**CS551 Advanced Software Engineering**

**Spring 2015**

**Project Title: PickMeUp**

**Submitted by**

PG6 (SG14 and SG15)

Ponnada Rahul (Class ID-39)

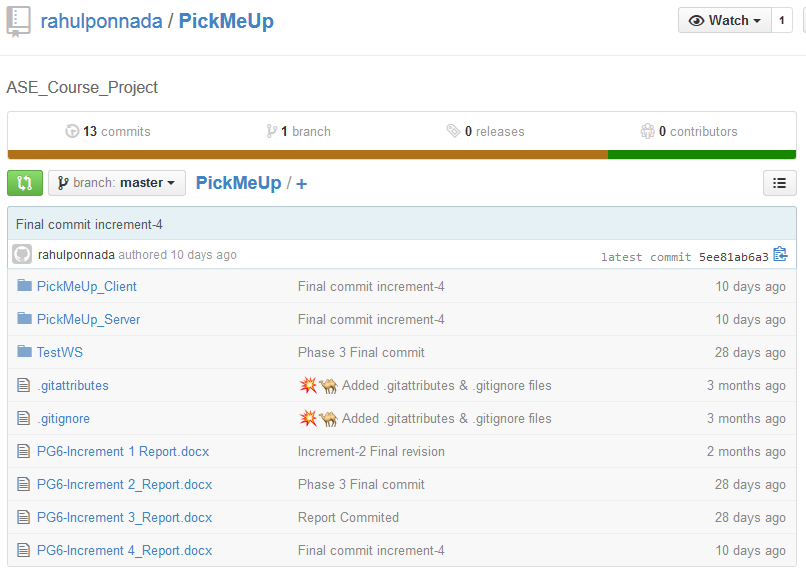
Ghanta Surya Prabha (Class ID-19)

Tummala Anvesh (Class ID-48)

Anumolu Satish Chowdary (Class ID-2)

1. **Project Deployment**

* **Source code:** <https://github.com/rahulponnada/PickMeUp>



* **Project video:** <https://www.youtube.com/watch?v=B10WNs5gBkk>

## 

## 

## PickMeUp Project - User Manual

**Contents**

[Introduction: 5](#_Toc419147138)

[Interaction with app: 6](#_Toc419147139)

[Sample Interaction: 7](#_Toc419147140)

[Bugs and Deficiencies: 8](#_Toc419147141)

# Introduction:

Once, as prospective master students coming to USA for the first time, we had undergone the unorganised, un automated and tedious process of following the google spread sheets for updating the travel information and checking the document multiple times a single day for any volunteers assigned from the university for the airport pickup. The harrowing experience inspired us to simplify and automate the process, making it easy for any future students coming to the country.

We feel there is a real need of this application for all universities at the start of every semester. We decided to materialize the idea through our android app “PickMeUP”. Our app enables the students, volunteers and admins of different universities to register and it assigns volunteers automatically to the registered students. Any changes made by students in their schedule are notified to the volunteers and vice versa. The specific objectives of the project include providing the common place to meet the volunteers and new coming students and to make the process of airport pickup easy. In addition, this application provides climate details and provides easy access of navigation to the respective destinations.

We feel there is a real need of this application for all universities at the start of every semester. This application makes the process of pickup so easy. As the new coming students do not make calls during flight journey, this application will help to be in contact to volunteers.

# Interaction with app:

A user is either a student, volunteer or admin can register with the PickMeUp application using their student ID’s. A student is a user who is fresher’s joining that semester to pursue their higher studies. A volunteer is user who takes the responsibility of picking the students from the airport and dropping them in their respective destinations. An admin is a user responsible for the respective university taking care of both students and volunteers.

**Student**

A student will initial register for the application providing their personal as well travel information like airlines, flight no, arrival date etc. to use the pickup service. On successful registration, the system automatically checks among the registered volunteers for the particular university and assigns the volunteer available during that time slot. An email notification will be sent to student, volunteer and admin regarding the new assignment. The student can login and see their personal details, Flight status, weather details of the particular day in the respective destination and assigned volunteer details on his home page. The student also has the feasibility to change the information provided at the time of registration later. In case of the change in the arrival date or time, the system again assigns a new volunteer based on the updated arrival timings. In this case, both the volunteer and the admin gets notified about the changes through e-mail. The student can check status of the flight and monitor the weather of the destination through the application.

**Volunteer**

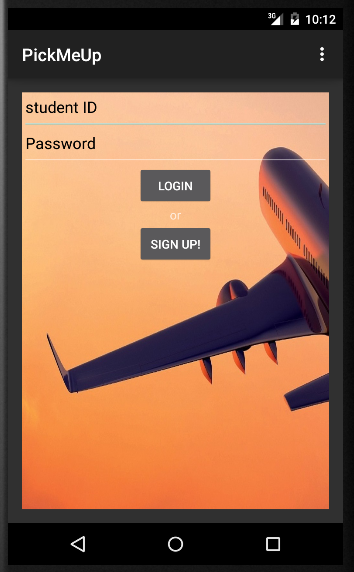
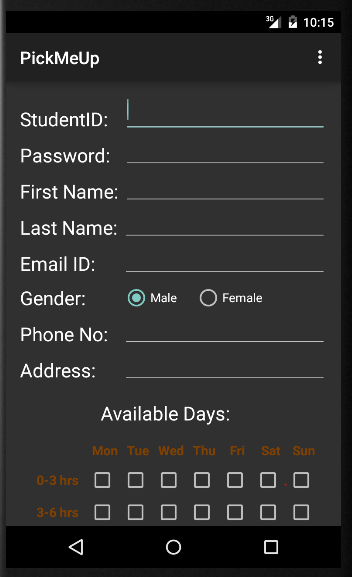
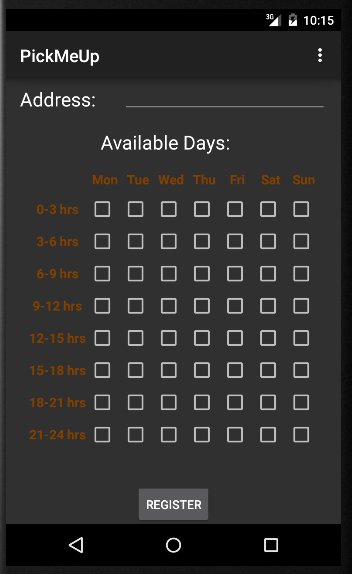
The volunteer will initially register for the application providing their personal as well as their available timings providing his availability in a day throughout the week. Upon successful registration, the system automatically assigns unassigned students (if any) to the volunteer based on his availability. An email notification will be sent to student, volunteer and admin regarding the new assignment. Once the volunteer logs in, he can see the list of students assigned to him on his home page. Each student is associated with a google navigate which will help through the navigation process in dropping the student in their respective destination address. The volunteer also has the feasibility to change the information provided at the time of registration later.

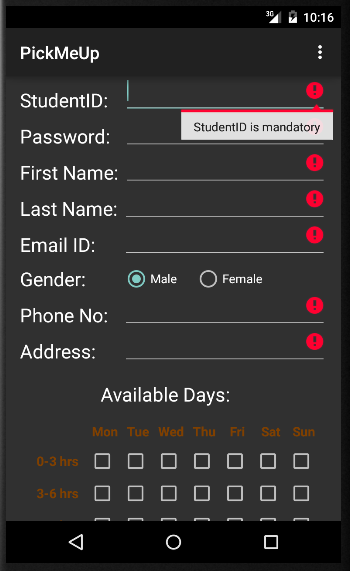
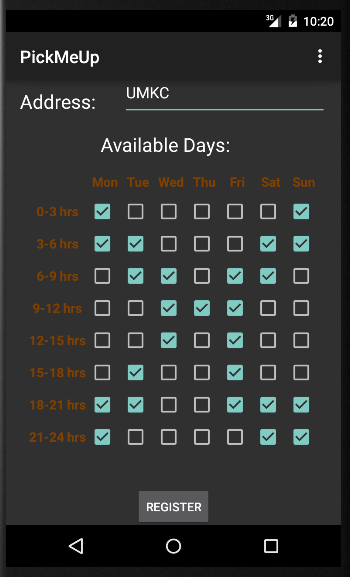
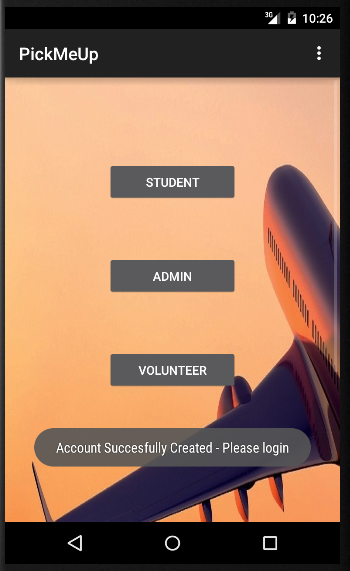
**Admin**

An admin will initially register for the application providing their personal as well as the university details. Upon registration and successful login, admin will be able to see the list of all volunteers of that university and students assigned for that respective volunteer. An email notification will be sent to student, volunteer and admin regarding the new assignment. Admin also has the feasibility to unassign any particular student from a volunteer. Admin can also make any announcements of any important updates to all the students and volunteers of that respective university. Upon successful announcement, an email will be sent to all the registered students and volunteers with the message.

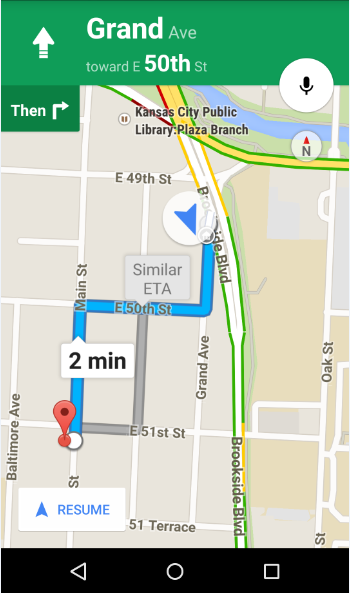
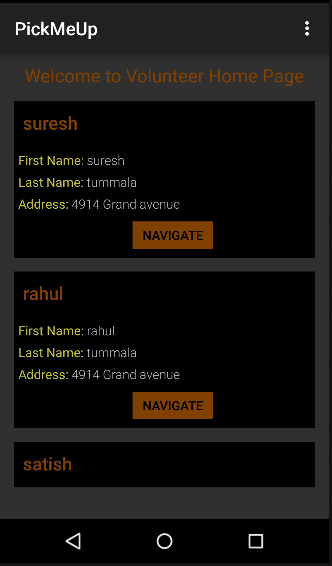
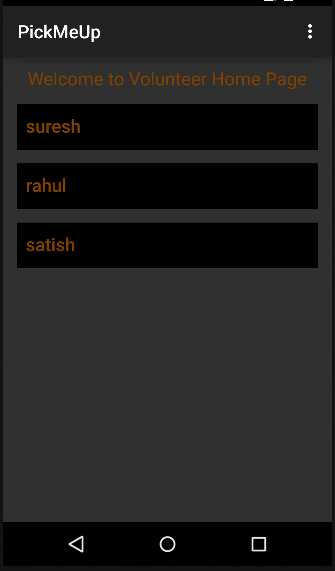
# Sample Interaction:

This is a sample interaction of volunteer with the PickMeUp application. User as a volunteer registers with the application. While registration he provides his available timings. Each day of the week is divided into 8 time slots. Validations will be there for all the required fields during the registration process. On successful registration, his account will be created and navigated to the main screen.

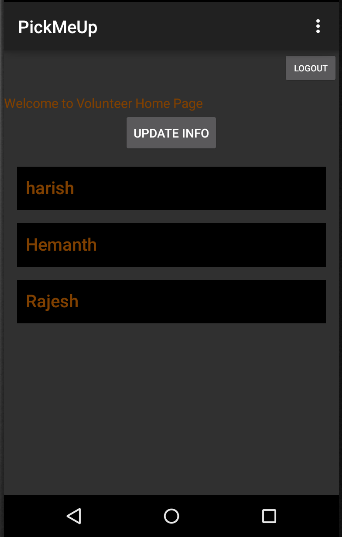
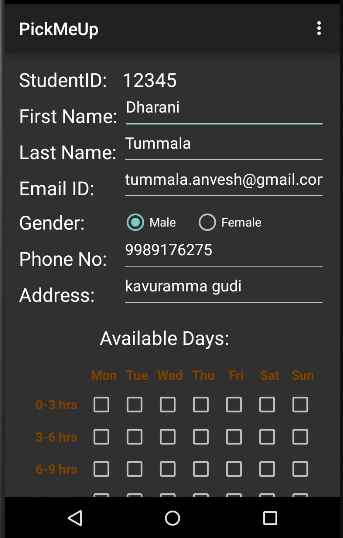
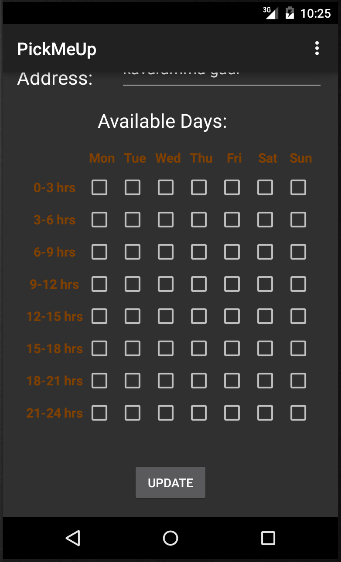
  

On login, he is taken to the home page where he can see the list of all students assigned to him. Upon expanding any student, a navigate button will be available to help the volunteers in the navigation process from the airport to the students drop address.



He can also update his personal and availability information using the update info button.

# Bugs and Deficiencies:

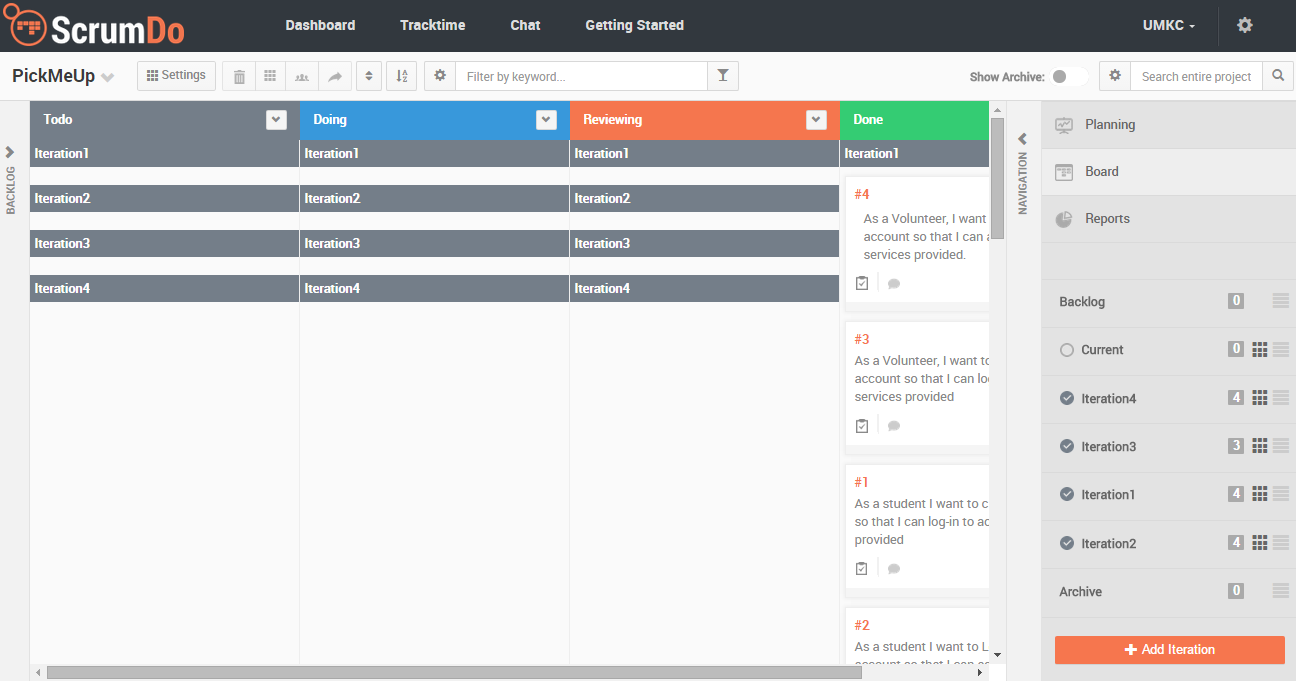
There are no bugs identified in our application. All the developed features are working properly without any issues.

1. **Project Management**

* **Project management report:**

We are a team of 4 who worked on the project called PickMeUp. We followed agile methodology to implement the project. So a basically an agile methodology works in iterations with stories in each iterations. Once they are completed they are called deliverable of that iteration. For this project we divided the time frame of 3 months into 4 iterations. Each iteration is having at least four stories to cover. Having said we are a team of 4, each member of the team took a story for each iteration and delivered that story at the end of the iteration.

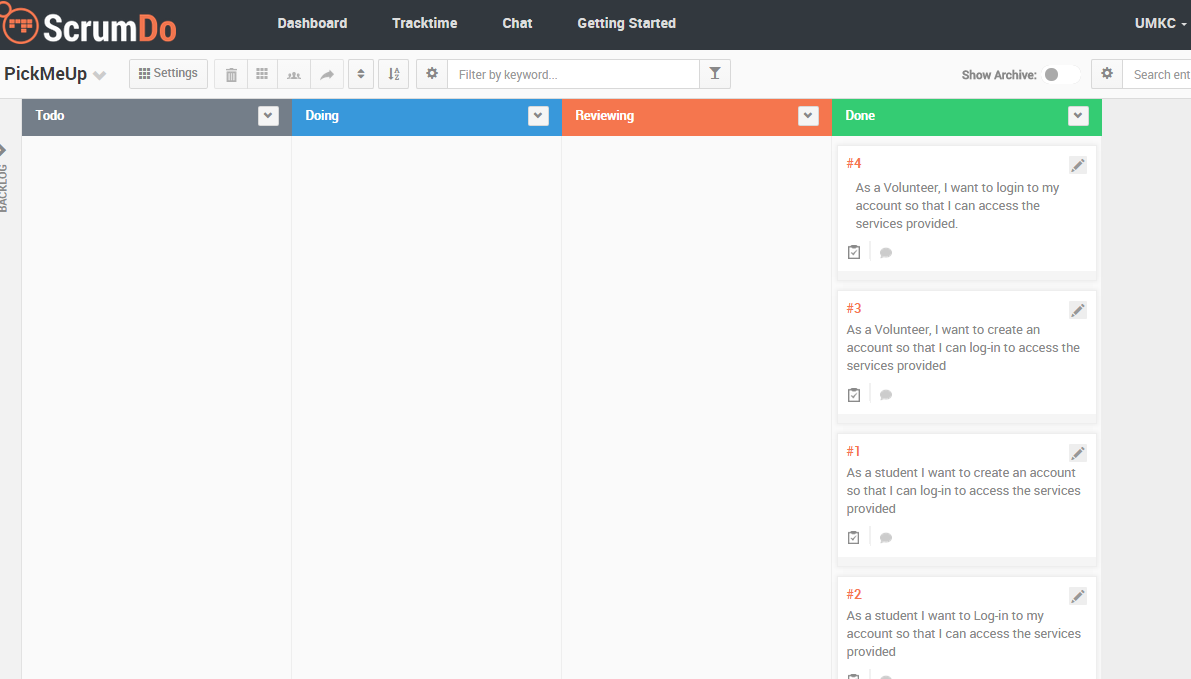
To keep track of all of these thing we used ScrumDo, an agile methodology tool which helped us to collaborate our work. The following screenshot is the dashboard our ScrumDo tool.



