Team Members: Aaron Geesink, Garrett Geesink, Milo Fisher, Noah Agudelo

Baseline Story (1 point) - User story #4 (main menu)

User stories that will be assigned to sprint 1: Story 1, 2, 3, 4, 7
User stories that will be assigned to sprint 2: Story 5, 8, 9, 10, 11, 12
User stories that will be assigned to sprint 3: Story 6, 13, 14, 15

Sprint 1 deadline: September 23, 2019
Sprint 2 deadline: October 7, 2019
Sprint 3 deadline: October 21, 2019

1. As a developer, I want a city class so that I can instantiate city data
    a. Description:
        i. Backend code for the city class that so that city class data can be instantiated with relevant private data
        ii. City class private data will be stored within in SQL database once it is implemented
    b. Assumptions: None
    c. Tasks:
        i. Create the private data for the city class, which will include
            1. the name of the city as a string
            2. A vector (or other STL container if more suitable) for the distances in km from the current city to every other city as integers
            3. A vector (or other STL container if more suitable) for the name of each city that can be traveled to, with an index number corresponding to the distance to be traveled by the distance container
            4. a class representing food that can be purchased in each city (food class implementation in user story #2)
        ii. Create mutator and accessor functions to modify the instantiated data
        iii. Create a function to output the ending cities and their distance to the current city to the console for testing
        iv. Create a class or struct to dynamically store instantiated city classes into a vector which can be passed to algorithms that need to process city data
        v. As we begin to create the SQL database, implement the functionality to store the data within the SQL database
    d. Test Scenarios:
        i. Instantiate a city class with the name of Amsterdam
        ii. Add the name of the 10 other basic ending cities to the endingCities container as well as their distances from Amsterdam in the distances container

iii. Call the function to output the ending cities and their distance to the current city to the console, and verify that each ending city has a distance from the current city.
iv. Dynamically Create a city class with the data for Amsterdam and another with the data for Berlin and store them within a vector. Call the function to output the distance from each city to the ending cities to the console.

e. Assignee: Aaron
f. Story Point Estimation: 2
g. Priority: 1
h. Definition of done:
    i. A city class can be instantiated with the data from the European Distances and Foods spreadsheet, and the data can be outputted to the console
    ii. Multiple instances of the class can be dynamically added and deleted to a vector

2. As a developer, I want a food class so that I can instantiate food data
   Put the food class within the city class
    a. Description:
        i. Create a food class that will contain the name of foods and their price
        ii. Food class will be a private data member of the city class
        iii. Could potentially implement food class separately from city class if it becomes more convenient for future algorithms dealing with food prices
    b. Assumptions: none
    c. Tasks:
        i. Create the private data for the city class, which will include
            1. A vector or other STL container of the name of each food item
            2. A vector or other STL container of the price of each food item
        ii. Create accessor and mutator functions for food class private data
        iii. Create function to output food data to the console for testing
        iv. Add the food class to the private data of the city class
    d. Test Scenarios:
        i. Instantiate a food class with the Traditional food items for Amsterdam as well as the cost of each food item, and output it to the console
        ii. Instantiate a food class within a city class with the Traditional food items for Amsterdam as well as the cost of each food item, and output it to the console.
    e. Assignee: Aaron
    f. Story Point Estimation: 1
    g. Priority: 2
    h. Definition of done:
        i. Food class can be instantiated with food names and their corresponding prices

        ii. Food class can be accessed within the city class
        iii. Food class can be outputted to the console

3. As a developer, I want an SQL database so that I can store food, city and password data
    a. Description:
        i. Create an SQL framework which stores data for cities, food, and passwords for use within the program.
    b. Assumptions: None
    c. Tasks:
        i. Research and learn about SQL
        ii. Create an SQL table for cities. Each city will have data for Traditional food items, as well as relevant data for calculating distances
        iii. Create an SQL table for password
    d. Test Scenarios:
        i. Verify that the SQL database stores city, food, username, and password data.
    e. Assignee: Milo/Noah
    f. Story Point Estimation: 13
    g. Priority: 1
    h. Definition of done:
        i. The SQL database has a table for cities
        ii. Each city has Traditional Food data
        iii. Each city has data used to calculate distances
        iv. The SQL database has a table for usernames and passwords

