CitySim2017 Master Test Plan

Version 1.0

Last Edited: 07/06/2017

Created by Jack Waayer

**Introduction**

This is the highest level document of the two Test Plan documents. It provides a high level overview of what is required for successful development of the CitySim2017 test suite. The plan is constrained solely to the CitySim2017 application and with the small amount of time provided for development and planning, the testing has been limited to solely unit testing.

The following provides a guideline for lower level plans such as the Unit Testing Plan and Test Cases. It details what is to be tested, how it will be tested, what is required and the risks involved.

**References**

CitySim2017 Requirements Specification

CitySim2017 Unit Test Plan

testingToDo.tdl

**Test Items**

CitySim2017 console application - v1.0

**Features to be Tested**

Input / output of functions within the solution

**Features to not be Tested**

* Compatibility - This is assumed to be out of the scope of the project.
* The applications performance - The application is too small to gain benefit from performance testing.
* Non-functional features - Non-functional features cannot be tested with unit tests.
* Security - It is assumed that security is out of the scope of the project.
* Usability - The application would not benefit from usability testing.
* Acceptance of the application - This is assumed to be out of the scope of the project.

**Approach**

It is assumed that the application will be based on graph theory for navigating through the city. A matrix will be used to simulate this. The following is the interpreted version of the client’s requirements specification:

* FUN-CITY-LOCS: Four locations which the drivers can visit; Akina, Mayfair, Mahora and Stortford Lodge. It is assumed that locations are only connected to their neighbouring locations for example, Akina is connected to Stortford Lodge and Mayfair but not Mahora.
* FUN-OUTSIDE-CITY: Outside City is a fifth location which a driver can visit. This location can be visited after leaving any of the FUN-CITY-LOCS and is accessed via one of the adjacent FUN-EXITS.
* FUN-EXITS: Four main exits from the city; Karamu Road, Havelock Road, Railway Road, Omahu Road. Each FUN-EXITS is connected to two adjacent FUN-CITY-LOCS via FUN-STREETS.
* FUN-STREETS: A path of one or more linked streets/roads which connect the FUN-EXITS to their adjacent FUN-CITY-LOCS.
* FUN-FIVE-DRIVERS: Five drivers numbered 1 through 5 will drive through the city one after the other. It is assumed the simulation will stop after the fifth driver has completed the drive.
* FUN-AKINA-COUNT: At the end of each drive, the application shall print out how many times that driver visited Akina. The format shall be “Driver n met with John Jamieson x time(s).”, where n is the driver number and x is the number of times the driver visited the Akina location. If Akina is the FUN-START-LOC this also counts as an Akina visit.
* FUN-AKINA-EDGES: If a driver visits Akina three or more times, an additional line shall be printed stating “This driver needed lots of help!”. If a driver never visits Akina, the additional line shall be printed stating “That passenger missed out!”. This additional line shall be printed as the last line for each drive. Any other number of visits does not require an additional line to be printed.
* FUN-START-LOC: A driver can start in any of the four locations in FUN-CITY-LOCS. Drivers cannot start outside the city.
* FUN-ITERATION: An iteration is a driver driving from their current location to one of the connected locations whether it be one of the FUN-CITY-LOCS or FUN-OUTSIDE-CITY. The next location is chosen randomly using a random number generated from a seed provided by the user in FUN-ARGS. If the next location is one of the FUN-CITY-LOCS, the driver will drive from the current location on the appropriate FUN-STREETS to the FUN-EXIT between the two locations and then continue on the FUN-STREETS to the next location. If the next location is FUN-OUTSIDE-CITY, the simulation will randomly pick a FUN-EXIT adjacent to the current location and then the driver will take the FUN-STREETS to that FUN-EXIT and exit the city via that FUN-EXIT to the FUN-OUTSIDE-CITY.
* FUN-END: A drive will end when the driver goes to the FUN-OUTSIDE-CITY location.
* FUN-ARGS: The application shall accept an integer seed from the command line for the pseudorandom number generator. No other arguments shall be accepted. If no arguments are provided, more than one argument is provided, or the single argument is not an integer, the application shall print an error and exit.
* FUN-OTHER-CITIES: If a driver exits the city via Karamu road, then the application shall print an extra line stating “Driver n has gone to Napier.”. If a driver exits the city via Omahu Road, then the application shall print an extra line stating “Driver n has gone to Flaxmere.”, where n is the drivers number. Railway and Havelock Road are assumed to not have an extra line when used as an exit.
* FUN-DASHES: After each drive displays on the screen, a line of five dashes (i.e., -----) shall be printed. This line of dashes shall occur after all information from that drive has been printed out.

