

Summary of Changes:

As we gained better understanding of our problem we made some changes in the Use Cases while preparing Report No. 2. Hence, we have revised the following parts of Report 1 to help in understanding Report 2. Report 2 is in continuation after the end of Report 1.

1. Casual Use Case Description
2. Use Case Diagrams.
3. Fully dressed use case description
4. System Sequence Diagrams

In the coming days, changes will be made to the other parts of Report 1 which need revision.

TRAFFIC MONITORING SYSTEM

GROUP NUMBER: 7



GROUP MEMBERS:

Kush Patel
Tejas Ravi
Jeffrey Gillen
Shreyas Bhandare
Kisholoy Vinayak Ghosh
Aaditya Shukla

kush.patel@rutgers.edu
tr332@scarletmail.rutgers.edu
jeffrey.gillen@rutgers.edu
shreyas.bhandare@rutgers.edu
kisholoy.v.ghosh@rutgers.edu
aaditya.shukla@rutgers.edu

Table of Contents

I.	Problem Statement.....	3
II.	Algorithm.....	5
III.	Data Collection.....	6
IV.	Functional Features.....	7
V.	Glossary of Terms.....	9
VI.	System Requirements	
	a. Functional Requirements.....	10
	b. Non-Functional Requirements.....	10
	c. On-Screen Appearance Requirements.....	11
VII.	Individual Breakdown.....	11
VIII.	Functional Requirement Specification.....	12
IX.	Use Cases	
	a. Casual Description	13
	b. Use Case Diagram.....	14
	c. Traceability Matrix.....	15
	d. Fully Dressed Description	16
	e. System Sequence Diagram.....	18
X.	User Interface Specification	
	a. Preliminary Design.....	20
	b. User Effort Estimation.....	21
XI.	Domain Model	
	a. Concept Definition for use case 1.....	23
	b. Association definitions for use case 1.....	23
	c. Attribute Definitions for use case 1.....	24
	d. Concept Definition for use case 2.....	24
	e. Association definitions for use case 2.....	25
	f. Attribute Definitions for use case 2.....	25
	g. Traceability Matrix.....	26
XII.	System Operation Contracts.....	27
XIII.	Prediction Model.....	28
XIV.	Plan of Work.....	30
XV.	References.....	31
XVI.	Summary of changes.....	33

Problem Statement

Problem definition:

The internet has advanced tremendously over the past decade. In today's day and age, our lives have become dependent on using the internet. From checking email to deciding what food to eat, every activity is governed by various websites. Websites also offer map applications that offer services such as navigation and traffic updates. Navigation is very essential if someone wants to go to an unknown location. Live time traffic is also very essential to determine congestion on a particular route and plan routes accordingly.

There are several services that offer this feature; for example, "Traffic.com" and "Yahoo! Maps Live Traffic" etc. All these services only account for current traffic conditions in a given geographical area. Even though current traffic models give a good idea of the congestion on a particular highway, it still is not completely accurate because the prediction can be based off of very few traffic reports. If there are no traffic reports, one cannot know what the traffic levels look like at a specific time of the day. With the help of current traffic intimation models, a user will get to know about current traffic updates dynamically as he travels along the route. However, it would be more meaningful if the user knows what to expect in terms of traffic even before starting from his starting location. This project aims to take traffic prediction to another level. Instead of just using current traffic patterns, we will use historic traffic data to give the user a better estimate of traffic patterns. Below we have proposed a solution to this problem. Several questions one would ask are What? How? Why? Who? Etc. We will explain each aspect below.

Solution to the Problem:

This project will create a One Stop Site for a user on a busy weekday or a relaxed weekend. It will give the user the ability to query for traffic information using a zip code, day of the week and time of the day. So when a user does perform a query, the traffic prediction would be based on the current predictions as well as the historic traffic data that the route experienced over the past few months. In this way, the user will be able to see what route is experiencing congestion and hereby decide whether to take another route. As add on, we also will be collecting weather information every time a traffic report is collected. If there was any correlation between bad weather and traffic congestion, this will be used towards our prediction algorithm if the query day has bad weather. The fact that the weather was bad can be used towards our model to make prediction more accurate. Hence we will be using a weightage algorithm that will give output based on current traffic conditions; old traffic data as well as weather. The service will ideally be able to suggest routes that avoid areas with a very high projected concentration of traffic. This will allow the user to find an ideal route with minimal exertion. The assimilated data and traffic information could be used in the future to help in improving road infrastructure or traffic diversion based on traffic patterns and weather conditions.

The front end of the software would include a website where:

1. The user will be able to enter the zip code [destination or (source & destination)],
2. The day of the week and
3. The time of the day of the area for which the traffic reports are required.

The service will then consult a directions service, which in our case is Google Maps, in order to find the traffic reports. This route will be highlighted with different colors referring to different intensities of traffic congestion, as per our algorithm. The key for the different colors will be available for the user to understand. Along with the traffic report, the user will be able to see the weather predictions based on the input that the user gives. As add-ons to the traffic and weather forecast, the application will provide the user with an option to look for restaurants and parking lots in the area.

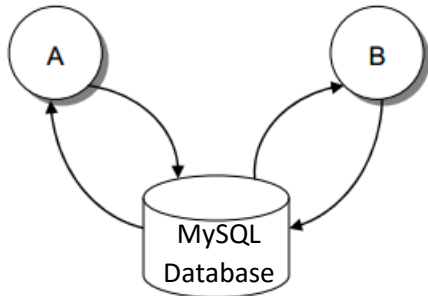
Algorithm:

How does our algorithm predict traffic congestion? A lot of the traffic websites like Yahoo, Bing, and Mapquest use only the current traffic incident reports to predict traffic. A typical use case would be a user logging into Yahoo at 7am to check traffic conditions for a particular highway. Due to it being 7am, Yahoo will show no congestions because normally the congestion starts around 8am. If the user has a long commute, they would get stuck in traffic because by 8AM, the traffic has increased tremendously and it's too late for him to take a detour. Hence to handle that use case, our algorithm would also take into account previous day's data. So when a user does login to our website, they will be given an option to select a time and if the time is past the current time (future), the algorithm will query the database for previous historic data for the same time and make a decision accordingly.

Another aspect of traffic prediction is accident conditions. We all know accidents do not occur every day on the same road; if an accident caused traffic congestion today, it would not cause traffic congestion tomorrow. In order to handle this use case, we will be using a weighted algorithm prediction hereby analyzing current traffic data as well as historic traffic data.

Let's assume the user wants to predict traffic for time 9am and the query was executed at 8am. This means that our current database has traffic incident data up to 8am. Our weighted algorithm will analyze the current traffic data and check whether there has been any change in traffic conditions in the past hour (between 7AM-8AM). If there have been any changes, we know traffic congestion is starting to increase. In this case, our traffic predictions will put more weight on the current traffic and less weight on yesterday's traffic. The reason we chose to have more weight on today's traffic is because if there was an accident in the morning, we simply do not want to predict light traffic based on yesterday's traffic. Additionally, we are also using the weather to help with predictions. If the weather is raining or snowing, it will perform a query on the database for snow/rain days and use the traffic incident data for those days. The weather data will have a weight factor associated as well, if and only if the weather was bad; meaning it rained/snowed.

Data Collection:



Our project website will be accessing data via a shared element as shown on the left. To predict traffic conditions, our algorithm will require traffic incident data. This data has been uploaded by users themselves who are out on the road via their smartphones. Our backend will have a data collection script running continuously that will be pulling data from the Google API and storing them into our local MySQL database. For our prediction, we will be storing data into 3 separate databases, namely historic traffic, current traffic and weather

Database	Data Storage Interval	Total Data Reports
Current	Once every 2 seconds	Max 24 hours
Historic	Once every hour	Max 4 Months
Weather	Once every hour	Max 4 Months

The above table shows the data collection scheme of the three separate databases.

The method we intend to use in realizing this idea is:

- **Current Database:**
The map and 'Traffic information' which will be seen on the site will be gathered from Google Maps. Google maps allow up to 25,000 requests every day. This would be sufficient to make a database with an update on the traffic once every 2 seconds.
- **Historic Database:**
This is similar to our Current Database, however we will be storing once every hour. The major difference between current and historic, besides the update interval, is historic has data upto 4 months whereas current will only have upto 24 hours.
- **Weather Database:**
The 'Weather information' will be gathered from sites like 'wunderground.com' or 'openweathermap.org'

As a side note, please keep in mind "Total Data Reports" mean for how many hours/months of data the data will be stored. If the database has reports for any time greater than that, the newest incident will be saved and oldest incident will be deleted.

Functional features

The previous versions of the project gave traffic predictions for the entire tri-state. They did not have any additional features. This project will bring out some additional features to the application that will enhance the user experience.

- Restaurants and Parking
- Navigation with suggested routes
- Plan a vacation
- Alternative Mode of transportation
- Mobile Android Application

Restaurants and Parking:

There are plenty of applications out there that will give you locations for food on the map. The only issue there is you do not know whether the restaurant is open or not. A lot of times users look up for restaurants late at night or during the night time. Different restaurants have different operating hours. Hence in our application, we will have a layer for restaurants and food, however it will only show you restaurants that are open in the nearby vicinity. No user wants to show up to a restaurant not knowing its closed or will be closed soon. Additionally, when the map points are enabled for the restaurants, it will also display the current estimated time to that place, so a user can make a decision.

Similar to restaurants, we will be adding another layer for parking as well. This can be used to find parking spots in the nearby vicinity. This is different from applications like google because Parking Spots is a layer and can be just enabled easily whereas in google, one has to cancel the current route and perform a search for parking.

Navigation:

Applications with only traffic updates are useless. Hence this application will allow a user to perform a navigation query with an origin and destination address. The user will be able to see the traffic conditions on this navigated route and hereby make a decision whether they want to take the suggested route. The application will also give the user an option to take a detour if they would like.

Vacation Planning:

If a user wants to plan out a vacation in the upcoming weeks, they will have an option to query for weather and possible traffic conditions during the travel days. The application will use the date and the day of the week and perform queries for predicted traffic conditions. This prediction however will be inaccurate due to not having enough traffic data, but this will still be helpful to plan out expected travel time.

Alternate Mode:

If the user's query returns a heavily congested route, and there are no possible detours, the user will be stuck in a limbo. To tackle this condition, our application will also have an alternate mode which will look for closest train stations that the user can take. This will reduce user's overall travel time and hereby tackle traffic.

Mobile Android Application:

Smartphones have become very popular in the past few years and users are very dependent on accessing application on their devices. It would be very convenient for the user to access our map application on their handheld. Hence, we plan to build an android application which will give the user the same features as the desktop application.

Glossary of Terms

1. User: A person who wishes to use the traffic monitoring application to view traffic congestion in a user-entered zip code and be able to choose a faster route to destination
2. ZipCode: A 5 digit number that defines a city or town in the United States. This will be an entry field for the user
3. MySQL Database: A relation database management system that will be used to store traffic incident data as well as weather information
4. Google Maps API: An application programming interface (API) that our system will use to access traffic incident data provided by google. This data will be stored into the MySQL Database
5. Weather Service: A website that contains weather data and will be accessed by our application to store into MySQL Database
6. Traffic Severity: This will depict the different levels of traffic congestion. For the first phase of our project, we will use (Low, Moderate, Moderate High and High)
7. Administrator: A person who is responsible for the management of the traffic monitoring webservice. They also need to ensure the data collection script is also running smoothly
8. Desktop Application: A web application that a user will be able to access on their web browser of a laptop/desktop computer
9. Mobile Android Application: An android application that a user can access on their android smartphone. The application will provide all the features a desktop application provides.

System Requirements

Functional Requirements:

Below are the functional requirements of our application

Identifier	PW	Requirement
REQ 1	5	Upon logging in to the web interface, the application shall prompt the user to enter time of day and zip code and display the most upto date traffic predictions for the entered zip code
REQ 2	5	The application shall display the most upto date weather data.
REQ 3	4	If the user selects Navigation Tab, the application shall prompt the user for a start and end destination and output a navigation route.
REQ 4	5	The application shall collect data in the background for traffic and weather and store them in the database. It also needs to keep the website in sync with the database
REQ 5	3	The application shall give the user to plot the restaurants and parking lots on the map and also display the estimated travel time to each of these applications
REQ 6	3	The application shall give the user an option to select other modes of transportation if there is heavy traffic on the selected route. The output will contain the estimated travel time from the closest train station from the origination to the closest train station from the destination.
REQ 7	4	If there is congestion along the navigation route, the application shall provide the user with an alternate route

Non - Functional Requirements:

Identifier	PW	Requirement
REQ 8	4	The application shall provide the user with a neat and easy to use Interface for the user
REQ 9	3	The application shall overlay the roads with different colors depending on the appropriate traffic conditions experienced at that instant.
REQ 10	2	The application shall provide traffic incident graphing and charting abilities on the webpage (trend over months) for informational purposes
REQ 11	5	The application shall use a local algorithm to predict traffic conditions
REQ 12	2	The application database shall have enough memory to store traffic incident data for over 2 months for different zip codes
REQ 13	3	The administrator shall easily be able to change the time interval of data collection for weather and traffic

On-Screen Appearance Requirements:

Identifier	PW	Requirement
REQ 14	5	Based on the prediction made by the application using the user's input, the application shall display the relevant information in an ergonomic way.
REQ 15	5	The application website shall provide an interface which will display a large size map with pan and zoom functionality.
REQ 16	5	The application website shall provide a key along the side of the map to understand the color code seen on the roads in the map.
REQ 17	3	The application website shall continuously prompt current traffic and weather news of the zip code entered by the user.
REQ 18	5	The application shall provide a drop down menu for all the different features that it provides and for fields that requires the user to make a choice(Ex: time, day of week etc.)

Individual Breakdown

The project can be broken down into 3 mini projects. These projects are parts that can be developed without any dependency. As time passes, these mini-projects will be integrated into our final project

- 1) Kush and Tejas: Kush and Tejas will work on the data collection piece of the project. The historic data collection is very vital since all the functions and algorithms are dependent on it.
- 2) Kisholoy and Shreyas: Kisholoy and Shreyas will be doing the development of the website and designing the user interface. This is very vital since this is the front end of the application and how the user will be accessing the service.
- 3) Jeff and Aaditya: Jeff and Aaditya will be working on the algorithm development and develop functions to integration our data on the website map.

Functional Requirement Specification

Stakeholders

- Users
- Administrators

Actors and Goals

- User:
 - Initiating Actor
 - The user will be accessing the application on their web client to obtain traffic predictions and also navigation between a start and end point. The application will also be providing other features that will go hand in hand with the traffic predictions
- Administrator:
 - Initiating Actor
 - The administrator's primary role is to ensure the traffic and weather data collection modules are working smoothly. The administrator must also ensure the webserver is in sync with the database (display the most up-to date traffic information).
- Backend Script:
 - Initiating Actor
 - The backend script is very vital for data collection. The script will be running continuously on the backend and collect traffic and weather data at periodic intervals. The intervals are defined in the "Data Collection" sections of this document. This data will be store into a MySQL database.
- Database:
 - Participating Actor
 - The database will serve as a central repository for all the traffic and weather data written by the backend script.
- Mapping Service:
 - Participating Actor
 - The goal of the mapping service is to plot the output of the traffic prediction algorithm on the map for the user to display. It will have different colors to depict different congestion levels.
- Weather Service:
 - Participating Actor
 - The goal of the weather service is to provide data to our application to store into the database. The weather service will have the data available and the backend script will be grabbing the data from the website
- Traffic Service:
 - Participating Actor
 - The goal of the traffic service is to provide data to our application to store into the database. The traffic service will have the data available and the backend script will be grabbing the data from the API.

- Restaurant Service:
 - Participating Actor
 - This is one of the enhancements our application will provide. The restaurant service will display nearby restaurants in the nearby vicinity that are open at the time of execution. It will not display restaurants that are closed at the time of the day. It will also display the estimated travel time to each of these restaurants which the user can decide to choose.
- Parking Lot Service:
 - Participating Actor
 - This is one of the enhancements our application will provide. The parking lot service will display the nearby parking lots/ decks in the vicinity.

Use Case Casual Description

UC-1: Get Traffic Predictions and Weather Data

User will enter the time of day and the zip code into the application. Current traffic predictions with traffic incidents and weather data will be displayed to the user.

Requirements: 1, 2

UC-2: Navigation

User will enter a starting and ending locations into the application. The application will display a navigation route with an estimated travel time. The user can view traffic incidents along the way, as well.

Requirements: 3

UC-3: Data Collection

A script will be running to continuously collect traffic and weather data and store the data in the database. The script will also keep the website updated with the latest data.

Requirements: 4

UC-4: Adjust Weather and Traffic Collection

The administrator will be able to change the time interval of data collection for weather and traffic.

Requirements: 13

UC-5: Restaurant/Parking lot/Alternate route

User can plot restaurants and parking lot spaces on maps. He can also view an alternate route if he observes heavy traffic.

