4/4/2017

# Test Plan

EXIT6 – Trip Planning Application for Singapore

**Delivered by,**
**EXIT6 Team**
Sim Long Siang(U1522053H)
Seshadri Madhavan (U1322790J)
Huang Jian Wei(U1521567A)
Lim Zi Yang(U1522218E)
Lim Hao Zhe(U1521589B)
Tan Jun Qiu(U1321638C)

# Document Change Record

| Revision | Description of Change | Approved by | Date |
|---|---|---|---|
| 0.1.0 | Initial Template | EXIT 6 Team | 21/03/2017 |
| 0.5.0 | Addition of Test Cases | Lim Hao Zhe | 28/03/2017 |
| 1.0.0 | Release | Seshadri Madhavan | 04/04/17 |

## Contents

## List of Tables

## Test Plan Identifier

Test Plan Identification Number: MTP 1.0 - EXIT6 version 1.0

The structure of this document is primarily based on the IEEE 829-1998 Standard for Software Test Documentation. Additional reference standards include IEEE 1008 (Unit Testing), 1012 & 1059(Validation & Verification) and 1074(Software Life Cycle process).

## Introduction

This document outlines the tests that are planned and should be applied to our android application EXIT6.

“EXIT6”, an android based mobile application, is a trip recommendation application that aims to generate a recommended itinerary of Singapore based on the user’s interests and preference, employing the use of algorithms to recommend a personalized itinerary of Singapore to tourists without the need for them to plan one manually. It will feature both a Recommendation mode where an itinerary is generated by our application algorithm and a Do-it-Yourself (DIY) mode which provides flexibility to users who desire control over their itinerary.

Details such as travel times, travel directions, location information, recommended amount of time spend and special event information will be provided for each location in the generated itinerary. The user can save the recommended itinerary as a “Saved Trip” under their own account profile and rate/review such saved trips after they have completed them.

Testing of EXIT6 consists of the following phases:

1. Unit Testing – This will be done in conjunction with our regular code reviews, to ensure consistency of code and correctness of individual module output
2. Integration Testing – To be done at the point when modules of each application is completed and are integrated together. We will ensure that the modules are working simultaneously without significant error.

3. System Testing – To be done when functionalities have been implemented. We will ensure that the functionalities work and do not conflict with other functionalities already present. Factors to consider include.
    a. Usability
    b. Reliability
    c. Performance
4. Acceptance Testing – To be done post implementation, when EXIT6 as a software product is complete. We will verify that the installed system works in an end-user environment.

## Type of Tests

Two different types testing are performed to find errors within the application.

1. Black-box testing

   Black-box testing is performed to ensure the functionality is working as intended. All functionalities of the application are tested thoroughly without viewing at the codes. The testing is done by entering different inputs, both correct and incorrect types of input, to test the response of the application. The testing is considered a success if only the expected outcome of the application is the same as the actual response.

2. White-box testing

   In white box testing the code will be inspected to check the internal structure of the application for correctness. The testing will be applied at both the unit and integration levels. Through white-box testing, codes can be optimized which improves the performance of the application.

## Test Items

EXIT6 comprises multiple individual software components

The items that need to be tested includes

1. EXIT6 v1.0 Android Application
2. PostgreSQL Database
3. FTP server (Glassfish)

## Features to be tested

### Unit and Integration Level Test Plan

The following are the features to be tested for the application based on the initial analysis:

*Table 1 Unit and Integration Test Plan*

| **Feature** | **Importance** | **Remarks** |
|---|---|---|
| User Log-In | High | Core Functionality of Platform, required for all other features to be tested and implemented |
| User Log-Out | Medium | Core Functionality, required to ensure user security |
| User Create Account | High | Core Functionality of Platform, required for all other features to be tested and implemented |
| Create Recommended Itinerary | High | Core Functionality of Platform, required for all itinerary based features to be implemented |
| Select Places of Interest | High | Core Functionality that is an itinerary feature |
| View Top Rated Itineraries | High | Core Functionality |
| Modify Recommended Itinerary | High | Core Functionality that is an itinerary feature |
| Update Account Details | Medium | Core Functionality |
| Save Trip | High | Core Functionality |
| Export Trip | High | Core Functionality |

## System-Level Test Plan

The following are the features to be tested for the application based on the initial analysis:

*Table 2 System Level Test Plan*

| **Feature** | **Importance** | **Remarks** |
|---|---|---|
| Displaying user details on main page when user logs in | High | Core Functionality of Platform, required for all other features to be tested and implemented |
| Displaying trip details from server when user submits a trip request | High | Core Functionality, required to ensure user security |
| Sending user input data to the server | High | Core Functionality of Platform, required for all other features to be tested and implemented |
| Displaying trip itineraries of other users | Medium | Core Functionality of Platform, required for all event based features to be implemented |
| Displaying of user's personal trip itineraries | High | Core Functionality |
| Uploading completed trips to the server | Medium | |
| Modifying trip itineraries | Low | |

### User Acceptance Test Plan

The following are the features to be tested for the application based on the initial analysis:

*Table 3 User Acceptance Test Plan*

| Feature | Importance | Remarks |
|---|---|---|
| Multiple users simultaneously logging in | High | Core Functionality of Platform, required for all other features to be tested and implemented |
| Time taken for trip details retrieval when 100 users are online | High | Core Functionality, required to ensure user security |
| Installation of the application on different Android operating systems | High | Core Functionality of Platform, required for all other features to be tested and implemented |
| Navigation throughout the application with no application termination errors | Medium | Core Functionality of Platform, required for all event based features to be implemented |

## Features not to be tested

The following list of features are not to be tested based on the initial analysis:

*Table 4 Reasoning of Tests to be eliminated*

| Feature | Reason for not including |
|---|---|
| Rating of trips | Less relevant in the current stage of development |

## Approach

The EXIT6 team will employ priority-based testing as our main testing approach. Each of our test cases will have a priority assigned to it; High, Medium or Low priority. Test cases with high priority will be executed first, followed by test cases with medium priority and finally test cases with low priority. This ensures the key functionalities of the application are tested and functioning as intended first before focusing on other aspects of the application.

For white-box testing, we will be conducting automated testing using the JUnit test framework to test correct or incorrect inputs.  Furthermore, we will manually inspect our code to ensure a consistent coding style and readable code for easier maintainability.

As for black-box testing, we will formulate test cases for all functionalities based on boundary value testing and equivalence class testing. This will ensure a sufficient coverage of test scenarios, with minimal effort and time. For each test case, a range of values (valid and invalid) of inputs will be tested. Different set of input combination will be entered to test the response of the application. This ensures all errors that user may encounter is caught during this phase. Appropriate error messages will be displayed to prevent user from making same error again.

Also, during the testing process execution, the source code will be frozen and no changes will be executed simultaneously, so that the test cases will not be affected. However, changes to rectify the errors and bugs surfaced will be permitted.

Basic metrics such as test effort, test cases executed, and issues identified from test case will be logged. More specific test metrics such as code coverage will be omitted due to the limited time and resources available.

## Items Pass/Fail Criteria

Our project team has different criterion for different test items. This would depend on the priority level of the test items. The pass percentage score for each level is as follows:

*Table 5 Criteria for Test Result Evaluation*

| Priority Level | Definition | Pass Percentage |
|---|---|---|

| | | |
|---|---|---|
| Highest | Critical system functions | 100% |
| High | Important functions, Fundamental | 100% |
| Medium | Non-core functions | 98% |
| Low | Additional Features, low significance | 90% |

We will categorize the issues into 1 of 4 categories, and the criteria for passing the test will be dependent upon that. Occurrence of issues above the maximum allowable percentage is discouraged. The issues will be addressed to ensure maximum allowable percentage is not hit.

*Table 6  Acceptable Failure Rate of Tests*

| **Category** | **Description** | **Maximum allowable percentage** |
|---|---|---|
| Critical | Failure to add/modify database, Unexpected application crash, Data Loss | 0% |
| Significant | Major issues such as wrong event information, unable to execute core functions (create trip, login etc.) | 0% |
| Minor | Minor issues such as information format and layout | 4% |
| Low | Interface elements not showing, Page not properly scaled | 2% |

## Suspension Criteria and Resumption Requirements

Testing must be suspended temporarily when certain criteria are met. Suspending is necessary as the criteria prevent testing from occurring or may lead to inaccurate results. The following are the list of suspensions criteria:

1. Connection to server is unstable
2. Server is down
3. Connection to database is unstable

4. Database is down
5. Bug which crashes the application

Testing is resumed when the following conditions are met:

1. Server is up and working properly
2. Database is up and working properly
3. Any major bug(s) found are addressed

Some test cases have entrance criteria prerequisites. For example, Plan Trips test case is dependent on the Log-In test case passing first. If the perquisite test case has failed, the subsequent test cases might be skipped as well.