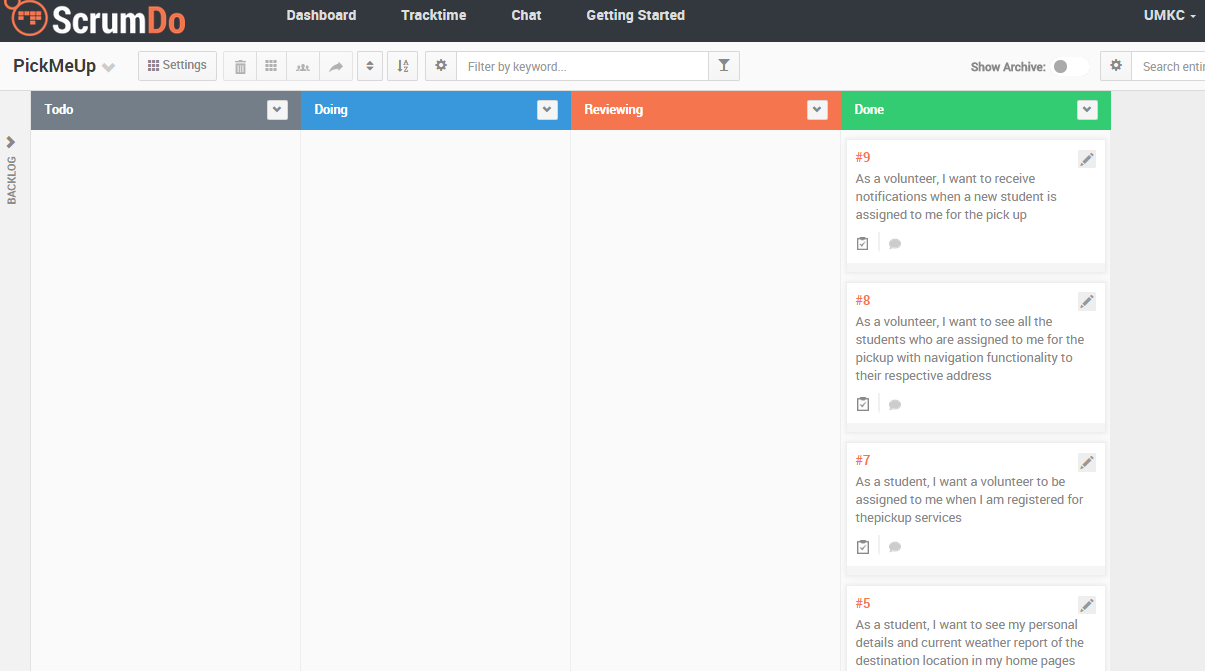
As you can see there are no stories left in any of the phases ToDo, Doing, Reviewing. All the stories have been completed and are in done phase. Important step in working teams is collaboration. ScrumDo has this feature to bring people together. So one can see the progress of others can comment on the stories. For each iteration each member of the team gets at least a story to develop. Usually an iteration span from 7 – 15 days. Once the stories are finalized for the project. These stories are scheduled to develop in each iteration. After the stories are placed in respective iterations, development has started.

Screenshots of all iterations can be viewed below.

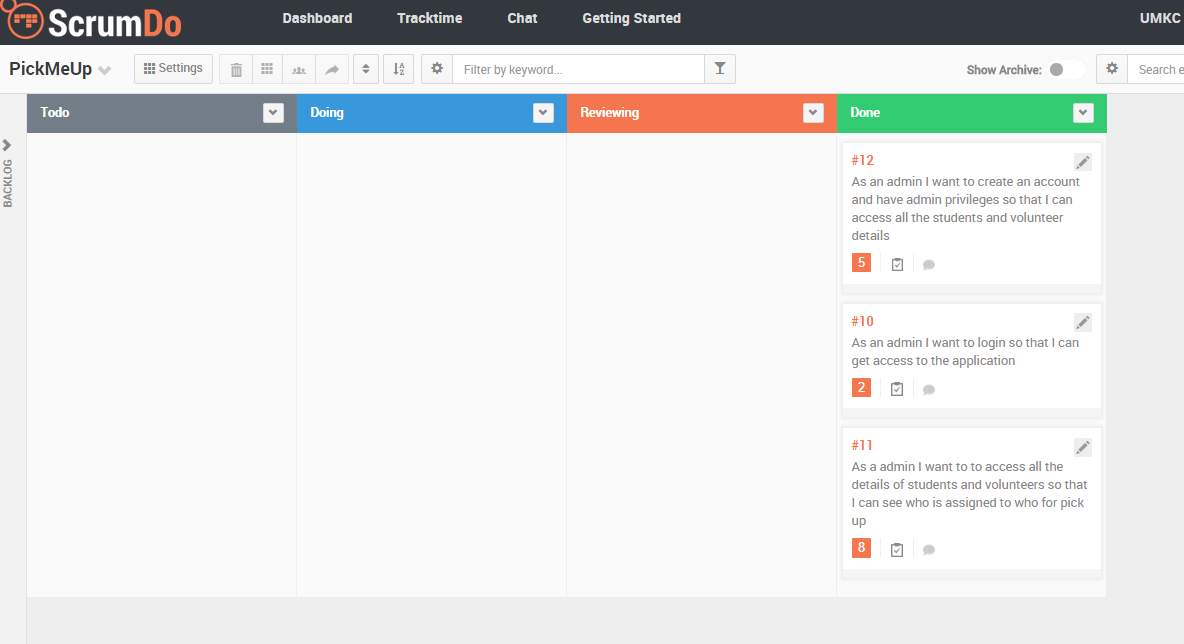
*Iteration-1:*



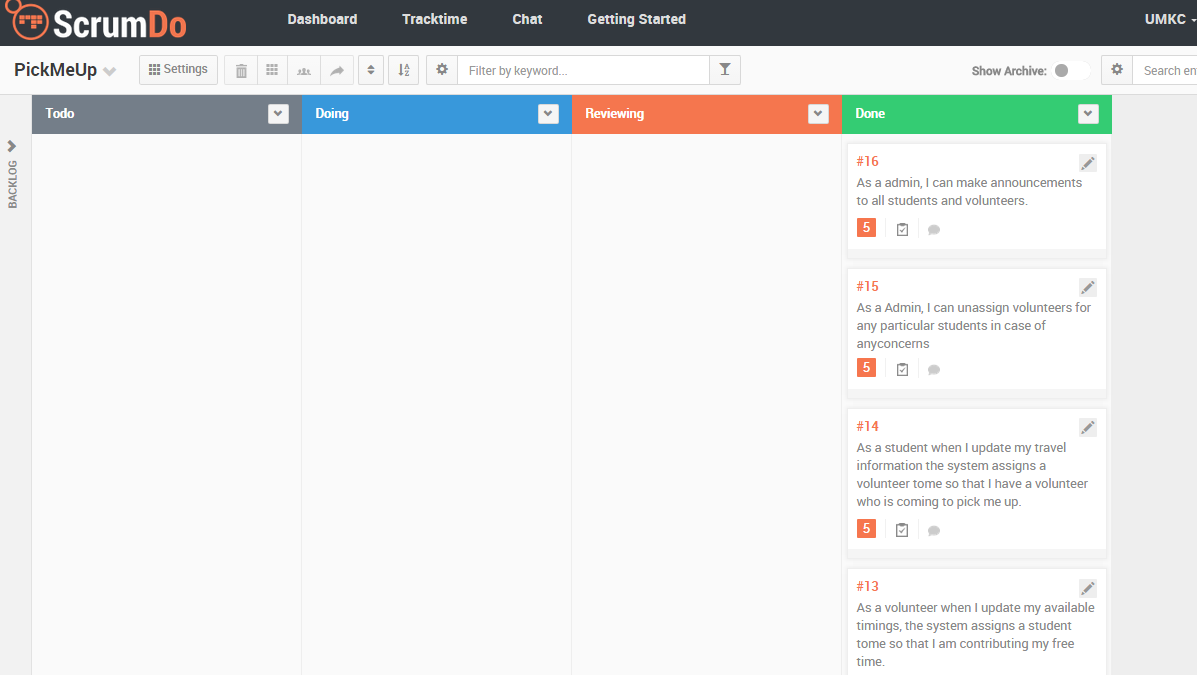
*Iteration-2:*



*Iteration-3:*



*Iteration-4*



Likewise every one worked on each story. We had team meetings at least 2 or 3 time in an iteration where in which we discuss the outcome of deliverables and any issues/troubles were addressed.

* **Final project evaluation:**

To evaluate our project we need to look at the stories that we created initially and what we have accomplished. There were lot of changes to be made to satisfy our requirements of stories. That’s the beauty of agile development it provides flexibility to change our stories based on the new requirement and helps in faster development. For we have accomplished what we intended to do. We were very much satisfied with design process and development phase. We met our original requirements and made some new additional requirements.

As I mentioned, agile helps in faster development with greater flexibility. I very much recommend agile development if anyone asks me for the development strategy. Next time, if I am asked to do an agile development I will make sure to have more team meetings and discuss the outcome, that way we can deliver better products. Although we have pretty much stick to our project schedule, there were hiccups due to some critical issues and were resolved promptly. Low level planning is the management strategy we have used within our group. Yes, it resembles to the structure we have planned. Better evaluation of stories ahead not just planning stories ahead improves the better handling of iterations. Coming to team we know each other pretty much well. That means we know each other strengths and weaknesses. So this kind of understanding helped to work efficiently. The recommendation for the next would be to have a close understating of each other and better scheduling of stories in iterations.

1. **Project Proposal:**

**Project name: Pick me up**

1. **Project Goal and Objectives :**

**Motivation :**

Currently there is no application for making the process of airport pick up easy. We traditionally follow the students filling of spreadsheets and manually assigning the volunteers. There are times a volunteer has to wait more than 3 hours in Airport because of flight delays. Therefore, we thought of an application that resolves all these problems.

**Significance :**

We feel there is a real need of this application for all universities at the start of every semester.

**Objectives :**

An android based application for the airport pick up of the students coming for an university. It will allow assigning volunteers to pick students and providing temporary accommodation if required.

**System Features :**

* New students has to register for pickup by providing their flight details, Phone Number, accommodation status, place to drop etc.
* A notification will be sent to admin once a new student is registered.
* Easy accesses for an admin to assign volunteers for picking up registered students.
* Flight delays can be monitored for every 5 minutes on the day of picking and the delay of flights can be notified to the volunteers.
* The climate details can be show at the time of the arrival using the respective APIs.
* Google maps are integrated to the application for easy access of navigation to the respective destinations.

1. **Project Plan:**

**I. Introduction:**

Currently there is no application for making the process of airport pick up easy. At the start of every semester, we traditionally follow the students filling spreadsheets and manually assigning the volunteers. There are times a volunteer has to wait more than 3 hours in Airport because of flight delays. Therefore, we thought of an application that connects the students, volunteers and make the pickup process easy.

**II. Project Goal and Objectives (revised)**

* **Overall goal :**

The main goal of this project is to build an android-based application for the airport pick up of the students coming for a university. It will allow assigning volunteers to pick students and providing temporary accommodation if required.

* **Specific objectives (problem statement)**

The specific objectives of the project include providing the common place to meet the volunteers and new coming students and to make the process of airport pickup easy. In addition, this application provides climate details and provides easy access of navigation to the respective destinations.

* **Significance**

We feel there is a real need of this application for all universities at the start of every semester. This application makes the process of pickup so easy. As the new coming students do not make calls during flight journey, this application will help to be in contact to volunteers.

**III. Project Background and Related Work**

We have come up with this project idea as we have seen the difficulty in the normal process of creating the excel sheets and updating manually. When we look for an application that help this process, we have not found any related applications.

**IV. Proposed System**

**1) Requirement Specification**

**Functional requirements:**

1. New students will create an account by entering their personal details like first name, last name, Phone number, email id and password.
2. Student will update the travel information if they need a pick up. Travel information includes baggage information, arrival date, flight details and accommodation address details.
3. Interested volunteers will register by providing their personal detail like first name, last name, Phone number, address, email id and password along with their available timings.
4. Once the new students are registered, system should assign the volunteers based on their available timings.
5. Admin should be able to see entire pickup information.
6. Volunteers should be able to see his assignee pickups.
7. Volunteers should be able to check the flight status all the time.
8. Volunteers should be assisted with google maps to navigate to respective addresses provided by the students.
9. Students should be able to see the weather information upon the time of their arrival.
10. Volunteers and students can tweet or post status in Twitter and Facebook about the application or any related information.

**Non-functional requirements:**

1. The system should assign the volunteers automatically without any redundancy.
2. All the validations should be done during the registration of students and volunteers.
3. The user interface should be very user friendly and easy navigable.

**Technical/business Requirements (prioritized)**

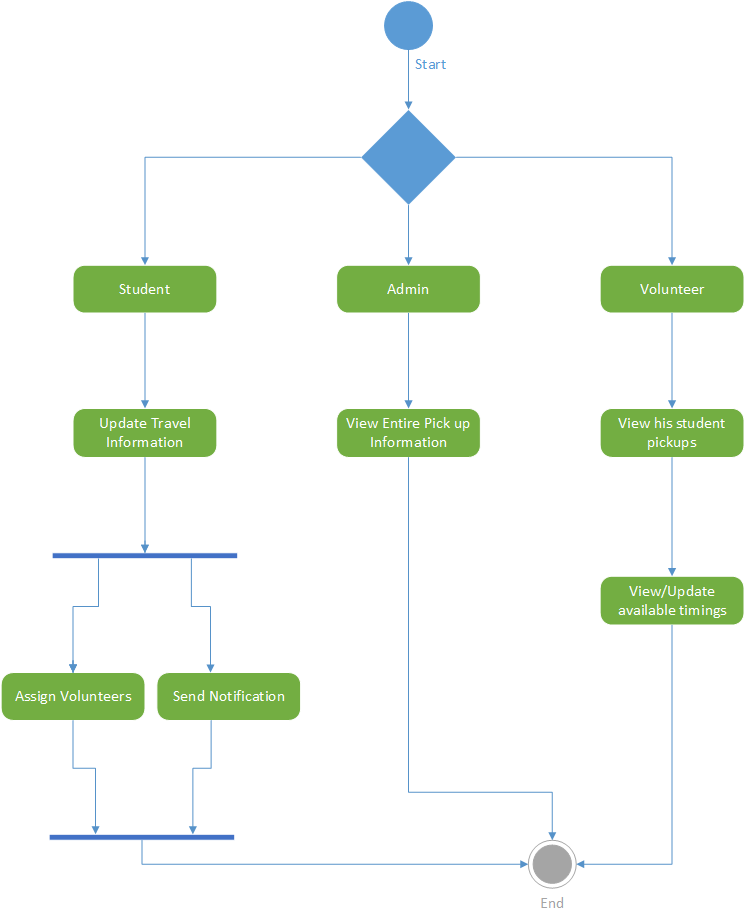
**System:** The system should assign volunteers without any redundancy.

**Students:** All the users should have the updated information time-to-time.

**Admin:** Admin should be able to change the assigned volunteers any time.

**Volunteers:** Volunteers should have the feasibility to change their available timings.

**Business Process/Workflow analysis (UML Activity Diagram)**



**Technological and Architectural requirements**

Required technologies are HTML5, JavaScript, and CSS for the user interface. AJAX is also used to get the asynchronous information like weather details and flight status information. Architectural requirements include a layered structure, which will be managing the workflow of the system. We require SQL database to store the information. A collection of web services is required to access data from different sources.

**2) Framework Specification: Build an overall system model**

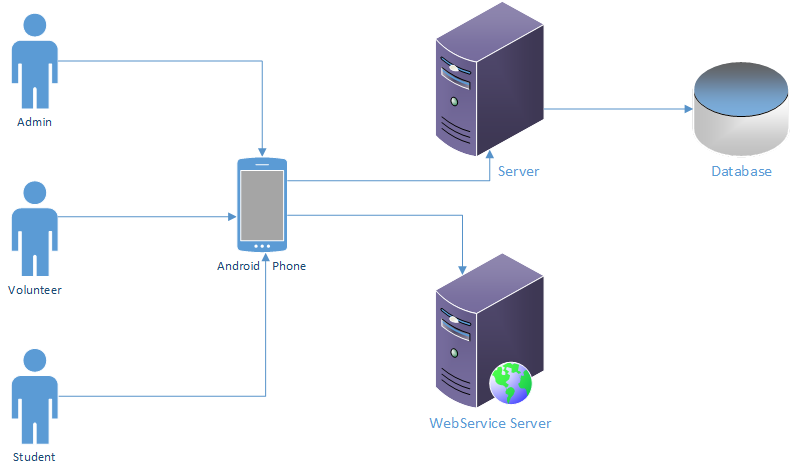
* **Assumptions and Principles**

1. The users are assumed to have an Android mobile to use the application.
2. Registration is restricted to UMKC students only.

* **Methodologies and Algorithms (if any)**

1. We are using scheduling algorithm to assign volunteers to students based on their availability.
2. This algorithm is also responsible for updating the volunteers based on the flight delays or any changes to available timings.

* **System Architecture Diagram**



**3) System Specification:**

* **Existing Services:**

1. Name: Google Maps API.

Description: Used to provide navigation for volunteers from their current location to airport Pickup location and to dropping location.

URL: <https://developers.google.com/maps/documentation/javascript/tutorial>

1. Name: Weather API.

Description: To get weather information.

URL: <http://openweathermap.org/api>

1. Name: Flight Status

Description: To get information about the flight status.

URL: <https://developer.flightstats.com/>

1. Name: Facebook

Description: To post updates on Facebook and for Login.

URL: <https://developers.facebook.com/docs>

1. Name: Twitter

Description: To tweet about their status on twitter.

URL: <https://dev.twitter.com/overview/documentation>

1. Name: Google Cloud Messaging for Android

Description: For notification about assigning volunteers and pickup students.

URL: <https://developer.android.com/google/gcm/index.html>

* **New Services to be built:**

1. **Scheduling the volunteers for pickups:** This service provides the scheduling of volunteers for the pickups based on match of available timing of volunteers and the pickup times.

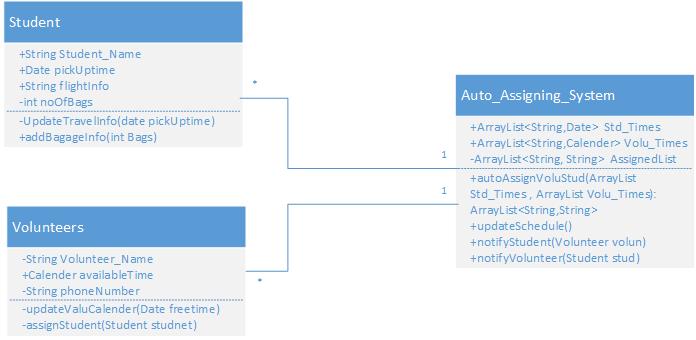
**Service Specifications:**

**Input:** The Pickup times of the newly coming students, available timing for volunteers.

**Output:** An efficient scheduling of volunteers for picking new students.

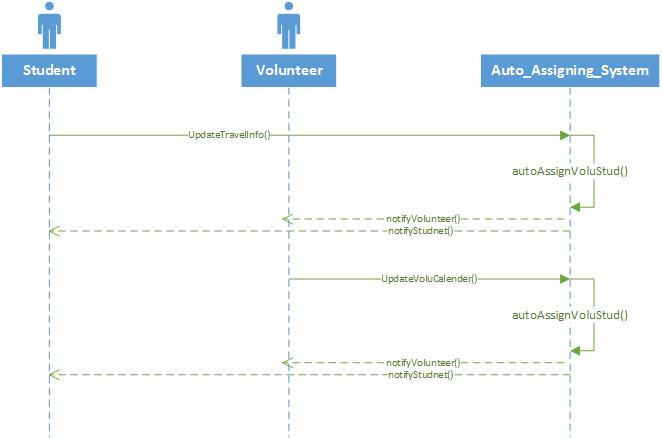
**Exception:** Ifthere are no volunteers available for picking a student, the system has to notify admin for assigning a volunteer available at that time.

