Rakesh Reddy Bandi (06)

Ravi kiran Yadavalli (67) Sri Naga Sarvani Jakkula (30) Nihar Dudam (16)

|  |  |
| --- | --- |
| Multi Messenger | Abstract  Multi Messenger application is one of the best approaches for making people interact with their friends without any interruption. |

# **CS 5551 FOURTH INCREMENT REPORT**

# Group: 1 Project Title: **Multi Messenger Application**

Nihar Dudam (16) Sri Naga Sarvani Jakkula (30)

Rakesh Reddy Bandi (06) Ravi kiran Yadavalli (67)

## **Introduction**

Multi Messenger application is one of the best approaches for making people interact with their friends without any interruption. Messaging friends through various applications by switching over them is tedious. This overhead can be minimized by our Multi Messenger Application. A user can message to his friend through various messengers at same time using our application. Our Application mainly aims for the new feature “Search Conversation with a keyword” and displays the result as a whole conversation involving that keyword. Search Conversation with a keyword is the main advantage of our application. It makes the user to gather all the conversation with the keyword matched. This makes the user to easily gather the useful information all at once. By not switching between applications there can be relatively less battery drain. The idea of our project can be found in project proposal document.

The project has been divided into four phases with improving implementation features. For the first iteration of our project we want to complete all design section of the application with login, dashboard page design with synchronizing the local mobile message service into our application. We have chosen the android platform to develop our application. For this first iteration, we have designed the UML class diagram, activity diagram along with wireframes. We concentrated mainly on the design part which play a major role in implementing our project.

## **Objectives/Features**

In recent times, the impact of mobile applications in socializing and communicating has been huge. With increase in demand for more sophisticated applications in managing and summarizing among tens of popular social networking mediums, it is great to have a single application which could aggregate and summarize through multiple messengers for different users.

Our application aims in providing a platform which could enable a user to send and receive messages from all friend lists across multiple instant messengers and synchronize them under a single window per unique person (friend) in your contacts overall. Our application also primarily focusses on analyzing and processing the conversation data per contact (friend) from multiple messengers. The application user would be able to “search” with a keyword in the window dedicated per friend of Multi Messenger App, to fetch the relevant data from the multiple messengers.

Below are the objectives of Multi Messenger Application:

1. To synchronize contacts from different instant messengers.
2. To enable each user with Send and Receive messages functionality from single window for the synchronised messenger.
3. Reduce the latency and overhead of switching across messengers.
4. To provide a user friendly and rich application.
5. To provide the user with the functionality of searching the conversation with a specific keyword to fetch the all relevant data across multiple messengers.

## **Project Background and Related Work**

There have been multiple instant Messenger Aggregators like Meebo, Nimbuzz which have been successful in providing the users to converse through different messengers from a single login. Yet, there is no application which could let users to view messages from multiple messengers in a single window.

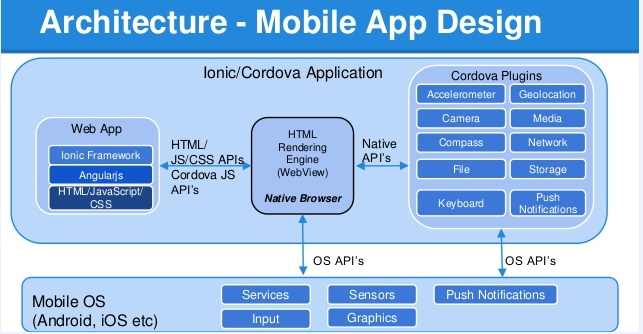
Achieving synchronization on conversation from multiple messengers would further lead an application to smartly search and process the information for better the knowledge on conversation history. To implement our project, we need several API’s to integrate. For this second increment we want to implement the synchronize operation of local mobile SMS service within our application and fetch the contacts from GTalk and use the contacts API plugin to the fetch the contacts from Device. Instant Messenger Applications such as Yahoo and Twitter do provide their messenger API’s to third Party Applications. We plan to receive the necessary OAuth credentials from respective applications and provide messaging facility through our application. Our upcoming deployments would be on IBM’s Blue Mix service which could host Mobile Applications in a highly scalable environment. The database provider for App being MongoDB, which is highly suitable for the mobile content accessibility and retrieval from various mobile nodes.

**Gtalk API** to integrate Gtalk service API in our application we need the Gtalk API from developers.google.com. It needs authentication and authorization of the user. This can be done using OAuth 2.0. The respective credentials such as key for accessing google API and secret key and client id for OAuth 2.0 to implement in the application. We can obtain these credentials from developers.console.com.

**ARCHITECURTE DIAGRAM**

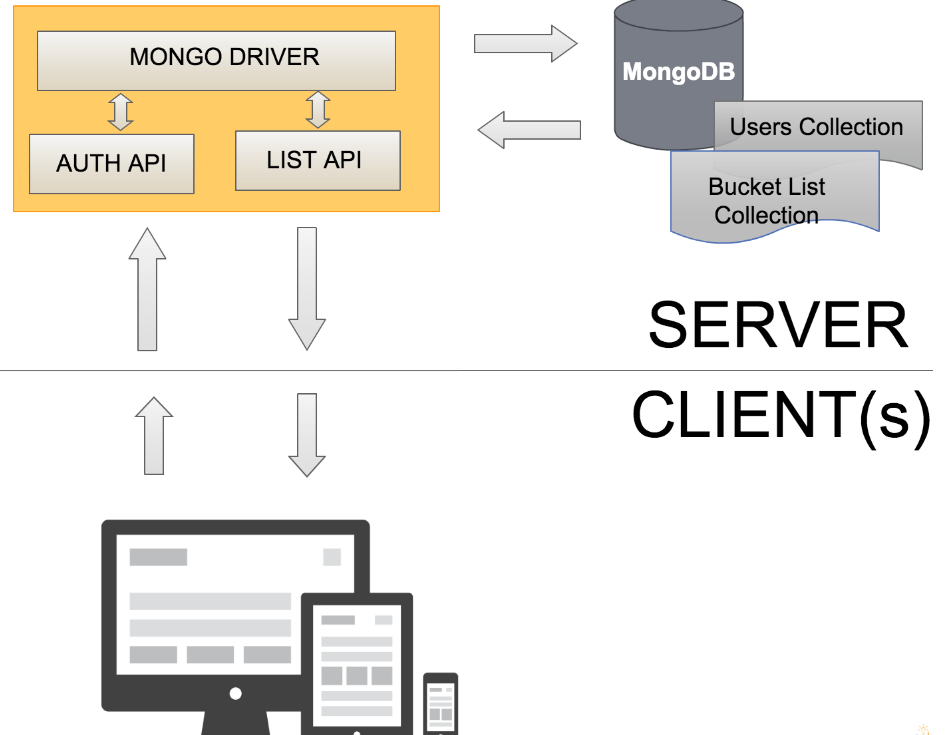


Mash Ups could deliver scalable content over hybrid architure.Mash Ups are the technology for the future.

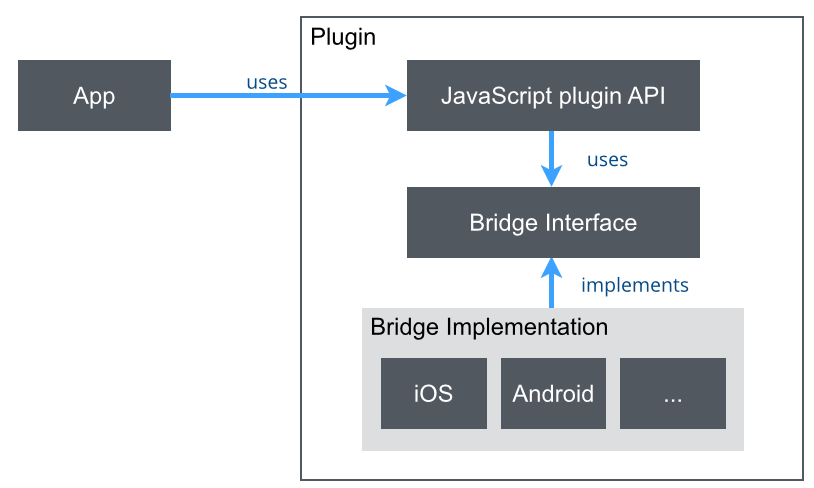


|  |
| --- |
| TECHNOLOGY USED |
| Ionic/Cordova |
| Angular JS, JavaScript |
| HTML 5.0 |
| Android OS and IOS compitable |
| IBM Bluemix |
| Mongo DB |

The above table mentions about the various technologies used in our application



The above mentioned diagram illustrates about architecture.



The above figure illustrates the plugin usage in our Profile Application

## **Proposed System**

1. Requirement Specification:

* Functional Requirements:
  + 1. User should have a single sign in.
    2. User must able to view his recent chats.
    3. User must able to view all messenger’s friends list.
    4. User must able to search in his friends list.
    5. User must able to search in the recent conversation with a keyword.
    6. User’s search should yield in a collaborative and meaningful data from all the instant messages.
    7. User must be able add various messenger accounts.
    8. User must be able to configure respective accounts.
    9. User must be able to sync his phone contacts.
    10. User must be able to sync messenger contacts.
    11. User must be able to chat consecutively through various instant messengers.

## **Import Existing Services/API**

* 1. **Google Talk API**

Google has provided its Instant Messenger Gtalk’s API to third party Developers. Gtalk’s API uses XMPP protocol to communicate the messages from Google’s Server. Unlike the most of instant messaging protocols today, XMPP is defined in an open standard and uses an open systems approach of development and application, by which anyone may implement an XMPP service and interoperate with other organizations' implementations. Because XMPP is an open protocol, implementations can be developed using any software license; although many server, client, and library implementations are distributed as free and open-source software, numerous freeware and commercial software implementations also exist.

Below is the architecture of XMPP Client Server Model:

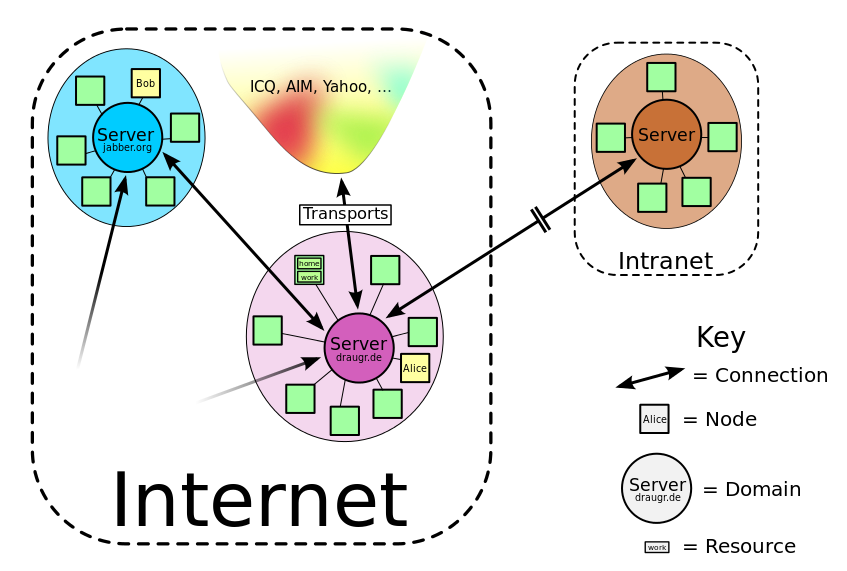


Figure 1: XMPP Architecture Diagram

The google OAuth is an Authorization API which is a two-step process in order to connect with GTalk.

* To Create a Token scoped for Chat Login.
* To Use the generated Token to authentication Google Talk Connection Servers.

* 1. **Twitter API**

Twitter is one of the most popular social Networking Site. In the current Increment we had been working on integrating three API’s of Twitter .Below are the three API’ and their functionality.

* + 1. *: GET direct messages/Show*

Returns a single Directed message with the ID. Like the /1.1/direct\_messages.format request, this method will include the user objects of the sender and recipient. This method requires an access token with RWD (read, write & direct message) permissions. Consult The Application Permission Model for more information.

Resource URL: https://api.twitter.com/1.1/direct\_messages/show.json

* + 1. *: GET direct messages/Sent*

Returns the 20 most recent directed messages sent by authenticating the user. Includes detailed information about the sender and recipient user. You can request up to 200 direct messages per call, up to a maximum of 800 outgoing DMs.

Resource URL: https://api.twitter.com/1.1/direct\_messages/sent.json

* + 1. *: POST direct messages/new*

Sends a new direct message to the specified user from the authenticating user. Requires both the user and text parameters and must be a POST. Returns the sent message in the requested format if successful.

Resource URL: <https://api.twitter.com/1.1/direct_messages/new.json>

Twitter requires Twitter OAuth to authenticate the Twitter User.

**5.3 Instagram API:**

The Instagram API uses the OAuth 2.0 protocol for simple, but effective authentication and authorization. OAuth 2.0 is much easier to use than previous schemes and developers can start using the Instagram API almost immediately. The one thing to keep in mind is that all requests to the API must be made over SSL (http**s**:// not http://).

### Do you need to authenticate?

The Instagram API requires authentication - specifically requests made on behalf of a user. Authenticated requests require an **access\_token**. These tokens are unique to a user and should be stored securely. Access tokens may expire at any time in the future.

### 1.Direct your user to authorization URL:

<https://api.instagram.com/oauth/authorize/?client_id=CLIENT-ID&redirect_uri=REDIRECT-URI&response_type=code>

### 2.Receive the redirect from Instagram:

<http://your-redirect-uri?code=CODE>

If your request for approval is denied by the user, then we will redirect the user to your redirect\_uri with the following parameters:

error: access\_denied

error\_reason: user\_denied

error\_description: The user denied your request

<http://your-redirect-uri?error=access_denied&error_reason=user_denied&error_description=The+user+denied+your+request>

### **3. Request the access\_token**

Now you need to exchange the code you have received in the previous step for an access token. In order to make this exchange, you simply have to POST this code, along with some app identification parameters, to our access\_token endpoint. These are the required parameters:

client\_id: your client id

client\_secret: your client secret

grant\_type: authorization\_code is currently the only supported value

redirect\_uri: the redirect\_uri you used in the authorization request. Note: this has to be the same value as in the authorization request.

code: the exact code you received during the authorization step.

This is a sample request:

curl -F 'client\_id=CLIENT\_ID' \

-F 'client\_secret=CLIENT\_SECRET' \

-F 'grant\_type=authorization\_code' \

-F 'redirect\_uri=AUTHORIZATION\_REDIRECT\_URI' \

-F 'code=CODE' \

<https://api.instagram.com/oauth/access_token>

* 1. **LinkedIn:**

The REST API is the heart of all programatic interactions with LinkedIn. All other methods of interacting, such as the JavaScript and Mobile SDKs, are simply wrappers around the REST API to provide an added level of convienence for developers. As a result, even if you are doing mobile or JavaScript development, it's still worth taking the time to familiarize yourself with how the REST API works and what it can do for you.

Requesting data from the APIs

Unless otherwise specified, all of LinkedIn's APIs will return the information that you request in the XML data format.

GET

https://api.linkedin.com/v1/people/~

sample response

<?xml version="1.0" encoding="UTF-8"?>

<person>

<id>1R2RtA</id>

<first-name>Frodo</first-name>

<last-name>Baggins</last-name>

<headline>Jewelery Repossession in Middle Earth</headline>

<site-standard-profile-request>

<url>https://www.linkedin.com/profile/view?id=…</url>

</site-standard-profile-request>

</person>

Add a format=json URL argument to the end of your API call.

Add this HTTP header to your API call: x-li-format: json

For example:

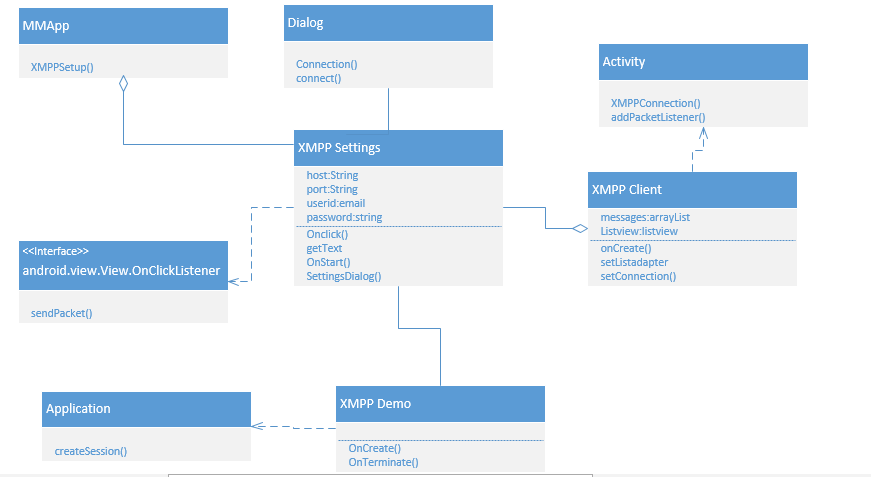
GET

[**https://api.linkedin.com/v1/people/~?format=json**](https://api.linkedin.com/v1/people/~?format=json)

## **Detail Design of Services**

**Class Diagrams**:

1. Class Diagram for establishing XMPP client connection and sending a message.



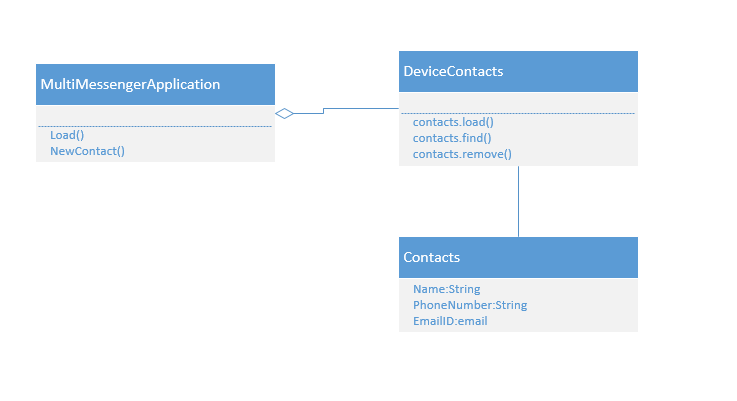
Description:

We have 3 main classes that needs to be implemented in order to establish a XMPP client connection and exchange packets.

They are:

* XMPP Settings: This class is used to setup XMPP connection with the Gtalk server by getting the necessary configuration parameters from the user. It extends Dialog class of Smack API
* XMPP Client: This class is used to send and receive packets of messages from the user and the user at other end. It extends Activity class of Smack API
* XMPP Demo: This class extends Application class of Smack API in which it is used to get the gtalk server interact with our host application.

1. Class Diagram for Synchronizing Contacts from the device



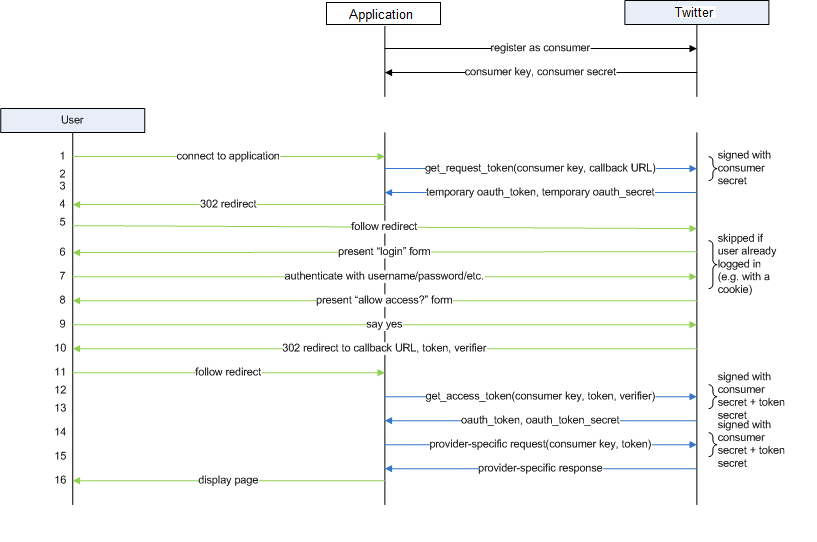
Description:

To synchronize device contacts from the phone, we implement this by using 3 classes.

* MultiMessengerApplication: This is our application class which is used to interact with the device to retrieve the contacts
* DeviceContacts: This class is used to retrieve the contacts from the device Database in order to display in our application
* Contacts: This class has the details of the contact.

**Sequence Diagrams:**

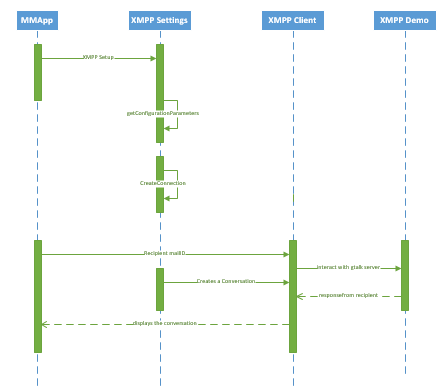
1. Sequence diagram for connecting to Twitter from the Application



Description:

* Here first our application register as a consumer with twitter developers account and get the consumer key and consumer secret.
* Later whenever user wants to connect his app with twitter he signs in through our application and grant access to our application.
* The application request the twitter for access token by passing the consumer key and secret and callback URL also as a parameter .
* Twitter returns temporary oauth\_token and Oauth\_secret to our application.
* Using this we follow to Twitter Login page were the user enters his/her credentials to give access to our application.
* And our application requests the access tokens from the twitter.
* Twitter sends access tokens both Oauth\_token & Oauth\_token\_secret to our application.
* Application stores these tokens in MongoDb database.
* We use this tokens for sending the further requests such as fetching the contacts and sending direct messages.

