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 THIRD 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.

**Yahoo Messenger API** We can get the Yahoo API from [https://developer.yahoo.com/messenger/](https://developer.yahoo.com/messenger/.%20). We can get different services based on type of device on which we are implementing. The available and popular types are iOS, Android, Web etc. For the first increment we want to implement the native SMS service within the app.

## **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:

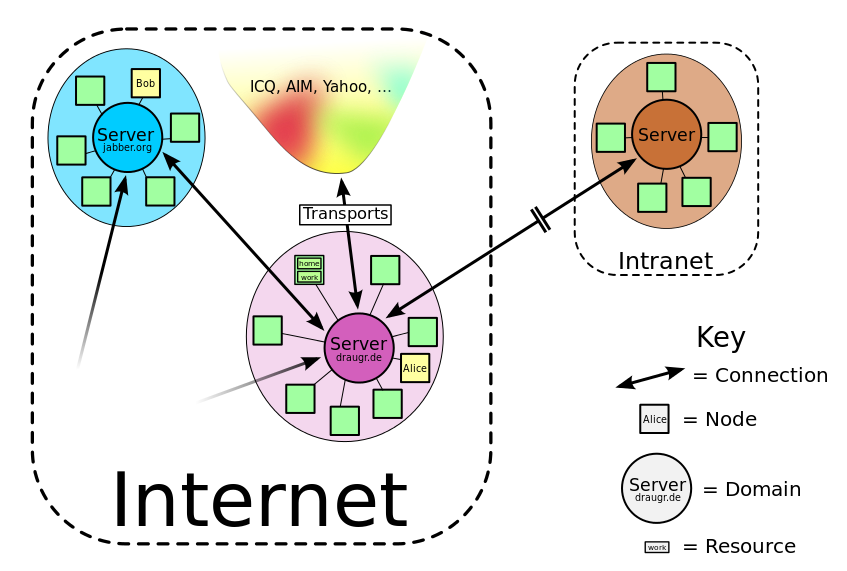


Figure 1: XMPP Architecture Diagram

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.

* 1. **Yahoo API**

The below are the steps in Integrating Yahoo Instant Messenger client.

* Create a Yahoo Messenger Open Authentication (OAuth) API Key.
* Authenticate with the Yahoo Messenger IM SDK servers.
* Create a new session.
* Obtain and update presence information.
* Obtain the current contact list, as well as more detailed information on a particular contact.
* Send and receive a message from another contact.

1. There are two API calls that are necessary to authenticate with the Yahoo Direct OAuth API. The first obtains a Pre-Authorized Request Token (PART), which requires user’s login username, password, and the OAuth Consumer Key (API Key) that was just generated through developer.yahoo.com.

2. You would receive a Request token as an output when Log-in API is called. The below is login API: [*https://login.yahoo.com/WSLogin/V1/get\_auth\_token?&login=username&passwd=mypassword&oauth\_consumer\_key=consumerkey*](https://login.yahoo.com/WSLogin/V1/get_auth_token?&login=username&passwd=mypassword&oauth_consumer_key=consumerkey)*.*

3. To receive pre-approved request token the API is: *https://login.yahoo.com/WSLogin/V1/get\_auth\_token*

4. To fetch Contacts from Yahoo Messenger the API is: *GET /v1/contacts?sid=msgrsessionid ; Host: rcore1.messenger.yahooapis.com*

5. To receive notifications from Yahoo Messenger the API is: *GET/v1/notifications?sid=msgrsessionid&seq=5 ;Host: rproxy1.messenger.yahooapis.com*

6. To send messages through Yahoo Messenger is *POST/v1/message/yahoo/targetYahooId?sid=msgrsessionid Host: rcore1.messenger.yahooapis.com*

**5.5 Outlook API:**

For this increment we have implemented synchronizing Microsoft Outlook profile and contacts associated with the account. We also got the SkyDrive documents and pictures list.

We implemented this service using Live SDK provided by Microsoft. For connecting to outlook through our app it has oAuth protocol implemented.

Outlook Rest API is the one in which we are using to implement Outlook and One Drive. We use IMAP OAuth 2.0 for connecting to the outlook server.

The Live Connect APIs use industry standard protocols such as OAuth 2.0 and JavaScript Object Notation (JSON). To call the APIs, you primarily use Representational State Transfer (REST) requests that return info formatted in JSON. This architecture enables us to support a variety of platforms, including those for web, desktop, and mobile apps, like those for the iOS and Android operating systems.

Live Connect exposes info from these services:

* **OneDrive** for working with documents and media. Your apps can use the Live Connect APIs to create, read, update, and delete a OneDrive user's folders, files, albums, photos, and videos, and to read, create, and delete any associated tags and comments.
* **Outlook.com** for working with contacts and calendars. With the Live Connect APIs, your apps can create and read an Outlook.com user's contacts, and can create, read, update, and delete the user's calendars and their associated events.
* **Live** for authenticating users and accessing their profile info.

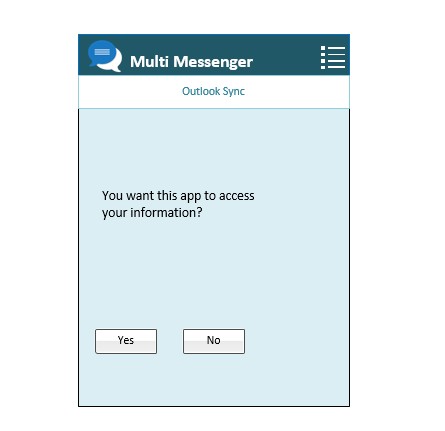
**OAuth 2.0** is the latest version of the OAuth protocol, an open standard for authenticating users' credentials. Live Connect, along with other social networking APIs, has adopted OAuth 2.0 as its authentication standard.

**REST** is an architectural style that's become popular in web services. Our use of REST makes it easy for you to request users' info through the Live Connect APIs. This REST implementation supports standard HTTP methods like **GET**, **PUT**, **POST**, and **DELETE**. We also include a few shortcuts to make certain requests even simpler to implement. For example, to get info about the signed-in user, you can use our shortcut **me** in your request.

## **Detail Design of Services**

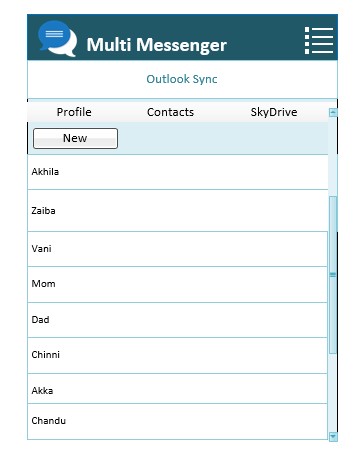
**Wireframes:**

1. **User Permission Access Screen**



Our application asks user whether he is willing to give access to the application to retrieve his private information from his profile.

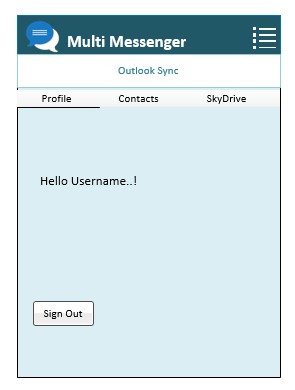
1. Outlook Contacts Sync Screen

****

Description:

We get the list of all contacts that are on outlook profile when we attempt to synchronize the contacts into our application. We give an option to create a contact on the device to the user in our application.

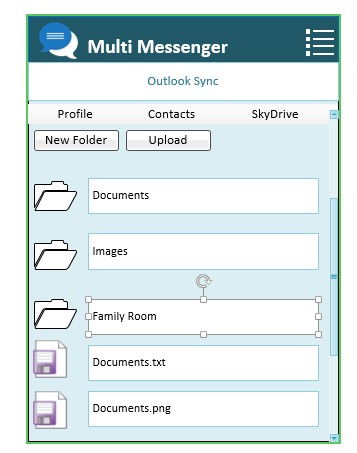
1. Outlook Profile Page



Description:

The profile information of outlook is displayed in this tab.

1. Outlook Skydrive Page



The information that is stored in skydrive is displayed in this page.

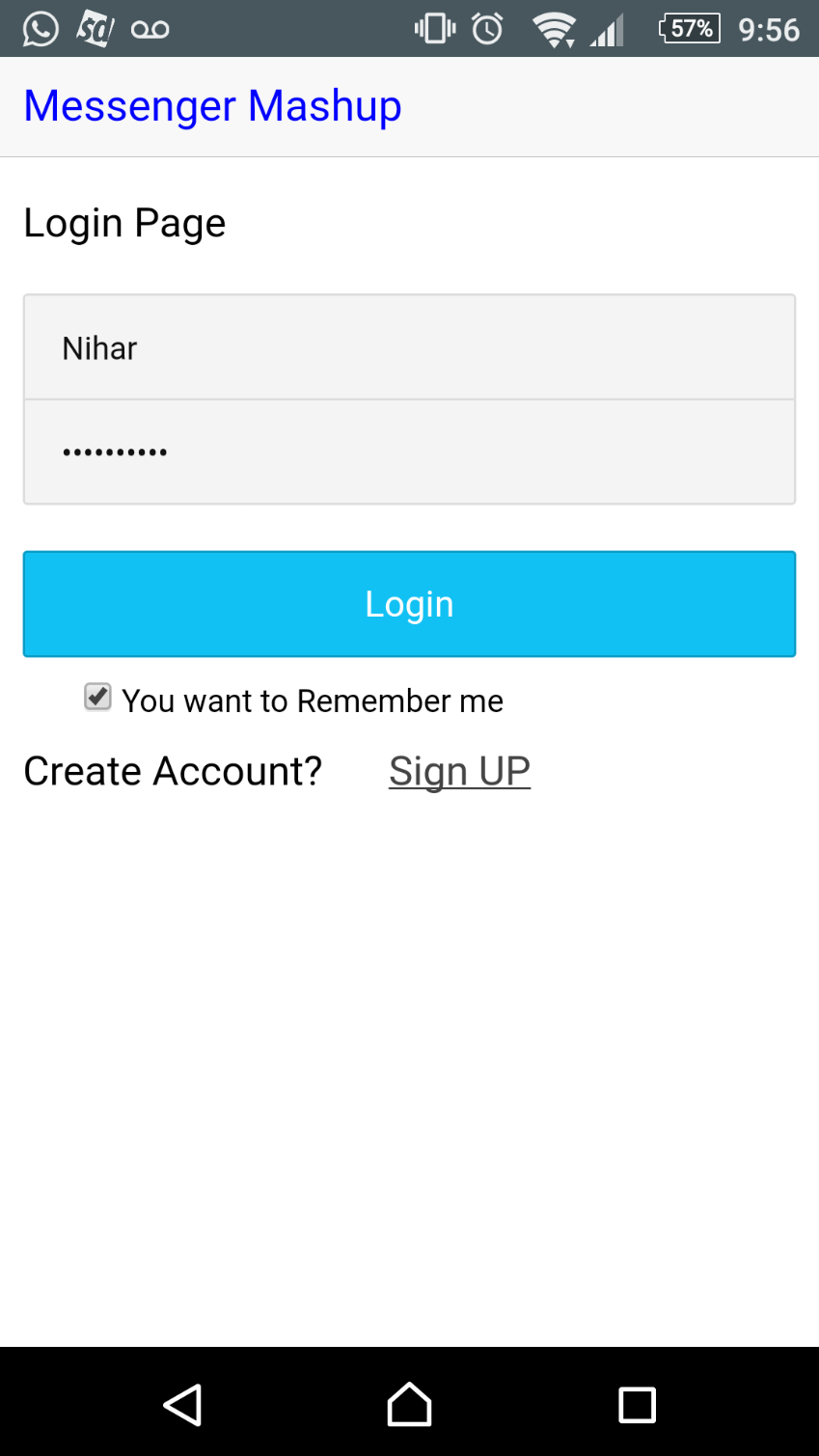
5. Signin Screen



User is allowed to signin into Outlook through this page.

**Mockups:**

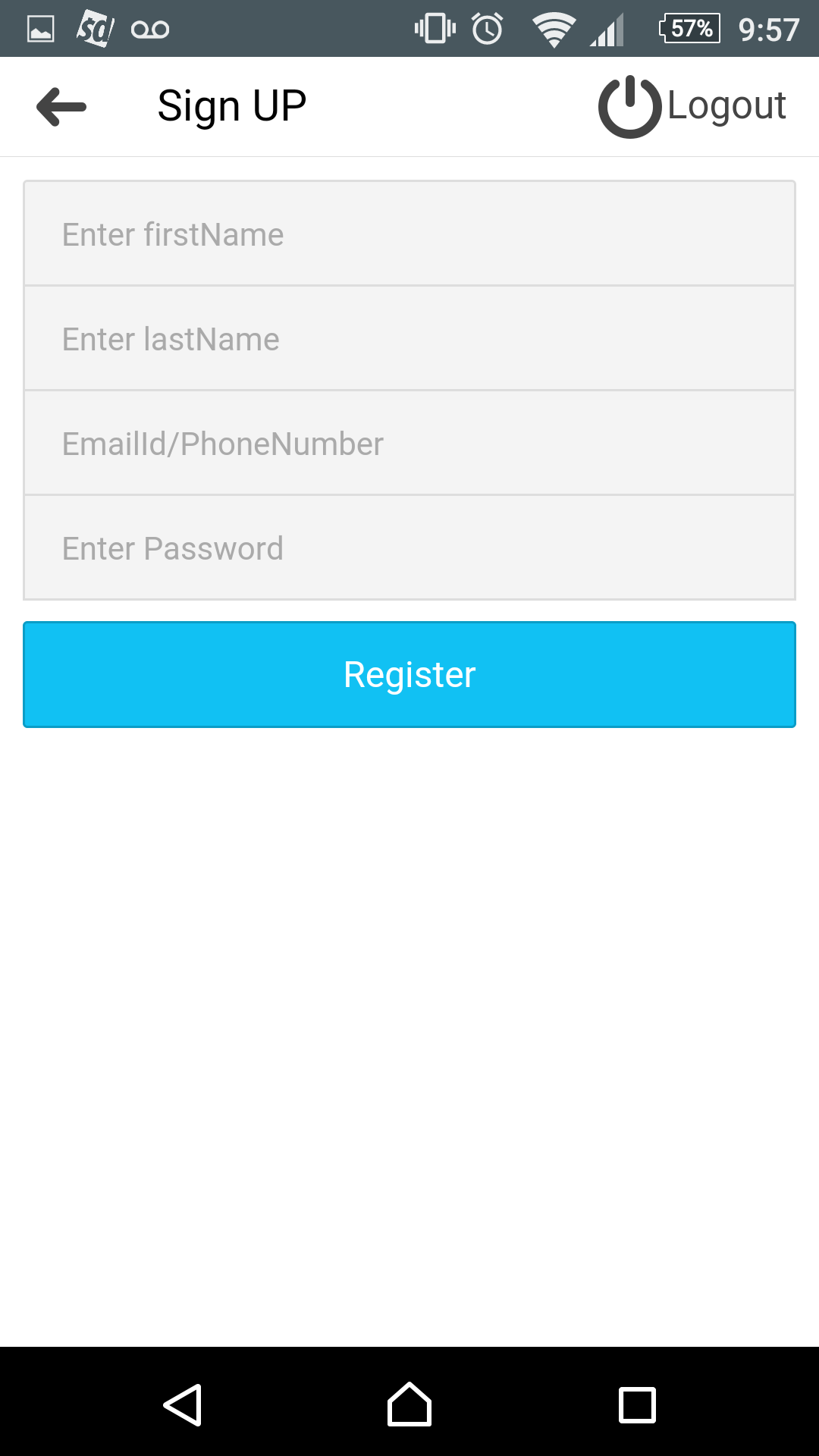
1. Login Activity:

****

Description:

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

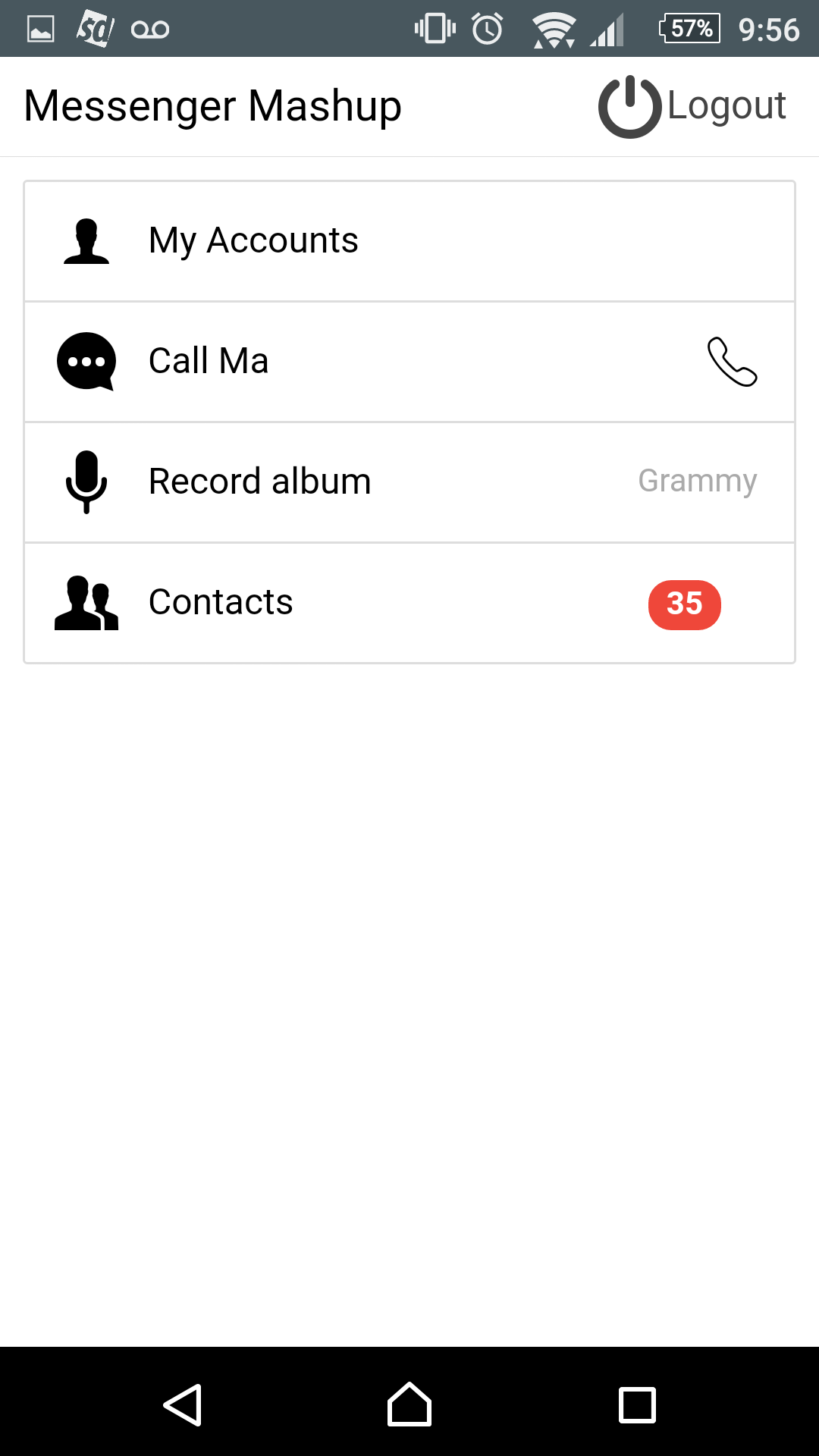
2.Register Activity:



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.

