**CS551 Advanced Software Engineering**

**Fourth Increment Report**

**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)

**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.

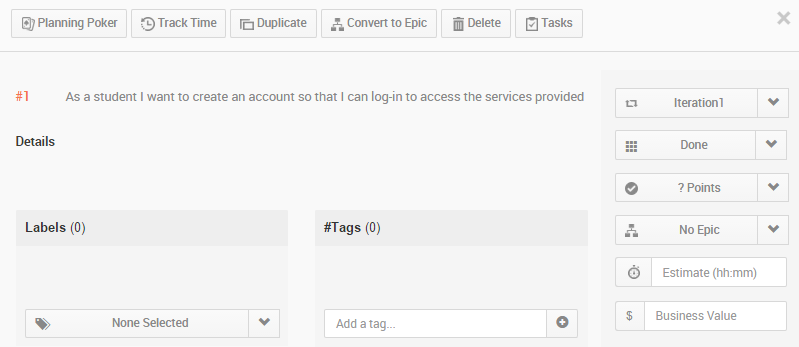
- Google Map API used by Volunteers for navigating to the student specified destination locations.

**Detail Design of Services**

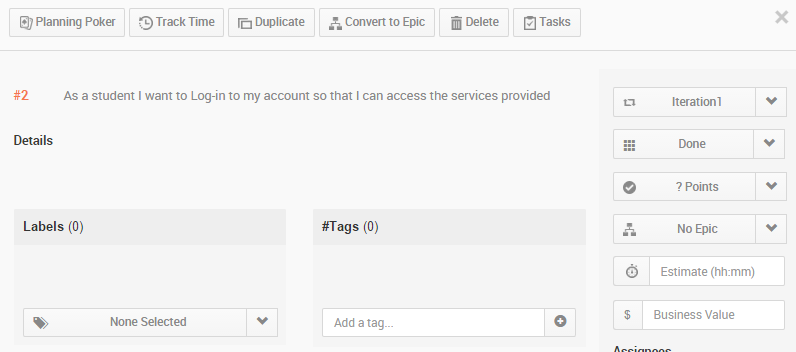
**User Stories:**

We have fifteen stories in this project

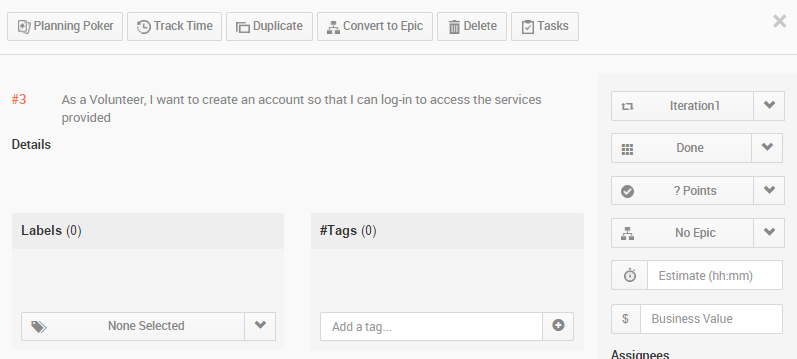
1. As a student, I want to create an account so that I can log-in to access the services provided.



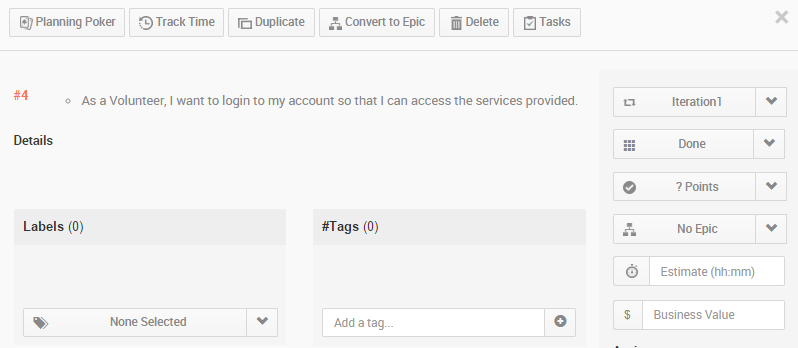
1. As a student, I want to log-in to my account so that I can access the services provided.



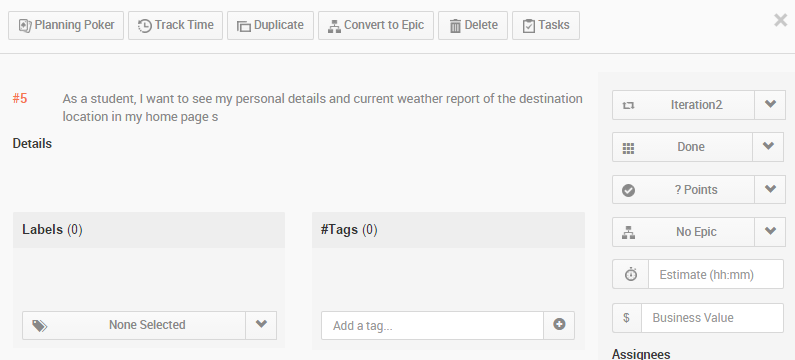
1. As a volunteer, I want to create an account so that I can log-in to access the services provided.



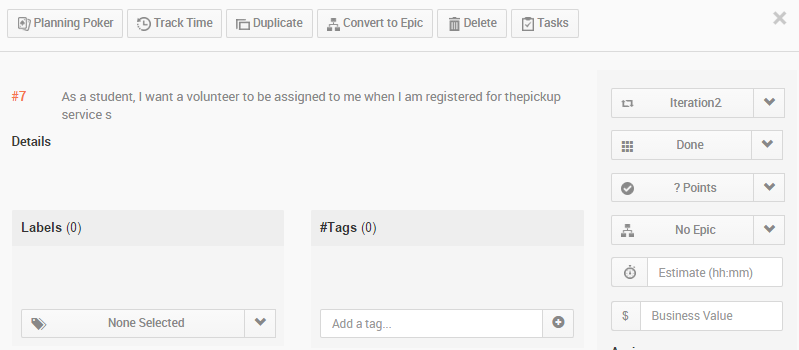
1. As a volunteer, I want to login to my account so that I can access the services provided.



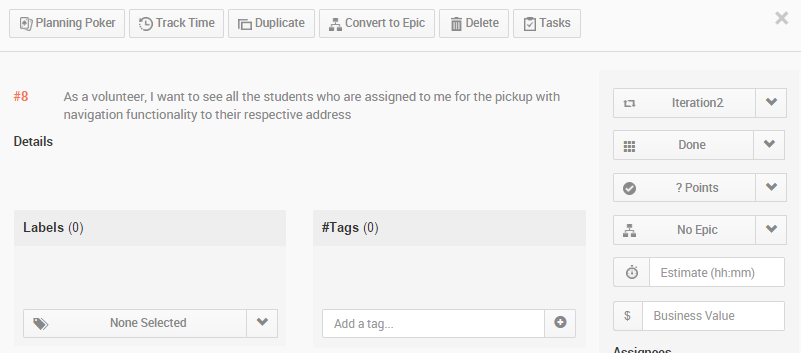
1. As a student, I want to see my personal details and current weather report of the destination location in my home pages.



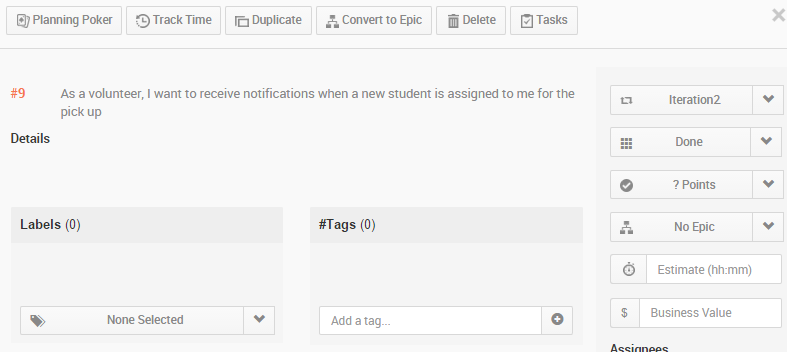
1. As a student, I want a volunteer to be assigned to me when I am registered for the pickup services.



