332:452: Software Engineering

Report 2: System Design

Group 7
Aditya Devarakonda
Vamshi Chilukamari
Akhilesh Madalli
Vladimir Samokhin
Sanket Wagle

Project: Traffic Monitoring
URL: https://sites.google.com/site/452trafficmonitor/
3/11/2011

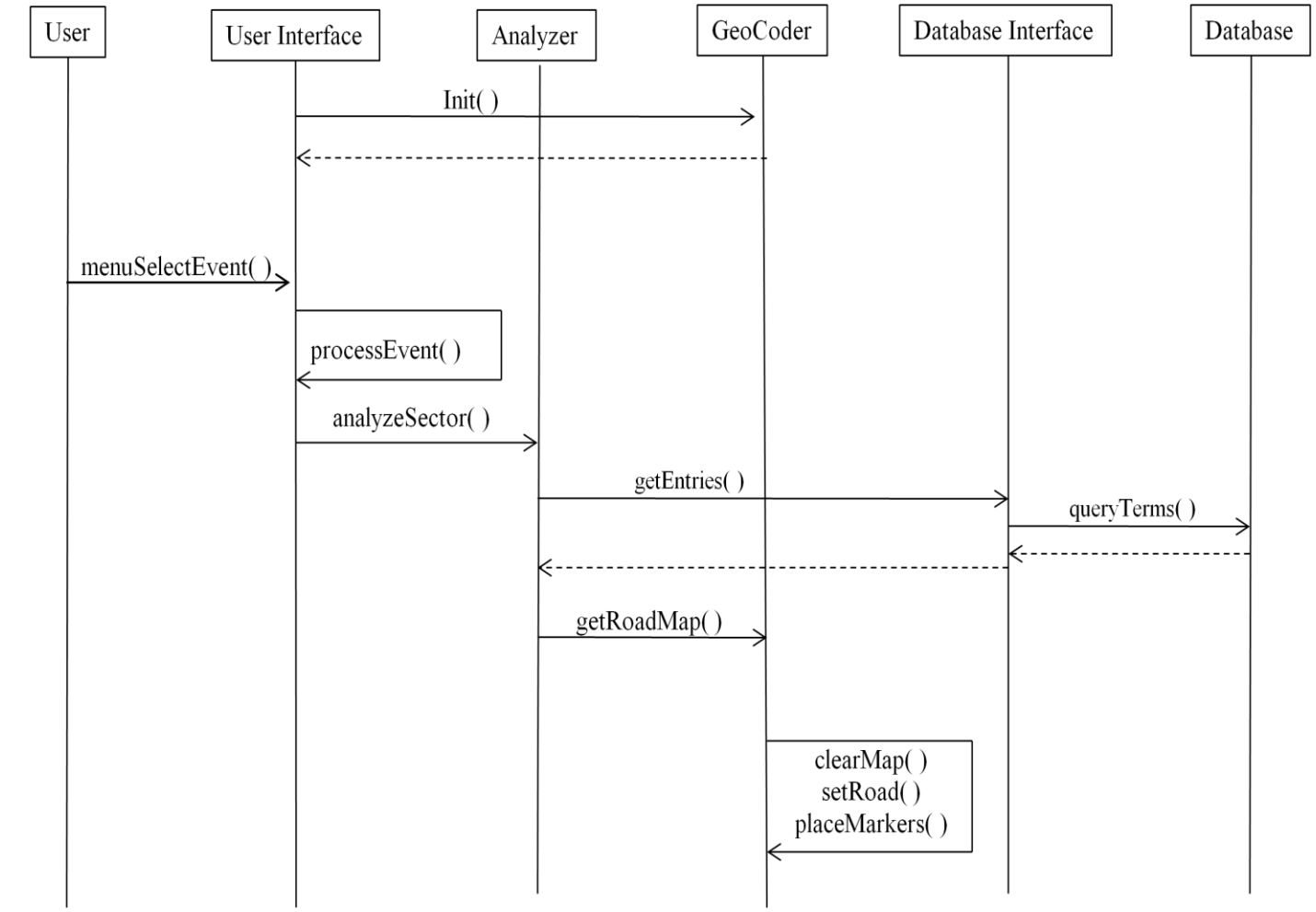
Breakdown of Contributions

All members contributed equally for this report.

