*<Take me out to the ballgame>*

test plan

Product Owner: Oscar Lopez

Scrum Master: Anthony Karapetian

Developers: Luke Schrezenmeier, Yuchen Yao, Alek Peratoner

*<04/20/2019>*

**TABLE OF CONTENTS**

1. **INTRODUCTION…………………………………………………………....… 3**
   1. **Purpose of the test plan document……………………………………....… 3**
   2. **Overall Test Strategy…………………………………………………..…....3**
2. **TEST ITEM………………………..………………………………………..….. 5**
   1. **Project description……………..……………………………………….…. 5**
   2. **Entry Criteria…………………..………………………………………….. 5**
   3. **Exit Criteria…………………..………………………………...………….. 6**
   4. **Item to be test from a baseball fan’s perspective.…………….………,.…6**

**2.5 Item to be test NOT from a baseball fan’s perspective……………..……7**

**2.6 Test Approach(s)………………………………………………………...… 9**

**2.7 Test Pass / Fail Criteria………………………………………………....... 10**

**2.8 Test Deliverables……………..………………………………………….…11**

**2.9 Test Suspension Criteria ……………………….………………...…..……12**

**2.10 Test Environments and Training Needs..…….…….……………………12**

1. **Roles and assigned responsibilities………………………………….…………13**
2. **Version control(GIT) philosophy…………………..……………….….……....14**
3. **AGILE STORY APPROVAL……………………….………..……..…………15**
4. **GLOSSARY OF TERMS………………………………………..…..…………16**
5. **INTRODUCTION**
   1. **Purpose of the test plan document**

This test plan describes the approach our team will take when testing the program, including the methods we will utilize for testing our program. It is our goal to ensure that each and every aspect of the project has been tested and officially checked off by the Product Owner, and that the segments meet each and every requirement provided by both baseball fans and admin of MLB. Test plan also provide extensive testing to verify that the baseball fan user interface, file parsing software, as well as admin of MLB privileges perform quickly and efficiently within program. The test plan will also aid in error reporting and hopefully reduce incoming errors in future sprints.

* 1. **Overall Test Strategy**

**White Box Testing:**

Used to check the actual code and its implementation.

Making sure the information of survivor, including its name and price added to a list correctly.

Making sure the information of stadium is modified correctly.

Basically, making sure everything that is written does what it needs to do correctly and the outcome is what the developer expected.

Generally, will be done during the programming time.

**Black Box Testing:**

Testing the program as a whole without knowing what the code may look like

Test the program from the point of view of the baseball fan or admin and be able to see what can be improved with the interface and any flaws that may have not been caught during white box testing.

Making sure all the sorted information is what the baseball fans want the way to sort and the baseball team and stadiums are corresponding.

Generally, will be done after the program/part of the program is finished.

**Story Testing:**

Some of the testing will be done from a baseball perspective. As in, can the program sort the information by what baseball fan wants, start the trip from baseball fan choose, choose the teams or stadiums baseball fan wants and gets the shortest distance of the trip.

Some of the testing will be done from admin of MLB. As in, the program add/remove survivors for the stadium and modify the information of a stadium/ survivors.

Some of the testing will be done from a developing perspective. As in, can the baseball teams/stadiums information be read from a file and displayed correctly, can the BFS/ DFS displays correctly and the minimum spanning tree(MST) displays well.

**Unit Testing:**

Unit testing will be used throughout the development of the program

It will be broken down into several major parts.

One will be reading the information of the baseball teams and stadiums from a file and testing to make sure that is correct

One will be displaying the sorted information correctly.

One will be creating a list of stadiums and its corresponding teams that the baseball fan wants to visit and it will be in the most efficient order possible

And finally, one will be making sure results, such as, total distance/ grand total of costs, are correct.

Once we have more than one units completed, we will also start integration testing to make sure the units work independently and together.

Desk checking will not be necessary since most of the code will have been put through the debugger and tested multiple times before being put together.

All of the testing will be manual and in the hands of the team members.

After we coding the program, we plan to use **White Box Testing** by using the **Unit Test** with the **Test Framework** of **Google Test**.

