UNIVERSITY OF NORTH CAROLINA AT CHARLOTTE

Report on:

SocioBot (Mobile Application Development)

By:

Group 3C

Pranay Reddy Manda

Aravindharaj Rajendran

Jonathan Smith

Submitted to:

Dr. MOHAMED SHEHAB

Table of Contents

1. Login and Signup feature:

- User can login to app using his email address and password, which he provided while signing up with the application. User can also login using Facebook credentials or Twitter login credentials.
- **Email address and password login**: User will provide these details while signup and those details are saved in parse.com "User" table. When user enters login button, the application will check if the email address and password fields are empty or not. If empty, a message is displayed using Toast asking the user to enter all fields. If not empty, then Parse will validate the user details. If details are correct user will login to the application and the Messages activity will open. If User login details are incorrect then appropriate Toast message will be displayed.

- **Facebook login**: We used the **com.facebook.widget.LoginButton** class for adding the "Log In with Facebook" button. When the button is clicked, the Facebook login page or the Facebook application (if available on your android phone) is opened and it asks the User to login to Facebook. A Callback Manager is registered to Facebook to bring the control back to SocioBot. If Facebook login is successful then using ParseFacebookUtils, the User will login to SocioBot App.

  If user is logging in with Facebook for first time then we are using Graph API to retrieve the basic user information like name, email, profile picture and gender from Facebook and saving it onto parse User data. After saving the user data in parse the app will be redirected to Messages activity. If user is logging in for the next time then he will be directly redirected to the Messages page.

  For each action (success or login failure) an appropriate Toast message is displayed allowing the user to understand what is happening.

  Graph API is a low-level HTTP-based API that is the primary way to get data in and out of Facebook's platform. Using that we are retrieving the basic user information like name, email, profile picture, first name, last name and gender from Facebook and saving it onto parse User data.

- **Twitter login:** We customized a button to appear similar to the official Twitter login button and used it implement the Log in with Twitter feature. When the Twitter login button is pressed we are using ParseTwitterUtils to login to twitter and then to SocioBot. If login is successful then the user will move to next page and a Toast message indicating that login is successful will be displayed.

  If the user is logging in for first time then we are establishing a Http Secure (Https) connection to "**https://api.twitter.com/1.1/users/show.json?screen_name=screen_name"** and using OAuth to create a signed request to Twitter API for retrieving user information as JSON object. Then, the JSON object is parsed for email address, user name and profile picture, and this

data is stored in parse User table as user information. If the user log-ins for next time then he will be directly redirected to next activity.

OAuth is an open protocol to allow secure authorization in a simple and standard method from web, mobile and desktop applications. In our application OAuth was used to send secure authorized requests to the Twitter API.

- The images are stored as files in Parse Database.
- On the Login page, a link is provided for the users to signup.

