Software Engineering 332:452

Traffic Monitoring Project
Group 13 – May 5, 2015

http://www.justincoding.com/trafficProject

Maxine Dienes

Charu Jain

Mehul Salhotra

Akshay Sardana

Justin Silang

Jason Yang

Responsibility Matrix

Responsibilities	Maxine	Charu	Mehul	Akshay	Justin	Jason
Summary of Changes (5)	3	1	0.25	0.25	0.25	0.25
Customer Statement of Requirements (6)	1	1	1	1	1	1
Glossary of Terms (4)	1	1	1	1		
System Requirements (6)	1	1	1	1	1	1
Functional Requirements Specification (30)	5	5	5	5	5	5
Effort Estimation (4)		1	1	1	1	
Domain Analysis (25)		5	5	5	5	5
Interaction Diagrams (40)	4	5	5	8	9	9
Class Diagram and Interface Specification (20)	3	4	2	5	3	3
System Architecture and System Design (15)	3	2	5		2	3

Algorithms and Data Structures (4)		1		1	1	1
User Interface Design and Implementation (11)	4	3				4
Design of Tests (12)	.25	.50	5	2.25	4	
History, Current, Future of Work (5)	1	1	1		1	1
Project Management (13)	7	2	1	3		
Total 200	33.25	33.50	33.25	33.50	33.25	33.25

Table of Contents

Summary of changes

- **We removed social media and music because we decided not to focus on non-functional topics too much and rather, to focus on functional requirements**

- **We edited the use cases by adding a GetUserReport and GetPolice Report. We did this so that there would a distinct separation between receiving the data for these since on the website there are two separate buttons for receiving these bits of information.**

  We changed User Interface by adding voice activation to make it easy to use for users. This was one of our biggest advances for the Mobile Application because we do not want users to be engaged on their phones while driving.

- **We added several buttons for the users to choose the specific action they want the application to do. This makes the Application much easier to use and clearer to understand.**

- **We implemented the GetDirections function so that our Application would have more beneficial use for the Users.**

- **We incorporated a Fastest Route algorithm that analyzed all traffic patterns based off of live and historical data to return an optimal route for users. This increased the efficiency and accuracy of our Application.**

- **We added algorithms in the database to set a threshold of user report entries so that the entries are accurate. For example, if the threshold is two, then there must be two or more user reports of the same time** and **at the same location for it to be shown on the map. This increased the accuracy of our Application significantly by keeping misinformation and accidentally entered information away from our display.**

- **We set up a cleaner script in the Database to delete old entries so that the entries are kept new and historically valid at the same time.**

Customer Statement of Requirements

Problem Statement

The Problem

If there is one daily aspect of the modern lifestyle that people all around the world express annoyances over, it is undoubtedly being stuck in traffic. Perhaps you are a university student running late to your class, and every traffic light seems to change to red before you. Or maybe an emergency forces you to drive to the hospital, but every red light delays your treatment minute by minute. Whatever the case may be, everyone who has ridden a motor vehicle has experienced this frustrating and tedious phenomenon before. Traffic congestion poses problems not only in wasted time but also monetary costs in wasted fuel when your car is idle, to the tune of upwards of $755 annually per driver (Morgan). If traffic is such a wasteful and universal problem, shouldn't there be a way to reduce the frequency of it, or at least avoid it yourself?

The Solution

Traffic congestion will always be inevitable, but there can be methods to reduce its severity. Currently, services exist for individuals to avoid areas of heavy traffic. Popular map services such as Google Maps and Mapquest Traffic give information about traffic only based on current accidents, construction, road closures, and other incidents. Most of these services collect data in intervals and release them to show current traffic congestion. While it is essential to provide current traffic information for users, it often becomes unreliable because these services frequently fail to provide up-to-date and accurate traffic information. Instead, it is often more helpful to incorporate an archive of historical traffic conditions for where the user requests.

Thus, the reliability of historical traffic data becomes the fulcrum of our proposed traffic application. Our project will stray away from only using live traffic reports to give information about traffic congestion. It will use this traffic data to utilize past

traffic patterns in a particular area. The purpose of obtaining past traffic data is to enable us to locate areas that will most likely form traffic again and anticipate alternative routes early on, to create a faster and more efficient system. The traffic information can be further used to collect weather along routes and day of week for each data. We aim to give a complete presentation of recent and past traffic data to determine traffic behavior.

Main Specifications

There are three main factors that influence the probability of a driver encountering traffic along a certain route, and consequently will be incorporated into our application:

1. **Historic Traffic Patterns**

   As previously mentioned, a route which historically experiences heavy congestion may exhibit patterns of similar traffic activity. Our service will take this into consideration, past accident reports on a route which should increase the probability of traffic within its respective incident locations. These reports will be retrieved from traffic information sites freely available on the Internet.

2. **Inclement Weather Conditions**

   During periods of inclement weather, a driver's behavior changes in response. He or she will likely slow down because of the reduced vision, which inevitably leads to higher congestion. Our service will look mainly into three weather effects: fog, heavy rain, and snow, all of which increase the probability of traffic in affected areas. We can easily access and retrieve relevant weather data from online weather services.

3. **Day of Week**

   The day of week will heavily affect how many cars will likely be on the road. We will adjust the levels of predicted traffic congestion depending on whether it is a weekday or weekend, which becomes especially relevant during busy hours, like rush hour during the week.

Utilizing all these factors together, we can then obtain a clear representation of how traffic should act in a given area.

Other Features

After the core functionality of displaying traffic info based on past traffic, weather, and day of week is fully implemented, we will look at several more desirable features, some of which are commonplace competitor products, but others which are unique to ours. Our overall goal of the finished product is to integrate many different functionalities of already existing services and combine them into one highly useful application. Here are our additions:

- Users of our service may report areas where there have been police monitoring and activity. This information will then be passed to other users, notifying them so they can either avoid highlighted areas or slow down to prevent getting ticketed by the authorities. This will promote wary and safe driving.

- Users of our service may also report areas where there is anything that may affect traffic – such as potholes, accidents, etc. This information will then be passed to other users, notifying them so they can avoid highlighted areas.

- A sleek GUI to truly provide a user-friendly experience. The interface should be intuitive to the user as we are providing a service.

- When getting directions on our application, the user can select a fastest route option that integrates our traffic algorithm to calculate the most optimal route for the conditions at the time.

- Voice Activation for improved usability, allowing consumers to not get distracted while driving and thus reduce the amount of casualties from cellular distraction.

- The addition of scenic route availability, this will lessen the amount of frustration received from “rush hour” traffic. Most consumers respond to visual stimulus, thus we want to increase consumer satisfaction by introducing the ability for users to rate the scenic value of a specific route,

eventually giving users the option to choose either the fastest route or the scenic route.

Example Outline of User Activities

The desired traffic services will view traffic predictions within the state. First, the user selects his region (north, central, south) or a specific road to view traffic information in his vicinity. This process is outlined below:

[Mobile]

"Traffic Summary within Region" [The user is presented additional choices]

1. "Select Desired Region"
    - user selects corresponding button
2. "Day Selection"
    - user selects "weekend" or "weekday"
3. "Weather Conditions"
    - user selects which weather was present during their target traffic location
4. "Display Live Traffic"
    - user checks display live traffic if desired
5. "Voice Activation"
    - user toggles the voice activation feature

[Web is same as Mobile but without Voice Activation]

The user will be able to look up a traffic summary based on their selection of region, day of week, and weather conditions. These basic selections that users will find intuitive and easy to understand will help provide an accurate match to current traffic conditions when analyzing past data.

Another way of viewing traffic predictions will be along a route. The user will input a starting and destination location. The service will consult with Google Maps to

find the fastest route disregarding traffic information and display expected traffic along that route. The service will now suggest routes that avoid traffic congested areas to the user. The process shown below:

[Mobile]
“Traffic Summary using Main Road” [The user is presented additional choices]

1. Select “a road”
    - user selects corresponding option from dropdown
2. “Day Selection”
    - user selects “weekend” or “weekday”
3. “Weather Conditions”
    - user selects which weather was present during their target traffic location
4. “Display Live Traffic”
    - user checks display live traffic if desired
5. “Voice Activation”
    - user toggles the voice activation feature

“Fastest Route” [The user is presented a minimal traffic route]

1. “Start”, “End”
    - The user will input these two boxes then the service will present a route between these two locations via GUI
2. “Voice Activation”
    - user toggles the voice activation feature
3. “Shortest Travel Time”
    - user clicks button to view different routes
4. “Suggested Routes”
    - user selects their preferred route

Mobile Application

After the web application is fully functional, we are looking to port our product onto cellular devices, namely smartphones. This project will eventually be extended to

mobile applications, where the user can access the traffic monitoring application on the road. It should retain all the core functions of the original web application, but still be simplistic and intuitive enough for people who are driving or on the go. This will increase the overall portability of the product, as well as providing users with a way to access traffic data in a situation where he or she most likely finds it the most relevant: while actually driving.

The mobile app should not trim away any of the main functions of the original application. Such features include presenting traffic data "within a region" or "along a route", inclusion of day of week, weather, police report data, user reported data, and voice activation features. Furthermore, all of the original methods in calculating the traffic congestion will remain the same, along with the methods of taking a user's inputted data and display relevant traffic information based on data taken from traffic and weather services. The only changes from the web application will be in the form of the GUI, which will be optimized for a smartphone experience. The map interface will take up more than half of the screen.

One major consideration of a smartphone app will be in its ease and safety of use. Because users will most likely be multitasking between driving and navigating the app, it is crucial the app remains simple and intuitive, with the fewest number of key-presses as possible. Also, voice activation may greatly reduce the distractions upon a user, and can potentially mitigate safety risks while driving

Glossary of Terms

- *Administrator* - An entity which has privileges to customize, alter and update all facets of the traffic monitoring system.
- *Algorithm* - Procedure or formula for solving a problem.
- *Application* – The main program that the user will be interacting with. This program is made to be user friendly and is located on the website.
- *Current Traffic*– Traffic that is currently stored on the Traffic Service websites at the time the user is utilizing the traffic monitoring service. This data is concrete, and describes the known conditions of traffic at the moment.
- *Database* - A storage device that can be used to store the weather data and traffic data being collected.
- *Developer* - Someone who is involved with creating the website's front-end and its back-end
- *Google Maps*- A service that contains maps. The application will display traffic information to the user using this service.
- *Google Traffic* - A service that contains traffic data that will be collected, parsed, and stored in a database.
- *Graphical User Interface (GUI)* - A type of interface that allows users to interact with the website using graphical components (buttons, checkboxes, etc.) and images instead of text-based commands.
- *Input* - Information that is entered by the user into the application in order to specify the output desired.
- *Mobile Application* - Software for an Android mobile device that is available to all users of the traffic system. Mimics the basic function of the web application.

- *Mobile Device* - A device that is meant primarily for mobile use, including tablets and smartphones

- *Module* - Any of a number of distinct but interrelated units from which a program may be built up or into which a complex activity may be analyzed.

- *Parsed* - Analyze (a string or text) into logical syntactic components, typically in order to test conformability to a logical grammar.

- *User* – A person who intends to access historical traffic data by using the traffic

System Requirements

Enumerated Functional Requirements

Identifier	PW	Requirement
REQ1	4	The system interface shall allow the user to specify traffic summary information by region, day of week, weather conditions, and live conditions.
REQ2	5	The system shall pull traffic data from inner Google Traffic API.
REQ3	5	The system shall analyze the traffic data.
REQ4	5	The system interface shall display the traffic summary.
REQ5	4	The system's interface shall include text boxes in which the user can input their starting location and destination for directions.
REQ6	5	The system shall pull weather data for every few hours.
REQ7	3	The system shall allow the user to zoom in/out of a map to different areas.
REQ8	2	The system shall allow user non-essential features i.e. voice activation, police watch, user reporting
REQ9	3	The system shall extend services to mobile devices.

Table 1: Enumerated Functional Requirements for a traffic monitoring system

Enumerated Nonfunctional Requirements

Identifier	PW	Requirement
REQ10	3	The system shall have an intuitive and user friendly web page.
REQ11	4	The system shall display Google Map's suggested route as well as the suggested route based on traffic history.
REQ12	2	The system shall have a mobile version of this system.
REQ13	5	The system shall use an algorithm which determines the status of traffic.
REQ14	2	The system shall have preferences for route such as fastest route.
REQ15	4	The system shall allow the administrator to control the frequency and time of data gathering scripts.
REQ16	3	The mobile application shall provide analyzed traffic information depending on the user's current location.

Table 2: Enumerated Nonfunctional Requirements for a traffic monitoring system

FURPS Table

Functionality	❖ Features Web and Mobile based Applications for ease of use for the Users. ❖ Features inputs such as Region, Weather, and Day of Week in order for the User to get an accurate depiction of traffic history they wish to see. ❖ Features a police report system and user report system. ❖ Features hands free usage with voice activation.
Usability	❖ The home page of the Website Application has an intuitive menu that provides options for all of its features. ❖ All of the pages on the site are created with a similar interface, allowing the user to navigate easily. ❖ Input boxes are labeled and easy to see.
Reliability	❖ Evaluations are done to check the validity of the data the Users entered onto the Website Application as well as the Mobile Application. ❖ Many versions and configurations will be implemented in the Application to promise that no information will be lost (Version Control).
Performance	❖ Multiple Users can access the Application at the same time. ❖ Efficiency evaluations are done when designing the maps in order check that unnecessary information is not processed.
Supportability	❖ Website Application is compatible with many browsers. ❖ Mobile Application performs on all updated Android devices. ❖ Classes are documented and separated clearly in order to develop an easy and secure expansion with the Application.

Table 3: FURPS requirements for a traffic monitoring system

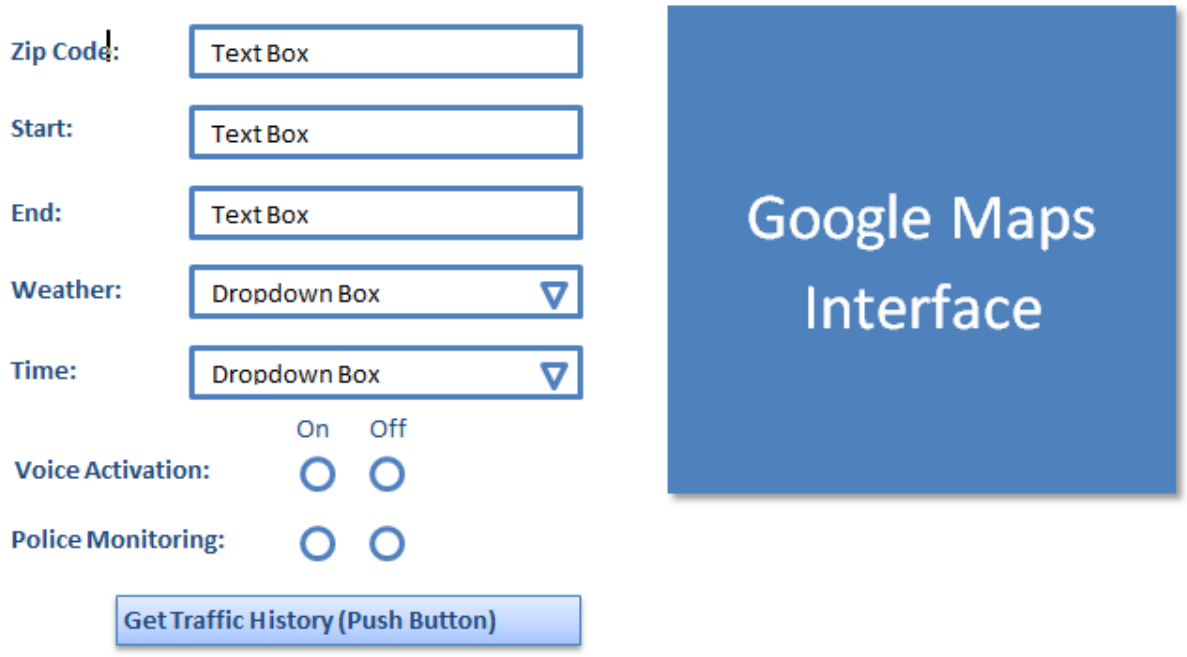
On-Screen Appearance Requirements

Identifier	PW	Requirement
REQ17	5	The system shall contain a Google Maps interface in its display.
REQ18	4	The system shall have a simple and concise mobile app GUI similar to the website version. (Traffic Monitoring System)

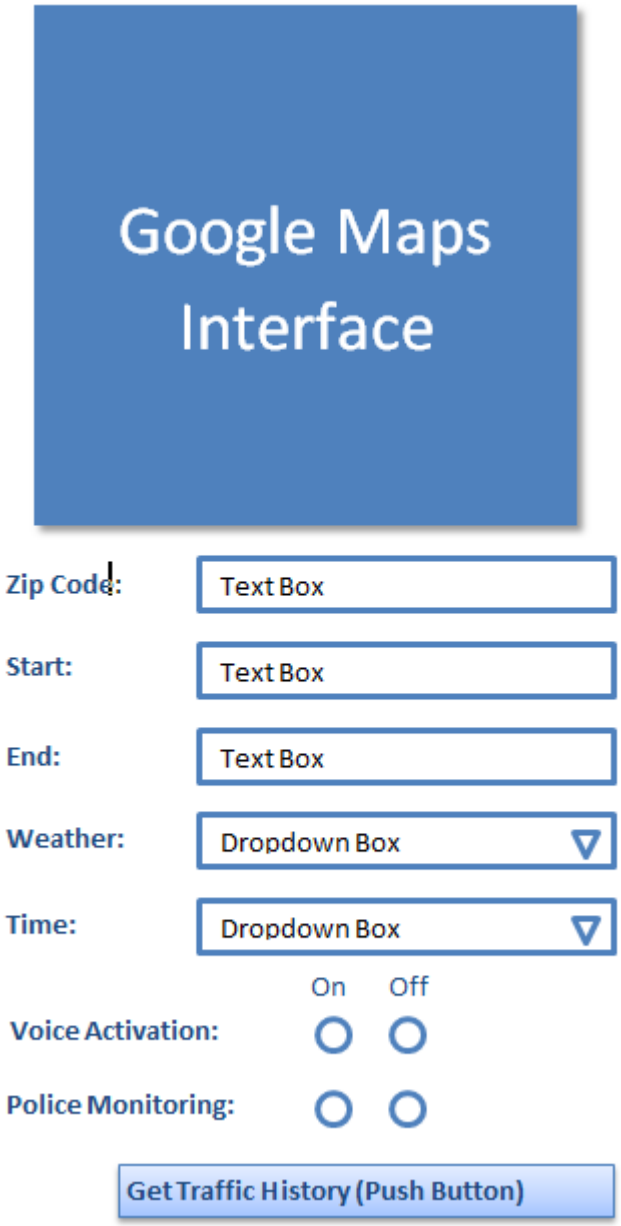
Table 4: On-screen Appearance Requirements for a traffic monitoring system

On-Screen Appearance Illustrations

❖ REQ17



❖ REQ18



These illustrations were polished in our actual design. These are only a rough skeleton that we used to base our design off of. We kept these in our report to provide our initial design visuals. Please look at User Interface and Design section of report to get a more detailed view of our design.

Functional Requirements Specification

- Computer Users
- Administrator
- Mobile Users

Actors & Goals

- *User*
  - Initiating Actor
  - The user accesses the main application to receive traffic history reports and directions of their choosing

- *Administrator*
  - Initiating Actor
  - The administrator's goal is to run and maintain the traffic and weather collection services

- *Application*
  - Participating Actor
  - The goal of the Application is to provide the user accurate results of their traffic and directions queries in the form of text or images. The Application platform supports many web browsers and will run on a website.

- *Database*
  - Participating Actor
  - The goal of the Database is to book-keep all the data to be used. It will store the weather and traffic data sent by the Weather and Traffic Services. Police report data and User report data will be received through forms on our website. It will also store the parsed traffic data that is used to display the traffic history. The database is scalable.

- *Mapping Service*
  - Participating Actor
  - The goal of the Mapping Service is to plot traffic history data on a map that will be intuitive for the user to understand.

- *Geocoding Service*
  - Participating Actor
  - The goal of the Geocoding Service is to obtain the longitude and latitude coordinates of the address.

- *Directions Service*
  - Participating Actor
  - The goal of the Directions Service is to get a route from starting to end destination locations. A further goal is to implement the route based on traffic history and real time traffic.

- *Police Report Service*
  - Participating Actor
  - The goal of the Police Report Service is to store user submitted police reports into the database. It parses the reports by region and day of week.

- *User Report Service*
  - Participating Actor
  - The goal of the User Report Service is to store user submitted user reports into the database. It parses the reports by region and day of week.

- *Weather Service*
  - Participating Actor
  - The goal of the Weather Service is to provide weather data relating to the roads that the Application finds in the route. This data will be stored in the Database.

- *Traffic Service*
  - Participating Actor
  - The goal of the Traffic Service is to provide traffic data relating to the roads used that the Application finds in the route. This data will be stored in the Database.

Use Cases

- Casual Description

User Case	Name	Description
UC1	GetTrafficHistory	The web interface will ask the user for the region, day selection, and weather. After the user inputs the data for a specific zipcode, it will display which roads have a history of traffic and how congested they are.
UC2	GetDirections	The User will input their starting location and destination. The interface will display on the map the suggested route based on routing information from Google and traffic history in the route area.
UC3	DisplayMap	This use case was implemented solely using Google Maps API. In the future, this could be implemented to be more efficient and custom to our application.
UC4	GetTrafficInfo	Provides traffic data requested by the Traffic Service. The Traffic Service and its specifications are controlled by the Administrator.
UC5	GetWeatherInfo	Provides weather data requested by the Weather Service. The Weather Service and its specifications are controlled by the Administrator.
UC6	GetMobileInfo	Provides the user with a traffic report in a given location via the mobile application on an Android smartphone
UC7	VoiceActivation	Provides the user to allow voice activation while using the application.

