

American International University-Bangladesh (AIUB)  
**Department of Computer Science  
Faculty of Science & Technology (FST)**

**Writer your PROJECT TITLE here**

A Software Quality and Testing Project Submitted

By

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Semester: Spring\_24\_25** | | | | **Section:** | **Group No:** |
| **SL** | **SN** | **Student Name** | **Student ID** | Individual  Contribution (in %) | Total Marks: 50 |
| Earned Marks: |
| **A** |  |  |  |  |  |
| **B** |  |  |  |  |  |
| **C** |  |  |  |  |  |
| **D** |  |  |  |  |  |

The project will be Evaluated for the following Course Outcomes

|  |  |  |
| --- | --- | --- |
| **EVALUATION CRITERIA** | **Total Marks (50)** | |
|  | |
| Revision History, Test Plan Identifier, Reference Materials, Problem Background, Solutions | [10 Marks] |  |
| Requirements Specification (System feature, Quality Attributes, System Interface, Project Requirements) | [10 Marks] |  |
| Item Not to be tested, Testing approach (Testing levels, tools, meetings), Test cases | [10 Marks] |  |
| Item pass/fail criteria, Test deliverables, Staffing and Training, Responsibilities, Scheduling, Risk | [10 Marks] |  |
| Approval, Format, Submission, and Defense | [10 Marks] |  |

Software Test Plan

for

<Project>

Version 1.0 approved

Prepared by <author>

<organization>

<date created>

Table of Contents

[Revision History 3](#_Toc136846938)

[1. TEST PLAN IDENTIFIER: AT-TP01.3 4](#_Toc136846939)

[2. REFERENCE MATERIALS 4](#_Toc136846940)

[3. INTRODUCTION 4](#_Toc136846941)

[3.1 Background to the Problem 4](#_Toc136846942)

[3.2 Solution to the Problem 4](#_Toc136846943)

[4. REQUEIREMNT SPECIFICATION 4](#_Toc136846944)

[4.1 System Features 4](#_Toc136846945)

[4.2 System Quality Attributes 5](#_Toc136846946)

[4.3 System Interface 5](#_Toc136846947)

[4.4 Project Requirements 5](#_Toc136846948)

[5. FEATURES NOT TO BE TESTED 5](#_Toc136846949)

[6. TESTING APPROACH 5](#_Toc136846950)

[6.1 Testing Levels 5](#_Toc136846951)

[6.2 Test Tools 5](#_Toc136846952)

[6.3 Meetings 5](#_Toc136846953)

[7. TEST CASES/TEST ITEMS 6](#_Toc136846954)

[8. ITEM PASS/FAIL CRITERIA 6](#_Toc136846955)

[9. TEST DELIVERABLES 6](#_Toc136846956)

[10. STAFFING AND TRAINING NEEDS 6](#_Toc136846957)

[11. RESPONSIBILITIES 6](#_Toc136846958)

[12. TESTING SCHEDULE 7](#_Toc136846959)

[13. PLANNING RISKS AND CONTINGENCIES 7](#_Toc136846960)

[14. APROVALS 7](#_Toc136846961)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Updated by** | **Update Comments** |
| 0.1 | 2007.06.04 | Scot Robinson | First Draft |
| 0.2 | 2007.06.19 | Amit Nimse |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# TEST PLAN IDENTIFIER: AT-TP01.3

# REFERENCE MATERIALS

* Any reference documents with the test plan. For example: Software Requirement Specification (SRS) Document

# INTRODUCTION

## Background to the Problem Effective pet care management is becoming increasingly critical in today’s fast-paced world, not only for pet owners but also for veterinarians, groomers, pet sitters, and adoption organizations. Despite the growing number of pet lovers and advancements in veterinary care, many individuals and service providers still rely on manual or disconnected systems to manage essential tasks such as scheduling veterinary appointments, tracking vaccinations, organizing grooming routines, accessing pet care resources, and finding adoptable pets. This fragmented approach often results in missed appointments, incomplete health records, delayed care, and poor coordination between pet owners and service providers. New pet owners, in particular, face difficulties in accessing reliable information and managing their pet’s needs effectively. The main cause of this problem lies in the absence of a centralized and efficient digital platform that integrates all aspects of pet care. Without a unified system, users must depend on various applications, paper documents, or manual scheduling leading to disorganization and neglect of important pet care responsibilities. To address this challenge, the Pet Care Management App has been proposed as a comprehensive digital platform designed to assist pet owners in managing their pets’ health, safety, and well-being. Users can securely log in, register detailed pet profiles, and retrieve forgotten passwords with ease. The app offers essential features such as:

## Booking veterinary appointments

## Connecting with pet care professionals (e.g., trainers, groomers)

## Receiving vaccination reminders and health alerts

## Saving emergency contacts and automatically initiating calls in urgent situations

## With its strong focus on convenience, data security, and pet safety, the app ensures fast access to vital services and is designed to work across multiple platforms for seamless use. Its goal is to empower pet owners with timely support and effective tools to manage all aspects of their pets’ care, ultimately improving the quality of life for both pets and their caregivers.

