CS 5551 Advance Software Engineering
FALL 2015 Project

Expense Tracker

By

Group 7

Pardha Saradhi Koye	16210833
Koushik Nallani	16208484
Nithin Sai Peram	16208687

CS 5551 Project Proposal

Group No: 7

Project Title: “Expense Tracker”

Group Members

1. Pardha Saradhi Koye (40)
2. Koushik Nallani (48)
3. Nithin Sai Peram (50)

Project Goals and Objectives

- **Motivation:** In a busy day to day life, unknowingly a man spends lot of money for various needs. Sometimes these expenses may exceed his/her monthly budget or income. In order to organize cash flow and financial management there is a necessity of a personal expense tracking application that stores your daily expenses and provides a clear view of your finances.

- **Significance:** This application allows users to add their income and expenses with details like category and date. It provides a list of all the incomes and expenditures that are added by user in an order. It also notifies the user about his total expenses in regular intervals. User can always see his total incomes, expenditures and remaining balance.

- **Objectives:** The main objective of this project is to implement a simple and robust mobile application for tracking expenses. This system helps users to record their daily expenses and helps them to plan their future expenses. It keeps a track of user money and gives a views on the items in which user spends most of his/her money. It also provides financial stability to the users by providing monthly reports.

- **System Features:**
    1. User can add income to his account
    2. User can add expenses which includes expense name, amount, category and date
    3. User can view list of all incomes added by date
    4. User can view list of all expenses added by date
    5. User can view total income, expenses and remaining balance at any time
    6. User can reset all the values to start a fresh tracking.

Related Works

There are some applications already developed in android and iOS like ispending and expense manager which are paid applications. Many applications which can hold more than one person expenses are also in development.

CS5551 Project Plan

Introduction

Technology is increasing day by day, with the increase in technology sales of smart phones reached maximum over the last decade. People in every country are using smart phones and they have become the part of their life style. The main feature in a smart phone is the ability to run different mobile applications that allows user to access, store or capture different kinds of information right from their mobile. The importance of mobile applications is that they are in to different categories like shopping, entertainment, sports, games, finance, food, fitness and online booking.

Expense Tracker is a mobile application that falls in to the finance category and its purpose is to manage the finances of the user which is very important for every person. In a busy day to day life, unknowingly a man spends lot of money for various needs. Sometimes these expenses may exceed his/her monthly budget or income. In order to organize cash flow and financial management there is a necessity of a personal expense tracking application that stores your daily expenses and provides a clear view of your finances.

Project Goal and Objectives

Overall Goal:

The main objective of this project is to implement a mobile application for tracking expenses. This system helps the users to plan their future expenses as per the past history of expenses. Tracking all your expenses and incomes so as to put you on the path to financial stability and tracking expenses and incomes day by day. This application gives an overview to user about his expenses with the help of visualizations like charts.

Specific Objectives:

Any financial management application should have some basic features like adding data and representing that information using pictures or charts. Our application objective too deals with collecting information from user, storing data collected and presenting data in a manner where user can analyze his financial status

The main objectives of this system are

- This system helps users to record their daily expenses and helps them to plan their future expenses.
- It keeps a track of user money and gives a views on the items in which user spends most of his/her money.
- Displays data using charts.
- It also provides financial stability to the users by providing monthly reports.
- It can also provide user to create goals and suggests him with a plan to reach his goal.

Significance:

This application allows users to add their income and expenses with details like category and date. It provides a list of all the incomes and expenditures that are added by user in an order. It also notifies the user about his total expenses in regular intervals. User can always see his total incomes, expenditures and remaining balance.

Proposed System

1) Requirements Specification:

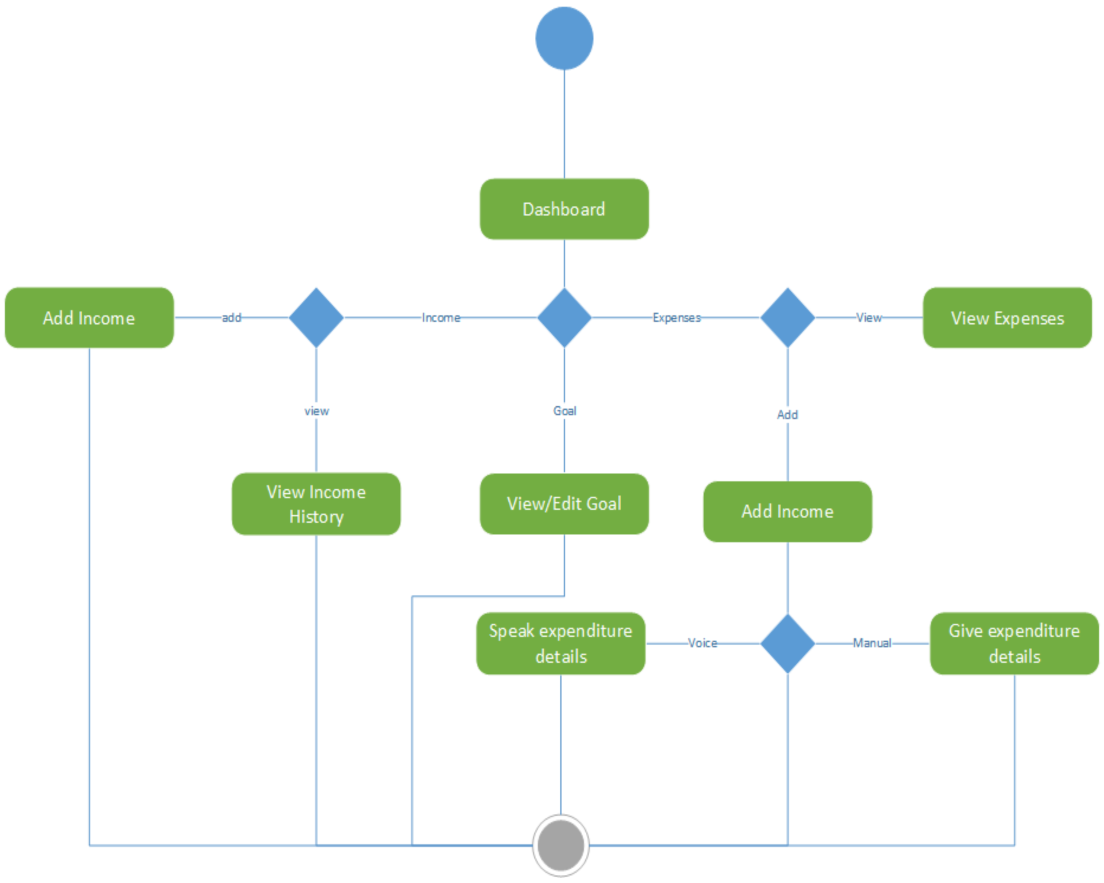
Functional requirements:

- Expense Tracker phone application should allow end users to record all sort of daily expenses like grocery, medical, education, entertainment, automobile, insurance etc.
- Expense Tracker in offline mode should allow expenses to be recorded and saved locally on the device
- Application should report these expenses in very useful manner for user to help him make decision about future expenses
- Expense Tracker allows you to reset all the incomes and expense
- Expense Tracker will ease and speed up the planning, decision-making, and process, secure, confidential and reliable reports.
- Display summary using charts
- Allow users to input data with speech.

Non Functional Requirements:

- Performance: Performance is always a programming issue not related to any physical or hardware part of the system. Since we have gathered every piece of information is needed for the development and performance orientation.
- Maintainability: The main aim of any application is to save the data efficiently and its perfect maintenance avoiding the mistakes. This application can be able to provide such maintenance of data through its functional modules that has been generated.
- Usability: The app developed must be simple enough that user with average background in using mobile phones can quickly experiment with the system and learn how to use the project. The system must have user friendly interface.
- Security: This application mainly deals with the user data that he enters during the usage of this app so the data should be kept confidential to maintain users privacy.
- Reliability: Since the applications backend is being developed through java, the most famous efficient and reliable language, so it is reliable in every aspect until and unless there is an error in the programming side. Thus the application can be compatible and reliable one.

Workflow Analysis:
Work flow in our app is based on the interaction by the user with the application. Clicking a button or tab in the application takes user to another page and each activity from user has a specific output.



Activity Diagram for Expense Tracker

When user opens the application a dashboard is displayed. When he choose any option either income or expenses user further requested with two options add or view income/expenses. User can also add a goal from the dashboard. When the user tries to add expenses the app requests for manual or by voice. If the user choose by voice user needs to give input by voice then the service is used to transfer the speech to text and updated in the data base.

Technological Requirements:
As we are developing a mobile application we require software development kits like android sdk or iOS sdk to develop the application. These sdks provide us the ease to create the front end of the application by just dragging and dropping the elements. For the backend we use technologies like Java and JavaScript so java developing kit and java run time environment should be configured in the system that is used to develop the application. Expense Tracker also requires a technology to store and retrieve the information from the database so a knowledge on how manage the data is required. Data base technology like SQLite, Mango DB is required to develop this application.

Architectural Requirements:

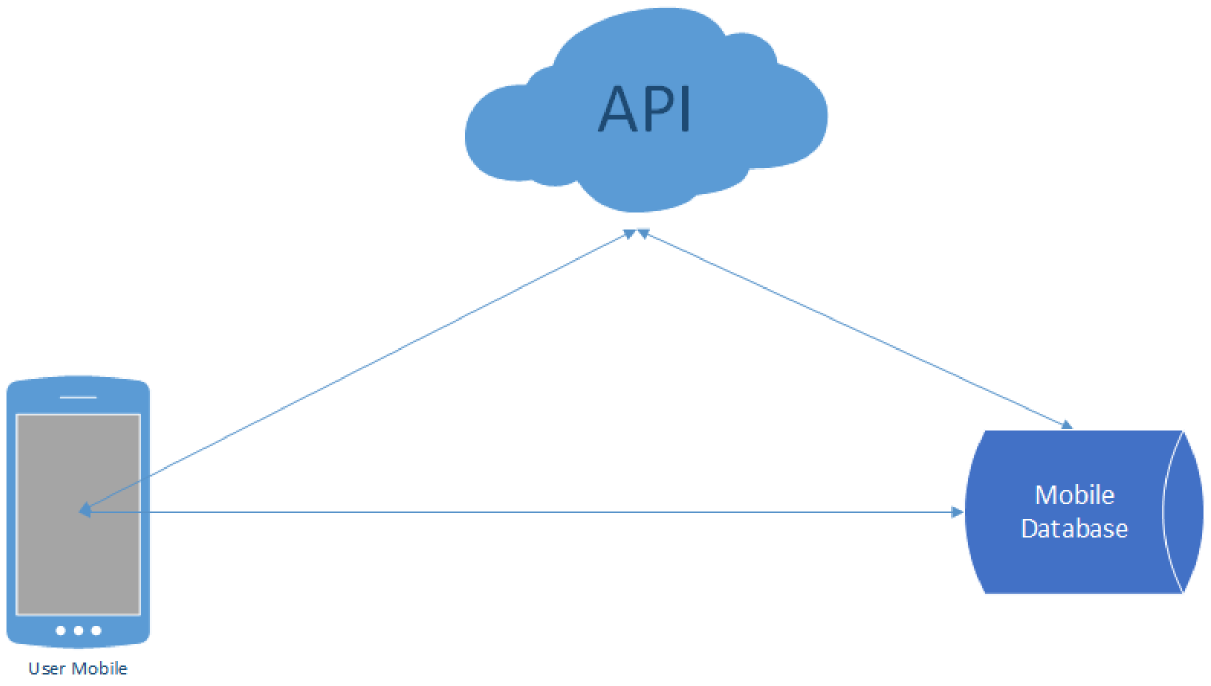
Architecture is the important aspect in a project as it involves the major elements that are used for the development of project. Expense Tracker also contains a three layers. First layer consists of the system that is used by the user to interact with the application as we are developing a mobile application the first part of our architecture will be the mobile phone or tablet.