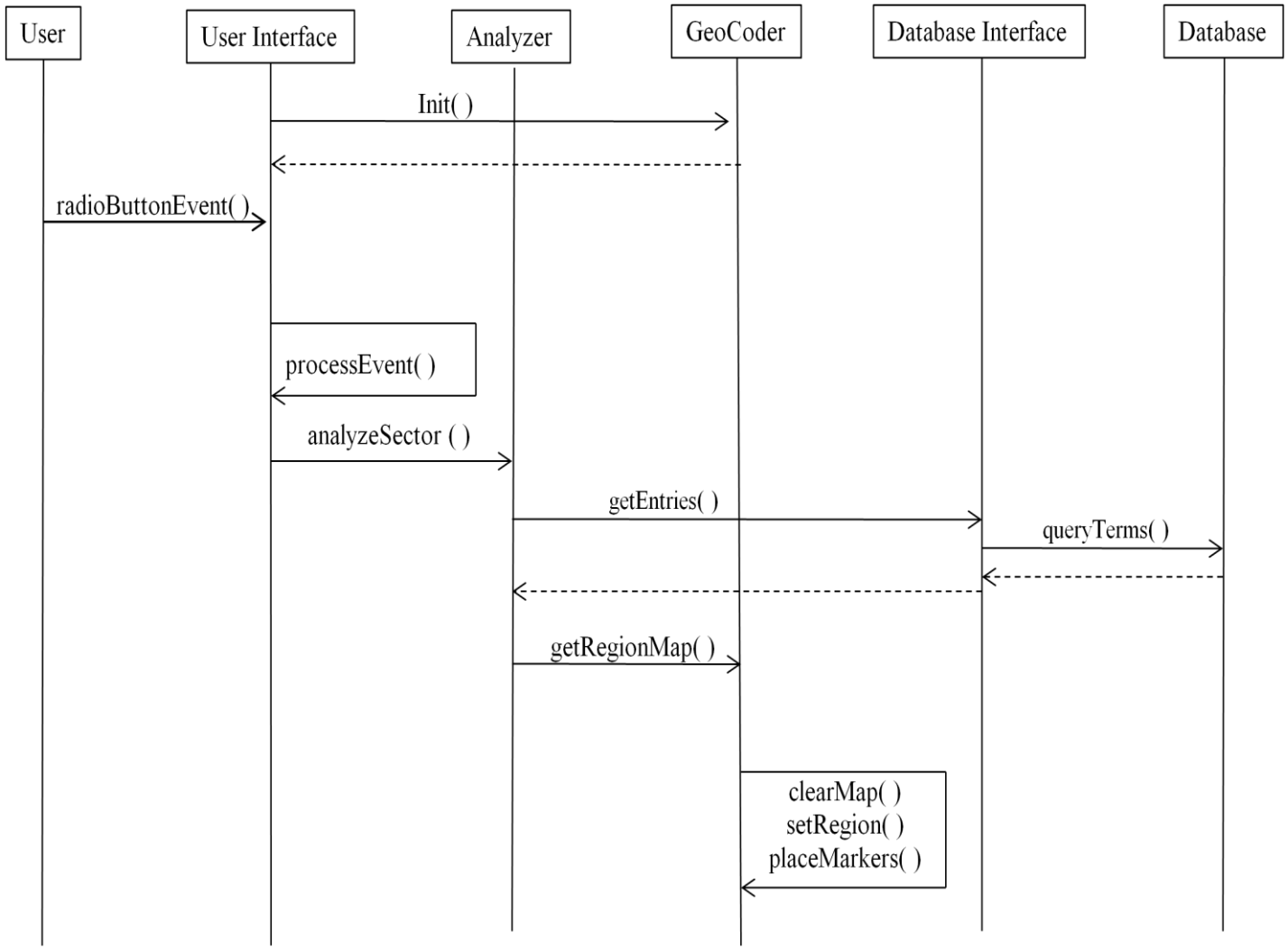
Table of Contents

Interaction Diagrams

View Statistics Along Road



View Statistics Along Region



Interaction Diagram Descriptions:

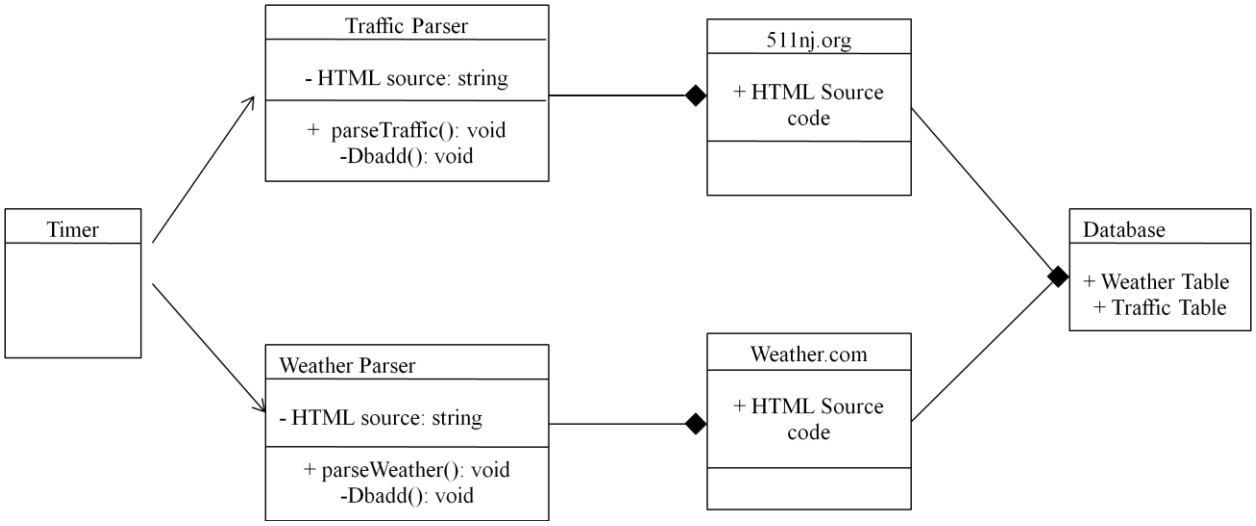
Method	Description
radioButtonEvent()	Called when the user changes the radio button options for the regions (North, Central, and South)
menuSelectEvent()	Called when the user selects a road option from the drop-down menu.
analyzeSector ()	It analyzes the historical data for that sector of the road selected and predicts the traffic pattern for the user based on region and road selected from the button/menu events.
getEntries()	Form the database key terms based on the user inputs and differ control to the database interface class.
queryTerms()	Interfaces with the database and returns the database tables containing the entries matching the user options.
Init()	It initializes Google maps and displays it on user interface.
getRoadMap()	It gets the map of the road user wants to view.
getRegionMap()	It gets the map of the region user wants to view.
clearMap()	It is used to clear the map of all previous markers and roads.
setRoadMap()	Sets the map returned by the geocoder to the road specified by the user.
setRegionMap()	Sets the map returned by the geocoder to the region specified by the user.
placeMarkers()	It places colored markers on the map based on the severity algorithm with appropriate colors.
processEvent()	Takes the users criteria and retrieves the relevant traffic information.

When the web page is first loaded the init() function is called to initialize the map display and create the GUI. The user fills in the form with the criteria for the traffic query and when a region is selected the radioButtonEvent() function is called which displays the region. When a road is selected the menuSelectEvent() function is called. When these functions are called UI processes the input with the the processEvent() function and calls analyzeSector(). The analyzer queries the database using getEntries() for appropriate entries which in turn uses queryTerms() to access the database. Then the analyzer analyzes the data and calls getRoadMap() or getRegionMap() depending on the query. The geocoder then clears the map (clearMap()), sets the view to the road or the region (setRoad() or setRegion()), and places markers(placeMarker()).