UC8	ReportIncident	Allows the user to report anything inhibiting traffic in their area via the mobile application on an Android smartphone and the web application. The entered reports will be added to the database. (This includes Police Reports).
UC9	GetUserReport	Provides user reported data requested by the User-submit Service. The User-Submit Service and its specifications are controlled by the Administrator. This information is given the the Application to further call the Mapping Service and display the information. This was not implemented directly for the final report but this would help in gathering data more efficiently in the future.
UC10	GetPoliceReport	Provides user reported data specific to police reporting, requested by the User-submit Service. The User-Submit Service and its specifications are controlled by the Administrator. This information is given the the Application to further call the Mapping Service and display the information. This was not implemented directly for the final report but this would help in gathering data more efficiently in the future.

Table 5: Casual Descriptions of use cases for a traffic monitoring system

♦ Use Case Diagram

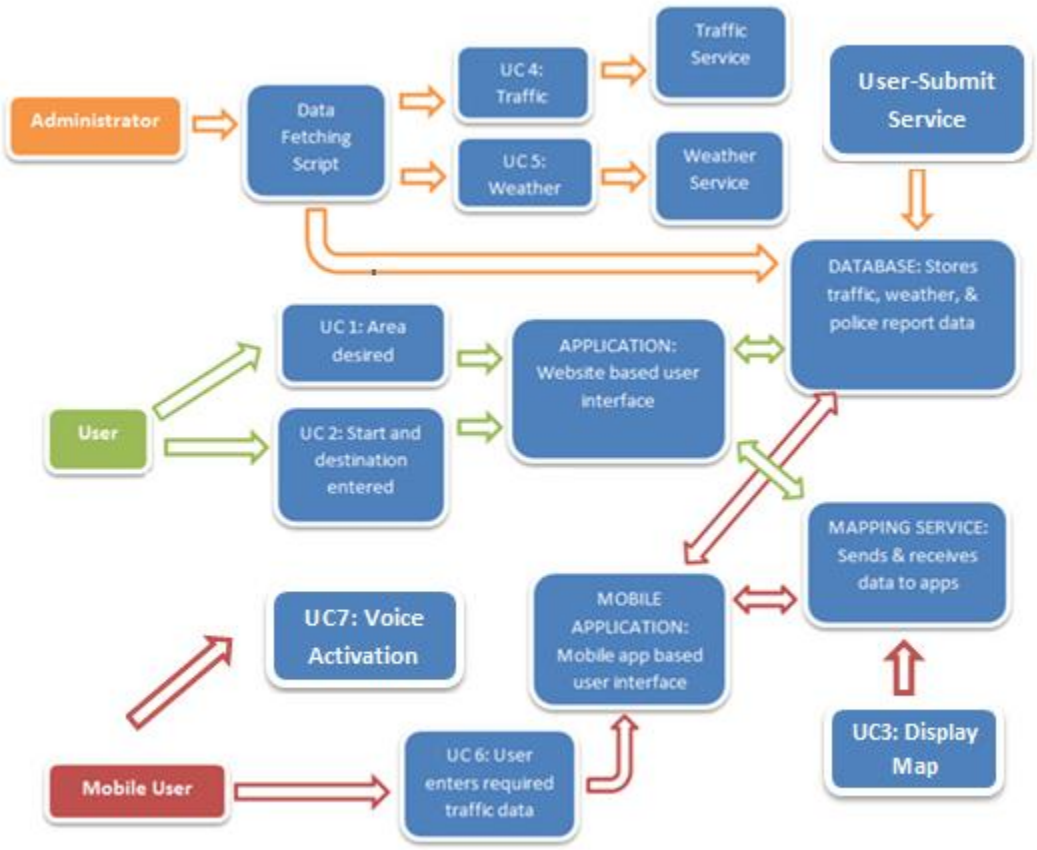


Figure 1: Use Case Diagram for a traffic monitoring system

♦ Traceability Matrix

	UC 1	UC 2	UC 3	UC 4	UC 5	UC 6	UC 7	UC 8	UC9	UC10
REQ 1	X	X								
REQ 2	X		X	X						
REQ 3	X	X	X	X						
REQ 4			X	X						
REQ 5		X						X		
REQ 6					X					
REQ 7		X		X						
REQ 8							X	X		
REQ 9		X				X		X		
REQ 10	X	X						X		
REQ 11	X									
REQ 12		X				X		X		
REQ 13	X	X	X	X						
REQ 14		X								
REQ 15				X	X				X	X
REQ 16										

REQ 17	X	X							X	X
REQ 18	X	X	X	X	X	X	X	X	X	X
TOTAL PW	**35**	**40**	**24**	**31**	**13**	**9**	**6**	**18**	**13**	**13**

Table 6: Traceability Matrix of use cases for a traffic monitoring system

♦ Fully-Dressed Description

UC1: Get Traffic History

Initiating actor: User

Goal: To view the traffic history based upon the user's current region, day selection, and weather.

Participating Actors: Application, Database, Mapping Service, Weather Service

Preconditions:

- Application is available
- Database is not empty
- Services needed are available – time and weather

Postconditions:

- Traffic history is displayed on the map

Main Success Scenario:

1) The User inputs a region/a road and day of week. User must select weather conditions from a drop down box.

2) The Application will send a query to the Database based on the region/road, weather, and day of week.

3) The Database will return all the information it has that matches with those specifications.

4) The Application will create a new array that will give the best indication of the traffic history that meets those specifications.

5) The Mapping Service gets called and it returns the map image to be displayed.

6) The Application displays the map image for the user to view.

Extensions:

5) An empty array is returned

I. Application will return only the raw traffic history.

II. Mapping service will be called and only history based on the region and any other available specifications will be requested.

III. The image will be displayed along with a message indicating the missing variables.

UC2: Get Directions

Initiating actor: User

Goal: To obtain driving directions based on traffic conditions along with an image of the route on a map.

Participating Actors: Application, Database, MappingService, Geocoding Service, and DirectionsService

Preconditions:

- Application is available
- Database is not empty
- All services are available

Postconditions:

- Directions and route images are displayed for the user on the Application.

Main Success Scenario:

1) The User inputs a start and end destination.
2) The User clicks the "Shortest Travel Time" button on the Application.
3) The Application takes the inputted addresses and calls the Geocoding Service.
4) Geocoding Service returns the longitude and latitude of the addresses.
5) The Application uses the longitude and latitude to create a call to the Database to find all the traffic points between the starting and end location.
6) The Database returns all the traffic points and the severity.
7) The Application takes the traffic points and creates the call to the Directions Service.
8) The Directions Service returns a suggested route, which has been optimized with the traffic severity points.

9) The Application calls the Mapping Service with the route information.

10) The Mapping Service returns an image of the map.

11) The Applications displays the map of the route for the User along with the directions.

Extensions:

4) The user input an invalid or not found address.

I. The Application will stop processing and display on the page that the address was invalid or not found.

7) The Application received no traffic data from the database.

I. The Application continues processing. The route will not be influenced by the traffic data and will be a simple quickest route.

UC4: Get Traffic Info

Initiating actor: Application

Goal: To obtain traffic data.

Participating Actors: Database, Traffic Service

Preconditions:

- Database is available
- Traffic Service is available

Postconditions:

- Traffic data stores into the database

Main Success Scenario:

1) Application seeks to obtain the traffic data.

2) The traffic is obtained using a traffic service and the data is reviewed.

3) A script parses the data from the Database to be used for the Application.

UC5: Get Weather Info

Initiating actor: Application

Goal: To obtain weather data.

Participating Actors: Database, Weather Service

Preconditions:

- Database is available
- Weather Service is available

Postconditions:

- Weather data stores into the database

Main Success Scenario:

1) The weather is obtained using a weather service and the data is reviewed.

2) A script parses the data from the Database to be used for the Application.

Extensions:

1) The primary weather service is unavailable.

I. A secondary service will be used to obtain weather data.

UC6: Get Mobile Info

Initiating actor: User on smartphone

Goal: To obtain traffic report on a mobile device.

Participating Actors: Mobile Application, Database, Mapping Service

Preconditions:

- Network communication available

Postconditions:

- Traffic report on mobile device

Main Success Scenario:

1) User enters region/road, day of week, and weather, or start and end locations.
2) Application collects location from phone.
3) Application sends data to web based system.
4) Web based system returns report to application.
5) Application displays the results.

Extensions:

1) The user enters an invalid destination.

I. The user is notified and asked to try again.

UC7: Voice Activation

Initiating actor: User on smartphone

Goal: To obtain the information the user orally inputted.

Participating Actors: Mobile Application, Database

Preconditions:

- Network communication available
- Microphone available
- Voice control available

Postconditions:

- Mobile application obtains user's voice commands.

Main Success Scenario:

1) User selects Voice Activation setting for any action.
2) Mobile Application displays a confirmation screen indication that voice activation is on.
3) Mobile Application prompts User to speak.
4) User enters a command by voice.
5) Mobile Application confirms the user's input.
6) User says Yes if correct.
7) The information is inputted and sent to the Web Application for further analysis.

Extensions:

1) The User says no when the mobile application confirms the user's input.

II. The user is notified and asked to try again.

UC8: Submit Incident

Initiating actor: User on web application or smartphone

Goal: To allow users to share and update traffic conditions using the smartphone app.

Participating Actors: Mobile Application, Database, Mapping Service

Preconditions:

- Network communication and GPS available

Postconditions:

- User's submissions entered into database

Main Success Scenario:

Web Application:

1) User selects incident type from a variety of choices.

2) Report is sent to the database.

Mobile Application:

1) User says incident type using voice activation

2) Application confirms incident type

3) User says "yes" or "no"

4) If yes, report is sent to the database

5) If no, user is prompted to try again

Extensions:

2) The database fails to collect the report.

I. The user is notified and asked to try again.

System Sequence Diagrams

❖ *Use Case 1: Get Traffic History*

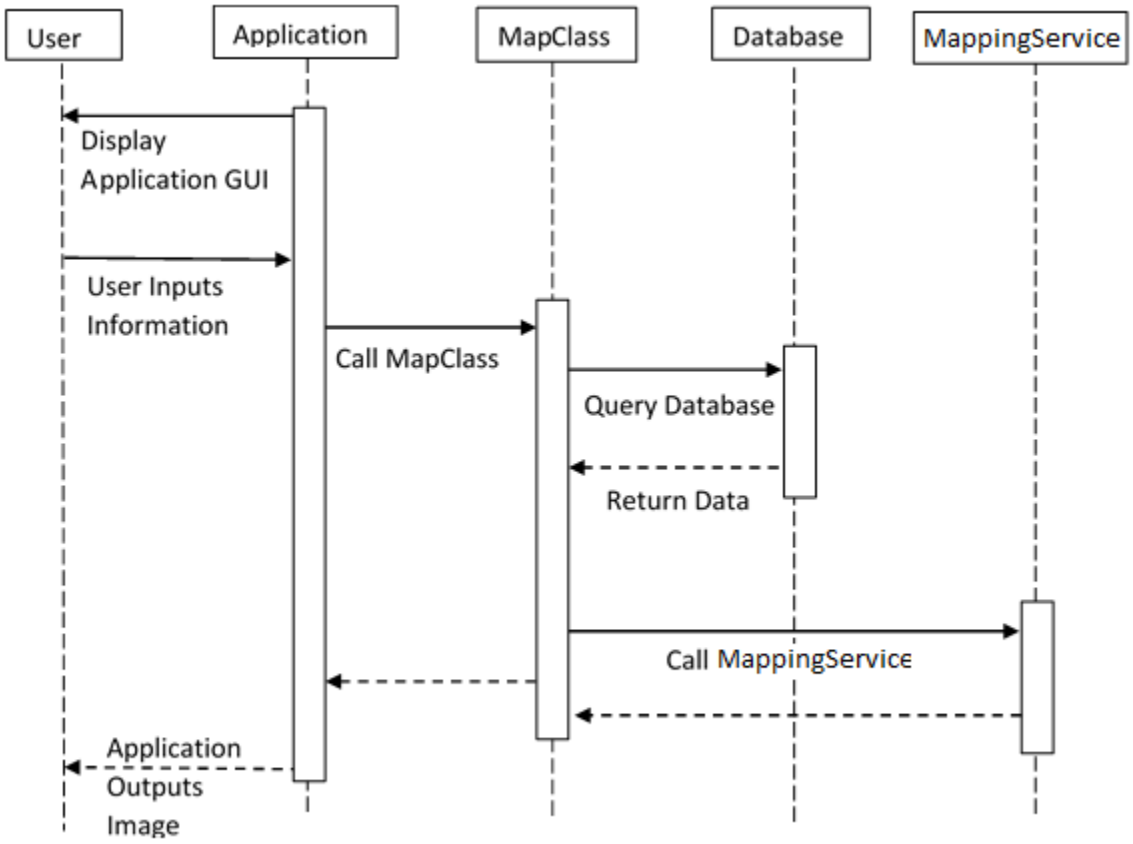


Figure 2: System sequence diagram for Use Case 1

Description:

First, the application presents the user with a GUI that allows him to enter in relevant information, including region, weather conditions, and day of week. User then inputs said information. The application calls it's internal class "MapClass", which then queries the database for the data pertaining to the user's inputs of selected region, weather, and day of week. Database returns the information, which is then parsed by MapClass. Next, MapClass class calls "Mapping Service", returning a map image, able to be displayed on the GUI. Mapclass sends this image to the Application, which then displays the image on the screen for the user.

❖ *Use Case 2: Get Directions*

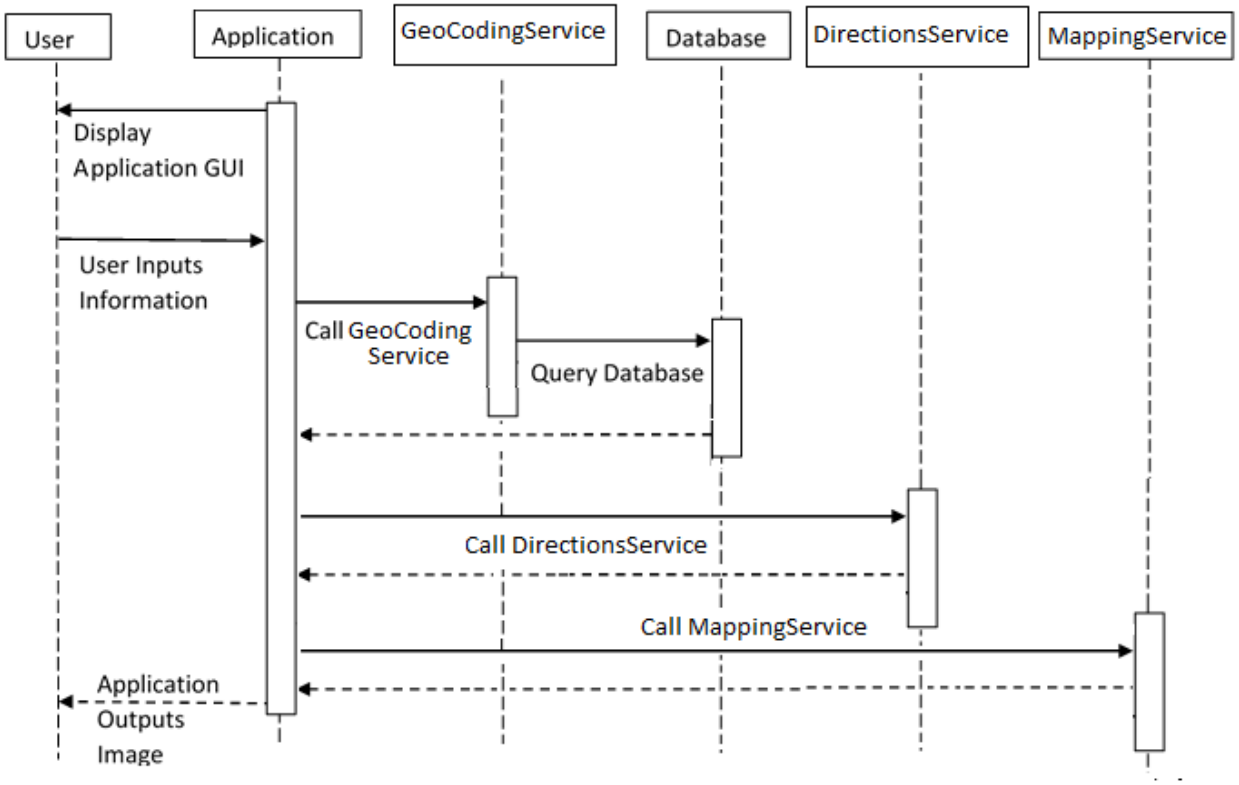


Figure 3: System sequence diagram for Use Case 2

Description:

Application starts by displaying the GUI to the user, who then inputs information relating to his start and end destinations to the system. It then calls class "GeoCoding Service", which returns the longitude and latitude points to the Application. The Application then calls the database with the proper start and end locations. The database returns all the relevant data regarding weather conditions, traffic, etc. to the Application. Next, the Directions Service is called and it returns a suggested route. Lastly, the Application calls the Mapping Service with the route information and this service returns an image of the map which the Application displays for the User.

❖ *Use Case 4: Get Traffic Info*

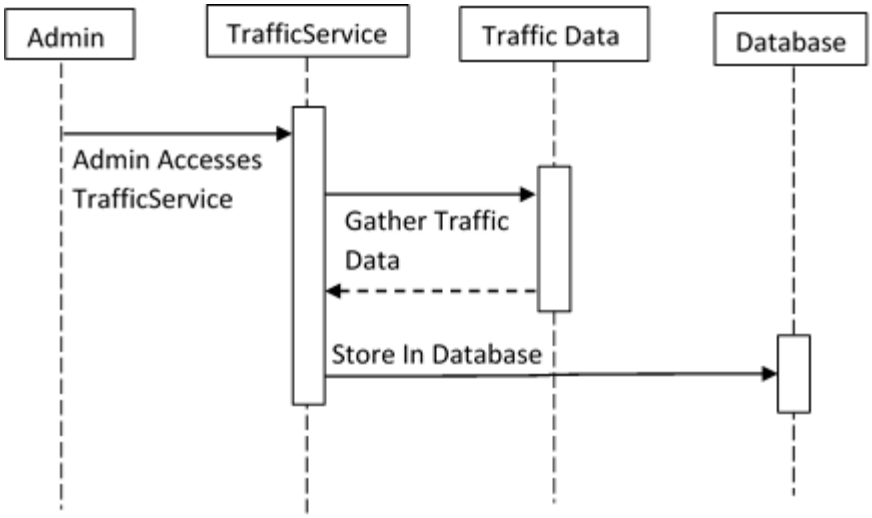


Figure 4: System sequence diagram for Use Case 4

Description:

The administrator either accesses the TrafficService or sets it to be updated periodically. TrafficService class then gathers traffic data from the web, which is sent back to the class. The class then stores the relevant traffic data into the database. This use case can either be done manually by the admin or programmed to be updated every X amount of time.

❖ *Use Case 5: Get Weather Info*

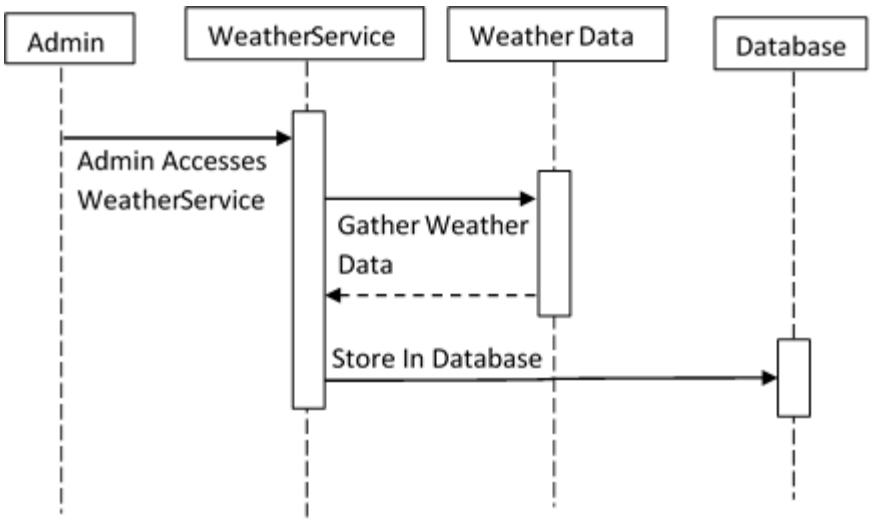


Figure 5: System sequence diagram for Use Case 5

Description:

This is very similar to the use case 4, except instead of traffic data, the Administrator uses the WeatherService to gather weather data. Similarly, this process can either be performed manually or automated.

