

# CS 5551 Second Increment Report

Pardha Saradhi Koye    Class ID: 40  
Koushik Nallani        Class ID: 48  
Nithin Sai Peram       Class ID: 50  
Group ID: 7  
CS-5551 – FALL 2015

## I. Introduction

This document is the summary of work done by us for second increment of our android project expense tracker. In the first increment we have focused mainly on the requirements, required services, platforms, API and their usage. Details of the first increment are available in first increment report [1] submitted by us earlier.

For this increment, we mainly focused on the development of the basic functionalities of the expense tracker integrated with database. These basic functionalities include user interface design, login, registration, adding incomes, adding expenses. In addition to these we have also implemented a simple view of incomes and expenses that are added by the user.

The database tier of expense tracker was designed by using Mongo DB [2] a NoSQL database system. Database connectivity was done using Mongo DB API. Tables like incomes, expenses and users are implemented as collection and are accessed by using Mongo DB API url.

## II. Objectives and Features

The main objective of expense tracker application is to provide user with an easy interface for adding income, expense and goals. In addition to this it should also report user data with pictures and charts.

Objectives covered in this increment are

- Registering users when the application is opened for the first time.
- Login in to the application whenever user opens the application.
- Add incomes and expenses with time stamp.
- Providing a list view of incomes and expenses added by users.
- Updating existing information in the application like user details, passwords, income and expense details.
- Representing incomes and expenses using charts.

## III. Existing Services/API

Expense tracker application uses some API's like charts, voice to text and mongo db which are already existing. In this increment we have used mongo db api and chart services which is provided by ionic.

Chart Services [3]: At first during the first increment we tried to use google chart api [4] for developing analytics to our expense tracker application but after introduction of ionic framework, we used chart.js to develop bar and line charts.

Mongo DB API [5]: We used Mongo DB to store our application data. Mongo DB is an open source NoSQL database which stores tables data in the form of JSON documents. Mongo DB provides API for user to make calls using urls.

## IV. Detail Design

### Wireframes

#### 1) Login page for Expenses Tracker


The wireframe illustrates a login page titled "LOGIN". It features a central form with two input fields: "Username" and "Password". Below these fields is a "LOGIN" button. At the bottom of the page, there are two navigation options: a "Login" link accompanied by a checkmark icon, and a "Register" link accompanied by a document icon with a pencil.


We have designed a basic login page for the users to access their account in a secure way. The details of the users are stored in the mongo lab database.

## 2) Registration page for Expenses Tracker

### REGISTER

SIGN UP

  
Login

  
Register

We have designed a basic registration page for the users to create their own account to store all the details of income and daily expenses in a secure way. So that they can access their data where they want.

