# **Technical Manual for Distance and Directions with Google Maps**

# By Greg Bayne, Kari Franklin, Michael Booser

1. **Overview**

This document describes how a system must be setup in order to be able to run the Google Maps program.

1. **System Requirements**

* Java version 1.4.0
* Memory 512 MB
* Free disk space 300 MB
* Processor speed 800 Mhz

1. **Assumptions**

* It is assumed the user has met all the minimum system requirements.
* It is assumed the user has a stable internet connection.

1. **Installing Eclipse**

**4.1 Installing Eclipse**

* Please reference the user manual for a step-by step guide to installing eclipse

1. **Running Distance and Directions Application**

**5.1 Opening Project and Launching Springboot Application**

* Open the project in eclipse as a maven project and it should automatically import all dependencies. Find the class “DistanceAndControllerApplication” and run it.
* For a step-by-step guide, reference the user manual

1. **Google Maps Classes**

**CreateMap.java**- This class builds URL with given address data (Origin and Destination) to create static map with google API. The URL includes markers for start and end locations.

**DatabaseQuery.java**- Queries coordinate data from provided ID.

**DistanceAndDirectionsApplications.java**- Application class to launch website on localhost:8080

**GeoGrabber.java**- Handles geocoding and reverse-geocoding events.

**DistanceAndDirectionsController.java**- The controller class for handling interactions between the SeachRepository (CRUD Repository) and corresponding “search” entities.

**PolyCluster.java**- Used to organize each set of long/lat coordinates into clusters.

**Search.java**- Depicting object model fro query data stored in the H2 database.

**SearchRepository.java**- Repository implementation extending from CRUD repository.

**DataStore.java**- Holds data for map queries, stores the starting and ending locations as well as the distance and DirectionsHolder.

**DirectionsAPI.java**- Generates a request for the Google Directions API and parses the response.

**DirectionsHolder.java**- A class that holds parsed information from Google Directions API including the directions summary.

**DistanceMatrixAPI.java**- Used for obtaining distance using Google Maps API.

**API\_Directions.java**- Generates a request for the google directions API and parses the response.

**API\_DistanceMatrix.java**- Used for obtaining distance using Google Maps API.

**API\_GeoCode.java**- This class handles geocoding and reverse-geocoding events.

**DataStore.java (GoogleAPIRoadsData)**- Holds data for map queries, stores the staring the ending locations as well as the distance and DirectionsHolder.