1. **TEST ITEM**
   1. **Project description**

The program we developed is mainly aimed at Major League Baseball. Through our program, baseball fans can get the information they want and search information for the convenience of baseball fans. Most of the information has been processed, for example by sorting the information in different directions, such as, baseball teams, stadium names, park typology and so on. Baseball fans can also choose a trip that has been arranged or choose the stadium they want to visit and the corresponding teams. Our program will generate the shortest distance and provides the shortest distance for baseball fans. Baseball fans can also buy souvenirs and our program will figure out the cost of each course and the total cost of all visits to the course. Our program also provides admin of MLB with the information of modifying courses or souvenirs. Admin can add or delete souvenirs through our program.

* 1. **Entry Criteria**

If the users want to search some sorted information or plan a trip, first they must login in as a baseball fans.

If the users want to implement admin functions, they must first login correctly as an admin.

There must be a UI system set up for users to interact with.

* 1. **Exit Criteria**

Once baseball fans get the sorted information that they want to get and wants to start the trip.

Once baseball fans choose their first stadium to visit, finish the trip plan and get the shortest distance of the trip.

Once baseball fans finish purchasing the survivors and get the ground total.

Once the admin of MLB can modify the information of the stadium.

Once the admin of MLB can create new souvenirs for the new stadium.

Once the admin of MLB can delete souvenirs for the stadium.

Once the admin of MLB can modify the price of the souvenir.

* 1. **Item to be test from a baseball fan’s perspective**

|  |  |  |  |
| --- | --- | --- | --- |
| **Item to Test** | **Test Description** | **Test Date** | **Responsibility** |
| Menu | Check whether the information of the stadium and its corresponding team will be sorted as the baseball fans wants.  Check the menu displays clearly  Check the connection between each menu.  All the test is using the **Black Box Testing and Story Testing**. | 04/22/2019 | Luke  Alek |
| Trip plan | Test whether every stadium allows baseball fans to choose to be the start of the visit they want. Baseball fans can then choose the stadium they want to visit.  All the test is using the **Black Box Testing**. | 04/22/2019 | Alek  Oscar |
| Price | Calculate the total price of the baseball fans purchased and the grand total of costs. Check the price change after the administrator modify the data. Consider the price as a unit to use **Unit Test of Google Test** | 04/24/2019 | Yuchen  Anthony |
| Login | Enter the user name and password which is valid and distinguish between admin and baseball fan logins. Using **Black Box Testing and** **Google test in the Qt** and check whether it success. | 04/25/2019 | Oscar  Yuchen |

* 1. **Item to be test NOT from a baseball fan’s perspective**

|  |  |  |  |
| --- | --- | --- | --- |
| **Item not to Test** | **Test Description** | **Test Date** | **Responsibility** |
| Database | We plan to test by **Black Box Testing** to make sure the maximum of the database to make sure the database will not be overloaded. | 04/20/2019 | Oscar  Anthony |
| Agile story | Agile stories are designed to make programmers more familiar with and complete programs on schedule. | 04/16/2019 | Luke  Alek  Anthony  Oscar  Yuchen |
| Modify data | The information can be modified once admin logs in. Information about the stadiums, for example. Admin can also delete or add souvenirs. After the admin operation is completed, use **Black Box Testing** and **Story Testing** to confirm that the information displayed is consistent with the operation of admin. | 04/23/2019 | Anthony  Oscar |
| BFS | Check Breadth-first search (BFS), which is an algorithm for traversing or searching tree or graph data structures. It starts at the tree root and explores all of the neighbor nodes at the present depth prior to moving on to the nodes at the next depth level.  Verify that BFS is displayed correctly and the corresponding minimum distance is obtained.  We will use the **Black Box Testing** and **Story Testing** to start with Coors Field. | 04/23/2019 | Luke  Yuchen |
| DFS | Depth-first search (DFS) is an algorithm for traversing or searching tree or graph data structures. The algorithm starts at the root node and explores as far as possible along each branch before backtracking.  Verify that DFS is displayed correctly and the corresponding minimum distance is obtained.  We will use the **Black Box Testing** and **Story Testing** to start with Fenway Park. | 04/23/2019 | Luke  Anthony |
| MST | Minimum spanning tree (MST) is a subset of the edges of a connected, edge-weighted undirected graph that connects all the vertices together, without any cycles and with the minimum possible total edge weight, a spanning tree whose sum of edge weights is as small as possible. More generally, any edge-weighted undirected graph has a minimum spanning forest, which is a union of the minimum spanning trees for its connected components.  Verify that MST is displayed correctly and using the Prim’s or Kruskal’s algorithm and get the corresponding minimum distance is obtained.  Using the **White Box Testing** to test the total distance. | 04/25/2019 | Luke  Oscar |

* 1. **Test Approach(s)**

All the information about stadiums and baseball teams are sorted and displays well on the screen.