4. As a traveler, I want a main menu so that I can select the different ways that I can plan my trip
    a. Description:
        i. When the program is initially loaded, the first thing that a traveler will see is a main menu which will enable to access all of the program's features.
        ii. The traveler will be able to click these buttons to take them to the other menus in the program
    b. Assumptions: None
    c. Tasks:
        i. Implement the main menu in a QWidget class, complete with buttons that go to windows for all of the program's features, which includes:
            1. Login (for admins)
            2. View European City and Food Data
            3. Plan a Trip
            4. Contact Us
            5. Edit city/food data (admin only)
        ii. All buttons will be in the center of the main window

iii. The menus that each button takes you to are implemented within their relevant user stories

d. Test Scenarios:
    i. Verify that the Login button opens the login window
    ii. Verify the Trip Planning button opens the Trip planning menu
    iii. Verify the contact us button opens the contact us window
    iv. Verify the purchasing button opens the purchasing window
e. Assignee: Garrett/Noah
f. Story Point Estimation: 1
g. Priority: 1
h. Definition of done:
    i. All of the program's features are accessible starting from the main menu
    ii. The main menu is shown upon program startup

5. As an admin, I want a login screen within the software which will allow access to admin features within the program upon a successful login.
    a. Description:
        i. In the main menu there will be a button labeled "Login" which will cause a login window to pop up when clicked.
        ii. An admin can input their username and password into the login window and click "Login". If their username and password are valid, they will be granted admin access to the program. If their credentials are invalid, they will be given a warning at the bottom of the login window telling them "Your username or password is incorrect."
        iii. The login window can be closed by clicking on a small X on the top right of the window.
    b. Assumptions: Main Menu done
    c. Tasks:
        i. Develop the look of the UI using a QWidget. It will have:
            1. Username and Password fields to input username and password
            2. "Login" button to attempt to login using the given username or password.
            3. Small X on the top right to close the login window
        ii. Develop a system which will read in the user's username and password, and compare it usernames and passwords in the SQL database. If a matching username and password combination is found, then the user will see "Login successful" in the login windows and will be granted access to admin features
    d. Test Scenarios:
        i. Test that the login window can be opened from the main menu
        ii. Verify that a user can login to the program through the login window using the correct username and password combination

iii. Verify that a user can not login without having the correct username and password
iv. Verify that a successful login grants a user access to the program's admin features
v. Verify that a regular user can not access admin features

e. Assignee: Garrett/Noah
f. Story Point Estimation: 2
g. Priority: 4
h. Definition of done:
i. A user can open the login window from the main menu by clicking a button labeled "Login"
ii. Login windows has fields for a username and password, a login button, and an X on the top right of the window to close the login window
iii. A user can type a username and password and click login. If the username and password combination is valid, they will be granted admin access to the program. Otherwise, they will be given a warning and not be granted admin access.

6. As an admin, I want a password encryption/decryption system so that I can securely store hashed passwords within SQL (optional)
    a. Description:
        i. There will be a system which will allow an admin to decrypt a password, and store this decrypted password in the SQL database.
        ii. Upon a login attempt, the username and password combination input by the user will be compared to the encrypted password key stored in the SQL database to see if the login credentials are valid.
        iii. This is so that passwords stored in the program's database are encrypted instead of being stored as plain text.
    b. Assumptions: Login window done, and a valid username and password grants admin access to the program
    c. Tasks:
        i. Study and learn about methods of password encryption/decryption
        ii. Develop an algorithm which will encrypt a password, and store that encrypted password into the SQL database
        iii. Develop a system for comparing user input to encrypted password hashes in order to determine a valid password
    d. Test Scenarios:
        i. Verify that the encryption system can turn ascii text into an encrypted password hash
        ii. Verify that a password hash can be checked against a user's password input in order to determine if the user's password is valid
        iii. Verify that no passwords are stored in the SQL database as unencrypted ASCII text

    e. Assignee: Milo
    f. Story Point Estimation: 4
    g. Priority: 10 (optional)
    h. Definition of done:
        i. All passwords stored in the SQL database are encrypted
        ii. When a user inputs a password to login, that password is compared to the encrypted password in the SQL database in order to determine if that password is valid.