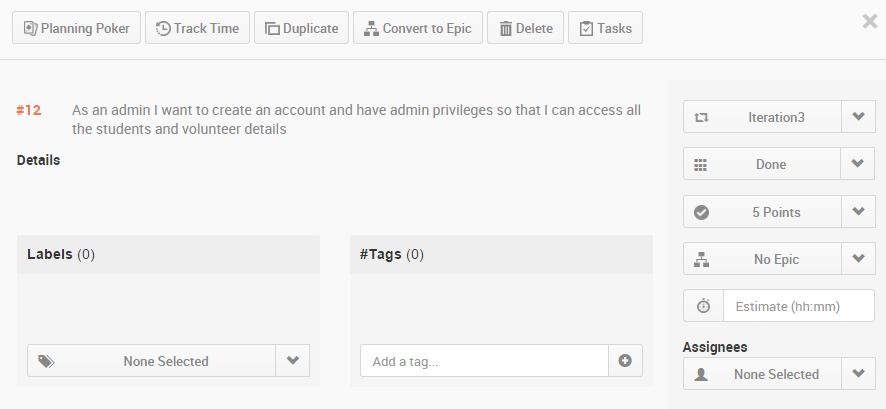
1. As a volunteer, I want to see all the students who are assigned to me for the pickup with navigation functionality to their respective address.



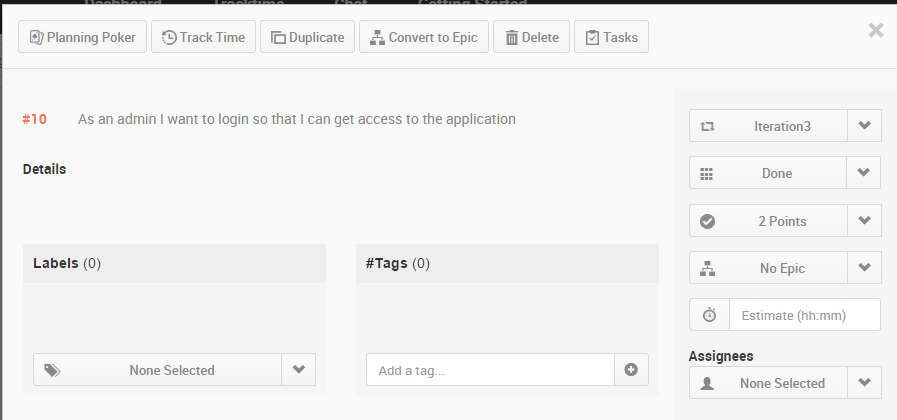
1. As a volunteer, I want to receive notifications when a new student is assigned to me for pick up.



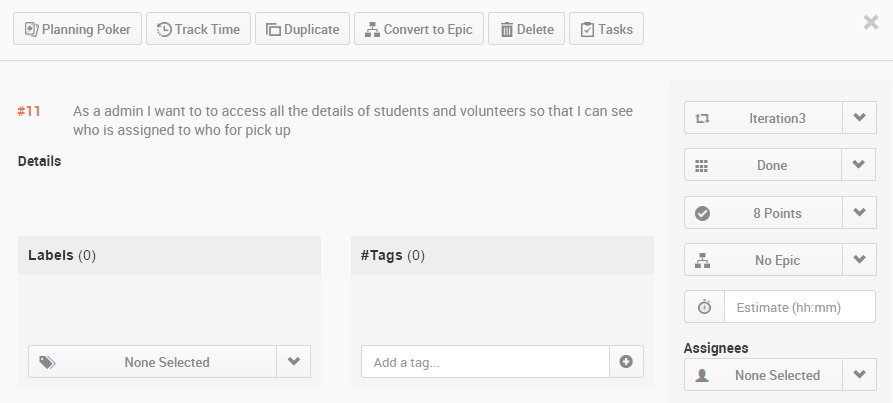
1. As an admin, I want to create an account and have admin privileges, so that I can access all the students and volunteer details.



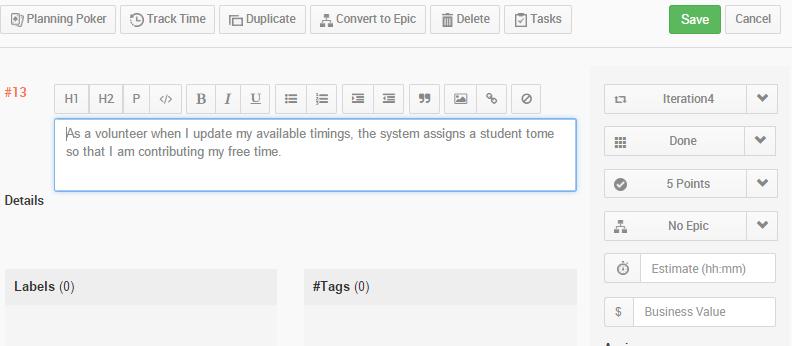
1. As an admin, I want to login to my account so that I can get access to the application.



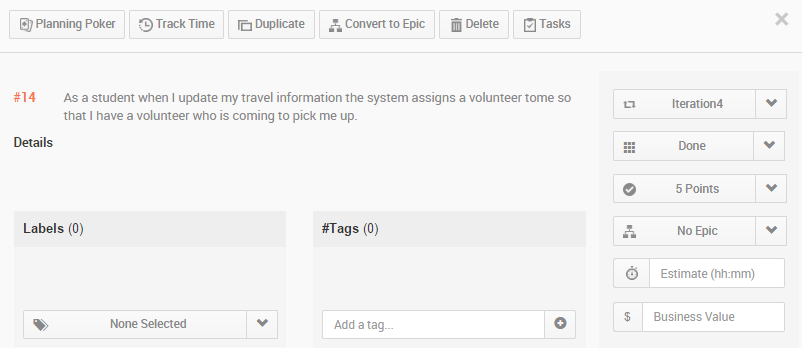
1. As an admin, I want to access all the details of students and volunteers so that I can see who is assigned to who for pick up.



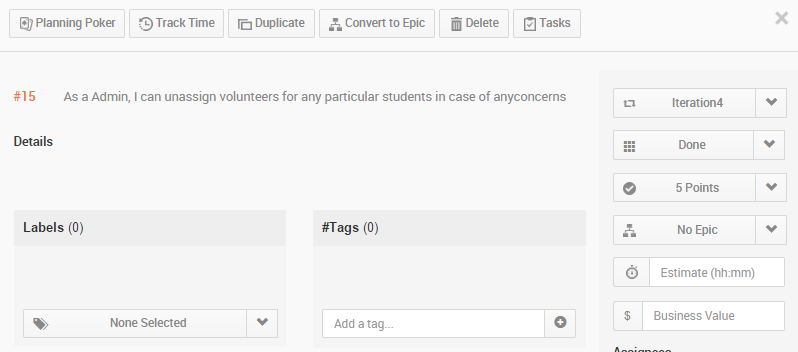
1. As a volunteer when I update my available timings, the system assigns a student to me so that I am contributing my free time



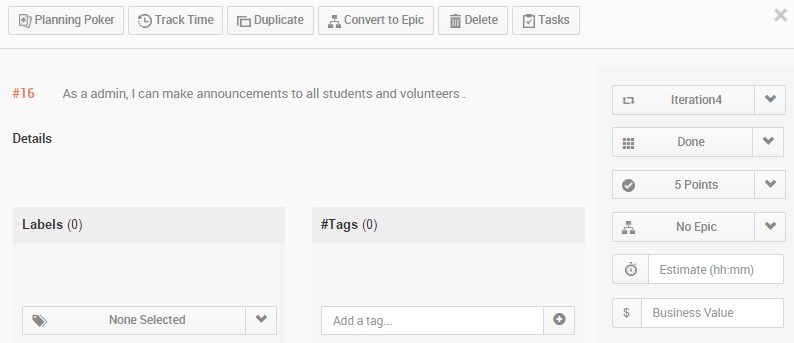
1. As a student when I update my travel information the system assigns a volunteer to me so that I have a volunteer who is coming to pick me up



1. As a Admin, I can unassign volunteers for any particular students in case of any concerns.



1. As a admin, I can make announcements to all students and volunteers.



**Service description:**

Login webservice 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. Hence, this service connects to the underlying database to fetch passwords of the respective users for validation. Student and volunteer registration webservice are pretty much similar except in the parameters they provide and underlying database table they access. It validates the data provided by these two users and records those data onto a database table for further accessing.