### 3) Password update page for Expenses Tracker

## CHANGE PASSWORD

User Name

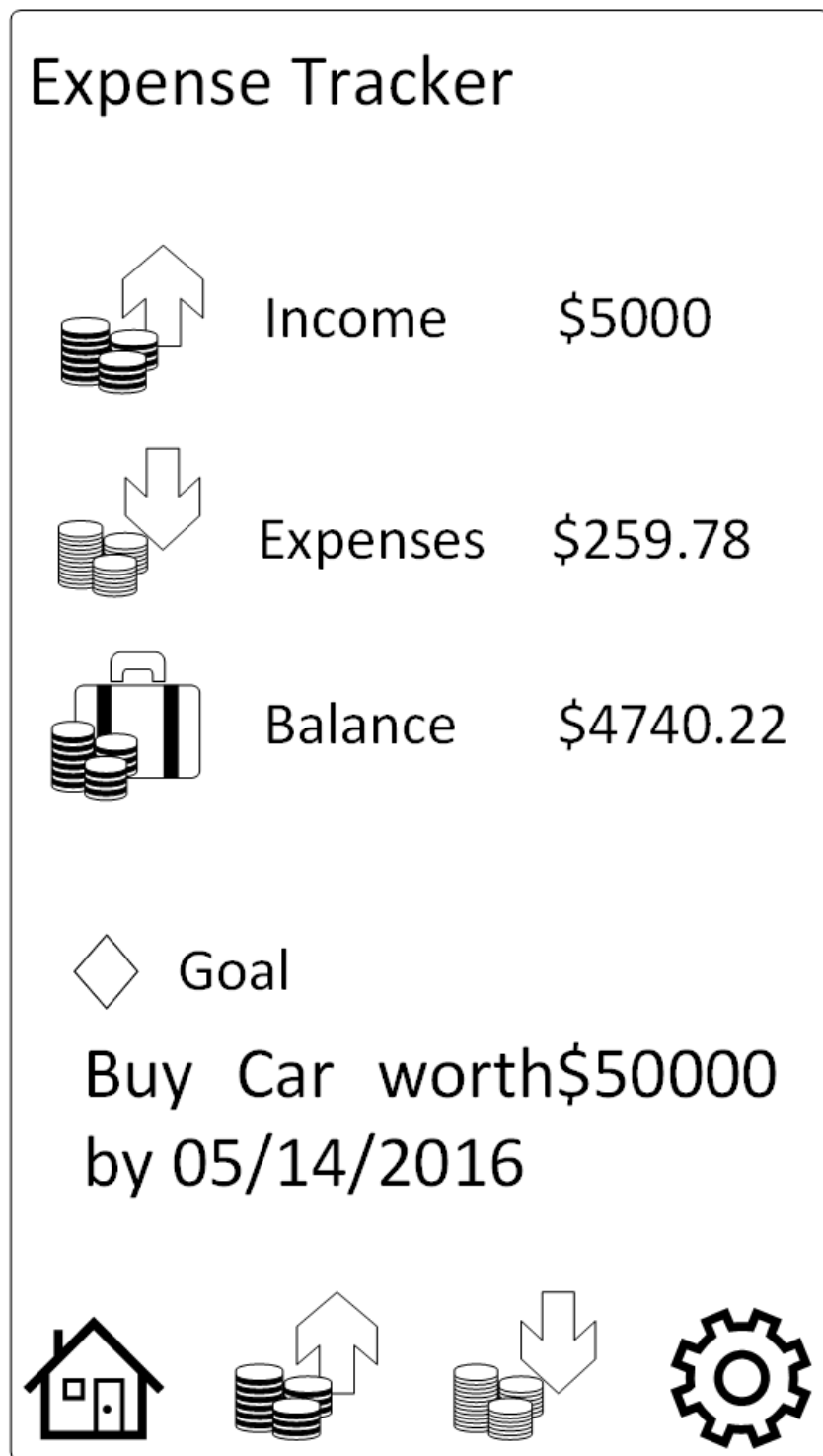
Old Password

New Password

UPDATE

We have also designed a page to update password to maintain the account in a secure way. Users can regular change their password.


#### 4) Add Expenses page for Expenses Tracker





This is the home page where the user can see all is statistics briefly.


## 5) Add Expenses page for Expenses Tracker

### ADD EXPENSES

  
Login

  
Add  
Expenses

  
Expenses

  
Account

In this page users can add expenses into their account. This page is flexible to use and user friendly. When we add expenses automatically money reduces from income. So that user can easily know how money to be spend in future.

