TRAVELER’S EYE

**Introduction:**

Being international students we all understand the intricacies involved in travel. The stress of keeping track of all the information about the destination and the travel, especially when it comes to travelling alone, takes away from the fun one can have while travelling. Traveler’s eye is aimed to make a traveler’s life easy by providing relevant useful information whenever the user needs it, at a simple button click or the touch of a finger.

Services Designed in Second Increment:

Weather service.

Traffic service.

Validating Jquery mobile GUI

Part of database

API’S and Web Service:

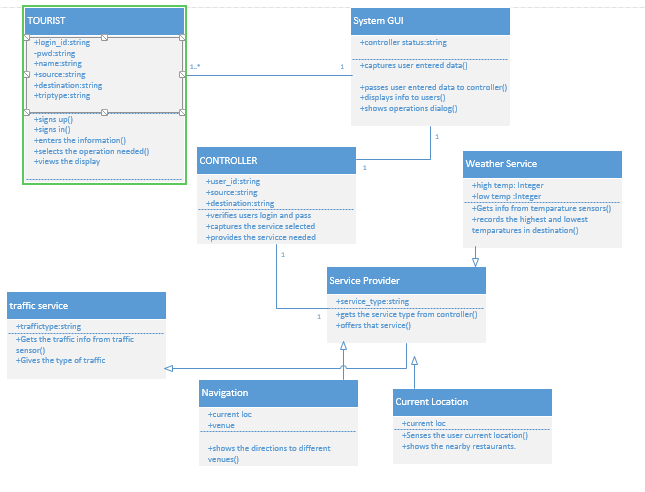
Google traffic layer

cdyne weather service

Validation plugins: <http://ajax.aspnetcdn.com/ajax/jquery.validate/1.11.0/additional-methods.js>

<http://ajax.aspnetcdn.com/ajax/jquery.validate/1.11.0/jquery.validate.min.js>

CLASS DIAGRAM:



The class diagram describes the static structure of the application. The class diagram consists of classes, attributes and operations.

**Classes:**

Tourist: This class refers to the user of the application. The user should first sign up with application and them login into it and give his details. And then can use the services of the application.

System GUI: This is the interface between the user and the application. It captures the user’s data and passes it on to the controller .And displays the useful information to user as requested by the controller.

Controller: This is the heart of the whole application. It analyses everything. It updates and verifies database. Contacts the service provider and brings the service needed into play.

Service Provider: This is used to provide services to the application. Each service is a sub class to this super class.

1) Weather service: This gives the weather conditions at the given destination.This gives the highest and lowest temperatures at the given position.

2) Traffic service: This gives the traffic updates at the place required by the user.

3) Current Location: The current location of the user is identified.

4) Navigation: This shows the navigation from to user’s current position to the place he wants to move.

**Relationship-Multiplicity:**

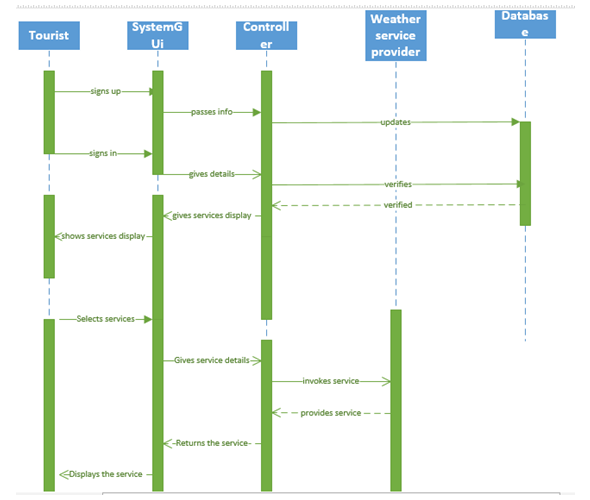
Tourist-SystemGUI: The SystemGUI is same and it can be used by multiple user’s .Hence the multiplicity is one or many-one. The relationship is association relationship.

SystemGUI-Controller: There is a purely one to one relationship between these two. As only single piece of them exists.They have an aasociation relationship.

Controller-Service Provider: These both also have association relaitonsip.And they have one to one multiplicity.

Service provider-weather service ,Service provider-traffic service ,Service provider-navigation, service provider-current location: These have parent child relationship. The Service provider is the parent class and the individual services are the child classes.

**SEQUENCE DIAGRAM FOR WEATHER SERVICE:**

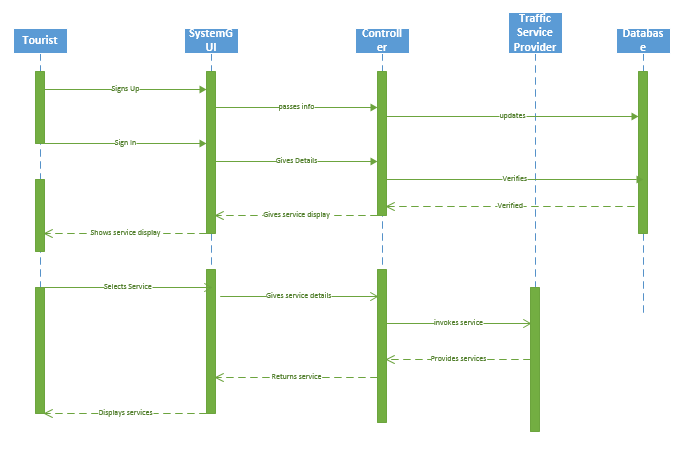
****

This is an interaction diagram which shows the interaction of one class with another class with respect to time.

**Steps:**

* The user first signs up using SystemGUI.
* The GUI informs the controller.
* The controller stores the information in the database.
* The user now logs in again by taking the help of SystemGUI.
* The SystemGUI informs the controller.
* The controller verifies the database.
* The database sends the authentication.
* The controller asks the SytemGUI to display the services.
* The SystemGUI displays the services to the user.
* The user selects one of the services.
* This is informed to the controller with the help of the SystemGUI.
* That service is provided by the service provider by contacting its sub classes.
* The service provider provides the service to the controller.
* The controller passes it to SystemGUI.
* Finnally,the required service is displayed to the user.

