

TOUR LOGGER

ASE Project Document Phase 2

Version 1.0

03/18/2015

Team Members:

1. Rajesh Kapa
2. Raghu Vital Gummadi

Contents

1. OBJECTIVES.....	3
2. IMPORT EXISTING SERVICES/API.....	3
3. DETAIL DESIGN OF SERVICES.....	3
3.1 Write User Stories /Use Case (Using ScrumDo – Include the screenshots)	3
3.2 Service description.....	4
3.3 Sequence diagram (using Visio).....	5
3.4 Class diagram (using Visio)	6
3.5 Design of Mobile Client Interface.....	6
3.6 Design of Unit test cases (using NUnit tool).....	7
4. IMPLEMENTATION.....	7
4.1 Implementation of REST services	7
4.2 Implementation of user interface (Mobile Apps)	9
4.3 Implementation of test cases.....	16
5. TESTING:	16
6. DEPLOYMENT:	16
6.1 Project ScrumDo link	16
6.2 GitHub site URL.....	16
7. PROJECT MANAGEMENT:	17
8. RESOURCES:	17

1. Objectives

In this Increment, there are several objectives which are to be achieved are described below. They are,

- a. Creating a WCF service for the end user login and consume it in our android application TOUR LOGGER.
- b. Creating another WCF service for the end user registration and consume it in our application.
- c. Using Google Places API in order to retrieve the list of places.
- d. Using Openweathermap API in order to show the weather for a selected place.
- e. Making Validations for user login and registration forms.
- f. Making use of google + button for sign in with google + option.
- g. Using DatePicker tool for android in order to have a Start date and end date for a trip.
- h. Creating several activities with customized action bar and background images.

2. Import Existing Services/API

The services that are used for this increment are

Service1: WCF Service using Microsoft visual studio for the Login process.

Service2: WCF Service using Microsoft visual studio for the registration process.

Service3: Google places API is used for displaying the search results of places when a string is entered in a search box.

Service4: OpenWeatherMap API is used for displaying the weather details like current temperature, humidity, pressure, the symbols relating to the weather and also the date and time of the last update.

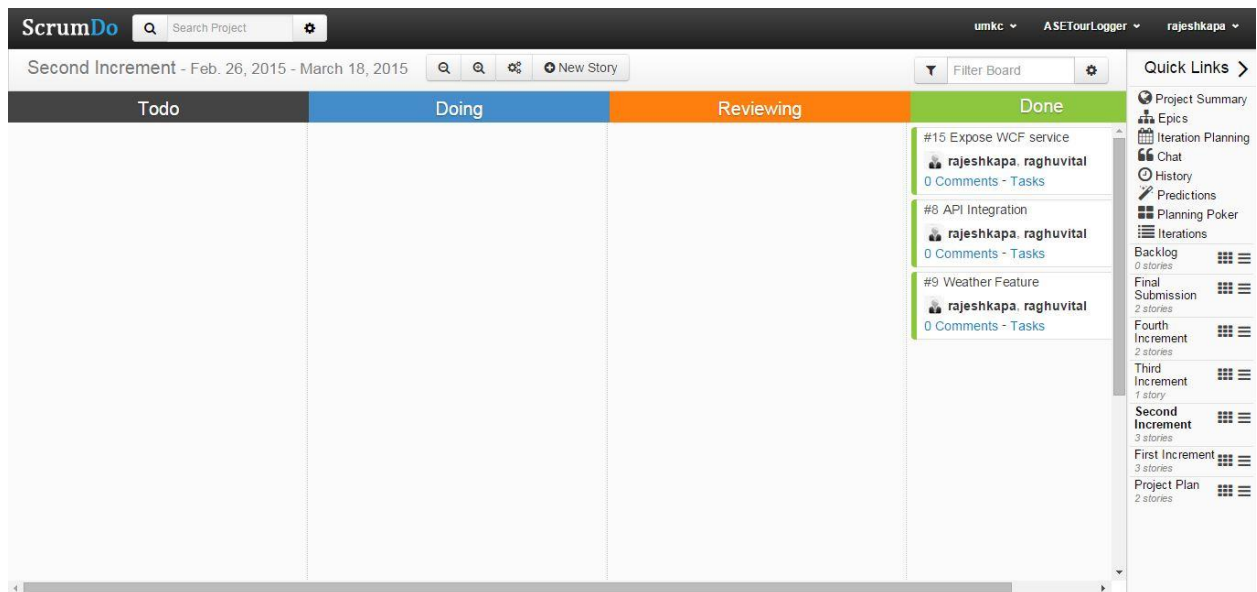
3. Detail Design of Services

3.1 Write User Stories /Use Case (Using ScrumDo – Include the screenshots)

Below three stories are implemented in iteration 2

1. Expose WCF service to store user data in sql server.
2. Integrate Google API to get auto completion of text for cities search and store these details.
3. Implement openweathermap to show weather of particular city on the trip day.