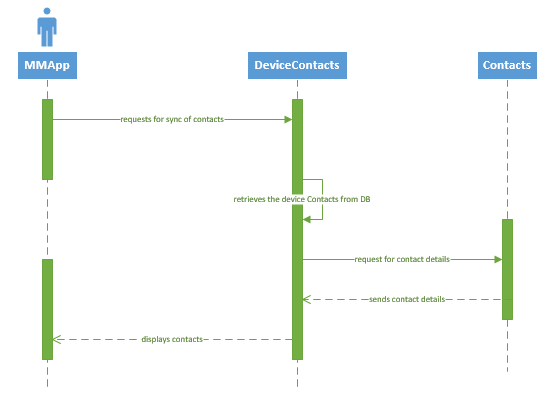
1. Sequence diagram for connecting to gtalk through XMPP



Description:

Here from user interface we request a setup for XMPP connection by entering configuration parameters. These are passed into XMPP settings class in which it creates a connection of XMPP client and sends the request for accessing gtalk to XMPP client class. Then we send the recipient mail ID entered from UI. It establishes a conversation with the recipient. It is displayed on the UI.

1. Sequence Diagram for Synchronizing Device contacts



Description:

Here, application requests for synchronizing the device contacts from the mobile. The contacts are retrieved from the DB through Device Contacts class then this requests the details of the contacts from Contacts class. The contacts are displayed on the UI.

## **Testing**

* **Unit Testing:**

Unit Testing is software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use.

* + - * Testing login :

**Test Data**

|  |  |  |
| --- | --- | --- |
| **Test Id** | **Library ID** | **Password** |
| T001 | tarun | A |
| T002 | tarun | Abc |
| T003 | nihar | Nihar123 |
| T004 | Nihar | nihar123 |
| T005 | Ravi | Raviultimate |
| T006 | Ravi | Ravinotultimate |

**Test Conditions**

|  |  |  |
| --- | --- | --- |
| **Test ID** | **Condition to be tested** | **Expected Result** |
| T001 | Login Validation | Login Successful |
| T002 | Login Validation | Login Failed |
| T003 | Login Validation | Login Successful |
| T004 | Login Validation | Login Failed |
| T005 | Login Validation | Login Successful |
| T006 | Login Validation | Login Failed |

* Synchronize Contacts page:

1. **Test Data :** Click load, don’t click load
2. **Test Conditions :**

|  |  |  |
| --- | --- | --- |
| **Test ID** | **Condition to be tested** | **Expected Result** |
| T001 | Sync Contacts Validation  On clicking Load | Successful |
| T002 | Sync Contacts Validation  On not clicking Load | Successful |

* + - * Twitter Functionality:

**Test Data**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Id** | **Host** | **Username** | **Password** |
| T001 | www.twitter.com | [rakesh@gmail.com](mailto:rakesh@gmail.com) | raki |
| T002 | www.twitter.com | [rakeshreddys](mailto:rakesh@gmail.com) | raki |
| T003 | www.app.com | [rakesh@our.com](mailto:rakesh@our.com) | raki |
| T004 | www.twitter.com | [rakesh@gmail.com](mailto:rakesh@gmail.com) | \*\*\* |
| 0000T005 | www.twitter.com | [rakesh@yah.com](mailto:rakesh@yah.com) | raki |
| T006 | www.twitter.com | Empty | raki |
| T007 | www.twitter.com | Empty | Empty |
| T008 | Empty | Empty | Empty |
| T009 | www.twitter.com | [rakesh@gmail.com](mailto:rakesh@gmail.com) | Empty |

**Test Conditions**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test ID** | **Condition to be tested** | **Expected Result** | **Test Cycle** | | | |
|  |  |  | S | 1 | 2 | 3 |
| T001 | Twitter Validation | Successful |  | | | |
| T002 | Twitter Validation | Successful |  | | | |
| T003 | Twitter Validation | Failed |  | | | |
| T004 | Twitter Validation | Failed |  | | | |
| T005 | Twitter Validation | Failed |  | | | |
| T006 | Twitter Validation | Failed |  | | | |
| T007 | Twitter Validation | Failed |  | | | |
| T008 | Twitter Validation | Failed |  | | | |

* + Performance Testing:

Performance testing is done using Yslow plugin of Google Chrome. This testing done to login page, register page and for Gtalk page where we will be sending messages to various other users.

## **Implementation**

**Twitter API:**

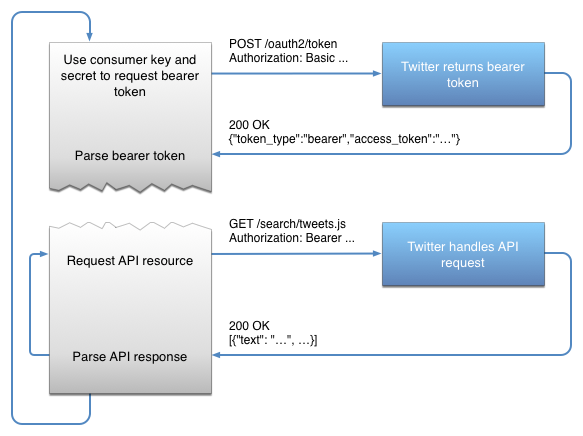
Twitter REST APIs provide programmatic access to send and receive direct receive messages ,to read and write Twitter data, to update statuses, to retrieve status, author a new Tweet, read author profile and follower data, and more. The REST API identifies Twitter applications and users using OAuth; responses are available in JSON.

Implementation Steps:

1. We need to create a project for our application in order to access twitter data in our application, we will need to get a twitter application Consumer key and Consumer Secret key and configure it to use .You need to have an Twitter account for creating a project.
2. There are mainly five steps for using this Yahoo API.
   1. Creating a Twitter Application Open Authentication (OAuth) API Key.
   2. Authenticate our application using the consumer key and consumer secret key.
   3. Creating a new session and obtain the access tokens from the twitter.
   4. Obtaining contact list.
   5. Sending and receiving the Direct messages.
3. Creating a Twitter Application Open Authentication (OAuth) API Key:

We can get Customer Key and Customer secret for an individual user by creating an app and selecting option of creating API key in <http://apps.twitter.com>

1. Authenticate with the Twitter servers

There are two API calls that are required to authenticate with the Twitter Direct OAuth API. The first obtains a Pre-Authorized Request Token (PART), and requires your login username, password, and the OAuth Consumer Key (API Key) that was just generated through developer.twitter.com

1. **Authorization header has been wrapped**:

POST /oauth2/token HTTP/1.1

Host: api.twitter.com

User-Agent: My Twitter App v1.0.23

Authorization: Basic eHZ6MWV2RlM0d0VFUFRHRUZQSEJvZzpMOHFxOVBaeVJn

NmllS0dFS2hab2xHQzB2SldMdzhpRUo4OERSZHlPZw==Content-Type: application/x-www-form-urlencoded;charset=UTF-8

Content-Length: 29

Accept-Encoding: gzip

grant\_type=client\_credentials

By calling this we will get a session id which is user for various operations like fetching contact details, sending messages etc.

1. **Authenticate API requests with the bearer token:**

GET /1.1/statuses/user\_timeline.json?count=100&screen\_name=twitterapi HTTP/1.1

Host: api.twitter.com

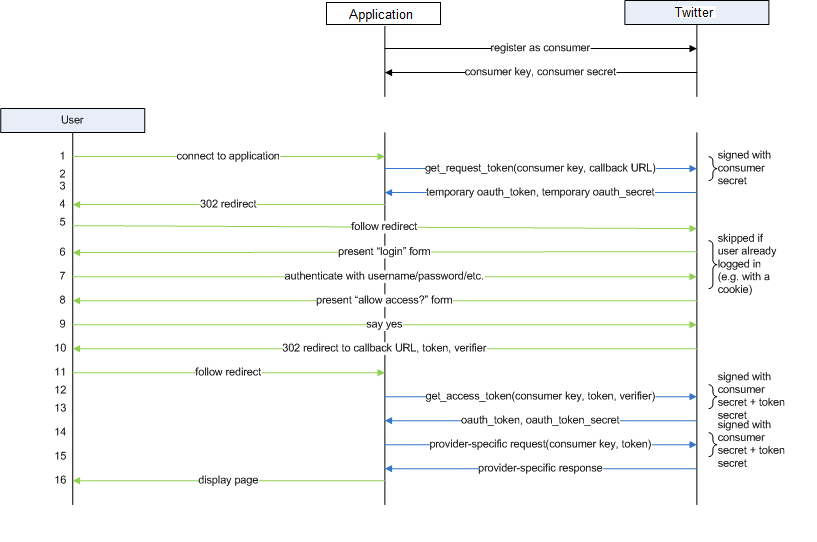
User-Agent: My Twitter App v1.0.23

Authorization: Bearer AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA%2FAAAAAAAAAAAA

AAAAAAAA%3DAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Accept-Encoding: gzip

**Sequence Diagram for Oauth authentication in Twitter:**



## **Deployment**

**Project Management:**

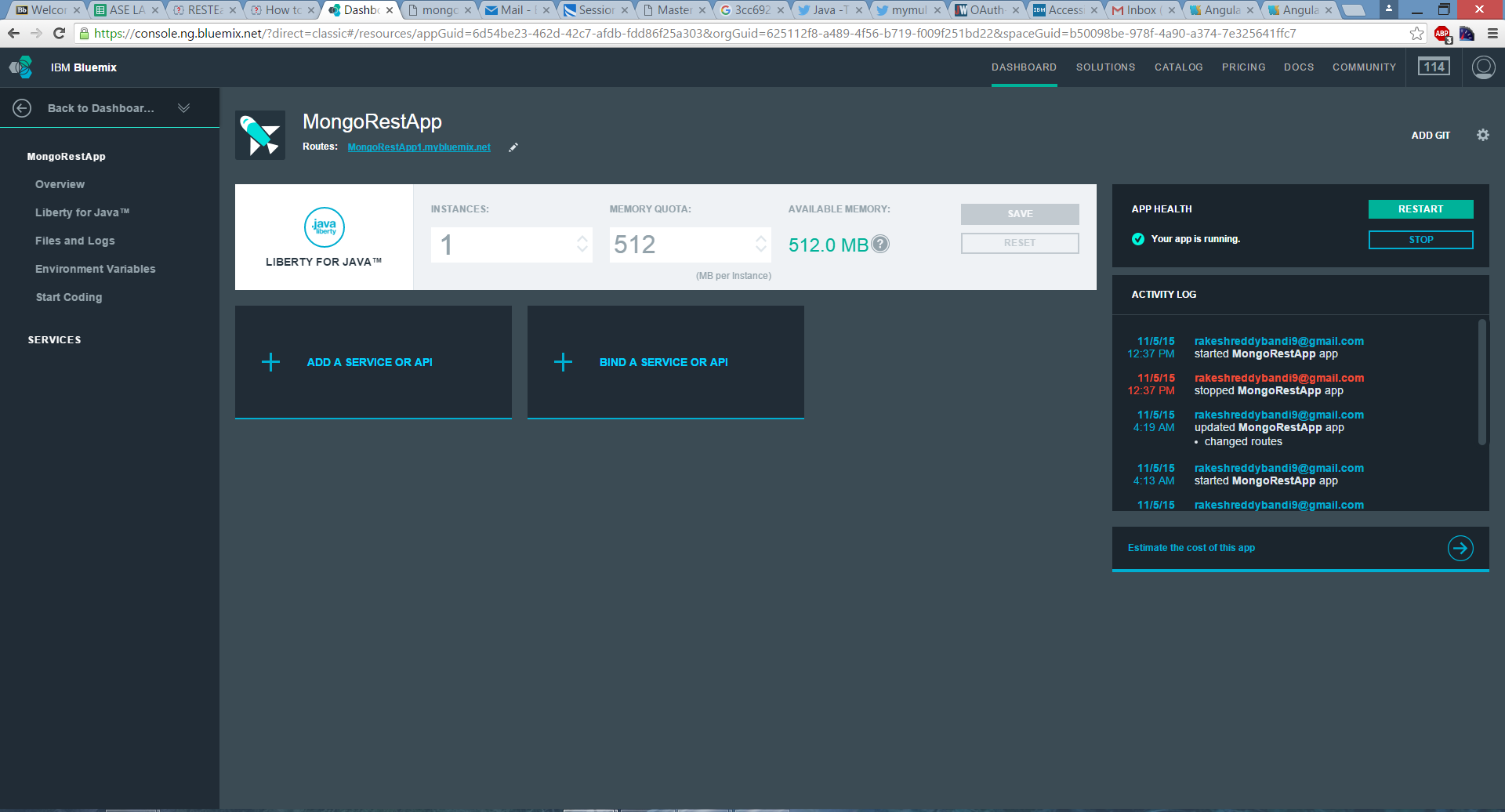
The Project management link of the Project is <http://niharase.kanbantool.com/b/180890-ase_project>.

**GitHub Link:**

The Link to second increment documentation and source code in our GitHub is <https://github.com/rakeshreddybandi/Multi-Messenger-ASE-GROUP1>.

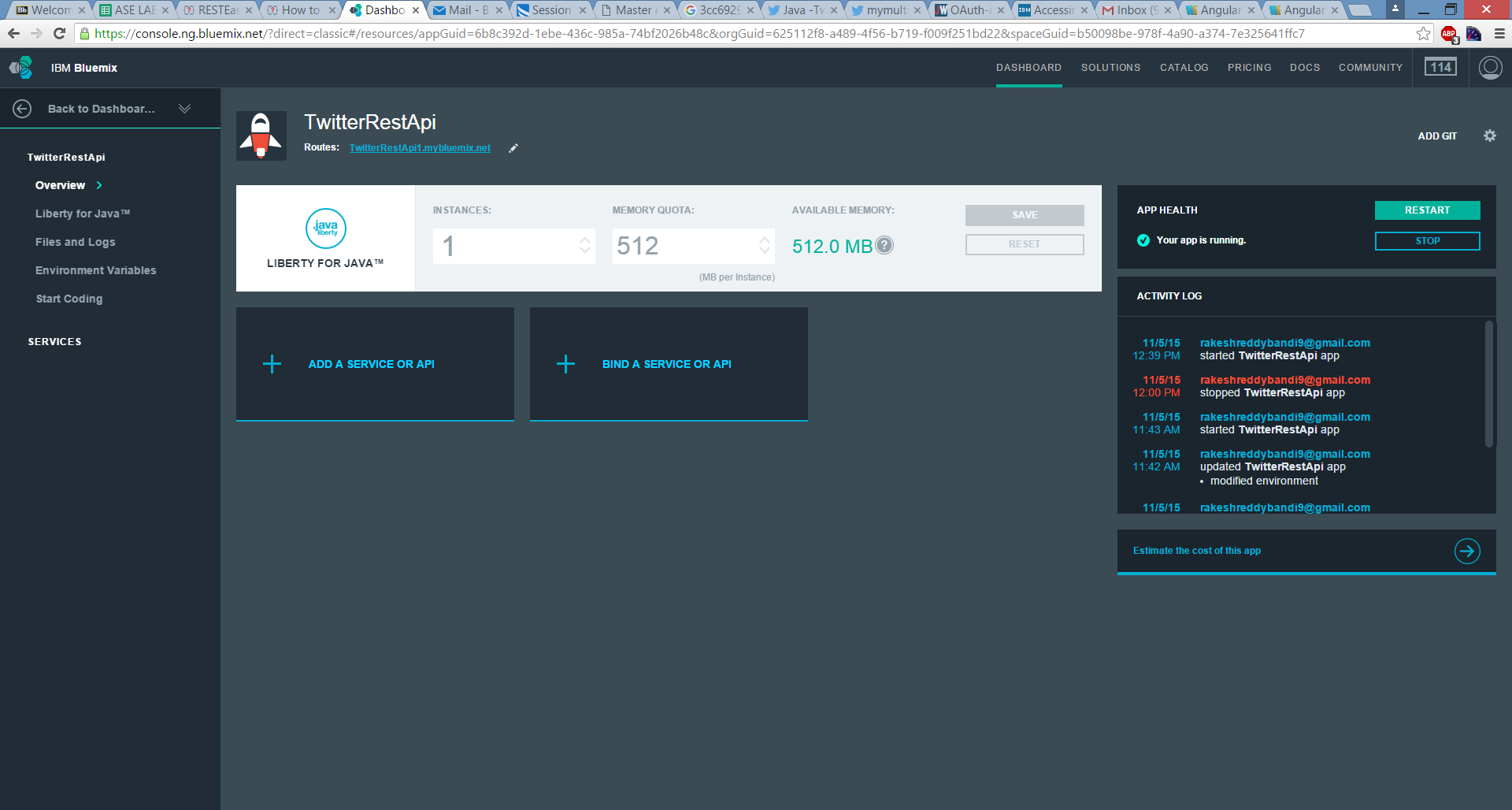
**IBM Bluemix Deployment Link:**

The mongodb database Rest api is deployed in bluemix and the link is: [www.MongoRestApp1.mybluemix.net](http://mongorestapp1.mybluemix.net/)



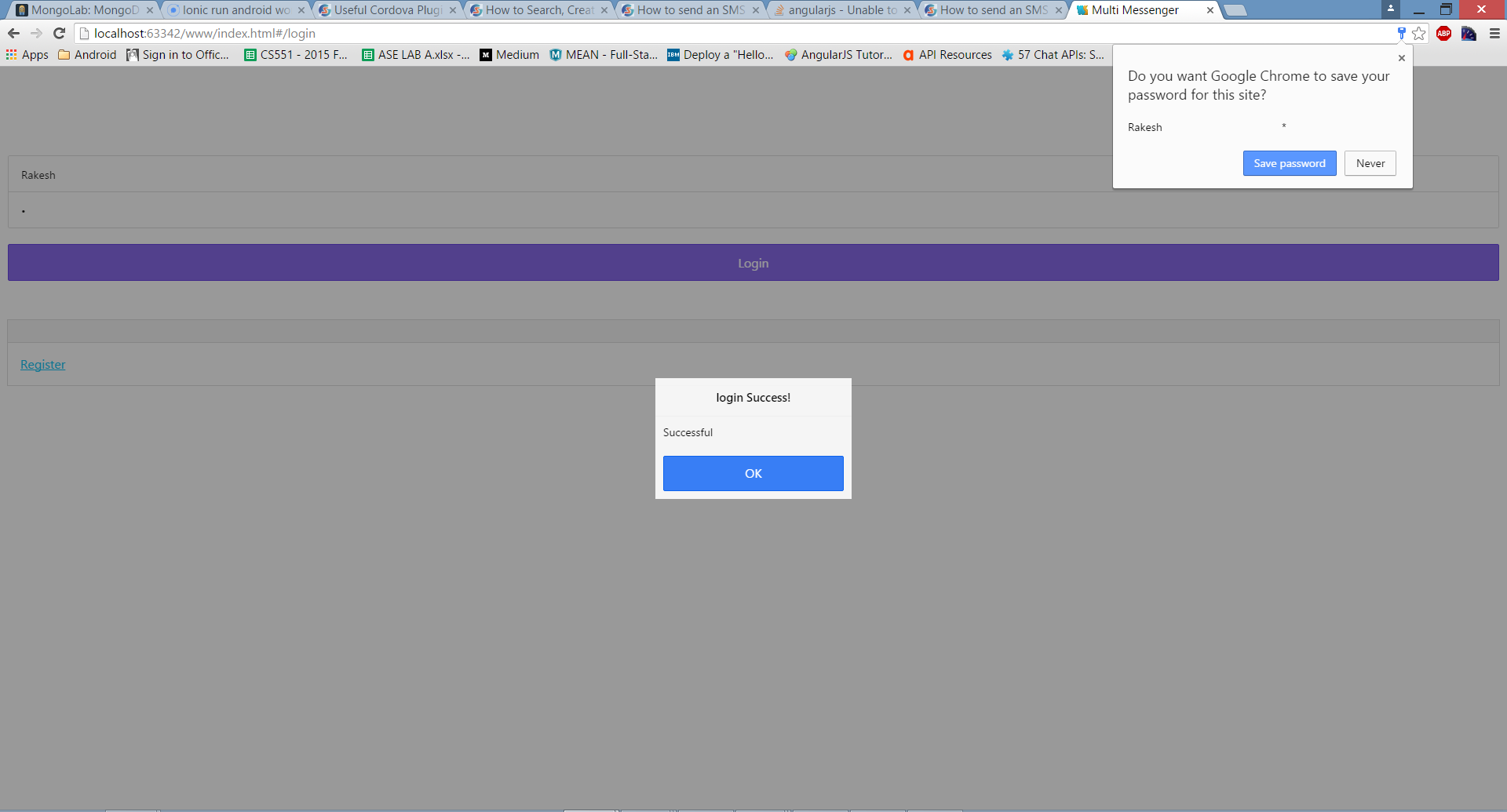
The Twitter Rest api that supports the twitter authentication and twitter calls for getting the contacts and sending the direct messages to the contacts list is:

[www.TwitterRestApi1.mybluemix.net](http://twitterrestapi1.mybluemix.net/)