**Root Cause of the Problem**

The primary cause of the issue is the absence of a unified, well-structured digital platform that addresses every aspect of pet care. These days, pet owners frequently utilize various apps, paper notes, or reminders to keep track of things like health records, grooming appointments, and veterinary visits. This makes it difficult to remember everything, which can result in confusion, missed appointments, and even the neglect of crucial things like immunizations. The lack of a single location to obtain trustworthy pet care information presents another challenge for new pet owners. This may lead to bad choices or a delay in providing pets with the care they require.  
Without a common framework, pet service providers like doctors and groomers find it difficult to effectively manage their workload and interact with pets.

**Why This Problem is Important**

Addressing this problem is essential because proper pet care significantly affects the **health, safety, and quality of life of animals**, and it also reduces stress and time consumption for pet owners and care providers. Inadequate management can lead to missed vaccinations, poor hygiene, or even preventable illnesses. Furthermore, service providers such as veterinarians and groomers benefit from a unified system that allows better client management and timely service delivery.

By implementing a **Pet Care Management System (PCMS)** that offers:

* Real-time scheduling reminders,
* Health and vaccination tracking,
* Pet adoption listings, and
* Access to reliable pet care resources,

the overall pet care experience can be transformed. The PCMS aims to **streamline pet management**, reduce human error, and foster a more responsible and informed pet ownership culture — ultimately improving the well-being of pets and the satisfaction of all parties involved.

## Solution to the Problem

To address the challenges faced by pet owners and care providers, we propose developing a **Pet Care Management App** — a centralized digital solution that combines health tracking, scheduling, emergency handling, and access to pet services in one secure, user-friendly platform.This solution is particularly appropriate because it directly targets the core issues: disorganized records, scattered tools, missed appointments, and lack of immediate support during emergencies. By integrating all essential pet care features into one system, users will experience greater convenience, faster access to services, and better overall care for their pets.

Moreover, the solution is **feasible and aligned with business objectives** because:

* It meets the growing demand for digital pet care tools.
* It provides multiple monetization opportunities (e.g., premium features, advertising, affiliate links with pet care providers).
* It improves customer engagement and loyalty through useful, time-saving features.
* It enhances brand trust by offering security, support, and real value to pet owners.

**Description of the Proposed Software**

The **Pet Care Management App** is a smart, multi-platform application developed to help pet owners manage every aspect of their pets’ lives. Its main features include:

* **User Registration & Secure Login**
* **Detailed Pet Profiles** (health records, breed, age, allergies, etc.)
* **Appointment Booking** with veterinarians, trainers, and groomers
* **Vaccination & Medication Reminders**
* **Emergency Contact Storage** with automatic call initiation in urgent situations
* **Educational Resources** on pet care, diet, and behavior
* **Adoption Listings** to help users find or rehome pets
* **Data Privacy & Cloud Backup** for secure access from any device

**Benefits**

* Reduces the risk of missed vet visits and important vaccinations
* Keeps health records organized and accessible
* Provides instant help in emergencies
* Supports informed pet ownership through trusted resources
* Saves time and effort by offering all features in one place

**Objectives**

* Develop a centralized and secure system for managing pet care
* Improve communication between pet owners and service providers
* Support better health and well-being for pets through timely reminders and data access

**Goals**

* Enhance the quality of pet care
* Make pet ownership easier and more organized
* Increase efficiency for veterinarians and pet care professionals

**Existing Studies and Software Solutions**

Several existing applications offer partial solutions to the problem, including:

1. **PetDesk**
   * Focuses on appointment reminders and pet health tracking
   * Mostly used by veterinary clinics to stay connected with clients
2. **11pets**
   * Offers pet care reminders and health tracking features
   * Includes a grooming and medical log for home use
3. **Pawtrack**
   * Specializes in GPS tracking for pets, particularly cats
   * Less focused on comprehensive health or service management
4. **Pet First Aid by American Red Cross**
   * Provides emergency care tips and guides
   * Does not include scheduling, reminders, or service connection

While these solutions address specific areas such as health tracking, GPS, or first aid, **none of them offer an all-in-one experience** combining emergency support, care provider integration, pet adoption, and multi-feature scheduling. The proposed Pet Care Management App fills this gap with a **holistic, user-friendly, and secure system** that meets the real needs of modern pet owners.

# REQUEIREMNT SPECIFICATION

## System Features

* List down the system functional requirements that describe the system’s functionalities.

**1. System Login (Example)**  
**Functional Requirements**

* 1. The software shall allow users to login with their given username and password.
  2. If the username and/or password has been inserted wrong for more than three times, the random verification code will be generated by the system to retry login.
  3. If the number of login attempt exceed its limit (5 times), the system shall block the user account login for one hour *[optional function]*

Priority Level: High  
Precondition: user have valid user id and password.

