

# **CMPE 277 Term Project**

## **Spring 2017**

**Project Title: ConnActivity – An Android App**

### **Project Team Members:**

**By:**  
**Ekta Sorathia**  
**Gaurav Chodwadia**  
**Keyur Golani**  
**Rushin Naik**

**Demo URL:**  
<https://youtu.be/KMV5xSXoe7Q>

## Table of Contents

Motivation and Introduction .....	3
High Level Design .....	4
Component Level Design .....	5
Technology Choices .....	6
Presentation Tier.....	6
Middleware Tier(Network) .....	6
Data Tier.....	6
Server Tier .....	7
Description of features .....	9
Android App Intro Module.....	9
User Email Authentication .....	9
Private Messaging .....	11
Timeline and Post.....	11
Notifications.....	12
Profile Settings and Update .....	12
Friends and Followers .....	13
Session Management using Shared Preferences .....	14
Testing Plan Executed with Results .....	15
Learning .....	16
Future Work .....	16
Theme Color Palette .....	16
Album .....	16
Push Notifications .....	16

## Motivation and Introduction

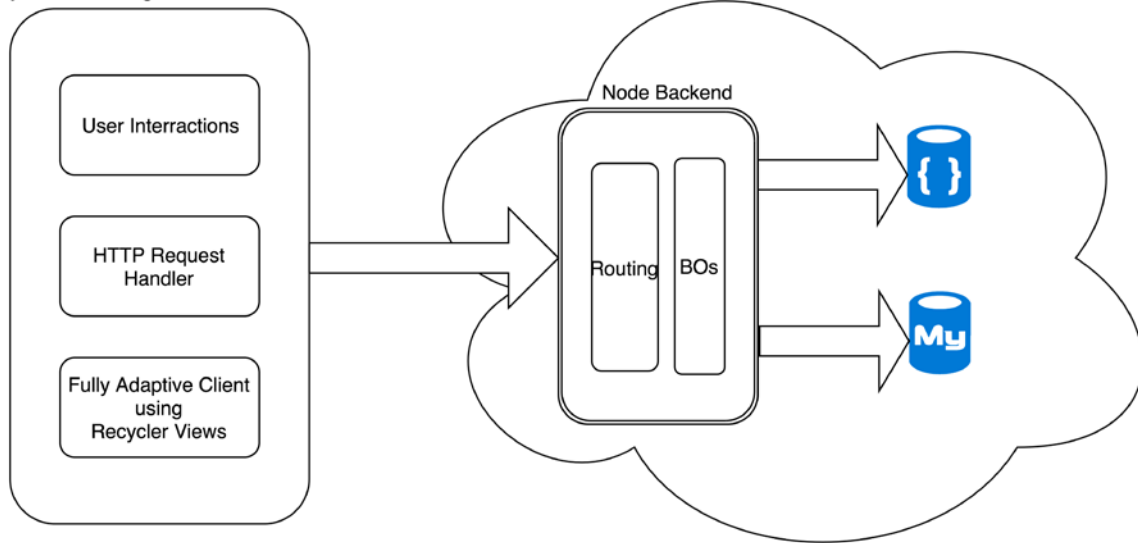
The motivation of creating this is application is to create interactive social media and provide users with a fun experience interacting with each other. Following are the core features of our application:

- ConnActivity is a social media application.
- Create account with your email ID and you're done.
- You can upload pictures, share photos, build profiles and more...!
- Also, find friends with their screen names and email to add as friend or follow.
- Send messages, follow updates of a friend and get notified of their activities.

