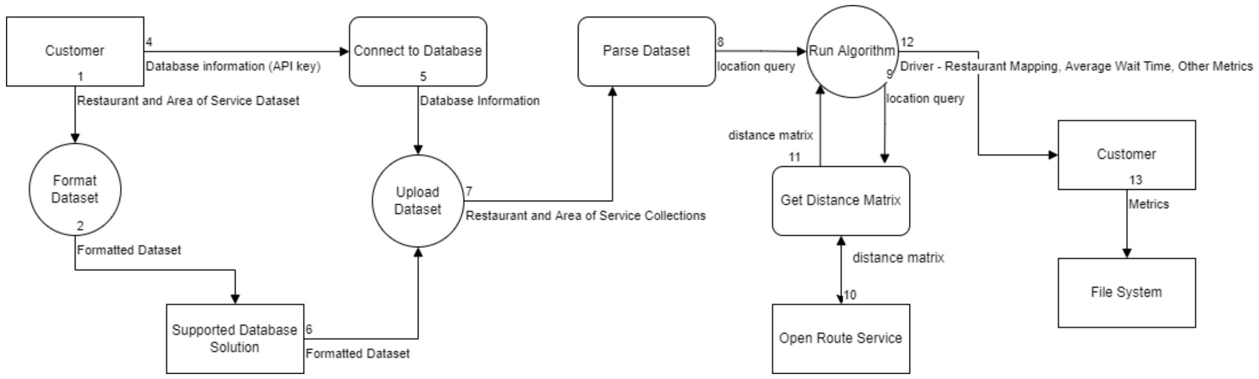
Requirements engineering

- Rob created a requirements table to reflect the current state of the requirements after the specification changes.

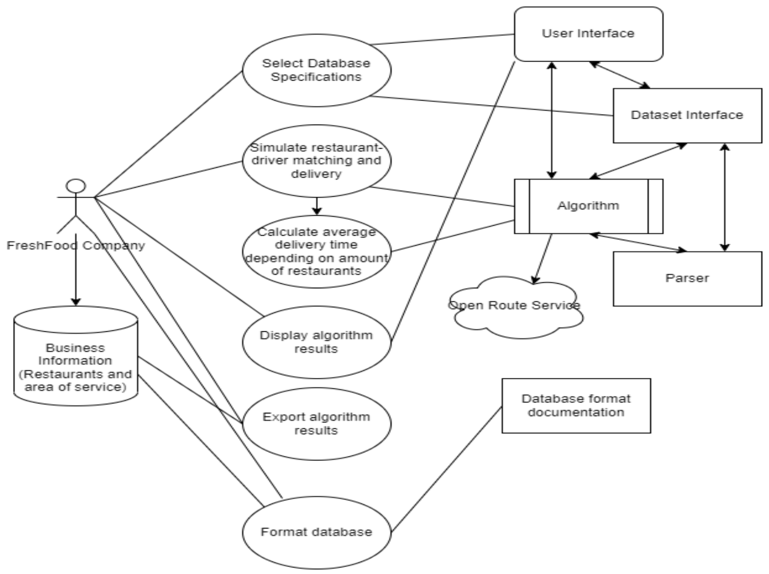
Functional Requirements	Non Functional Requriements
Must allow input of a database of locations of restaurants and points within the area of service.	Must have a document instructing users on how to format their restaurant and area of service database.
Must now allow databases with less than 25 restaurants.	Must be easy to use.
Must assume that a maximum of 1000 orders will be served at any given time.	Must display data graphically.
Should calculate the average wait time per order.	Must be able to output data to a local file system.
Should output a mapping of restaurant orders and drivers that achieve the target average wait time.	
Must have an interface for uploading a database or inputting database connection credentials.	
Must include middleware for interfacing with routing service API	
Must have a document with instructions on formatting the company dataset.	