*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.

**UnasignVolunteer:**

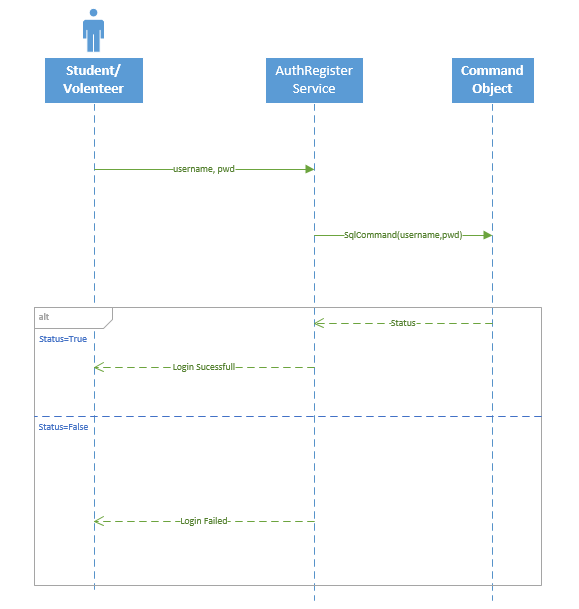
Used by Admin to unassign a volunteer for the student if he got some request from Volunteer or Student.

**NotifyAll:**

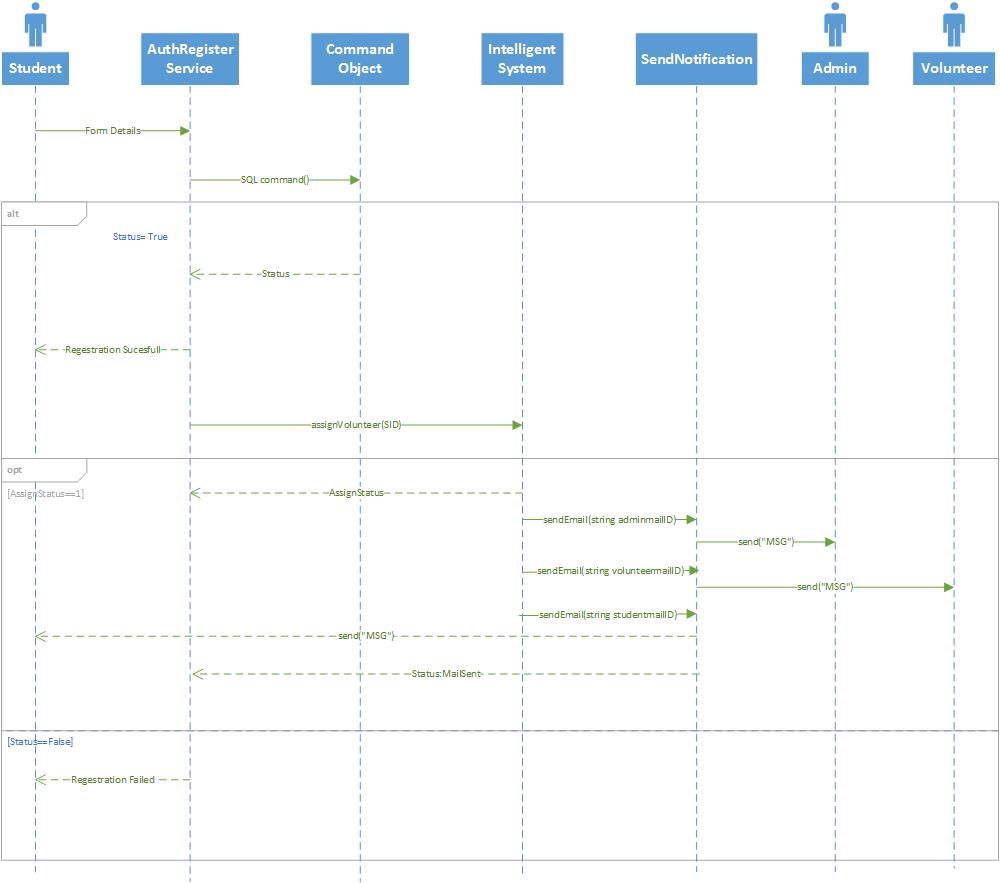
This service is used by Admin to announce a message to all students and Volunteers. This fetches all the students and volunteer emails from Database and used notification service to send message to all of them.

