G-One Debuggers

Moway

Test Plan Report

For Virtual self-driving car using reinforcement learning

Version 1.0

V.Sajeevan

160544C

5th April, 2019

Revision History

Date	Version	Description	Author
5/04/2019	1.0	Document created with initial plans	`V.Sajeeavn

Table of Contents

Test Plan

1. Evaluation Mission and Test Motivation

1.1. Background

This project involves creating a virtual self-driving car using RL simulation framework based on the requirements and design documents which were submitted in the previous iteration. This product will be developed by the implementation using Python. A comprehensive test plan has been developed to ensure that the system conforms to the specifications, design and to perform quality assurance on the final product. This will enable to release as much as possible bug-free virtual self-driving car simulation framework and minimize the risk of software failure. The test plan will allow verifying if the final product successfully meets the specifications which were specified in the previous documents (requirement and architecture document) with a variety of testing techniques. The plan will also help in fault detection with the test cases that have been designed.

1.2. Evaluation Mission

The main objectives of the testing plan are to make sure that the specified requirements of the SRS have been achieved successfully, making sure that the specifications of the architecture document have been achieved successfully, ensuring that the risk of software failure is minimized as much as possible and detecting the defects and correct them before go-live. Fulfilling these objectives will help to produce a stable version which obtains the expected outcomes successfully. Although there are no approaches or methods to guarantee that the system is completely free of defects, the following test plan will help to reduce faults of the system and provide an improved product.

1.3. Test Motivation

The targeted test items listed below will be the motivation for testing in this phase.

Function Testing: Will ensure that the use cases have been met.

User Interface Testing: Will verify if the requirements of the GUI have been implemented as specified.

Data and Database Integrity Testing: Units will be integrated with other units to see if they work correctly together. It is done by giving the input details programmatically and checking the activities.

Performance Profiling: Ensure that the system's performance is at an acceptable level.

Load Testing: See how the system performs when being used at its limits.

Configuration Testing: Ensure that the system works correctly under different environment configurations.

Security and Access Control Testing: Ensure to protect data and maintain functionality as intended and check whether the requests to access a particular resource should be granted or denied.

Failover and Recovery Testing: Testing verifies product in terms of ability to confront and successfully recover from possible failures, arising from software bugs, hardware failure or communication problems.

2. Target Test Items

The following have been identified as targets for testing:

In this section, I will list the target test items. For ease of reference, I have categorized the test items by motivation.

Integration Testing

During integration testing, we will be testing components separately, and then integrating them together one by one, and testing them again. It is done by giving the input details programmatically and checking the activities. Below is a list of the test items for which integration tests were documented and tested. Donor registration, view donor detail, updating details and finding nearby locations. Add/remove obstacles, Add Manned vehicles, Choose the map, Store trained data.

Function Testing

Function testing consists of testing all the requirements and specifications, as per the requirements and specifications document. In essence, the list of functions to test corresponds to the list of use cases and requirements in the requirements document. Due to the importance of function testing, we have included detailed test cases for all the product functions. Below is the list of functions that were tested. Add/remove obstacles, Add Manned vehicles, Choose the map, Store trained data.

User Interface testing

User interface testing is concerned with making sure that each functionality concerning the user interface works as per the requirements defined in the design document and check whether the user can go back and forward from any particular activity, button presses. For the user interface, the possible interactions with the system will be tested in great detail. Below is a list of the User Interface items that were tested. Simulating, Manned vehicle running, Autonomous vehicle running.

Performance Profiling

Performance profiling is concerned with testing the different response times of the software and, how many concurrent users can be used in this system at one time. In these types of tests, we have focused mainly on the following test items. Simulating, Get training.

Security & access control

Security and access control are concerned with testing for unauthorized control. In these types of tests, we have focused mainly on the following test items. Store trained data, Access trained data.

3. Test Approach

Where possible tests will be automated by using the Robot Framework. Before any code is checked it must pass the appropriate unit tests. Then it must be fixed. Also, re-testing of the code will be done to make sure that defect has been fixed and there no bugs produced due to change in code.

3.1. Testing Techniques and Types

3.1.1. Function Testing

Functional testing will be performed to verify all functional requirements are met successfully. In functional testing basically, the testing of the functions of component or system is done. It refers to activities that verify a specific action or function of the code. Functional test tends to answer the questions like “can the user do this” or “does this particular feature work”. This is typically described in a requirements specification or in a functional specification

Technique Objective:	• Verify system functional requirements and Ensure proper application navigation, data entry, processing, and retrieval.

Technique:	• Ensure new changes did not adversely affect other parts of the system. • Ensure every function produces its expected outcome. • Check whether the expected results occur when valid data is used. • Check whether the appropriate error or warning messages are displayed when invalid data is used. • Ensure all functions combine to deliver the desired result. • Ensure every line of code executes properly.
Oracles:	
Required Tools:	Pycharm and Debugger
Success Criteria:	• All planned tests have been executed. • All identified defects have been addressed.
Special Considerations:	

3.1.2. User Interface Testing

This would help to identify whether the requirements of the user interfaces have been implemented as specified.

Technique Objective:	• Identifies the presence of defects in the product. • Navigation through the application properly reflects business functions and requirements, including a window to window, field to field, and use of access methods. • It evaluates design elements such as layout, colors, fonts, font sizes, labels, text formatting, captions, buttons, maps, and vehicles.
Technique:	• Verify All Navigations • Check Screen Validations • Check whether the button presses are working correctly • Check whether the clickable buttons are shown correctly • Check whether the user can go back and forward from any particular screen
Oracles:	
Required Tools:	Squish GU testing tool
Success Criteria:	• All window objects can be exercised, proper navigation through test target, and test target acts as expected.