❖ *Use Case 6: Get Mobile Info*

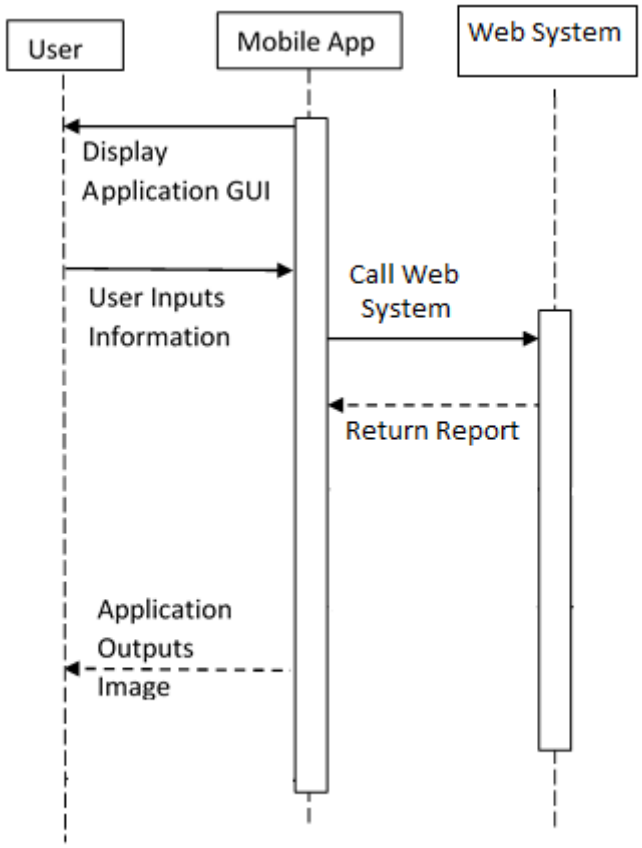


Figure 6: System sequence diagram for Use Case 6

Description:

The mobile app presents a GUI to the user's smartphone screen. The user then inputs relevant information regarding his region, weather, and day of week to the system. The Mobile Application queries the Web System, which internally accesses database and returns relevant information. The Web System also internally requests an image of a relevant map from MappingService class, which sends it back to the Mobile Application. Finally, the Application presents the map image to the user through the screen. The Mobile Application relies on the Web System to complete tasks.

❖ *Use Case 7: Voice Activation*

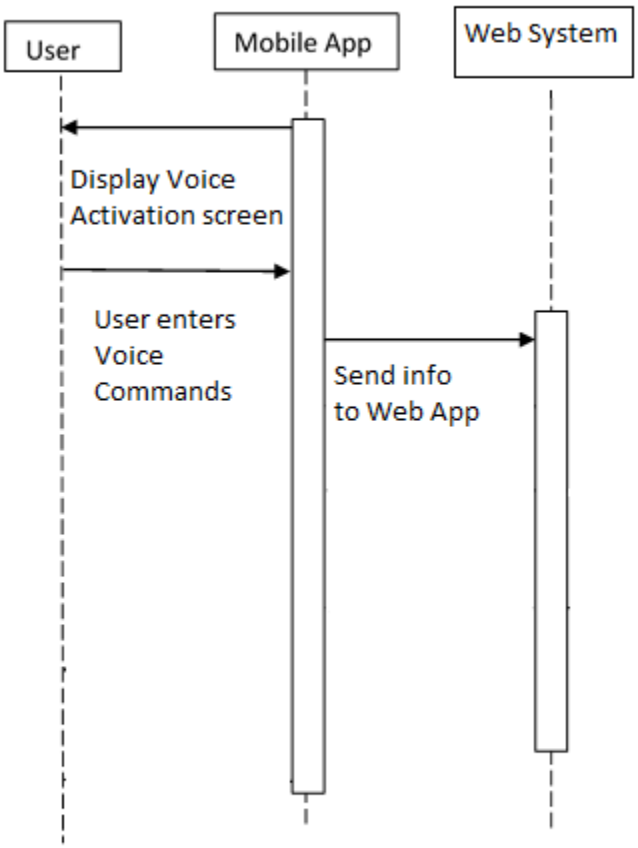


Figure 7: System sequence diagram for Use Case 7

Description:

The User selects the Voice Activation button to proceed with an action. The mobile app presents a GUI to the user's smartphone screen displaying the Voice Activation screen and prompting the User to speak. The User enters commands via voice which the Mobile Application inputs and then sends to the Web Application to process. There is an internal process with the Mobile Application and the User which confirms the inputted commands by the User which are not shown because of redundancies.

❖ *Use Case 8: Report Incident*

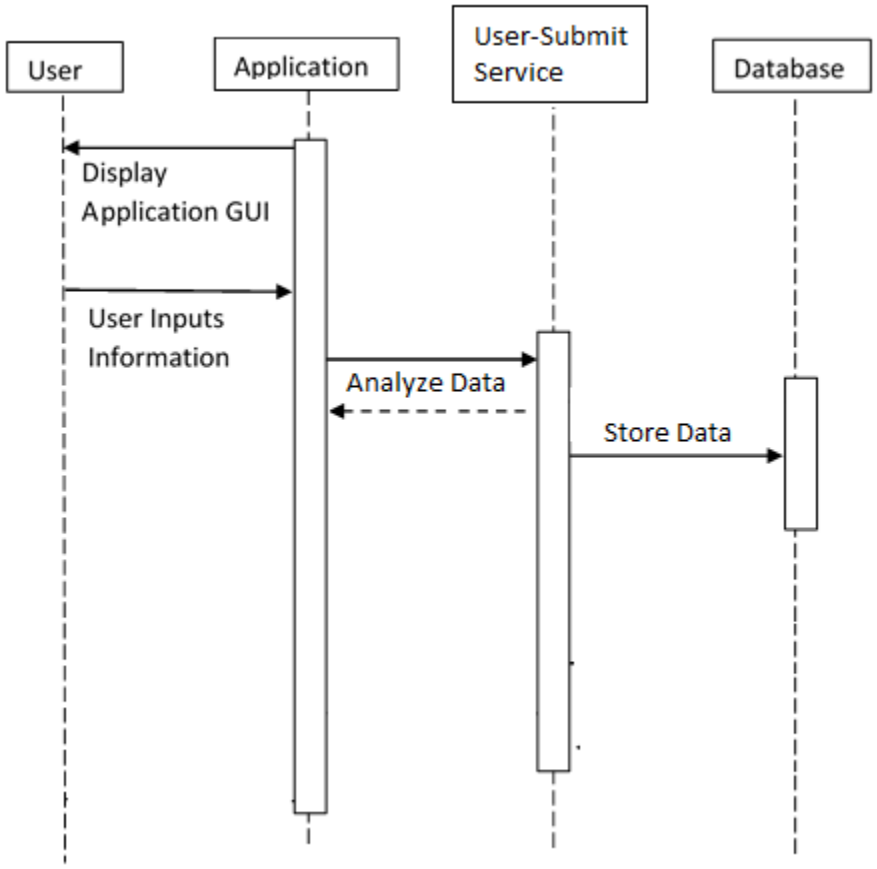


Figure 7: System sequence diagram for Use Case 8

Description:

First, the application presents the user with a GUI that allows him to select what kind of incident to report from a dropdown menu. User then inputs said information.The application calls the User-Submit Service to analyze the inputted data for any discrepencies (though this is very unlikely). User-Submit Service sends this the data to the Database to be used for GetPoliceReport (UC 10) and GetUserReport (UC 9).

User Effort Estimation

- UCP: Use Case Point
- UUCP: Unadjusted Use Case Points
- TCF: Technical Complexity Factor
- ECF: Environment Complexity Factor

UCP = UUCP x TCF x ECF

Actor Type	Description of how to recognize the actor type	Weight
Simple	The actor is another system which interacts with our system through a defined application programming interface (API)	1
Average	The actor is a person interacting through a text-based user interface, or another system interacting through a protocol, such as a network communication protocol.	2
Complex	The actor is a person interacting via a graphical user interface.	3

Table 7: Definition and weight of different actor types

- UUCP = unadjusted actor weight (UAW) + Unadjusted use case weight (UUCW)

Finding UAW:

Actor Name	Description of relevant characteristics	Complexity	Weight
Administrator	Administrator interacts with system through command lines and/or interface.	Complex	3
User	User interacts with system through graphical user interface.	Complex	3
Mobile User	Mobile user interacts with system through graphical user interface.	Complex	3

Application	The Application is a system interacting with our system.	Average	2
Database	The Database is a system interacting with our system.	Simple	1
Mapping Service	The Mapping Service is a system that our system interacts with.	Simple	1
Directions Service	The Directions Service is a system that our system interacts with.	Simple	1
Police Report Service	The Police Report Service is a system that our system interacts with.	Simple	1
Weather Service	The Weather Service is a system that our system interacts with.	Simple	1
Traffic Service	The Traffic Service is a system that our system interacts with.	Simple	1
Social Media/Music Service	The Social Media/Music System is a system that our system interacts with.	Simple	1

Table 8: Unadjusted Actor Weight for a traffic monitoring system

- UAW = (5 x Simple) + (1 x Average) + (3 x Complex) = 5x1 + 1x2 + 3x3 = 16

Finding UUCW:

Actor Type	Description of how to recognize the actor type	Weight
Simple	Simple user interface. Up to one participating actor (plus initiating actor). Number of steps for the success scenario: <= 3. If presently available, its domain model includes <=3 concepts.	5
Average	Moderate interface design. Two or more participating actors. Number of steps for the success scenario: 4 to 7/ If presently available, its domain model includes between 5 and 10 concepts.	10
Complex	Complex user interface or processing. Three or more participating actors. Number of steps for the success scenario: >= 7. If available, its domain model includes >= 10 concepts.	15

Table 9: Definition and weight of different actor types for unadjusted use case weight

Use Case	Description	Category	Weight
Traffic Account (UC 1)	Complex user interface. Three participating actors. 7 steps for success scenario.	Complex	15
Get Directions (UC 2)	Complex user interface. Four participating actors. 12 steps for success scenario.	Complex	15
Get Traffic Info (UC 4)	Simple user interface. Two participating actors. 3 steps for success scenario.	Simple	5
Get Weather Info (UC 5)	Simple user interface. Two participating actors. 3 steps for success scenario.	Simple	5
Get Mobile Info (UC 6)	Moderate interface design. Three participating actors. 5 steps for success scenario.	Average	10
Social Media/Music (UC 7)	Simple user interface. Two participating actors. 3 steps for success scenario.	Simple	5
Report Traffic (UC 9)	Moderate interface design. Three participating actors. 5 steps for success scenario.	Average	10

Table 10: Unadjusted Use Case Weight for a traffic monitoring system

- UUCW = (3 x Simple) + (2 x Average) + (2 x Complex) = 3x5 + 2x10 + 2x15 = 65
- **UUCP = UAW + UUCW = 16 + 65 = 81**

Finding TCF:

Technical Factor	Description	Weight
T1	Distributed system (running on multiple machines)	2
T2	Performance objectives (are response time and throughput performance critical?)	1

T3	End-user efficiency	1
T4	Complex internal processing	1
T5	Reusable design or code	1
T6	Easy to install (are automated conversion and installation included in the system?)	0.5
T7	Easy to use (including operations such as backup, startup, and recovery)	0.5
T8	Portable	2
T9	Easy to change (to add new features or modify existing ones)	1
T10	Concurrent use (by multiple users)	1
T11	Special security features	1
T12	Provides direct access for third parties (the system will be used from multiple sites in different organizations)	1
T13	Special user training facilities are required	1

Table 11: Technical Complexity Factors weights

Technical Factor	Description	Weight	Perceived complexity	Calculated factor
T1	Distributed system	2	2	4
T2	Expected fast loading pages & good performance on scripts	1	4	4
T3	End-user efficiency has no expectations/is not predictable	1	3	3
T4	Complex internal processing	1	4	4
T5	Reusable web pages and database tables	1	3	3
T6	Installation not required	0.5	1	0.5
T7	Easy to use	0.5	5	2.5
T8	Portability is not a concern	2	1	2
T9	Easy to change by developers	1	3	3
T10	Concurrent use available	1	2	2
T11	Security with user passwords	1	2	2
T12	No direct access for third parties	1	0	0
T13	No special user training required	1	0	0

Table 12: Technical Complexity Factors for a traffic monitoring system

- TCF Total: 30
  - C1 = 0.6
  - C2 = 0.01
  - **TCF = 0.6 + (0.01 x 30) = 0.9**

Finding ECF:

Environmental Factor	Description	Weight
E1	Familiar with the development process	1.5
E2	Application problem experience	0.5
E3	Paradigm experience	1
E4	Lead analyst capability	0.5
E5	Motivation	1
E6	Stable requirements	2
E7	Part-time staff	-1
E8	Difficult programming language	-1

Table 13: Environmental Complexity Factors weights

Environmental Factor	Description	Weight	Perceived Impact	Calculated Factor
E1	Beginner familiarity with the UML-based development	1.5	1	1.5
E2	Some familiarity with application problem	0.5	2	1
E3	Object-oriented knowledge needed	1	2	2
E4	Average lead analyst	0.5	1	0.5
E5	Highly motivated, some members occasionally slacking	1	4	4
E6	Stable requirements expected	2	3	6
E7	No part-time staff involved	-1	0	0
E8	Programming language is of average difficulty	-1	3	-3

Table 14: Environmental Complexity Factors for a traffic monitoring system

- ECF Total = 12
- C1 = 1.4
- C2 = -0.03
- **ECF = 1.4 + (-0.03x12) = 1.04**
- **UCP = UUCP x TCF x ECF = 81 x 0.9 x 1.04 = 75.82 = 76 use case points**
- **Duration = UCP x PF**
- **Duration = 76 x 28 = 2128 hours**

- We will assume that each developer will spend 20 hours per week working on this project. With 6 group members that is 120 hours per week being spent.
- 2128 hours / 120 = 17.73 weeks
- So the project would take about 18 weeks to be completed.

Domain Analysis

Domain Model

- Application Users:

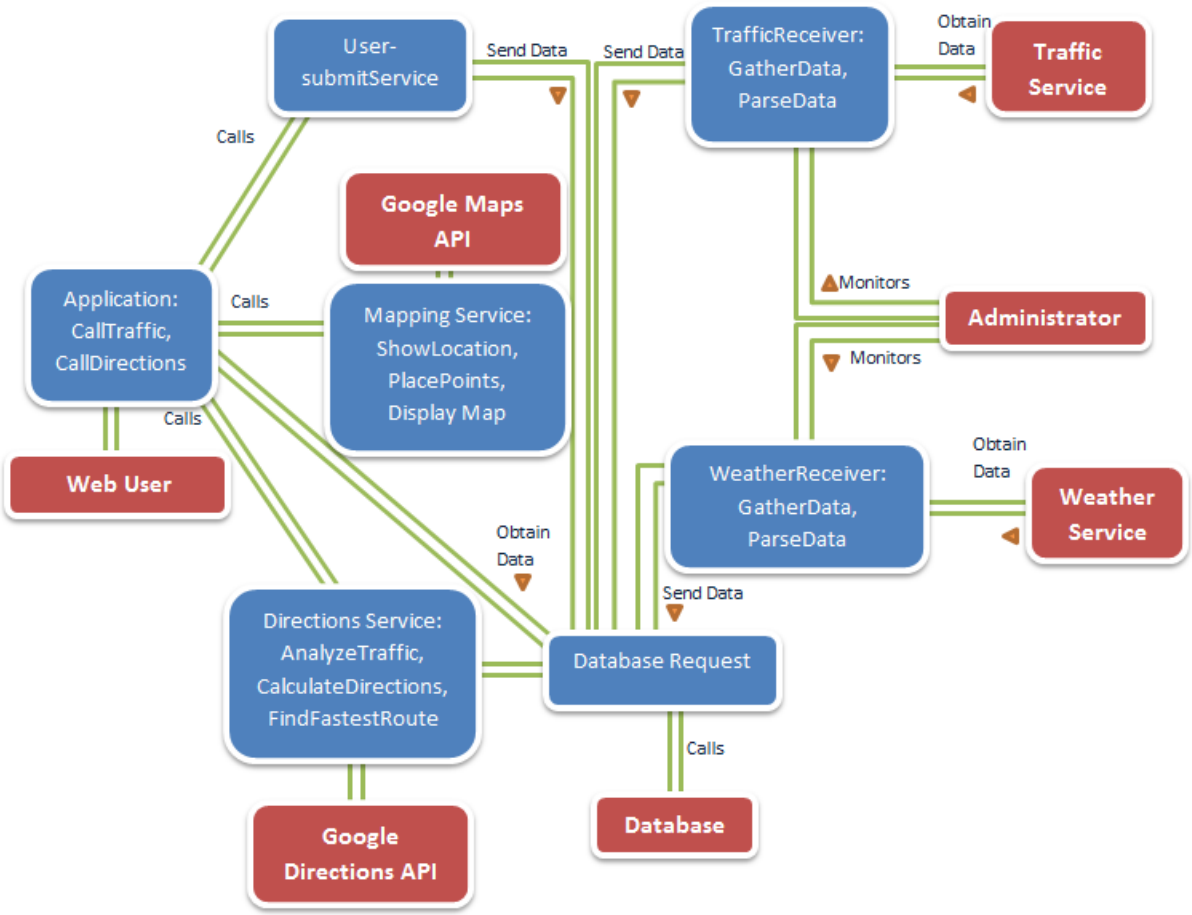


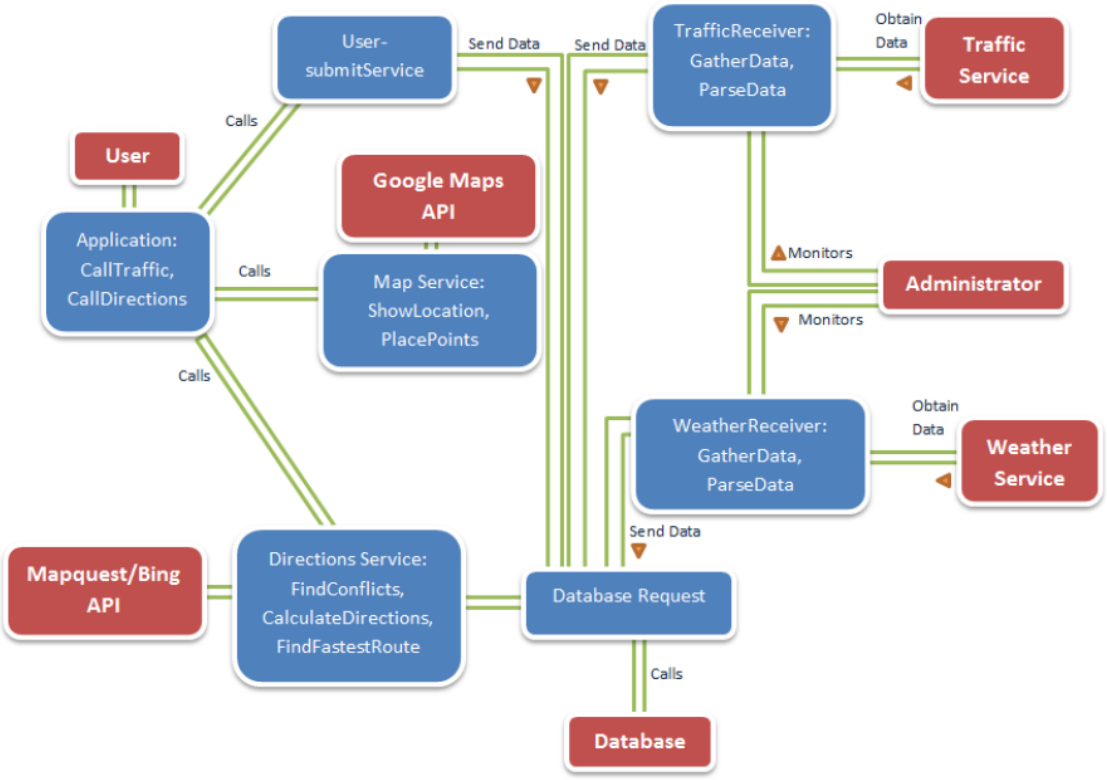
Figure 10: Domain Model diagram

Description:

The Domain Analysis shows that the application is accessible to the User. These bounds are defined so as to create easy user experience. The Application can call the Mapping, Directions Service, or User-Submit Service. Although these services are similar, the procedures needed to complete each action require different APIs to be used for each function. The Google Maps API is more suited for placing unique points on the map, while the Google Directions API is preferable for finding routes while avoiding locations that have high traffic severities. These functions call the Database to receive the required information about the traffic points they wish to analyze or display.

The Administrator has control of the two scripts that can receive data from Web Services. The traffic and weather scripts gather and parse the data received from the Web Services, and they store that data in the database to be used by the main Application. The user reporting data script will gather data from user submitted service and store that data into the database.

The Domain Model for Application users from Report 1 is the following:



The updated version of the domain model was almost identical to the one from Report #1 (shown above). The biggest changes were that we decided to use the Google Maps API instead of the MapQuest/Bing API because it was easy to incorporate and much more accurate than the MapQuest and Bing. The other change was seen within the Mobile Application, where we added the GetDirections function. We made note to add this into the domain analysis because it is one of the key functions of the Mobile Application. We did not have this function completely designed or implemented for the first report and that is why it was left out then. Lastly, we added a link between the Application and Database that obtains data. This was imperative because the Application needs to be able to access the data from the Database and then call other Services to analyze or manipulate it. Without this link, data would not be appearing on our Application.