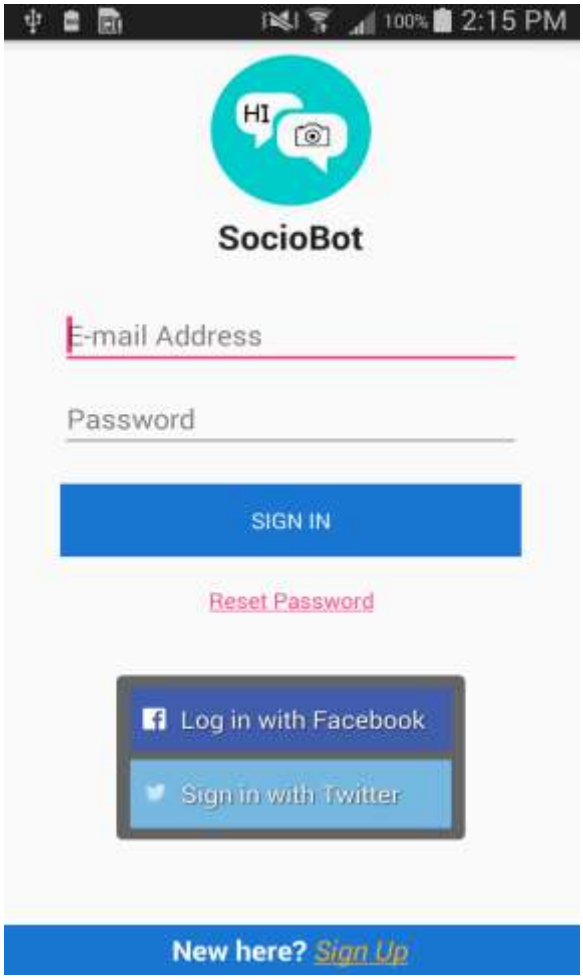


Figure 1.1 login page

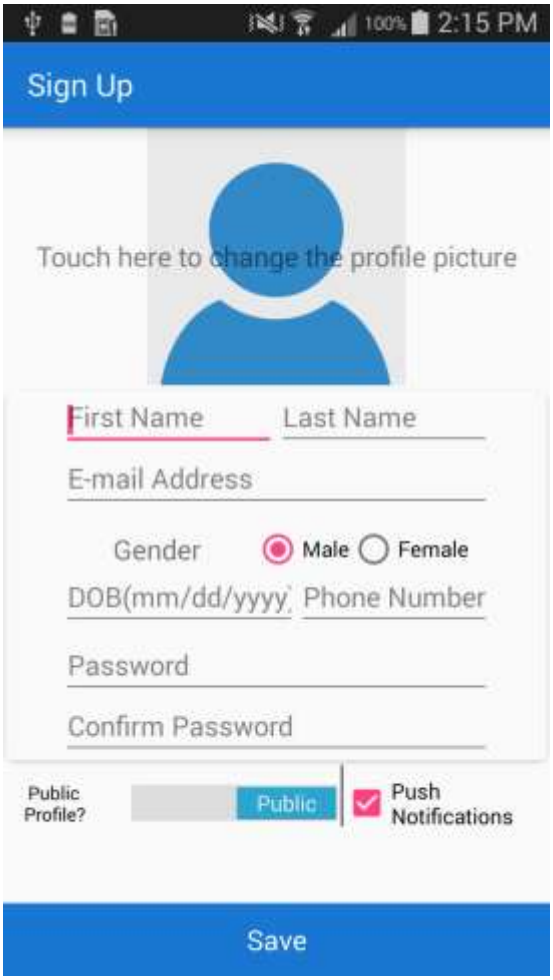


Figure 1.2 sign up page.

- **Reset Password option**: If you click on this link an alert dialog will appear asking for email address. If you enter an email address, which is registered with any user on Parse, then a password-reset link will be sent to that email address. If not account is registered with that email then a Toast message will appear asking for the correct email address.
- **Sign Up feature**: If user clicks on Sign Up link in log in page, he will be redirected to Sign up page.

1. In Sign up page all the fields are mandatory. If users skips any of the fields and enters save button then a Toast message will appear asking the user to enter all fields.
2. If user clicks on image avatar gallery will open and he/she can select an image as profile picture.
3. Password and confirm Password should match.
4. If user unchecks the Push notifications, then he will not receive any notifications.
5. After entering all the fields, if user enters save button only then the Parse query is created, all fields are saved to Parse database and login page will appear.

2. Profile Feature:

While Sign up, the User enters all the details like first name, last name, email address, gender, profile picture, date of birth, phone number, password, and others that are required to display in the user profile. Even if user logins through Facebook or Twitter some basic information will be retrieved and it will be displayed in "My Profile".

- In the Profile page user can view all these details and can also edit them.
- A user can view the profiles of all the other public profiles but he cannot edit other user profiles.
- User can control his privacy by altering between public and private at any time.
- User can chose to receive notifications or turn them off.

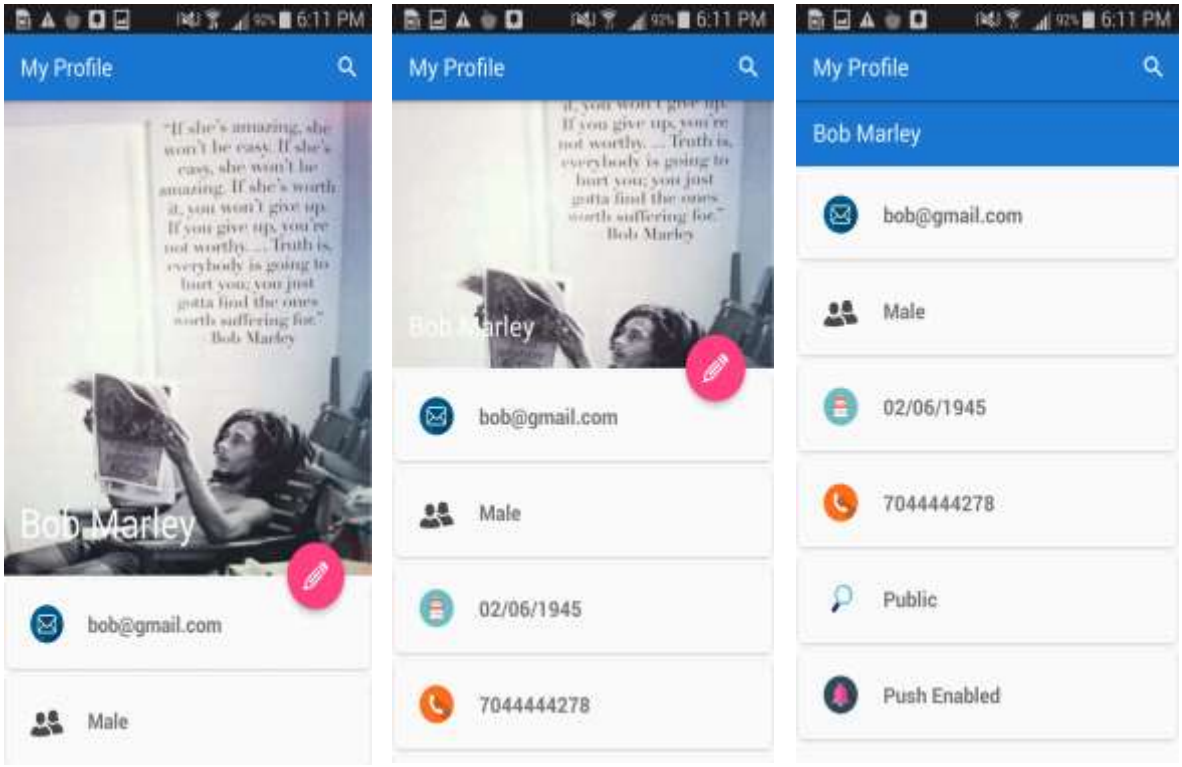
When a user logs-in successfully, message activity will appear. On the message activity, if he clicks on menu there will be an option called "My Profile" which will lead to the profile activity.

