**Tempeturs**

Master Test Plan

Version 1.2

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 02/11/18 | 1.0 | Initial Test Plan | David Millard |
| 02/11/18 | 1.1 | updates | Parakh Jaggi |
| 02/11/18 | 1.2 | Reformatting | David Millard |
|  |  |  |  |

Table of Contents

[1. Introduction 4](#_Toc528965213)

[1.1 Purpose 4](#_Toc528965214)

[1.2 Background 4](#_Toc528965215)

[1.3 Scope 4](#_Toc528965216)

[2. Target Test Items 4](#_Toc528965217)

[3 Test Approach 5](#_Toc528965218)

[3.1 Testing Techniques and Types 5](#_Toc528965219)

[3.1.1 Data and Database Integrity Testing 5](#_Toc528965220)

[3.1.2 Function Testing 5](#_Toc528965221)

[3.1.3 UI Testing 6](#_Toc528965222)

[3.1.4 Performance Profiling 6](#_Toc528965223)

[3.1.5 Load Testing 7](#_Toc528965224)

[3.1.6 Security and Access Control Testing 7](#_Toc528965225)

[3.1.7 Configuration Testing 8](#_Toc528965226)

[4 Resources 8](#_Toc528965227)

[4.1 People and Roles 8](#_Toc528965228)

[4.2 Base System Hardware 10](#_Toc528965229)

[5 Master Milestones 11](#_Toc528965230)

[6 Deliverables 11](#_Toc528965231)

[6.1 Test Evaluation Summaries 11](#_Toc528965232)

[6.2 Test Logs 11](#_Toc528965233)

[6.3 Perceived Quality Reports 11](#_Toc528965234)

**Master Test Plan**

# Introduction

## Purpose

The purpose of the Master Test Plan is to gather all of the information necessary to plan and control the test effort for a given Master. It describes the approach to testing the software, and is the top-level plan generated and used by managers to direct the test effort.

This *Test Plan* for the Tempeturs supports the following objectives:

* Identify project conditions and software components that should be tested.
* List the necessary requirements for testing
* Recommend and describe the testing strategies to be employed
* Identify the required resources
* List the deliverable elements of the project

## Background

The Tempeturs Project allows users to find other users who can watch their pets and schedule a sitting.

The system is comprised of a few major subsystems, including

* Website Manager: this subsystem handles requests from the users, displays profiles, an account, and acts as the medium between users and the rest of the system.
* Database: this subsystem stores all information needed for users and pets, one for each.
* Account: this subsystem has all information about its user and allows its user to change variables related to their profile and schedule a sitting.

## Scope

The Tempetur Project will be unit tested and system tested. Unit tests will address functional quality, while system testing will address issues of scalability and performance.

The interaction of the subsystems will be tested as follows:

1. Website manager to Database
2. Website Manager to Account

The following interfaces will be tested

1. Website manager to its online server

The most critical testing will be that of load and performance testing. This will be addressed as follows:

1. We will have multiple owners simultaneously schedule a sitting with the same sitter for the same time.
2. We will see if a user can schedule a sitting with multiple sitters for the same time.

# Target Test Items

The listing below identifies those test items,software, hardware, and supporting product elements that have been identified as targets for testing. This list represents what items will be tested.

Database Testing

Verify that a user can be entered and retrieved

Verify that a pet can be entered and retrieved

Verify that updating a petlist will update the user’s profile

Functional Testing

Verify that an owner can find sitter with an available time slot.

Verify that a sitter can enter a time schedule that can be modified later.

Verify that an owner can get a list of nearby sitters

Verify that users can add and delete pets.

Verify that sitting prevents the owner and sitter from accepting/requesting a sitting at the time

UI Testing

Verify that the layout is intuitive for inexperienced users.

Performance Profiling

Verify response time of interface from the server

Verify response time of interface to the Tempeturs system

Load Testing

Verify system response with 20 concurrent members

Verify system response with 100 concurrent members

Verify system response with 1000 concurrent members

Verify system response with 5000 concurrent members

Security and Access Control Testing

Verify that non-users cannot view profiles

Verify that users cannot view the account of other users.

