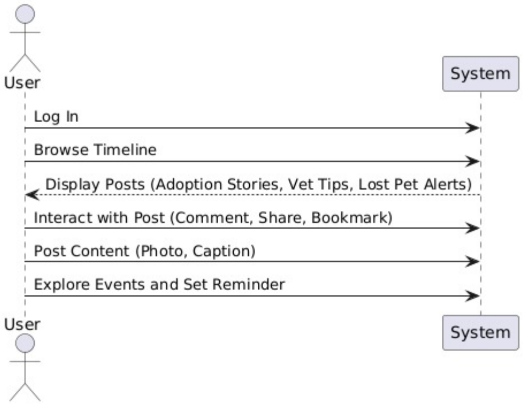

User
System
Log In
Browse Timeline
Display Posts (Adoption Stories, Vet Tips, Lost Pet Alerts)
Interact with Post (Comment, Share, Bookmark)
Post Content (Photo, Caption)
Explore Events and Set Reminder
User
System

Fig. 6. Sequence Diagram for Timeline Interaction

5) Emergency Assistance Process

The sequence diagram below illustrates the process of creating an emergency assistance announcement and the interactions involved.

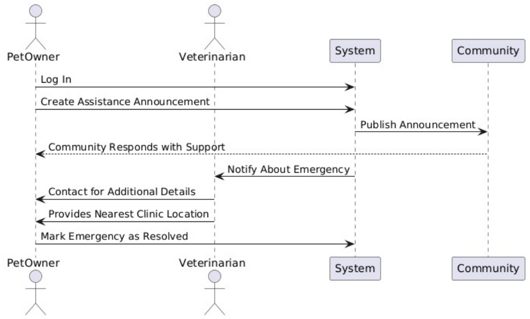

PetOwner
Veterinarian
System
Community
Log In
Create Assistance Announcement
Publish Announcement
Community Responds with Support
Notify About Emergency
Contact for Additional Details
Provides Nearest Clinic Location
Mark Emergency as Resolved
PetOwner
Veterinarian
System
Community

Fig. 7. Sequence Diagram for Emergency Assistance Process

D. Class Diagram

The class diagram defines the structure of the PetSoLive system, detailing its main classes, attributes, and relationships.

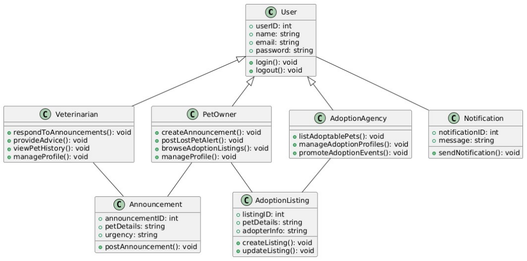


Fig. 8. Class Diagram for PetSoLive

XIV. System Architecture

This section provides an overview of the architectural design of the PetSoLive platform, which adopts a 3-Tier Architecture. The design focuses on modularity, scalability, and maintainability while integrating Windows IIS for efficient file storage management.

A. Overview

The PetSoLive platform leverages a layered architecture to ensure a clear separation of responsibilities, enhancing the system's scalability and maintainability. The integration of Windows IIS enables secure and efficient handling of user-uploaded files, such as photos for adoption listings and assistance announcements.

As shown in Figure 9, the platform adopts a 3-Tier Architecture that separates responsibilities into Presentation, Application, and Data layers.

B. Presentation Layer

Purpose: The presentation layer provides the user interface for interacting with the platform.

Responsibilities:

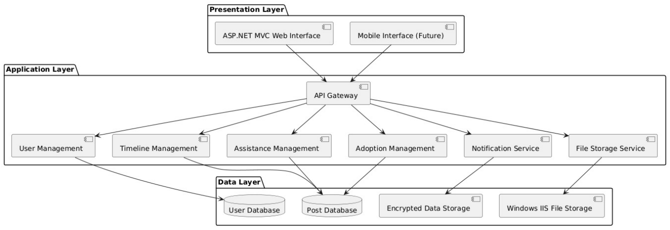


Fig. 9. Architectural Design of PetSoLive

- Displaying the timeline with posts such as adoption stories, assistance announcements, and lost pet alerts.
- Enabling user actions, including creating, filtering, and managing posts.
- Managing notifications, event reminders, and user profile interactions.

Technologies:

- **ASP.NET MVC:** Framework for building the web interface.
- **HTML5/CSS3:** For structuring and styling user interfaces.
- **JavaScript:** For interactivity and client-side operations.

C. Application Layer

Purpose: The application layer acts as the business logic layer, processing workflows and managing communication between the presentation and data layers.

Responsibilities:

- Handling user authentication, session management, and data validation.
- Managing assistance announcements, adoption posts, and notifications.
- Coordinating with the Windows IIS file storage system for file uploads and retrievals.
- Providing APIs for communication with the front-end.

Technologies:

- **ASP.NET Core:** Backend framework for implementing business logic.
- **RESTful APIs:** Enabling seamless communication between the frontend and backend.
- **Windows IIS Servers:** Hosting file storage services and facilitating secure uploads/downloads.

D. Data Layer

Purpose: The data layer handles secure storage and retrieval of structured data and media files.

Responsibilities:

- Storing structured data such as user accounts, pet profiles, and posts.

- Managing media files, including photos and documents, through Windows IIS.
- Ensuring data security using encryption techniques and regular backups.

Technologies:

- **Microsoft SQL Server:** Relational database for structured data storage.
- **Windows IIS:** Manages file storage to ensure scalability and secure access.
- **Encryption Standards:** AES for protecting sensitive information.

E. Advantages of Including Windows IIS

- **Scalable File Management:** Windows IIS ensures secure and efficient handling of user-uploaded files, such as images for posts and announcements.
- **Seamless Integration:** IIS integrates seamlessly with ASP.NET Core, providing a unified backend for file management and data handling.
- **Enhanced Performance:** IIS improves file retrieval speeds and reduces server load by delegating file management to a dedicated service.

XV. System Evolution

The system evolution section outlines the fundamental assumptions on which the PetSoLive platform is built and explores the potential changes expected in response to hardware advancements, evolving user needs, and emerging technologies. This section aims to guide system designers in making decisions that ensure the platform's scalability and adaptability to future requirements.

A. Fundamental Assumptions

The following assumptions form the foundation of the PetSoLive system:

- **Increasing User Base:** The platform assumes steady growth in the number of users, including pet owners, veterinarians, and adoption agencies, requiring scalable infrastructure.
- **Hardware Advancements:** The system is designed to leverage improvements in server hardware, such as increased processing power, storage capabilities, and faster network speeds.
- **Adoption of Mobile Devices:** A significant portion of users will access the platform through mobile devices, necessitating a responsive design and a future mobile application.
- **Regulatory Changes:** The platform anticipates compliance requirements with evolving data protection regulations, such as GDPR and similar laws in other regions.

B. Anticipated Changes

To ensure the platform remains relevant and effective, the following changes are anticipated over time:

- **Hardware Evolution:** Advances in cloud computing and storage technologies will enable the platform to handle increased user activity and larger volumes of data with improved efficiency.
- **Changing User Needs:** As user demands grow, new features such as AI-powered recommendations, real-time video consultations with veterinarians, and integration with IoT devices are expected to become essential.
- **Increased Security Requirements:** With the rise in cyber threats, the platform must adopt advanced encryption protocols and continuous monitoring to ensure data protection.
- **Expanded Language Support:** As the user base diversifies, the platform will need to support additional languages and cultural preferences for accessibility.
- **Integration with Emerging Technologies:** The adoption of technologies like blockchain for secure transaction logging and machine learning for behavior prediction is anticipated.

C. Design Considerations for Future Changes

To accommodate these anticipated changes, the following design considerations are prioritized:

- **Modular Architecture:** The platform employs a modular design to enable the seamless addition or replacement of components without disrupting existing functionalities.
- **Scalability:** Cloud-based infrastructure is used to support horizontal and vertical scaling as the user base and data volume grow.
- **Backward Compatibility:** Future updates will ensure compatibility with existing features to prevent disruptions for current users.
- **Extensibility:** APIs and interfaces are designed to facilitate the integration of third-party services and emerging technologies.
- **Continuous Feedback Loop:** Regular feedback from users and stakeholders will guide iterative development and adaptation to changing needs.

This section aims to provide a roadmap for the system's long-term evolution, ensuring that the platform remains adaptable, secure, and user-focused as new challenges and opportunities arise.

XVI. Risk Management

This section outlines the identified risks in the PetSoLive project, their potential impacts, and proposed alternative scenarios. The project management tool utilized is the GitHub Kanban Board, and the project code is stored in a GitHub repository.

A. *Risks Related to Project Management Tool and Code Repository*

Risk: Potential issues accessing the GitHub Kanban Board and repository (e.g., network disruptions or access permissions). **Alternative Scenarios:**

- Regularly maintain a local backup of project progress plans and task lists.
- Prepare a temporary project management structure on an alternative platform (e.g., Trello or Asana) in case of emergencies.
- Backup the code daily using external tools (e.g., Google Drive or OneDrive).

B. *Technical Risks During Development*

Risk: Errors arising from a lack of expertise in the technical stack (e.g., inefficiencies due to limited knowledge of ASP.NET Core). **Alternative Scenarios:**

- Conduct quick training sessions or share resources (e.g., ASP.NET Core documentation and video tutorials) for team members as needed.
- Hold regular team meetings to facilitate knowledge sharing and optimize project progress.

C. *Non-Compliance with Project Schedule*

Risk: Inability to complete tasks within the planned timeline. **Alternative Scenarios:**

- Review progress on the GitHub Kanban Board weekly to monitor task completion rates.
- Prioritize delayed tasks and allocate additional resources (e.g., having multiple team members work on a single task).

D. *Server and Hosting Issues*

Risk: Performance issues or crashes on the hosting servers. **Alternative Scenarios:**

- Keep alternative hosting services (e.g., Firebase or AWS) available as backups.
- Integrate monitoring and error-reporting tools (e.g., New Relic or Sentry) into the system.
- Automate user notifications to inform them about any service disruptions.

E. *Security Risks Related to User Data*

Risk: Potential breaches or mishandling of sensitive user data. **Alternative Scenarios:**

- Implement strong encryption techniques for sensitive data storage and transmission.
- Regularly conduct security audits and penetration testing.
- Provide users with transparency regarding data handling policies and obtain necessary consents.

XVII. Project Management

In order to successfully complete the PetSoLive project, a well-defined project management process has been implemented, utilizing effective tools and strategies for task management, scheduling, and communication.

A. *Project Management Tools*

For efficient project tracking and management, the GitHub Kanban Board was utilized. This tool facilitated the organization and monitoring of tasks throughout the project:

- **Task Management:** Tasks were divided into columns such as *To Do*, *In Progress*, and *Done* to streamline the tracking process.
- **Time Management:** Weekly sprints were planned to ensure the project stayed on track and deadlines were met.
- **Communication:** GitHub Issues were used to handle team communication and to provide a platform for feedback and suggestions.

B. *Project Planning Process*

The project was planned and executed in the following steps:

1) Requirement analysis and defining the project scope.
2) Technology selection and infrastructure design.
3) Task allocation and creation of a detailed project timeline.
4) Sequential implementation of development and testing phases.

Regular progress checks were conducted using the GitHub Kanban Board to ensure that tasks were being completed within the expected timeframe.