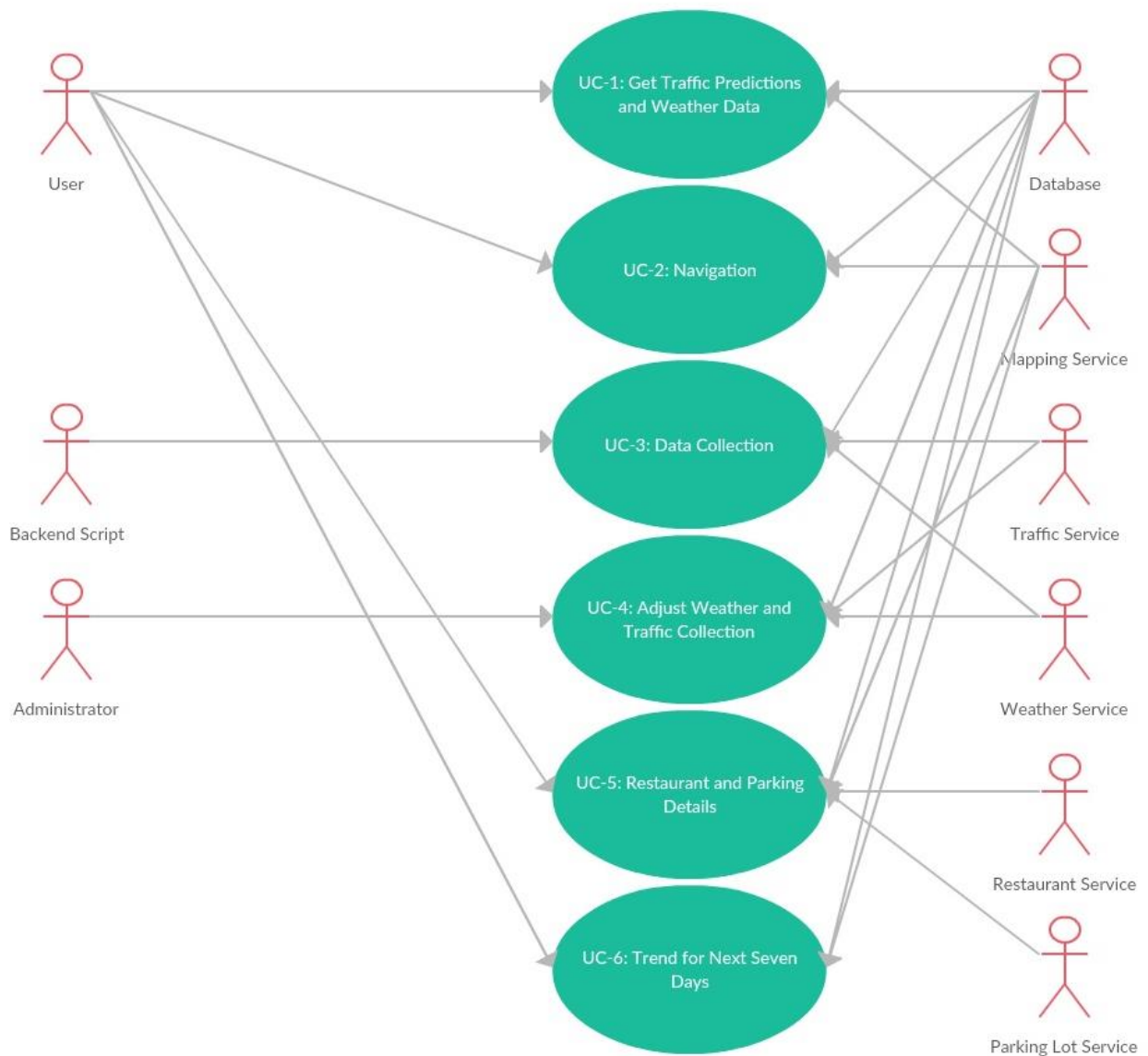
Requirements: 5, 7

UC-6: Seven Days Trend

Once the user logs in, he can view traffic trend for next seven days on his home page. If the user continues as a guest user he can view traffic trend for New Brunswick, NJ as it is the default location of the map.

Requirements: New

Use Case Diagram



Traceability Matrix

Requirement	Weight	UC-1	UC-2	UC-3	UC-4
REQ-1	5	X			
REQ-2	5	X			
REQ-3	4		X		
REQ-4	5			X	X
REQ-5	3		X		
REQ-6	3		X		
REQ-7	4		X		
REQ-8	4	X	X		
REQ-9	3	X	X		
REQ-10	2				
REQ-11	5	X	X		
REQ-12	2			X	X
REQ-13	3			X	X
REQ-14	5	X	X		
REQ-15	5	X	X		
REQ-16	5	X	X		
REQ-17	3	X		X	X
REQ-18	5	X	X		
Weight		45	46	13	13

Fully Dressed Use Cases

Use Case UC-1: Get Traffic Predictions and Weather Data

Related Requirement: REQ 1, REQ 2

Initiating Actor: User

Actor's Goal: To obtain current traffic predictions and weather data

Participating Actors: Database, Mapping Service

Preconditions: User has logged into the application

Postconditions: Application has displayed traffic predictions and weather data

Flow of Events for Main Success Scenario:

1. **User** enters time of day and zip code into the application
2. **System** requests traffic and weather data from the **Database**
3. **Mapping Service** will display the traffic predictions and weather data

Flow of Events for Alternate Scenarios

1. **User** enters invalid information
 - a. **System** prompts user to re enter zip code and time of day
2. Requested data is unavailable
 - a. **System** will display that the service is currently unavailable

Use Case UC-2: Navigation

Related Requirements: REQ 3

Initiating Actor: User

Actor's Goal: To obtain a navigation route from their starting location to their destination

Participating Actors: Database, Mapping Service

Preconditions: User had logged into the application

Postconditions: Application has displayed a navigation route and guided the user to their destination

Flow of Events for Main Success Scenario:

1. **User** enters starting location and destination into the application
2. **Mapping Service** displays the most direct route to the destination
3. **System** requests traffic data from **Database** along the suggested route, checking for heavy congestion
4. **User** navigates along the suggested route and reaches the destination

Flow of Events for Alternate Scenarios:

1. User enters invalid information
 - a. **System** will prompt the user to re-enter starting location and destination
2. Navigation route not available
 - a. **System** will display that the service is currently unavailable
3. Heavy traffic detected
 - a. **System** will request a less congested route
 - b. **Mapping Service** will display the route

Use Case UC-5: Restaurant and Parking lot details

Related Requirements: 5

Initiating Actor: User

Actor's Goal: To plot Restaurants and Parking lot details on map.

Participating Actors: Database, Mapping Service, Restaurant Service, Parking Lot Service

Preconditions: User had logged into the application

Post conditions: Application has displayed Restaurants and Parking lot details on map.

Flow of Events for Main Success Scenario:

1. **User** enters the location address where we wants to view Restaurant and Parking lot details.
2. **Mapping Service** displays the details.
 - a. **Restaurant Service** will display restaurants and their estimated travel time
 - b. **Parking Service** will display parking lots and their estimated travel time

Flow of Events for Alternate Scenarios:

1. User enters invalid information
 - a. **System** will prompt the user to re-enter location address

Use Case UC-6: Trend for next seven days

Related Requirements: NEW

Initiating Actor: User

Actor's Goal: To view traffic trend for next seven days

Participating Actors: Database, Mapping Service

Preconditions: User had logged into the application

Post conditions: Application has displayed traffic trends for next seven days on map.

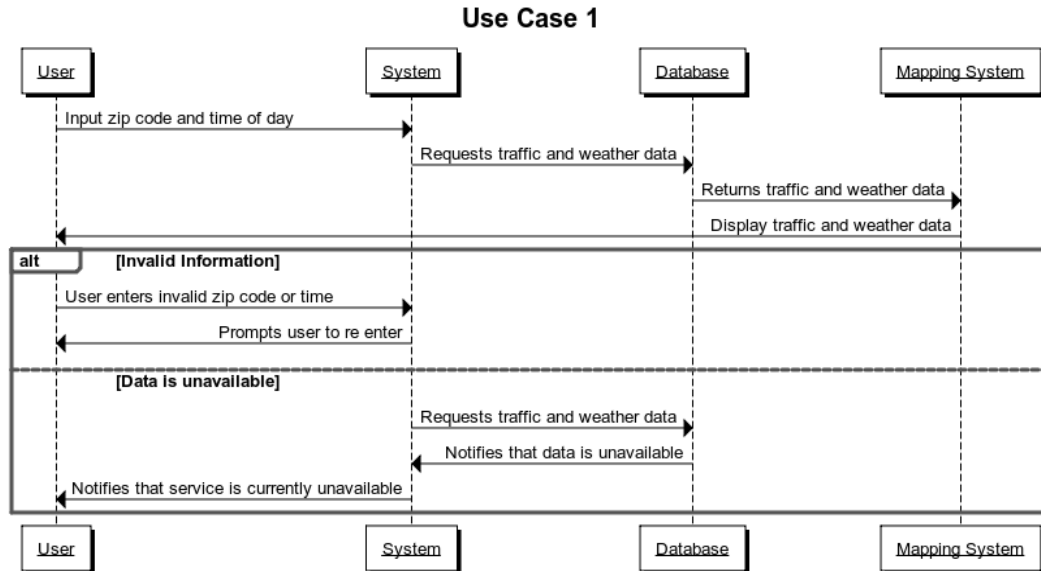
Flow of Events for Main Success Scenario:

1. User logs into the application
2. **Mapping Service** displays the details.

Flow of Events for Alternate Scenarios:

1. User is using the application for the 1st time.
 - a. **System** will fetch default zip code weekly trends and display to the user.
2. User enters invalid information
 - a. **System** will prompt the user to re-enter zip code for which the trends are required
3. Requested data is unavailable
 - a. **System** will display that the service is currently unavailable

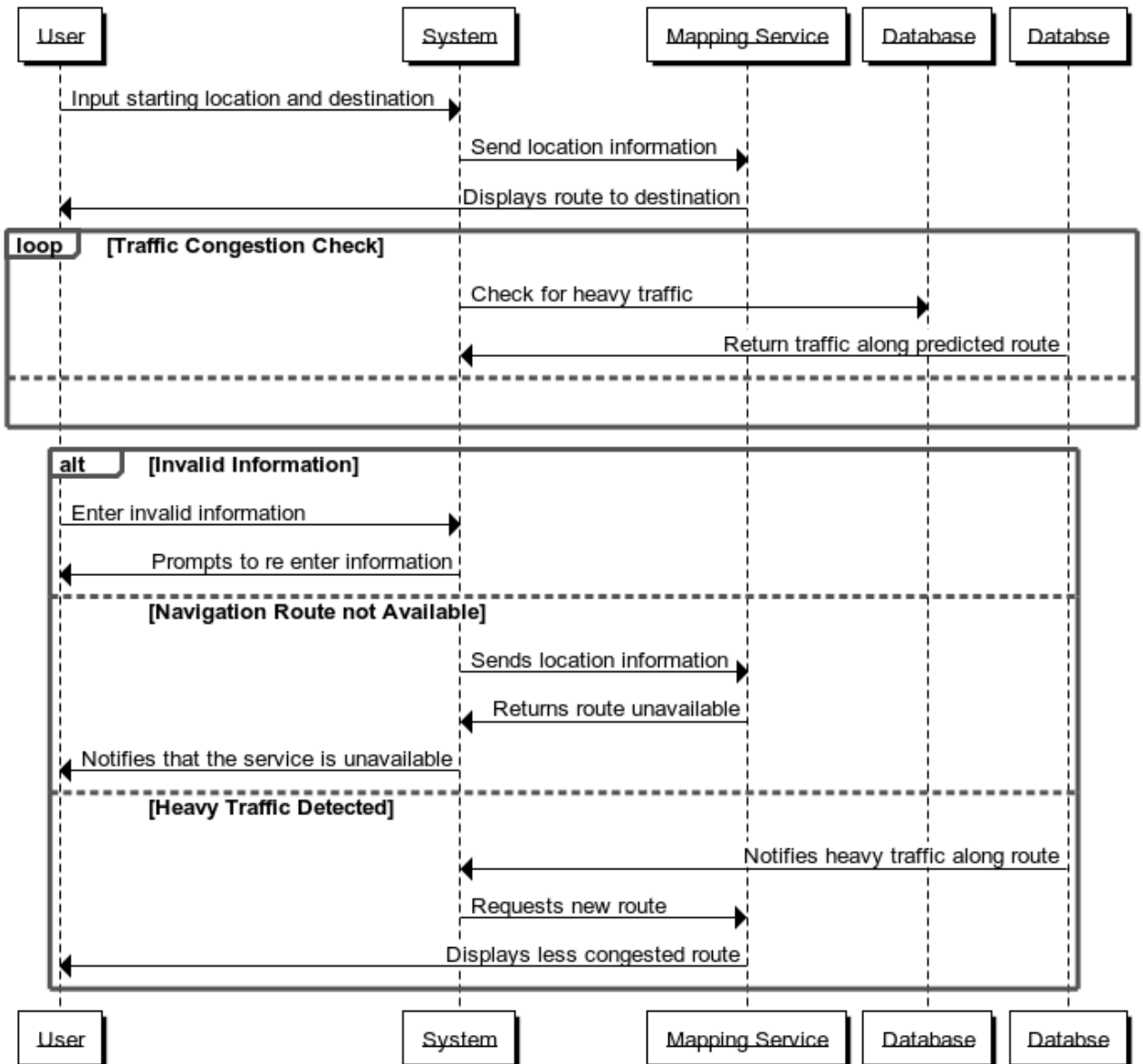
System Sequence Diagrams



This system diagram shows the normal flow of **Use Case 1** and the alternate flow for if the array is empty. The Sequence Flow can be summarized as:

- The user will enter the application with his login credentials.
- The User will then enter the data and submit it on the Application.
- The Application queries the database for traffic and weather data within the inputted zip code and time of day.
- The Application then calls the Mapping class to process these inputs, and calls the Database requesting the traffic and weather points in the area desired.
- The Mapping class parses the information and constructs the link to the Mapping Web Service.
- The Mapping Web Service is called and returns a map image to the Application. The Application will display the image for the user.
- For the alternate flow if the array of data for any specific area is empty, the map will show the zip code still, but a text box will appear at top the show that the zip code has no traffic history.

Use Case 2



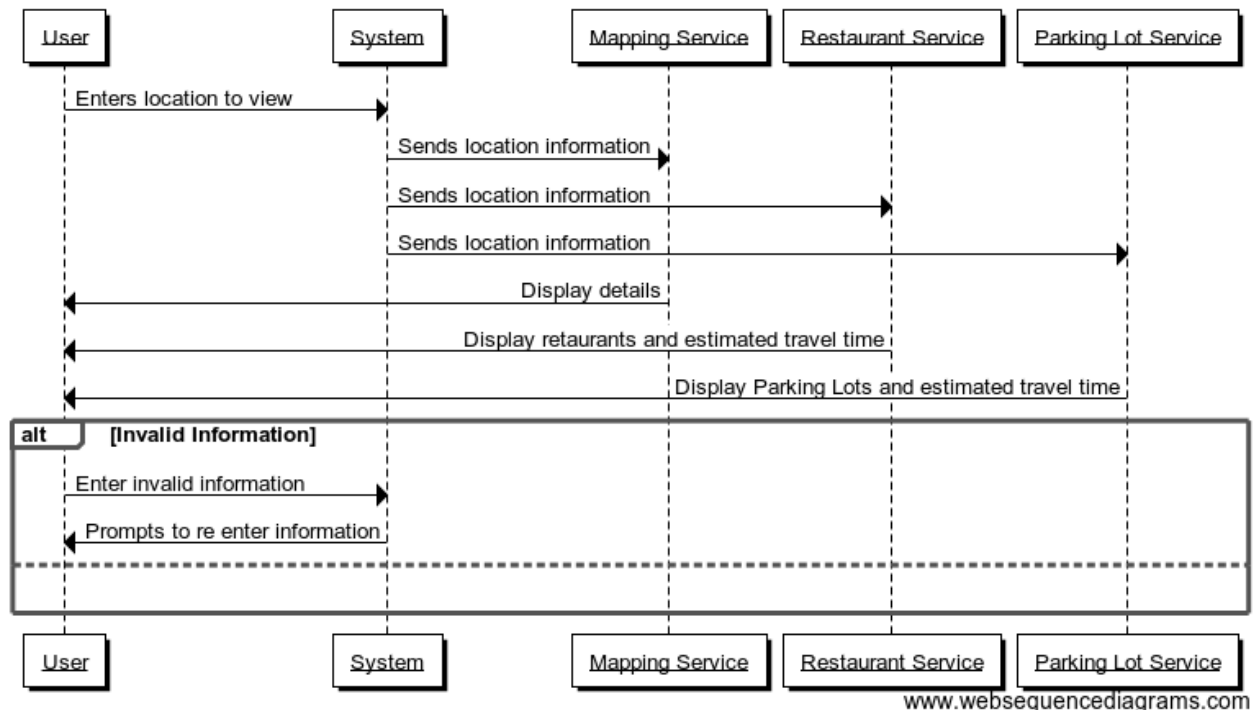
www.websequencediagrams.com

This system diagram shows the normal flow of **Use Case 2** and the alternate flow for if the array is empty. The sequence can be summarized as

- The user will enter the application with his login credentials.
- The User will then enter the data (start and destination zip codes) and submit it on the Application.
- The Application calls the Mapping Class with the given inputs and initializes the Map.
- The Class then calls the Prediction model that runs the data from the database in a loop with the current input from the user to generate results.
- The Database returns an array of results.

