Technical Documentation

Onward

Group 4

Team Members

Name	Email
Ridwan Khan	ridwankhan101@gmail.com
Brian Monticello	b.monticello23@gmail.com
Mhammed Alhayek	almoalhayek@gmail.com
Sean Olejar	solejar236@gmail.com
Lauren Williams	laurenwilliams517@gmail.com
Shubhra Paradkar	shubhra.paradkar@gmail.com

Code Files

File Name	Short Description
Landing Page	
-index.html	Constructs and displays the landing page
Heatmap	
-index.php	Constructs and displays the heatmap web page
screen_display.js	Sets up the map with severities displayed on it.
Route	
-index.php	Constructs and displays the route feature
screen_display.js	Sets up the map with severities displayed on it.
graph_print.js	Sets up the graph.
General	
-map_comminicator.php	Collects route information
-traffic_collector.php	Collects traffic information from database based on user inputs
-weather_collector.php	Collects weather information for forecasted weather
-loc_converter.php	Converts addresses or zip code to latitude and longitude pairs and receives zip code for addresses
-controller.php	Handles parameters and function calls
-gas_calculator.php	Calculates the cost of gas for route(s)
-info_entry.js	Updates map when parameters are changed on the form

Data Collection Files

FileName	Short description
-masterCurl.php	Main data collection file.
-find_startstop.php	File to get starting and ending coordinates for roads.
-cron_tab.txt	Cron Job

Landing Page

index.html

This file is called when the user directs to http://www.onwardtraffic.com/ and constructs the landing page (or Home Page). This will also link users to the “Heat Map” and “Route” features. This file uses HTML and CSS elements as well as Javascript. and is based off of a bootstrap template, as other the other index files.

Heat Map

This file exists in the “heatmap” folder and are used for the Heat Map feature.

index.php

This file is the one called once the user clicks “Heat Map” on the landing page and can be directed by the URL http://onwardtraffic.com/heatmap/. This file displays an embedded Google Maps and a form which will accept user inputs to generate the Heat Map. This file contains HTML, CSS, Javascript and PHP. The file uses HTTP POST request method to pass the form variables to the server through a php script. This embedded Google Maps Javascript API displays traffic information based on user inputs by the use of polylines (https://developers.google.com/maps/documentation/javascript/examples/polyline-simple).

JS Folder

screen_display.js

This file contains the javascript code necessary to set up the map on the route page. It contains the code to run the Google Maps Javascript API.

Route

This file exists in the “route” folder and are used for the Route feature.

index.php

This file is the one called once the user clicks “Route” on the landing page and can be directed by the URL http://onwardtraffic.com/route/. This file displays an embedded Google Maps and a form which will accept user inputs to generate the Route, as well as show an alternate route and estimate gas cost(s). The file uses HTTP POST request method to pass the form variables to the server through a php script. This file contains HTML, CSS, Javascript and PHP. This embedded Google Maps Javascript API displays traffic information based on user inputs by the use of polylines along the user’s desired route(s) between addresses. Additonally, this embedded Google Charts API creates and displays a line graph to show traffic information at different hours in the day (https://developers.google.com/chart/interactive/docs/gallery/linechart).

JS Folder

screen_display.js

This file contains the javascript code necessary to set up the map on the route page. It contains the code to run the Google Maps Javascript API.

graph_print.js

This file contains the javascript code necessary to set up the graphs on the route page. It uses the Google Charts API. The charts are set up by a callback on the drawChart() function that is called when the page is loaded.

General

These files contain the functions used in our program, for both Heat Map and Route features, and exist in the “php” folder and are all written in PHP.

controller.php

This file handles all the “General” files as it “include”s the other files. It is responsible for storing and organizing parameters passed by the index php files, making function calls and passing appropriate parameters to the other PHP files as well as handling the returns from function calls.

loc_converter.php

This file is responsible for converting user inputted zip codes and addresses to latitude and longitude pairs and returning these pairs to the controller.php. The latitude and longitude pairs for zip codes, starting and ending addresses are returned from calling the Google Geocoding API (https://developers.google.com/maps/documentation/geocoding/intro). This file in addition returns the zip code that is needed for the WeatherCollector.php. If the Heat Map feature is being used, then it simply returns the inputted zip code. If the Route feature is being used, then it returns the zip code of the starting address using the Google Geocoding API.