Below is the screenshot of the ScrumDo.



3.2 Service description

Login service: This is a WCF service, when a user logs in with their details, this service is called which takes inputs such as username and password. The password stored in the database and the entered password are compared.

Registration service: This is also a WCF service, when a user inputs details such as first name, last name, email, password, and re-enter password, this service is called to take the inputs from the user and insert them into the table.

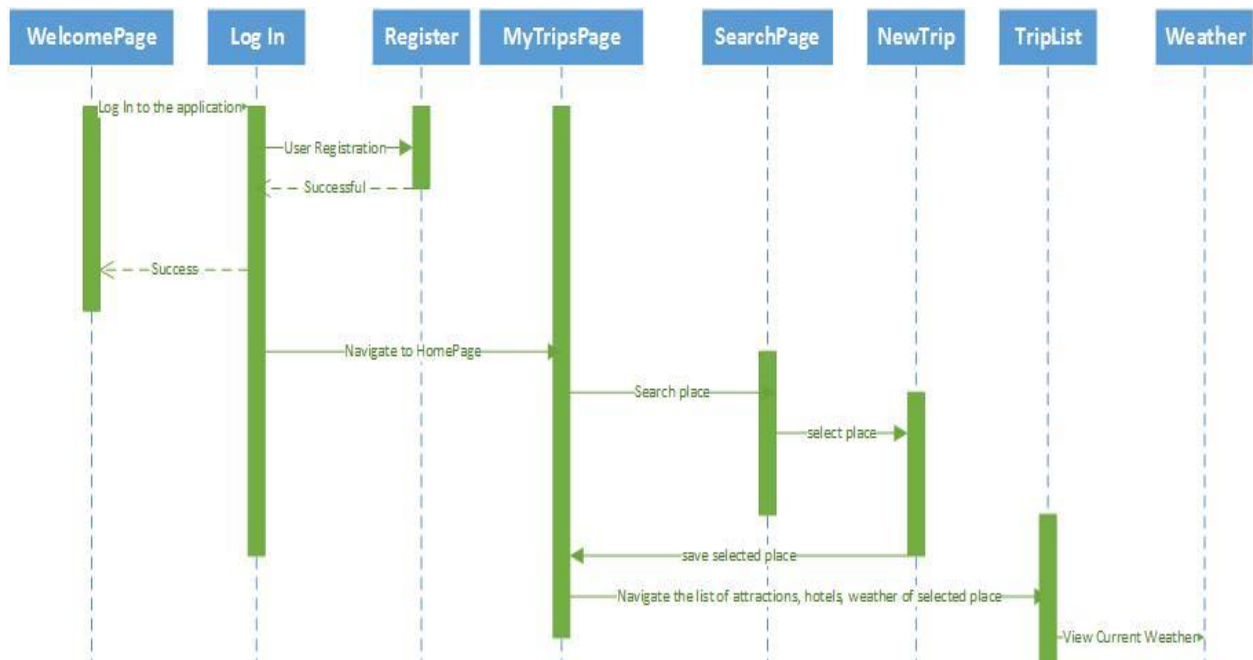
Google Places API: Through this API, users can use several features like Place details which shows more detailed information about the selected place, Place add allows to add the data present in the Google database with our application, place searches return a list of places depending upon the user input or current location, place autocomplete automatically completes the name of the place as the user types in the search box.

In this increment, we are using the place autocomplete feature of the Google Places API. When the user enters a string with a minimum of one letter, this feature shows the available list of cities present starting with that entered string. So, we can select one place in that list.