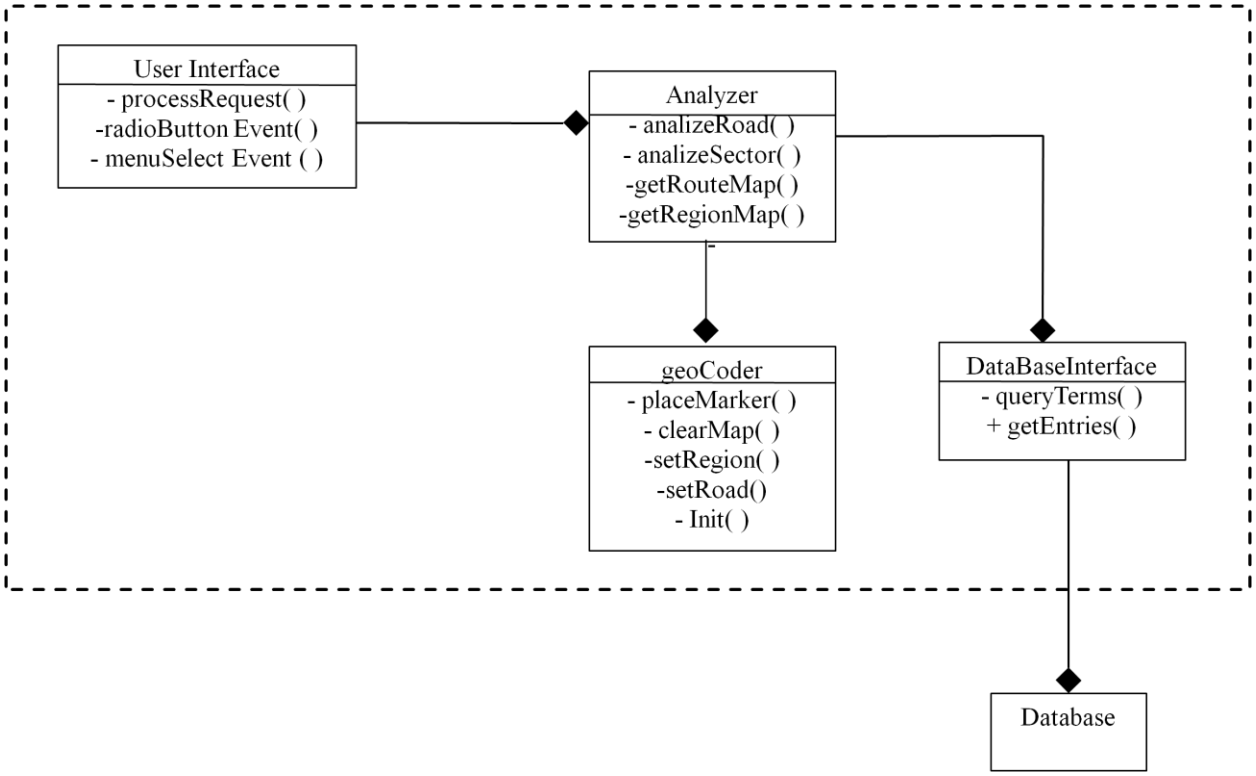
Class Diagrams and Interface Specification

Class Diagrams:

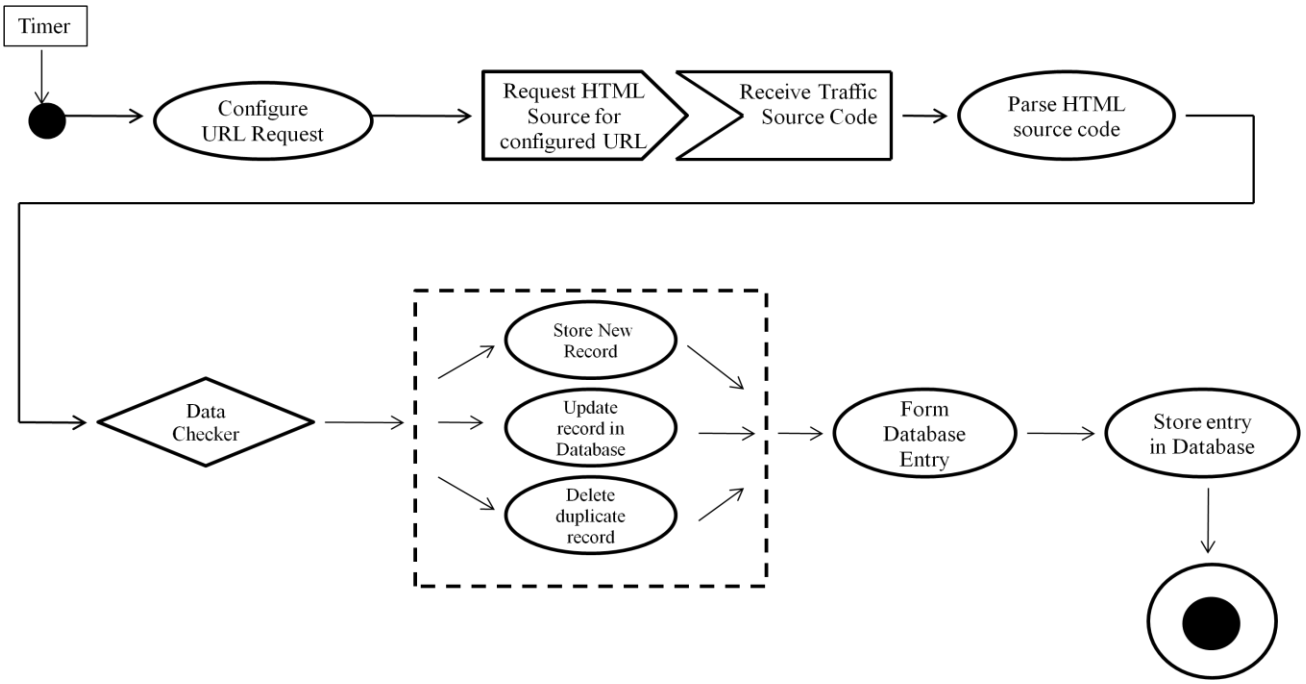
Data Collection System:



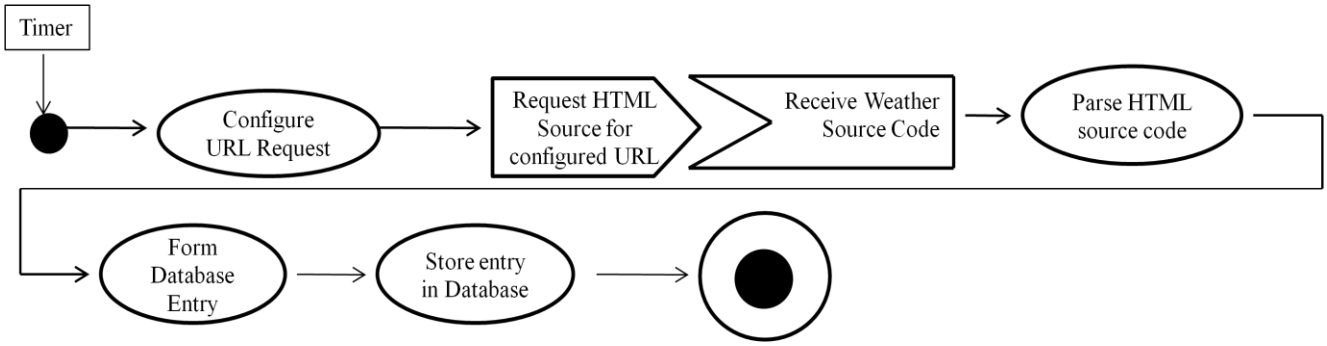
User Interface System:



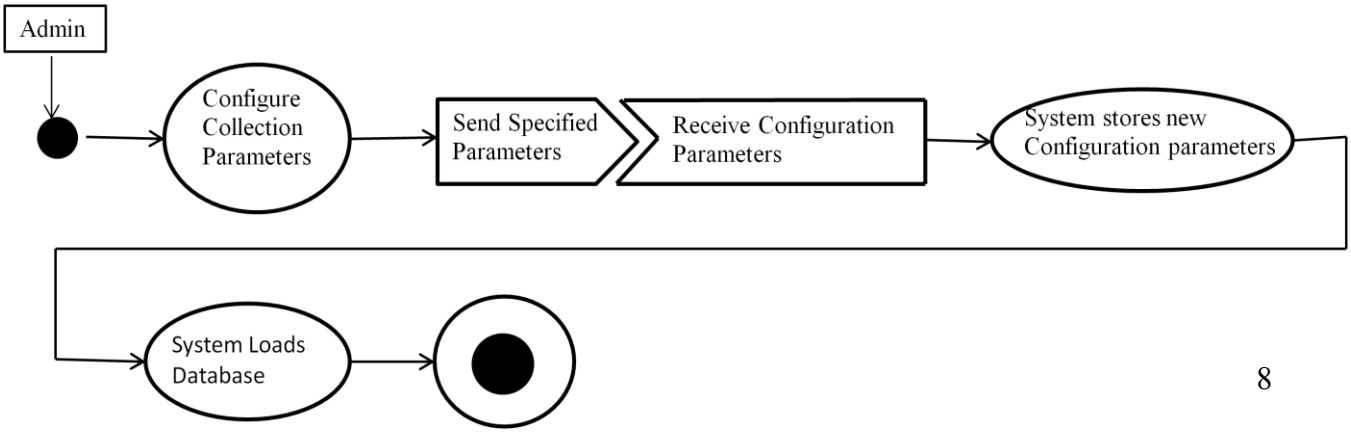
Activity Diagram for Traffic Collection:



Activity Diagram for Weather Collection:



Activity Diagram for Admin Configuration:



Data Types and Operation Signatures

User Interface System

UserInterface
To be determined during implementation
- processRequest() - radioButtonEvent() - menuSelectEvent()

Analyzer
To be determined during implementation
- analyzeRoad() - analyzeSector() - getRouteMap() - getRegionMap()

GeoCoder
To be determined during implementation
- placeMarker() - clearMap() - setRegion() - setRoad() - Init()

DatabaseInteraction
To be determined during implementation
- queryTerms() - getEntries()

As of this moment, we do not have a web server to begin implementation of the user interface portion of the system. However, we have a clear idea of the main methods we need to implement and foresee the above methods as absolutely essential to the user interface, data processing/analysis and database interactions. We will be updating this section frequently once the web server is setup and we have begun implementation of this process.

Data Collection System

Timer
- tParse : Traffic Parser - wParse : Weather Parser

The timer method uses built in function and will not create new function utilized anywhere else. It will contain instances of the traffic parser and weather parser. This design decision was made with the consideration of portability. We can instantiate new Traffic Parsers or Weather Parsers anywhere else without worrying about having a pre-configured hourly data collector. Now, the callee program can instigate the collection based on other factors and events rather than just time.

Traffic Parser
- myhtml : string - htmlTag : string - tagCount : int - incidentList : string - roadName : string - incidentDesc: string - Latitude : float - Longitude : float - sock : http socket
+ parseTraffic() - dbAdd()

Weather Parser
- myhtml : string - htmlTag : string - tagCount : int - parseList : string - cityName : string - condition: string - Temperature : float - precipitationType: string - sock : http socket
+ parseWeather() - dbAdd()

The data collection system is currently functional. If any changes are to be made, the corresponding updates will be made to the tables above to incorporate any additional data types and/or methods. Currently both parsers only retrieve and parse data, if we feel that additional functionality is needed the corresponding changes will be made to these tables.