The second layer consists of the APIs or the services that records the data from the user in different formats and converts them to the format that can be stored in the database.

Third Layer consists of the database which may be the local database in the android phone or the database that is present in the cloud.

2) Framework Specification:

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.



This shows the system architecture of the expense tracker application.

3) **System Specification:**

Existing Services:

In our project we are using some services like google chart services and speech to text service which are already existing

Google Chart Services: https://developers.google.com/chart/?hl=en

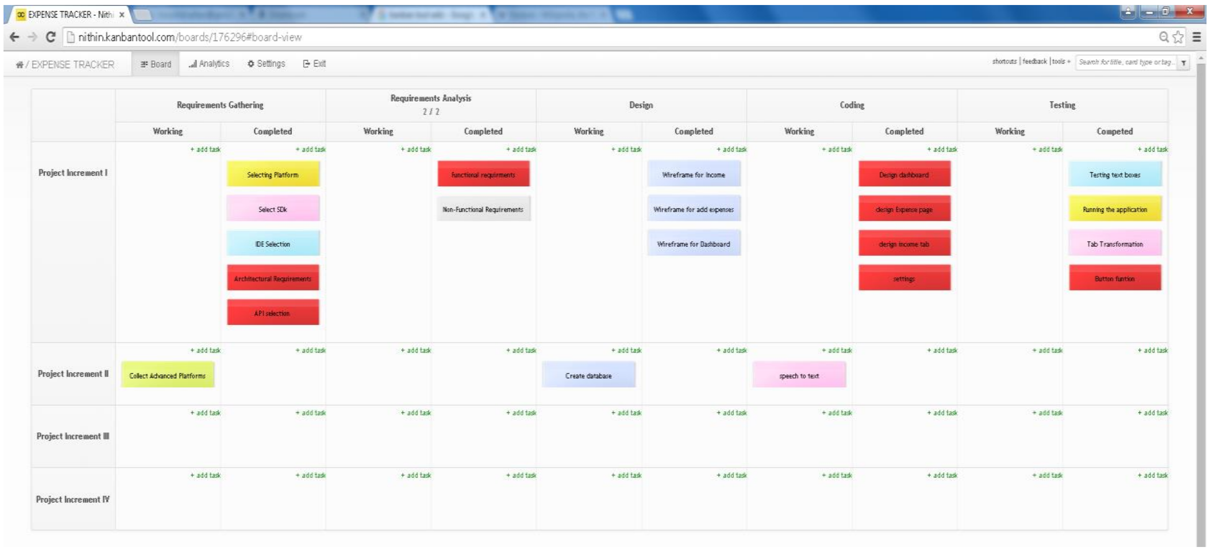
Google Charts API provides us with a wide variety of charts that can be used in our project to represent the user data in a pictorial way. This API is free to use and converts the table data stored in database to a chart with rich details.

Speech to Text API: https://dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html

Our application also requires a API that recognizes speech from the user and converts them in to the text which can be later saved to database. This API provides user a facility to give inputs to the app by speech. It will act as a mini voice assistant to the application to record user inputs.

Project Plan:

Kanban tool is best for collaborative working and for just in time tasks. It is easy view the tasks that are completed by different members in the project. It can be operated by the all the members in the project and it is very to understand the tasks completed in the project. It is the best tool to be used in project to analysis the present status of the project and also to now the status of individual work done by the project members.


EXPENSE TRACKER - Nith
nithin.kanbantool.com/boards/176296#board-view
/ EXPENSE TRACKER
Board
Analytics
Settings
Exit
shortcuts | feedback | tools
Search for title, card type or tag...

	Requirements Gathering		Requirements Analysis 2 / 2		Design		Coding		Testing	
	Working	Completed	Working	Completed	Working	Completed	Working	Completed	Working	Competed
Project Increment I		Selecting Platform Select SDk IDE Selection Architectural Requirements API selection		functional requirments Non-Functional Requirements		Wireframe for Income Wireframe for add expenses Wireframe for Dashboard		Design dashboard design Expense page design income tab settings		Testing text boxes Running the application Tab Transformation Button funtion
Project Increment II	Collect Advanced Platforms				Create database		speech to text			
Project Increment III										
Project Increment IV										

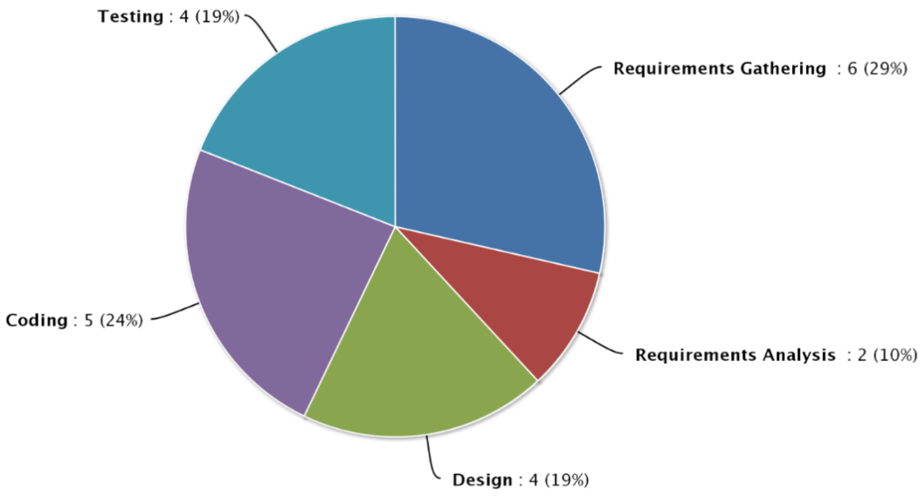
Project board created using Kanban Tool.

Tasks Included on Board:

1) We have selected the platforms, SDk , IDE, architectural designs and API which required for the project increment-I.
2) We have analyzed all the functional and non-functional requirements.
3) Also Designed wire frames for the income of user, add expenses and dashboard.

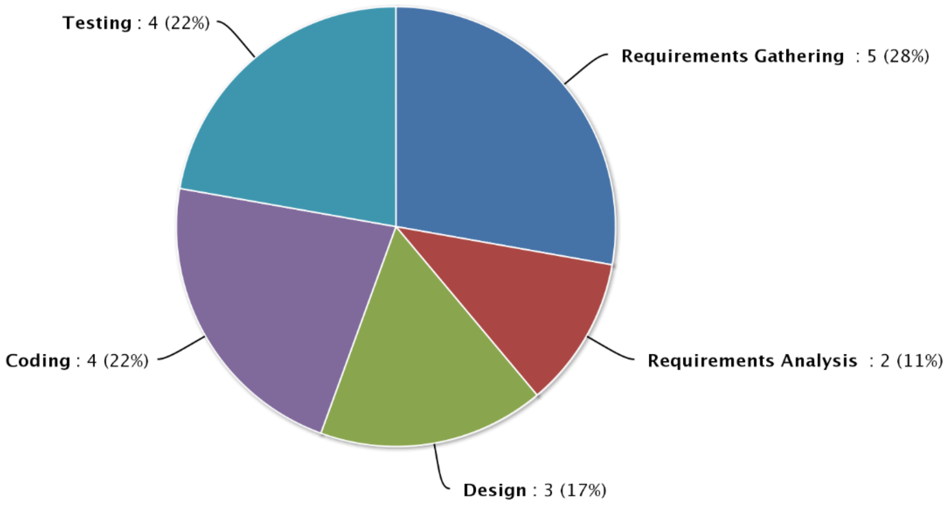
4) We have done coding for design dashboard, design add expenses, design income tab and settings tab.
5) We tested whole application by running on the hand set.
6) We have also stared to more advanced applications and tools to improve the interface.
7) We started designing database for storing data in the handset and for that we working on the coding.

Analytics Pie Chart:

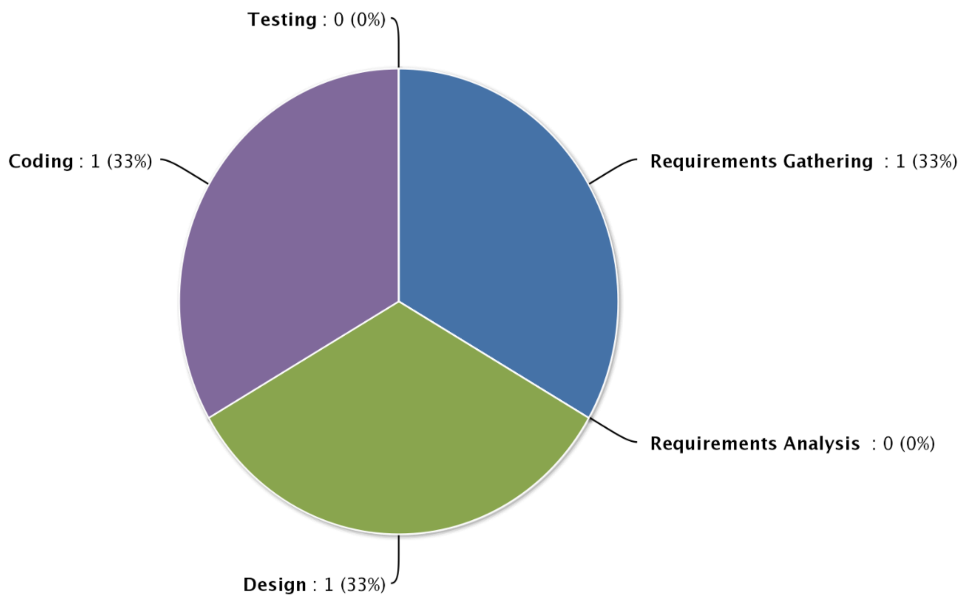


Pie Chart for all tasks

This analysis for the whole tasks in the board. In Kanban tool we can divided different tasks and can analyze the chart. For example if we want to analyze project increment-I we can do that very easily. It is the best user friendly tool.