**Sequence diagrams:**

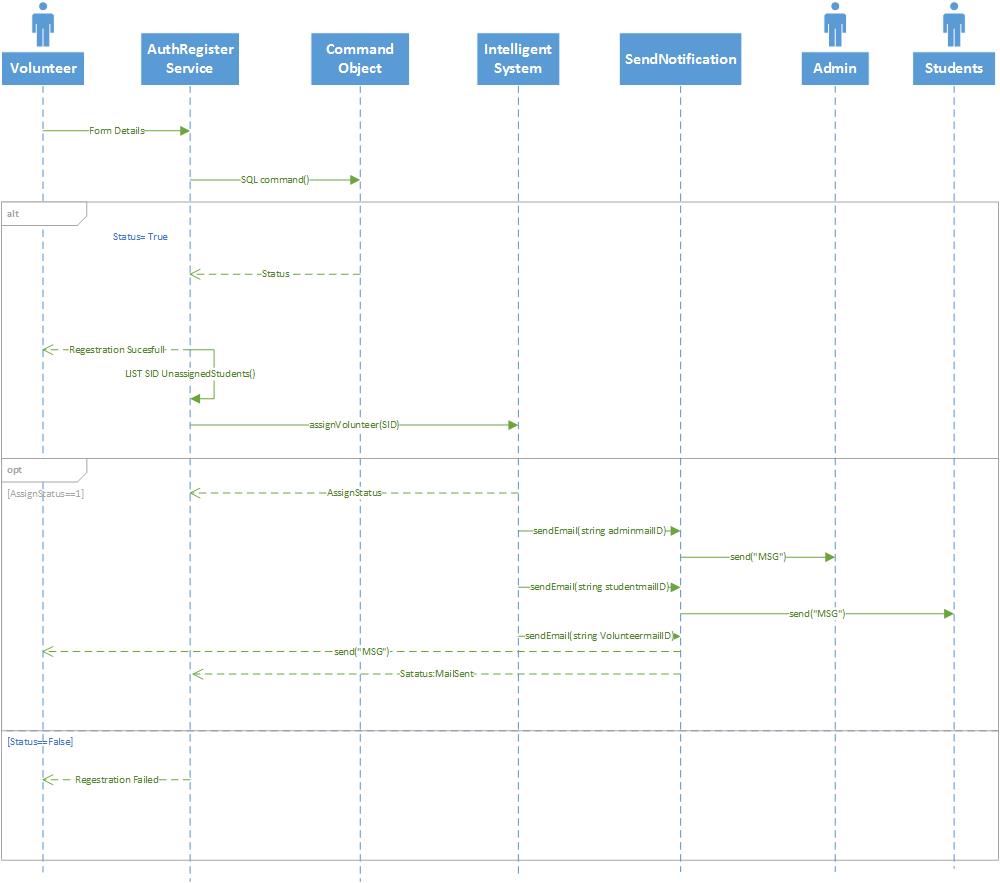
Student/Volunteer/Admin Login sequence



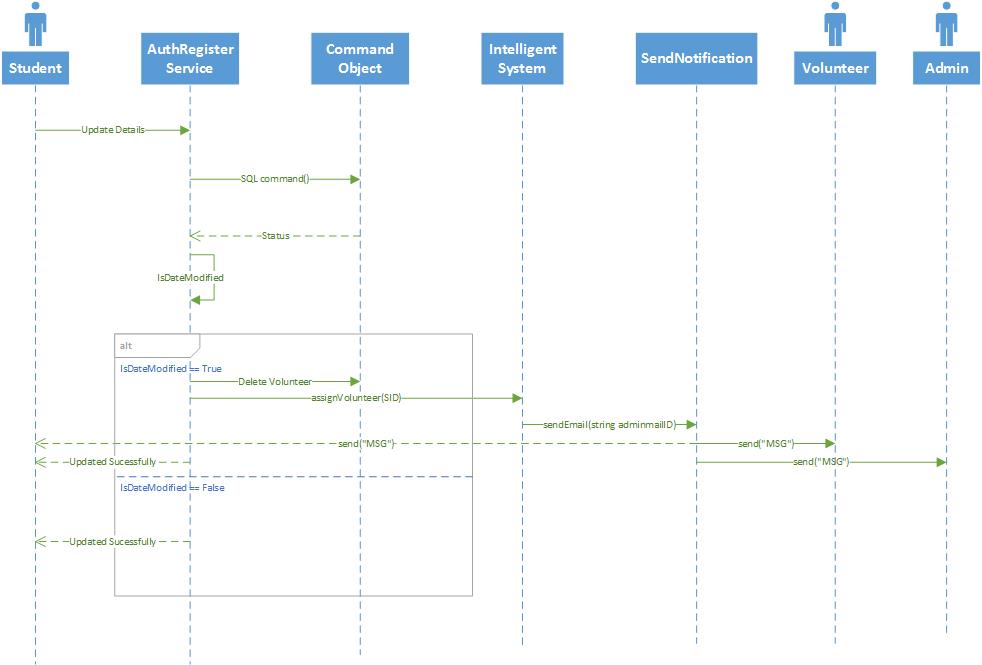
**Student Registration:**



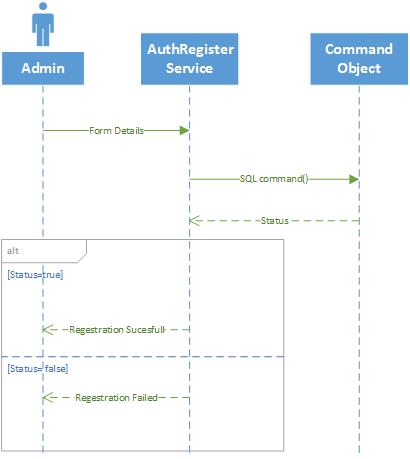
**Volunteer Registration:**

****

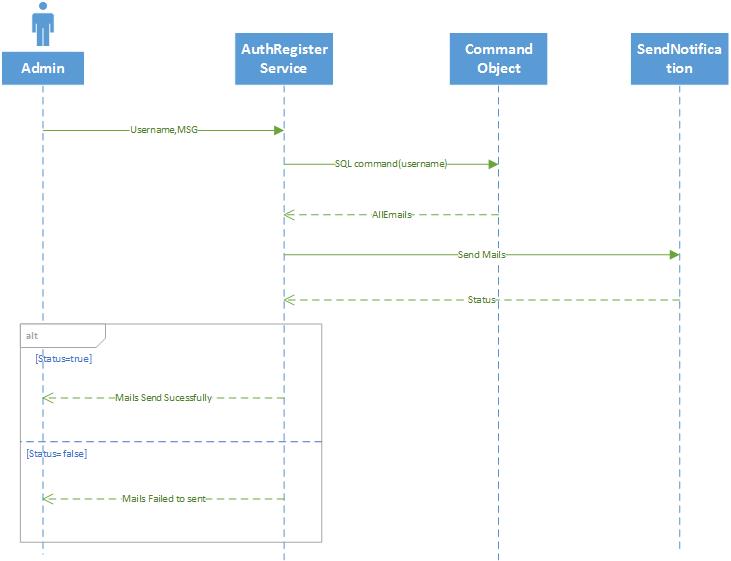
**Student Update Info:**

****

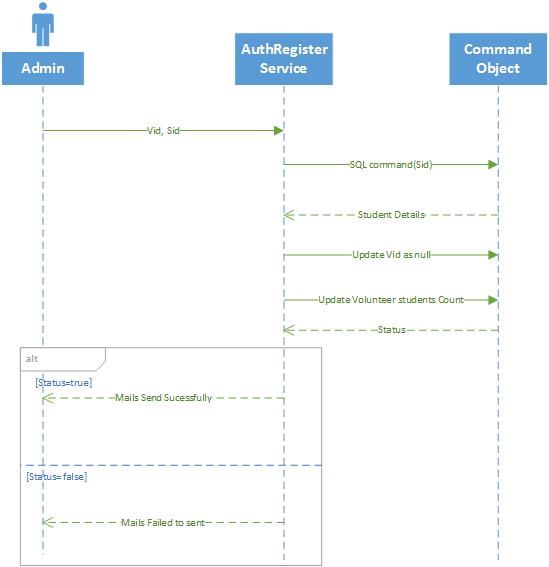
**Admin Registration:**

****

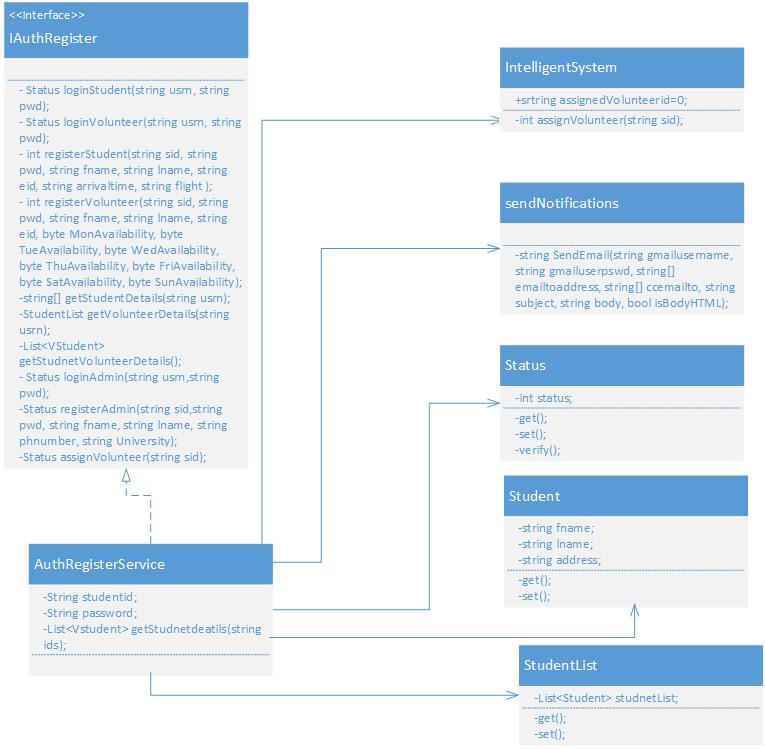
**Admin NotifyAll:**

****

**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. Start Screen
2. Login Screen
3. Student Registration Screen
4. Volunteer Registration Screen
5. Admin Registration screen
6. Student Home Screen
7. Volunteer Home Screen
8. Admin Home Screen
9. 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.

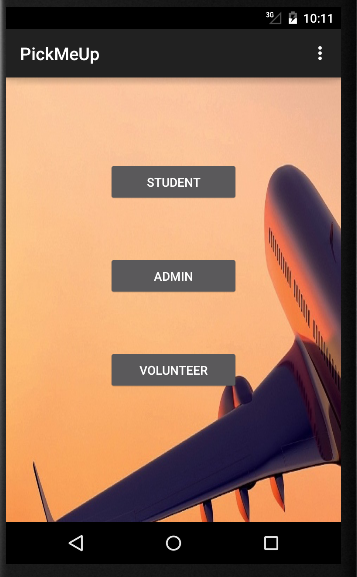
Notification service is used by intelligent system to send mail to volunteer when a student is being assigned. It used the Gmail server to send mail. When Intelligent system assigns student to Volunteer then this notification service is invoked to send mail to volunteer.

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.

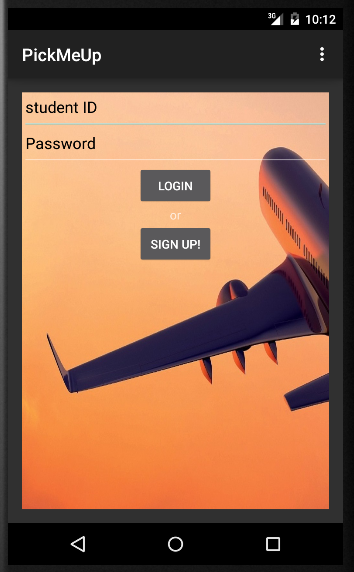
**Implementation of user interface (Mobile Apps):**

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.

