### Overall Evaluation Report for Pet Clinic Application

#### 1. Application Overview

The Pet Clinic Application is designed to manage pet owners, their pets, and veterinary visits. It provides functionalities for adding, updating, and viewing pet owner information, pet details, and veterinary visits. The main features include:

* CRUD operations for pet owners.
* CRUD operations for pets.
* Managing veterinary visits for pets.
* Viewing the list of veterinarians.

#### 2. Screens Evaluated

The following screens have been evaluated:

1. Home Page
2. Owners List Screen
3. Owner Details Screen
4. Add/Edit Owner Screen
5. Pet Details Screen
6. Add/Edit Pet Screen
7. Visits Screen
8. Veterinarians List Screen

#### 3. Functional Evaluation

* Home Page:
  + Contains no functionality but serves as a landing page for navigation.
* Owners Management:
  + Adding new owners: Functionality works as expected, allowing the creation of new owners with valid data.
  + Editing existing owners: Owners can be edited, and changes are reflected in the system.
  + No delete functionality: The application does not provide an option to delete an owner, which might be necessary for data management.
* Pets Management:
  + Adding new pets: Pets can be added to an existing owner with valid data.
  + Editing pets: Existing pet details can be updated.
  + No delete functionality: Similar to owners, there is no option to delete a pet.
* Visits Management:
  + Adding visits: New visits can be added for existing pets.
  + Viewing visits: Visit details are displayed correctly under the respective pet.
  + No update or delete functionality: The application does not allow updating or deleting a visit.
* Veterinarians Management:
  + Viewing veterinarians: The list of veterinarians is displayed, showing names and specialties.
  + Incomplete data handling: Some veterinarians do not have a specialty listed.
  + No create, edit, or delete functionality: The application does not allow managing veterinarian data.

#### 4. UI/UX Evaluation

* The UI is clean and straightforward, making it easy to navigate through different sections.
* Forms are user-friendly, with clear labels and expected input fields.
* Error handling for invalid inputs is present but can be enhanced for better user experience.
* The home page serves as a simple navigation point without any interactive functionality.

#### 5. Automation Feasibility

* The application is suitable for automation testing due to its well-defined UI elements and straightforward workflows.
* Recommended tools: Selenium WebDriver for UI automation, along with JUnit/TestNG for test management.
* Test cases that can be automated include:
  + Creating, updating, and viewing owners and pets.
  + Adding and viewing veterinary visits.
  + Verifying the list of veterinarians.

#### 6. Suggested Improvements

* Implement delete functionality for owners, pets, and visits.
* Implement create, edit, and delete functionalities for veterinarians.
* Add DOM element IDs to ensure that tests could be more idempotent.
* Enhance error messages for better clarity.
* Add date validations to prevent the addition of past or unrealistic dates.
* Ensure all veterinarians have a specialty listed to maintain data completeness.

**7. Selected Test Cases for Automation and Justifications**

**Criteria for Selecting Test Cases for Automation**

1. Repetitive Tests for Multiple Builds: Tests that need to be run frequently during each build, ensuring that the application’s core functionalities are not broken.
2. Tests Prone to Human Error: Automated tests minimize the risk of human error during repetitive and detailed manual testing.
3. Tests Requiring Multiple Data Sets: Automating these tests helps in efficiently managing and validating various data inputs.
4. High-Risk Functionalities: Automating critical functionalities ensures stability and reduces risk in the application.
5. Tests Impossible Manually: Some tests may not be feasible to execute manually due to their complexity or the volume of data.
6. Time-Consuming Tests: Automation saves significant time for tests that are lengthy and labour-intensive when done manually.

Based on the above criteria, the following tests cases were automated:

#### API Test Cases

1. TC\_003: Create a new owner
   * Why: This is a repetitive and critical functionality that needs to be tested across multiple builds.
2. TC\_005: Get owner details by ID
   * Why: This functionality is frequently used and introduces high-risk conditions if it fails. Ensuring that owner details can be correctly retrieved is essential for various other operations within the application.
3. TC\_012: Add a new pet
   * Why: This functionality is frequently used and important for ensuring the integrity of the owner's pet records.

