Traffic Monitoring Project

Group #13
Xuan Li
Chih-Ting Cho
Ting-Chieh Huang
Jonathan Hong
Jan Racoma
Kevin Cundey

Software Engineering 14:332:452
First Report

https://sites.google.com/site/452trafficmonitoring/home

<u>Individual Contributions Breakdown</u>

All team members contributed equally.

Table of Contents

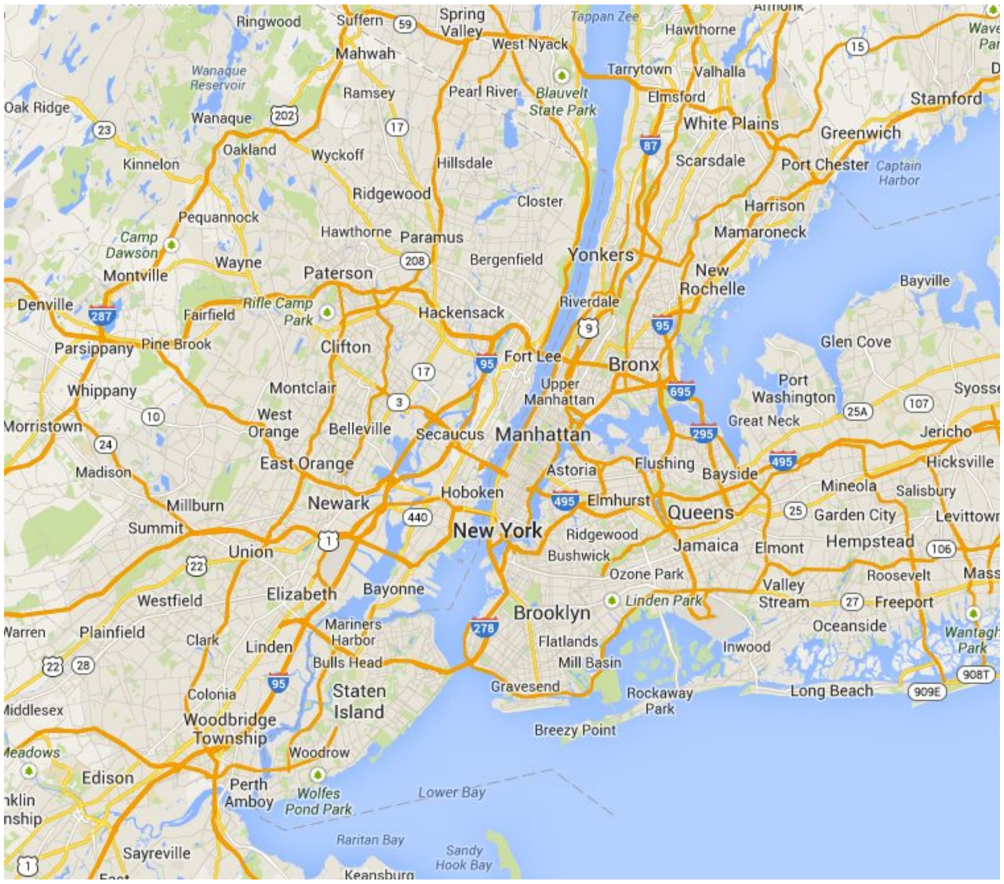
Customer Statement of Requirements

Traffic is a daily commuter's worst enemy. Most commuters are only armed with their familiarity of their commute and a GPS-bad device. Most traffic information services only provide current information about traffic conditions in a given area. Although current information is valuable, the main caveat is that these services provide incomplete reports. For example, if a route is notoriously known for traffic congestion or accidents, drivers would know to avoid taking that route. However, with live traffic monitoring systems, incidents are only reported after they occur; thus, they are unable to show how traffic may develop as the day progresses. In some cases, traffic conditions at a certain location may even go unreported.

Along with congestion and accidents, weather is rarely indicated along with traffic but can also be a determinant to traffic conditions. As a result, one cannot make an assumption that traffic is minimal on his or her intended route. Therefore, there is an apparent need for a software system that aims to provide real-time feedback on traffic conditions using historical traffic data, current weather conditions, and time of day. With this software, commuters will be able to plan out the most efficient route and arrive at their destination without any issues.

Current services provide users with a couple of route choices and estimated travel time based on current traffic conditions, but none of them provide predictions with a certain degree of probability. It is true that no one can predict the future, but with the assistance of statistics, trends may be discovered through analytics. By analyzing this data, predictions on future traffic conditions can be made and suggestions on a new route, when necessary, will be systematically provided to users. The data will highlight known problem areas during certain time periods of the day and weather conditions. Furthermore, it will be able to show when rush hour is likely to occur on major roadways. In addition to displaying historical traffic trends, construction schedules will be integrated with traffic predictions on the intended routes. If construction will be present on the intended route, the software can take this information into consideration into making routing decisions. With this information, users can then decide which route to take in order to minimize their time of travel and decrease the chances of encountering traffic. This software will simplify the decision making by automating as much of the process as possible.

Traffic and weather information will be collected hourly and imported into a database system. Additionally, scheduled constructions will also be collected and stored in the same manner. This collected information should be reliable, such as Google, MapQuest, or Weather.com. Collecting information hourly will allow ample time between collections for minor incidents to be cleared yet major incidents will remain and thus be made more apparent. As proposed by previous iterations, expanding the service outside of New Jersey is essential. Our primary focus will be the New York Metropolitan Area, which includes New York, New Jersey, and Connecticut. Traffic is much more prevalent for residents in this area due to their close proximity to New York City. Traffic on major highways, freeways, bridges, and tunnels will be the service's main concern; however, it is subject to change. Currently, we do not anticipate integrating local roads in this service. In general, because there is a low volume of vehicles on local roads, they are usually less congested than major roadways. Therefore, we will assume the lowest level of traffic on local roads. This assumption will be verified throughout the course of the project, but the principles behind the assumption seem plausible. For example, the Garden State Parkway, New Jersey Turnpike, George Washington Bridge, and Lincoln Tunnel are among the busiest roadways during rush hour. For users of this software, knowing the traffic trends of these busy areas will allow them to be prepared beforehand.



Also, many GPS and traffic systems require a purchase of specific hardware and software. This software is set with the goal of being accessible anywhere an internet connection on a computing device is available. The software should then be easily accessible through a computer, laptop, tablet, phone, and other similar type portable devices through a web browser. This will allow the system to be very portable and convenient to use for everybody with any of the required devices.

The program may be used as a planning tool prior to travelling, whether the night before or before heading out the door. Users will input a source and destination. The system will then take the points and determine possible routes of travel. Next, the system will take the possible routes and determine possibilities of incidents and presence of construction occurring on the displayed routes. The system will then present the route with the least amount of travel time. Users will have the luxury of having the system send an alert of current or probable delays and notifying the user to leave by a certain time in order to arrive at their destination on time. Notifications will be available either through email or by text message. By receiving this notification ahead of time, users can plan an alternative route to their destination in the event that the usual route is congested or closed due to traffic and/or scheduled construction.

The user interface will contain the following input fields:

(A) *Starting Zip Code*

- The user enters the source ZIP code

(B) *Destination ZIP Code*

- The user enters the destination ZIP code

(C) *Type of Day (Optional)*

- The user selects the type of day desired (ie. weekday, weekend, holiday)

(D) *Time (Optional)*

- The user selects a time from intervals of one hour (ie. 7AM)

(E) *Weather (Optional)*

- The user selects the weather condition that is current or anticipated

(F) *Method of Notification (Optional)*

- The user selects email or text message to receive departure alerts
- The user enters email address or phone number

After a route is selected, an option to view the gas prices along the selected route will become present. This feature will enable the driver to be able to view gas prices along the route and determine the best choice for refueling. Gas prices across the Metropolitan area can be easily accessed online. As a software primarily intended for commuters, having access to the gas prices at nearby stations or along a desired route is helpful. The program will allow users to record fuel usage information. Users will input gallons of fuel refilled, total miles, and gas price to calculate the average price per mile using a specific route.

The user interface will contain the following input fields:

(A) *Fuel Grade*

- The user will select desired fuel grade

(B) *Station (Optional)*

- The user will select a desired fuel station (ie. Shell, BP)

(C) *Distance (Optional)*

- The user will enter the maximum allowable distance from the route

If time permits, an additional feature we may entertain is public transportation recommendations. Public transportation information provides an alternative form of transportation. If congestion is excessive, travelling on public transportation may allow the user to arrive at his or her destination in time.

Glossary of Terms

API: Application programming interface. This is a software-to-software interface that specifies how applications interact with each other. In practice, API is often in the form of a library that contains specifications for routines, data structures, object classes and variables.