**DirectionsHolder.java**- Holds parsed information from Google Directions API including the directions summary.

**KEYS.java**- Used for accessing and editing API keys in GoogleAPIRoadsData.

**DataController.java**- Runs the xlsx parser and adds its data to a Table object.

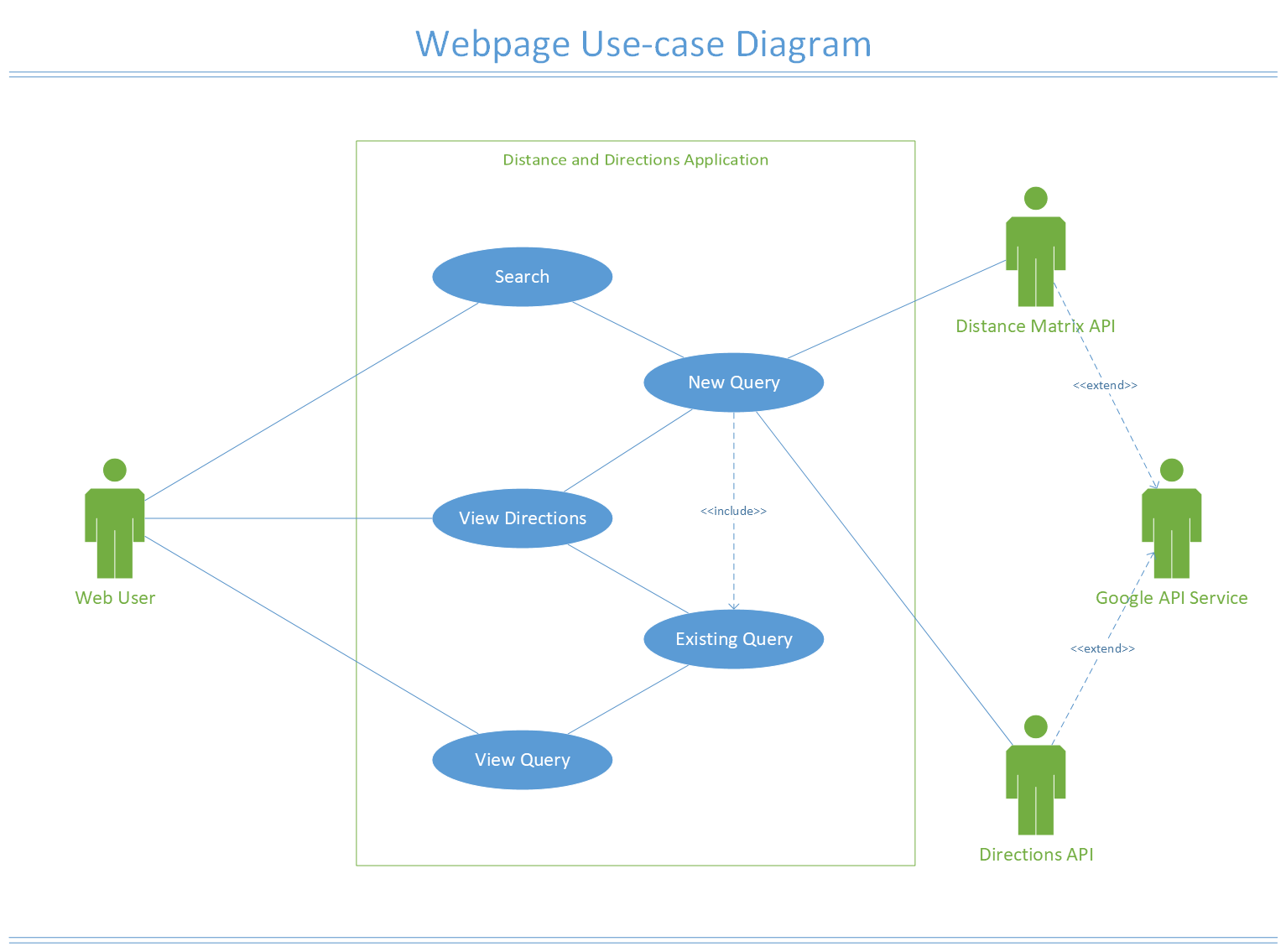
**DataObject.java**- It is a container for all the data to be stored.

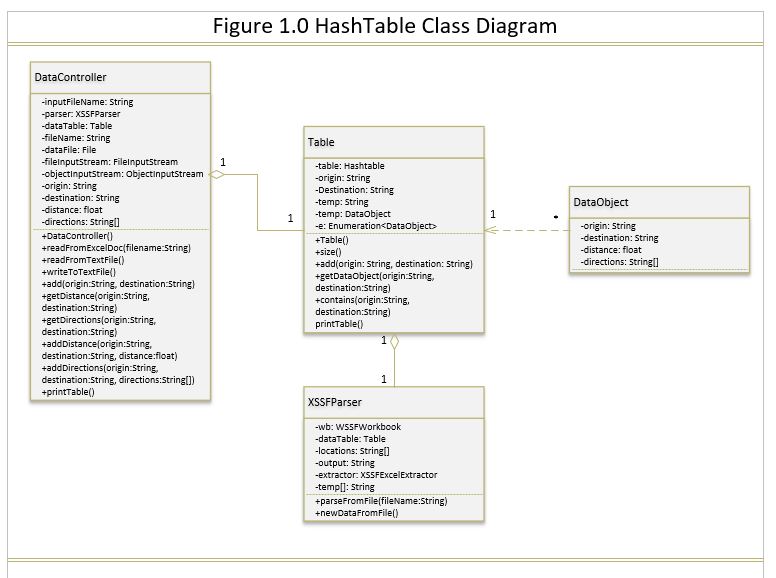
**Table.java**- Used to make an instance of a Hashtable. This holds the data structure that we are using to store data.