This is the analysis chart for project increment-I.



This is the analysis chart for project increment-II.

Bibliography

[1] http://www.nytimes.com/2014/01/04/your-money/household-budgeting/review-apps-to-track-income-and-expenses.html?_r=0

[2] https://itunes.apple.com/us/app/ispending-expense-tracker/id484100875?mt=8

[3] https://play.google.com/store/apps/details?id=com.expensemanager

[4] https://developer.android.com/sdk/index.html

[5] https://developers.google.com/chart/?hl=en

[6] https://dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html

[7] https://docs.google.com/spreadsheets/d/1vBMJ4gp22y1JzebvTcwkDS1r9Aals2H7H-ml_wznHSg/edit#gid=1973386343

[8] https://kanbantool.com/

[9] https://github.com/pardha5/Expense-Tracker

[10] http://www.androidhive.info/2014/07/android-speech-to-text-tutorial/

CS 5551 First Increment Report

Introduction

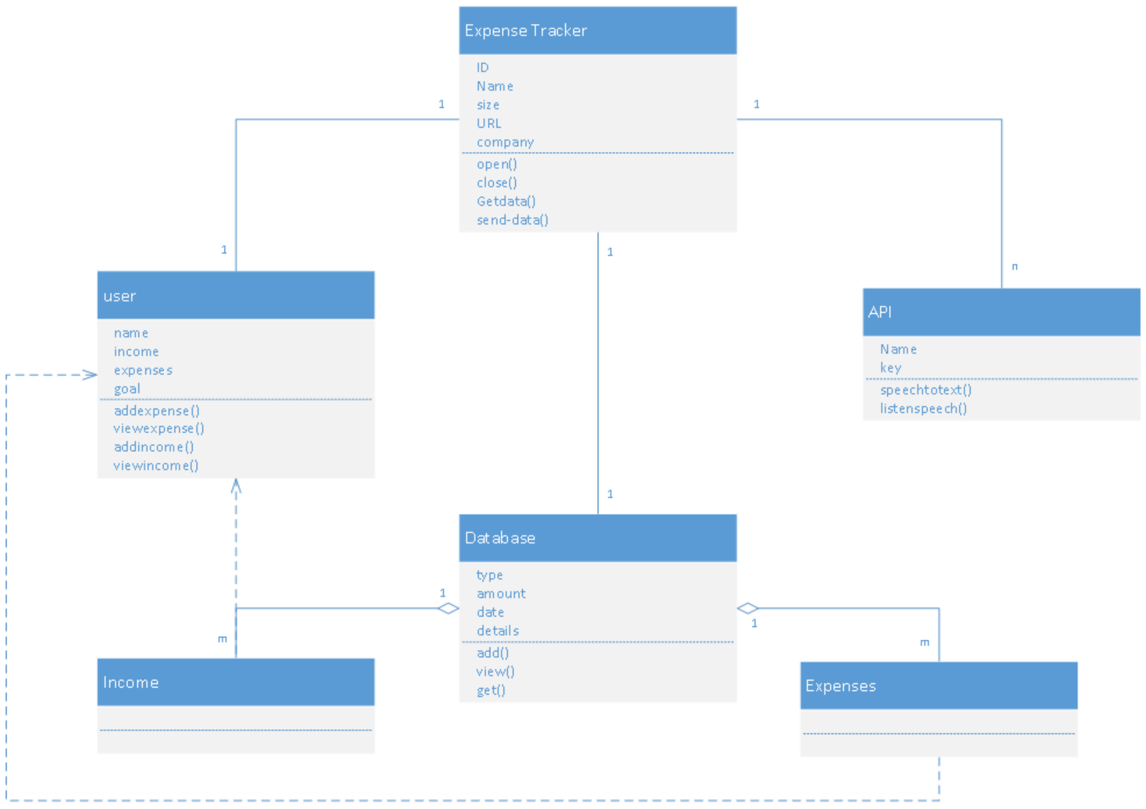
This is the summary report of the first iteration of the work done in Expense Tracker application. This application is used to control and track the income and expenses details of the user and also to remind the user about his goals.

For the first iteration we have focused our work on the requirement phase in gathering the required services, platforms, API and their usage. Also we have designed the architecture of the application which we should design.

Design Overview:

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.

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 organisation 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.

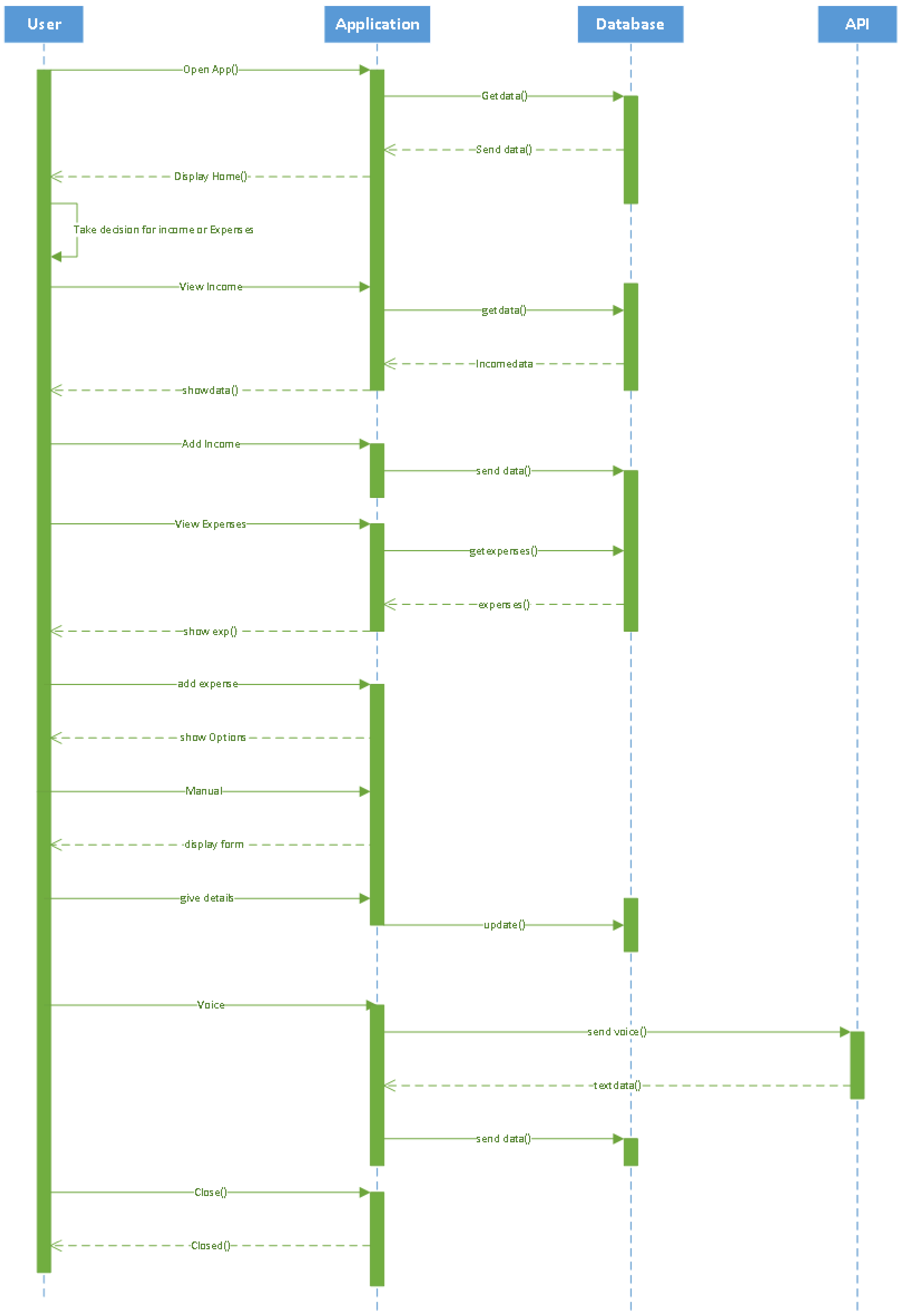
Database is a class in which all the data is stored. Its members are type of entry income or expenses, amount or value of entry, date of entry, details about the entry. Its functions are to add, view, and get the entries to the database and from the database.

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

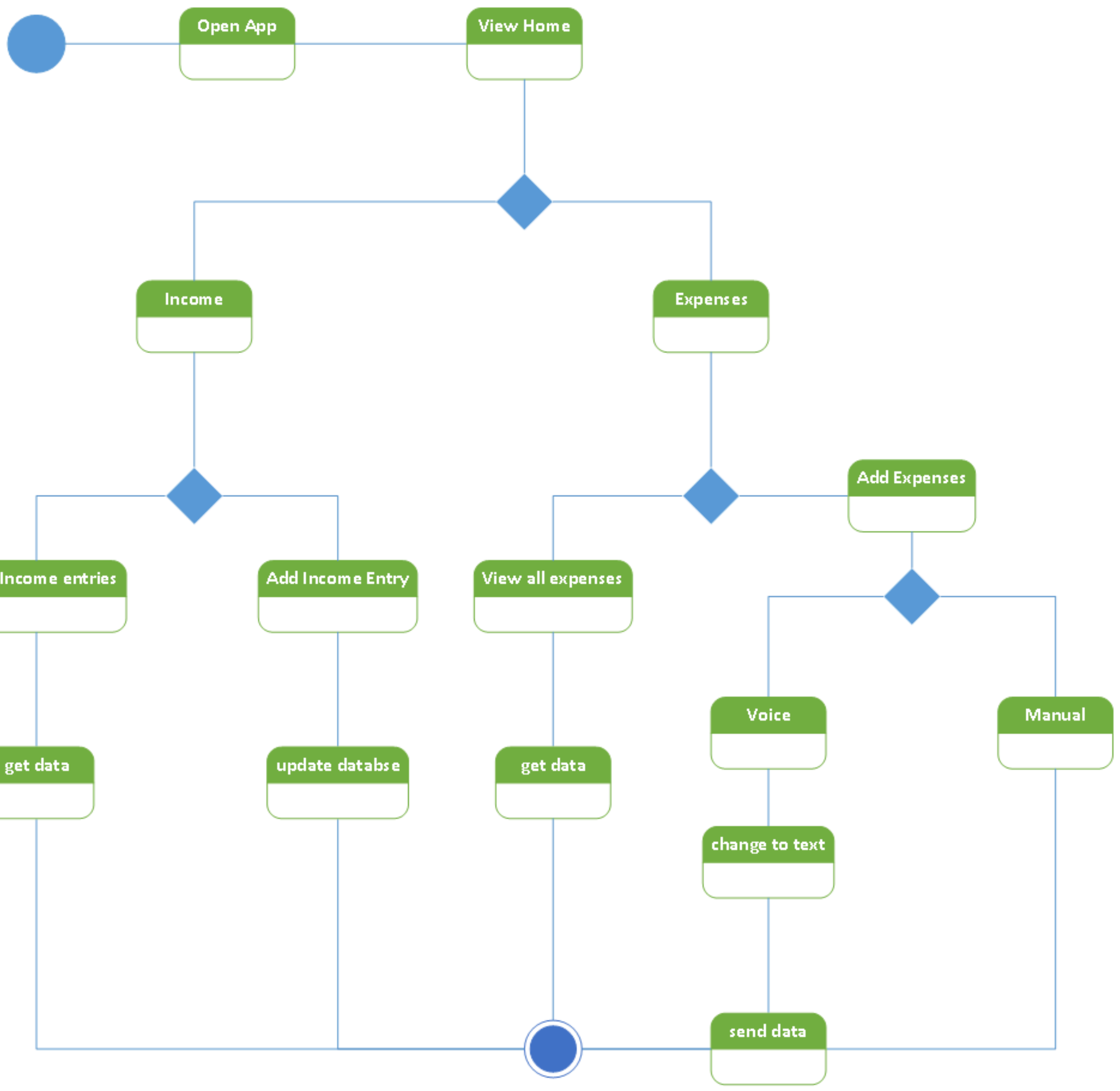
Sequence Diagram:

Sequence diagram represents the flow of the application processes done by the different roles. In the initial sate user opens the application, the application fetch the data form the database and view in the dashboard. Then user decide for the operation he should be done. In first case if use press income button then the application shows two options either to add or view income. If user selects for view option then the application request the database for the all income history to display in on the screen. If user selects for add income, application displays a form for the user to enter the inputs. After the entry the user press add then the application send the data to the database and the balance gets updated on the dashboard.

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

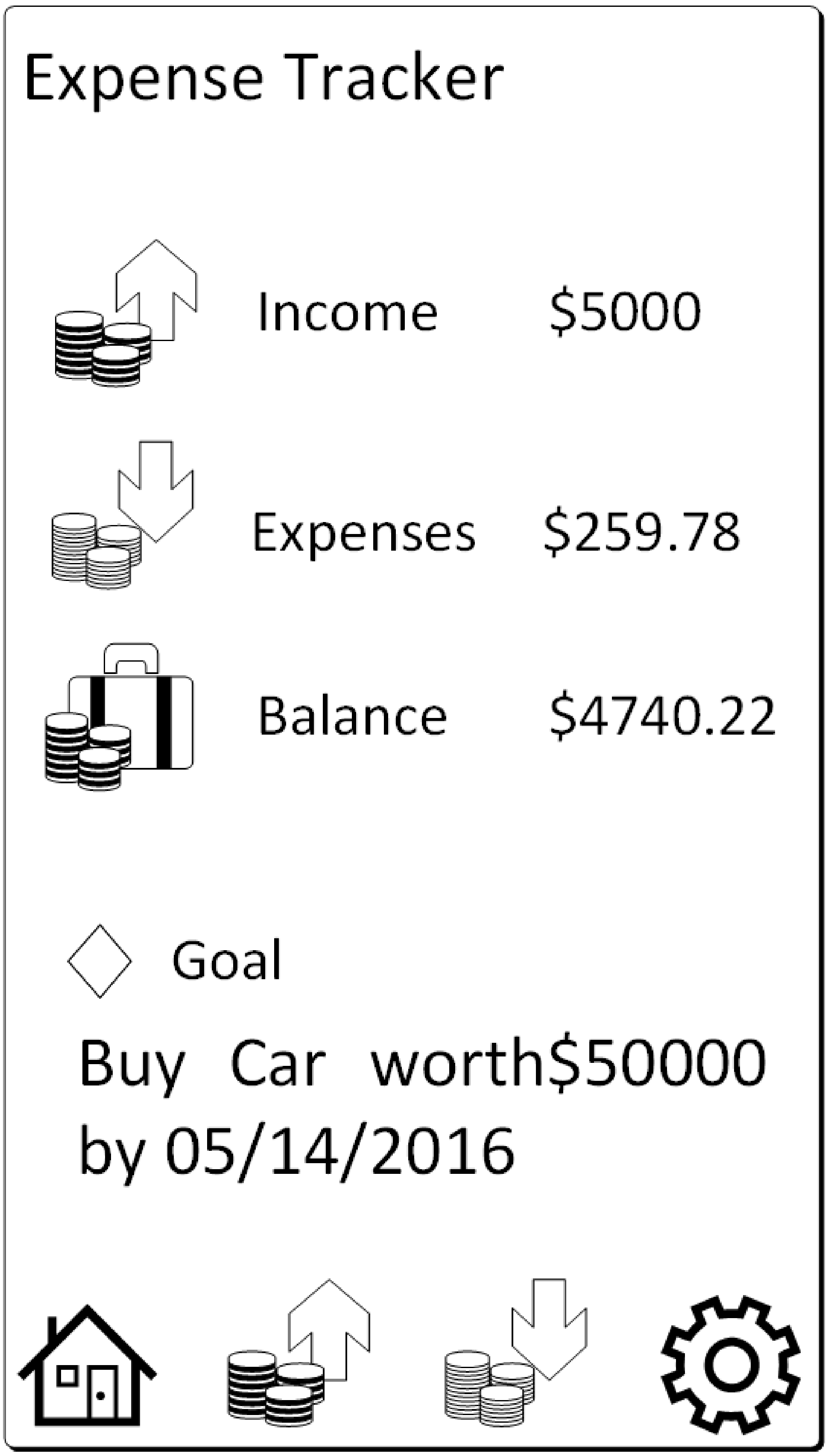

User
Application
Database
API
Open App()
Get data()
Send data()
Display Home()
Take decision for income or Expenses
View Income
get data()
Incomedata
showdata()
Add Income
send data()
View Expenses
get expenses()
expenses()
show exp()
add expense
show Options
Manual
display form
give details
update()
Voice
send voice()
text data()
send data()
Close()
Closed()

State Chart Diagram:



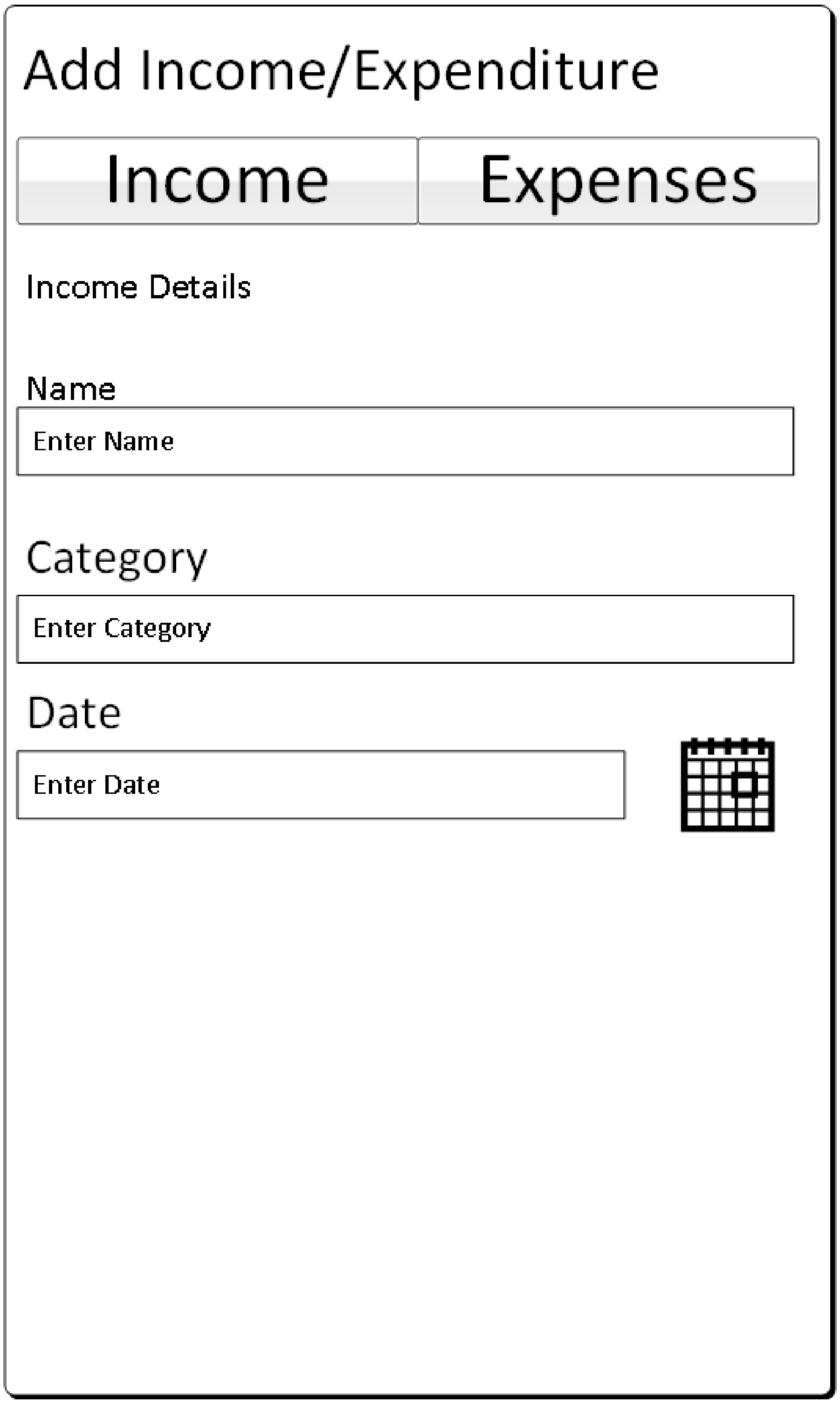
Initially user is in idle state user opens the app and view home. In the dashboard user can view the summary of income, expenses, balance and goal. User decides either income or expenses, if the user decide for expenses he can add or view the income. Similarly hew can add or view expenses. When the user decides to view income or expenses the application request the database and view the results. In case of add of income user gives the input and the application updates in the database. While in adding of expenses user can give inputs manual by entering the data or by give a voice speech. In case of voice speech the API is invoked to change the speech to text, after transformation the text file is sent to the application and application is stores the data in the database. Finally user closes the application.