System Modeling

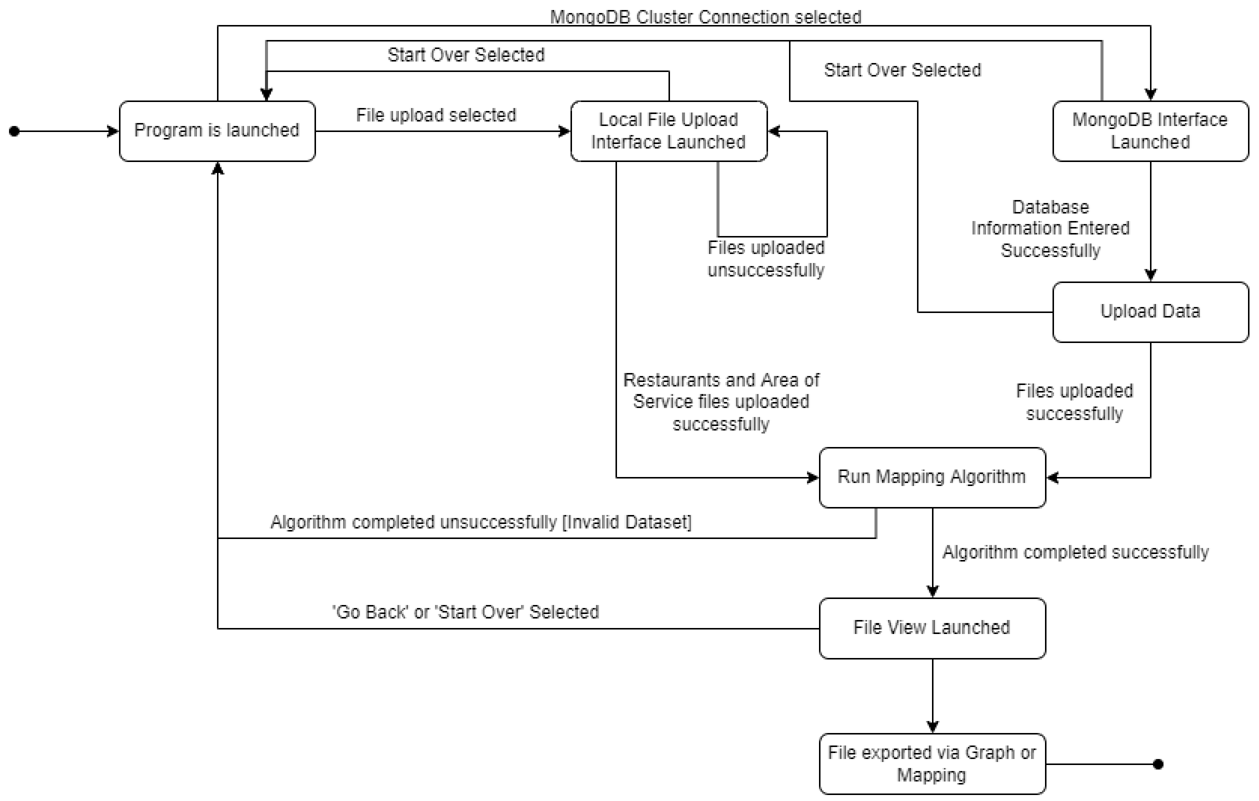
- Rob created a data flow diagram which illustrates how the input data is processed into business metrics