**XSSFParser.java**- Uses a third party library called Apache POI that is used to manipulate .xlsx files.

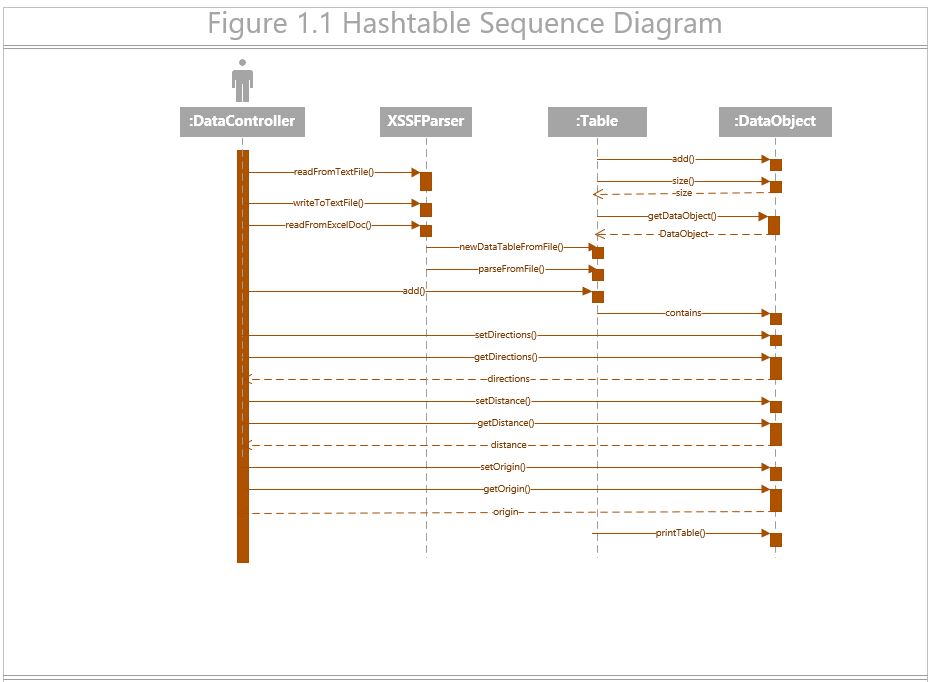
1. **UML Diagrams on functionality of Google Maps**

The following UML diagrams will show the interaction between different classes, how they are being used, and their functionality together as a whole.

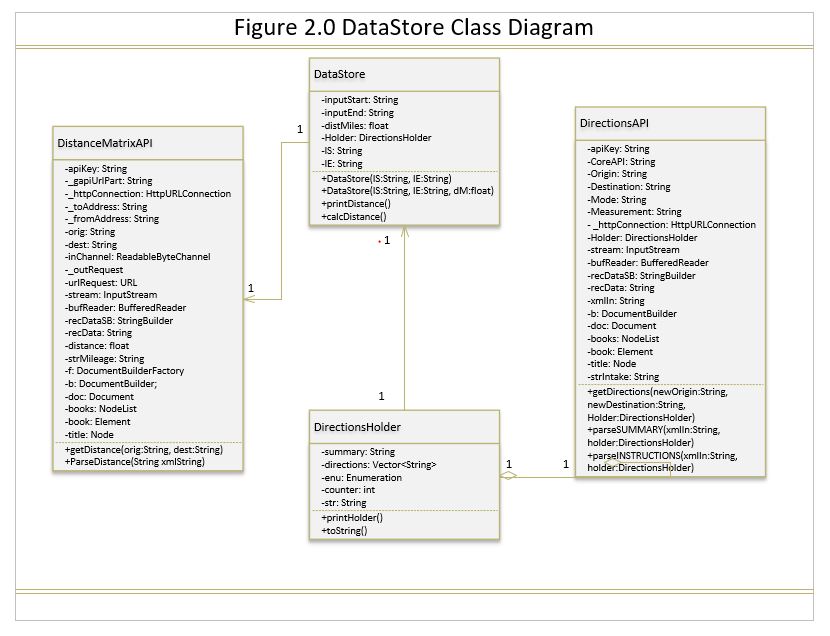
This diagram shows the use case relation between the webuser and the Google API