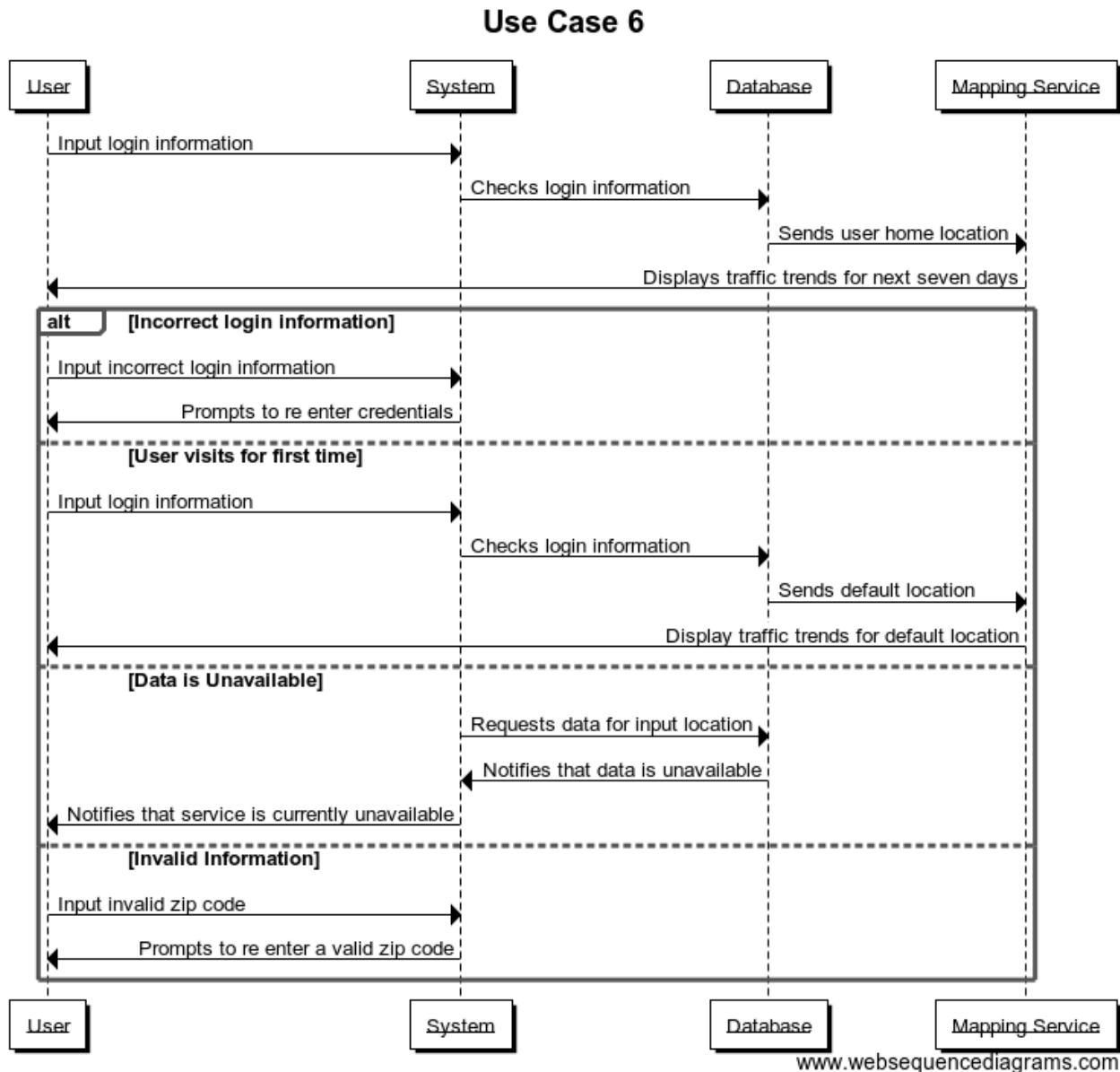
- The Mapping class parses the data from the Mapping Service and the processed data and constructs an image URL to display.
- The Application displays the image of the route along with the directions.
- For alternate flow if the array of data for any specific area is empty, the map will show the zip code still, but a text box will appear at top to show that the route between target zip codes has no traffic history.

Use Case 5



This system diagram shows the normal flow of **Use Case 5** and the alternate flow for if the input is invalid. This use case is designed to be present in the Traffic as well as Navigation flow. The Sequence Flow can be summarized as:

- The user would be on the traffic or navigation page and will click on the buttons corresponding to the additional services like restaurant/parking lot service.
- This will lead to the system interacting with the mapping service to initialize the map for the location that is desired.
- Then the information based on the click of the additional service button will call the RestaurantService/ ParkingLotService function that will be responsible for gathering that service's information from the database and will result in markers places on the map around the desired location
- For alternate flow, we consider a case where the user inputs invalid information for location. In this case, the system will check the database and verify if the zip code is present. If it's not present,, the system will display an error asking the user to re-enter information.



This system diagram shows the normal flow of **Use Case 6** and the alternate flow for other possible conditions. The Sequence Flow can be summarized as:

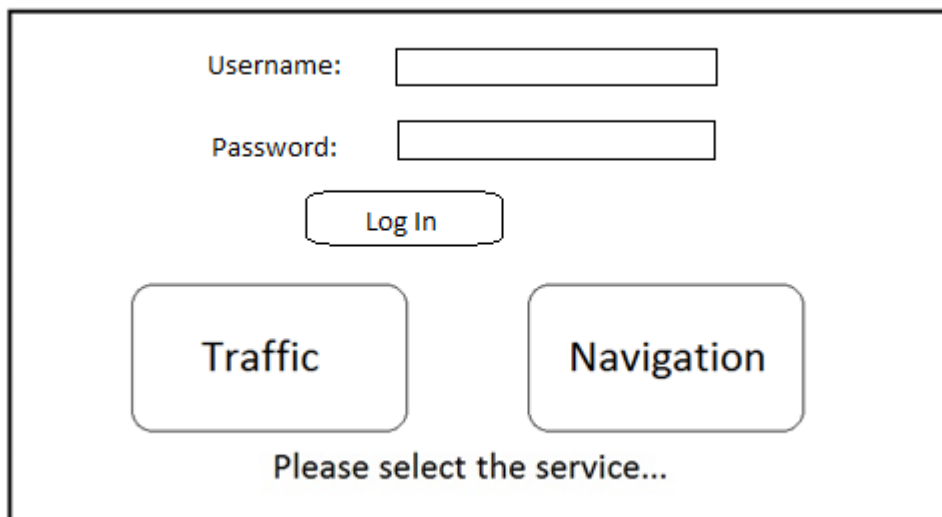
- The user will log into the application with his login credentials.
- The system will then verify the user's login credentials by checking the database.
- Along with the login checking, the system will provide traffic trends based on the most frequent areas (zip codes) that the user has searched which will be fetched from the database.
- A trend for the 7 days of the week of the area will be displayed for the user to view.
- The user will have an option to enter a zip code (area) for which he/she would like to see the trends and the flow will be similar to what is mentioned above.
- For alternate flow, we can consider 3 cases: i.e. when a user uses the application for the first time or when the user enters invalid information or the information for that area is currently unavailable.

- a) In the case that the user logs in for the 1st time, the system will fetch the default zip code from the Database (New Brunswick, in our case) and provide trends for this area. The user can then input values for other results.
- b) In both the other cases the system will check the database to find valid outputs and when output is not found, the system will prompt an error message to the user asking to re-enter the information.

User Interface Specification

Preliminary Design

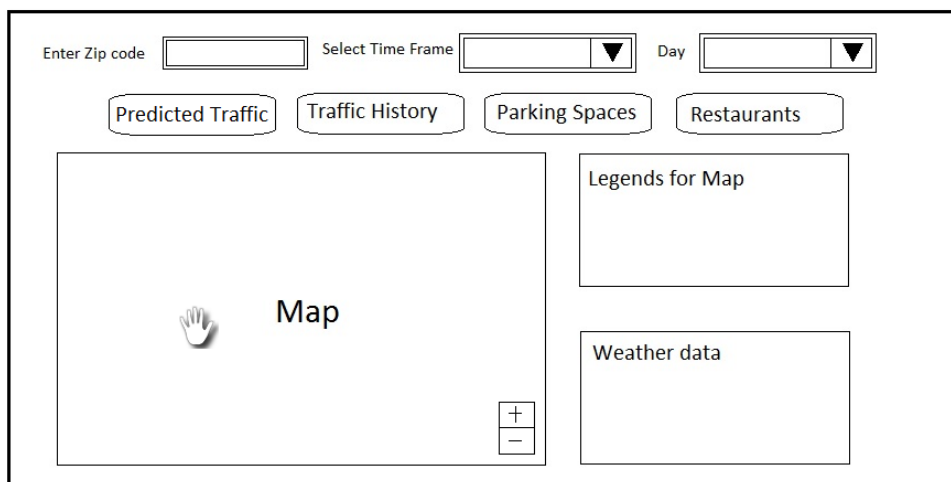
Preliminary User Interface design for fully dressed **Use Case - I**



The diagram shows a home screen window with the following elements:

- Username:** A text label followed by a rectangular input field.
- Password:** A text label followed by a rectangular input field.
- Log In:** A rounded rectangular button centered below the password field.
- Traffic:** A large rounded rectangular button on the left.
- Navigation:** A large rounded rectangular button on the right.
- Please select the service...:** Text centered below the Traffic and Navigation buttons.

Fig. 1: Home screen window options.



The diagram shows a Basic Service window with the following elements:

- Enter Zip code:** A text label followed by a rectangular input field.
- Select Time Frame:** A text label followed by a dropdown menu with a downward arrow.
- Day:** A text label followed by a dropdown menu with a downward arrow.
- Predicted Traffic:** A rounded rectangular button.
- Traffic History:** A rounded rectangular button.
- Parking Spaces:** A rounded rectangular button.
- Restaurants:** A rounded rectangular button.
- Map:** A large rectangular area containing a hand cursor icon and the word "Map". It has zoom-in (+) and zoom-out (-) buttons in the bottom right corner.
- Legends for Map:** A rectangular box to the right of the map.
- Weather data:** A rectangular box below the legends.

Fig. 2: Basic Service window.

Stepwise data entry and displaying of on-screen results:

- 1) Select the 'Traffic' service on homepage (fig.1) by clicking the appropriate option.
- 2) A New window (fig.2) will appear on the screen.
- 3) Enter the zip code, select the time frame and day of the week in respective fields.
- 4) Select 1 or many out of 4 options, viz. Current Traffic, Traffic history, Restaurant Suggestions and Parking Availability.
- 5) Corresponding information will be shown on the map. Also weather update for given time frame, zip code and day of the week will be shown regardless of selected service.
- 6) User can later zoom in/out and move the map using pan cursor.

User Effort Estimation

The following data shows the effort that needs to be put in by the user to get all the information required, including mouse clicks and keystrokes.

1) NAVIGATION:

The basic requirements to get traffic data of a desired destination would consist of 3 mouse clicks, as follows:

1. Open preferred web browser to load project website.
2. Click on the required service (Traffic Info or Navigation) to enter their designated pages.
3. Click "Show Traffic" to display the traffic in the selected region on entered day (after completing data entries as shown below).

The add-ons to get information about the available Parking lots and Restaurants in the region will be displayed on each of the pages and would require one click each to display the result.

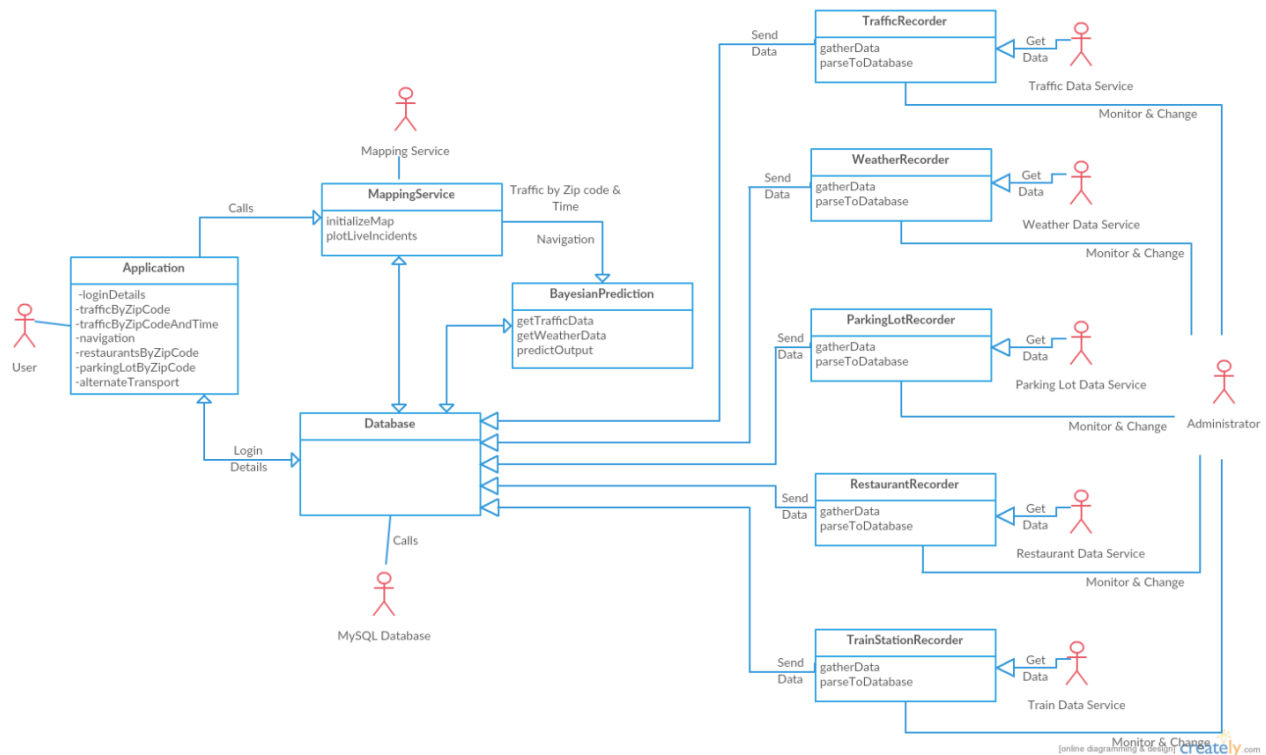
2) DATA ENTRY:

Total of 8 mouse clicks and 5 keystrokes will be required to complete a selected service.

1. Click the Text Box to type zip code of the destination to travel to (assisted by a drop down with prediction of destination name based on user input).
2. Click on the Drop down Menu to Click and select the day of travel.
3. Click on the Drop down Menu to Click and select the time frame.
4. Click the radio button to display the result of whichever service (Traffic History/Restaurant Location/Parking Lot) the user wants to see.
5. The map will be equipped with a zoom and pan functionality which will require a mouse click and drag respectively.

Domain Analysis

Domain Model



The Domain Analysis figure above shows the analysis of related software systems in a domain to find their common and variable parts. From the above diagram, it is clear that the User has access only to the Application. To keep the User experience easy and personalized, the User interface is made basic and has a login to prompt the user based on the history of how they used the application. The login details of the user are verified from the database and his most frequently searched result is prompted. Based on the input from the User, the Application calls Mapping Class. All of the functionality of the application happens through this specified Mapping Class. We are making use of two different paths thereon to provide functionality to the application. One path is where the map has to display values only by taking using entries present in the database. In this path, once the user input is got the Map is initialized and the database is invoked to provide data which is used to display the result on the map. The other path makes use of the Predictive Model to provide the best guess of the data requested by the user. Here, once the user inputs his data, the Map is initialized and then the prediction class is called to use the input and derive a result. The prediction class will invoke the database and use the current data inputs and database entries to compute the result which will be given back to the Mapping service to display.

The Administrator has control of the other features that can receive data from Web Services. The administrator will monitor the data that comes from these Services. These scripts

gather and parse the data received from the Web Services. The Administrator can use these scripts as per the requirement of the application and then pass required information to be stored in the database which in turn is used by the main web application.

To provide a personalized touch to the application, a login feature has been incorporated. This functionality allows the application to access the database at the time of login and prompt or predict the user's general choice location and time to display information based on the users data stored in the database. The application sends and receives data from the database where the User details are stored. Also, a few new features have been added keeping in mind a user who is travelling to an unknown location. The user will be able to find parking lots which are accessible to the general public based on the data that we receive from the Web service which is parsed through the ParkingLotRecorder Class and stored in our Database. The application will also provide restaurant locations at the destination location. The data of the restaurants which are got from the Web Services will also be parsed through the RestaurantRecorder Class and stored in the database. As an add-on, the application will also suggest closest train stations and a train schedules based on the ZipCode of the source and destination addresses fed in by the user. The data for the parking lots and the restaurants will be fed into the database fewer times than the weather and traffic data because it is less prone to frequent changes.

Use Case 1: Get Traffic Predictions and Weather Data

Concept Definitions

Responsibilities	Type	Concept Name
Allows user to login into the application	D	Application
Displays the predicted traffic congestion and weather data based on zip code and time	D	Application
Contains the traffic and weather data	K	Database
Predicts the future traffic congestion	D	BayesianPrediction
Initialize the map and plot the traffic congestion	D	MappingService

Association Definitions

Concept Pair name	Associated Definition	Association Name
Application - MappingService	Application sends the location address.	Plot the graph
BayesianPrediction - MappingService	BayesianPrediction sends the predicted traffic congestion on the route	Provides data

MappingService - Database	MappingService requests for the data from Database and then plots on the graph	Provides data
BayesianPrediction - Database	BayesianPrediction requests for historical data from the database.	Provides data

Attribute Definition

Concept	Attribute	Attribute Description
Application	weather	Weather of the area which user has queried
	time	Time of day
	zipCode	Zip code of source or source
Database	trafficData	Live and historical traffic data
	weatherData	Live and historical weather Data
BayesianPrediction	futuretrafficdata	Predicted congestion data
MappingService	initializeMap	Load the map based on entered zip code

Use Case 2: Navigation

Concept Definitions

Responsibilities	Type	Concept Name
Allows user to login into the application	D	Application
Displays the predicted traffic congestion and weather data based on zip code and time	D	Application
Contains the traffic and weather data	K	Database
Predicts the future traffic congestion	D	BayesianPrediction
Initialize the map and plot the traffic congestion	D	MappingService
Plot the suggested route from source to destination	D	Application

Association Definitions

Concept Pair name	Associated Definition	Association Name
Application - MappingService	Application sends the location address.	Plot the graph
BayesianPrediction - MappingService	BayesianPrediction sends the predicted traffic congestion on the route	Provides data
MappingService - Database	MappingService requests for the data from Database and then plots on the graph	Provides data
BayesianPrediction - Database	BayesianPrediction requests for historical data from the database.	Provides data

Attribute Definition

Concept	Attribute	Attribute Description
Application	weather	Weather of the area which user has queried
	time	Time of day
	zipCode	Zip code of source or source
Database	trafficData	Live and historical traffic data
	weatherData	Live and historical weather Data
BayesianPrediction	futuretraffidata	Predicted congestion data
MappingService	InitializeMap	Load the map based on entered zip code
	suggestedRoute	Load the map based on entered zip code and plot the suggested route on the map

■ Traceability Matrix

Use Case	Weight	Application	Mapping Service	Bayesian Prediction	Database	Traffic Recorder	Weather Recorder	Parking Lot Recorder	Restaurant Recorder	Train Station Recorder
UC-1	45	X	X	X	X					
UC-2	46	X	X	X	X					
UC-3	13				X	X	X	X	X	X
UC-4	13				X	X	X	X	X	X

System Operation Contracts

1) **Operation:** Writing into database

Precondition:

Memory is allocated for different type of data in database.