Cross reference: 4.1, 7.2, 9 (example)

// Start from here//

**Functional Requirements**

1. **User Registration**

**1.1** The System shall allow new users to register with name, email and password.

1.2 The system shall validate the email format before submission.

1.3 The system shall validate password length is more than 6.

1.4 A OTP shall be sent on the varified email addres.

1.5 User shall be able to enter the OTP.

1.6 A confrimation email shall be sent upon successful registration.

**Priority Level: High**

**Precondition:** User has a valid email address.

**Cross Reference:**

1. **User Signin with third parties**

2.1 The software shall allow the user to sign up and log in with third-party authentication providers such as Facebook, Google or Apple

**Priority: High**

**Precondition:** User must have a valid and active third\_party account (Facebook, Google, or Apple)  
**Cross Reference:** 1.1 (User Authentication), 4.1 (Security Requirements), 9.2 (Test Case ID PMS-2024-006)

1. **System Login**

**3.1** The software shall allow users to login with their given username and password.

3.2 If the username and/or password has been inserted wrong for more than three times, the random verification code will be generated by the system to retry login.

**Priority: High**

**Precondition:** User has a valid Facebook, Google or Apple id.  
**Cross Reference:**

1. **Upload Profile Picture**

**4.1** The software shall allow users to upload and update their profile picture from their local device in supported image formats (e.g., JPG, PNG, JPEG) with a maximum size of 10MB.

**Priority: Medium**

**Precondition:** User must have an active account.

**Cross Reference:**

1. **Search About Pet**

**5.1** The software shall users to search about pets using keywords such as name, breed, or age.

5.2 The software shall provide filtering options in the search feature to narrow down reasults by breed, age, gender.

5.3 The software shall display search reasults.

5.4 If no result found then the system should display no matches found.

**Priority: Medium**

**Precondition: At least one macthing result must exist.**

**Cross Reference:**

1. **Side Navigation Menu(Sidebar)**

6.1 The software shall display a side navigation menu (sidebar) on all authenticated pages, allowing users to access the main sections of the system easily.

6.2 The sidebar shall include links to the following modules:

* Pet Profile
* Create Pet Profile
* Adoption
* Doctor Appoinment
* Pet Sitter
* Pet Grommer
* Health Tracking
* Rating

**6.3** The sidebar shall highlight the currently active page with a different background color or icon indicator.

**6.4** The sidebar shall display right side of the navbar and the sidebar shall display the user’s profile picture and name at the top section of menu.

**Priority: Medium**

**Precondition:** User must have login and user has uploaded a profile picture.

**Cross Reference:**

1. **Create Pet Profile**

7.1 The software shall allow users to create pet profile with pet name, pet owner name, pet age, pet gender, breed, species into a structured form.

7.2 The software shall validate all input field to ensure no field is left empty and data is in a valid format before submission.

7.3 The system shall store the pet profile in the database and display a confirmation message upon successful creation.

7.4 Each pet profile shall be li ked to the users account and accessible from the user dashboard for future update or deletion.

7.5 The system shall allow users to optionally upload a photo of the pet in image formats such as JPG, JPEG and PNG.

**Priority: High**

**Precondition:** User must have logged in and have a valid account.

**Cross Reference:**

1. **Pet Profile Display**

**8.1** The software shall display pet profile with pet picture, pet name, pet id, pet age, gender, breed, species and medical history.

8.2 If the user has not upload pet picture then a default placeholder image shall be shown.

8.3 The software shall allow users to edit or delete pet profile.

**Priority: High**

**Precondition:** At least one pet profile is created.

**Cross Reference:**

1. **Edit Pet Profile**

**9.1** The software shall fetch pet data from database to edit pet profile information.

**9.2** The software shall store the new pet data to the database and update pet profile.

**Priority:** Medium.

**Precondition:** At least one pet profile is created.

**Cross Reference:**

1. **Adoption Application**

**10.1** The software shall allow users to submit adoption application which contains date, adoption id, application id, applicant id , and status.

10.2 The status shall be set to “Pending” by default and not changeable by user.

10.3. The software shall create a adopter profile with valid data.

**Priority:** Medium.

**Precondition:**

**Cross Reference:**

1. **Adoption Approval**

**11.1** If the agency approve the application then it shall set the status “Adopted”.

**Priority:** Medium.

**Precondition:** Must have a adoption application.

**Cross Reference:**

1. **Pet Sitting and Grooming Service**

12.1The software shall allow users to hire pet sitter and groomer for the pet.

12.2 The list of groomers and pet sitters shall be shown to the users with name , distance, and price per hour.

**Priority:** Medium.

**Precondition:**

**Cross Reference:**

1. **Appointment Booking**

13.1 The software shall allow users to book an appointment for veteran with date, time, appointment id, pet id, user id and type

13.2 The software shall auto generate appoinment id.

13.3 The software shall validate all input field to ensure no field is left empty and data is in a valid format before submission.