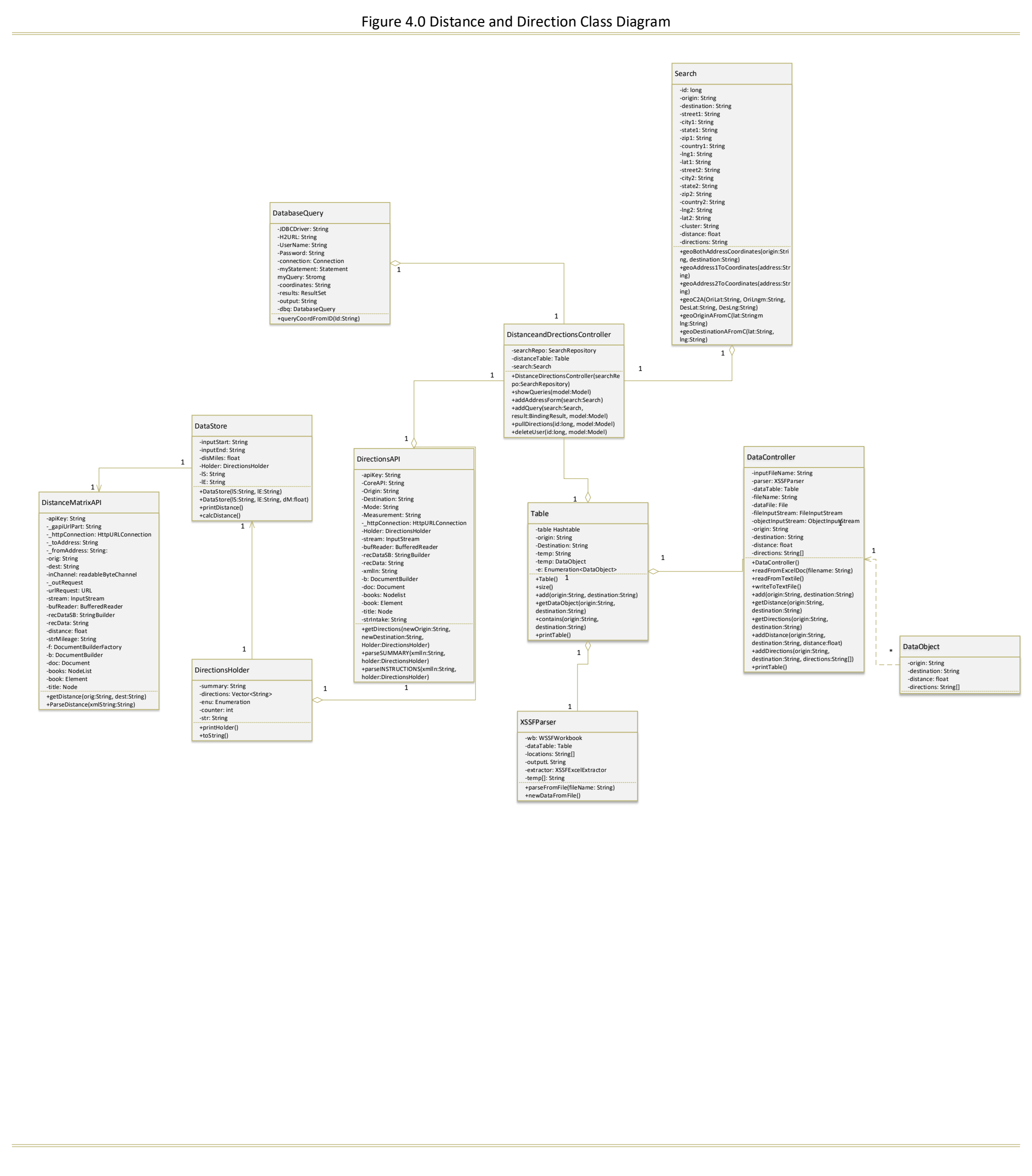


This UML class diagram shows the interactions of classes that are used to create and use the HashTable data structure. The Hashtable is what is used to load and save data for the project. There is a Table class that sets up the table, an XSSFParser that allows reading and writing to a text file. A DataObject that makes getters and setters for the Hashtable info and finally a DataController class that is used to implement all these classes together. Lastly, this shows the associations between these classes.