Post condition:

New data is inserted in database.

2) **Operation:** Displaying dynamic data on Navigation

Pre-condition:

Traffic is predicted according to weighted algorithm in application.

Post-condition:

Detour/alternate route is suggested according to the predicted traffic.

3) **Operation:** Service request from database

Precondition:

There is enough data in the database provided by Traffic recorder, weather recorder, restaurant recorder and parking recorder.

Post condition:

This operation returns user with markers overlaying on the map. If there is no data in database, it returns nothing to the mapping devices showing only the blank map. That notifies user that there is no data available in the database for particular user request criteria.

4) **Operation:** Calculate results

Precondition:

Criteria provided by the user must be available in the database.

Post condition:

According to weighted algorithm, calculations for traffic predictions are performed and returned to the application.

Prediction Model

Prediction Strategies:

These are machine learning algorithms we use to predict the future value, when a set of past events is given.

Curve fitting is the basis of prediction. It is a process of drawing a curve that best fits with a given data set. Given a particular set of points, we draw a graph that passes through all the points or closest to all the points and passing through most of them. Thus making it possible for predicting further values. In this process it is important to carefully select the order, there is a chance that the curve may pass too far from the points if the order is too low and if the order is too high there is a chance that there are unnecessary oscillations and the prediction may not be right.

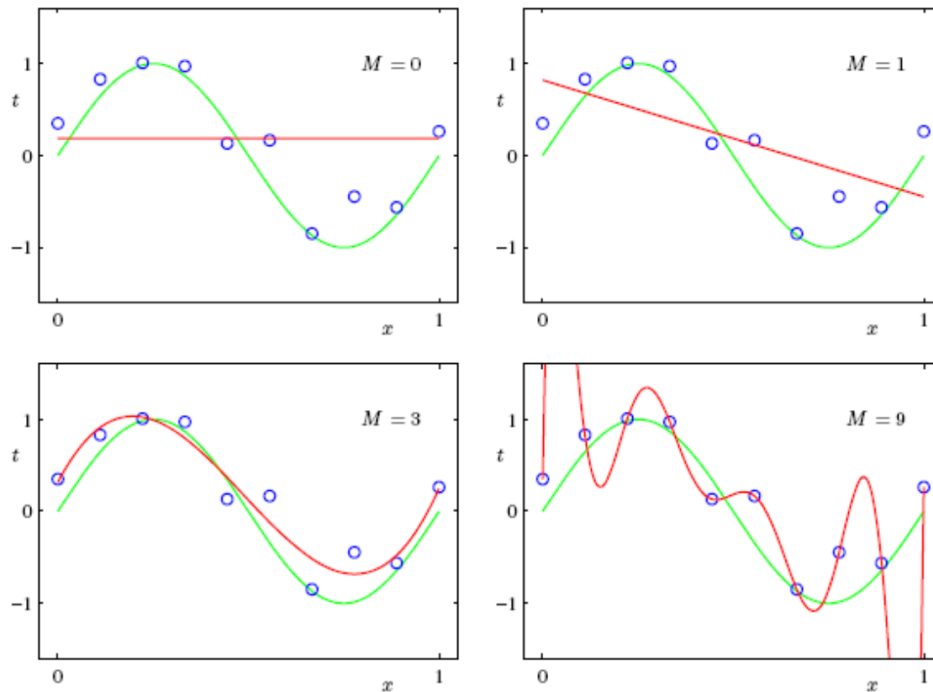


Fig: It shown the curve for different values of order M .

There are many types and strategies used to implement curve fitting. We will be using a couple of them in the project to achieve our results with the most probability of it being right. Some of the strategies used have been described below.

Bayesian Curve Fitting:

The Bayesian Curve fitting uses bayesian probability, that is we try to predict the distribution of data. We consider the previous data as training data X and t and another test point x to predict another t . So the distribution we get is $p(t|x, X, t)$.

$$p(t|x, x, t) = p(t|x, w)p(w|x, t) dw$$

Use of weighted average method in the prediction strategy, giving more weight to inputs that are more relevant to output than the ones less relevant. For example in this project taking the case of predicting the traffic at 4P.M. We give more weightage to the traffic data from previous days than that of the past hour.

In this project, we are taking two datasets in particular a) Historic data collected from a particular time and b) Live data collected in the last 1 hour. In this case when you take the case of a particular time, for example, 5P.M. is the time when people start coming home and traffic suddenly increases. If we use the live data to predict it then it may not show the traffic congestion with more precision than that of the case if we consider historical data. If we consider the historical data of 5P.M. on every working day. Then we will be able to estimate/predict precisely.

Since the historical data is more important for prediction we will be giving it more weight compared to live data. And when we are predicting for a different date since it is not possible to get the live data we just consider the historical data.

Traffic congestion	Weight
High	5
Moderate high	4
Moderate	3
Moderate low	2
Low	1

So in the equation:

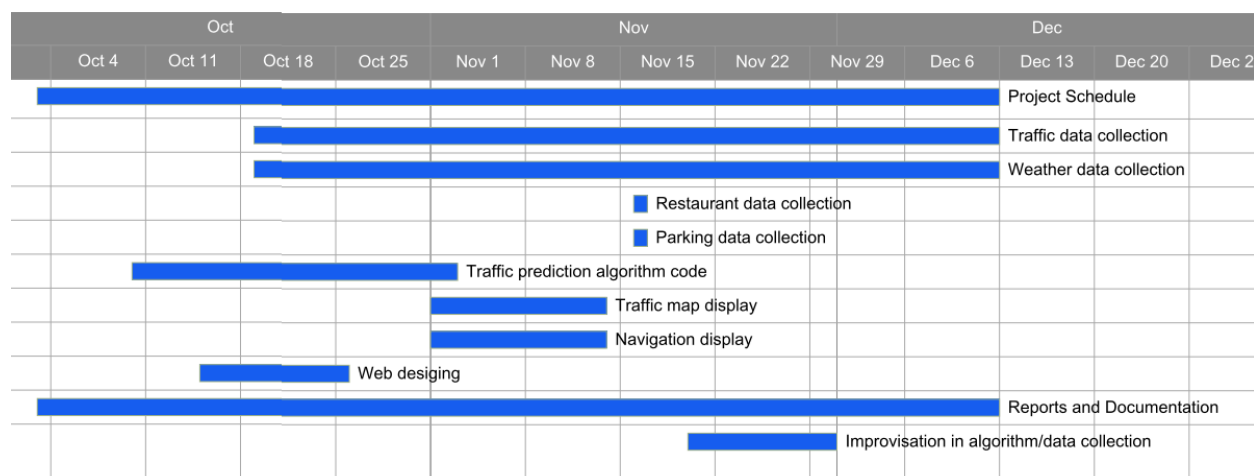
$$p(t|x, x, t) = p(t|x, w)p(w|x, t) dw$$

We take t value as the Weight of congestion and x is date/time so we predict the value of t depending on the test set, that is the future date/time. So that the value we obtain is the congestion weight at the particular time.

Plan of Work

	Task Name	Start Date	End Date	Assigned To	Duration
1	Project Schedule	10/03/15	12/12/15		71d
2	Traffic data collection	10/19/15	12/12/15	Kush, Tejas	55d
3	Weather data collection	10/19/15	12/12/15	Kush, Tejas	55d
4	Restaurant data collection	11/16/15	11/16/15	Kush, Aaditya, Tejas, Jeffrey	1d
5	Parking data collection	11/16/15	11/16/15	Kush, Aaditya, Tejas, Jeffrey	1d
6	Traffic prediction algorithm code	10/10/15	11/02/15	Everyone	24d
7	Traffic map display	11/01/15	11/13/15	Shreyas, K.Vinayak	13d
8	Navigation display	11/01/15	11/13/15	Shreyas, K.Vinayak	13d
9	Web designing	10/15/15	10/25/15	Shreyas, K.Vinayak	11d
10	Reports and Documentation	10/03/15	12/12/15	Everyone	71d
11	Improvisation in algorithm/data collection	11/20/15	11/30/15	Not assigned currently	11d

Gantt chart continued below..



Individual Responsibility Breakdown:

1) Kush and Tejas: Kush and Tejas will start working on traffic and weather data collection shortly. They will help Aaditya and Jeffrey to collect restaurant and parking information data.

2) Kisholoy and Shreyas: As mentioned earlier Kisholoy and Shreyas will be working on web development. Also they will be work on and Navigation and Traffic map display. Web designing is 50% complete.

3) Jeff and Aaditya: Jeff and Aaditya have started working on traffic prediction algorithm. Rest of the groups will be helping them in this task considering it is the most important task of the project.

Apart from this, everyone is participating in making reports. 10 days are reserved for improvisation in algorithm/data collection and additional features (if any).

References

- 1) Chen, H., Roseman, R. and Rule, A. (2015). *Curve Fitting & Multisensory Integration*. [online] <http://www.cogsci.ucsd.edu/>. Available at: http://www.cogsci.ucsd.edu/~ajyu/Teaching/Cogs202_sp14/Slides/lect3.pdf [Accessed 15 Oct. 2015].
- 2) Creately.com, (2015). *Online Diagram Software to draw Flowcharts, UML & more | Creately*. [online] Available at: <http://creately.com> [Accessed 15 Oct. 2015].
- 3) Dev.virtualearth.net, (n.d.). [online] Available at: http://dev.virtualearth.net/REST/v1/Locations?postalCode=07657&key=AvITzUaX5nqGg0JKU43_QSP30BB_ovX37MIND9WahtRbBQpeV4ugoB3HQ60LRxQL8&o=xml [Accessed Oct. 2015].
- 4) GenMyModel, (2015). *Software Modeling in the Cloud*. [online] Available at: <http://genmymodel.com> [Accessed 16 Oct. 2015].
- 5) <https://developers.google.com/maps/?hl=en>, (n.d.). *Google Maps API*. [online] Available at: <http://google.com> [Accessed Oct. 2015].
- 6) Marsic, I. (2012). *Software Engineering*.
- 7) Msdn.microsoft.com, (n.d.). *Find a Location by Address*. [online] Available at: <https://msdn.microsoft.com/en-us/library/ff701714.aspx> [Accessed Oct. 2015].
- 8) Msdn.microsoft.com, (n.d.). *Get Traffic Incidents*. [online] Available at: <https://msdn.microsoft.com/en-us/library/hh441726.aspx> [Accessed Oct. 2015].
- 9) Msdn.microsoft.com, (n.d.). *Traffic Incident Data*. [online] Available at: <https://msdn.microsoft.com/en-us/library/hh441730.aspx> [Accessed Oct. 2015].
- 10) Openweathermap.org, (n.d.). *OpenWeatherMap current weather and forecast*. [online] Available at: <http://openweathermap.org> [Accessed 4 Oct. 2015].
- 11) Project Report Spring 2011, (2011). *Project: Traffic Monitoring*. [online] Available at: <http://www.ece.rutgers.edu/~marsic/books/SE/projects/Traffic/2011-g7-report3.pdf> [Accessed 2015].

- 12) Project Report Spring 2013, (2013). *Traffic Monitoring Service*. [online] Available at: <http://www.ece.rutgers.edu/~marsic/books/SE/projects/Traffic/2013-g7-report3.pdf> [Accessed 2015].
- 13) Smartsheet, (n.d.). *Online Project Management Software | Smartsheet*. [online] Available at: <http://smartsheet.com> [Accessed 15 Oct. 2015].
- 14) Stackoverflow.com, (n.d.). *Stack Overflow*. [online] Available at: <http://stackoverflow.com/questions/33069298/bing-traffic-api-not-returning-any-data-for-accidents-delays> [Accessed 2 Oct. 2015].
- 15) W3schools.com, (2015). *W3Schools Online Web Tutorials*. [online] Available at: <http://w3schools.com> [Accessed 10 Oct. 2015].
- 16) Websequencediagrams.com, (n.d.). *WebSequenceDiagrams.com - Draw and Edit Sequence Diagrams in seconds*. [online] Available at: <http://websequencediagrams.com> [Accessed 8 Oct. 2015].
- 17) Wikipedia, (2015). *Curve fitting*. [online] Available at: https://en.wikipedia.org/wiki/Curve_fitting [Accessed 15 Oct. 2015].
- 18) Youtube.com, (n.d.). *YouTube*. [online] Available at: <http://youtube.com> [Accessed 7 Oct. 2015].

TRAFFIC MONITORING SYSTEM

GROUP NUMBER: 7

REPORT 2



GROUP MEMBERS:

Kush Patel
Tejas Ravi
Jeffrey Gillen
ShreyasBhandare
KisholoyVinayakGhosh
AadityaShukla

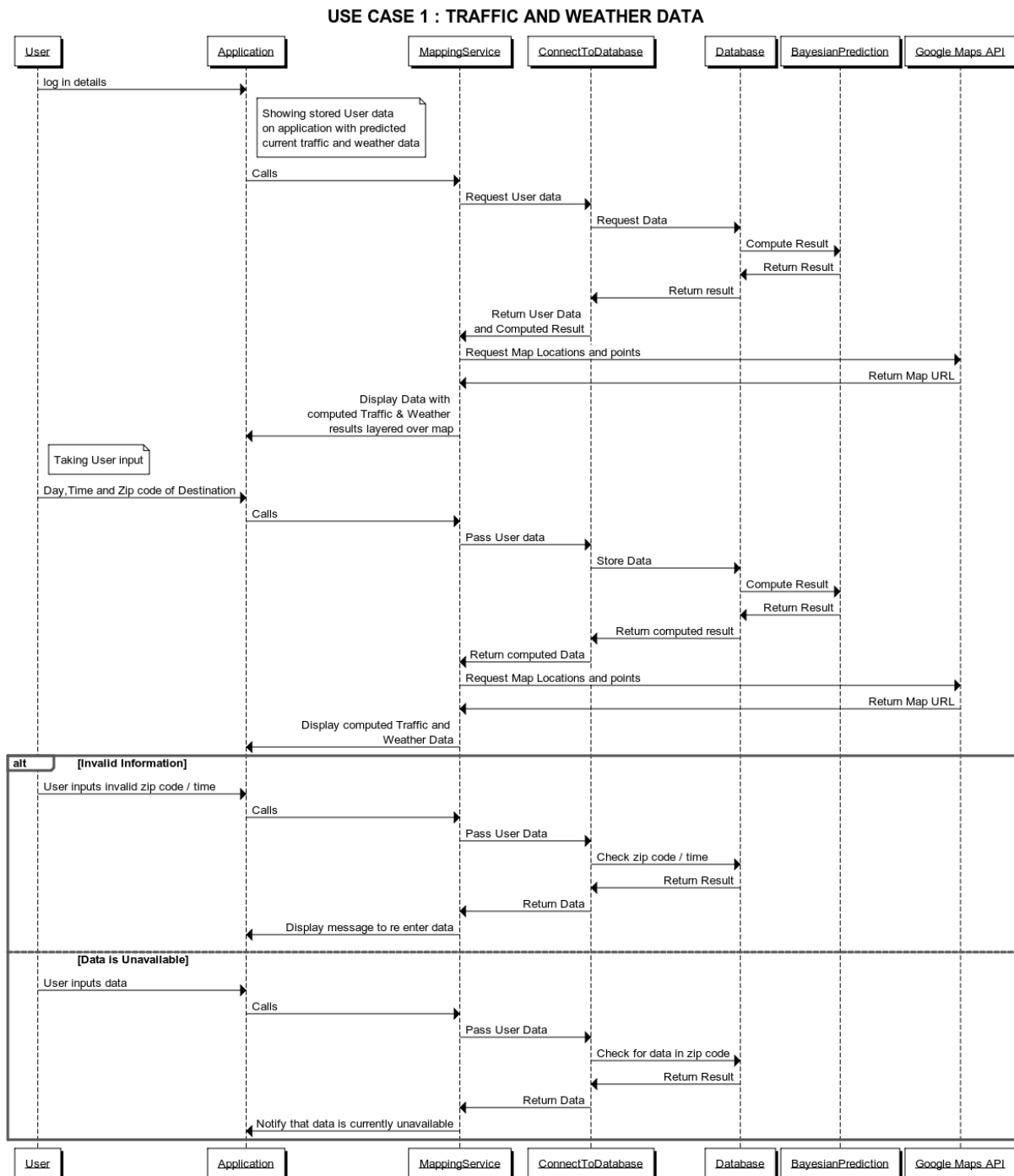
kush.patel@rutgers.edu
tr332@scarletmail.rutgers.edu
jeffrey.gillen@rutgers.edu
shreyas.bhandare@rutgers.edu
kisholoy.v.ghosh@rutgers.edu
aaditya.shukla@rutgers.edu

Table of Contents

I.	Table of Contents.....	2
II.	Interaction Diagrams.....	3
III.	Class Diagrams and Interface Specification	
	• Class Diagrams.....	11
	• Data Types and Operation Signatures.....	15
	• Traceability Matrix.....	18
IV.	System Architecture and System Design	
	• Architectural Styles.....	19
	• Identifying Subsystems.....	20
	• Global control flow.....	21
	• Hardware Requirement.....	21
V.	Algorithms and Data Structures.....	22
VI.	User Interface Design and Implementation.....	24
VII.	Design of Tests.....	27
VIII.	Project Management and Plan of Work	
	• Merging the Contributions from Individual Team Members.....	28
	• Project Coordination and Progress Report.....	28
	• Plan of Work.....	29
	• Breakdown of Responsibilities.....	30
IX.	References.....	31