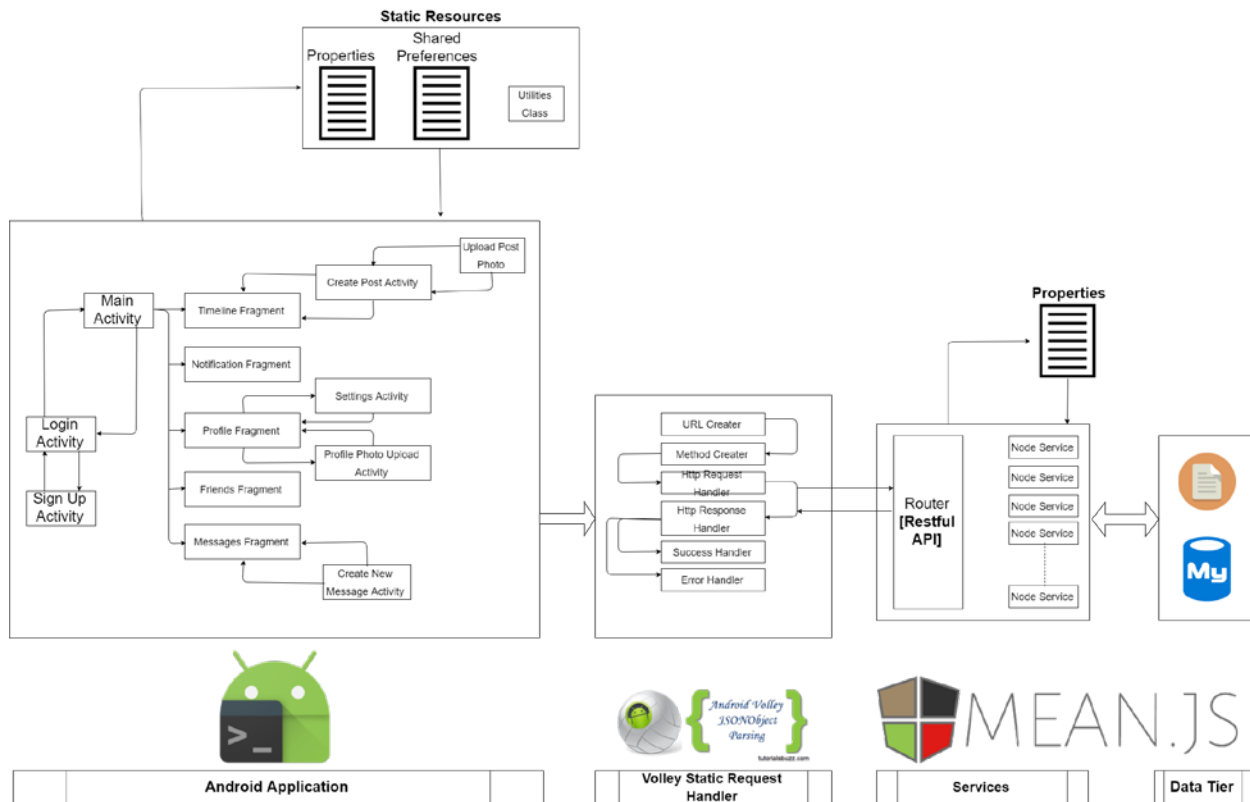
Demo Video URL: <https://youtu.be/KMV5xSXoe7Q>