Mobile Users:

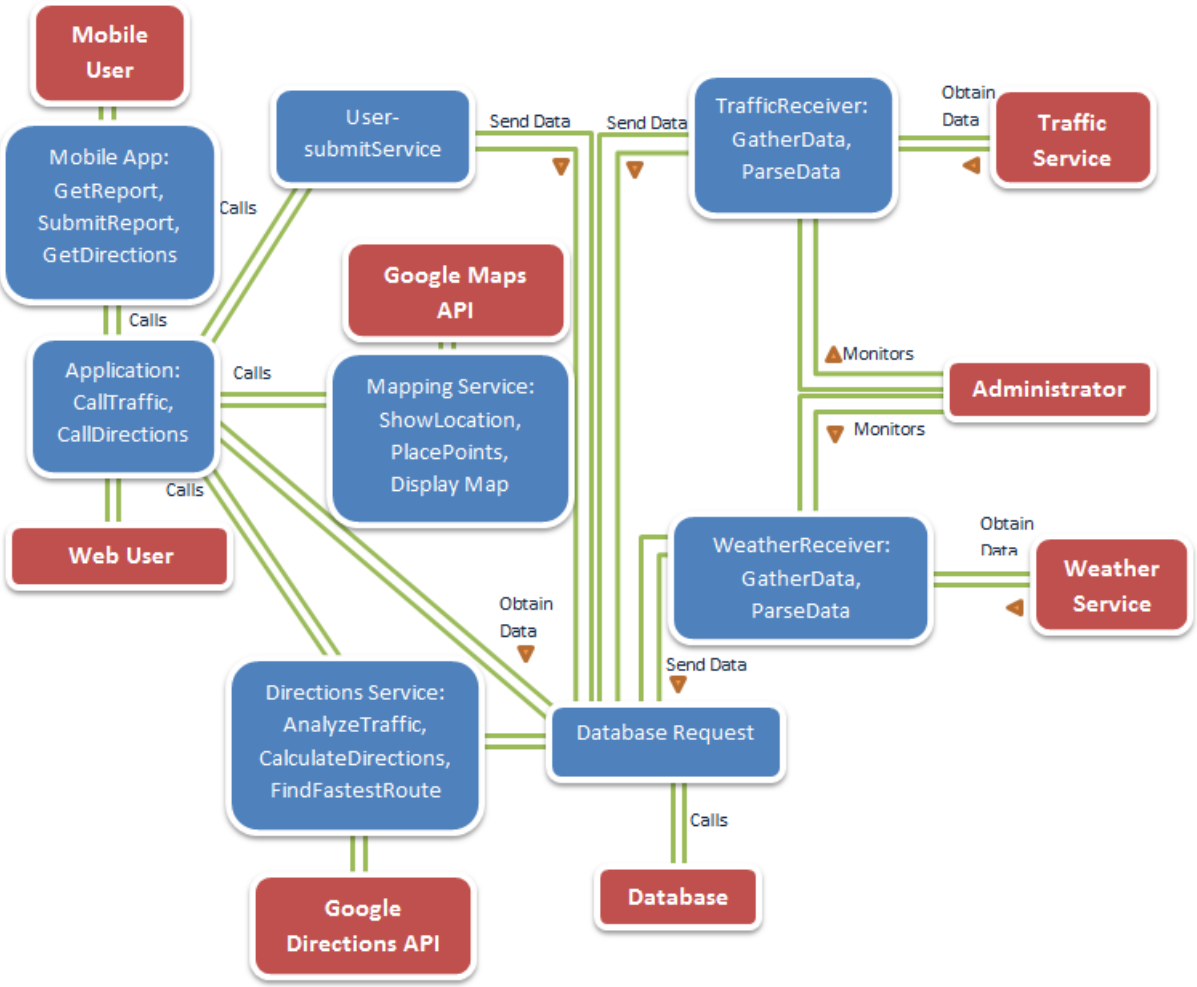


Figure 11: Domain Model diagram for Mobile Users

Description:

The Mobile Application is made to mimic the functionalities of the website Application. Because of this, the best way to replicate the functions of the website would be to link the Mobile Application to the Web Application. The Mobile Application gets a series of inputs from the Mobile User, and the Mobile Application sends that data to the main Application. The Application then parses the data from the Mobile Application. At this point, the Application behaves in the exact same way as the previous Domain Diagram. The Mobile Application receives a series of inputs for a mobile traffic report. These inputs are again sent to the main Application. This time, the Application parses the data received, and it enters it straight into the database, bypassing the MappingService and DirectionsService.

The same changes were implemented from Report 1 into Report 3 as described for the Application User Domain Model.

♦ Traceability Matrix

	Application	Database	Mapping Service	Directions Service	User Report Service	Weather Service	Traffic Service	Voice Activation	Mobile Application
UC1	X	X	X			X	X		
UC2	X	X	X	X		X	X		
UC3	X	X	X			X	X		
UC4		X					X		
UC5		X				X			
UC6	X							X	X
UC7								X	X
UC8	X	X			X				
UC9	X	X			X				
UC10	X	X			X				

Table 15: Traceability Matrix for Domain Analysis

Description of Traceability Matrix for Domain Analysis:

The following Domain Concepts are used during the course of this project: Application, Database, Mapping Service, Directions Service, User-Submit Service, Weather Service, Traffic Service, Voice Activation and Mobile Application. They are all imperative to the success of the project. The Application is the most important of the concepts, as it links the user to the project, without the application, there would be nothing for the User to access. The Database also plays a crucial role in the Domain Analysis as it holds all the historical traffic information, weather information, and user reported information – all key portions of the data that is outputted and displayed on the maps for users to see. This is the concept that is used the most from the use cases, because most of the use cases need to be able to access data so that they can analyze it, return it, or store it. The Mapping Service and the Directions Service are the significant concepts for the GetDirections use case. The Directions Service calls the Google API and combines this with the historical traffic patterns from our database to return a fastest route for the user. The Mapping Service is vital to displaying the map. The User Report Service is used for ReportIncident, GetUserReport, and GetPoliceReport. This service accesses the database when the “Submit Incident” button is clicked, and it also accesses the database to retrieve the information for getting user/police report. Once it retrieves this information, it links to the Application which then calls the Mapping Service to display the map. The Weather and Traffic Services, though not used for many use cases, are extremely important in parsing the traffic and weather data. These services pull relevant data from websites and specific APIs, run it through algorithms that combine it with our previous historical information, and then push it into the Database. The Voice Activation concept is only implemented for the Mobile Application. This concept is thus only traced to Use Case 6 (Voice Activation) and Use Case 7 (Get Mobile Info). Lastly, the Mobile Application concept is linked to the use cases that deal with the Mobile Application – use case 6 and use case 7. It is clear that there are significant correlations between the concepts and the use cases, as should be for a domain analysis.

❖ Use Case 1: Get Traffic History

Concept definitions

Responsibility Description	Type	Concept Name
Takes in user input to find which areas the user is requesting traffic history.	Doing	Application
Database returns the information matching user input and application return output that displays most accurate estimation of traffic history.	Doing	Application
Mapping service is called and displays the map image for user.	Communicating & Doing	MappingService
Should previously have the traffic, time, and weather that is necessary for user operation.	Knowing	Database

Table 16: Concept Definitions for Use Case 1

Association Definitions

Concept Pair Name	Associated Definition	Association Name
Application ↔ Mapping Service	The application will send the current user location to the Mapping Service. The Mapping service then returns the map area to the user. The Mapping service returns the map area to the user	Provides Data
MappingService ↔ Database	The Mapping Service uses the Database information to display the proper direction, time, weather, traffic congestion, location, etc. through the internal algorithm, and displays it for the user.	Provides Data

Table 17: Association Definitions for Use Case 1

Attribute Definitions

Concept	Attributes	Attribute Description
Application	Weather	Weather desired by user
	Day	Day of week desired by user
	Location	Geographical location desired by user
Database	Traffic Data	Traffic congestion associated with the time, date, and location
	Weather Data	Previous weather history associated with certain areas

Table 18: Attribute Definitions for Use Case 1

- Use Case 2: Get Directions

Concept Definitions

Responsibility Description	Type	Concept Name
Receives input from user based on where he would like to receive traffic information.	D	Application
Stores all relevant data from user/police reports, traffic reports, and weather data.	K	Database
Reads starting and end locations from the user. Estimates the fastest route from start to end, taking into account weather, time of day and historic traffic data.	D	DirectionsService

Table 19: Concept Definitions for Use Case 2

Association Definitions

Concept Pair Name	Associated Definition	Association Name
Application ↔ DirectionsService	DirectionsService receives the user's starting and end locations from the Application, and sends back directions and map image of the route.	Provides Data
DirectionsService ↔ Database	DirectionsService queries the database and retrieves the relevant information regarding traffic.	Provides Data

Table 20: Association Definitions for Use Case 2

Attribute Definitions

Concept	Attributes	Attribute Description
Application	Start Location	Starting address location
	End Location	Destination address location
Database	Traffic Data	Data of traffic reports based on time of day and location
	Weather Data	Data of weather conditions of current input location
DirectionsService	Directions	List of directions from address of Start Location to End Location

Table 21: Attribute Definitions for Use Case 2

- Use Case 4: Get Traffic Info

Concept Definitions

Responsibility Description	Type	Concept Name
Keeps archive of traffic history	K	Database
Obtain the traffic history from traffic service website	D	TrafficService

Table 22: Concept Definitions for Use Case 4

Association Definitions

Concept Pair Name	Associated Definition	Association Name
TrafficService → Database	TrafficService uses the information gained from Google Maps API and keeps it in the data base	Provides Data

Table 23: Association Definitions for Use Case 4

Attribute Definitions

Concept	Attributes	Attribute Description
TrafficService	Region	Location where traffic is happening
	Severity	Congestion levels of traffic
	Day	Day of week for Traffic
Database	Traffic Data	Traffic intensity associated with location and day of week

Table 24: Attribute Definitions for Use Case 4

❖ Use Case 5: Get Weather Info

Concept Definitions

Responsibility Description	Type	Concept Name
Keeps archive of weather history	K	Database
Obtain the weather history from a weather service website	D	WeatherService

Table 25: Concept Definitions for Use Case 5

Association Definitions

Concept Pair Name	Associated Definition	Association Name
WeatherService → Database	WeatherService uses the information gained from Google API and keeps it in the database	Provides Data

Table 26: Association Definitions for Use Case 5

Attribute Definitions

Concept	Attributes	Attribute Description
WeatherService	Region	Location where weather was recorded
	Weather Type	Type of weather recorded
	Day	Day of week for Weather
Database	Weather Data	Weather associated with location and time

Table 27: Attribute Definitions for Use Case 5

- Use Case 6: Get Mobile Info

Concept Definitions

Responsibility Description	Type	Concept Name
Receives input from user and display traffic information based on what the user inputs into the mobile application.	D	Mobile Application
Mobile Application sends user inputs to main web application	D	Application
Contains relevant weather, traffic and user report data	K	Database
Mapping service is called and displays the map image for user.	D	MappingService

Table 28: Concept Definitions for Use Case 6

Association Definitions

Concept Pair Name	Associated Definition	Association Name
Mobile Application → Application	Mobile app has same functionality as main application.	Software Port
MappingService ↔ Database	The Mapping Service accesses the database to display the proper direction, day selection, weather, traffic congestion, region, etc. through the internal algorithm, and displays it for the user on the mobile app.	Provides Data

Table 29: Association Definitions for Use Case 6

Attribute Definitions

Concept	Attributes	Attribute Description
Mobile Application	Weather	Type of weather inputted
	Day	Day Selection inputted
	Region	Region inputted by user
	Start Location	Starting address location
	End Location	Destination address location
Database	Traffic Data	Data of traffic reports based on time of day and location
	Weather Data	Data of weather conditions of current input location
	User Report Data	User submitted user/police data reports

Table 30: Attribute Definitions for Use Case 6

❖ Use Case 8: Report Traffic Activity (Web)

Concept Definitions

Responsibility Description	Type	Concept Name
User inputs user report of the area they are in based upon longitude/latitude coordinates provided by selection point on map.	D	Application
Receives the user report and location inputted by user.	K	Database
Analyzes this data and starts to create patterns based upon time and location.	D	Database
Database sends this updated information to the Application.	D	Application

Table 31: Concept Definitions for Use Case 9

Association Definitions

Concept Pair Name	Associated Definition	Association Name
Application -> Database	Application receives user input and sends this data to the Database.	Provides data
Database -> Application	After storing and analyzing this data, the application sends this new information to the application.	Provides data

Table 32: Association Definitions for Use Case 9

Attribute Definitions

Concept	Attributes	Attribute Description
Application	Time	Time taken from mobile device
	Report Type	User inputted traffic intensity
	Location	Location taken from mobile device
Database	User Report Data	User reports with their types, location and times, creating a pattern along the way.

Table 33: Attribute Definitions for Use Case 9

Please keep in mind that process for submitting a user report is slightly different on the Mobile Application. The commands are entered through voice activation, so the user orally selects the user report type, rather than the user manually filling it out on the application. The Mobile application then links to the Web Application, so the rest of the steps are the same.

Interaction Diagrams

❖ Use Case 1: Get Traffic History

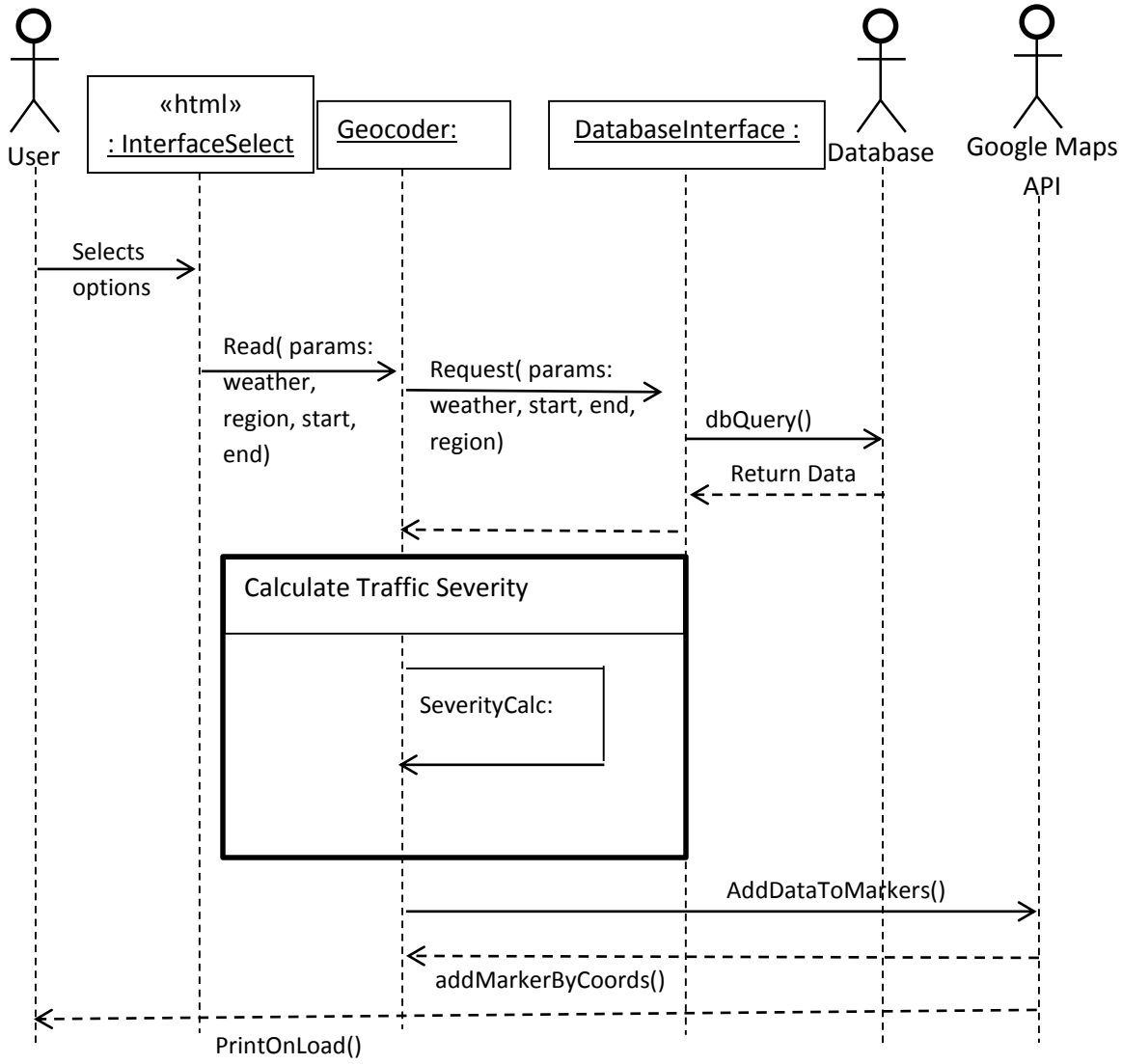


Figure 12: Interaction diagram for Use Case 1

Description:

The user starts by selecting his/her options on the webpage on the dropdown menus. The Geocoder object is then initialized, which calls the Database interface. The database interface performs a series of queries to the SQL database, based on the inputs the user entered. After retrieving the data, the geocoder calculates the traffic

severity using SeverityCalc() function, and calls the Google Maps API service. The function AddDataToMarkers() takes the data and assigns each a marker type for differing traffic severity. Finally, addMarkerByCoords() plots the markers to the Google Map, and PrintOnLoad() prints the map to load to the web application.

This interaction diagram employs the design practice of Publisher-Subscriber, in the form of the Geocoder and DatabaseInterface classes. This is because the Geocoder sends information to the DatabaseInterface (the parameters of the user's input requests), and the DatabaseInterface performs querying of the database, disregarding the specific information being sent to it. Also, in this way, this means the two modules are very loosely coupled.

❖ Use Case 2: Get Directions

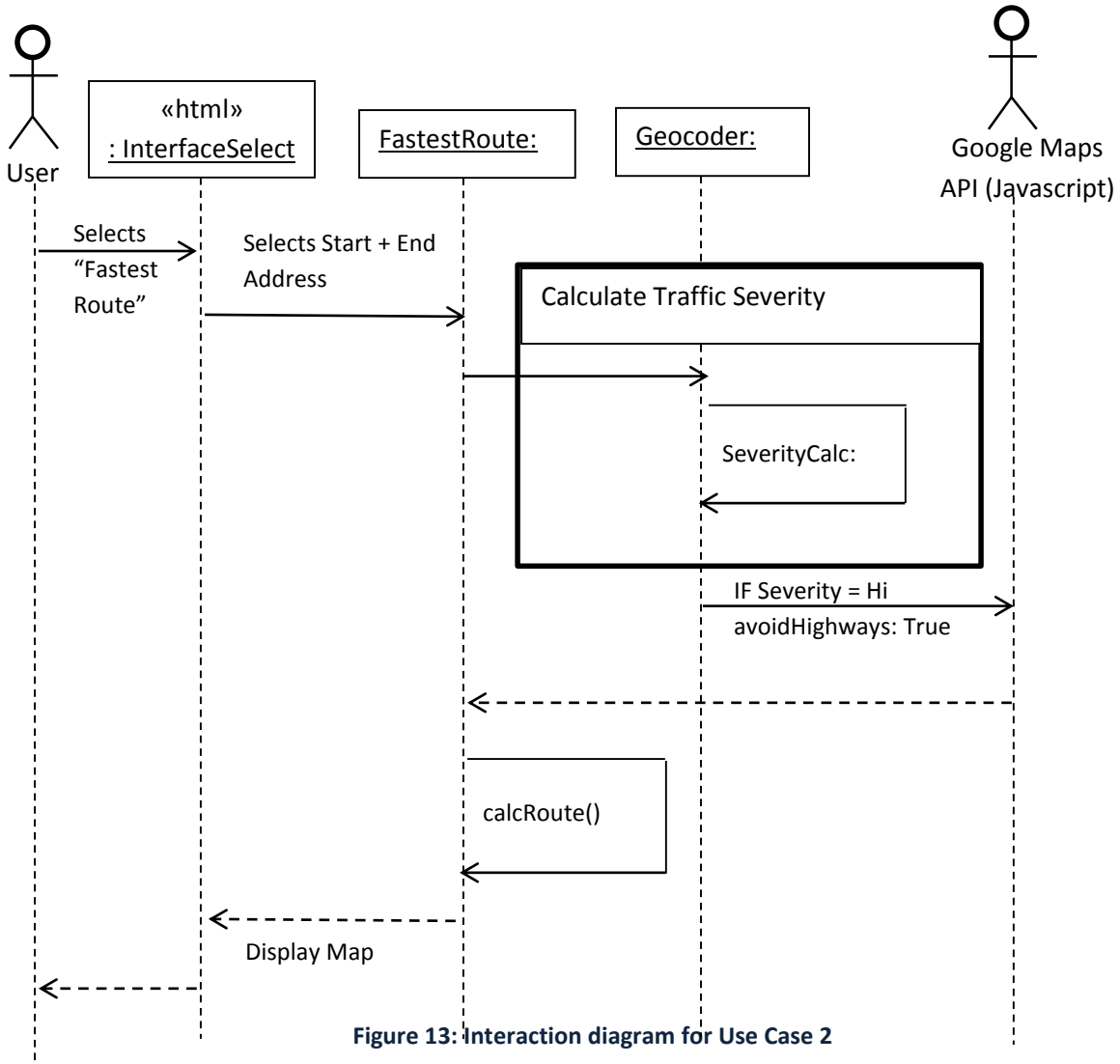


Figure 13: Interaction diagram for Use Case 2

Description:

First, user selects “Fastest Route” to be brought to the Fastest Route page. Then he/she selects start and end address for directions. This calls the Geocoder with the proper information in terms of start and end locations. Not shown in this diagram (to reduce clutter and save space) are the same interactions between geocoder → databaseinterface → database in order to get relevant traffic and weather data. After severity is calculated, the Google Maps API is called, displaying the map and using its functions to get ideal directions (calcRoute()). The avoidHighways setting is set to true if the severity along the highway is high.