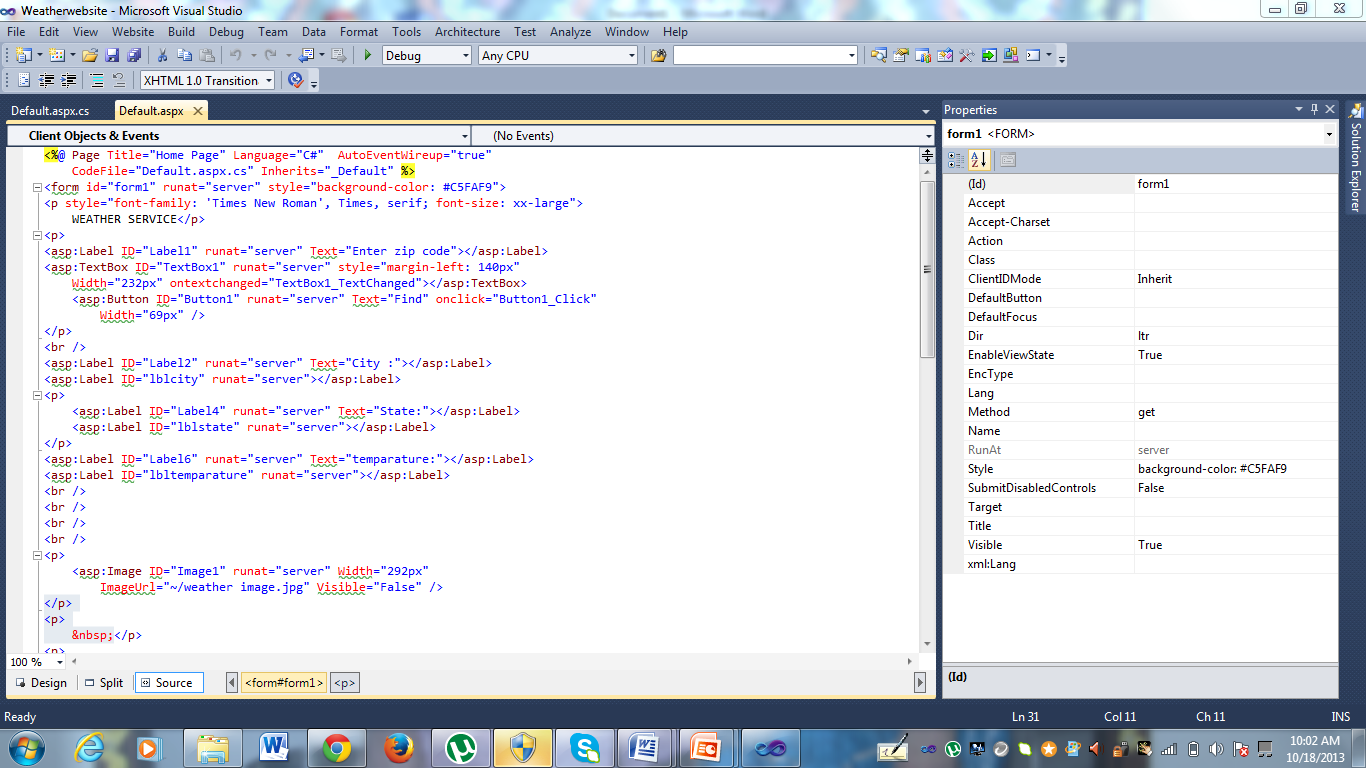
**SEQUENCE DIAGRAM FOR TRAFFIC SERVICE:**

****

**WEATHER SERVICE:**

This service is used to provide the user with the temperature conditions at the destination he wants to reach or about the place he wants to know. This is a dynamic service which uses CYDNE web services.

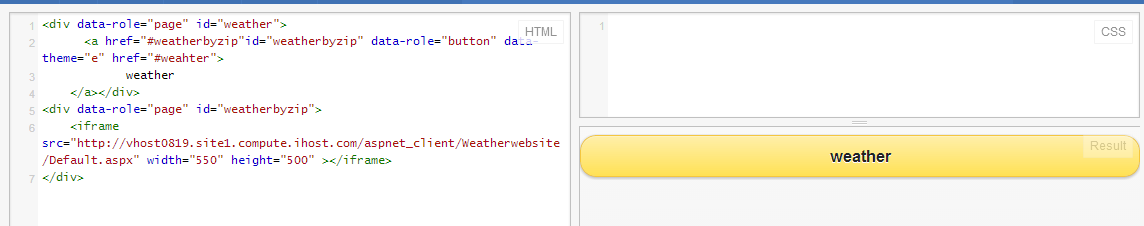
First of all we create a web site in visual studio and then add the web reference .After this we write the code accordingly and design the web page.



Then the code for the button click is written in a way to use the web service in our application.Then the web service is used in our application locally.

Then our service is deployed into cloud by following the required steps.

After deploying it into cloud our client communicates with the service provided. And finally displays the output.



We can access the weather service by clicking the button.The output will be as follows after clicking the button.



When we enter the zip code we want and click on find button.

We get the city and state and temperature of that place dynamically.

**Traffic service:**

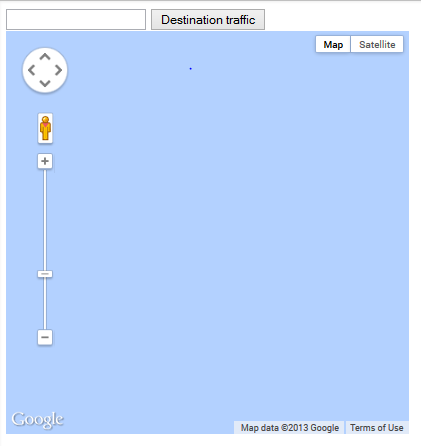
This service enables user to have a look on traffic of a region, as traveler’s eye assured him to predict the traffic of destiny before reaching there.