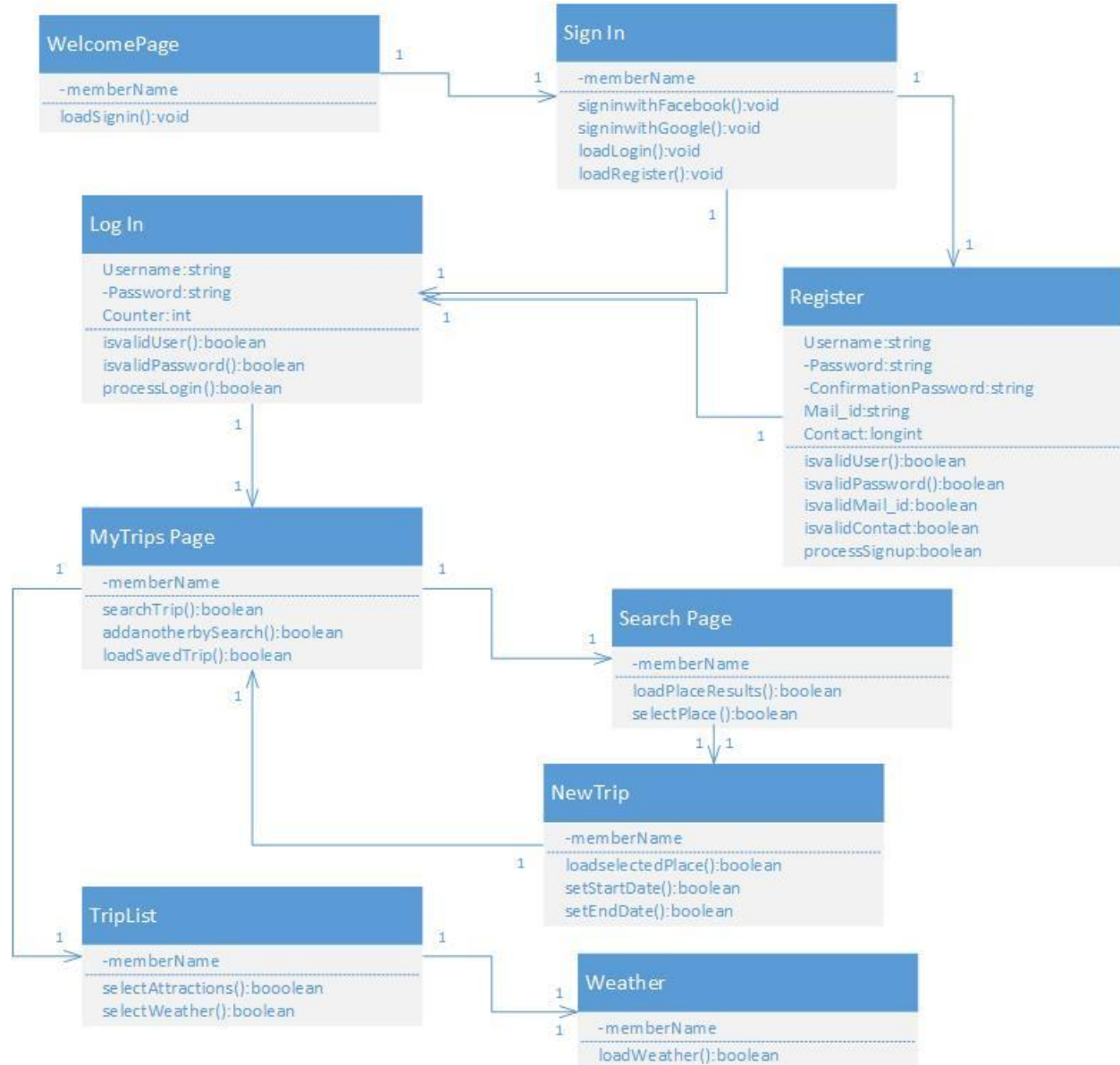
OpenWeatherMap API: This API provides various features like Current Weather data, 5 and 16 day forecast, historical data, weather stations, and weather map layers.

In this increment, we are using the Current Weather data feature which gives weather details of almost any location on earth. It is updated frequently depending on the global models and on the data from various weather stations. This data is available in JSON, XML, or HTML format. We can call this service by using place names, place IDs, and also using latitudes and longitudes of the place.

3.3 Sequence diagram (using Visio)



3.4 Class diagram (using Visio)



3.5 Design of Mobile Client Interface

Our TOUR LOGGER application's client Interface has several screens which are designed user friendly. These are MyTrips Page, SearchLocation Page, NewTrip Page, TripList Page and Weather Page. After Logging in the user interacts with the MyTrips Page that has option to add a place and show a place that is selected. The Search location page helps user to search for places and after selecting a place he will be navigated to another page NewTrip where he inputs the start date and end date and saves his trip. Then the details about the place and dates are saved in the database. The saved trip will be displayed in the MyTrips page and for this trip there will be a TripList Page which has a list view that displays several content like attractions, hotels and weather etc. The weather page displays the weather of the saved city by using the OpenWeatherMap API.

3.6 Design of Unit test cases (using NUnit tool)

Unit Test cases are designed in NUnit tool. Test cases are written for login, registration and trip add services. They are written and tested in positive scenarios.

4. Implementation

4.1 Implementation of REST services

Below REST Services are implemented in Tour Logger project are implemented as below. Generated google API key and used in service call.

Key for Android applications

API KEY	AIzaSyCImwX3a5VOeGwWUhzRq0ZE8_usAD1oPO0
ANDROID APPLICATIONS	FF:0F:2C:B3:68:49:0C:98:E7:9E:66:E8:B9:B1:FC:C7:FB:11:9A:E4;asepg15.umkc.edu.tourlogger
ACTIVATION DATE	Mar 14, 2015, 7:08:00 AM
ACTIVATED BY	rajesh.tist@gmail.com (you)
<div>Edit allowed Android applications Regenerate key Delete</div>	

a. Google Maps REST Service:

By using this API key auto generation of cities was implemented and tested manually. Below screenshot shows the code.