**Class diagram**

****

**Class diagram for scheduling the volunteers for pickups**

**Sequence diagram**

****

**Sequence Diagram for the service of auto schedule**

* **Design of Mobile Client:**

**Features:** Register, Login, Request\_Pickup, Cancel\_Request, Navigation

**Styles:**  CSS

**Technologies**: HTML5

JavaScript

JQuery Mobile

Twitter Bootstrap

Java (Back end)

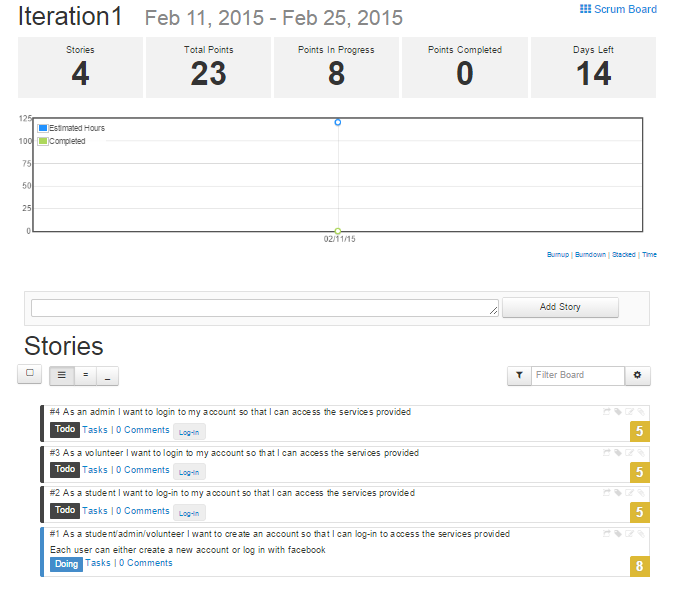
**Database:** Microsoft SQL Server 2008

**Tools:** Android Studio, SQL developer

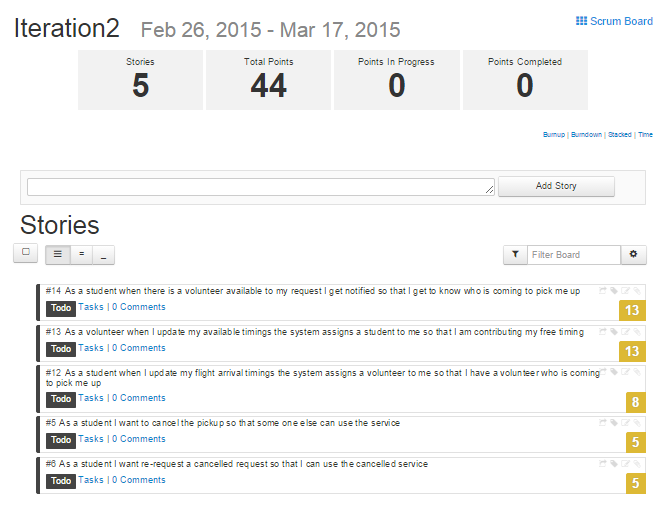
**V. Plan by Services (using ScrumDo)**

**Schedule for the four different increments**

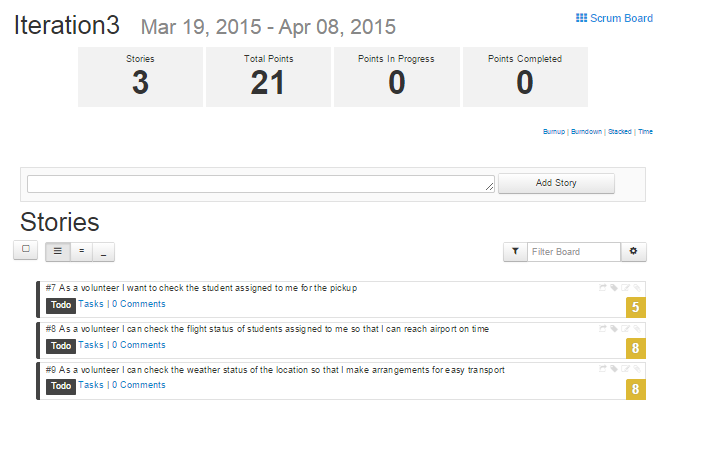
**Increment 1:**

****

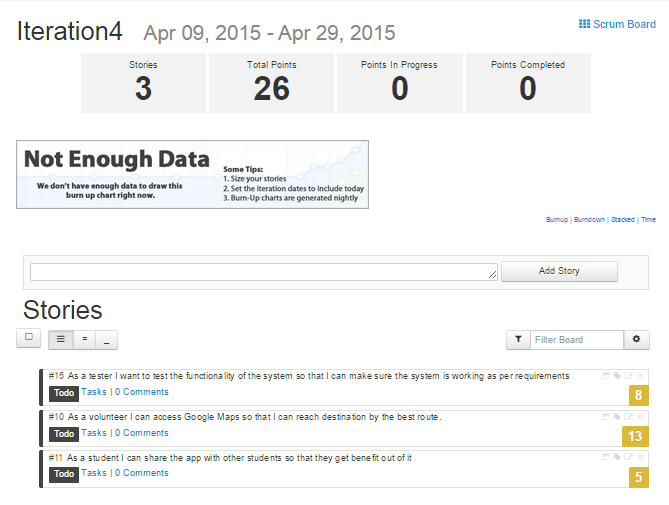
**Increment 2:**



**Increment 3:**



**Increment 4:**



**VI. Risk management**

Risks in a project should be identified and resolved by using some recovery methods. Minor risks involved in our project are:

* Users may forget their passwords.
* Validity of the volunteers is very important and that is taken care by admin.

**VIII. Bibliography**

* <http://www.w3schools.com/>
* <https://developer.flightstats.com/api-docs/flightstatus/v2>
* <https://developers.google.com/maps/documentation/android/>
* <http://openweathermap.org/api>
* <https://developer.android.com/training/index.html>

1. **First Increment:**

**Import Existing Services/API**

In this Increment, we implemented three web services each one for a different purpose. The web services are as follows -Student and volunteer login, student registration and volunteer registration. There are no external API used as of now. The main purpose of these web services is to validate the student and volunteers then register their details to a centralized storage.

-Login web service for students and volunteers to login

-Registration web service for students and volunteers to register with the app

**Detail Design of Services**

**User Stories:**

We have four stories in iteration1

1. As a student, I want to create an account so that I can log-in to access the services provided.
2. As a student I want to log-in to my account so that I can access the services provided.
3. As a volunteer, I want to create an account so that I can log-in to access the services provided.
4. As a volunteer I want to login to my account so that I can access the services provided.

**Service description:**

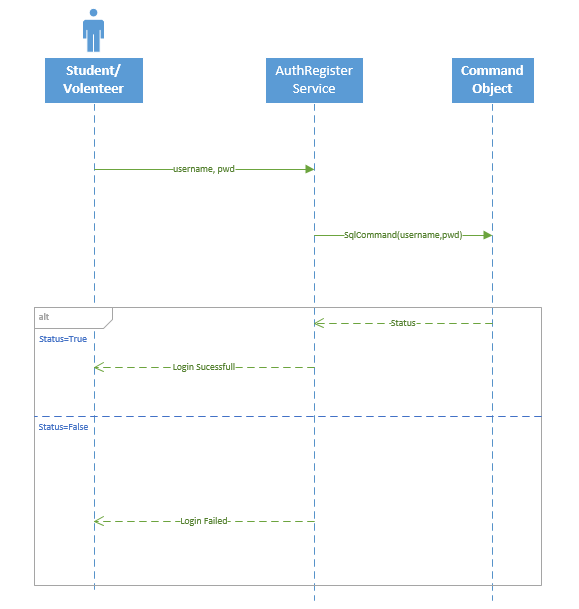
Login is a web service that takes *student id* and *password* as parameters. Then the service authenticates whether the specified student exists or not. If a student exists, then his password is authenticated against the password from his record from a centralized database and the return status as true, otherwise false. Hence, this service connects to the underlying database to fetch passwords of the respective users for validation.

Student and volunteer registration web services are pretty much similar except in the parameters they provide and underlying database table they access. It access the data provided by the respective user (Student or Volunteer) and records those data onto a respective database table and returns 1 for successful registration.

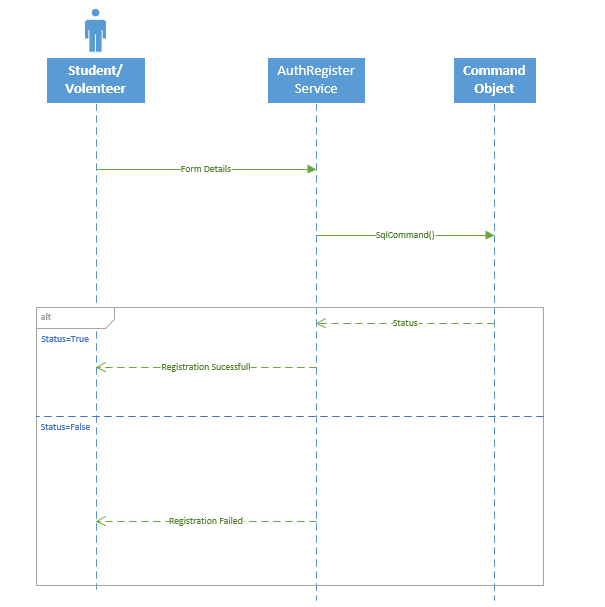
These web services return JSON data back to the client to verify the operation has successfully completed on the server side. REST uses JSON as it data exchange format so as in here for all the web services. Microsoft SQL server is the persistence storage that these services store data on to table and retrieve them for later verification, validation and population purposes.

**Sequence diagrams:**

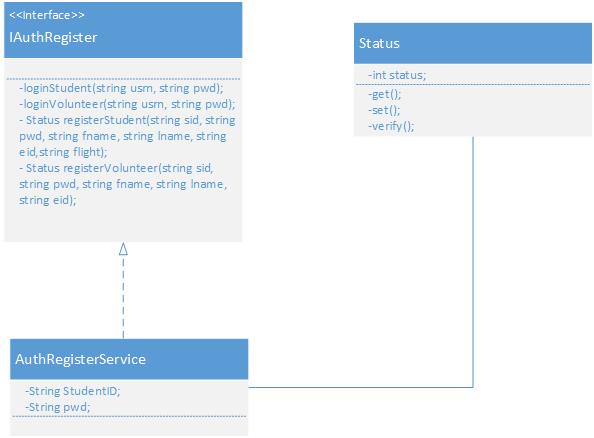
**Student/Volunteer Login**



**Student/Volunteer registration:**

****

**Class Diagram:**

****

**Design of Mobile Client Interface:**

Mobile client interface is an Android application which is a rich client. An application is considered rich client if it has all the UI required on client’s side. Overall UI design is developed using Android studio. We are using the Base.V21.Theme.AppCompat theme for our application and Nexus 5 API 21 as emulator for testing our application. As of now, for the first iteration, app consists of six different screens.

1. Start Screen which prompts user to select student or volunteer.
2. Login Screen for student and volunteer.
3. Student Registration screen.
4. Volunteer Registration screen.
5. Student home screen after successful login
6. Volunteer home screen after successful login.

User Interacts by a touch based smartphone there by navigating to other screens and perform operations on the server. Typical mobile client flow of operations is as follows. When the student or a volunteer install the app, they are asked to identify themselves (to distinguish between student and volunteer). Then they are redirected to a login page where already existing users can login and new users can register. After successful registration, users are navigated back to the Start screen form where they can login to their respective home pages (Student Home Screen & Volunteer Home Screen).

**Design of Unit test cases (using NUnit tool):**

Test cases are designed to test the Login and Registration services. This is implemented using visual studio and executed by NUnit Client. Our test case consists of four methods to test the Student login, Volunteer Login, Student Registration and Volunteer Registration functionalities.

**Implementation**

**Implementation of REST services:**

WCF (Windows Communication Framework) is used to implement REST web services on Visual Studio 2010. Web service project has an endpoint IAuthRegister.cs, which is also called as contract and Implementation of these resources is in AuthRegister.svc.cs. The implementation has several resources implemented and are ready to be consumed from a client. Resources communicate directly with the underlying database.

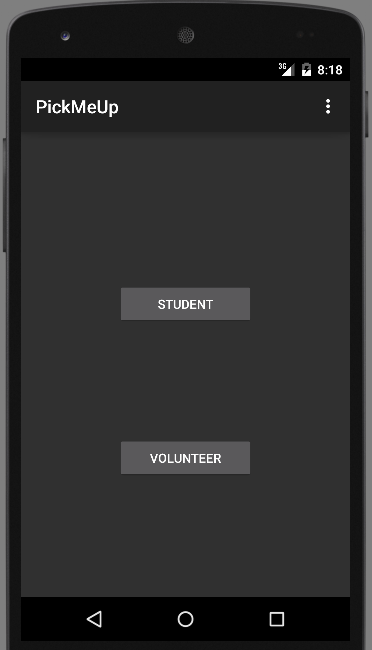
Login service validates existing student or volunteer by the taking the username and password from the mobile client side and runs a select query on the respective table. Response will be returned by the service as true if the login is successful or false otherwise.

Registration services saves student and volunteer account information in the system. Both the services will take the respective account information from the Mobile client side and Updates the respective student or volunteer tables respectively.

**Implementation of user interface (Mobile Apps):**

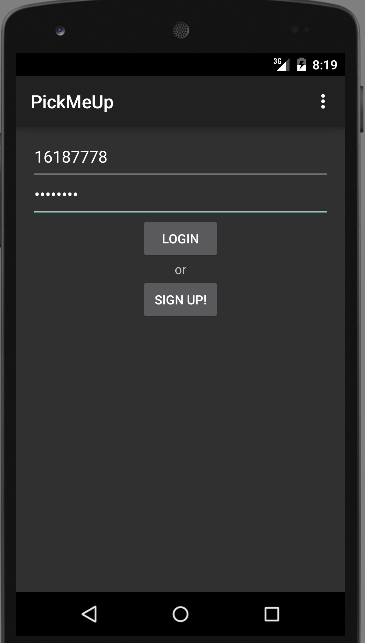
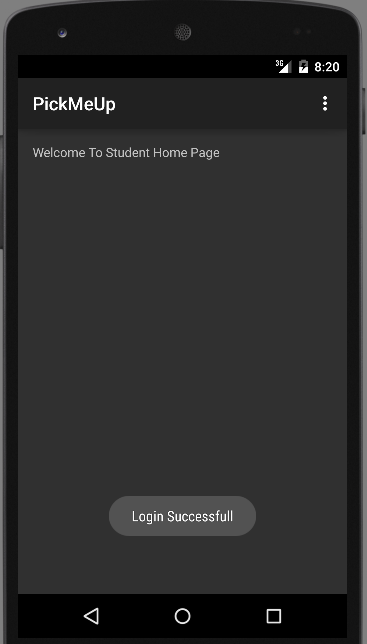
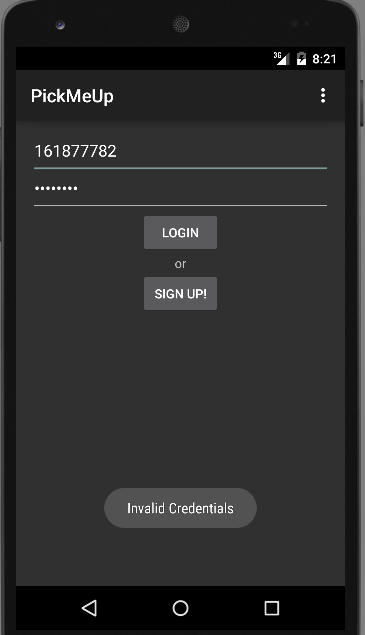
Android studio is being used to implement the Mobile App. User Interface of the app is XML based and is relatively changeable to the screen size. A total of six activities are created for the six screens designed for this increment.

1. **Start Screen:** It is the MainActvity consists of two buttons student and volunteer. Upon click of any button, it will navigate to Login screen and will also pass the respective button name to distinguish between student and volunteer for the next levels of navigation from login screen.

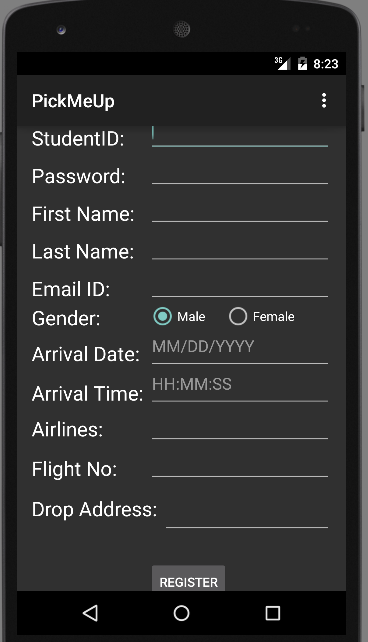
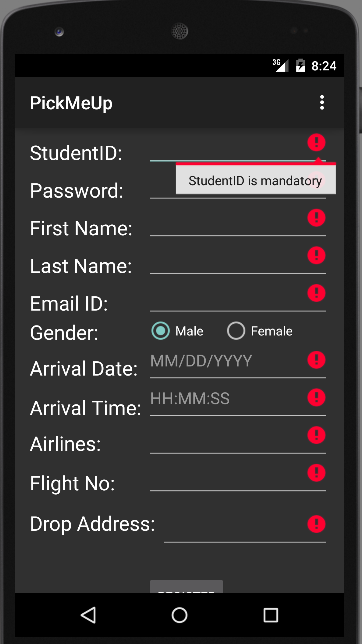
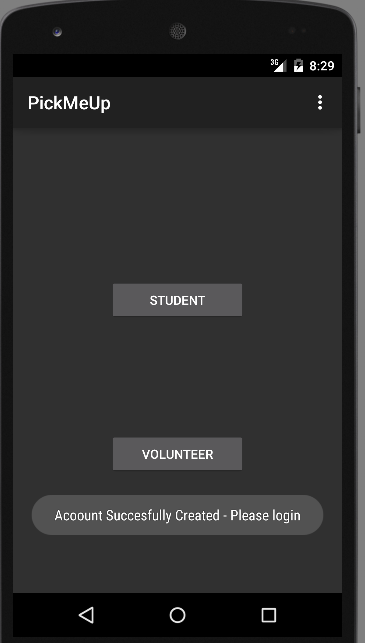


1. **Login Screen:** It is the LoginActivity which consists of StudentID, Password fields and Login, Sign Up buttons. Existing users will navigate to their respective Home screen by the providing the login details. A new private class AuthenticationService is written inside the LoginActivity which uses HTTP request to call the Login Service from the Mobile client side. Upon click of the login button, respective login service is called to validate the login credentials and response of true or false is returned. If true is returned, a Toast message “Login Successful” is displayed and the user will be navigated to the respective Home screens. If false is returned, a Toast message “Invalid Credentials” will be displayed.