7. As a developer, I want an algorithm which will calculate the closest city to another city.
    a. Description:
        i. This algorithm will find the closest city to any other city in the SQL database.
        ii. The algorithm can also map out the shortest path across multiple cities
    b. Assumptions: SQL database completed
    c. Tasks:
        i. Develop a class which will take in any amount of cities as an input and then calculate the shortest path which will hit every one of the input cities.
            1. The class will have a function which will receive a vector of cities as an input, and will sort the cities in the order they will be visited such that the total distance travelled is as short as possible, known as “Shortest Distance” Order
        ii. Allow the class to work with the city class as a parameter (or a vector of city classes)
    d. Test Scenarios:
        i. Give the algorithm multiple different vectors of cities, and verify that the algorithm sorts the cities in the order they will be visited such that the total distance travelled is as short as possible.
    e. Assignee: Noah
    f. Story Point Estimation: 10
    g. Priority: 3
    h. Definition of done:
        i. The shortest distance algorithm takes in a vector of cities as an input, and then sorts that vector in “Shortest Distance” order (i.e. the cities are in the order they will be visited such that the total distance travelled is as short as possible.)

8. As a traveler, I want to see a list of European Cities with their distances from Paris and the food that I can purchase there.
    a. Description:
        i. A traveler can click a button on the main menu that will bring them to a list of European cities and their distances from each other

ii. The traveler will have a dropdown menu of cities to choose from, and selecting one will display the distances from the city to all other cities that are available

b. Assumptions: main menu done, city class done, loading from SQL database done
c. Tasks:
    i. Create a qwidget menu that the “City and Food Data” button on the main menu will take you to when clicked
    ii. In this menu, add a drop down menu containing all of the European city classes that are loaded into the program (Either hard coded or loaded from the SQL database if that is implemented
    iii. Implement functionality so that when a city in the dropdown is selected, the city class data for that city will be displayed within a table to the right of the dropdown, including:
        1. Distance from the selected city to every other city
        2. The food that can be purchased at the city
        3. The price of each food item
d. Test Scenarios:
    i. In the dropdown menu, select every single possible city and verify that it changes the distances and food in the tables to the corresponding data for the selected city
    ii. In admin mode, add a new city to the SQL database, load it to the program, and verify that the newly added city appears within the dropdown menu and it’s data is loaded to the tables with selected
e. Assignee: Aaron
f. Story Point Estimation: 4
g. Priority: 5
h. Definition of done:
    i. Clicking the “City and Food Data” button on the main menu changes the program to the City and Food Data Menu
    ii. Every city currently in the SQL database appears in the dropdown menu, and the food and distance data for each city is displayed onto a table when selected

9. As a traveler, I want a menu where I can select the different types of trips I can plan
    a. Description:
        i. A traveler can click a button on the main menu that will bring them to menu where they can select between 3 different trip plans
        ii. This menu will contain buttons to plan 3 different kinds of trips:
            1. 11 Cities starting at Berlin
            2. Shortest Trip from London
            3. Custom Trip
        iii. Each button will take you to a menu for the selected trip setting

b. Assumptions: main menu done
c. Tasks:
    i. Create a qwidget menu that the “Plan a Trip” button on the main menu will take you to when clicked
    ii. Create a button for the three kinds of trips that can be planned, which will take you to their corresponding menus when clicked:
        1. 11 Cities starting at Berlin (specified in User Story 10)
        2. Shortest Trip from London (specified in User Story 11)
        3. Custom Trip (specified in User Story 12)
    iii.
d. Test Scenarios:
    i. Verify that the Trip Planning menu can be accessed from the main menu
    ii. Verify that the trip planning menu can be exited to return to the main menu
    iii. Verify that there are options in the trip planner to select different types of trips which take the user to the corresponding menus
e. Assignee: Garrett
f. Story Point Estimation: 1
g. Priority: 5
h. Definition of done:
    i. The Trip Planning menu can be accessed from the main menu and can be exited to return to the main menu
    ii. The options in the trip planner take you to their corresponding menus for planning the specified type of trip

10. As a traveler, I want to plan a trip starting at Berlin or Paris which will take me to 11 predetermined European cities with the shortest distance travelled.
    a. Description:
        i. In the “Trip Planner” menu, a user can select a trip starting at Berlin or Paris. The program will then show them the shortest trip which will take them to the program’s 11 default cities.
    b. Assumptions: Shortest distance algorithm done, SQL database done
    c. Tasks:
        i. Using the shortest distance algorithm, create two vectors which stores the 11 default cities order, starting from both Paris and Berlin. Store these vectors in the SQL database
        ii. The vectors must be in “Shortest Distance” order (i.e the cities in the order they will be visited such that the total distance travelled is as short as possible.)
        iii. In the Trip Planner, allow these trips to be selected from the list of trip options.
        iv. Add a menu that can be accessed by clicking the “11 Cities starting at Berlin” button in the Trip Menu.