View Profile:

Profile activity at first will look like this:
There are few interesting feature to note here:

1. Collapsing Toolbar Layout for the profile picture.
2. Floating action button (fab) for edit the current user profile.
3. Search Manager in the menu to search for other users.

Collapsing Tool bar Layout:



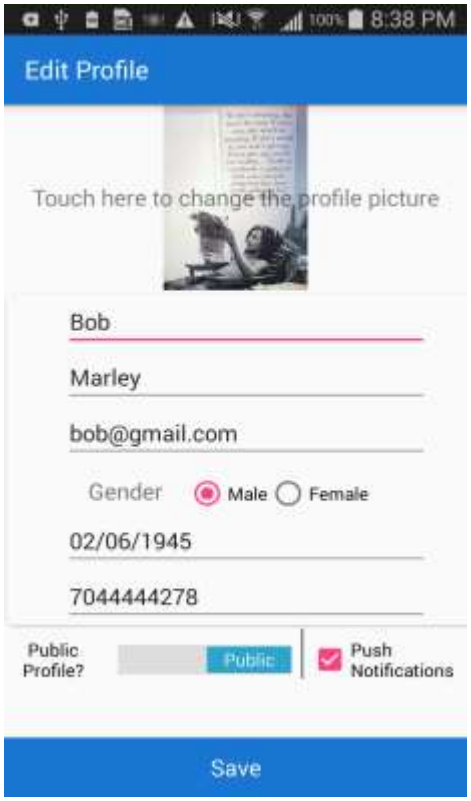
These three images will show you how the Collapsing Tool bar layout works.

- At first users complete profile picture is displayed and few other details like email address and gender are displayed (figure to left).
- When user touches anywhere and scrolls upwards the profile picture gets compressed gradually (as in second or center figure).
- While the profile picture gets compressed, other details will be visible from below.
- The profile picture gets compressed completely and finally only the user name will be displayed at the top and all other details below (as in right corner figure).
- If the user scrolls down the page then the size of the profile picture will increase and if he/she scrolls upwards then the profile picture will compress.
- More details about collapsing toolbar layout can be viewed here: http://developer.android.com/reference/android/support/design/widget/CollapsingToolbarLayout.html

Edit Profile:

User can edit his profile any time. He can also update any field based on his requirements. He can make his profile privacy public or private. He can choose to receive push notifications or disable push notifications.

To edit user's profile, he/she can click on the **floating action button (fab)** which is below the profile picture on his "My Profile" activity.
The fab appears to be attached to the profile picture and it will move along with profile picture.
It will disappear when the profile picture disappears.
When the user clicks on that floating action button, Edit Profile activity will open.



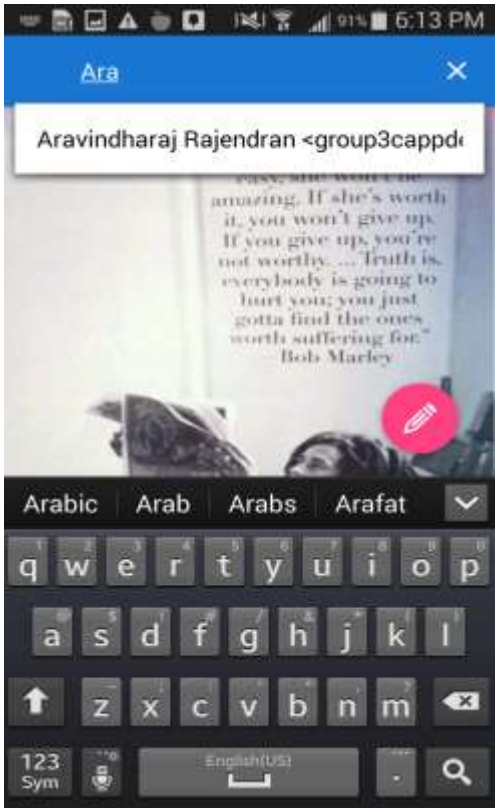
This is the edit profile activity.

- On the profile picture, there is a hint text telling the user that he can change the profile picture by clicking on it.
- User can change his profile picture and update any other field.
- If user leaves any field empty and clicks save button then a Toast message will appear asking him to enter all fields.
- By clicking on save button, all the changes will be updates to parse user and those changes will reflect in "My Profile" activity. The edit profile activity will finish and My Profile activity will appear again with updates.

User Listing:

- My Profile activity's action menu consists of **Search View** which is responsible for the User listing feature and viewing other users profile feature.
- If the user clicks on the search button on action menu, search view will start.
- As you can see in the image a hint is display which says "Search for other users".
- When user starts typing the names of other users, their names will be listed.
- Search view is **case-sensitive**.

- As the user starts typing in search menu, if the user name of any other user matches that substring then their user name along with email address will be displayed.
- For e.g.: Consider there are two users “James” and “John”. If user types ‘J’ both names will be listed. If user continue and types ‘Ja’ only “James” will be entered.