Interaction Diagrams

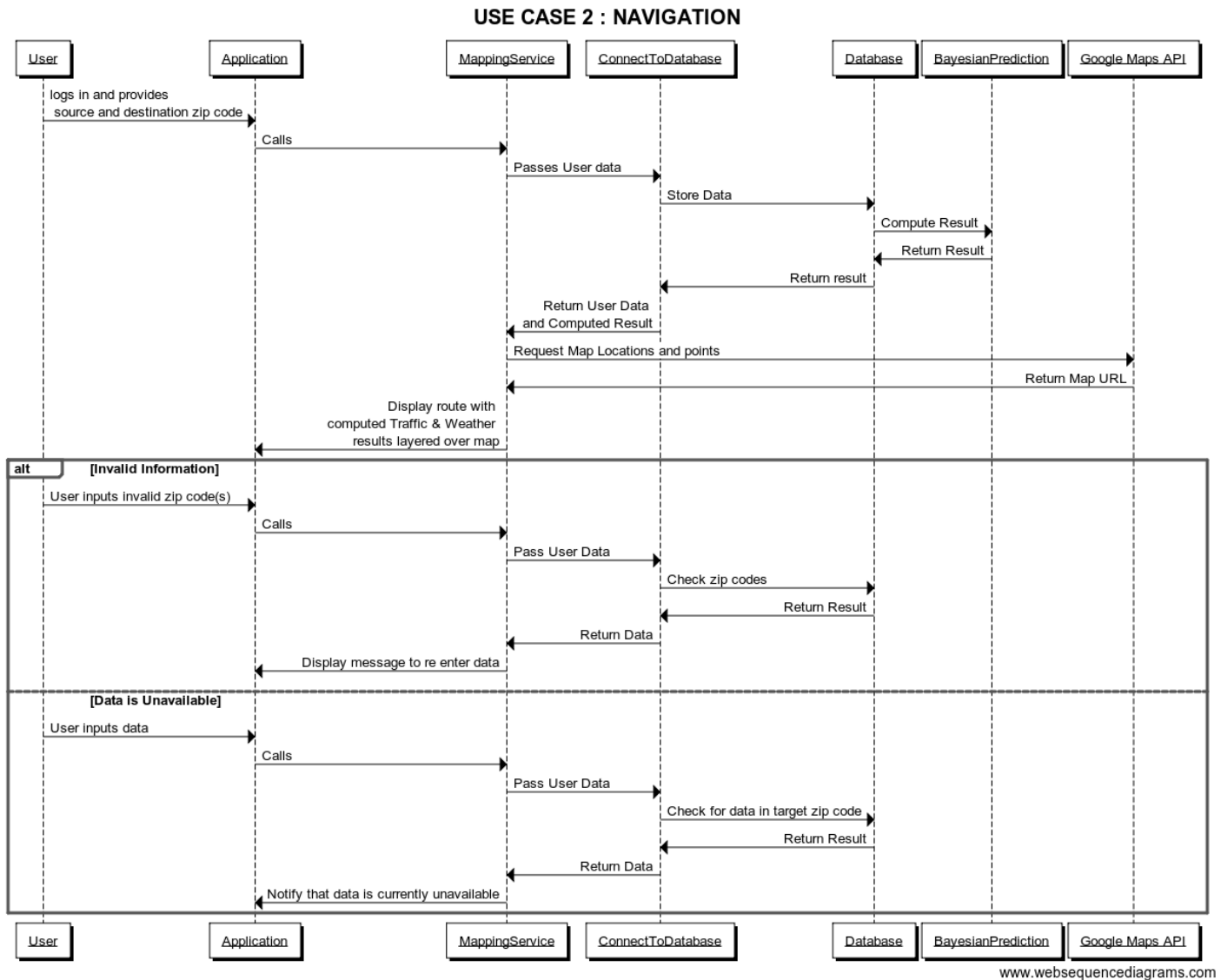
The interaction of the project is broken into 2 basic flows which depend on the path that the user selects: Traffic or Navigation. The add-on services which include the Parking Lot Information, Restaurant Information and Alternate mode information is presented to the user in both these flows.



The above Interaction diagram shows the interaction between various components of the system when the User accesses the Application for Traffic and Weather data (**Use Case 1**). Right after the user logs into the application and chooses the Traffic Service, the application initiates the map by calling the Mapping Service and requests the database for the most frequently searched zip code of the user. This request is made by calling the ConnectToDatabase. ConnectToDatabase is used as an intermediate between the controller and the MySQL database. This ensures that the controller does not do too many computations when attempting to access data from the Database. The Database then returns the information through ConnectToDatabase back to the MappingService, where the MappingService will use that data to call the Google Maps API and place points on a map that is to be returned to the User. The Database also invokes the BayesianPrediction algorithm which calculates the traffic in that zip code based on historical data that is stored in the database. This data is also sent back to the MappingService which overlays the traffic and weather data on the map which the user can see.

The User can then enter information regarding the traffic information they require. Based on these inputs, the Application will call the MappingService to initiate the map and the procedure followed will be similar to the one mentioned above i.e. the MappingService calls the Database using ConnectToDatabase and passes the user input to be stored. The Database will then invoke the BayesianPrediction based on user's current input and history present in the database to compute the traffic results which it send back to the MappingService. The MappingService will then call the Google Maps API to load the points and request for the Map URL. Once that is got, the results of the prediction will be layered on the map and shown to the user.

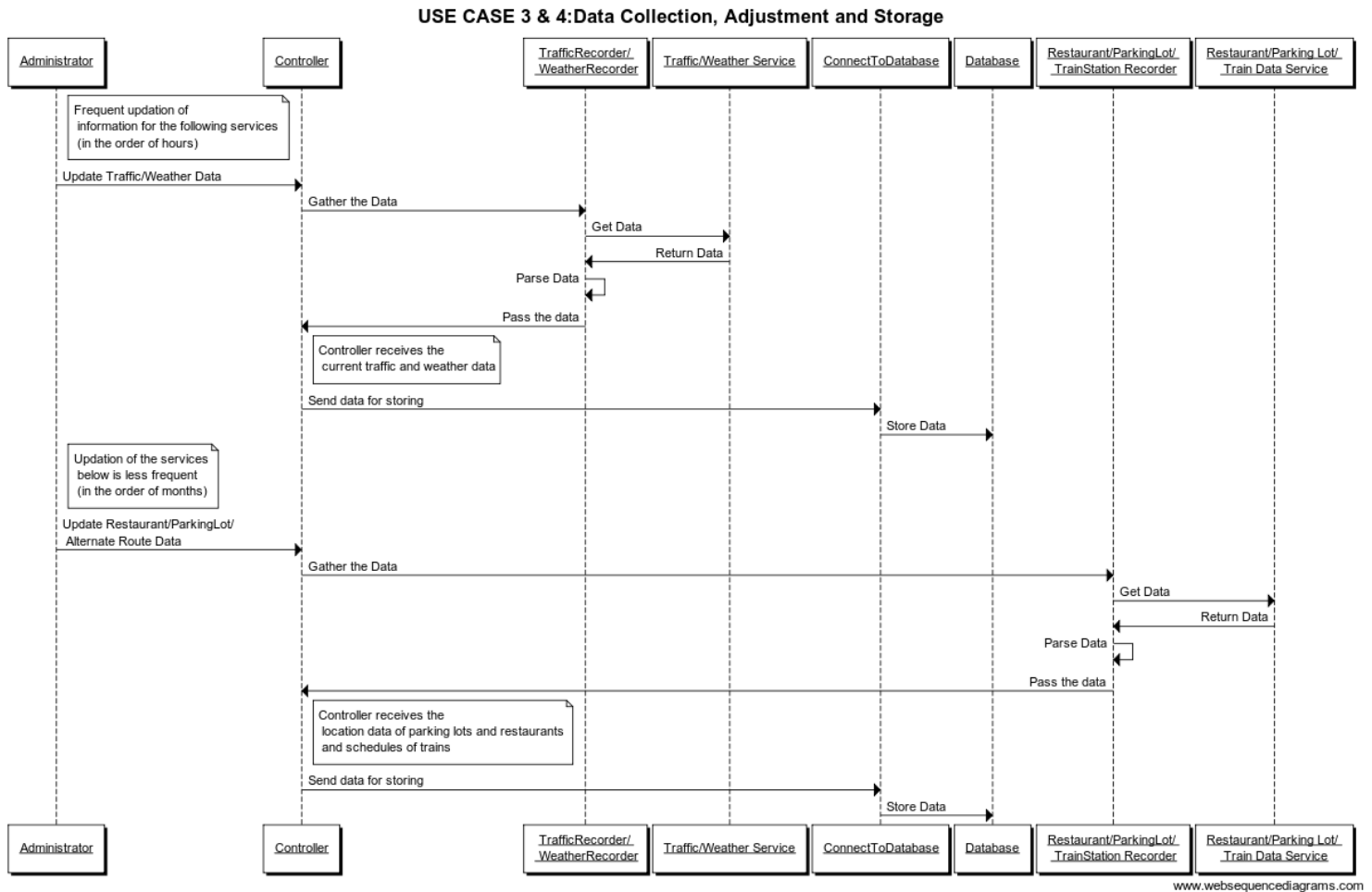
In the alternate paths that this Use Case can take i.e. when the user enters invalid zip codes or the data for the specified zip code is not available, the Application will call the MappingService and pass the user data to the Database using the ConnectToDatabase and request for results. The Database will process the information and return a message through the MappingService to the Application to display the respective error messages.



The above Interaction diagram shows the interaction between various components of the system when the User accesses the Application for Navigation (**Use Case 2**). Here, once the user logs in and chooses the Navigation service, the user will enter the source and destination zip codes. The Application will then call the MappingService to invoke the map. The MappingService will then pass the user data to the Database using the ConnectToDatabase. Once the data is stored in the MySQL database, the BayesianPrediction algorithm is invoked to compute the results based on the users input and historical data that is stored in the Database. This result is then passed back to the MappingService through the ConnectToDatabase. The MappingService then calls the Google Maps API to create the map URL and then layers the traffic and weather data on the map and presents it to the user.

In the alternate paths that this Use Case can take i.e. when the user enters invalid zip codes or the data for the specified zip codes is not available, the Application will call the MappingService and pass the user data to the Database using the ConnectToDatabase and request for

results. The Database will process the information and return a message through the MappingService to the Application to display the respective error messages.

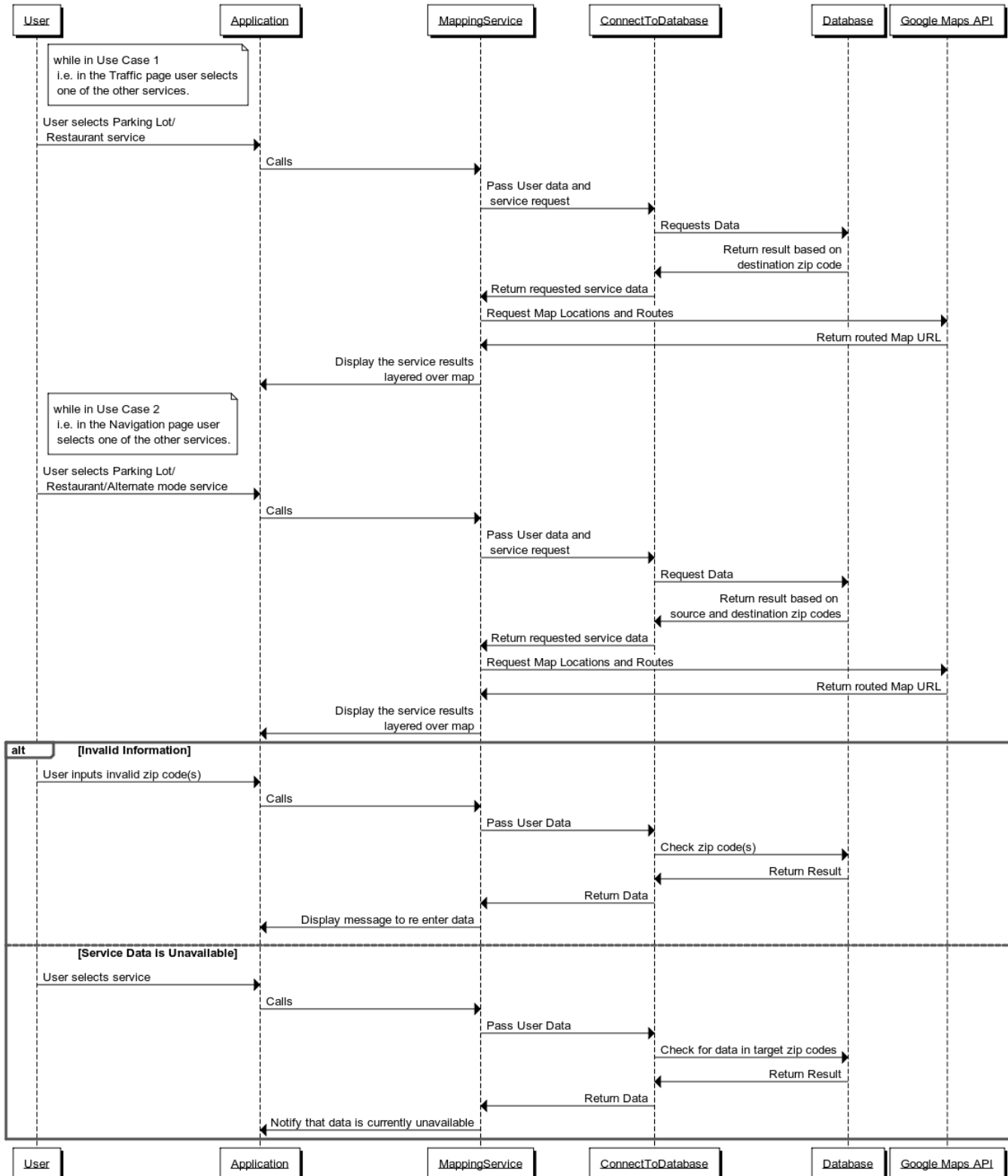


The above Interaction diagram shows the interaction between various components of the system using the Administrator's point of view (**Use Case 3 & 4**). The interactions used for Traffic and Weather data and the add on services i.e. Restaurant, Parking Lot and Alternate Route data are very similar but are shown separately only to specify that the intervals between gathering data for them are different. The approach we take is that the data about Restaurant locations, Parking Lot locations and Train Schedules change less frequently as compared to traffic and weather and hence, the querying of this information is done once every 2 months. However, the traffic and weather data is queried once every 2 hours.

Since the interaction is similar for all services, the Administrator invokes the controller to update the information about the services from their respective websites. The Controller will call the respective recorder. Ex: TrafficRecorder. The TrafficRecorder will gather data from the Traffic Service website (511nj.org) and parse the data back to the controller. The data will include details like latitude, longitude, time and congestion details. The controller will then send this data to be stored in the Database through the ConnectToDatabase. Similar flow and interactions will be noticed in terms of weather services, Parking Lot services, Restaurant

Services and Alternate Route Services which will differ only in the type of information and the interval between each update.

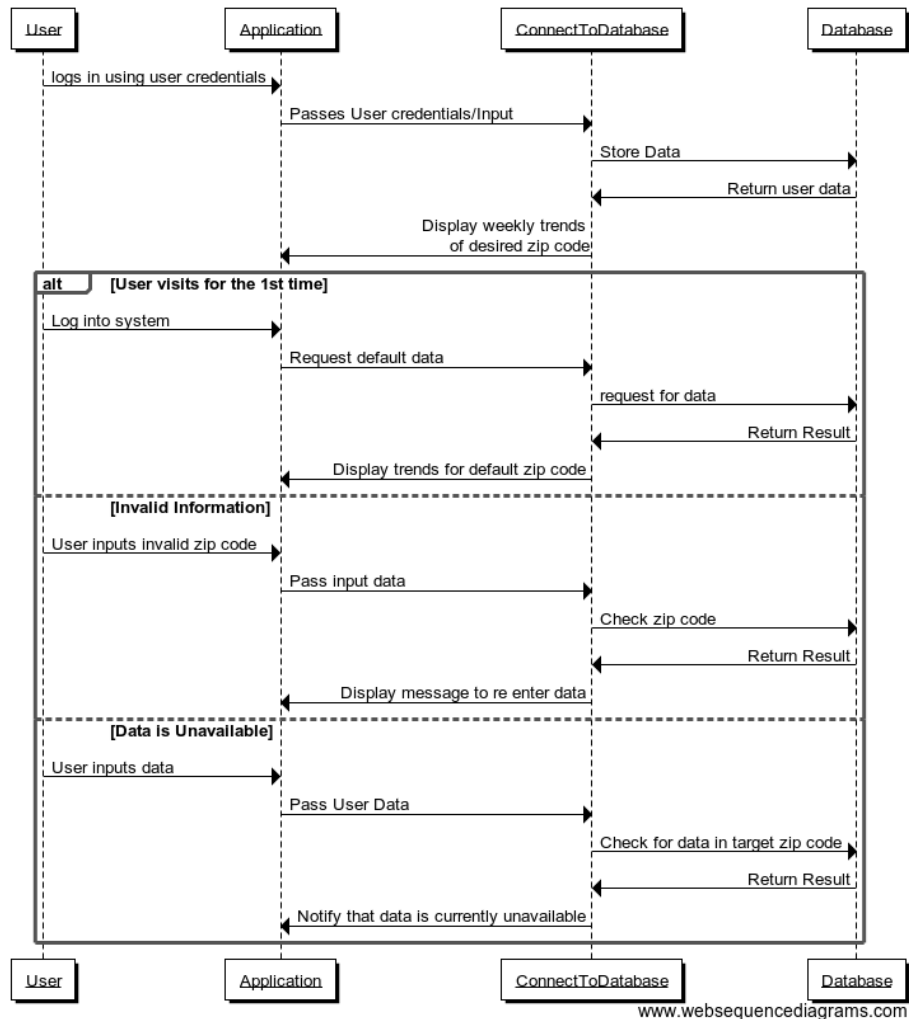
USE CASE 5 : RESTAURANT/PARKING LOT/ALTERNATE ROUTE



The above Interaction diagram shows the interaction between various components of the system when the User accesses the Application for additional Services viz. Restaurant/Parking Lot/Alternate Route services (**Use Case 5**). Since these services are going to be present in both the flows (Use Case 1 & 2) an interaction diagram between the user and the components are shown keeping Use Case 1 and 2 as the basis. The flow is similar in both use cases and includes the user inputting the relevant information(basically the zip codes) and then choosing a service which the user wants to use. The user request is sent to the Application which then calls the MappingService to initiate the map. The MappingService then passes on the user request to the Database through the ConnectToDatabase. The Database has the stored and updated information about these services which is got through the respective web services. The Database then returns the data to the ConnectToDatabase and back to the MappingServices. The MappingServices then requests the Map URL from the Google Maps API and layers the result on the map and displays it to the user.

