Test Plan

Date: 10/13/2017

Version: 1.0

Author: Matt Dulany

Table of Contents

* Introduction
* Requirements
* Strategy
* Resources
* Deliverables

Introduction

Purpose

The goal of this document is to:

1. Identify the components of the Tempeturs system that need to be tested.
2. Define the requirements needed to test the system.
3. Layout testing strategies
4. Identify the resources needed for testing purposes
5. List deliverables to be created by testing.

Background

The Tempeturs system is a pet sitter matching site, designed to match possible pet sitters with pet owners. This system contains three main systems. The online front end needed to display information and provide content to user. The java based backend needed to connect users to each other and the elastic search database where user information is stored.

Scope

This front end and backend portion of this system will be tested separately. The front end will be tested manually, while the backend will be unit tested. The overall system will also be tested manually tested.

Backend unit tests will cover the following:

1. The connection to the elastic search database.
2. Interaction between subsystems
3. System functionality

Requirements

The following list lays out the requirements that testing must accomplish.

Database Testing

Must ensure that users information can be entered and stored correctly

Must ensure that user data can be retrieved correctly

Must ensure that appointment data can be entered and retrieved

Must ensure the payment information can be entered and retrieved.

Functional Testing

Verify that a user can be registered successfully

Verify that a user can be logged in correctly

Verify that a pet owner can add a pet as needed

Verify that sitters can enter calendar appointments correctly

Verify that Owners can create pet sitting appointments

Verify that Sitters can accept possible sitting appointments.

Verify that Owners can rate past sitters

Verify that payment transactions are processed correctly

Verify that sitter suggestions can be made for appointments

Verify that alexa can communicate with the system to create an appointment

User Interface Testing

Navigate through each use case verifying that each page can be easily understood

Verify that all documentation is presented clearly

Performance Testing

Verify that request minimize delay

Verify that responses are received without backend delay

Security Testing

Verify Authentication and Online security measures protected data

Strategy

##### Data Integrity and Database Integration Testing

|  |  |
| --- | --- |
| Test Objective: | Ensure database access methods process data properly and without data corruption. |
| Technique: | * Invoke each database access method and process, seeding each with valid and invalid data (or requests for data). * Inspect the database to ensure the data has been populated as intended, all database events occurred properly, or review the returned data to ensure that the correct data was retrieved (for the correct reasons) |
| Completion Criteria: | All database access methods and processes function as designed and without any data corruption. |
| Special Considerations: | * Processes should be invoked manually. * Small or minimally sized databases (limited number of records) should be used to increase the visibility of any non-acceptable events. |

##### Functional Testing

|  |  |
| --- | --- |
| Test Objective: | Ensure proper target-of-test functionality, including navigation, data entry, processing, and retrieval. |
| Technique: | Execute each use case, use case flow, or function, using valid and invalid data, to verify the following:   * The expected results occur when valid data is used. * The appropriate error / warning messages are displayed when invalid data is used |
| Completion Criteria: | * All planned tests have been executed. * All identified defects have been addressed. |
| Special Considerations: | None. |

##### User Interface Testing

|  |  |
| --- | --- |
| Test Objective: | Verify the following:   * Navigation through the target-of-test properly reflects business functions and requirements, including page to page methods * Web objects and characteristics, such as menus, size, position, state, and focus conform to standards. |
| Technique: | Manually run through each page looking for specific criteria. |
| Completion Criteria: | Each window successfully verified to remain consistent with benchmark version or within acceptable standard |
| Special Considerations: | None |

##### Performance Profiling

|  |  |
| --- | --- |
| Test Objective: | Verify performance behaviors for designated transactions or business functions under the following conditions:   * normal anticipated workload * anticipated worse case workload |
| Technique: | Manually verify that data stats for process meet acceptable standards. |
| Completion Criteria: | All statics conform to an acceptable level. |
| Special Considerations: | None |

Resources

We will be using JUnit to perform backend testing. Matt Dulany will be the test engineering in charge of performing and meeting each test requirement as well as ensuring that each test meets completion criteria. These tests will be developed throughout the development process. The will be developed and run alongside functional development to ensure that each step is being designed correctly.

Deliverables

Test Model

This document will be a guide as to what and how each test will need to be created in order to ensure that each test is completed correctly. It is due as of 10/13/2017.

Test Documentation

Logs and output of each test run will be collected created in word documents

Completed Tests

**Data Integrity and Database Integration Testing**

Registration Tests:

Each form of user and user data has a integration test that runs against the database. It attempts to place a user into the database and then pulls the data out and tests that it is correctly stored.:

User Test:

Input: User with username jwild7777.

Criteria: TestUser pulled from database should be equal to User

Output: True.

Owner Test:

Input: Owner with username jwild7777.

Criteria: OwnerUser pulled from database should be equal to Owner

Output: True.

Sitter Test:

Input: Sitter with username jwild7777.

Criteria: TestSitter pulled from database should be equal to Sitter

Output: True.

Pet Test:

Input: Pet with id = to next id in database.

Criteria TestPet pulled from database should be equal to Pet

Output: True.

Each one of these tests has run successfully and is run each time the project is rebuilt on GitLab.

We also have run several integration tests against the backend for every functional flow method. These are the methods like registering a sitter, creating an appointment, cancelling an appointment etc. This functional flow test does not actually run on each build due to the fact that it seems to create a race case in the elastic search database. However, each one of the sections of this test have been run independently and has passed successfully.

ScheduleSitterTest:

Input: Appointment with username jwald77777 and ownerUsername jwild77777

Criteria: TestAppointment pulled from database should be equal to Appointment

Output: True.

##### Functional Testing

We are currently in the process of run end-to-end test on each one of the function calls to make sure that they work correctly with front end integration. So far we have successfully tested both getAppointments methods as well as the getPets method. The rest of the methods are being tested currently. These require certain front-end pages to be developed around them, and these pages are also currently in development which has slow the testing phase down.

GetAppointmentTest:

Input: Ask for appointments of jwild77777

Outpus: The two appointments for jwild77777 that are currently in the database.

##### User Interface Testing

##### Performance Profiling

##### Both of these tests have yet to take place yet as we have not fully developed the user interface or have fully set up the system in order to test the performance of the system. We plan to begin these tests within the next week.