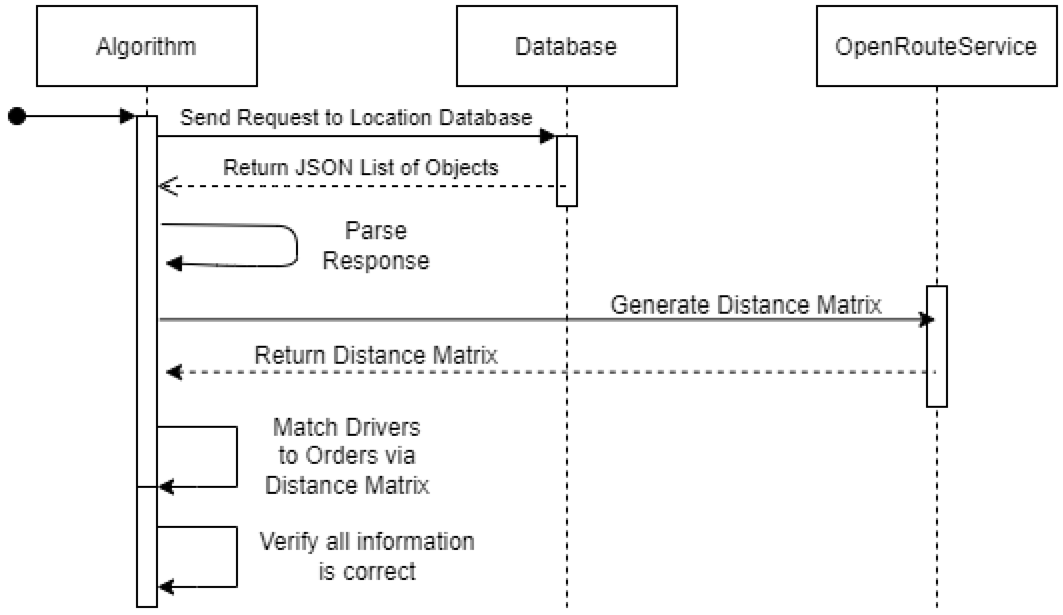
- Rob updated the use case diagram to reflect the changes made in the user interface which allows the user to specify the operation performed by the program


User Interface
Select Database Specifications
Dataset Interface
Simulate restaurant-driver matching and delivery
Algorithm
FreshFood Company
Calculate average delivery time depending on amount of restaurants
Parser
Open Route Service
Business Information (Restaurants and area of service)
Display algorithm results
Export algorithm results
Database format documentation
Format database

- Kiernan created State Diagram representing FreshTime

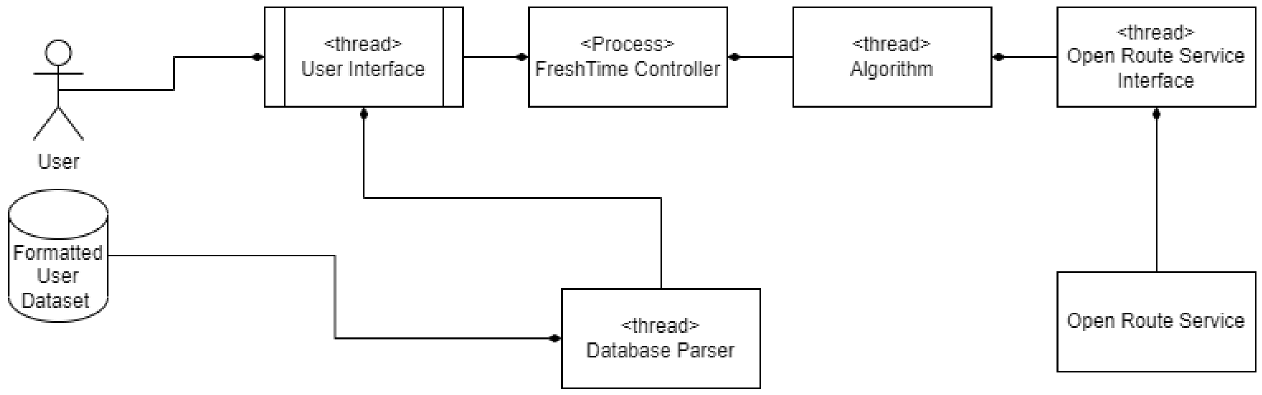

MongoDB Cluster Connection selected
Start Over Selected
Start Over Selected
Program is launched
File upload selected
Local File Upload Interface Launched
MongoDB Interface Launched
Files uploaded unsuccessfully
Database Information Entered Successfully
Upload Data
Restaurants and Area of Service files uploaded successfully
Files uploaded successfully
Run Mapping Algorithm
Algorithm completed unsuccessfully [Invalid Dataset]
Algorithm completed successfully
'Go Back' or 'Start Over' Selected
File View Launched
File exported via Graph or Mapping

- Kiernan generated Sequence Diagram


Algorithm
Database
OpenRouteService
Send Request to Location Database
Return JSON List of Objects
Parse Response
Generate Distance Matrix
Return Distance Matrix
Match Drivers to Orders via Distance Matrix
Verify all information is correct