System Architecture and System Design

Architecture style:

Traffic monitoring system follows client/server software architecture. There is a database server used to save data for traffic along the route and weather. Client communicates with the database server using the web page, which is saved in the database server. Server verifies the user criteria and processes the request to generate result and is displayed on the web page. The architecture has three subsystems: User Interface Subsystem, Collector Subsystem and Database. The User Interface Subsystem is used to display the output to user. The Collector Subsystem is to collect and parse weather and traffic data. The Database Subsystem stores the database tables for the collected traffic and weather data.

User Subsystem has following classes associated with it:

➔ class UserInterface , class Analyzer , class GeoCoder , and class DataInterface.

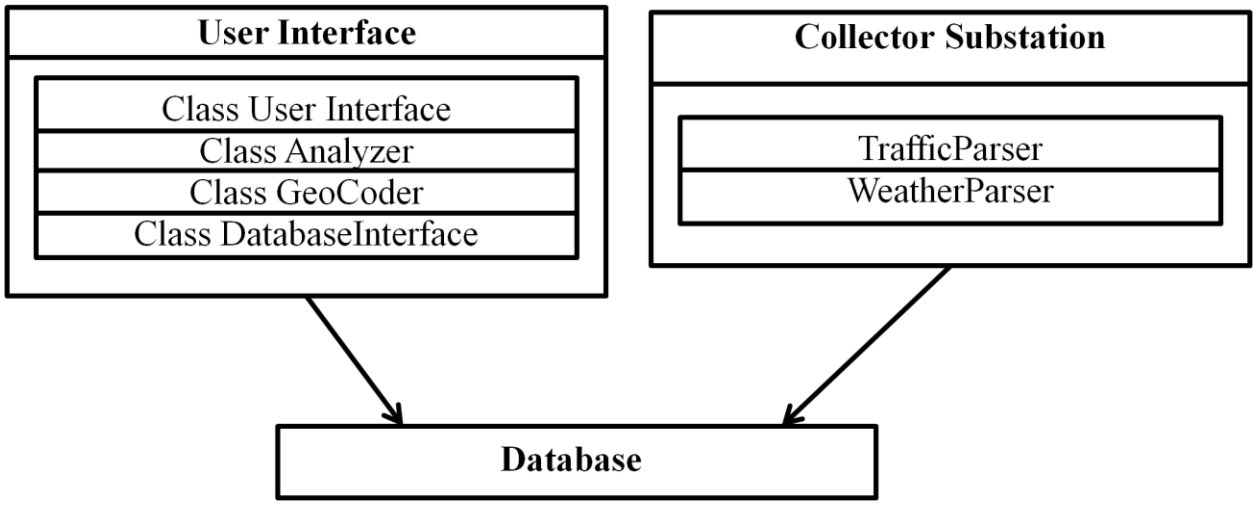
Collector Subsystem class has following classes associated with it:

➔ class TrafficParser , class WeatherParser.

Benefits of Client/Server architecture in Traffic Monitoring System:

- Since data is stored on a server, it offers greater security and there will not be that many errors in the data.
- Traffic monitoring system requires constant updates in weather and traffic data, with client/server architecture, it is easier to access the data on server and update it rather than constantly downloading and processing data on a local user machine.
- Client/server architecture provides ease of maintenance so that client is not affected by server maintenance and upgrade.

Identifying Subsystems:



Mapping Subsystems to Hardware:

The data collected in the "Traffic Monitoring System" will be stored in a database, which will be located in a server. In addition to the database, the graphical user interface will also be stored in the server. The server we are going to be using to store all our data as well as the GUI is a web-hosting service. The client will be able to access the GUI from the server and run the "Traffic Monitoring System" on the server. The client opens a TCP/UDP socket to access and communicate the server.

Persistent Data Storage:

The following database schemas correspond to the stored database tables for the collected weather data and traffic data.

Traffic Database Schema

Field Name	Type	NULL	Default
Create_Time	timestamp	YES	0000-00-00(date) 00:00:00(time)
Latitude	decimal(0,0)	NO	0.000000
Longitude	decimal(0,0)	NO	0.000000
Road Name	varchar(255)	NO	----------

Weather Database Schema

Field Name	Type	NULL	Default
Create_Time	timestamp	YES	0000-00-00(date) 00:00:00(time)
City Name	varchar(255)	NO	----------
Temperature	decimal(5,2)	NO	0.00000
Condition	varchar(255)	NO	----------

The design decision to implement a database instead of a flat file arose due to performance concerns. We will potentially have megabytes of data and the database lookup will be computed much faster than parsing a file. We feel that once we collect significant amounts of data, the decision to use a database will be justified.

Network Protocol:

In Order for the "Traffic Monitoring System" to interact with the server and the client a HTTP protocol must be used. The same protocol is used to access "Google Maps" for the map images, "511nj.org" to collect the traffic information, and "Weather.com" to collected the weather information. An IPC socket connection is opened so that the server can interact with database. We are using an HTTP protocol because it is the most commonly used protocol and it is a standard protocol in any server as well as browser.

Global Control Flow:

- Execution Order: The execution of the Traffic Monitoring System begins with some sort of an event, in the form of a user selecting options from the standard GUI. The User Interface PHP class will react to the user options and begin calling respective function in Analyzer and GeoCoder. So in our architecture User Interface is event-driven and everything else is procedure-driven. Both of the parsers (Traffic Parser and Weather Parser) is procedure-driven. It might seem like event-driven because of the Timer, but the Timer itself is written by us and uses system calls to sleep and call the parsers.

- Time Dependency:The Traffic Monitoring System is a real-time program in that it does the computations and processing when the user selects options to view statistics along a route or statistics in a region. Naturally, there will be a delay between the user request and the final processed display however we are very early in the implementation stage and will measure these statistics when appropriate. The two Parsers, which primarily collect and parse weather and traffic data are event-response. They respond to the Timer and collect and parse data when woken up. We have initially set the delay time to 1 hour but this can be re-configured easily buy modifying the Timer sleep time.