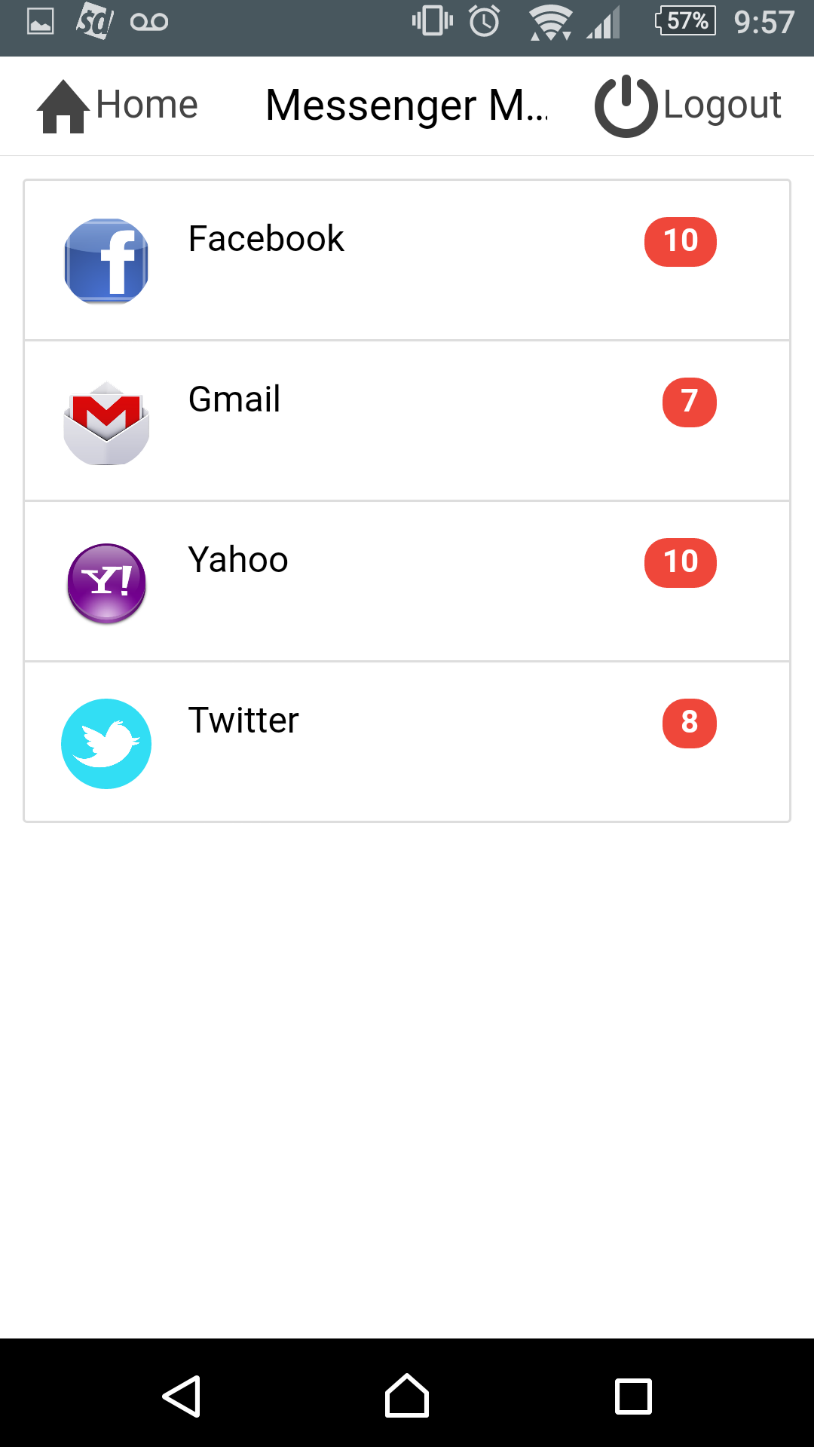
3. User Home Page:



Description:

The above screen displays the user homepage where he will be provided with various options to perform.

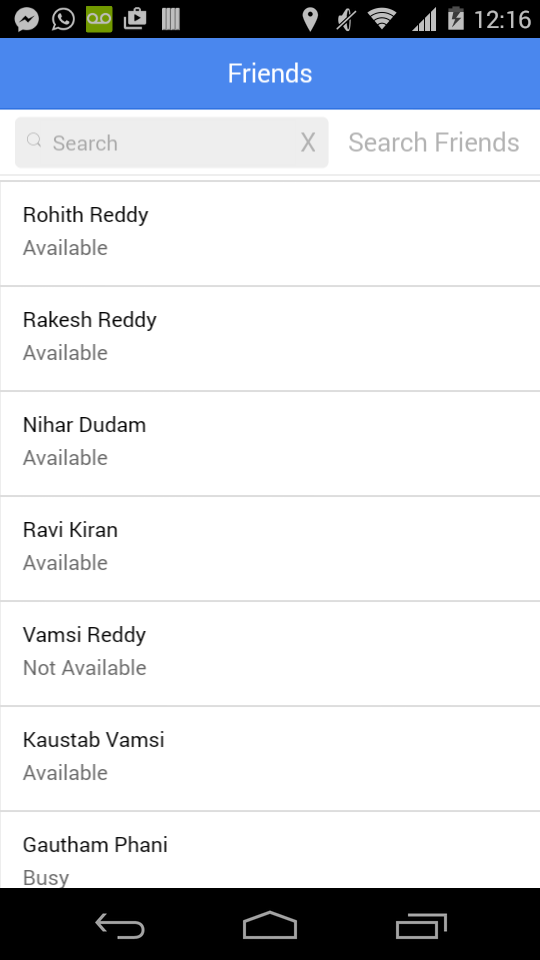
4.User Contacts Page :



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.

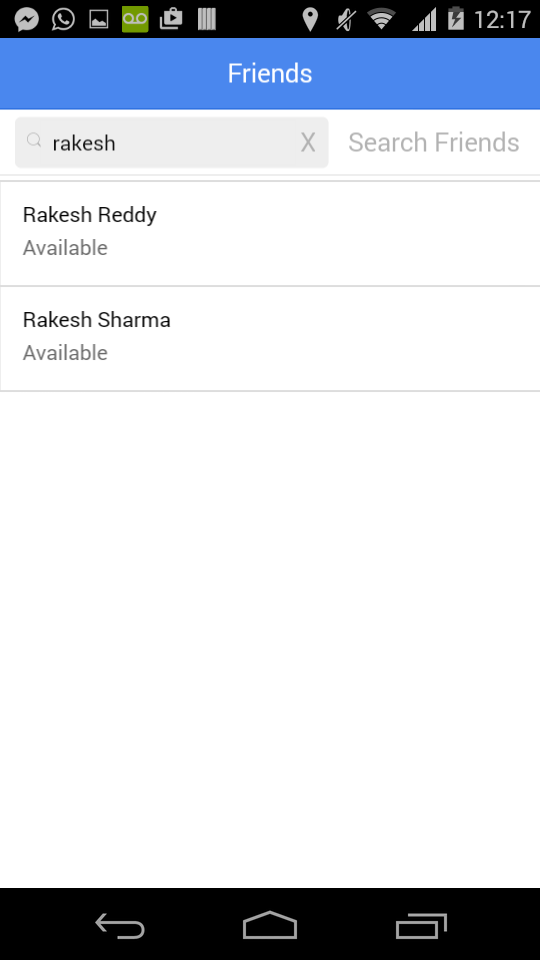
1. User Gtalk contacts:

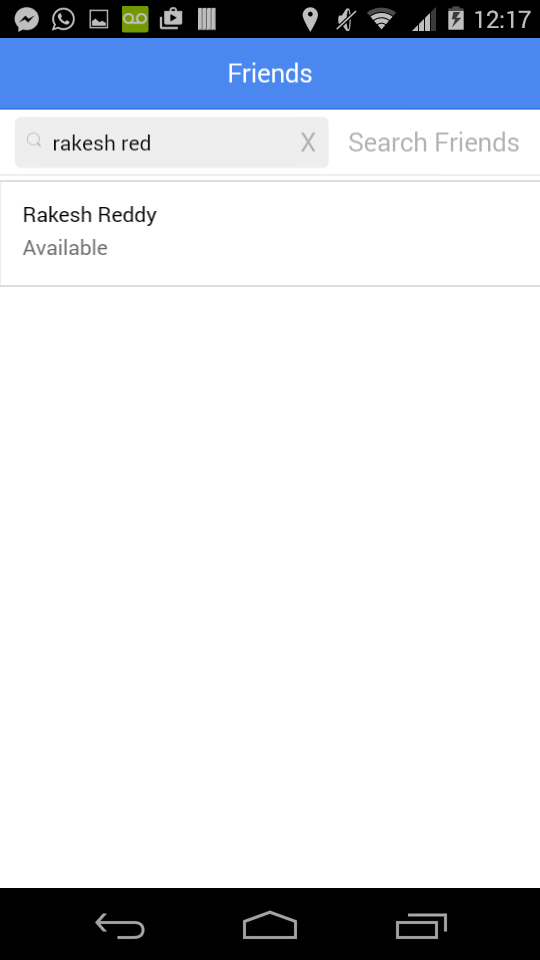


Description:

The above window displays us the gtalk contacts.

6. Friends Search :

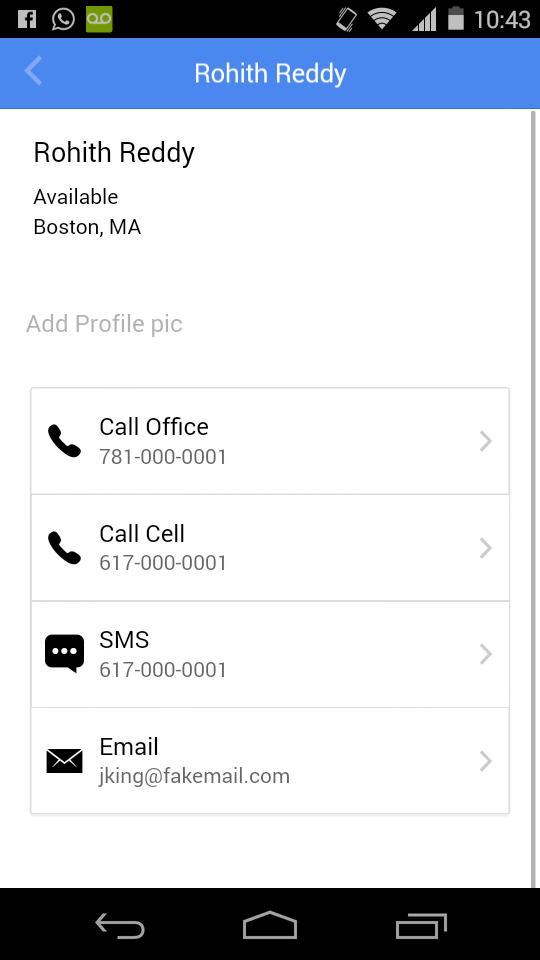




Description :

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

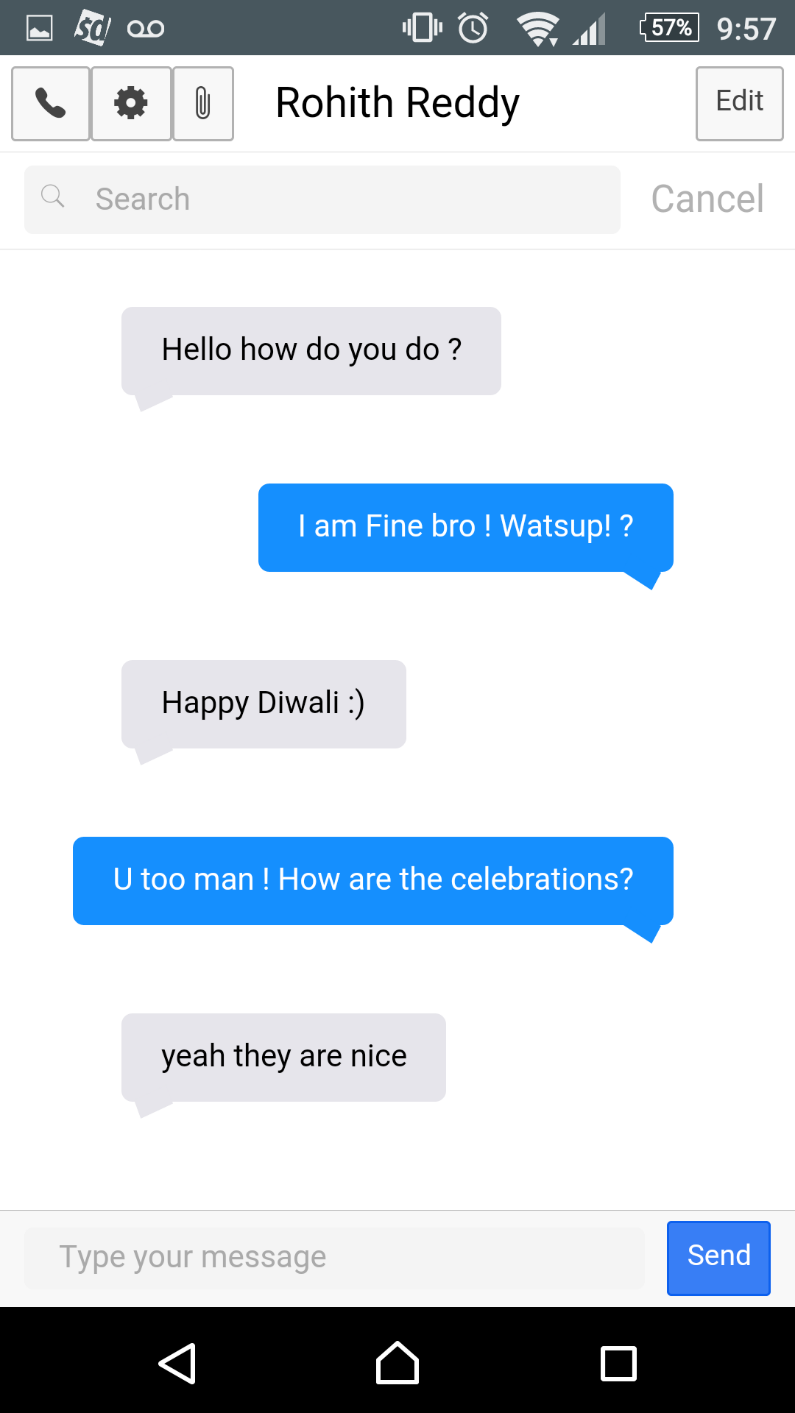
7.Specific Friend window :



Description :

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

8.Chat Screen :



Description :

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

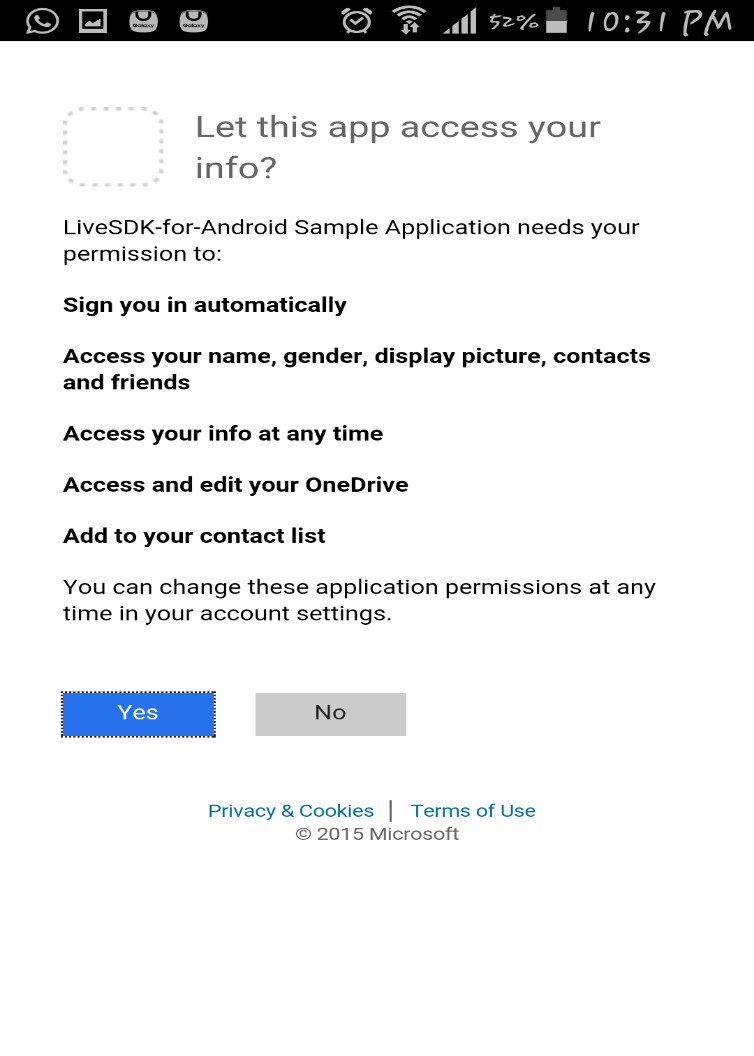
**Outlook Mockups**:



Fig: Home Screen

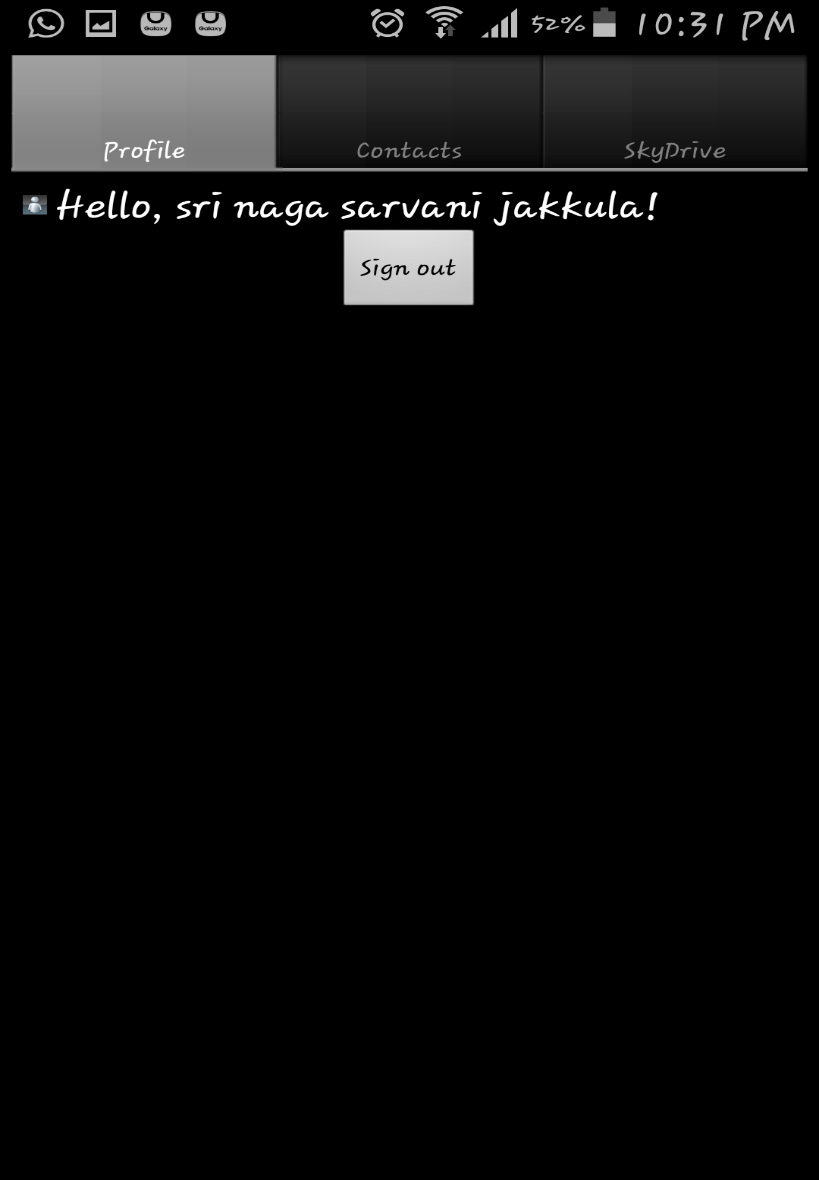
This Screen allows the user to sign in into outlook account. Here clicking Sign in button takes into outlook sign in page.

3 User Permission for access Page



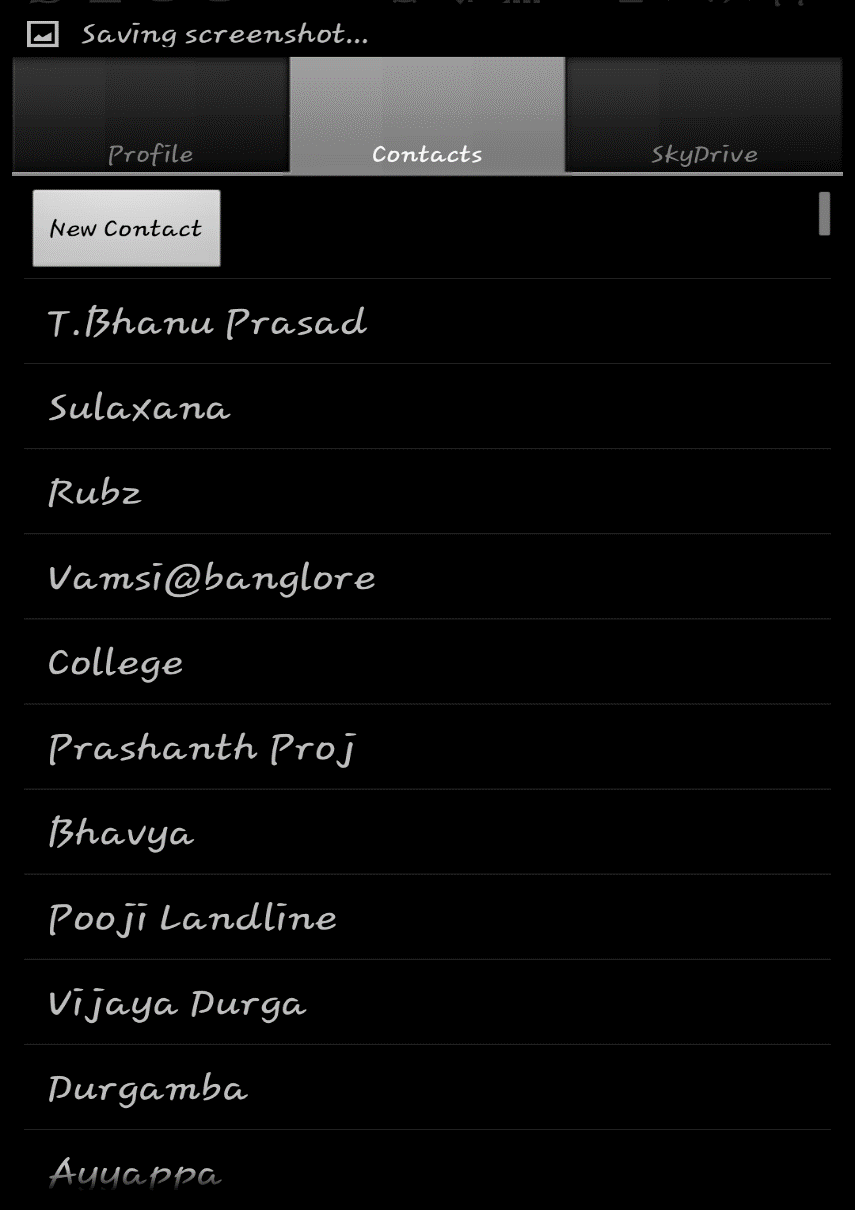
This screen asks the user whether he is willing to share his profile information with the application.