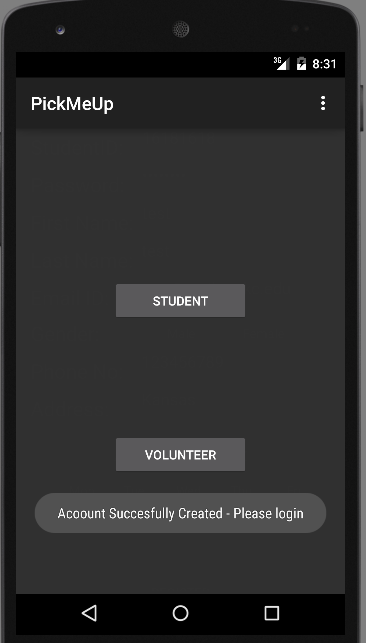
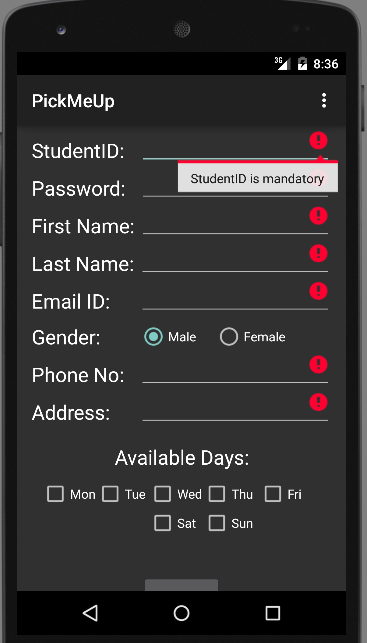
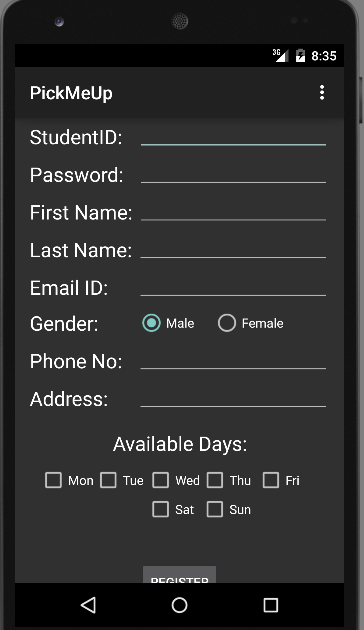
Upon click of the signup button, users will be navigated to Student Registration screen or Volunteer registration screen respectively.

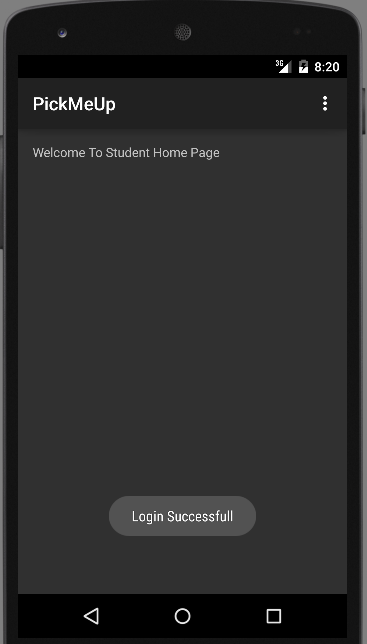
1. **Student Registration Screen:** It is the StudentRegActivity which consists of fields like StudentID, Password, FirstName, LastName, Email ID, Gender, Arrival Date, Arrival Time, Airlines, Flight No and Drop Address to create a student account. Here the studentID field is numeric type, password is Password type, Gender is Radio Button type, Arrival Date and Time are of Date type, Airlines is autocomplete Text view type fields and remaining fields are normal text fields. Many of these are mandatory fields and Validated after the click of Register button. A new service StudentRegistration is written inside the StudentRegActivty which uses HTTP request to call the Student Registration service from the mobile client side. Upon validation of validation of required fields, this service will be called to create an account for the student. A new Toast message “Account Created Successfully – Please Login” will be displayed to the user and is navigated to the Start Screen.

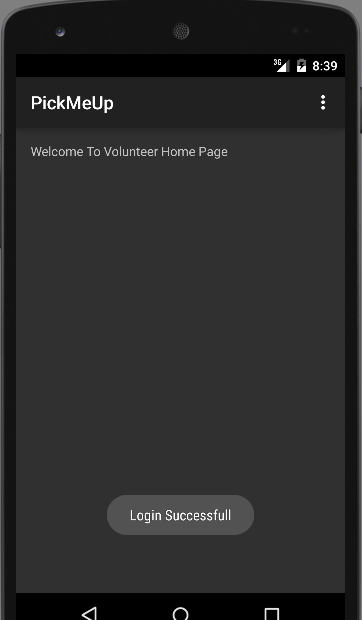
1. **Volunteer Registration Screen:** It is the VolunteerRegActivity which consists of fields like StudentID, Password, FirstName, LastName, Email ID, Gender, Phone No, Address and Available Days fields to create a volunteer account. Here the studentID field is numeric type, password is Password type, Gender is Radio Button type, Available Days consists of Check Boxes for all the days of the week and remaining fields are normal text fields. Many of these are mandatory fields and Validated after the click of Register button. A new service VolunteerRegistration is written inside the VolunteerRegActivty which uses HTTP request to call the Student Registration service from the mobile client side. Upon validation of validation of required fields, this service will be called to create an account for the student. A new Toast message “Account Created Successfully – Please Login” will be displayed to the user and is navigated to the Start Screen.



1. **Student Home Screen:** It is the StudentHomeActivity which consists of simple text message “Welcome to Student Home Page”. This screen will be designed in the next increments.



1. **Volunteer Home Screen:** It is the VolunteerHomeActivity which consists of simple text message “Welcome to Volunteer Home Page”. This screen will be designed in the next increments.

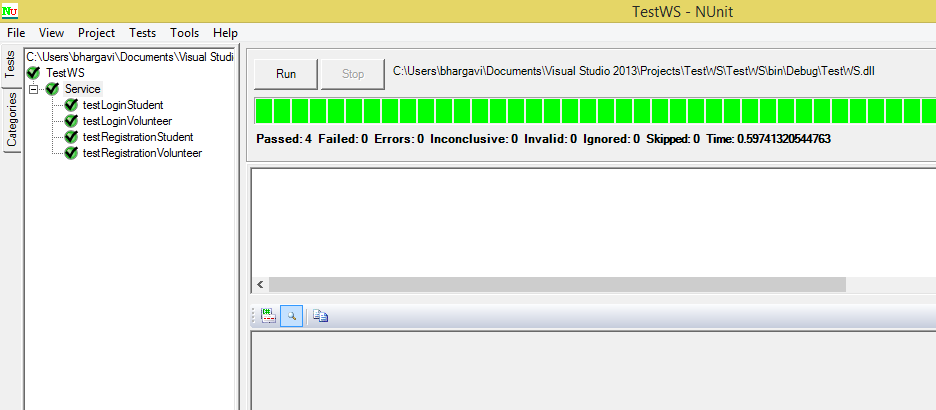


**Implementation of test cases:**

Test cases are implemented by using visual studio and executed by NUnit Client. We have a NUnit test class Service. It consists of four test methods, testLoginStudent- to test the student Authentication process, testLoginVolunteer - to test the Volunteer Authentication process, testRegistrationStudent – to test the student Registration process and testRegistrationVolunteer - to test the volunteer Registration Process.

**Testing: Perform Unit testing (using NUnit tool)**

All the implemented tests are executed using NUnit Client and all the four test cases are passed.

****

**Deployment:**

**WebService URLs**

**Student Login:**

[**http://kc-sce-cs551.kc.umkc.edu/aspnet\_client/Group6/PickMeUpService/AuthService.svc/student/login?username=12345&passwd=erty**](http://kc-sce-cs551.kc.umkc.edu/aspnet_client/Group6/PickMeUpService/AuthService.svc/student/login?username=12345&passwd=erty)

**Volunteer Login:**

[**http://kc-sce-cs551.kc.umkc.edu/aspnet\_client/Group6/PickMeUpService/AuthService.svc/volunteer/login?username=12345&passwd=erty**](http://kc-sce-cs551.kc.umkc.edu/aspnet_client/Group6/PickMeUpService/AuthService.svc/volunteer/login?username=12345&passwd=erty)

**Student Registration**

[**http://kc-sce-cs551.kc.umkc.edu/aspnet\_client/Group6/PickMeUpService/AuthService.svc/register/volunteer?studentid=123453&passwd=qwerty&firstname=george&lastname=bush&email=bush@whitehouse.com&gender=M&phone=8164821266&address=WashingtonDC&available=TuThFrSaSuMoWe**](http://kc-sce-cs551.kc.umkc.edu/aspnet_client/Group6/PickMeUpService/AuthService.svc/register/volunteer?studentid=123453&passwd=qwerty&firstname=george&lastname=bush&email=bush@whitehouse.com&gender=M&phone=8164821266&address=WashingtonDC&available=TuThFrSaSuMoWe)

**Volunteer Registration:**

[**http://kc-sce-cs551.kc.umkc.edu/aspnet\_client/Group6/PickMeUpService/AuthService.svc/register/volunteer?studentid=jtykkjdkj&passwd=cdkkjs&firstname=dkjsj&lastname=dskjksj&email=dsdsjhjskj&gender=Mnmdc&phone=dhshjk&address=dsdsas&available=jksak**](http://kc-sce-cs551.kc.umkc.edu/aspnet_client/Group6/PickMeUpService/AuthService.svc/register/volunteer?studentid=jtykkjdkj&passwd=cdkkjs&firstname=dkjsj&lastname=dskjksj&email=dsdsjhjskj&gender=Mnmdc&phone=dhshjk&address=dsdsas&available=jksak)

1. **Second Increment:**

**Import Existing Services/API**

In this Increment, we implemented three web services and have used two external services. The web services we implemented are as follows -Student registration a modified version including assignment of volunteers of previous increment, SendNotifications to send notification emails, IntelligentSysytem that assigns Volunteers to students based up on the time availability. The external APIs used are Weather API for Climate details, Google Map API for Navigation. The main purpose of these web services is to assign the students with volunteers based on their matching schedules and sending notifications to Volunteers when a student has been assigned.

-Student registration (Modified) service for students to register and then it uses IntelligentSysytem to assigns volunteers and gives the assigned Volunteers name after successful registration.

- IntelligentSysytem service is for assigning Volunteers to students based on their matching times.

- SendNotifications service is for sending notifications to Volunteers when a student has been assigned.

- Weather API used for showing the climate details of the arrival location for students in their home page.

- Google Map API used by Volunteers for navigating to the respective student specified destination locations.

**Detail Design of Services**

**User Stories:**

We have five stories in iteration2

1. As a volunteer, I want to receive notifications when a new student is assigned to me for the pick up.
2. As a volunteer, I want to see all the students who are assigned to me for the pickup with navigation functionality to their respective address.
3. As a student, I want an volunteer to be assigned to me when I am registered for the pickup service.
4. As a student, I want to see my personal details and current weather report of the destination location in my home page.

**Service description:**

*Intelligent system* is a web service that is the core component of this project. The main task of this system is allot volunteer to students upon their arrival timings.

Assigning a volunteer is a part of the web service that is invoked when a volunteer is registered or change his available timings. This service is invoked after the volunteer has successfully registered there by checking his availability timings against student arrival timing. The algorithm that we developed here follow certain rules in assigning a volunteer to the student.

Assigning a student is also a part the of web service that is invoked when the student gets registration. His arrival timing is checked against the volunteer availability. If any volunteer is available he is assigned to that volunteer.

These web services return JSON data back to the client to verify the operation has successfully completed on the server side. REST uses JSON as it data exchange format so as in here for all the web services. Microsoft SQL server is the persistence storage that these services store data on to table and retrieve them for later verification, validation and population purposes.

**sendNotification:**

sendNotification service will send email notifications to volunteer and users regarding their pickup. When a student is assigned to a volunteer. Both receive an email notification of the service. Email notification can be set to different services such as registration, change in timings and update to a new volunteer and so on.

**IntelligentSysytem:**

IntelligentSysytem service will assign volunteers to students based on the available times of Volunteers and students arrival times. If there are is no volunteer available for the pickup of a student, it will send email notification to Admin to have volunteer for picking that student. The algorithm we followed to assign Volunteers is as follows.

**Algorithm**

We have come up with our own algorithm to assign volunteers for picking students. We have taken input for the available times of Volunteers for a week, assuming he will have a recursive schedule that will be same for all weeks. For each day in the week, we have taken 8 bit input, each bit refers to his availability of every 3 hours. So, the volunteers 24 hours availability is taken input for each 3 hours i.e. {0-3, 3-6, 6-9, 9-12, 12-15, 15-18, 18-21, 21-24}.

**Requirement1:** The Volunteers availability is to be stored in DB as follows, if a Volunteer is available on Monday, Tuesday, Saturday and if he is available from 12-18h on Monday, from 21-24h on Tuesday, fully available on Saturday, he will input his availability as (48-00110000, 128-10000000, 0, 0, 0, 255-11111111, 0). This should be stored in DB as availability of Volunteers.

**Requirement2:** The arriving time of the students is to be stored in the DB as DateTime format.

**Input:** String: StudentID

**Step1:** Extract the arrival time of the student from DB based on input StudentID.

**Step2:** Get the dayOfWeek of arrival time of student, let say Sunday.

**Step3:** Look for the time slot student is coming. I.e is he is coming at 11:15 AM the his daySlot will be 8 - (00001000) bit 1 at (9-12 time period).

**Step4:** From the details of the dayOfWeek and time slot, query the Volunteer database for the week schedule of dayOfWeek

such that Get TOP of VolunteerAvaialability &(bitwise AND) daySlot !=0 AND sort by noassignedstudents ASC.

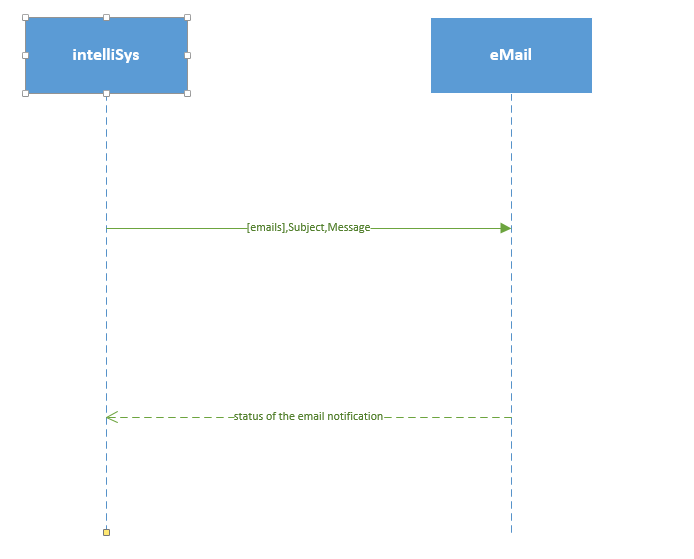
Then we will get the Volunteer who is available for the pickup of student having less no of students being assigned.

**Step5:** Notify the Volunteer about the assigned student details.

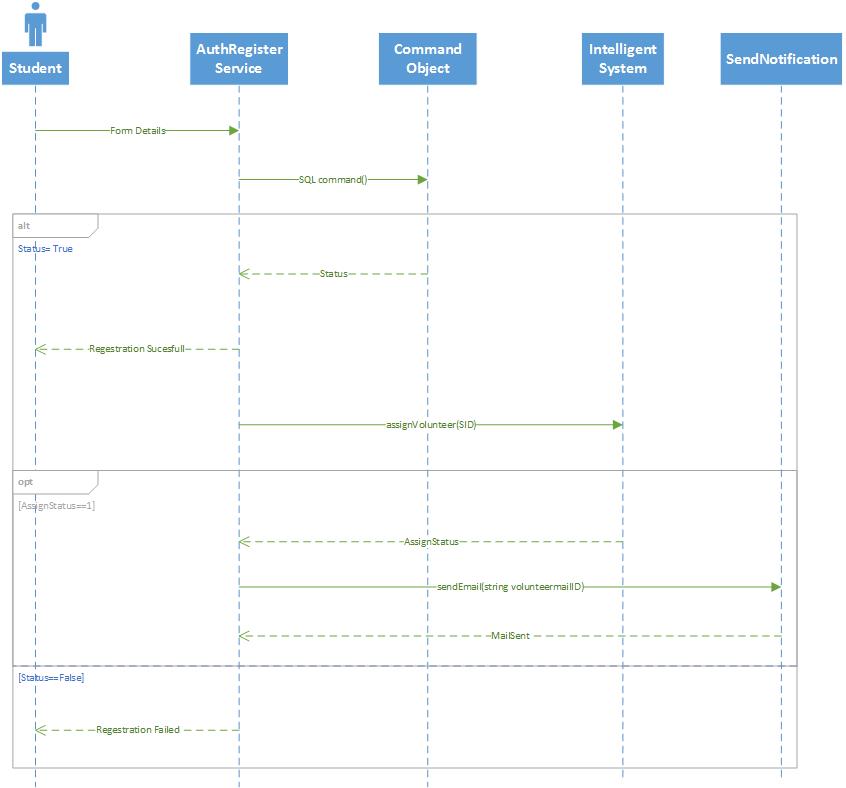
**Step6:** Return status.

**Sequence diagrams:**

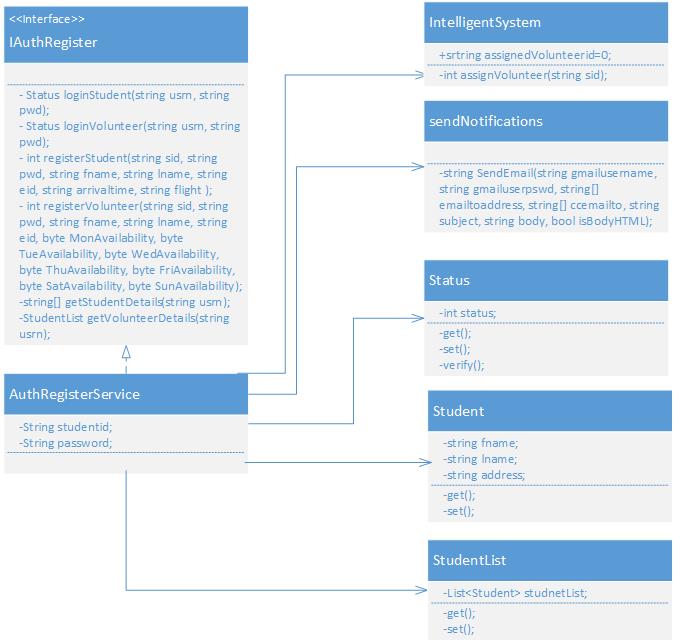
**EmailNotification:**



**Student Registration**



**Class Diagram:**

****

**Design of Mobile Client Interface:**

Mobile client interface is an Android application which is a rich client. An application is considered rich client if it has all the UI required on client’s side. Overall UI design is developed using Android studio. We are using the Base.V21.Theme.AppCompat theme for our application and Nexus 5 API 21 as emulator for testing our application. As of now, for the second, we have designed 2 screens.

1. Student home screen after successful login
2. Volunteer home screen after successful login.

User Interacts by a touch based smartphone there by navigating to other screens and perform operations on the server. Typical mobile client flow of operations is as follows. When the student or a volunteer install the app, they are asked to identify themselves (to distinguish between student and volunteer). Then they are redirected to a login page where already existing users can login and new users can register. After successful registration, users are navigated back to the Start screen form where they can login to their respective home pages (Student Home Screen & Volunteer Home Screen). Student Home Page will consist of the necessary student details and the current weather report of the destination location. Volunteer home page consists of primary details of the assigned students and the google navigation functionality to navigate to their respective dropping addresses.

**Design of Unit test cases (using NUnit tool):**

Test cases are designed to test the Login and Registration services. This is implemented using visual studio and executed by NUnit Client. Our test case consists of four methods to test the Student login, Volunteer Login, Student Registration, Volunteer Registration, get student Details and Get Volunteer details functionalities. Intelligent System and notification service are tested by testing Student Registration as that service invokes the call to the intelligent system and intern invokes call to notification.

