Project Title: System Verification and Validation Plan for Greenway

Author Name

November 2, 2022

1 Revision History

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

Contents

1 Revision History **i**

2 Symbols, Abbreviations and Acronyms **iv**

3 General Information **1**
- 3.1 Summary . . . 1
- 3.2 Objectives . . . 1
- 3.3 Relevant Documentation . . . 1

4 Plan **2**
- 4.1 Verification and Validation Team . . . 2
- 4.2 SRS Verification Plan . . . 2
- 4.3 Design Verification Plan . . . 3
- 4.4 Implementation Verification Plan . . . 3
- 4.5 Automated Testing and Verification Tools . . . 3
- 4.6 Software Validation Plan . . . 4

5 System Test Description **4**
- 5.1 Tests for Functional Requirements . . . 4
  - 5.1.1 Start Screen . . . 5
  - 5.1.2 Map Interactions . . . 7
  - 5.1.3 Backend Processing . . . 8
- 5.2 Tests for Nonfunctional Requirements . . . 10
  - 5.2.1 Interface Interactions . . . 10
  - 5.2.2 Performance Tests . . . 11
- 5.3 Traceability Between Test Cases and Requirements . . . 12

6 Appendix **14**
- 6.1 Symbolic Parameters . . . 14
- 6.2 Usability Survey Questions . . . 14

List of Tables

[Remove this section if it isn't needed —SS]

List of Figures

[Remove this section if it isn't needed —SS]

2 Symbols, Abbreviations and Acronyms

symbol	description
T	Test

[symbols, abbreviations or acronyms – you can simply reference the SRS (Author, 2019) tables, if appropriate —SS]

This document ... [provide an introductory blurb and roadmap of the Verification and Validation plan —SS]

3 General Information

3.1 Summary

The goal of Greenway is to create an application that allows users to provide car information and a destination, in return the application supplies the amount of money it will take for the user to reach that destination using the most fuel-efficient route calculated by the app. The application uses real-time gas price data, information on mileage data, and terrain data to calculate and provide the amount of money it would take to reach a destination for users who drive many different types of gas vehicles; this allows them to use this app in many ways, such as, figuring out which gym may provide the most bang for their buck based on distance from their location.

3.2 Objectives

The objective of the document is to test the various subsystems identified in the system requirements and the systems design documents to ensure correctness and demonstrate adequate usability. Often times software may have bugs that is experienced by the end-user. This document will try to mitigate those issues by ensuring the underlying logic for the subsystems are performing as expected. This will be done through a suite of Unit Tests to test the functional and non-functional requirements. The code tested will be the underlying logic functions that interact with the database and send information back to the client since the rest of the system is front-end UI aspects.

3.3 Relevant Documentation

The relevant documents include:

- Software Requirment Specification Document
- Systems Design Document
- Hazard Analysis Document

4 Plan

This section includes the following sections, "Verification and Validation Team", "SRS Verification Plan", "Design Verification Plan", "Implementation Verfication Plan", "Automated Testing and Verification Tools", "Software Validation Plan".

4.1 Verification and Validation Team

The following project members are responsible for all procedures of the verification and validation processes including writing and executing tests:

- Utsharga Rozario
- Jash Mehta
- Priyansh Shah
- Bilal Shaikh
- Sharjil Mohsin
- Pranay Kotian

4.2 SRS Verification Plan

The following plans indicate what our team intends to do for SRS verification:

- **Review by teammates:** This plan involves going through each SRS and verifying whether each SRS is still valid within our given scope.
- **Review by stakeholders:** This plan involves going through each SRS with our stakeholder and how they view them in perspective to your usage.
- **Task based inspection:** This plan involves going through each SRS as a task, generally done by a teammate or stakeholder.
- **Checklist:** This plan involves verifying if each SRS has been met through our previously set checklist in our SRS document.

4.3 Design Verification Plan

The following plans indicate what our team intends to do for Design verification:

- **Review by teammates:** This plan involves going through the design of the project with our teammates through a high-fidelity prototype or a functional prototype, and verifying if the design is meets the expectation stated in the SRS.
- **Review by stakeholders:** This plan involves going through the design of the project with our stakeholders through a high-fidelity prototype or a functional prototype, and verifying if the design is meets the expectation stated in the SRS.
- **Task based inspection:** This plan involves going through each feature as a task in a high-fidelity prototype or a functional prototype, generally done by a teammate or stakeholder.
- **Checklist:** This plan involves verifying if each design requirements have been met through our previously set checklist in our SRS document.

4.4 Implementation Verification Plan

The following plans indicate what our team intends to do for Implementation verification:

- **Code Inspection:** This will be used for the test plans in section 5.1.1.
- **Static Analysis:** This will be used for the test plans in section 5.1.2 and section 5.1.3.
- **Non-functional Testing:** Non-functional Requirements test plans are written in details in section 5.2.