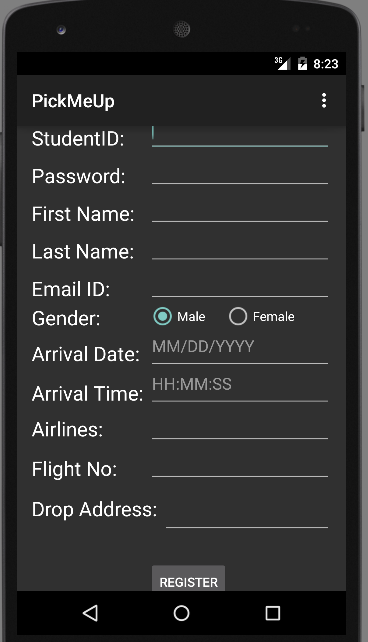
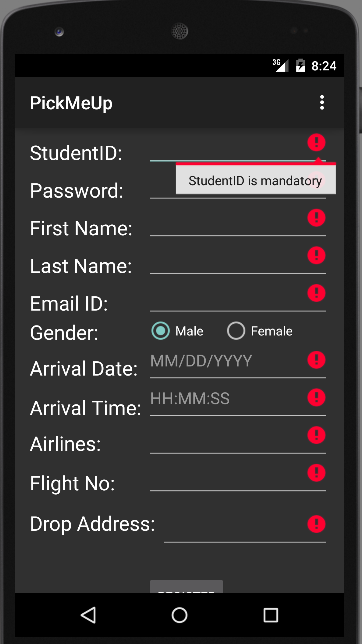
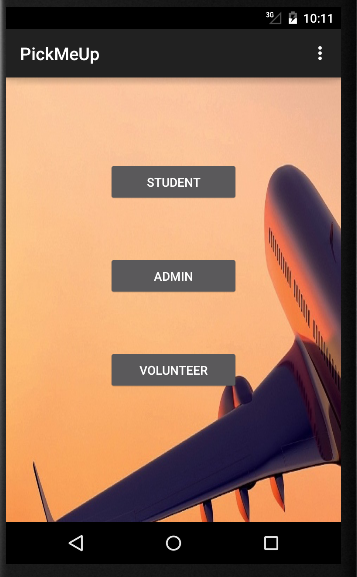
1. **Start Screen:** It is the MainActivity consists of three buttons student, admin and volunteer. Upon click of any button, it will navigate to Login screen and will also pass the respective button name to distinguish among student, admin and volunteer for the next levels of navigation from login screen.



1. **Login Screen:** 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.



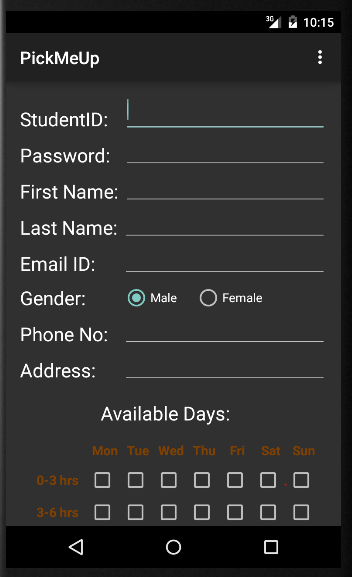
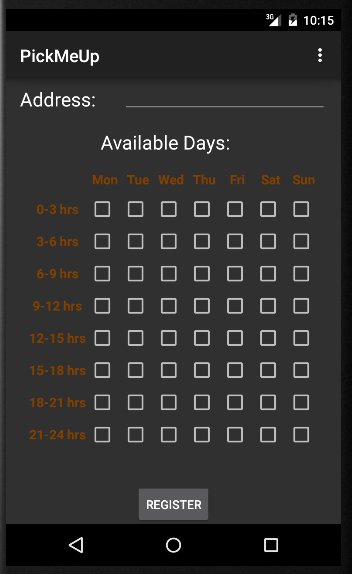
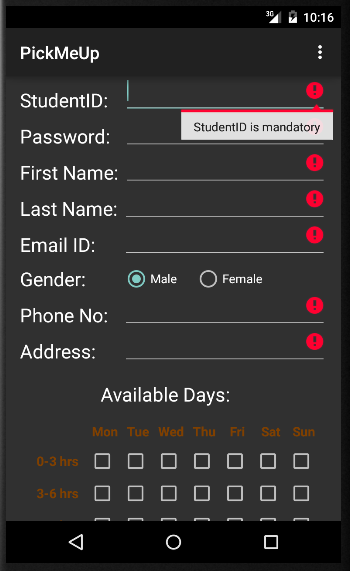
1. **Student Registration Screen:** It is the StudentRegActivity which consists of fields like StudentID, Password, FirstName, LastName, Email ID, Gender, Arrival Date, Arrival Time, Airlines, Flight No and Drop Address to create a student account. Here the studentID field is numeric type, password is 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.

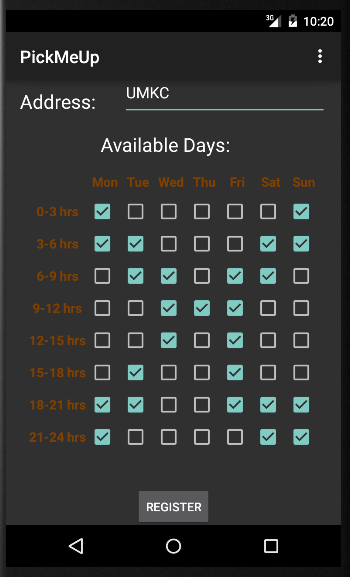
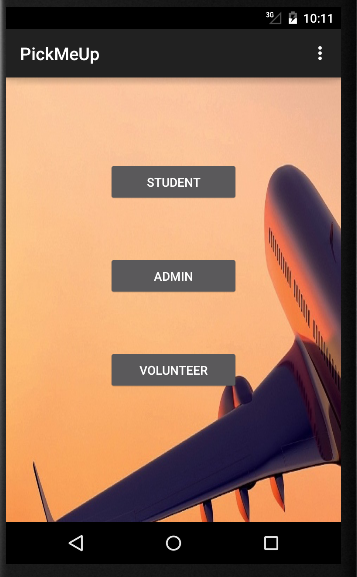
  

1. **Volunteer Registration Screen:** It is the VolunteerRegActivity which consists of fields like StudentID, Password, FirstName, LastName, Email ID, Gender, Phone No, Address and Available Days fields to create a volunteer account.

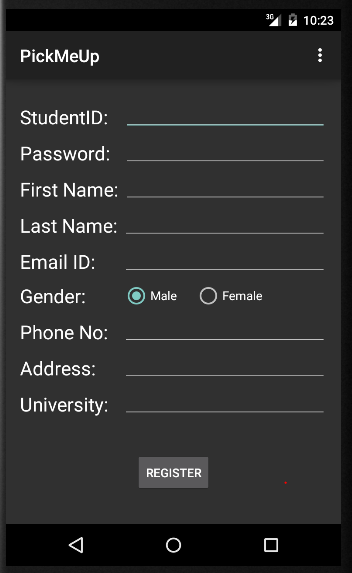
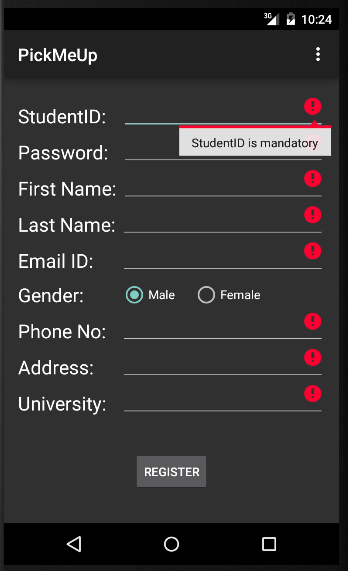
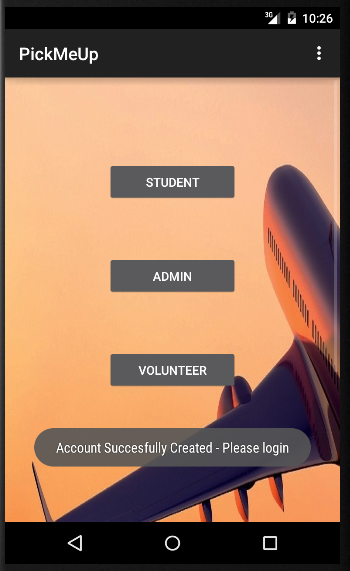
Available Days is divided into 7 days of week. Each day is further divided into 8 slots, each slot of 3 hrs of the total available 24 hrs. A check box is designed for each of the 8 slots for all the 7 days of the week. Upon selecting any checkbox, volunteer assures his availability and a new bit value 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.