## 6) Expenses page for Expenses Tracker


### EXPENSES

Eggs  
\$ 50

Apples  
\$ 30


Hot Food  
\$ 60

Meat  
\$ 30

  
Login

  
Add  
Expenses

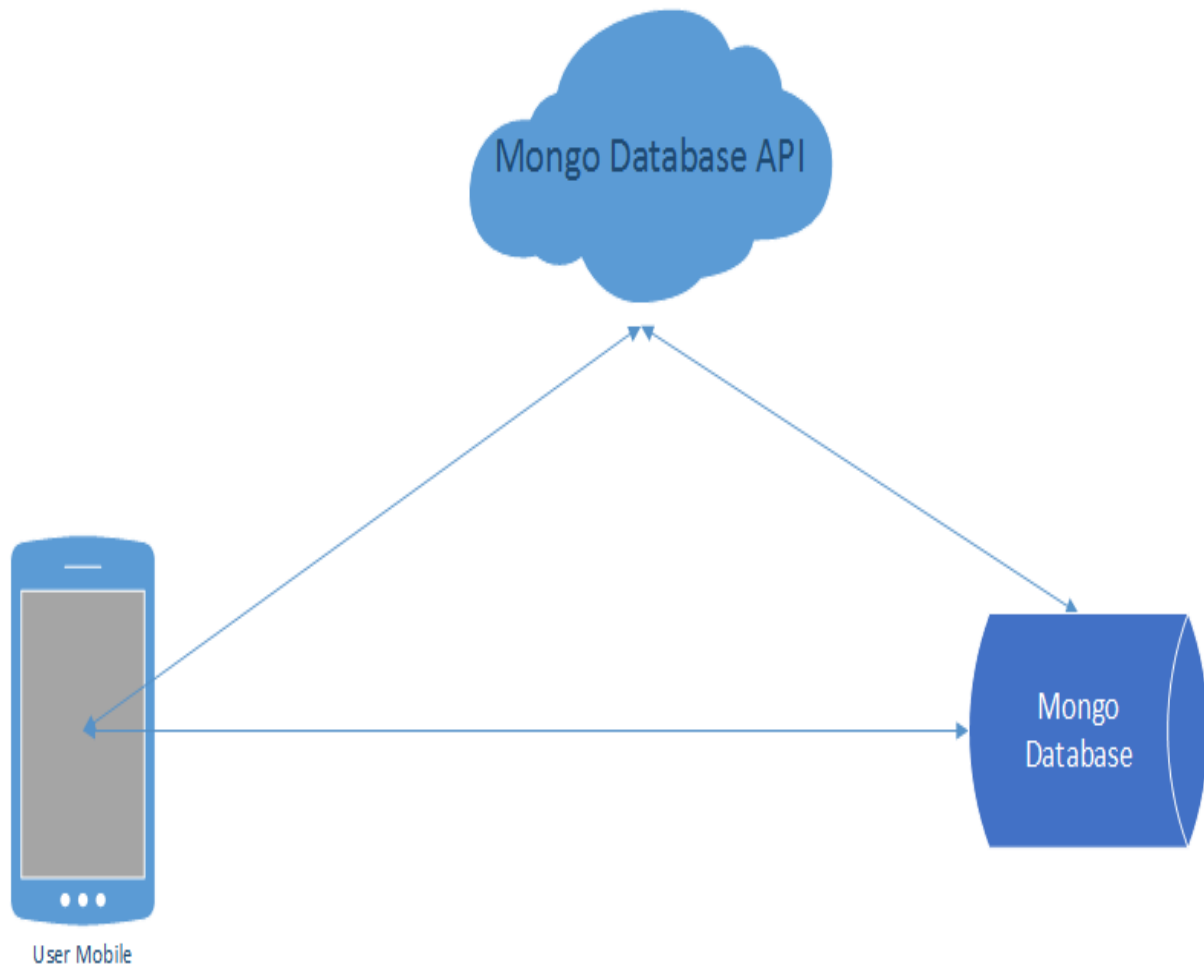
  
Expenses

  
Account

In Expenses page all the money spent by the user are stored here. User can store the data in the best way he/she can understand when they look into their expenses.

## Architecture

This shows the system architecture of the expense tracker application.

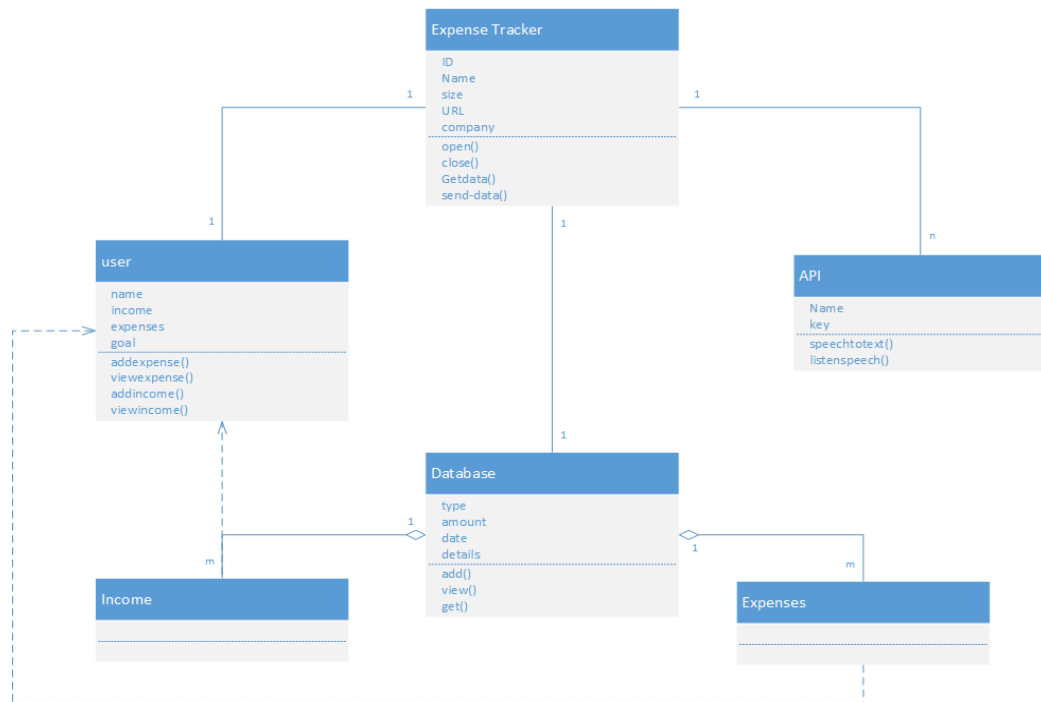


Our application follows a three tier architecture with client devices like mobile phones in the first tier that runs the application and according to user request a particular service is being called to ease the process of providing the input by the user. The last layer involves the database repositories that stores the inputs like incomes and expenses that are periodically updated by user and serves the data that is requested by the user.

## Class Diagram:

In design phase we have designed the general flow of the state of the user and the application. Structure of the data to be created is designed and represented as the following class diagram.





Expense tracker application consists of 5 classes Expense Tracker, User, Database, API, Income, Expenses. In Expense Tracker class the data values are ID, Name, size of application, URL of the project and company or organization name. Its functions are open application, close application, link with the database and get or send the data for the user or by the user.