WireFrames and Mockups:



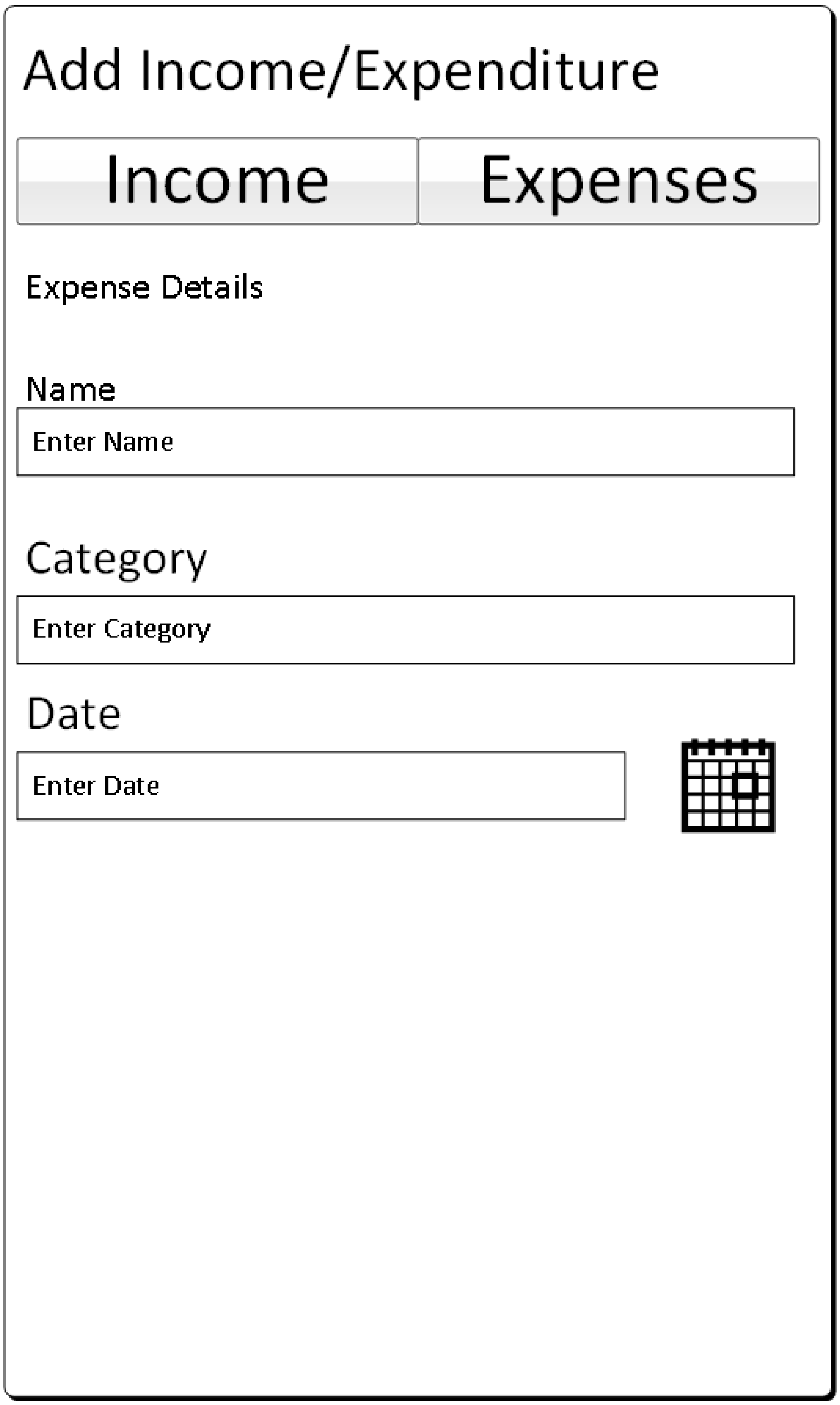
Wireframe of Dashboard

The above picture shows the view of the dashboard which consists of income value, value of expenditure spent and the final balance remaing. In addition we can also see the goal created by the user.



Wireframe of Income Window

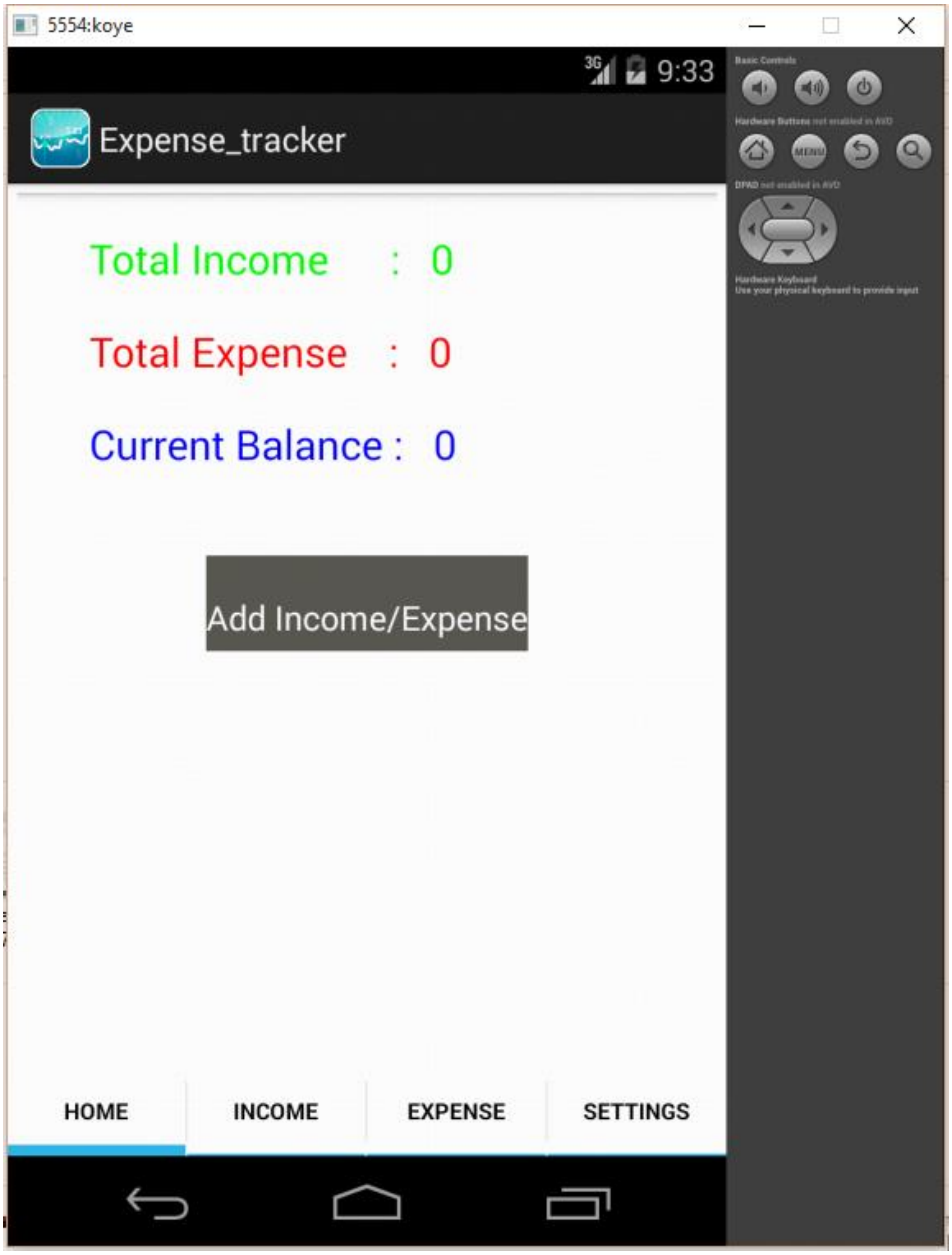
In the above wireframe the user gives the input to add an expense occurred by the user.



Wireframe of Add Expenses (manual)

In the above wireframe the user gives the input to add an expense occurred by the user.

Implementation Overview:

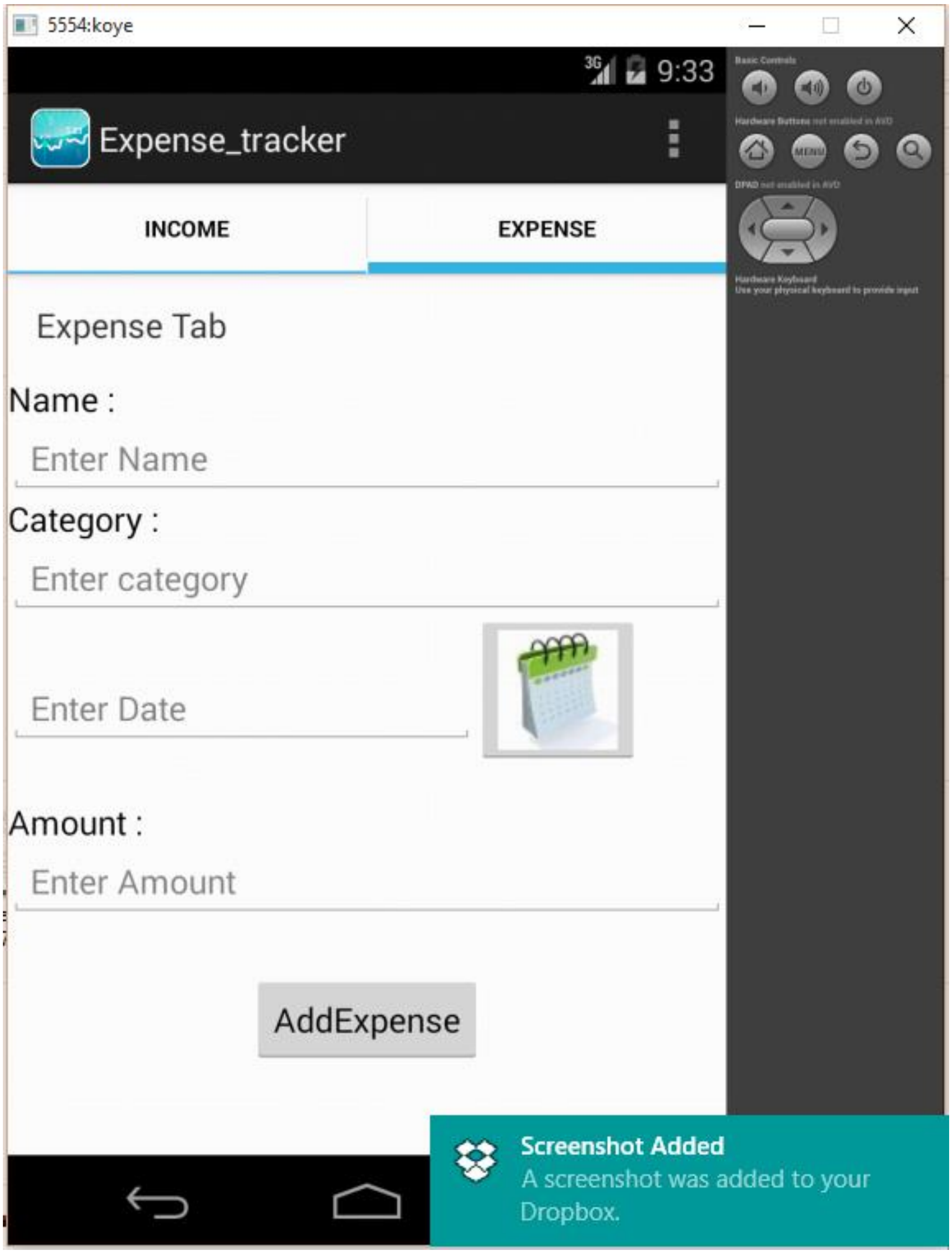


Total Income : 0

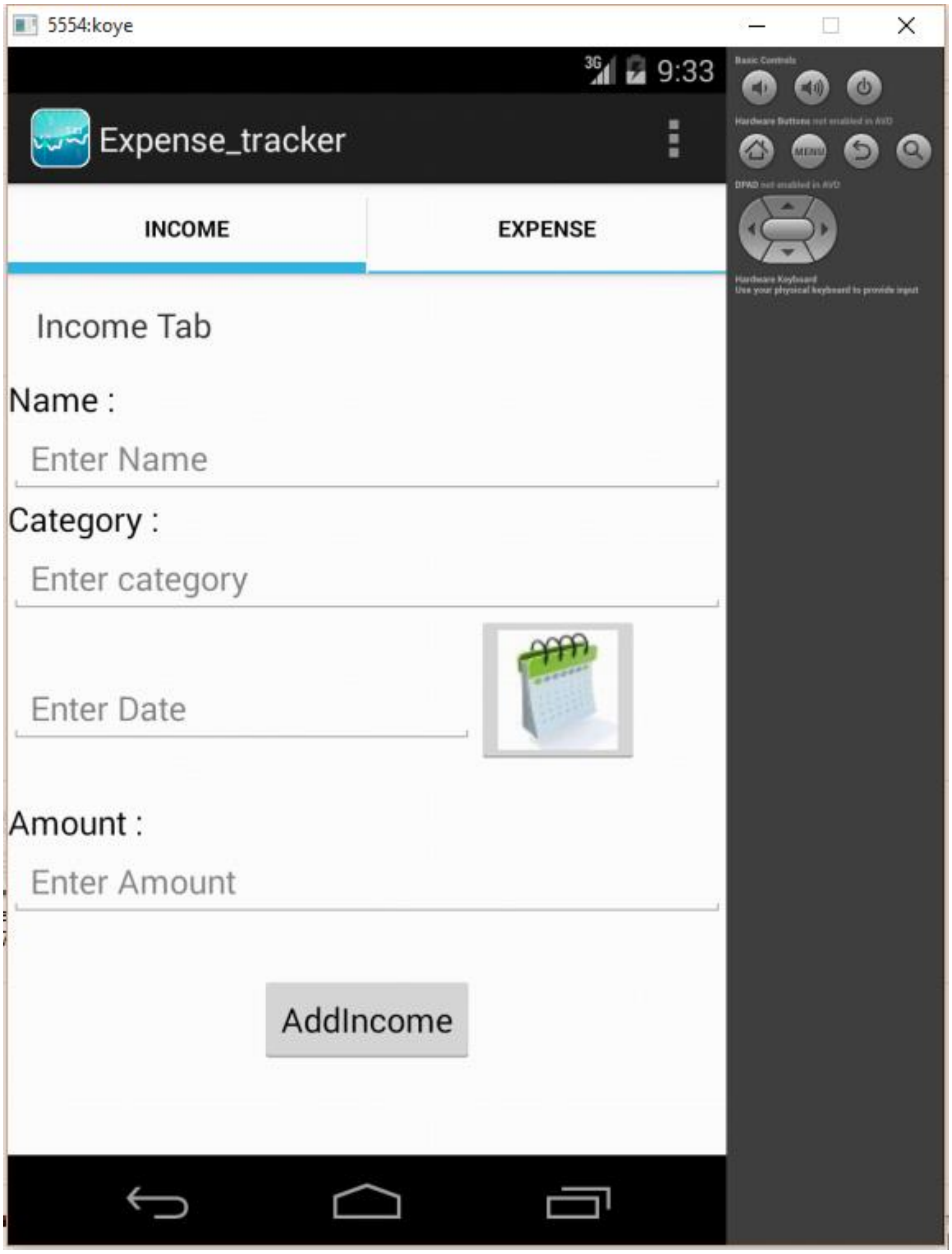
Total Expense : 0

Current Balance : 0

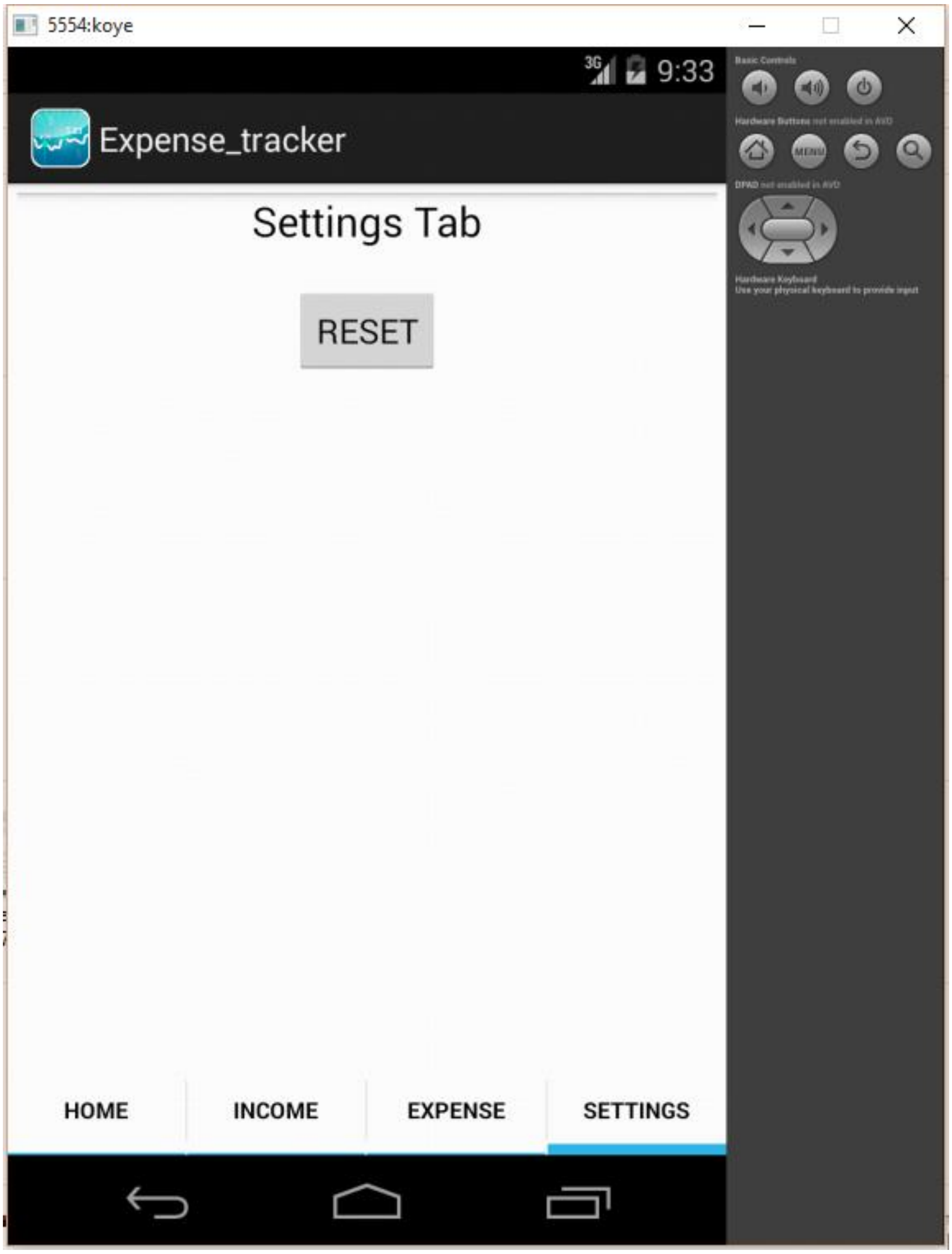
Mockup of dashboard



Mockup of Add Expense (Manual)



Mockup of Add Income Tab



Mockup of Settings Tab

The above images show the design of the real view of the dashboard, add income tab, and add expense tab and settings.

Reference

[1] http://www.nytimes.com/2014/01/04/your-money/household-budgeting/review-apps-to-track-income-and-expenses.html?_r=0

[2] https://itunes.apple.com/us/app/ispending-expense-tracker/id484100875?mt=8

[3] https://play.google.com/store/apps/details?id=com.expensemanager

[4] https://docs.google.com/spreadsheets/d/1vBMJ4gp22y1JzebvTcwkDS1r9Aals2H7H-ml_wznHSg/edit#gid=1973386343

[5] https://kanbantool.com/

[6] https://github.com/pardha5/Expense-Tracker

[7] http://www.androidhive.info/2014/07/android-speech-to-text-tutorial/

[8] https://developers.google.com/chart/?hl=en

[9] https://developer.android.com/sdk/index.html

[10] https://dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html

CS 5551 Second Increment Report

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.

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.

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.