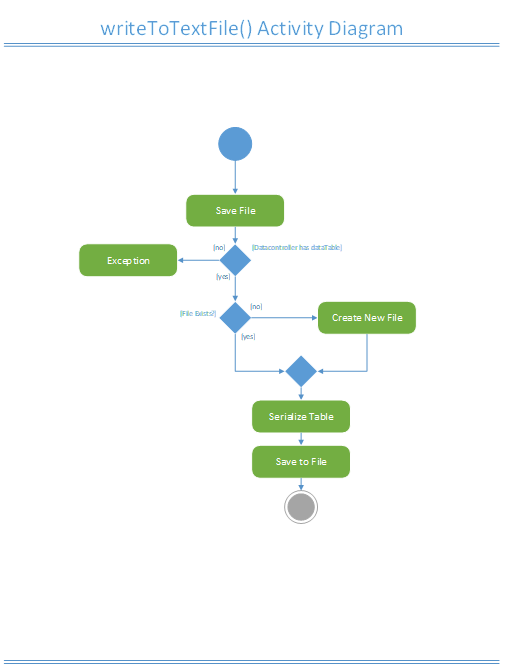


This UML sequence diagram shows the sequence of events existing between these four classes. This displays all the methods that are being used in each class as well as where they are being used and how they’re being used based on each class interaction. This diagram shows the sequence of information being used for the HashTable.

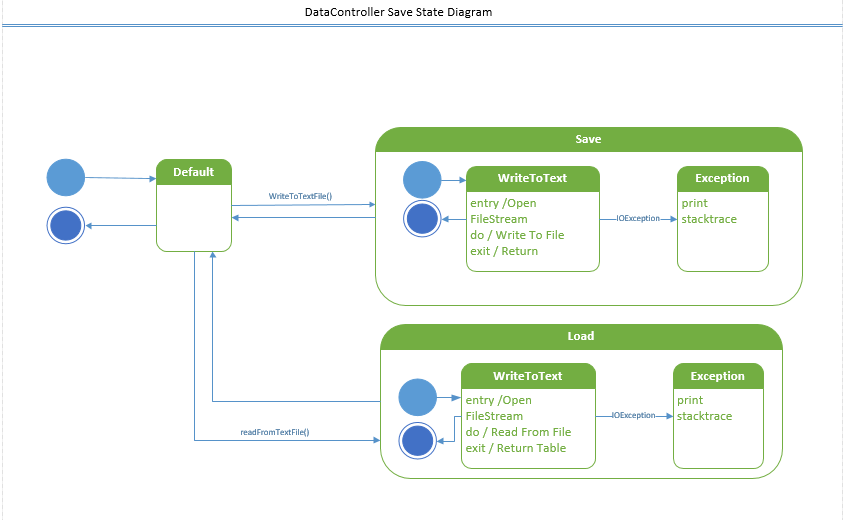
This is a class diagram describing the interaction between the two API and how they enter the datastore object



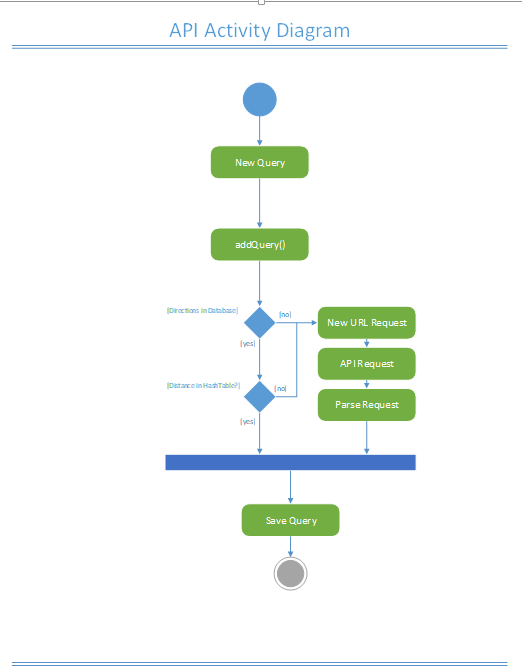
This UML class diagram shows the interactions of all the important classes in the program.