We determine user as another class consisting of name, income, expenses, balance and goal. User operations are open app, add income or expenditure, view previous incomes and expenses, user can add a goal with a period of time and value.

Goal is the class which is dependent on the user, user may create the goal or remove to control user expenses. Its values are start date, end date, time period, value or worth for the savings of amount and details. Its functions are to add or delete a goal and also to remind about the goal if user has a lot of expenditure.

API is the class in which we use API's of values name, key and functions. We use speech-to-text and listen speech functions in the current increment.

Database is a class in which all the data is stored. Its members are type of entry income or expenses, amount or value of entry, date of entry, details about the entry. Its functions are to add, view, and get the entries to the database and from the database.

Income and Expenses are the dependents of database classes which share the features of the database class.

## Sequence Diagram:

Sequence diagram represents the flow of the application processes done by the different persons. In the initial state user open the application, the application get the data form the database and view in the dashboard. Then user decide for the operation he should be done. Initially if use press income button then the application shows two options either to view or add income. If user selects for view option then the application request the mongolab for the all incomes to display in on the screen. If user want to add income, application displays a form for the user to enter the data. After the entry updated into mongolab then the application put the data to the database and the balance results on the dashboard.

In second case if the user select the expenses button user is provided with view options. User view the application request the database for the expenses list. The list is shown on the view screen. If user want to add expense user is provided with two options entry by voice or text. In voice mode the user gives the input in form of speech then the API captures it and send to the API, in API the speech converted to data and returned to the application. Application uses the data and store in the mongolab and updates the balance and expenses value.



## V. Testing

### Unit Testing:

In development unit testing is a software testing method in which individuals units or modules of the code or application are tested with the control data to determine if there are any error or any bugs in the process. Unit testing inspires the confidence. It also forces the developer to confront the errors or problem ahead.

In our application we have used karma for the unit testing. We can also use jasmine for the unit testing. We have to write the testcases for the units or modules in javascript. At the time of testing the javascript file is taken which consist of tests written in angularjs.

### Performance Testing:

In software development performance testing is the general testing performed to determine how the system performs in the terms of response and the stability under different workloads.

There are many tools for performance testing among those we use YSlow tool for performance testing. YSlow analyse the web page and result in why they are slow based on rules developed by the yahoo for high performance sites. YSlow is a friendly tool which can be installed just by downloading the extension of the browser.

### A/B Testing:

A/B testing is a process or method of comparing the versions of a webpage or web application against them to determine which application performs better. By A/B testing the developer or the organisation validates the design changes and can improve the conversion rate. In A/B testing A version is the control and the B version is the variation.

## VI. Implementation

### Server Side Implementation:

For the server side implementation we have used [Mongolab](#) as the database for which an API access token is generated and used in the application. While performing the REST operations we use the API key for operation on the database.

“Expense Tracker” is the database name in which it has a set of collections. The data is stored in the form of JSON data but for the view of developer data can also be viewed in the form of the table created by developer choice.