	• Each window successfully verified to remain consistent with the benchmark version or within an acceptable standard. • All button presses are working correctly. It's Easy to identify clickable buttons. The user can go easily back and forward
Special Considerations:	

3.1.3. Data and Database Integrity Testing

This testing verifies whether the data in the database is accurate and whether it functions as expected in the given application.

Technique Objective:	• Ensure Database access methods and processes function properly and without data corruption and Verify the data integrity of the imported schema independent of the UI.
Technique:	• Verify that Virtual self-driving car can access and analyze any data in tables. Verify that, when a particular set of data is saved to the database, each value gets saved fully, and the truncation of strings and rounding of numeric values do not occur.
Oracles:	
Required Tools:	Pycharm and Debugger
Success Criteria:	• Users authenticated correctly. All database access methods and processes function as designed and without any data corruption, All schema elements can be stored and retired with no data loss
Special Considerations:	

3.1.4. Performance Profiling

Performance testing measures response times, transaction rates, and other time-sensitive requirements. The goal of Performance testing is to verify and validate the performance requirements have been achieved.

Technique Objective:	• Measures the quality attributes of the system, such as scalability, reliability and resource usage.

	• Determine whether the system will be able to sustain the workload. • Discover the system's performance under sustained use. • Measure which parts of the system or workload cause the system to perform badly.
Technique:	• Check the response time of the decisions made by the system(finding shortest distance location) measured is at an acceptable level.
Oracles:	
Required Tools:	Multi-Mechanize
Success Criteria:	• Single vehicle: Successful emulation without any failures due to test implementation problems. • Multiple vehicles: Successful emulation of the workload without any failures due to test implementation problems.
Special Considerations:	• Performance testing should be performed on a dedicated machine or at a dedicated time. This permits full control and accurate measurement.

3.1.5. Load Testing

Load testing measures subject the system-under-test to varying workloads to evaluate the system's ability to continue to function properly under these different workloads. The goal of load testing is to determine and ensure that the system functions properly beyond the expected maximum workload. Additionally, load testing evaluates the performance characteristics (response times, transaction rates, and other time-sensitive issues).

Technique Objective:	• Verify System Response time, throughput, resource utilization, and maximum user load for designated transactions or business cases under varying workload conditions. • Determine how the application behaves when multiple users hit it simultaneously.
Technique:	• Modify data files (to increase the number of transactions) or the tests to increase the number of times each transaction occurs. • Check when there are more users try to access the response time of the donor details, the finding nearby location is acceptable.
Oracles:	
Required Tools:	Locust

Success Criteria:	• Multiple Vehicles: Successful completion of the tests without any failures and within acceptable time allocation.
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.

3.1.6. Security and Access Control Testing

It ensures to protect data and maintain functionality as intended and check whether the requests to access a particular resource should be granted or denied.

Technique Objective:	• Verify that Trained data is stored correctly and accessible by autonomous vehicles.
Technique:	• Test with different phenomena using stored data
Oracles:	
Required Tools:	Pycharm and Debugger
Success Criteria:	• For each known vehicle type the appropriate function/data are available and all function as expected and run in prior Application Function tests
Special Considerations:	

3.1.7. Failover and Recovery Testing

It verifies product in terms of ability to confront and successfully recover from possible failures, arising from software bugs, hardware failure or communication problems.

Technique Objective:	• Failover testing is a testing technique that validates a system's ability to be able to allocate extra resource and to move operations to back-up systems during the failure due to one or the other reasons. • Discover the specific point at which failure occurs. Recovery testing is basically done in order to check how fast and better the application can recover against any type of crash or hardware failure

Technique:	● While the application is running, suddenly restart the laptop or computer, and afterwards check the validness of the application's data integrity
Oracles:	
Required Tools:	Robot Framework
Success Criteria:	● One or more simulated recoveries involving one or more combinations of the application, database, and system to a known desired state.
Special Considerations:	Resources from the Systems (or Computer Operations), Database are required. These tests should be run after hours or on an isolated machine

3.1.8. Configuration Testing

Configuration testing verifies the test-target operates correctly under different software configurations and interacting with different software. It is the process of testing the system with each one of the supported software and hardware configurations. The Execution area supports configuration testing by allowing reuse of the created tests.

Technique Objective:	● Verify the test-target functions correctly on different platforms and under different configurations.
Technique:	● Normal test functions will be used to test the system with different configurations. ● Unit Tests as well as integration test and database access tests will be executed in different configurations. ● When configuring the system, the test configuration will be created to cover the maximum possible scope.
Oracles:	
Required Tools:	Robot Framework
Success Criteria:	● The test-target behaves as expected and the non test-target software also behaves as expected.
Special Considerations:	

4. Deliverables

4.1 Test Evaluation Summaries

The Test Evaluation Summary collects, organizes, and presents the Test Results and key measures of test to enable objective quality evaluation and assessment. The Test Evaluation Summary also presents an interim evaluation from the test, indicating the assessment of the software. The test evaluation summaries are realized in form of an internet survey.

I am going to submit below documents after finished the testing phase

- Robot Framework
- Test plan
- Develop Test cases
- Test Case Review
- Complete Defect Reports

4.2 Reporting on Test Coverage

Incident log entries will be made for all bugs found during testing. The log will be used to track the status of the bugs.

5. Risks, Dependencies, Assumptions, and Constraints

Risk	Mitigation Strategy	Contingency (Risk is realized)
All the functionalities may not be able to be checked by the automated testing.		Refractor and re-architect the design in order to enable easy mocking of components where possible in order to automate testing.
Desktop applications are tested for some OS and PCs, not for all, so there is possibility that applications won't work on some of them		Check application on many platforms as much as possible.work with widely used API's.
Database requires refresh.	System Admin will ensure the Database is regularly refreshed as required.	Restore data and restart Clear Database