C. *Team Collaboration and Task Allocation*

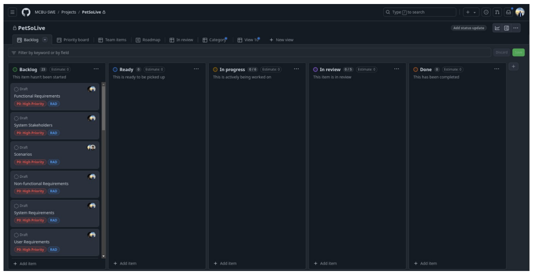
The project was divided into smaller tasks, which were assigned based on team members' expertise and availability. Communication channels, such as regular meetings and GitHub comments, ensured smooth collaboration and issue resolution. Additionally, code reviews were performed regularly to maintain high-quality standards.

D. *Kanban Board Progress*

To visualize the progression of tasks, the GitHub Kanban Board was updated regularly. Below are snapshots from different stages of the project:

1) Initial Setup

The initial setup included task identification and allocation into the *To Do* column.



Backlog 23 Estimate: 0 — This item hasn't been started
- Draft — Functional Requirements — P0: High Priority, RAD
- Draft — System Stakeholders — P0: High Priority, RAD
- Draft — Scenarios — P0: High Priority, RAD
- Draft — Non-functional Requirements — P0: High Priority, RAD
- Draft — System Requirements — P0: High Priority, RAD
- Draft — User Requirements — P0: High Priority, RAD

Ready 0 Estimate: 0 — This is ready to be picked up

In progress 0 / 6 Estimate: 0 — This is actively being worked on

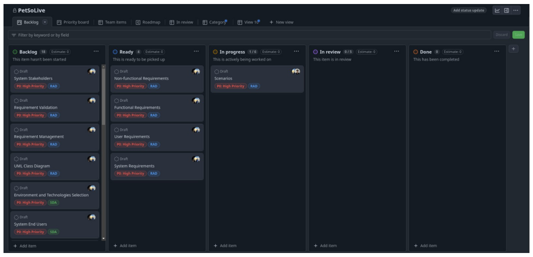
In review 0 / 5 Estimate: 0 — This item is in review

Done 0 Estimate: 0 — This has been completed

Fig. 10. Kanban Board – Initial Setup

2) Second-Development Progress

During the second-development phase, tasks moved from *In Progress* to *Done* as features were implemented and tested.



Backlog 18 Estimate: 0 — This item hasn't been started
- Draft — System Stakeholders — P0: High Priority, RAD
- Draft — Requirement Validation — P0: High Priority, RAD
- Draft — Requirement Management — P0: High Priority, RAD
- Draft — UML Class Diagram — P0: High Priority, RAD
- Draft — Environment and Technologies Selection — P0: High Priority, SDA
- Draft — System End Users — P0: High Priority, SDA

Ready 4 Estimate: 0 — This is ready to be picked up
- Draft — Non-functional Requirements — P0: High Priority, RAD
- Draft — Functional Requirements — P0: High Priority, RAD
- Draft — User Requirements — P0: High Priority, RAD
- Draft — System Requirements — P0: High Priority, RAD

In progress 1 / 6 Estimate: 0 — This is actively being worked on
- Draft — Scenarios — P0: High Priority, RAD

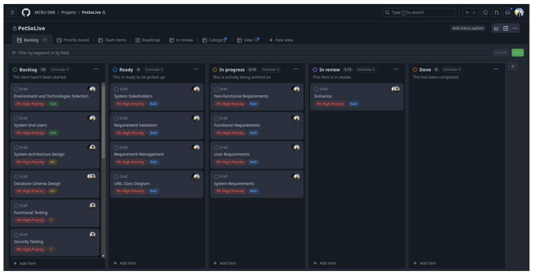
In review 0 / 5 Estimate: 0 — This item is in review

Done 0 Estimate: 0 — This has been completed

Fig. 11. Kanban Board – Second Development

3) Third-Development Progress

In the third-development phase, most tasks were completed, and the *Done* column reflected the project's progress.



Backlog 14 Estimate: 0 — This item hasn't been started
- Draft — Environment and Technologies Selection — P0: High Priority, SDA
- Draft — System End Users — P0: High Priority, SDA
- Draft — System Architecture Design — P0: High Priority, DD
- Draft — Database Schema Design — P0: High Priority, DD
- Draft — Functional Testing — P0: High Priority, T
- Draft — Security Testing — P0: High Priority, T

Ready 4 Estimate: 0 — This is ready to be picked up
- Draft — System Stakeholders — P0: High Priority, RAD
- Draft — Requirement Validation — P0: High Priority, RAD
- Draft — Requirement Management — P0: High Priority, RAD
- Draft — UML Class Diagram — P0: High Priority, RAD

In progress 4 / 6 Estimate: 0 — This is actively being worked on
- Draft — Non-functional Requirements — P0: High Priority, RAD
- Draft — Functional Requirements — P0: High Priority, RAD
- Draft — User Requirements — P0: High Priority, RAD
- Draft — System Requirements — P0: High Priority, RAD

In review 1 / 5 Estimate: 0 — This item is in review
- Draft — Scenarios — P0: High Priority, RAD

Done 0 Estimate: 0 — This has been completed

Fig. 12. Kanban Board – Third Development

E. Project Links

The following links provide access to the Kanban Board and the PetSoLive project repository for detailed reference:

- **Kanban Board:** https://github.com/orgs/MCBU-SWE/projects/4
- **PetSoLive Project:** https://github.com/MCBU-SWE/Online-Pet-Adopting-Site

XVIII. Future Enhancements

To ensure PetSoLive evolves to meet user demands and incorporates emerging technologies, the following enhancements are proposed for future development:

- **AI-Powered Recommendations:** Integrate machine learning models to suggest relevant adoption listings, veterinary advice, or community events based on user activity and preferences.
- **Mobile Application:** Develop a dedicated mobile application to enhance accessibility and improve user experience on mobile devices.
- **Video Consultation with Veterinarians:** Enable real-time video consultations for veterinary support, especially in emergency cases.
- **Gamification Features:** Introduce achievement badges, progress tracking for pet health, and other gamified elements to boost user engagement.
- **Integration with IoT Devices:** Link with smart pet devices (e.g., GPS trackers, health monitors) to provide real-time updates and insights.
- **Expanded Language Support:** Add support for more languages to cater to a diverse global audience.
- **Data Analytics Dashboard:** Provide administrators with detailed analytics on platform usage, trends, and performance for better decision-making.
- **Community Forums:** Create dedicated forums for pet-related discussions, enabling users to share tips, experiences, and advice.
- **Advanced Search Features:** Incorporate natural language processing (NLP) for enhanced search capabilities within adoption listings and assistance announcements.
- **Customizable Notifications:** Allow users to personalize notification preferences for specific events or updates.

These future enhancements aim to expand the platform's functionality, improve user engagement, and solidify PetSoLive's position as a comprehensive tool for pet welfare.

XIX. Appendices

This section provides detailed, specific information related to the development and operation of the PetSoLive application, including hardware requirements and database specifications.

A. Hardware Requirements

The following table outlines the minimal and optimal hardware configurations required for running the PetSoLive platform effectively:

TABLE I
Hardware Requirements

Component	Minimum Configuration	Optimal Configuration
Processor	Dual-core 2.4 GHz	Quad-core 3.0 GHz
Memory (RAM)	4 GB	16 GB
Storage	50 GB HDD	250 GB SSD
Operating System	Windows Server 2016	Windows Server 2022
Network Bandwidth	10 Mbps	100 Mbps

B. Database Design

The database for PetSoLive is structured to support efficient data storage, retrieval, and management. The logical organization and relationships between data entities are described below:

- **Users Table:** Stores information about users, including:
  - **Fields:** User ID, Name, Email, Encrypted Password, Role, Active Status.
- **Pets Table:** Contains details of pets available for adoption or related to assistance announcements:
  - **Fields:** Pet ID, Name, Species, Breed, Age, Health Status, Associated User ID.
- **Announcements Table:** Tracks emergency and assistance announcements posted by users:
  - **Fields:** Announcement ID, Pet ID, Urgency Level, Details, Timestamp, Status.
- **Adoption Listings Table:** Manages adoption listings:
  - **Fields:** Listing ID, Pet ID, Description, Adopter ID (if adopted), Listing Status.
- **Messages Table:** Records communication between users:
  - **Fields:** Message ID, Sender ID, Receiver ID, Content, Timestamp.
- **Vaccination Records Table:** Maintains pet vaccination schedules:
  - **Fields:** Record ID, Pet ID, Vaccine Name, Due Date, Veterinarian ID.

C. Entity-Relationship Diagram (ERD)

The following diagram represents the logical relationships between the main entities in the PetSoLive database:

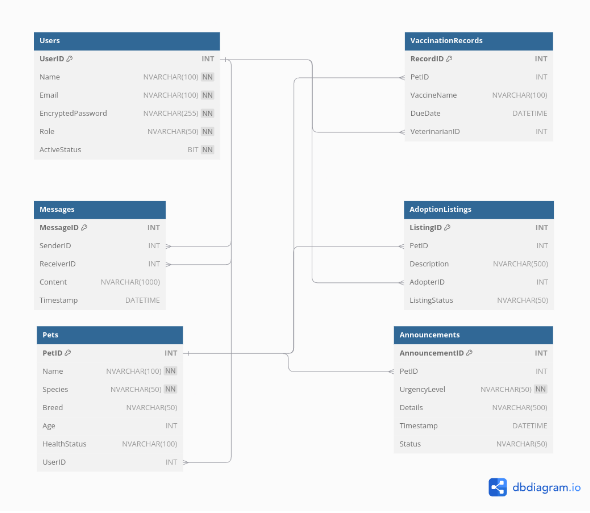


Fig. 13. Entity-Relationship Diagram (ERD) for PetSoLive Database

D. Additional Specifications

- **Backup and Recovery:** Scheduled daily backups are configured to ensure data integrity and availability during unforeseen events.
- **Scalability:** The database is designed to scale horizontally, supporting increased data volumes and user activity.
- **Security Measures:** Role-based access control (RBAC) and encryption are implemented to secure sensitive data and ensure compliance with industry standards.

This appendix serves as a reference for technical configurations and database organization to support the ongoing development and maintenance of the PetSoLive platform.

XX. Index

This section provides various indexes for the document, making it easier for readers to locate specific information. The following indexes are included:

A. Index of Diagrams

A list of all diagrams included in the document, along with their page numbers for quick reference.

B. Index of Tables

A list of all tables included in the document for quick reference.

XXI. Task Allocation

The responsibilities for the PetSoLive project have been distributed among the team members based on their expertise and project requirements. Below is a detailed breakdown of each team member's role and assigned tasks:

A. *Furkan BAYTAK*

- **Project Documentation and Report Writing:** Leading the preparation of the project report and technical documentation in line with IEEE standards, ensuring all project phases are thoroughly documented.
- **Requirements Engineering:** Gathering and analyzing user needs to define functional and non-functional requirements, ensuring they align with the project objectives.
- **Front-End Development:** Overseeing the development of the platform's user interface, focusing on creating a visually appealing and user-friendly design.
- **Kanban Board Management:** Maintaining and updating the Kanban board to ensure smooth task tracking and workflow management.
- **Project Coordination:** Coordinating with team members to align tasks, timelines, and overall project progress.