        v. In this menu, display the distance traveled between the 11 cities as well as the food that can be purchased at each city
    d. Test Scenarios:
        i. In the Trip Planner, verify that a user can select a shortest trip starting from Paris or Berlin as a trip option with all 11 cities appearing.
    e. Assignee: Milo/Garrett (SQL), Noah (algorithm), Aaron (displaying data)
    f. Story Point Estimation: 9
    g. Priority: 6
    h. Definition of done:
        i. From the Trip Planner, a trip starting from Berlin or Paris can be selected. Then, the program displays a trip with the shortest distance which starts at Berlin or Paris and takes them through the 11 default cities.

11. As a traveler, I want to plan a trip starting at London which will allow me to choose the number of cities I wish to visit, then take me on the shortest trip through that number of cities.
    a. Description:
        i. In the “Trip Planner” menu, a user can select a trip starting at London, as well as the number of cities they wish to visit. The program will then show them the shortest trip which will take them to that number of cities
    b. Assumptions: shortest distance algorithm done
    c. Tasks:
        i. Create a function which will receive two inputs: the Starting City, and the Number of Cities to visit. The function must then output a vector of cities which will start with the Starting city, and then have an additional amount of cities that the traveller wishes to visit.
        ii. The vector must be in “Shortest Distance” order (i.e the cities in the order they will be visited such that the total distance travelled is as short as possible.)
        iii. Add a menu that can be accessed by clicking the “Shortest Trip from London” button in the Trip Menu
        iv. In this menu, Add a dropdown menu or dialog box that prompts the user to select the starting city as well as the number of cities they want to visit. Once selected, the cities that will be visited and the food that can be purchased will be displayed on a table to the right of the menu.
        v. Add contingency handling so that the number of cities to visit cannot be greater than the maximum cities that are loaded from the database.
    d. Test Scenarios:
        i. In the Trip Planner, verify that a user can select shortest trip from London to go to its corresponding menu
        ii. Verify that the cities visited produce the shortest possible distance from London for a selection of 1, 5, 10, and 15 cities
    e. Assignee: Milo/Garrett (SQL), Noah (algorithm), Aaron (displaying data)

f. Story Point Estimation: 9
g. Priority: 6
h. Definition of done:
    i. From the Trip Planner, a trip starting from London can be selected. Then, the number of cities can be specified. Finally, the Program displays a trip with the shortest distance which starts at London and goes through the specified number of cities.

12. As a traveler, I want to plan a trip starting from any city that allows me to visit any number of selected cities in the shortest distance
    a. Description:
        i. “Custom Trip” option in the trip planner
        ii. A traveller can select a starting city as well as all of the cities they wish to visit. Then, the program displays a trip which will take them through all of the selected cities in shortest distance order
    b. Assumptions: Trip Planner UI implemented, shortest distance algorithm done
    c. Tasks:
        i. In the Trip Planner, implement a menu which will allow the traveler to select a starting city and all other cities they wish to visit.
        ii. Implement a button called “Calculate Trip”, which will format the selected cities into a vector, then display these cities in shortest trip order.
    d. Test Scenarios:
        i. In the “Custom Trip” menu, select a starting city and all other cities to visit. Then click “Calculate Trip”. Verify that the program displays all selected cities, and that the order they are in is the shortest distance.
        ii. Implement contingency handling for edge cases
            1. User elects no cities
            2. User selects only a starting city
            3. Use selects no starting city
    e. Assignee: Milo/Garrett (SQL), Noah (algorithm), Aaron (displaying data)
    f. Story Point Estimation: 8
    g. Priority: 7
    h. Definition of done:
        i. In the Trip Planner, a traveller can access a ‘Custom Trip” menu
        ii. Custom Trip which will allow travellers to select a starting city and all other cities they wish to visit
        iii. When the traveller click on “Calculate Trip,” they are shown the shortest trip which takes them through the selected cities