1. Profile Page



This page shows the profile data of the user after successful login into Outlook.com

6. Contacts Page



This page shows the synchronized contacts from outlook into the screen

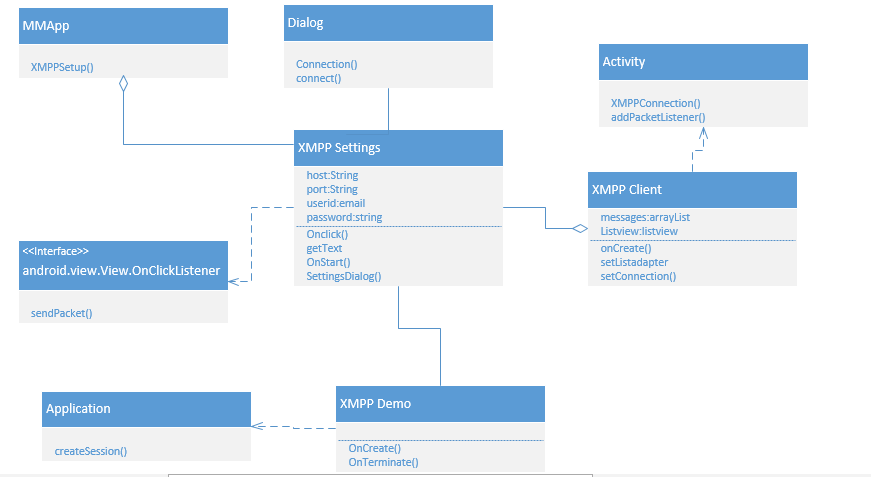
1. Skydrive Page



This page gives the relative sky drive information from the Microsoft one drive location

**Class Diagrams**:

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



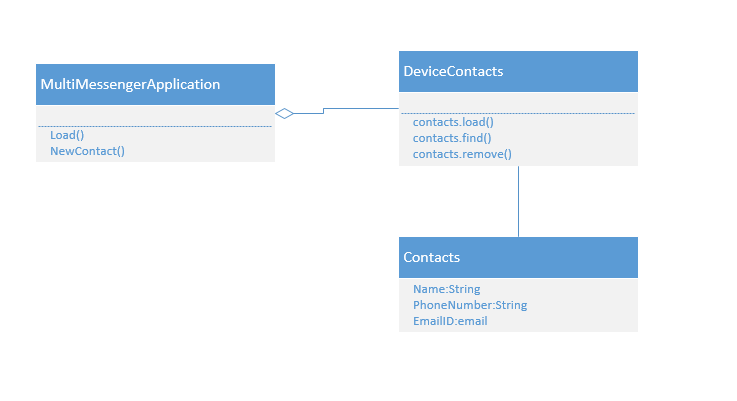
Description:

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



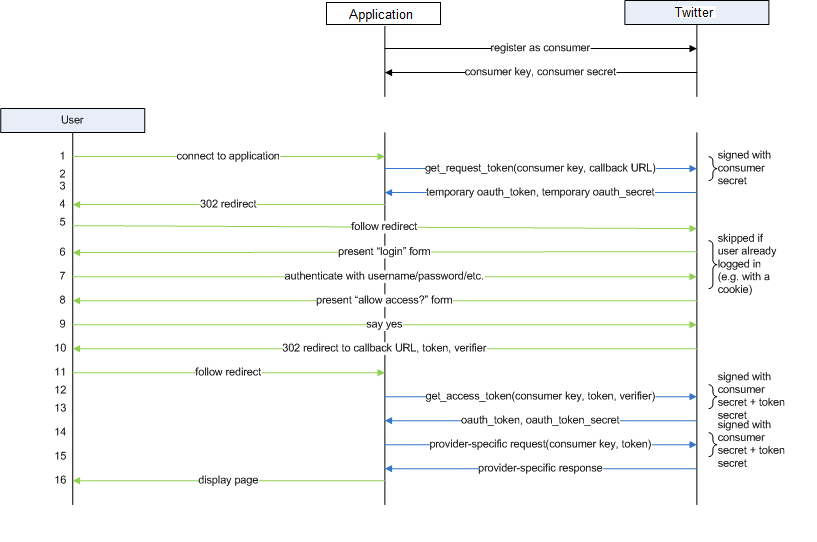
Description:

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:**

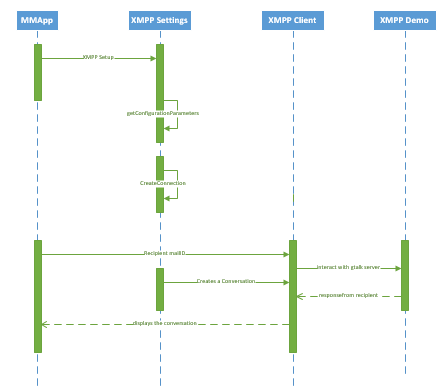
1. Sequence diagram for connecting to Twitter from the Application



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.

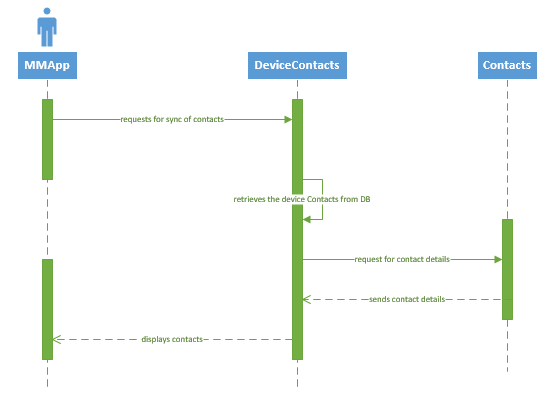
1. Sequence diagram for connecting to gtalk through XMPP



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 |

**Outlook Functionality**

1. **Author :**  Sarvani(30), Nihar(16), Ravi(67), Rakesh(6)
2. **Program Name :** Outlook Module
3. **Functionality Tested :** Sync Contacts
4. **Test Execution Procedure :** Manual
5. **Test Results Checking Method :** Manual
6. **Test Data :**

|  |  |  |
| --- | --- | --- |
| **Test Id** | **Username** | **Password** |
| T001 | [sar@live.com](mailto:sar@live.com) | Sarvani |
| T002 | [sar@yahoo.in](mailto:sar@yahoo.in) | Sarvani |
| T003 | [sar@live.com](mailto:sar@live.com) | Sarvani |
| T004 | [sar@live.com](mailto:sar@live.com) | @@@@@ |
| T005 | [sar@yahoo.in](mailto:sar@yahoo.in) | Sarvani |
| T006 | Empty | Sarvani |
| T007 | Empty | Empty |
| T008 | Empty | Empty |
| T009 | [sar@live.com](mailto:sar@live.com) | Empty |

1. **Test Conditions :**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test ID** | **Condition to be tested** | **Expected Result** | **Test Cycle** | | | |
|  |  |  | S | 1 | 2 | 3 |
| T001 | Outlook Validation | Successful |  | | | |
| T002 | Outlook Validation | Failed |  | | | |
| T003 | Outlook Validation | Failed |  | | | |
| T004 | Outlook Validation | Failed |  | | | |
| T005 | Outlook Validation | Failed |  | | | |
| T006 | Outlook Validation | Failed |  | | | |
| T007 | Outlook Validation | Failed |  | | | |
| T008 | Outlook Validation | Failed |  | | | |
| T009 | Outlook Validation | 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) | \*\*\* |
| T005 | 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**

**XMPP**, Extensible Messaging and Presence Protocol is a communications protocol for message-oriented middleware based on XML (Extensible Markup Language). It enables the near-real-time exchange of structured data between any two or more network entities. Using this we can send or receive messages or data packets on a network.

**Smack API**, Smack is an Open Source XMPP (Jabber) client library for instant messaging and presence. A pure Java library, it can be embedded into your applications to create anything from a full XMPP client to simple XMPP integrations such as sending notification messages and presence-enabling devices.

Connecting to XMPP server requires knowing of configuration parameters set by XMPP Server.

XMPP connectivity details for Google Talk:

*Host: talk.google.com*

*Port: 5222*

*Service: gmail.com*

*Username and Password*

There are certain methods in Smack API in order to connect to server through XMPP. The *XMPPConnection* class is used to create the connection to the XMPP server specified by the *ConnectionConfiguration* class which has all the configuration parameters, which uses configuration parameters for establishing connection with the server and to disconnect, use the disconnect() method. Once a connection is established, the user should log in with username and password using the login () method of the Connection class.

We use Chat class of smack API to send messages for the specified recipient. The Chat Manager instance is obtained from XMPPConnection using the getChatManager () method. Chat Manager keeps track on all current chats. Chat is created which will now be series of messages exchanged between two users.

The send Message (String msg) or send Message (Message msg) method is used for sending text messages or message object in the context of a given chat session. Moreover, *MessageListener* can be used to get callbacks of notification of Message from other users on chat. To receive messages, we use asynchronous mechanism through the use of packet listener.

We used Contacts plugin available in ***ngCordova*** list of plugins to synchronize device contacts into our application. ***ngCordova*** Contacts plugin is used to create, remove and search contacts of the device. This plugin defines a global navigator. Contacts object which provides access to the device contacts database. The following are the methods which we use

* navigator.contacts.create: to create a contact on the device
* navigator.contacts.find: to search the contact on the device.
* navigator.contacts.pickContact: method launches the Contact Picker to select a single contact.

**Twitter API:**

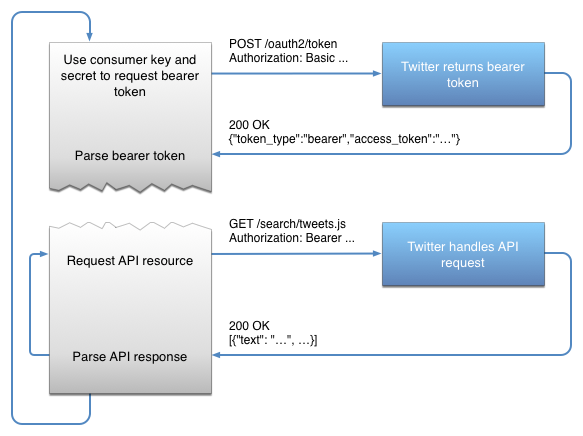
Twitter REST APIs provide programmatic access to send and receive direct receive messages ,to read and write Twitter data, to update statuses, to retrieve status, author a new Tweet, read author profile and follower data, and more. The REST API identifies Twitter applications and users using OAuth; responses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

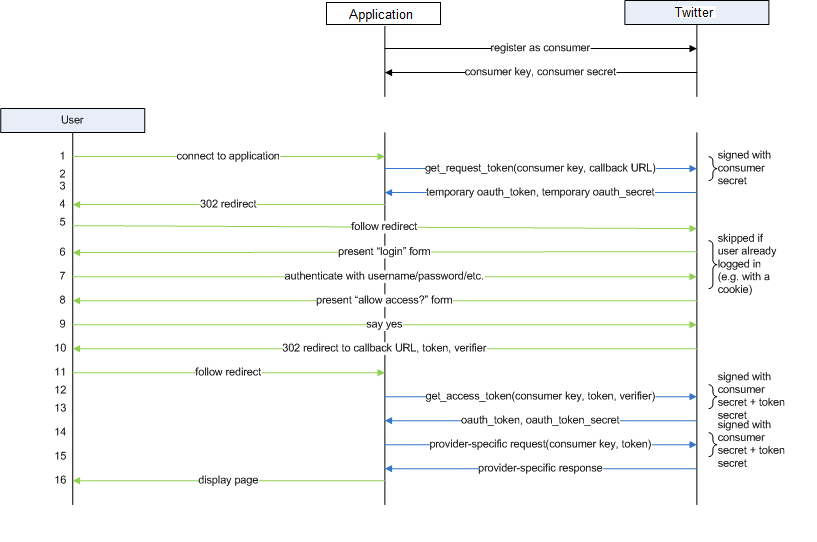
User-Agent: My Twitter App v1.0.23

Authorization: Bearer AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA%2FAAAAAAAAAAAA

AAAAAAAA%3DAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Accept-Encoding: gzip

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



**Yahoo API:**

Yahoo Messenger is instant messaging platform, used on a wide variety of desktop and mobile clients. Yahoo SDK offers developers a platform to manage contacts, real-time instant communications, and data transfer to and from clients throughout the world.

Implementation Steps:

1. We need to create a project for our application in order to access messenger data in our application, we will need to get a Yahoo API key and configure it to use Yahoo Messenger.You need to have an yahoo account for creating a project.
2. There are mainly five steps for using this Yahoo API.
   1. Creating a Yahoo Messenger Open Authentication (OAuth) API Key.
   2. Authenticate with the Yahoo Messenger servers.
   3. Creating a new session
   4. Obtaining contact list.
   5. Sending and receiving messages.
3. Creating a Yahoo Messenger 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://developer.yahoo.com>

1. Authenticate with the Yahoo Messenger servers

There are two API calls that are required to authenticate with the Yahoo 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.yahoo.com.

The API call looks like :

https://login.yahoo.com/WSLogin/V1/get\_auth\_token?&login=username&passwd=mypassword&oauth\_consumer\_key=consumerkey

The result of this call gives us a request token of format:

RequestToken=Juahfbjjbfsfsfnjjjnsj778sbfa3AYGU1KtB9vUbxlnzfIiFRLP...

The second API call gives us an access token which is used to perform various actions

<https://api.login.yahoo.com/oauth/v2/get_token>

Various input parameters for this API call are :

|  |  |
| --- | --- |
| oauth\_consumer\_key | The consumer key that will be validated |
| oauth\_signature\_method | PLAINTEXT |
| oauth\_nonce | random value |
| oauth\_timestamp | A Unix timestamp, expressed as "*seconds* from epoch." |
| oauth\_signature | Consumer Secret Key |
| oauth\_verifier | None |
| oauth\_version | 1.0 |
| oauth\_token | Request token obtained. |

Response of this API call gives us access token.

1. Creating session key :

*POST /v1/session*

*Host: developer.messenger.yahooapis.com*

*Authorization: < Standard OAuth credentials >*

*Content-Type: application/json;charset=utf-8*

*Content-Length: 2*

*{*

*}*

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

1. Obtaining contact list:

Example of fetching contact details:

*GET /v1/contacts?sid=msgrsessionid*

*Authorization: OAuth*

*realm="yahooapis.com",*

*oauth\_consumer\_key="dj0yJmk9nM9Y29uc3VtZXJzZWNyZXQmeD1lMg--",*

*oauth\_nonce="24829.2331",*

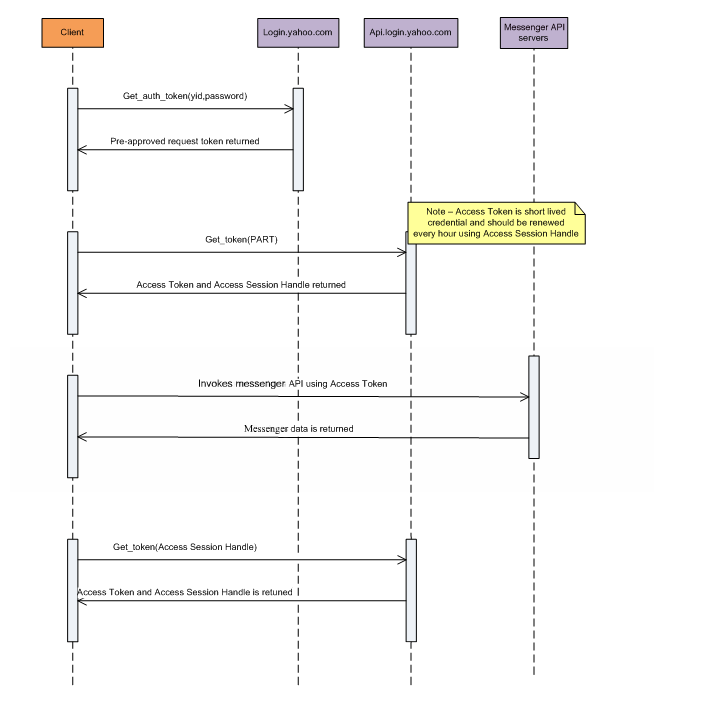
*oauth\_signature\_method="PLAINTEXT",*

*oauth\_timestamp="1219450170",*

*oauth\_token="A%3DuqkiebGpiTJl7ThQxU.jDXXaETYyfEy3xAKPyoavokwOOcZcz8Xs\_l1Nvnl.\_KmCEVCeLkxxT1Y6BgRqf5f98sQWHklBM\_anetveR7okK\_M\_5XEmQ1\_1reo3UgKQULT\_dQT8Gao3rgz5rJxgmnYrhdWWdfgTdMQVzpbJT2aGkz59NTK1O8yXVE1EvZUCqju7WiFYu.WHNEw.9TWq3g--",*

*oauth\_version="1.0",*

*oauth\_signature="O2AQipLITO0aYHKZc9266RzC94%260af0ef7f79bfb89dd6af87589e4c97b022f594a3"*



This is the sequence diagram for Yahoo API implementation using Oauth Authentication.

## **Deployment**

**Project Management:**

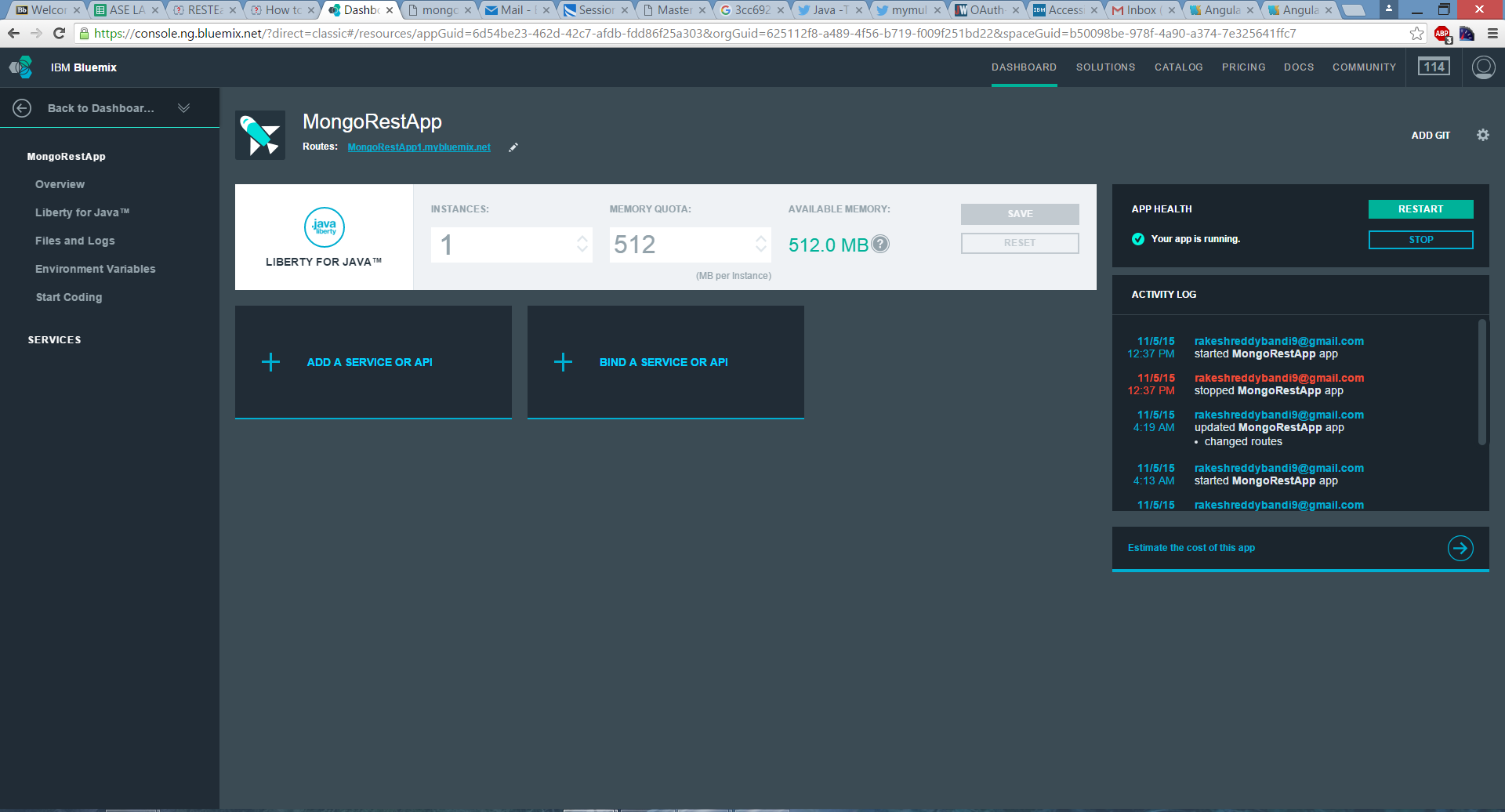
The Project management link of the Project is <http://niharase.kanbantool.com/b/180890-ase_project>.