In the alternate paths that this Use Case can take i.e. when the user enters invalid zip codes or the data for the specified zip codes is not available, the Application will call the MappingService and pass the user data to the Database using the ConnectToDatabase and request for results. The Database will process the information and return a message through the MappingService to the Application to display the respective error messages.

USE CASE 6: WEEKLY TRAFFIC TRENDS



The above Interaction diagram shows the interaction between various components of the system when a User accesses the Application (**Use Case 6**). This includes a case where the user is accessing the application for the first time and the case where the user is a regular. In this use case we are trying to incorporate weekly trends of the traffic patterns in a zip code based on the historical values that are present in the database. As seen in the interaction diagram, when the user logs in to the application, the login credentials of the user are verified and if the credentials match, the trends of the user's most frequently searched zip code is presented as an output on the home page for that user. The interaction starts off with the user accessing the application to log in. This login details is sent across to the MySQL database using the ConnectToDatabase. The database is requested to validate and provide the results in the form of the weekly traffic trends of the area that the user frequently searches. This information is passed back to the ConnectToDatabase and then displayed on the Home Screen of the user in the application. The user has an option to input a zip code for which he/she wishes to see the trends for, as well. The interaction will be similar to what was just described, just that instead of

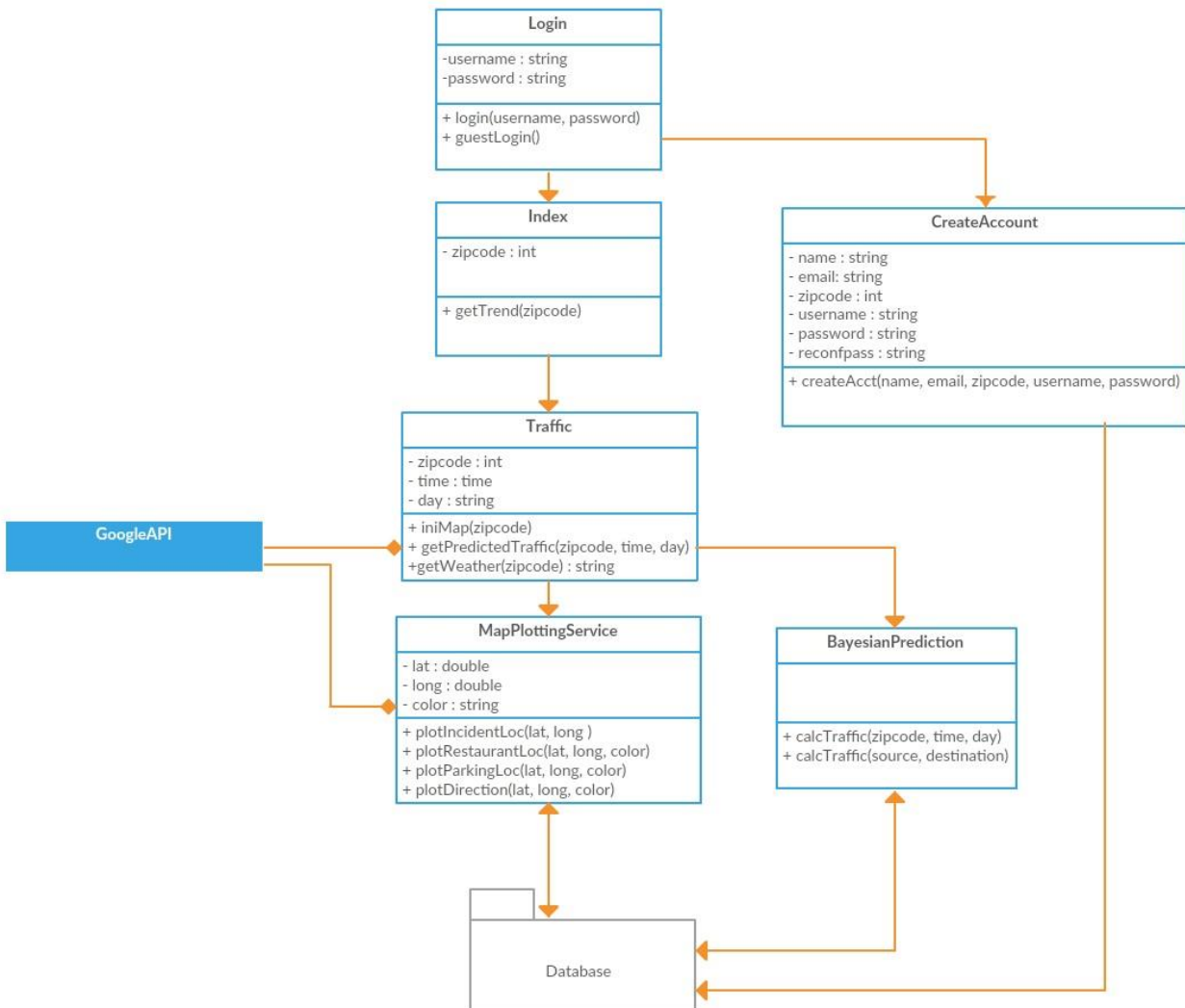
the user credentials now the user input will be stored in the database and the trends for that area will be fetched.

The alternate paths that this Use Case can take are that the user is using the application for the 1st time or the User inputs invalid information or the data for the inputted area is unknown. In the case that the user is using the application for the first time, the application connects to the Database using the ConnectToDatabase and fetches the default zip code (in our case New Brunswick) and displays trends of this default area. In the case that the user inputs invalid information, the Application passes the information to the Database using the ConnectToDatabase and looks for the zip code. If the zip code is invalid, an error message is prompted to the user and the user is asked to re-enter the data. The last alternate path for this use case includes a case where the information for the desired zip code is not available at the moment. Here, the application sends the user data to the Database using the ConnectToDatabase and looks for the zip code. On verifying that there is no stored information for the desired zip code, a message is displayed to the user on the application saying that the requested information is not available.

Class Diagrams and Interface Specification

Class Diagram for Website: part1

The class diagrams for website are divided into two parts according to our high priority use cases for 1) Traffic for particular zip code and 2) Navigation for particular source and destination.



Use case 1:

The Index class is the class that calls other classes to be executed and holds the data from the website.

From Index class, the program will jump to Traffic class. A Google API will initialize dynamic map according to home zipcode of the user.

Further zipcode, time and day will be passed to BayesianPrediction class through getPredictedTraffic(). The BayesianPrediction class has a function called calTraffic() which passed zipcode, time and day to database, receives the information through SQL queries and calculates the predicted traffic accordingly for given zipcode, time and day. The BayesianPrediction will return the predicted information and stored in data in database which will be accessed by MapPlottingService class to plot the predicted value. Weather information for particular zipcode will be accessed from database and displayed on screen by function getWeather().

Use Case 5:

The class MapPlottingService will have functions for plotRestaurantLoc(), plotParkingLoc() plotting the information regarding corresponding services.

These functions will get the data from database and plot it on map using GoogleAPI.

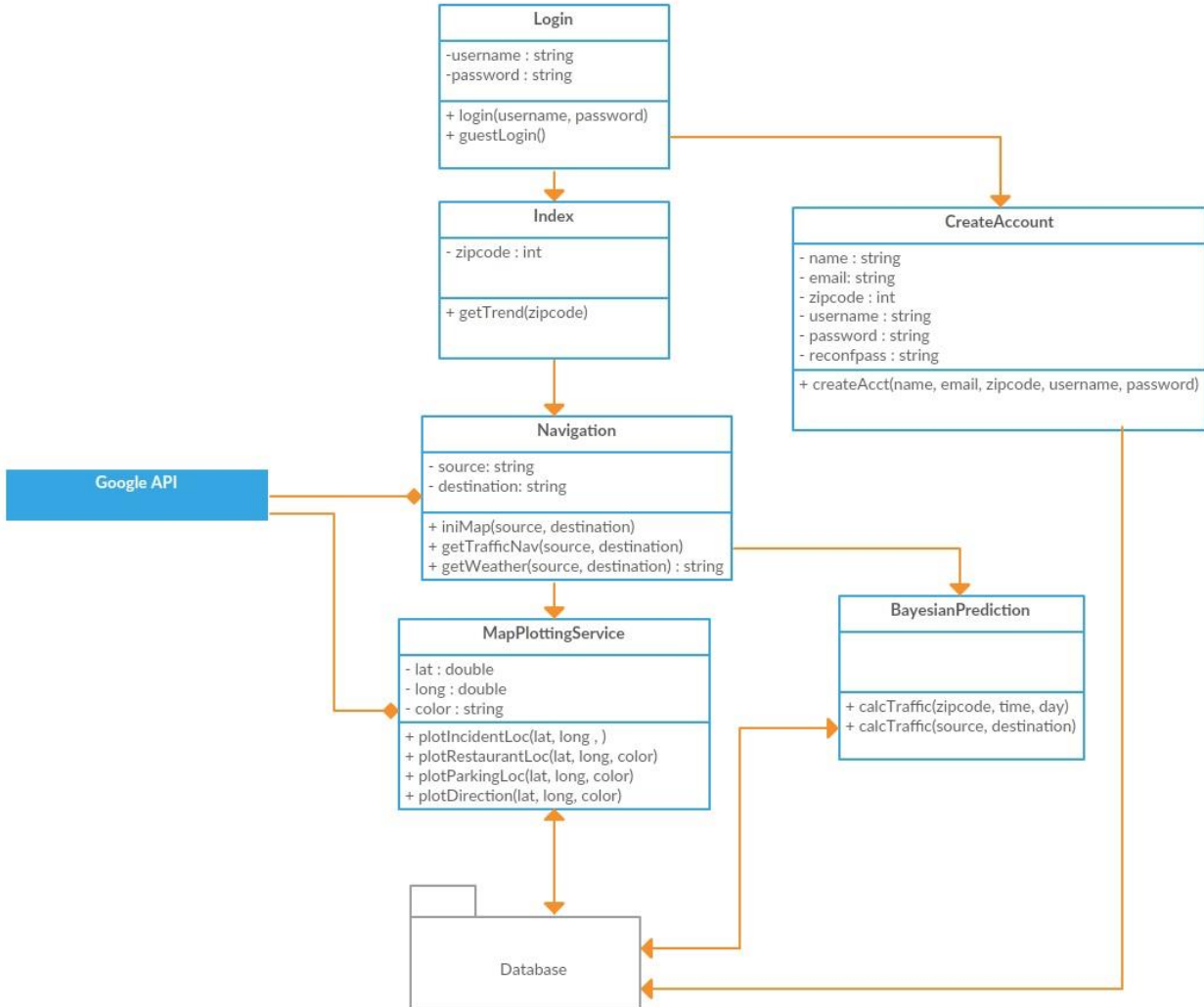
Use case 6:

We have identified one more use case where program begins from Login class and user logs in with username and password. User credentials are checked with the database function isValid(). The flow of program then jumps to the Index class where function named getTrend(), which gets the zipcode from user and display traffic trend for a particular week on home page.

If user not wishes to register or login, there will be a function call guestLogin(), which will take the program control to Index class and show a fixed traffic trends for previously stored zipcode.

If user is not a registered and wishes to register then program will jump to CreateAccount class. In this user entered data such as name, email, zipcode, username and password will be sent to database to create an account. In database, insert() query will update the database. After creating an account program will again go to Index class.

Class Diagram for Website: part2

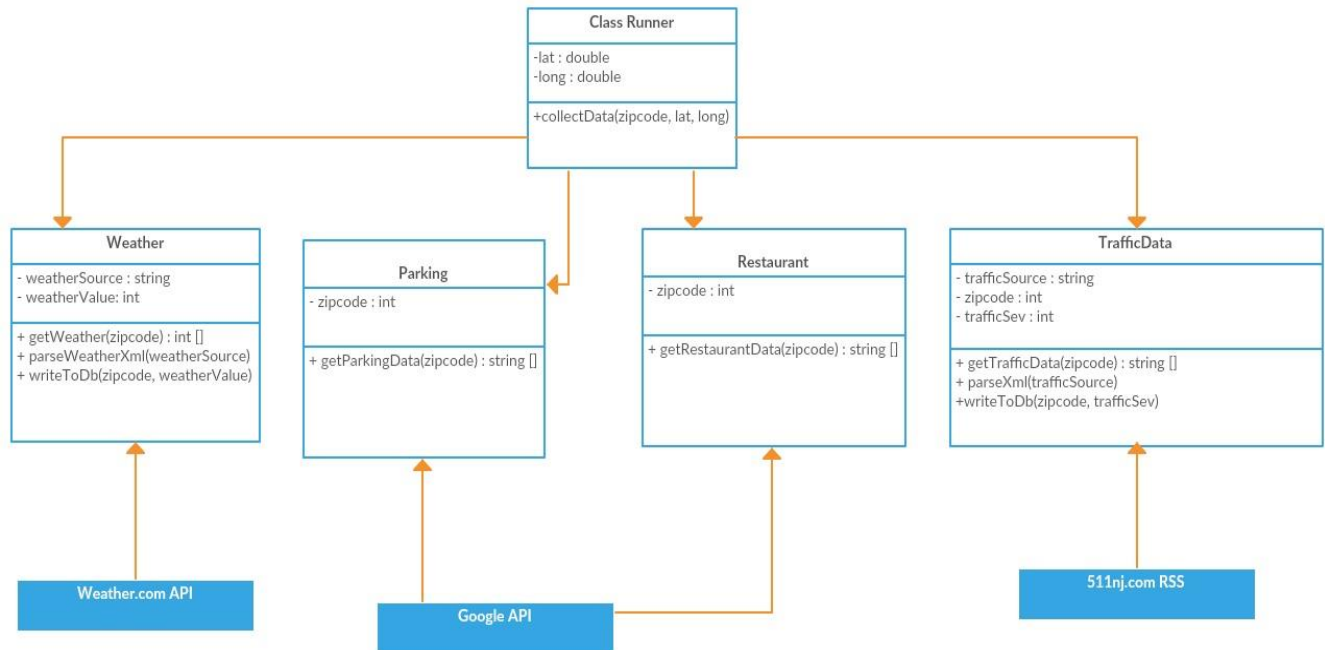


Use case 2:

The class **Navigation** has two attributes called `source` and `destination` with string datatype. A Google API will initialize dynamic map according to the most frequent source and destination of the user.

Further when user enters the source and destination, it will be passed to **BayesianPrediction** class through `getTrafficNav()`. The **BayesianPrediction** class has a function called `calcTraffic()` which passes source and destination information to the database, receives information for particular route through SQL queries and calculates the predicted traffic accordingly for the given source and destination. The **BayesianPrediction** will return the predicted information and stored in data in database which will be accessed by **MapPlottingService** class to plot the predicted value. Weather information will be accessed from database for source as well as destination and displayed on screen by function `getWeather()`.

Class Diagram for Scripts:



Use case 3:

The Class Runner will setup the data collection process with function `collectData()`.

In class **Weather** `getWeather()` will collect the data from **Weather.comAPI**, it will parse the data with `parseWeatherXML()` and write into database with `writeToDb()`.

In class **TrafficData** `getTrafficData()` will collect the traffic data from **511nj.comRSS**, it will parse the data with `parseXML()` and write into database with `writeToDb()`.

Similarly data collection for restaurant and parking is less frequent. Thus class **Parking** and **Restaurant** will have `getParkingData()` to collect the data from **GoogleAPI** and `writetoDb()` to insert the data into database.

Data Types and Operation Signatures:

Login ->

- Attributes:
 - 1) -string username – User input for username
 - 2) -string password – user input for password
- Operations:
 - 1) +login(username, password) – user logs into the application and it brings to class Index
 - 2) +guestlogin() – user logs in as a guest and continues to the class Index.

Index ->

- Attributes:
 - 1) -int zipcode - ser input for zipcode on home screen.
- Operations:
 - 1) +getTrend(zipcode) – user sees the traffic trend for on-going week for entered zipcode.

CreateAccount ->

- Attributes:
 - 1) -string name - user input for name
 - 2) -string email - user input for email
 - 3) -int zipcode – user input for home zipcode
 - 4) -string username – user input for entering username
 - 5) -string password – user input for password
 - 6) -string reconfpass – user input for confirming password
- Operations:
 - 1) +createAcct(name, email, zipcode, username, password) – function to create account.

Traffic ->

- Attributes:
 - 1) -zipcode – user input for zipcode
 - 2) -time – user input for selecting time slot
 - 3) -day – user input for selection day of the week
- Operations:
 - 1) +iniMap(zipcode) – initialize map for previously stored zipcode.
 - 2) +getPredictedTraffic(zipcode, time, day) – get predicted traffic from BayesianPrediction class.
 - 3) +getWeather(zipcode) – get weather data from database and print the data on webpage.