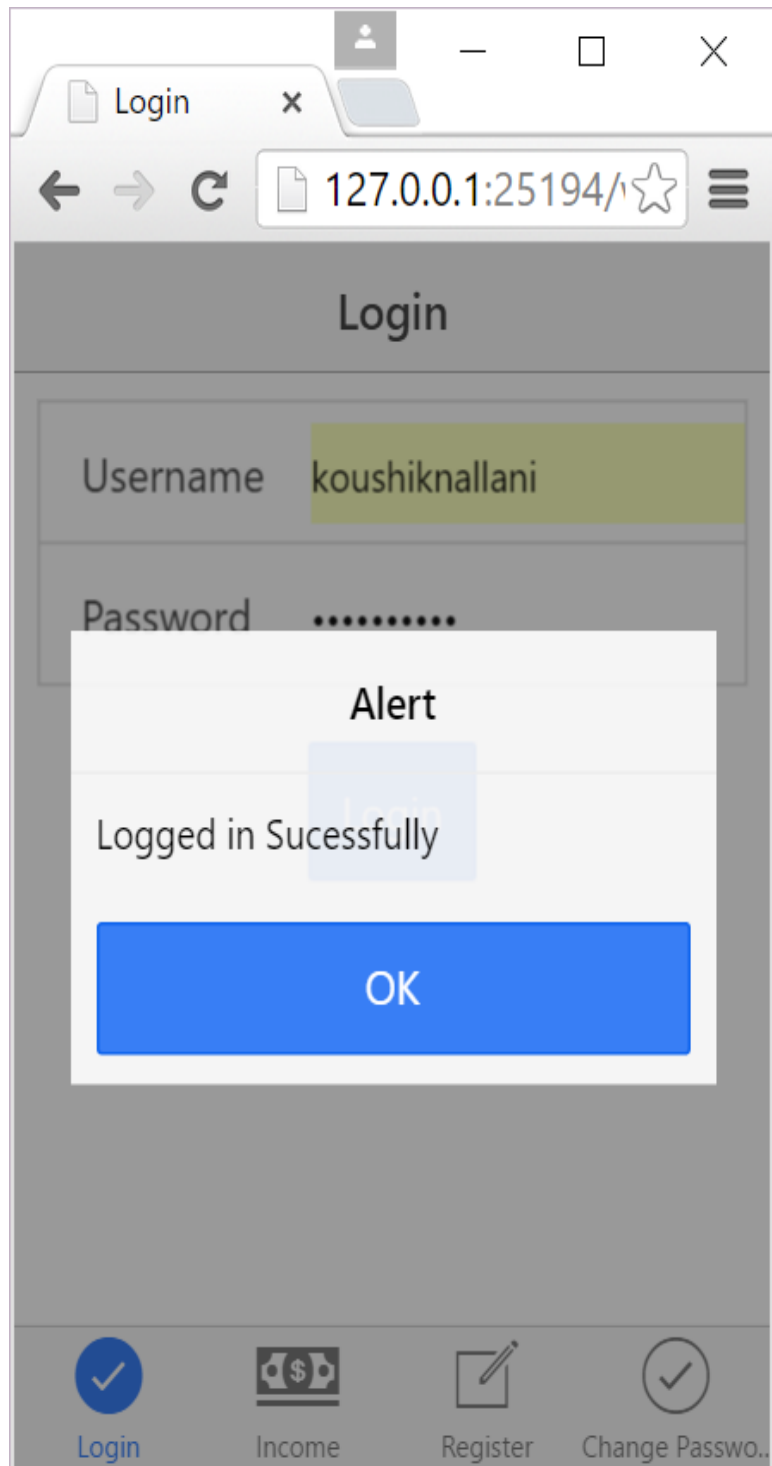
We commonly use GET, POST, PUT REST services for the application and DELETE is rarely used to delete any user or delete an entry of income or expense.

### Mobile Client Implementation:

User interface has been designed using ionic framework as a frontend, for the functionality AngularJS is used for the operations. Initially user login into the app, then the app invokes the database and validate the user. If the user is valid the dashboard is displayed else a

popup is shown as invalid user. If the user doesn't have an account user register into the application using registration interface. After the user given the inputs the backend creates a user profile in the database with default values.

Charts are used to show the statistics of the user. For charts we have used angular-charts.js and charts.js javascript files and chart.css for style.



Login validation

Register

127.0.0.1:25194/www/in

Register

First Name

koushik

Last Name

nallani

Email

k@g.com

Username

kshk

Password

....