B. *Furkan BULUT*

- **Back-End Development:** Developing the server-side functionality, including the creation and management of APIs, ensuring efficient system performance.
- **Database Design and Optimization:** Designing the database structure, optimizing queries, and ensuring data integrity and security throughout the platform.
- **System Integration:** Ensuring smooth integration between the front-end and back-end components, guaranteeing seamless functionality.
- **Technical Report Contributions:** Documenting technical details related to back-end implementation, database design, and integration processes.
- **Test Implementation:** Conducting unit and integration testing to verify the performance and functionality of the platform.

C. *Furkan ÖZKAYA*

- **Assistance with Report Writing:** Contributing to the technical documentation and preparation of the final project report, especially in the areas of design and analysis.
- **Scenario and Diagram Development:** Creating detailed user scenarios, workflows, and system diagrams, such as use case and sequence diagrams, to clarify system design and behavior.
- **Front-End Support:** Assisting in the development of the platform's user interface, focusing on implementing interactive and responsive elements.
- **Kanban Board Operations:** Assisting in managing task statuses and ensuring workflows are up to date on the Kanban board.
- **Testing and Quality Assurance:** Preparing test scenarios and executing functionality tests to validate system behavior and user requirements.

The tasks have been allocated to leverage the strengths and expertise of each team member, ensuring the successful completion of the PetSoLive project within the planned timeline.

XXII. Meeting Schedule

The following table outlines the meeting schedule for the PetSoLive project, detailing the key objectives and topics covered in each meeting:

A. *Meeting Dates and Details*

- **November 5, 2024:**
  - Initial review of project scope and requirements.
  - Discussing task allocation and team responsibilities.
  - Setting up the GitHub repository and Kanban board.
- **November 12, 2024:**
  - Progress evaluation on front-end development.
  - Reviewing initial drafts of use case diagrams and system workflows.
  - Aligning team members on database schema design.
- **November 19, 2024:**
  - Reviewing progress on front-end development.
  - Gathering feedback on the first draft of the project report.
  - Discussing potential improvements to user scenarios and interface designs.
- **November 26, 2024:**
  - Assessing the usability and functionality of the user interface.
  - Addressing any inconsistencies in the use cases and updating diagrams as needed.
- **November 27, 2024:**
  - Reviewing the near-final version of the project report.
  - Planning the defining test cases and scenarios.
- **November 28, 2024:**
  - Conducting a final review of the project report and technical documentation.
  - Reviewing and evaluating the progress on the front-end user interface, including discussions on the layout and functionality. (Figure 14 for the current interface design.)

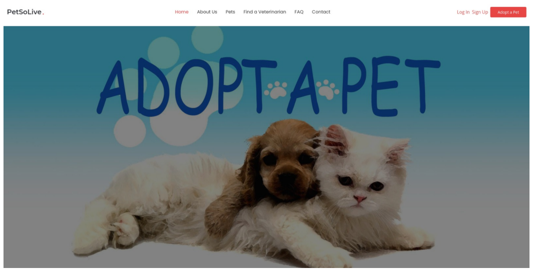


Fig. 14. Front-End Interface Design Example

B. Objective of Meetings

These meetings ensured that the team remained aligned on project goals, tracked progress effectively, and resolved challenges in a timely manner. Each meeting contributed to maintaining the project's momentum and ensuring the successful completion of all deliverables.

