Test Plan for PetClinic Application

# 1. Test Plan Identifier

* **Test Plan Version**: 1.0
* **Project Name**: PetClinic Application Testing
* **Prepared by**: Jodok Felder
* **Date**: 2025-01-09

# 2. Introduction

This test plan outlines the strategy, scope, and approach for testing the **PetClinic Application**, which consists of a **backend REST API** and a **frontend Angular application**

# 3. Objectives

The objectives of this test plan are:

1. To ensure the application meets the functional and non-functional requirements.
2. To validate that the core features (owners, pets, vets) work as expected on the backend and frontend.
3. To verify the application is user-friendly and generates accurate reports.

# 4. Scope

The testing scope includes:

* **Backend (API)**: CRUD operations for owners, pets, and vets.
* **Frontend (UI)**: Key user interactions (e.g., adding an owner, viewing pet details, managing vets).
* **Integration**: Ensure backend and frontend work seamlessly together.
* **Reports**: Verify reports generated are accurate and user-friendly.

# 5. Test Items

The following features will be tested:

1. **Owners**
   * Adding, editing, viewing, and deleting owners.
2. **Pets**
   * Adding, updating, and viewing pet details.
3. **Veterinarians**
   * Adding, editing, and deleting vet records.

# 6. Test Approach

* **Testing Levels**:
  + **Unit Tests**: Individual backend methods and frontend components.
  + **End-to-End Tests**: Verify interactions across the system.
  + **UI Tests**: Simulate user interactions in the Angular frontend.
* **Testing Types**:
  + **Functional Testing**: Ensure all features work as per requirements.
  + **Negative Testing**: Verify the system handles invalid inputs gracefully.7. Test Environment
* **Backend**: Spring Boot application running on http://localhost:9966/petclinic/api.
* **Frontend**: Angular application running on http://localhost:4200.
* **Tools**:
  + **Testing Frameworks**: JUnit, Selenium. (Chosen based on experience)
  + **Browser**: Google Chrome.

# 8. Test Cases

**Backend Test Cases**

1. **CRUD Operations**:
   * Add, read, update, and delete owners, pets, and vets.
   * Negative tests for invalid data (e.g., missing fields, incorrect formats).
2. **Integration Tests**:
   * Ensure correct communication between API endpoints.

**Frontend Test Cases**

1. **Owners**:
   * Add a new owner.
   * View owner details.
   * Update owner information.
   * Delete an owner.
2. **Pets**:
   * Add a pet to an owner.
   * View pet details.
   * Update pet details.
3. **Vets**:
   * Add a veterinarian.
   * Edit vet details.
   * Delete a vet.

# 9. Test Deliverables

1. Test Scripts:
   * Backend: JUnit test files.
   * Frontend: Selenium test files.
2. Test Reports:
   * Allure or similar framework for detailed reporting. (did not work in the end, so missing for now)

# 12. Risks and Contingencies

* **Risk**: Test data from one test may interfere with subsequent tests.
  + **Mitigation**: Isolate tests and clean up data after each run.
* **Risk**: Inconsistent results due to browser or network issues.
  + **Mitigation**: Use explicit waits in Selenium tests.