❖ Use Cases 4/5: Get Traffic Info, Get Weather Info

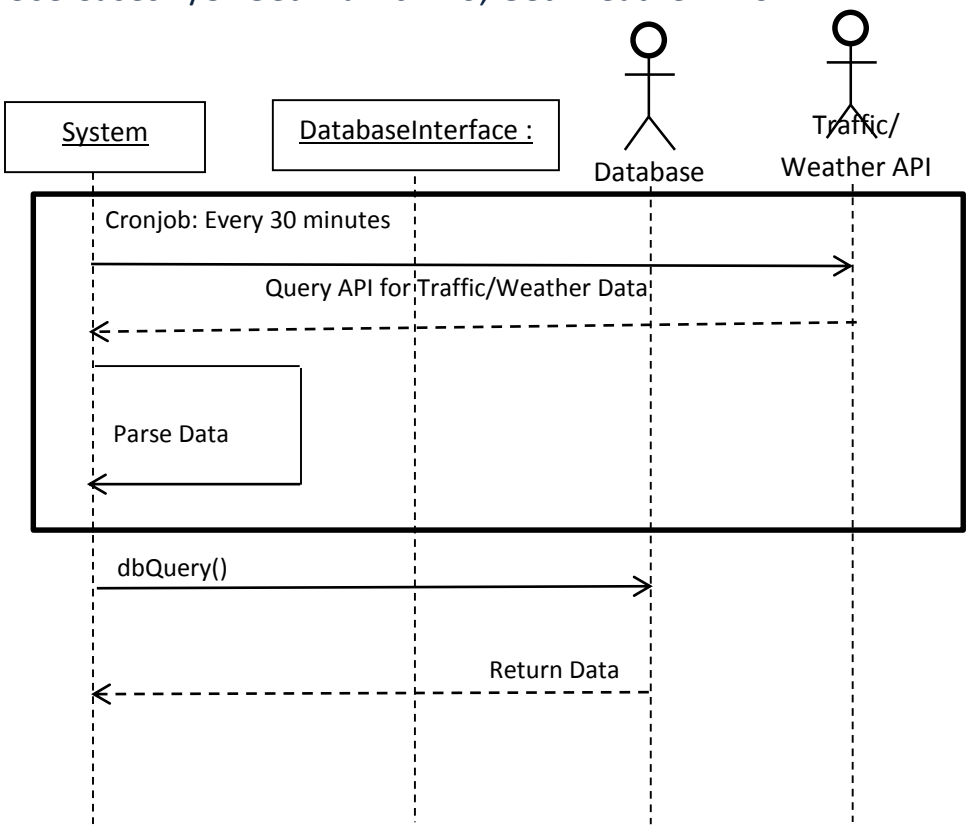


Figure 14: Interaction diagram for Use Cases 4/5

Description:

Because the use case for 4 and 5 are very similar except for the differences in content of data (traffic vs. weather), the interaction diagrams were combined into one

representing both. In order to collect traffic and weather data from online resources, the system first runs a script to call the online API to gather relevant data. After the websites supply the data, the system then parses the data in order to be able to store neatly into the database. A cron job runs this above process every 30 minutes to routinely collect data. Afterwards, the system can then query the database to view the data that was collected in the database.

❖ Use Case 6: Get Mobile Info

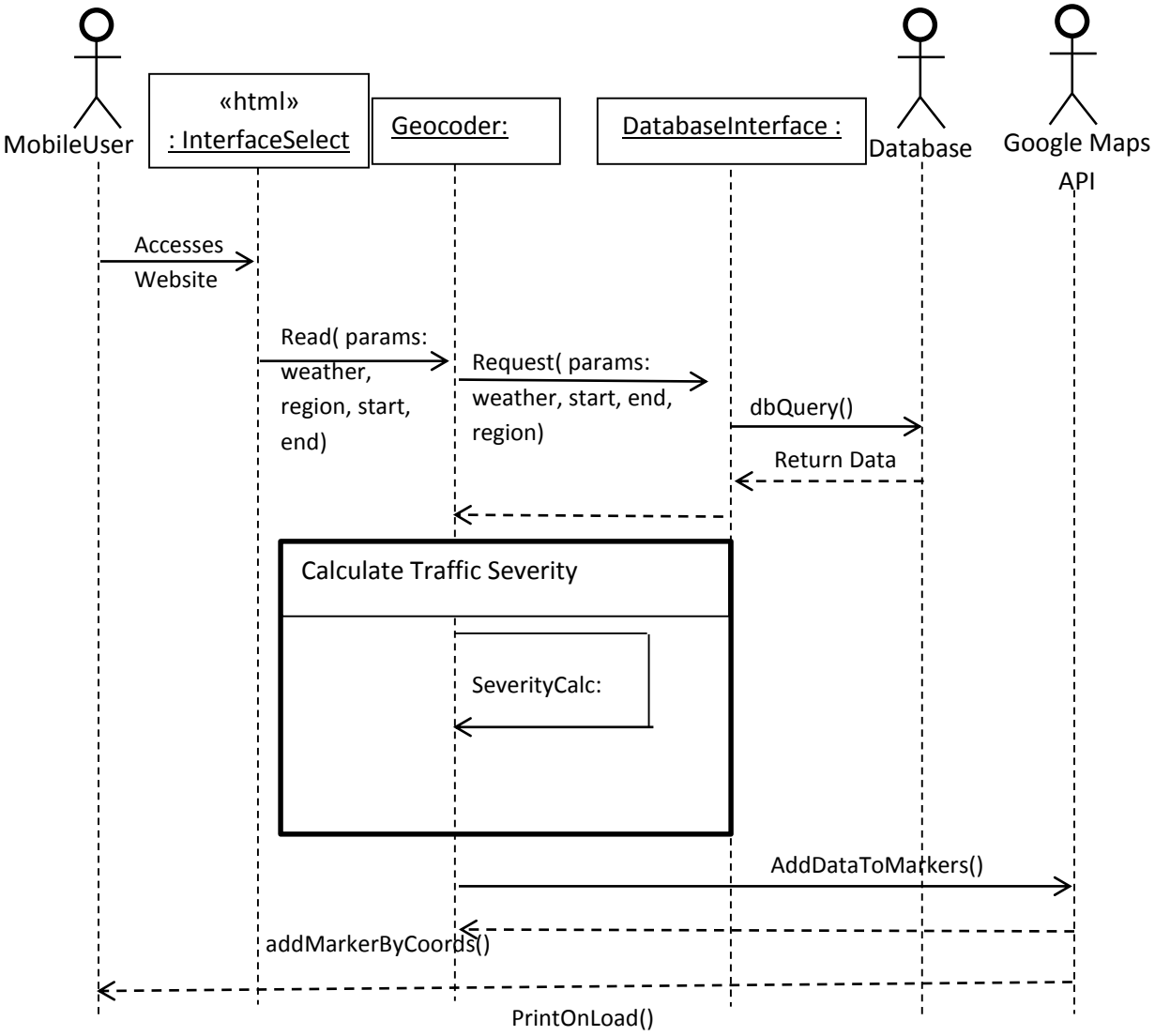


Figure 15: Interaction diagram for Use Case 6

Description:

As you can see, this interaction diagram is almost identical to the first use case, because all the mobile app does is access the content on the website and then returns the information in the same way the web app does. As such, since it has the same interactions as the web application, one can consider this utilizing the decorator pattern design principle. This is because the behavior of mobile access is added to the application, without changing the operation and functionality of the main application itself.

❖ Use Case 9: Report Incident

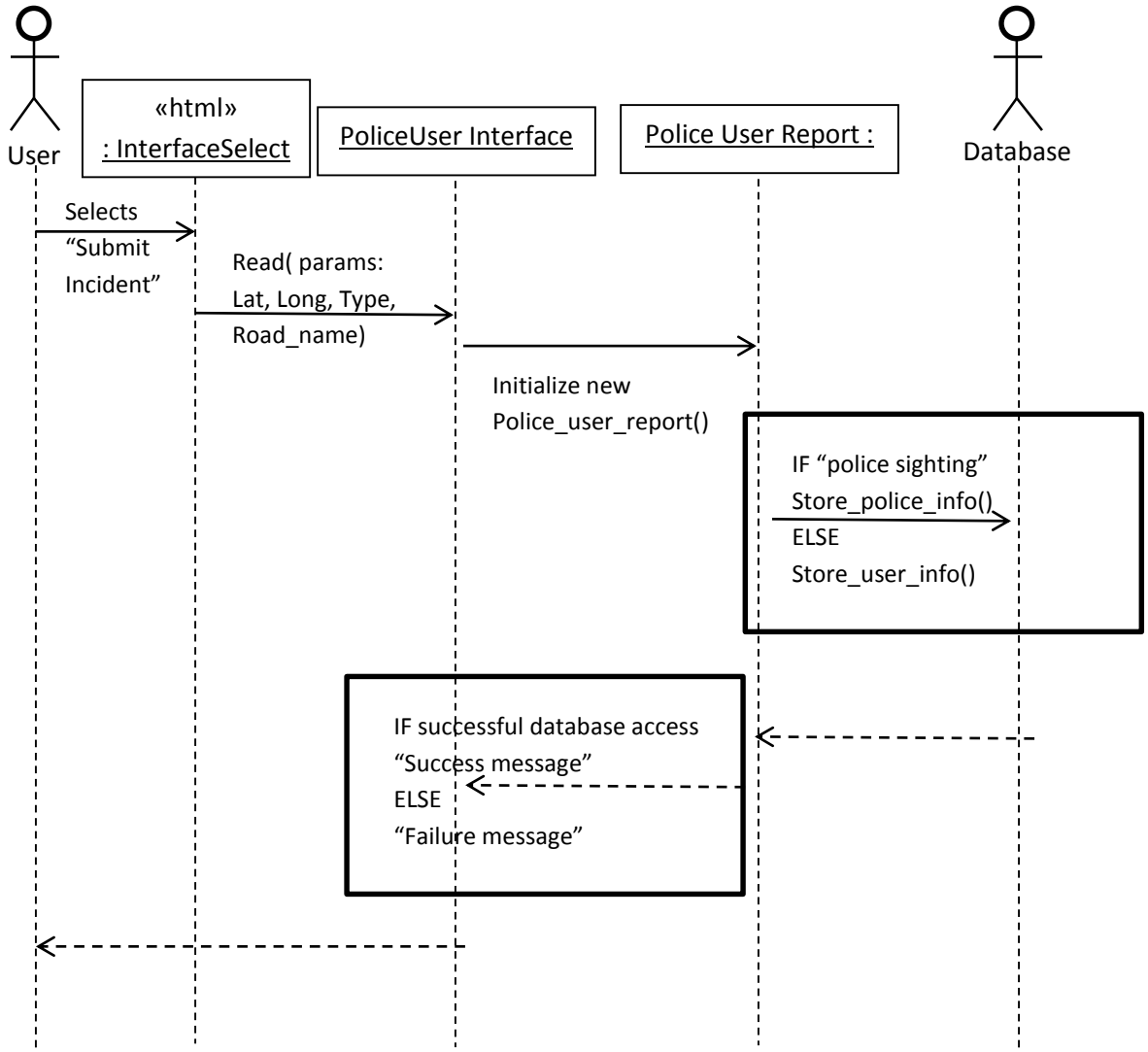


Figure 16: Interaction diagram for Use Case 9

Description:

The user must first select the option “Submit Incident”, which takes him/her to the incident report page, calling the police_user_interface. The user inputs the parameters of the incident report, including the type of report (police or user report). A new instance of “police_user_report()” class is instantiated from here. If the user input “police report” then store_police_info() is called, otherwise store_user_info() is called. These functions store the incident into the database. The application reports a success message if the incident was successfully stored, otherwise and error message appears. Then the user can go back to the main app.

Although it is not shown in the interaction diagram and should belong to its own separate use case, the option for users to view the incident reports from others actually uses the same class as submitting one, Police_user_report. Although there is no inheritance in this class, the different functions the class provides still acts like having two children classes, one for each operation. In this way, the Police_user_report class can be considered an example of a decorator design pattern, in which new features are added onto an existing class without changing the class’s main functionality.

Design Patterns:

The goal behind our design patterns was to ensure reusability between different use cases. We wanted our interaction diagrams to be easy to understand and follow a similar format, so that the format can be followed for other use cases. To do this we tried to use reusable actors between use cases, such as Mobile User and Database. We tried to make the objects reusable too, such as Application and DatabaseConnect. The actual interactions were created to maximize ease of use for the Mobile User. Ideally we want to limit the number of actions the mobile user has to conduct to get the end result. To do this we increased internal interactions between the InterfaceSelect and Database.

Use Cases 1, 2 and 6 follow the state pattern since the user initiates input, and the goal is to display output back to the user.

Use Cases 4 and 5 follow the publisher/subscriber pattern since once the weather or traffic data updates it is sent to the database, which is essentially subscribed to this needed information.

OCL Contracts:

Controller
Invariants: Only available to the application.
Precondition: Availability of at least one of the following: Traffic Parser, Weather Parser, Data Aggregator, Map Imaging, Query, User Interface
Postcondition: Extracts available information from concepts listed above through various classes.

Data Aggregator
Invariants: Only available to the application.
Precondition: Availability of data from primary domain concepts.
Postconditon: Combines data from database with live data to create traffic report.

Traffic Parser
Invariants: Accessible by database.
Precondition: Traffic data server is available.
Postcondition: Retrieve and parse traffic data.

WeatherParser
Invariants: Accessible by database.
Precondition: Weather data server is available.
Postcondition: Retrieve and parse weather data.

DatabaseConnection
Invariants: Only accessed by database.
Preconditions: Has access to database.
Postconditons: Connects database with parsers and aggregators.

Query
Invariants: Accessed by controller.
Preconditons: Access to database.
Postconditions: Traffic, weather, and police data will be queried from database.

UserInterface
Invariants: Available to all controller
Precondition: User is using application and inputting data.
Postcondition: Retrieves needed information.

UserReport Service
Invariants: Available to database and UserInterface

Precondition: Available submitted user reports.
Postcondition: Store reports in database.

MapImaging
Invariants: Hidden service
Precondition: Access to Google Maps API
Postcondition: Displays the route on a map.

Class Diagram

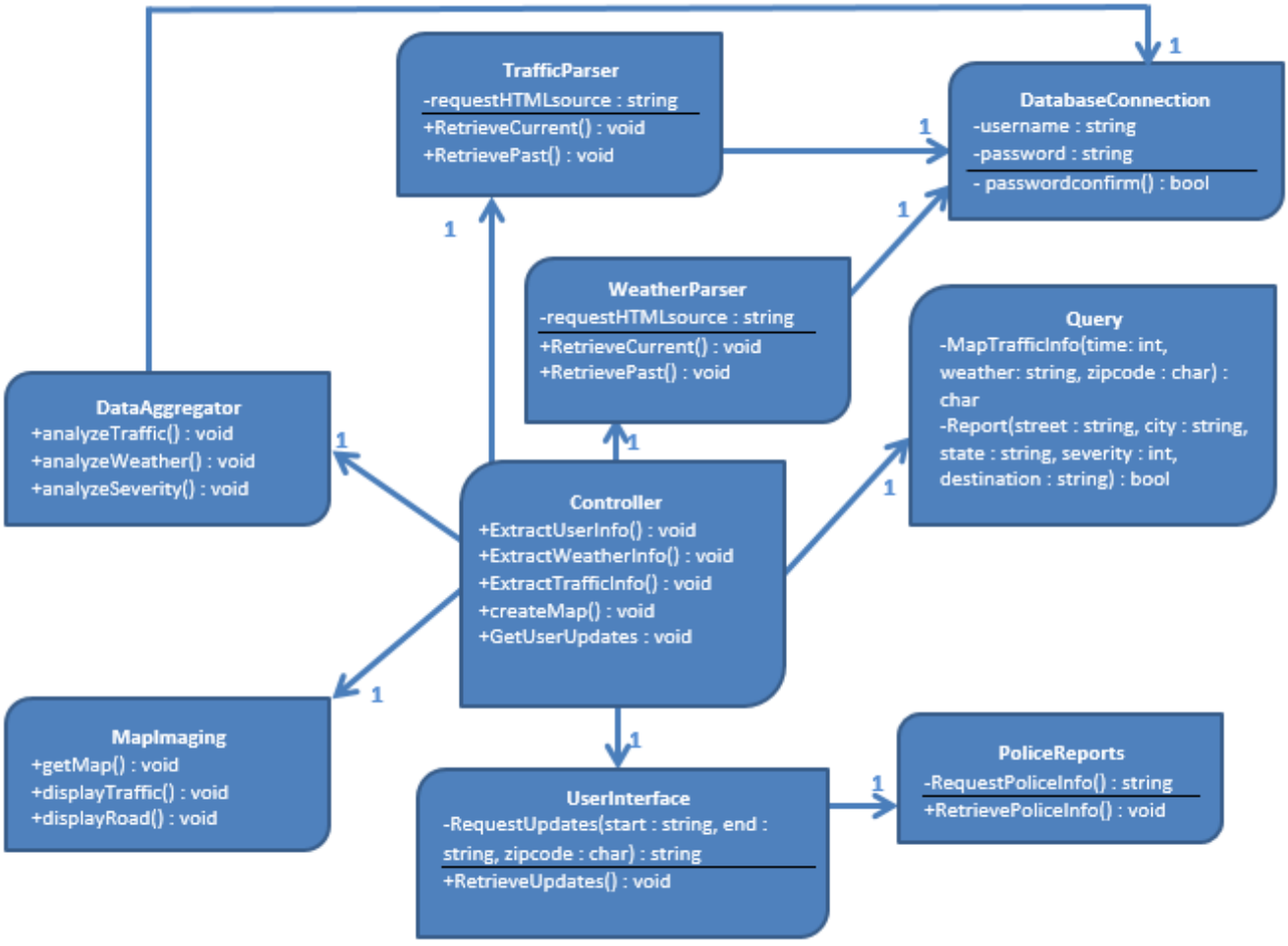


Figure 17: Class Diagram

Data Types and Operation Signatures

Weather Parser
-requestHTMLsource : string -htmltag: string -count: int -list: string -location: string -weather: string -precipitation: string -temperature: double -forecast: string -occurrencepossibility: int *(percentage chance it is going to be certain weather i.e. 10% chance precipitation)* -sockets: http socket
+RetrieveCurrent() : void +RetrievePast() : void

Table 34: Operations Table for Weather Parser

Traffic Parser
-requesthtmlsource : string -htmltag: string -count: int -incidents: string *(amount of car accidents/incidents that happened)* -latitude: double -longitude: double -road: string -sockets: http socket
+retrieveCurrent() : void +retrievePast() : void

Table 35: Operations Table for Traffic Parser

Both parsers retrieve and parse data, one in regards to traffic, and the other for the weather. There are several different attributes and operations, but they seem pretty self-explanatory. Yet, the ones that are more opaque are defined.

Query
-time: int -weather: string -zipcode: char -street: string -city: string -state: string -severity: int -destination: string
-mapTrafficInfo() : char -report() : bool

Table 36: Operations Table for Query

User Interface
-start: string -end: string -zipcode: char
+retrieveUpdates() : void -requestUpdates() : string

Table 37: Operations Table for User Interface

Database Connection
-username : string -password : string
-passwordconfirm() : bool

Table 38: Operations Table for Database Connection

To access the database, the username and password must match.

Map Imaging
-startlocation : string **-endlocation: string** **-routetype: string** ***(i.e. scenic route, fastest route, least use of highways)***
+getMap() : void **+displayTraffic() : void** **+displayRoad() : void**

Table 39: Operations Table for Map Imaging

MapImaging class is used for displaying the route.

Controller
To be determined when implementation begins
+ExtractUserInfo() : void **+ExtractWeatherInfo() : void** **+ExtractTrafficInfo() : void** **+createMap() : void** **+GetUserUpdates : void**

Table 40: Operations Table for Controller

Data Aggregator
-trafficintensity: float **-weather: string** **-time: int**
+analyzeTraffic() : void **+analyzeWeather() : void** **+analyzeSeverity() : void**

Table 41: Operations Table for Data Aggregator

User Submit Reporter
To be determined when implementation begins
+retrieveUserInfo() : void **-requestUserInfo(): string**

Table 42: Operations Table for User Submit Reporter

These are the attributes and operations that are utilized for these various classes.

Traceability Matrix

	Classes								
Domain Concepts	Map Imaging	Data Aggregator	User Interface	DB Connect	Weather Parser	Traffic Parser	User-Submit Reporter	Query	Controller
Application	X	X	X		X	X	X	X	X
Database		X		X	X	X	X	X	
Mapping Service	X	X						X	X
Directions Service		X						X	X
User-Submit Service		X					X	X	X
Weather Service		X			X			X	X
Traffic Service		X				X		X	X
Voice Activation			X						X
Mobile Application			X						X

Table 44: Traceability Matrix relating Classes and Domain Concepts

Description:

The classes for our system are listed across the top of the graph, while the domain concepts are listed along the left. From the domain concepts, we obtained the derived classes. First, the Mobile Application domain concept only maps to the User Interface class because that is the only way the Mobile Application interacts with the user; after that, the mobile application links to the Web Application so it does not associate with any other classes.

The Map Imaging class generates the maps that the user is able to view, therefore it is linked to the Application and the Mapping Service. The Application needs the map so that it can display it, and the Mapping Service has the abilities to create maps.

The Data Aggregator class serves mainly to gather various types of data (such as weather, traffic and user report data), and thus distributes it wherever it sees fit. This means that it will be involved with all of the main domain concepts, as it will usually act as an intermediate step between most processes.

The User Interface links the user to the application and thus only maps to the Application, Voice Activation, and Mobile Application.

Since the Database Connection only acts as a channel to the database, it is solely derived from Database.

For each of the Weather Parser, Traffic Parser, and User-Submit Reporter classes, all three are derived from their respective services in the domain concepts. They also map to databases, because the above classes interact with the database by storing their respective data inside it, and of course, all connect to the main Application, as that is where the data is fed.