Baseball fans can know stadium seating capacity.

The greatest/smallest distance to center field that the stadium(s) and corresponding team name(s) can be clearly seen by a baseball fan.

Baseball fans can choose to start at the Angel Stadium (Los Angeles Angels) and get the shortest distance.

Baseball fans can buy the souvenirs they want and the program will display the total amount spent at each stadium and a grand total for all stadiums visited.

Admin can modify the information of the stadium after logging in.

Admin can create new souvenirs for the new stadium after login.

Admin can log in and then delete souvenirs for the stadium.

Admin can login and modify the price of the souvenir.

* 1. **Test Pass / Fail Criteria**

If the menu connects well and can easily return to the main menu, the test passes; otherwise it fails.

If the list of information displays organized and sorted well, the test passes; otherwise it fails.

If the admin using the valid username and password to log in, the login page works, the test passes; otherwise it fails.

If the data changed by the admin and displayed organized and correct, the test passes; otherwise its fails.

If the programmer enters the data that the database cannot hold, the database is overloaded and the program will crash, then the test fails. Enter at least 100 databases to make sure that the program can work well and will crash.

If the baseball fans can enter his or her name to purchase items and cannot modify any information of items, the test passes; otherwise it fails.

If the result of total price is the same using the math method, the test passes; otherwise it fails.

If the baseball fans can choose to start at Comerica Park (Detroit Tigers) or Angel Stadium (Los Angeles Angels) and get its corresponding total distance correctly, the test passes; otherwise it fails.

If the DFS & BFS displays correct and clearly, also the associated mileage is correct, the test passes; otherwise it fails.

* 1. **Test Deliverables**

Our testing deliverables will include the correct, error-free program; this test plan; and our team’s agile stories. The test will be done on the program and will using the screenshots to compare the data. The test will follow as a store manager and an administrator to check the result of the display and the connection of each menu. We will test all the requirements and meet the requirements.

* 1. **Test Suspension Criteria**

Testing will be suspended when the segment of code being tested has failed to meet any of the requirements laid forth for it, when it has provided the wrong output, or when it has caused the entire program to crash. At this point, the segment will be returned to the code’s author for the appropriate corrections before it may be tested again. When the program runs error or cannot display at all the test will suspend. As a programmer, we need to enter at least 100 databases to make sure that the program will not crash. The test will suspend once the database is overloaded.

* 1. **Test Environments and Training Needs**

The Test environment includes a windows operating system (8 or newer) as well as the current version of Qt (5.11). Our UML diagrams as well as machine state diagrams will help support and develop tests for this project. All the team members should be familiar with C++ Language and how to design QT GUI. Also, the team will need a place to constantly have meeting to discuss progress of the project. The project owner should consider all the test and evaluate whether the test is necessary and sign for the useful test. All the members will have different units for the test.