#### Web Test Cases

1. TC\_001: View owner details
   * Why: This is a fundamental functionality that needs to work consistently and is frequently used.
2. TC\_007: Edit owner details
   * Why: Editing owner details is a common functionality that should be reliable to avoid human errors. This test ensures that user data can be updated accurately, which is important for maintaining correct and up-to-date information.
3. TC\_016: Add a new veterinary visit
   * Why: This functionality involves multiple steps and data inputs, making it prone to human error and time-consuming when done manually.

### Detailed Automated Test Case Plan

#### API Test Cases

TC\_003: Create a new owner

* Test Case ID: TC\_003
* Test Case Description: Validate the API to create a new owner.
* Preconditions: API endpoint is available.
* Test Steps:
  1. Send a POST request to the /owners endpoint with the owner details in the JSON body.
  2. Validate the response status code is 201 (Created).
  3. Verify the owner details in the response.
* Expected Result: The owner is created successfully, and the response contains the correct owner details.

TC\_005: Get owner details by ID

* Test Case ID: TC\_005
* Test Case Description: Validate the API to retrieve owner details by owner ID.
* Preconditions: An owner exists in the system.
* Test Steps:
  1. Send a GET request to the /owners/{ownerId} endpoint.
  2. Validate the response status code is 200 (OK).
  3. Verify the owner details in the response.
* Expected Result: The response contains the correct owner details.

TC\_012: Add a new pet

* Test Case ID: TC\_012
* Test Case Description: Validate the API to add a new pet to an existing owner.
* Preconditions: An owner exists in the system.
* Test Steps:
  1. Send a POST request to the /owners/{ownerId}/pets endpoint with the pet details in the JSON body.
  2. Validate the response status code is 201 (Created).
  3. Verify the pet details in the response.
* Expected Result: The pet is added successfully to the owner, and the response contains the correct pet details.

#### Web Test Cases

TC\_001: View owner details

* Test Case ID: TC\_001
* Test Case Description: Validate the functionality to view owner details on the web interface.
* Preconditions: Owner records exist in the system.
* Test Steps:
  1. Navigate to the owners list page.
  2. Click on an owner’s name to view details.
  3. Verify the displayed owner information matches the expected details.
* Expected Result: The owner details are displayed correctly.

TC\_007: Edit owner details

* Test Case ID: TC\_007
* Test Case Description: Validate the functionality to edit owner details on the web interface.
* Preconditions: Owner records exist in the system.
* Test Steps:
  1. Navigate to the owners list page.
  2. Click on an owner’s name to view details.
  3. Click the “Edit” button.
  4. Modify the owner details and submit the form.
  5. Verify the updated owner information is displayed.
* Expected Result: The owner details are updated and displayed correctly.

TC\_016: Add a new veterinary visit

* Test Case ID: TC\_016
* Test Case Description: Validate the functionality to add a new veterinary visit for a pet.
* Preconditions: A pet exists in the system.
* Test Steps:
  1. Navigate to the owner’s detail page.
  2. Click on the “Add Visit” button for the pet.
  3. Fill in the visit details and submit the form.
  4. Verify the visit appears in the list of visits for the pet.
* Expected Result: The new veterinary visit is added and displayed correctly.

**Issues Found**  
Title: Intermittent 500 Response in API Test for Pets Endpoint

Description: When running the API test for the Pets endpoint in the PetClinic application, we are experiencing intermittent failures where the expected status code is 204, but we receive a 500 status code instead. This inconsistency occurs sporadically and cannot be reliably reproduced.

Steps to Reproduce:

1. Run the API test suite for the Pets endpoint in the PetClinic application multiple times.
2. Observe the test results.

Expected Behavior: The API test should consistently return a 204 status code, indicating a successful operation.

Actual Behavior: Intermittently, the API test returns a 500 status code, indicating a server error.

Additional Information:

* This issue is occurring inconsistently and cannot be reliably reproduced.
* The test data used includes a generated pet with the following details:
  + Name: Sandshrew
  + Birth Date: 1976-10-20T22:22:00.671Z
  + Type ID: 1

Attachments:

* N/A

Impact: This intermittent issue affects the reliability of our API tests and may lead to false positives or negatives in our test results. It could also indicate potential stability issues in the PetClinic application.

Priority: Medium

Status: To Do

### Conclusion

The Pet Clinic Application is well-designed with core functionalities implemented. It provides a good basis for further development and refinement. The application is a suitable candidate for test automation, and implementing the suggested improvements will enhance its usability and maintainability.