Cron Job: A job or script run at certain times.

Database: An organized collection of data formatted in such a way that it can be efficiently analyzed and used.

Graphical User Interface (GUI): An interface helps the users to interact with electronic devices. This uses graphical components and visual indicators as opposed to text navigation. Usual GUI elements include icons, menus, buttons, and lists.

Library: A collection of implementations of behavior in terms of computer language.

Mobile Device: It is also known as handheld computer which is small. A device such as a smartphone that utilizes a touchscreen with a display.

Mobile Friendly: Allows users limited by the small screen space and touchscreen of a mobile device to easily operate the software.

MySQL: Open source relational database management system. This software uses SQL to interact with databases.

PHP: Scripting language primarily used in web development. Often used in conjunction with MySQL as it is able to interact with MySQL databases.

Severity: A method of classifying the traffic data based on the type of incident and duration of the incident (4 levels of severity).

SQL: Structured Query Language. A data manipulation programming language designed to perform tasks such as update data on a database or retrieve data from a database.

SQL Table: An organized set of data elements composed of columns and rows, forming "cells" where these intersect. A database is usually made up of many tables.

System Requirements

Functional Requirements:

Identifier	PW	Requirement
REQ 1	5	The system's interface shall accept the user starting and destination locations, inside the New York Metropolitan area, and return an interactive map accompanied with directions.
REQ 2	4	The system shall analyze historical traffic data and present the user the option to overlay the historical data over currently displayed route based on day (e.q., weekend, weekday) and time (e.q., one hour intervals).
REQ 3	4	The system shall analyze historical weather data and present the user the option to overlay the historical data over currently displayed route based on weather (e.q., clear, rain, snow, cloudy) and time (e.q., one hour intervals).
REQ 4	4	The system shall retrieve and record traffic data from MapQuest Traffic website every 1-3 hours.
REQ 5	4	The system shall retrieve and record weather data from Wunderground website every 1-3 hours.
REQ 6	4	The system shall provide alternative route suggestions and expected route times using MapQuest.
REQ 7	2	The system shall allow the user to input an email or phone number in order to receive departure alerts.
REQ 8	4	The system shall provide a mobile friendly website which can be viewed on a mobile device.
REQ 9	1	The system shall allow users to register with unique identifiers.

Nonfunctional Requirements:

REQ 10	3	The system shall present the user the option to show cheapest gas prices along the intended route.
REQ 11	5	The system shall mainly focus on the New York Metropolitan Area, which includes New York, New Jersey, and Connecticut.

REQ 12	3	The system shall allow the user to input a maximum distance of deviation from intended route for gas.
REQ 13	3	The system shall allow the user to delete existing alerts.
REQ 14	3	The system shall allow the user to input gallons of fuel refilled, total miles, and gas price to calculate the average price per mile using a specific route.

On-Screen Appearance Requirements:

REQ 15	5	The system shall use a display containing MapQuest interface.
REQ 16	4	The system shall have simple and concise display for both interface and the data display.
REQ 17	3	The system shall utilize different colors to represent the different intensities of traffic.

Analysis:
Initially, we began with a total of 21 requirements. As part 2 of the first report came by, it seemed that this number of requirements seemed excessive. We then re-examined the list and decided to consolidate. “The system shall have a fast response time to the user’s request.” was removed because this would be a common requirement that does not require to be specifically listed as a requirement. “The system should provide information on public transportation recommendations.” was removed to maintain focus on the primary requirements as this was an added feature if time allotted but we will note that here so keep it on record.

FURPS+ Model:

Functionality	1. Features both web and mobile applications for more ease of use for users. 2. Features inputs such as zip-code, weather, day, time interval, notification way for the user to get more accurate results and better suggestions to allow the user to arrive in time. 3. User's search information are secured and cannot be accessed by different IP addresses.
Usability	1. The main page allows the user to get started. 2. The user can refresh the page and start a new search anytime. 3. All pages have similar design to allow user easy to read. 4. User can retrieve their search history (recent 10 days). 5. User can save their prefered route. 6. Input boxes are clearly labeled 7. Notification is short and concise.
Reliability	1. Check whether users entered valid data on the website 2. An error occurs when user requests for a regions that are not included in the database. 3. The specific search request allows a more accurate result. 4. Allow large amount of the search at the same time to avoid system crash.
Performance	1. Short response time after user submits the search request 2. Multiple users are able to use the app at the same time.
Supportability	1. Website application runs on all kinds of browsers. 2. Classes are clearly specified to increase the ease of expanding the software. 3. Help button for reporting errors to administrators.

Acceptance Tests for Requirements

Acceptance Test Cases for REQ 1:

ATC1.1	Enter two locations inside of the New York Metropolitan area (pass)
ATC1.2	Enter one location inside of the New York Metropolitan area and one outside (fail)
ATC1.3	Enter two locations outside of the New York Metropolitan area (fail)

Acceptance Test Cases for REQ 2 & REQ 3:

ATC2/3.1	Display bar chart of collected data showing incidents per day and show what percentage of accidents occur on that day (pass)

Note: A specific test case for these two requirements proved to be difficult. A better acceptance test for these requirements are better included in UC-1.

Acceptance Test Cases for REQ 4 & REQ 5:

ATC4/5.1	Display tables of collected data (pass)

Acceptance Test Cases for REQ 6:

ATC6.1	Display tables of collected data (pass)

Note: Similar to REQ 2 & REQ 3, a test acceptance case is better implemented with UC-1.

Acceptance Test Cases for REQ 7:

ATC7.1	Enter an email into “Email Address/Phone Number” field (pass)
ATC7.2	Enter a phone number into “Email Address/Phone Number” field (pass)

Acceptance Test Cases for REQ 8:

ATC8.1	Load website on a mobile device browser (pass)

Acceptance Test Cases for REQ 9:

ATC9.1	A visitor not logged in fills out the registration form using an unused identifier (pass)
ATC9.2	A visitor not logged in attempts to fill out the registration form using an already used identifier

Acceptance Test Cases for REQ 10:

ATC10.1	Select “Display Gas Prices” on interactive map (pass)

Note: A test acceptance case is better implemented with UC-1.

Acceptance Test Cases for REQ 11:

Note: This test case is better implemented along with REQ 1.
Acceptance Test Cases for REQ 12:

ATC12.1 Select from a set of distances from intended route (pass)

Acceptance Test Cases for REQ 13:

ATC13.1 Select an alert to delete and "Delete" (pass)

Acceptance Test Cases for REQ 14:

ATC14.1 Enter acceptable numbers for gallons of fuel refilled, total miles, and gas price (pass)

Note: This is hard to implement because a user might forget to record on a refill and thus require to input a much higher allowed number for the inputs. Will have to decide on whether to strictly enforce ranges or whether to allow multiple refills to be inputted at once. Typical inputs should range from 5 - 18 gallons, 100 - 400 miles, and $2.50 - $5.00 gas prices.

Acceptance Test Cases for REQ 15, REQ 16 & REQ 17:
Note: Once again, hard to test case specifically but easier to implement with UC-1.

Functional Requirements Specification

Stakeholders:

- Users
- Administrator

Actors & Goals:

- User
    - Initiating Actor
    - The user initiates interface with the website to access traffic, weather, and gas price information.
- Administrator
    - Initiating Actor
    - The administrator sets up the traffic, weather, and gas price collection systems to be used in the statistical analysis of the data.
- Crontab
    - Initiating Actor
    - The goal of Crontab is to perform data collection at a scheduled rate set by the administrator. Current traffic data, weather, and 10-day forecast are retrieved and stored into the MySQL Database.
- Alerter
    - Initiating Actor
    - The alerter will execute the selected alert method to the user.
- MySQL Database
    - Participating Actor
    - The goal of the database is to store the parsed traffic and weather data for future use in the statistical analysis process. The database is an intermediary type since the data collection side and the user interface must have access to it.
- Traffic & Mapping Service (MapQuest)
    - Participating Actor
    - The goal of the mapping service is to provide an interactive map for the user to visualize his or her route.
- Weather Service (Wunderground)
    - Participating Actor
    - The goal of the weather service is to provide current weather and forecast data, which the data collection system can process and store for future use.
- Gas Prices Service (myGasFeed)
    - Participating Actor