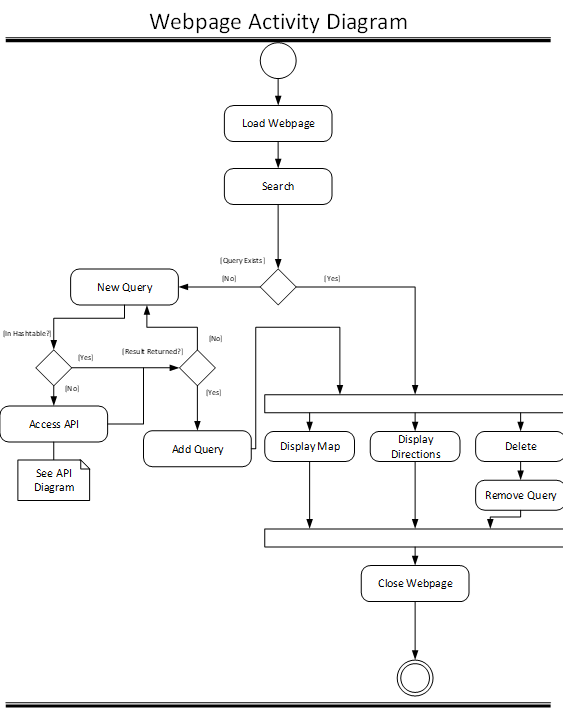
This is an activity UML diagrams that shows the different states involved in writing to a text file as well as the different steps towards saving it.



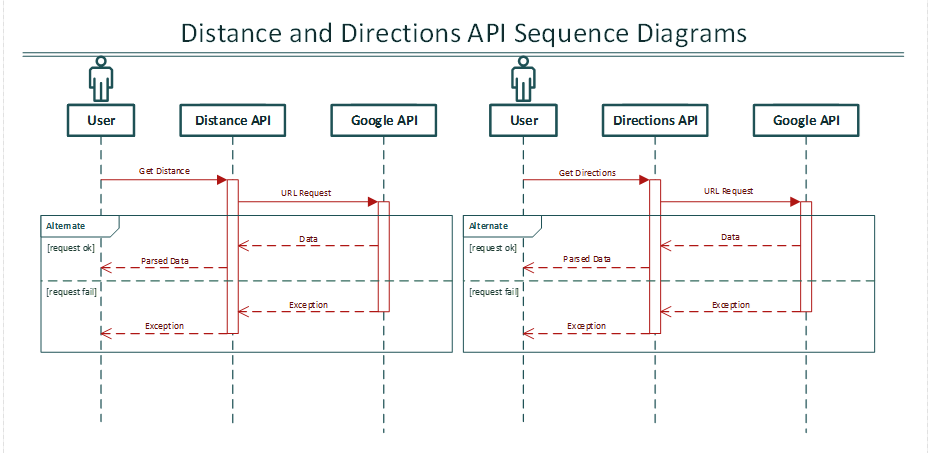
This is a state UML diagram that shows how the loading and saving works in the DataController class for the program.



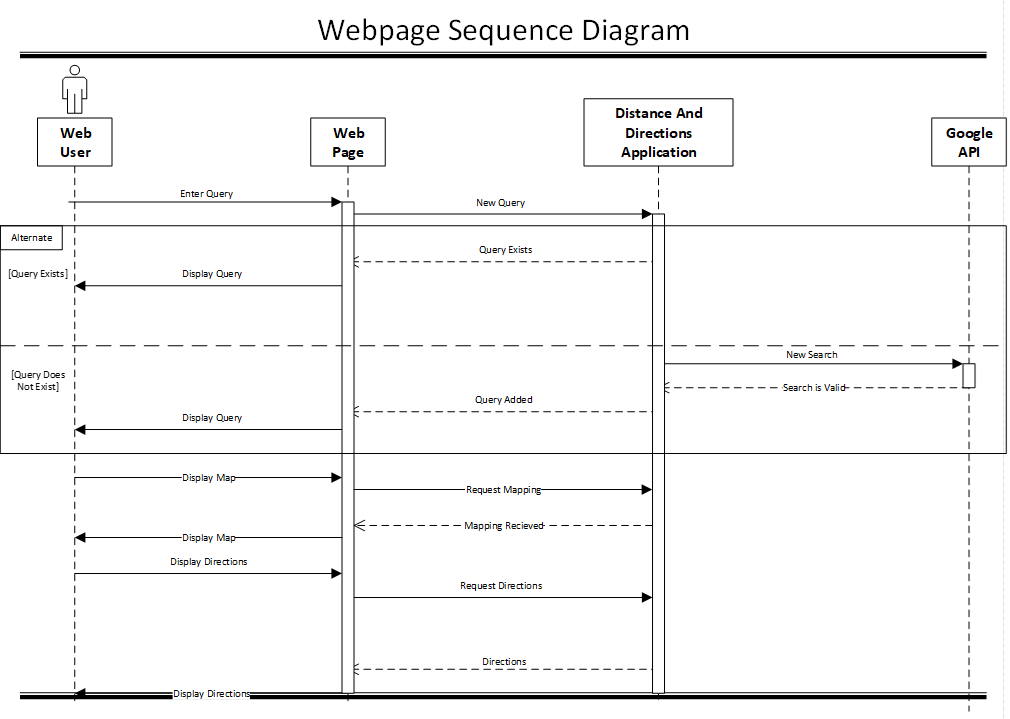
This is a activity UML diagram that shows the steps towards creating a new query and saving the query.