**Implementation**

**Implementation of REST services:**

WCF (Windows Communication Framework) is used to implement REST web services on Visual Studio 2010. Web service project has an endpoint IAuthRegister.cs, which is also called as contract and Implementation of these resources is in AuthRegister.svc.cs. The implementation has several resources implemented and are ready to be consumed from a client. Resources communicate directly with the underlying database.

Registration services saves student and volunteer account information in the system. Both the services will take the respective account information from the Mobile client side and Updates the respective student or volunteer tables respectively.

IntelligentSystem service takes the input as the studentID and get the details of student from DB. It will query the Volunteer DB for the Volunteers who are available for the student pickup and assigns him for pickup. From Mobile client side whenever the student updates his flight arrival time this service gets invoked and assigns him volunteer.

Notification service is used by intelligent system to send mail to Volunteer when a student is being assigned. It used the Gmail server to send mail. When Intelligent system assigns student to Volunteer then this notification service is invoked to send mail to volunteer.

Student Home page service which is used to retrieve student details like first name, last name, email id, airlines, flight No based on the username of the student and pass the respective details back to display in the student home page. By taking the username of the student as the input, it will query the Student DB and gets the respective details.

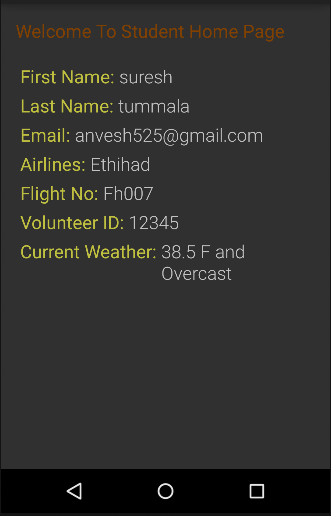
Volunteer Home page service which is used to retrieve primary details like first name, last name and dropping address of all the students who are assigned to the respective volunteer based on the username of the volunteer and display them in the home page of the volunteer. By taking the username of the volunteer as the input, it will query the Student DB and gets the necessary details of all the students assigned to that volunteer.

**Implementation of user interface (Mobile Apps):**

Android studio is being used to implement the Mobile App. User Interface of the app is XML based and is relatively changeable to the screen size. A total of six activities are created for the six screens designed for this increment.

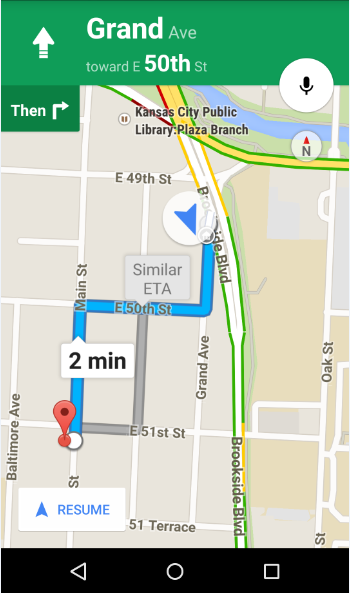
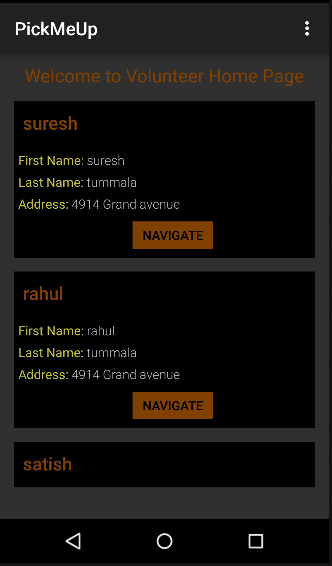
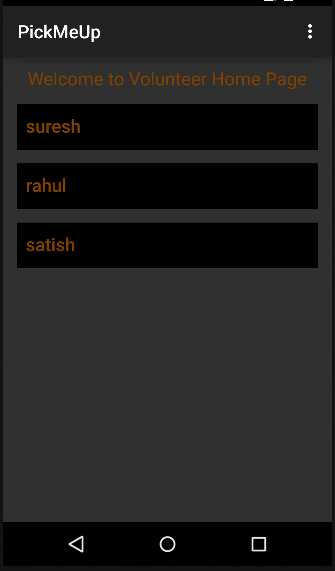
1. **Student Home Screen:** It is the StudentHomeActivity which consists of simple text message “Welcome to Student Home Page” followed by the personal details of the current student and the ID of the volunteer assigned to him and the current weather report of respective location. Upon successful login of the student, student home rest service will be called and of the response will be captured and displayed in the student home screen.

Along with student home service, wunderground external webservice is also called to get the weather report of the current location.



1. **Volunteer Home Screen:** It is the VolunteerHomeActivity which consists of simple text message “Welcome to Volunteer Home Page” followed by the collapsible layout with the names of the students assigned to that volunteer. After successful login of volunteer, Volunteer Home service is called and response of the student details will be captured and displayed. Once any of the student screen is the expanded it will show the first name, last name and dropping address of the student followed by navigate button.

After clicking on the navigate button, google navigation service will be called in a driving mode from the current location to the dropping address of the respective student.

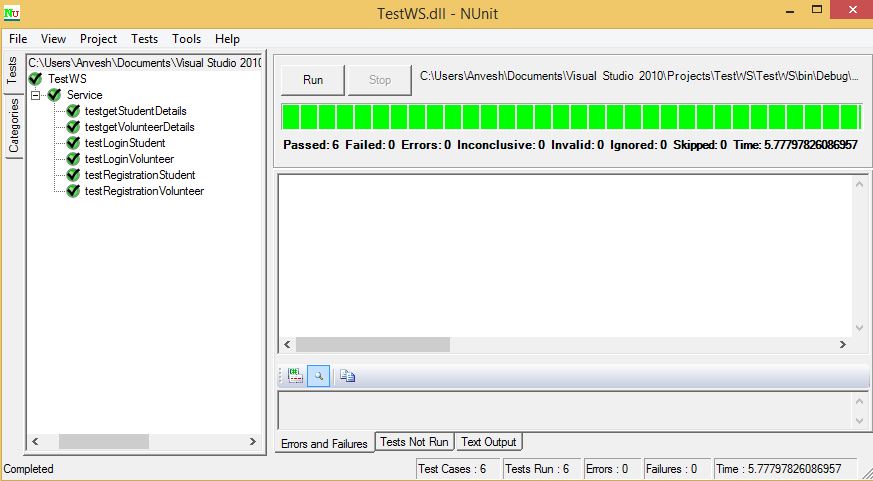


**Implementation of test cases:**

Test cases are implemented by using visual studio and executed by NUnit Client. Our Nuint class Service consists of six test methods, testLoginStudent- to test the student Authentication process, testLoginVolunteer - to test the Volunteer Authentication process, testRegistrationStudent – to test the student Registration process that successful assignment of the Volunteer for the student will return 1, testRegistrationVolunteer - to test the volunteer Registration Process, testgetStudentDetails to test the details of the students returned and testgetVolunteerDetails to test the details of the students assigned for Volunteers.

**Testing: Perform Unit testing (using NUnit tool)**

All the six test cases are successful when our service test in TestWS class is executed with NUnit Client.



**Deployment:**

The whole deployment of the project can be found in here

<http://kc-sce-cs551.kc.umkc.edu/aspnet_client/Group6/PickMeUpService>

**Webservice URLS:**

Get Student details:

<http://kc-sce-cs551.kc.umkc.edu/aspnet_client/Group6/PickMeUpService/AuthService.svc/get/student/{usrn}>

Get Volunteer details:

[http://kc-sce-cs551.kc.umkc.edu/aspnet\_client/Group6/PickMeUpService/AuthService.svc/get/volunteer/{usrn}](http://kc-sce-cs551.kc.umkc.edu/aspnet_client/Group6/PickMeUpService/AuthService.svc/get/volunteer/%7busrn%7d)

The other stories are C# classes which are used by these services to perform certain tasks such as allotting volunteer or student, sending mailing notifications of status like that.

C# classes

intelligentsystem.cs

sendNotification.cs

The final story is the external weather API used to forecast weather of the pickup date.

1. **Third Increment**

**Import Existing Services/API**

In this Increment, we implemented three web services. The web services we implemented are as follows - Admin registration to register for administrators, Admin Login to verify the admin credentials for validating him, Admin Home that will get the Array of list of Volunteers and their assigned students. The main improvement we made in this increment is creation of Admin, finalizing the algorithms with all the modifications required.

Modifications to our previous web services include:

Student Registration: After the successful registration student will be assigned a volunteer automatically and mail will be sent to admin, volunteer and student about the newly assigned pickup.

Volunteer Registration: After a volunteer is successfully registered, the service will look for all the unassigned students and calls for the assignVolunteer method for each of the unassigned students. This will assign the students for newly registered volunteer.

-Admin Registration service for admin to register and use the app.

-Admin Login service for verifying the admin login credentials and let him use the app.

-Admin Home Page for getting all the Volunteers and their assigned students to aloe him display about the volunteers and students details, unassigned students.

- IntelligentSysytem(Modifications) service is for assigning Volunteers to students based on their matching times. Also it looks for unassigned students when a new volunteer is available and assigns him the students for pickups.

- SendNotifications(Modifications) service is for sending notifications to Volunteers when a student has been assigned. As we have included Admin this time, the notification system will be used for sending mails to admin whenever a student has been assigned with a volunteer.

**Detail Design of Services**

**User Stories:**

We have three stories in iteration3

1. As an admin, I want to login so that I can get access to the application.
2. As an admin, I want to create an account and have admin privileges so that I can access all the students and volunteer details.
3. As an admin I want to access all the details of students and volunteers so that I can see who is assigned to who for pick up.

**Service description:**

Admin Registrationis for registering an administration by giving his details like name, Phone Number, Email ID, University Name. Admin will be only one for a university.

**Admin Login** is for verifying the login details entered by the user and checking them with the login details in the Database of the admin. If he is a valid admin, then he will be allowed to uses the application.

**Admin Home:** Once an administrator is being validated he has to get the details of all the Volunteers and their assigned students details. This service will give the arraylist of students with the volunteer at first and followed by the list of assigned students.

**Student Login:** After the successful login of the student, the intelligent system will be automatically invoked and assigns a volunteer and send the notification mail to admin, assigned volunteer and student. The admin is notified that a Volunteer has been assigned for picking a student, the volunteer will be notified that a student is being assigned for pickup and student will be notified with the assigned volunteer name.

**Volunteer Registration:** After a volunteer is successfully registered, the service will look for all the unassigned students and calls for the assignVolunteer method for each of the unassigned students. If this newly added volunteer is available for any unassigned student then those students will be assigned for newly added volunteer. Also Emails will be sent to all the assigned students, volunteers and admin.

*Intelligent system* is a web service that is the core component of this project. The main task of this system is allot volunteer to students upon their arrival timings.

Assigning a volunteer is a part of the web service that is invoked when a volunteer is registered or change his available timings. This service is invoked after the volunteer has successfully registered there by checking his availability timings against student arrival timing. The algorithm that we developed here follow certain rules in assigning a volunteer to the student.

Assigning a student is also a part the of web service that is invoked when the student gets registration. His arrival timing is checked against the volunteer availability. If any volunteer is available he is assigned to that volunteer.

These web services return JSON data back to the client to verify the operation has successfully completed on the server side. REST uses JSON as it data exchange format so as in here for all the web services. Microsoft SQL server is the persistence storage that these services store data on to table and retrieve them for later verification, validation and population purposes.

**sendNotification:**

sendNotification service will send email notifications to volunteer and users regarding their pickup. When a student is assigned to a volunteer. Both receive an email notification of the service. Email notification can be set to different services such as registration, change in timings and update to a new volunteer and so on. Notifications will be sent to the administrator, Volunteer and students.

**IntelligentSysytem:**

IntelligentSysytem service will assign volunteers to students based on the available times of Volunteers and students arrival times. If there are is no volunteer available for the pickup of a student, it will send email notification to Admin to have volunteer for picking that student. The algorithm we followed to assign Volunteers is as follows.

**Algorithm**

We have come up with our own algorithm to assign volunteers for picking students. We have taken input for the available times of Volunteers for a week, assuming he will have a recursive schedule that will be same for all weeks. For each day in the week, we have taken 8 bit input, each bit refers to his availability of every 3 hours. So, the volunteers 24 hours availability is taken input for each 3 hours i.e. {0-3, 3-6, 6-9, 9-12, 12-15, 15-18, 18-21, 21-24}.

**Requirement1:** The Volunteers availability is to be stored in DB as follows, if a Volunteer is available on Monday, Tuesday, Saturday and if he is available from 12-18h on Monday, from 21-24h on Tuesday, fully available on Saturday, he will input his availability as (48-00110000, 128-10000000, 0, 0, 0, 255-11111111, 0). This should be stored in DB as availability of Volunteers.

**Requirement2:** The arriving time of the students is to be stored in the DB as DateTime format.

**Input:** String: StudentID

**Step1:** Extract the arrival time of the student from DB based on input StudentID.

**Step2:** Get the dayOfWeek of arrival time of student, let say Sunday.

**Step3:** Look for the time slot student is coming. I.e is he is coming at 11:15 AM the his daySlot will be 8 - (00001000) bit 1 at (9-12 time period).

**Step4:** From the details of the dayOfWeek and time slot, query the Volunteer database for the week schedule of dayOfWeek

such that Get TOP of VolunteerAvaialability &(bitwise AND) daySlot !=0 AND sort by noassignedstudents ASC.

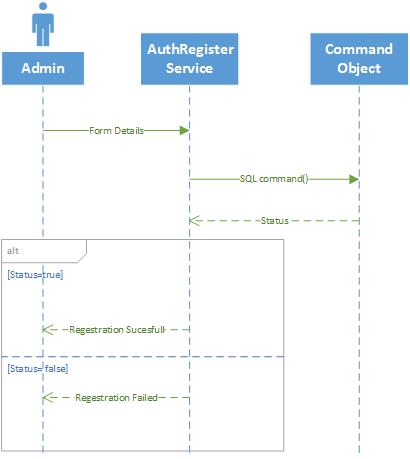
Then we will get the Volunteer who is available for the pickup of student having less no of students being assigned.

**Step5:** Notify the Volunteer about the assigned student details.

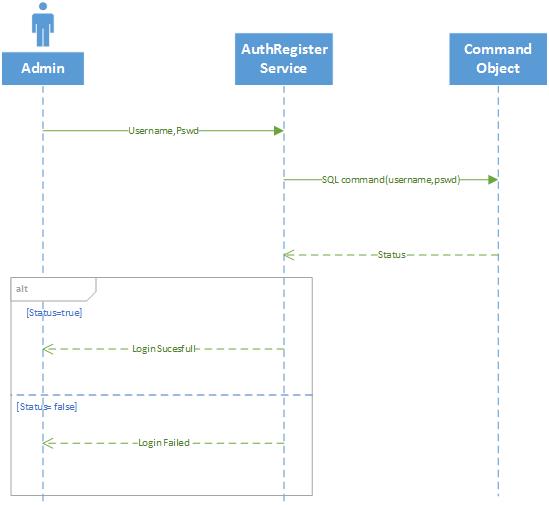
**Step6:** Return status.

