Jerry Berry

Test Plan

Product Owner Blake Molina
Scrum Master Xavier Lian
Arman Jafarinia
Stephen Lastimosa
Ryan Nguyen

Purpose of Test Plan: Intent of a the test plan is to give the team specific instructions to test the efficiency, sufficiency, and durability of our product.

Overall Strategy: The strategy behind the test plans is to try and break the program while running or find loops that might cause the program to produce errors, which will lead us to fix the issues we find.

Conditions Needed to Start the Testing: Need a working program that provides some type of input/output, a GUI if available, and testing information that will prove the program is working.

When to Stop Testing: Testing should stop when all a single test can no longer be tested due to errors that occur prior to the current state of the error.

Documents that support the test plan:

1) Waffle.io keeps track of the status for each story while being tested.

Features of test plan:

Test 1: Add stadium objects with all proper information to a database

1.1 Overview

We want to be able to import a collection of data that represent baseball stadiums. The baseball stadiums will contain data such as stadium name, team name, opening date, address, phone number, turf type, and stadium type. A picture of the stadium or the team logo will also be provided as a reference of important data.

1.2 Test Objective

Objective is to make sure that all the data is correctly input into a SQL database, and that the data can easily be retrieved by importing it at the startup of our program.

1.3 Test Type

This type of testing is a mixture of Whitebox and Blackbox testing. Whitebox testing because the actual code must be examined to ensure proper import. Blackbox testing because the ultimate proof the data was imported correctly is through the program GUI.

1.4 Test Procedure

1) Verify the program's internal database path is set to the correct file(initial data file given).
2) Reference copy that will be proof the information was correctly input and output.
3) Start the program.
4) Proceed to the administration page(Command+Enter)
5) Verify that the stadium object is added correctly which should include proper stadium name, team name, phone number, address, league, turf type, stadium type, capacity, opening date, and revenue.
6) Souvenirs should be listed below each selected stadium with the proper data entry of souvenirs for each team, which should also be listed with the proper pricing.
7) Repeat if needed.

Test 2: Sorting stadiums

2.1 Overview

We want to be able to sort the stadiums in the stadium information page based off of five different categories of sorting. Alphabetically by stadium name, park type, and team name, chronologically by opening date of the stadium, and capacity of the stadiums from lowest to highest integer value.

2.2 Test Objective

We want the baseball traveler or the administrator to be able to sort the stadiums when overlooking baseball stadiums different information for purposes like efficiency and user friendliness.

2.3 Test Type

The test case is of type Blackbox because we will only be looking at program functionality.

2.4 Test Procedure

1) Start the program.
2) Proceed to Stadium information.
3) Under the category labeled sorts there are five options to sort by. Stadium name, team name, opening date, capacity, and park type.
4) Choose one of the five options by clicking on a white push button which will turn blue when selected.
5) Stadiums listed above show a stadium picture, stadium name and team name. This list of stadiums should sort based on the category pushed. Alphabetically for stadium name, team name, and park type. Chronologically for the opening date and by ascending order for park capacity.
6) Verify the list is correctly sorted by referencing an already sorted version of the given list for each category.
7) Repeat the processes 3-6 by choosing another category to sort by.

Test 3: Filtering out stadiums

3.1 Overview

We want to be able to filter out stadiums to simplify stadium searches for a baseball traveler or an administrator.

3.2 Test Objective

We want to to be able to refine the searches or process of baseball travel planning by giving the traveler an option to filter out stadiums based upon four different categories: American leagues, national leagues, and whether the stadium uses natural or synthetic turf.

3.3 Test Type

The test case is of type Blackbox because we will only be looking at program functionality.

3.4 Test Procedure

1) Start the program.
2) Proceed to stadium information.