**Priority:** Medium.

**Precondition:**

**Cross Reference:**

1. **Health Record**

14.1 The software shall allow user to generate pet health record that includes details such as vaccination history, check-up dates, medication , allergies, and veterinary notes.

14.2 The software shall provide a structured input form for users to enter or update health information for each registered pet.

14.3 The health record shall be linked to the respective pet profile and viewable from the pet dashboard.

**Priority:** Medium.

**Precondition:**

**Cross Reference:**

1. **Rating**

15.1 The software shall allow users to submit a rating for the application experience.

15.2 The rating interface shall use 1 to 5 star scale.

15.3 Users shall be allowed to optionally write a short review or comment explaning their rating.

**Priority:** Medium.

**Precondition:**

**Cross Reference:**

## System Quality Attributes

**QA1 – Usability**

**Description:** A new user shall be able to register, add a pet profile, and book a grooming appointment within an average of 5 minutes and a maximum of 7 minutes.

**Priority Level:** High

**Precondition:** User must have a valid internet connection and device.

**Cross Reference:** QA3, QA5

**QA2 – Performance**

**Description:** The system shall respond to user actions such as search or booking within 2 seconds under normal load conditions (≤10,000 concurrent users).

**Priority Level:** High

**Precondition:** System is operating under optimal conditions.

**Cross Reference:** QA1, QA4

**QA3 – Security**

**Description:** All personal, payment, and pet medical data shall be encrypted in transit and at rest using industry standards (e.g., HTTPS, AES-256).

**Priority Level:** High

**Precondition:** The user is authenticated and authorized.

**Cross Reference:** QA2, QA6

**QA4 – Reliability**

**Description:** The system shall maintain an uptime of 99.9% during operational hours (24/7 access).

**Priority Level:** High

**Precondition:** Hosting infrastructure is properly maintained.

**Cross Reference:** QA2, QA5

**QA5 – Scalability**

**Description:** The system shall support scaling to 100,000 users with seamless service booking and health data tracking without degrading performance.

**Priority Level:** Medium

**Precondition:** System is deployed with scalable cloud infrastructure.

**Cross Reference:** QA2, QA4

**QA6 – Maintainability**

**Description:** System modules shall be modular and documented so that developers can implement bug fixes or updates within 4 hours.

**Priority Level:** Medium

**Precondition:** Developers have access to documentation and source code repository.

**Cross Reference:** QA3, QA7

**QA7 – Compatibility**

**Description:** The system shall run seamlessly on Chrome, Firefox, Safari, and Edge (latest versions), and mobile apps shall support Android (v10+) and iOS (v13+).

**Priority Level:** Medium

**Precondition:** User device meets minimum technical requirements.

**Cross Reference:** QA6, QA8

**QA8 – Data Integrity**

**Description:** The system shall ensure 100% accuracy of pet records by validating all user input during data entry (e.g., health info, adoption history).

**Priority Level:** High

**Precondition:** Input validation modules are active.

**Cross Reference:** QA3, QA7

**QA9 – Response Time**

**Description:** Critical operations like emergency vet contact or appointment booking must be completed in under 2 seconds.

**Priority Level:** High

**Precondition:** User has a stable internet connection.

**Cross Reference:** QA2, QA4

**QA10 – Availability**

**Description:** All major functionalities (login, profile, booking, health tracking) shall be available 24/7, excluding scheduled maintenance.

**Priority Level:** High

**Precondition:** No server outages or maintenance in progress.

**Cross Reference:** QA4, QA9

**QA12 – Correctness**

**Description:** The system shall ensure that all inputs, computations (e.g., appointment time validation, health tracking data), and outputs strictly conform to specified functional requirements and business logic (e.g., only registered pets can book appointments).   
**Priority Level:** High   
**Precondition:** All modules are implemented according to the approved SRS and verified through functional testing.   
**Cross Reference:** QA8 (Data Integrity), QA2 (Performance)

**QA13 – Completeness**

**Description:** The system shall provide full support for all core user tasks listed in the functional requirements — including user registration, pet profile management, booking, health tracking, emergency contact, and adoption listings — without missing any major workflow step.   
**Priority Level:** High   
**Precondition:** All features listed in the requirements document are implemented and testable.   
**Cross Reference:** QA1 (Usability), QA5 (Scalability)

**QA14 – Consistency**

**Description:** The system shall maintain consistent user experience across all platforms (web and mobile), ensuring uniform interface layout, terminology, color themes, error messages, and behavior across all modules (e.g., profile editing, notifications).   
**Priority Level:** Medium   
**Precondition:** UI/UX design standards are applied uniformly across all components.   
**Cross Reference:** QA1 (Usability), QA7 (Compatibility)

**QA11 – Accessibility**

**Description:** The system should conform to **WCAG 2.1 Level AA** accessibility standards, ensuring that users with visual, auditory, or motor impairments can navigate and use all essential features (e.g., booking, viewing health records, contacting emergency services) without external assistance.   
**Priority Level:** Medium   
**Precondition:** The device supports assistive technologies (e.g., screen readers, voice navigation).   
**Cross Reference:** QA1 (Usability), QA7 (Compatibility)

## System Interface

**UI / UX Design**

**1. Starting**

A screenshot of a cell phone

AI-generated content may be incorrect.