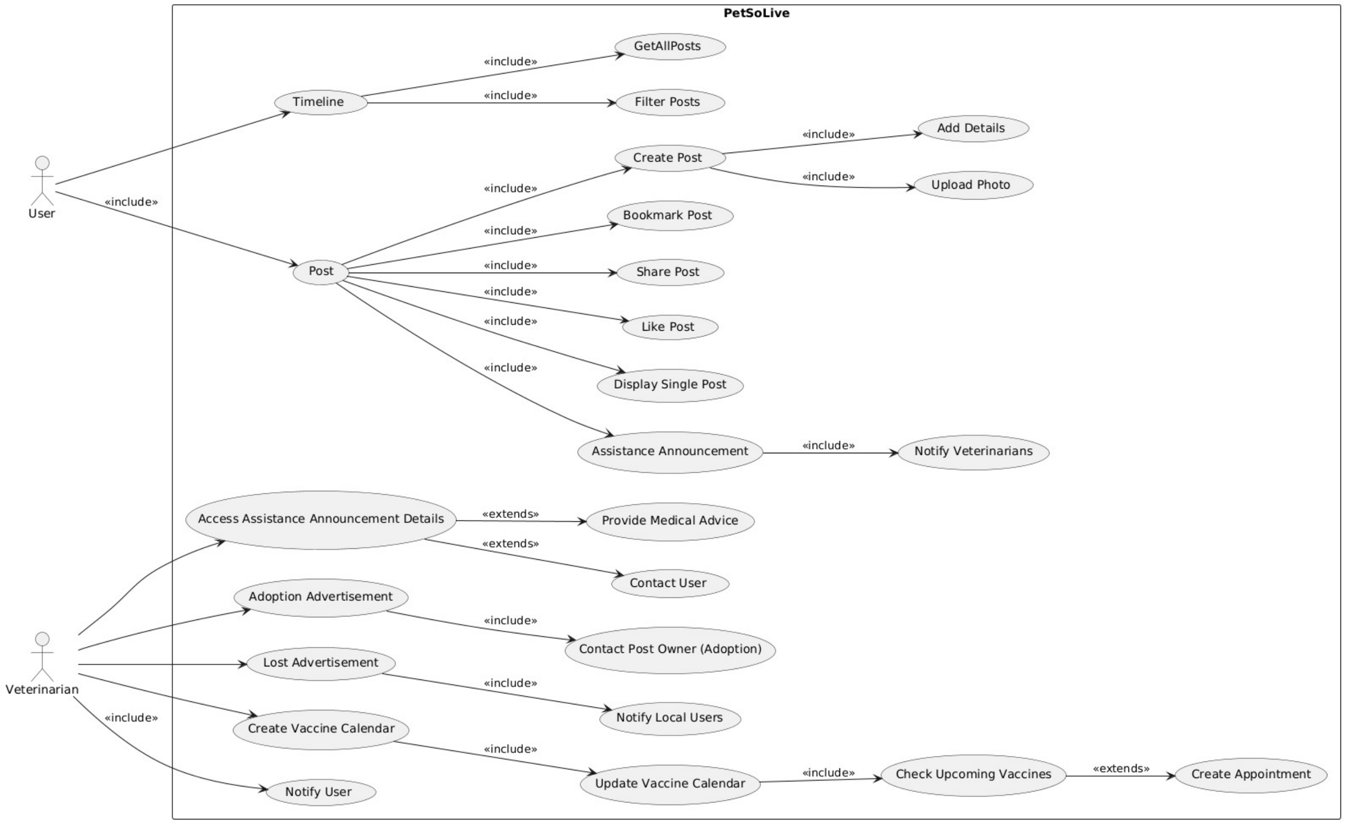


Figure 1: Use Case Diagram for PetSoLive

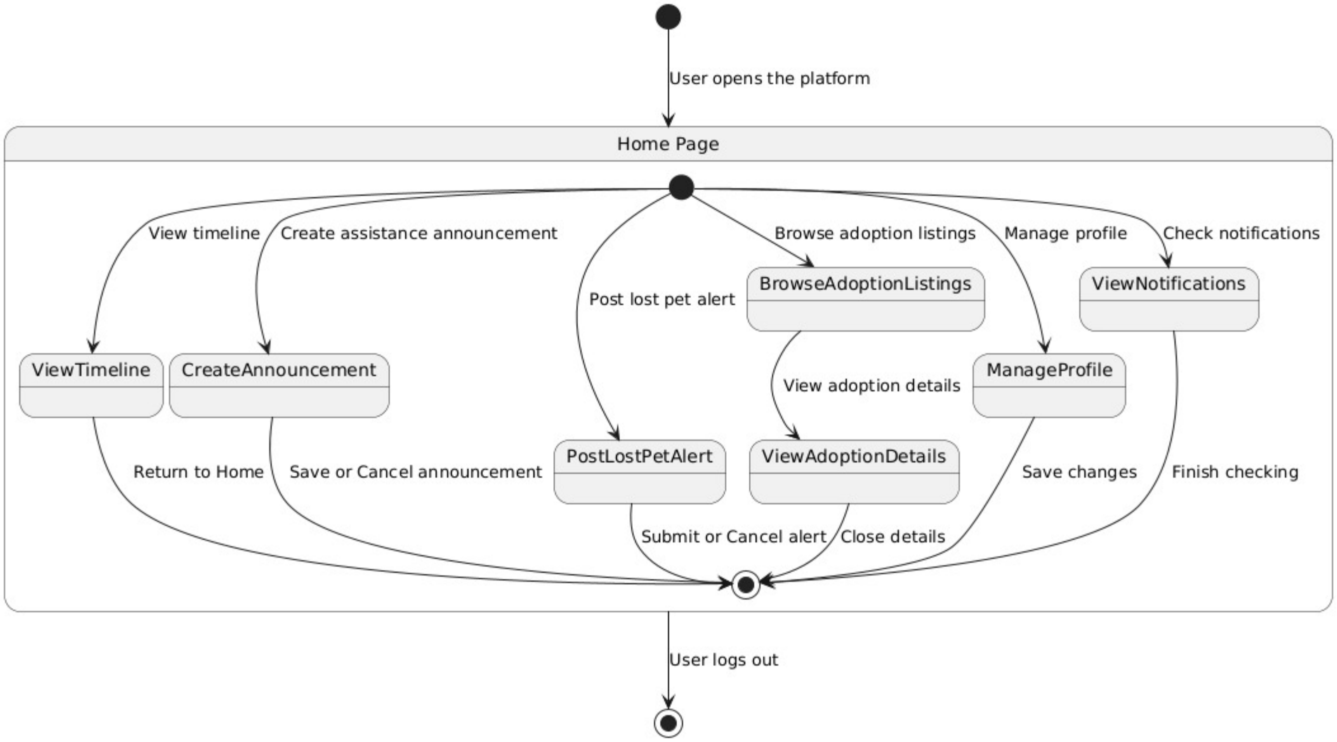


Figure 2: State Diagram for PetSoLive

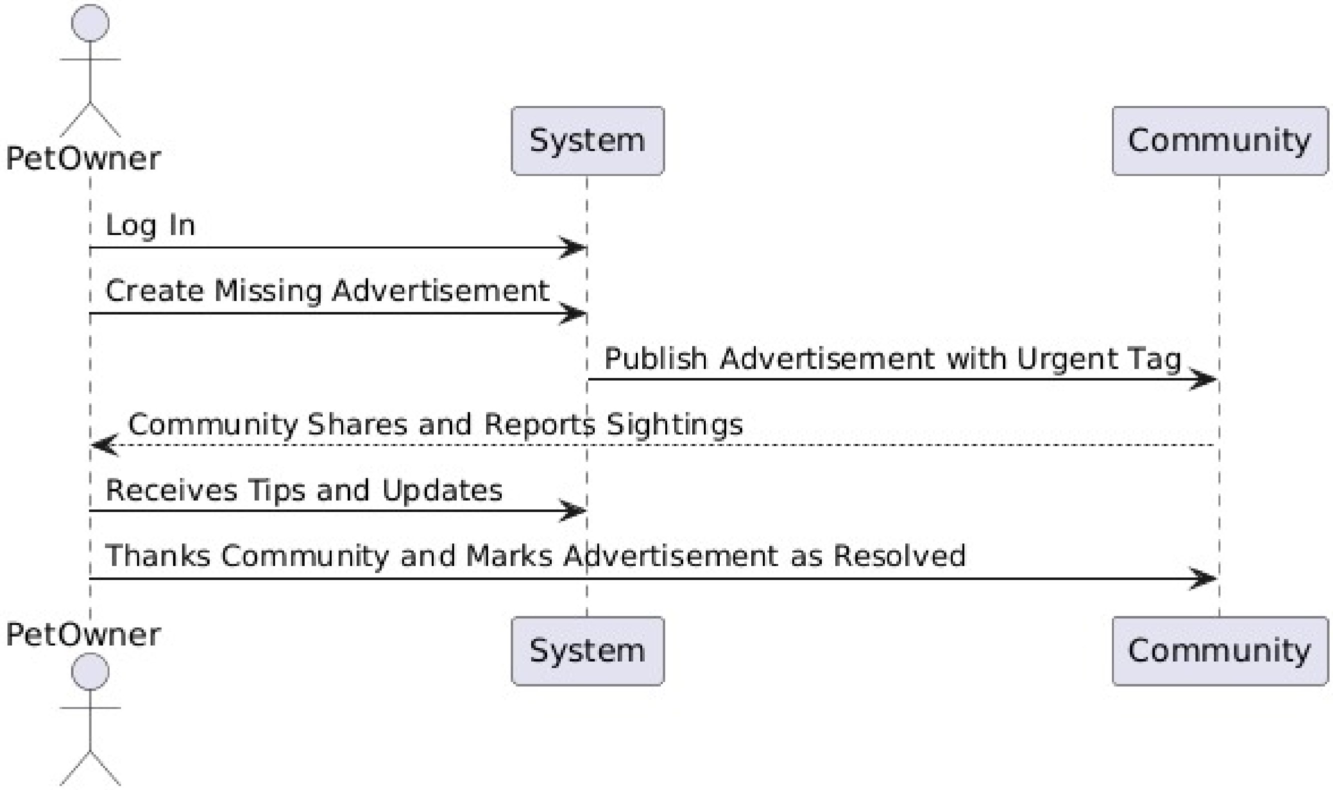

PetOwner
System
Community
Log In
Create Missing Advertisement
Publish Advertisement with Urgent Tag
Community Shares and Reports Sightings
Receives Tips and Updates
Thanks Community and Marks Advertisement as Resolved
PetOwner
System
Community

Figure 3: Sequence Diagram for Missing Advertisement Process

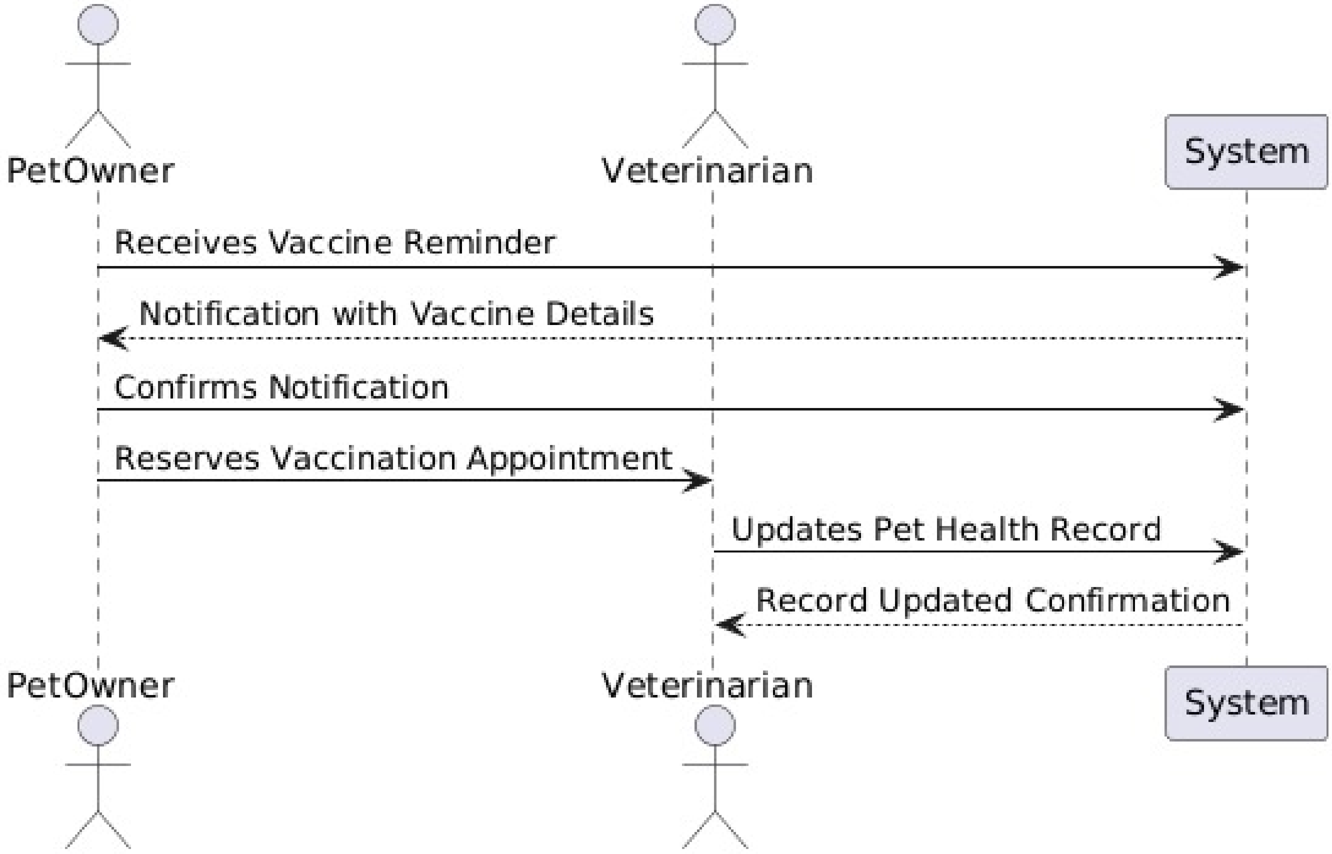

PetOwner
Veterinarian
System
Receives Vaccine Reminder
Notification with Vaccine Details
Confirms Notification
Reserves Vaccination Appointment
Updates Pet Health Record
Record Updated Confirmation
PetOwner
Veterinarian
System