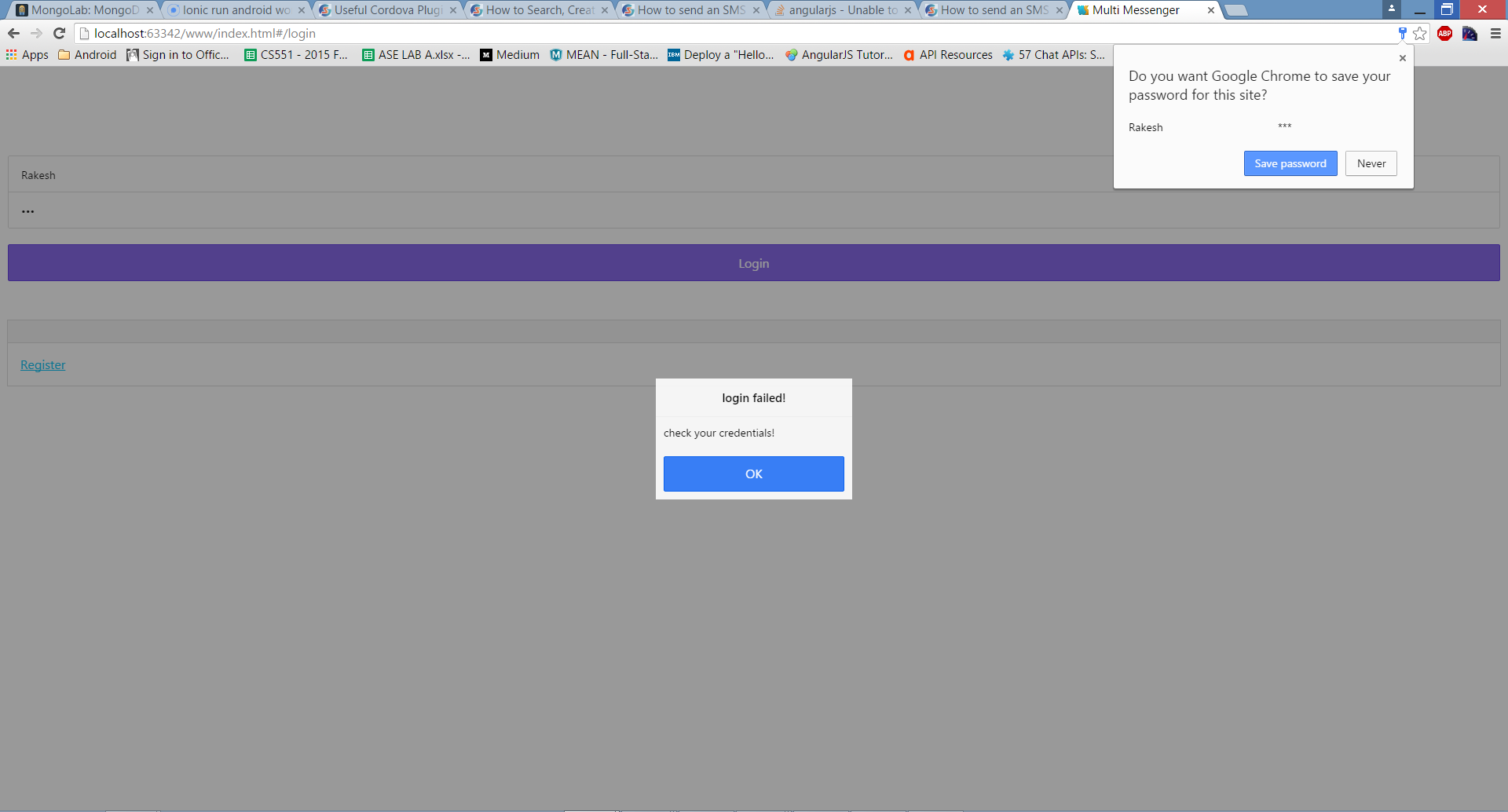
## **Report**

* + - Unit test cases screen shots:
      * Login success :



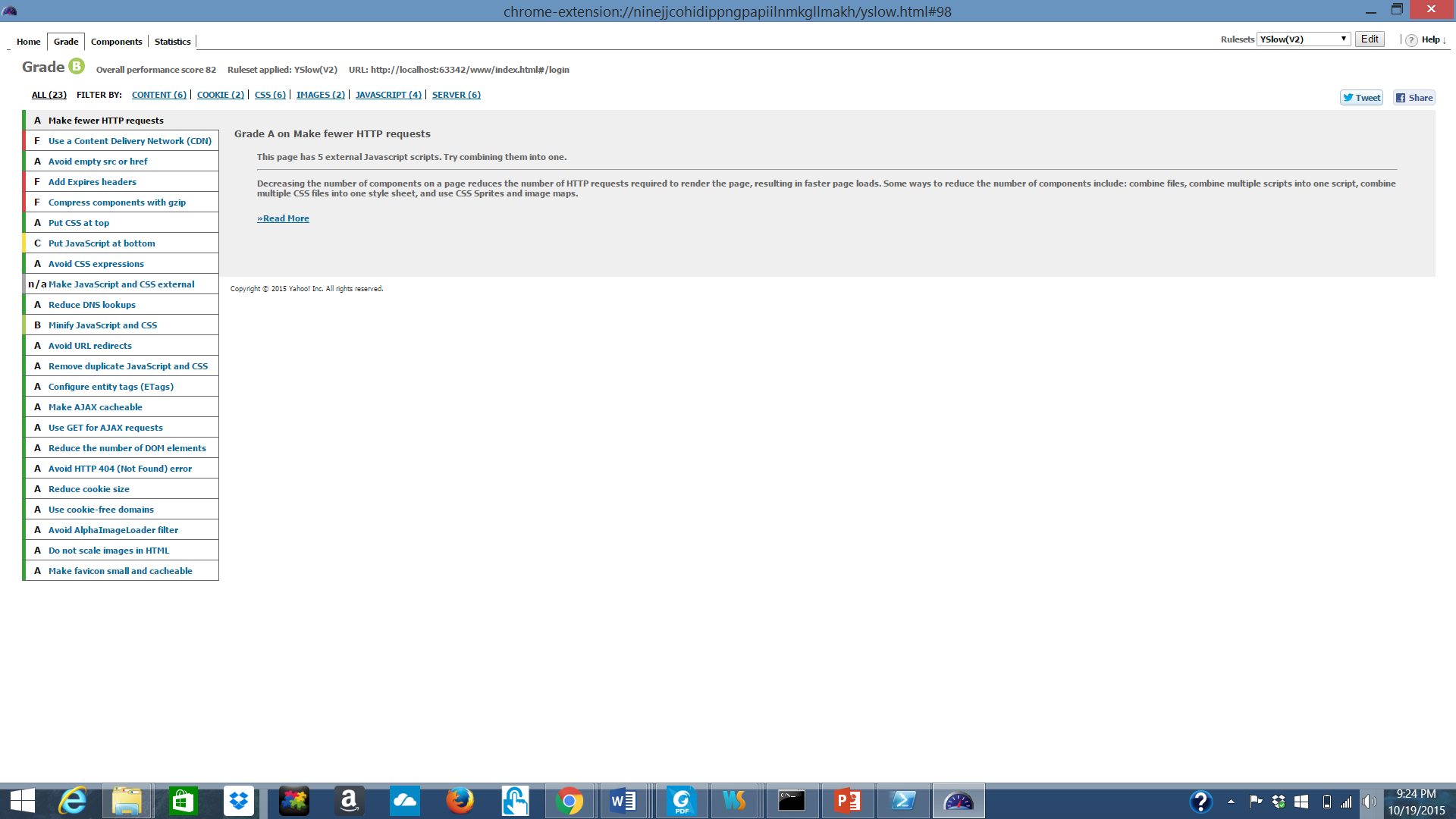
In this we have given the correct credentials username and password of user and logged in to the app. We get the Success message for correct credentials.

* + - * + Login Failure:



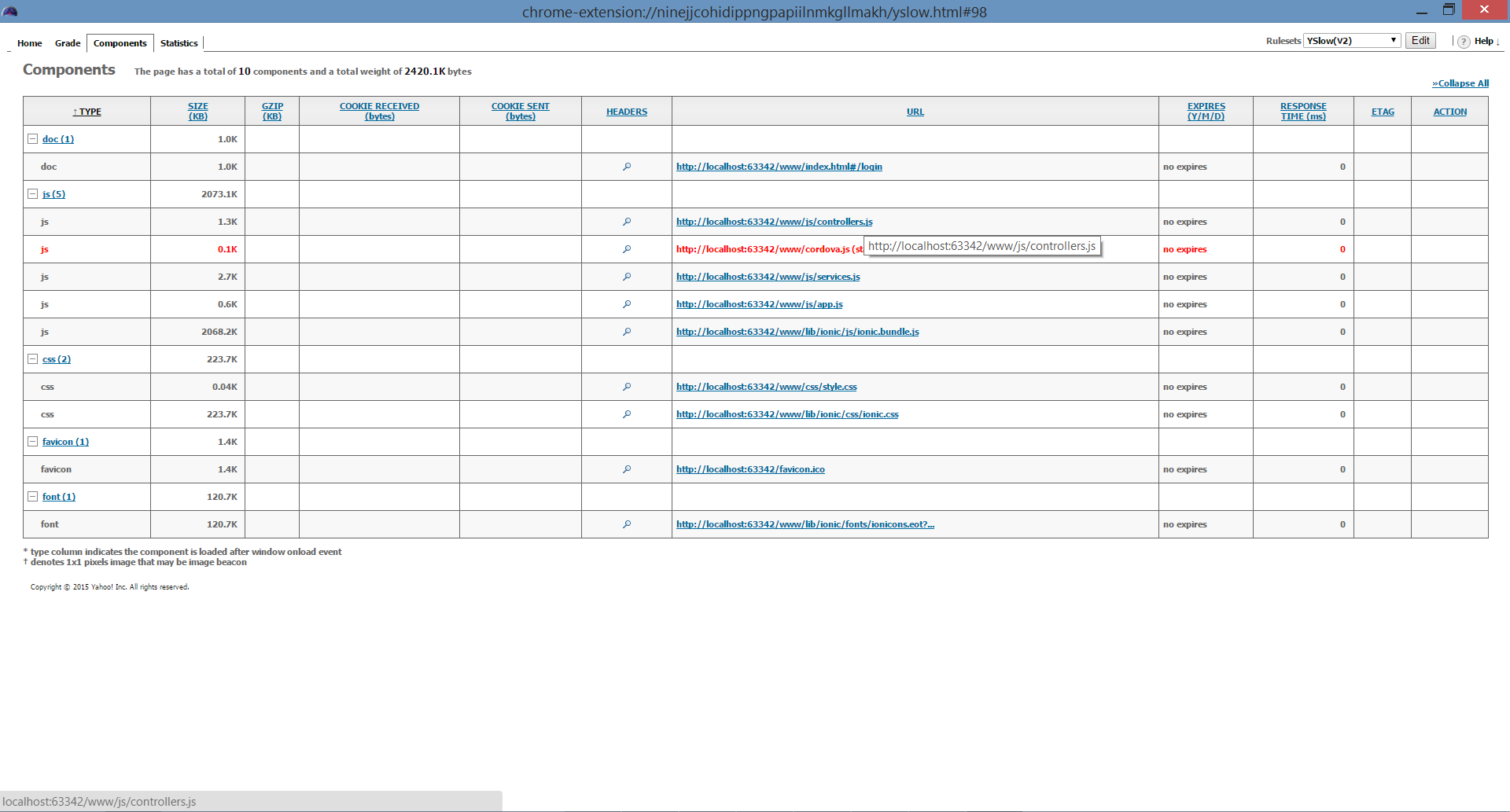
In this we have given the incorrect credentials username and password of user and logged in to the app. We get the Failure message for incorrect credentials.

* + Performance testing Screen Shots:
    - Login:
      * Yslow Grade:



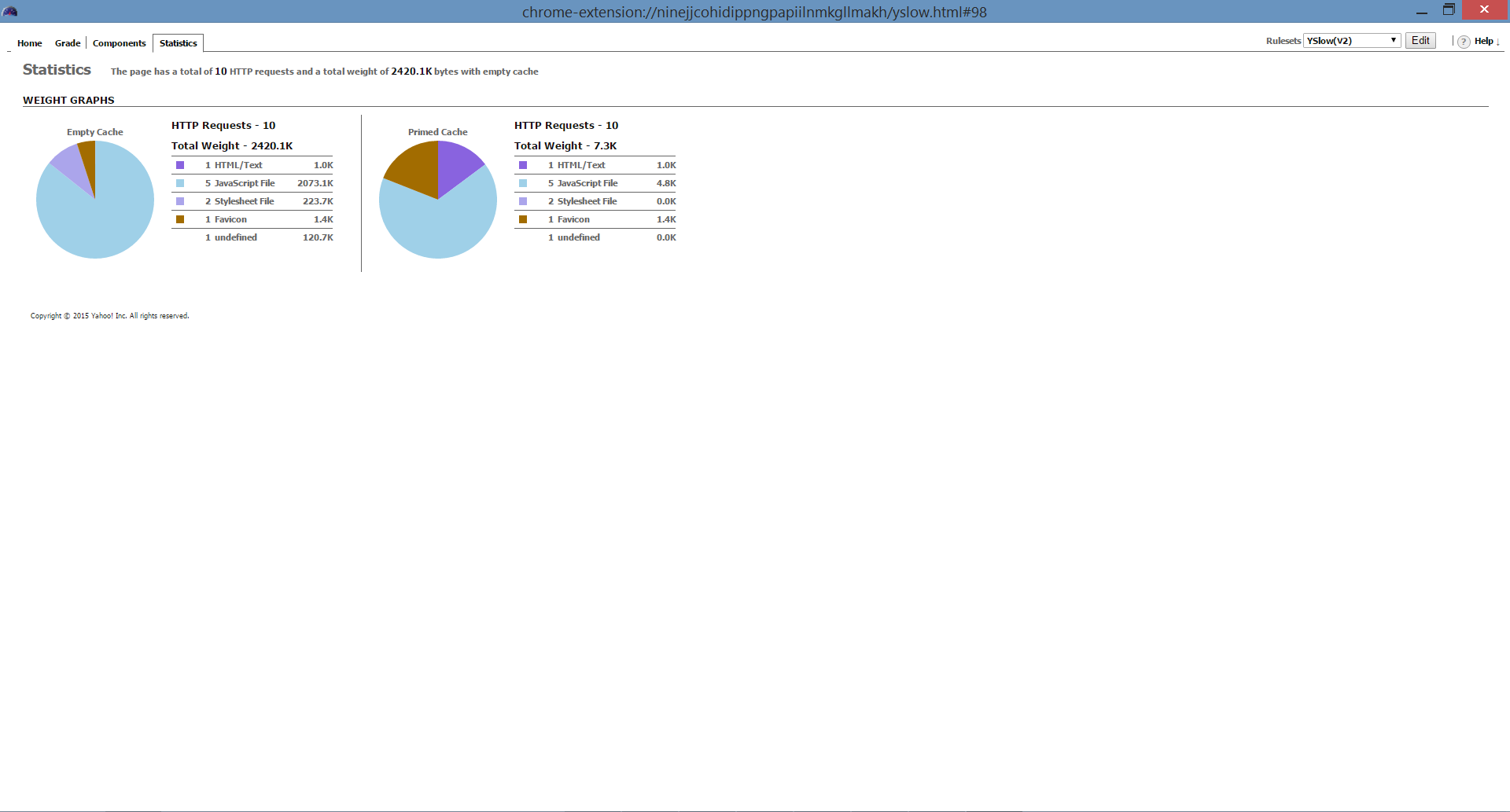
The above diagram describes about the Yslow grade for login

* + - * + Yslow components:



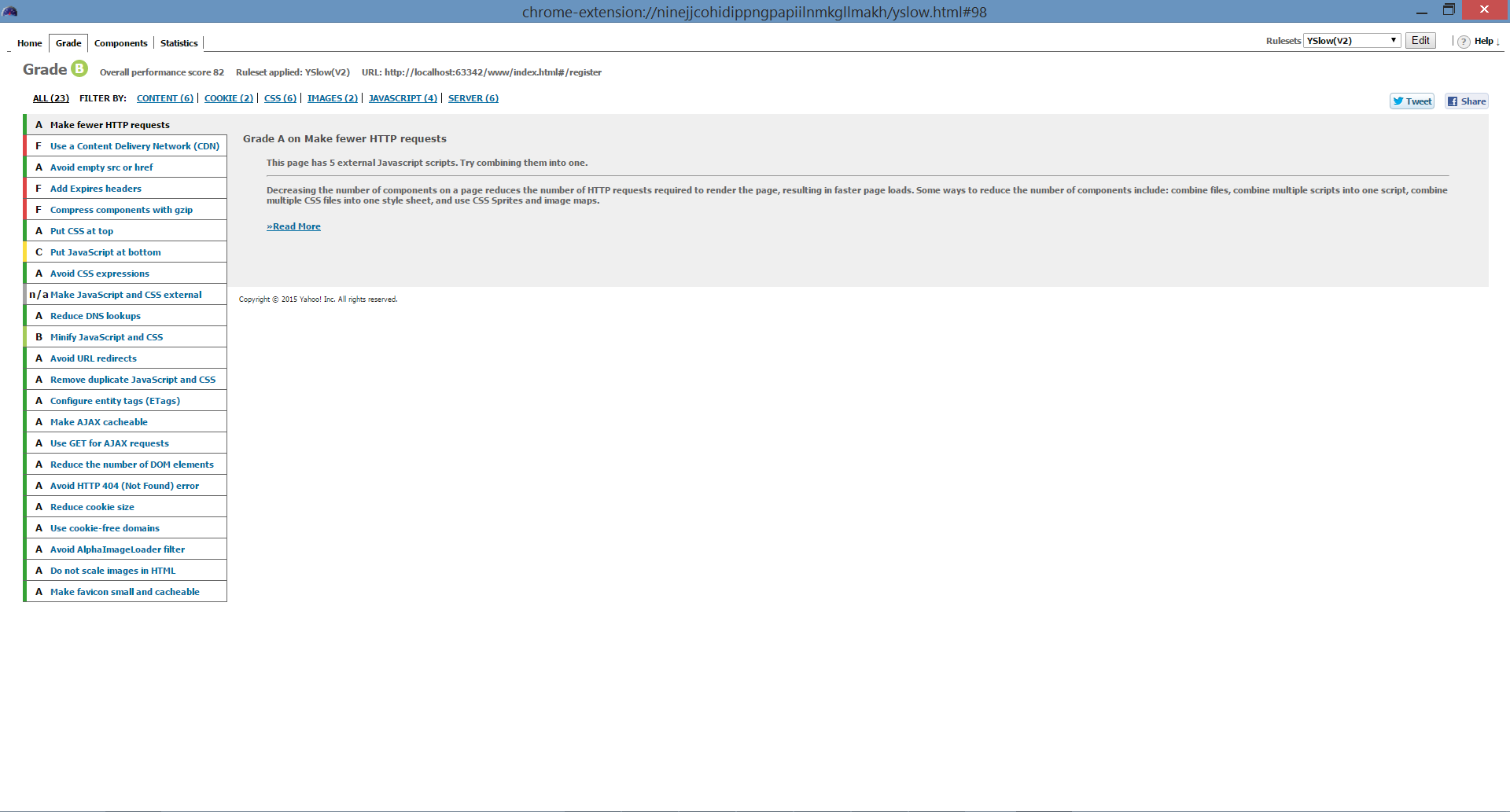
The above diagram explains about Yslow components for login page

* Yslow statistics:



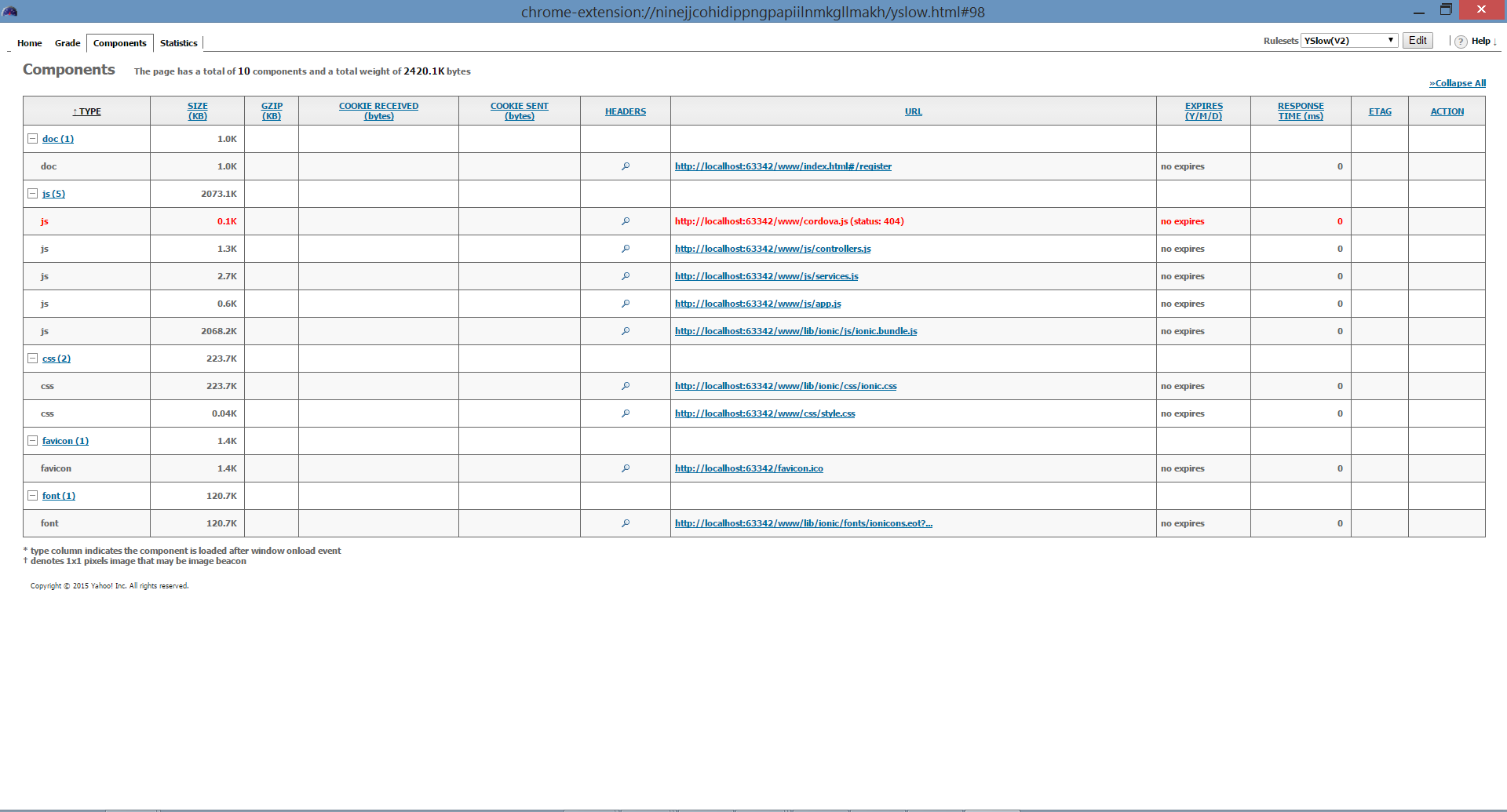
The above figure explains about Yslow statistics for login page.