**2.SignUp**

A screenshot of a phone

AI-generated content may be incorrect.

**3.Information**

A screenshot of a phone

AI-generated content may be incorrect.

**4.Dashbord**

A screenshot of a phone

AI-generated content may be incorrect.

**5.DashBord**

A screenshot of a phone

AI-generated content may be incorrect.

**6.Demo Profile**

A screenshot of a phone

AI-generated content may be incorrect.

**7.Create pet Profile**

A screenshot of a mobile application

AI-generated content may be incorrect.

**8.Adoption Info**

A screen shot of a application form

AI-generated content may be incorrect.

**9.Adopter**

A screenshot of a phone

AI-generated content may be incorrect.

**10.Adoption**

A screenshot of a cell phone

AI-generated content may be incorrect.

**11. Pet sitting and grooming**

A screenshot of a phone

AI-generated content may be incorrect.

**12.Appointment**

A screenshot of a appointment form

AI-generated content may be incorrect.

**13.Health Record**

A screenshot of a phone

AI-generated content may be incorrect.

**14.Review**

A screenshot of a phone

AI-generated content may be incorrect.

**15.Exit Page**

A black and white dog with a blue background

AI-generated content may be incorrect.

**16.Prototype**

****

## Project Requirements

The success of the **Pet Care Management System (PCMS)** depends on several critical project constraints and requirements that must be followed during the planning, development, and delivery phases. These requirements are defined in terms of time, budget, resources, environment, and effort estimation, which help guide the project management process and ensure timely and quality completion of the system.

**1. Time Constraint**

Time is a crucial constraint in this project. The total duration allocated for the completion of the PCMS is 8 weeks. This limited timeline has been divided into manageable development phases to ensure steady progress and deadline compliance. The initial two weeks are allocated for requirement analysis, followed by two weeks for system design. Development work is scheduled for the next two weeks, while the final phase, which includes system testing, debugging, and deployment, is planned for the last two weeks. Adhering to this time schedule is essential to meet academic and practical expectations.

**2. Budget Constraint**

Another key constraint of the project is the fixed budget of 50,000 TK. This budget is distributed across the different phases of development, including requirement analysis (5,000TK), project management (10,000 TK), system design (8,000 TK), development (15,000 TK), and testing (12,000 TK). Each team member is responsible for working within these financial limits to ensure that the project is completed economically. The low-cost nature of the project emphasizes the importance of resourcefulness, careful planning, and efficient use of open-source technologies.

**3. Human Resource Constraint**

The project is being developed by a team of four members, each contributing equally (25%) to the overall work. The roles have been distributed based on the strengths and interests of the team members. Responsibilities include project management, frontend and backend development, UI/UX design, testing, and documentation. Collaborative efforts and well-defined roles ensure that all tasks are covered efficiently, and team performance remains balanced throughout the project lifecycle.

**4. Environmental Constraint**

The system is developed using a modern MERN stack which includes HTML, CSS, JavaScript, and React.js for frontend development, while Node.js and Express.js are used for the backend. The system uses MongoDB as its primary database. The software is designed to be accessible on both desktop and mobile devices, supporting Android and Web Application. It is assumed that the system will be hosted in a cloud environment to ensure availability, scalability, and reliability.

**5. Effort Estimation (Using COCOMO Model)**

To estimate the amount of effort and time needed to complete this project, the COCOMO (Constructive Cost Model) is applied. COCOMO is a widely used model in software engineering for predicting project effort based on the size of the codebase.

For this project, we estimate the size of the code to be approximately 4 KLOC (thousands of lines of code). Since it is a relatively small project developed in a stable academic setting by an experienced team, it fits the Organic type of COCOMO. Based on the COCOMO formula:

* Effort (E) = 2.4 × (KLOC)^1.05
* Time (T) = 2.5 × (Effort)^0.38

Using the formula:

* Effort ≈ 10.17 person-months
* Time ≈ 5.85 months

However, as the actual development team has four members working in parallel and the features have been carefully scoped, the work is planned to be completed within the assigned 8 weeks. This is possible through task parallelization, time management, and simplified functionalities.

# FEATURES NOT TO BE TESTED

**1.Third-Party Payment Gateway Integration**

The system supports online payments (e.g., Bkash, Rocket and Banks Apps) the internal logic for initiating payment requests will be tested. But the actual functionality of the third-party payment gateway (e.g., transaction success/failure, bank verification processes) will not be tested directly, as it is managed by the third-party service providers.