Figure 4: Sequence Diagram for Vaccine Reminder Process

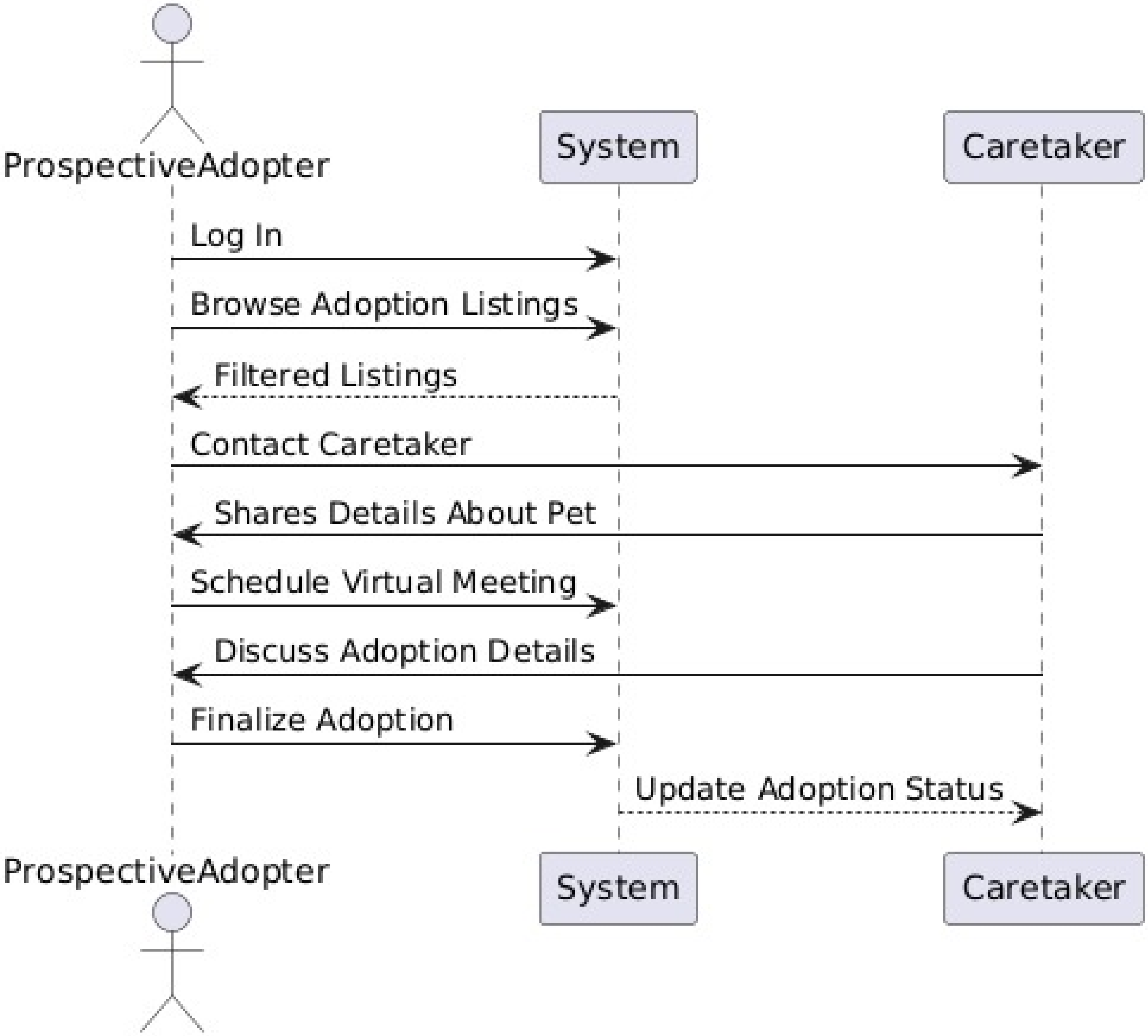


Figure 5: Sequence Diagram for Adoption Process

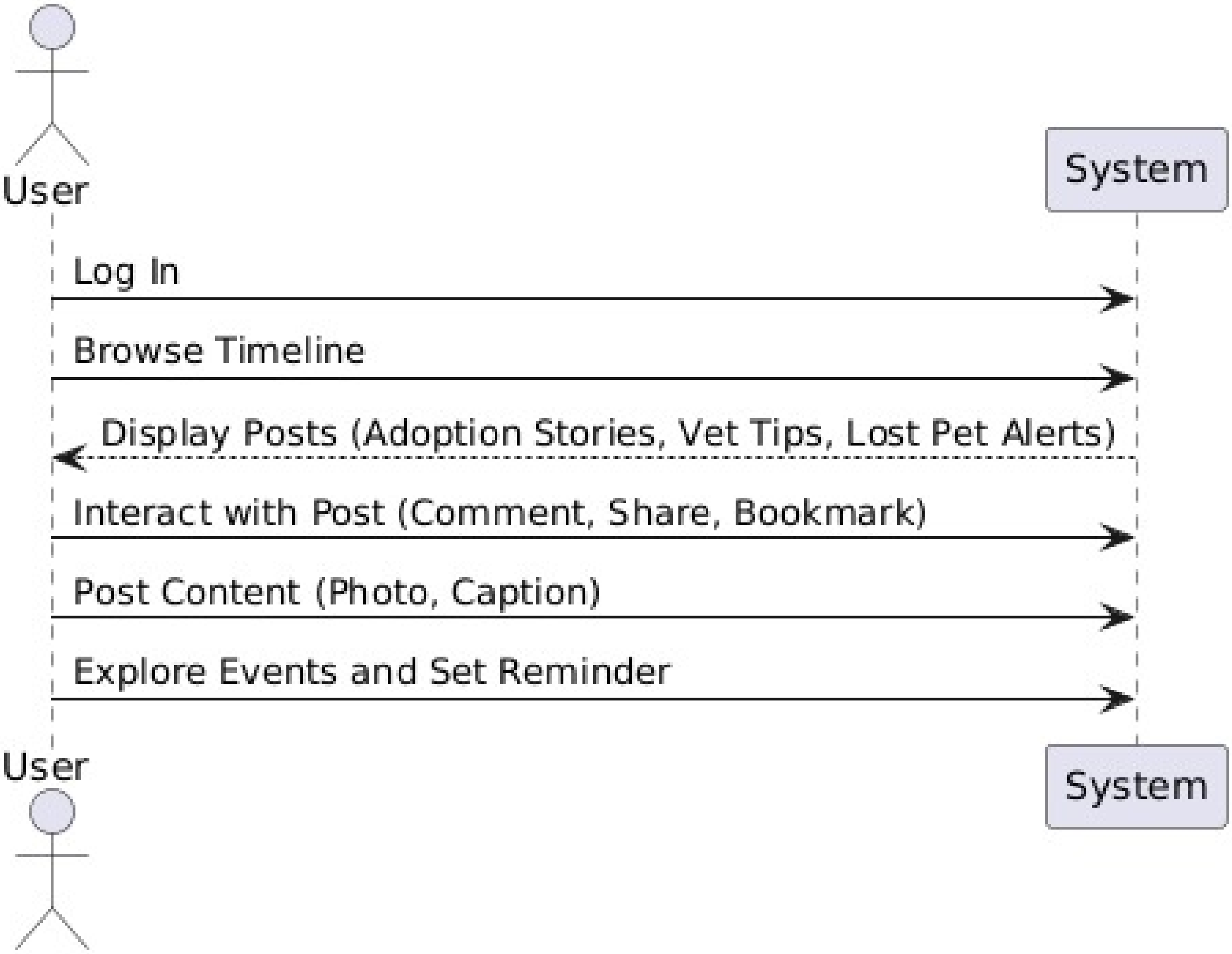

User
System
Log In
Browse Timeline
Display Posts (Adoption Stories, Vet Tips, Lost Pet Alerts)
Interact with Post (Comment, Share, Bookmark)
Post Content (Photo, Caption)
Explore Events and Set Reminder
User
System

Figure 6: Sequence Diagram for Timeline Interaction

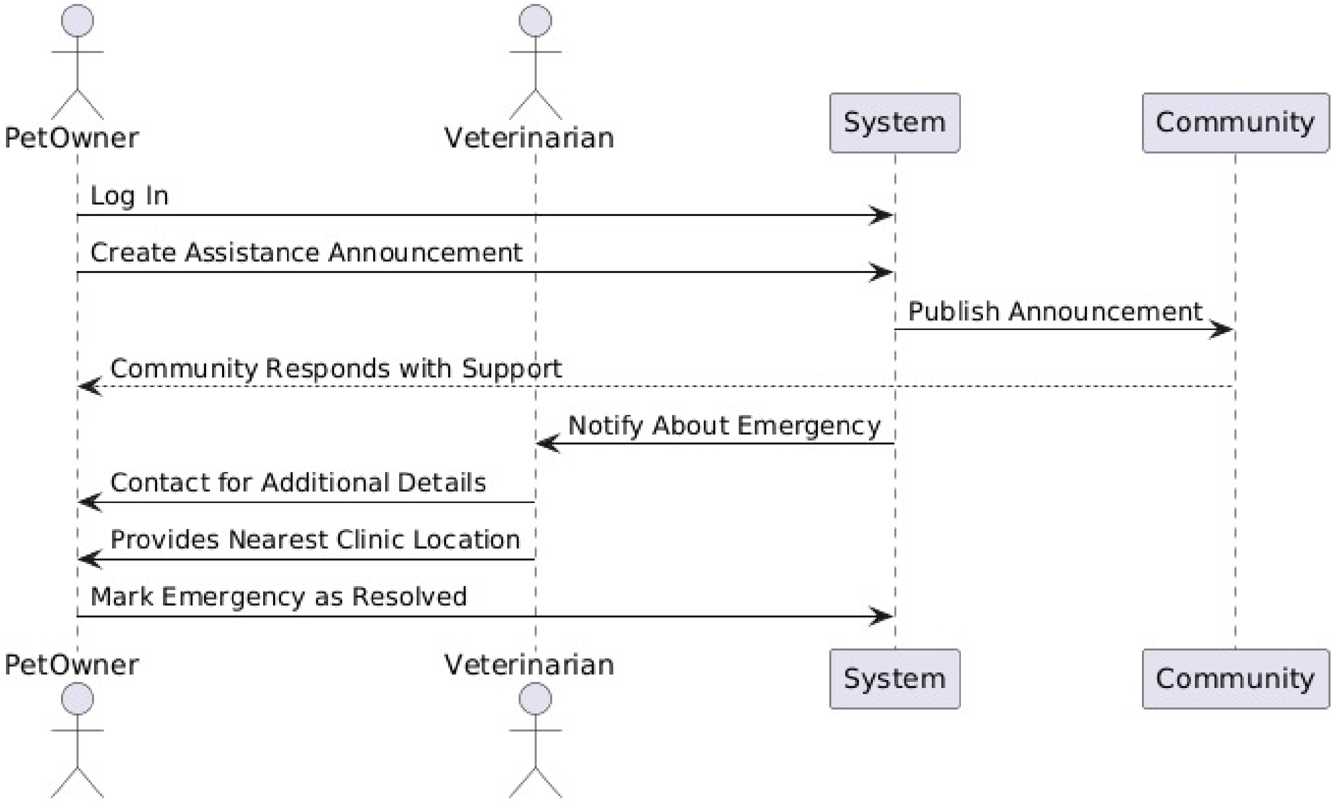

PetOwner
Veterinarian
System
Community
Log In
Create Assistance Announcement
Publish Announcement
Community Responds with Support
Notify About Emergency
Contact for Additional Details
Provides Nearest Clinic Location
Mark Emergency as Resolved
PetOwner
Veterinarian
System
Community

Figure 7: Sequence Diagram for Emergency Assistance Process

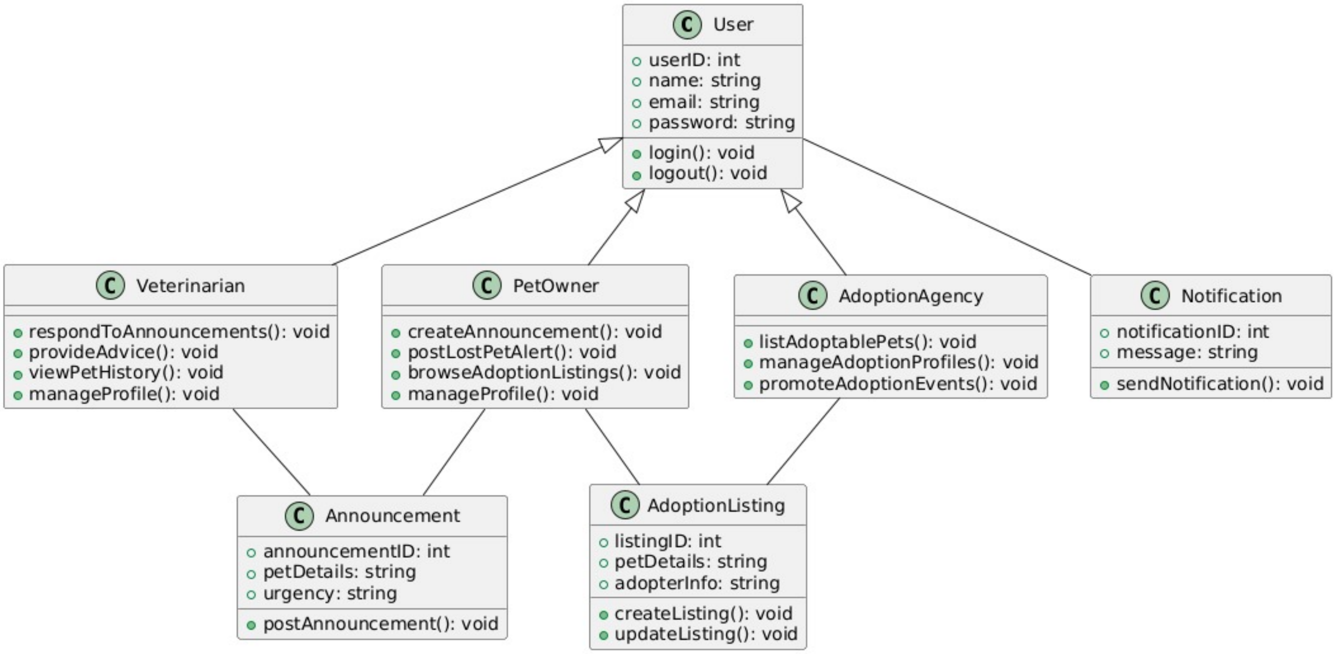
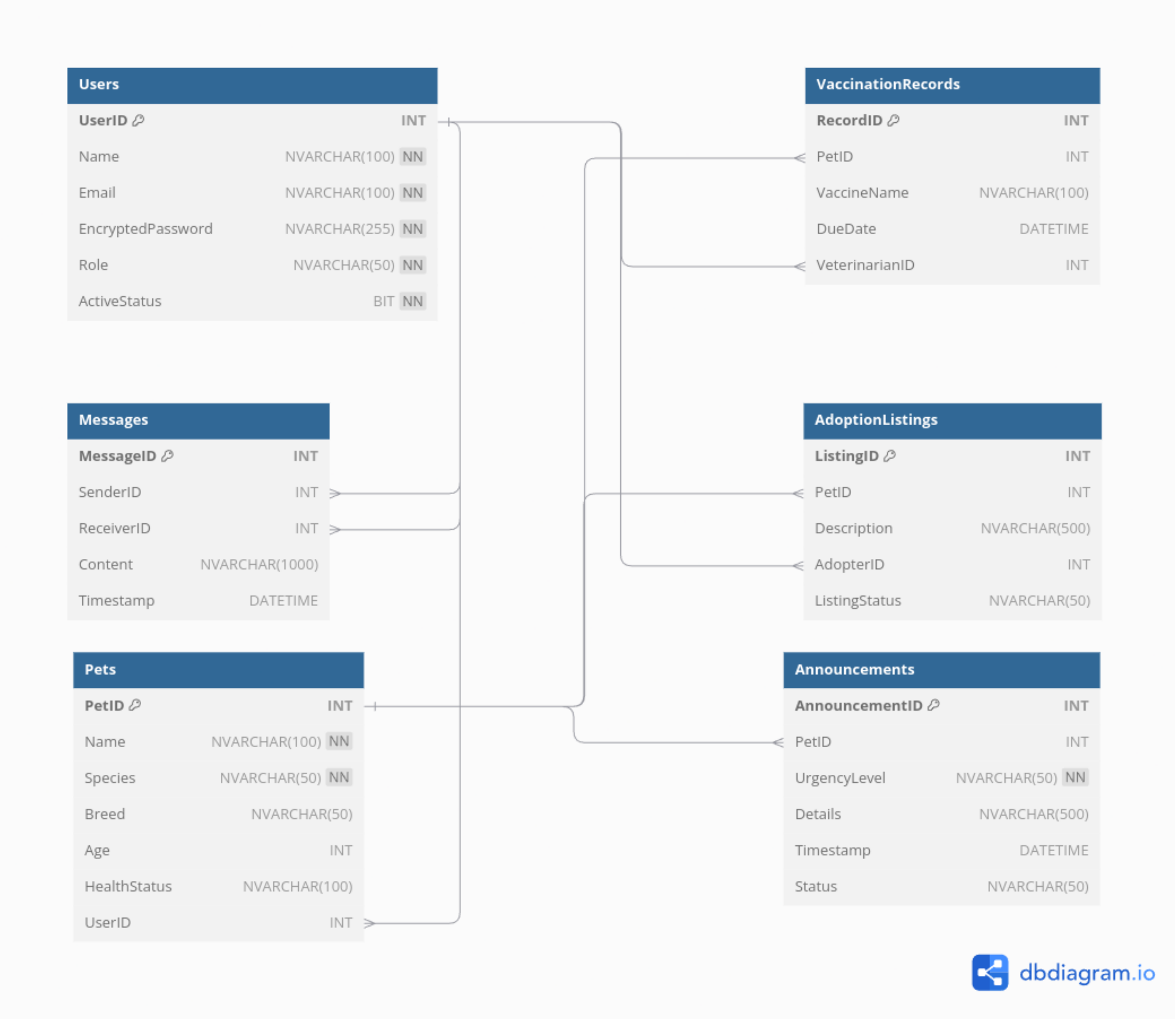