A screenshot of an IDE window showing several Java files: RegisterActivity.java, NewTripActivity.java, SearchLocationActivity.java, LoginActivity.java, and activity_new_trip.xml. The active file is SearchLocationActivity.java, which contains the following code:

```
private static final String PLACES_API_BASE = "https://maps.googleapis.com/maps/api/place";
private static final String TYPE_AUTOCOMPLETE = "/autocomplete";
private static final String OUT_JSON = "/json";

private static final String API_KEY = "AIzaSyA23jRoKzRI16Al7c88DpHxHDuW_OutLjU";

private ArrayList<String> autocomplete(String input) throws IOException, JSONException {
    ArrayList<String> resultList = null;

    HttpURLConnection conn = null;
    StringBuilder jsonResults = new StringBuilder();
    try {
        StringBuilder sb = new StringBuilder(PLACES_API_BASE + TYPE_AUTOCOMPLETE + OUT_JSON);
        sb.append("?key=" + API_KEY);
        sb.append("&types=(cities)");
        sb.append("&input=" + URLEncoder.encode(input, "utf8"));

        URL url = new URL(sb.toString());
        conn = (HttpURLConnection) url.openConnection();
        InputStreamReader in = new InputStreamReader(conn.getInputStream());

        // Load the results into a StringBuilder
        int read;
        char[] buff = new char[1024];
        while ((read = in.read(buff)) != -1) {
            jsonResults.append(buff, 0, read);
        }
    }
```

b. Using exposed WCF services:

Login service to check whether details are present in Database or not.

```
}  
else {  
    UserLoginCheck logincheck = new UserLoginCheck();  
    logincheck.execute(new String[] {"http://kc-sce-cs551-2.kc.umkc.edu/aspnet_client/MPG15/TourLoggerService/Service1.svc/login/user/"+ un + "/" + pw});  
}
```

Registration of user using REST WCF service

```
}  
else  
{  
    UserRegistration regcheck = new UserRegistration();  
    regcheck.execute(new String[] {"http://kc-sce-cs551-2.kc.umkc.edu/aspnet_client/MPG15/TourLoggerService/Service1.svc/register/user/"+ fn +  
}
```

c. openweathermap api is used to get weather of particular place. Below is the code used for consuming the existing service.


```

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

import org.json.JSONObject;

import android.content.Context;
import android.util.Log;

public class RemoteFetch {

    private static final String OPEN_WEATHER_MAP_API =
        "http://api.openweathermap.org/data/2.5/weather?q=%s&units=metric";

    public static JSONObject getJSON(Context context, String city){
        try {
            URL url = new URL(String.format(OPEN_WEATHER_MAP_API, city));
            HttpURLConnection connection =
                (HttpURLConnection)url.openConnection();

            connection.setRequestProperty("x-api-key",
                context.getString(R.string.open_weather_maps_app_id));






            BufferedReader reader = new BufferedReader(
                new InputStreamReader(connection.getInputStream()));

            StringBuffer json = new StringBuffer(1024);
            String tmp="";
            while((tmp=reader.readLine())!=null)
                json.append(tmp).append("\n");
            reader.close();

```

4.2 Implementation of user interface (Mobile Apps)

a. Registration Page: In this increment, we implemented the database table for the registration. When the user inputs his information, the data is stored and toast will be displayed as registration successful returning to the login page. Below are the screenshots of Registration and login pages after successful registration of the user.