Like the Data Aggregator, the Controller and Query both are derived from most of the primary domain concepts. The query class will query the database for data regarding traffic, user report data, directions data, and weather, so it will be derived from those classes.

Finally, the Controller acts as the main hub, delegating different tasks for each module, so it is involved with many domain concepts, including Voice Activation and Mobile Application as it needs to tell the system to link the data received from the voice to the mobile application and then link to the database or web application.

System Architectural Styles

Architectural Styles

This system uses the Client/Server style as its main architectural style. The system is separated into two distinct areas. On the client side of the system, the user interacts with the interface to send the input information and view the output traffic information. On the server side of the system, the system performs functions on data based on input parameters and outputs data that can be used for traffic information. The client side sends input requirements over to the server side, and requests this output information from the server side, and this is the only capacity in which the client side and server side will interact. That way, the data that the client side receives will only be the data necessary for the user in that given instance, and the server side would not have to deal with changing inputs from the client side before they submit their request.

Identifying Subsystems

The first subsystem is the User Interaction subsystem. The user accesses the system through this subsystem on the website or the Android application, and the user will only interact with the system through this subsystem. It contains the Display, the User's input data, the Map output (traffic and directions), the Directions output to accompany the Map output, the traffic report submission for users who want to submit their current traffic information, and the Voice Activation screen. This subsystem is an aggregation of the Database subsystem, as the database only works through the User Interaction subsystem but the database also exists regardless of the current user interaction.

The second subsystem is the Database subsystem, which contains the traffic, weather, mobile, and user report data gathered for the User Interaction subsystem to get and work with for desired user output.

The third subsystem is the Data collection subsystem, which contains the functions that store the data to the database from outside sources like the User Interaction system or relevant web services. This subsystem includes the Weather

Service, the Traffic Service, and the User-Submit Service. It is a composition of the database because these functions only occur with the existence of the database.

UML Package Diagram:

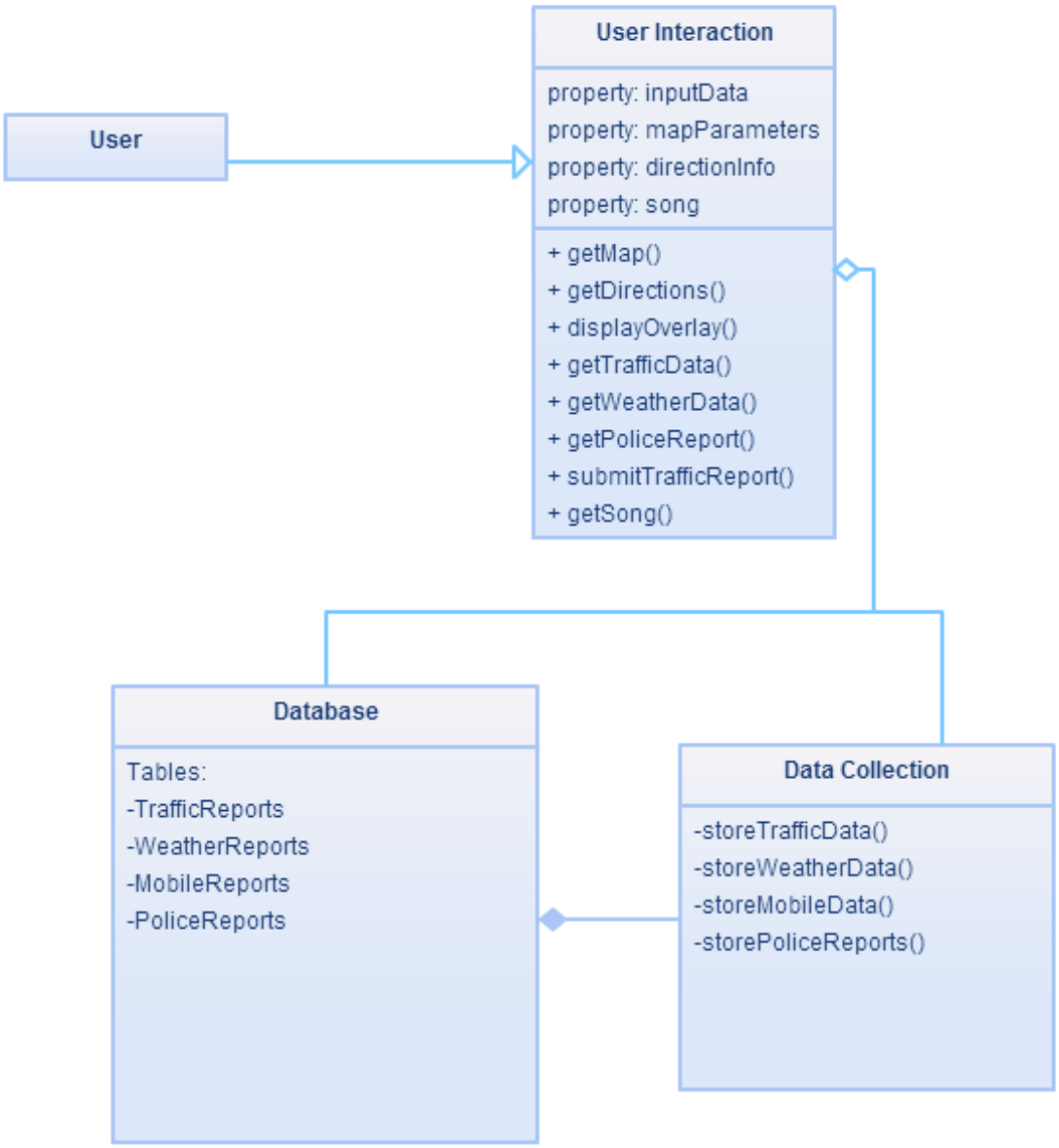


Figure 18: UML Package Diagram

Mapping Subsystems to Hardware

Subsystems do not have to be on the same computer. Our Database subsystem runs on a web server while the User Interaction subsystem runs on the clients' browsers. The Data Collection subsystems run through a web site called CPanel, which holds all the code for allowing the Traffic, Weather, and User-Submit Services to run.

Persistent Data Storage

MYSQL controls the database storage. There are three tables that the database stores: Traffic, Weather, and Police/UserReports. Keep in mind that the Mobile information all links to the Web Application so there aren't separate tables for Mobile data. There is an algorithm implemented in the database that sets a threshold for the user reports that are submitted by the users. In order for an inputted data entry to be outputted on the map, the entry must have occurred in the same location at least as many times as the threshold is set to. This is to ensure accurate data. The database also has another algorithm incorporated into it that deletes all entries after a month to guarantee modern data yet also maintain an efficient "history" of data.

Database Tables

Field	*Type*	*NULL*	*Default*
Traffic			
Create_Date	Date	Yes	NULL
Create_Time	Time	Yes	NULL
Update_Time	Time	Yes	NULL
Update_Date	Date	Yes	NULL
Longitude	Char(15)	Yes	NULL
Latitude	Char(15)	Yes	NULL
Incident_Type	Varchar(255)	Yes	NULL

Road_Name	Varchar(255)	Yes	NULL
Weather			
Create_Date	Date	Yes	NULL
Create_Time	Time	Yes	NULL
Conditions	Varchar(255)	Yes	NULL
Temperature	Char(30)	Yes	NULL
City_Name	Char(45)	Yes	NULL
Zipcode	Char(10)	Yes	NULL
Police/User Report			
Create_Date	Date	No	None
Create_Time	Time	No	None
Incident_Report	Varchar(255)	No	None
Latitude	Char(15)	No	None
Longitude	Char(15)	No	None
Road_Name	Varchar(255)	No	None

Table 45: Database Tables for Traffic, Weather, and Police/User Reports

Network Protocol

The network protocol will be HTTP. The traffic system service will be accessed through the web. The reason we are using HTTP, is because it will allow us to get a high amount of users, since it the most common protocol.

Global Control Flow

Execution Orderness

As a navigation device, our system will always have event driven attributes. An event driven system is centered on performing certain actions in response to user input. This is exactly what our system does; it waits for the user to select the action he wants to do: View Statistics, View User Report, View Police Report, Submit Incident, Start Over, or get the Fastest Route. Our system waits for one of these buttons to be clicked and then performs the appropriate actions after. Even within these actions there are more event driven attributes. For example, View Statistics, the system gives the user a choice of many options to view statistics from, whether it be a region, road, weather condition, etc. The system waits in a loop for events and it is clear that every user can generate actions in a different order.

Time Dependency

There are no time dependencies on our system. Our navigation devices puts no limits on our user, they can enter new destination points at any time or make selections at any time. The only timer is on our database which deletes all entries every 30 days. This is because we want the most accurate and recent data possible, in a reasonable time frame, to provide an efficient history of traffic data.

Concurrency

Our system does use multiple threads. One thread strictly deals with the interactions between the users and the application. Another thread handles the interaction between the application and web services, which includes weather and traffic information.

Hardware Requirements

The user must have Java and Flash installed to access the web application. Our application has been optimized for a 16:9 display ratio and can adjust to the popular screen resolutions.

The Android application only requires 50 MB of space since the database stores all traffic and weather related information. The Android phone must be able to access the Internet and preferably have a GPS chip.

Algorithms and Data Structures

The algorithm for traffic severity relies on Google Map's severity value.

For a more clear and concise view of traffic history, we developed our own algorithm to clean up the traffic data. Essentially, this deletes entries of traffic and weather data in our database that is older than 30 days. This allows for less clutter needed to be displayed on a map and can give the User faster loading times on our application. Our Database contains a table for traffic that contains all the traffic incidents with time and position of the incident. The second table is a weather table, which contains the weather at certain times for all supported Zip-codes. The third table is the User/Police Report table contains all reported incidents from all users. The algorithm takes all traffic incidents of the day and constructs a hash of incidents with position as a key and the severity as the value. The hash is then used to get all incident points along a certain road at the same time with the same weather. The algorithm also creates the same hash from the condensed traffic incident table. The hashes are compared by street. Along that same street traffic incidents of the day are added to the closest point within .5 miles. If there are no points within .5 miles that incident location becomes a new location to be displayed in the condensed traffic incident table. The severity values are updated each time a traffic incident is added to a condensed traffic incident point.

The algorithm decides how severe a condensed traffic incident point is by using the severity value and dividing it by the number of traffic calls the data collection script makes.

The algorithm to determine the route for directions uses the condensed traffic points. We have the start and end points. Using the start and end points, a bounded box

is formed. All points within that box that have high traffic intensities are then found in a list. Using the condensed traffic points we can construct a call to the Directions Service. In that call to Directions Service, we can indicate points between the start and end points that the route should avoid. We weight the points to be avoided based on the condensed traffic point's severity value. No complex data structures are used in this project. The majority of the complexity deals with manipulation of the data stored in the database.

In terms of the algorithm to output User incidents and Police sightings to the map, an algorithm that processes “verified” data points is used. The User/Police Database table contains every single report that has been collected from the “submit incident” page. However, in order for a certain location to be output to the map, it must first pass a “threshold” count to be verified as a legitimate point. If “n” number of other users have reported the same report within a certain radius of that latitude/longitude, then that point is verified and gets output to the map. This “n” can be changed easily and can reflect how popular the app gets used.

User Interface Design & Implementation

For this section, the preliminary design and final design are both displayed to show the change in design over the course of the project. Our final design uses the base structure of our preliminary design, however with many added features included. In Report 1 and 2, there were only some base functions implemented, thus the design was minimal to none. An important design change we made was making the voice activation accessible and easy to use for getting directions and submitting user reports on the Android application. We did this to ensure that the user is being safe and not physically engaging in their phones while driving. We also discarded the music feature because we wanted to focus on more of the functional requirements. Lastly, we created a main screen for the Web Application with many options for the user to choose from. We did this to make it easy for the User to navigate what actions they want to perform and to be able to fully visualize everything our application offers for them. Our Web Application design is written in HTML, CSS, PHP, and some Javascript. The Mobile Application is primarily designed in Java and XML; it also links to the Web Application and uses a mobile view at times. Now that the application has been completed, there are many mockups for the final design to show step by step what the user enters and the corresponding results that are shown after the preliminary design.

Preliminary Design

- Website Interface Specifications

Zipcode:
08902
Weather conditions:
Sun
Temperature:
Current time:
4 P.M.
Start:
End destination:
Submit

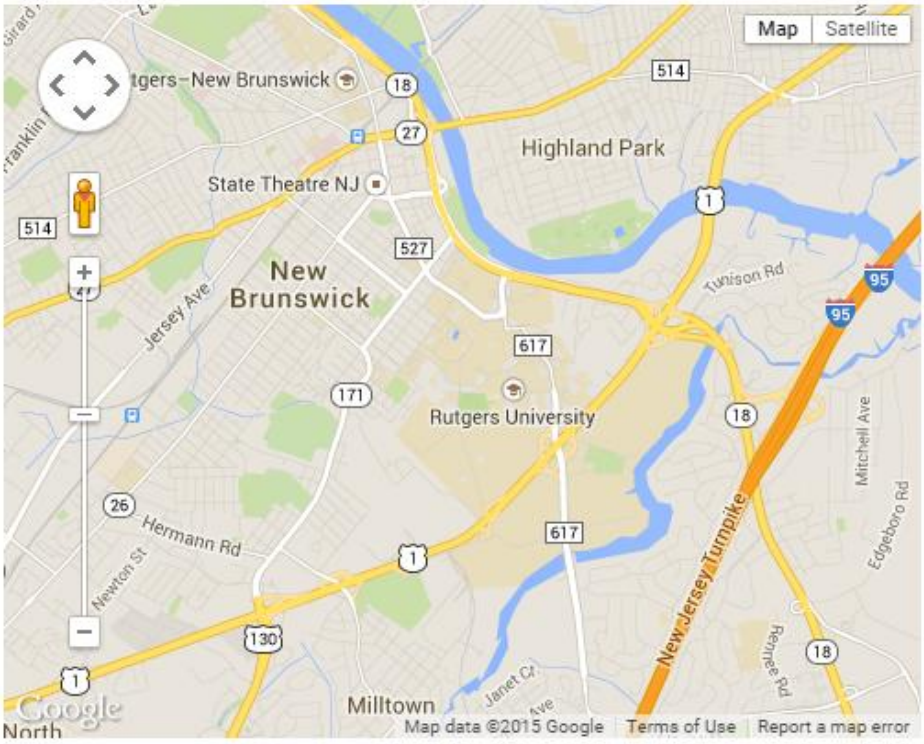


Figure 19: User Interface Specifications for Website

Zipcode:
08902
Weather conditions:
Sun
Temperature:
78
Current time:
4 P.M.
Start:
hill center
End destination:
rutgers student center
Submit

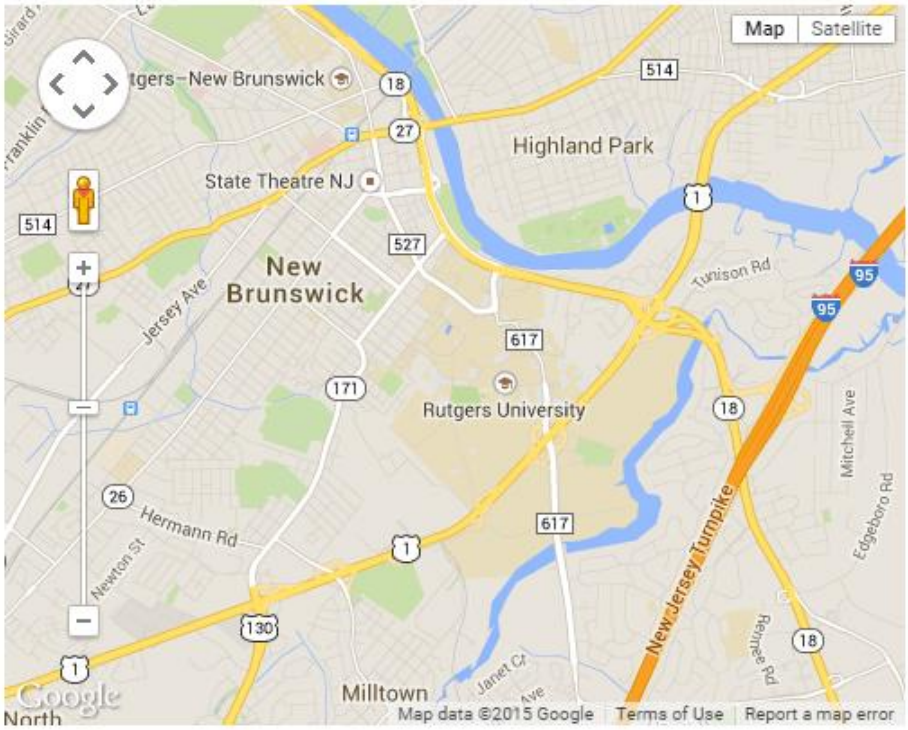


Figure 20: User Interface Specifications for Website Example

❖ Android Interface Specifications

Figure 21: User Interface Specifications for Mobile Application

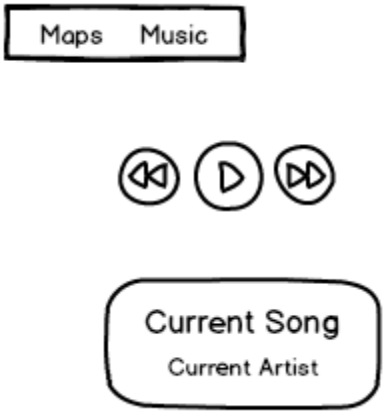


Figure 22: User Interface Specifications with Music Feature for Mobile Application

Final Design with Mockups

Please note: red shows what the user selects, green shows what will result from the users actions.

- Website Interface Specifications: This image is the main website application from where the user can perform many different actions: view statistics, view user report, view police report, submit incident, fastest route, or start over.

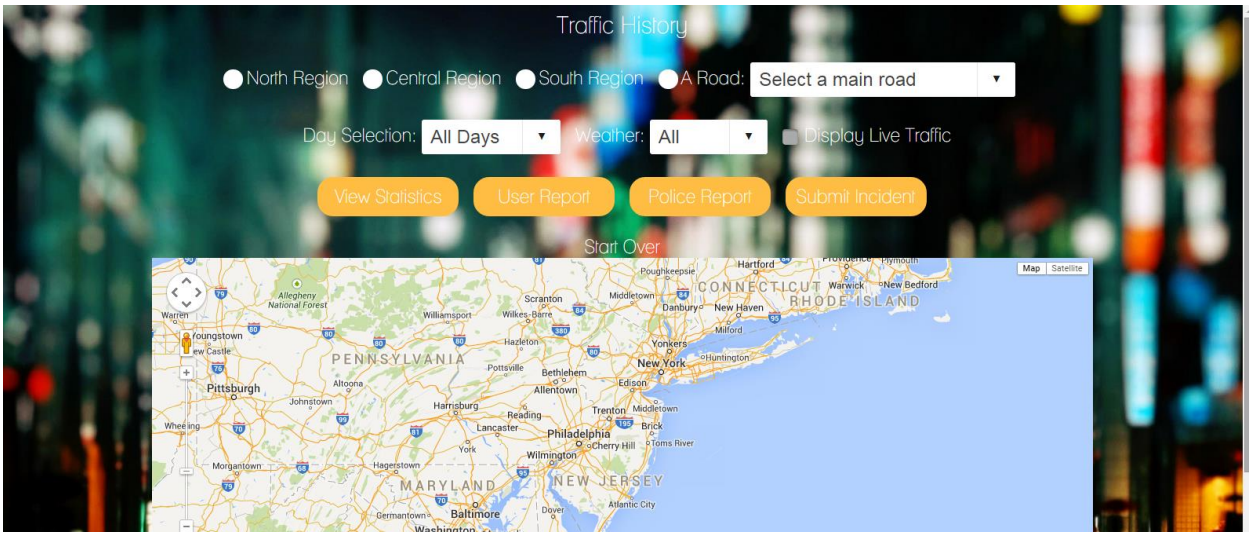


Figure 23: Main Website

1. View Statistics: The User selects a region or a road to view traffic congestion for. The User has the option of selection a day of the week and/or weather conditions. The User can also select Display Live Traffic to view the life traffic alongside the historical traffic congestion. The red boxes show all of these options and the green box shows the result after clicking “View Statistics”.

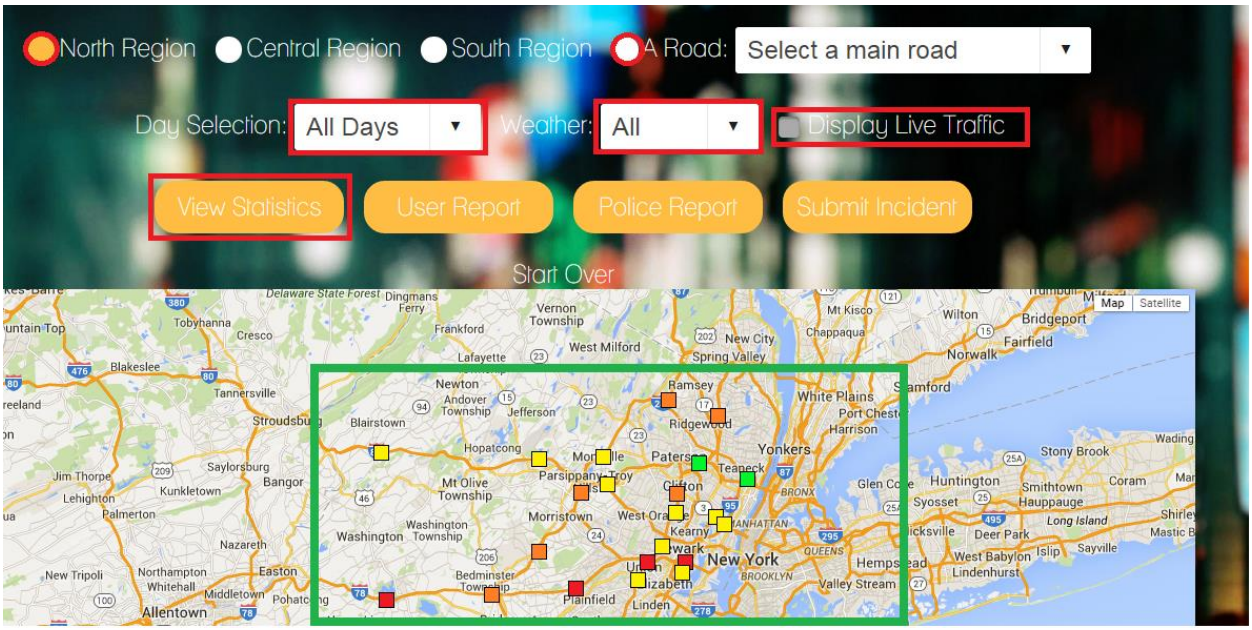


Figure 24: Website View Statistics

2. User Report: The User selects the button “User Report” and then the user reports display on the map. There are different symbols for different types of user reports, rose signifies scenic route, orange cone signifies bad road conditions, etc. The green box shows the result of this action.