In order to facilitate this traffic layer’s service from google maps are been utilized.

Initial step involving creating an object for google maps which gets rendered on the output screen with traffic intensity plotted over it.

Place on google can be plotted using the geocode or latitude and longitude of the location. But traveler’s takes in the name of location and plots it on the map by retrieving the geocode of the location.

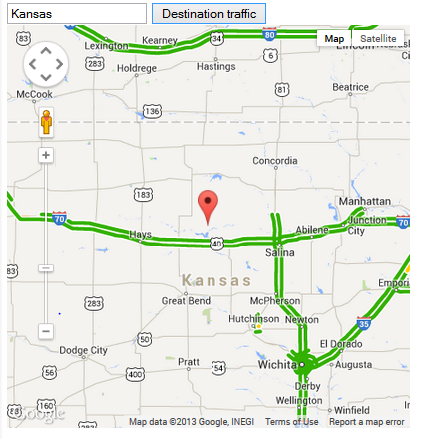
Once geocode of location is found, next step is to create an object for the traffic layer. Now traffic layer is overlaid on the map object with clear distinction of the location with a marker.



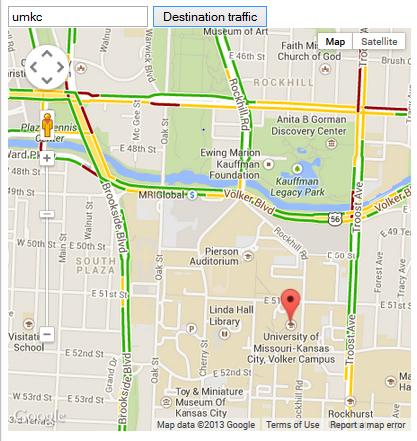
Above is the initial display that appears on the mobile output screen.



User intends to find the traffic in the region of Kansas and therefore he enters Kansas and the clicks on destination traffic button.



Green showing a free flow of traffic, red indicating a higher intensity of traffic. On adjusting the zooming levels user can a detailed look of traffic.



**Mobile GUI Validation:**

In this increment, I have completed client side validation on JQuery mobile GUI. For this validation I have used some JQuery plugins in to jsfiddle in order to complete the validation process for Registration page.

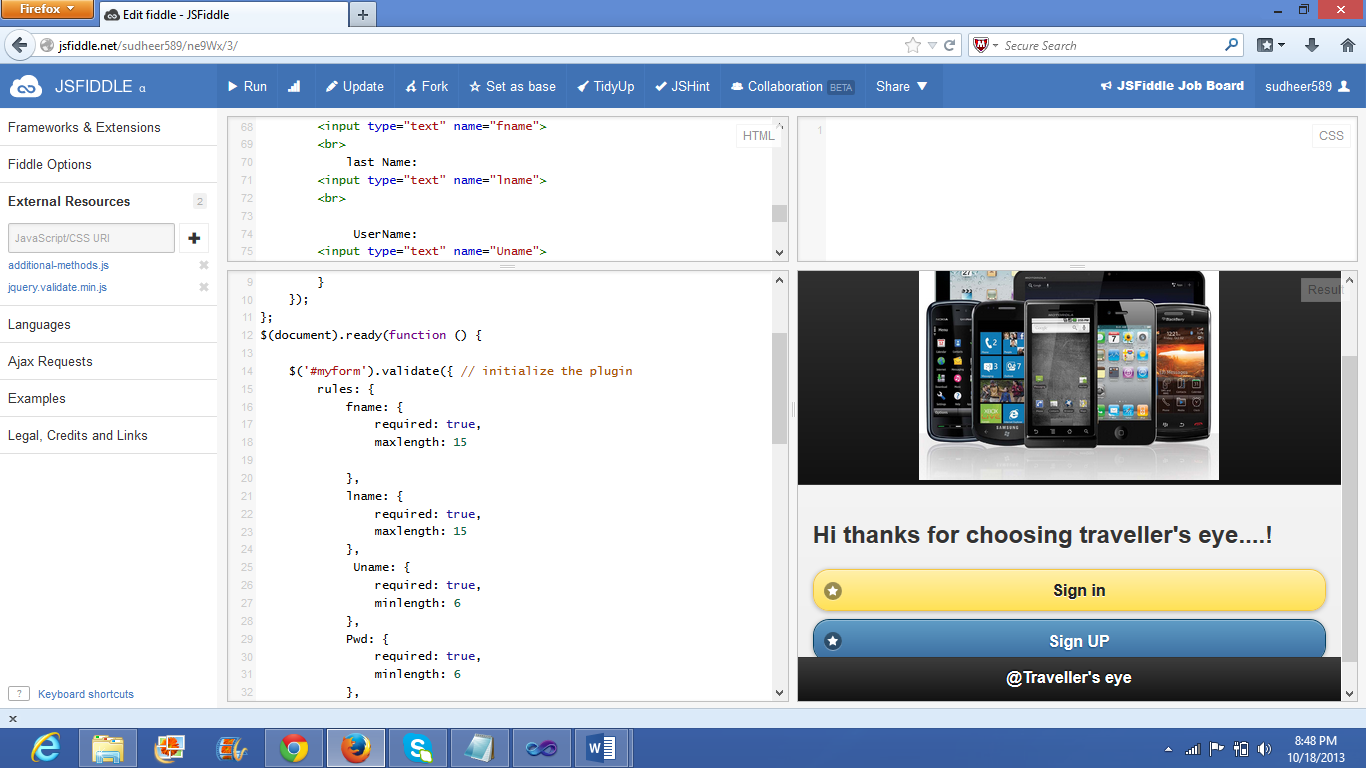
Jquery.validate.min.js:

<http://ajax.aspnetcdn.com/ajax/jquery.validate/1.11.0/jquery.validate.min.js>

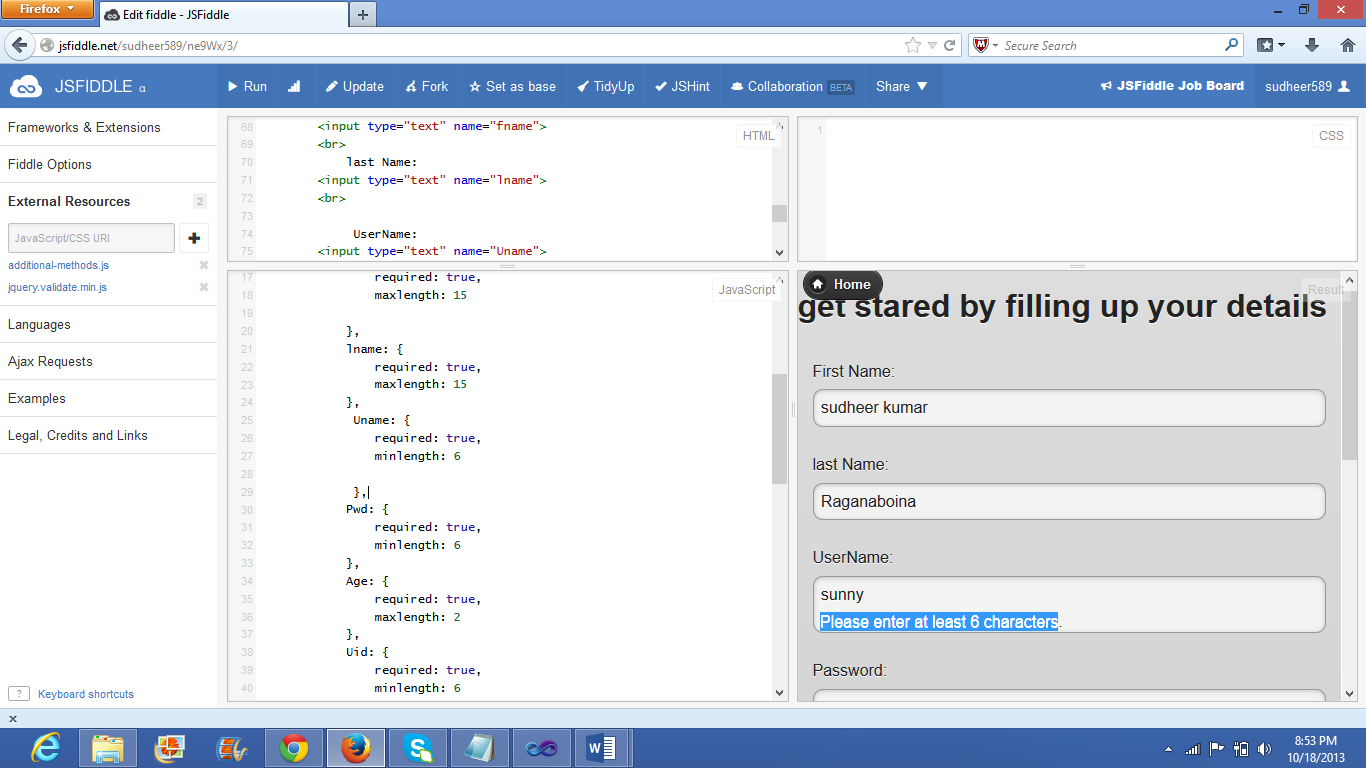
Additional- methods.js:

<http://ajax.aspnetcdn.com/ajax/jquery.validate/1.11.0/additional-methods.js>

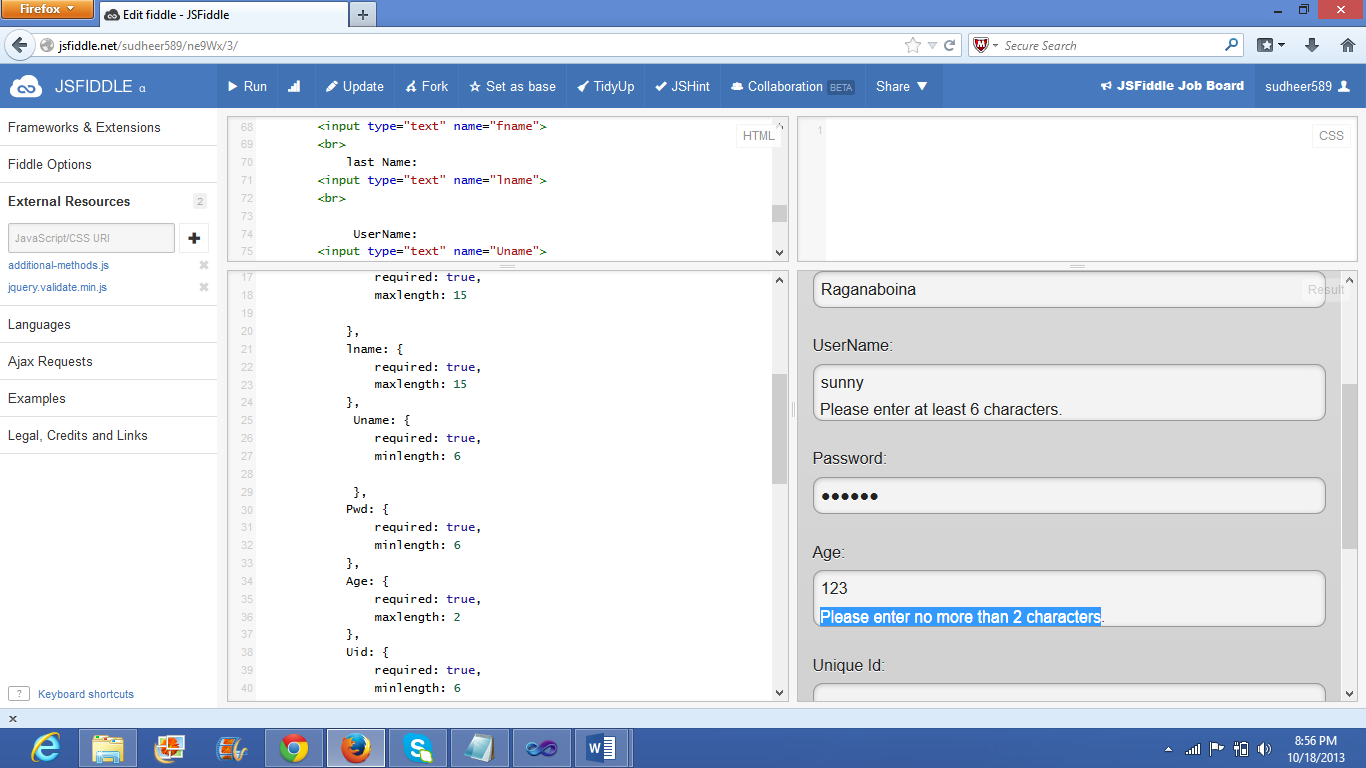
**Print screens:**



**Validating User Name:**



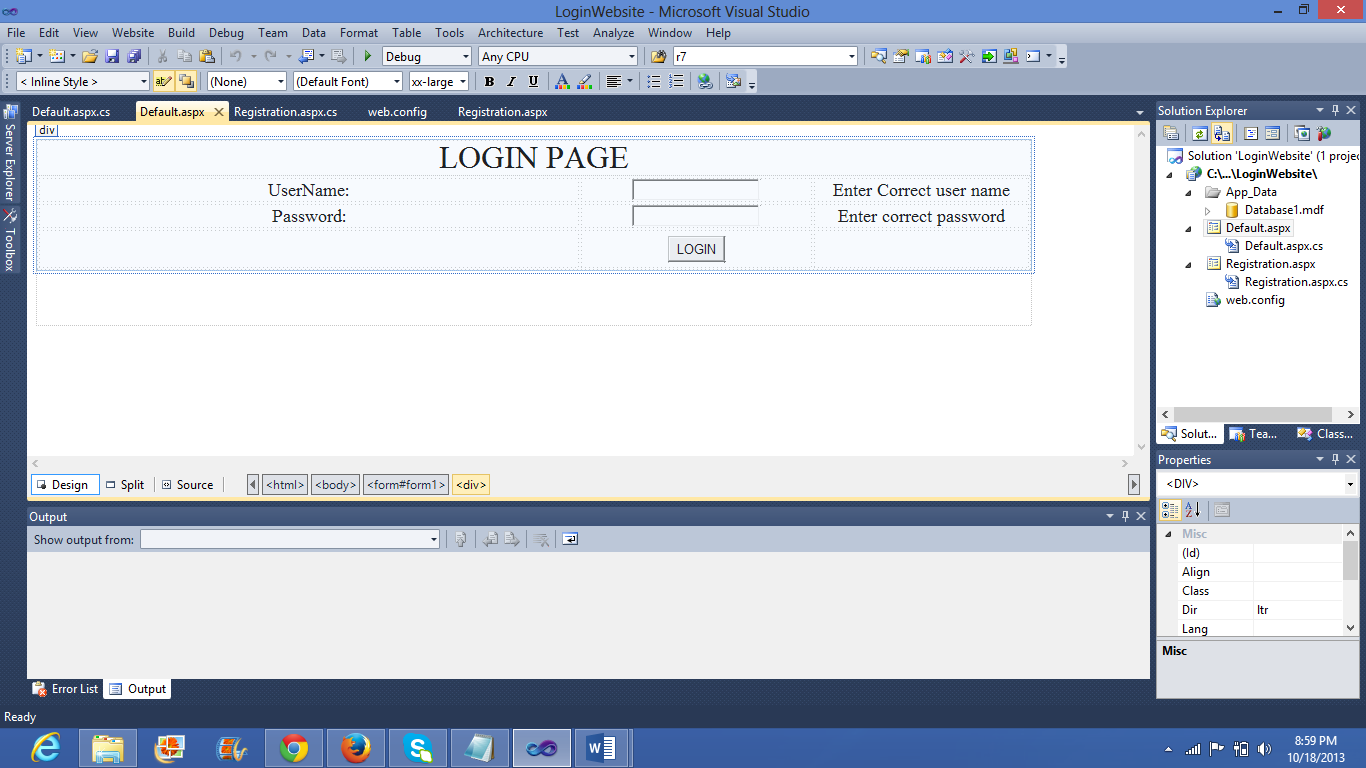