**GitHub Link:**

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

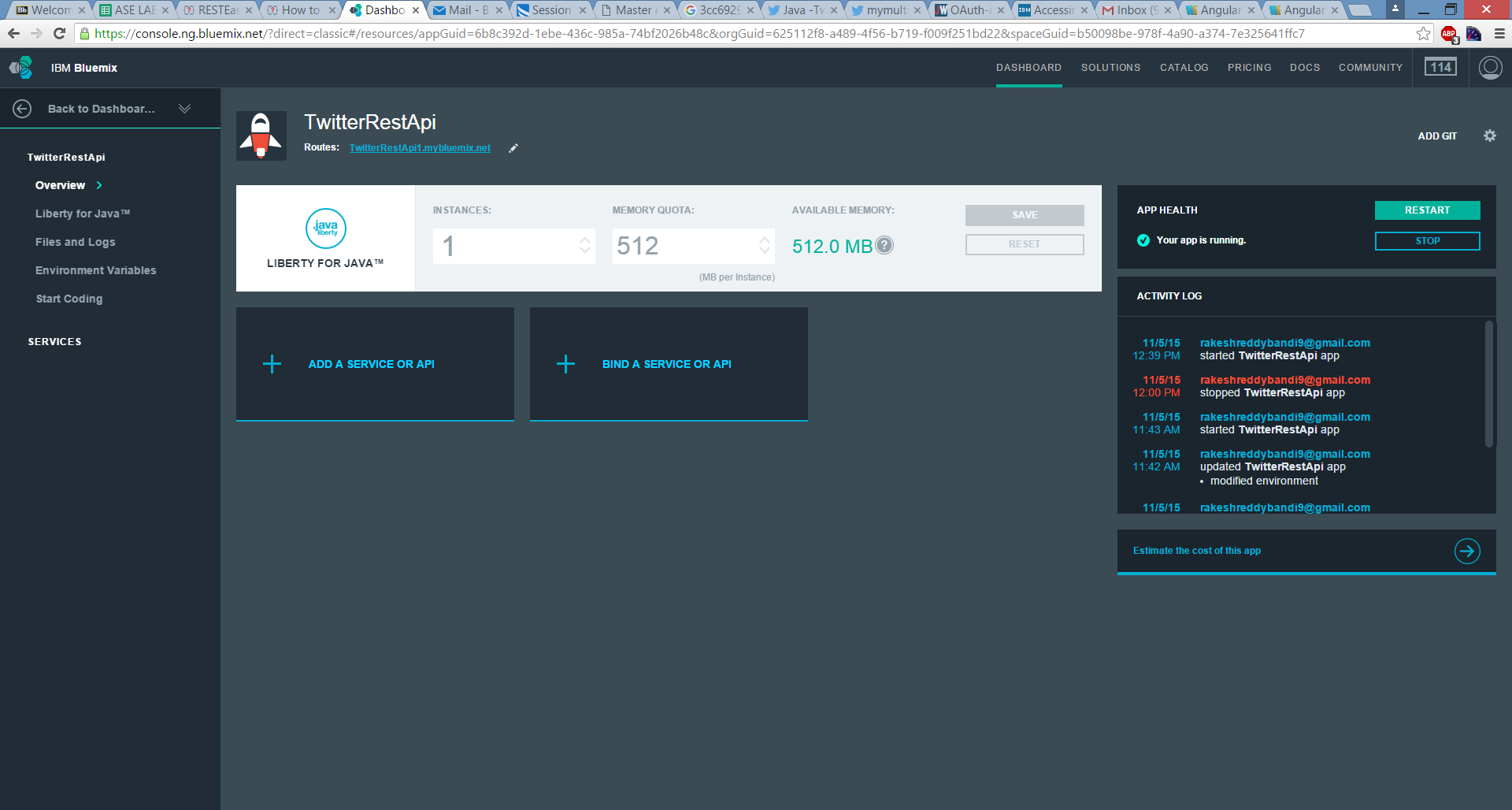
**IBM Bluemix Deployment Link:**

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



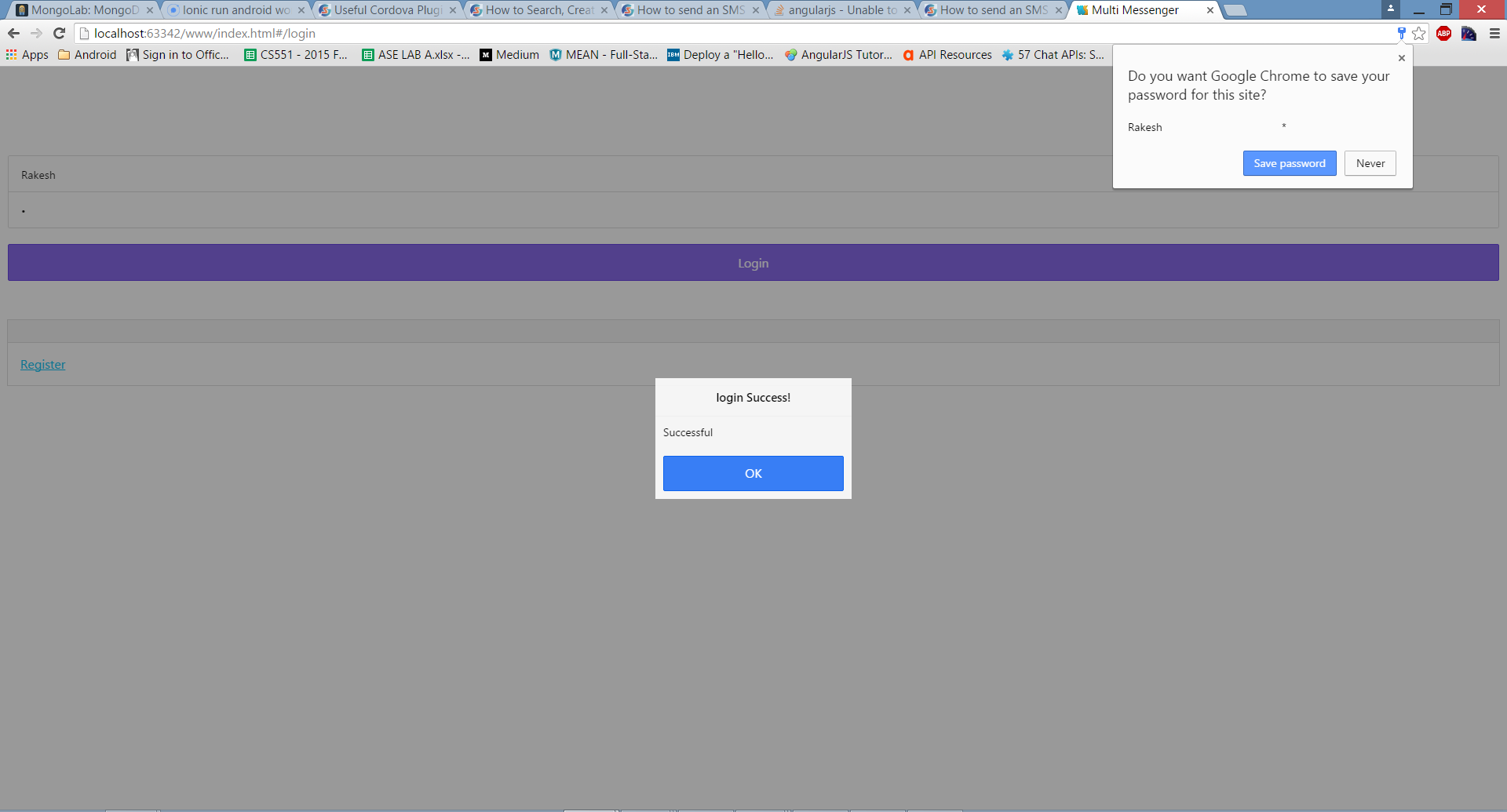
The Twitter Rest api that supports the twitter authentication and twitter calls for getting the contacts and sending the direct messages to the contacts list is:

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



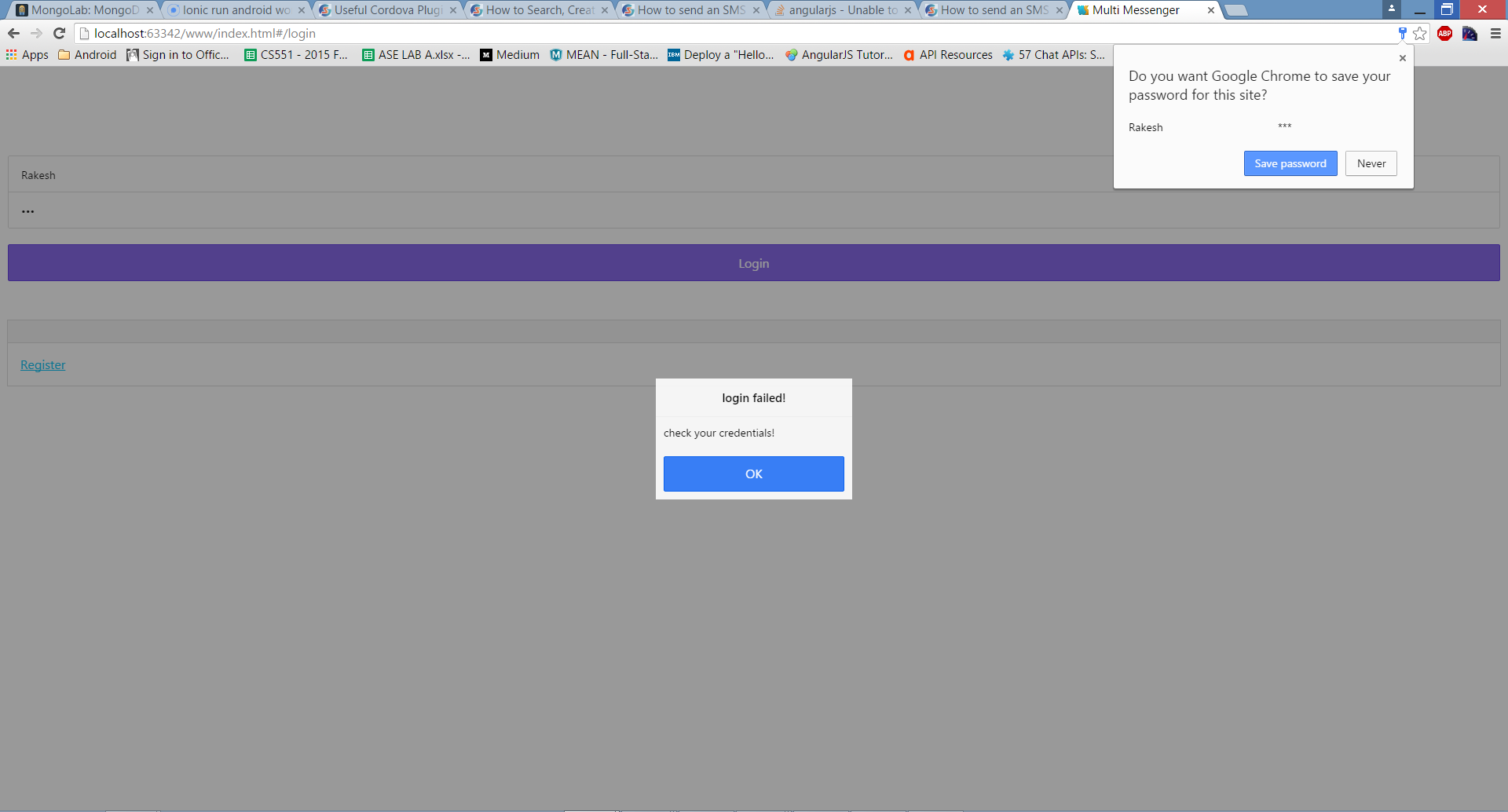
## **Report**

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



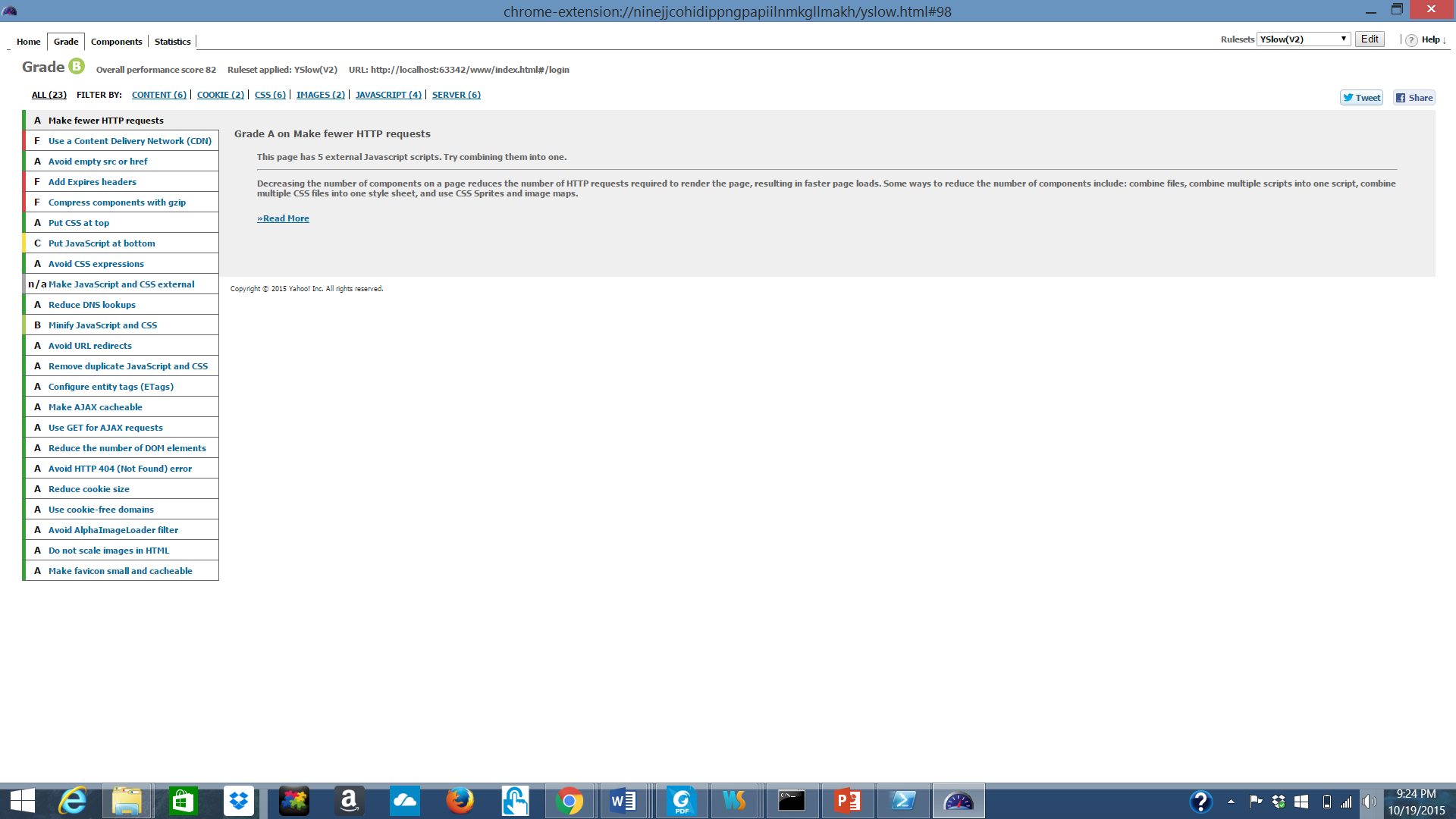
In this we have given the correct credentials username and password of user and logged in to the app. We get the Success message for correct credentials.

* + - * + Login Failure:



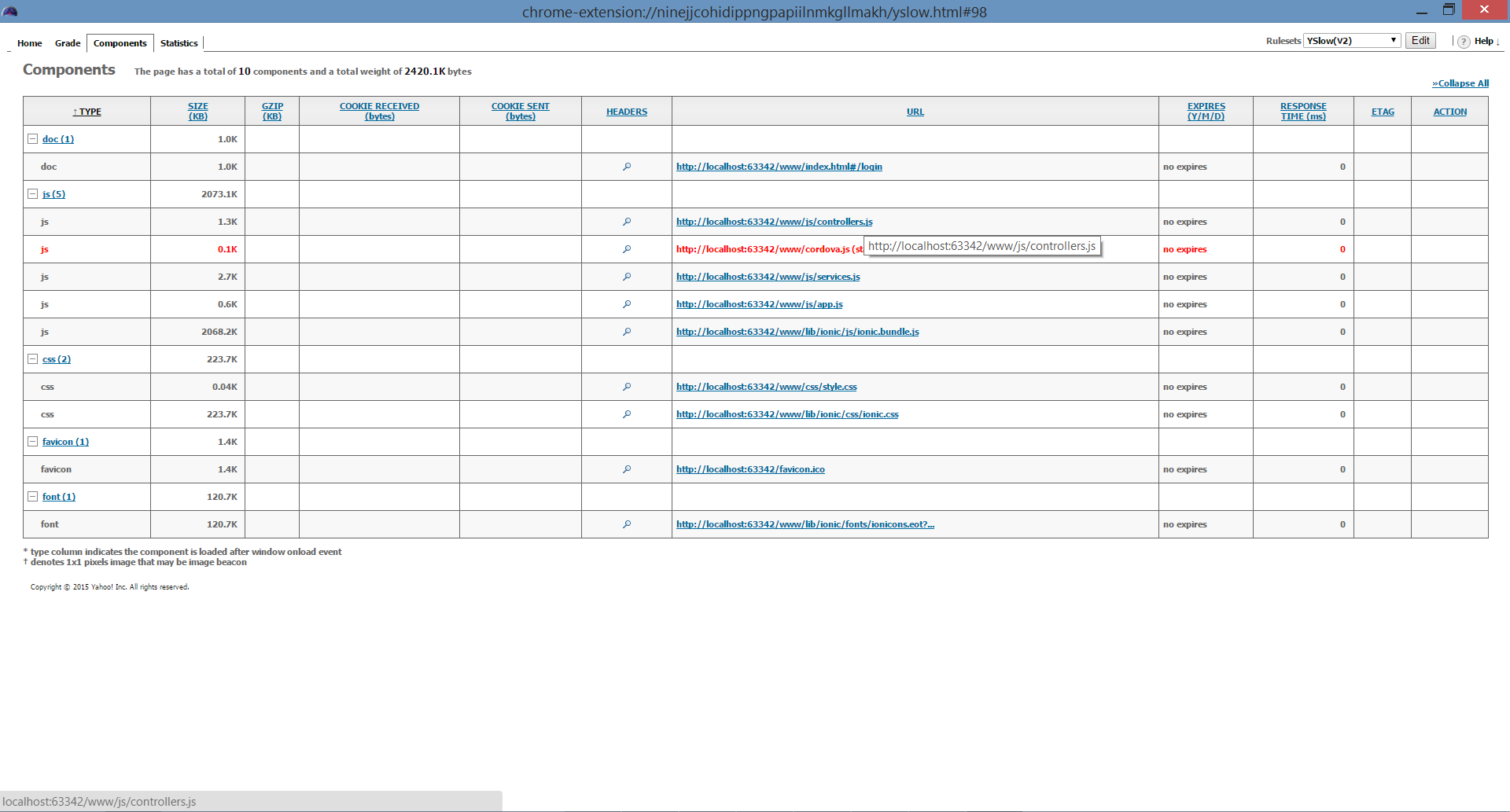
In this we have given the incorrect credentials username and password of user and logged in to the app. We get the Failure message for incorrect credentials.