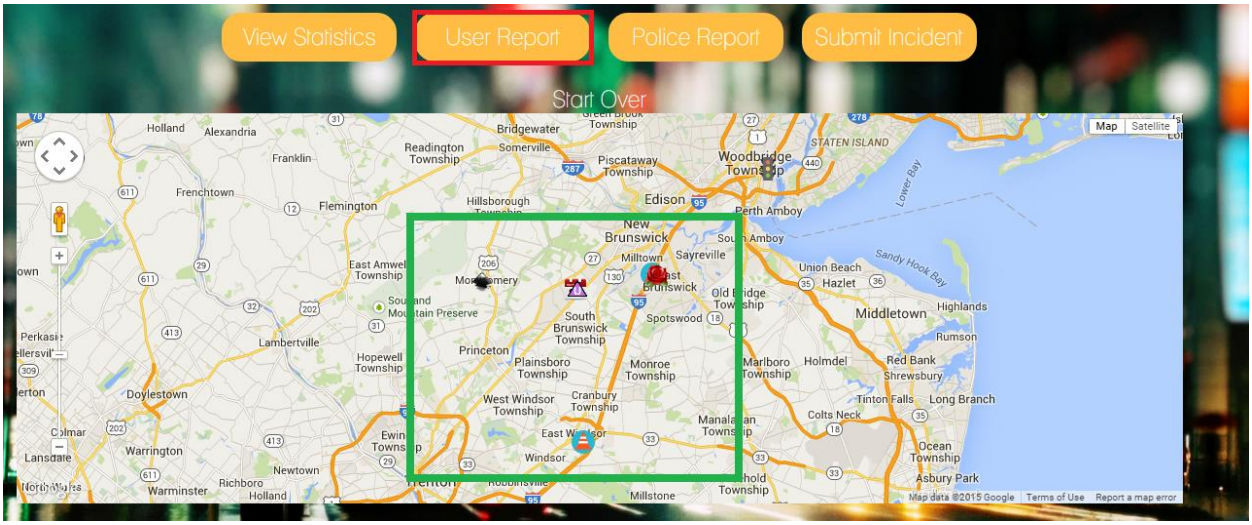


Figure 25: Website Police Report

3. Police Report: The User selects the button “Police Report” and then the police reports display on the map. The green box shows the result of this action.

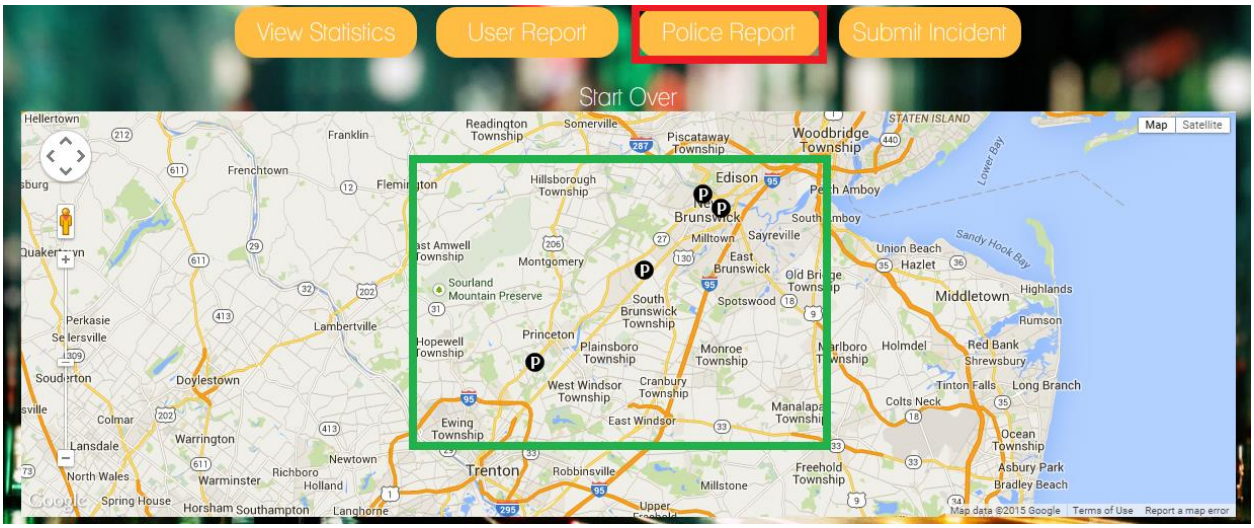


Figure 26: Website Police Report

4. Submit Incident: The user can submit an incident they saw on their route. Incidents can include police, traffic, road conditions, accident, etc. This data is then pushed into the database.

Figure 27: Website Report Options

After clicking the "Submit Incident" button, a new page is loaded. This new page gives the options to submit the incident. The user must click on the location (on the map) where they saw the incident. The longitude and latitude points for this location then appear below the map, these numbers need to be copied into the longitude and latitude boxes below that. The user then selects the type of report from the dropdown. If it is a police report, then the user must select the number of police from the dropdown on the right. The user also has the option of typing in a report type if it is not shown in the dropdown. The user clicks "Submit" to finish this process.

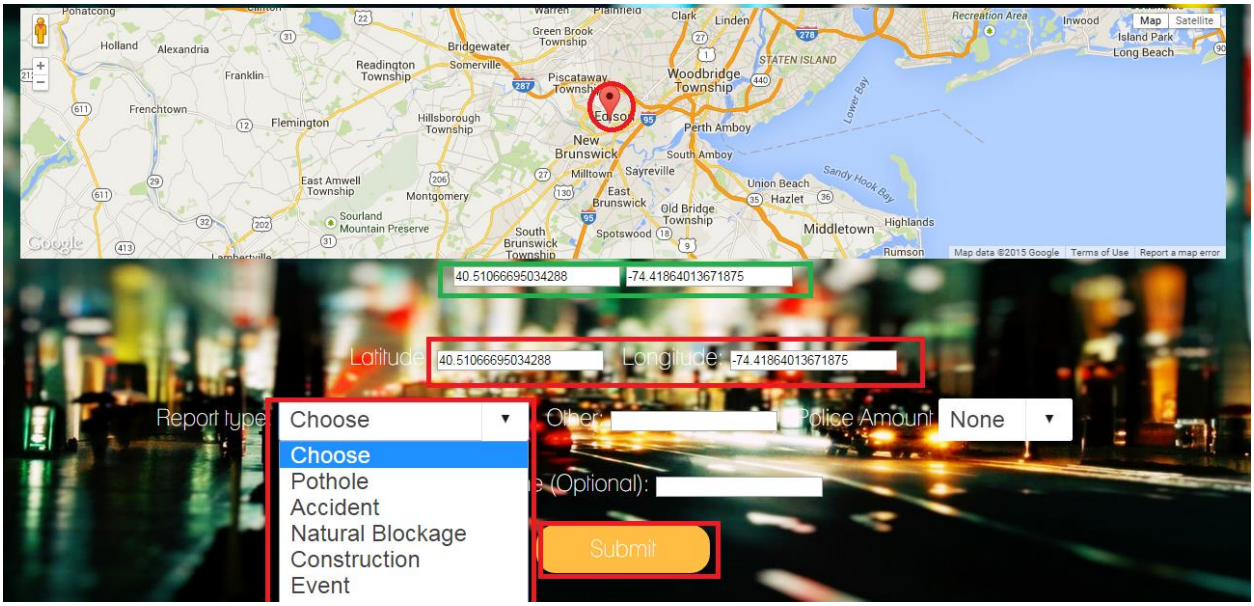


Figure 28: Website Submit Incident

5. Fastest Route: The user can get directions from one point to another by clicking on fastest route. This button is shown near the button of the main page of the web application.

Figure 29: Website Fastest Route

Once the user clicks this, a new page is loaded where the user inputs the start and end destinations for the route they are searching directions for. The User then clicks submit and the directions appear on the right of the screen, as shown in the green box. The map of the route appears on the left of the screen, as shown in the green box. The user can select any of the suggested routes; the map and listed directions will change correspondingly.

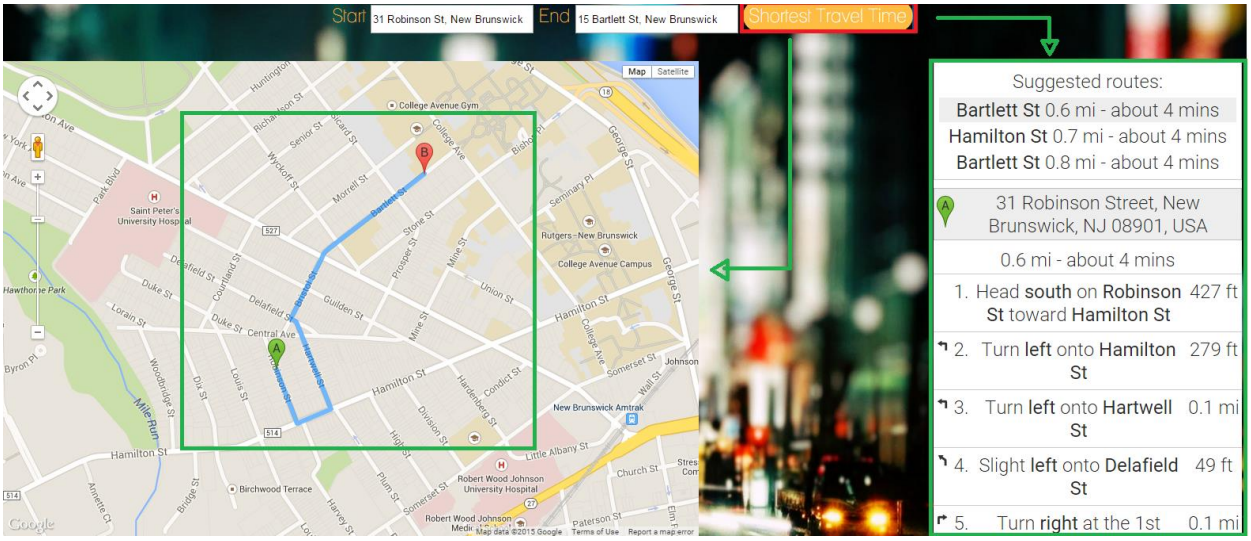


Figure 30: Website Suggested Route

6. Start Over: The User has the option of starting over.

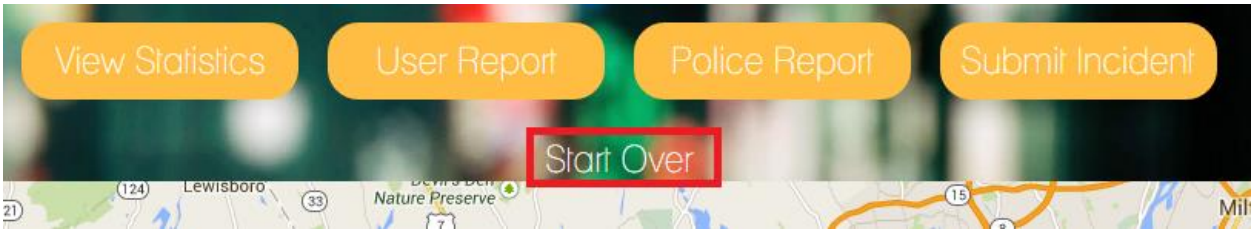


Figure 31: Website Start Over

- Android Interface Specifications: This image is the mobile application screen upon opening the app, from where the user can perform many different actions: Get Traffic Report, Get Directions, Submit Traffic Report.

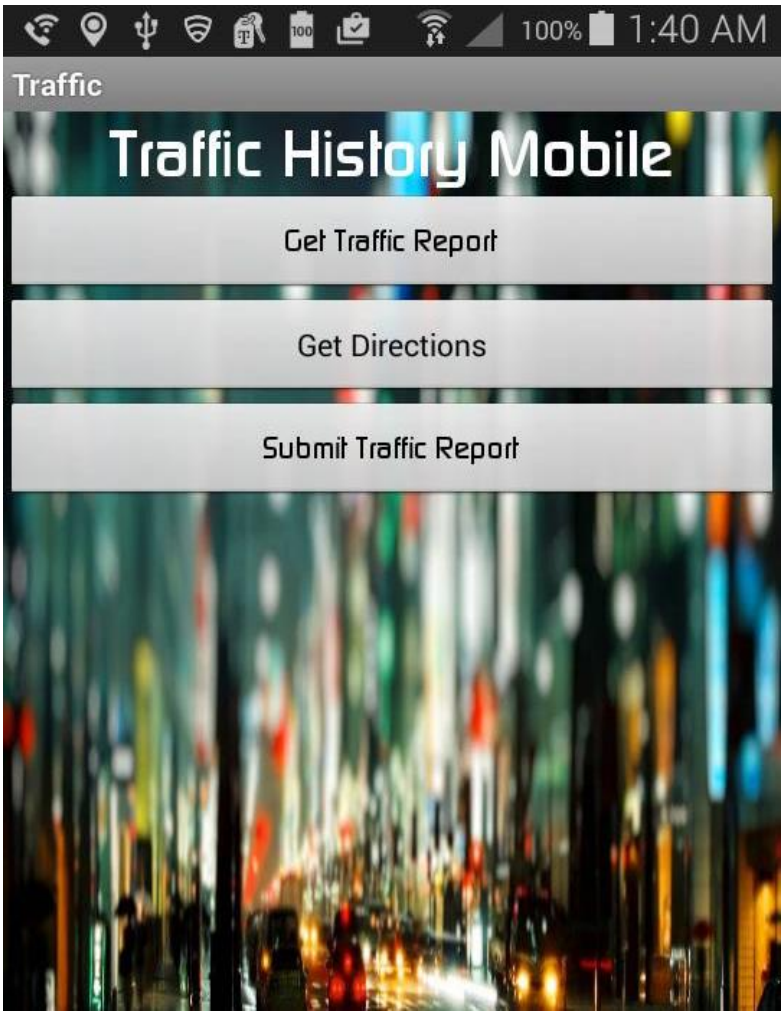


Figure 32: Mobile Interface

1. Get Traffic Report: This button links to the mobile view of the website and performs the same functions. The same user clicks are necessary for this part as for the web application. To avoid redundancy, these will not be shown. The first figure below shows the series of buttons the user could select (red boxes), and the figure after that shows the traffic report result (green box).

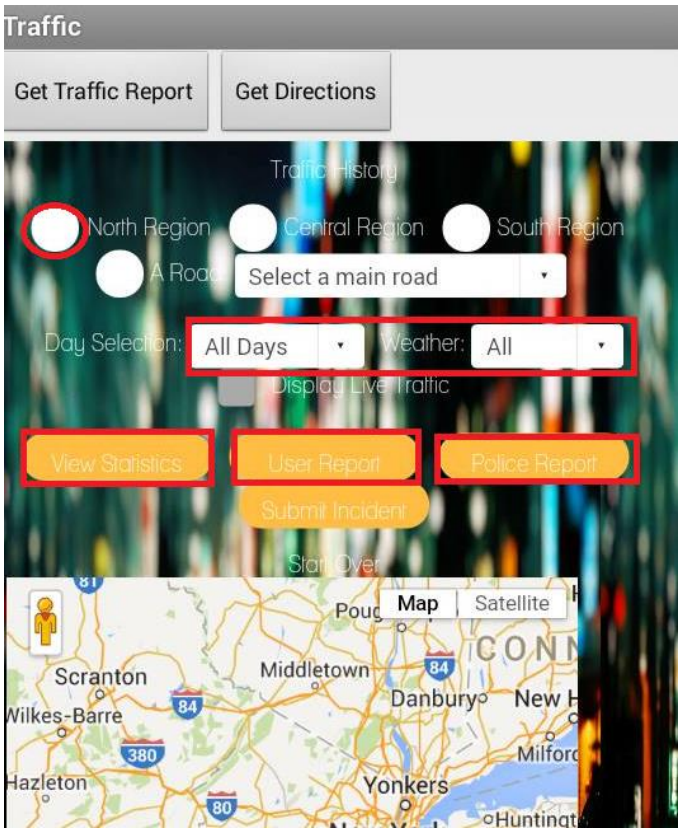


Figure 33: Mobile Main Options

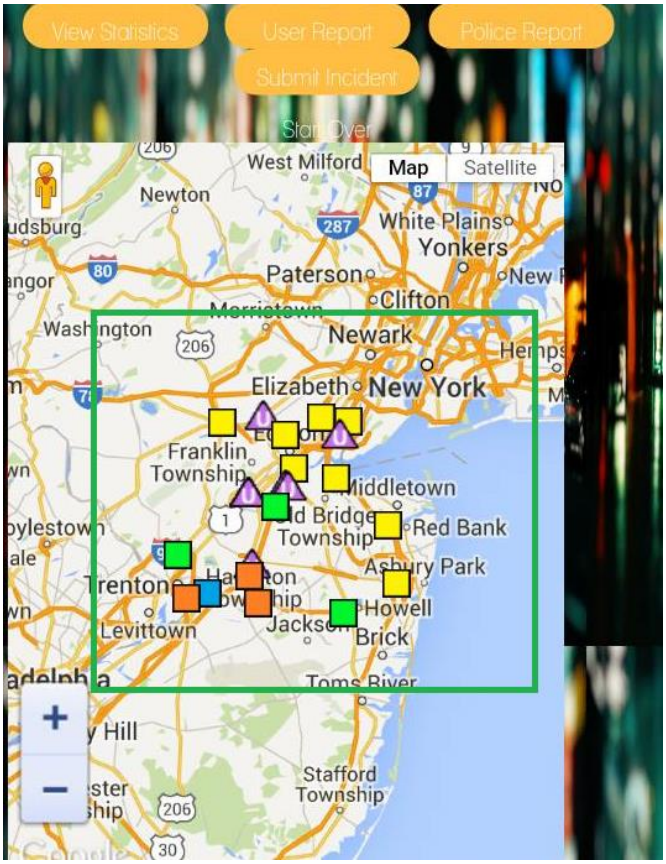


Figure 34: Mobile Get Traffic Report

2. Get Directions: The User can get directions to a certain destination by clicking on the “Get Directions” button in the main screen of the mobile application and the following screen will display. Keep in mind that the application pins the GPS location of the user, so the user does not need to enter a starting location. For the end destination, the User has two options: manually enter in the end destination and then click submit (red boxes), or click the voice activation button.

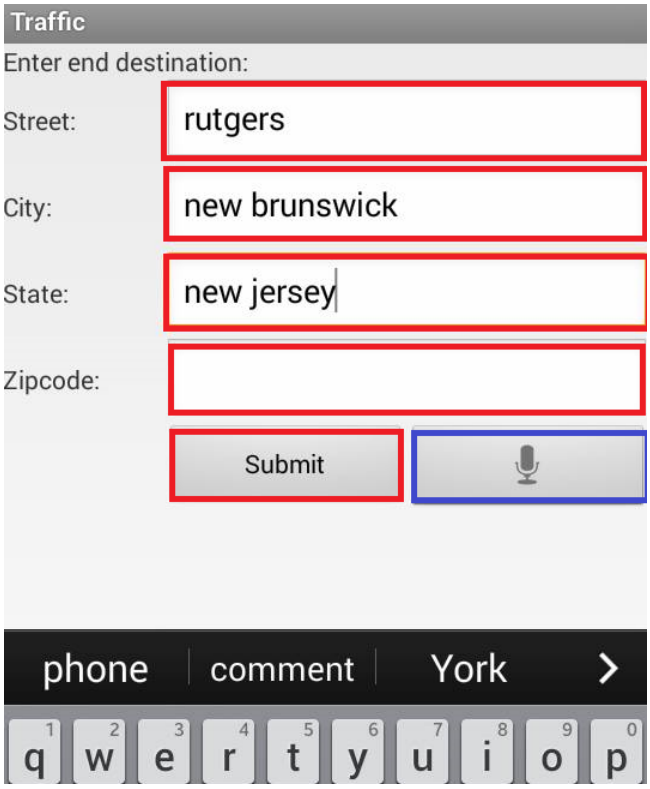


Figure 35: Mobile Get Direction

If the User chooses voice activation, the following screen will occur, prompting the user to speak into the mobile device. The information will then be inputted into the application by the device.