**2.External Emergency Contact Services**

The Pet Management System provides a feature to contact emergency pet services (e.g., veterinarians). However, the system will not be tested:

* The availability or response time of emergency contacts.
* The real-world outcome of contacting a service.

**3.Cross-Platform OS Compatibility**

As the application is designed to be compatible with Android only. Cross-Platform Compatibility testing is not possible at this moment.

**4. Load/Stress Testing beyond 5,000 Users**

Basic performance testing for up to 5,000 concurrent users will be conducted

However, large-scale stress testing (e.g., for 10,000+ users) is excluded due to limited resources.

**5.Multi-Languages Support**

The current version of the Pet Management System (PMS) supports English language only. Although Bangla language is planned for future releases, Bangla interface will not be tested in this version.

# TESTING APPROACH

## Testing Levels (Detailed Plan)

In the Pet Management System, the following structured testing levels will be implemented to ensure comprehensive quality assurance across all functional and non-functional aspects:

**Unit Testing**

* Purpose: To test individual functions, methods, and components in isolation.
* Conducted By: Development team using the Jest framework.
* Scope:
* Frontend: React component rendering, state management, input validation.
* Backend: API endpoints, controller logic, data models, middleware functions.
* Example Scenarios:
* Validate input sanitization on registration form.
* Ensure proper error handling for appointment booking API.
* Automation: Majority of unit tests will be automated and integrated into the CI pipeline via GitHub Actions.
* Metrics: Code coverage percentage (goal ≥ 80%).

**Integration Testing**

* Purpose: To test the interaction between integrated modules or services.
* Conducted By: QA and development teams collaboratively.
* Scope:
* Communication between front-end forms and back-end APIs (e.g., login, booking).
* Inter-module interaction (e.g., health tracking module updating the pet profile).
* External integrations (mocked) such as email/SMS notification services.
* Example Scenarios:
* When a user books a vet appointment, ensure confirmation notification is sent.
* Updating health data correctly reflects dashboard summaries.
* Tools: Postman (API), Jest (integration), MongoDB Test Instances.

**System Testing**

* Purpose: To verify the system as a whole against the defined requirements.
* Conducted By: QA team.
* Scope:
* End-to-end flow for all user roles (Pet Owner, Vet, Admin).
* Functional coverage: registration → profile creation → service booking → history access.
* Non-functional testing: performance, security, usability, accessibility.
* Tools: Selenium for UI testing, Lighthouse for performance and accessibility.
* Execution: Manual and automated scripts will be used to validate workflows across multiple devices and browsers.

**Acceptance Testing**

* Purpose: To validate the system with respect to business needs and user expectations.
* Conducted By: Product stakeholders and a selected group of real users (UAT group).
* Scope:
* Verify that the business logic aligns with the actual pet owner's needs.
* Assess real-world usability and ease of adoption.
* Validate key functionalities: profile management, booking, health logs.
* Methodology:
* Scenario-based testing with feedback sessions.
* Surveys and usability score collection after trials.
* Exit Criteria: ≥ 90% of test scenarios passed with acceptable feedback.

## Test Tools

•**Figma for Requirements & UI Design:** We used Figma to design the wireframes and high-fidelity interactive prototypes for the Pet Management System. This helped the team visualize workflows, navigation structures, and identify usability issues early in the design phase before implementation began.

•**Lucidchart / draw.io for System Diagrams:** These tools were utilized to create architecture diagrams, flowcharts, and use case mappings of the system. This helped define the system’s components, user flows, and interactions between modules and external APIs.

•**Visual Studio Code for Development:** As the primary code editor, VS Code was used for developing both frontend and backend modules. Its rich plugin ecosystem helped streamline JavaScript, Node.js, and React.js development.

•**React.js, Node.js, MongoDB Atlas as Development Stack:** React.js was used for the frontend, Node.js and Express.js for the backend APIs, and MongoDB Atlas served as the cloud-based database. This MERN stack allowed efficient development and data management for our cross-platform system.

•**Git & GitHub for Source Control:** Version control was managed using Git and GitHub. This enabled code collaboration, branching, pull requests, and CI integration across our development and testing teams.

•**Jest for Unit & Integration Testing:** Jest was used for testing JavaScript modules and React components. Unit and integration tests were written to verify input/output logic, component rendering, and form validation.

**• Postman for API Testing:** REST APIs were tested using Postman. We created and automated API collections to validate endpoint responses, authentication, error handling, and status codes.

**• Selenium for Functional UI Testing:** We used Selenium WebDriver to simulate user interactions across different browsers and screen sizes. This helped validate form submissions, button clicks, navigation flow, and booking actions.

•**Lighthouse (Chrome DevTools) for Performance & Accessibility Testing:** Lighthouse was used to evaluate system performance, mobile responsiveness, SEO readiness, and accessibility (e.g., color contrast, ARIA labels).