- Concurrency: We do not intend to use multiple threads in the User Interface or the Data Collection systems. We assume single threads for each system.

Hardware Requirements:

- Traffic monitoring system requires user to have a minimum bandwidth of 56 Kbps
- Screen resolution of 800 x 600
- We recommend user to have Microsoft Internet Explorer 7.0 and higher, Firefox 3.6 onwards, safari 3.1 and later, Google chrome because we are using Google maps and it supports only these web browser.
- 2 GB of free disc space per month for storing historical data
- Server provides following services:

PHP, MYSQL, Python with standard libraries and Apache HTTP server.

Algorithms and Data Structures

Algorithms

The most important algorithm we intend to implement involves the calculations of average traffic, severity and other metrics such as delay times, if supported. The current traffic website (511nj.org) uses very generic descriptions and simply states whether an incident has occurred or not. Based on these generic descriptions we need to develop a systematic way of deriving severity and average number of incidents statistics.

1) Average Incidents Statistics: The SQL database table for traffic will contain information (latitude, longitude and highway) which will differentiate the incidents based on the sectors of the highway they occur on. We count the number of incidents which occur within one sector of the highway by querying the database with the conditions, lat = sector_lat and long = sector_long and road_name = highway being queried. Once the matching results are gathered, we count the length of the array returned and then divide this by the total number of incidents on the highway in question. The resulting answer represents the average number of incidents which occurred within a particular sector of the highway. However this does not take into account cases where a highway has negligible traffic. If such a case is encountered then the averaging will be skewed. To remedy this we introduce time into the averaging. This way as more days go by without incidents the lower the calculated probability of encountering traffic. This model will now incorporate the number of days since the data collection started. Our final statistical averaging model will be formulated as follows:

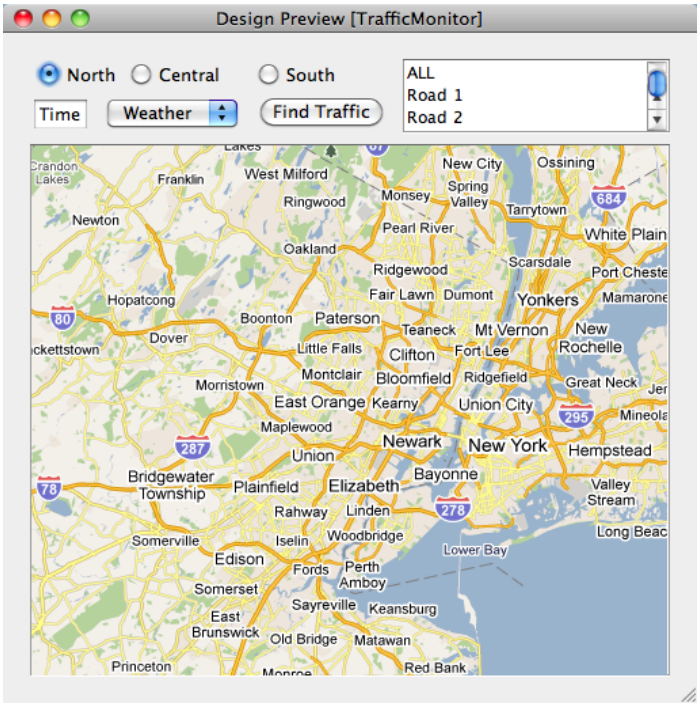
$$\frac{Number\ of\ incidents\ in\ sector}{Number\ of\ incidents\ on\ highway * total\ days\ of\ data\ collection}$$

   This gives a true estimate of the traffic and takes into account the days when there is no traffic, in order to preserve accuracy.
2) Severity: The severity of various sectors of the highway will be displayed by placing colored markers for each sector of the highway. Determining which color is used requires an effect algorithm which chooses color accurately and deterministically, meaning there is no ambiguity in determining a severity. We intend to implement a scale with cut off for color. However, this depends heavily on the data collected and the average incident statistics (above). So we intend to address this issue once we have enough data and have seen the outputs of the average incidents statistics for a set of data.

Data Structures

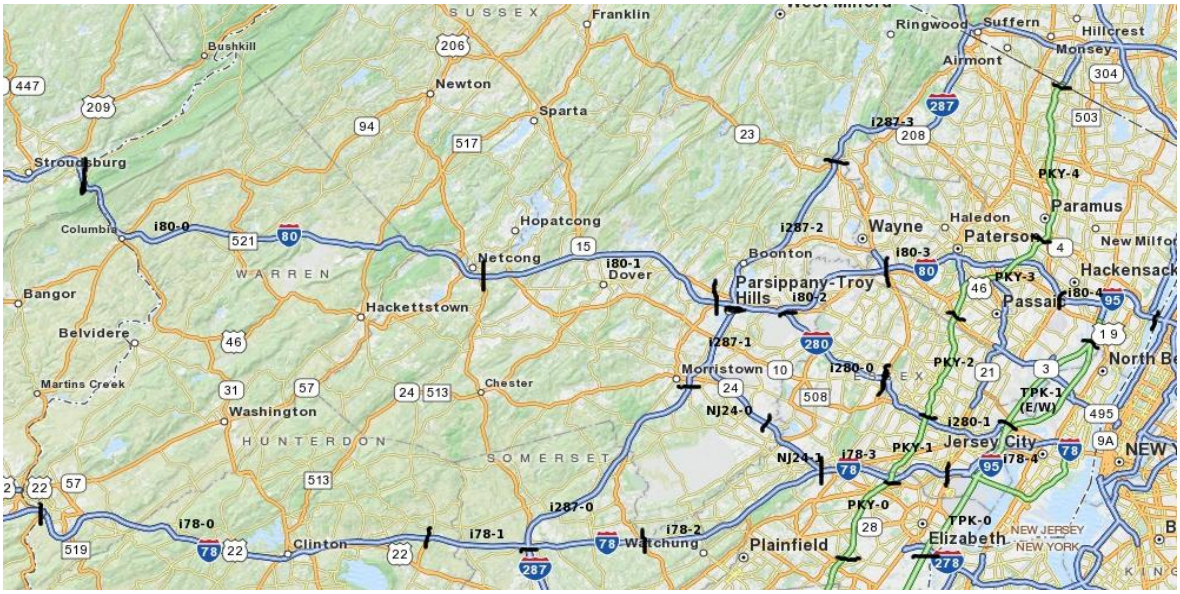
The main data structure in our Traffic Monitoring system is the database. The database will hold all the collected weather data and all the collected traffic data. The contents of the database are outlined in the database schema. We will be implementing lookup tables (Multi-dimensional arrays) in order to lookup the sectors for each highway. This way we can match incidents to their corresponding sectors. We do not anticipate any other major data structures (trees, linked list etc.)