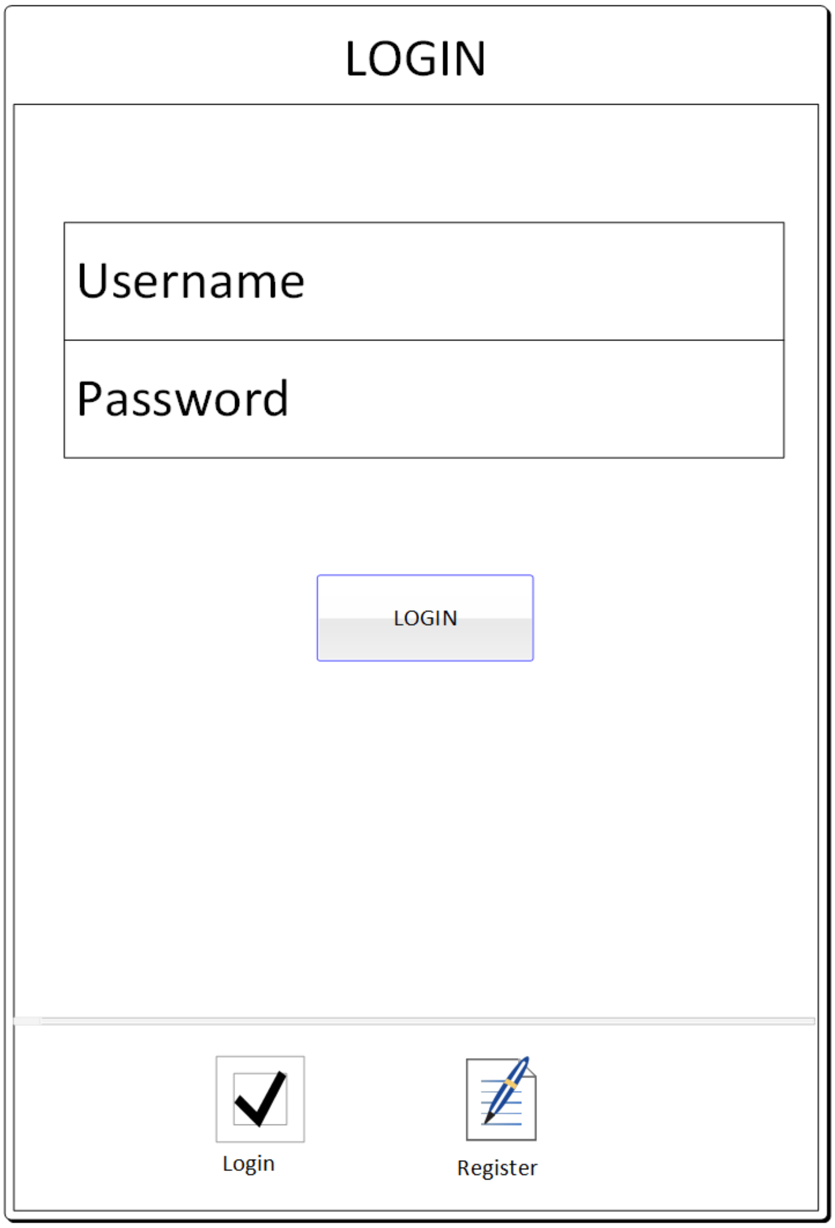
Chart Services [3]: At first during the first increment we tried to use google chart api [4] for developing analytics to our expense tracker application but after introduction of ionic framework, we used chart.js to develop bar and line charts.

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.

Detail Design

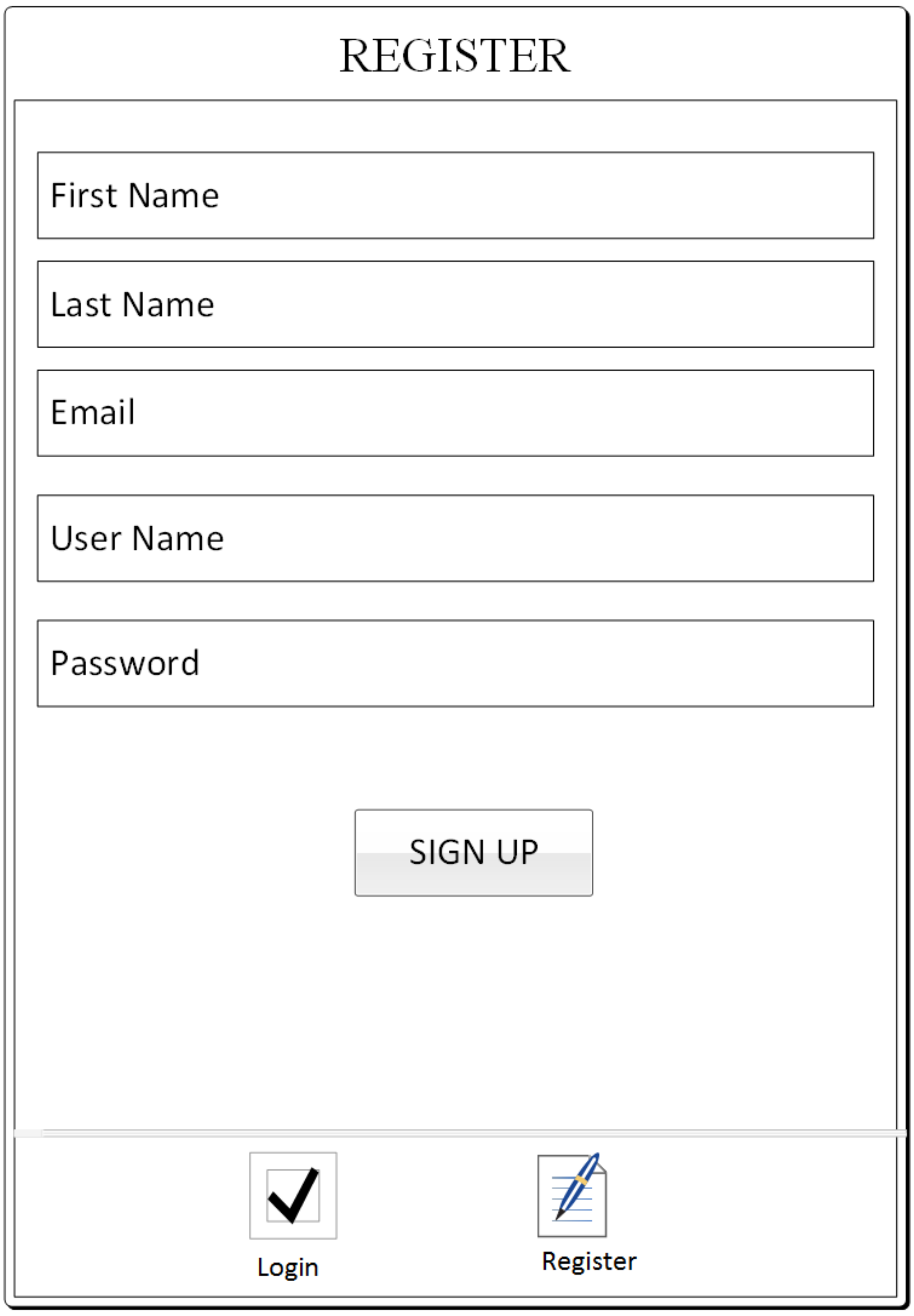
Wireframes

1) Login page for Expenses Tracker



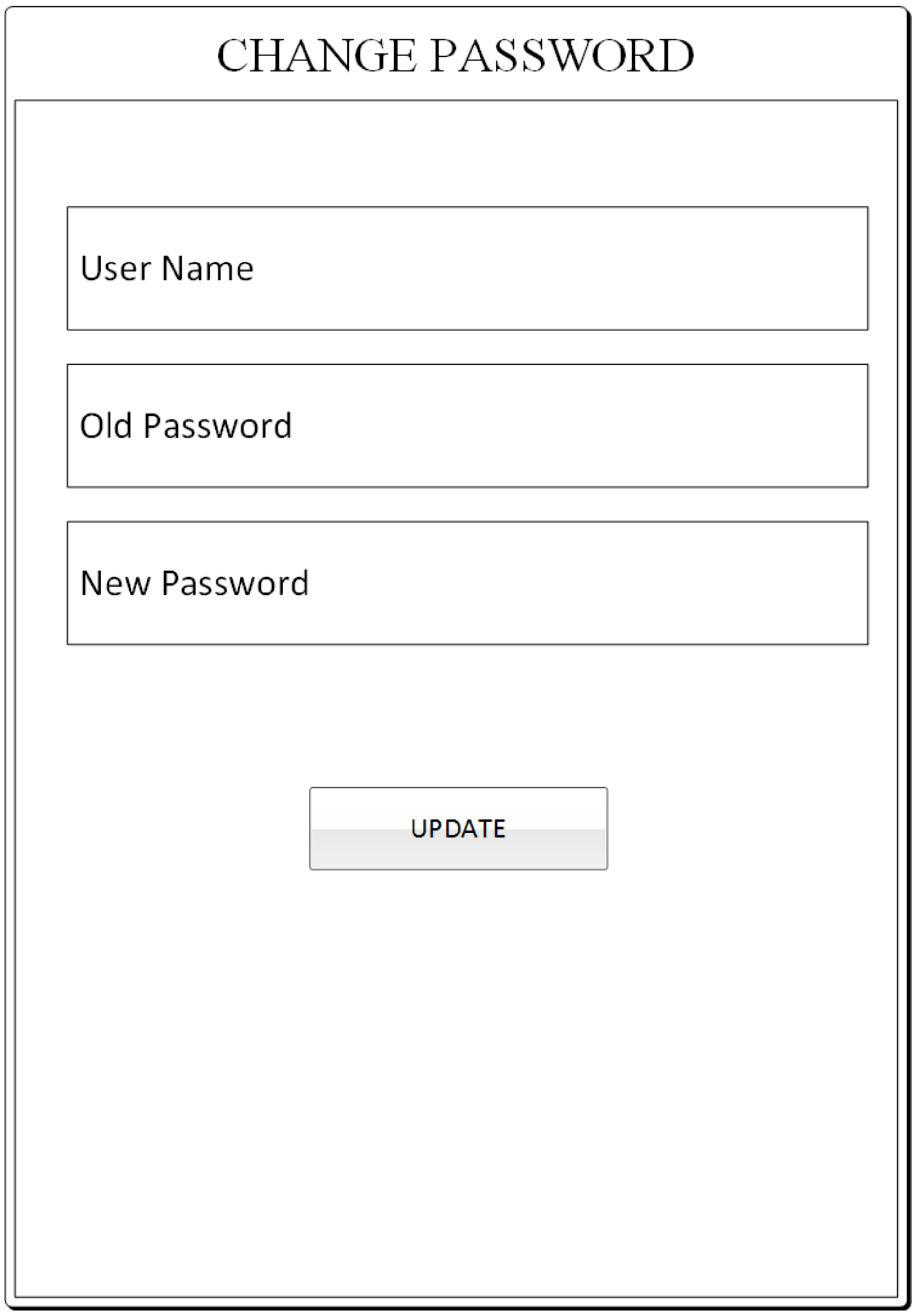
We have designed a basic login page for the users to access their account in a secure way. The details of the users are stored in the mongo lab database.