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



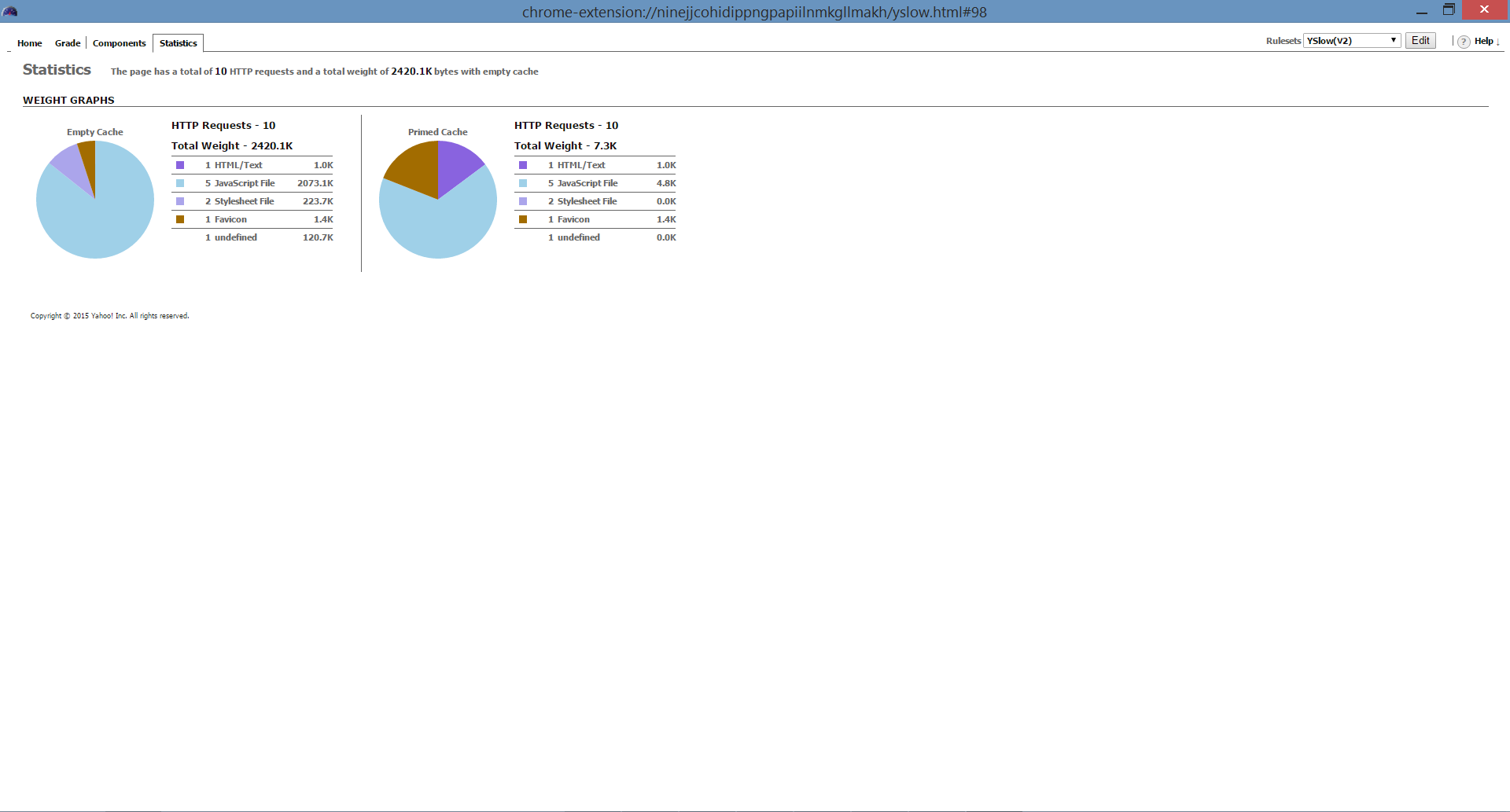
The above diagram describes about the Yslow grade for login

* + - * + Yslow components:



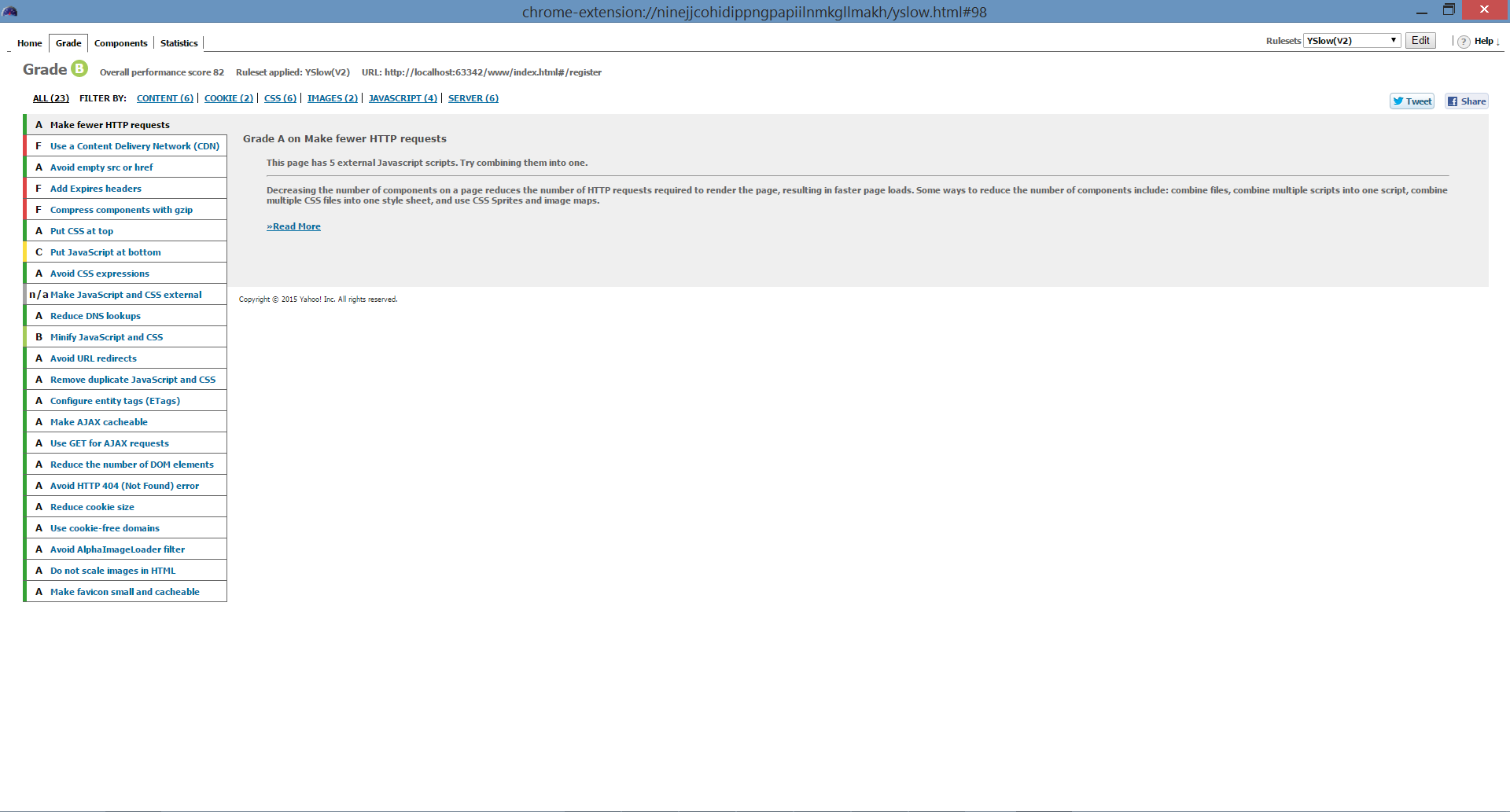
The above diagram explains about Yslow components for login page

* Yslow statistics:



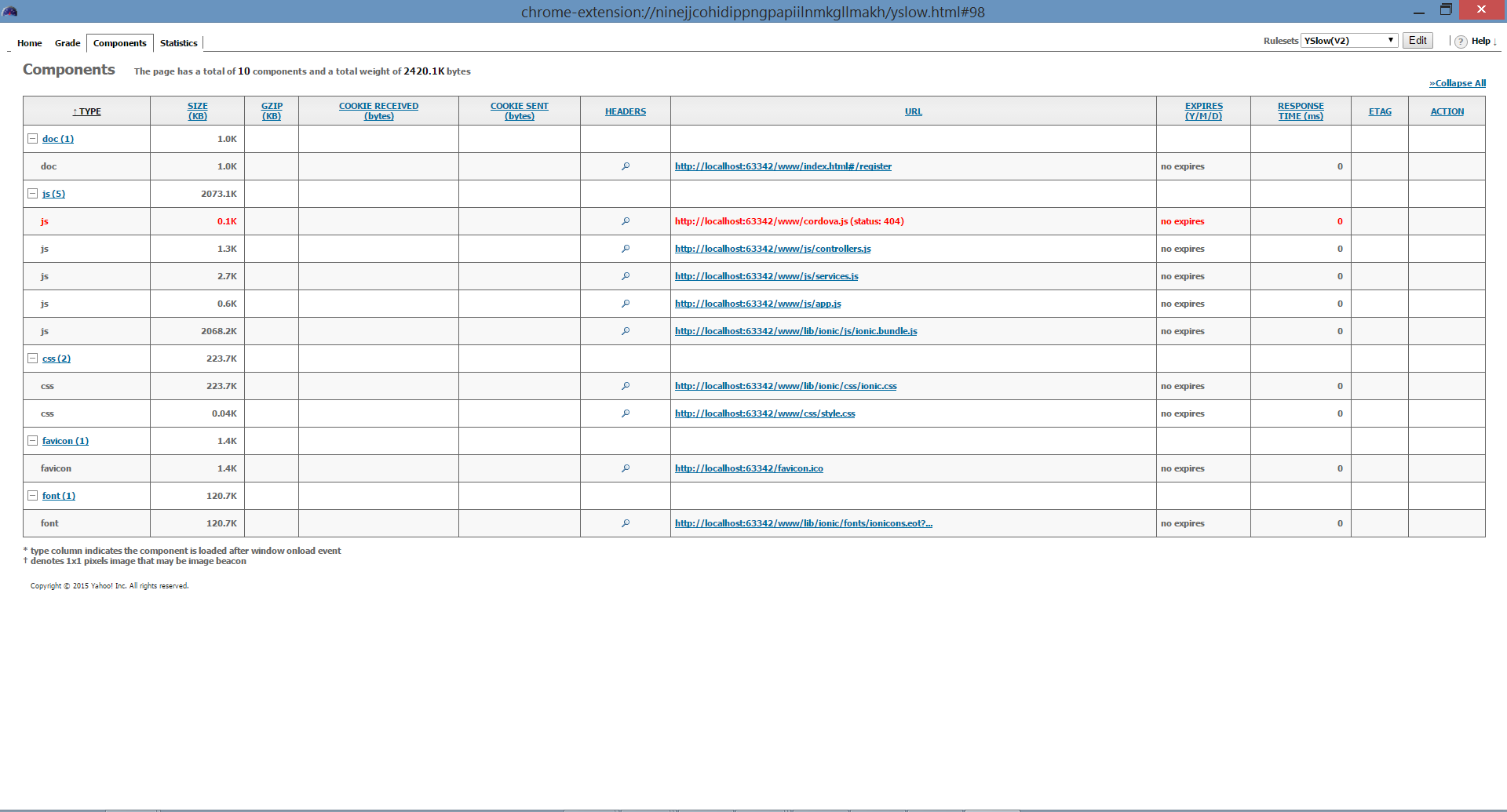
The above figure explains about Yslow statistics for login page.

* Register Page
* Yslow Grade:



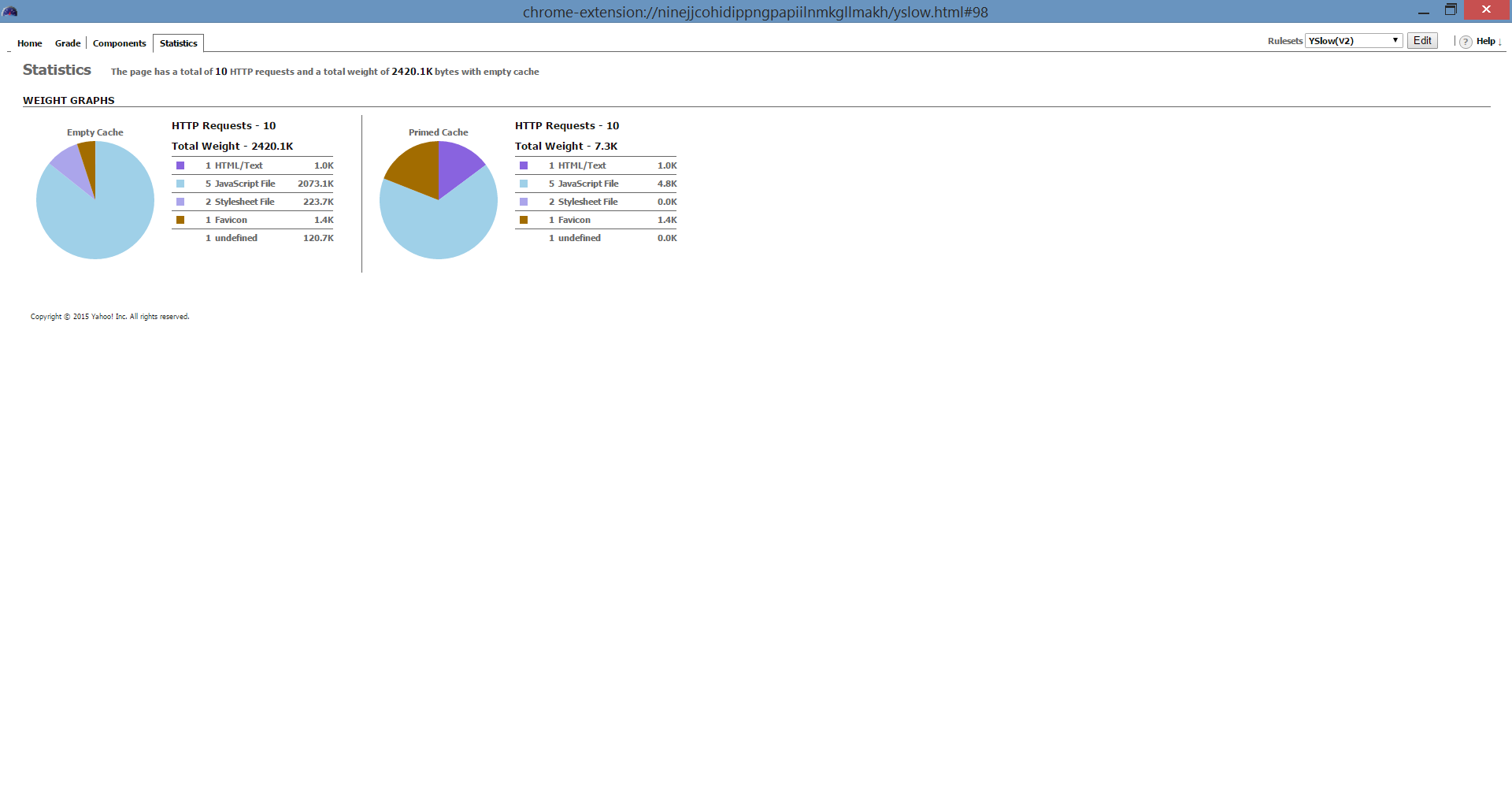
The above figure explains about Yslow grade for register page.

* Yslow component:



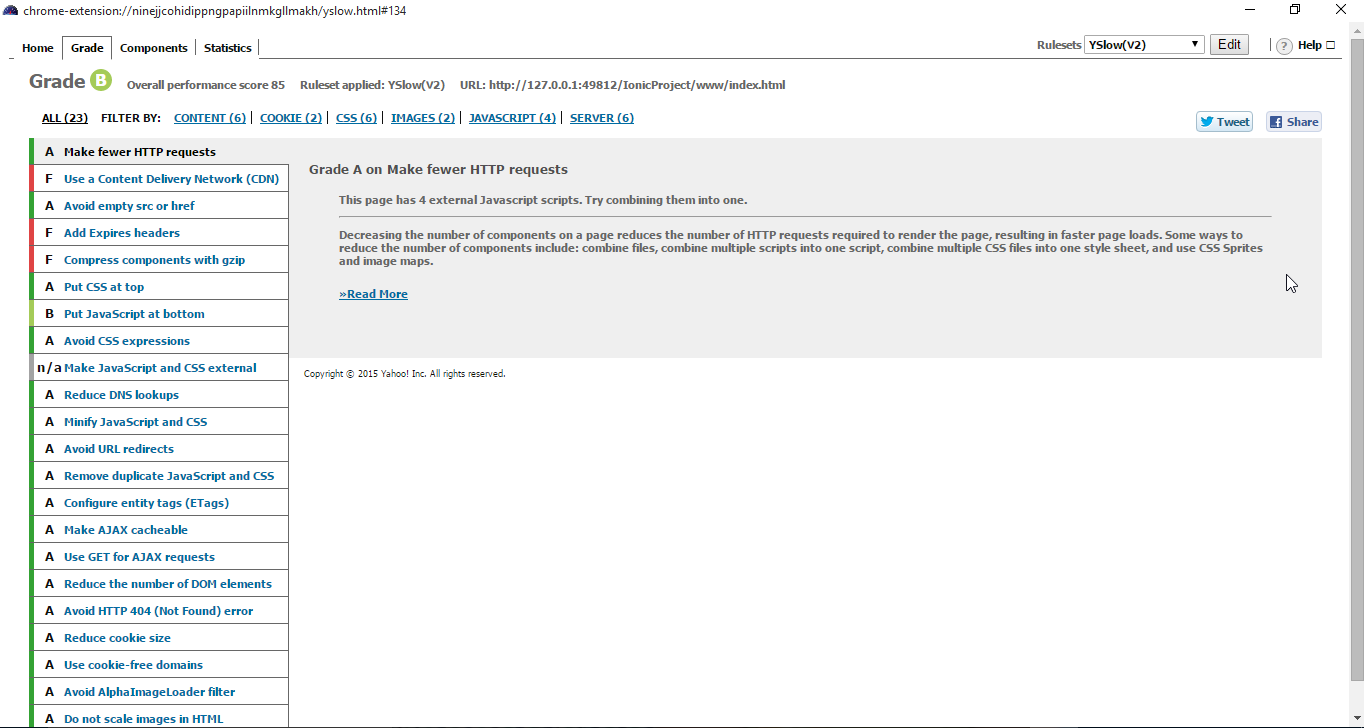
The above figure explains about Yslow components for register page

* Yslow statistics:



The above diagram describes about Yslow Statistics for register page:

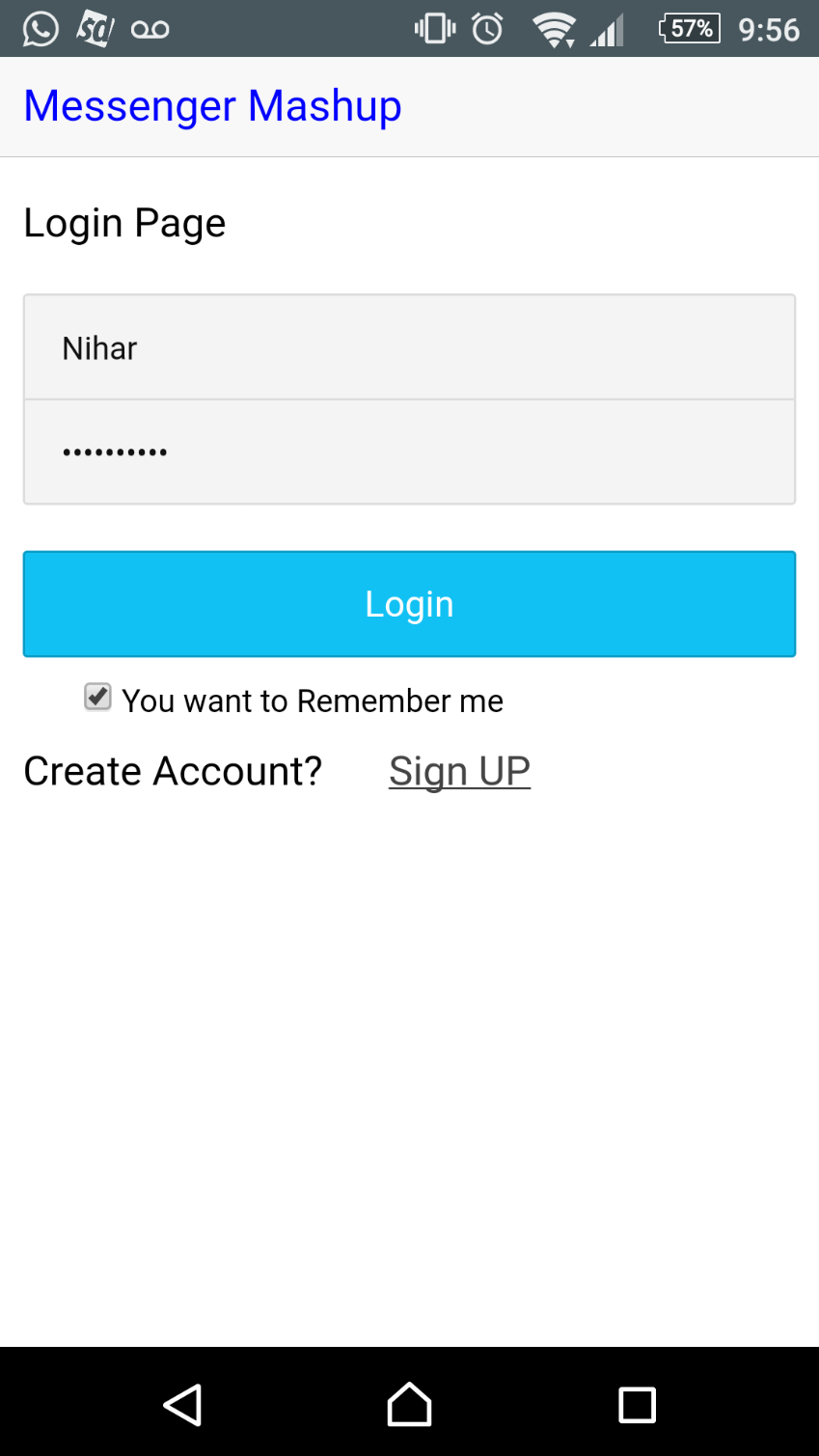
* Contacts Synchronization page :