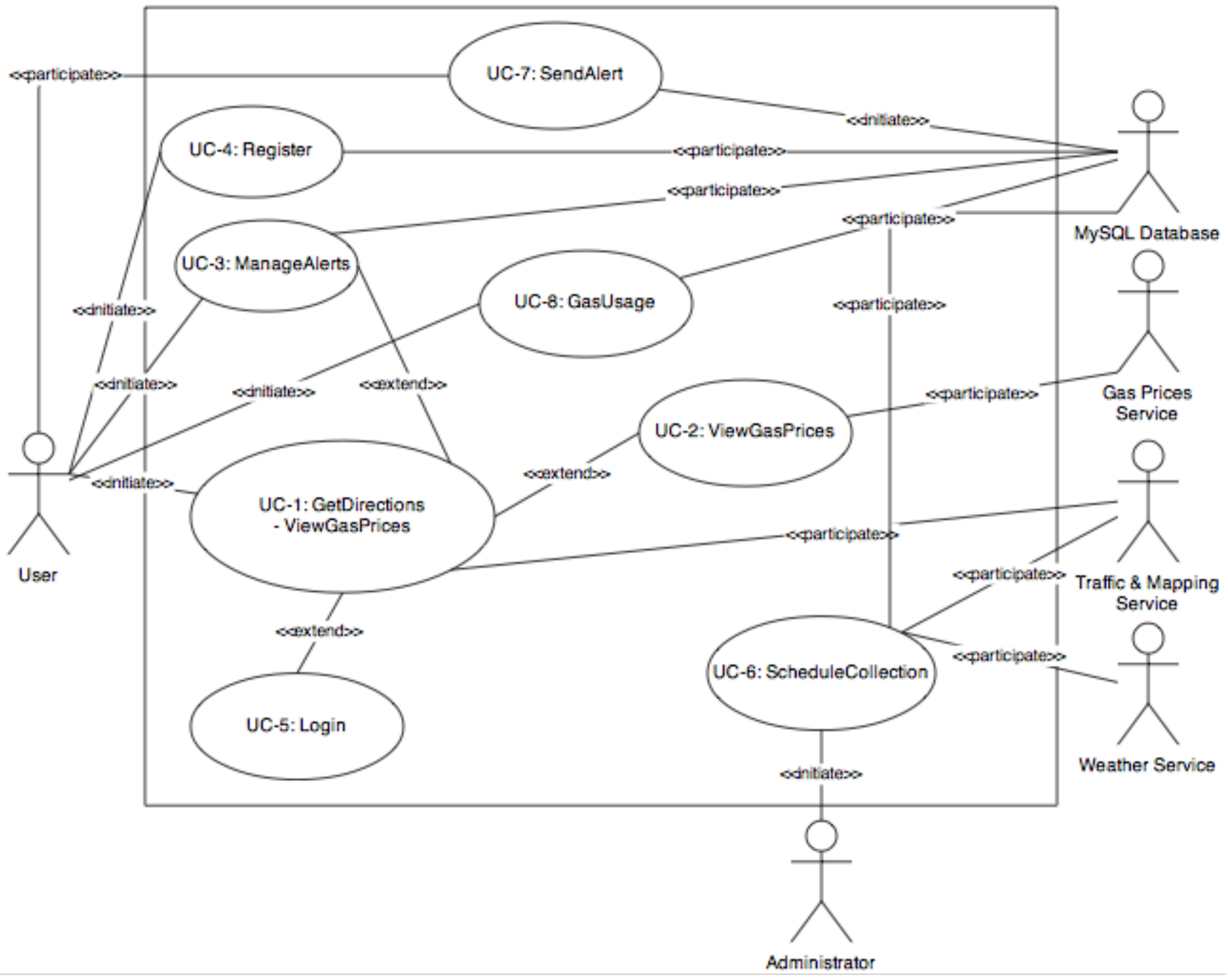
    - The goal of the gas prices service is to provide gas prices from various vendors.

Casual Description of Use Cases:

- UC-1: GetDirections - Allows user to input two locations and determine a best route of travel between the two locations with an overlay of current and historical traffic data and forecasted weather during time of travel.
- UC-2: ViewGasPrices - Allows user to overlay gas prices along a route.
- UC-3: ManageAlerts - Allows users access to manage existing alerts.
- UC-4: Register - Allows user to fill out registration form and become a member of the site.
- UC-5: Login - Allows user to log in to the website and gain access to routing, historical and alert features.
- UC-6: ScheduleCollection - Allows administrator to schedule collection scripts to retrieve data from the respective websites and store data into MySQL database.
- UC-7: SendAlert - Allows system to send departure alert to user.
- UC-8: GasUsage - Allows user to input gas usage information and view usage statistics.

Analysis:
It was first arranged to have a total of 10 use cases where UC-2 and UC-3 were "ViewTraffic" and "ViewWeather." As we began writing fully-dressed use case for UC-1, it became apparent that separating these two requirements would be non-ideal because in order for UC-1 to execute, it would require execution of those other two use cases. From there on, it was decided to combine all three into the one main and large use case. For a similar reason, a UC-9 of "AddAlert" was consolidated into a UC-1 to create a more streamline website where all the features are available on the same page.
UC-4 and UC-5 were created to handle the problems caused by having anonymous users. Although this was not stated specifically in the customer's requirements, it would alleviate the headache of having to determine whether or not a user was registered. If a user was not registered and wanted to create alerts, he or she would need to first register and then create the alert. With the Register/Login required, the users must first register and thus removes an anonymous use case. This also ties in with UC-3 because if the users are not logged in, they would not be able to view and edit saved alerts. In fact, they would have to delete all alerts and start with zero saved alerts. The Register/Login removed this odd function.
UC-7 was required to send out the required alerts. It's implementation was unknown at the time but came to realize that a similar script as the collection scripts could be run hourly to and send mass alerts then.

Use Case Diagram



Traceability Matrix

	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC-7	UC-8
REQ 1	X							
REQ 2	X							
REQ 3	X							
REQ 4						X		
REQ 5						X		
REQ 6	X							
REQ 7	X						X	
REQ 8	X							
REQ 9				X	X			
REQ 10		X						
REQ 11	X							
REQ 12		X						
REQ 13			X					
REQ 14								X
REQ 15	X							
REQ 16	X							
REQ 17	X							
Total PW:	**40**	**6**	**3**	**1**	**1**	**8**	**2**	**3**

Fully-Dressed Description of Use Case:

Use Case UC-1:	**GetDirections**
Related Requirements:	REQ 1 - 3, REQ 6 - 8, REQ 11, REQ 15 - 17
Initiating Actor:	User
Actor's Goal:	To generate directions from one starting location to destination location.
Participating Actors:	MySQL Database, Traffic & Mapping Service
Precondition:	- User is a registered user. - The system displays the required input requirements, starting and destination locations. - The source and destination addresses are valid locations within the application coverage area. - **Traffic & Mapping Service** is operational.
Success End Condition:	- The user is presented the optimal route to his or her destination
Extension Points:	*ViewGasPrices* (UC-4)
Flow of Events for Main Success Scenario: include::*Login (UC-7)* ← 1. **System** displays the application web interface awaiting **User** inputs of: - Starting Location - Destination Location - Time of Departure - a) Leave Now b) Depart By - (Optional) Will default to "Leave Now" unless "Depart By" specified then Date, a) Today b) Tomorrow c) Choose Date (Max up to 10 days ahead), and Time, 00:00 - 23:00, inputs will be required - Set Alert - a) On b) Off - (Optional) Will default to "Off" unless "On" specified then a) Email b) Cell Phone Number, inputs will be required → 2. **User** enters required inputs. ← 3. **System** sends latitude and longitude coordinates of starting and destination locations to **Traffic & Mapping Service**. → 4. **Traffic & Mapping Service** returns optimal route along with an interactive map. ← 5. **System** queries **MySQL Database** for route information. → 6. **MySQL Database** returns historical traffic data along specified route. ← 7. **System** queries **MySQL Database** for forecasted weather information on specified date. → 8. **MySQL Database** returns forecasted weather information along specified	

route.
← 9. **System** displays interactive map of optimal route along with step-by-step directions.

Flow of Events for Extensions (Alternate Scenario):

2a. **User** attempts to input locations outside of application coverage area. (e.g., any location not in the New York Metropolitan area.

← 1. **System** alerts **User** that a location is outside of the coverage area and **System** ignores request and returns to Step 1 of Main Success Scenario.

2a. **User** inputs invalid email address or cell phone number or leaves either blank

← 1. **System** alerts **User** that values are invalid and **System** ignores request and returns to Step 1 of Main Success Scenario.

Analysis: As stated earlier, the initial intention was to have separate use cases for GetDirections, ViewTraffic and ViewWeather. As we began writing detailed use case scenarios for GetDirections, it became apparent that this would prove to be more difficult than expected. For a succcessful UC-1 scenario, it would require completions of ViewTraffic and ViewWeather while it was performing it's own use case. Since this was the case, it was decided to combine the three into a main interface to avoid this.