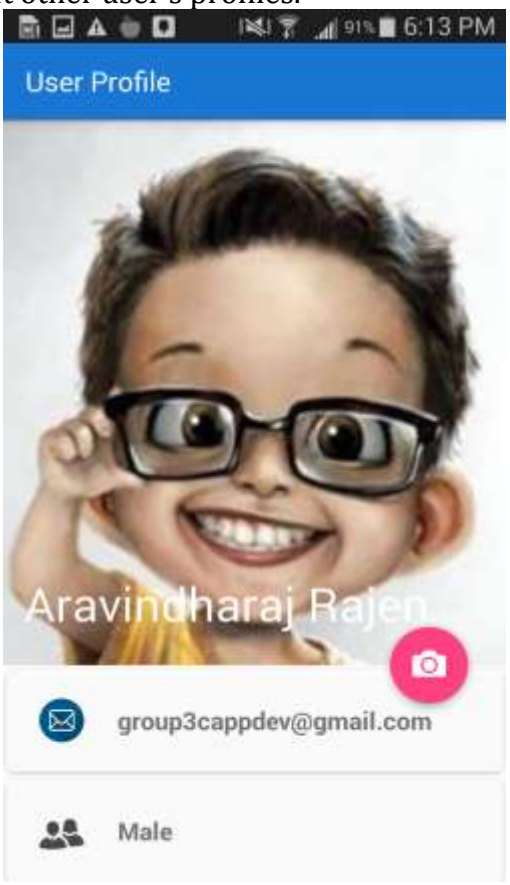
- Each time user enters a character, based on that input a new set of data is generated.
- That data is added as rows in **MatrixCursor**. It is a mutable cursor implementation backed by an array of Objects.
- MatrixCursor automatically expands internal capacity as needed.
- This MatrixCursor data is displayed using **SimpleCursorAdapter**.
- Simple cursor adapter is an easy adapter to map columns from a cursor to TextViews.
- This allows us to match the user selection with the respective user.

Listing all Users

- Only public user profiles will be displayed on user listing.
- If user types and deletes all characters in search menu then all the users will be listed.

View Other User Profiles:

If user clicks on any user in the user listing (SimpleCursorAdapter) then his profile will be displayed.
Current user cannot edit other user's profiles.



- When I clicked on "Aravindharaj Rajendran" in user's listing, his profile is displayed as shown here.
- Here, you can notice that the floating action button changed from edit image to camera image.
- That means that there is no option for current user to edit other user profile.
- If you click on camera floating button, user's public photos albums will be displayed.
- You can only view other user's profile but cannot edit it.

3.Album Feature:

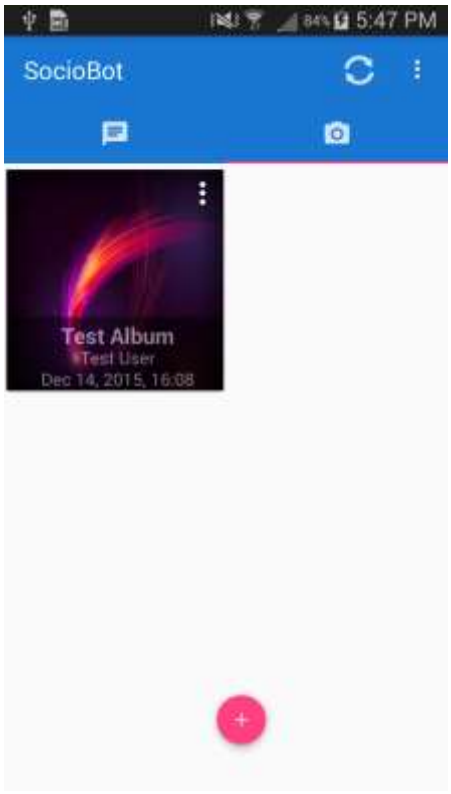


Figure 3.1: Album activity.

This is how album activity looks like:

- If the album contains any images then the first image is shown as album cover photo.

The main activity of the application contains a **Swipe View**. Using swipe view feature user can swipe between messages activity and albums activity.
An album activity displays:

1. Albums created by current user.
2. Public albums created by other users.
3. Other user's private albums to which the current user is invited.

Adding an album:

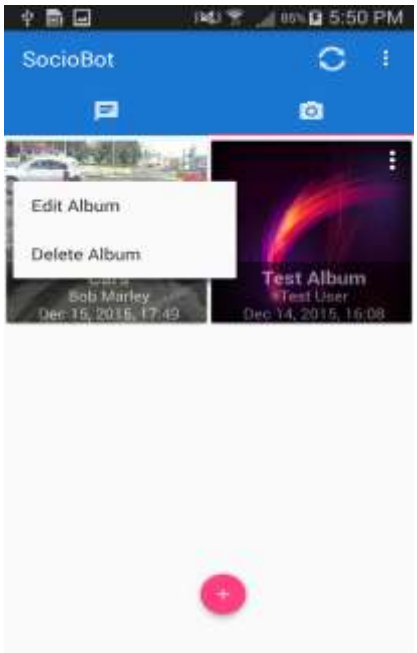
User can create a new album by clicking on the floating action button visible at the bottom center in the album activity.
When creating a new album, user has to add few basic details about the album like, album name, album description, album privacy – public/private and an initial image of that album.
If the privacy of album is set to private then the owner can select other public users who can view the album.
Public albums are visible to all the other users.
A private album is only viewable to users that are invited to view the album.
Below image shows how create album activity looks like.

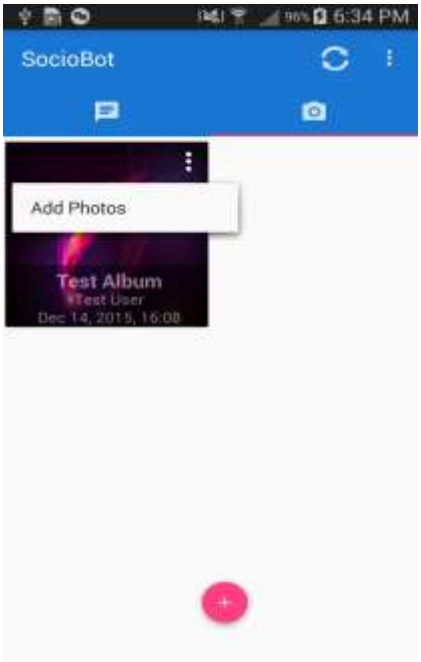
- By clicking on the (+) image button, album owner can add new photos to the album.
- Only the album owner can delete an image from the album by Long Clicking on the image.