Architectural Design

- Rob created a process view for the FreshTime project architecture

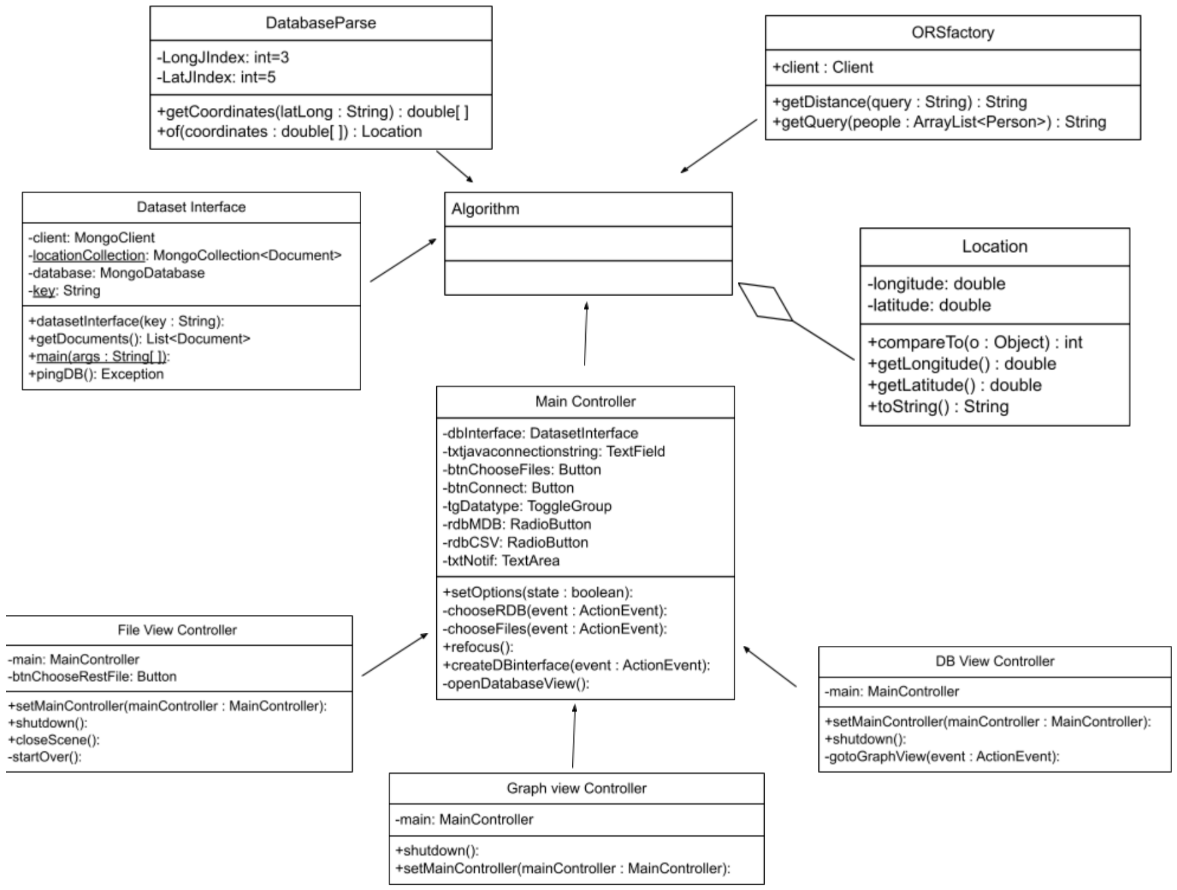


This shows how the system is made up of individual processes that spawn threads which perform a specific task.

Design Specification

- Rob updated Software Specification documents to reflect changes in the functional and nonfunctional requirements.
    - Identified supported databases, mongoDB and local file upload.
- Kiernan finalized the Database Import Specification Document in tandem with our chosen database architecture.
- Kiernan updated the current Javadoc comments for all classes.
- Kiernan generated Javadoc and uploaded it to Github.
- Kiernan drafted a Database Format Document.