## High Level Design

Fully Material Design Android Thin Client



## Component Level Design

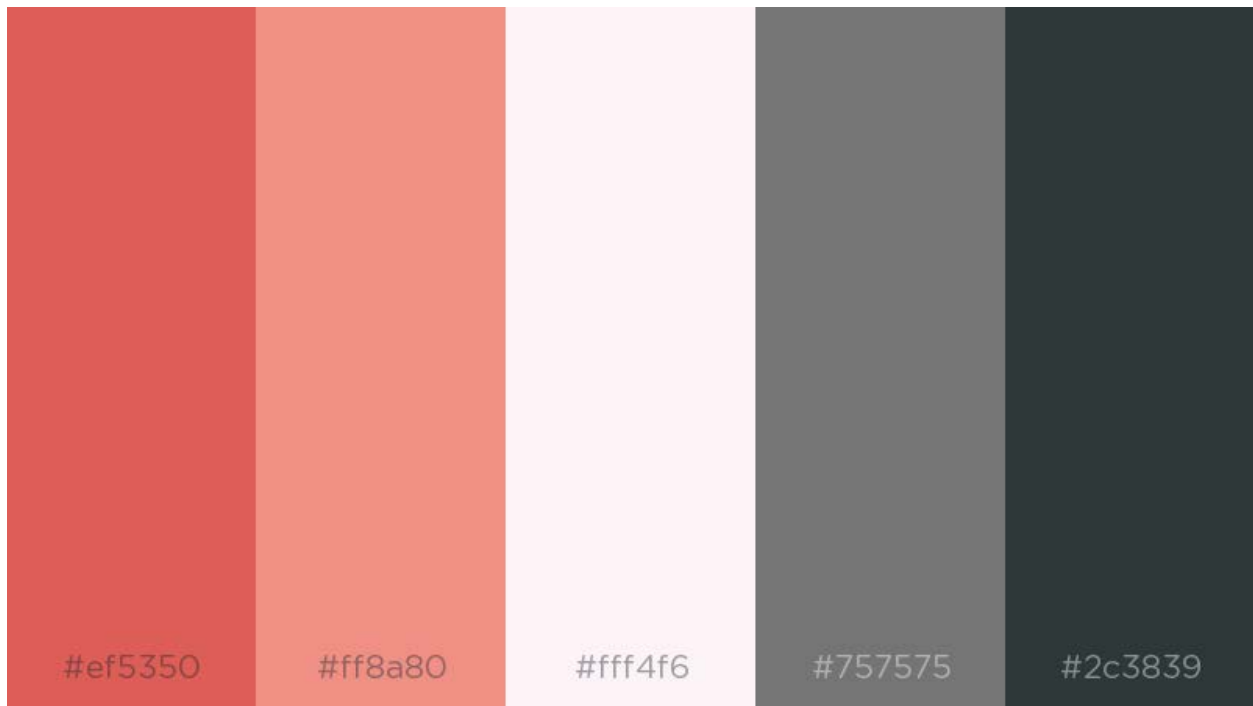


## Technology Choices

### Presentation Tier

We have used android fragments and activities for the UI implementation. RecyclerViews and Adapters are also core component of our UI.

Following is the colour palette for our app.



### Middleware Tier(Network)

The use of Volley was inspired from a comparison of a couple of APIs and the features that they provided for the HTTP communication between Android App and a backend server. Volley is intuitive to use and effective. Developed by Google Employees in house, which makes it compatible and reliable product for Android integration.

To use Volley with the application, we developed a RequestHandler class where we developed our own method to handle HTTP requests being made. This class contains methods to get base URL from properties file, create requesting method and execute an HTTP request using the passed parameters. This class provides the static method for request making and accepts an instance of an interface called ResponseHandler which needs to implement handleSuccess and handleError functions in order to handle the responses from the HTTP requests made. This makes the system work asynchronously and populates the fetched data asynchronously in non-blocking way.

### Data Tier

1. MySQL and Google Cloud Platform

MySQL is used to store the logical relationships between different entities. It related different entities using foreign keys and thus simplifies the query logic. The full data set is stored in the following tables:

- following tables:
- account\_details
- profile\_details
- notification\_details
- message\_details
- connection\_details
- follow\_details
- interest\_details
- post\_details
- preference\_details

The same data can also be implemented on the cloud using Google Cloud Platform. It provides a SQL dashboard to interact with it and thus helps to keep the SQL data online and easily accessible.

## 2. MongoDB and mLab

It is a free and open-source cross-platform document-oriented database program. It stores data in the JSON objects and it is considered to be one of the nonrelational databases. We used this document based data structure to store heavy data like images which were easily intractable in the form of key value pair.

The same database can be hosted on cloud for global access. We also implemented the cloud method using '**mLab**'. mLab is a cloud database service that hosts MongoDB databases and is easily managed.

## Server Tier

Server tier consists of a highly distributed and highly robust server application developed in Node.js and its web routing module Express.js. Our backend has routing logic in a file and different dedicated services running integrally which are responsible for different elements of the system. The Account Business Object is responsible for handling account related business logic. The Profile BO deals with the profile related logic, Timeline BO deals with timeline logic and so on. The routing logic has none but one responsibility which is to route the request to appropriate Business Object. The relevant BO takes care of the business logic to be performed.

On the next stage of modularity, the BOs take care of integrating the Business Logic together and the responsibility to connect to the DB and interact with it for fetching and saving data is taken care by the DAO object. The DAO object creates a connection pool of database connections which it uses one by one and waits for a free connection upon need while interacting with the database. Here, the DAO object exposes the methods to talk to the database for the BOs through which it can be used and managed modularly.

For ease of use and code reusability we have also created global variables and functions into the application that can be used anywhere in the project just by the name. These functions would be the functions to check if a variable exists or not, in other words if it is null, undefined, 0, "" or {} in JavaScript, it would be a non-existent value. Same way the regex validation patterns for email

and other values are also saved in this module. This module allows the ease of use and code reusability to the developer.

**Security:**

We used Bcrypt module for Node.js to hash the credentials stored in the database where we are using 10 fold salt generation to give the maximum security to the credentials stored. We have used Nodemailer API to send the emails to Email ID of any user about the notifications, messages, new posts and account verification codes which is also defined into the Utilities module as a global function to be used generically.