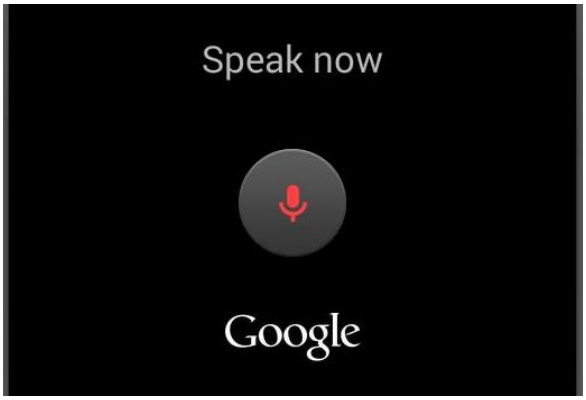


Figure 36: Voice Activation

After either manually or orally entering in the end destination, the application will result in the following screen. (green boxes) (The user must scroll down to see the listed directions). The user can select any of the suggested routes; the map and listed directions will change correspondingly.

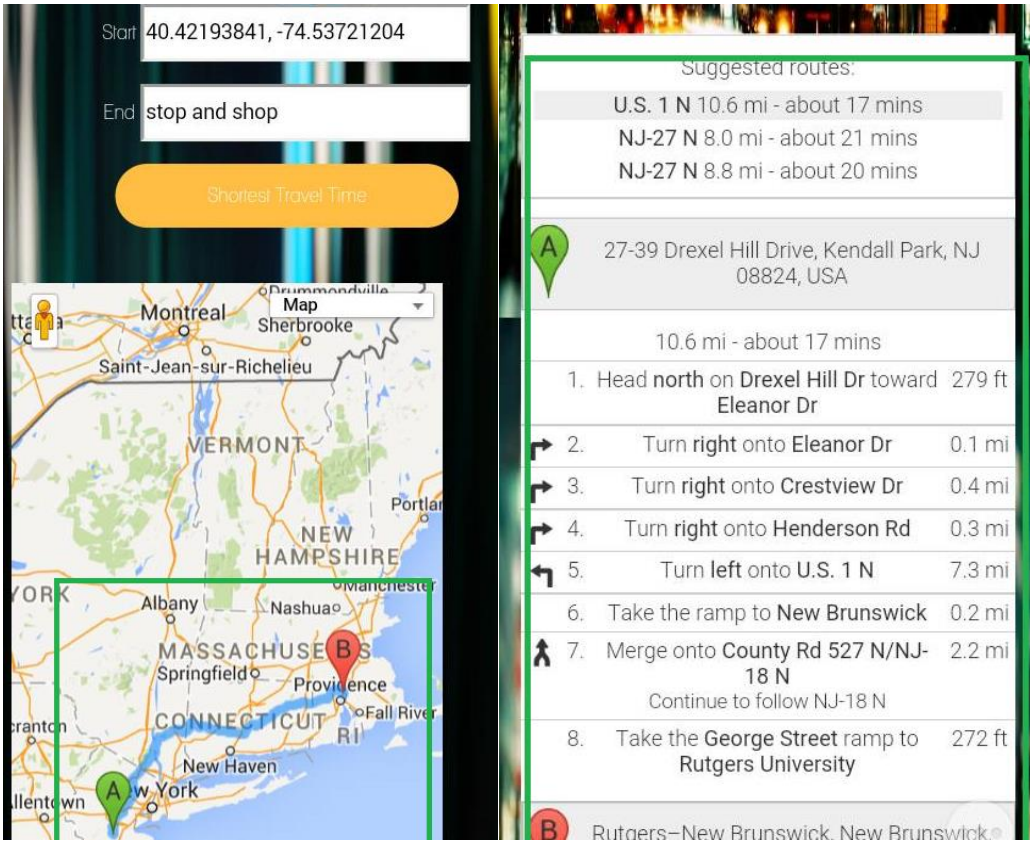


Figure 37: Mobile Suggested Route

3. Submit Traffic Report: The user can input a user report while driving by using the voice commands. The User simply clicks this button and the voice activation screen will appear. The User will engage with the device and submit a user report. The Application pins the location of the User so that the User does not have to enter a location.

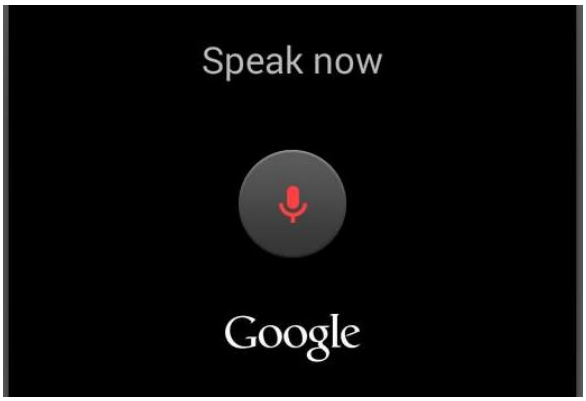


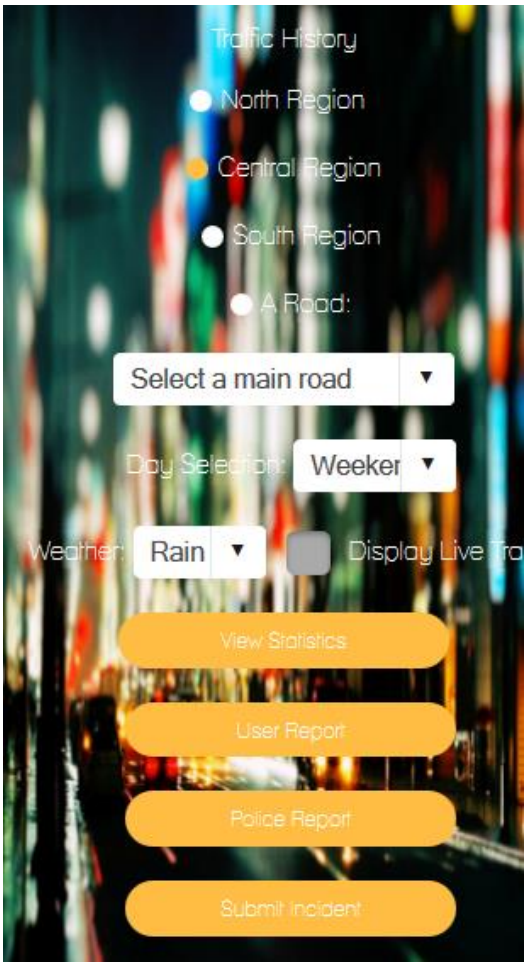
Figure 38: Mobile Report Voice Activation

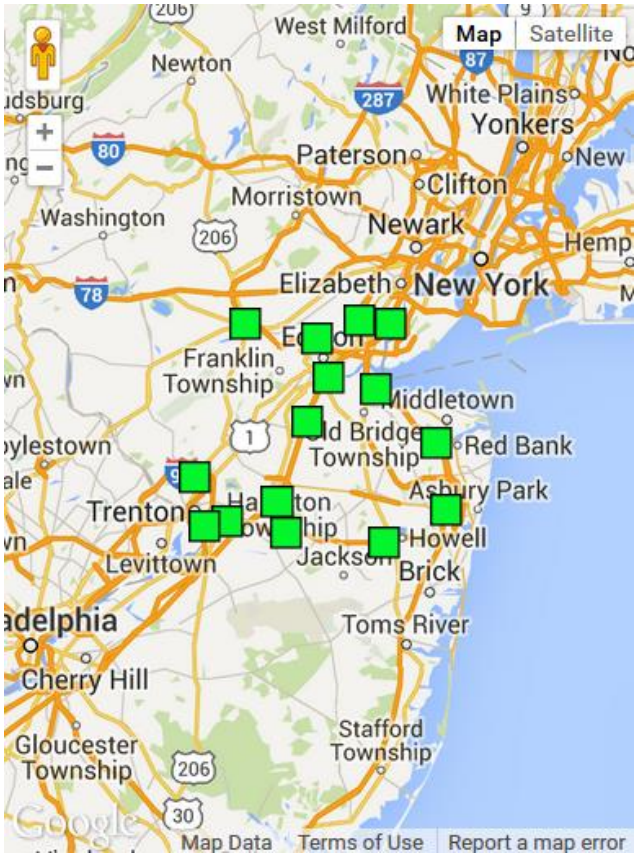
Use Case Testing

Use Case 1 (Get TrafficHistory):

General:
Input: Region, Day Selection, Weather
Device: Web Interface
Ideal Result: Display history of traffic

Test:
Input: Region (*Central Region)*, Day Selection (*Weekend*), Weather (*Rain)* Output:





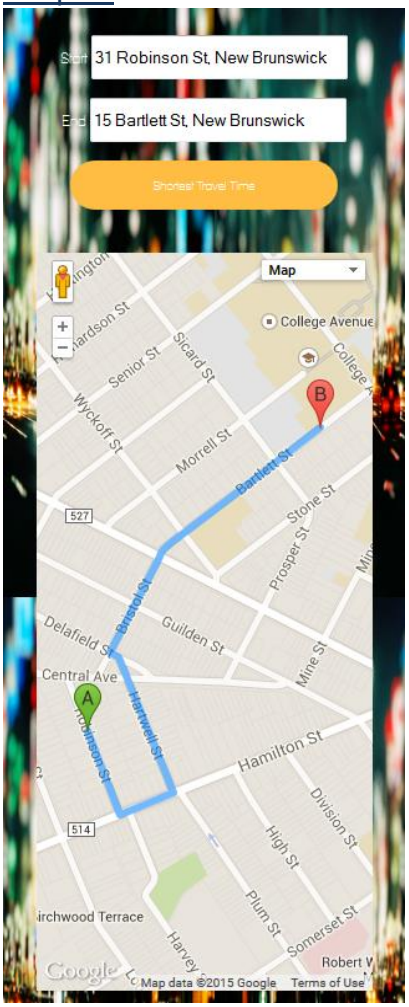
Listed above is a demonstration of Use Case 1 being tested on a web interface. We inputted the required information which was the region, day selection, and weather. Once we inputter this we clicked on View Statistics to see what the traffic is under those conditions. The result we have are logically compatible with the inputted results. It would make sense for there to be less traffic on weekends in Central Jersey due to rainy weather as reflected through the output.

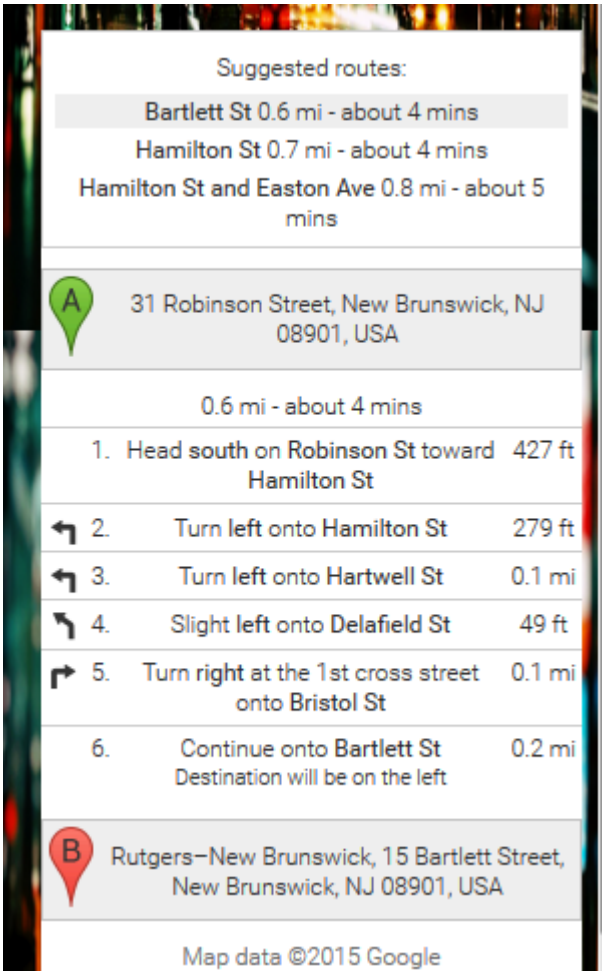
Use Case 2 (GetDirections):

General:
Input: Starting location, destination
Ideal Output: Display suggested route on map

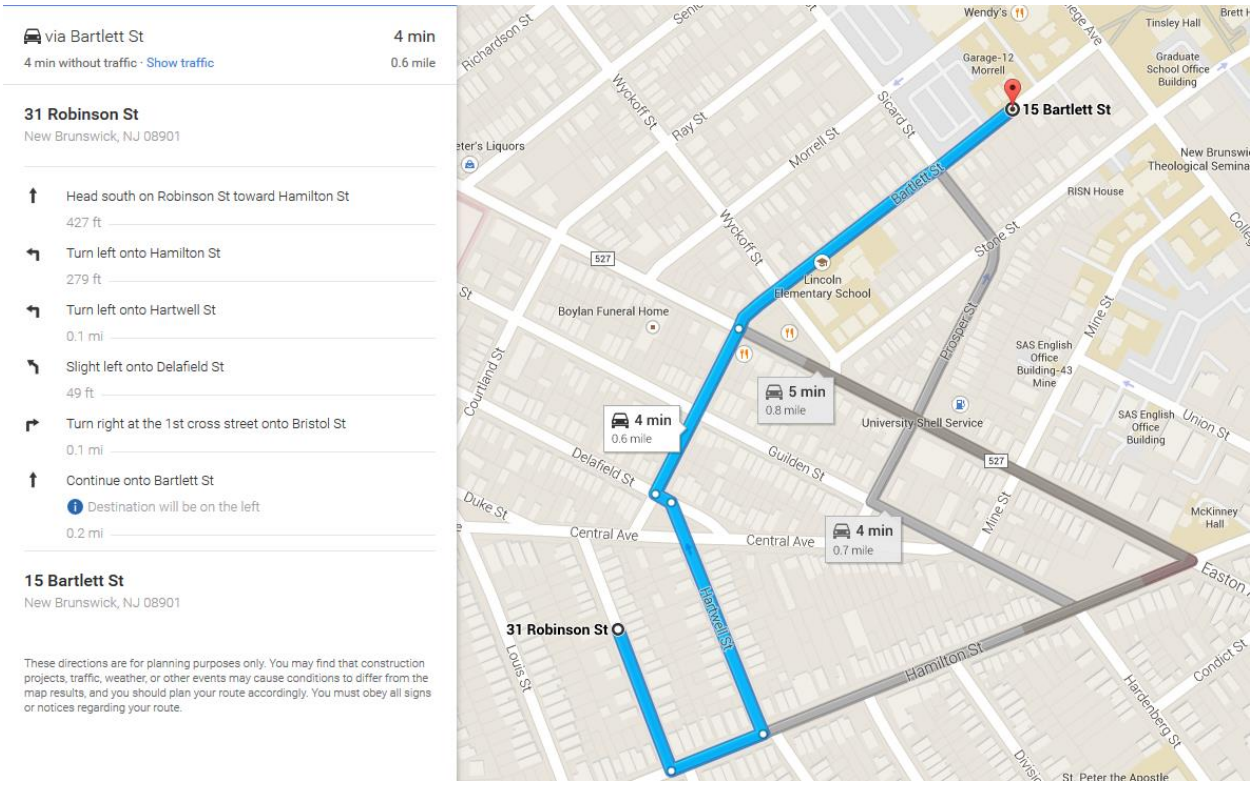
Test:
Input: Starting location (31 Robinson Street), destination (15 Bartlett Street)

Output:





Google Maps Output:



Result:

As you can see for the above test case, the shortest route and directions match each other on the website our group created alongside google maps. This confirms that we get the shortest and correct route and that the actual output matches the expected output. We performed equivalence testing to verify that these different use cases were fulfilled. For example, to test Use Case 2 we logged onto the website http://m.justincoding.com/trafficProject on different platforms to make sure it worked universally. Also, we inputted what was listed above and confirmed it on the website to test that the correct and ideal output had been relayed. We confirmed that the shortest possible route was displayed by looking at the time displayed and compared it to other websites like google maps. The driving time was a little bit longer in some cases, as we accounted for whether the route avoided any severe traffic or incidences that google maps could not account for. Yet, for most of the time the time displayed was equivalent.

Use Case 8 (ReportIncident):

General:

Input: User's traffic, police report
Device: Android smartphone, Laptop
Ideal Result: The values are stored in the database

Test:

Traffic Activity:

Input: Latitude (40.438847170052), Longitude (-74.42790985107),
Report Type (Police Sighting)
Device: Laptop

Input:

Result:

CREATE_DATE	CREATE_TIME	INCIDENT_REPORT	LATITUDE	LONGITUDE	ROAD_NAME
2015-05-02	12:05:00	Traffic	40.086477293808	-75.13755798339	
2015-05-02	11:47:47	Police Sighting	40.438847170052	-74.42790985107	
2015-05-02	12:08:34	Traffic	40.526326510744	-75.44105529785	

Police Activity:

Input: Latitude (40.438847170052), Longitude (-74.42790985107),
Report Type (Police Sighting)
Device: Laptop

Input:

Result:

CREATE_DATE	CREATE_TIME	INCIDENT_REPORT	LATITUDE	LONGITUDE	ROAD_NAME
2015-05-02	11:47:47	Police Sighting	40.438847170052	-74.42790985107	
2015-05-01	00:26:53	Natural Blockage	40.430	-74.670	Sunset Road
2015-05-01	13:47:09	Accident	41.52288	-74.46083492	
2015-05-01	13:32:54	Accident	41.5223178	-74.46015643	
2015-05-01	12:59:15	Police Sighting	41.5230807	-74.46283015	
2015-05-01	11:48:22	Accident	41.52313127	-74.46266056	
2015-05-01	12:58:33	Accident	41.52254274	-74.46329263	
2015-04-30	21:27:22	event	40.42224686	-74.53734483	
2015-04-30	21:30:50	Other	40.4223684	-74.5373464	
2015-04-30	21:50:30	Police Sighting	40.42240076	-74.53741718	
2015-04-30	15:15:07	Natural Blockage	40.430	-74.670	Sunset Road.

Similarly, for Use Case 8 we inputted the traffic report and the police report and confirmed it in the database to make sure it got relayed properly. We performed this on several Android smartphone devices as well as laptops to make sure we performed the proper equivalence testing.

History of Work

Milestones:

Everyone in this group has contributed equally; we ensure this to keep a positive environment and to keep the project moving forward. We all work towards the same format so that when the time comes to create a full report, formatting and consistency are not issues. We work towards the most modern look possible so that the reports do not seem dry and boring. We have successfully achieved a simple compilation of the final copy by heavy communication so that every member is aware of their task and how to implement it.

As with any project, issues do arise. One issue that we dealt with at first was the distribution of the workload, which we quickly fixed so that no one member was doing too much work – we strive for everyone to contribute equally. Another was the scheduling of group meetings to discuss programming implementations and project ideas. As engineers it is very hard to find time for everyone to be available, as everyone has extracurricular activities, jobs, and studying to do. We found that the best way for everyone to interact was through Google Hangouts, Google Docs and GroupMe messaging. These methods of communication and report implementation allow for every member to commit a lot of time during the day, however at their own convenience. We were all able to discuss the report breakdowns, any questions at hand, and write the report whenever we had time during the day. This solution has led to a much better group dynamic and overall improved awareness of the project at hand.

Project Coordination and Progress Report

We are first working on implementing use case 1 (TrafficAccount) and use case 2 (GetDirections) as a whole as they are the basis of this project. We plan to have it completely implemented by the end of Spring Break, as we are working out some kinks with the user interface. We also plan to implement the MapService use case very soon so that we can extract the information from the map API's, something that affects a large portion of the project. Once we can get the User interface to work efficiently, we will include the ReportTraffic and ReportPoliceActivity use cases. In a more project

management perspective, we are a little behind schedule, as the plan of work we had from the last report may have been much tighter than we realized. We are bearing in mind all new dates and deadlines that we have set and are working towards achieving all of them. Currently, we are using a GitHub account that has shared access for each member. We are using this to communally collaborate on code for the project

Key Accomplishments:

1) Subsystem optimization
2) Use Case Implementation
3) Database implementation in SQL
4) Webserver implementation in PHP, Javascript
5) Mobile App Development on Android Systems
6) Google Maps API to receive proper routes
7) Thorough Equivalence Testing
8) Adding several unique features to our system in regards to user reporting
9) User interaction through police, traffic, pothole, etc. reporting
10) Live traffic data combined with historic traffic data for accurate route
11) Even distribution of team work that resulted in fully functioning and accurate product

Future Work:

1) Music Integration:
   a) Integrate various music playing apps such as Spotify, Pandora, and Soundcloud into our system
   b) Develop encompassing algorithm that accounts for various variable such as weather, mood, traffic, etc.
2) Social Media Integration:
   a) Integrate various social media apps such as Yelp, Facebook, Instagram, and Twitter into our system
   b) Develop encompassing algorithm that lets user know when nearby restaurant, friends, or events are occurring

3) Mobile App Refinement:
   a) Make the GUI look sleeker and easier for the user
   b) Decrease the amount of clicks necessary for the driver
4) Scenic Route:
   a) Formulate algorithm to account for the utility gained from happiness due to scenic route

References

511 Traffic Incidents by 511NJ Services:
http://www.511nj.org/

Google Maps by Google:
https://maps.google.com

“Software Engineering” by Ivan Marsic:
http://www.ece.rutgers.edu/~marsic/books/SE/

“The Costs of Highway Congestion” by David Morgan
http://abcnews.go.com/US/story?id=94064

Traffic.com by Nokia HERE:
https://www.here.com/traffic

Voice activation for android application licensed under Apache 2.0 :
http://developer.android.com/reference/android/speech/RecognizerIntent.html

Weather.com RSS Feed by The Weather Channel, LLC weather.com:
http://www.weather.com/