Registered

OK

Sign Up

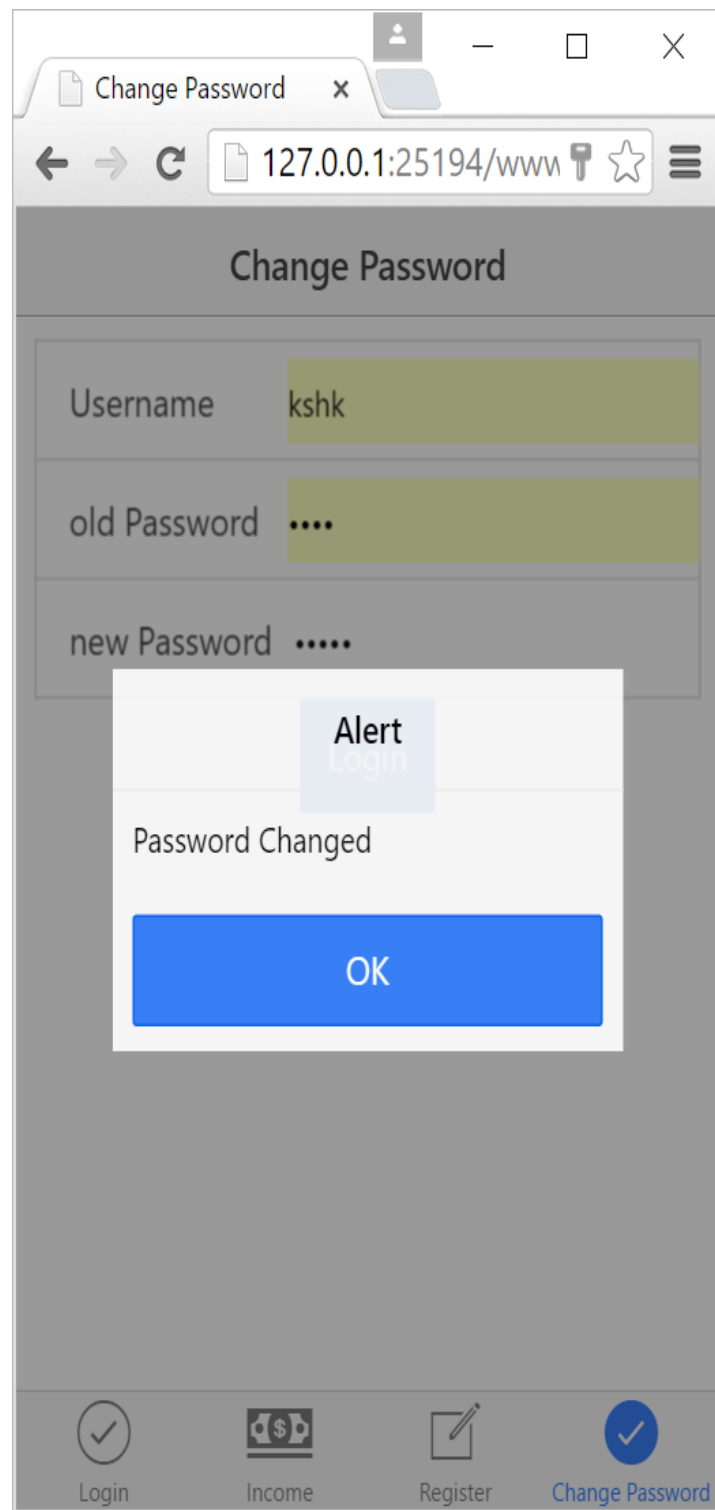
Login

Income

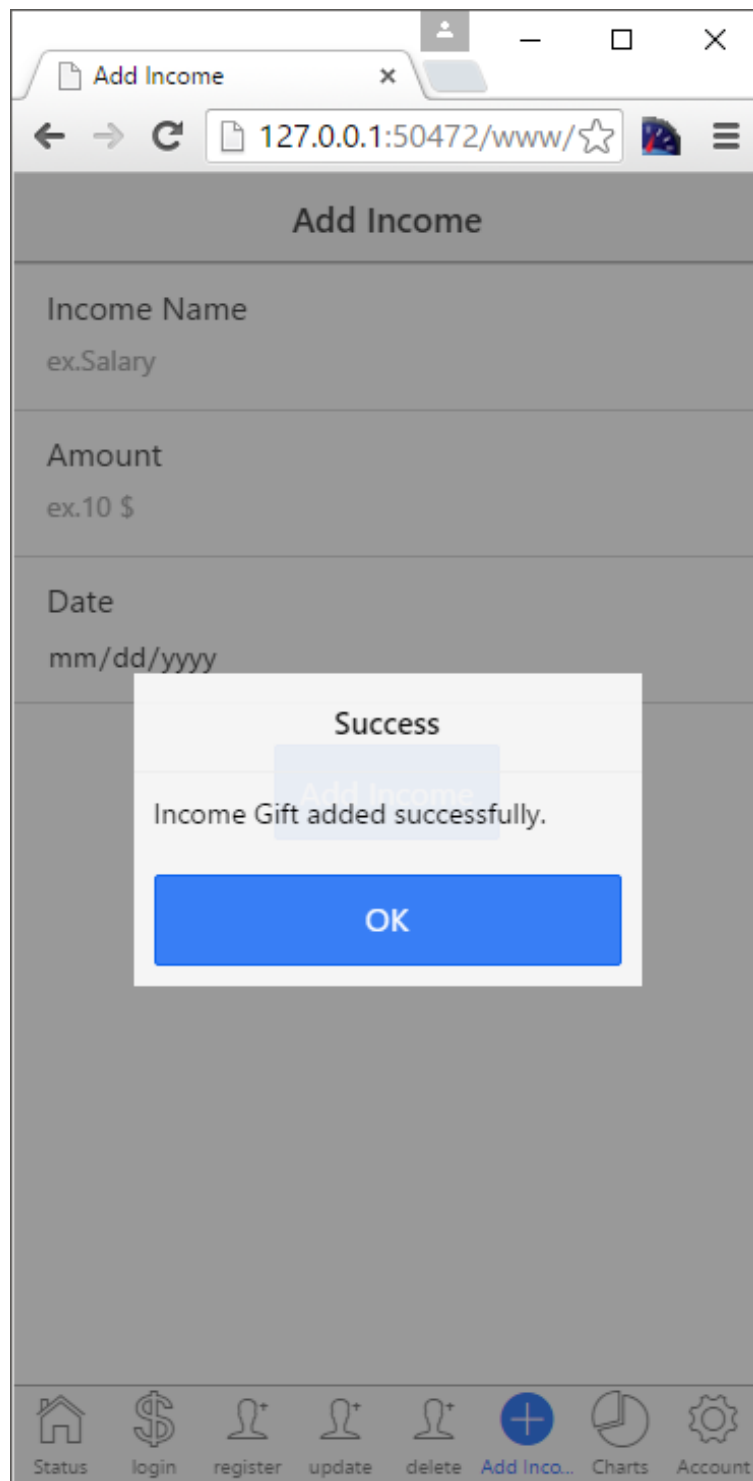
Register

Change Password

Registration



Changing password



Adding Income

## VII. Deployment

Git Hub Link: <https://github.com/pardha5/Expense-Tracker>

## VIII. Report

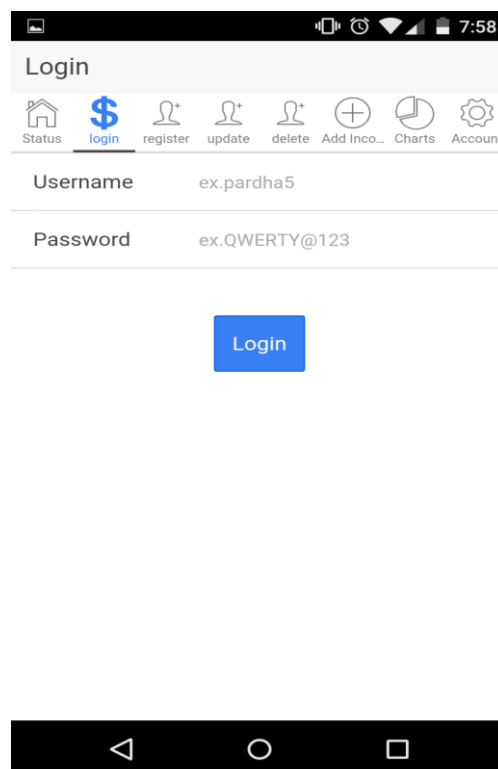
Expense tracker application is a hybrid mobile application that is developed using ionic framework. Expense tracker application architecture is pretty simple and it includes a mobile device that can run the application, a database server that stores the information generated by users and an API that connects this application to the database.

Implementation of the application is done by using ionic frame which is an open source frame work for creating android applications. Ionic frame work uses simple commands to create, build and execute applications. It comes with preloaded