Album listings:

Once user enters all details and clicks SAVE button album is created and the albums activity is displayed. For each album, on top right corner there is an action menu.

- If album owner clicks that button, “Edit album” and “Delete album” options are provided as shown in below left figure.
- If other users clicks that button, only the “Add Photos” option is provided as shown in below right figure.





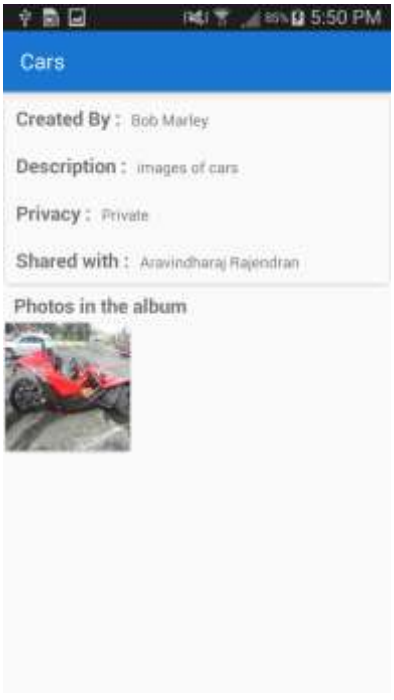
Edit Album:

As shown above, **only the owner of the album** can edit the album.

- ✓ Owner can add or delete any image in the album.
- ✓ Owner of the album can delete the whole album.
- ✓ Owner can change the privacy settings of the album.
- ✓ Owner can invite any other users to his private album.
- ✓ All the album information is managed on Parse.

View Album Details:

The albums are listed in a Grid view in album activity. On each tile, album cover image, album name, album owner and date of creating is shown.
If the user clicks on any album then the album details are displayed on a different activity.
So, a clear listing of all the details of an album are shown to the user.



If a user has a permission to a particular album then he can view all the images of that album.
User can also view each image individually.

Figure 3.7 Individual view of an image.

Other Users adding an image to album:

Other users who have permission to view an album can also add image to that album but before the image is shown in the album, it needs owner's permission.
Parse implementation:
In parse album table, each image has an "approved" flag.
If owner adds an image to album, the image is stored in images table, its "approved" flag is set to "yes" and its objectId is stored in album table.
If other user adds an image to an album:

- The image is saved in Parse database, and its approved flag is set to "no".
- There is one more table called "Approval". In this table the image object id, album object id, user object Id, album owner object id, and approval status are stored.
- A notification is sent to owner to approve the image.
- If user approves the image, its status changes to "approved" = "yes" in the "Images" and "Approval" tables.
- If the approved status changes to yes, then the image is visible in the album. If it's no then the image is not visible.
- If an image is rejected then it is deleted in the parse data store as well.

When an invited user adds an image to an album then this activity will appear.

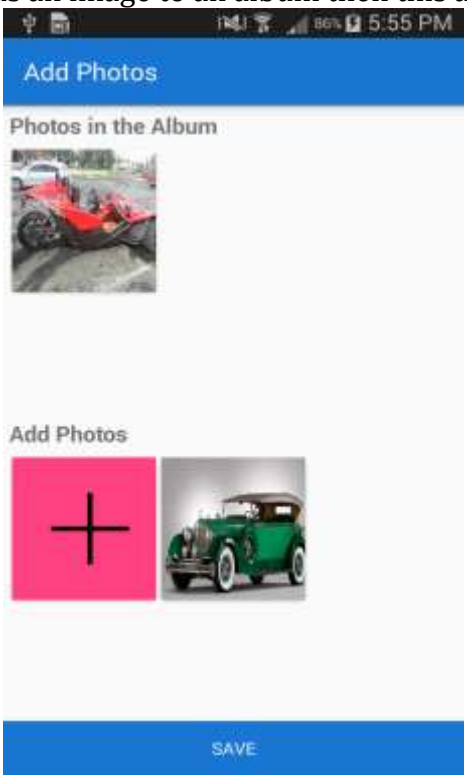


Figure: 3.8: Invited user adding an image to album.

- When an invited users adds image to an album, the owner of the album can view that request as shown here.
- Owner can either accept or reject the image request.
- Only if owner accepts the request then the image is added to the album.
- If he rejected then the image is deleted.

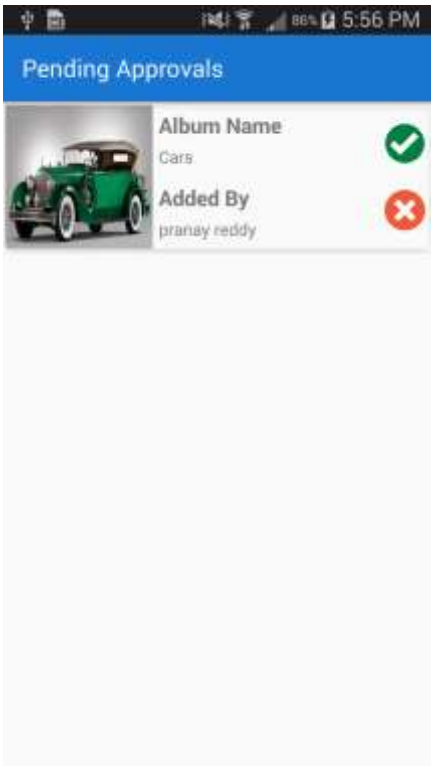


Figure 3.9: Owners pending approvals activity.

