Team Scrumbags Test Plan (Test Plan ID # 1):

Members: Ryan Solanki, Reed Wilson

Purpose of Test Plan

To ensure the College Touring Project runs successfully and meets all of the project requirements. Throughout our project, we will be testing individual components and the program as a whole to check for errors as the program develops. This will include the SQL database connection, making sure users can easily take either of the three paths, and successful communication between each window and the database class.

Scope of Test Plan

We will be testing various components of the program throughout our timeline such as:

- Being able to view the college campuses and their distance from Saddleback College as a student.
    - Giving the student the option of the campus to start on
    - Plan the shortest trip
    - Create a custom trip
- Outputting the names and prices of souvenirs from each campus to the student.
- Being able to add/change souvenirs from each school and their corresponding prices as an admin

Test Environment

The environments we will use include…

1) Qt Creator
2) SQLite Database and DB Browser

3) GitHub -- Collaborate on the project and for version control
4) Clubhouse -- Agile Management Tool

Overall Test Strategy

We will use Unit, Integration, White Box and Black Box Testing. We will use *Unit Testing* to test our SQLITE database, classes, functions, and that requirements of each story are being met and are working properly. Also, we may utilize test case C++ files within our project in order to help us isolate and test each unit as we develop. (Note: Unit tests do not test the interactions between these modules).

We will use *Integration Testing* to verify that the interactions between units are working properly. We will do these tests after each sprint to make sure that the implementations of each story are working together with the others. Optimally we will do enough *Unit Testing* that the *Integration Testing* is less consequential and time consuming.

We will use *White Box Testing* to verify the internal structures of the program and to test each possible input, verifying that each path from the input has the desired output. This testing will be utilized in both *Unit Testing* and *Integration Testing.*

We will use *Black Box Testing* to verify the source code requirements. This strategy is about testing the overall functionality of the project without being able to look at its internal structures. Again, testing will also be utilized in both *Unit Testing* and *Integration Testing.*

Test Plan Deliverables

Each story and sprint will have an accompanying test to verify completion. These tests will mainly be implemented in test cases as separate C++ files.

Features Tested From User's Perspective

- Viewing the college campuses and their distance from Saddleback College or UCI.
- Planning the shortest trip across all the campuses while starting at Saddleback College, UCI, or ASU.
- Create a custom trip and view the shortest way to travel to all the colleges selected on the trip.
- Seeing the total distance traveled throughout the trip.
- Being able to view and purchase the traditional souvenirs from each school.
- Buttons/Widget functionality

Features NOT tested from User's Perspective

- Being able to add new college campuses
- Being able to add/change the traditional souvenirs and their corresponding prices.
- Being able to populate the database with a set amount of schools, including their name and distances from Saddleback College.
- Linking of the SQlite database with QT

Roles and Responsibilities:

- **Scrum Master:** The leader of a Scrum team and is responsible for facilitating communication among team members and ensuring that all agile practices are followed by team members. The Scrum Master also keeps track of the Scrum Log.
- **Product Owner:** The person who works with all the team members to determine what features will be used in the product release. The Product Owner also keeps track of the backlog.
- **Tester:** The person who tests code through black box and white box. If errors are found, the tester will report bugs to the product owner and developer to ensure that the problem is resolved.
- **Developer:** Anyone on the team who helps to develop the code.

Environment Description:

- MacBook Pro with macOS Big Sur
- Windows 10 Devices
- Qt 5.15.1 clang 64 bit

Documents that Support the Test Plan:

UML diagrams, QT documentation, SQLite documentation

Configuration Management (GITHUB):

Each team member will push code changes to their own individual branch. This code will be tested using said methods before we merge it onto the master branch. This use of branches will also enable us to catch and fix merge conflicts much more easily.

Entry Criteria:

- Testers need to be proficient in QT, C++, and SQL for all whitebox testing.
- For blackbox testing the Tester need only be aware of the functionality requirements of the application itself.

Exit Criteria:

- When the program can successfully run with no bugs and all objectives/testing strategies have been completed.

Suspension Criteria:

- When a bug has been encountered causing the program to act differently than previously expected.

Approval Process

- All team members will agree on the direction that the program is being taken. If agreement issues arise, we will contact Professor Lebowski to decide on the best path to be taken.

Glossary of Terms

- **Scrum Master:** Person in charge of keeping track of the Scrum Log
- **Product Owner:** Person in charge of keeping track of the backlog
- **Developer:** Person who develops the code for the project.

- **Tester:** Person who employs white and black box testing to ensure that the code performs optimally without errors.
- **Github:** An online repository used for version control and collaboration.
- **SQlite:** A database software used to manage local/client storage in applications.
- **Qt Creator:** An Integrated Development Platform (IDE) used for the development in which GUI's are necessary.
- **Git:** The most commonly used version control system that allows you to track changes made to files.