4.5 Automated Testing and Verification Tools

The following Automated Testing Tools:

- **Cypress:** Cypress is an automated testing tool from front-end development and has ability to take screenshots and video recordings of tests, allows developers to do asynchronous testing, and has In-browser debugging environment. Hence will be used to perform front-end testing on React.

- **Mocha:** Mocha is one of the oldest and most well-known testing frameworks for Node.js and hence will be used for our project. It has also evolved with Node.js and the JavaScript language over the years, such as supporting callbacks, promises, and async/await which we intend to use in our back-end.

- **Mongo Orchestration:** Mongo Orchestration is an HTTP server providing a RESTful interface to MongoDB process management running on the same machine and will be used to test our MongoDB database.

The following Verification Tools:

- **EsLint:** ESLint is a tool for identifying and reporting on patterns found in ECMAScript/JavaScript code, with the goal of making code more consistent and avoiding bugs and hence will used for linting our front-end (React) and back-end (Node.js).

4.6 Software Validation Plan

There are no available external data that can help validate the software. Hence, there are no plans for validation for now.

5 System Test Description

5.1 Tests for Functional Requirements

The areas of testing listed below are the following: Start screen, Map Interactions, Backend processing, Results Display. Start screen covers functional requirements 1-5 from the SRS as all those requirements are related to the inputs and functionality that exists in the program before any processing or output is shown. Map Interaction covers functional requirements 6-8 as these requirements are concerned with functionality related to the map on

display. Backend processing covers functional requirements 9-12 as these requirements are concerned with calculations and, external data collection and processing alike.

5.1.1 Start Screen

Start screen covers functional requirements 1-5 from the SRS as all those requirements are related to the inputs and functionality that exists in the program before any processing or output is shown

Preliminary Information Tests

1. preliminary-information-test-1

   Control: Automatic

   Initial State: No input in the start screen

   Input: Start Location

   Output: Status Message stating if the Location was accepted

   Test Case Derivation: The status message should be accepted the location as FR1 in the SRS requires the system to be able to take that as an input.

   How test will be performed: The testing framework will use a valid start location to input it into the field and ensure the status message is accepted the location.

2. preliminary-information-test-2

   Control: Automatic

   Initial State: No input in the start screen

   Input: Location of the Destination

   Output: Status Message stating if the Location was accepted

   Test Case Derivation: The status message should be accepted the location as FR2 in the SRS requires the system to be able to take that as an input.

How test will be performed: The testing framework will use a valid location for a destination to input it into the field and ensure the status message is accepted the location.

3. preliminary-information-test-3

   Control: Automatic

   Initial State: No input in the start screen

   Input: Car Details

   Output: Status Message stating if the Car Information was accepted

   Test Case Derivation: The status message should be accepted the Car Information as FR3 in the SRS requires the system to be able to take that as an input.

   How test will be performed: The testing framework will use valid Car details for details to input it into the field and ensure the status message is accepted the Car Information.

4. preliminary-information-test-4

   Control: Automatic

   Initial State: Accepted Car details in the start screen

   Input: Car mileage/fuel economy information

   Output: Status Message stating if the Car mileage/fuel economy information was accepted

   Test Case Derivation: The status message should be accepted the Car mileage/fuel economy information as FR4 in the SRS requires the system to be able to take that as an input.

   How test will be performed: The testing framework will use valid Car mileage/fuel economy information for details to input it into the field and ensure the status message is accepted the Car mileage/fuel economy information.

5. preliminary-information-test-5

Control: Automatic

Initial State: Accepted Car details in the start screen

Input: Car information

Output: Status Message stating if the Car information was updated

Test Case Derivation: The status message should be accepted the Car information as FR5 in the SRS requires the system to be able to take that as an input.

How test will be performed: The testing framework will use valid Car information for details to input it into the field and ensure the status message is updated the Car information.

5.1.2 Map Interactions

Map Interaction covers functional requirements 6-8 as these requirements are concerned with functionality related to the map on display.

Map Tests

1. map-test-1

   Control: Manual

   Initial State: Start Screen finished

   Input: None

   Output: If the Map displays a route from start to end based on start screen input.

   Test Case Derivation: The Map should display a route from start to end as specified in the Start screen, as FR6 States.

   How test will be performed: The tester will ensure that the map has the correct route showing on it for a given start and end destination with a correct route to compare with.

2. map-test-2

   Control: Manual

Initial State: Start Screen finished

Input: None

Output: If the Map displays all gas stations that it can possibly encounter along the route from a start to end destination.

