Last edited by Maya Amjad Al Hourani 10 hours ago

Testing Documentation

Testing Documentation

Testing Documentation for PetPal

Version	Date	Author(s)	Summary of Changes
0.1	Nov 28	Maya Al Hourani	Document Outline created
0.2	Dec 1	Maya Al Hourani	Test Cases added to test plan. Roughly completed Section 1 and 3

1.0 Introduction

1.1 Overview

This document outlines the testing approach for **PetPal**, ensuring that all system requirements are met through unit, integration, validation, and system testing. The application allows users to care for a virtual pet by feeding, playing, and resting it, maintaining health, happiness, and hunger metrics.

1.2 Objectives

The objectives of this testing document and the project as a whole include:

- Providing a structured, maintainable design that supports the core functionalities outlined in the requirements document.
- Creating an intuitive and accessible user interface that enhances engagement with the pet simulation.
- Ensuring that the codebase is well-documented, tested, and adaptable for future improvements.
- Applying software engineering principles and design patterns that reinforce code quality and efficiency.
- Facilitating team collaboration, code consistency, and effective project management.

References

Design Documentation: Design document.
Requirements Documentation: Requirements document.
JUnit User Guide: JUnit User Guide.

2.0 Test Plan

This section discusses the detailed test plan for your software system. This plan should indicate both the test cases that you are using to assess the quality of your software, as well as a justification for their inclusion as part of your test plan. This discussion should indicate why your plan is both sufficient and complete. In delivering its detail, this plan should include a number of sub-pages, each covering one of the following:

2.1 Unit Testing

Unit tests focus on individual methods and logic within the application. Key public methods in the VirtualPet class, such as feed(), play(), and rest(), are tested using JUnit to ensure they perform as expected. All major units in the code, including the pet's attributes and behavior, are covered.

Integration testing ensures that components interact correctly. We use Bottom-Up integration, incrementally combining smaller modules into larger components. Stubs are created for unimplemented modules where necessary.

Test Case 1	Feed Integration Test
Test Case Name	FeedIntegration
Test Case Description	Validate feeding action correctly updates hunger and energy metrics.
Test Steps	1. Initialize a virtual pet object. 2. Call the feed() method. 3. Verify that hunger decreases and energy increases.
Pre-Requisites	A working instance of the Pet class.
Expected Results	Hunger decreases, energy increases.
Test Category	Integration Test.
Requirement	Feeding functionality as specified in requirements.
Automation	Yes (JUnit).
Date Run	2024-11-30.
Pass/Fail	Pass.
Test Results	Metrics updated correctly.
Remarks	None.

2.2 Validation Testing

Validation testing ensures all requirements are met and the application functions as described in the specification.

Test Case 2	Gameplay Validation Test
Test Case Name	GameValidation
Test Case Description	Ensure all actions (feed, play, rest) maintain metrics within bounds.
Test Steps	1. Create a virtual pet object. 2. Call all action methods sequentially. 3. Confirm metrics stay within 0–100.
Pre-Requisites	Fully implemented action methods.
Expected Results	Metrics never exceed bounds.
Test Category	Validation Test.
Requirement	Consistent metric behaviour.
Automation	No (Manual).
Date Run	2024-11-30.
Pass/Fail	Pass.
Test Results	Metrics behaved correctly.
Remarks	None.

2.3 System testing

System testing evaluates the application as a whole, including GUI functionality and cross-platform compatibility.

Test Case Name	WinCompatability
Test Case Description	Ensure the application runs smoothly on Windows.
Test Steps	1. Install the application on a Windows system. 2. Execute the application. 3. Verify all features function correctly.
Pre-Requisites	Deployed application and Java runtime.
Expected Results	Application starts and all features work as expected.
Test Category	System Test.
Requirement	General system functionality.
Automation	No (Manual).
Date Run	2024-11-30.
Pass/Fail	Pass.
Test Results	Application worked on Windows.
Remarks	None.

3.0 Summary

This document provides a comprehensive approach to testing the PetPal, covering unit, integration, validation, and system testing. The combination of white-box and black-box testing ensures thorough verification of all requirements. Future iterations will include expanded test cases as new features are added.