- Wiktor created a class diagram

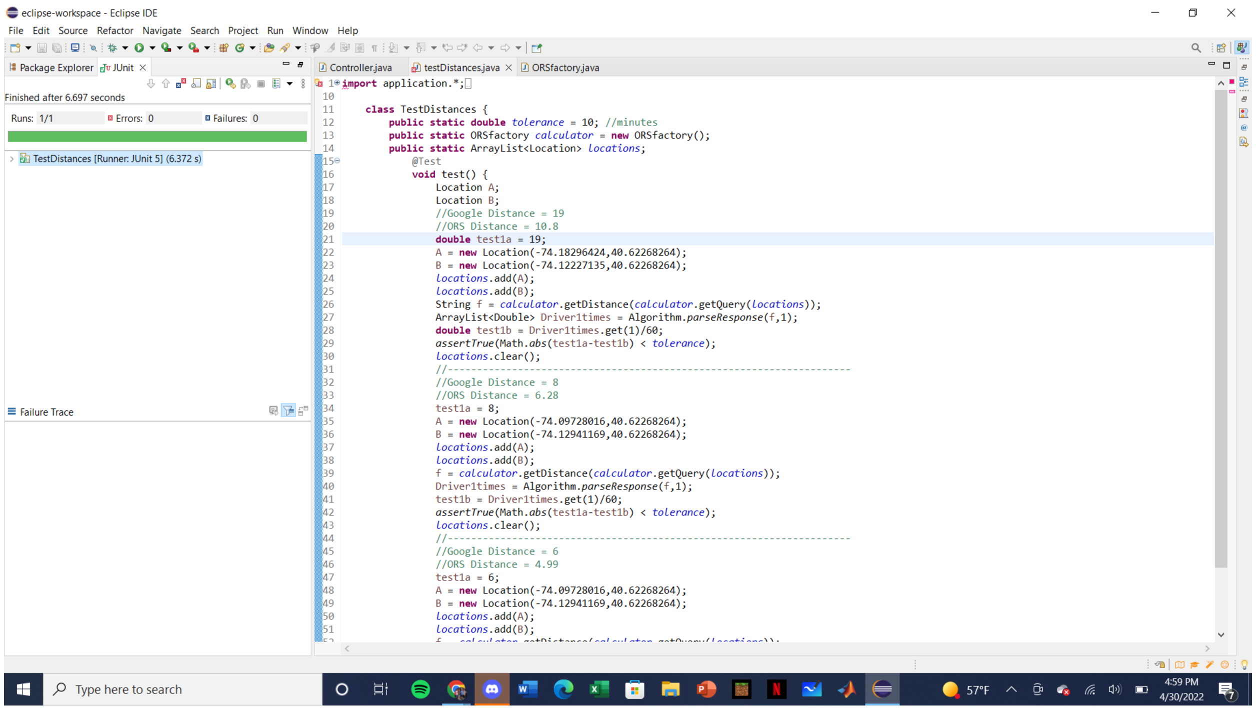

DatabaseParse
-LongJIndex: int=3
-LatJIndex: int=5
+getCoordinates(latLong : String) : double[ ]
+of(coordinates : double[ ]) : Location
ORSfactory
+client : Client
+getDistance(query : String) : String
+getQuery(people : ArrayList<Person>) : String
Dataset Interface
-client: MongoClient
-locationCollection: MongoCollection<Document>
-database: MongoDatabase
-key: String
+datasetInterface(key : String):
+getDocuments(): List<Document>
+main(args : String[ ]):
+pingDB(): Exception
Algorithm
Location
-longitude: double
-latitude: double
+compareTo(o : Object) : int
+getLongitude() : double
+getLatitude() : double
+toString() : String
Main Controller
-dbInterface: DatasetInterface
-txtjavaconnectionstring: TextField
-btnChooseFiles: Button
-btnConnect: Button
-tgDatatype: ToggleGroup
-rdbMDB: RadioButton
-rdbCSV: RadioButton
-txtNotif: TextArea
+setOptions(state : boolean):
-chooseRDB(event : ActionEvent):
-chooseFiles(event : ActionEvent):
+refocus():
+createDBinterface(event : ActionEvent):
-openDatabaseView():
File View Controller
-main: MainController
-btnChooseRestFile: Button
+setMainController(mainController : MainController):
+shutdown():
+closeScene():
-startOver():
DB View Controller
-main: MainController
+setMainController(mainController : MainController):
+shutdown():
-gotoGraphView(event : ActionEvent):
Graph view Controller
-main: MainController
+shutdown():
+setMainController(mainController : MainController):