3) Under the category Filters there are four options to sort by: American League, National League, Use of Natural Turf, and use of Synthetic Turf.
4) Choose one of the five options by checking on a white check box which will turn blue when selected.
5) Stadiums listed above show a stadium picture, stadium name and team name.
6) When a option is checked it will show all the stadiums of that specific filter type. Multiple filters are allowed to be turned on at once.
7) Verify that the checked options display the correct stadiums based on their information by referencing to a copy sheet of the correct stadium data.
8) Repeat processes 3-7 for all the filters individually or together.

Test 4: Administration creation of teams

4.1 Overview

As an administrator we would like to be able to create a new team which also includes a new stadium, and save all the input data back into the SQL database that is connected with our program.

4.2 Test Objective

It is necessary to test the program capability and ensure that there is an option to add a new team followed by a new stadium. This means that the administrator must include information like stadium name, team name, address, capacity, turf type, opening date, total revenue, and league type.

4.3 Test Type

This test case is of type Whitebox testing because the SQL database must be examined to verify that the stadium was added with all the necessary information.

4.4 Test Procedure

1) Start the program.
2) Hold down CTRL key and press ENTER.
3) Enter “1234” into the textbox.

4) Select option to add a new team.
5) Fill out the necessary information to add a new team. NOTICE, an error will appear if the stadium the administrator wishes to add to is already filled.
6) The GUI will notice the added stadium immediately; however, to save the new stadium to the database select the button that allows to save and restart.
7) The official verification is shown in the sqlitebrowser application which relates the program database to the SQL database. Open sqlitebrowser application and browse the team or stadium list for the newly added team.

Test 5: Edit a team or stadium information to the database

5.1 Overview

As administrators we want to be able to edit the team or stadium.

5.2 Test Objective

We want to be able to edit all or any of the information of the team or stadium. Administrators should be able to edit the stadium number, stadium name, team name, league type, phone number, address, capacity, turf, opened, type, and revenue.

5.3 Test Type

The test type is a mixture of Whitebox and Blackbox. Whitebox testing because the actual database must be viewed to verify the edit worked. Blackbox because the GUI does show that the edit was initiated.

5.4 Test Procedure

1) Start the program.
2) Hold down CTRL key and press ENTER.
3) Enter “1234” into the textbox.
4) From this point there is a list of stadiums which includes the stadium number, stadium name, team name, league type, phone number, address, capacity, turf, opened, type, and revenue.
5) Double click any of the information boxes that you would like to change.

6) If the input is invalid an error box should appear, otherwise input will be saved to the GUI screen.
7) Select the option save & restart in order to save to the database.
8) To confirm the data edited was correctly re-entered, open the sqlitebrowser and view the SQL database that relates the program to the database.
9) This test can be repeated for any of the options, just repeat steps 2 - 6.

Test 6: Administrator has ability to add a new souvenir

6.1 Overview

We would like to be able to add a new souvenir to any stadium that also has a set price for the baseball traveler to see.

6.2 Test Objective

Objective of this test is for the administrator to be able to add a new souvenir to a stadium and also fix the price. The new item should be added to a souvenir list and be persistent in memory.

6.3 Test Type

This type of testing is Whitebox testing. It requires the administrator to check the SQL database and confirm the item was added.

6.4 Test Procedure

1) Start the program.
2) Hold down CTRL key and press ENTER.
3) Enter “1234” into the textbox.
4) Selection the option that allows the ability to add a new souvenir.
5) Select the stadium to add the the new souvenir to.
6) Enter the new item name.
7) Fix the price by either entering in the price or using the up down clicker to increase or decrease the price.
8) Click ok.
9) A new souvenir will be shown in the GUI under the selected stadium.

10) To validate the insertion of a new souvenir it should appear it two places. In the souvenir list box where it was initially added, or in the stadium information page.
11) If necessary backtrack to the stadium information page and select the stadium that the souvenir was added to. The new souvenir should have been added to that souvenir list with the price also showing.

Test 7: Administrator has the ability to delete a souvenir

7.1 Overview

It is necessary for the administration to have the ability to delete a souvenir from a stadium.