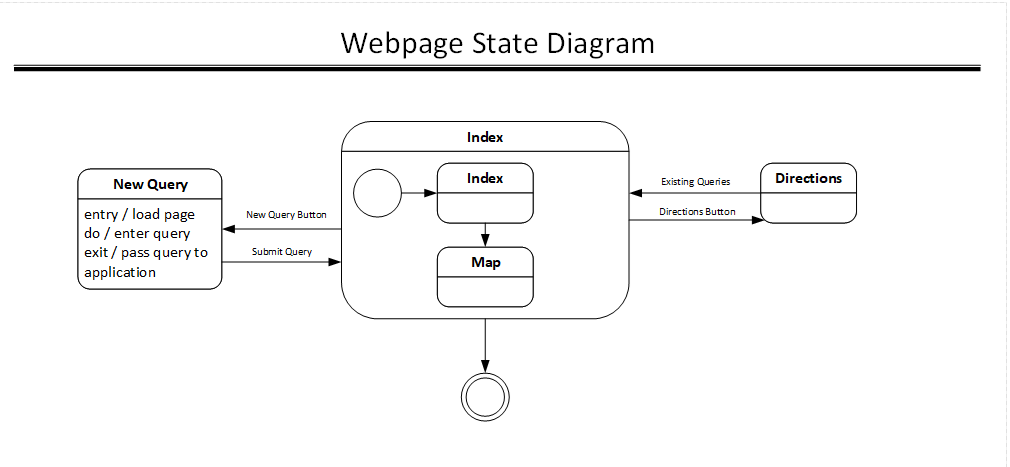
This is an activity UML diagram that shows the different flow of events for the webpage.



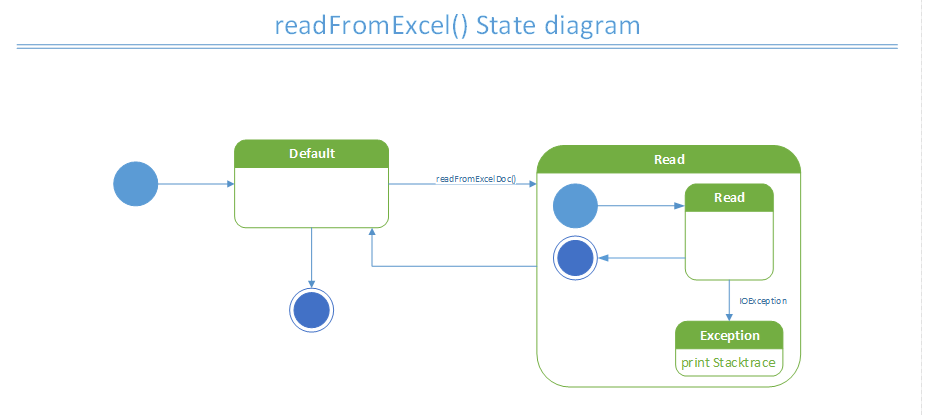
This is a UML sequence diagram that shows the parsing of data methods involved with the DirectionsAPI and the DistanceAPI classes.



This is a sequence UML diagram depicting the sequence of events involved with the operation of the webpage showing the different methods used.



This is a State UML diagram that shows the different states of the webpage and how it’s working.



This is a state UML diagram that is depicting how the readFromExcel() function is working.