Design Implementation

- Kiernan uploaded a location database file (.csv) into our MongoDB database.
- Kiernan updated Interface, Algorithm and Parsing methods with code that better represents our project requirements.
- Rob added to the FreshTime project designing the user input checking system for valid symbols.

Software Testing (across all levels)

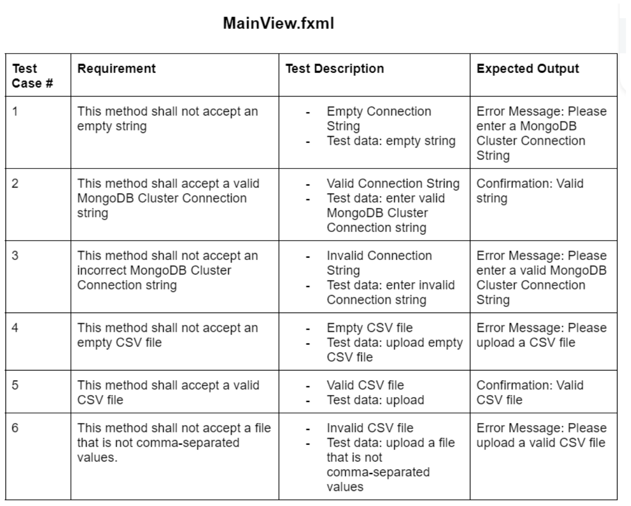
- Dan created a spreadsheet to compare the data returned from the ORSfactory class distance matrix times returned versus that of google maps.
- Dan drafted Junit tests which tested Rob’s getDistance method of the ORSfactory class which passed.



```
Finished after 6.697 seconds
Runs: 1/1    Errors: 0    Failures: 0
TestDistances [Runner: JUnit 5] (6.372 s)
```

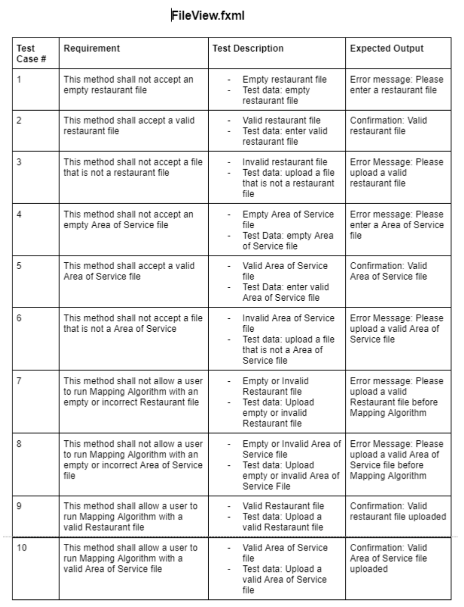
```java
import application.*;

    class TestDistances {
        public static double tolerance = 10; //minutes
        public static ORSfactory calculator = new ORSfactory();
        public static ArrayList<Location> locations;
            @Test
            void test() {
                Location A;
                Location B;
                //Google Distance = 19
                //ORS Distance = 10.8
                double test1a = 19;
                A = new Location(-74.18296424,40.62268264);
                B = new Location(-74.12227135,40.62268264);
                locations.add(A);
                locations.add(B);
                String f = calculator.getDistance(calculator.getQuery(locations));
                ArrayList<Double> Driver1times = Algorithm.parseResponse(f,1);
                double test1b = Driver1times.get(1)/60;
                assertTrue(Math.abs(test1a-test1b) < tolerance);
                locations.clear();
                //--------------------------------------------------------------------
                //Google Distance = 8
                //ORS Distance = 6.28
                test1a = 8;
                A = new Location(-74.09728016,40.62268264);
                B = new Location(-74.12941169,40.62268264);
                locations.add(A);
                locations.add(B);
                f = calculator.getDistance(calculator.getQuery(locations));
                Driver1times = Algorithm.parseResponse(f,1);
                test1b = Driver1times.get(1)/60;
                assertTrue(Math.abs(test1a-test1b) < tolerance);
                locations.clear();
                //--------------------------------------------------------------------
                //Google Distance = 6
                //ORS Distance = 4.99
                test1a = 6;
                A = new Location(-74.09728016,40.62268264);
                B = new Location(-74.12941169,40.62268264);
                locations.add(A);
                locations.add(B);
```