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

**Project Implemenation Screenshots:**

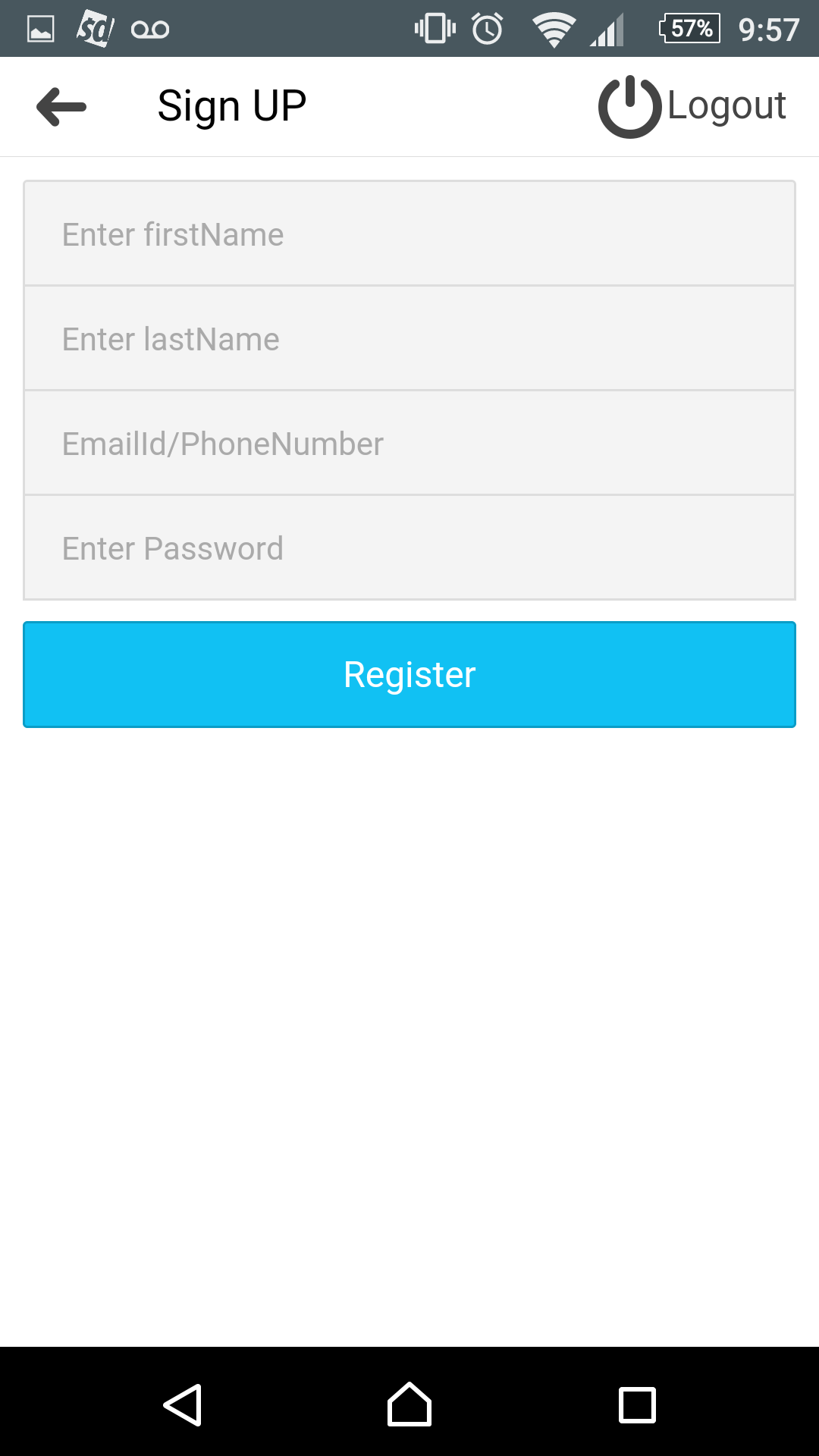
1. Login Activity:

****

Description:

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

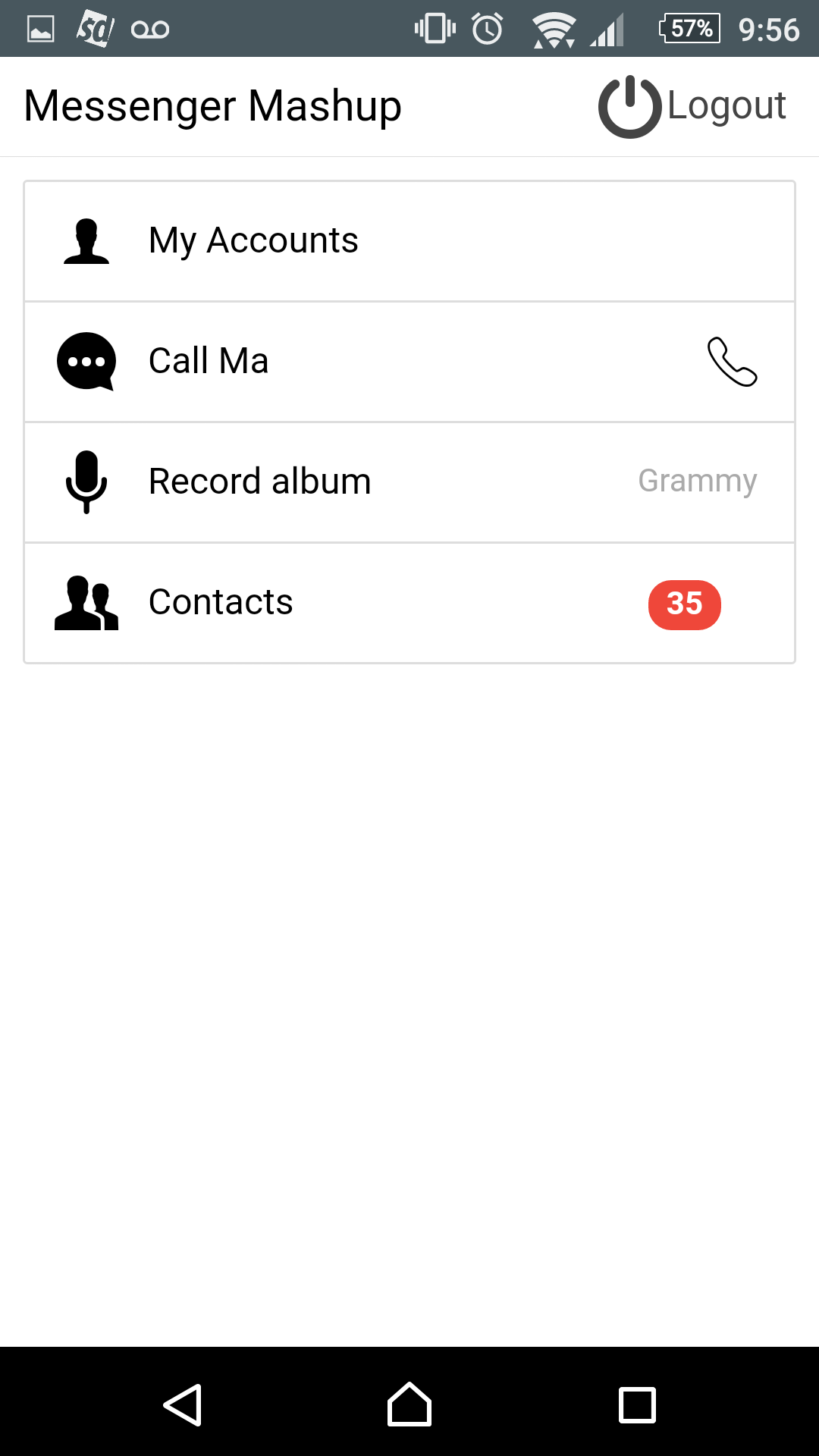
2.Register Activity:



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.

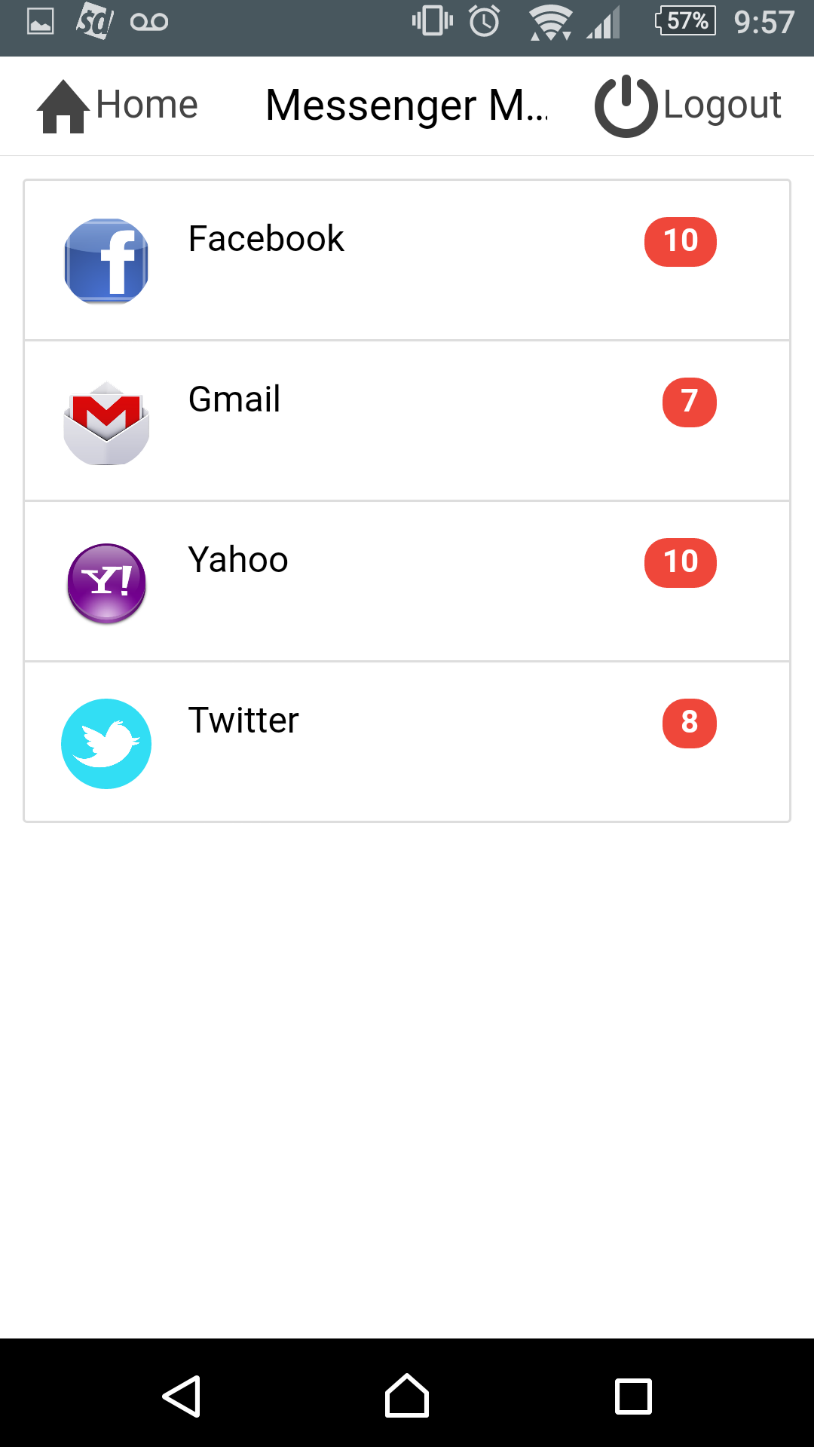
3. User Home Page:



Description:

The above screen displays the user homepage where he will be provided with various options to perform.

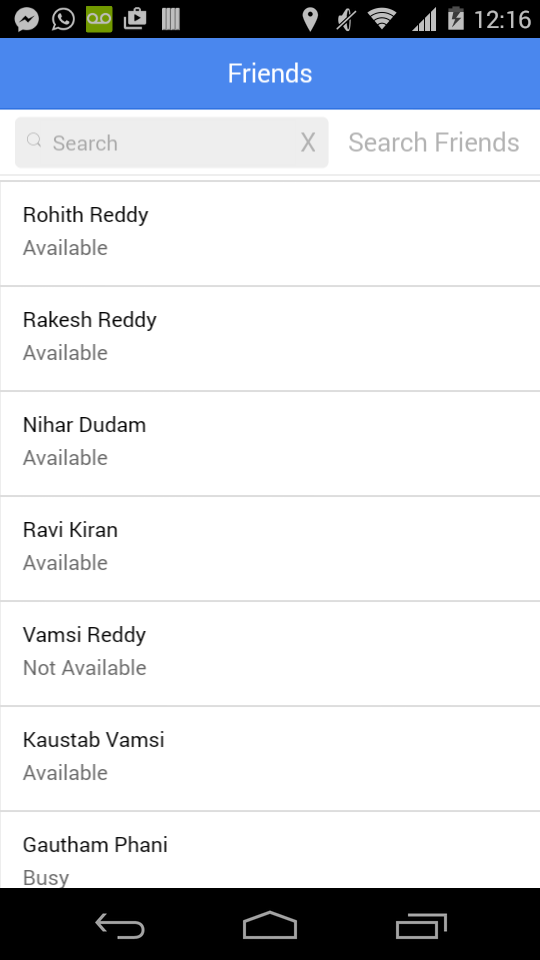
4.User Contacts Page :



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.

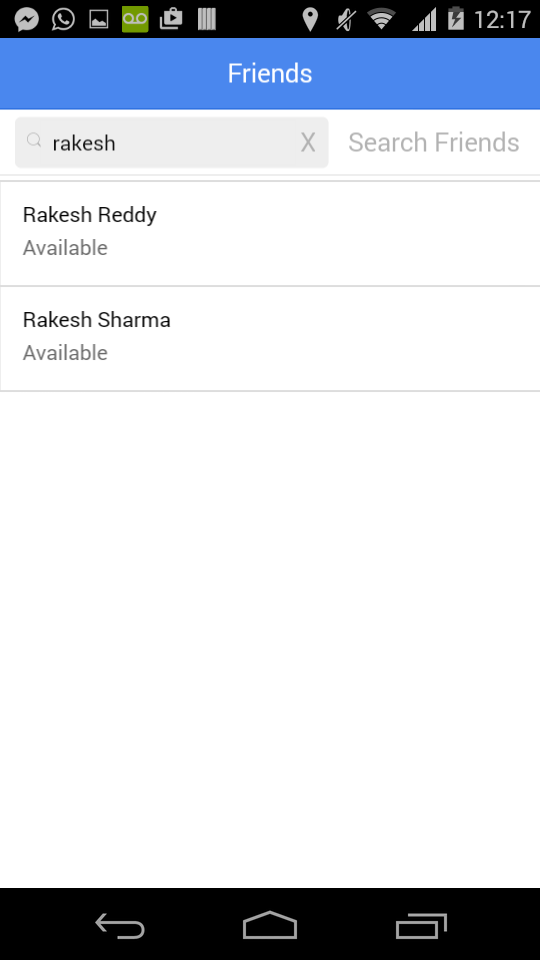
1. User Gtalk contacts:

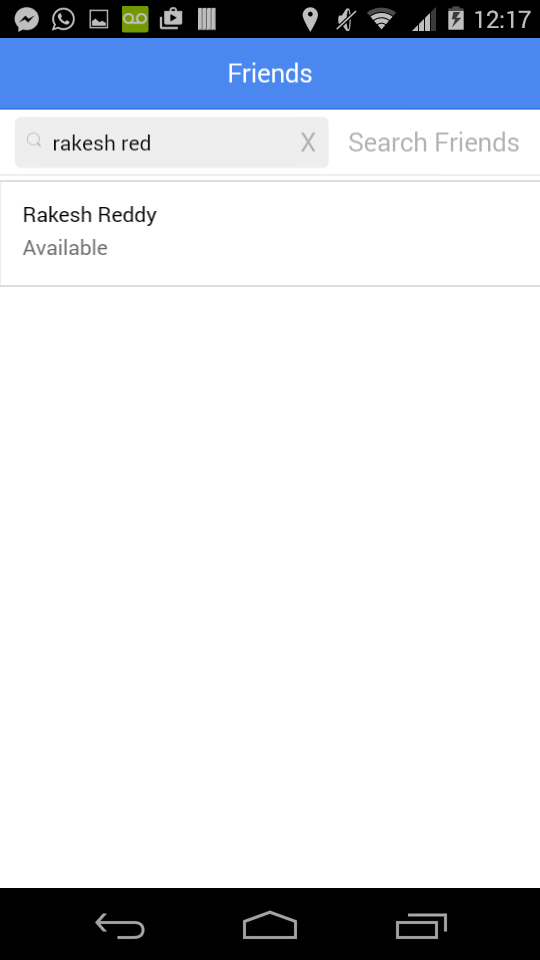


Description:

The above window displays us the gtalk contacts.

6. Friends Search :

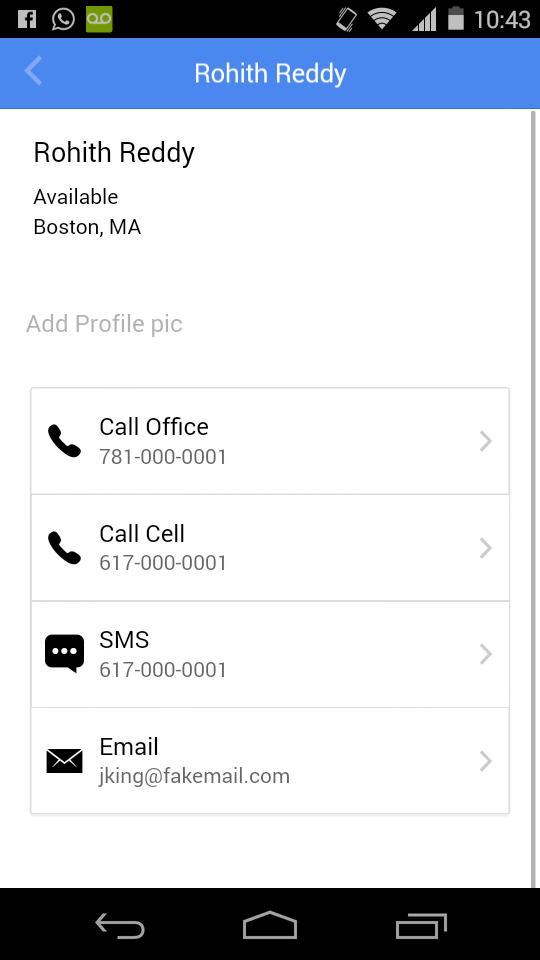




Description :