Use Case UC-2:	**ViewGasPrices**
Related Requirements:	REQ 10, REQ 12
Initiating Actor:	User
Actor's Goal:	To view nearby gas prices along specified route.
Participating Actor:	Gas Prices Service
Preconditions:	- User is a registered user. - User has completed *GetDirections* (UC-1).
Success End Condition:	Gas prices of nearby gas stations are displayed on the interface.

Flow of Events for Main Success Scenario:

→ 1. **User** selects to enable nearby gas prices.

← 2. **System** prompts **User** to select fuel type, a) Regular b) Mid-Grade c) Premium d) Diesel) and the maximum distance from the user's location, a) < 5 mi b) < 10 mi c) < 15 mi

→ 3. **User** enters required inputs.

← 4. **System** sends latitude, longitude, fuel type, and distance to the **Gas Prices Service**.

→ 5. **Gas Prices Service** returns location of nearby stations with their last reported prices.

← 6. **System** displays these as markers on the interactive map.

Analysis: This was considered to be implemented along with UC-1 but was finally decided to leave as a separate use case. While it would have been easy to implement along with UC-1 and the main interface, a dilemma of a user just wanting directions quickly became apparent. Having to navigate through an additional set of inputs would delay the user. This lead to the decision to keep this use case separate to avoid delay but still provide this function. Even though this is a separate use case, this use case does depend on completion of UC-1 to be viewable.

Use Case UC-3:	**ManageAlerts**
Related Requirements:	REQ 13
Initiating Actor:	User
Actor's Goal:	To create new alerts through email or texting, and remove old alerts.
Participating Actor:	MySQL Database
Preconditions:	- Alert already exists in the database.
Success End Condition:	Existing alert is deleted.
Flow of Events for Main Success Scenario:	
←	1. **System** displays application web interface.
→	2. **User** selects "Manage Alerts."
←	3. **System** queries **MySQL Database** for existing alerts related to current **User**.
→	4. **MySQL Database** returns results for **User**.
←	5. **System** displays existing alerts in tabular format to **User**.
→	6. **User** selects alert(s) to be deleted.
←	7. **System** sends index(es) of selected alerts to **MySQL Database**.
→	8. **MySQL Database** deletes selected alert(s) and returns updated existing alerts.
←	9. **System** displays existing alerts for **User**.
Flow of Events for Extensions (Alternate Scenario):	
9a.	**MySQL Database** returns an empty table.
←	1. **System** notifies **User**, "You currently do not have any alerts set."

Use Case UC-8:	**GasUsage**
Related Requirements:	REQ 14
Initiating Actor:	User
Actor's Goal:	To input gas usage information for statistical view.
Participating Actor:	MySQL Database
Preconditions:	- User is a registered user. - User has navigated to "Gas Usage" web interface.

Success End Condition: The user will be shown gas usage statistics for the user.

Flow of Events for Main Success Scenario:

← 1. **System** displays “Gas Usage” web interface.

→ 2. **User** inputs gallons of fuel used, total miles traveled, and gas price.

← 3. **System** queries **MySQL Database** to insert information into database for current **User**.

→ 4. **MySQL Database** returns successful insert into table.

← 5. **System** queries **MySQL Database** for **User** usage information.

→ 6. **MySQL Database** returns selected table for **User**.

7. **System** calculates:
- Average $ / mi per Fillup
- Total Average $ / mi

← 8. **System** displays information to the **User**.

Flow of Events for Extensions (Alternate Scenario):

2a. **User** fails to input one of the required fields or has inputted an invalid value.

← 1. **System** ignores inputs, alerts **User** of “Invalid/Missing Inputs,” and returns to Step 1 of Main Success Scenario.

Acceptance Tests for Use Cases

Test Case Identifier:	TC-1.01
Use Cases Tested:	UC-1: GetDirections/UC-2 ViewGasPrices - main success scenario
Pass/Fail Criteria:	If an interactive map is displayed with step-by-step directions from a starting location to a final destination location with an overlay of historical traffic and weather data. If no interactive map is displayed and no traffic/weather data is displayed.
Input Data:	Starting Location: 98 Brett Rd, Piscataway Township, NJ 08854 Destination Location: 1185 Avenue of the Americas, Manhattan, NY 10036 Time of Departure: Tomorrow Time: 0800 Set Alert: On Alert Method: Email Address Email Address / Phone Number: test@testing.com

Test Procedure:	**Expected Result:**
Set Up: Load the main application web interface and Logged In.	System displays main application web interface.
Step 1. Input the initial input data into appropriate fields. Step 2. User clicks on “Get Directions!” Step 3. User clicks on “Show Nearby Gas Prices.” Step 4. User selects fuel grade “Regular.” Step 5. Use selects maximum distance from route “< 10 mi.”	System accepts inputted data and returns with an interactive map with step-by-step directions from starting location to destination location with an overlay of traffic and weather history. System will send an appropriate alert to “test@testing.com” alerting user to leave at desired time of departure. Systems displays and places markers to nearby gas stations with a list of their gas prices.

Analysis: Although this acceptance tests verifies UC-1 functionality, UC-2 test is included to verify a complete the test case. Since UC-2 requires a successful UC-1 execution, it is easier to continue and verify UC-2 afterwards.

Test Case Identifier:	TC-3.01
Use Cases Tested:	UC-3: ManageAlerts
Pass/Fail Criteria:	If an existing alert is deleted. If an existing alert is not deleted.

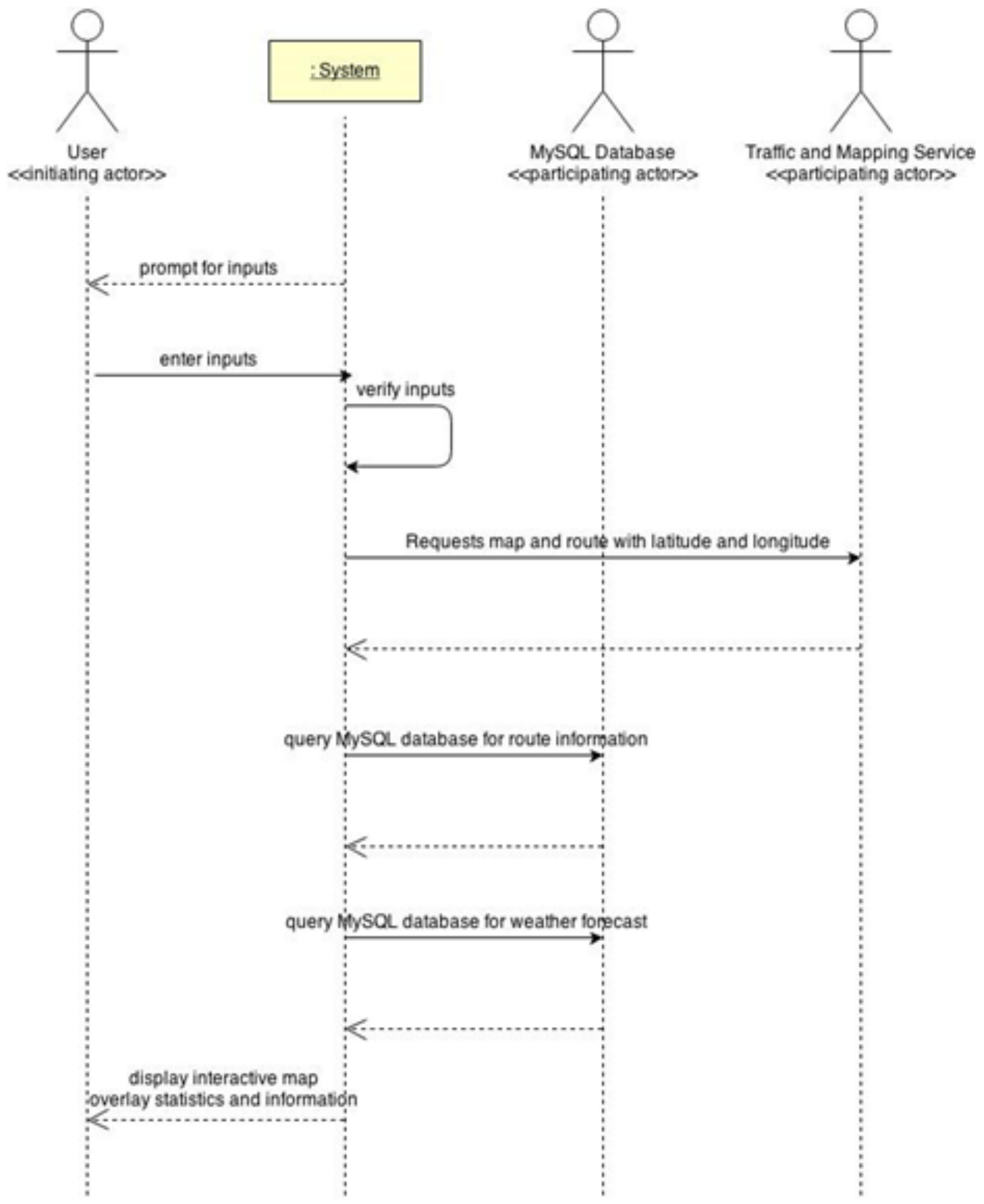
Test Procedure:	**Expected Result:**
Set Up: Load the main application web interface, Logged In and on "Manage Alerts" interface.	System displays "Manage Alerts" web interface.
Step 1. Select an existing alert by clicking checkbox Step 2. Click "Submit."	System deletes existing alert from database and displays updated database.