4.Messaging Feature:

When a user successfully logins, he will be redirected to the messages activity. Messages activity will display all the received messages in list view. The message will contain message text and message sender photo.

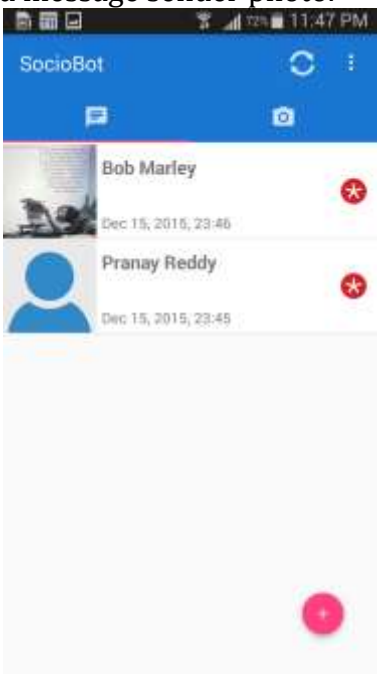


Figure 4.1: Message activity

If you click on any message you can read the message.

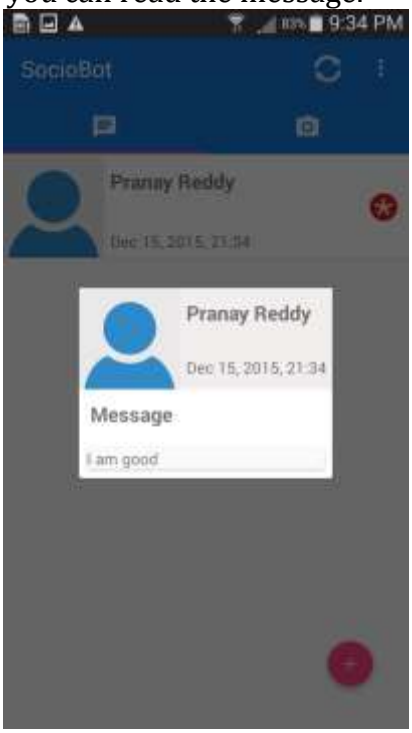


Figure 4.2: Reading a message

Compose a message:

User can send message to any other public user.

- User cannot send message to a private user
- And he cannot send message to himself.

Message activity has a floating action button similar to album activity. If a user clicks on it compose activity will start.
In compose activity user has to type his message and select the recipient.

Selecting the recipient:

If user starts typing the name of the recipient, suggestions will appear below based on user's entry. If user clicks on any of the suggestions, that recipient will be selected.

- **User can send messages to multiple users**

In the Parse database we are storing the sender objectId, message and array of recipient's objectIds.
We are creating an additional table called "Flag" to keep a track of which user is reading the message.

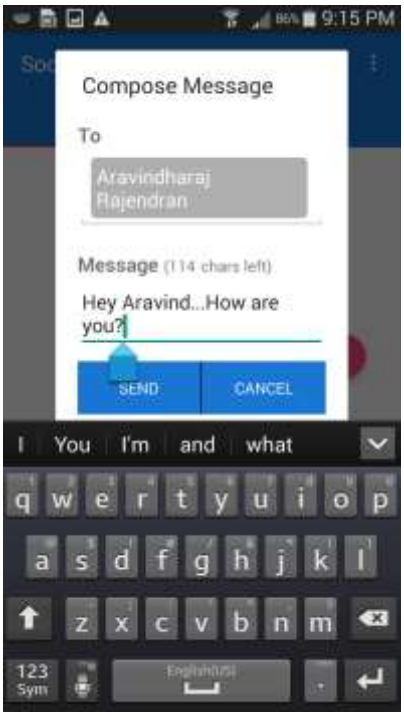


Figure 4.3: Compose a message

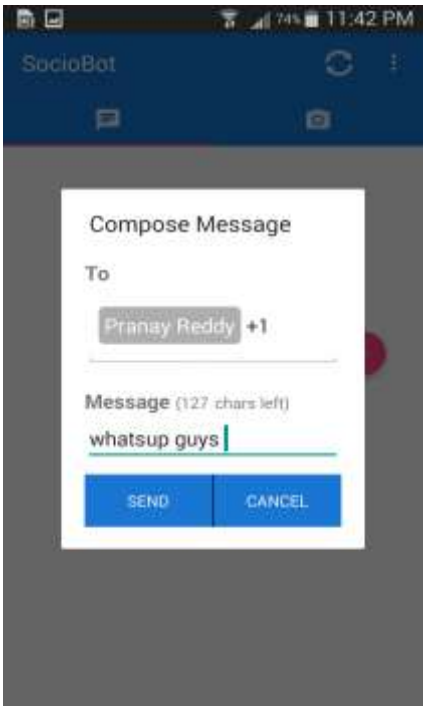


Figure 4.4: Sending message to multiple users

Reply and Delete Message:

To **Reply** swipe a message to right. To **Delete** swipe a message to left.
We implemented this using **"SwipeActionAdapter".** The SwipeActionAdapter is meant to wrap around the existing implementation of Adapter and provide callbacks that you can use to perform actions when someone swipes an item in the ListView.