1. **Roles and assigned responsibilities**

|  |  |
| --- | --- |
| **Role** | **Responsibility** |
| **Scrum master:**  Anthony Karapetian | Keeps track of the meetings and overall program. Helps with testing by ensuring the team carries out the vital steps in the testing process. Ensuring the team lives agile values and principles and follows the processes and practices that the team agreed they would use. Everything from coaching team about agile scrum to delivering successful deliverables. Job description of scrum master is challenging as it’s the single most hectic job if things do not go well or deadlines couldn’t be kept. He also publishes tasks on **Waffle** so that everyone can clearly understand what and when we need to do. |
| **Project owner:**  Oscar Lopez | Oversees the backlog and ensures there are no problems with the code. Responsible for successful delivery of the project. The role requires effective coordination of the projects and their inter-dependencies, including other resources, and any risks and other issues that may arise. He the person who most knows the project and have well communication with project members. He will also explain our project to the third person, such as professor. |
| **Project members:**  Luke Schrezenmeier  Yuchen Yao  Alek Peratoner | Help with testing because they are the ones who are primarily documenting what is wrong with the code, and why it is not performing as it should. They will work together to determine how best to solve the problem.  Design the project according to the agile story.  Project members are the individuals who actively work on one or more phases of the project. Sometime, project members work together and both focus on the same problem of the project. |
| **Tester:**  Luke Schrezenmeier  Alek Peratoner  Oscar Lopez  Yuchen Yao  Anthony Karapetian | The tester tests a computer program that has completed all or part of its functions and modules before formal use to ensure that the program runs correctly in a predetermined manner. The correctness of software has not been fundamentally solved, software testing is still the main means to find software errors and defects. In order to detect errors in the system, you should strive to design test cases that expose errors. Because of the limited number of people on the team, each of us will act as testers of the team and test different parts. To avoid test limitations, we will arrange for two testers to test the third team member's part at the same time to ensure that the test part runs well. That way it ensures the **tester independence** is satisfied and bias won’t occur |

1. **Version control(GIT) philosophy**

We use **GitHub** to share our code. Team members can download existing files from **GitHub**. And we can also upload completed parts via **Git**. With **Git**, there is one **Master branch** that everyone pushes their code to after project owner makes sure the code works correctly and each one can pull in his or her own branch. The nice thing about **Git** is that we don't upload the wrong code. **Git** is able to discover newly added code by scanning and comparing. And the project owner can decide whether to merge the code. This prevents us from making careless changes to the code. In addition, Git has a role in restoring code. We can change the code at will, if we find that the modified code is not as good as the previous code can be restored to the original.

1. **Agile story and Test plan APPROVAL**

The agile story will first be created by each team members individually with a topic of their choosing. Then the team will hold a meeting to read all the agile stories together. Members will share their thoughts on each story, make changes or merge on the stories if necessary. After the stories for the entire project has been checked by the entire team and make sure to satisfy all the requirements. The team will go to a process of planning poker to set the priority of each story. The team will discuss again along the poker process. After the planning poker process is done, the agile story is officially received approval from the team and will put into effect. When the developer finishes their segment of code, they will hand their code over to another team member for testing. When the team member has tested the code, and has ensured it functions as required, the team member will hand the code over to the Product Owner for checking. The Product Owner will determine if the segment of code has met all of the requirements for that portion of the program. If it has, the Product Owner will then notify the original author that their story is done, and that they may move onto the next portion of the program (whatever that may be). The tester will write down the part that he or she plan to test. The project owner will consider the test whether it is useful or necessary to test. If he agrees to test it, he will sign the approval. Once the code for a specific task has been complete, the group must review and test the code currently waiting to be pushed. The test must confirm that the code runs through all cases (expected, unexpected, edge, ect) and the test plan results and expected output fall within the range as dictated in the test plan. Once the group consensus has been reached, the issue can be pushed to master.

1. **GLOSSARY OF TERMS**

**Black Box Testing -** Application testing that requires the tester to have no knowledge of the internal structure and logic of the code. Black-Box-Testing checks the functionality of the software which can be applied at any stage of testing.

**White Box Testing -** Application testing that requires the tester to possess knowledge of the internal structure and logic of the code. The software tester should be able to test the application for its input and output behavior.

**Github -** GitHub is a web-based version-control and collaboration platform for software developers.

**Waffle-** A project management tool powered by GitHub Issues and Pull Requests. Interact with team Issues and Pull Requests from one or more repositories in the form of cards on a board.

**UI** - The visual interface that prompts to the user when the software is in process.

**Google Test -** The tests themselves could be run one at a time, or even be called to run all at once. This makes the debugging process very specific and caters to the need of many programmers and coders alike.

**Regression testing -** A type of software testing designed to verify that the original functionality of the software remains intact after modification.

**QT -** Qt Creator is a cross-platform C++, JavaScript and QML integrated development environment which is part of the SDK for the Qt GUI application development framework

**Test Frameworks -** Frameworks or software used for Unit Testing and Google Test.

**Tester Independence** **-** The level of independence involved in team’s testing activities other than developer to avoid author bias.

**Unit Testing -** Unit Testing is a level of software testing where individual units/ components of a software are tested.