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

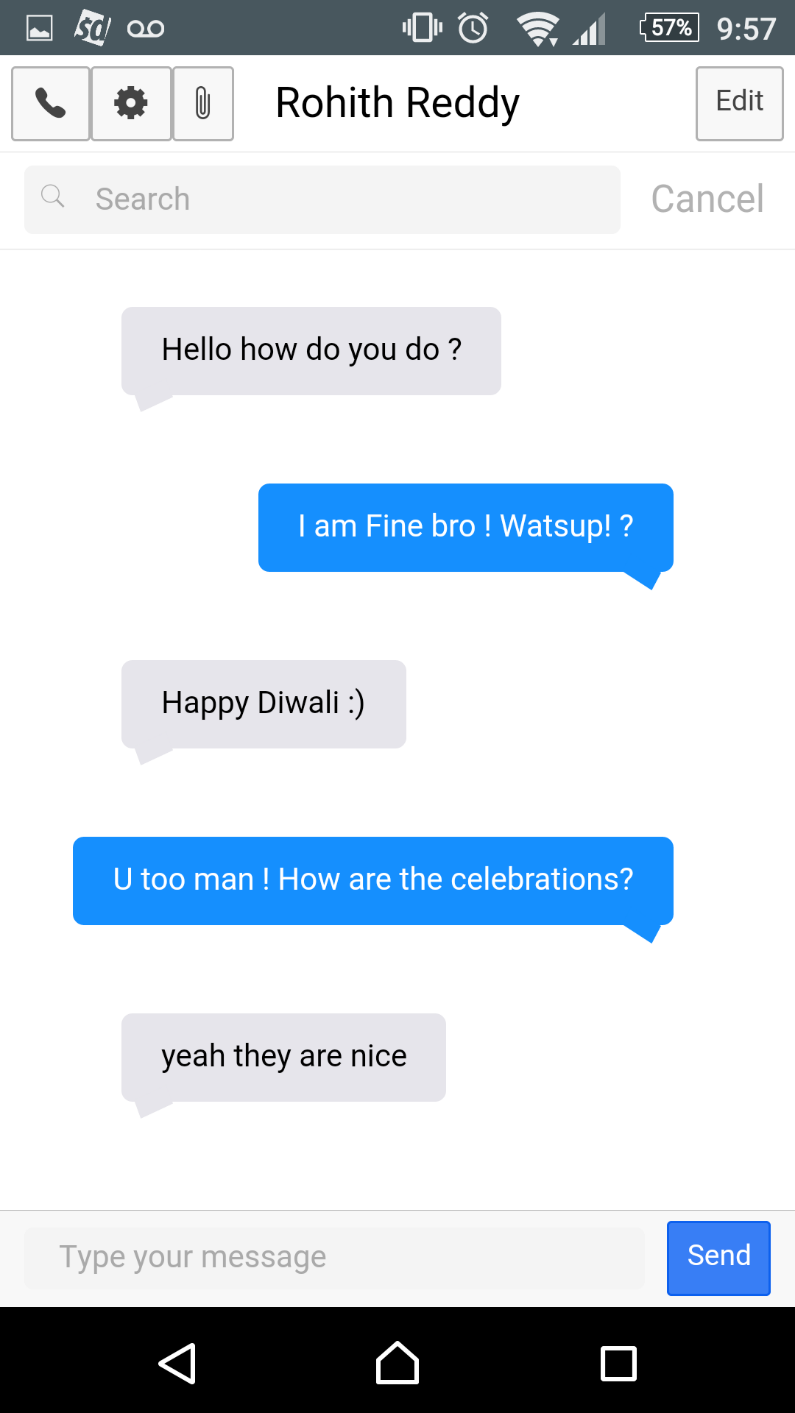
7.Specific Friend window :



Description :

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

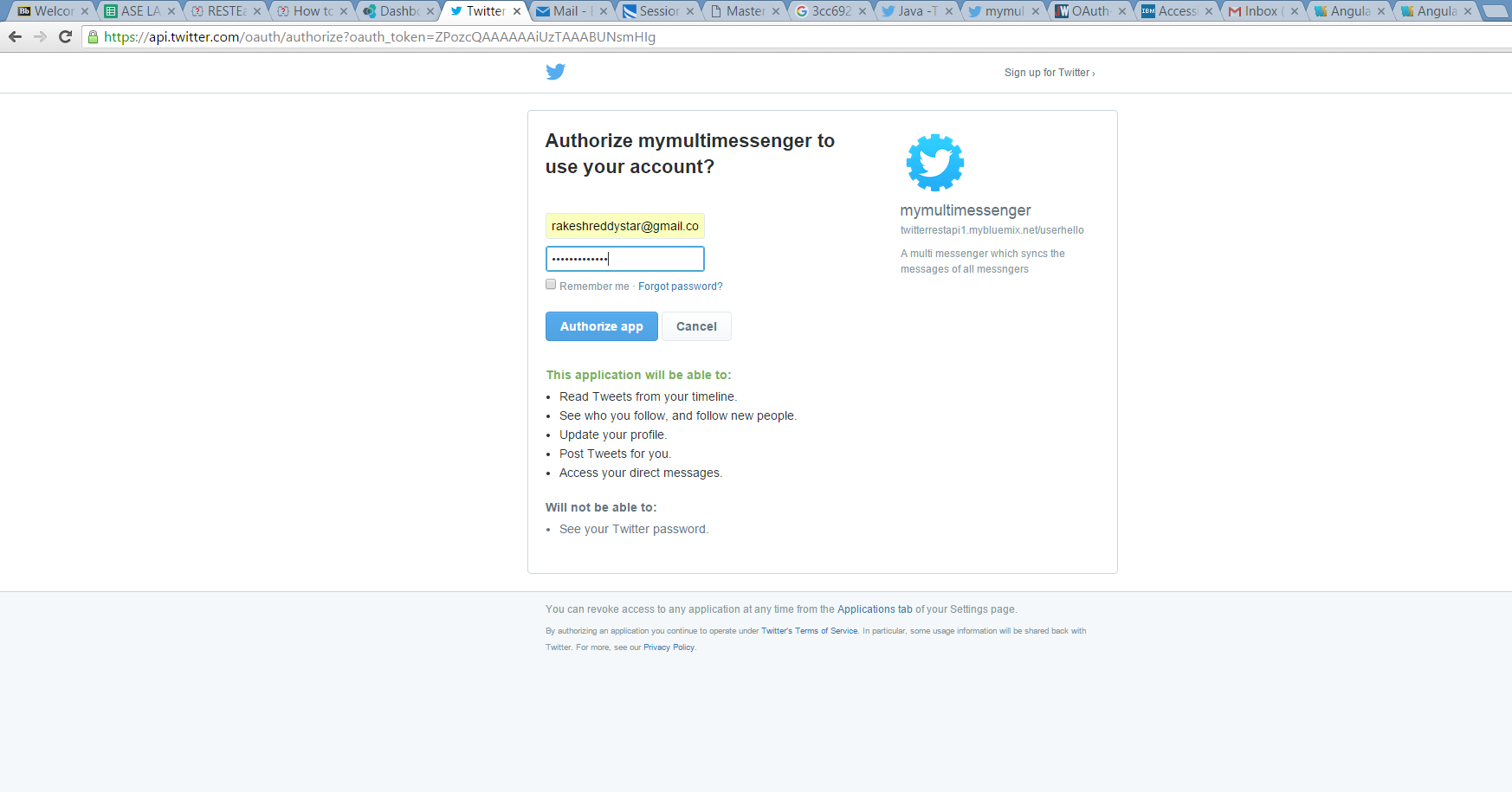
8.Chat Screen :



Description :

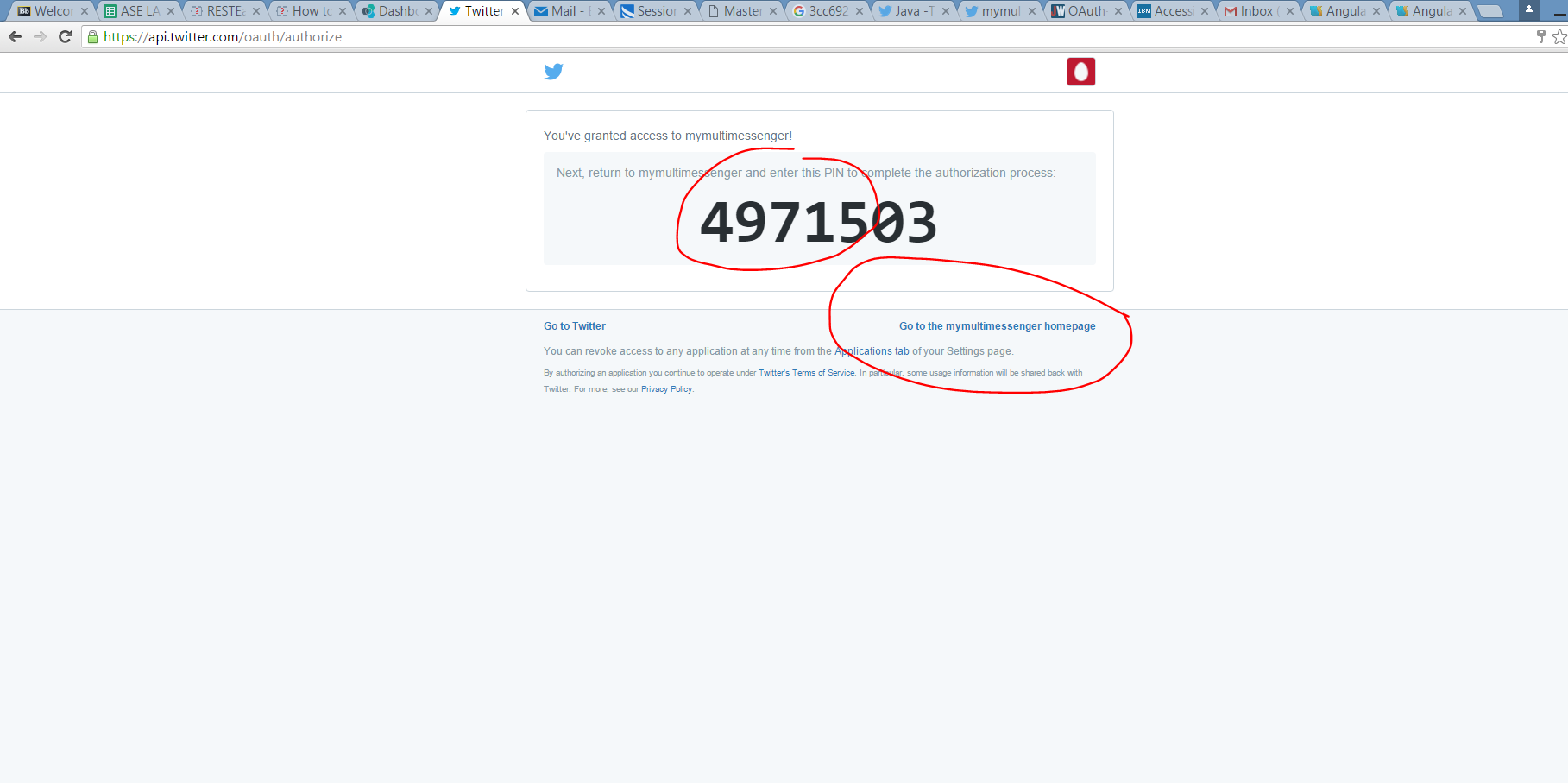
The above screen illustrates the chat screen where user can send and receive the messages from a specific friend.

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.

**Outlook Screen Shots :**



Fig: Home Screen

This Screen allows the user to sign in into outlook account. Here clicking Sign in button takes into outlook sign in page.

2 Outlook Sign In Page

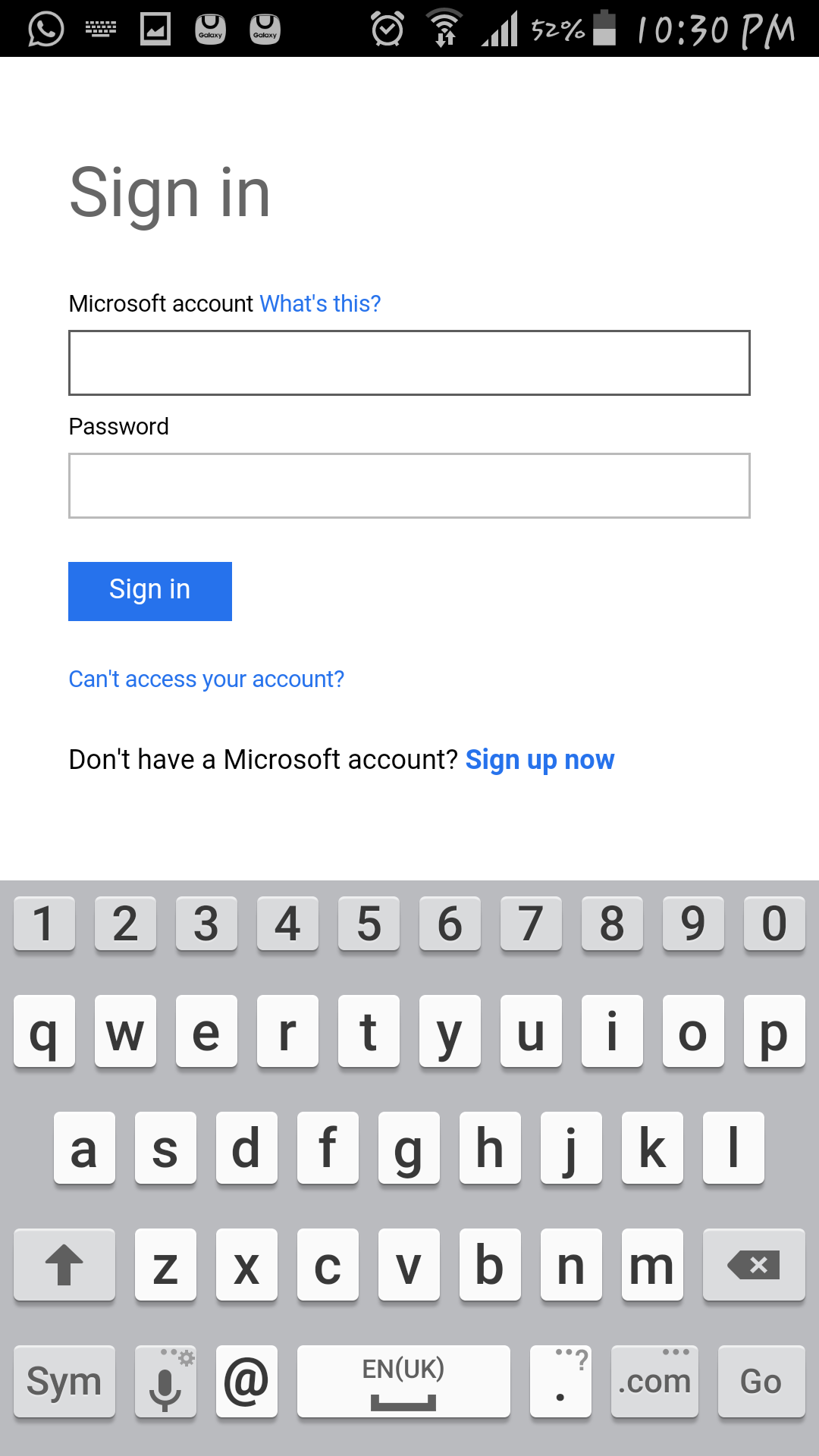
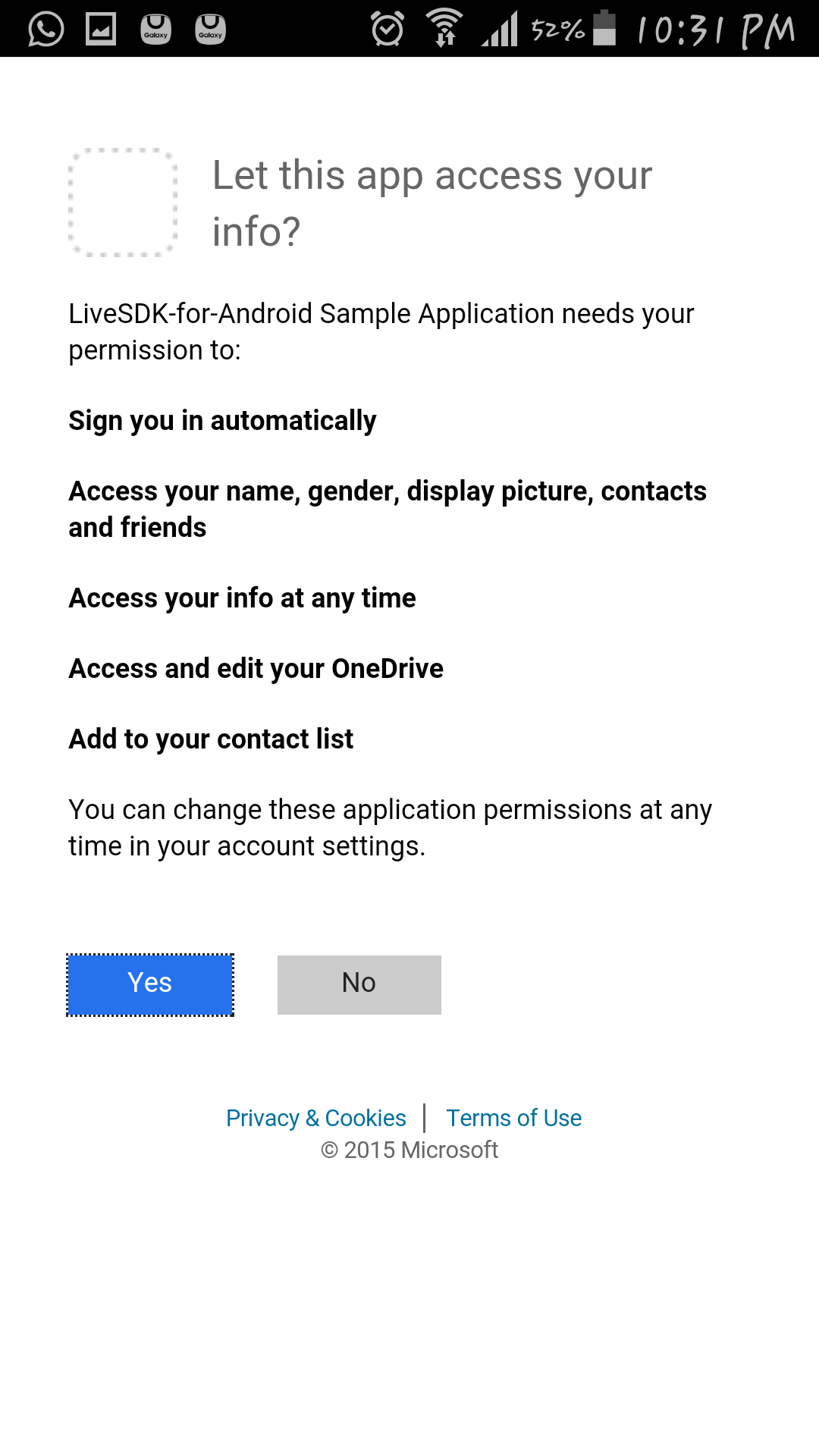


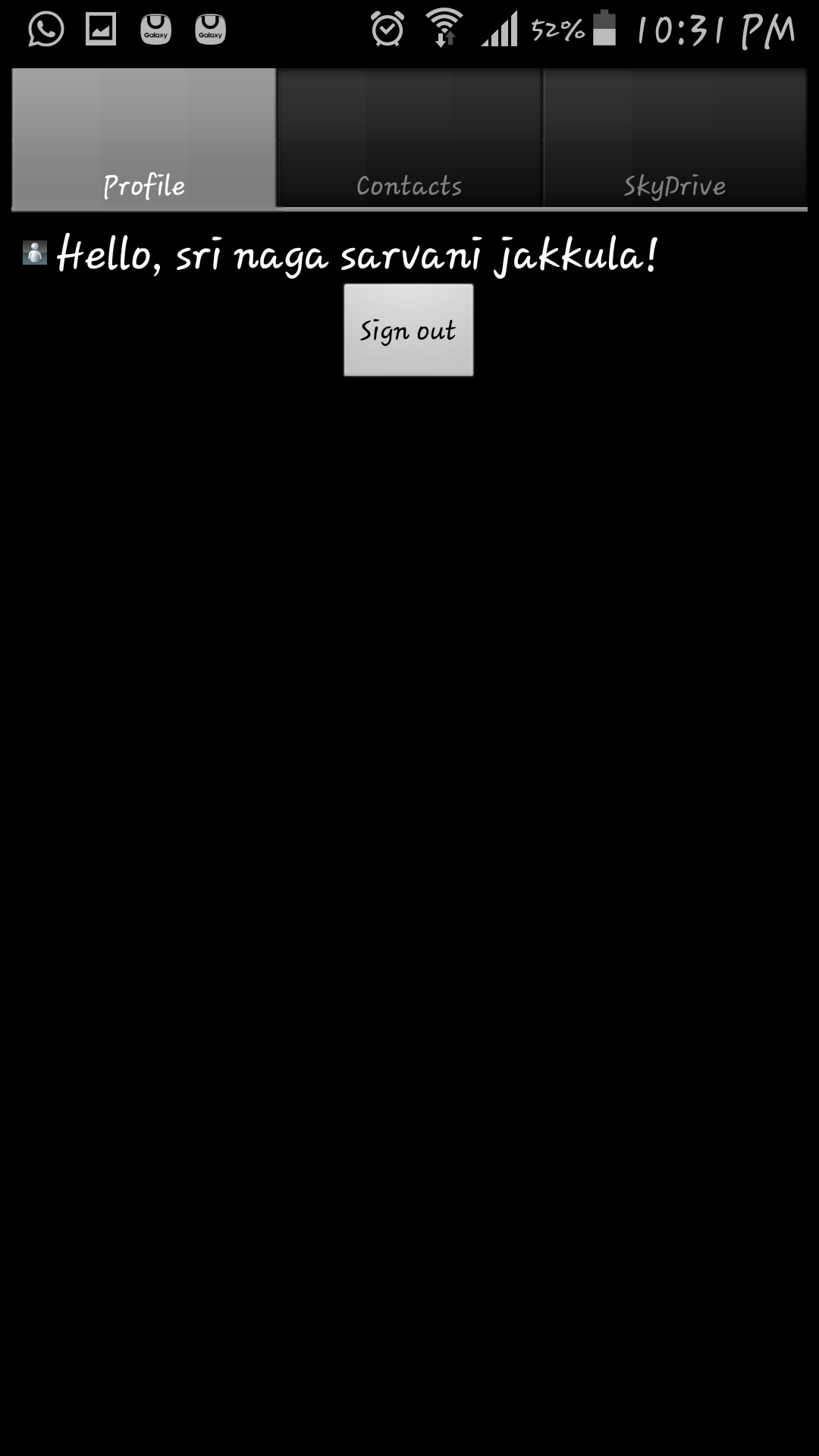
Fig: Outlook Signin screen

3 User Permission for access Page



This screen asks the user whether he is willing to share his profile information with the application.