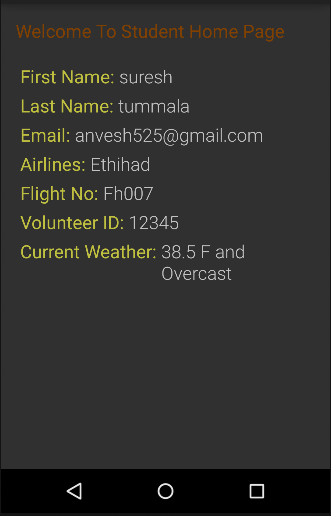
 

1. **Admin Registration Screen**: It is the AdminRegActivity which consists of fields like StudentID, Password, FirstName, LastName, Email ID, Gender, Phone No, Address and University fields to create an Admin account. All the fields of this activity are mandatory fields validated after the click of 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.

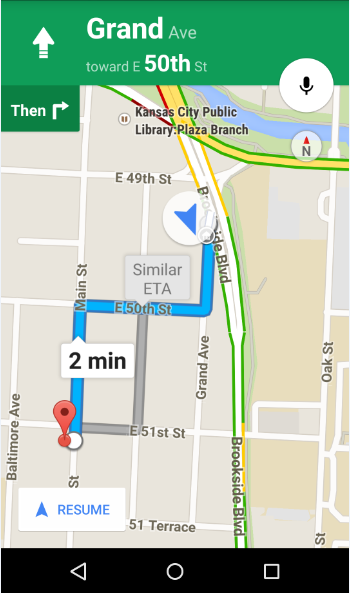
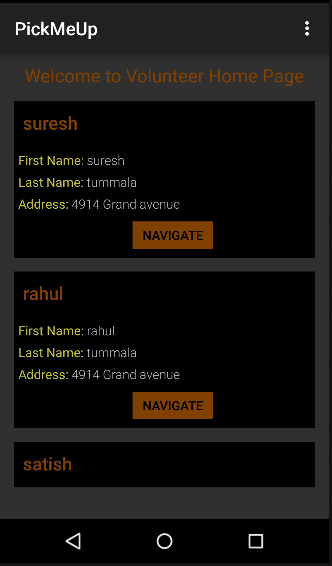
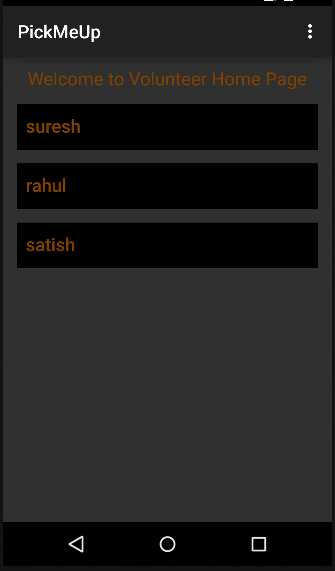
1. **Student Home Screen:** It is the StudentHomeActivity which consists of simple text message “Welcome to Student Home Page” followed by the personal details of the current student and the ID of the volunteer assigned to him and the current weather report of respective location. Upon successful login 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.

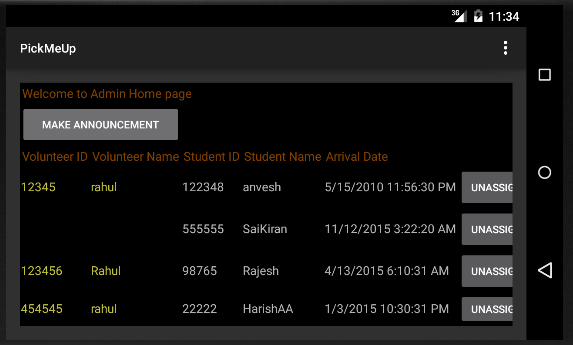


1. **Admin Home Screen:** It is the AdminHomeActivity which consists of table which all the details of the volunteer and the students assigned to that respective volunteer. Volunteer details include the volunteer ID and Volunteer username whereas student details include student ID, student Name and their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

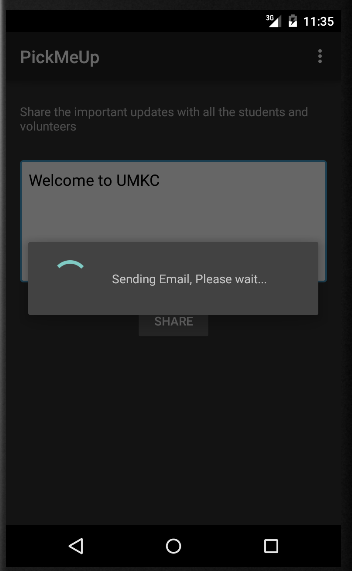
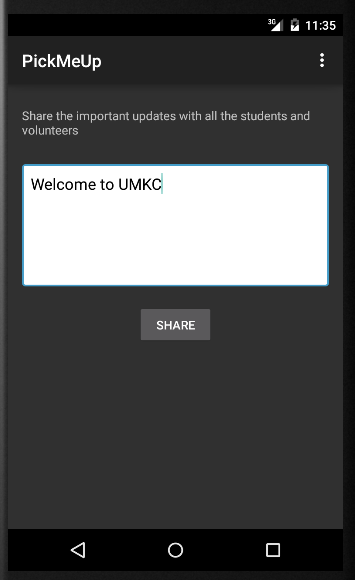
A new service AdminHomeService is written inside the AdminHomeActivty which uses HTTP request to call the Admin home service from the mobile client side.

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.

