**Team AlphaOmegaAwesome**

Cody Thompson (Team Member) Chase Deric (Team Member)

Chaz Del Prato (Scrum Master) Mohamed Samatar (Product Owner)

**Lab 4 AGILE stories**

* **Baseline Story:** As a vacationer, I want to be able to view all the traditional foods offered and its cost in any given European city.

**Story Point Estimation: 1**

**Assumptions:** The foods and their costs are already stored and organized in a data structure.

**Description:** Foods and their respective costs will be contained within a data structure and have their attributes, food name and cost, output through established methods.

**Assignee**: Cody Thompson

**Priority (1-3): 1**

**Tasks & Tests:** Ensure that the foods and their costs have been assigned to the proper city. Output the attributes and reference the data given on the cities to check accuracy.

**Definition of Done:** The GUI will have a combo box that will have all the names of the cities that can be chosen. Once you select what city to view, you hit go and it will display the menu items with name and cost into a table.

* **Story:** As a vacationer, I want to be able to view a list of European cities and their distances from Berlin.

**Story Point Estimation: 2**

**Assumptions:** All European cities and their distances from Berlin are already stored and organized in a data structure.

**Description:** Every European city available for the trip will be have their city name and distance from Berlin recorded into a City object. All cities will be contained within a data structure and output through an established method.

**Assignee**: Cody Thompson

**Priority (1-3): 1**

**Tasks & Tests:** Ensure that each city name has been assigned to the proper distance from Berlin. Output the attributes of the City object and reference the data given on the cities to check accuracy.

**Definition of Done:** If the vacationer selects the city from the combo box and hits go it will display the city the vacationer can go to and the distances to each into a table.

* **Story:** As an admin, I want to be able to create a database that holds the starting city, ending city, distance between those cities, city’s food, and cost of the food.

**Story Point Estimation: 5**

**Assumptions:** The database will use SQLite commands and QT API to fill vectors in order to run algorithms on.

**Description:** The QT API for SQLite has built in functions that allow me to open/close the database and fill the database through a callback function. The purpose of this story is to show that the program is communicating with the database.

**Assignee**: Chase Deric

**Priority (1-3): 1**

**Tasks & Tests:** Make sure that the database is setup correctly. To ensure the database has been successfully connected, we will be outputting the list of cities and food items into the GUI to determine if the database is being read correctly.

**Definition of Done:** The GUI will display a side by side window. On the right side will display the food options for all European Cities. On the left Side will be the list of European Cities that can be visited. At the bottom of the window, there will be text that states of the database has been loaded successfully.

* **Story:** As an admin, I want to be able to add cities to the database.

**Story Point Estimation: 2**

**Assumptions:** The database can be added to using SQL commands through the C++ API.

**Description:** This includes adding to the data structure that the C++ API created originally of the cities, food, and costs.

**Assignee**: Chase Deric

**Priority (1-3): 3**

**Tasks & Tests:** Make sure that the database has been added to correctly. For testing, output the distanceTable where the city is the new city and determine if the table has the correct city.

**Definition of Done:** An admin page will be added to the GUI which will display all cities, their food, and their costs. The admin will be able to add or delete from this page. The database will be updated and redisplayed on the page to reflect these changes made.

* **Story:** As an admin, I want to be able to change the cost of food as the vacationer learns the reality of the cost of food in the given city.

**Story Point Estimation: 13**

**Assumptions:** The database will use SQL commands and C++ API to change the data structure in order to better reflect the vacationers’ experience.

**Description:** The C++ API for SQLite has built in functions that allow me to change any given table inside the database. Furthermore, I can alter columns by using the alter table command.

**Assignee**: Chase Deric

**Priority (1-3): 2**

**Tasks & Tests:** Make sure that the database reflects the changes that I made to it. For testing, output the food\_costs table where I changed the cost and determine if it matches the true cost of food.

**Definition of Done:** An admin page will be added to the GUI which will display all cities, their food, and their costs. The admin will be able to modify information about the cities from this page. The database will be updated and redisplayed on the page to reflect these changes made.

* **Story:** As a vacationer, I want to log all foods I’ve purchased from every city I’ve visited, and I want to see a total for each city along with a grand total of all food.

**Story Point Estimation: 8**

**Assumption:** The cities and food items are all recorded in a database.

**Description:** Each person object should have a way to store every food item they’ve purchased in an array or list. When calculating the grand total, add every food item’s price.

**Assignee: Mohamed**

**Priority (1-3): 3**

**Tasks & Tests:** Use an array or list to store each food item, make a member function to total each food item price

**Definition of Done:** Whenever the traveler takes a food item, the price is recorded and totaled, and when the trip is finished, the prices from every city are totaled. This grand total is then shown to the traveler.

* **Story:** As a vacationer, I want to have the option to visit every possible city, starting from London, with the ability to see the distance I’ve traveled and the ability to purchase food items from each city.

**Story Point Estimation: 5**

**Assumption:** Each city has a corresponding distance and list of food

**Description:** Assign the traveler to London, then add distance from current city upon travel to another city. Depending on which city the traveler is in, certain food items become available.