components that can be added easily to any application. Total frontend of the application is designed by using ionic frame work. In the back end we used Mongo DB to store the data and logic was implemented using angular JS.

Expense tracker was tested in many ways, unit testing was performed using jasmine and karma to ensure that each and every unit of code written in the application works well. On the other hand performance testing was done using Yslow[10], a plugin that grades your application based on the number of requests handled by the application at one time. User interface preferences are done by discussing different layouts of the application with our team members.

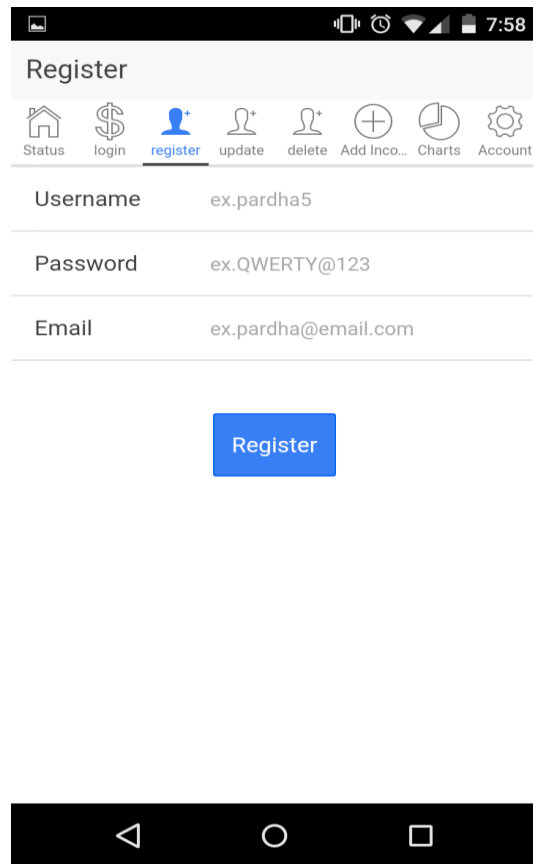
Expense tracker with basic functionalities developed till second increment was uploaded in an android mobile and it is working well. Source code of this project was deployed in github and bluemix.