- Dan Created a document to test the GUI component of FreshTime.



MainView.fxml

Test Case #	Requirement	Test Description	Expected Output
1	This method shall not accept an empty string	- Empty Connection String - Test data: empty string	Error Message: Please enter a MongoDB Cluster Connection String
2	This method shall accept a valid MongoDB Cluster Connection string	- Valid Connection String - Test data: enter valid MongoDB Cluster Connection string	Confirmation: Valid string
3	This method shall not accept an incorrect MongoDB Cluster Connection string	- Invalid Connection String - Test data: enter invalid Connection string	Error Message: Please enter a valid MongoDB Cluster Connection String
4	This method shall not accept an empty CSV file	- Empty CSV file - Test data: upload empty CSV file	Error Message: Please upload a CSV file
5	This method shall accept a valid CSV file	- Valid CSV file - Test data: upload	Confirmation: Valid CSV file
6	This method shall not accept a file that is not comma-separated values.	- Invalid CSV file - Test data: upload a file that is not comma-separated values	Error Message: Please upload a valid CSV file



FileView.fxml

Test Case #	Requirement	Test Description	Expected Output
1	This method shall not accept an empty restaurant file	- Empty restaurant file - Test data: empty restaurant file	Error message: Please enter a restaurant file
2	This method shall accept a valid restaurant file	- Valid restaurant file - Test data: enter valid restaurant file	Confirmation: Valid restaurant file
3	This method shall not accept a file that is not a restaurant file	- Invalid restaurant file - Test data: upload a file that is not a restaurant file	Error Message: Please upload a valid restaurant file
4	This method shall not accept an empty Area of Service file	- Empty Area of Service file - Test Data: empty Area of Service file	Error message: Please enter a Area of Service file
5	This method shall accept a valid Area of Service file	- Valid Area of Service file - Test Data: enter valid Area of Service file	Confirmation: Valid Area of Service file
6	This method shall not accept a file that is not a Area of Service	- Invalid Area of Service file - Test data: upload a file that is not a Area of Service file	Error Message: Please upload a valid Area of Service file
7	This method shall not allow a user to run Mapping Algorithm with an empty or incorrect Restaurant file	- Empty or Invalid Restaurant file - Test data: Upload empty or invalid Restaurant file	Error message: Please upload a valid Restaurant file before Mapping Algorithm
8	This method shall not allow a user to run Mapping Algorithm with an empty or incorrect Area of Service file	- Empty or Invalid Area of Service file - Test data: Upload empty or invalid Area of Service File	Error Message: Please upload a valid Area of Service file before Mapping Algorithm
9	This method shall allow a user to run Mapping Algorithm with a valid Restaurant file	- Valid Restaurant file - Test data: Upload a valid Restaraunt file	Confirmation: Valid restaurant file uploaded
10	This method shall allow a user to run Mapping Algorithm with a valid Area of Service file	- Valid Area of Service file - Test data: Upload a valid Area of Service file	Confirmation: Valid Area of Service file uploaded

Dependable Attribute

Reliability - Brian has laid out plans to test all possible states of the GUI next week. The week after we have plans to test our Delivery Time Algorithm against valid and invalid conditions and inputs. These are indicated in the reliability plan document in the sprint 5 folder. These plans are intended to ensure the reliability of our program by having a formal code review process which stems from the initial system requirements and ensures they will be fulfilled and provide correct service to the user.