•**MongoDB Compass for Data Validation:** Compass was used to browse collections and validate test data directly inside the database, ensuring data integrity and correct schema population after operations.

•**Allure / HTML Reporters for Test Result Visualization:** These tools were used to generate automated reports from test runs. They visualized pass/fail ratios, logs, and screenshots of failed test cases for better debugging.

**• Google Sheets & Excel for Manual Test Management:** Test scenarios and execution logs were maintained in structured spreadsheets. Each test case included steps, expected output, actual output, status, and notes.

**• Trello for Project & Bug Management:** The team used Trello boards to plan sprints, assign testing tasks, and track bug tickets with severity levels. This ensured clear visibility and accountability across the QA lifecycle.

**• Notion & Google Docs for Collaboration:** All test plans, schedules, and review notes were shared via Google Docs or Notion. This allowed asynchronous collaboration and version tracking.

**• GitHub Actions for CI/CD Pipelines:** We configured GitHub Actions for automatic building, testing, and deploying the application. This minimized manual intervention and accelerated release cycles.

**• Vercel / Netlify for Frontend Hosting:** The React frontend was deployed and tested live on these platforms, allowing mobile and cross-browser testing in real environments.

**• Render / Railway for Backend Deployment:** Node.js backend and MongoDB database were deployed on Render and Railway. This supported full-stack staging environments for realistic testing.

## Meetings

The testing team will decide the frequency of meetings according to the meeting purpose. The meeting setup will include the following:

1. **Daily Stand-up Meetings**

**Purpose:** Quick check-in on individual progress, identify roadblocks, and coordinate efforts.

**Frequency:** Daily (15-20 mins)

**Participants:** All testers

**Agenda:**

What was tested yesterday?

What will be tested today?

Any blockers or issues?

1. **Weekly Review Meetings:**

**Purpose:** The main objective is to have a comprehensive discussion about the status of testing, the test results, the critical issues, and to make the plan for the next week.

**Frequency:** Once weekly (1 hour).

**Participants:** Testing team, developers, project manager.

**Agenda:**

* Progress report which includes executed vs. pending test cases.
* Summary of bugs that were found.
* Review of requirement traceability.
* Reassigning escalates unresolved issues.
* Update the test schedule if needed.
* Discussion on critical issues and potential solutions.
* In-depth analysis of test results (including defect trends).

1. **Sprint Planning:**

**Purpose:** The goal of sprint planning is to define the tasks, objectives, and scope of the upcoming sprint cycle according to the project goals and team capacity.

**Frequency:** At the beginning/ end of each sprint (as per sprint cycle)

**Participants:** Project Manager, Testers, Developers, Product Owner, Scrum Master.

**Agenda:**

* Go through the product backlog items.
* Establish sprint goals.
* Determine effort for each task.
* Assign tasks to the team members.
* Recognize the testing scope.

1. **Monthly Planning:**

**Purpose:** Review overall testing strategy, identify long-term trends, plan for future sprints or releases, and discuss process improvements.

**Frequency:** Once a month.

**Participants:** Testing team, Project Manager, potentially other stakeholders.

**Agenda:**

* Review of overall testing strategy and alignment with project goals.
* Analysis of key performance indicators(KPIs) related to testing.
* Discussion of long-term trends.
* Planning of the next month/sprint.

# TEST CASES/TEST ITEMS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Project Name: | | | Test Designed by: | | |
| Test Case ID: FR\_1 | | | Test Designed date: | | |
| Test Priority (Low, Medium, High): Medium | | | Test Executed by: | | |
| Module Name: Login Session | | | Test Execution date: | | |
| Test Title: verify login with valid username and password | | |  | | |
| Description: Test website login page | | |  | | |
| Precondition (If any): User must have valid username and password | | | | | |
| Test Steps | Test Data | Expected Results | | Actual Results | Status (Pass/Fail) |
| 1. Go to the website 2. Enter username 3. Enter password 4. Click submit | Username: 99999999999  Password: 321 | User should login into the application | | As expected, | Pass |
| Post Condition: User is validated with database and successfully login to account. The account session details are logged in the database. | | | | | |

Table 1: Test Case for **Login Session**

# ITEM PASS/FAIL CRITERIA

* Discuss how you would define your testing pass/fail criteria.

# TEST DELIVERABLES

**9.1 Test Plan**

The Test Plan document outlining the overall testing approach, objectives, scope, resources, schedule, roles and risk management.

**9.2 Test Cases**

The detailed test cases including test ids, steps, expected results, actual outcome, and pass/fail status. The test cases are focused on functional and non-functional requirements.

9.3 **Tools**

Document that describe the hardware, software and tools used for testing.

**9.4 Bug/ Defect Report**

The Bug/ Defect report documenting bugs discovered during testing.

**9.5 Test Summary Report**

A final report that summarize overall test results, bug statistics, and outstanding issues.