1. Profile Page



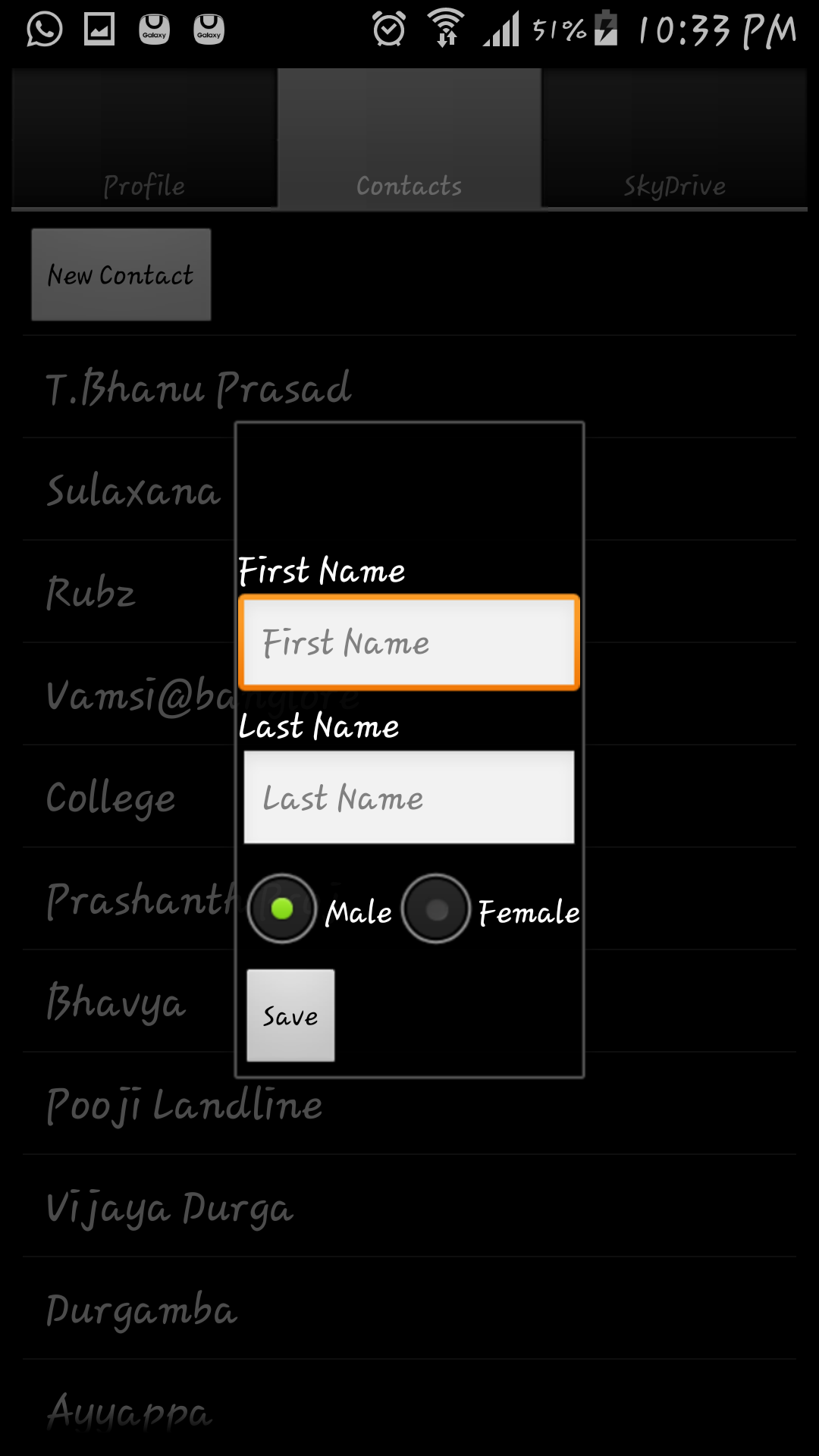
This page shows the profile data of the user after successful login into Outlook.com

6. Contacts Page



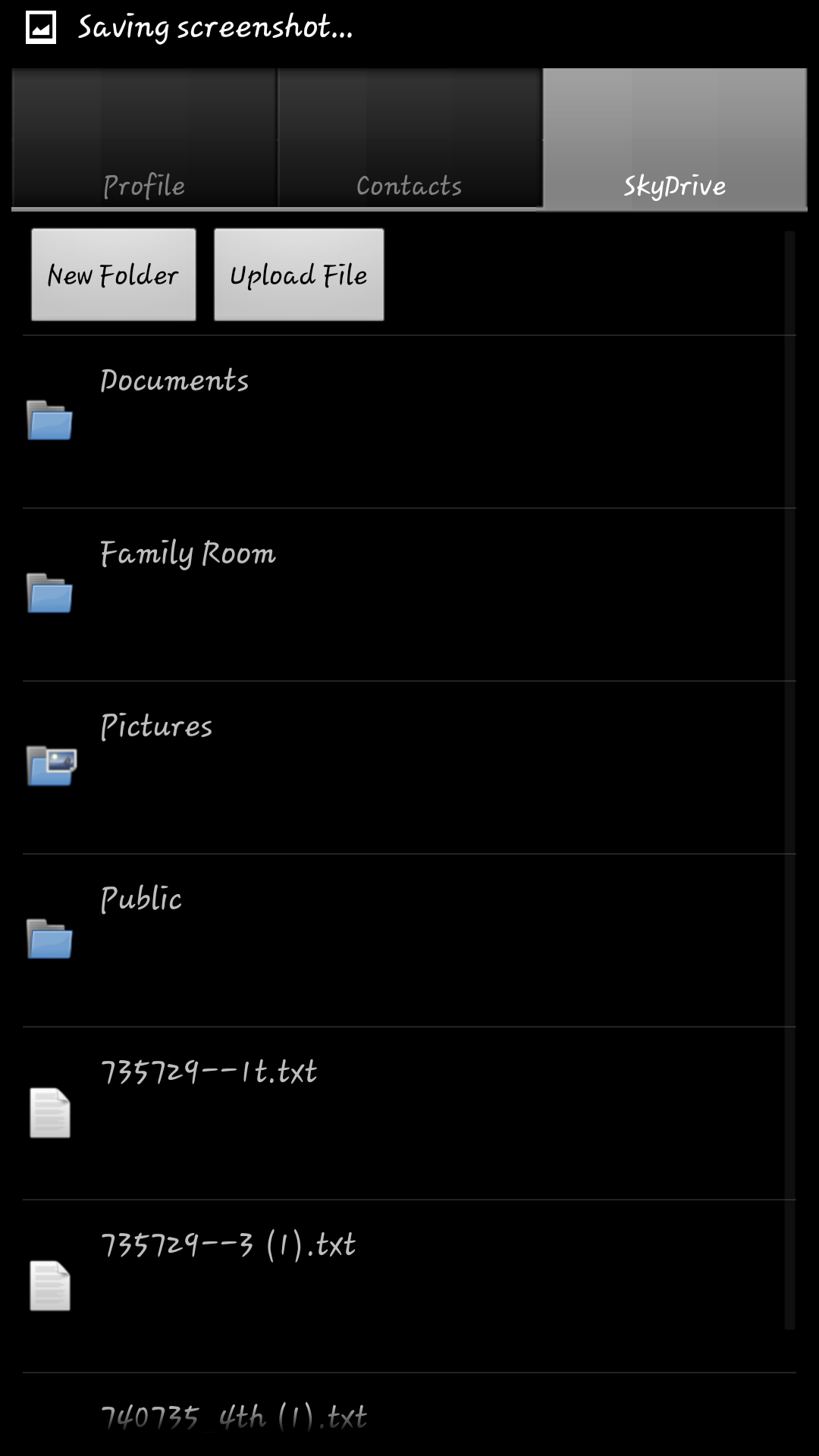
This page shows the synchronized contacts from outlook into the screen

1. New Contact Page



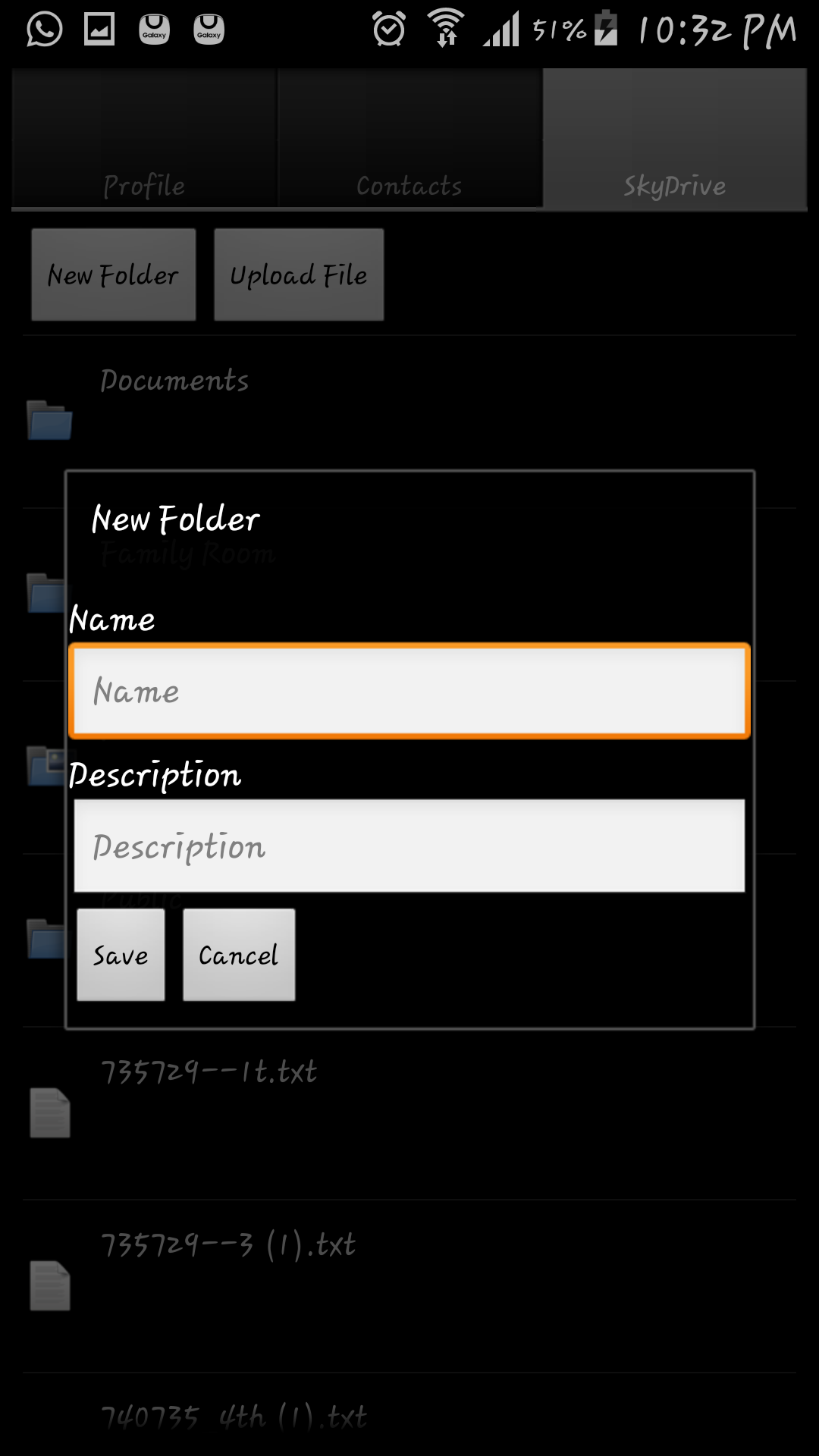
Here we can add a new contact to our outlook account.

1. Skydrive Page



This page gives the relative sky drive information from the Microsoft one drive location

1. Creating new folder in Sky Drive



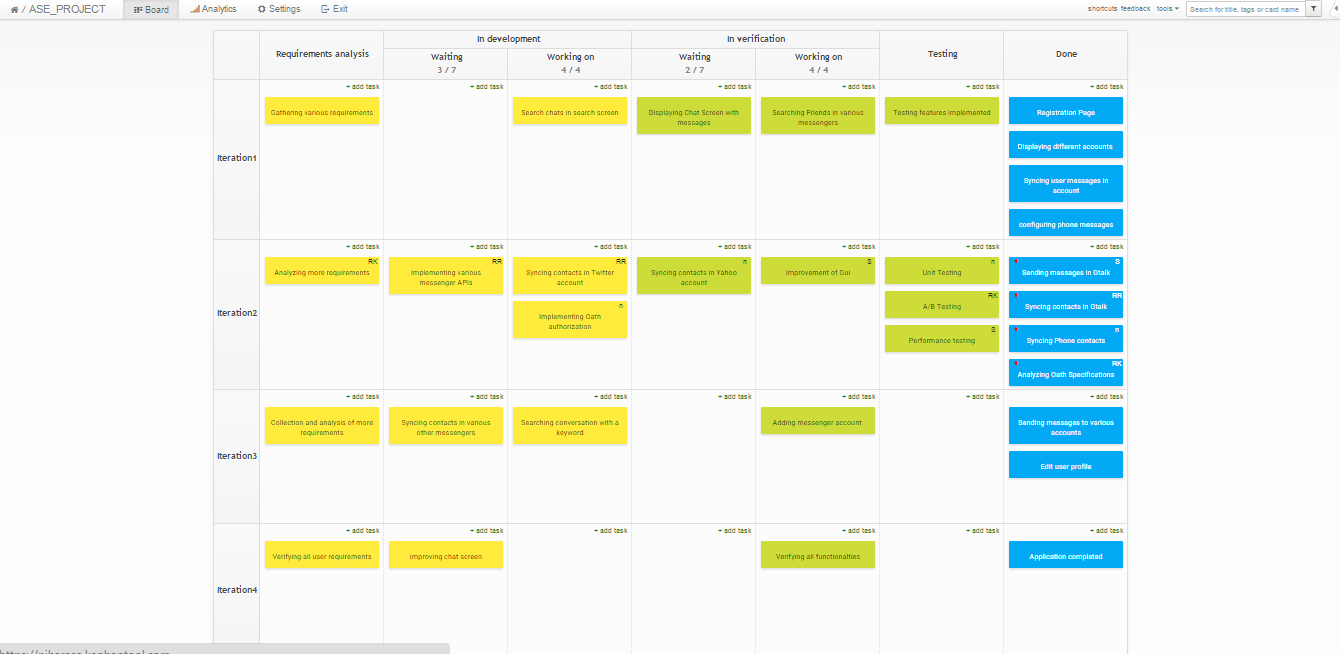
This page allows to create a new folder in one drive location.

## **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://niharase.kanbantool.com/

* Kanban tool iterations



The above figure explains about various iterations of the project.

As we can see, for the first increment Registration of a page and syncing phone messages are completed.

In the second increment, various requirements users may have are gathered and analysed. Later we have concentrated in fetching phone contacts and Gtalk API and we were successful in fetching contacts of the user phone and also we were able to send messages through our messenger to another Gtalk user using Gtalk API and XMPP connections.

In the third increment we mainly concentrated on integrating the whole project into a single module. We have been successful in integrating with some issue which we will try to resolve in the next increment. We have improved GUI with some more features which are available in ionic framework. We have stored and retrieved the details of the user from **MongoLab** and authentication is done by retrieving the details from the mongo lab, even the contact information is also stored in MongoLab and displayed to client. We were able to send and retrieve messages through twitter, gtalk and yahoo. From outlook we were able to fetch the contacts through the SDK which is available in developers section of outlook and we would look forward to integrate it as well in our next increment. The main feature of our project which is searching the messages and retrieving specific conversation details is also achieved. We will try increase the optimization of search in the next increment.

All the tasks are divided in a good proportion among the group members by assigning specific time and deadline to a task. Tasks which are in the development and verification stage could be completed in iteration 3.

* Implementation status report:

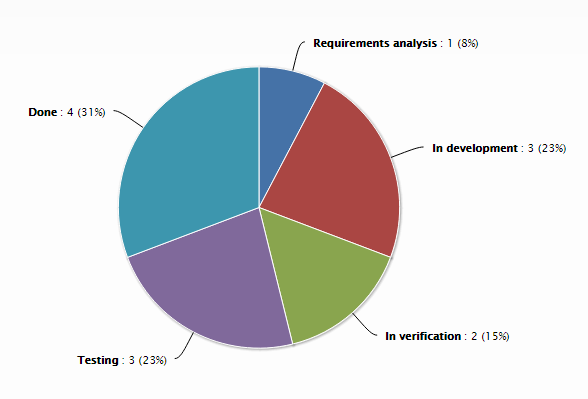
1. Work completed

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Member | Task Description | Member Responsibility | Contribution % | Time (hours) | Comments  /Issues |
| Rakesh | 1. Syncing contacts in Twitter.  2. Implementing various messenger API’s  3.Sending Messages to the contacts in twitter account  4. Integrating the project. | Develop, design, build and testing the task | 100 | 18 |  |
| Nihar | 1.Implementing Oath authorization  2.Syncing contacts in Yahoo API.  3. Sending messages in yahoo  4. Implementing search in messages.  4. Integrating the project. | Develop, design, build and testing the task | 100 | 20 |  |
| Sarvani | 1.Syncing contacts in outlook.  2. Improvement of GUI  3. Integrating the project | Develop, design, build and testing the task | 100 | 22 |  |
| Ravi KIran | 1. Sharing the data via facebook.  2.Analyzing Oath specifications  3.Authentication in facebook  4.Integrating the project | Develop, design, build and testing the task | 100 | 20 |  |

1. Work yet to be completed

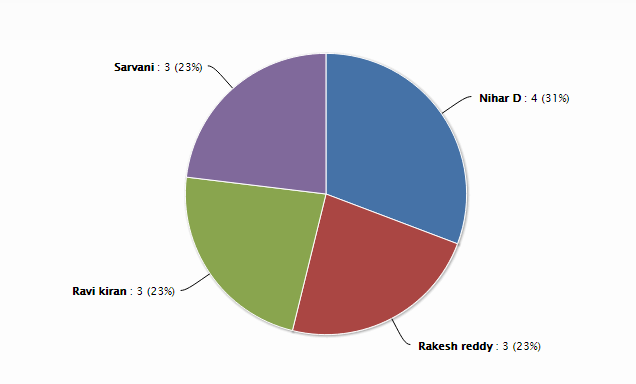
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Member | Task Description | Member Responsibility | Time (hours) | Comments  /Issues |
| Rakesh | 1. Integrating the whole project | Develop, design, build and testing the task | 20 |  |
| Nihar | 1. Integrating the whole project. | Develop, design, build and testing the task | 20 |  |
| Sarvani | 1. Integrating the whole project. | Develop, design, build and testing the task | 20 |  |
| Ravi KIran | 1. Integrating the project. | Develop, design, build and testing the task | 20 |  |

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



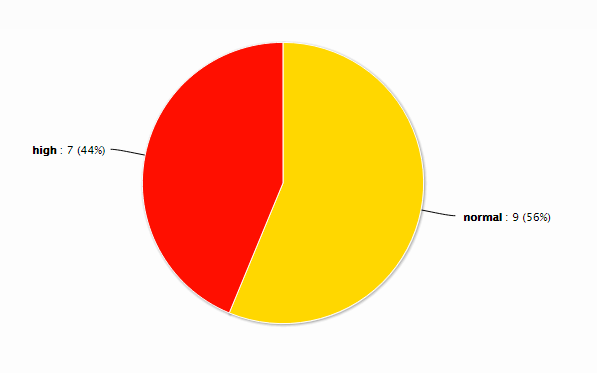
The above diagram explains about the various tasks position in various divisions available in Kanban tool

* + - Assignment division graph



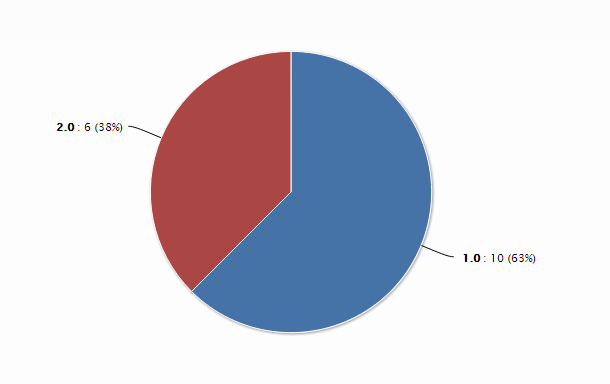
The above diagram describes about the amount of task difficulties faced by each group member.

* + - Priorities analysis graph



The above diagram explains about priority level given to various tasks.

* + - Task difficulty graph:



The above figure describes about difficulty level of various tasks in this specific increment.

## **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>