Navigation ->

- Attributes:
 - 1) -string source – user enters particular source location.
 - 2) -string destination – user enters particular destination location.
- Operations:
 - 1) +iniMap(zipcode) – initialize map for previously stored zipcode.
 - 2) +getPredictedTraffic(source, destination) – get predicted traffic from BayesianPrediction class.
 - 3) +getWeather(zipcode) – get weather data from database and print the data on webpage.

MapPlottingService ->

- Attributes:
 - 1) -double lat – store Latitude value from database
 - 2) -double long – store Longitude value from database
 - 3) -string color - color for particular incident
- Operations:
 - 1) +plotIncident(lat, long, color)
 - 2) +plotRestaurantLoc(lat, long, color)
 - 3) +plotParkingLoc(lat, long, color)
 - 4) +plotDirection(lat, long, color)

BayesianPrediction ->

- Operations:
 - 1) +calcTraffic(zipcode, time, day) – predicts the traffic using Bayesian prediction method for particular zipcode, time and day.
 - 2) +calcTraffic(source, destination) - predicts the traffic using Bayesian prediction method for particular source and destination route.

Runner ->

- Attributes:
 - 1) -double lat – initiating latitude value
 - 2) -double long – initiating longitude value
- Operations:
 - 1) +collectData(zipcode, lat, long) – initiates data collections script from weather, traffic, restaurants, parking services.

Restaurant ->

- Attributes:
 - 1) -int zipcode – initiating the zipcode for collecting restaurant data.
- Operations:
 - 1) +getRestaurantData(zipcode) – restaurant data collection script

Parking ->

- Attributes:
 - 1) -int zipcode – initiating the zipcode for collecting parking data.
- Operations:
 - 1) +getParkingData(zipcode) - parking data collection script.

Weather ->

- Attributes:
 - 1) -string weatherSource – storing values of current weather.
 - 2) -int weatherValue – storing parsed value of weather.
- Operations:
 - 1) +getWeather(zipcode) – script for storing weather updates
 - 2) +parseWeatherXML(weatherSource) – parsing the weather data to store into the database
 - 3) +writeToDb(zipcode, weatherValue) –writing zipcode and weatherValue into database.

TrafficData ->

- Attributes:
 - 1) -int zipcode – storing value of zipcode.
 - 2) -string trafficSource - storing value of traffic location/incident
 - 3) -int trafficSev – storing parsed value of traffic severity.
- Operations:
 - 1) +getTrafficData(zipcode) – script for storing traffic updates
 - 2) +parseXML(trafficSource) – parsing the traffic data to store values in database
 - 3) +writeToDb(zipcode, trafficSev) – writing zipcode and trafficSev into database.

Traceability Matrix:

Column1	Use Case 1	Use Case 2	Use Case 3	Use Case 5	Use Case 6
Login					Allows User to log in and guest log in
Index					Display weekly trends for particular zipcode
CreateAccount					Creating user account
Traffic	Receives User Inputs and initiates map plotting service				
Navigation		Receives User Inputs and initiates map plotting service			
MapPlottingService	Creates images of points as per severity of traffic	Creates images of navigation route and traffic severity points on		Creates images of restaurant and parking spots.	
BayesianPrediction	Traffic prediction based on zipcode	Traffic prediction based on source and destination			
Runner			Initiates data collection		
Restaurant			finds restaurants places from google api and stores into database		
Weather			finds weather updates from weather.com and stores into database		
Parking			finds restaurants places from google api and stores into database		
TrafficData			finds traffic updates from 511nj.com and stores into database		

System Architecture and System Design

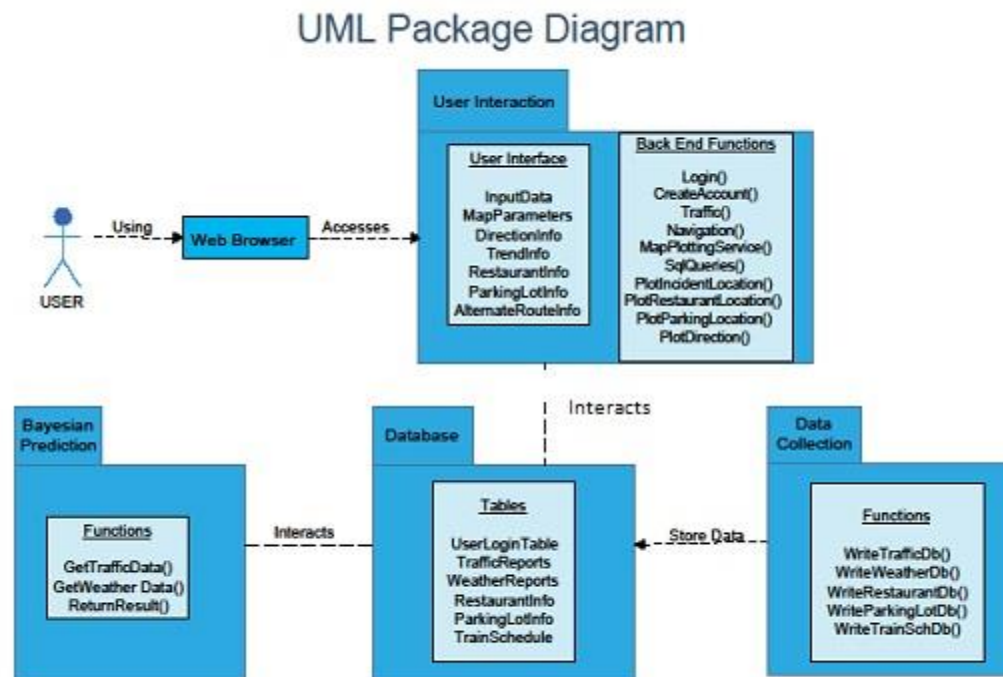
Architectural Types:

The most important actors of the traffic monitoring system are the users who will use the application and the system (which includes the application, database etc.) that'll provide the information after processing the data. Hence, the primary architectural style used in this project is the Client- Server style. The Client-Server model of computing is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients. In our case, the client is going to be the user and the server will be the system.

The clients and servers will communicate over a computer network on separate hardware. Clients and servers exchange messages in a request-response messaging pattern: The client sends a request, and the server returns a response. A server host runs one or more server programs which share their resources with clients. A client does not share any of its resources, but requests a server's content or service function. Clients therefore initiate communication sessions with servers which await incoming requests. The user of the application is only meant to interact with the basic user interfaces of the Web Application. After the user inputs their relevant information, it is the server's job to process all of the information given in order to display what the user needs.

For an easy user experience, it is important to reduce the number of interactions of the user with the many processes or classes that will be present in the application to process the user information. In this project the user will need to control only a series of inputs (text boxes, dropdown menus, radio buttons, etc.) and has to understand the response based on the well-known application protocol, i.e. the content and the formatting of the data for the requested service. These aspects of the project are all very simple to interact with and designed to have minimum user effort. The server sends requests to server side classes based on the input from the user and hence, the user will never interact with these classes. Using these simple inputs from the user, a complex output is displayed for the user, by the server. The classes execute in a similar way each time. The same functions and classes are used every time, so the Client-Server style is a good fit.

Identifying Subsystems:



The traffic monitoring service will be a website that the clients will interact with. These users will access the web Application through the browser of their choice. The network protocol in this case is HTTP with POST request method. This allows us to reach the highest number of users through a web Application.

All user interaction with the system takes place in the User Interaction subsystem. The user interaction subsystem contains all of the user inputs, and the services that have to be displayed by the system. These services include the map output, the directions output, the add-ons like restaurant locations, parking lot locations and train schedules. The user will interact with only the interface and provide his inputs. The other part of the subsystem includes the backend functions that will run during their use of the Application. This subsystem connects directly to the database of the system. The database holds all of the information necessary to achieve the output the user desires. These functions are not accessible by the user.

The second subsystem is the Data Collection subsystem. This system involves the functions to get and write the information got from the web services to the Sql Database. The various service receivers access the relevant web services in order to find more data to store into the database. Because it stores data into the database, this subsystem clearly also connects to the database.

The next subsystem is the Prediction subsystem. This system is responsible for generating results based on the input from the user and the data stored in the database. This system uses the data from the user and the database data to predict the traffic that could be expected in any given area. This subsystem returns the result in the form of the predicted values for the traffic. Hence, this subsystem is connected to the Database and interacts with the Database.

The final subsystem is the database. The database is responsible for storing the data got from all the other subsystems. This database will make tables which will hold information like the user login details, the traffic data, weather data, the restaurant location data, the parking lot location data and the train schedules. The other subsystems will access and interact with this subsystem whenever it requires to access historical data.

Global Control Flow:

1) Execution Orderness-

The system is event driven as users can generate actions in different orders each time they use the system. For example, a user could use request traffic and weather predictions, or they could request navigation. While navigating they could select a restaurant, or a parking lot, or neither.

2) Time Dependency-

The rate that traffic and weather data is updated is on a timer. The rate that the data is updated can be adjusted by an administrator. There are no timers that are related to what a user would be using the system for. Users are able to perform actions at any time.

Hardware Requirements:

The website will be hosted on a remote server. And the data collected is stored in a database, on multiple servers, that is the host server deploying the website and the project members. The database being used is MySQL and the server being used presently is Apache TOMCAT 8.0. We are currently working on making sure the Database containing the traffic data and the website are on the same host.

Algorithms and Data structures

Our major challenge was to construct an algorithm and assigning a severity value to each event data which is present in our database. We are going to use an weighted algorithm to decide the severity of live traffic data. This algorithm is based on Congestion data, Special events data, Construction data and weather data.

In Our congestion data we will look for two main keywords which are "delay" and "Grid lock alert".

High Traffic. Severity = 5

Severity 5 is assigned for heavy traffic. To assign severity 5 we will look into four main factors such as Congestion, Special events, Construction and weather conditions. Below we have explained the priority list.

- Congestion (Grid lock alert) + Special events + construction events
- Congestion (delay) + Special events + severe climate
- Congestion (delay) + severe climate
- Congestion (delay) + Special events
- Congestion (Grid lock alert)

Moderate High. Severity = 4

Severity 4 is assigned for Moderate high traffic. To assign severity 4 we will look into four main factors such as Congestion, Special events, Construction and weather conditions. Below we have explained the priority list.

- Congestion (delay) + special events
- Congestion (delay) + construction events
- Special events + construction events
- Construction events + Severe climate

Moderate. Severity = 3

Severity 3 is assigned for Moderate traffic. To assign severity 3 we will look into four main factors such as Congestion, Special events, Construction and weather conditions. Below we have explained the priority list. Here we assume that there is no Congestion due to Grid Lock.

- Special events
- Construction events
- Congestion (delay)

Moderate Low. Severity = 2

Severity 2 is assigned for Moderate low traffic. To assign severity 2 we will look into four main factors such as Congestion, Special events, Construction and weather conditions. Below we have explained the priority list. For severity 2 we take into consideration that there is no Congestion (delay) or (Grid lock alert) and there are no special events. Here we assume two events are occurring at given time.

- Construction events + Severe climate

Low. Severity = 1

Severity 1 is assigned for low traffic. To assign severity 1 we will look into two main factors such as Construction and weather conditions. Below we have explained the priority list. For severity 1 we take into consideration that there is no Congestion (delay) or (Grid lock alert) and there are no special events. Here we consider that only one event is occurring at a given time.

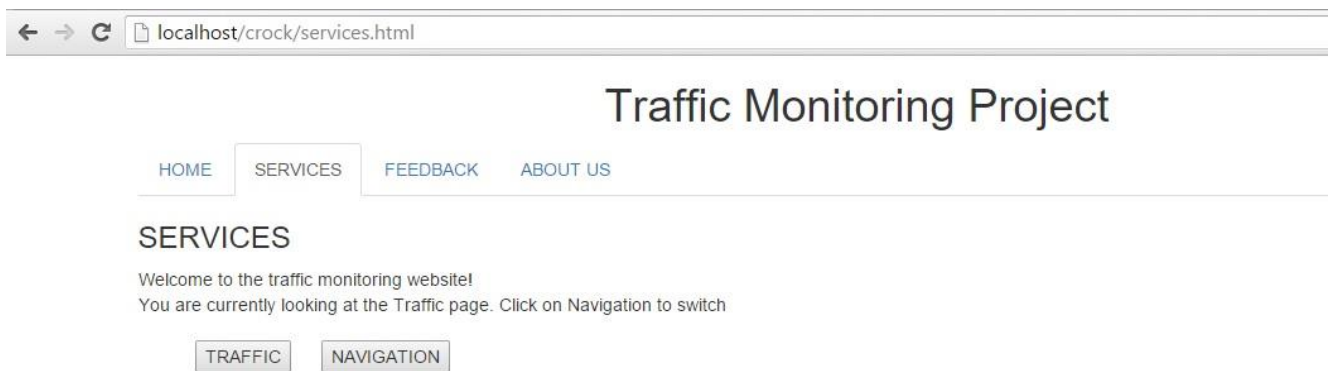
- Construction events
- Severe climate

No complex data structures are used in this project.

User Interface Design and Implementation

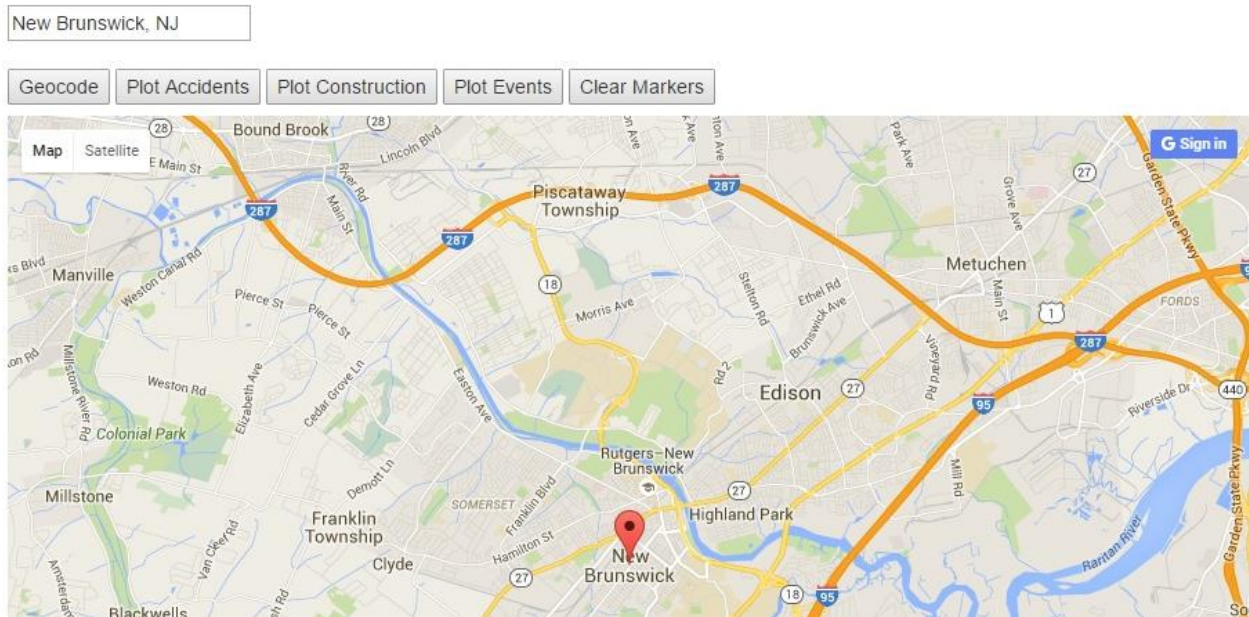
Since the main objective of the project is to display results on a web page which can be easily understood by the user, the interface implementation plays a key role in achieving this goal. We have tried to create an interface which is easy to use and understand and will improve on it as and when we figure out a better way to implement its functionality.

In the preliminary design of the User interface, the approach adopted by us was to provide drop down menu's to select from among the various services that could be accessed by the user. However, while implementing the Interface, we realized that since the layout of the major paths of our services (Traffic and Navigation) are identical to a great extent. Hence, we decided to make buttons to toggle between the services on each page. This reduces the user effort by at least 1 mouse click each time he/she wants to switch between services. With the current implementation of our interface, the switching between services with the help of buttons can be seen below:

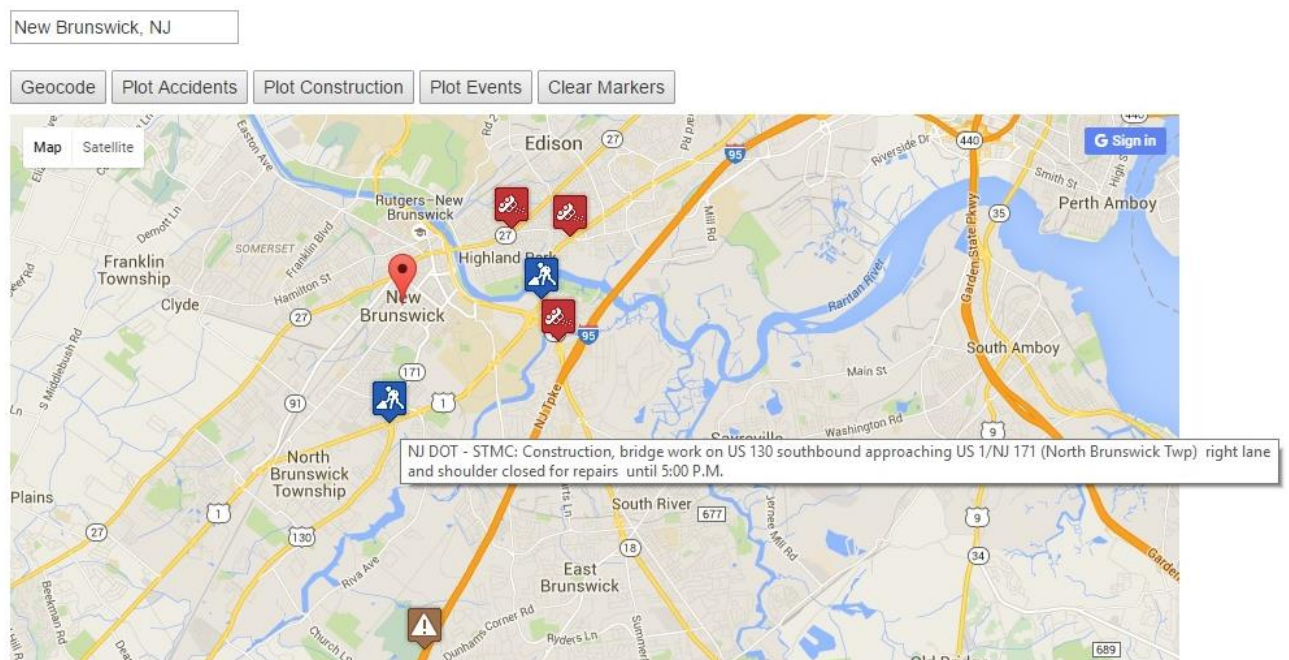


When the user chooses the respective service that he/she wishes to access, there will be text boxes which will be filled by the user to get the result which is desired.