13. As an admin, I want the capability to add, delete, or modify European cities and their corresponding food items in the SQL database
    a. Description:
        i. After logging in, an admin can access an “Edit City/Food Data” menu

ii. This menu allows the admin to add new cities that can be visited, add new food items at each city, and delete food items and cities
iii. These changes will be stored within a vector of city classes before being saved to the SQL database (as specified in user story 14)

b. Assumptions: SQL Database created, admin login done
c. Tasks:
    i. Create a QWidget menu that can be accessed when the “Edit City/Food Data” button is clicked on the main menu
    ii. In the menu, create form fillable UI elements that prompt the admin for city information, including the name of the city, the distance from the new city to every other existing city, the food that can be purchased, and the price of each food
    iii. Add an “Add City” button that will instantiate a new city class with the form fillable elements and store the city class into a vector of city classes.
    iv. Add an “Delete City” Button that will delete a selected city from the vector of loaded city classes
    v. Add a “Edit City” Button that will allow the admin to modify a selected city’s data from the vector of loaded city classes. The cities data will be loaded to form fillable UI elements which can be modified and then saved to the class by calling the mutator functions.
d. Test Scenarios:
    i. When “Edit City/Food Data” button is clicked, a menu opens up with options to edit a city data, add a new city, and delete a city.
    ii. When selecting edit a city, a list of cities will be displayed so the admin can click on one of those cities to edit
        1. this menu contains fillable forms that can have data inputted into them.
    iii. When selecting to add a new city, a form is opened up that allows the admin to type into it.
    iv. When selecting to delete a city, the application will show a list of cities for the admin to delete from.
        1. When the admin selects a city to delete, the application will print out to the console the list of cities before and after the deletion to verify the city has indeed been deleted.
e. Assignee: Aaron and Noah
f. Story Point Estimation: 5
g. Priority: 8
h. Definition of done:
    i. Verify that the “Add City” button instantiates a new city class with the data from the form fillable UI elements by calling the function to display the city data to the console
    ii. Verify that the “Delete City” button deletes an existing city class by outputting the city name of all elements in the city class vector

iii. Verify that the “Edit City” Button loads an existing city class to the form fillable elements and can be saved back to the vector of city classes by printing the modified class to the console.

14. As an admin, I want to save any changes that I make to European cities and food item to the SQL database
    a. Description:
        i. Once an admin has made changes in the “Edit City/Food Data”, they can click a “Save Changes” button to save those changes to the SQL database.
        ii. Saved data is kept even between instances of the program
    b. Assumptions: Edit City/Food Data menu done, SQL database Done, Admin Login Done
    c. Tasks:
        i. Learn how to save to an existing SQL database
        ii. Implement a parser which turns user data in the qwidget UI into SQL data to be saved in the database
        iii. Show that data has been saved.
        iv. Prevent program from saving invalid data (i.e. display an error when trying to save a number to a food data field)
    d. Test Scenarios:
        i. Verify that the file Saving system saves user input from the UI to the SQL database
        ii. Verify that an error message appears when the user attempts to save invalid data, and that no data is saved
    e. Assignee: Garrett and Milo
    f. Story Point Estimation: 6?
    g. Priority: 9
    h. Definition of done:
        i. File saving system saves user input from the “Edit City/Food Data” menu to the SQL database
        ii. Saving system prevents invalid data from being saved.

15. As a traveler, I want to be able to purchase traditional food items when visiting a European city
    a. Description:
        i. A traveller can see traditional food items from any city they visit
        ii. A traveller can purchase food items from any city they visit
        iii. A traveller can see the total amount they have spent at each European city, as well as a grand total for all cities visited.
    b. Assumptions: Travellers can plan trips in the program which will take them to multiple European Cities.
    c. Tasks:

i. Implement a system which will:
    1. Allow a user to purchase any food item from the cities they visit
    2. Keep track of the food items purchased at each city.
    3. Display the total amount spent at each city
    4. Display the grand total for all cities visited

d. Test Scenarios:
    i. Select a trip plan. Then, purchase food items for each city in the trip.
    ii. Verify that the amount spent at each trip is displayed correctly in the purchasing menu
    iii. Verify that the total spent for all cities is displayed correctly in the purchasing menu
e. Assignee: Noah
f. Story Point Estimation: 7
g. Priority: 8
h. Definition of done:
    i. User can purchase traditional food items from the cities that they visit
    ii. User can see the amount spent at each city, and the total amount for the trip.