* Register Page
* Yslow Grade:



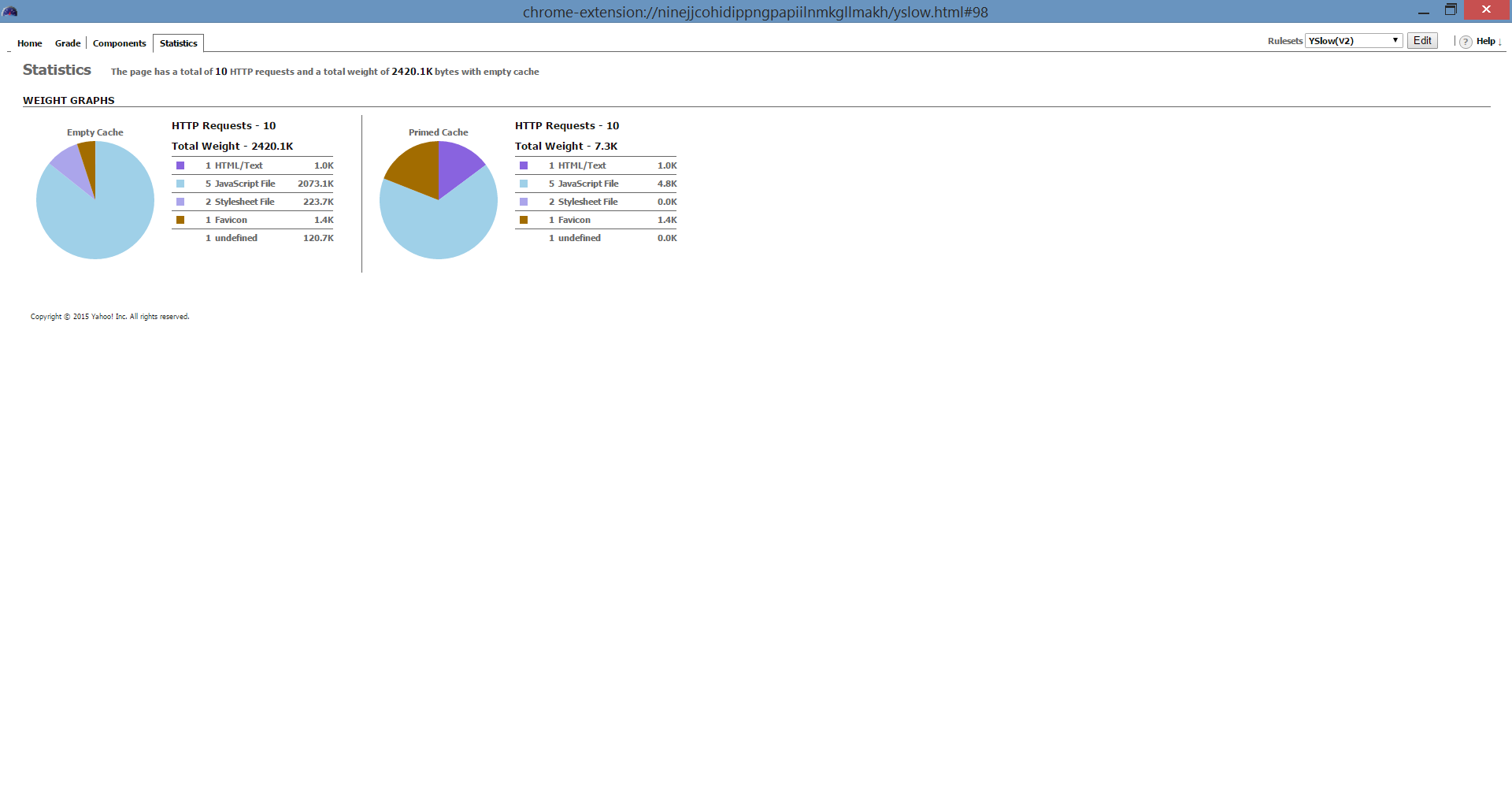
The above figure explains about Yslow grade for register page.

* Yslow component:



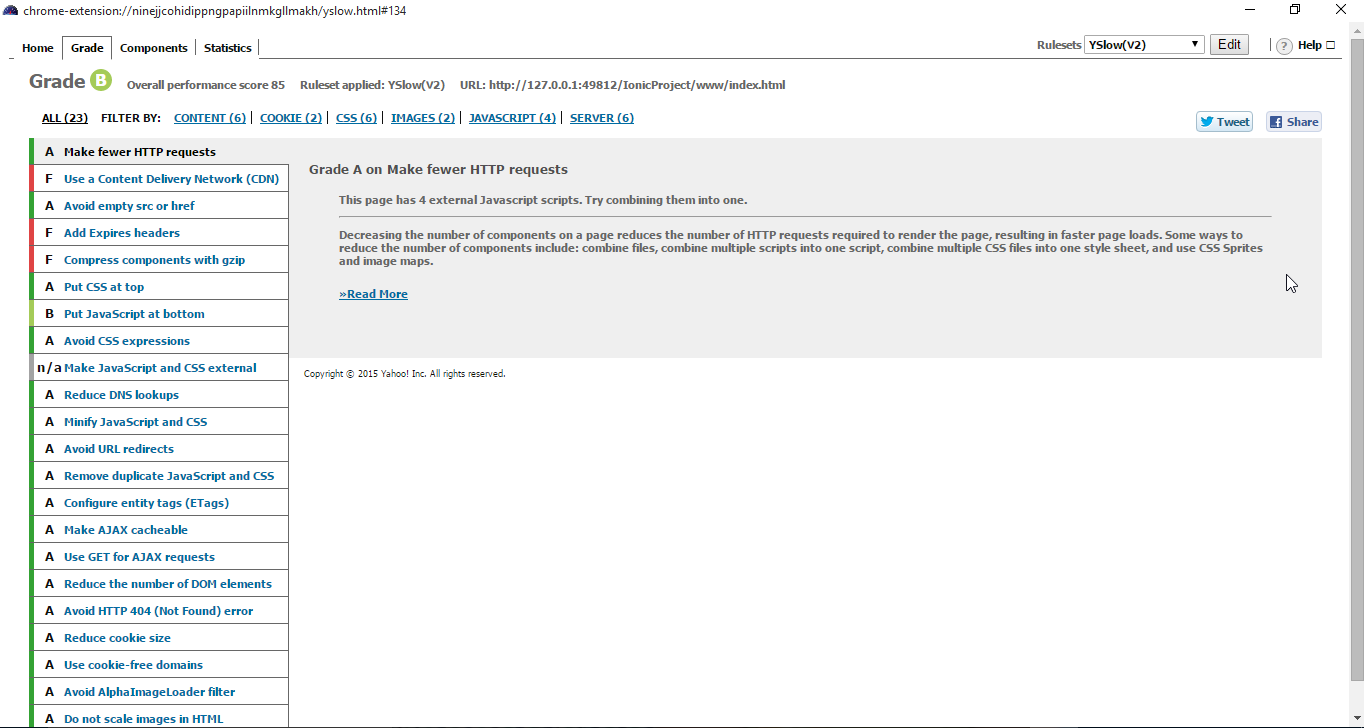
The above figure explains about Yslow components for register page

* Yslow statistics:



The above diagram describes about Yslow Statistics for register page:

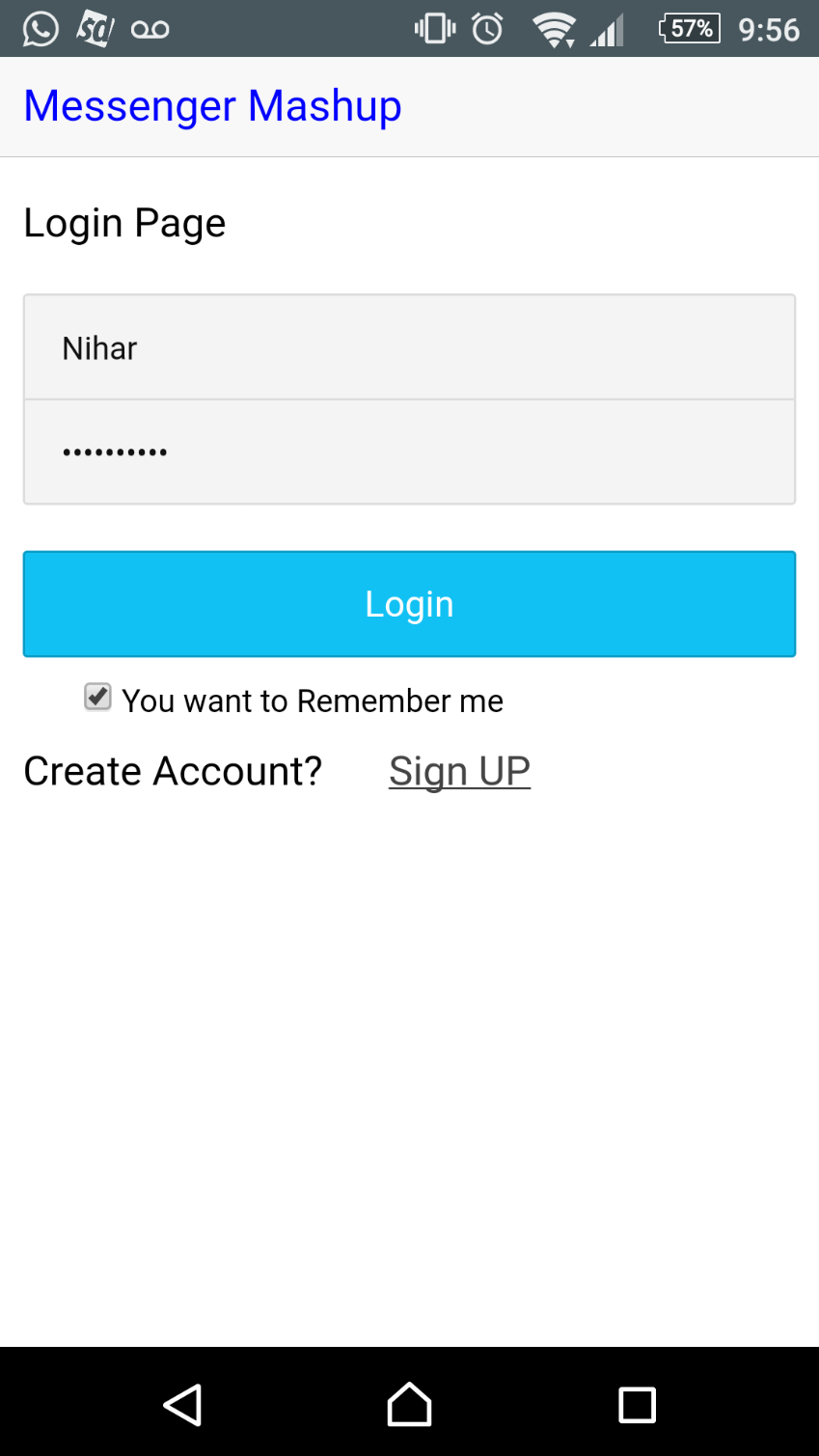
* Contacts Synchronization page :



The above diagram explains about yslow grade for contacts synchronization page.

**Project Implemenation Screenshots:**

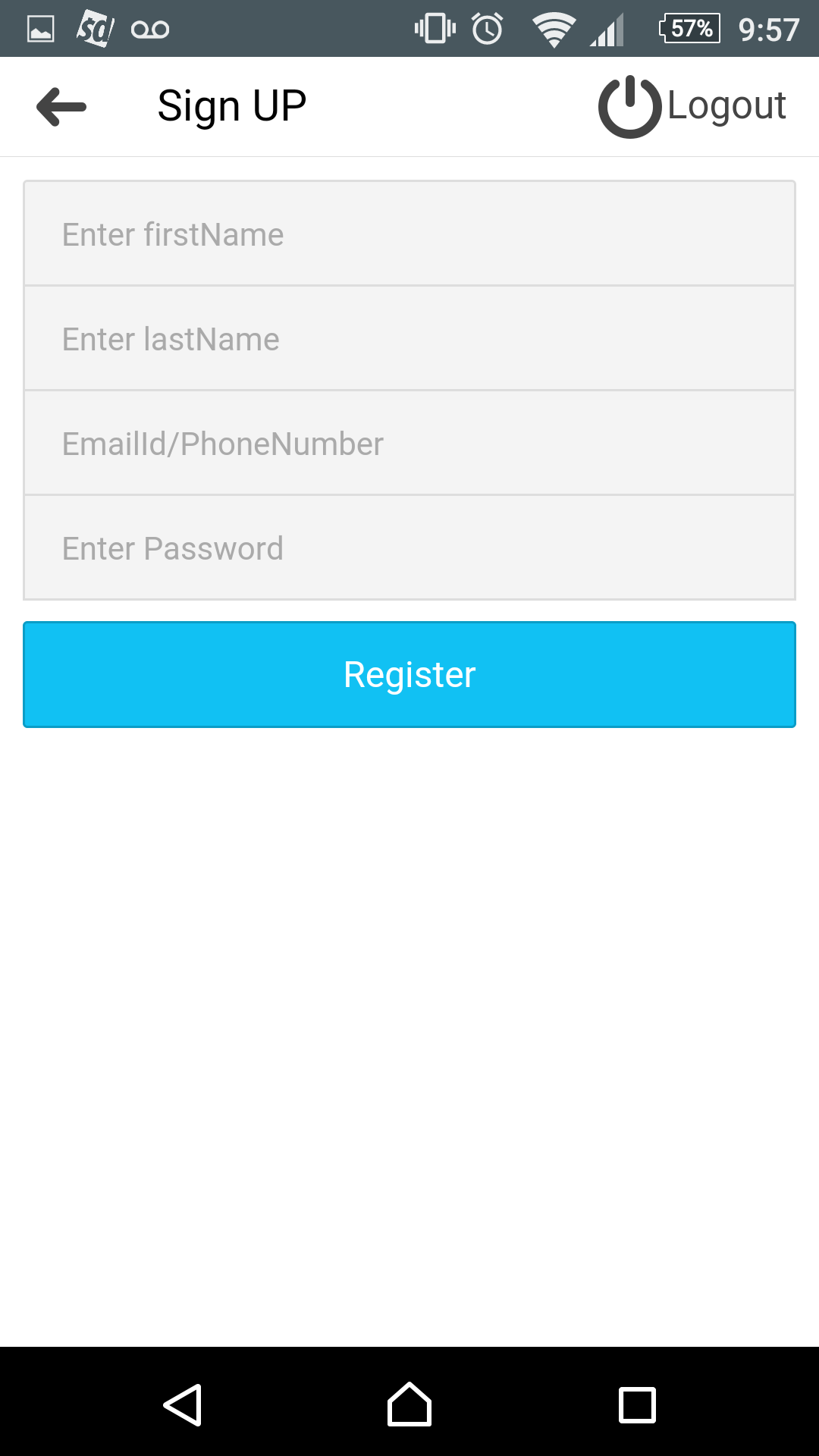
**Mockups:**1. **Login Activity:**

****

Description:

This is the Login Activity of the Multi-Messenger app where user can login to the app using the correct credentials.

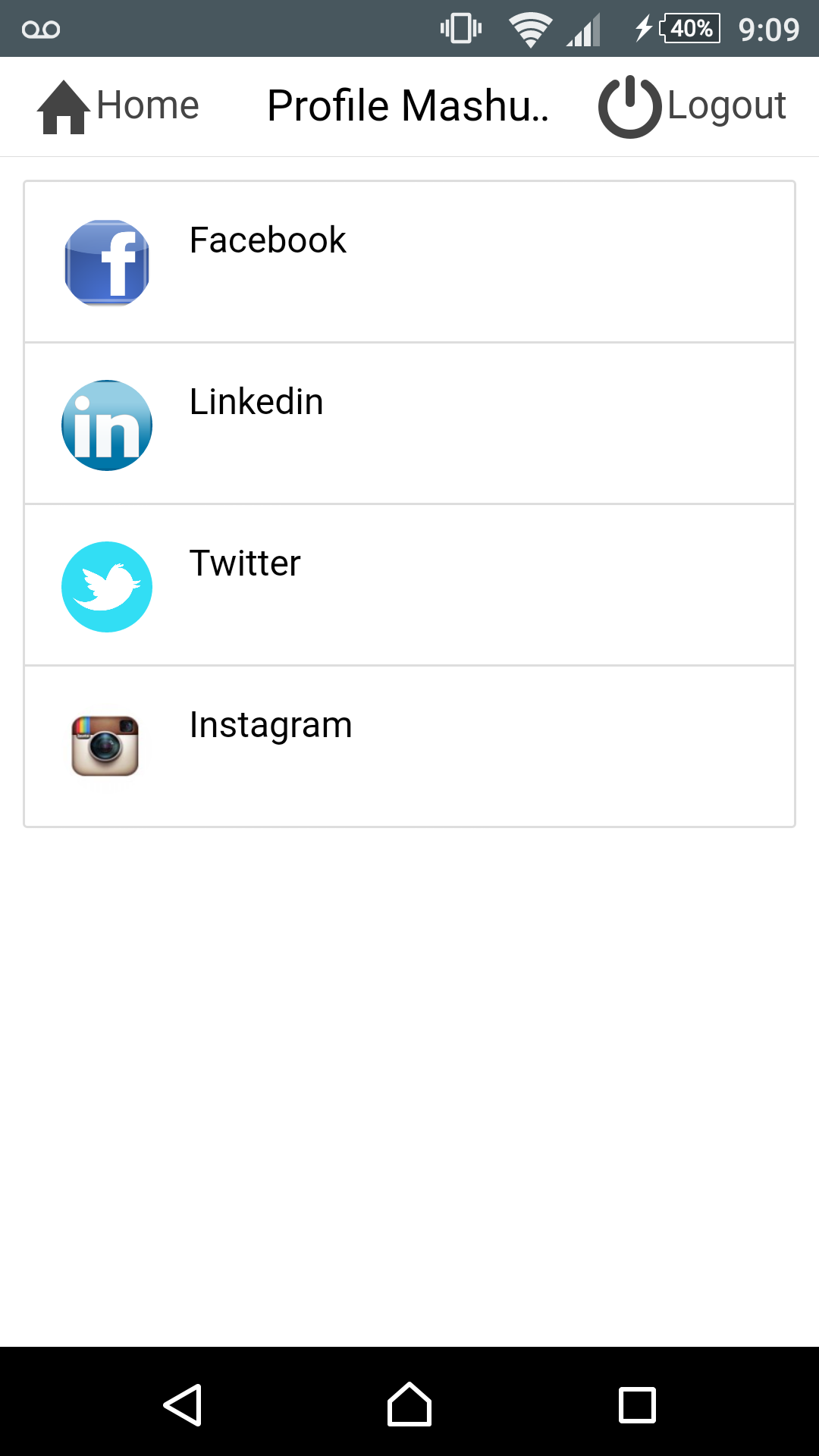
**2.Register Activity:**



Description:

This is the Register Activity of the Multi-Messenger app where user can register to the app where the user details are stored in Mongolab using MongoDB.

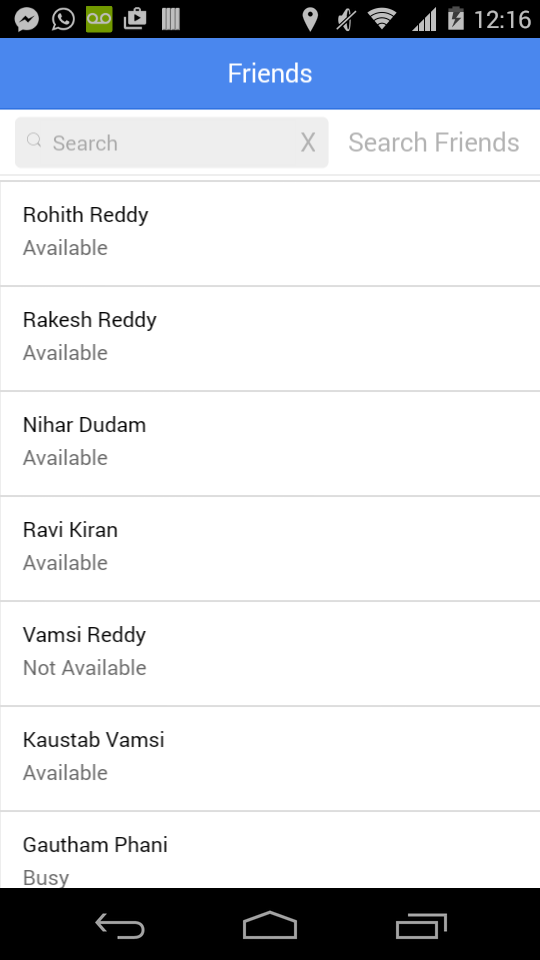
**4. User Contacts Page :**



Description:

The above page displays user contacts page where user can select various messengers to communicate with and perform various actions , send, receive messages etc.