Figure 8: Class Diagram for PetSoLive



Users		
UserID	INT	
Name	NVARCHAR(100)	NN
Email	NVARCHAR(100)	NN
EncryptedPassword	NVARCHAR(255)	NN
Role	NVARCHAR(50)	NN
ActiveStatus	BIT	NN

VaccinationRecords	
RecordID	INT
PetID	INT
VaccineName	NVARCHAR(100)
DueDate	DATETIME
VeterinarianID	INT

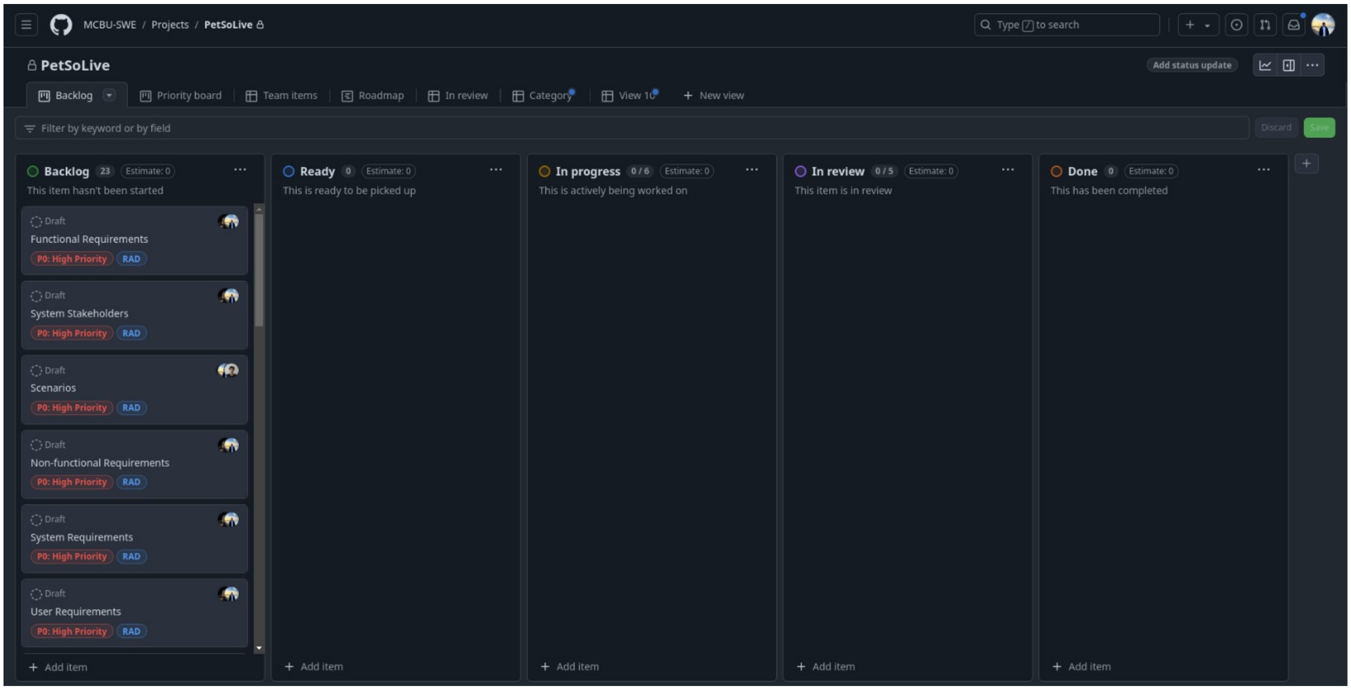
Messages	
MessageID	INT
SenderID	INT
ReceiverID	INT
Content	NVARCHAR(1000)
Timestamp	DATETIME

AdoptionListings	
ListingID	INT
PetID	INT
Description	NVARCHAR(500)
AdopterID	INT
ListingStatus	NVARCHAR(50)

Pets		
PetID	INT	
Name	NVARCHAR(100)	NN
Species	NVARCHAR(50)	NN
Breed	NVARCHAR(50)	
Age	INT	
HealthStatus	NVARCHAR(100)	
UserID	INT	

Announcements		
AnnouncementID	INT	
PetID	INT	
UrgencyLevel	NVARCHAR(50)	NN
Details	NVARCHAR(500)	
Timestamp	DATETIME	
Status	NVARCHAR(50)	

Figure 9: Entity-Relationship Diagram (ERD) for PetSoLive Database



Backlog 23 Estimate: 0
This item hasn't been started

Draft
Functional Requirements
P0: High Priority RAD

Draft
System Stakeholders
P0: High Priority RAD

Draft
Scenarios
P0: High Priority RAD

Draft
Non-functional Requirements
P0: High Priority RAD

Draft
System Requirements
P0: High Priority RAD

Draft
User Requirements
P0: High Priority RAD

Add item

Ready 0 Estimate: 0
This is ready to be picked up

Add item

In progress 0 / 6 Estimate: 0
This is actively being worked on

Add item

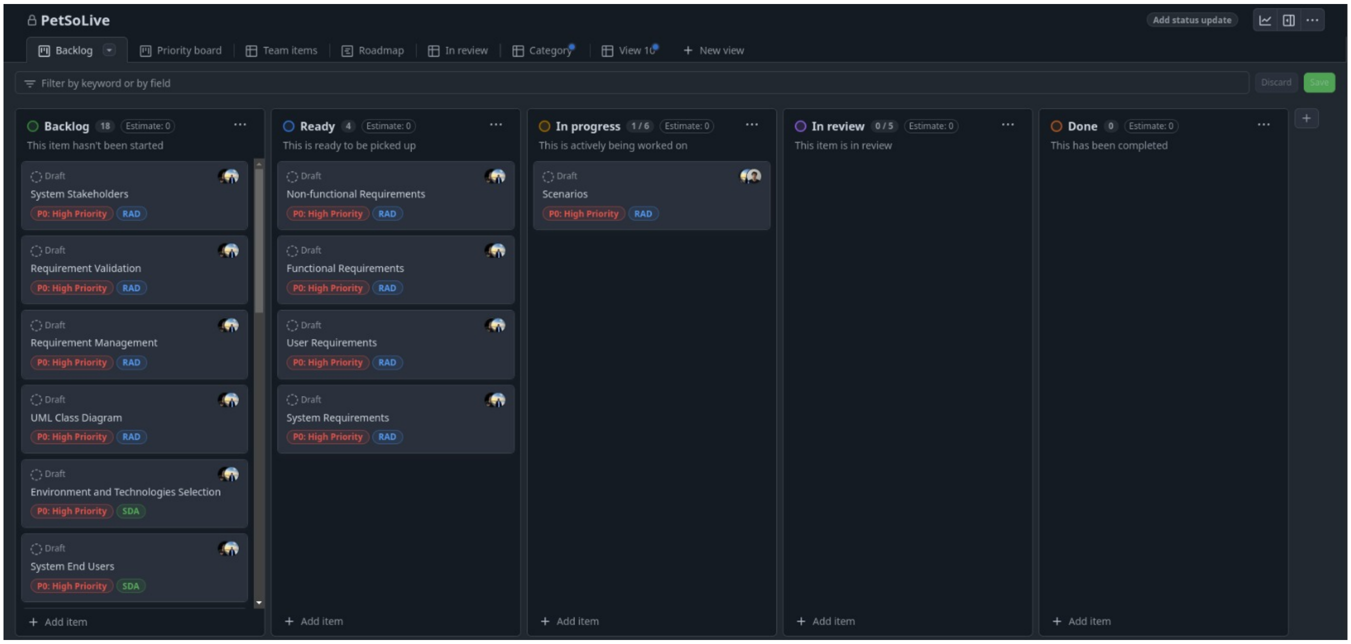
In review 0 / 5 Estimate: 0
This item is in review

Add item

Done 0 Estimate: 0
This has been completed

Add item

Figure 10: Kanban Board – Initial Setup



Backlog 18 Estimate: 0
This item hasn't been started

- Draft — System Stakeholders — P0: High Priority, RAD
- Draft — Requirement Validation — P0: High Priority, RAD
- Draft — Requirement Management — P0: High Priority, RAD
- Draft — UML Class Diagram — P0: High Priority, RAD
- Draft — Environment and Technologies Selection — P0: High Priority, SDA
- Draft — System End Users — P0: High Priority, SDA

Ready 4 Estimate: 0
This is ready to be picked up

- Draft — Non-functional Requirements — P0: High Priority, RAD
- Draft — Functional Requirements — P0: High Priority, RAD
- Draft — User Requirements — P0: High Priority, RAD
- Draft — System Requirements — P0: High Priority, RAD