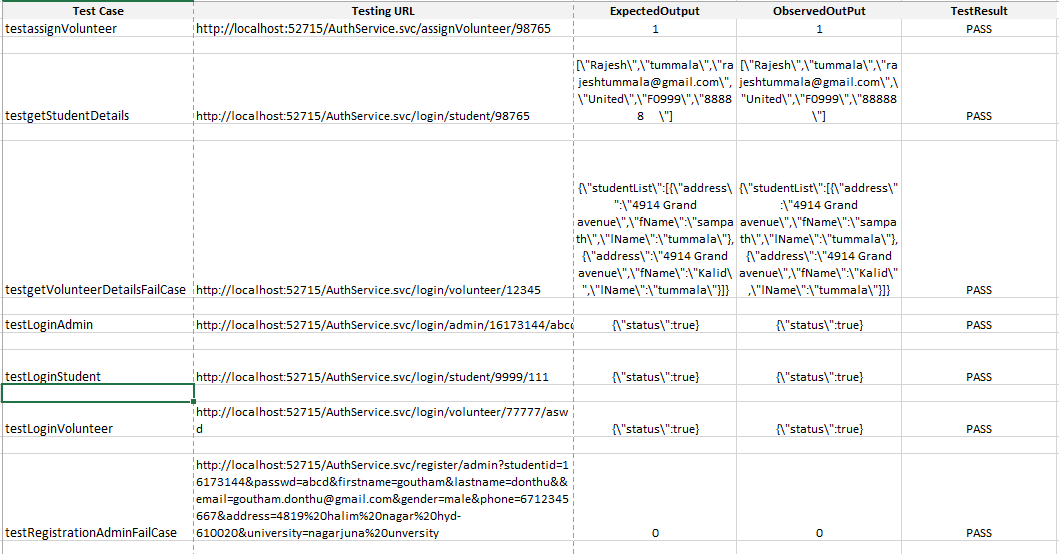


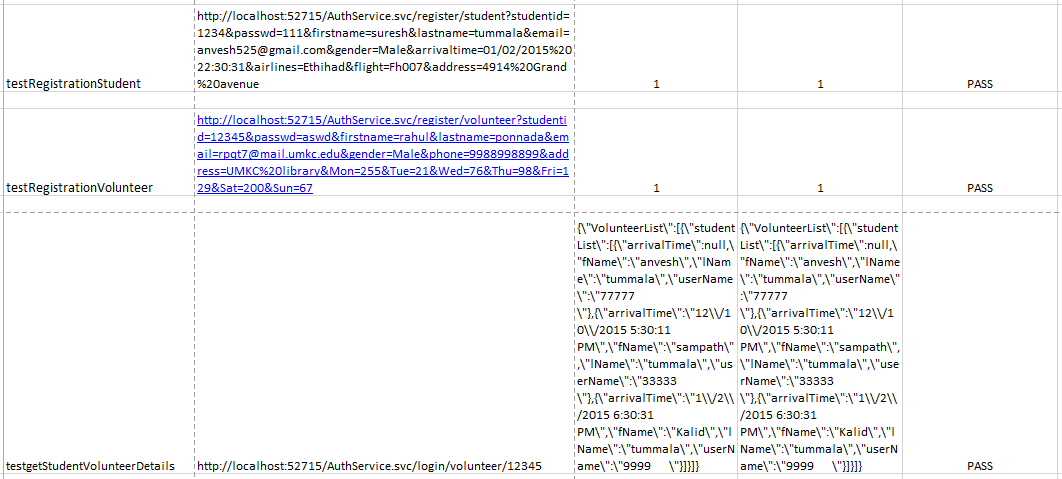
**Implementation of test cases:**

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.

**Test Cases:**

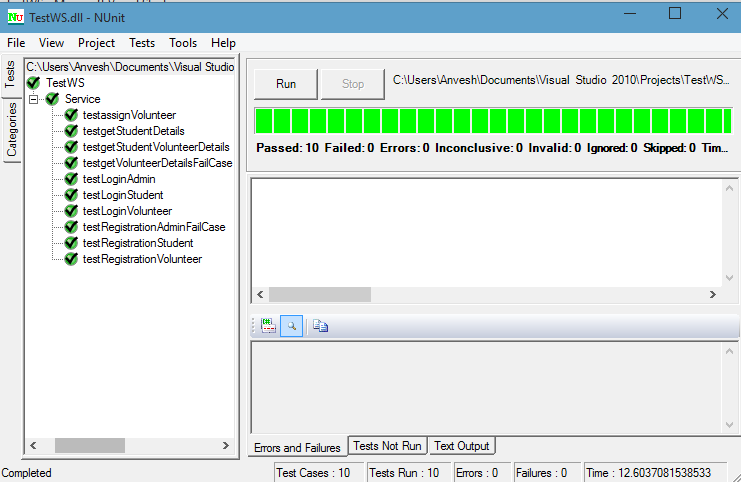




**Testing:**

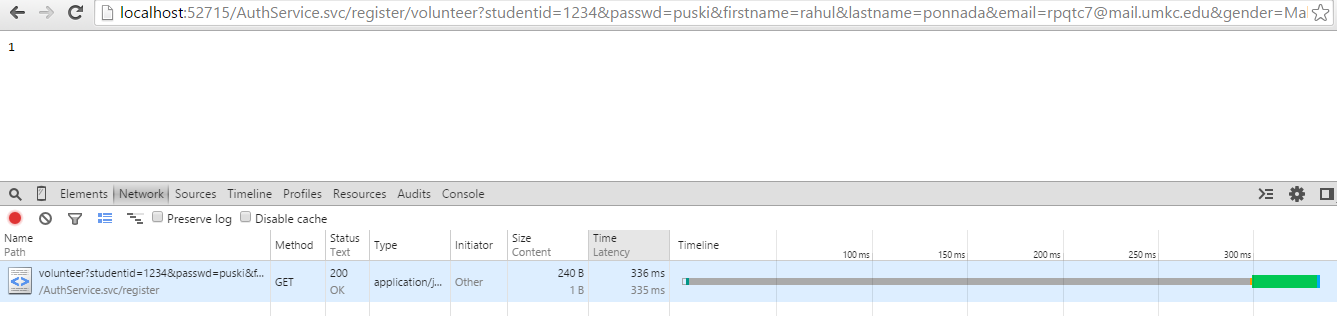
**Functional Testing : Perform Unit testing (using NUnit tool)**

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



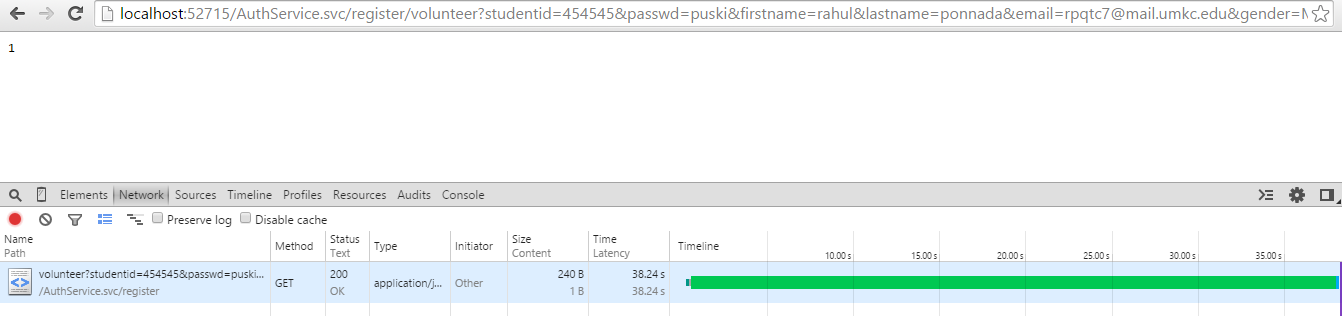
**Deployment Testing : Runtime performance testing**

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



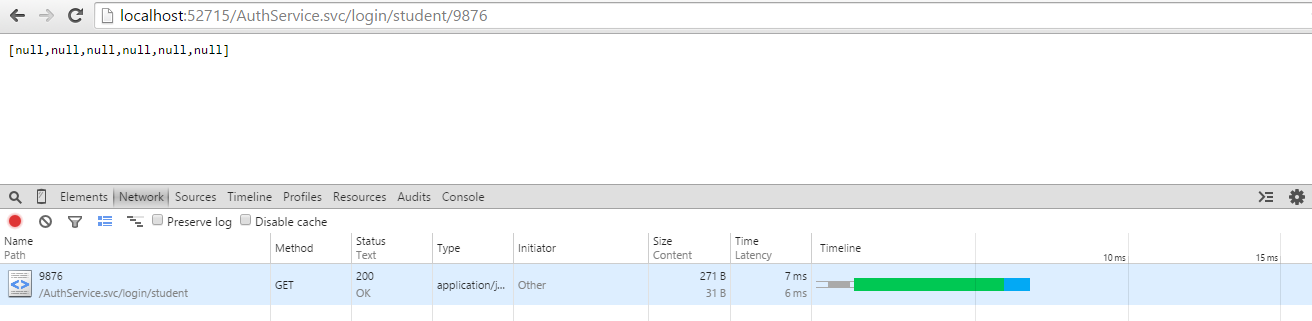
**Volunteer Registration with automatically assigning the students (assigning volunteers for unassigned students)**

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

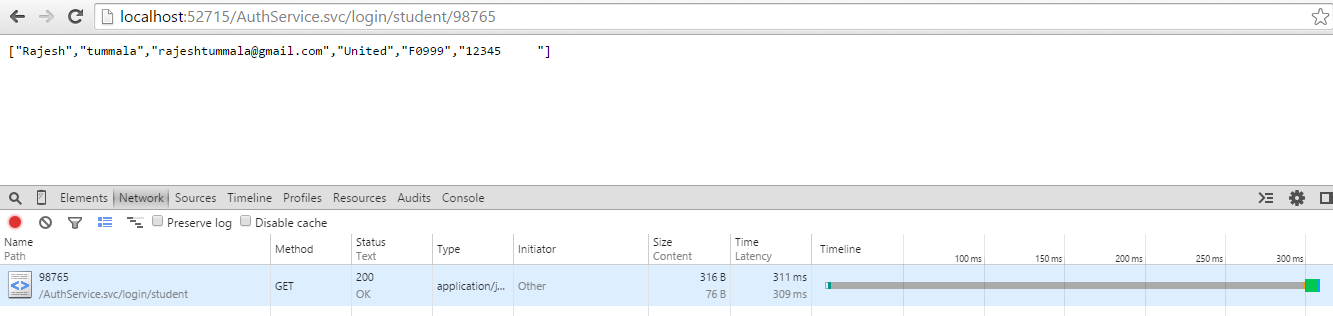


Interesting result: For assigning volunteers for all the 10 students and to send mails(almost 30 mails (3\*10)) it took 38 sec. I think its not that bad, at the same it is not high response time.