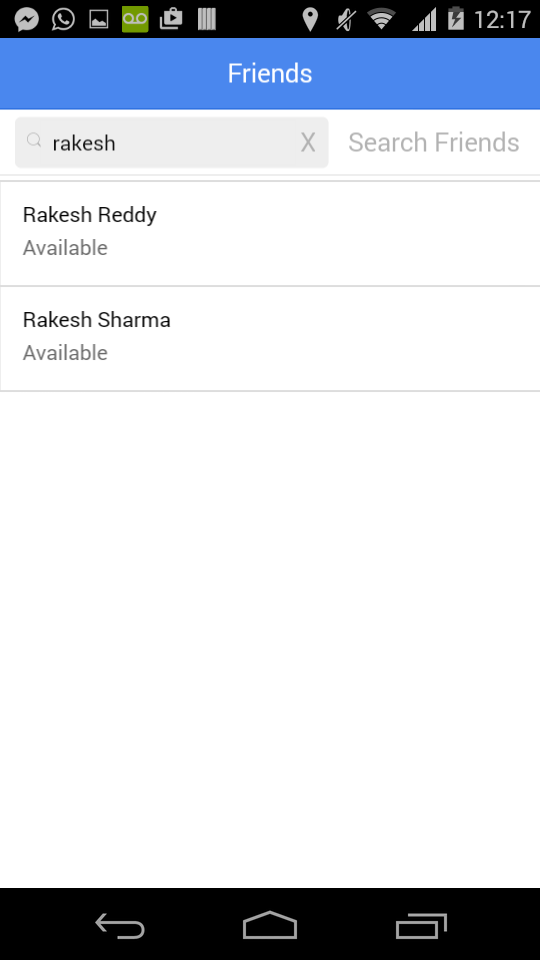
**5.User Gtalk contacts:**

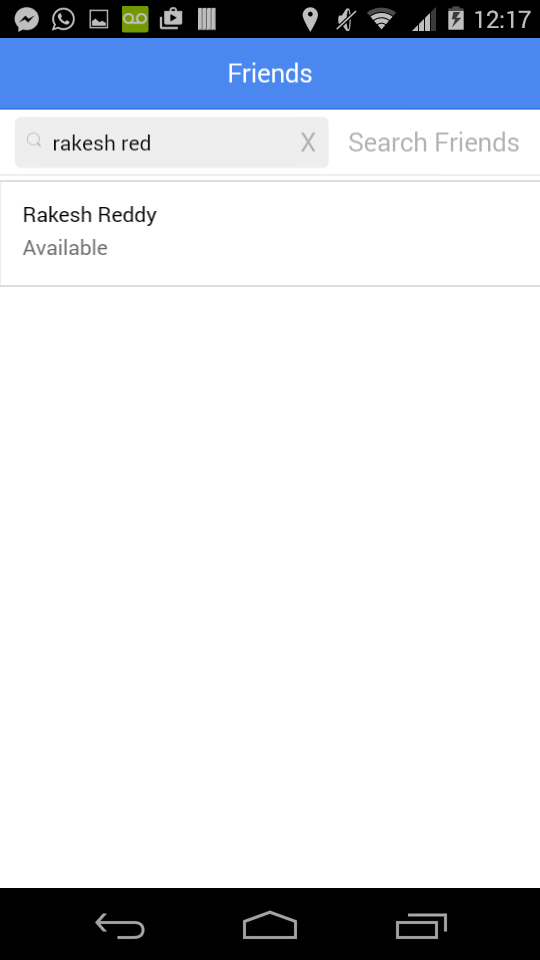


Description:

The above window displays us the gtalk contacts.

**6. Friends Search :**

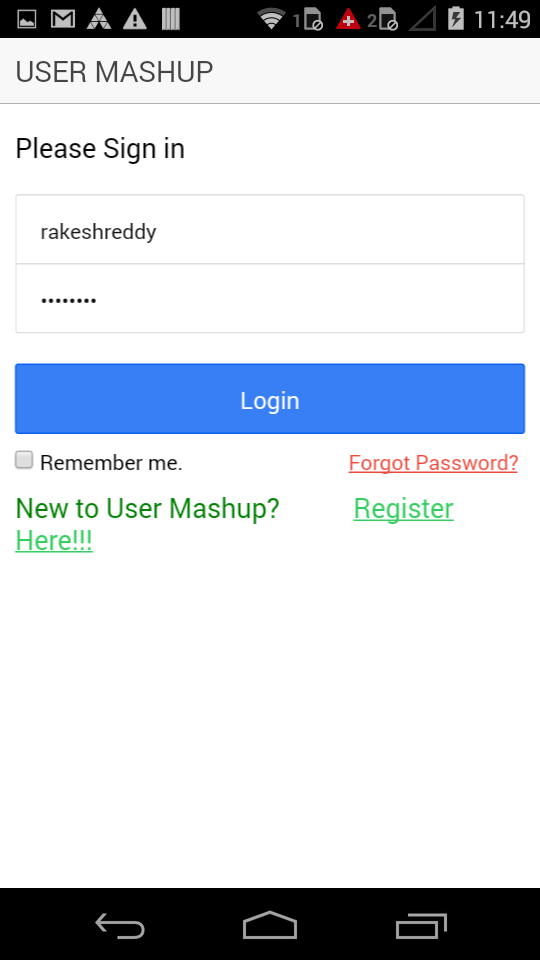




Description :

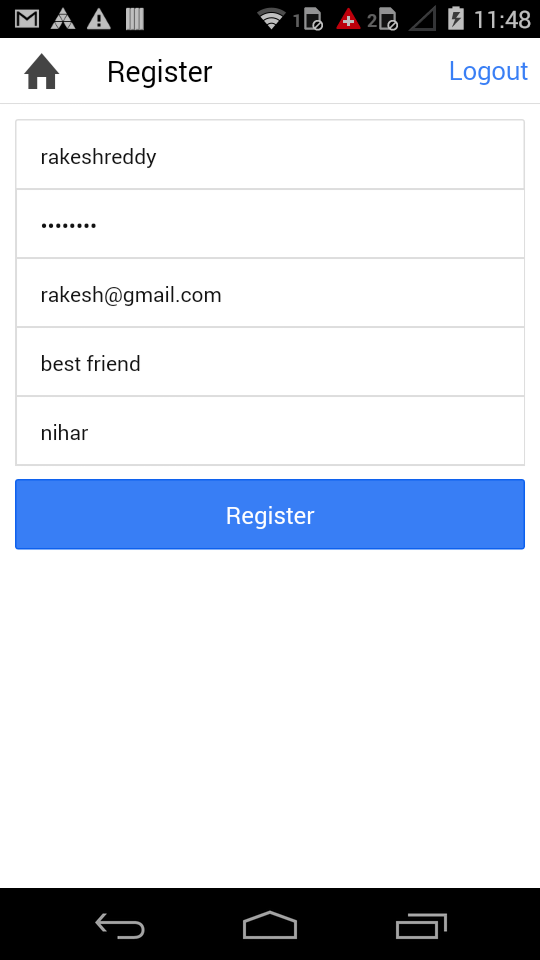
The above screen illustrates us the search functionality of the project where the user can search the contacts using a keyword.

**7.Login Page:**

****

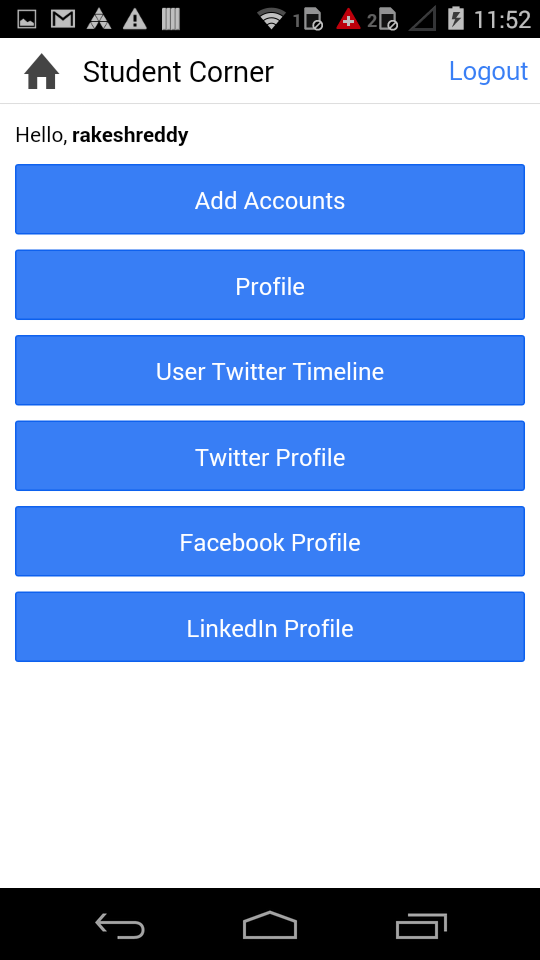
The above figure depicts login page

**8.Register Page**

****

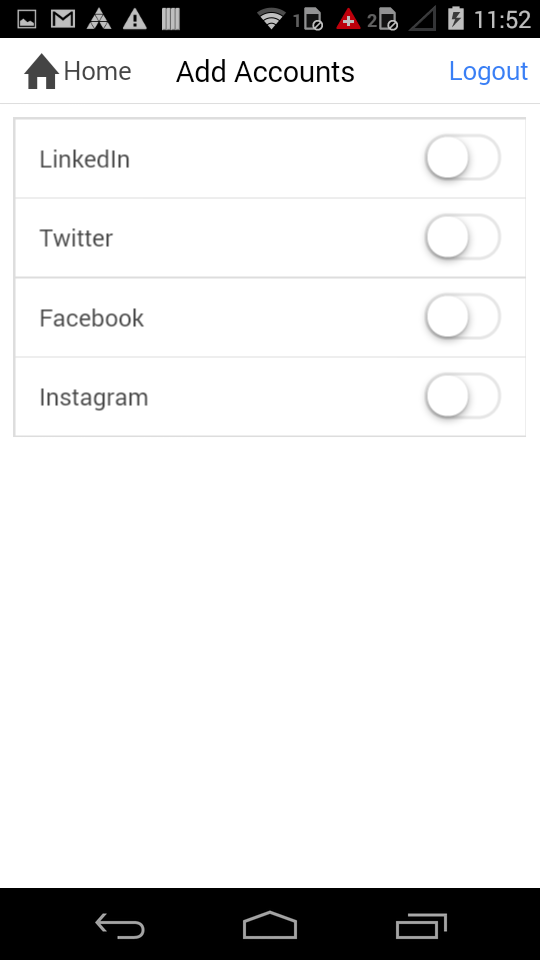
The above figure explains about register Page

9.Home Page

****

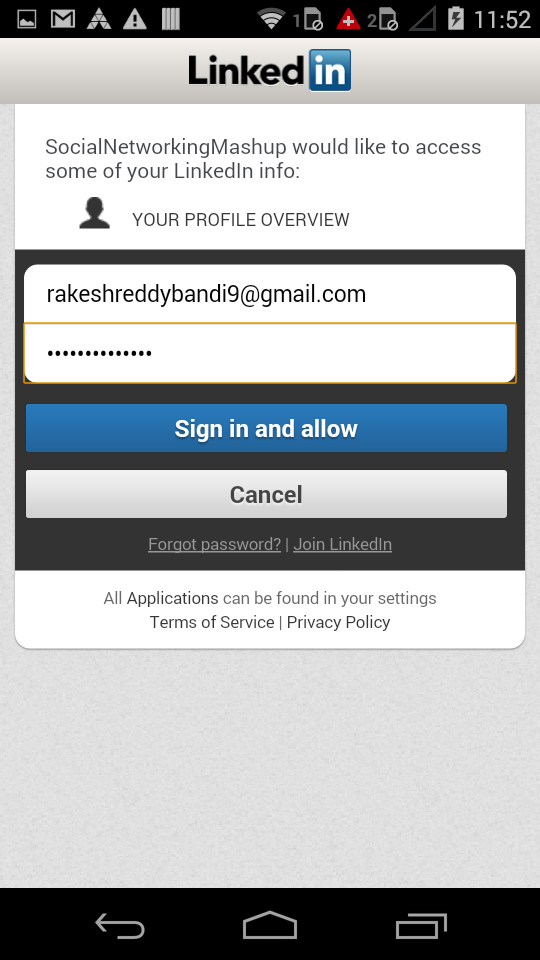
The above figure explains about Home Page of our application

**10.Add Accounts Page**

****

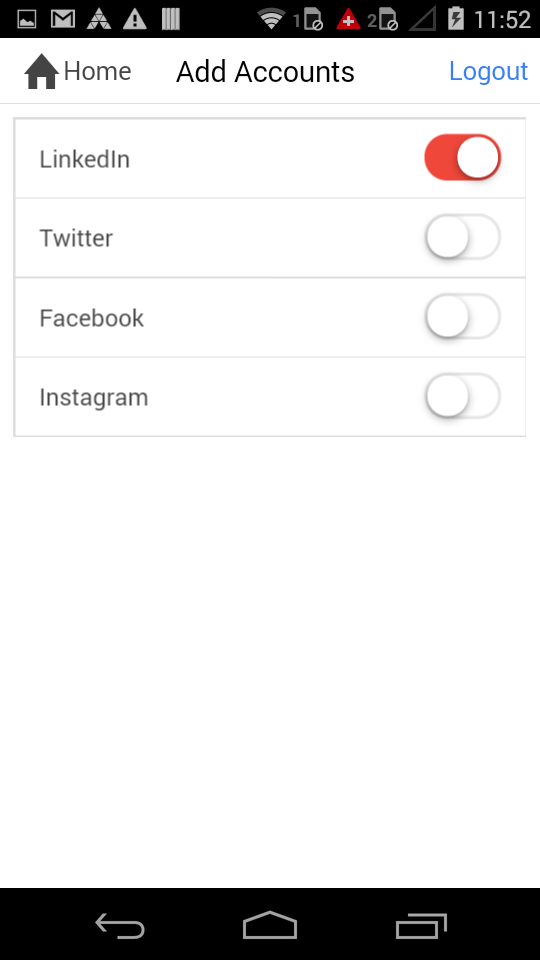
The above diagram explains about adding accounts in our application

