**USER STORIES**

**Team: my group jeff (Bryce, Jared, Brian, Brandon)**

**#1** As a traveler I want to visit each NFL stadium starting from Los Angeles Memorial Coliseum.

**a) Description:** This story will use Dijkstra’s algorithm to plan the most efficient path starting from LA.

**b) Task:** use the algorithm and show the trip **c) Test:** Upon starting from LA, check if path is correct

**d) Priority:** 3rd sprint

**e) Estimation:** 3 **f) Done:** The path is the correct one.

**g) Assignee:** Brandon

**h) Assumptions:** Database will be up and running and dijkstra’s algorithm already implemented.

**#2** As a traveler I should be able to create my own custom trip where i can select a starting stadium and visit all the other ones I want to visit.

**a) Description:** This story uses the dijkstra’s algorithm to calculate the shortest distance using the order specified. It will display the total distance traveled.

**b) Task:** Let user choose stadium names and make trip **c) Test:** Choose various stadiums and test with those random values.

**d) Priority:** 3rd sprint

**e) Estimation:** 5

**f) Done:** The traveler should see exactly where to go for the most efficient path and we should see the correct distance displayed onto the screen.

**g) Assignee:** Brian

**h) Assumptions:** Database and dijkstra algorithm will have been implemented.

**#3** As a traveler, I want to view the mileage between all stadiums starting at Hard Rock Stadium. **a) Description:** Traveler will choose to view the mileage, and a DFS will be performed starting at Hard Rock Stadium. After the search, the mileage will be displayed. **b) Task:** Implement and use the DFS algorithm on a tree structure that contains all the stadiums, starting at Hard Rock Stadium. **c) Test:** Execute the search, and make sure the mileage concurs with the manually added mileage from the given data. **d) Priority:** 2nd sprint **e) Estimation:** 5

**f) Done:** When the traveler can successfully view the correct mileage.

**g) Assignee:** Brian

**h) Assumptions:** Database will be completed and a gui will be there for them to choose from.

**#4** As a traveler, I want to view the mileage between all stadiums starting at Lambeau Field. **a) Description:** Traveler will choose to view the mileage, and a BFS will be performed starting at Lambeau Field. After the search, the mileage will be displayed.. **b) Task:** Implement and use the BFS algorithm on a tree structure that contains all the stadiums, starting at Lambeau Field. **c) Test:** Execute the search, and make sure the mileage concurs with the manually added mileage from the given data. **d) Priority:** 2nd sprint  **e) Estimation:** 3

**f) Done:** The user sees the correct milage from Lambeau Field to all other stadiums.

**g) Assignee:** Jared

**h) Assumptions:** Database will be completed and a gui will be there for them to choose from.

**#5** As an admin we should be able to modify (add/delete/change) the souvenirs list and their prices per NFL team. (administrator only)

**a) Description:** The admin can modify delete and change the souvenir list and their prices for each and every NFL Team.

**b) Task:** Modify souvenirs at the stadiums

**c) Test:** Test by changing the values **d) Priority:** 1st Sprint **e) Estimation: 1**

**f) Done:** Information is modified and changed. The admin can also see that souvenirs are deleted properly.

**g) Assignee:** Bryce

**h) Assumptions:** Database will have been completed and gui will be done.

**(Baseline)#6** As a traveler I want to know how much seat capacity each stadium has.

**a) Description:** Displays a list of NFL stadiums and their corresponding team names sorted by seating capacity (smallest to largest). Be sure to display seating capacity. Display the total capacity of all NFL teams.  **b) Task:** UI display that shows the user the seating capacity of the stadiums. **c) Test:** Check if the list is sorted by the seating capacity.  **d) Priority: 1st Sprint  
 e) Estimation: 1**

**f) Done:** The UI shows a list that contains the stadium name and the corresponding teams ordered by the seating capacity (smallest to largest). It will also display the seating capacity for each stadium and the total seating capacity.

**g) Assignee: Brandon**

**h) Assumptions:** Database should be done.

**#7** As an administrator I should be able to maintain football stadiums(Admin only), so that the program can be updated with new teams that will pop up. **a) Description:** Provide the capability to modify stadium information including capacity if the team moves into a new stadium.

**b) Task:** Move LA rams to farmers field 2019

**c) Test:** Check if it successfully read in data and new colleges were added to the list. **d) Priority:** 2rd sprint **e) Estimation:** 1

**f) Done:** Teams can be moved and capacity is changed appropriately.

**g) Assignee:** Bryce

**h) Assumptions:** Database should be done.

**#8** As an administrator I want to maintain football teams and stadiums if the NFL adds new teams.

**a) Description:** The admin should be able to add new values for the new teams that are added by the NFL and all the other associative data with each task.

**b) Task:** Add new football teams

**c) Test:** Test if they changed or not.

**d) Priority:** 3rd Sprint

**e) Estimation:** 2

**f) Done:** Should be able to see database change to see respective adding of new teams.

**g) Assignee:** Bryce

**h) Assumptions:** Database should be done.

**#9** As a traveler I want to determine the minimum spanning tree that connects all the NFL stadiums.

**a) Description:** The MST algorithm should display the associated mileage for the trip.

**b) Task:** Perform Prim’s or Kruskal’s algorithm

**c) Test:** Check if the mileage is correct

**d) Priority:** 2nd sprint

**e) Estimation:** 5

**f) Done:** Done when the mileage and the algorithms output match

**g) Assignee:**Brandon

**h) Assumptions:** Database should be done.

**#10** As a traveler I would like a list of various information regarding what I select.

**a) Description:** The program should be able to display:

* all the information related to a particular football team (team name, stadium name, seating capacity, location, conference, surface type, stadium roof type, star player).
* a list of the NFL teams sorted by team name.
* a list of NFL stadiums and their corresponding team name sorted by stadium name.
* a list of only the American Football Conference teams sorted by team name.
* a list of only the National Football Conference teams sorted by team name.
* a list of only the stadiums that have an “open” stadium roof type and their corresponding team name sorted by stadium name. Display the number of stadiums have an “open” roof.
* a list of NFL star players and their corresponding team name sorted by team name.
* a list of NFL teams, their stadium names, their surface type, and their corresponding location sorted by surface type.

**b) Task:** Make lists of certain filters

**c) Test:** show output on a table

**d) Priority:** 1st Sprint

**e) Estimation:** 1

**f) Done:** Each list is functional and displays information accurately

**g) Assignee:** Brian

**h) Assumptions:** Database should be done.

**TEAM RULES**

* Communication through Discord (VOIP service)
* Fulfill your assigned work in a timely manner.
* Have descriptive commit messages
* Let everyone know via Discord when you’re working on something/when you push something to the repository
* Meet at least twice a week (in lab)

**CODING STANDARDS**

* Using [Linux Kernel Coding Standards](https://www.kernel.org/doc/html/latest/process/coding-style.html) that are applicable for C++

Among the methods used in our software is a Depth-First Search. In this method, each vertex n of our graph is initially labeled as unexplored, which can be identified as n labels. Each edge m of the graph is also labeled as unexplored, identified as m labels. As the function loops, each vertex will be labeled again as visited, another n labels. Each edge is labeled as discovery if used in our function, and then labeled as back if not used, another m labels. Therefore, the operation can be visualized as n + n + m + m operations. This is simplified to an O(n + m) running time for DFS.