In the preliminary description of the interface, the design being used was based on zip code. However, while implementing the project, we realized that it was possible to achieve the functionality of geocoding the location even when the user inputs just the name of the location. This reduces the effort of the user in a way that the user doesn't have to remember the zip code of the desired location and can search just on the basis of the name of destination. The following screenshot shows how the user can input the name of the location instead of the zip code and still locate the place on the map.



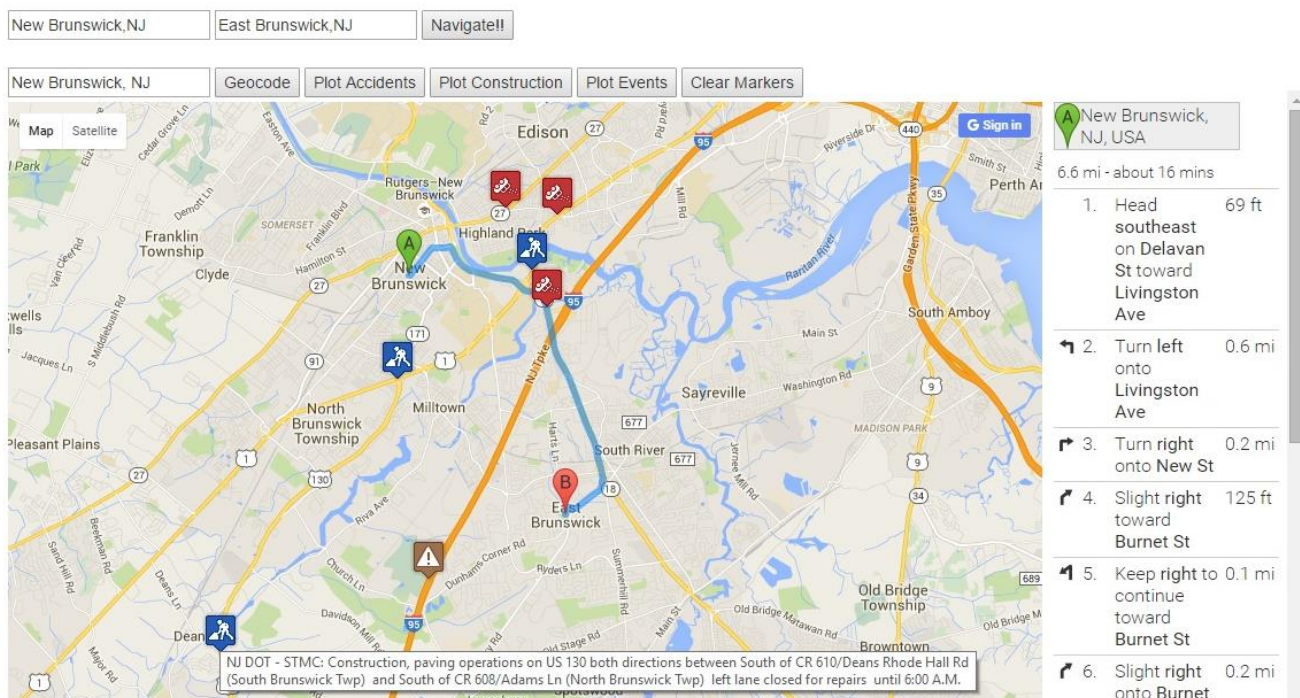
While collecting data we realized that additional functionality could be given to the application by helping the user see live traffic incidents that are occurring in the areas of interest. This is achieved by providing buttons on the interface that will place markers on the locations where there are traffic related incidents like construction work, accidents and special events. These buttons are places right above the displayed map to make it easy for the user to access these services while looking at the map. The following image displays the buttons and how it appears on the interface when the user chooses to see the incidents:





As seen from the above image, the markers provide a detailed description as to what incident has occurred at the area. The user simply needs to take the mouse pointer over one of the markers and it'll prompt the description immediately.

On the Navigation page, the basic functionality and layout of the page is similar to the Traffic page. This helps in making the interface easy to understand for the user. The major difference between the traffic page and the navigation page is that for navigation we require the user to input a source and destination location. Just like the traffic page, here also, the user can input the name of the source and destination rather than zip codes. The navigated route is highlighted in blue which is different from the regular color of the roads and so it can be easily recognized. The traffic incidents are still available with the same functionality as the traffic page for the user to see incidents along the suggested route. A scroll bar which is the size of the map is added with directions from the source to destination alongside the map so that the user can look through all the directions while looking at the route. The scroll bar for the directions reduces user effort and increases ease of using the application because the user doesn't have to scroll up and down on the web page to look at the directions and try to locate it on the map. The following image shows the navigation functionality that is currently implemented by us.



The current implementation of the project provides live traffic incidents and navigation functionality which has been mapped into the user interface implementation. In the coming days we will be implementing the additional functionalities like Parking Lot info and Restaurant Info which will be incorporated into the website keeping the user experience and effort in mind.

Design of Tests

In this phase we have designed our website to open a particular location when loaded. We have chosen New Brunswick since it is the place where the server is located.

Test case 1, Navigation: In the navigation.html user can enter a desired destination address and the source or starting point. Then the page gives directions to the particular location. Then the User can use the section below the map which shows "Accidents", "Construction" and, "Special events". To display where there is congestion on road. This is quite useful when the user wants to go out and check if there is congestion on any route to avoid it and take an alternate route.

Test case 2, Current traffic: User enters a particular location. It may be their home or any other location, then the map is centered to the entered location. The below the map section many options can be seen- Events, Construction, Accidents. The user clicks on the desired section which they want to see, either one or all of them, which can be cleared at any time

Integration Test: An android application will be integrated to the website. A user in possession of an android device can dump the .apk on the android device and run it to perform the same functions which the web application does. Mobile app is being made, Since it is convenient to carry a mobile device. And can be used while traveling.

The mobile application can also be tested on eclipse, with an android sdk.

The further Test cases before the final demo:

- Giving an alternate route to the user which has no congestion. In case there is a major delay along the path which is projected.
- Providing prediction of traffic for the whole week, just like the weather predictions we see today. User enters a Zip code and a webpage showing traffic trends for the next 7 days will be shown.
- Plotting place along the route, that is we will be plotting the places nearby like restaurants, parking lots along the path being navigated. For the user going on long drives or searching for a parking lot near the destination.

Project Management and Plan of Work:

A. Merging the Contributions from Individual Team Members:

When you have six people working on a single project they have six different approaches regarding documentation, coding. There must be one single person who collects all the data from 6 different people and present it like a whole unit.

Shreyas Bhandare and Kisholoy Vinayak Ghosh were responsible for merging the entire member's contribution in terms of reports and code and present it as a whole group.

There are several issues we faced. When we divided the work, all of them submitted the reports in different font size, font type, spacing and different numberings. Thus compiling full report-1 became a nightmare. We tackled this issue by laying certain ground rules for documentation. Since then we have followed Calibri font, size 11 for normal text, size 12 for highlighted points and size 14 for Main point. We resolved the spacing issue using 'Justify' tool in MS-Word everywhere (with few exceptions).

We faced the same issue in the coding part. Earlier few wrote comments for the code and some didn't. While preparing demo-1 documents, we had to look every code and add comments if there aren't any. Thus for the demo and electronic archival everyone has decided to comment his code by own to save time and confusion.

B. Project Coordination and Progress Report:

Current working use cases:

Use case-1: Displaying current traffic is implemented. Our aim was to show traffic intensities for particular area. But since our source is not providing us consistent and uniform information, we are currently able to show incidents such as accidents, construction and special events.

Use case-2: Navigating from source to destination is been implemented by showing current incidents, accidents and construction on navigated route.

Use case-3: Traffic data collection scripts are running on server machine.

Project Management Activities:

We meet once in a week at SERC reading room to monitor our project progress. We analyze current work, discuss the issues with everyone, correct the flaws, plan the future work and assign it to respective subgroup. Also Project coordination plays important role as we divide our work in subgroups and then coordinate with every other subgroup to make project work. Usually every project in industry is assigned a project coordinator who

plans, monitors and controls all aspects of your project. In our project, we divided our work in three subgroups:

- Kush Patel and Tejas Ravi (Kush is a subgroup coordinator)
- Shreyas Bhandare and Kisholoy Vinayak Ghosh (Shreyas is a subgroup coordinator)
- Jeffrey Gillen and Aaditya Shukla (Jeffrey is a subgroup coordinator)

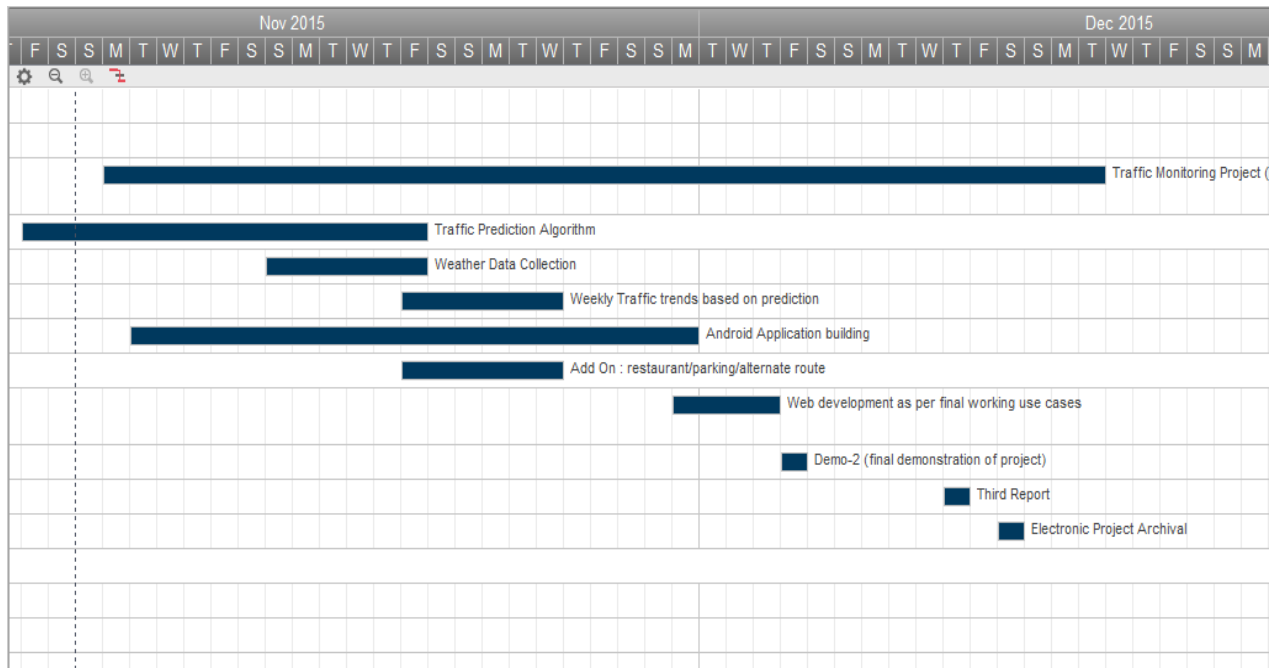
Every person from each subgroup acts as a local level coordinator while Kisholoy Vinayak Ghosh/Tejas Ravi is on top of the hierarchy who acts a whole project coordinator.

Currently we are working on traffic prediction model from historical traffic data. This model is based on Bayesian prediction strategy. Details of current work and future work for rest of the project duration are shown below in section C Gantt chart.

C. Plan of Work:

Task Name	Start Date	End Date	Assigned To	Duration
Traffic Monitoring Project (work remaining after demo-1)	11/06/15	12/12/15		37d
Traffic Prediction Algorithm	11/06/15	11/20/15	Tejas, Shreyas, K.Vinayak, Aaditya	15d
Weather Data Collection	11/15/15	11/20/15	Kush, jeffrey	6d
Weekly Traffic trends based on prediction	11/20/15	11/25/15	Shreyas, K.Vinayak	6d
Android Application building	11/10/15	11/30/15	Aaditya, Tejas	21d
Add On : restaurant/parking/alternate route	11/20/15	11/25/15	Kush, Jeffrey	6d
Web development as per final working use cases	11/30/15	12/03/15	Shreyas, K.Vinayak	4d
Demo-2 (final demonstration of project)	12/04/15	12/04/15	Everyone	1d
Third Report	12/10/15	12/10/15	Everyone	1d
Electronic Project Archival	12/12/15	12/12/15	Everyone	1d

Gantt Chart Continued Below..



D. Breakdown of Responsibilities

Now onwards, according to our skills, we are re-organizing our teams a little bit to achieve better performance.

- Aaditya and Tejas will be working on mobile android application. Testing for this module also will be performed by them.
- Shreyas, Kisholoy, Tejas and Aaditya will be working on Bayesian prediction strategy for traffic prediction and weekly traffic trends display. Testing for this module also will be performed by them. This work is currently being progressed.
- Kush and Jeffrey will be working on parking lot display, restaurant display, and alternate route. Testing for this module also will be performed by them.
- Integration testing will be performed by Kush Patel and Jeffrey Gillen.

References

- [1] Chen, H., Roseman, R. and Rule, A. (2015). *Curve Fitting & Multisensory Integration*.
[online] <http://www.cogsci.ucsd.edu/>. Available at:
http://www.cogsci.ucsd.edu/~ajyu/Teaching/Cogs202_sp14/Slides/lect3.pdf [Accessed 15 Oct. 2015].
- [2] Creately.com, (2015). *Online Diagram Software to draw Flowcharts, UML & more / Creately*. [online] Available at: <http://creately.com> [Accessed 15 Oct. 2015].
- [3] Dev.virtualearth.net, (n.d.). [online] Available at:
http://dev.virtualearth.net/REST/v1/Locations?postalCode=07657&key=AvITzUaX5nqGg0JKU43QSP30BB_ovX37MIND9WahtRbBQpeV4ugoB3HQ60LRxQL8&o=xml
[Accessed Oct. 2015].
- [4] GenMyModel, (2015). *Software Modeling in the Cloud*. [online] Available at:
<http://genmymodel.com> [Accessed 16 Oct. 2015].
- [5] <https://developers.google.com/maps/?hl=en>, (n.d.). *Google Maps API*. [online] Available at:
<http://google.com> [Accessed Oct. 2015].
- [6] Marsic, I. (2012). *Software Engineering*.
- [7] Msdn.microsoft.com, (n.d.). *Find a Location by Address*. [online] Available at:
<https://msdn.microsoft.com/en-us/library/ff701714.aspx> [Accessed Oct. 2015].
- [8] Msdn.microsoft.com, (n.d.). *Get Traffic Incidents*. [online] Available at:
<https://msdn.microsoft.com/en-us/library/hh441726.aspx> [Accessed Oct. 2015].
- [9] Msdn.microsoft.com, (n.d.). *Traffic Incident Data*. [online] Available at:
<https://msdn.microsoft.com/en-us/library/hh441730.aspx> [Accessed Oct. 2015].
- [10] Openweathermap.org, (n.d.). *OpenWeatherMap current weather and forecast*. [online]
Available at: <http://openweathermap.org> [Accessed 4 Oct. 2015].
- [11] Project Report Spring 2011, (2011). *Project: Traffic Monitoring*. [online] Available at:
<http://www.ece.rutgers.edu/~marsic/books/SE/projects/Traffic/2011-g7-report3.pdf>

[Accessed 2015].

[12] Project Report Spring 2013, (2013). *Traffic Monitoring Service*. [online] Available at:

<http://www.ece.rutgers.edu/~marsic/books/SE/projects/Traffic/2013-g7-report3.pdf>

[Accessed 2015].

[13] Smartsheet, (n.d.). *Online Project Management Software / Smartsheet*. [online] Available at:

<http://smartsheet.com> [Accessed 15 Oct. 2015].

[14] Stackoverflow.com, (n.d.). *Stack Overflow*. [online] Available at:

<http://stackoverflow.com/questions/33069298/bing-traffic-api-not-returning-any-data-for-accidents-delays> [Accessed 2 Oct. 2015].

[15] W3schools.com, (2015). *W3Schools Online Web Tutorials*. [online] Available at:

<http://w3schools.com> [Accessed 10 Oct. 2015].

[16] Websequencediagrams.com, (n.d.). *WebSequenceDiagrams.com - Draw and Edit Sequence*

Diagrams in seconds. [online] Available at: <http://websequencediagrams.com> [Accessed 8 Oct. 2015].

[17] Wikipedia, (2015). *Curve fitting*. [online] Available at:

https://en.wikipedia.org/wiki/Curve_fitting [Accessed 15 Oct. 2015].

[18] Youtube.com, (n.d.). *YouTube*. [online] Available at: <http://youtube.com> [Accessed 7 Oct.

2015].