**Assignee: Mohamed**

**Priority (1-3):** 2

**Tasks & Tests:** Have each traveler default to London, then allow the user to set their city. Show food that is available to the traveler depending on their city.

**Definition of Done:** As the vacationer goes through the trip, they will stop at each city and it will display the cities food options and a corresponding price. The vacationer will be able to select a food item to purchase and will display a receipt of what was purchased. At the end of the trip there will be a trip summary of how much was spent and the total distance traveled.

* **Story**: As a vacationer, I want to be able to plan a trip to 11 different European cities starting from London.

**Story Point Estimation: 8**

**Assumptions:** We will be assuming that the database holds all of the cities and distances from each other. We will also assume that there is an algorithm that will plan the most economical route to the cities.

**Description:** This will be a capability for the vacationer to plan one trip visiting all of the default cities in the most economical order. The vacationer will be able to see the total distance traveled at the end of the trip.

**Assignee**: Chaz Del Prato

**Priority (1-3): 2**

**Tasks** **& Tests:** Need to make sure that the planning algorithm will be working correctly by testing to see if the route is in the most economical form. Make sure that the total distance is correctly keeping a running sum throughout the trip.

**Definition of Done:** The algorithm should be used to determine the most economical route through all cities from London. The GUI will show a window that will have a table that has the menu of the current city. The vacationer will be able to hit a next button that will bring them to the new city. At the end of the trip it will display the total distance of the trip.

* **Story:** As a vacationer, I want to be able to plan a trip starting from Paris and provide the number of cities I want to visit.

**Story Point Estimation: 8**

**Assumptions:** We will be assuming that the database holds all of the cities and distances from each other.

**Description**: This capability will be able to let the vacationer start from Paris and choose the number of cities the vacationer would want to visit after. After Paris, the program will visit the closest city from there and repeat until the number of places is satisfied.

**Assignee:** Chaz Del Prato

**Priority (1-3): 2**

**Tasks & Tests:** Need to make sure that the planning algorithm will be working correctly by checking to see if the cities are being visited in the most economical order. We will need to check to see that the running sum for the total distance is accumulating correctly. We will want to test for the user invalid input of entering a number of cities that does not exist. We will need to develop an algorithm that will be able to plan a trip with the least amount of distance in between the following city.

**Definition of Done:** The display of this capability will show the vacationer that the trip will start at Paris and then will give the vacationer the option of how many cities they would like to visit after Paris. Then it will display each city in the planned trip and then will display at the end of the trip the total distance that was traveled.

* **Story:** As a vacationer, I want to be able to create my own custom trip, starting from the place of my choice and picking all the cities I want to visit.

**Story Point Estimation: 2**

**Assumptions:** We will be assuming that the database holds all the cities and their distances from each other. There should be an algorithm that will efficiently plan the custom trip. We assume a visible list of cities that the user can choose from.

**Description:** The user will have the capability of picking a custom trip. The user will be able to choose the starting point and then they can choose every city that they would want to go to. After choosing the trip the program will determine the most economical route to the chosen cities.

**Assignee:** Chaz Del Prato

**Priority (1-3): 2**

**Tasks** **& Tests:** Need to make sure that the planning algorithm will be working correctly by testing to see if the route is in the most economical form. Make sure that the total distance is correctly keeping a running sum throughout the trip. Test the program so that it does not duplicate the chosen cities.

**Definition of Done:** A list of cities will be displayed on the GUI and the vacationer will be able to select a starting point and every city they want to visit. Next our planning algorithm will create a trip plan in the most efficient order, least amount of traveling distance, and will display the plan in order from start to finish. The total traveling distance will be displayed at the end.

* **Story:** As an admin, I want to view a diagram of all classes used in the program

**Story Point Estimation:** **2**

**Assumptions:** All classes (minus qt’s built in ui classes) have been planned out from other stories

**Description:** UML diagram shows non-ui classes as well as relationships between classes

**Assignee:** Mohamed Samatar

**Priority (1-3):** 1

**Tasks and Tests:** Diagram all classes, fields, and methods. Display class relationship arrows

**Definition of Done:** Diagram shows all planned non-ui classes with relationships

**Plan for Sprint One:**

* **Story:** As an admin, I want to be able to create a database that holds the starting city, city’s food, and cost of the food.

**Story Point Estimation: 1**

**Assumptions:** The database will use SQLite commands and QT API to fill vectors in order to run algorithms on.

**Description:** The QT API for SQLite has built in functions that allow me to open/close the database and fill the database through a callback function. The purpose of this story is to show that the program is communicating with the database.

**Assignee**: Chase Deric

**Priority (1-3): 1**

**Tasks & Tests:** Make sure that the database is setup correctly. To ensure the database has been successfully connected, we will be outputting the list of cities and food items into the GUI to determine if the database is being read correctly.

**Definition of Done:** The GUI will display a side by side window. On the right side will display the food options and the costs for all European Cities. On the left Side will be the list of European Cities that can be visited. At the bottom of the window, there will be text that states of the database has been loaded successfully.