# STAFFING AND TRAINING NEEDS

For the successful execution of testing activities in the Pet Management System (PMS) project, the recruitment and training of the testing team must be strategically planned. The quality of testing largely depends on the right skill set, clear role distribution, and proper knowledge transfer among team members. Both horizontal and vertical staffing approaches have been considered for this academic project.

**1. Staffing Strategy**

To manage testing tasks efficiently, the project follows a combination of horizontal and vertical staffing models:

* • Horizontal Staffing:   
    In this approach, all testing responsibilities are distributed equally across the team members. Each team member participates in writing test cases, executing them, and reporting bugs. This method is suitable for a small academic project where the workload can be balanced, and collaboration is emphasized.
* • Vertical Staffing:   
    Here, specific roles are assigned based on expertise or preference. One or two team members may act as test coordinators or leads, responsible for test planning, prioritizing test cases, and maintaining documentation. Other members perform test execution and report bugs. This helps streamline communication and maintain test quality.

For this project, a hybrid model is used to ensure flexibility while maintaining accountability.

**2. Training Needs**

Even in an academic environment, a minimal training plan is necessary to ensure all testers are familiar with:

* • The system requirements and user flows   
  • Testing tools or techniques (e.g., black-box, white-box)   
  • Bug tracking and test reporting standards

The training plan includes:

* • Introduction sessions on the Pet Management System architecture, use cases, and modules   
  • Basic testing workshops covering:   
     - Writing test cases and scenarios   
     - Executing manual tests   
     - Using bug reporting templates   
     - Maintaining traceability between requirements and test cases   
  • Review sessions led by experienced members or guided by a mentor/supervisor for continuous learning

**3. Role Assignment Summary**

|  |  |  |
| --- | --- | --- |
| **Team Member** | **Testing Focus Area** | **Role Type** |
| **Member 1** | Test Planning, Risk Analysis | Vertical (Lead) |
| **Member 2** | Appointment & Health Module Testing | Horizontal |
| **Member 3** | UI/UX and Compatibility Testing | Horizontal |
| **Member 4** | Functional & Integration Testing | Horizontal |
| **Member 5** | Test Documentation & Reporting | Vertical (Support) |

The hybrid staffing model allows the team to balance responsibilities while giving individuals leadership opportunities. Through guided training sessions and hands-on testing activities, the team is prepared to deliver high-quality results, ensuring that the PMS meets all functional and non-functional requirements.

# RESPONSIBILITIES

|  |  |  |
| --- | --- | --- |
| Name | Role | Responsibilities |
| Test Lead | Coordinates test execution | * Create test plans and assign tasks to testers. * Review test case and prioritize defects * Act as a bridge between testers and managers. |
| Test Manager | Oversees testing strategy and governance. | * Plan test timelines, budgets, and resources. * Ensure compliance with QA standards. |
| Testers (QA Engineers) | Execute tests and report defects. | * Write and execute test cases * Log defects that found with steps to reproduce. * Retest fixes and validate patches. |
| Developers | Build the product | * Write unit/ integration tests. * Fix defects raised by QA. |
| Business Analyst(BA) | Define requirements. | * Document functional requirements and acceptance criteria. * Verify test cases align with business needs. |
| Product Owner | Represents end-users. | * Prioritize features for testing. * Approve test scope and release readiness. |
| Design Team Member | UI/UX Designer | * Support usability testing * Review UI-related bugs. |
| Assigned Team Member | Documentation Lead | * Maintain requirement traceability matrix. * Compile test reports and meeting records. |

# TESTING SCHEDULE

* Provide a Gantt chart showing the schedule of various testing activities in the project.

# PLANNING RISKS AND CONTINGENCIES

Table 2: Risk Mitigation Plan for testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S/N** | **Risk Description** | **Probability** | **Impact** | **Mitigation Plan** |
| 1 | Unrealistic time estimate | 40% | Delay project 2 weeks | Use multiple estimation techniques(e.g., COCOMO, expert review) |
| 2 | Lack of test data | 30% | Incomplete or skipped test cases | Create mock data sets early in the test planning phase |
| 3 | Unclear or changing requirements | 50% | Rework of test cases and invalid results | Maintain versioned requirements and hold review meetings regulary |
| 4 | Team member unavailability | 10% | Slower test coverage | Cross-train team members to handle multiple roles or have backup team member. |
| 5 | Test case not aligned with updated requirements | 35% | Invalid or irrelevant test results | Implement regular traceability between requirements and test cases |
| 6 | Environment Issues (e.g., bugs in development server or system failure) | 25% | Delay in test execution | Maintain backup environments and ensure local testing environments are setup |
| 7 | Bugs blocking test progress | 40% | Incomplete feature testing | Assign developer-tester pairs for fast communication and immediate bug fixes. |

# APROVALS

* Provide a list of personnel along with their designations to get their approval for this test plan.

**Text Format**

* Style: Times New Roman
* Size: 12
* Line and Paragraph Spacing: 1.00
* Alignment: Justify