User Interface Design and Implementation

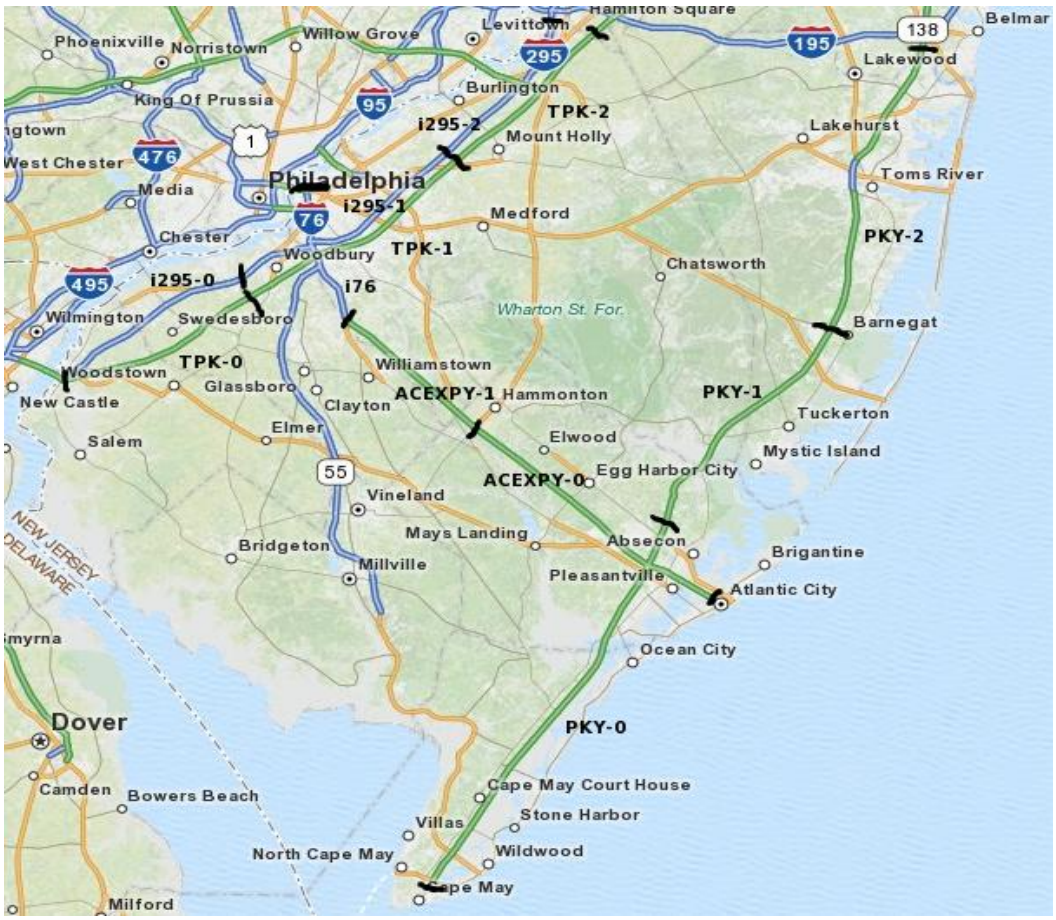


The user will select a region, time, weather, and optionally a road. Then the website will display a map plotting markers in sectors of the roads. These sectors represent a segment of the road that will have similar traffic conditions throughout. In rural areas the sectors are larger and they are small in urban areas. The map will display a marker of a certain color depending on the traffic prediction for that sector. The user interface has not changed visually since the last report. However, now it will be implemented in PHP and not Java or JavaScript because Google maps does not support Java.

The north, central, and south regions of NJ with pre-determined road partitions.




Clinton
Watchung
Plainfield
Elizabeth
NEW JERSEY
NEW YORK
Brookl
Staten Island
RICHMOND
KINGS
PKY-3
TPK-3
i287-1
Somerville
i287-0
SOMERSET
Edison
Perth Amboy
RARITAN
BAY
Flemington
New Brunswick
TPK-2
PKY-2
Sayreville
North Brunswick
East Brunswick
Keansburg
Matawan
Strathmore
Highlands
Spotswood
TPK-1
Lambertville
Princeton
Jamesburg
PKY-1
Red Bank
Little Silver
Long Branc
MONMOUTH
Eatontown
i295-1
Lawrenceville
Ewing
Hightstown
Twin Rivers
Freehold
Newtown
Morrisville
TRENTON
i195-2
TPK-0
Asbury Park
PKY-0
Belmar
i195-1
i195-0
Levittown
i295-0
Spring Lake Heigh
Manasquan


Hamilton Square
Belmar
Phoenixville
Norristown
Willow Grove
Levittown
Lakewood
King Of Prussia
Burlington
TPK-2
i295-2
Mount Holly
Lakehurst
West Chester
Philadelphia
Toms River
Media
i295-1
Medford
PKY-2
Chester
Woodbury
TPK-1
Chatsworth
i295-0
i76
Wharton St. For.
Wilmington
Swedesboro
Barnegat
Woodstown
TPK-0
Williamstown
Glassboro
ACEXPY-1
Hammonton
PKY-1
New Castle
Clayton
Tuckerton
Elmer
Salem
Elwood
Mystic Island
Egg Harbor City
ACEXPY-0
Vineland
Mays Landing
Absecon
Brigantine
Bridgeton
Millville
Pleasantville
Atlantic City
NEW JERSEY
DELAWARE
Ocean City
Dover
PKY-0
Camden
Bowers Beach
Cape May Court House
Villas
Stone Harbor
North Cape May
Wildwood
Cape May
Milford

User effort estimation:

The effort put in by the user to get all the information required, including mouse clicks and keystrokes. (***Using an example time period of 9:00A.M.***)

- **NAVIGATION**: Total of two mouse clicks, as follows
  - Open preferred web browser to load project website

  ---*After completing data entries as shown below*---
  - Click *"Show Traffic"* to display the traffic in the selected region.