Verify that no two users can have the same emails

Configuration Testing

Verify operation using Google Chrome

Verify operation using Mozilla Firefox

## Test Approach

## Testing Techniques and Types

### Data and Database Integrity Testing

|  |  |
| --- | --- |
| Technique Objective: | Ensure database access methods and process function properly without corruption even after connection error |
| 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 and all database events have occurred properly, or review the returned data to ensure that the correct data was retrieved for the correct reasons. |
| Success Criteria: | All database methods and processes work as intended without data corruption |
| Special Considerations: | Processes should be invoked only when the system requests them. |

### Function Testing

|  |  |
| --- | --- |
| Technique Objective: | Exercise target-of-test functionality, including navigation, data entry, processing, and retrieval to observe and log target behavior. |
| Technique: | Execute each use-case scenario’s individual use-case flows or functions and features, using valid and invalid data, to verify that:   * the expected results occur when valid data is used * the appropriate error or warning messages are displayed when invalid data is used * each business rule is properly applied |
| Success Criteria: | The technique supports the testing of:   * all key use-case scenarios * all key features |
| Special Considerations: | None |

### UI Testing

|  |  |
| --- | --- |
| Technique Objective: | Exercise the following to observe and log standards conformance and target behavior:   * Navigation through the target-of-test reflecting business functions and requirements, including window-to-window, field-to- field, and use of access methods (tab keys, mouse movements, accelerator keys). * Window objects and characteristics can be exercised–such as menus, size, position, state, and focus. |
| Technique: | Create or modify tests for each window to verify proper navigation and object states for each application window and object. |
| Success Criteria: | The technique supports the testing of each major screen or window that will be used extensively by the end user. |
| Special Considerations: | Not all properties for custom and third-party objects can be accessed. |

### Performance Profiling

|  |  |
| --- | --- |
| Technique Objective: | * Exercise behaviors for designated functional transactions or business functions under the following conditions to observe and log target behavior and application performance data: * normal anticipated workload * anticipated worst-case workload |
| Technique: | * Use Test Procedures developed for Function or Business Cycle Testing. * Modify data files to increase the number of transactions or the scripts to increase the number of Masters that occur in each transaction. * Scripts should be run on one machine (best case to benchmark single user, single transaction) and should be repeated with multiple clients (virtual or actual, see Special Considerations below). |
| Success Criteria: | * The technique supports testing: * Single Transaction or single user: Successful emulation of the transaction scripts without any failures due to test implementation problems. * Multiple transactions or multiple users: Successful emulation of the workload without any failures due to test implementation problems. |
| Special Considerations: | Comprehensive performance testing includes having a background workload on the server.  There are several methods that can be used to perform this, including:   * “Drive transactions” directly to the server, usually in the form of Structured Query Language (SQL) calls. * Create “virtual” user load to simulate many clients, usually several hundred. Remote Terminal Emulation tools are used to accomplish this load. This technique can also be used to load the network with“traffic”. * Use multiple physical clients, each running test scripts, to place a load on the system.   Performance testing should be performed on a dedicated machine or at a dedicated time. This permits full control and accurate measurement.  The databases used for Performance Testing should be either actual size or scaled equally. |

### Load Testing

|  |  |
| --- | --- |
| Technique Objective: | Exercise designated transactions or business cases under varying workload conditions to observe and log target behavior and system performance data. |
| Technique: | * Use Transaction Test Scripts developed for Function or Business Cycle Testing as a basis, but remember to remove unnecessaryinteractions and delays. * Modify data files to increase the number of transactions or the tests to increase the number of times each transaction occurs. * Workloads should include (for example, Daily, Weekly, Monthly and so forth) Peak loads. * Workloads should represent both Average as well as Peak loads. * Workloads should represent both Instantaneous and Sustained Peaks. * The Workloads should be executed under different Test Environment Configurations. |
| Success Criteria: | The technique supports the testing of Workload Emulation, which is the successful emulation of the workload without any failures due to test implementation problems. |
| Special Considerations: | * Load testing should be performed on a dedicated machine or at a dedicated time. This permits full control and accurate measurement. * The databases used for load testing should be either actual size or scaled equally. |