### Application ScreenShots

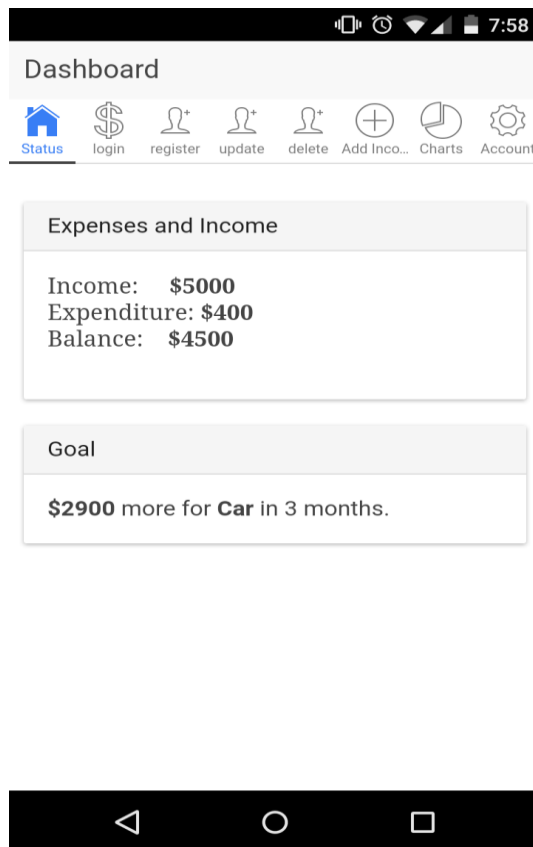


Login page





Register page



Dashboard

7:58

## Add Income

Status login register update delete **Add Inco...** Charts Account

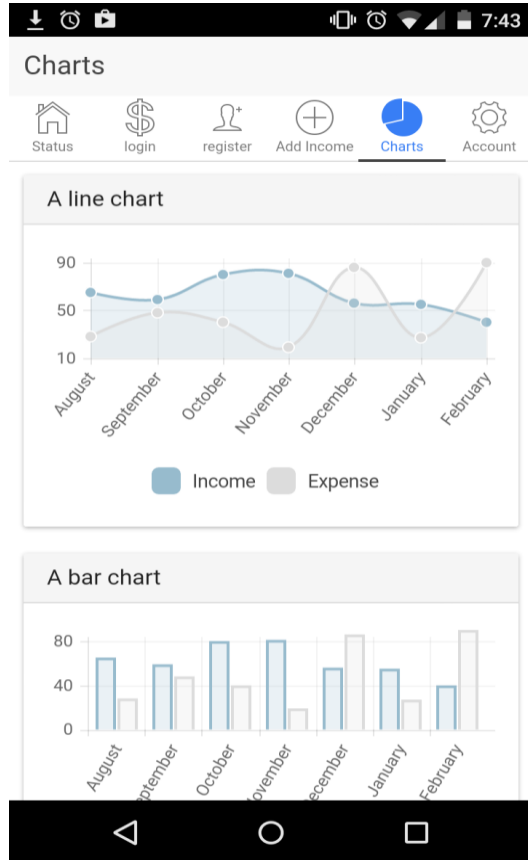
Income Name  
ex.Salary

Amount  
ex.10 \$

Date

Add Income

Add income page



Charts page

7:58

Add Income

StatusloginregisterupdatedeleteAdd Inco...ChartsAccount

Income Name

ex.Salary

Amount

ex.10 \$

Date

Add Income



## Add income page

8:02

Expenses

StatusAdd ExpenseExpensesAccount

\$

Grocery

200\$

\$

Movie

70\$

\$

Clothing

100\$

\$

Food

30\$

\$

Gas

50\$



## Expense list

## **IX. Project Management**

### **Implementation Status Report**

#### **Work completed**

- Description:

We have completed the designing mock-ups, wireframes, class diagram, sequence diagram, state diagram and developed user interface, successfully established connection between the database and mobile application.

- Responsibility (Task, Person):

Pardha Saradhi koya (40): Adding Income, View Income and Charts.

Koushik Nallani (48): Login, Registration and Change Password.

Nithin Sai Peram (50): Expense List, Add Expense and Dashboard

- Time taken:

20 hours each person.

- Contributions (members/percentage)

Each 33.3 percentage.

#### **Work to be completed**

- Description

Integration of speech to text api for taking input using voice.

Goal setting for user and database implementation.

Combining all modules to make one single app.

- Responsibility (Task, Person)

Goal setting, Pardha Saradhi Koye

Voice to text, Koushik Nallani

Database Implementation, Nithin Sai Peram

- Time to be taken (estimated #hours)

50 hours for all.

## X. Bibliography

- [1] <https://github.com/pardha5/Expense-Tracker/blob/master/Documentation/Project%20Increment%201.pdf>
- [2] <http://mongolab.com/databases/expensetracker>
- [3] <http://www.chartjs.org/>
- [4] <https://developers.google.com/chart/interactive/docs/gallery?hl=en>
- [5] <https://mongolab.com/databases/expensetracker>
- [6] <http://docs.mongolab.com/>
- [7] <http://ionicframework.com/docs/components/>
- [8] <http://www.w3schools.com/angular/>
- [9] <https://www.genymotion.com/#/>
- [10] <http://yslow.org/>
- [11] <http://karma-runner.github.io/0.13/index.html>
- [12] <http://jasmine.github.io/2.0/introduction.html>