## Description of features

### Android App Intro Module

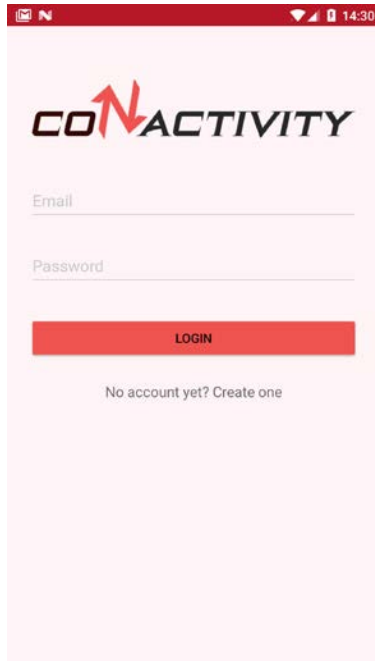
The app intro feature is very unique feature that we have implemented. It tells the user about our application. It provides a small and clear note about the salient feature of the application. The user can either skip or can click on the next arrow to view the next note and so on.

This feature is only shown to the first-time user and so it doesn't bother any returning who already knows about the app.



### User Email Authentication

The feature deals with the account creation and verification module of the registration. When the user clicks on the launcher icon of the our application, the user is directly guided towards to Login Activity if the user has not logged in. If the user has logged in already, then directly the timeline view of is opened.



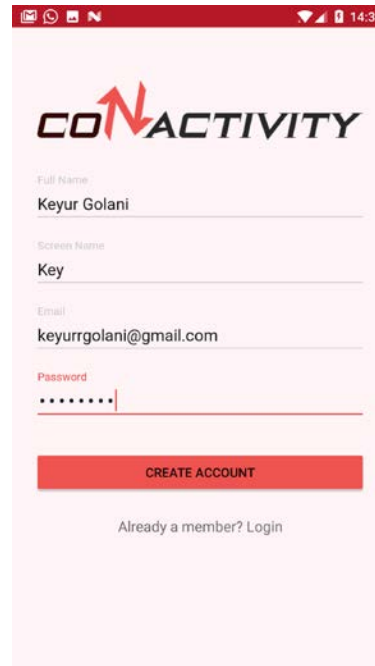
ConnActivity

Email

Password

LOGIN

No account yet? Create one



ConnActivity

Full Name  
Keyur Golani

Screen Name  
Key

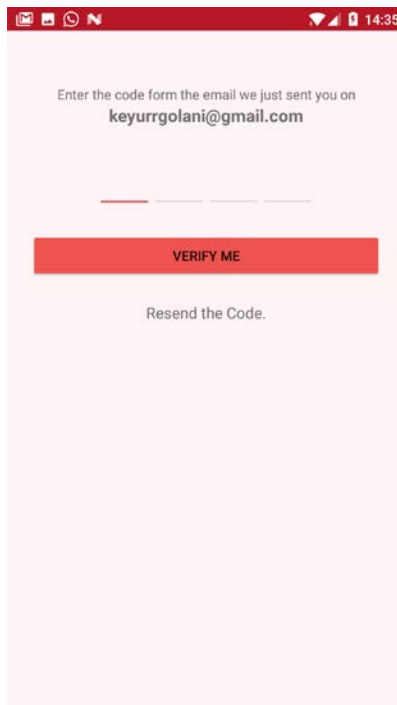
Email  
keyurrgolani@gmail.com

Password  
.....

CREATE ACCOUNT

Already a member? Login

The email verification is required to verify that the user has a genuine email id. This is implemented using the 4-digit 'Verification Code'. Once, the user has filled all the details in the sign up activity, the user will get a verification code on in an email. The code when entered on your app, will verify the account.

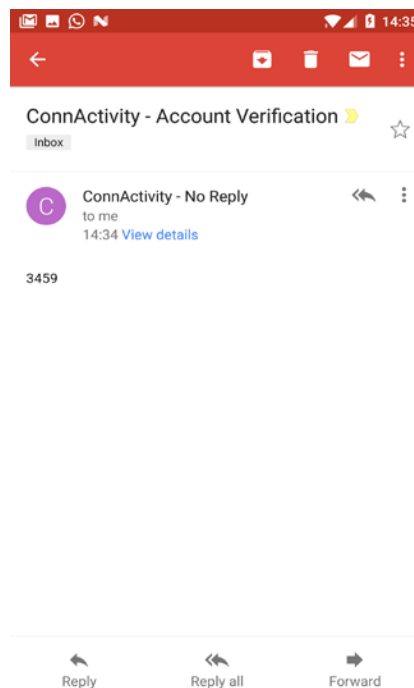


Enter the code form the email we just sent you on  
keyurrgolani@gmail.com

.....