**Validating age:**



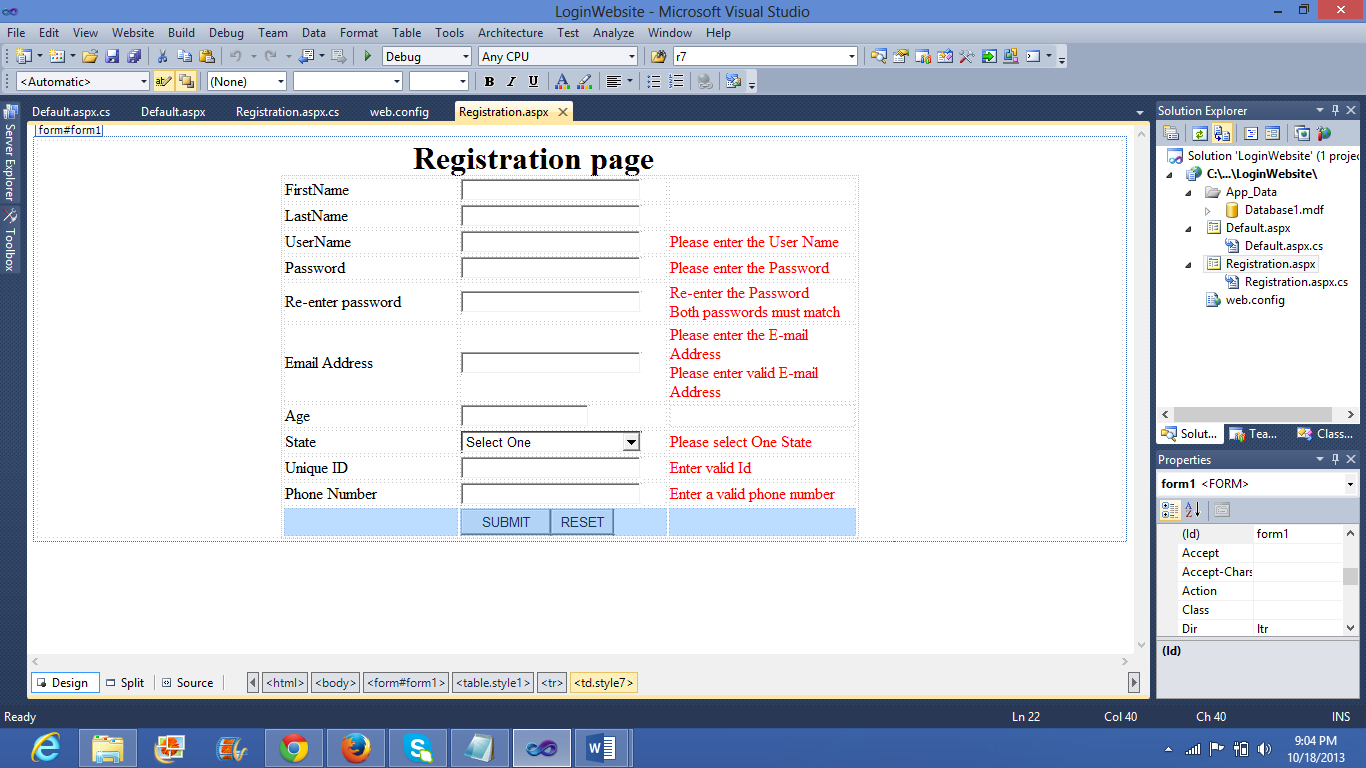
Even I have worked on the web client for creating and validating the login and registration pages in Microsoft visual studio.

**Screen shots:**

Creating and validating the login page:

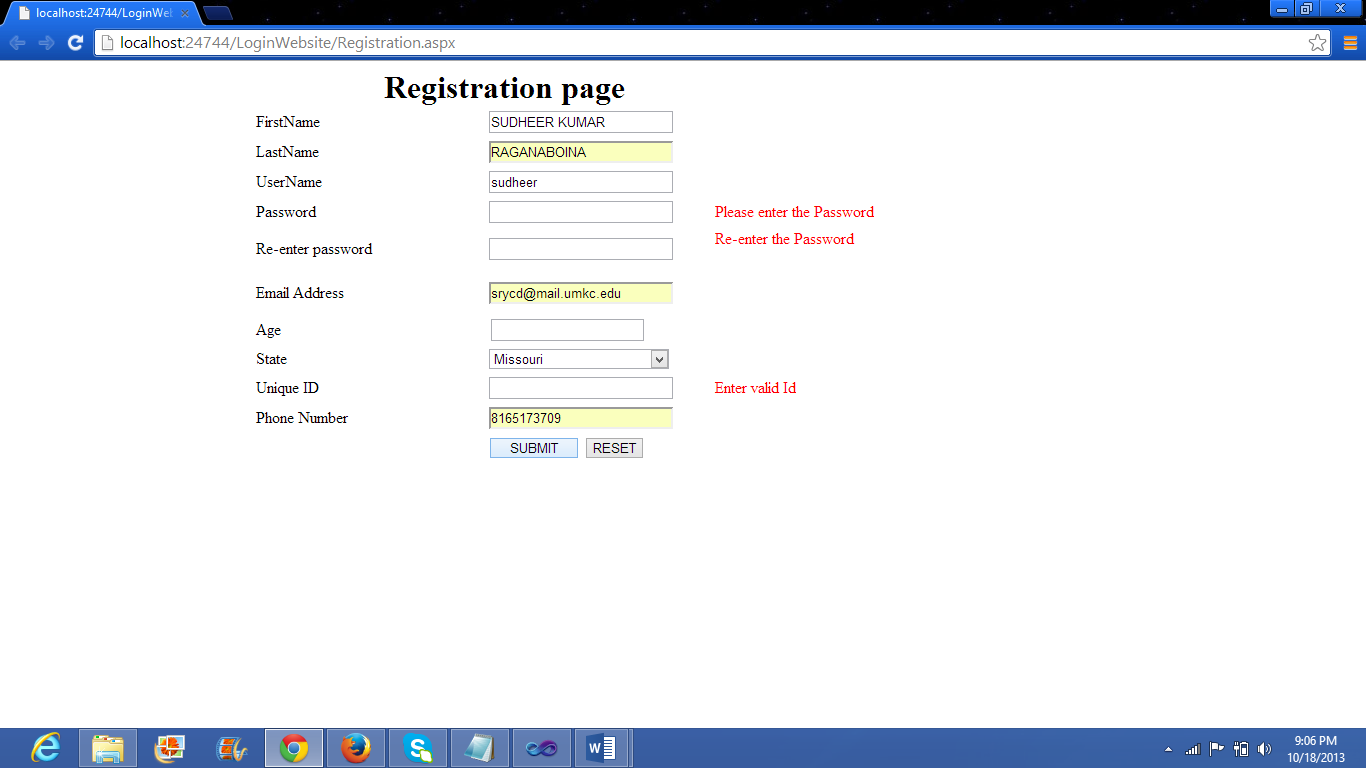


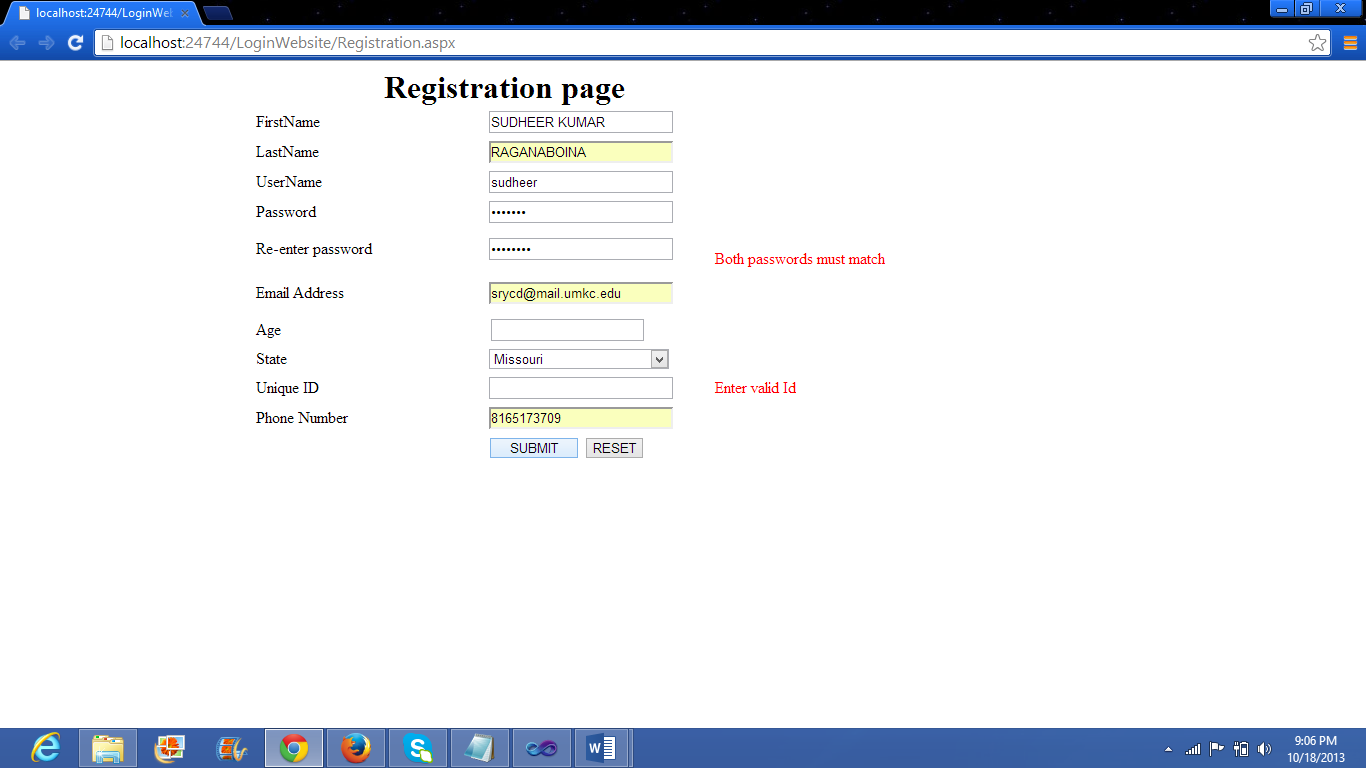
**Creating and validating Registration page:**