### Security and Access Control Testing

|  |  |
| --- | --- |
| Technique Objective: | Exercise the target-of-test under the following conditions to observe and log target behavior:   * Application-level Security: an actor can access only those functions or data for which their user type is provided permissions. * System-level Security: only those actors with access to the system and applications are permitted to access them. |
| Technique: | * Application-level Security: Identify and list each user type and the functions or data each type has permissions for.   + Create tests for each user type and verify each permission by creating transactions specific to each user type.   + Modify user type and re-run tests for same users. In each case, verify those additional functions or data are correctly available or denied. * System-level Access |
| Success Criteria: | The technique supports the testing of for each known actor type the appropriate functions or data affected by security settings can be tested. |
| Special Considerations: | Access to the system must be reviewed or discussed with the appropriate network or systems administrator. This testing may not be required as it may be a function of network or systems administration. |

### Configuration Testing

|  |  |
| --- | --- |
| Technique Objective: | Verify that the browsers function properly on the hardware with the necessary software configurations. |
| Technique: | * Use Function Test scripts. * Open and close various non-target-of-test related software, such as Microsoft Excel and Word applications, either as part of the test or prior to the start of the test. * Execute selected transactions to simulate actors interacting with the target-of-test and the non-target-of-test software. * Repeat the above process, minimizing the available conventional memory on the client workstation. |
| Success Criteria: | The technique supports the testing of one or more combinations of the target test items running in expected, supported deployment environments. |
| Special Considerations: | * What non-target-of-test software is needed, is available, and what is accessible on the desktop? * What applications are typically used? * What data are the applications running; for example, a large spreadsheet opened in Excel or a 100-page document in Word? * The entire system’s netware, network servers, databases, and so on, also needs to be documented as part of this test. |

# Resources

## People and Roles

This table shows the staffing assumptions for the test effort.

|  |  |  |
| --- | --- | --- |
| **Human Resources** | | |
| **Role** | **Minimum Resources Recommended**  **(number of full-time roles allocated)** | **Specific Responsibilities or Comments** |
| Test Manager | 1 | Provides management oversight.  Responsibilities include:   * planning and logistics * agree mission * identify motivators * acquire appropriate resources * present management reporting * advocate the interests of test * evaluate effectiveness of test effort |
| Test Analyst | 1 | Identifies and defines the specific tests to be conducted.  Responsibilities include:   * identify test ideas * define test details * determine test results * document change requests * evaluate product quality |
| Test Designer | 1 | Defines the technical approach to the implementation of the test effort.  Responsibilities include:   * define test approach * define test automation architecture * verify test techniques * define testability elements * structure test implementation |
| Tester | 1 | Implements and executes the tests.  Responsibilities include:   * implement tests and test suites * execute test suites * log results * analyze and recover from test failures * document incidents |
| Test System Administrator | 1 | Ensures test environment and assets are managed and maintained.  Responsibilities include:   * administer test management system * install and support access to, and recovery of, test environment configurations and test labs |
| Database Administrator, Database Manager | 1 | Ensures test data (database) environment and assets are managed and maintained.  Responsibilities include:   * support the administration of test data and test beds (database). |
| Designer | 1 | Identifies and defines the operations, attributes, and associations of the test classes.  Responsibilities include:   * defines the test classes required to support testability requirements as defined by the test team |
| Implementer | 1 | Implements and unit tests the test classes and test packages.  Responsibilities include:   * creates the test components required to support testability requirements as defined by the designer |

## Base System Hardware

The following table sets forth the system resources for the test effort presented in this *Test Plan*.

|  |  |  |
| --- | --- | --- |
| **System Resources** | | |
| **Resource** | **Quantity** | **Name and Type** |
| Database Server |  | ElasticSearch |
| —Network or Subnet |  |  |
| —Server Name |  |  |
| —Database Name |  |  |
| Client Test PCs |  |  |
| —Include special configuration requirements |  |  |
| Test Repository |  |  |
| —Network or Subnet |  |  |
| —Server Name |  |  |
| Test Development PCs |  |  |

# Master Milestones

|  |  |  |
| --- | --- | --- |
| **Milestone** | **Start Date** | **End Date** |
| Plan Test | 6/11/18 |  |
| Design Test |  |  |
| Implement Test |  |  |
| Execute Test |  |  |
| Evaluate Test |  |  |

# Deliverables

## Test Evaluation Summaries

A brief form will be created with then name and ID of the test, and will cover the pre and post conditions required of the tests and the actual results of the test.

## Test Logs

A text editor, probably Microsoft Word, will be used to record the test logs.

## Perceived Quality Reports

Defects will be recorded in a similar fashion.