## Test Deliverables

Following are the deliverables for the QA Test Engineering Process:

1. Test Plan Document (This Document)
2. Test Cases
3. Test Design Specifications
4. Error Logs and Execution Logs from the tools used (JUnit test logs)
5. Problem reports and corrective actions
6. Test Input and Output report

The abovementioned deliverables need not be presented in separate documents, but they will have to be sectioned accordingly.

## Testing Tasks

The testing tasks include and is carried out in the following sequence:

1. Preparation of the test plan

2. Preparation of the test design specification
3. Preparation of the test cases
4. Preparation of the Client-side test environment
5. Preparation of the Server-side test environment
6. Performing the test procedures and generate error and execution logs
7. Resolve problem reports by taking corrective actions
8. Repeat tasks (6) - (9) until all test procedures are successful
9. Prepare the test summary report

## Environmental Needs

For environmental needs, we will be focusing on two parts, namely the Client-side environment and the Server-side environment.

### Client-side Environment

The following client-side hardware is needed for testing purposes. The testing team will use their own android phones to test the application.

**Client Mobile Devices Specifications**

1. Mobile devices running on Android OS with API 14 (Android 4.0 Ice Cream Sandwich) or above.
2. Mobile devices must have at least 70 Megabytes of free space for installation.
3. Mobile devices must support 3G wireless communication standards.
4. Mobile devices must have a minimum screen size of 4.0 inches and support a screen resolution of 640 x 960.

## Server-side Environment

The following server-side hardware is needed for testing purposes. Two exact servers will be created and deployed. One server will be used by the development team for unit and integration testing while the other server will be used by the testing team for system testing.

**Server Specifications**

- 64 bits Windows operating system running Windows 7 or higher.
- Server computer with at least 8 Gigabytes of memory space (4GB RAM)
- Server computer with 100 Gigabytes of free space
- Server computer with Intel i5 CPU
- Server computer with PostgreSQL database installed
- Server computer with Eclipse J2EE IDE installed to run Glassfish server

- Glassfish server version 4.1.1 or higher is required
- Notepad++ to facilitate viewing and coding of .xml files (Optional)

**Additional Testing Resources**

- Android Studio built-in emulators
- Eclipse J2EE IDE
- Notepad++
- NTU Virtual Servers
- pgAdmin (PostgreSQL administration and management tools)

## Responsibilities

Following are the responsibilities of the team members as per the Job Scope in the project:

*Table 7 Responsibilities for Team Members*

| **Plan Area** | **Person Responsible** |
|---|---|
| Risk Identification and Management of Test plan related items | Tan Jun Qiu |
| Determining Features/Requirements to be tested | Huang Jian Wei, Sim Long Siang |
| Deciding and Drafting overall test strategy | Lim Hao Zhe, Lim Zi Yang |
| Coordinating all test elements and coordinating test schedule with other work tasks | Seshadri Madhavan |
| Managing changes required during the test process | Sim Long Siang |
| Provision of required training/Subject Matter Expert | Lim Hao Zhe, Lim Zi Yang |
| Execution of Test Cases | Lim Hao Zhe, Lim Zi Yang |

## Staffing and Training Needs

The Quality Assurance Manager and Lead Developer, Long Siang and Jian Wei will oversee generating the test cases, and planning and scheduling of the testing processes. To do so, they would have to have expertise and knowledge in the subject areas of:

- Control-flow testing
- Equivalence Class testing
- Boundary Class Testing

And others, if required. In addition, they are required to know how to use JUnit to generate feasible white-box test cases and execute them.

## Schedule

The testing process is integral in all stages of our implementation, from the start to the end. Since we are using the AGILE methodology, from the very first implementation of the prototype till the EXIT6 platform is complete, the testing process will be carried out every iteration cycle. Our unit-level testing and integration testing will take place during each iterative development cycle. Our system level testing will be done after near the end of each iterative cycle, to test the prototype. User Acceptance Testing will take place nearing the end of the entire project.