Swipe Action Adapter implementation.

- All the received messages are displayed is a ListView. So the messages activity has a ListView.
- That ListView has a list view adapter to display each message as a row.
- And then the ListView is set to SwipeActionAdapter.

Swipe action adapter provides swipe action listener which can identify to which direction ListView items are being swiped.
So we can know to direction to which each message is swiped. In this way we implemented the reply and delete features.
For replying, swipe a message to right, enter the reply message and click send.
For deleting, swipe a message left and then confirm to delete.

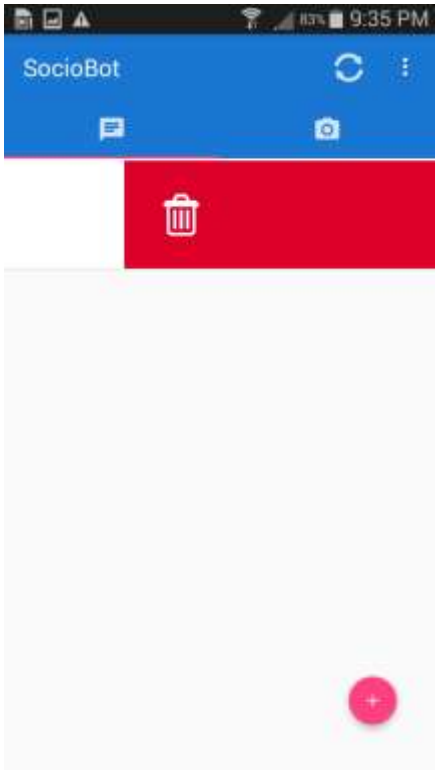


Figure 4.5: Delete a message

Figure 4.6: Reply to a message

Read and Unread messages:

If a message is unread then a red star will appear on that message. If the user clicks on the message and closes it then it is marked as read message.

In the image below, first message is read while second in unread.

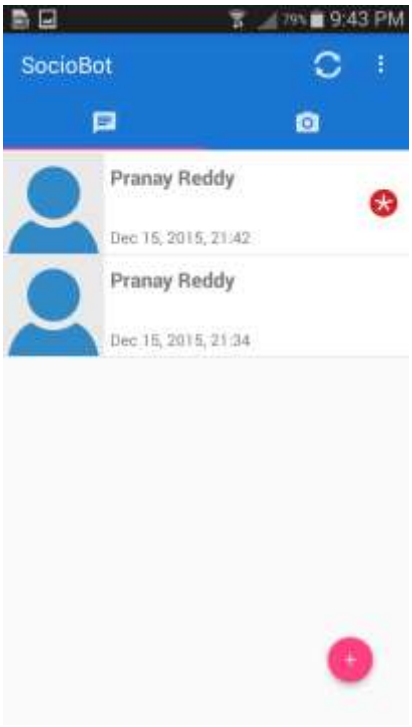


Figure 4.7: Read and Unread messages

5.Push Notifications Feature:

Every user has an option to control his push notification feature.
If a user disables his push notification then he will not receive any notifications. But when the user enables the push notifications he can receive them in these circumstances:

- ✓ When user A shares a photo album with user B, then user B will get a push notification alerting him of this event.
- ✓ When user A submits a photo in the album owned by user B, then user B will get a push notification alerting him to this event and requesting user B to verify the submitted photo. If the photo is verified by user B then user A will get a push notification.
- ✓ When user A sends a message to user B, then user B will get a push notification alerting him of this event.
- ✓ When a new user successfully registers to the app as a **public** user then all the users in the system will get a push notification alerting them to this event.
- ✓ When a new user successfully registers to the app as a **private** user and later if he changes his privacy to public, he is considered as new user and all the users in the system will get a push notification alerting them to this event.