Test Case Identifier:	TC-8.01
Use Cases Tested:	UC-8: GasUsage
Pass/Fail Criteria:	If the user's total average dollar per mile and average dollar per mile is calculated based upon their usage statistics pulled from the database. If the user's total average dollar per mile is not calculated, is an invalid number, or incorrect.
Input Data:	Gallons of fuel used: 10 Total miles traveled: 150 Gas price: 3.50

Test Procedure:	**Expected Result:**
Set Up: Load the main application web interface and Logged In.	System displays main application web interface.
Step 1. User clicks on "Gas Usage" Step 2. User inputs gallons of fuel used. Step 3. User inputs total miles traveled. Step 4. User inputs gas price. Step 5. User clicks on "Get Statistics"	System accepts the user's data and displays average dollar per mile and total average dollar per mile.

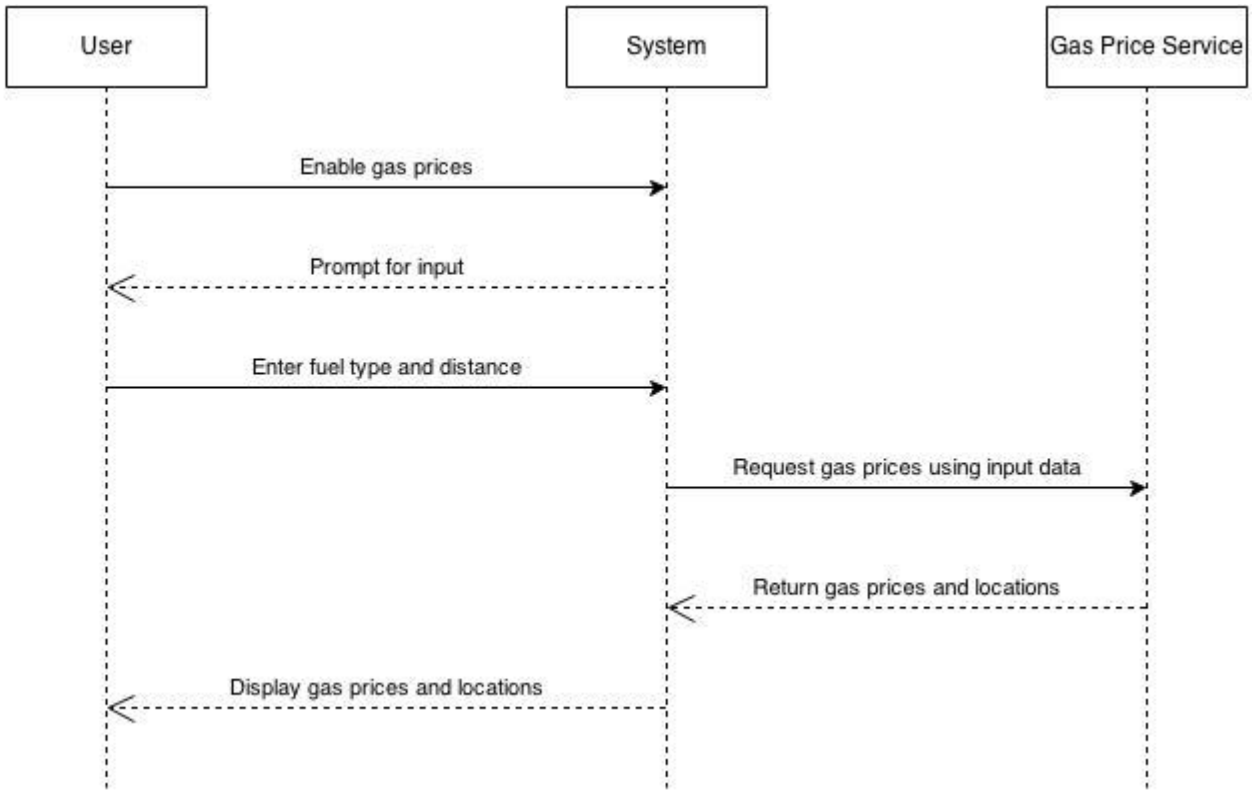
System Sequence Diagrams

Use Case 1:


: System
User
<<initiating actor>>
MySQL Database
<<participating actor>>
Traffic and Mapping Service
<<participating actor>>
prompt for inputs
enter inputs
verify inputs
Requests map and route with latitude and longitude
query MySQL database for route information
query MySQL database for weather forecast
display interactive map
overlay statistics and information

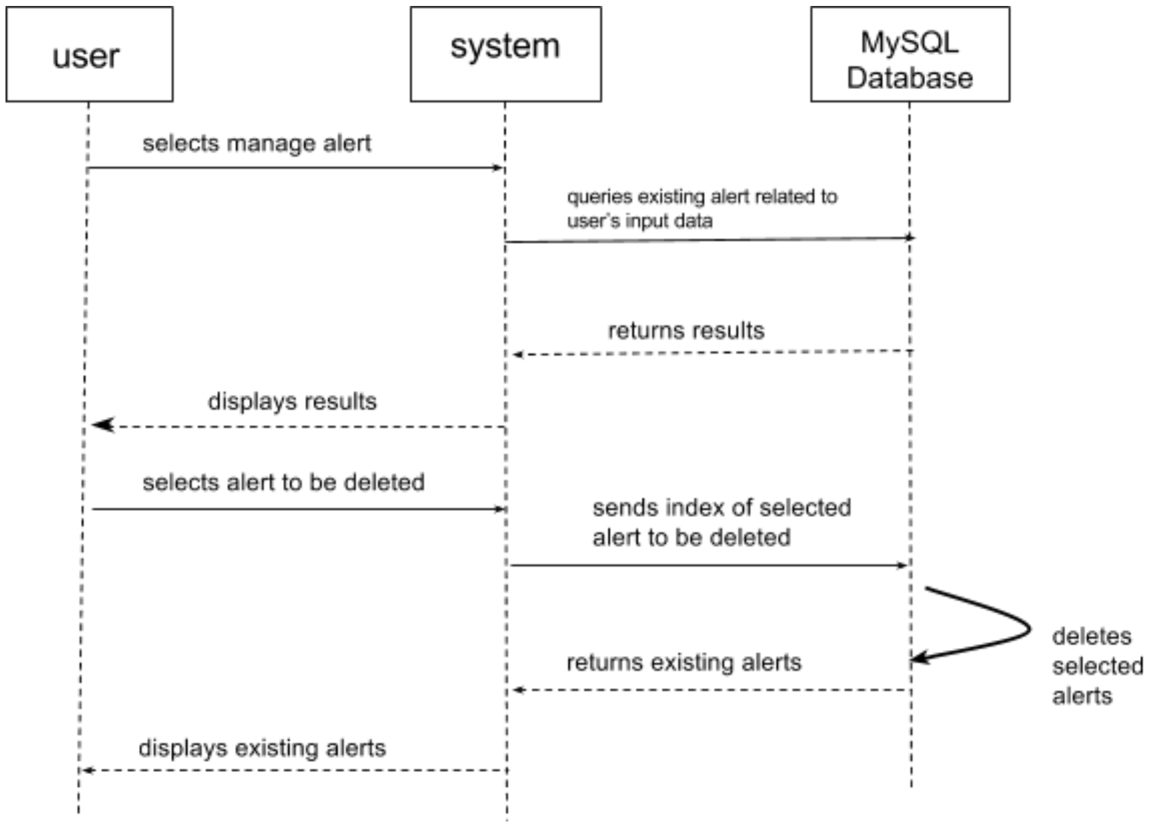
In UC-1, the system displays the user interface to the user, prompting for inputs. The user enters the required inputs and sends it back to the system, which is then verified. The system requests for the map and route with the latitude and longitude from MapQuest. Next, the system queries the MySQL Database for the requested traffic data and weather forecast. This information is then sent back to the system. Finally, the system displays the interactive map and overlays statistics and information back to the user.