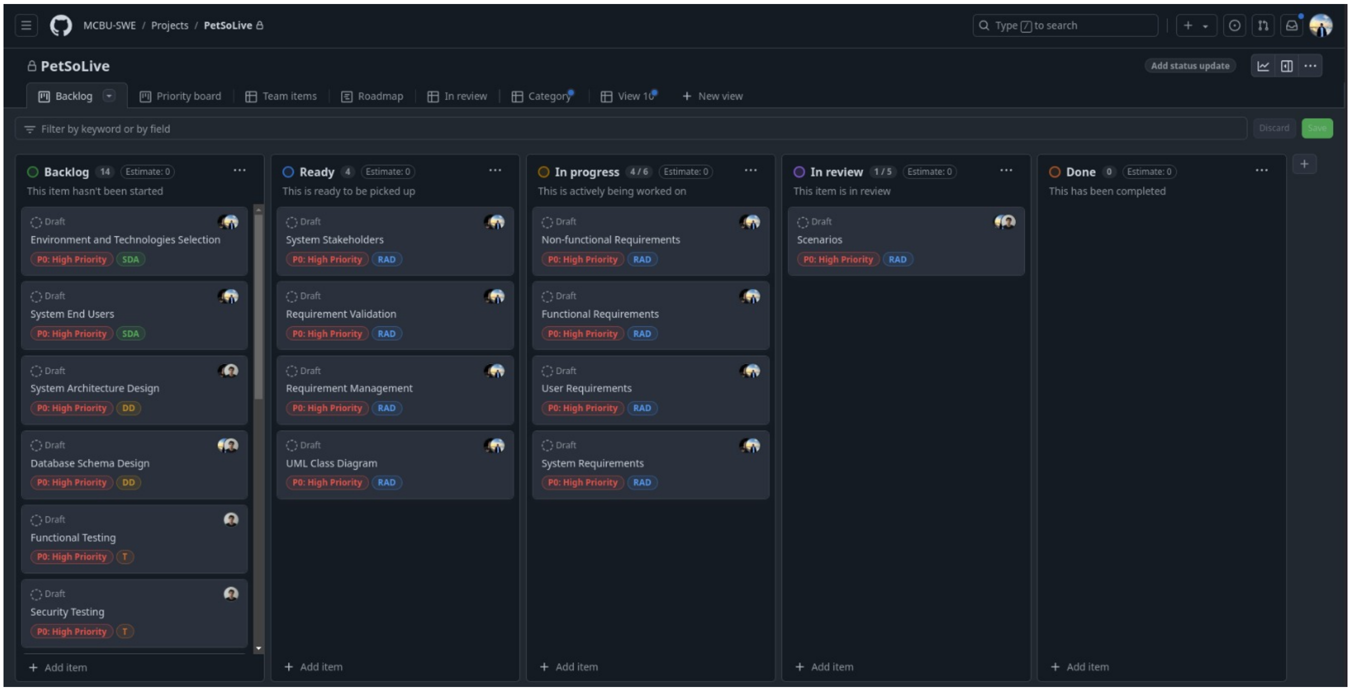
In progress 1 / 6 Estimate: 0
This is actively being worked on

- Draft — Scenarios — P0: High Priority, RAD

In review 0 / 5 Estimate: 0
This item is in review

Done 0 Estimate: 0
This has been completed

Figure 11: Kanban Board – Second Development Progress



MCBU-SWE / Projects / PetSoLive

Type / to search

PetSoLive

Add status update

Backlog | Priority board | Team items | Roadmap | In review | Category | View 10 | + New view

Filter by keyword or by field

Discard Save

Backlog 14 Estimate: 0

This item hasn't been started

Draft
Environment and Technologies Selection
P0: High Priority SDA

Draft
System End Users
P0: High Priority SDA

Draft
System Architecture Design
P0: High Priority DD

Draft
Database Schema Design
P0: High Priority DD

Draft
Functional Testing
P0: High Priority T

Draft
Security Testing
P0: High Priority T

+ Add item

Ready 4 Estimate: 0

This is ready to be picked up

Draft
System Stakeholders
P0: High Priority RAD

Draft
Requirement Validation
P0: High Priority RAD

Draft
Requirement Management
P0: High Priority RAD

Draft
UML Class Diagram
P0: High Priority RAD

+ Add item

In progress 4 / 6 Estimate: 0

This is actively being worked on

Draft
Non-functional Requirements
P0: High Priority RAD

Draft
Functional Requirements
P0: High Priority RAD

Draft
User Requirements
P0: High Priority RAD

Draft
System Requirements
P0: High Priority RAD

+ Add item

In review 1 / 5 Estimate: 0

This item is in review

Draft
Scenarios
P0: High Priority RAD

+ Add item

Done 0 Estimate: 0

This has been completed

+ Add item

Figure 12: Kanban Board – Third Development Progress

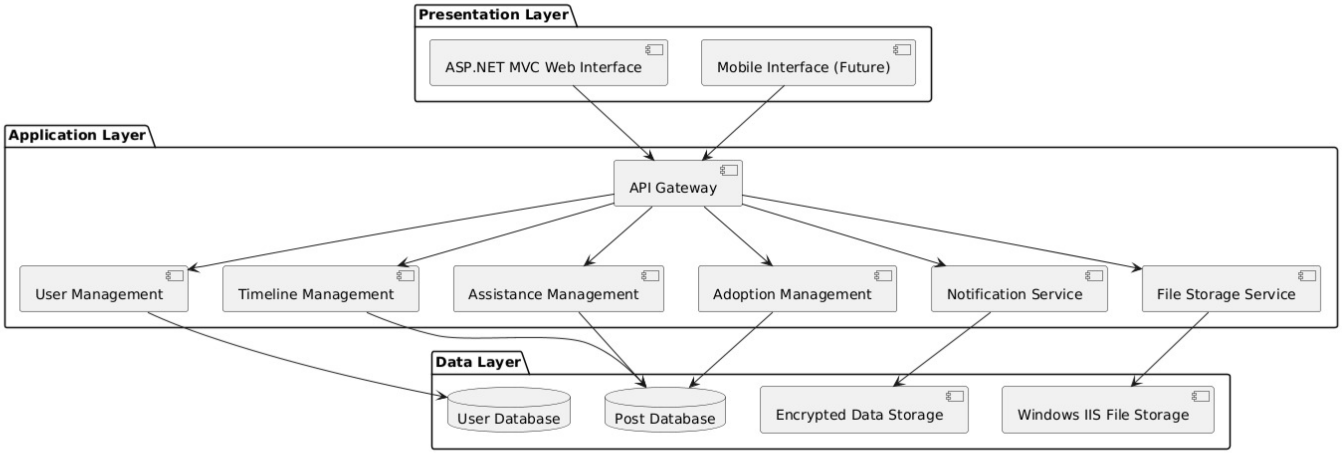


Figure 13: Architectural Design of PetSoLive

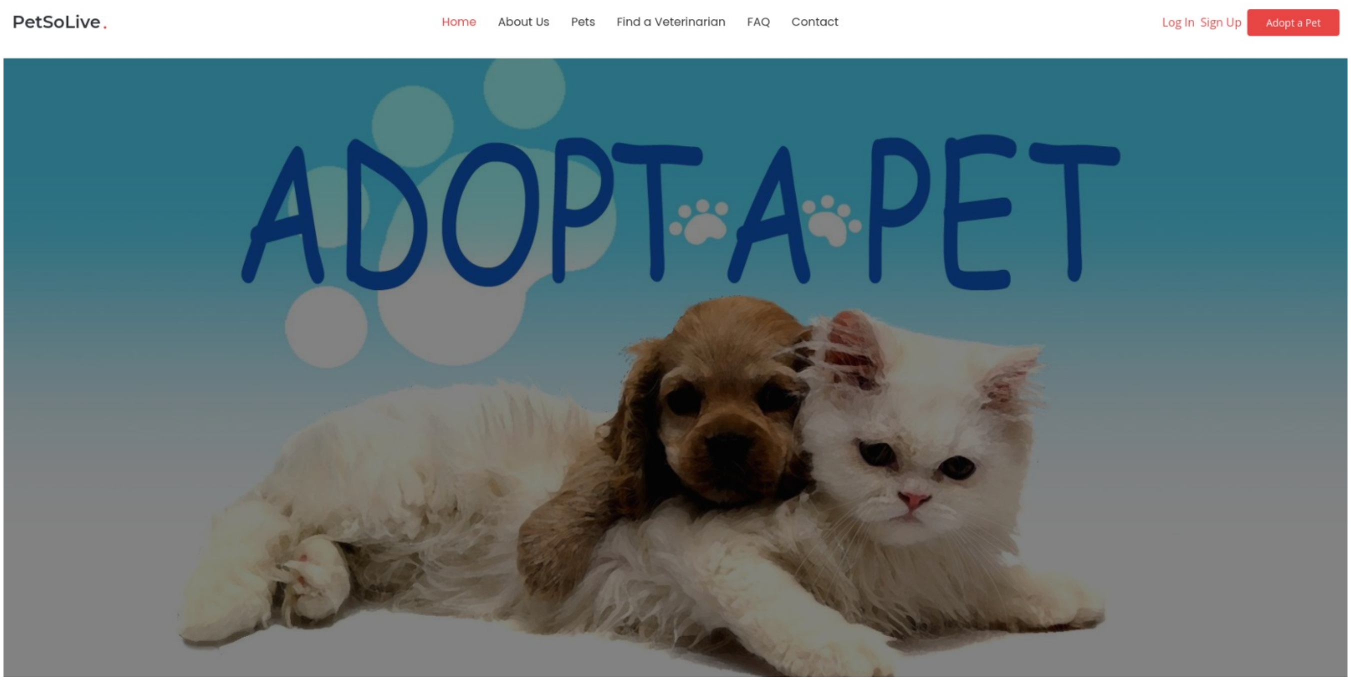
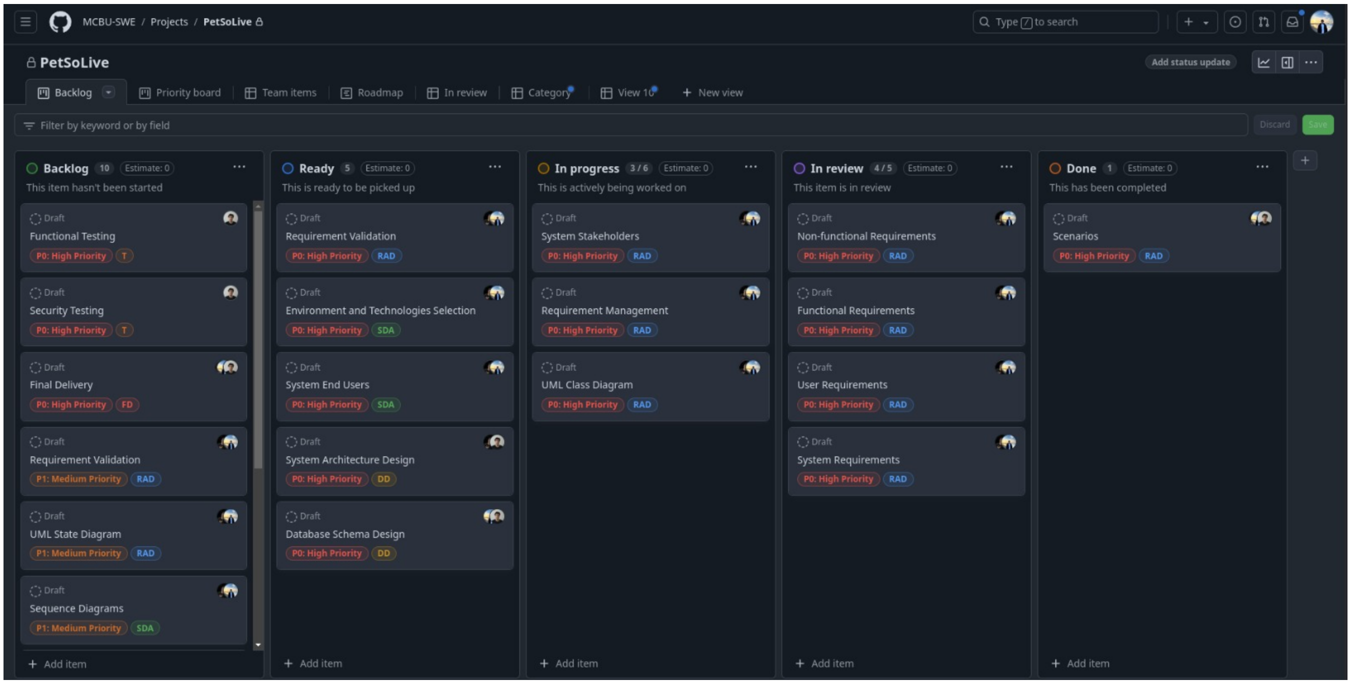