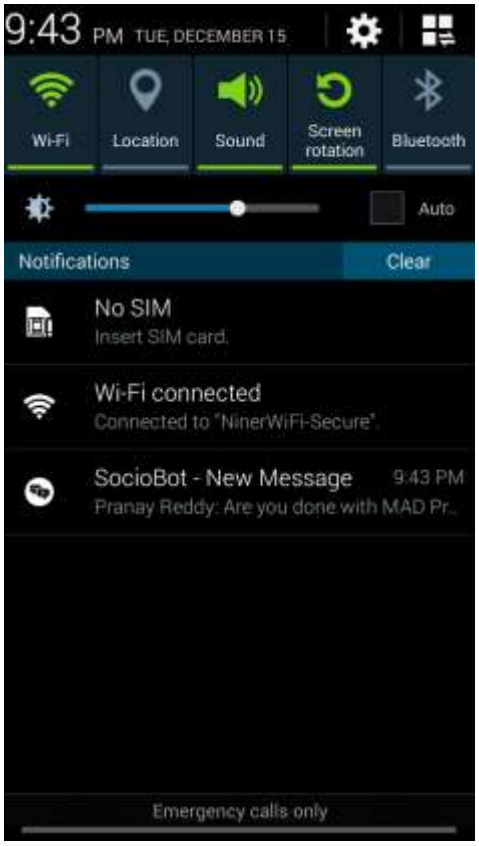


Figure5.1: Message notification

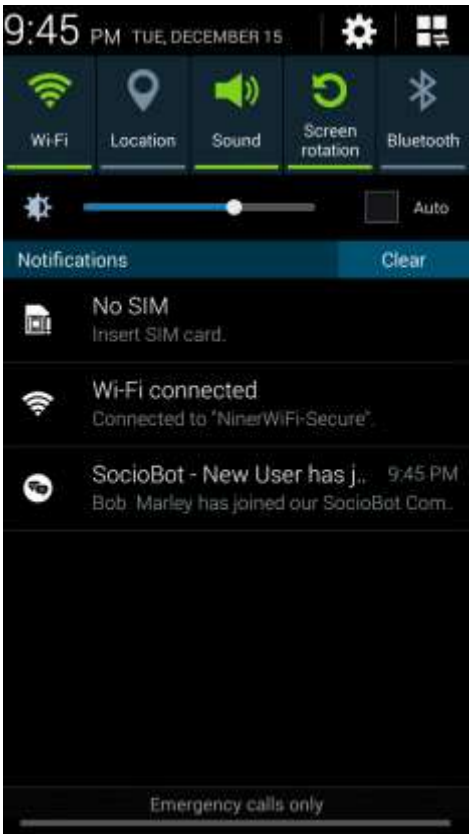


Figure 5.2: New User notification

6.Privacy Feature:

Privacy and Push notifications:

While signing-up to the application itself user can choose to be either public/private user and to enable/disable push notifications.
If users logins through Facebook or Twitter then by default he is considered as Private user and his push notifications are enabled.
User can updated this features in “My Profile” any time.

If user is Private: he will not be shown in user listings. That means other users cannot see him profile and cannot send him messages.
If user is Public: he will be shown in user listings. Other users can send him messages and they can view his profile.

If user push notifications are disabled: he cannot receive any push notifications.

Album Privacy:

User specifies the privacy of an album while creating a new one.

A public album can be viewed by every other user. If user creates a Private album then he can specify which other users can view that album.

Owner of the album can change the album privacy policy at any time.

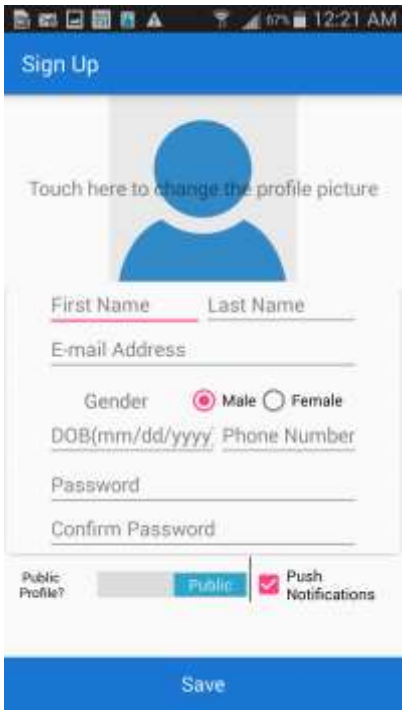


Figure 6.1: Privacy and Push notification settings while signup.

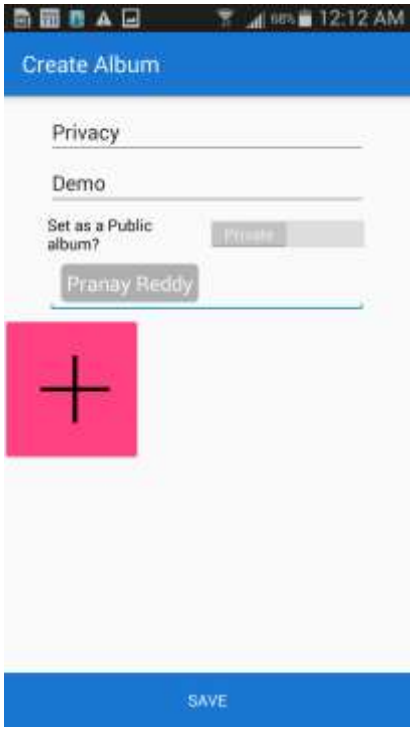


Figure 6.2: Album privacy settings.

7.Summary Table:

Requirements	*Implemented/Not Implemented*
1. Login and Sign Up Feature • Login using username & password • Login using Facebook • Login using Twitter	**Implemented as per the requirements**
2. Profile Feature • A User Profile has First name, Last Name, Gender, Privacy, and Profile avatar. • A User can edit his Profile to make changes to any field. • Display of a list of users, their profiles and their albums.	**Implemented as per the requirements**
3. Album Feature • Creating, Editing, Deleting and setting Privacy of an Album. • Implementing Public and Private Privacy Constraint of an Album. • Adding and Deleting photos to and from an album. • Listing of albums all public and private (Shared) in the album view. • Shared albums to be accessed by the receiver and are capable of being updated with new image data. (Upon owner's discretion) • Parse cloud for storing all the album and image data.	**Implemented as per the requirements**
4. Messaging Feature • Composing and Sending of messages to all public users in the system. • Messages to include both text and image fields. • Inbox feature for each user to access his received messages. • Reply and delete feature for each message. • A feature to distinguish read and unread messages in the inbox.	**Implemented as per the requirements**

5. Push Notification Feature • Push notifications when an album was shared between users. • Push notifications for images added into the shared album between receiver and sender for data verification. • Push notifications for each message when users try to interact via messaging feature in the system. • Push notification to all users notifying of a new user registration.	**Implemented as per the requirements**
6. Privacy • Users can set privacy (public/private) to an album at a time when it's created/ edited. • Users can set privacy to their profiles. • Users can toggle Push notifications to On/Off.	**All the features under "Privacy" are implemented and tested to work as per the requirement.**