Use Case 2:


User
System
Gas Price Service
Enable gas prices
Prompt for input
Enter fuel type and distance
Request gas prices using input data
Return gas prices and locations
Display gas prices and locations

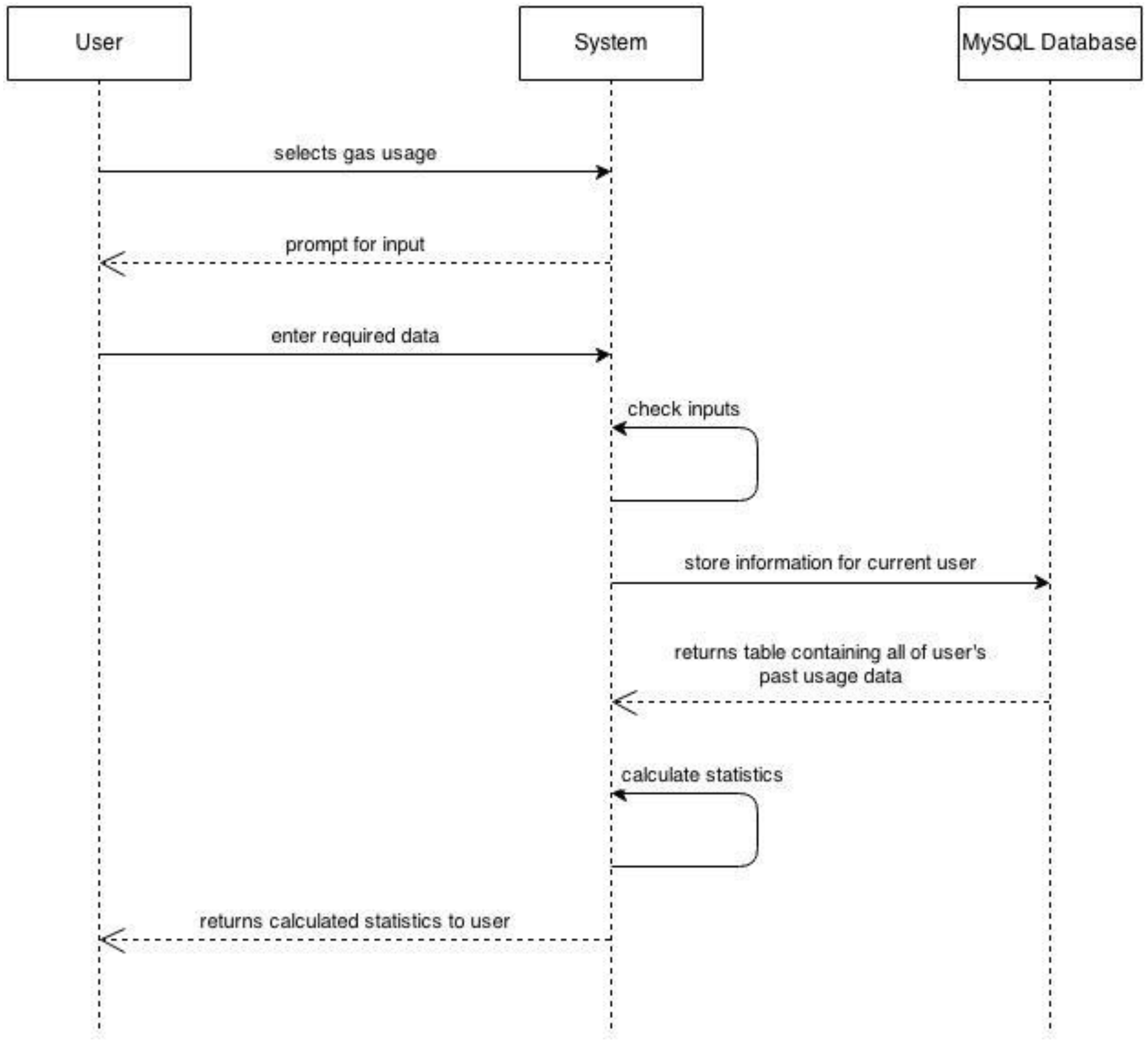
In UC-2 the user tells the system it wants to view nearby gas prices. The system then prompts the user to input the fuel type and maximum distance desired. The user supplies this information to the system, then the system relays this data to the Gas Price Service. This service will return prices and locations for nearby gas stations supplying the specified fuel type to the system. The system will then display this information to the user on the interactive map.

Use Case 3:



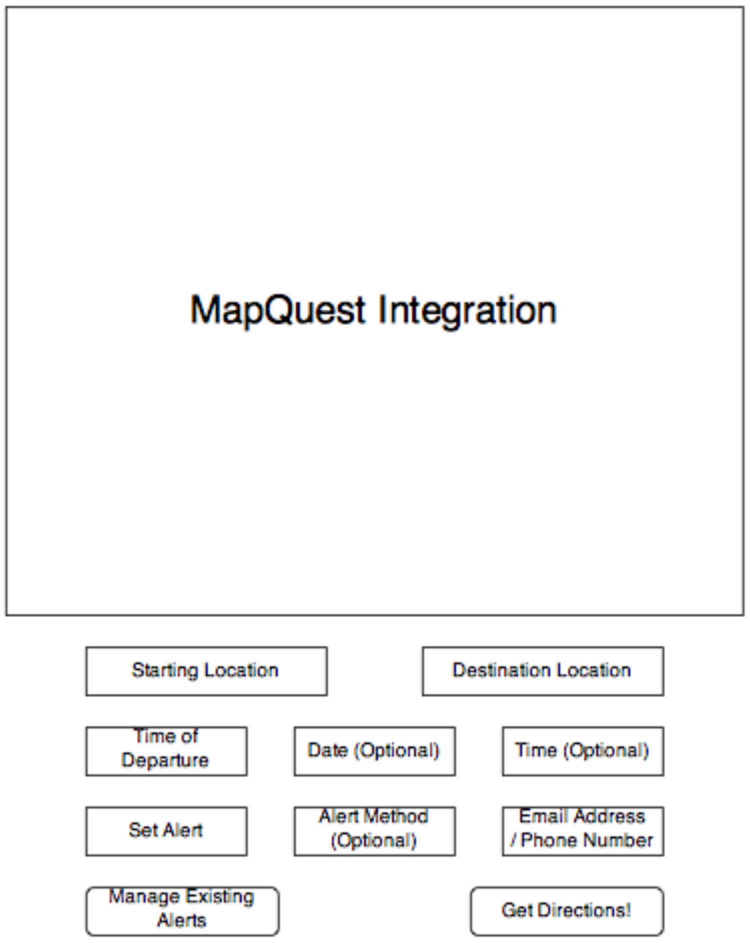
In UC-3, the user selects manage alert and turns it on. Based on the route information user inputs in UC-1, the system queries existing alert from MySQL Database. MySQL database returns results to the system. The system then displays the results to user. The user then is able to select the alert to be deleted. The system then sends the index of selected alerts to MySQL Database. MySQL Database deletes the selected alerts, and returns the existing alerts to the system. The system displays existing alerts to the user.

Use Case 8:


User
System
MySQL Database
selects gas usage
prompt for input
enter required data
check inputs
store information for current user
returns table containing all of user's past usage data
calculate statistics
returns calculated statistics to user

In UC-8, the user selects to display gas usage. The user is then prompted to input gallons of fuel used, total miles traveled, and gas price. The system will then check the inputs for errors and send the data to the MySQL database if none are found. The database will store the information for the current user's ID and return a table containing all of the user's stored gas usage information. The system will calculate the statistics based on this data and display it for the user.

User Interface Specification



Example of User Input for UC-1:
User inputs starting location and destination address into “Starting Location” and “Destination Location” input fields respectively. User will then use the dropdown box on “Time of Departure” and select “Depart By.” This will enable in the “Date” and “Time” dropboxes. The user will select “Today” from the “Date” dropdown and then select “05:00” from the “Time” dropdown box. The user will submit the form using the “Get Directions!” push button. The recommended route will then be shown inside the “MapQuest Integration” and step-by-step directions will be listed underneath it.

Example of User Input for UC-2
User selects fuel type Premium and inputs maximum distance 10 miles from user’s location. Locations of nearby stations with their last reported prices is displayed to user and shown as markers on the interactive map.