**11.Linked in Sign in page**

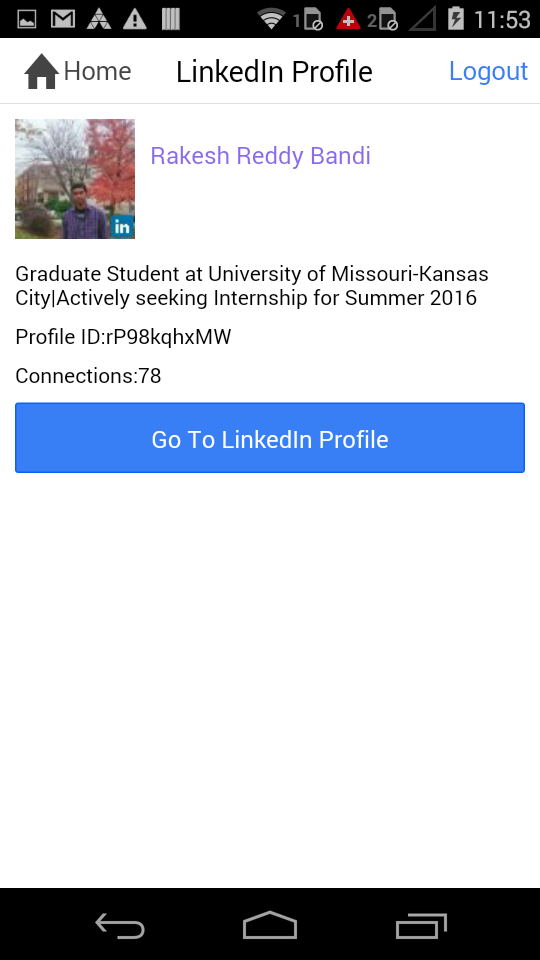
****

The above diagram depicts adding linked in Sign in page

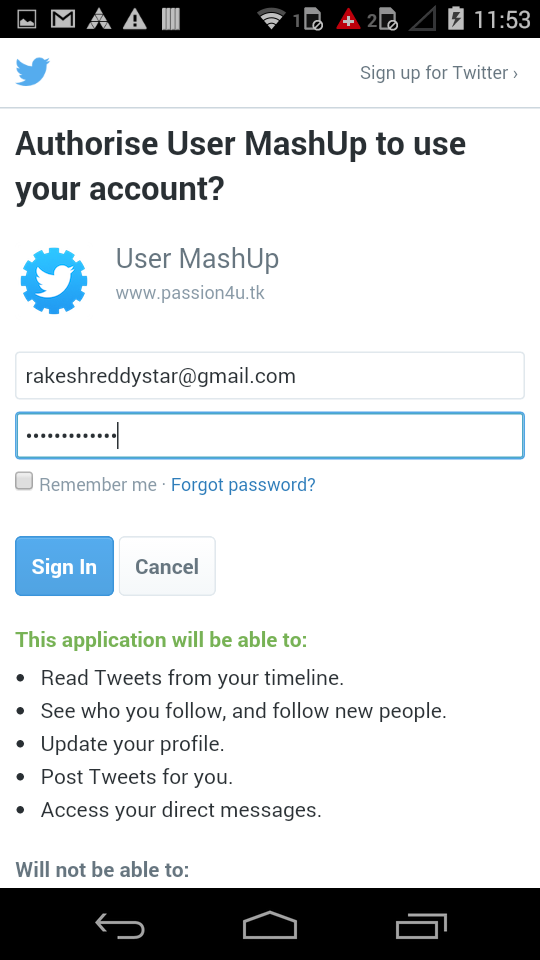
**12.Enabling Linkedin account**

****

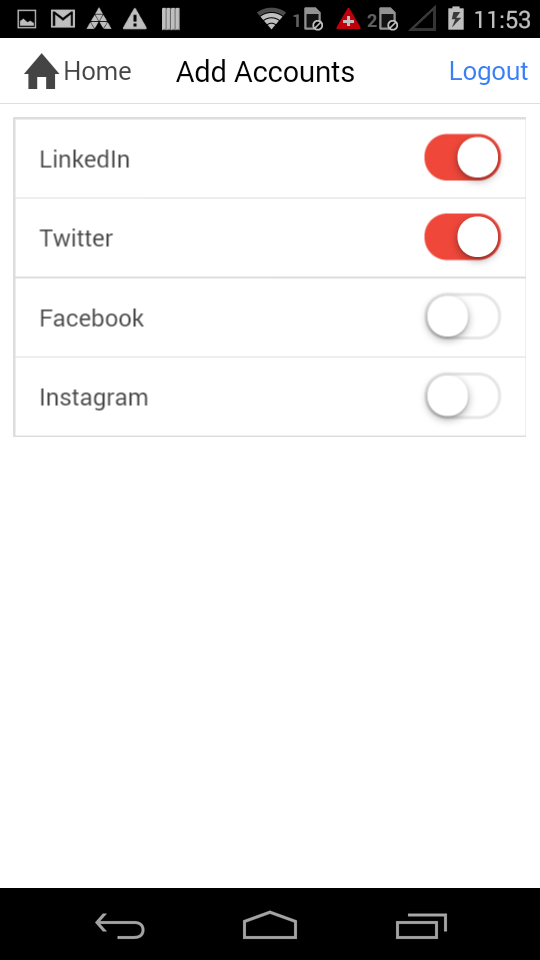
**13.Linkedin profile and feed**

****

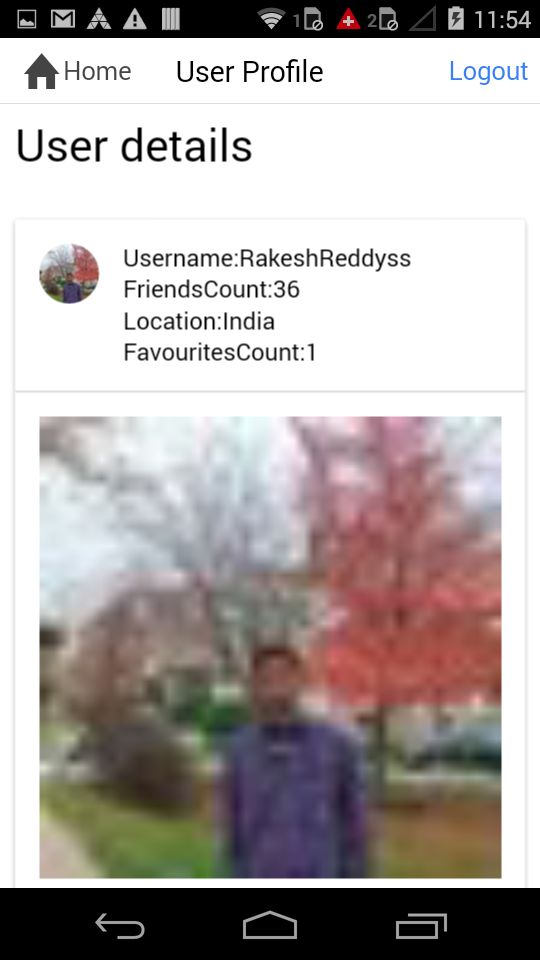
**14. Authorization of twitter App**

****

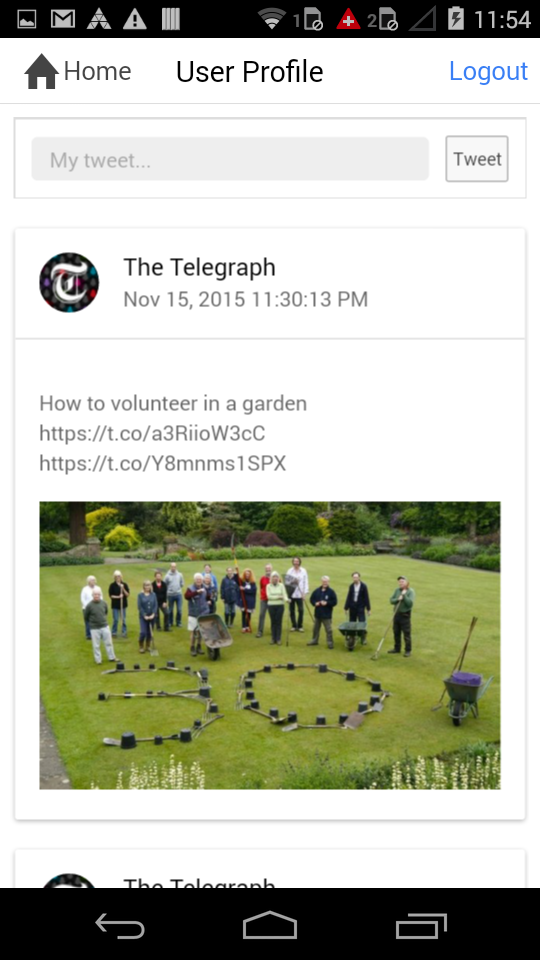
**15.Enabling Twitter account**

****

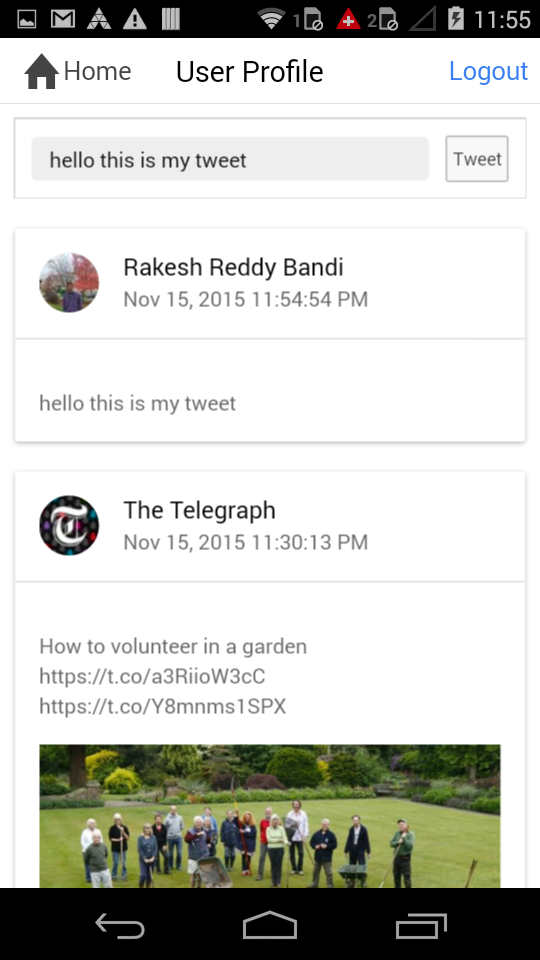
**16.Twitter feed**

****

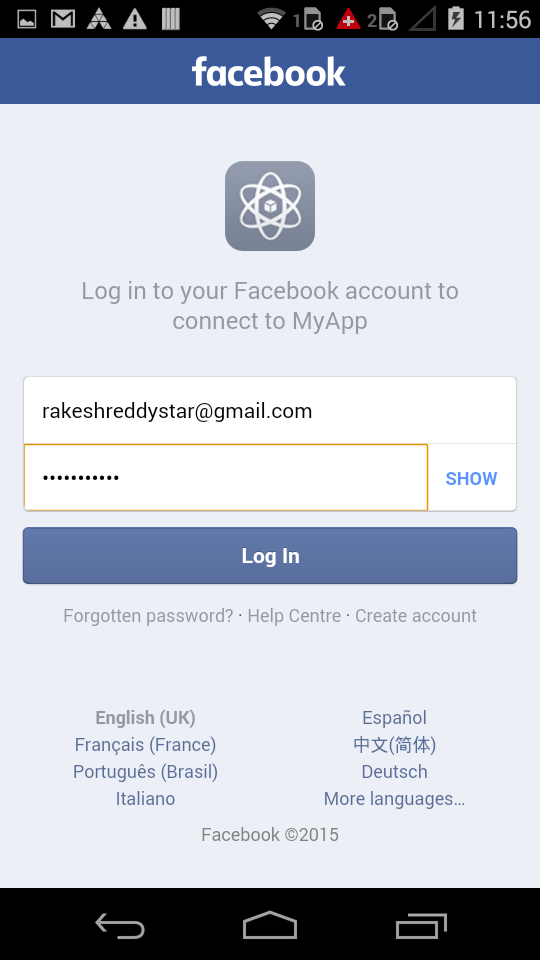
**17.Twitter Feed**

****

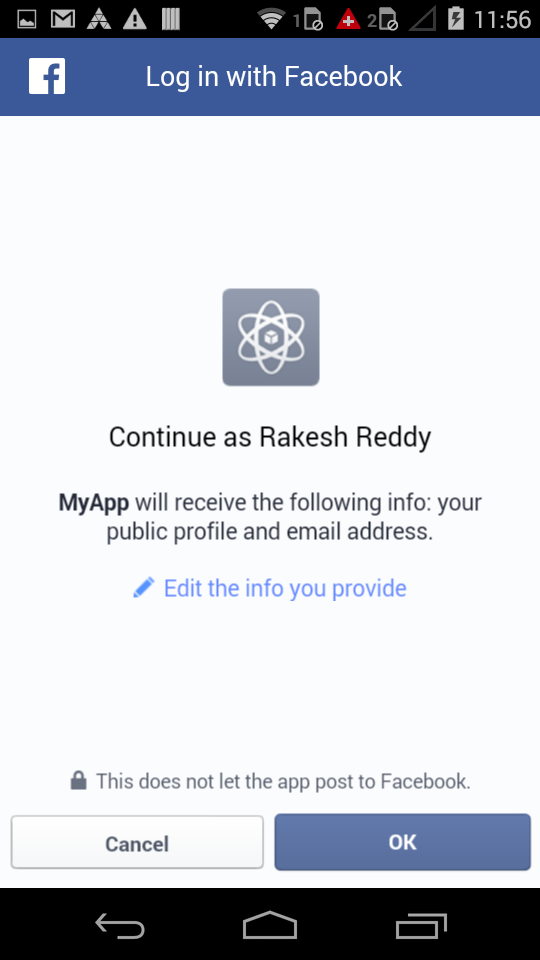
**18.Posting to Twitter wall**

****

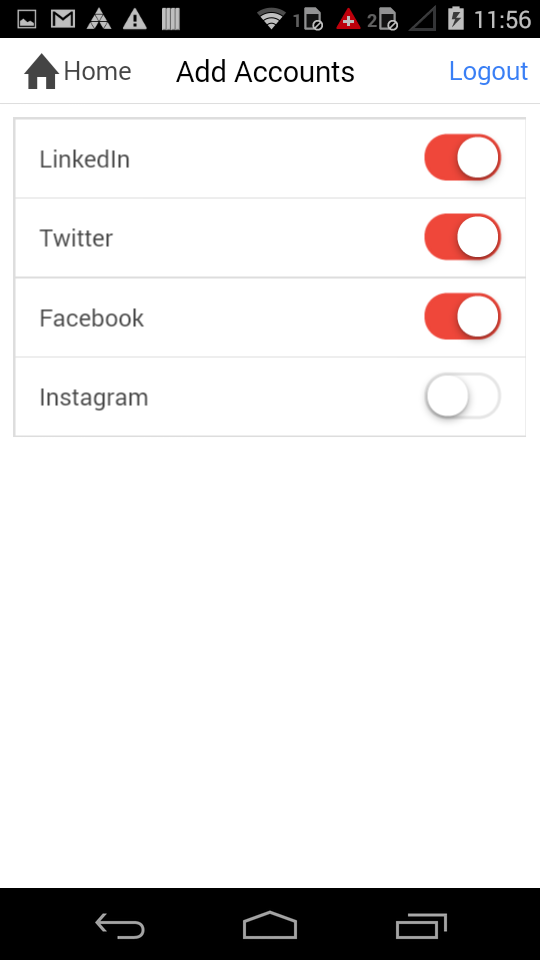
**19. Authorizing facebook**

****

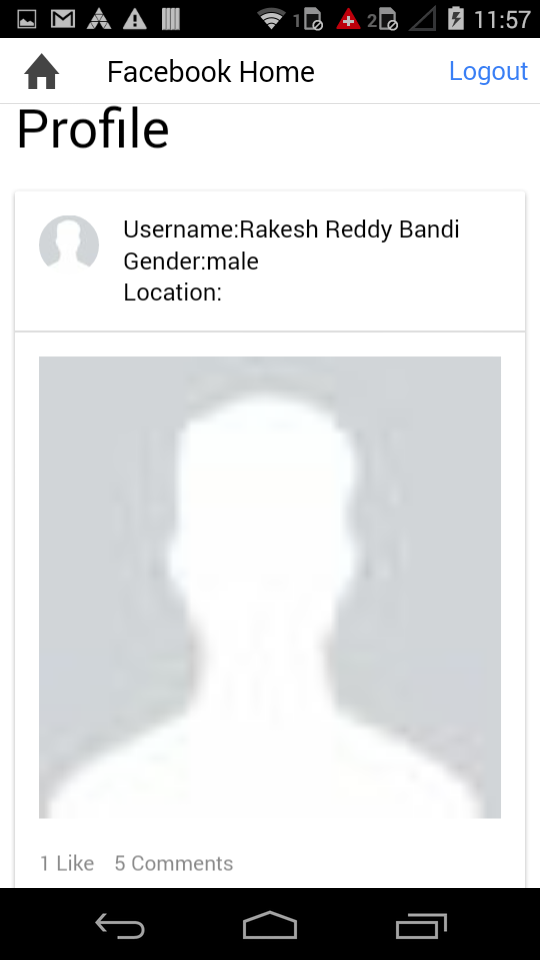
**20. logging into facebook**

****

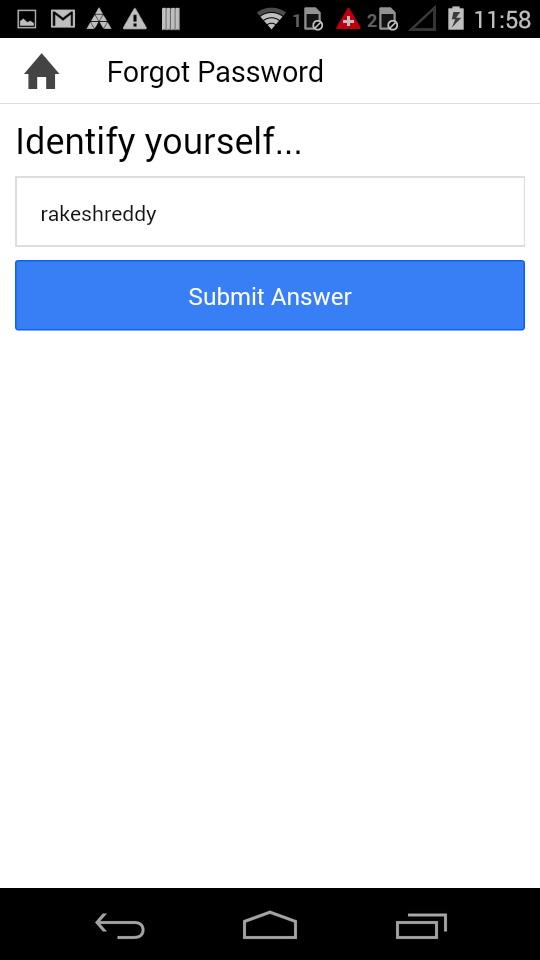
**21. Enabling Facebook account**

****

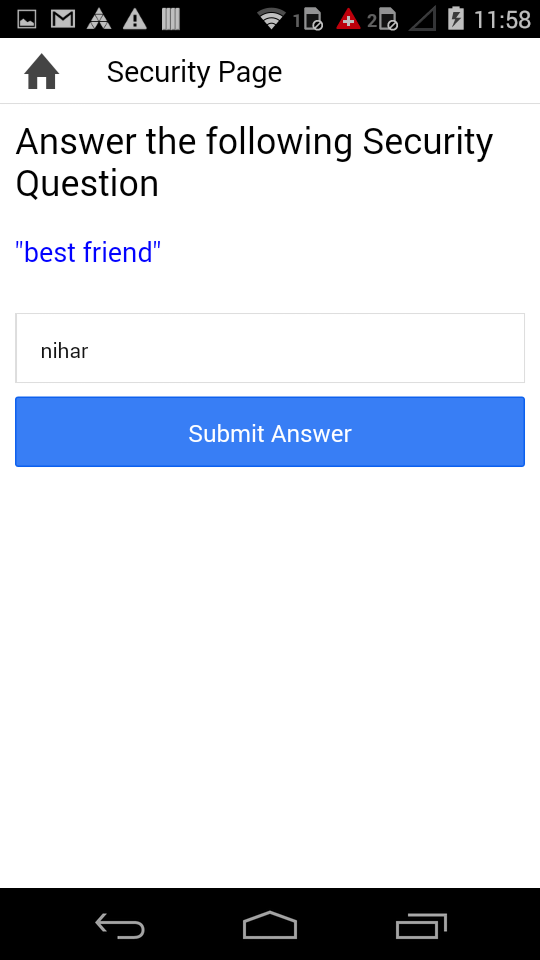
**22.Facebook profile feed**

****

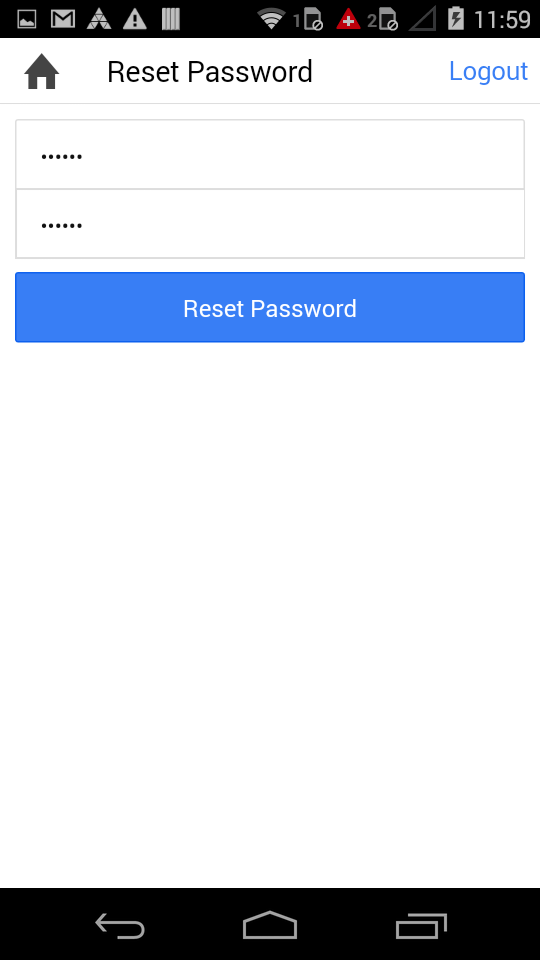
**23. Forgot password**

****

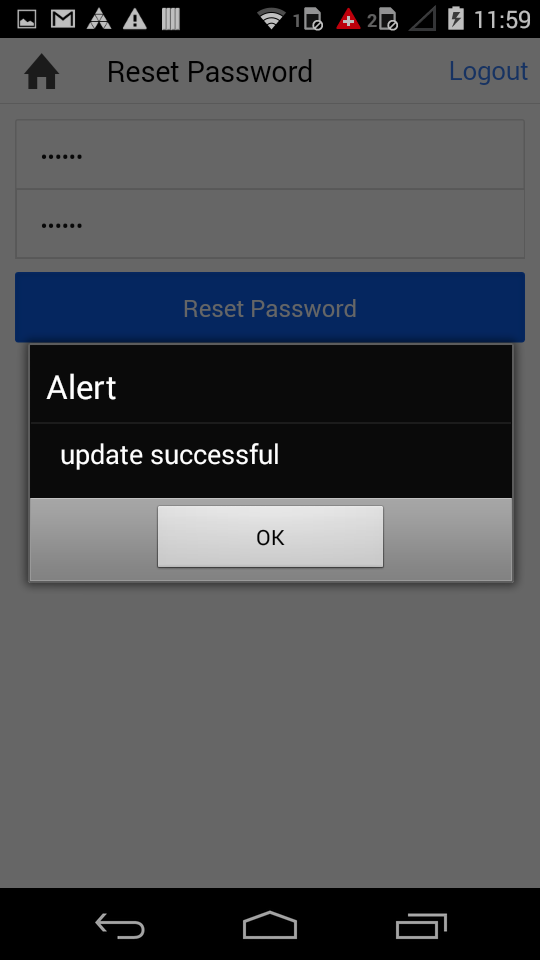
**24. Entering Security Question**

****

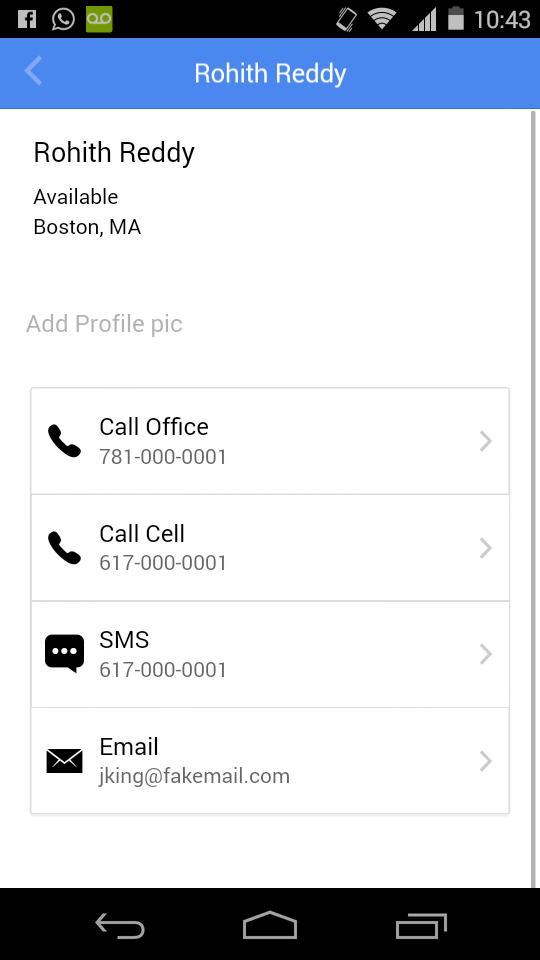
**25. Finally resetting a password**

****

**26. Success message of resetting password**

****

**27.Specific Friend window :**



Description :

The above figure illustrates the specific friend window where he can send mail or message to him

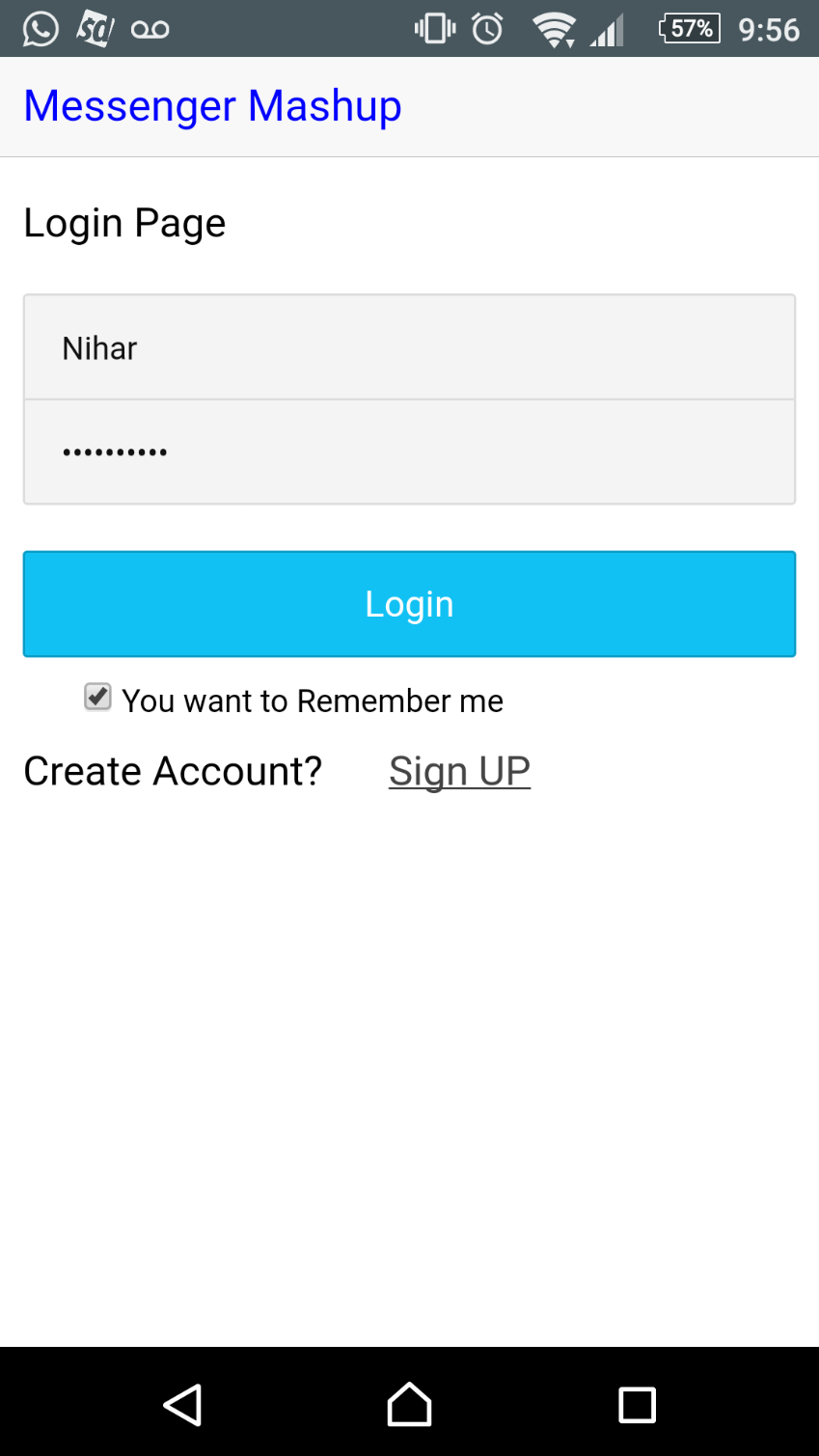
**28.Chat Screen :**



Description :

The above screen illustrates the chat screen where user can send and receive the messages from a specific friend.