Figure 14: Front-End Interface Design Example



MCBU-SWE / Projects / PetSoLive

Type / to search

PetSoLive

Add status update

Backlog | Priority board | Team items | Roadmap | In review | Category | View 10 | + New view

Filter by keyword or by field

Discard | Save

Backlog 10 Estimate: 0

This item hasn't been started

- Draft – Functional Testing – P0: High Priority – T
- Draft – Security Testing – P0: High Priority – T
- Draft – Final Delivery – P0: High Priority – FD
- Draft – Requirement Validation – P1: Medium Priority – RAD
- Draft – UML State Diagram – P1: Medium Priority – RAD
- Draft – Sequence Diagrams – P1: Medium Priority – SDA

+ Add item

Ready 5 Estimate: 0

This is ready to be picked up

- Draft – Requirement Validation – P0: High Priority – RAD
- Draft – Environment and Technologies Selection – P0: High Priority – SDA
- Draft – System End Users – P0: High Priority – SDA
- Draft – System Architecture Design – P0: High Priority – DD
- Draft – Database Schema Design – P0: High Priority – DD

+ Add item

In progress 3 / 6 Estimate: 0

This is actively being worked on

- Draft – System Stakeholders – P0: High Priority – RAD
- Draft – Requirement Management – P0: High Priority – RAD
- Draft – UML Class Diagram – P0: High Priority – RAD

+ Add item

In review 4 / 5 Estimate: 0

This item is in review

- Draft – Non-functional Requirements – P0: High Priority – RAD
- Draft – Functional Requirements – P0: High Priority – RAD
- Draft – User Requirements – P0: High Priority – RAD
- Draft – System Requirements – P0: High Priority – RAD

+ Add item

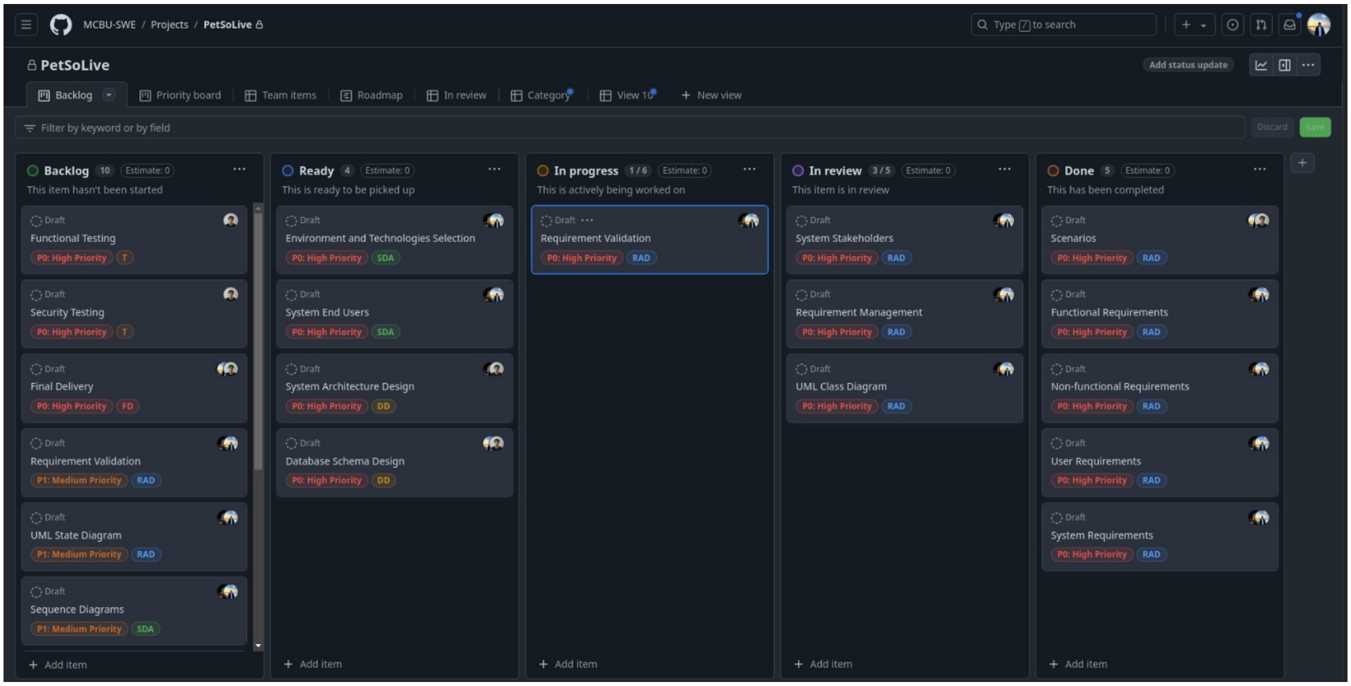
Done 1 Estimate: 0

This has been completed

- Draft – Scenarios – P0: High Priority – RAD

+ Add item

Figure 15: Kanban Board – 4



Backlog 10 Estimate: 0
This item hasn't been started

- Draft — Functional Testing — P0: High Priority — T
- Draft — Security Testing — P0: High Priority — T
- Draft — Final Delivery — P0: High Priority — FD
- Draft — Requirement Validation — P1: Medium Priority — RAD
- Draft — UML State Diagram — P1: Medium Priority — RAD
- Draft — Sequence Diagrams — P1: Medium Priority — SDA

+ Add item

Ready 4 Estimate: 0
This is ready to be picked up

- Draft — Environment and Technologies Selection — P0: High Priority — SDA
- Draft — System End Users — P0: High Priority — SDA
- Draft — System Architecture Design — P0: High Priority — DD
- Draft — Database Schema Design — P0: High Priority — DD

+ Add item

In progress 1 / 6 Estimate: 0
This is actively being worked on

- Draft — Requirement Validation — P0: High Priority — RAD

+ Add item

In review 3 / 5 Estimate: 0
This item is in review

- Draft — System Stakeholders — P0: High Priority — RAD
- Draft — Requirement Management — P0: High Priority — RAD
- Draft — UML Class Diagram — P0: High Priority — RAD

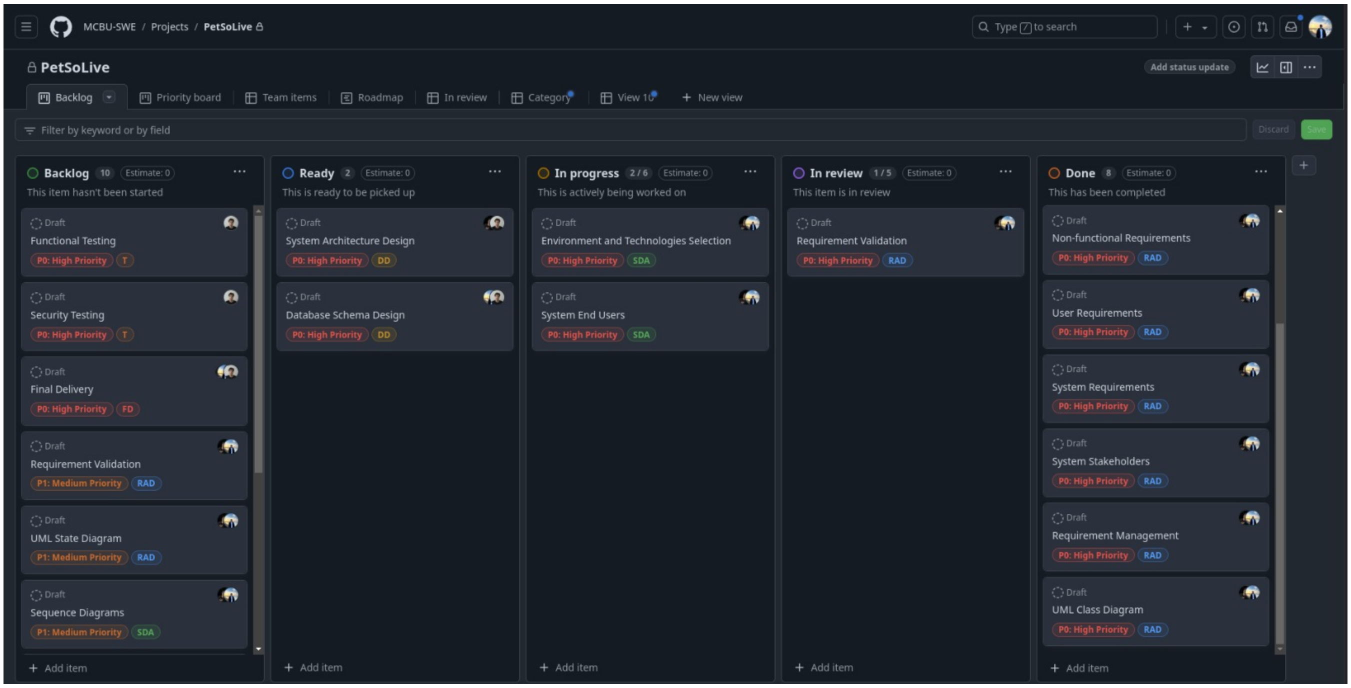
+ Add item

Done 5 Estimate: 0
This has been completed

- Draft — Scenarios — P0: High Priority — RAD
- Draft — Functional Requirements — P0: High Priority — RAD
- Draft — Non-functional Requirements — P0: High Priority — RAD
- Draft — User Requirements — P0: High Priority — RAD
- Draft — System Requirements — P0: High Priority — RAD

+ Add item

Figure 16: Kanban Board – 5



Backlog 10 Estimate: 0
This item hasn't been started

- Draft — Functional Testing — P0: High Priority, T
- Draft — Security Testing — P0: High Priority, T
- Draft — Final Delivery — P0: High Priority, FD
- Draft — Requirement Validation — P1: Medium Priority, RAD
- Draft — UML State Diagram — P1: Medium Priority, RAD
- Draft — Sequence Diagrams — P1: Medium Priority, SDA

+ Add item

Ready 2 Estimate: 0
This is ready to be picked up

- Draft — System Architecture Design — P0: High Priority, DD
- Draft — Database Schema Design — P0: High Priority, DD

+ Add item

In progress 2 / 6 Estimate: 0
This is actively being worked on

- Draft — Environment and Technologies Selection — P0: High Priority, SDA
- Draft — System End Users — P0: High Priority, SDA

+ Add item

In review 1 / 5 Estimate: 0
This item is in review

- Draft — Requirement Validation — P0: High Priority, RAD

+ Add item

Done 8 Estimate: 0
This has been completed

- Draft — Non-functional Requirements — P0: High Priority, RAD
- Draft — User Requirements — P0: High Priority, RAD
- Draft — System Requirements — P0: High Priority, RAD
- Draft — System Stakeholders — P0: High Priority, RAD
- Draft — Requirement Management — P0: High Priority, RAD
- Draft — UML Class Diagram — P0: High Priority, RAD

+ Add item

Figure 17: Kanban Board – 6