Wesley Chok

Matt Jang

Daniel Olaes

Mohamed Soliman

Big O(mega) G's Test Plan

Test Plan ID: 2

Purpose of Test Plan: The purpose of the test plan is to design a GUI application using the QT software development tool to allow a baseball fan to create a Baseball Fan Stadium Dream Trip. The testing will check that there are no error present throughout the software process, ensuring that the program runs successfully, meets all requirements, and is suitable for delivery.

Scope of the Test Plan: We will be testing the program requirements and agile stories for the Baseball Fan Trip and admin privileges.

- Based on the admin's display option, validate the legitimacy of the information displayed corresponding to the information held in the database and the admin's corresponding choice.
- Based on the baseball fan's trip selection, validate that the proper trip is taken corresponding to the fan's choice.
  - Dodgers trip: Dodger Stadium is the starting location and the fan can select 1 other available trip.
  - Start + All Others trip: Allow the fan to select their starting stadium, then visits all other stadiums.
  - Marlins trip: The fan will be given a trip starting at Marlins Park and then visits all other stadiums.

        - Custom: The fan will be able to create their own dream vacation, allowing them to choose their starting stadium and as many other stadiums as they desire.
            - For all trips, the program will sort the fan's trip using the Dijkstra's algorithm and a recursive sort function.
- Based on the baseball fan's trip selection, validate that the fan will be able to visit the souvenir shop of the corresponding stadiums selected, and purchase as many souvenirs as desired.
- Based on the baseball fan's traversal selection (MST, BFS, DFS), validate that the traversal and information correspond to the fan's choice.
- From the admin's perspective, validate that the admin is able to read in new stadiums, and edit stadium information and souvenir information.

<u>Overall Test Strategy</u>:

- When a story is completed, the team member should present the stories to the Product Owner to receive approval for the completion of the task.
- If an error is encounter by the Product Owner or another team member, the code should be sent back to the developer for debugging until the error has been corrected.
- Each developer must follow the definitions of done for the agile stories, and error check the software prior to presenting to the Product Owner.
- Unit testing should be utilized to test singular units of the program, ignoring all collaborations with other units.
- White box testing will be utilized to test the internal structure of the application and derive inputs to test the program's algorithms and functionality.

- Black box testing will be utilized to ensure that the application is functional from a user's perspective, rather than peering into the application's code.

<u>What features will be tested from a user's perspective</u>:

- The correct retrieval of information from the SQL database to the application display tables.
- The correct responses and calculations corresponding to the user's selection for traversals and trips. i.e. total distances and the calculated shortest paths.
- The sending and receiving of information from the database.
    - The addition of new stadiums
    - The addition and deletion of souvenirs
    - Edits on the MLB team information
- The contingency handling of the application. i.e. valid inputs and their corresponding outputs, and invalid inputs and the way the program handles these contingencies.

<u>What features will not be tested from the user's perspective</u>:

- The code that connect the QT application to the SQL database.
- The code that connects the widgets throughout the application.
- The code that allows the user to read in the information from a .txt file into the SQL database.
- The SQL query function that allow the program to receive and send information to and from the database.
- The graph used to hold the vertices and edges for the stadiums.
- The algorithms (Dijkstra's algorithm) used to recursively calculate the most efficient trip plan and distance.

- The algorithms for the DFS traversal, BFS traversal, and MST.

Entry Criteria:

- For unit testing, each developer must have their code written so it functional without relying on parts of the code. i.e. classes and their methods should be functional as singular units.
- For white box testing, the developers should have multiple inputs and paths that require testing that have the potential to cause errors.
- For black box testing, every developer must have ensured that their code is easily combinable other developer's code. Next, the entire team needs to test the code with the business requirement for the application to ensure that it is deliverable.

Exit Criteria:

- When all possible bugs are handled, and the application outputs the correct output to their corresponding input.
- All requirements are met and meet the expectation of the product owner.

Suspension Criteria:

- If a bug or a potential source of error appear, the code will be sent back to the developer for correction and testing.
- If a requirement is missing, a developer will be need to create and test code to fulfill this requirement.

Roles and Requirements:

- Product Owner: as the head of the team, the product owner is in charge of approving the product according requirements and customer's priorities.

- Scrum Master: Responsible for making sure the team adhere to Scum protocol. Removes barriers for the team and ensure that the team is working at peak performance.
- Team Members: Develop code adhering to the demand of the product owner and the application requirement. Need to be respectful and work efficient to ensure no problems occur during the development process.

Approval Process:

- When the definition of done is meet by the developers, and the product owner has approved the application for delivery.
- White box testing will confirm the integrity of the internal code, and black box testing will ensure that all requirement for the application are met.

Schedule:

- Initially, a developer will utilize unit testing during the process of developing a single story, ensuring that it function independently.
- Next, the developer should do thorough white box testing to ensure the integrity of the code. Checking for contingency handling.
- Finally, the developer should black box test to ensure that the application meets all the requirements.

Necessary Training Needed for Testing:

- Each developer should have some familiarity with using agile management software such as GitHub.
- Advanced knowledge of QT, C++, and SQL isn't necessarily required, but each developer must be proficient and willing to learn more if necessary.

Environmental Description:

- Hardware: Laptop/ Desktop with internet connection is required for coding, sharing code, and meeting.
- Software: QT Creator, SQLITE database browser, GitHub and Discord/Zoom allowing the group to work from home and meet online during quarantine.

Configuration Management:

- When a story is completed, the developer should upload the finished code to their own branch on the GitHub.
- The product owner will test the code with the developer for bug and requirements. If errors are found, the developer must debug the code and reupload. If the code is approved, it can be merged with other completed pieces of code.
- Once the code is merged, white box and black box testing is required to ensure that the code is still operational after the merge. If the code fails, the team need to locate the problem and correct where necessary.
- Once all the code has been merged successfully, it can be merged to the master branch for delivery.

Document that Support the Test Plan:

- UML diagrams
- QT class reference pages
- SQL command reference pages
- Agile stories.

Test Deliverables:

- UML diagrams

- Agile stories
- Team coding standards
- Team rules
- Doxygen documentation
- Project test plan
- Scrum log
- Product backlog

Glossary of Terms:

- Unit testing: A testing process that involves testing a single piece of a program, such as a method or a class, ignoring the other piece of the program.
- White box testing: A method of testing software that tests internal structures or workings of an application. Used to design test cases to exercise paths through the code and determine the appropriate output.
- Black box testing: a method of testing software that examines the functionality of an application without peering into the internal structure of the code.