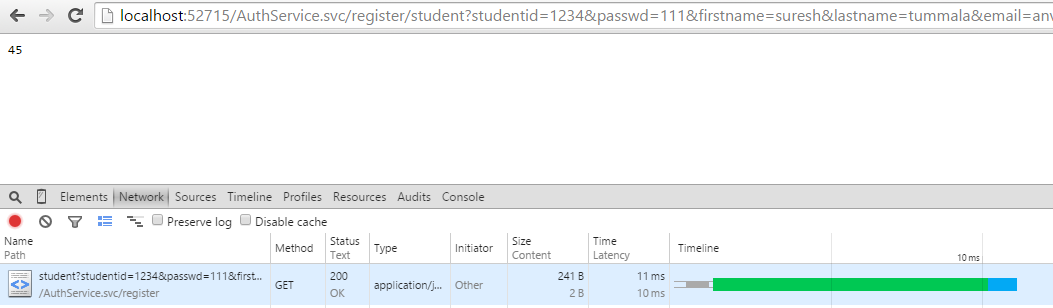
**Login Student response time: *(Unsuccessful Login)***



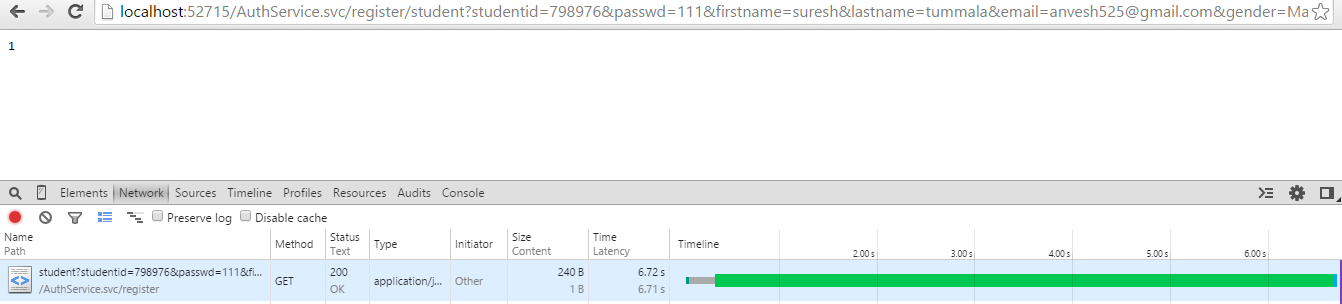
**Login Student response time: *(Successful Login)***



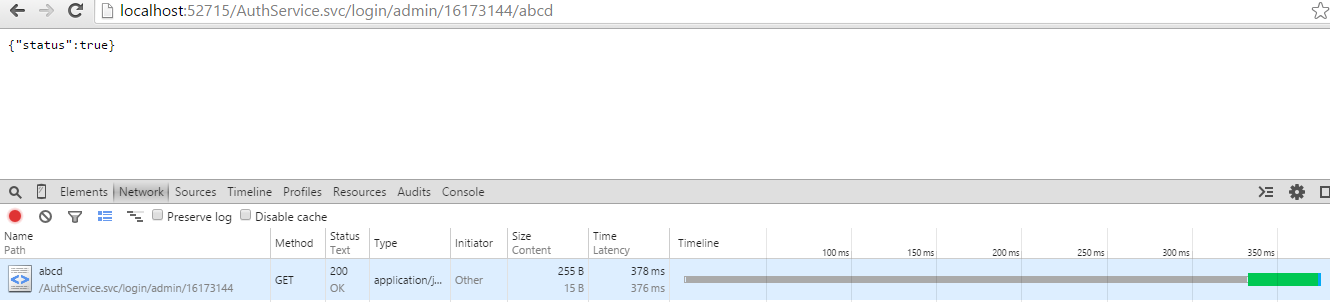
**Student Registration response time: (Failed case)**



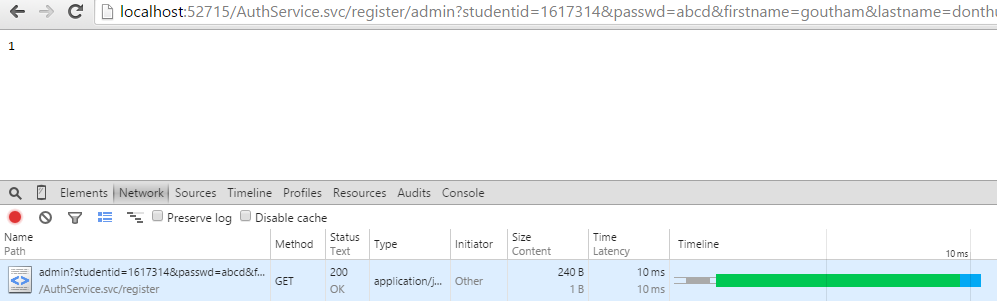
**Student Registration response time: (Successful case)**



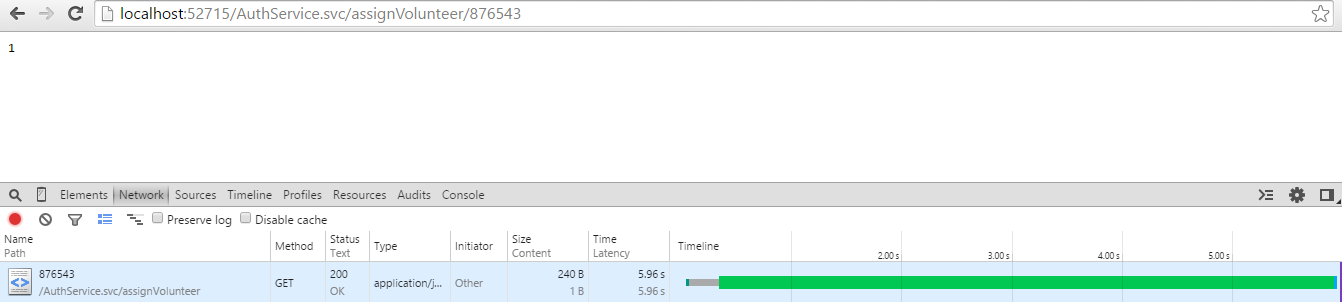
**Admin Login response time:**



**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>

As described in the report there are 4 stories completed in this iteration. You can find url of web services created for this project.

**Webservice URLS:**

**Student Login:**

<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>

**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>

**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>

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 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}

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}

**Report GitHub URL:**

[https://github.com/rpqt7/PickMeUp/blob/master/PG6-Increment 4\_Report.docx](https://github.com/rpqt7/PickMeUp/blob/master/PG6-Increment%204_Report.docx)

**Video Youtube URL:**

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

**Project Management:**

**ScrumDo Link:** <https://app.scrumdo.com/projects/pickmeup/board#/view/iteration/1473>

**Implementation status report:**

**Work Completed:**

**Description:**

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.

Responsibility: Rahul

Time Taken: 20 hrs

Contribution: 100%

1. 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.

Responsibility: Satish

Time Taken: 20 hrs

Contribution: 100%

1. As a Admin, I can unassign volunteers for any particular students in case of any concerns.

Responsibility: Anvesh

Time Taken: 20 hrs

Contribution: 100%

1. As a admin, I can make announcements to all students and volunteers.

Responsibility: Prabha

Time Taken: 20 hrs

Contribution: 100%

**Work To be completed:** None

**Issues/Concerns:**

* We have difficulty in accessing Flight Status as there is no freely available web service for that.
* Performance issues due to high processing of data at server side. We may think of improvements of performance by including JavaScript logic processing at user side.
* Build issues were there.