The application will be tested using the C# language within the Visual Studio 2015 development environment using the .NET Framework.

This approach is focused purely on automated, white-box testing. All functions will be thoroughly tested using unit tests. All unit tests will be contained within a new project in the existing solution. Tests which test the same function will be contained in the same .cs file.

Each unit test will contain:

* Setup code
* Pre-conditions
* Execution steps
* Post-conditions
* Tear down code

Features which are not required will be given lower priority as they are not essential to complete the project.

Version control shall be maintained through GitHub (<https://github.com/>). Team members maintain issues and commit messages detailing what was done and what they will be doing next.

A schedule will be maintained using ToDoList (<http://abstractspoon.weebly.com/>). Team members must keep this up to date throughout the testing process.

**Item Pass/Fail Criteria**

Unit Test Plan successfully executed and completed.

**Suspension Criteria and Resumption Requirements**

The testing process will be too short for suspension criteria to be of any benefit.

**Test Deliverables**

Executive Summary

Interpreted Requirements

Master Test Plan

Unit Test Plan

Findings Report

**Testing Tasks**

* Interpret Client Requirements – Must be done before any other activities can commence. Approximate one day completion.
* Create Master Test Plan – Must be completed before the Unit Test Plan. Approximate one week completion.
* Create Unit Test Plan – Must be completed before Test Cases and Unit Tests. Approximate one week completion.
* Create Test Cases – Can be created along with Unit Tests. Approximate two week completion.
* Create Unit Tests – Once each Test Case is created a Unit Test can be created for it. Approximate three week completion.
* Execute Unit Tests
* Create a Findings Report - Detailing test findings, discoveries, difficulties and observations etc. Approximate one week completion.
* Write an Executive Summary – Overviews testing framework and rationale for the chosen language. Approximate one day completion.

**Environmental Needs**

Microsoft Visual Studio 2015 - Version 14.0.25425.01

Microsoft .NET Framework - Version 4.7.02046

ToDoList - Version 7.1.1.0

Git - Version 2.13.0

Moq.4.7.25

**Responsibilities**

Jack Waayer (Project Lead, Testing Developer) - Keep ToDoList up to date, interpret client requirements, develop a Master Test Plan, develop a Unit Test Plan, create Test Cases, create unit tests following the Master and Unit Test Plans, maintain Test Logs, create a Findings Report and write up an Executive Summary. Maintain backups throughout the entire process.

Jake Hussey (Testing Developer) - Assist with the Master Test Plan, assist with the Unit Test Plan, create unit tests following the Master and Unit Test Plans.

**Staffing and Training Needs**

Team members must learn the Visual Studio 2015 environment, the basics of the C# language and the .NET Framework, and how to use ToDoList. Simple Git commands must be learnt in order to create and maintain the GitHub repository.

Unit testing is the focus of the test plan and best practices must be adopted by the team to ensure the best outcome.

**Schedule**

Refer to testingToDo.tdl within the ToDoList.exe.

**Risks and Contingencies**

The following is a matrix showing potential risks, the likelihood they will occur and impact they will have on the testing process.

|  |  |  |
| --- | --- | --- |
|  | Likelihood (1 Low – 5 High) | Impact (1 Low – 5 High) |
| Loss of data | 3 | 5 |
| Lack of human resources | 2 | 3 |
| Late delivery of test deliverables | 2 | 5 |
| Lack of technical knowledge | 1 | 4 |

Loss of data – Dependant on the scale of data lost the entire testing process could extend by a matter of weeks if no mitigation is put in place. Mitigation for this includes, cloud storage backups and external HDD backups of all documents and Visual Studio solutions.

Lack of human resources – This will majorly effect the schedule for the testing process. Tasks within the testingToDo.tdl will have to be updated and re-allocated to accommodate for the lack of human resources. The final deadline will not be changed so the amount of time and effort exerted into tasks must be limited to ensure completion of the testing process. To help minimize the likelihood of this, both team members are to attend all classes and communicate outside campus.

Late delivery of test deliverables – This is not an option. Thorough planning/scheduling must be put in place to ensure deliverables are complete before deadlines. If running behind schedule, update tasks to only include essential details for the completion of the testing process.

Lack of technical knowledge – This would result in slow development of tests and program code causing delays in the schedule. Both team members are expected to research the C# language, .NET framework and the Visual Studio 2015 environment. Once comfortable with these technologies they can proceed with the test plan.

**Approvals**

This document is complete and meets expected standards.

Signed: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_