traffic_collector.php

This file will query the database based on the user inputs to receive traffic information from our SQL Database. If the user inputs parameters to create a Heat Map, traffic_collector.php will creates a SQL WHERE clause based on the latitude and longitude of the zip code. This WHERE clause is used to find traffic incidents within the database within the radius (range of miles) of the zip code specified. This clause along with parameters for a specific day of week, time and weather from the user input will query the database to find incidents which fit the user parameters. For our design of the database, we have divided our region of interest for Onward into grid boxes, that have latitude and longitude values for the 4 sides of the box as well as center of the grid box. We query the “grid” table in our database to see which grids fall in the range, using the WHERE clause and then using these grids, time, day and weather parameters we query the “severity” table to find the roads and severities in the grid boxes of interests, and then using the road name and gridid we query the “bounds” data table that tells us the starting and end latitude longitude pairs for the roads in the grids of interest to display, which along with severity values is returned to controller.php which will then send this back to respective index.php files to display on the map. If the user inputs parameters for Route, then TrafficCollector.php queries the “grid” and “severity” tables to find the roads (and legs of roads) returned from MapCommunicator.php to find the severities of these roads based on parameters of time, day and weather.

map_communicator.php

This file is used to get route information for the user specified starting and ending addresses and parses the route to get the different legs along the route so that later, these legs of the route can be associated with severities and be shown on the map. To get the route and direction information the Google Maps Directions API was used (https://developers.google.com/maps/documentation/directions/).

weather_collector.php

This file is exclusively used when the user clicks on “Forecast” from either the Heat Map or Route page. The weather collector will use a zip code, date, and time to call the Weather Underground API (https://www.wunderground.com/weather/api/) to find the forecasted weather, and returns the weather as either“Clear”, “Snow”, “Cloudy”, “Rain”, and “Fog”. We generalize all weather returns from the API to these 5 weathers for the sake of simplicity. The weather collector is able to find the weather forecast up to 10 days in advance. The API requires a specific time since it uses the hourly 10 day forecast, therefore there is not an “All Times” option for user input.

gas_calculator.php

This file is used to calculate the cost of gas along a route, based on user inputs of MPG and fuel type. It gets the gas price from the Fuel Economy Web Services API (http://www.fueleconomy.gov/feg/ws/index.shtml#ft9). It uses this gas price, the user inputs, and distance information received from map_communicator.php, to calculate an estimated gas cost for a route.

info_entry.js

This file contains the javascript code that updates the traffic incidents displayed on the map when the form is changed.

Data Collection

These files are specifically for our data collection, masterCurl and find_startstop are written in PHP and cron_tab is a simple txt file.

masterCurl.php

This file is what manages our data collection, using MYSQL and PHP.. It is executed by a Cron Job that runs on the 5th minute of every hour. It uses several APIs for data collection such as the Bing Maps Traffic API (https://msdn.microsoft.com/en-us/library/hh441725.aspx) to receive

traffic incidents, the Weather Underground API (https://www.wunderground.com/weather/api/) to receive weather data, the Google Geocoding API (https://developers.google.com/maps/documentation/geocoding/intro) to get road names from Google for consistency with the embedded Google Maps Javascript. This file connects with multiple data tables. The “traffic_data” table contains the the pulled traffic information from the Bing Maps API. We choose this over using Google and other APIs as this best fitted the information we wanted for traffic incidents.The “frequency” table keeps a count of the 5 weather types for different zip regions. The “grid” table stores the gridID and latitude longitude pairs as discussed in TrafficCollector.php as well as associated zip codes. The “severity” table that has a row for all road segments in each grid for each day of the week for each hour and has the total severities for the 5 weathers and the respective average severities, that is calculated using the frequency table. This file also calls the find_startstop.php.

find_startstop.php

This file connects to the “bounds” data table and for every grid id and road name pair assigns pairs of starting and ending latitude longitude pairs. This file is called my masterCurl.php and is therefore run with the Cron Job.

cron_tab.txt

This file runs our Cron Job on the 5th minute of every hour to pull in all our necessary data.