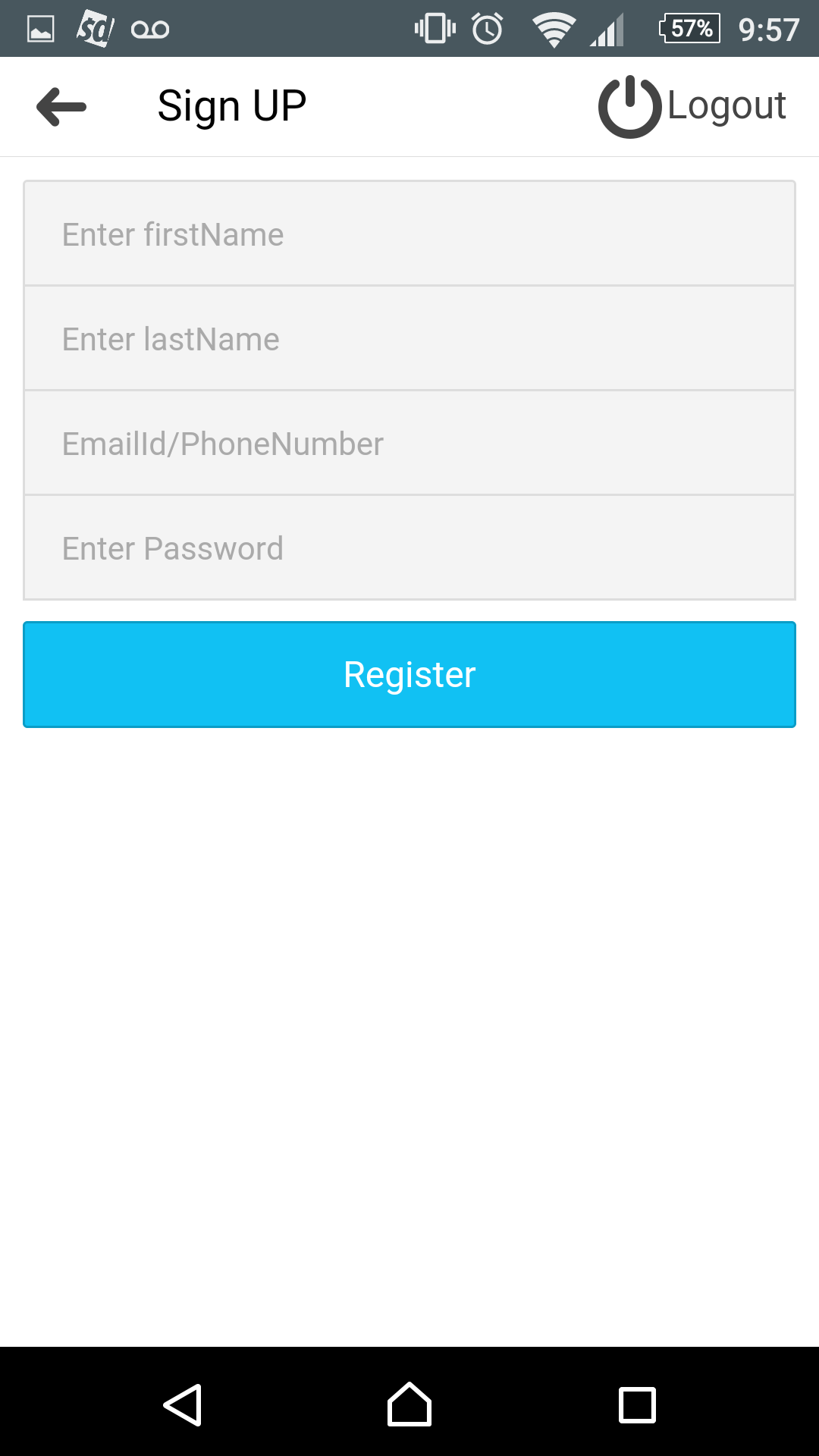
29. Login Activity:

****

Description:

This is the Login Activity of the Multi-Messenger app where user can login to the app using the correct credentials.

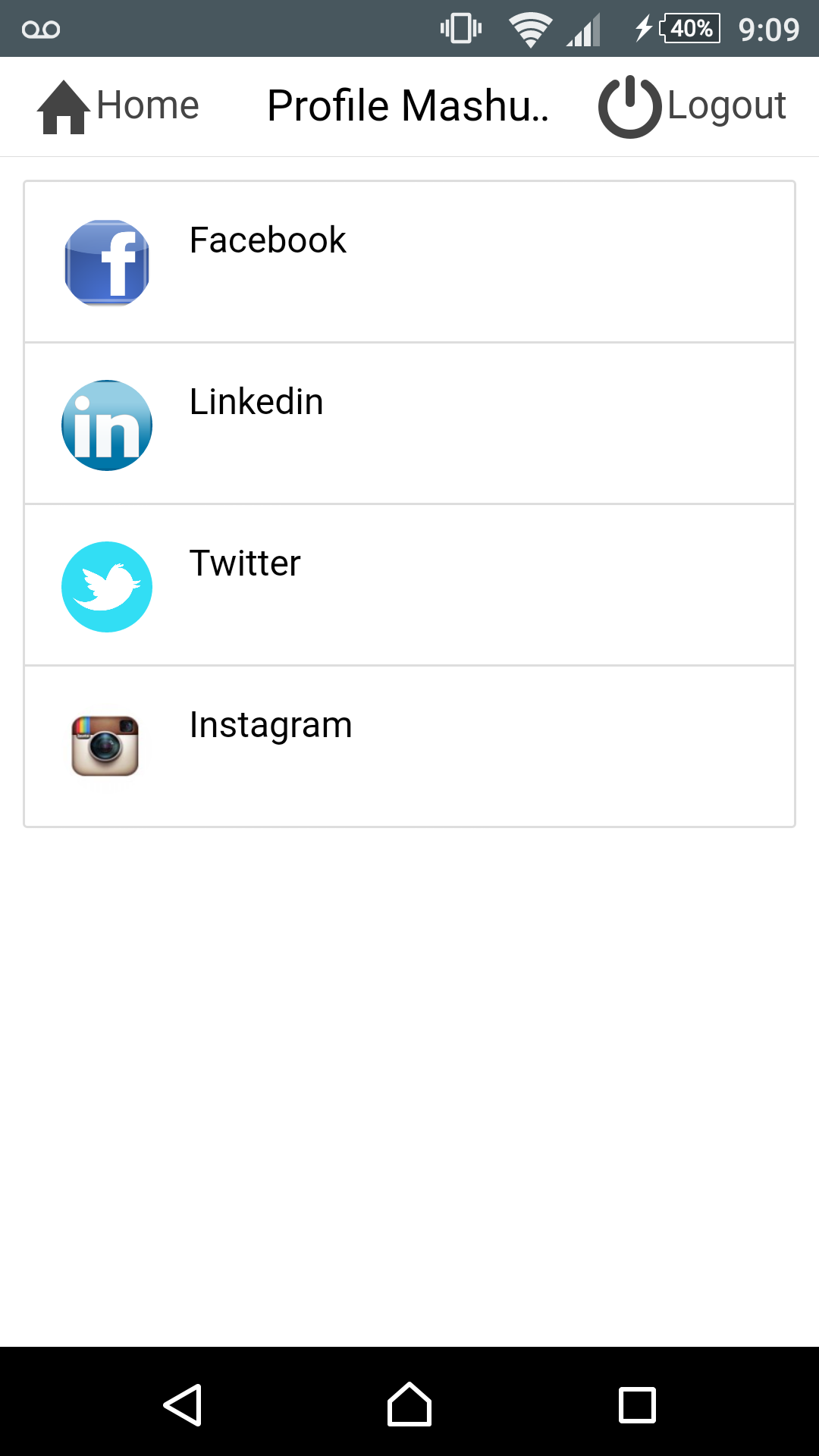
30.Register Activity:



Description:

This is the Register Activity of the Multi-Messenger app where user can register to the app where the user details are stored in Mongolab using MongoDB.

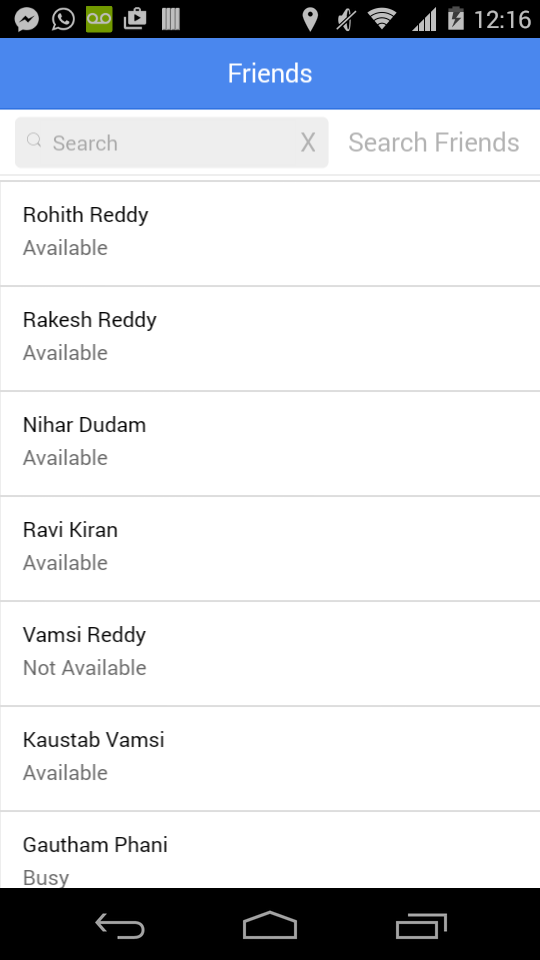
31.User Contacts Page :



Description:

The above page displays user contacts page where user can select various messengers to communicate with and perform various actions , send, receive messages etc.

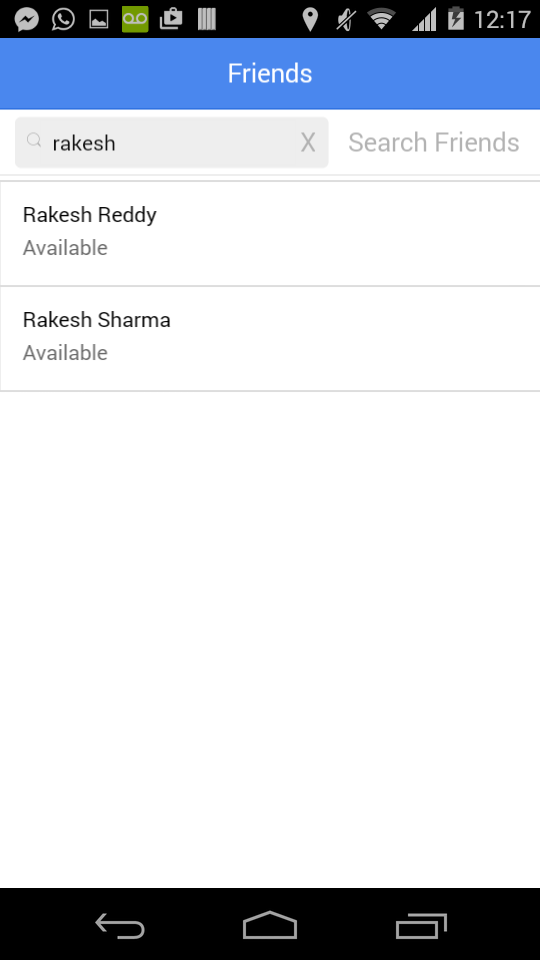
32.User Gtalk contacts:

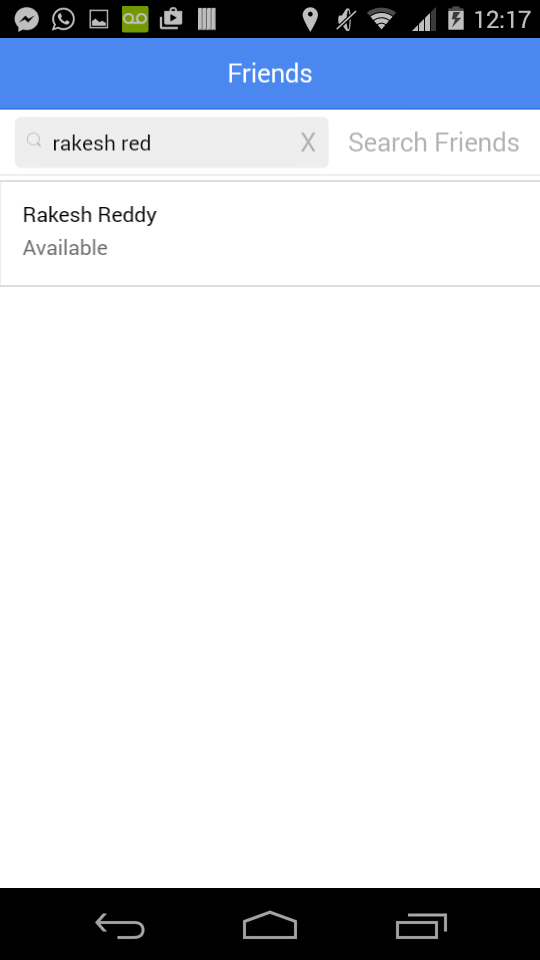


Description:

The above window displays us the gtalk contacts.

32. Friends Search :

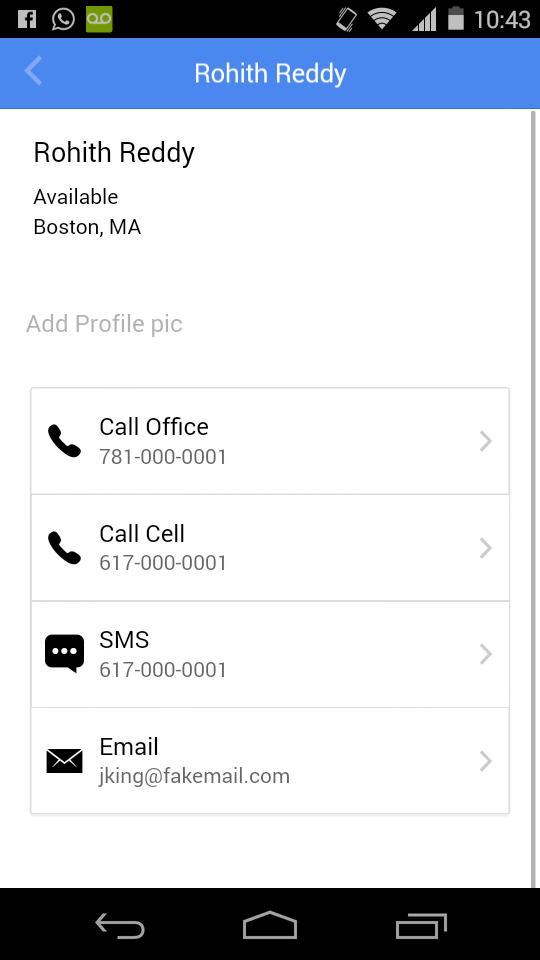




Description :

The above screen illustrates us the search functionality of the project where the user can search the contacts using a keyword.

33.Specific Friend window :



Description :

The above figure illustrates the specific friend window where he can send mail or message to him

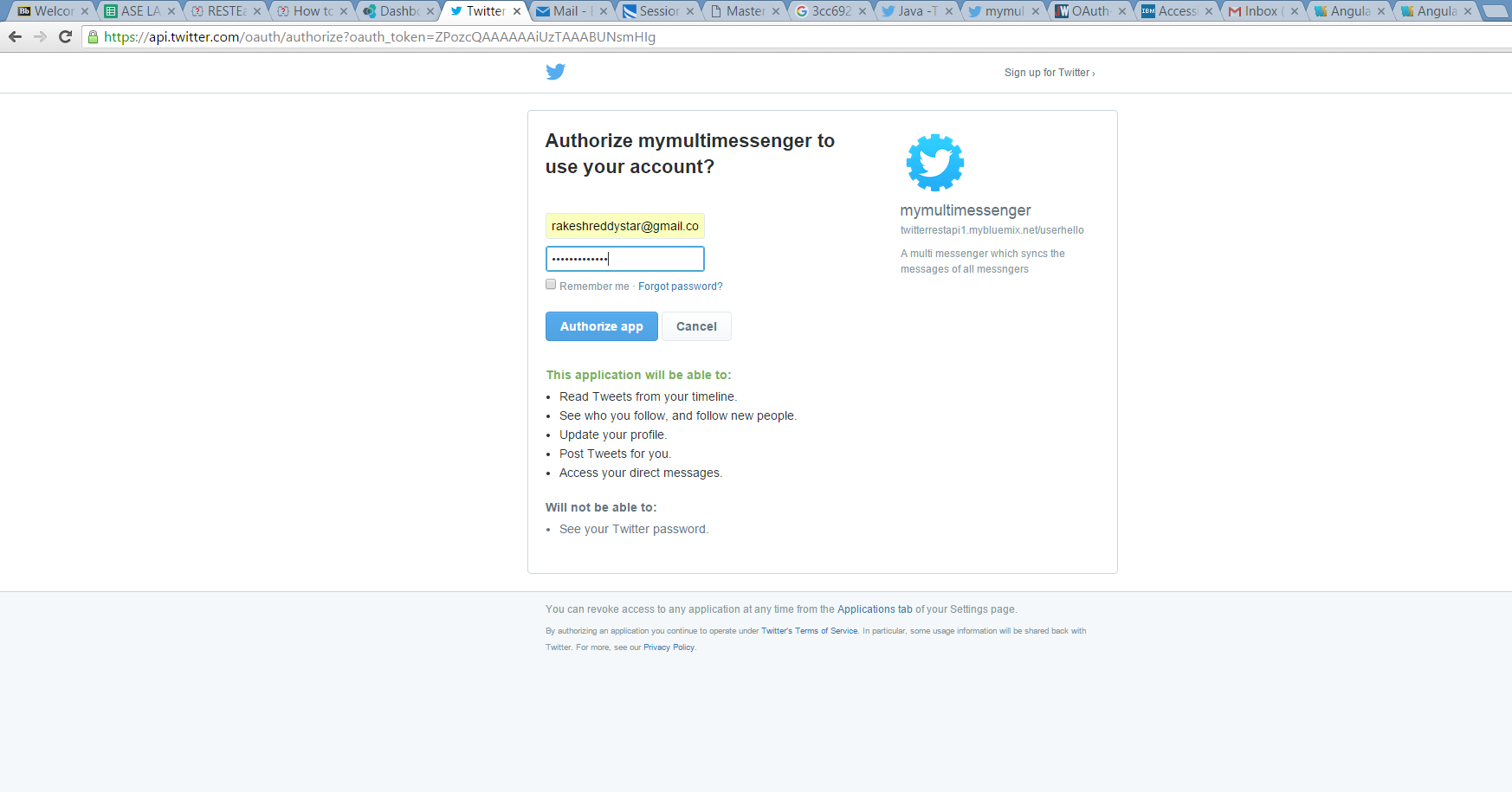
34.Chat Screen :



Description :