- **DATA ENTRY:** Total of 5 mouse clicks and 6 keystrokes
  - Click the radio button to select region to travel in.
  - Press the *"Tab"* key to move to the text field (*"Time"*).
  - Press the keys *"9", "0", "0", "A", "M"* (Enter time in a 12-hour format without the colon for nine in the morning).
  - Click the drop down menu named *"Weather"* and select a weather condition by clicking on a choice.
  - Click on the preferred highway OR click on ALL to select all roads.
  - Click the radio button to display which map the user wants to see.

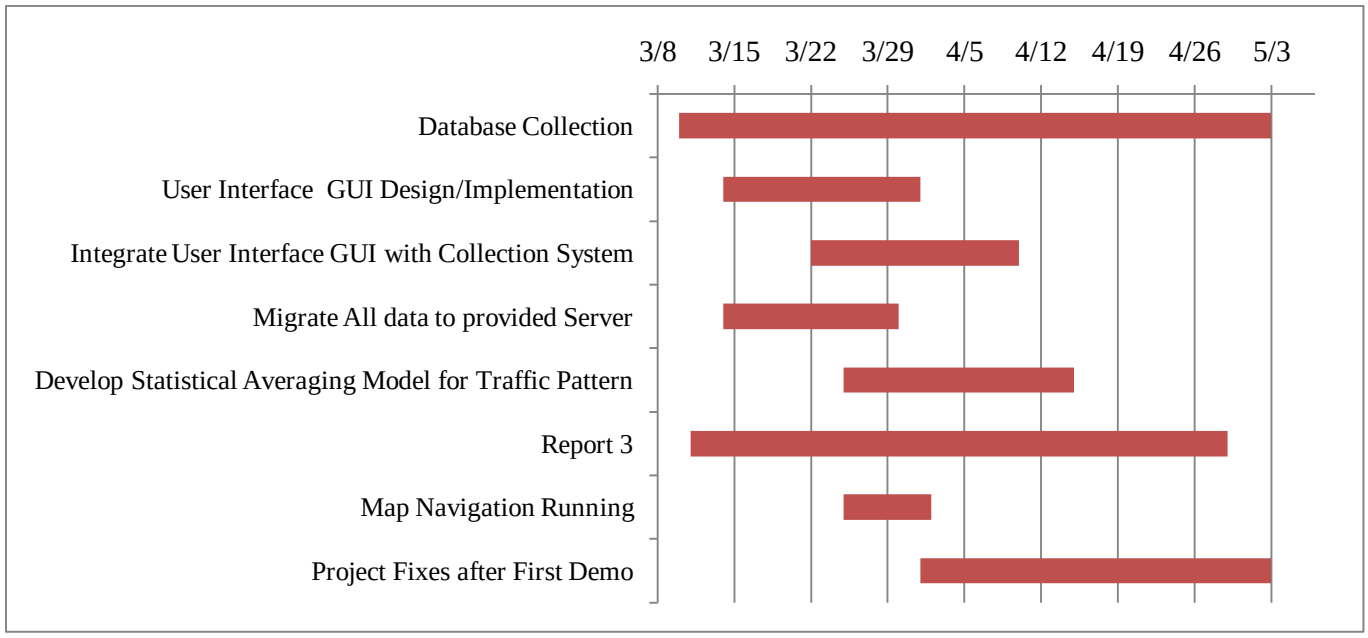
Note: The same procedure was used as described in Report 1. The reason for the same procedure is because we are awaiting approval of the ECE server to implement our website.

Progress Report and Plan of Work

Progress Report:

- All the use cases, Traffic and Weather Collection, are functional. They have been through an implementation test period and are functioning as per planned.
- The group is awaiting the approval of an ECE server. All the use case(s) implementation and data collection will be set-up once all information has been migrated to the provided server.
- The group has decided consensually to use *php* to program/code all necessary database collection methods. We are all simultaneously working on tutorials on learning this language as well as programming the collection methods.

Plan of work:



Breakdown of Responsibilites:

- All group members will perform testing of the integrated system.
- All group members will coordinate the integration.
- At the present moment:
    - Aditya Devarakonda has been working on the database interface, traffic and weather parsers and the timer. This part of the program deals with data collection.
    - Vladimir Samokhin is working on the GUI. This part of the program deals with taking the user input(s) and transmitting each input to the appropriate classes and methods in order for data analyses to begin.
    - Sanket Wagle is working on creating the analyzer module class. This part of program deals with traffic prediction.

- Akhilesh Maddali is working on creating the geocoder module class. This part of the program deals with taking user inputs and translating the selected region into latitude and longitude. This helps in the data collection.
- Vamshidhar Chilukamari is working on the database interface. This part of the program helps in interaction between all the methods and the database for collection all necessary data.

Resources

- Gantt Chart Tutorial video on youtube: http://www.youtube.com/watch?v=HQwE0Xv1lAA&feature=relmfu
- Google Maps API: www.code.google.com/apis/maps/index.html
- Weather information: www.weather.com
- Traffic Information: www.511nj.org
- PHP Tutorial: www.w3schools.com/php/default.asp
- SQL Database documentation: www.mysql.com
- PHP 5 for Dummies by Janet Valade