Example of User Input for UC-3
User completes all steps of UC-1 prior to “Get Directions.” User clicks the dropdown box “Set Alert” and selects “On.” This will enable “Alert Method” and “Email Address/Phone Number” field. User selects “Email” from “Alert Method” dropdown box. User then inputs email into “Email Address/Phone Number” field. The user will submit the form using the “Get Directions!” push button. The recommended route will then be shown inside the “MapQuest Integration” and step-by-step directions will be listed underneath it. The alert will be sent out according to the user’s inputs.

Example of User Input for UC-8
User inputs gallons of fuel used, total miles traveled, and gas price. System displays information to User.

User Effort Estimations

User Effort Estimation for UC-1:

NAVIGATION: 8 mouse clicks, as follow:

a. Click cursor to “Starting Location”

--- after completing the data entry below ---

b. Click “Time of Departure” and click “Depart By”
c. Click “Date” and click “Today”
d. Click “Time” and click “05:00”
e. Click “Get Directions!”

DATA ENTRY: about 60 keystrokes total, as follows:

a. Input Starting and Destination Locations (About 30 keystrokes for each address, e.q., 1185 6th Ave, New York, NY)

User Effort Estimation for UC-2:

NAVIGATION: completion of UC-1, then 2 mouse clicks, as follows:

a. Click "Enable Nearby Gas Prices"

--- after completing the data entry below ---

b. Click "Get Gas Prices"

DATA ENTRY: 4 mouse clicks, as follows:

a. Click "Fuel Type" and select "Regular"
b. Click "Maximum Distance" and select "<10 mi"

User Effort Estimation for UC-3:

NAVIGATION: completion of UC-1, Navigation to Step d, 3 mouse clicks, as follows:

a. Click “Set Alert” and click “On”
b. Click “Alert Method” and click “Email”

--- after completing the data entry below ---

c. Click “Get Directions!”

DATA ENTRY: about 10-20 keystrokes total, as follows:

a. Input email address “test@testing.com”

User Effort Estimation for UC-8:

NAVIGATION: 2 mouse clicks, as follows:

a. Click "Gas Usage"

--- after completing the data entry below ---

b. Click "Get Gas Usage"

DATA ENTRY: 3mouse clicks, and about 8 keystrokes, as follows:

a. Click "Fuel Used" and enter "10” gallons
b. Click "Miles Traveled" and enter "150” miles
c. Click "Gas Price" and enter $”3.33"

Domain Model

Use Case 1: GetDirections

Concept Definitions

Responsibility	**Type**	**Concept**
R1: Display main application web interface.		Interface
R2: Store list of registered users	K	UserDatabase
R3: Accept user inputs and forward to appropriate entities	D	Application
R4: Use MapQuest API to create route from starting location to destination location	D	DirectionService
R5: Contain historical traffic data for use on overlays on interactive map	K	TrafficDatabase
R6: Contain historical traffic data for use on overlays on interactive map	K	WeatherDatabase
R7: Store requested alert settings from user	D	SetAlert
R8: Store alerts per individual user	K	AlertsDatabase
R9: Manage interactions with the database		DB Connection

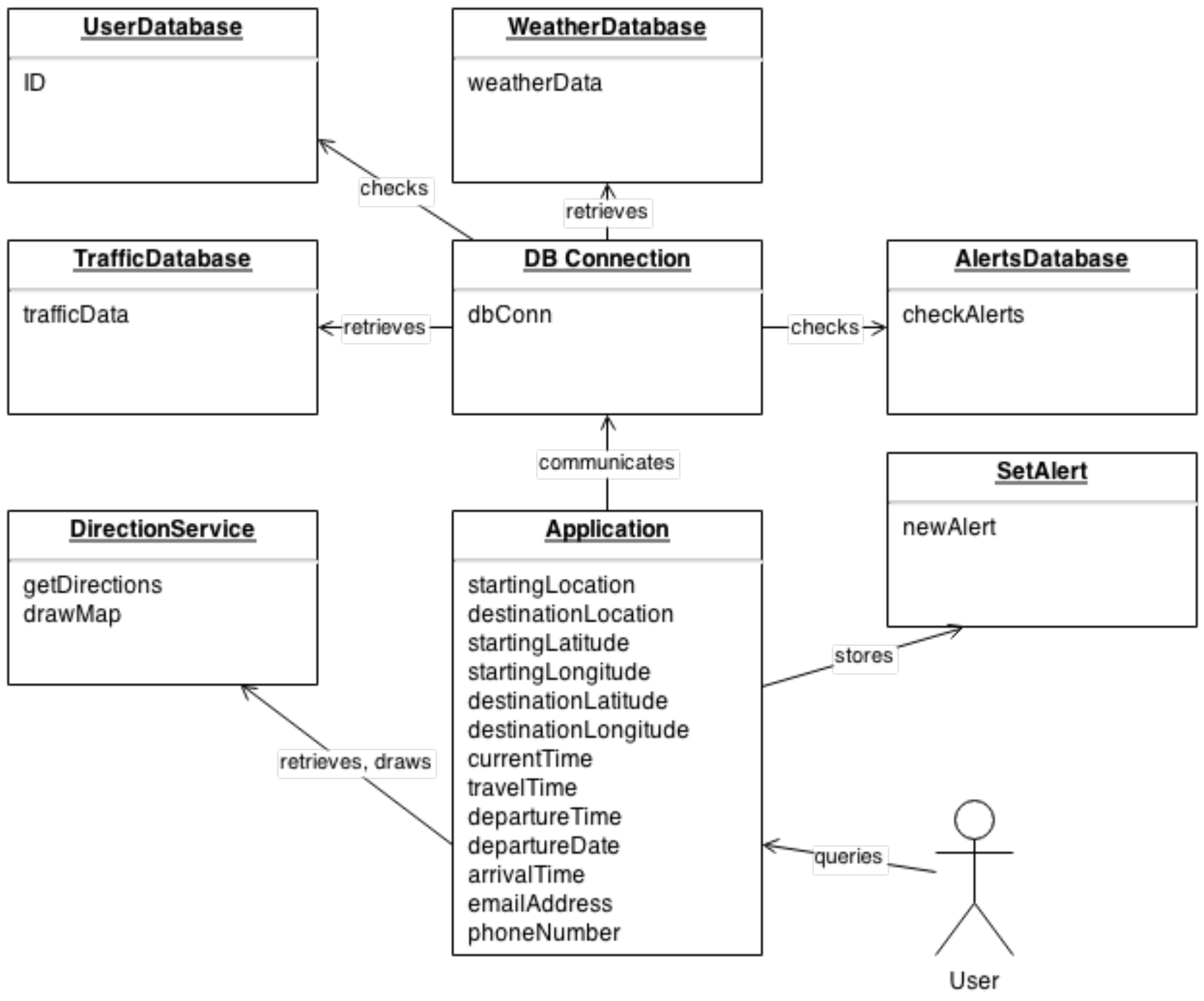
Attribute Definitions

Responsibility	**Attribute**	**Concept**
R10: DB network connection	dbConn	DB Connection
R11: User's identity	ID	UserDatabase
R12: Starting address R13: Destination address R14: Latitude of starting address R15: Longitude of starting address R16: Latitude of destination address R17: Longitude of destination address R18: Current time R19: Intended route's travel time R20: User's planned departure time	startingLocation destinationLocation startingLatitude startingLongitude destinationLatitude destinationLongitude currentTime travelTime departureTime	Application

R21: User’s planned departure date R22: Intended route’s arrival time R23: User’s email address R24: User’s phone number	departureDate arrivalTime emailAddress phoneNumber	
R25: Retrieve step-by-step directions from Mapping Service R26: Display interactive map	getDirections drawMap	DirectionService
R27: Traffic intensities based on times and locations	trafficData	TrafficDatabase
R28: Weather history based on locations	weatherData	WeatherDatabase
R29: Store new alert	newAlert	SetAlert
R30: Verify alert does not already exist	checkAlerts	AlertsDatabase

Association Definitions

Concept Pair	**Association Description**	**Association Name**
DB Connection↔ UserDatabase	Application verifies user exists in the UserDatabase and is properly logged in	checks
Application↔ DirectionService	Application submits user’s inputs to the DirectionService to generate a step-by-step direction list with an interactive map	retrieves, draws
Application↔ DB Connection	Application communicates user data and database data via DB Connection	communicates
DB Connection↔ TrafficDatabase	Application retrieves historical data from database	retrieves
DB Connection↔ WeatherDatabase	Application retrieves historical data from database	retrieves
Application↔ SetAlert	Application stores new alert from user	stores
DB Connection↔ AlertsDatabase	Application checks if alert already exists	checks



Analysis: In the fully dressed use case, the databases were all described as one. As we delve deeper into development, each part will require separate databases along with another **UserDatabase**.

It was questioned whether or not a **Controller/Communicator** would be required but decided that all users will handle their individual connections separately. The only communication will be between user and application to complete the use case.