7.2 Test Objective

Objective is to delete a souvenir from a souvenir list at any stadium without causing any run time errors and having the list be persistent in memory.

7.3 Test Type

Testing type is Whitebox and BlackBox. Whitebox testing because in order to prove the deletion was executed correctly the actual program data must be examined. Blackbox testing because you can check program persistency by viewing the souvenir list in the stadium information section of the program.

7.4 Test Procedure

1) Start the program.
2) Hold down CTRL key and press ENTER.
3) Enter “1234” into the textbox.
4) Click on a stadium that a souvenir needs to be deleted from.
5) Below is a souvenir list of souvenirs at that stadium, click on one of the souvenirs to delete.
6) Find and click the button provides the option to delete a souvenir.
7) If delete was selected without selecting a souvenir to delete, an error box shall appear prompting the user to choose an item to delete.
8) Find and click option to save and restart.

9) The souvenir should no longer appear in the list of souvenirs for the stadium it was deleted from.
10) To check for program persistency return to the page called stadium information and click on a stadium that had a souvenir deleted from it. The souvenir deleted should not appear in the list of souvenirs for the selected stadium.
11) To verify the change was persistent open the sqlitebrowser and confirm the souvenir was deleted from the SQL database that is related with the program.

Test 8: Administrator has the ability to edit souvenirs

8.1 Overview

It is desired by administration to be able to edit souvenir names and prices.

8.2 Test Objective

Objective of the test is to be able to edit any souvenir at any stadium.

8.3 Test Type

Test type is Blackbox because the administrator can see the functionality of the program without seeing the internal infrastructure.

8.4 Test Procedure

1) Start the program.
2) Hold down CTRL key and press ENTER.
3) Enter “1234” into textbox.
4) A list of stadiums and their souvenirs should be listed.
5) To edit the name double click on the box that holds the name, it should become highlighted which signifies it is ready to be edited. Edit the name.
6) To verify the name was edited return to the startup page and continue to stadium information. At the stadium where the souvenir was changed the souvenir name should appear as changed.

7) To edit the price double click on the box that holds the price, it should become highlighted which signifies it is ready to be edited. Edit the price.
8) To verify the price was edited return to the startup page and continue to stadium information. At the stadium where the souvenir was changed the souvenir price should appear as changed.

Test 9: Use of Prim's algorithm to find optimal way of travel

9.1 Overview

It is desired by the administration to use Prim's algorithm to find the best connected map of all the stadiums(Minimum Spanning Tree).

9.2 Test Objective

Objective of the test is to show virtually the best way to connect all the stadiums while eliminating extraneous traveling distances that are not of the least mileage.

9.3 Test Type

This type of testing is Whitebox because in order to visually see accurate map after executing Prim's algorithm the programmer must have a correct analysis of the map that is not easily shown by the program. However, the GUI does provide a statement map that says which distances connect the map cost efficiently using Prim's algorithm.

9.4 Test Procedure

1) Start the program.
2) Hold down CTRL key and press ENTER.
3) Enter "1234" into the textbox.
4) Select the option that refers to Prim's MST
5) A new window will appear which starts at Anaheim stadium and calculates the shortest routes for all other stadiums creating a new map, eliminating unnecessary travel options, stating the distances between connected stadium options, and states the total distance coverage of the new MST.
6) To verify the algorithm worked a reference to the correct MST of the map provided by the program database must be

cross referenced and the total distance of the MST will also verify the MST is correct.

Test 10: ADD/REMOVE a stadium to the itinerary when planning a trip

10.1 Overview

We want to be able to ADD or REMOVE a stadium to the itinerary when planning a trip.

10.2 Test Objective

In order to have an itinerary the baseball traveler must be able to ADD or REMOVE stadiums to the itinerary list.

10.3 Test Type

This type of testing is Blackbox because the internal infrastructure is not necessary to look at.