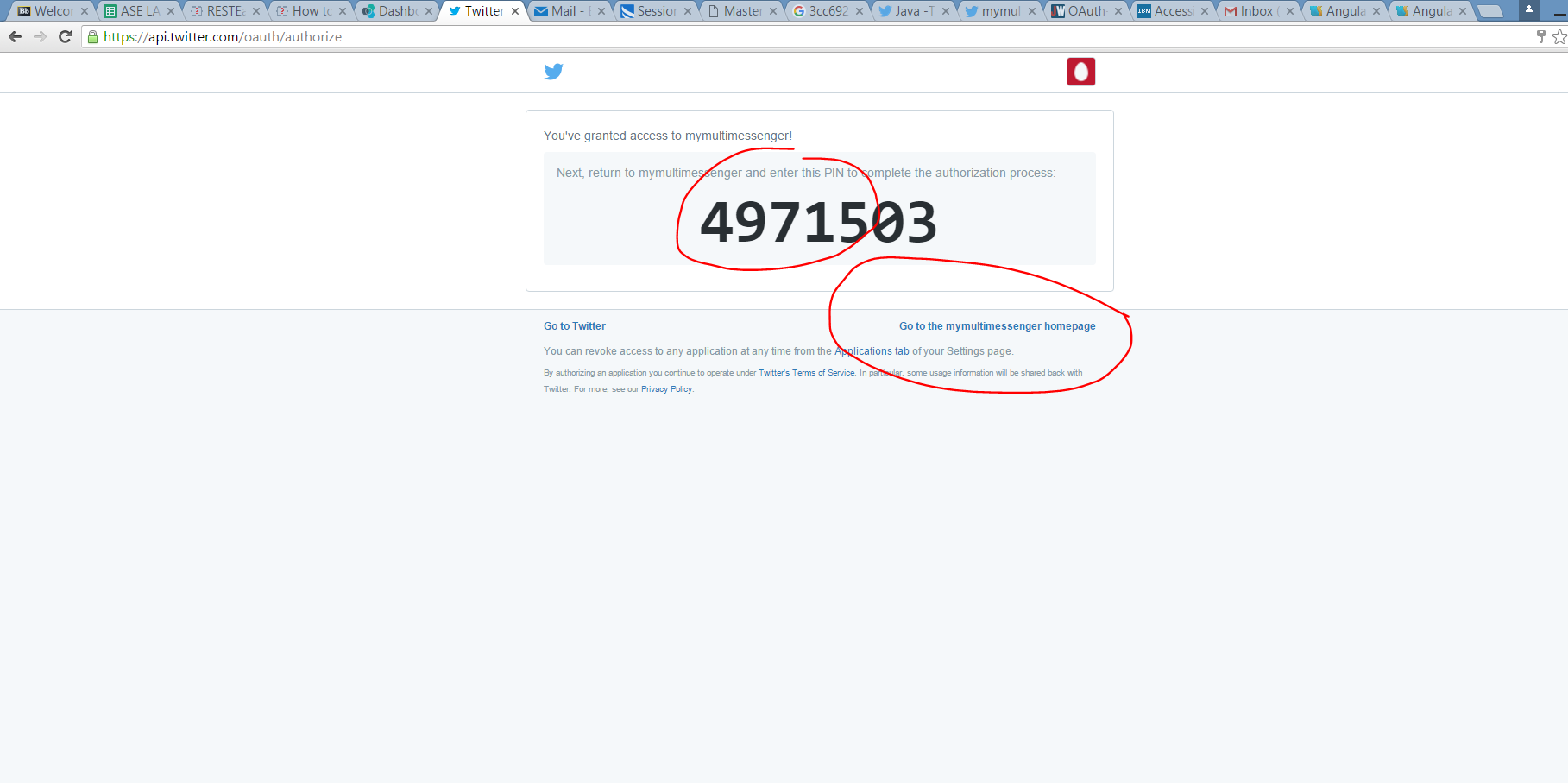
The above screen illustrates the chat screen where user can send and receive the messages from a specific friend.

9. Twitter OAUTH screen shots:



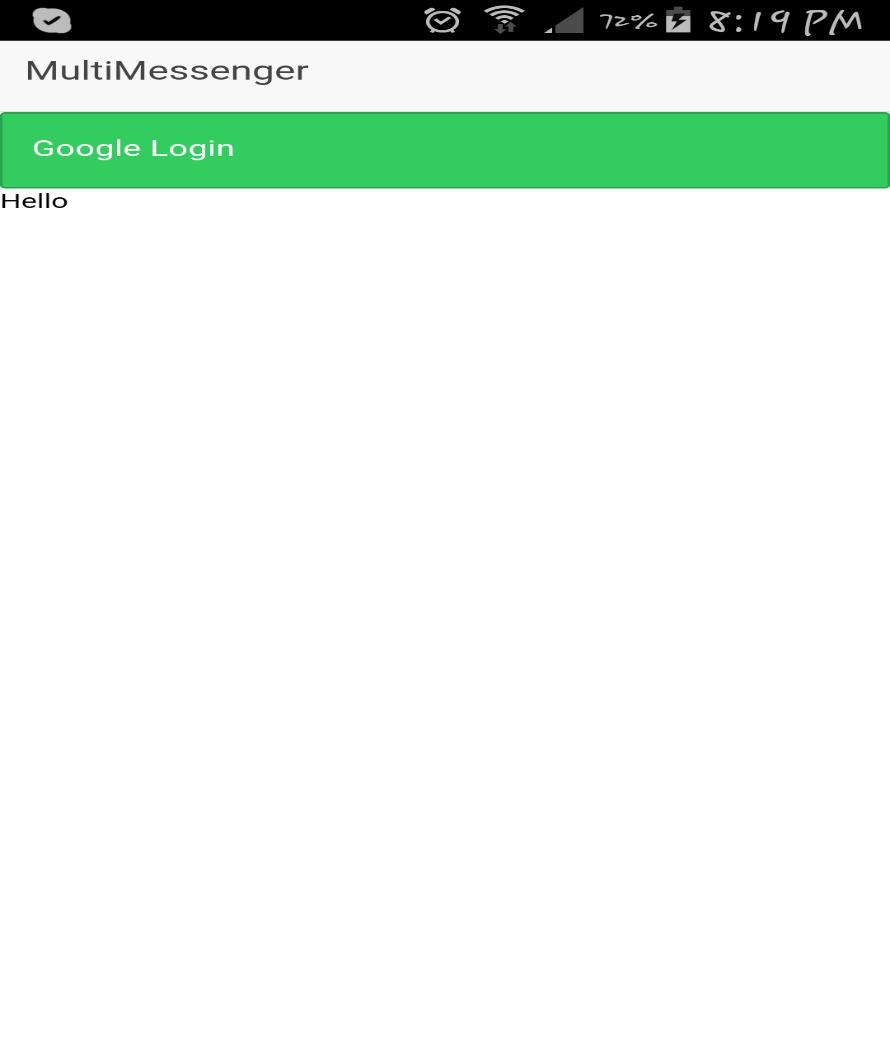
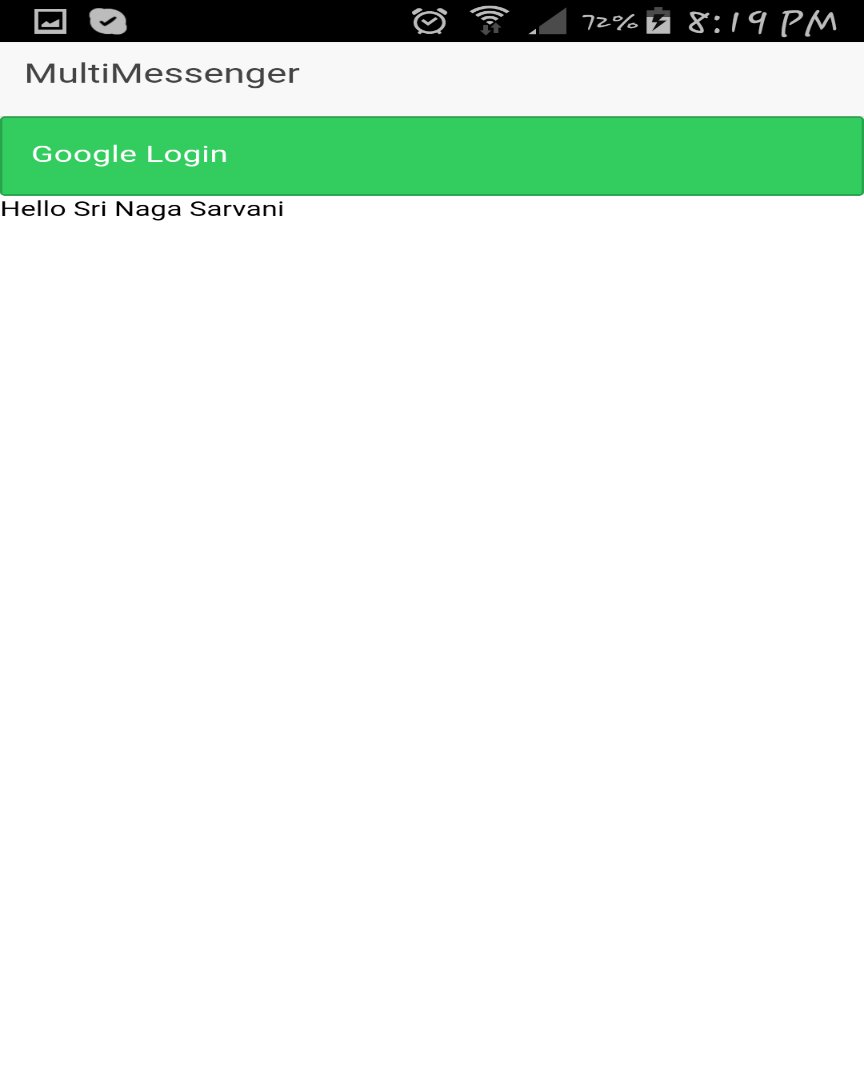
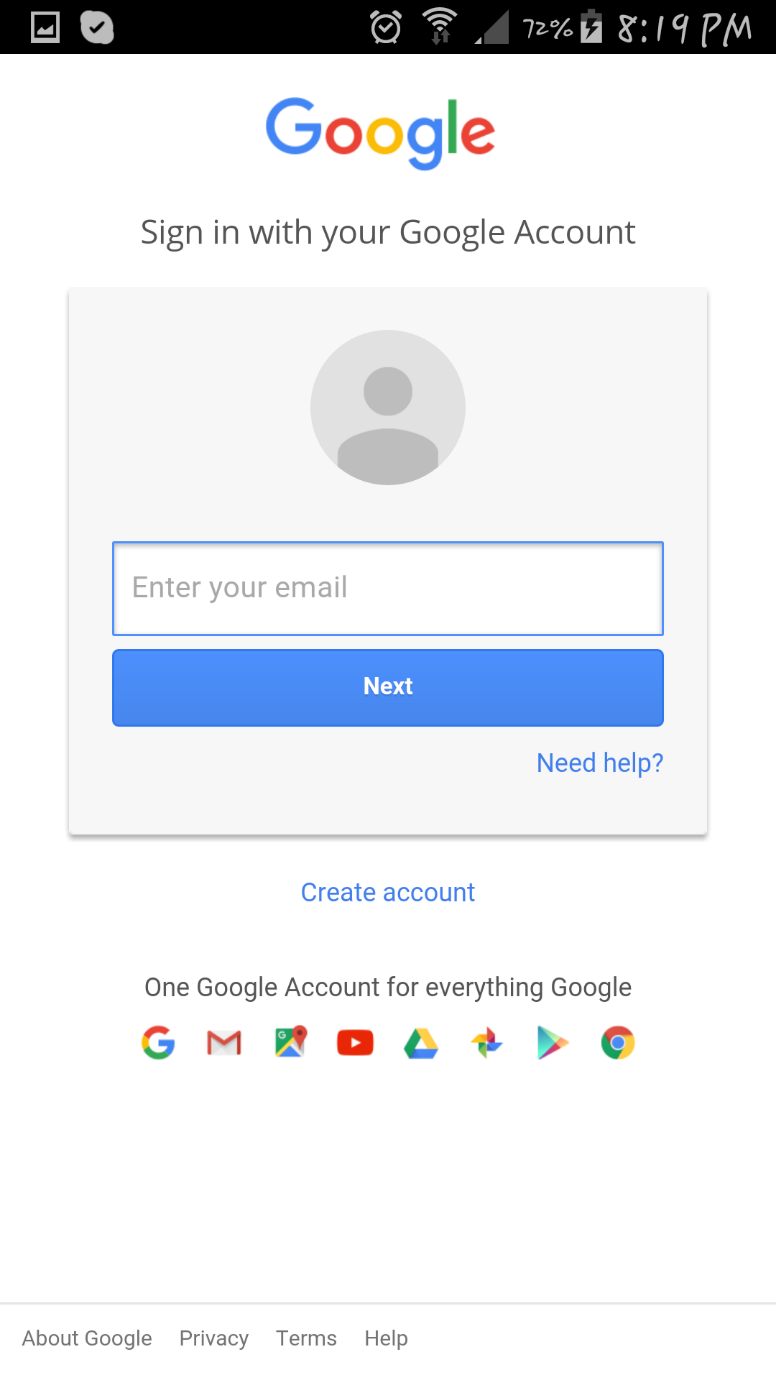
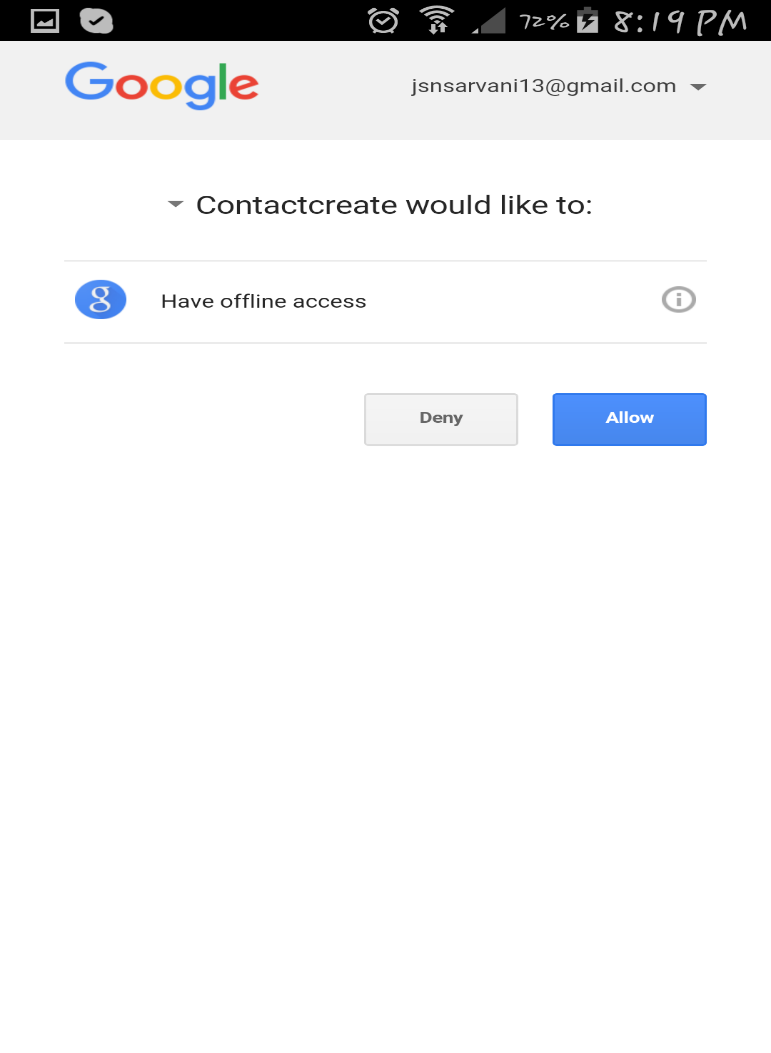
Description:

This the authentication page from twitter. When we add account from our application we redirects to this page to authorize mymultimessenger application to use the User’s account. We authorize the app for our application.



Description:

This is the code we need to enter to authorize the application to access the users data in twitter.

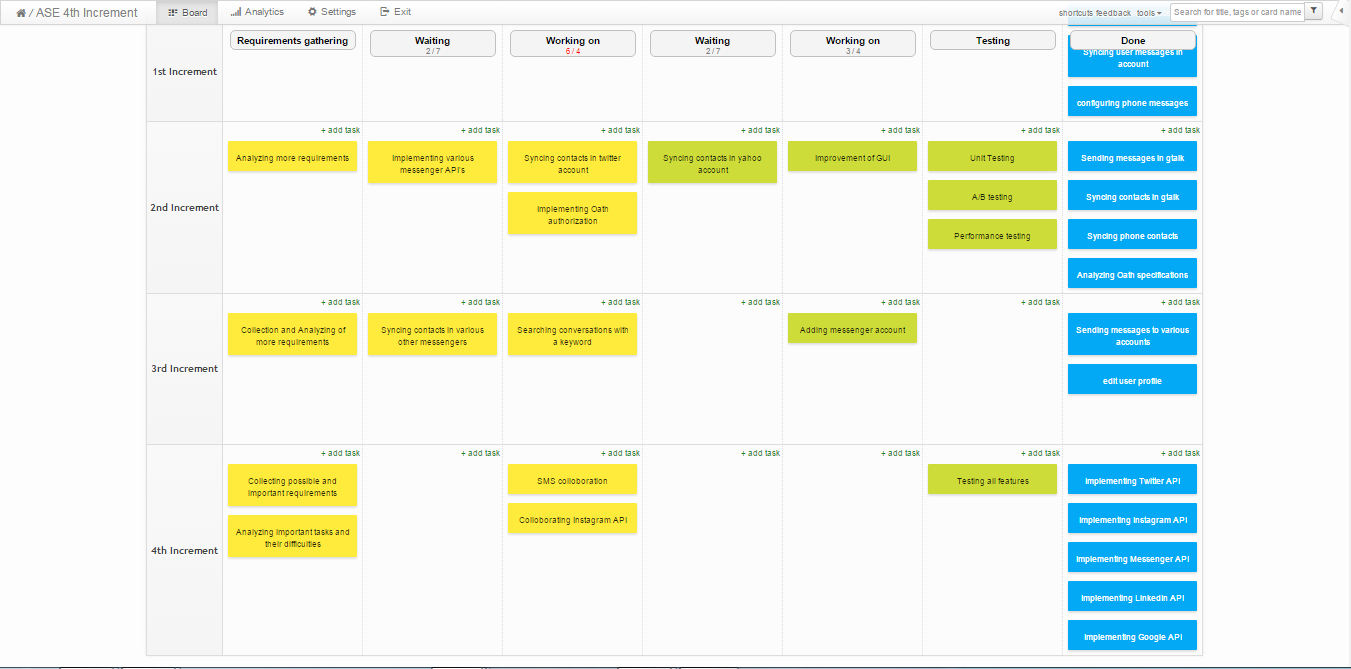


## **Project Management**

Project Management is done through popular Kanban tool where entire project can be completed efficiently by dividing the tasks among all group members and analyzing the completed work and work yet to be completed effectively.

Project Management URL: https://nihard.kanbantool.com/b/191348

* Kanban tool iterations



The above figure explains about various iterations of the project.

All the tasks are divided in a good proportion among the group members by assigning specific time and deadline to a task.

In the fourth increment, we mainly concentrated on the important and implementable tasks. We have worked on various other API’s where they are providing us feasibility to work on their API’s. So we have chosen Instagram API, Google API, Facebook API, LinkedIn API and Twitter API. We have integrated all the API’s in a single window and user can interact with various messengers, can see their profiles and respond to the notifications, they can even post their respective views.

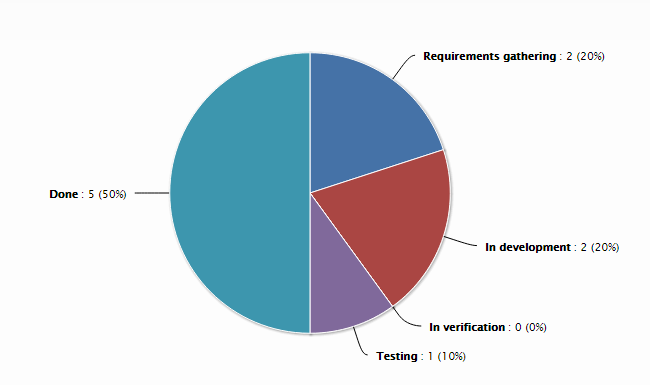
* Implementation status report:

1. Work completed

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Member | Task Description | Member Responsibility | Contribution % | Time (hours) | Comments  /Issues |
| Rakesh | 1. LinkedIn API  2. Twitter API  3. Integrating the project. | Develop, design, build and testing the task | 100 | 18 |  |
| Nihar | 1. Instagram API  2. Integrating the project | Develop, design, build and testing the task | 100 | 20 |  |
| Sarvani | 1. Google API  2. Integrating the project | Develop, design, build and testing the task | 100 | 22 |  |
| Ravi KIran | 1. Facebook API  2. Integrating the project | Develop, design, build and testing the task | 100 | 20 |  |

\

* **Analysis graphs**:
  + - Increment Analysis graph:



The above diagram explains about the various tasks position in various divisions available in Kanban tool

## **Bibliography**

[https://developer.yahoo.com/messenger/](https://developer.yahoo.com/messenger/.%20)

<https://dev.twitter.com/rest/reference/get/direct_messages/sent>

<https://dev.twitter.com/rest/reference/get/direct_messages/show>

<https://dev.twitter.com/rest/reference/post/direct_messages/new>

<https://en.wikipedia.org/wiki/XMPP>

<https://developers.google.com/talk/jep_extensions/oauth>

<http://developer.samsung.com/technical-doc/view.do?v=T000000119>

<http://ngcordova.com/docs/plugins/>

<https://en.wikipedia.org/wiki/Unit_testing>

<https://www.instagram.com/developer/authentication/>

<https://developer.linkedin.com/docs/rest-api>