DB Connection is set as a separate responsibility from the other databases because while the databases contain information, **DB Connection** will handle the interface between application and database.

SetAlert will be the main controller for handling new alerts. It will accept the new alert parameters and use **DB Connection** to store it in **AlertsDatabase**.

Use Case 2: ViewGasPrices

Concept Definitions

Responsibility	Type	Concept
R1: Display main application web interface.		Interface
R2: Accept inputs for distance and fuel type.	D	Application
R3: Use myGasFeed API to retrieve the locations and prices determined by the user's inputs.	D	GasService
R4: Store list of registered users	K	UserDatabase
R5: Manage interactions with the database		DB Connection

Attribute Definitions

Responsibility	Attribute	Concept
R6: Maximum distance. R7: Fuel Type. R8: Latitude R9: Longitude	maxDistance fuelType gasLatitude gasLongitude	Application
R10: Retrieve locations and prices from GasService. R11: Display locations on interactive map.	getGas appendMap	GasService
R12: DB network connection	dbConn	DB Connection
R13: User's identity	ID	UserDatabase

Association Definitions

Concept Pair	Association Description	Association Name
DB Connection↔ UserDatabase	Application verifies user exists in the UserDatabase and is properly logged in	checks
Application↔ DB Connection	Application communicates user data and database data via DB Connection	communicates
Application↔ GasService	Application sends user inputs to GasService. GasService sends back locations and prices to display on the interactive map.	retrieves, draws

Analysis: The database connection allows the application to contact the UserDatabase in order to confirm that the user is registered. After the user is confirmed, that application proceeds to contact the GasService in order to perform its function. The gas service will retrieve the required information then append its data onto the interactive map.

Use Case 3: ManageAlerts

Concept Definitions

Responsibility	**Type**	**Concept**
R1: Display main application web interface.	D	Interface
R2: Accept user inputs and forward to appropriate entity	D	Application
R3: retrieve the records from the MySQL Database		DB Connection
R4: Render the retrieved records into an HTML document for sending to actor's Web browser for display.	D	Page Maker

Attribute Definitions

Responsibility	**Attribute**	**Concept**
R5: User's identity that used to determine the actor's credentials, which in turn specify what kind of data this actor is authorized to view.	ID	UserDatabase
R6: Retrieve existing alerts R7: Delete selecting alerts	getAlert deleteAlert	Application
R8: Allows the user to see their existing list of the alerts.	AlertList	MySQL Database

Association Definitions

Concept Pair	**Association Description**	**Association Name**
Application↔ UserDatabase	Application verifies user exists in the UserDatabase and is properly logged in	checks
Application↔	Application passes requests to Page Maker and	conveys request

Page Maker	receives back pages prepared for displaying	
Page Maker↔ DB Connection	Database Connection passes the retrieved data to Page Maker to render them for display	Provides data
Page Maker ↔ Interface	Page Maker prepares Interface	Prepares
Application↔ DB Connection	Application passes delete selected alerts request to Database connection	Conveys request

Analysis: Data connection coordinates the work between the action of user and MySQL database. Based on the user's ID, the application retrieves the UserDatabase and displays the current alert lists user has. Database Connection is also able to receive the data that is the index of the alerts to be deleted by user, and then pass it to the MySQL database to delete the selected alerts. The page maker then receive the new list from Database connection and displays them to user.

Use Case 8: GasUsage

Concept Definitions

Responsibility	**Type**	**Concept**
R1: Display main application web interface.		Interface
R2: Store list of registered users	K	UserDatabase
R3: Accept inputs for fuel used, total miles traveled, and gas price. Calculates average dollar per mile.	D	Application
R4: Stores information of fuel usage.	K	GasDatabase
R5: Manage interactions with the database		DB Connection

Attribute Definitions

Responsibility	**Attribute**	**Concept**
R6: DB network connection	dbConn	DB Connection
R7: User's identity	ID	UserDatabase
R8: Gallons of fuel used R9: Total miles traveled R10: Gas price	fuelUsed totalMiles gasPrice	Application

R11: Average dollars per mile R12: Total average dollars per mile	avgDPM totAvgDPM	
R13: User’s past gas usage information.	gasData	GasDatabase

Association Definitions

Concept Pair	**Association Description**	**Association Name**
DB Connection↔ UserDatabase	Application verifies user exists in the UserDatabase and is properly logged in	checks
DB Connection↔ GasDatabase	Application sends new input to be stored in database. Database returns with table to be used in dollar per mile calculation.	stores, receives
Application↔ DB Connection	Application communicates user data and database data via DB Connection	communicates

Analysis: The database connection is the link that allows the application to share with the two databases. The UserDatabase is used to confirm that the user is registered. The user's ID then allows the GasDatabase to store and retrieve gas usage statistics for the specific user. The application then calculates and displays the average dollars per mile of the user.

Traceability Matrix

		Domain Concepts										
Use Case	**PW**	DB Connection	Application	Interface	DirectionService	GasService	Page Maker	UserDatabase	TrafficDatabase	WeatherDatabase	AlertsDatabase	GasDatabase
UC-1	40	X	X	X	X			X	X	X	X	
UC-2	6	X	X	X		X		X				
UC-3	3	X	X	X			X	X			X	
UC-4	1	X	X	X				X				
UC-5	1	X	X	X				X				
UC-6	8	X										
UC-7	2	X	X	X							X	
UC-8	3	X	X	X				X				X
	Max PW	40	40	40	40	6	3	40	40	40	40	3
	Total PW	64	56	56	40	6	3	54	40	40	43	3

System Operation Contracts

Operation:	CollectData
Preconditions:	- Traffic & Weather Service operational
Postconditions:	- Data is continuously updated and available

Operation:	CalculateStatistics
Preconditions:	- TrafficDatabase/WeatherDatabase not empty
Postconditions:	- Data is divided into days and then separated by time

Operation:	SetAlert
Preconditions:	- UserDatabase available - User □ UserDatabase
Postconditions:	- Alert stored into AlertsDatabase

Operation:	StoreGasStats
Preconditions:	**-** UserDatabase available - User □ UserDatabase - GasDatabase available
Postconditions:	**-** User’s gas usage statistics stored into GasDatabase

Plan of Work

Original Plan of Work:

Plan of Work

There are six people in our group and due to the limited time constraints and the demanding task of data collection and analysis, any fewer than the allotted six members would prove difficult. Even though the project workload will be divided evenly among the six members, specifics groups of two will focus on certain aspects due to strengths and abilities as follows:

Jan & Jonathan: Departure Alerts & Public Transportation
Chih-Ting & Ting-Chieh: Probable Delays & Alternate Routes
Kevin & Xuan: Fuel Prices & Consumption

As of February 23, 2014:

In an attempt to make this project as agile as possible, we will maintain the set of groups working on their separate features. Each minigroup has been working on their separate use cases, test cases, and design. As of now, the system has been collecting data since the beginning of February.

The plan is to develop the website and maintain a weekly goal. The goal is to have a fully functional website by the March 23, the week prior to the first demo. This last week should be spent fine tuning small details and ironing out small kinks. This plan of action will most like be altered as time goes and individuals might shift around or take extra work as needed.

Traffic Monitoring	Start Date	End Date	Time	Team	2/24/2014 - 3/2/2014	3/3/2014 - 3/9/2014	3/10/2014 - 3/16/2014	3/17/2014 - 3/23/2014	3/24/2014 - 3/28/2014
					Implement..				
Main Scenario	24-Feb	7-May	72	Jan/Jon	Statistics on Data	Map Overlays / Alternate Routes	Total System Integration	Testing/Last Minute Changes...	First Demo
GasUsage	24-Feb	7-May	72	Kevin/Chih-Ting	Interface for Data Input	Usage Calculations & Statistics	Total System Integration	Testing/Last Minute Changes...	First Demo
Alerts	24-Feb	7-May	72	Ting-Chieh/Xuan	Interface for ManageAlerts	Send out Alerts	Total System Integration	Testing/Last Minute Changes...	First Demo

References

1. Marsic, I. 2012. *Software Engineering*.
2. *MapQuest Platform Web Services : Traffic (v2).* [Online]. Available: http://www.mapquestapi.com/traffic/. [1 February 2014].
3. *API | Weather Underground*. [Online]. Available: http://www.wunderground.com/weather/api/d/docs?d=index. [1 February 2014].
4. *myGasFeed.com - Api requests for developers.* [Online]. Available: http://www.mygasfeed.com/keys/api. [1 February 2014].
5. *W3Schools Online Web Tutorials.* [Online]. Available: http://www.w3schools.com/. [1 February 2014].