     5:54

Register

First Name

Last Name

Mobile Number

Email Address

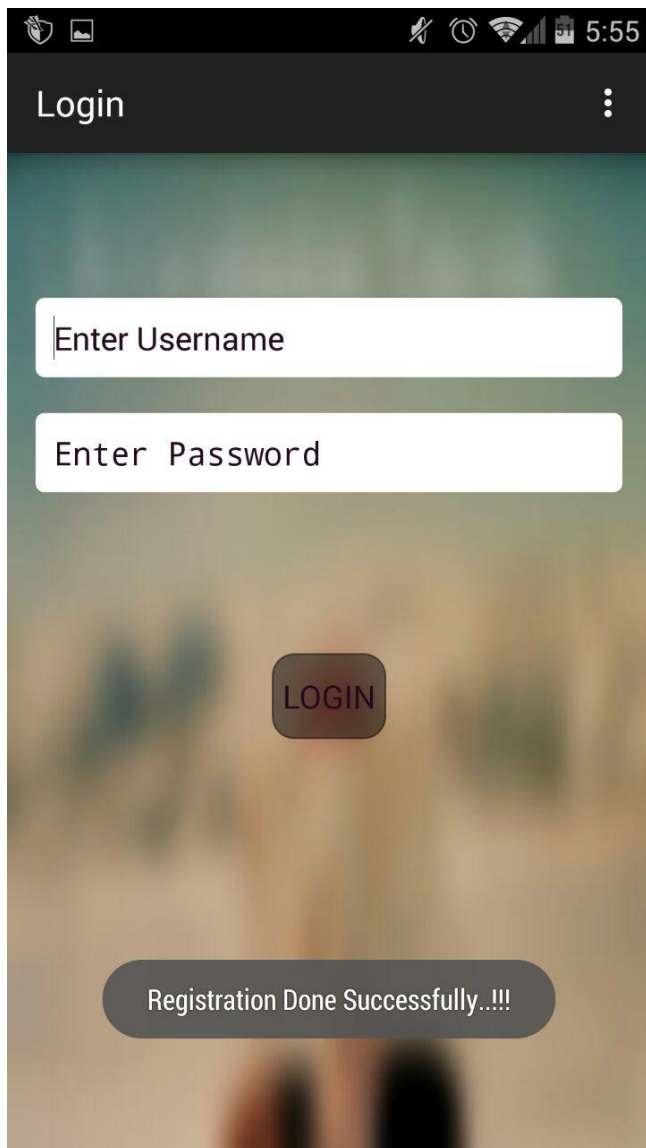
Password

Re-enter Password

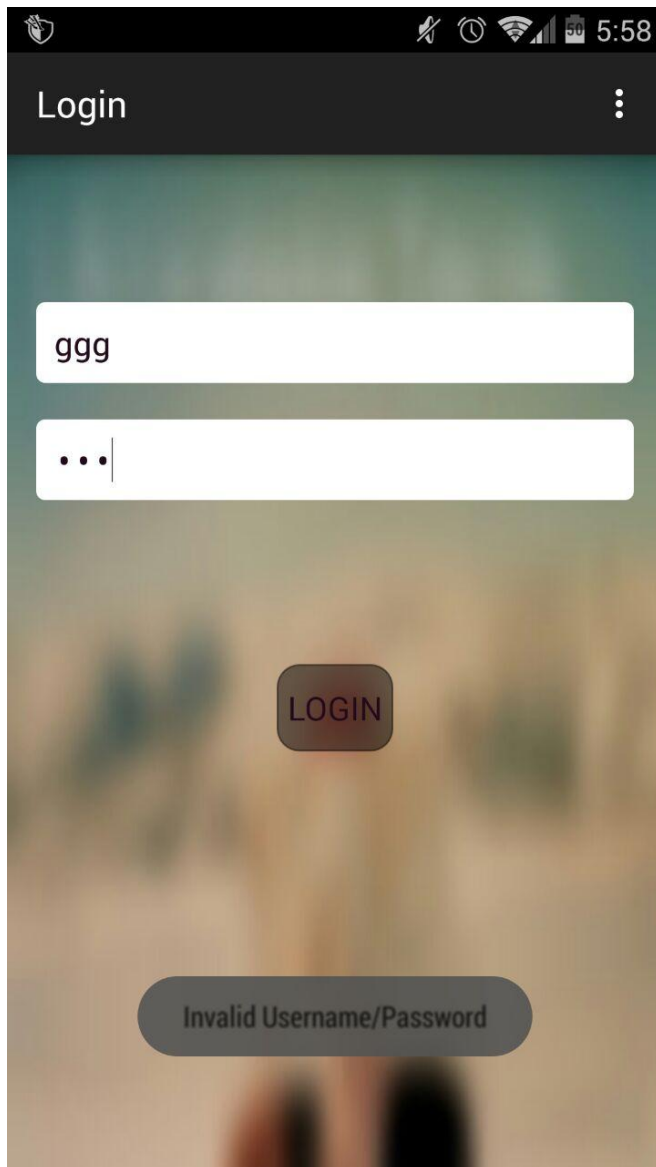
Back

Register

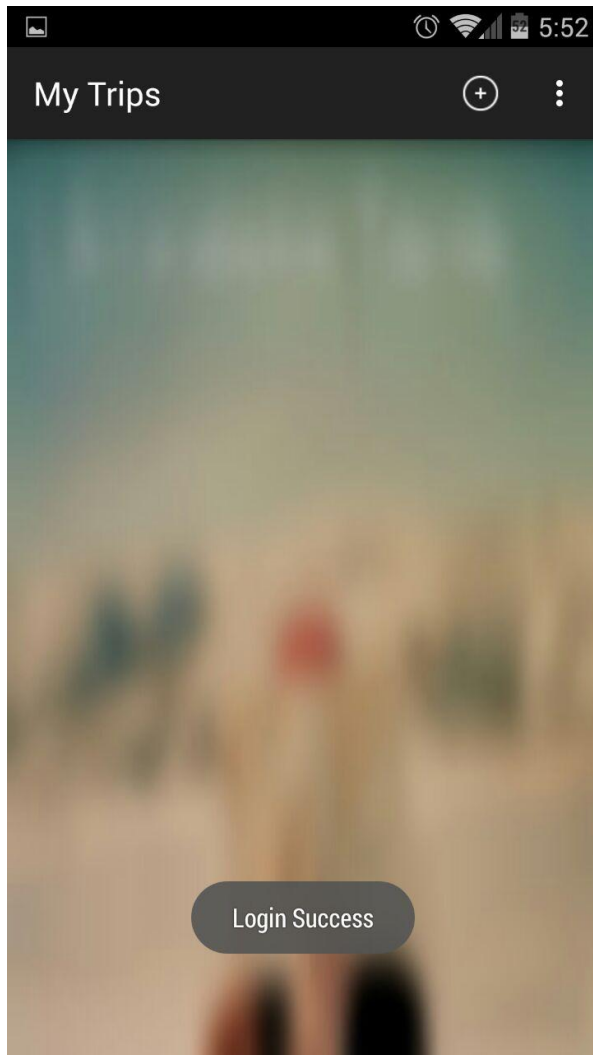
Please enter a first name



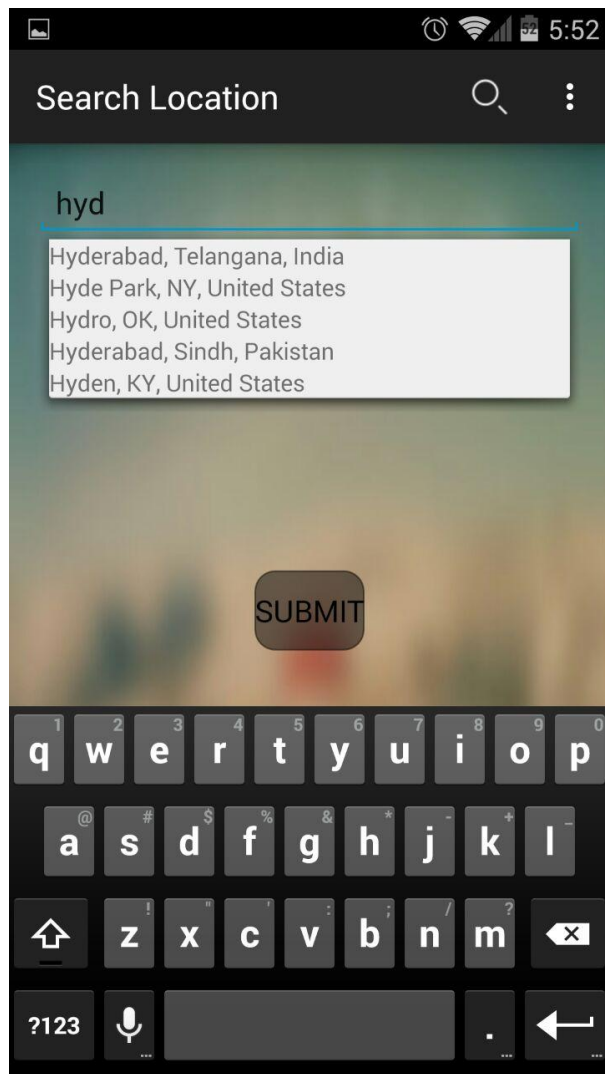
b. Login Page: So, after successful registration, user inputs the username which is an email and password with which he registered. If the details are correct, the MyTrips Activity is invoked. If the details entered are incorrect, a toast will be displayed as invalid username and password. When Login is clicked it calls the WCF service and it checks the username and password. Below is a screen that shows an error message.