Test Case Derivation: The Map should display all gas stations along the route from start to end as specified in the Start screen, as FR7 States.

How test will be performed: The tester will ensure that the map has all the gas stations along the route it shows and that they are being correctly displayed with a map to reference.

3. map-test-3

   Control: Manual

   Initial State: Start Screen finished

   Input: None

   Output: If correct gas prices come up when gas stations along the displayed route are clicked upon.

   Test Case Derivation: The Map should display all gas prices in the gas stations along the route from start to end as specified in the Start screen, as FR8 States.

   How test will be performed: The tester will click on all gas stations along the route and check that they are displaying gas prices in a range given beforehand to the tester.

5.1.3 Backend Processing

Backend processing covers functional requirements 9-12 as these requirements are concerned with calculations and, external data collection and processing alike.

Backend Tests

1. backend-test-1

Control: Automatic

Initial State: Start Screen finished

Input: Route Details

Output: If the route returned to the user is one that is optimizing the distance and elevation perfectly to reduce fuel costs.

Test Case Derivation: The route displayed to the user should be the most fuel efficient route, as FR9 States.

How test will be performed: The testing framework will have correct nodes pertaining to the most fuel efficient route and will compare them with nodes outputted by the back end to ensure that they are correct.

2. backend-test-2

   Control: Automatic

   Initial State: Start Screen finished

   Input: Route Details

   Output: If the database call on the backend returns correct elevation data for a route.

   Test Case Derivation: The most fuel efficient needs accurate elevation data and as such the FR10 is derived from the needs of FR9. And the test is designed to ensure that FR10 is fulfilled in the design.

   How test will be performed: The testing framework will test that the backend is getting the correct elevation data from the database with a reference dataset with correct elevation data.

3. backend-test-3

   Control: Automatic

   Initial State: Start Screen finished

   Input: Route Details

   Output: If correct fuel consumption is given by backend for different parts of the route which have different elevation metrics.

Test Case Derivation: The application should be able to calculate accurate fuel consumption in different elevation conditions as distance is not the only metric for determining the most fuel efficient route. This test is made for FR11.

How test will be performed: The testing framework will test that the backend is calculating the correct fuel consumption data from the database with a reference dataset with correct fuel consumption data.

4. backend-test-4

   Control: Automatic

   Initial State: Start Screen finished

   Input: Route Details

   Output: If correct total cost is calculated by the backend for a certain route with a start and end decisions as the route details.

   Test Case Derivation: The application should be able to calculate accurate total cost for a certain route, as FR12 states.

   How test will be performed: The testing framework will test that the backend is to ensure that the a correct total cost is being returned with a correct total cost for a route as reference.

5.2 Tests for Nonfunctional Requirements

5.2.1 Interface Interactions

Usability Tests

1. usability-1

   Type: Functional, Manual

   Initial State: N/A

   Input/Condition: N/A

   Output/Result: N/A

   Test Case Derivation: As NFR1 states, the system must have a modern, minimalist user interface.

How test will be performed: This test will be verified using responses to Usability Survey Question 1

2. usability-2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

5.2.2 Performance Tests

...

5.3 Traceability Between Test Cases and Requirements

Requirement #	**Description**	**Test ID(s)**
FR1	The system shall allow the user a way to input a start location.	preliminary-information-test-1
FR2	The system shall allow the user a way to input a final destination.	preliminary-information-test-2
FR3	The system shall allow the user a way to select any car model and make.	preliminary-information-test-3
FR4	The system shall allow the user a way to input the fuel economy of their car.	preliminary-information-test-4
FR5	The system shall have a way to update available car models from either user input or external sources.	preliminary-information-test-5
FR6	The system shall have a map to show journey from start to final destination.	map-test-1
FR7	The system shall display the gas stations along the route.	map-test-2
FR8	The system shall display gas prices at stations on route to destination.	map-test-3
FR9	The system shall have a way to calculate the most fuel efficient route.	backend-test-1
FR10	The system shall have a way to collect elevation data along the suggested route.	backend-test-2
FR11	The system shall have a way to calculate fuel consumption on different terrain elevations.	backend-test-3
FR12	The system must calculate the total cost of the journey.	backend-test-4

References

Author Author. System requirements specification. https://github.com/..., 2019.

6 Appendix

6.1 Symbolic Parameters

The definition of the test cases will call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

6.2 Usability Survey Questions

1. On a scale of 1 - 10, how aesthetically pleasing is the user interface?
2. Does the user interface have a consistent graphical style throughout the system?
3. On a scale of 1 - 10, how easy to navigate is the user interface?
4. Does the user interface feel minimalist?
5. Does the system run on your favorite web browser?

Appendix — Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Lifelong Learning. Please answer the following questions:

1.

2.