VERIFY ME

Resend the Code.



ConnActivity - Account Verification

Inbox

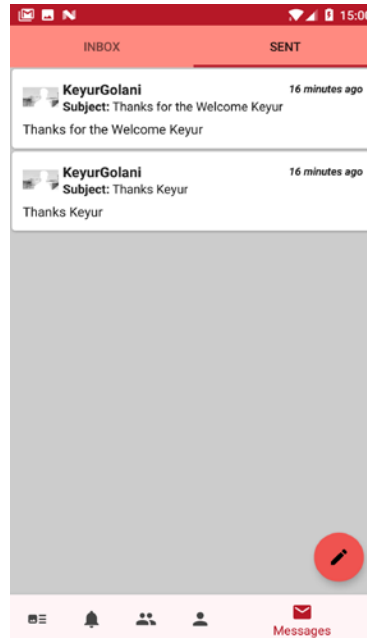
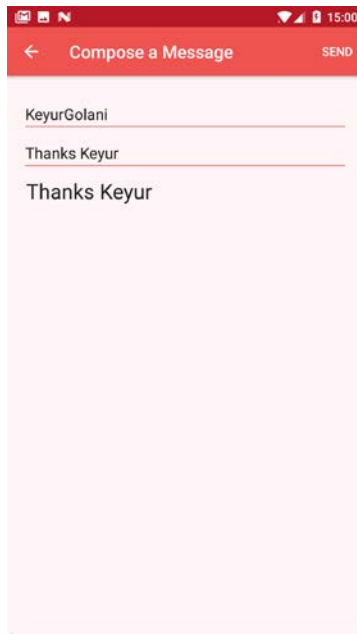
ConnActivity - No Reply  
to me  
14:34 [View details](#)

3459

Reply Reply all Forward

## Private Messaging

The messaging feature works using the RestFul API calls. The user can send email like messages using the Screen Name and Email Id of the friend. The following images shows the UI screen for the private messaging implementation. It is very interactive and easy to understand.



## Timeline and Post

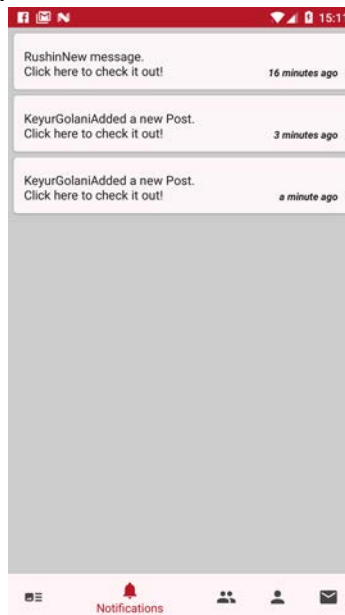
The timeline of user is populated with the post of the friends and following profiles. The user can also write post of any sizes and post them.

Also, user can add images to a post using Camera or Gallery. These images would be easily shown on the timeline as shown in the figure



## Notifications

The notification are the alert like entities of the project that tell the user about any post that a friend or a following person has made. The user can view those notification in the 'Notification Tab', which can be selected from Bottom Navigation Bar. The pictures shows the content that will be loaded in the notification fragment.

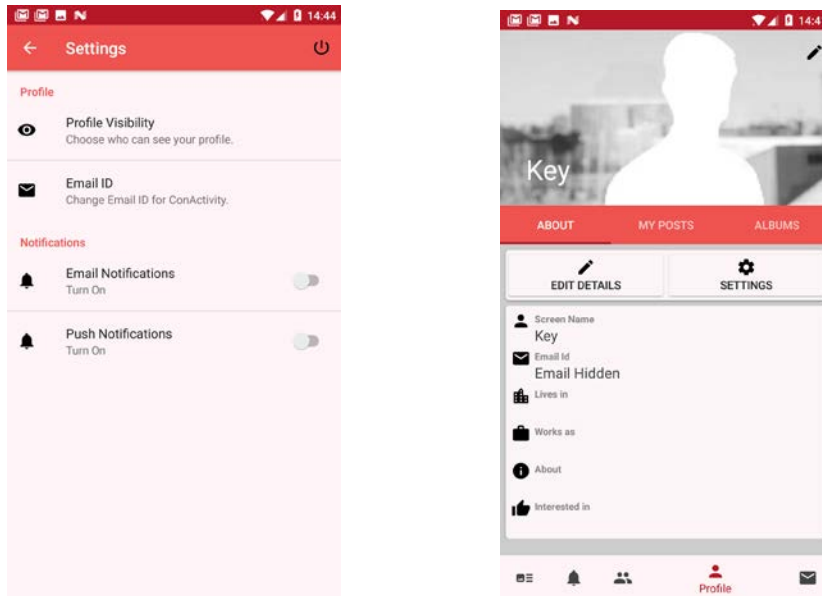


## Profile Settings and Update

This feature takes care of preference of the user. The user can select preferences related to the following two properties:

1. Visibility of the Profile (Public/Private)
2. Email Notification (On/Off)

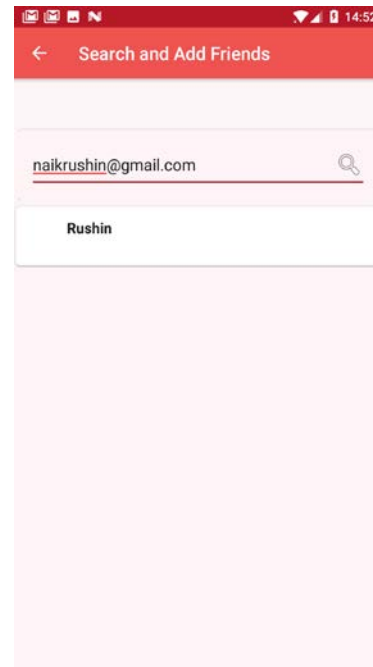
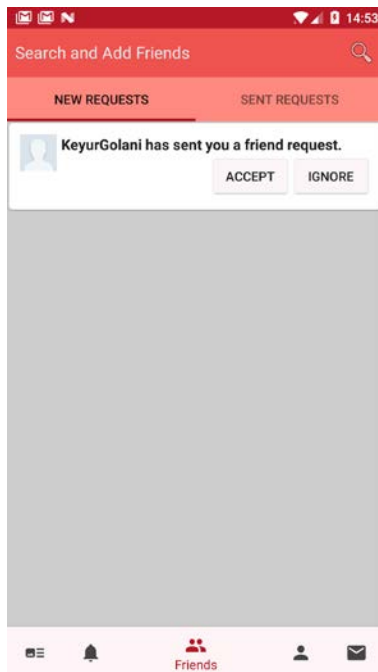
The visibility feature helps the user to decide who can view the user's profile and thus makes it a secure account. Random users may not be able to search your profile if it's private. The user can get email notification when he or she is offline. Thus, if the user is offline, a notification about any new message or new post is sent to registered email id.



## Friends and Followers

The user can have friends and also follow another user for the updates. The user can send friend request and receive friend request from other profiles. All these request can be accessed in the navigation bar.

Search Profile module helps the user to browse other users profile. The user just needs to enter a valid Email Id or valid ScreenName(unique).



### Session Management using Shared Preferences

As this is a social networking application, it consists of user profile and other related features. Session Management is really important to persist the user details. We used the Shared Preferences provided by android to overcome this barrier.

We cached the details of the user in a HashMap data set and stored in the Shared Preference object of the application. We took care of the reusability by creating a Preference Handler Class that was reused again and again to alter user profile in the session.

## Testing Plan Executed with Results

For the testing of the application, we did two types of testing. First was our standard unit testing for the Java classes and methods. The client being a thin client, we didn't need to test much on the front end side. However, the node.js backend APIs had to be tested extensively. For the Android thin client we used JUnit testing framework and for the REST APIs we used Mocha framework and Requests module together for the extensive testing.

The second testing that we did was integration testing where we used the application as a product for nearly half a day extensively and had it used by other friends too. This introduced us to any integration issues, out of memory issues with the loading images and other issues with the application which were crucial for the right result from the app.

## Learning

The project really helped us to understand how the real world social media application like Facebook, Twitter and LinkedIn work. It introduced us to various android functionalities:

- Managing user with Shared Preferences
- RecyclerView to save memory
- Managing Network Calls using Volley
- Dynamically loading multiple Fragments
- Showing time lapse using Moment.js

It also helped us to understand Cloud technologies like Google Cloud Platform and mLab.

## Future Work

### Theme Color Palette

The user can select theme colour in the settings. The theme colour will decide how the application will look inside. The user will be provided with a pop up grid to select the colors.

### Album

We created a section to add albums to the profile. We would surely like to implement the album feature if could get some more time. But, in future user can create albums and save images and also manage the albums easily.

### Push Notifications

The user can select preferences and opt for push notifications. This can be implemented using Google Cloud Messaging and JavaScript features.