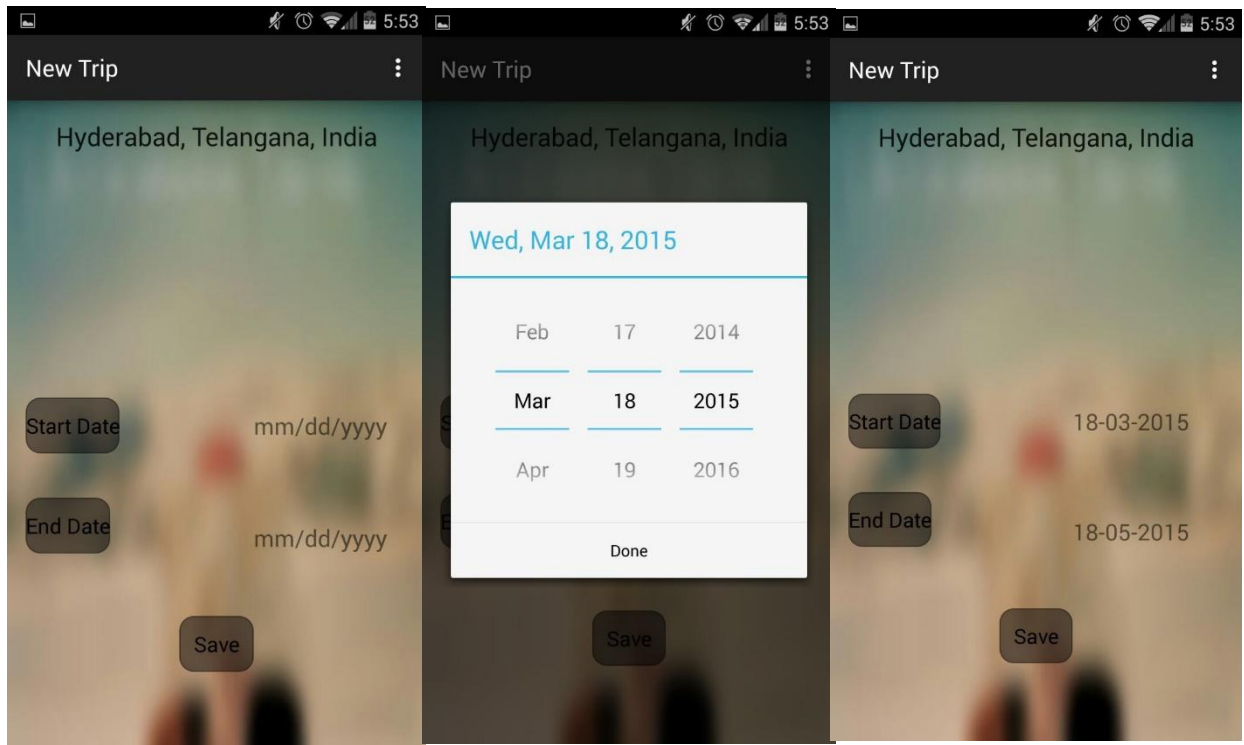
c. MyTrips Page: After successful login, a toast will be displayed as login success. And this activity has an option in the action bar with a plus symbol on it. When it is clicked it goes to another activity SearchLocationActivity. And after adding and saving a trip, the saved trips are shown in this page. These details are stored in the database along with the user id since these are referenced to a single unique user id. Below is the screen of this MyTrips Page.



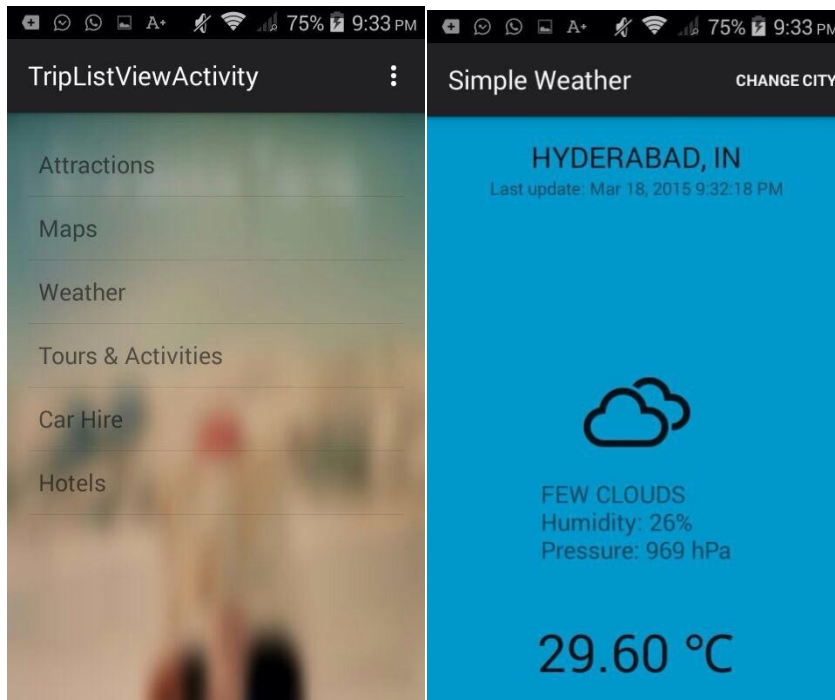
d. Search Location Page: This page has a search bar and on click on it enables the user to give an input. When user enters a minimum of one string, the search displays a list of different places related to the search string entered by the user. This process involves calling the google places API service and having the JSON object values and placing them in a listview. Below is the screenshot of this page showing some results when a string is entered.



e. NewTrip Page: This Page takes the place selected in the previous page and has options to select start date and end date. When the user clicks on the start date and end date, a dialog appears which is a date picker dialog. It is a tool in android. It allows user to select date with a good interface. When a date is selected, the dates are saved in the database and are shown the text view in that page. Below are the screens of the new trip page.



f. TripList Page and Weather Page: After selecting the place, the attractions, hotels, weather all in listview are displayed. And on click of weather list displays the current weather of the selected place. Below are the screens of the TripList Page and Weather Page.