**Sequence diagrams:**

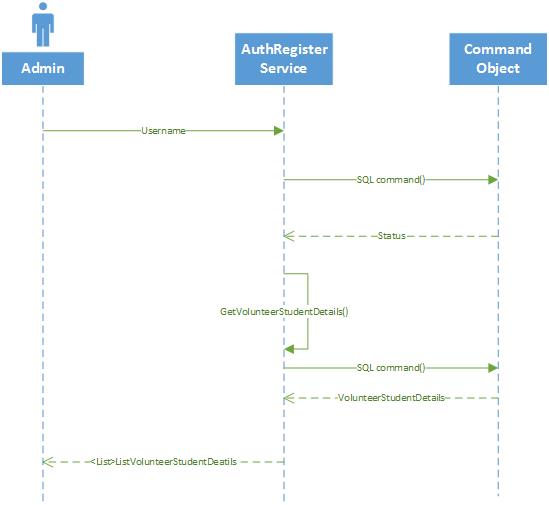
**Admin Registration:**

****

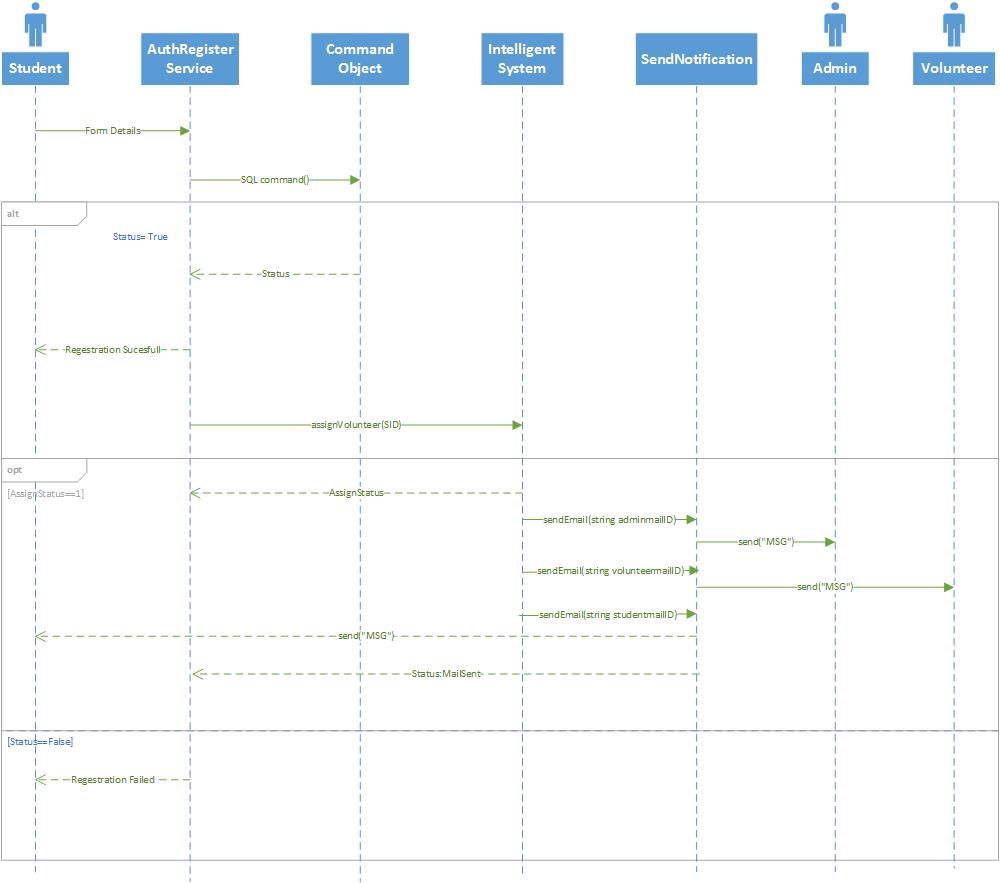
**Admin Login:**

****

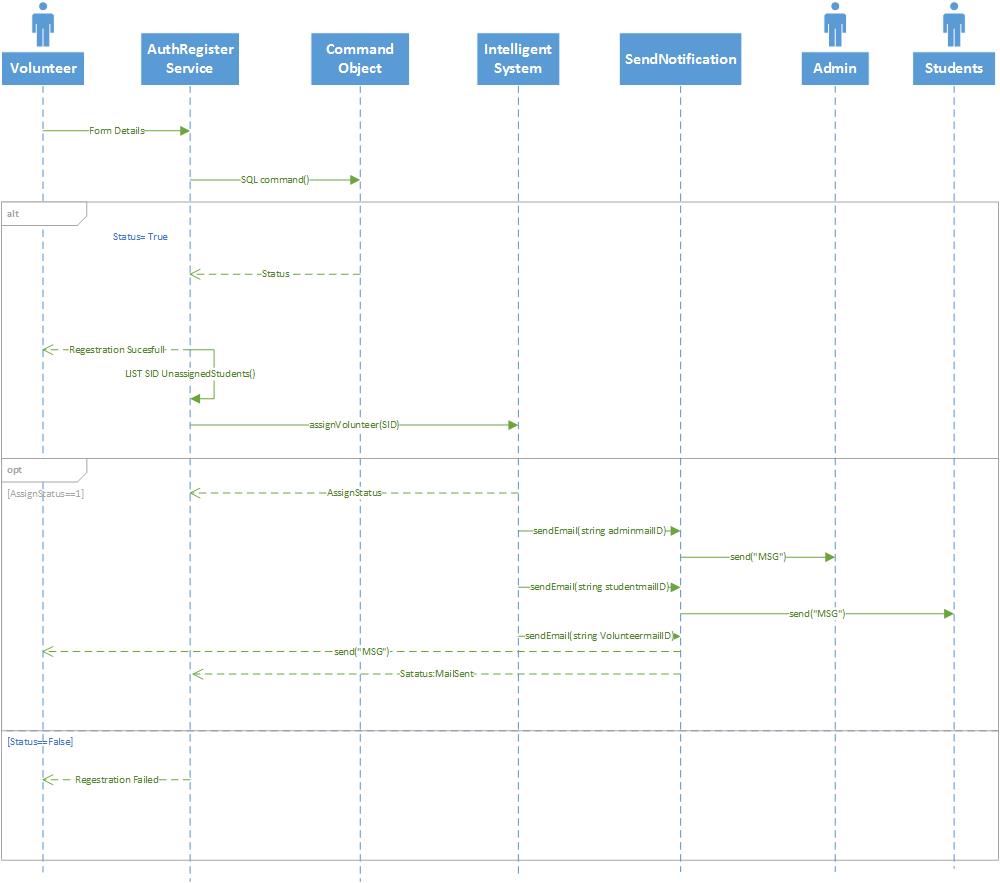
**Getting Admin Home Page Details :**

****

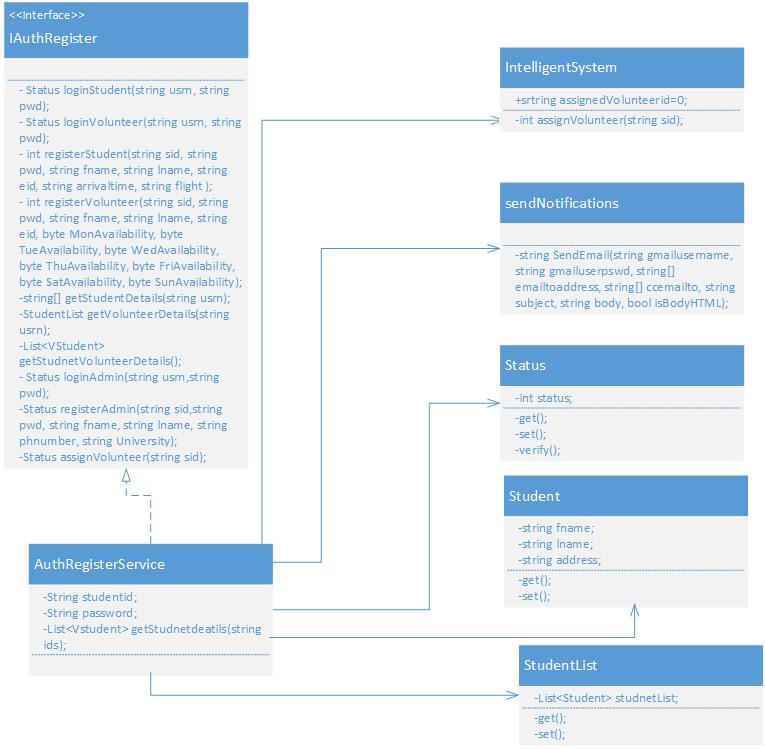
**Student Registration:**



**Volunteer Registration:**

****

**Class Diagram:**

****

**Design of Mobile Client Interface:**

Mobile client interface is an Android application which is a rich client. An application is considered rich client if it has all the UI required on client’s side. Overall UI design is developed using Android studio. We are using the Base.V21.Theme.AppCompat theme for our application and Nexus 5 API 21 as emulator for testing our application. As of now, for the third iteration, we have designed 2 screens and updated 3 screens that already exist.

**New Screens**

1. Admin Registration screen
2. Admin home screen after successful login.

**Updated Screens**

1. Start Screen which prompts user to select Student, Admin or Volunteer
2. Login Screen for Student, Admin and Volunteer
3. Volunteer Registration Screen.

User Interacts by a touch based smartphone there by navigating to other screens and perform operations on the server. Typical mobile client flow of operations is as follows. When the student, admin or a volunteer install the app, they are asked to identify themselves (to distinguish between student, admin and volunteer). Then they are redirected to a login page where already existing users can login and new users can register. Volunteer registration is modified in order to allow the volunteer to select the available time slots for all the 7 days of a week. After successful registration, users are navigated back to the Start screen form where they can login to their respective home pages (Student Home Screen, Admin Home Screen & Volunteer Home Screen). Admin Home Page will consist of details of all the volunteers and the students assigned for each volunteer.

**Design of Unit test cases (using NUnit tool):**

Test cases are designed to test the Login and Registration services, Assign Volunteer service, Admin Home details. This is implemented using visual studio and executed by NUnit Client. Our test case consists of four methods to test the Student login, Volunteer Login, Admin Login Student Registration, Volunteer Registration, Admin Registration, Admin Home details, get student Details, Get Volunteer Student Details and Get Volunteer details functionalities. Intelligent System and notification service are tested by testing Student Registration as that service invokes the call to the intelligent system and intern invokes call to notification. Also the assigning of the students after a volunteer registration is tested.

**Implementation**

**Implementation of REST services:**

WCF (Windows Communication Framework) is used to implement REST web services on Visual Studio 2010. Web service project has an endpoint IAuthRegister.cs, which is also called as contract and Implementation of these resources is in AuthRegister.svc.cs. The implementation has several resources implemented and are ready to be consumed from a client. Resources communicate directly with the underlying database.

Registration services saves student and volunteer account information in the system. Both the services will take the respective account information from the Mobile client side and Updates the respective student or volunteer tables respectively.

IntelligentSystem service takes the input as the studentID and get the details of student from DB. It will query the Volunteer DB for the Volunteers who are available for the student pickup and assigns him for pickup. From Mobile client side whenever the student updates his flight arrival time this service gets invoked and assigns him volunteer.

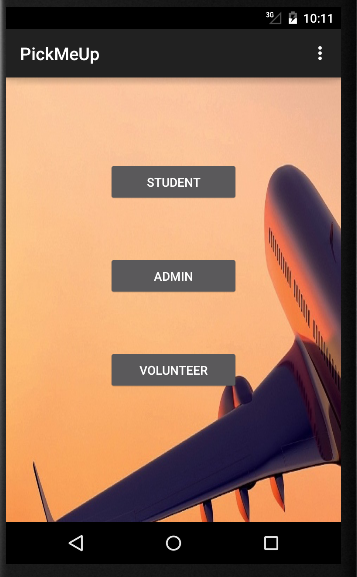
Notification service is used by intelligent system to send mail to volunteer when a student is being assigned. It used the Gmail server to send mail. When Intelligent system assigns student to Volunteer then this notification service is invoked to send mail to volunteer.

The Volunteer Registration will include the getting of the unregistered student details and assigning the new volunteer for their pick up if the volunteer availability matches with the pickup times of students.

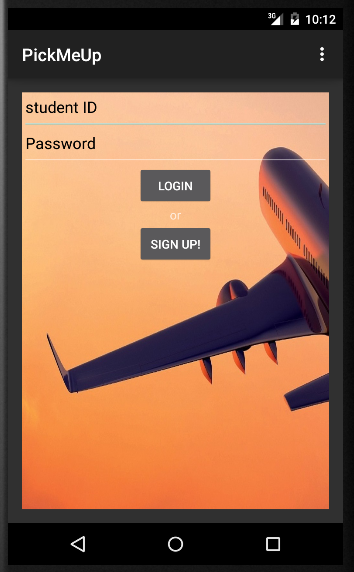
**Implementation of user interface (Mobile Apps):**

Android studio is being used to implement the Mobile App. User Interface of the app is XML based and is relatively changeable to the screen size. Two activities are created for the 2 new screens designed for this increment and 3 existing screens are modified.

1. **Start Screen (updated):** It is the MainActivity consists of three buttons student, admin and volunteer. Upon click of any button, it will navigate to Login screen and will also pass the respective button name to distinguish among student, admin and volunteer for the next levels of navigation from login screen. A new background image is also added for the start screen layout file to make the application more meaningful and attractive.



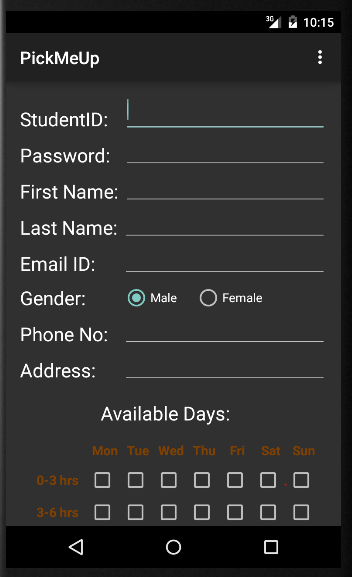
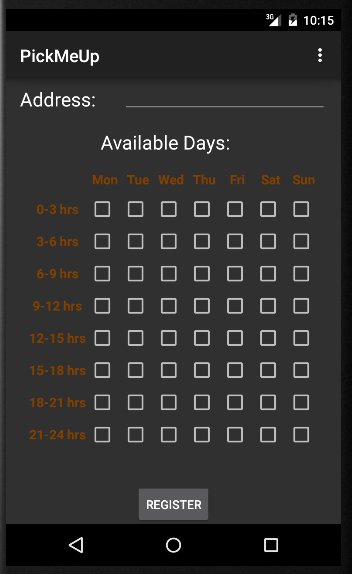
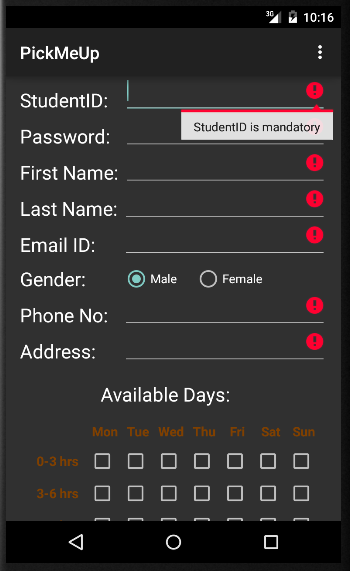
1. **Login Screen (updated):** It is the LoginActivity which consists of StudentID, Password fields and Login, Sign Up buttons. Now, we have provided the login and registration functionality for the Admins. Existing users will navigate to their Home screen by the providing the login details. A new private class AuthenticationService is written inside the LoginActivity which uses HTTP request to call the Login Service from the Mobile client side. Upon click of the login button, respective login service is called to validate the login credentials and response of true or false is returned. If true is returned, a Toast message “Login Successful” is displayed and the user will be navigated to the respective Home screens. If false is returned, a Toast message “Invalid Credentials” will be displayed. A new background image is also added for the start screen layout file to make the application more meaningful and attractive.

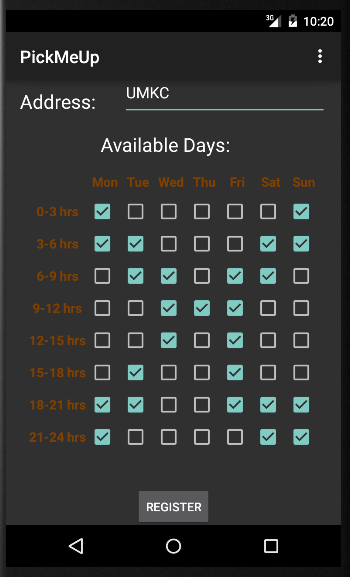
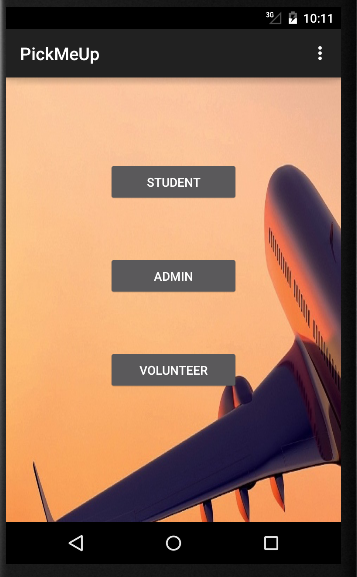


1. **Volunteer Registration Screen:** It is the VolunteerRegActivity which consists of fields like StudentID, Password, FirstName, LastName, Email ID, Gender, Phone No, Address and Available Days fields to create a volunteer account.

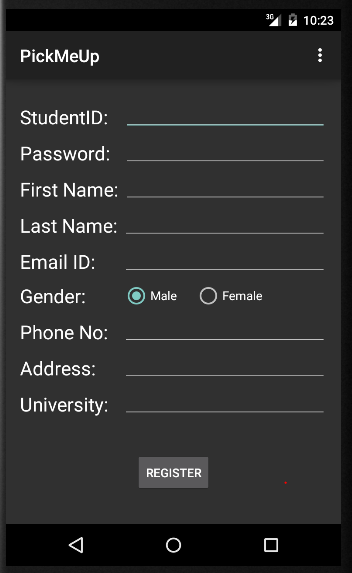
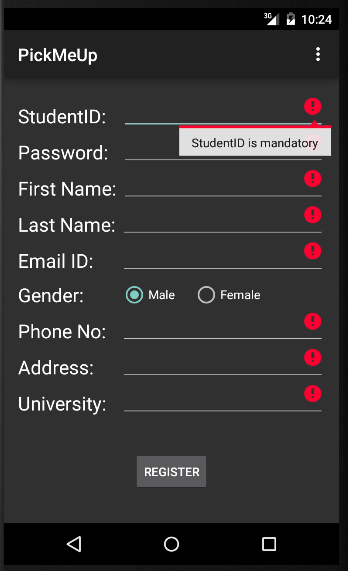
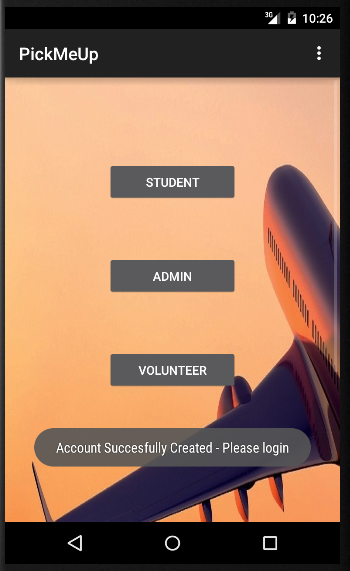
Available Days is divided into 7 days of week. Each day is further divided into 8 slots, each slot of 3 hrs of the total available 24 hrs. A check box is designed for each of the 8 slots for all the 7 days of the week. Upon selecting any checkbox, volunteer assures his availability and a new bit value 1 (0 if not checked) is assigned and final value is calculated for each day. Based on this final value of each day, backend REST service will know the availability of the volunteer for that specific day of the week.

Except the available days, all the other fields are mandatory fields and Validated after the click of Register button. A new service VolunteerRegistration is written inside the VolunteerRegActivty which uses HTTP request to call the Volunteer Registration service from the mobile client side. Upon validation of validation of required fields, this service will be called to create an account for the volunteer. A new Toast message “Account Created Successfully – Please Login” will be displayed to the user and is navigated to the Start Screen.

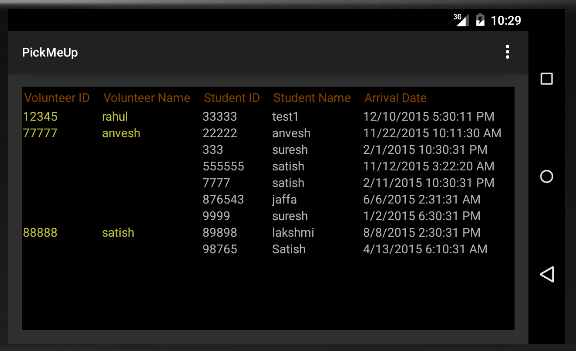
1. **Admin Registration Screen**: It is the AdminRegActivity which consists of fields like StudentID, Password, FirstName, LastName, Email ID, Gender, Phone No, Address and University fields to create an Admin account. All the fields of this activity are mandatory fields validated after the click of the registration button. A new service AdminRegistration is written inside the AdminRegActivty which uses HTTP request to call the Admin Registration service from the mobile client side. Upon validation of validation of required fields, this service will be called to create an account for the admin. A new Toast message “Account Created Successfully – Please Login” will be displayed to the user and is navigated to the Start Screen.

1. **Admin Home Screen:** It is the AdminHomeActivity which consists of table which all the details of the volunteer and the students assigned to that respective volunteer. Volunteer details include the volunteer ID and Volunteer username whereas student details include student ID, student Name and their Arrival Date.

A new service AdminHomeService is written inside the AdminHomeActivty which uses HTTP request to call the Admin home service from the mobile client side.

Upon successful login of the admin, admin home rest service will be called and the response of the respective will be captured and displayed in the Admin Home screen. TableRows will be programmatically appended based on the JSON response of the web service.