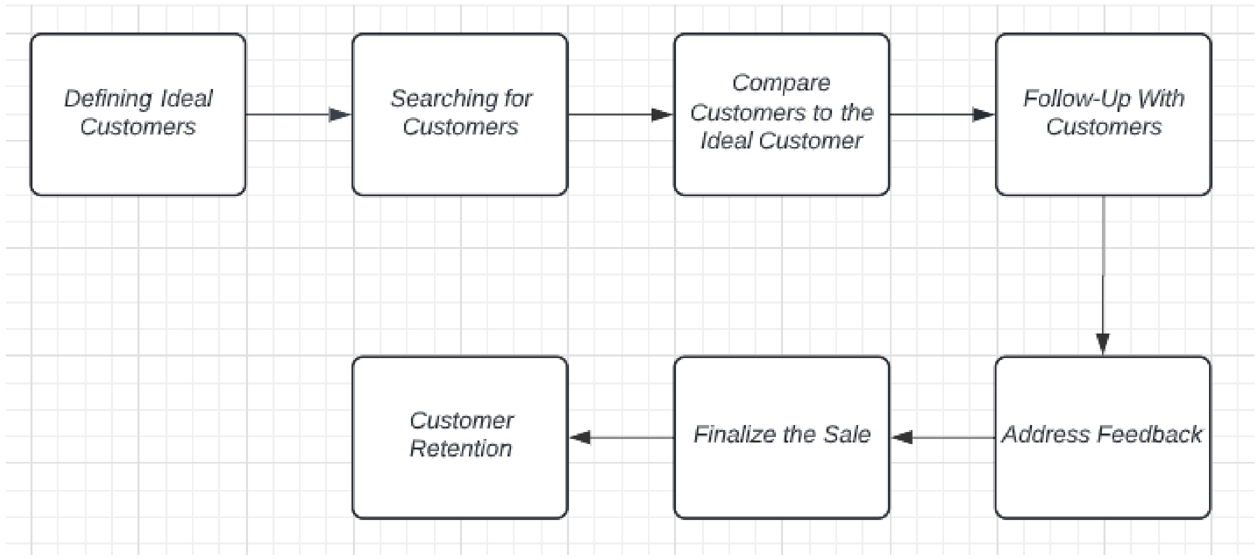
Project Management Concept

- Brian drafted a Software Quality Assurance Plan detailing the future development of the project.
    - SQA Plan_04/30/22

Advanced SWE topics

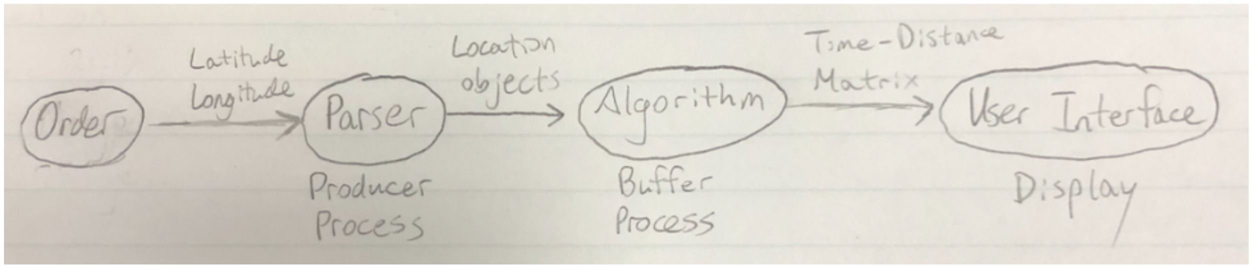
Distributed Software Engineering - Software as a Service (SaaS)

- Brian conducted research on the future possibility of developing a web based version of our software to offer as a service and then created the SaaS Customer Interaction Plan FURTHER DETAILS in the "Software as a Service Plan" Document



Real-time Software Engineering System Analysis

- Wiktor researched and created the process pipeline diagram and a real time system analysis document that outlines the analysis of time sensitive elements in our system.


Order
Latitude
Longitude
Parser
Producer Process
Location objects
Algorithm
Buffer Process
Time-Distance Matrix
User Interface
Display