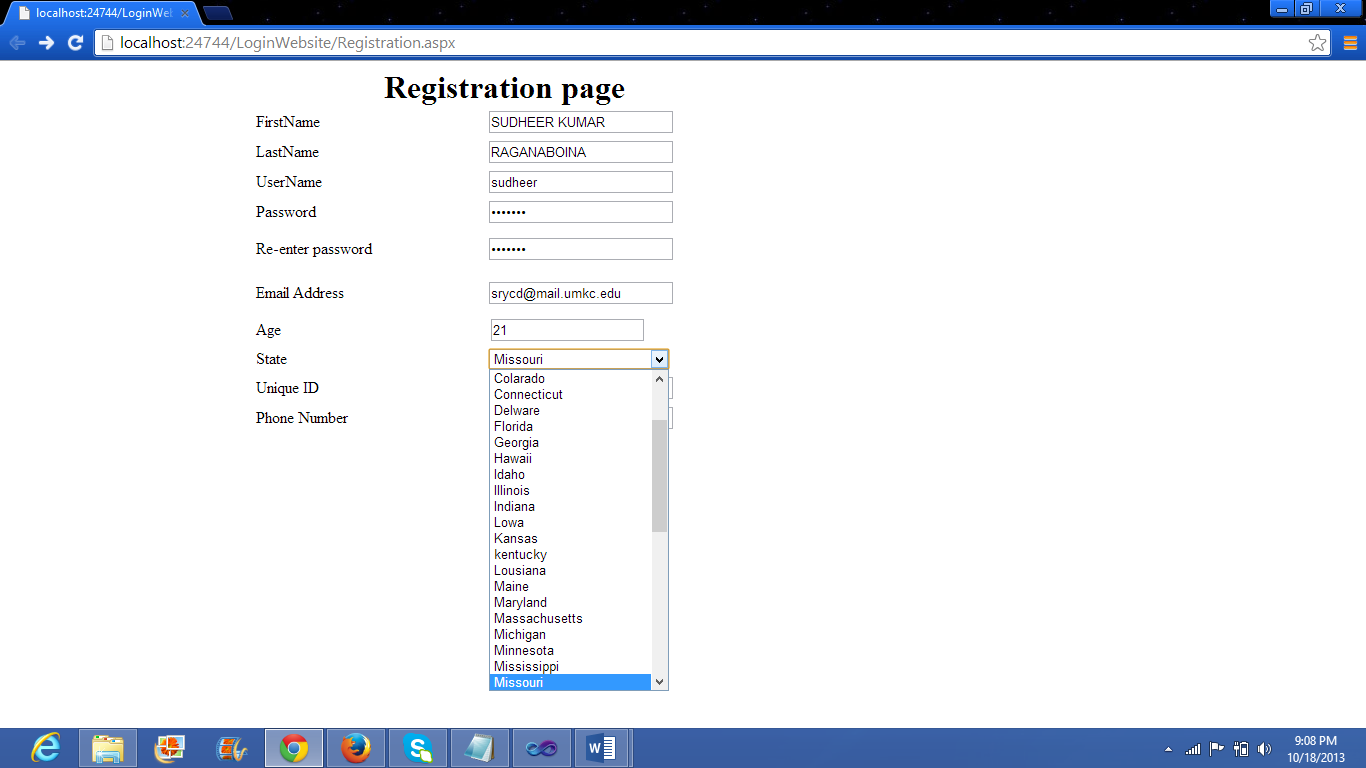


Validating and executing the registration page in the **local host**:

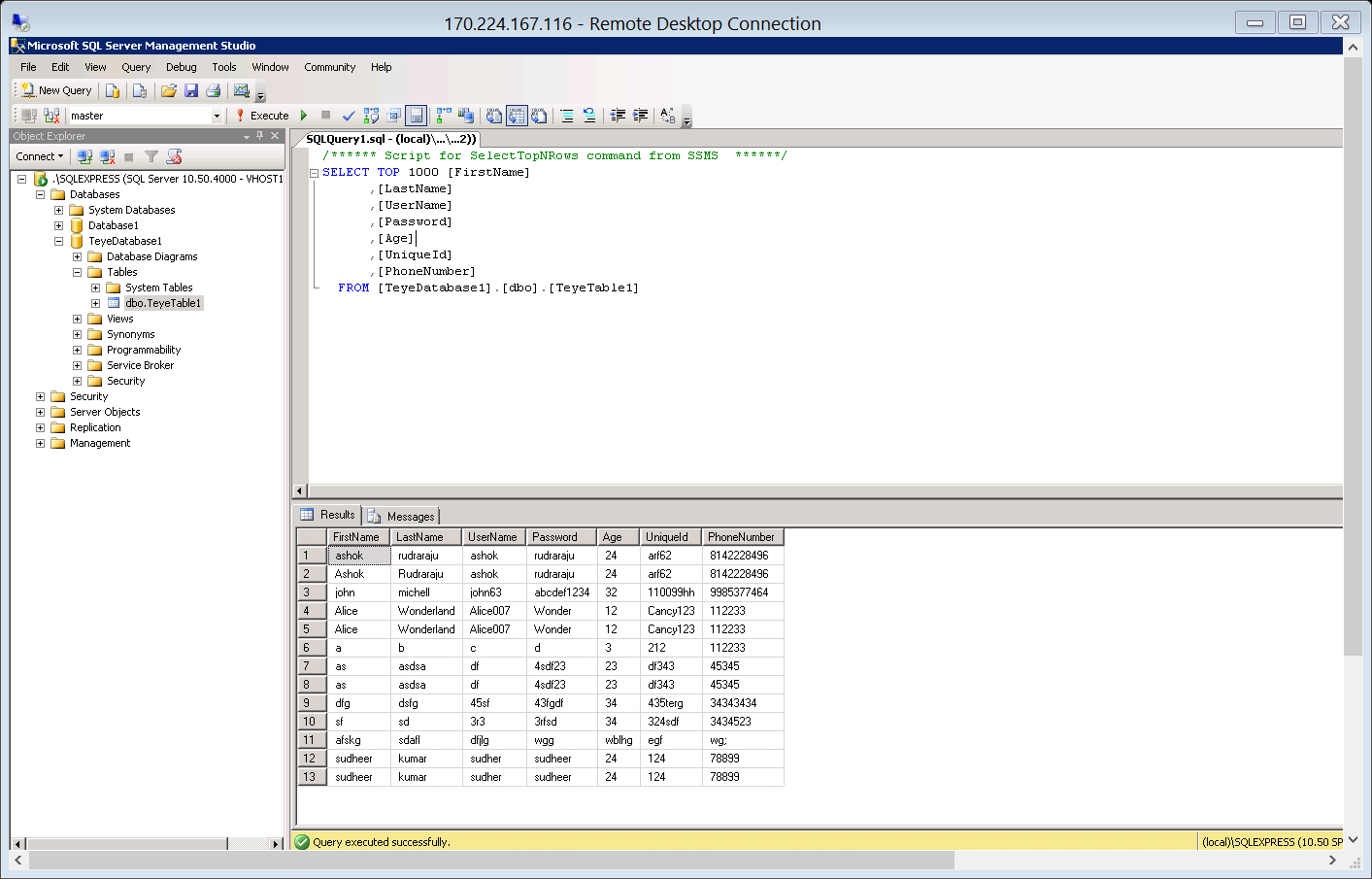
<http://localhost:24744/LoginWebsite/Registration.aspx>







Database:



**Testing the GUI (Graphical User Interface):**

1. When the application is executed on the first click it is displaying the proper screen fit output.
2. Each and every field in the page is aligned properly i.e; text boxes, radio buttons.
3. By clicking on each input text field the pointer of the mouse is changing as the Cursor.
4. When the “Tab” button is clicked, the cursor is shifting its control to the next input text box.
5. The application is allowing the user to select a single radio button at a single instance.
6. It is allowing multiple selection of check boxes.
7. When the user enters the password the password should not be displayed and the password is not displayed as characters.

**Project Management:**

Sindhu Koneru-weather service

Sujitha Onteru-Traffic service

Sudeer kumar rangaboina-Validating Jquery mobile GUI ,Created and validated login and registration pages in dotnet and working on database.

Ashok Rudraraju-part of database.