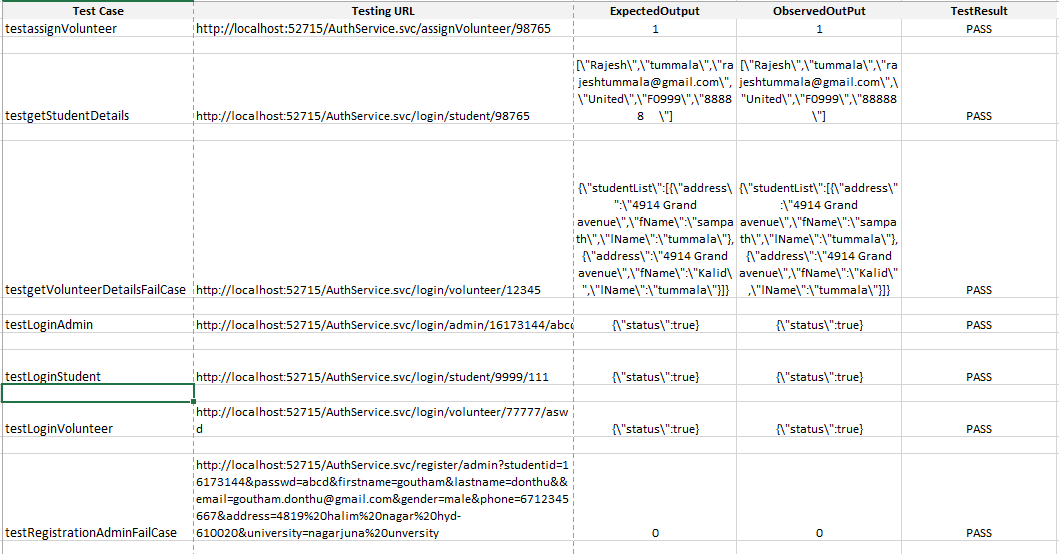


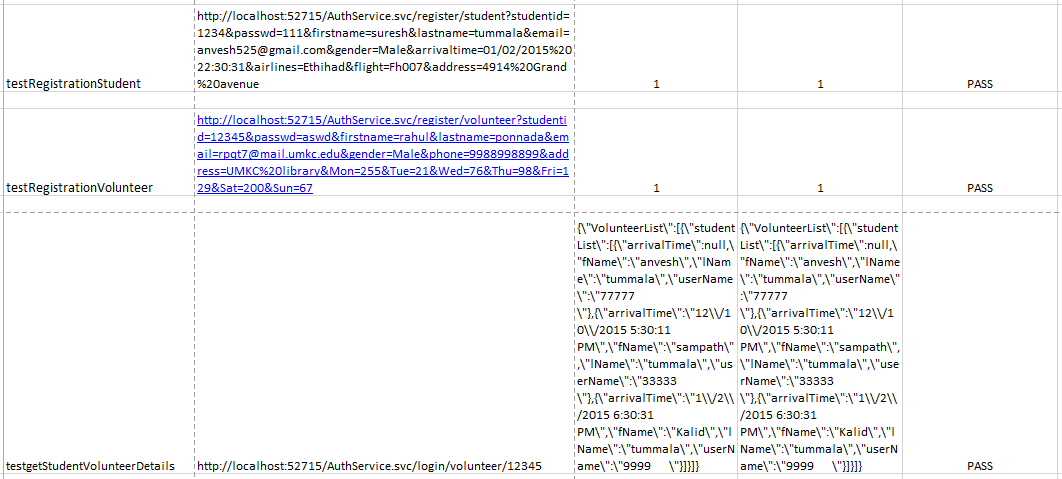
**Implementation of test cases:**

Test cases are implemented by using visual studio and executed by NUnit Client. Our Nuint class Service consists of Ten test methods. They are as follows:

* testassignVolunteer - to test for assigning a volunteer for a student.
* testgetStudentDetails - to test the details of the students returned.
* testgetVolunteerDetailsFailCase - to test the details of the students assigned for Volunteers. This is a fail case as the expected output is excluding the newly assigned student.
* testLoginAdmin - to test the admin Authentication process.
* testLoginStudent- to test the student Authentication process.
* testLoginVolunteer - to test the Volunteer Authentication process.
* testRegistrationAdminFailCase – to test the fail case of the admin by registering with the already existing ID. So it will not allow us to register with duplicate ID.
* testRegistrationStudent - to test the student Registration process that successful assignment of the Volunteer for the student will return 1
* testRegistrationVolunteer - to test the volunteer Registration Process.

**Test Cases:**

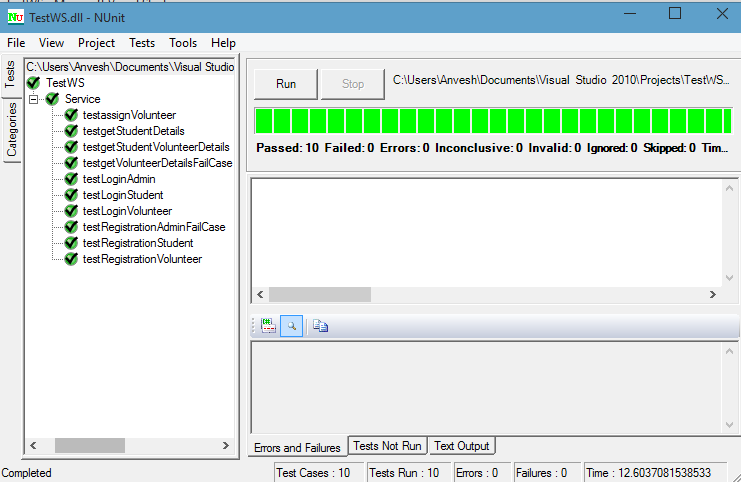




**Testing:**

**Functional Testing : Perform Unit testing (using NUnit tool)**

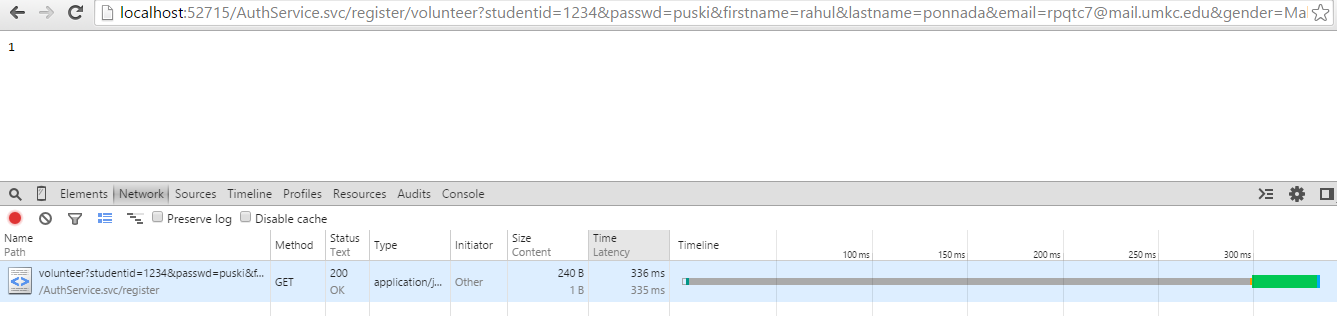
All the six test cases are successful when our service test in TestWS class is executed with NUnit Client.



**Deployment Testing : Runtime performance testing**

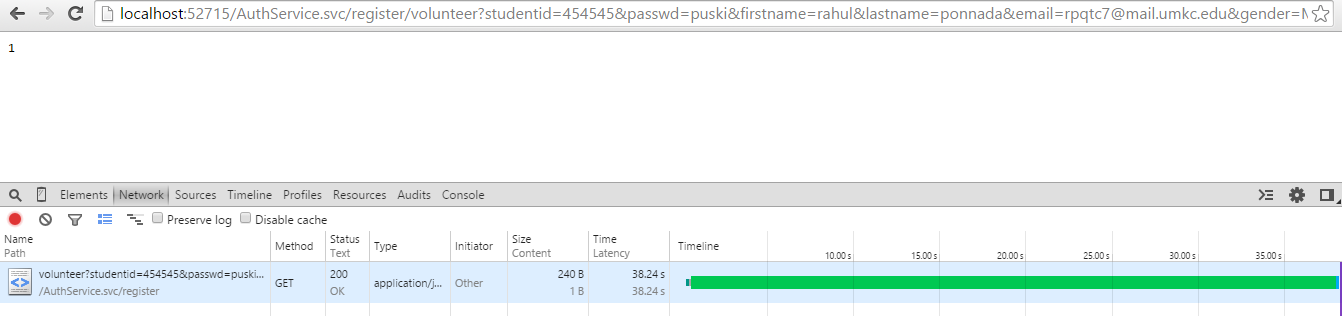
**Tool Used for performance testing : Chrome Dev Tool**

**Volunteer Registration response time (Successful):**



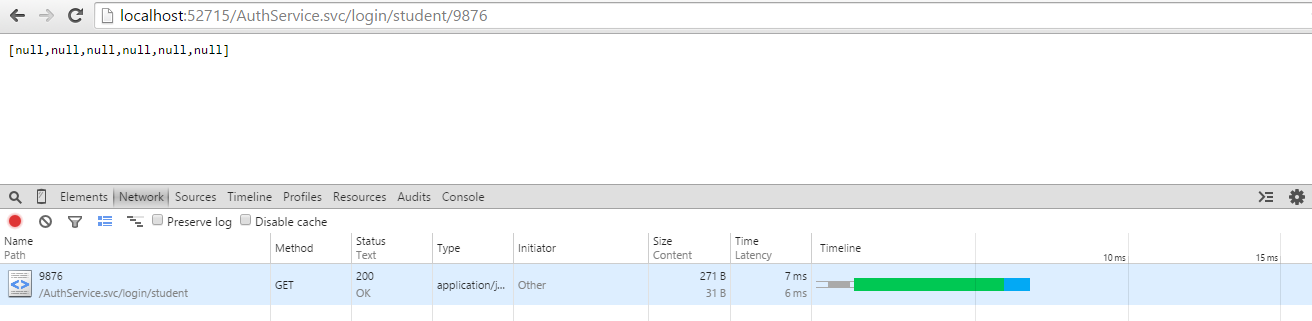
**Volunteer Registration with automatically assigning the students (assigning volunteers for unassigned students)**

We have made 10 students unassigned and have seen the response time for assigning volunteers, mailing for all those students.

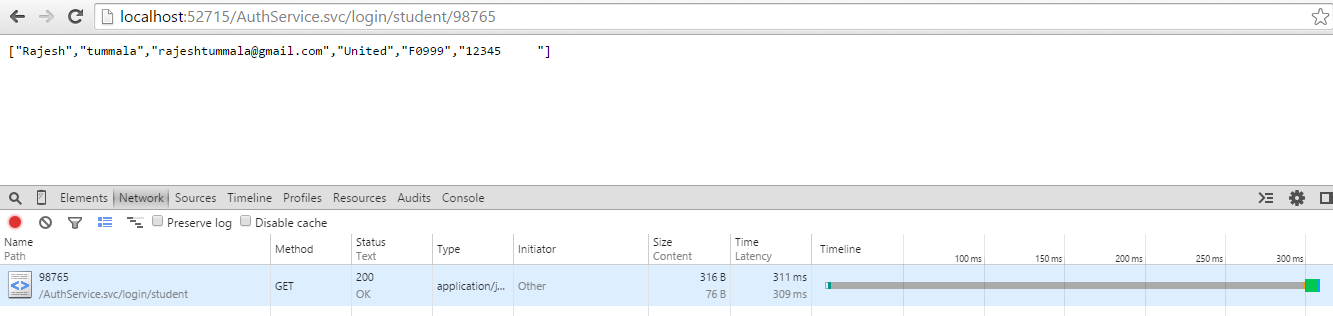


Interesting result: For assigning volunteers for all the 10 students and to send mails(almost 30 mails (3\*10)) it took 38 sec. I think its not that bad, at the same it is not high response time.

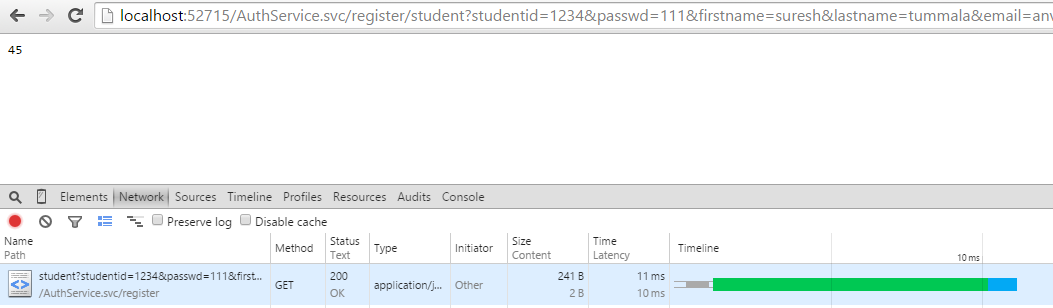
**Login Student response time: *(Unsuccessful Login)***



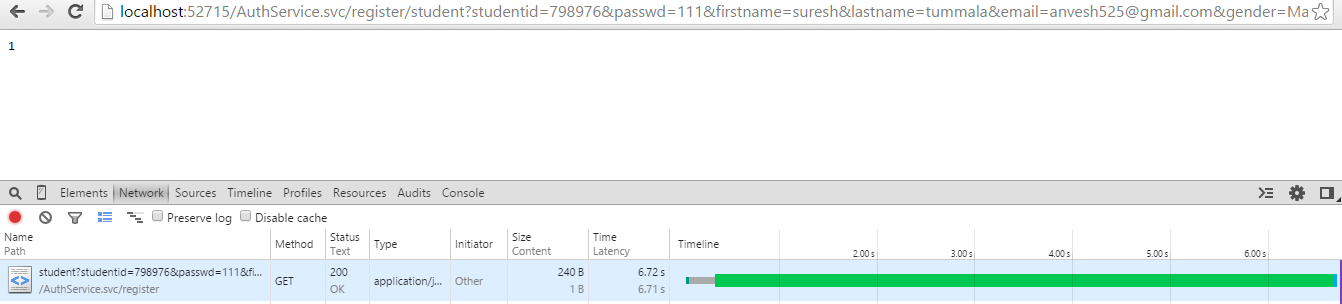
**Login Student response time: *(Successful Login)***



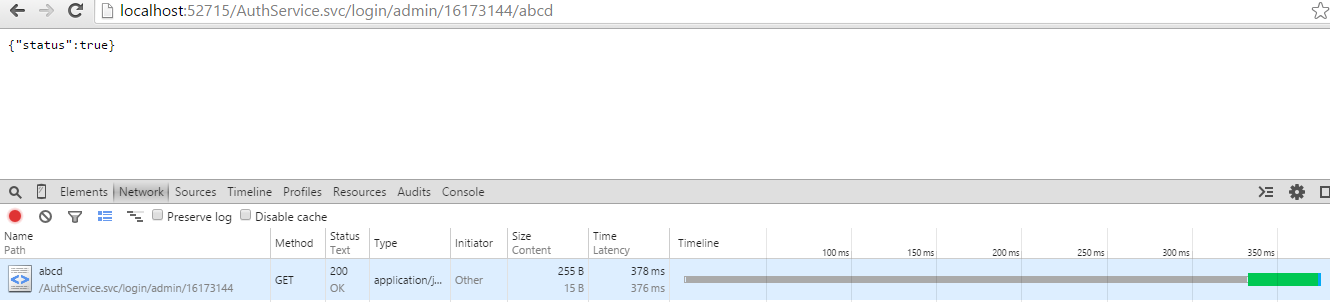
**Student Registration response time: (Failed case)**



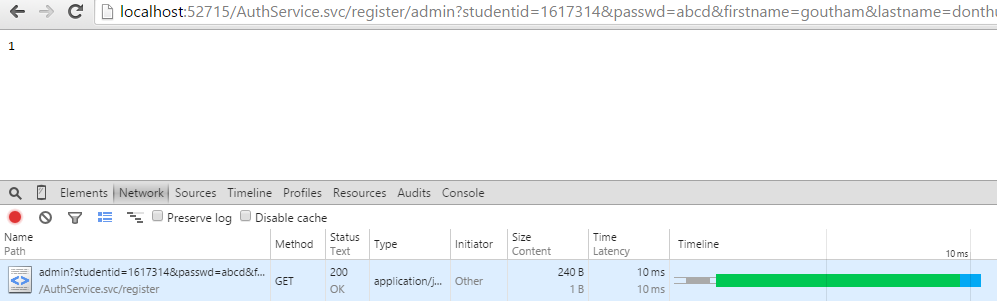
**Student Registration response time: (Successful case)**



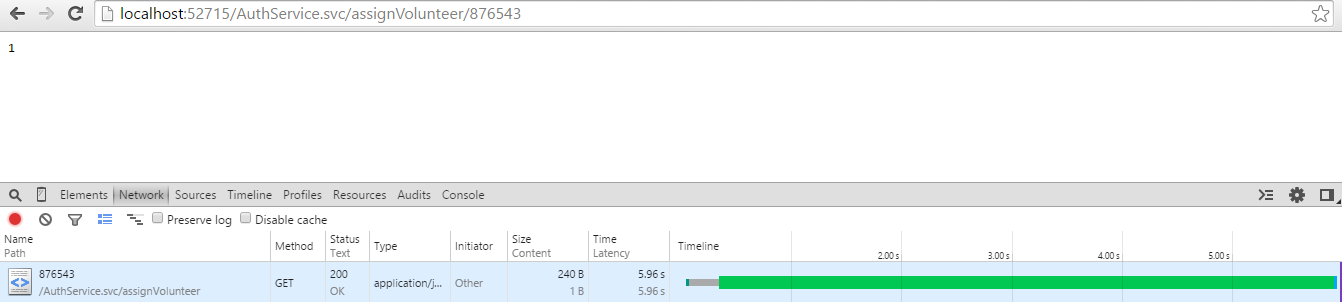
**Admin Login response time:**



**Admin Registration Response Time:**



**Assign Volunteer:**



**Results of the Performance:**

* The results include the high response time for login(<300ms).
* The response time for registration is around (700ms).
* The response time for Volunteer is variant. It depends on the number of assigned students for the registered volunteer, involve sending them mails. The response time we observed for 10 students assigned when a volunteer is registered is 38 secs.

**Deployment:**

The whole deployment of the project can be found in here

<http://kc-sce-cs551.kc.umkc.edu/aspnet_client/Group6/PickMeUpService>

**Webservice URLS:**

Admin Login Service:

<http://kc-sce-cs551.kc.umkc.edu/aspnet_client/Group6/PickMeUpService/AuthService.svc/login/admin/{usrn}/{pwd}>

Admin Registration Service:

[http://kc-sce-cs551.kc.umkc.edu/aspnet\_client/Group6/PickMeUpService/AuthService.svc/ register/admin?studentid={sid}&passwd={pwd}&firstname={fname}&lastname={lname}&email={eid}&gender={sex}&phone={ph}&address={address}&university={university}](http://kc-sce-cs551.kc.umkc.edu/aspnet_client/Group6/PickMeUpService/AuthService.svc/get/volunteer/%7busrn%7d)

Admin Home Service:

http://kc-sce-cs551.kc.umkc.edu/aspnet\_client/Group6/PickMeUpService/AuthService.svc/ login/admin/{usrn}

1. **Fourth Increment**

**Import Existing Services/API**

The project implements twelve web services each one for a different purpose and have used two external services. The web services are as follows Student, Volunteer and Admin login, student, volunteer and Admin registration, Student and Volunteer info Update, Admin unAssignVolunteer and NotifyAll, SendNotifications to send notification emails, IntelligentSysytem that assigns Volunteers to students based up on the time availability. The external APIs used are Weather API for Climate details, Google Map API for Navigation. The main purpose of these web services is to validate the student, volunteers and Admin then register their details to a centralized storage and then to assign the students with volunteers based on their matching schedules and sending notifications to Volunteers when a student has been assigned.

-Login web service for Students, Volunteers and Admins to login into the system for using PickUp application.

-Volunteer Registration service for volunteers to register and then it looks for unassigned students and assigns the volunteer if he is available for the pick of any unassigned students.

-Student registration (Modified) service for students to register and then it uses IntelligentSysytem to assigns volunteers and gives the assigned Volunteers name after successful registration.

- Admin UnAssignVolunteer service is used by Admin for unasssigning any volunteer for students upon receiving any complaints or for any security reasons.

- NotifyAll service is used by Admin for announcement messages to all Students and Volunteers.

- IntelligentSysytem service is for assigning Volunteers to students based on their matching times.

- SendNotifications service is for sending notifications to Volunteers when a student has been assigned.

- Weather API used for showing the climate details of the arrival location for students.

- Google Map API used by Volunteers for navigating to the student specified destination locations.