**Plan for Sprint Two**

* **Story:** As an admin, I want to view a diagram of all classes used in the program

**Story Point Estimation:** **2**

**Assumptions:** All classes (minus QT’s built in UI classes) have been planned out from other stories

**Description:** UML diagram shows non-UI classes as well as relationships between classes

**Assignee:** Mohamed Samatar

**Priority (1-3):** 1

**Tasks and Tests:** Diagram all classes, fields, and methods. Display class relationship arrows

**Definition of Done:** Diagram shows all planned non-UI classes with relationships.

* **Story:** As a vacationer, I want to be able to view all the traditional foods offered and its cost in any given European city.

**Story Point Estimation: 1**

**Assumptions:** The foods and their costs are already stored and organized in a data structure.

**Description:** Foods and their respective costs will be contained within a data structure and have their attributes, food name and cost, output through established methods.

**Assignee**: Cody Thompson

**Priority (1-3): 1**

**Tasks & Tests:** Ensure that the foods and their costs have been assigned to the proper city. Output the attributes and reference the data given on the cities to check accuracy.

**Definition of Done:** The GUI will have a combo box that will have all the names of the cities that can be chosen. Once you select what city to view, you hit go and it will display the menu items with name and cost into a table.

* **Story**: As a vacationer, I want to be able to plan a trip to 11 different European cities starting from London.

**Story Point Estimation: 8**

**Assumptions:** We will be assuming that the database holds all of the cities and distances from each other. We will also assume that there is an algorithm that will plan the most economical route to the cities.

**Description:** This will be a capability for the vacationer to plan one trip visiting all of the default cities in the most economical order. The vacationer will be able to see the total distance traveled at the end of the trip.

**Assignee**: Chaz Del Prato

**Priority (1-3): 2**

**Tasks** **& Tests:** Need to make sure that the planning algorithm will be working correctly by testing to see if the route is in the most economical form. Make sure that the total distance is correctly keeping a running sum throughout the trip.

**Definition of Done:** The algorithm should be used to determine the most economical route through all cities from London. The GUI will show a window that will have a table that has the menu of the current city. The vacationer will be able to hit a next button that will bring them to the new city. At the end of the trip it will display the total distance of the trip.

* **Story:** As a vacationer, I want to be able to view a list of European cities and their distances from Berlin.

**Story Point Estimation: 2**

**Assumptions:** All European cities and their distances from Berlin are already stored and organized in a data structure.

**Description:** Every European city available for the trip will be have their city name and distance from Berlin recorded into a City object. All cities will be contained within a data structure and output through an established method.

**Assignee**: Cody Thompson

**Priority (1-3): 1**

**Tasks & Tests:** Ensure that each city name has been assigned to the proper distance from Berlin. Output the attributes of the City object and reference the data given on the cities to check accuracy.

**Definition of Done:** If the vacationer selects the city from the combo box and hits go it will display the city the vacationer can go to and the distances to each into a table.

* **Story:** As an admin, I want to be able to change the cost of food as the vacationer learns the reality of the cost of food in the given city.

**Story Point Estimation: 13**

**Assumptions:** The database will use SQL commands and C++ API to change the data structure in order to better reflect the vacationers experience.

**Description:** The C++ API for SQLite has built in functions that allow me to change any given table inside the database. Furthermore, I can alter columns by using the alter table command.

**Assignee**: Chase Deric

**Priority (1-3): 2**

**Tasks & Tests:** Make sure that the database reflects the changes that I made to it. For testing, output the food\_costs table where I changed the cost and determine if it matches the true cost of food.

**Definition of Done:** An admin page will be added to the GUI which will display all cities, their food, and their costs. The admin will be able to modify information about the cities from this page. The database will be updated and redisplayed on the page to reflect these changes made.

**Style Standard:**

* Use alignment wherever it enhances readability.
* Goto should not be used.
* The use of magic numbers in the code should be avoided. Numbers other than 0 and 1should be considered declared as named constants instead.
* The conditional should be put on a separate line.
* The form while(true) should be used for infinite loops.
* Variables should be initialized where they are declared.
* Exception classes should be suffixed with *Exception*.
  + class AccessException
* Variables representing GUI components should be suffixed by the component type name.
  + mainWindow, propertiesDialog, widthScale, loginText
* Names representing methods or functions must be verbs and written in mixed case starting with lowercase.
  + getName(), computeTotalWidth()
* Named constants (including enumeration values) must be all uppercase using underscore to separate words.
  + MAX\_ITERATIONS, COLOR\_RED, PI
* Variable names must be in mixed case starting with lowercase.
  + line, savingsAccount

**Team Rules:**

1. Be on time
2. Come prepared to lab meetings
3. Mandatory work times: MW 5:00-6:20pm
4. Additional work times: Weekends
5. C++ language
6. Same style
7. Adhere to Agile guidelines
8. No slacking off
9. Flexibility with software changes
10. Each team member should know every aspect of our program
11. Be a reliable teammate
12. Help teammates if one does not have a full understanding
13. Team members must communicate in an efficient, reliable manner
14. Meet all our given deadlines (No exceptions)
15. NO PROCRASTINATING