4.3 Implementation of test cases

The Following Test cases are implemented in Nunit Tool.

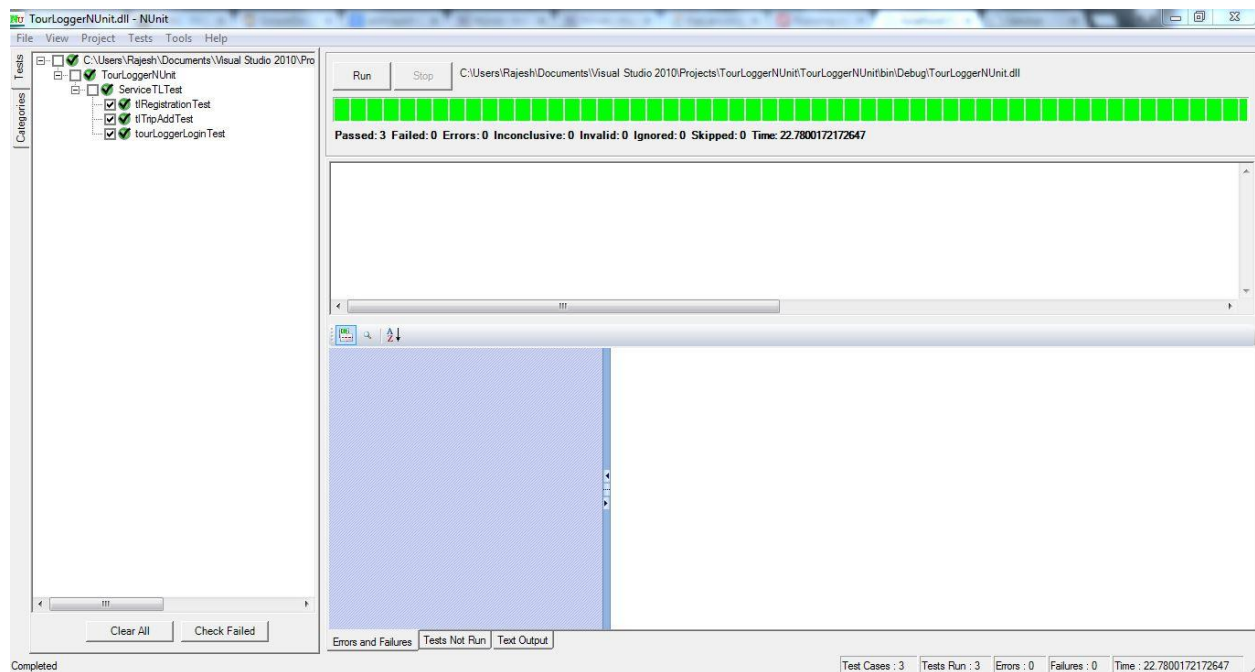
tourLoggerLoginTest() – To test whether the username, password details are present in user's table.

tlRegistrationTest() – To test insertion of user registration details into user's table.

tlTripAddTest() – To test insertion of trip details into my trips table.

5. Testing:

Below screenshot shows the test cases run in Nunit and the results.



6. Deployment:

6.1 Project ScrumDo link

<https://www.scrumdo.com/projects/project/asetourlogger/iteration/121729/board>

6.2 GitHub site URL

GitHub link: <https://github.com/rajeshkapa/ASETourLogger>

7. Project Management:

ScrumDo link: <https://www.scrumdo.com/projects/project/asetourlogger/iteration/121729/board>

Implementation status report:

Work Completed:

Description:

1. Expose WCF service to store user data in sql server
Responsibility: Login service by Raghu Vital, Register service by Rajesh
Time Taken: 20 hours
Contribution: 100%

2. Integrate Google API to get auto completion of text for cities search and store these details.
Responsibility: Rajesh
Time Taken: 20 hours
Contribution: 100%

3. Implement openweathermap to show weather of particular city on the trip day.
Responsibility: Raghu Vital
Time Taken: 20 hours
Contribution: 100%

Work To be completed: None

Issues/Concerns:

1. Issues faced during deployment of WCF service in remote server.
2. Issues occurred with openweathermap API. The url does not give results sometimes since this is a free account and there are limited requests available.

8. RESOURCES:

<http://openweathermap.org/appid#get>

<https://developers.google.com/places/documentation/>