10.4 Test Procedure

1) Start the program.
2) Proceed to the option that allows planning a trip.
3) There are three columns to choose from on the screen. The one on the right should enable the baseball traveler to add a stadium to the list. Choose the add option.
4) The stadium should appear on the right box portion of the screen.
5) Once added the option to add a stadium should change to a remove option. To test the REMOVE go to the stadium that was added and select the option to remove. The stadium added should not appear in the itinerary on the right anymore once the stadium was removed.
6) In addition there is an add all option as well which can be found at the top of the column add.
7) Press ADD ALL; all the stadiums should be added to the itinerary.

Test 11: Optimize the best travel route(uses Djikstra’s algorithm)

11.1 Overview

We want to be able to plan a trip to different baseball stadiums, but from our starting stadium be able to calculate the best route to each stadium based on distances.

11.2 Test Objective

When adding stadiums to the itinerary it is not always the case that the stadiums in a top down order would be the optimal route. So it is desired as an administrator to give the baseball traveler the option to optimize their itinerary in an order that from the starting position it finds the next shortest distance from the current position to the next stadium, this process is repeated until a new least is created.

11.3 Test Type

This test case is of Blackbox and Whitebox testing. Blackbox testing because the result of optimizing a list of stadiums is a surface function of the program, but in order to ensure correct travel options it is necessary to reference the program infrastructure and verify the result is correct.

11.4 Test Procedure

1) Start the program.
2) Proceed to the option planning a trip.
3) Choose any amount of stadiums to add to the itinerary.
4) Below the itinerary is an option to optimize. Click this option.
5) The stadiums in the itinerary should now reorder. This new order is ordered by shortest distance from each current point starting at the first, the next, and so on.
6) Verify the stadiums are in the correct order by referencing a map of the correct stadiums and calculating the correct route of the chosen stadiums by hand. This will confirm if the Djikstra's algorithm worked.

Test 12: Adding and removing from the wishlist

12.1 Overview

We want to be able to create a shopping cart or wishlist while traveling to all the stadiums. This list should have the ability to be added to or removed from.

12.2 Test Objective

Objective for this test is to test the functionality of the wish list like the adding or removing from the wish list.

12.3 Test Type

Test type is Blackbox testing because it not necessary to critique the internal infrastructure of the program.

12.4 Test Procedure

1) Start the program.
2) Proceed to option planning a trip.
3) When adding a stadium to the itinerary, there is also another page called Wishlist.
4) This page should allow the user to view items they would like to purchase over the course of their trip.
5) Choose any item or items at any stadium. Once clicked it should appear in the wishlist and below the souvenir clicked on should now appear a number box that will represent a quantity for the souvenir desired.
6) In order to remove from the wishlist simply click on the souvenir orignally added and it should no longer appear on the wishlist.
7) A total cost of all the souvenirs added to the wishlist will appear at the bottom of the wishlist.
8) A total number of souvenirs is also shown at the bottom of the wishlist.

<u>Test 13: Checking for total revenue for each stadium</u>

13.1 Overview

We want to be able find the revenue for all the stadiums to ensure that souvenirs are cost are kept track of.

13.2 Test Objective

Objective for the test is to ensure that revenue for each stadium is being kept track of in the database of the program. It is important for the administration to keep track of this.

13.3 Test Type

This test type is Blackbox and Whitebox. Blackbox because the revenue can be provided to view without viewing the internal infrastructure, but Whitebox because it is necessary to view the SQL database and confirm the revenues are correct.

13.4 Test Procedure

1) Start the program.
2) Proceed to the planning a trip.
3) Order whatever can be calculated easily by hand by any amount of stadiums visited.
4) Return back to the startup page.
5) Hold down CTRL key and press ENTER.
6) Enter “1234” into the textbox.
7) A list of stadiums should appear, what we need is the revenue which is listed on the right of the stadium informations. Refer to the hand calculated revenues and confirm if the revenues were calculated correctly.
8) To ensure the database stored the revenues correctly, open the sqlitebrowser and verify by viewing the database that the revenues were calculated correctly.