**Detail Design of Services**

**User Stories:**

We have three stories in this iteration

1. As a volunteer when I update my available timings, the system assigns a student to me so that I am contributing my free time.
2. As a student when I update my travel information the system assigns a volunteer to me so that I have a volunteer who is coming to pick me up
3. As a Admin, I can unassign volunteers for any particular students in case of any concerns.
4. As a admin, I can make announcements to all students and volunteers.
5. **Service description:**

**UnasignVolunteer:**

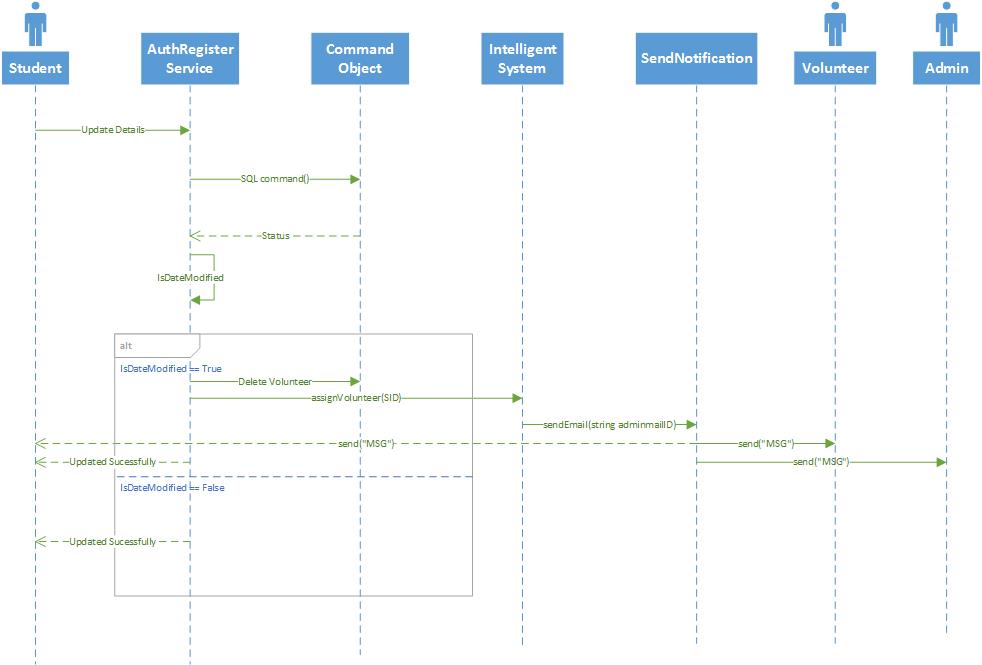
Used by Admin to unassign a volunteer for the student if he got some request from Volunteer or Student.

**NotifyAll:**

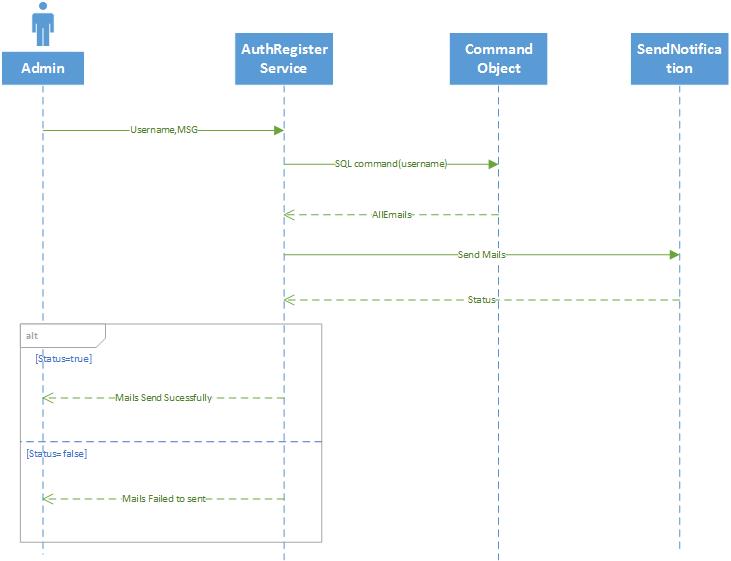
This service is used by Admin to announce a message to all students and Volunteers. This fetches all the students and volunteer emails from Database and used notification service to send message to all of them.

**Sequence diagrams:**

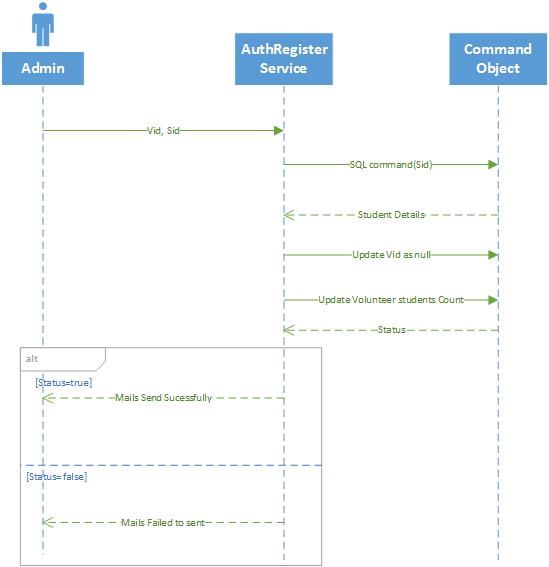
**Student Update Info:**

****

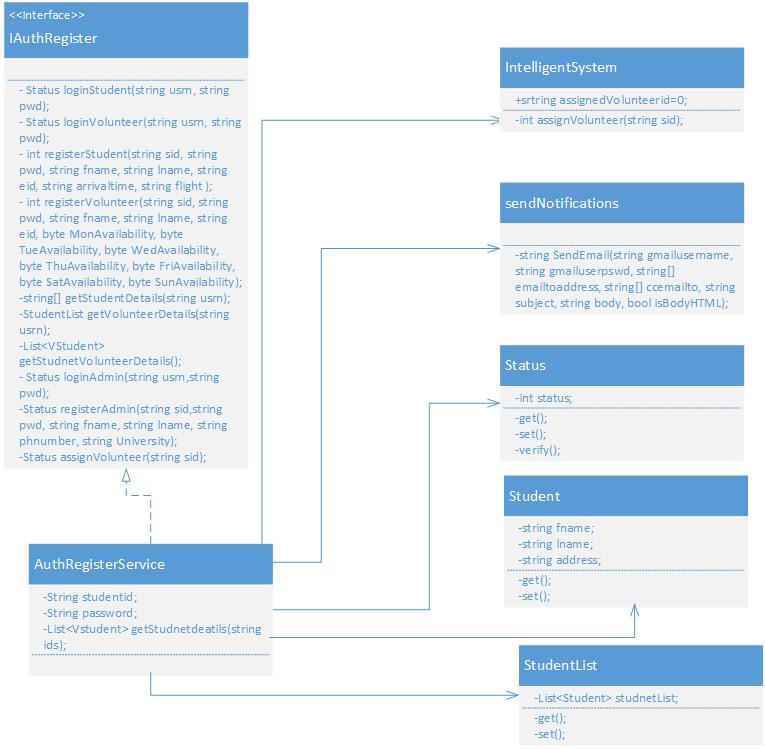
**Admin NotifyAll:**

****

**Admin Unassigning volunteer:**

****

**Class Diagram:**

****

**Design of Mobile Client Interface:**

Mobile client interface is an Android application which is a rich client. An application is considered rich client if it has all the UI required on client’s side. Overall UI design is developed using Android studio. We are using the Base.V21.Theme.AppCompat theme for our application and Nexus 5 API 21 as emulator for testing our application. We have a total of 9 screens as listed below.

**Screens**

1. Student Update Screen
2. Volunteer Update Screen
3. Admin Home Screen
4. Make Announcement Screen.

User Interacts by a touch based smartphone there by navigating to other screens and perform operations on the server. Typical mobile client flow of operations is as follows. When the student, admin or a volunteer install the app, they are asked to identify themselves (to distinguish between student, admin and volunteer). Then they are redirected to a login page where already existing users can login and new users can register. Volunteer registration is modified in order to allow the volunteer to select the available time slots for all the 7 days of a week. After successful registration, users are navigated back to the Start screen form where they can login to their respective home pages (Student Home Screen, Admin Home Screen & Volunteer Home Screen). Admin Home Page will consist of details of all the volunteers and the students assigned for each volunteer.

**Design of Unit test cases (using NUnit tool):**

Test cases are designed to test the Login and Registration services, Assign Volunteer service, Admin Home details. This is implemented using visual studio and executed by NUnit Client. Our test case consists of four methods to test the Student login, Volunteer Login, Admin Login Student Registration, Volunteer Registration, Admin Registration, Admin Home details, get student Details, Get Volunteer Student Details, NotifyAll, Update Student Details, Update Volunteer Details, Unassign Volunteer and Get Volunteer details functionalities. Intelligent System and notification service are tested by testing Student Registration as that service invokes the call to the intelligent system and intern invokes call to notification. Also the assigning of the students after a volunteer registration is tested.

**Implementation**

**Implementation of REST services:**

WCF (Windows Communication Framework) is used to implement REST web services on Visual Studio 2010. Web service project has an endpoint IAuthRegister.cs, which is also called as contract and Implementation of these resources is in AuthRegister.svc.cs. The implementation has several resources implemented and are ready to be consumed from a client. Resources communicate directly with the underlying database.

Registration services saves student and volunteer account information in the system. Both the services will take the respective account information from the Mobile client side and Updates the respective student or volunteer tables respectively.

IntelligentSystem service takes the input as the studentID and get the details of student from DB. It will query the Volunteer DB for the Volunteers who are available for the student pickup and assigns him for pickup. From Mobile client side whenever the student updates his flight arrival time this service gets invoked and assigns him volunteer.

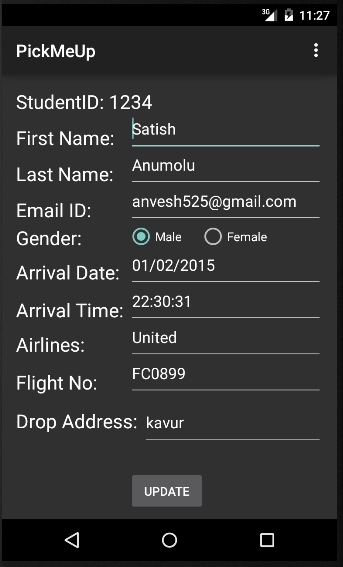
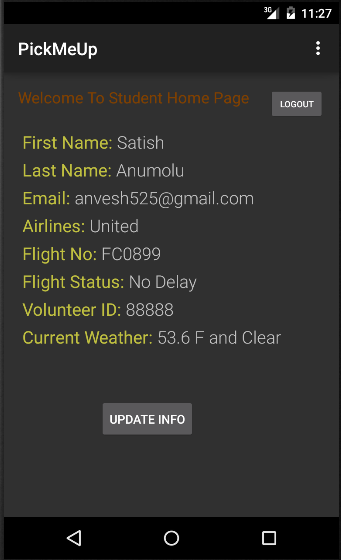
Notification service is used by intelligent system to send mail to volunteer when a student is being assigned. It used the Gmail server to send mail. When Intelligent system assigns student to Volunteer then this notification service is invoked to send mail to volunteer.

The Volunteer Registration will include the getting of the unregistered student details and assigning the new volunteer for their pick up if the volunteer availability matches with the pickup times of students.

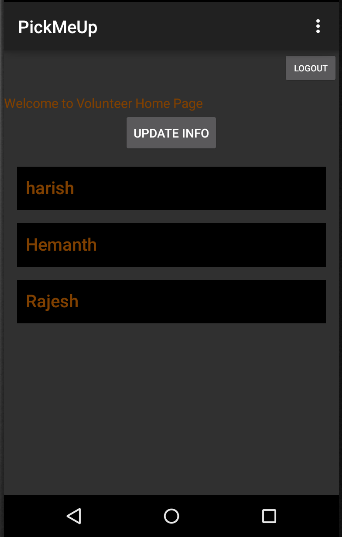
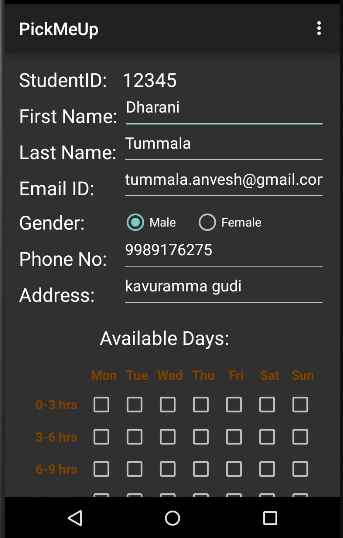
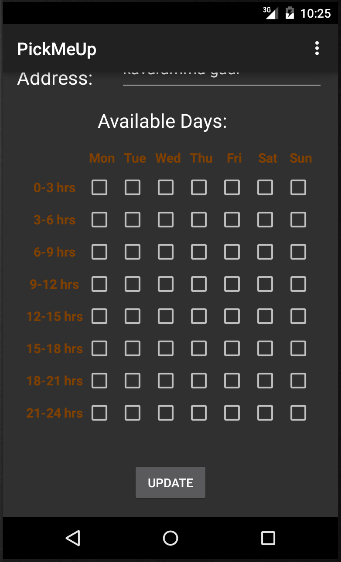
**Implementation of user interface (Mobile Apps):**

Android studio is being used to implement the Mobile App. User Interface of the app is XML based and is relatively changeable to the screen size. Two activities are created for the 2 new screens designed for this increment and 3 existing screens are modified.

1. **Student Update Screen:**  This is for updating the student personal and travel details from the home page.



1. **Volunteer Update Screen:**  This is for updating the volunteer personal and availability from the home page.

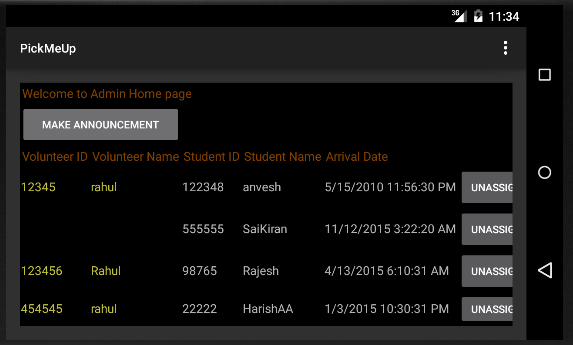
  

1. **Admin Home Screen:** It is the AdminHomeActivity which consists of table which all the details of the volunteer and the students assigned to that respective volunteer. Volunteer details include the volunteer ID and Volunteer username whereas student details include student ID, student Name and their Arrival Date.

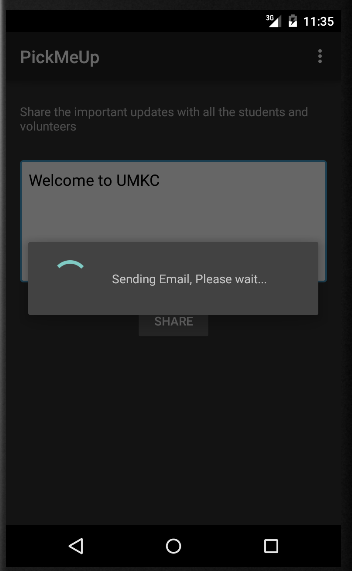
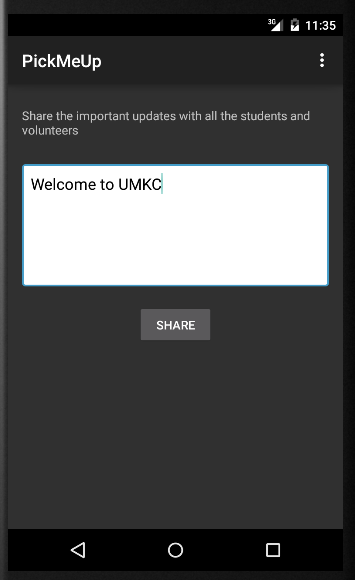
A new service AdminHomeService is written inside the AdminHomeActivty which uses HTTP request to call the Admin home service from the mobile client side.

Upon successful login of the admin, admin home rest service will be called and the response of the respective will be captured and displayed in the Admin Home screen. TableRows will be programmatically appended based on the JSON response of the web service.

There is “Make Announcement” button which will allow the admin to send notifcations to all the students and volunteers about some important updates. Admin can also unassign any volunteer by clicking on “Unaassign” button available for each students entry.



1. **Make Announcement screen:** It is a NotificationActivity which consists of MultiLine text view followed by as share button. Admin can send any important updates using this screen. Upon click of share button, an email will be sent to all the students and volunteers registered for the application and will navigate back to the Admin home screen.



**Implementation of test cases:**

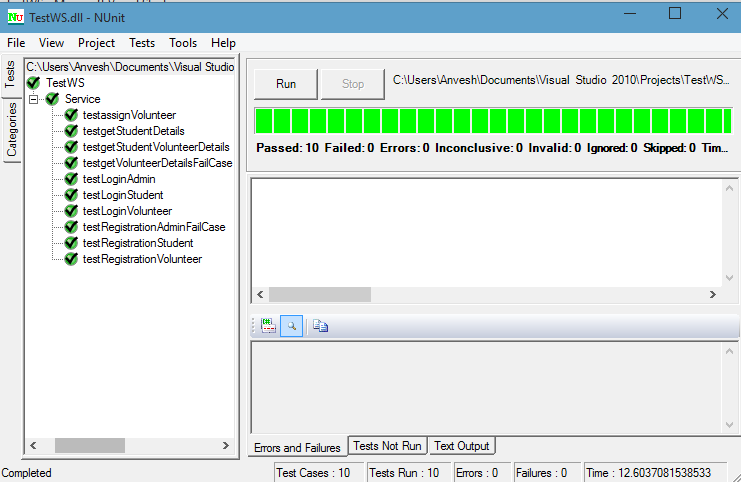
Test cases are implemented by using visual studio and executed by NUnit Client. Our Nuint class Service consists of Ten test methods. They are as follows:

* testassignVolunteer - to test for assigning a volunteer for a student.
* testgetStudentDetails - to test the details of the students returned.
* testgetVolunteerDetailsFailCase - to test the details of the students assigned for Volunteers. This is a fail case as the expected output is excluding the newly assigned student.
* testLoginAdmin - to test the admin Authentication process.
* testLoginStudent- to test the student Authentication process.
* testLoginVolunteer - to test the Volunteer Authentication process.
* testRegistrationAdminFailCase – to test the fail case of the admin by registering with the already existing ID. So it will not allow us to register with duplicate ID.
* testRegistrationStudent - to test the student Registration process that successful assignment of the Volunteer for the student will return 1
* testRegistrationVolunteer - to test the volunteer Registration Process.

**Testing:**

**Functional Testing : Perform Unit testing (using NUnit tool)**

All the six test cases are successful when our service test in TestWS class is executed with NUnit Client.



Deployment:

The whole deployment of the project can be found in here

<http://kc-sce-cs551.kc.umkc.edu/aspnet_client/Group6/PickMeUpService>

**Webservice URLS:**

Get Student details:

<http://kc-sce-cs551.kc.umkc.edu/aspnet_client/Group6/PickMeUpService/AuthService.svc/get/student/{usrn}>

Get Volunteer details:

[http://kc-sce-cs551.kc.umkc.edu/aspnet\_client/Group6/PickMeUpService/AuthService.svc/get/volunteer/{usrn}](http://kc-sce-cs551.kc.umkc.edu/aspnet_client/Group6/PickMeUpService/AuthService.svc/get/volunteer/%7busrn%7d)

Admin Home Service:

http://kc-sce-cs551.kc.umkc.edu/aspnet\_client/Group6/PickMeUpService/AuthService.svc/ login/admin/{usrn}

Unassign Volunteer Service:

http://kc-sce-cs551.kc.umkc.edu/aspnet\_client/Group6/PickMeUpService/AuthService.svc/ unassignvolunteer/{studentId}/{volunteerId}

Notify All service:

http://kc-sce-cs551.kc.umkc.edu/aspnet\_client/Group6/PickMeUpService/AuthService.svc/notifyall/{text}

Student Update Info Service:

http://kc-sce-cs551.kc.umkc.edu/aspnet\_client/Group6/PickMeUpService/AuthService.svc/ student?studentid={studentId}&firstname={firstname}&lastname={lastname}&email={email}&gender={gender}&arrivaltime={arrivaltime}&airlines={airlines}&flight={flight}&address={address}