2) Registration page for Expenses Tracker



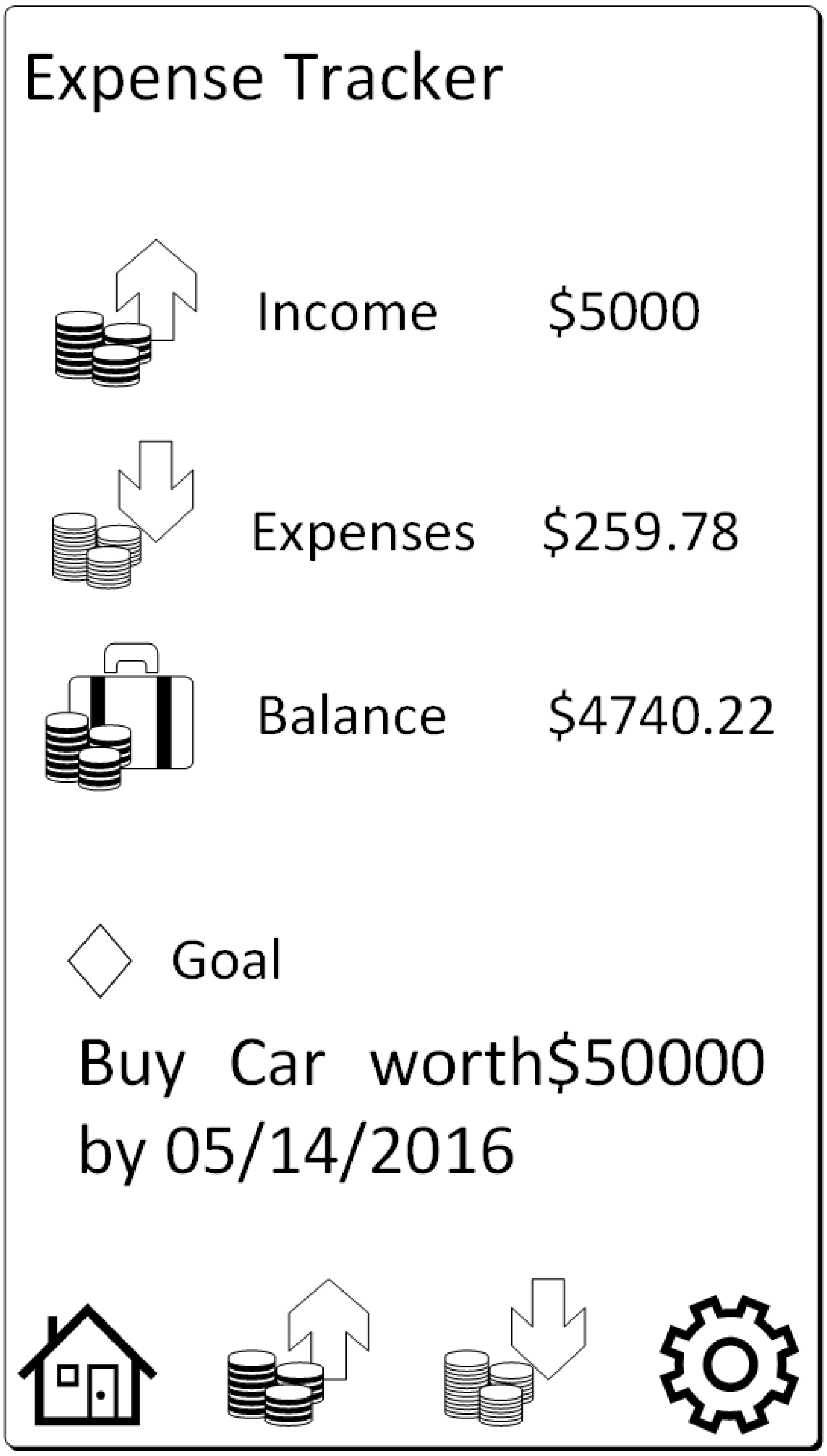
We have designed a basic registration page for the users to create their own account to store all the details of income and daily expenses in a secure way. So that they can access their data where they want.

3) Password update page for Expenses Tracker



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


Expense Tracker

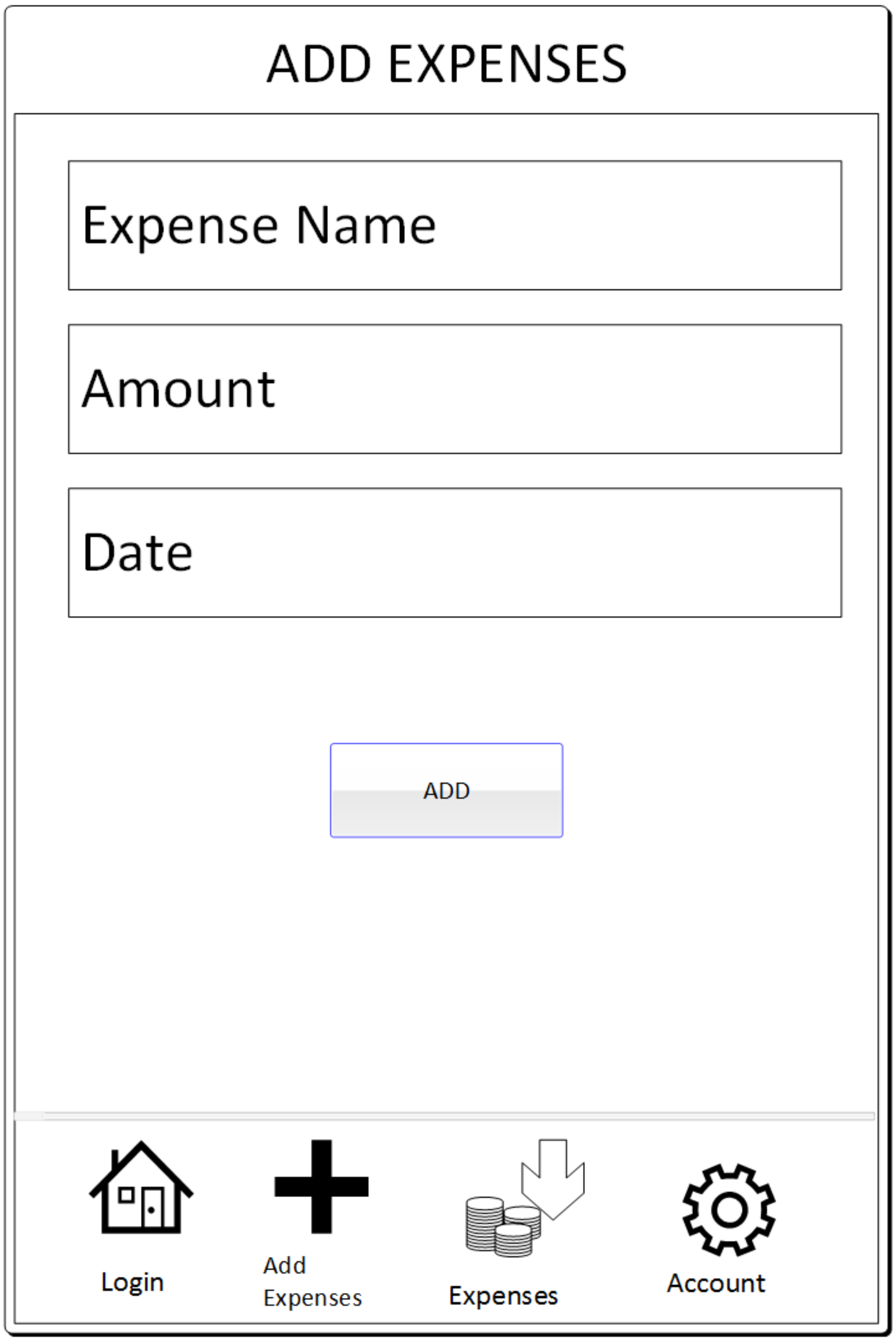
Income	$5000
Expenses	$259.78
Balance	$4740.22

Goal

Buy Car worth$50000 by 05/14/2016

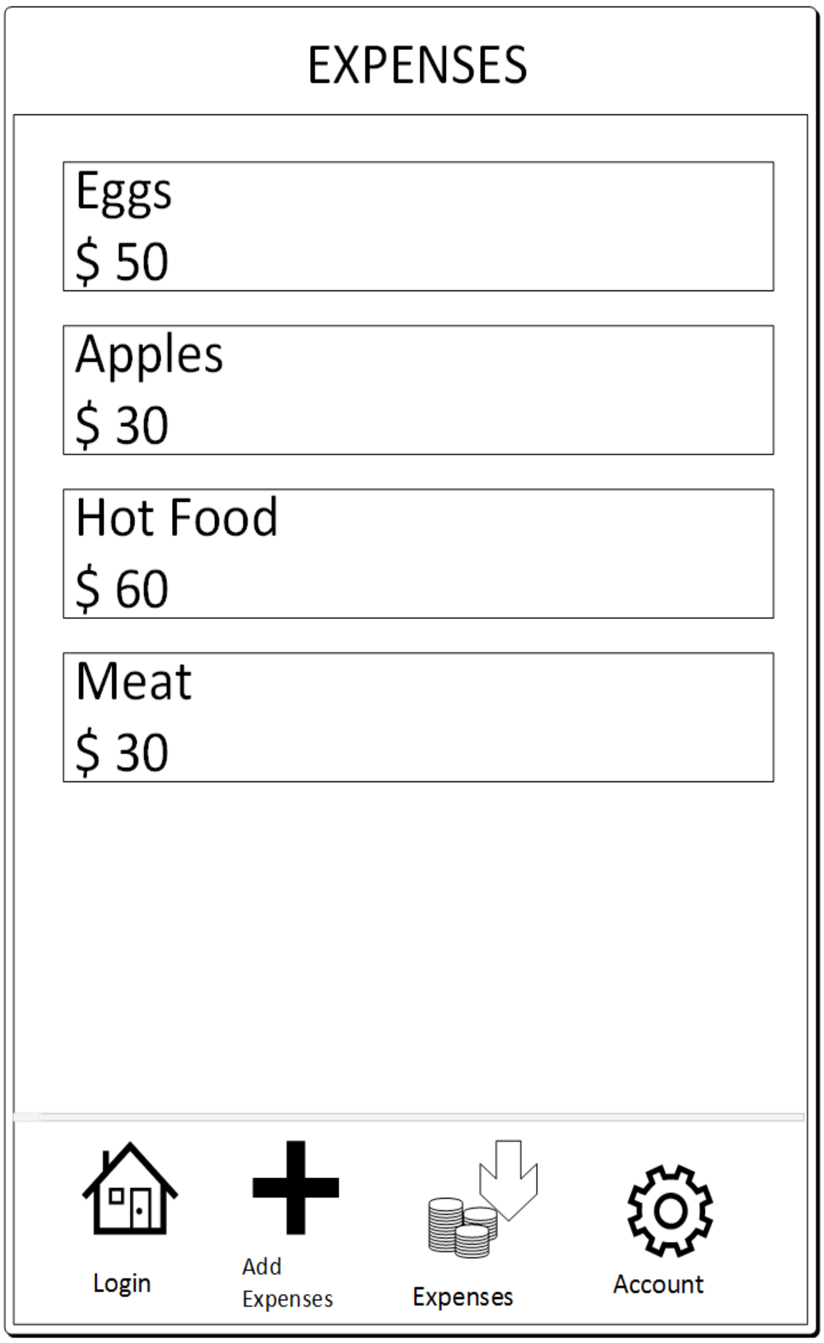
This is the home page where the user can see all is statistics briefly.

5) Add Expenses page for Expenses Tracker



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.

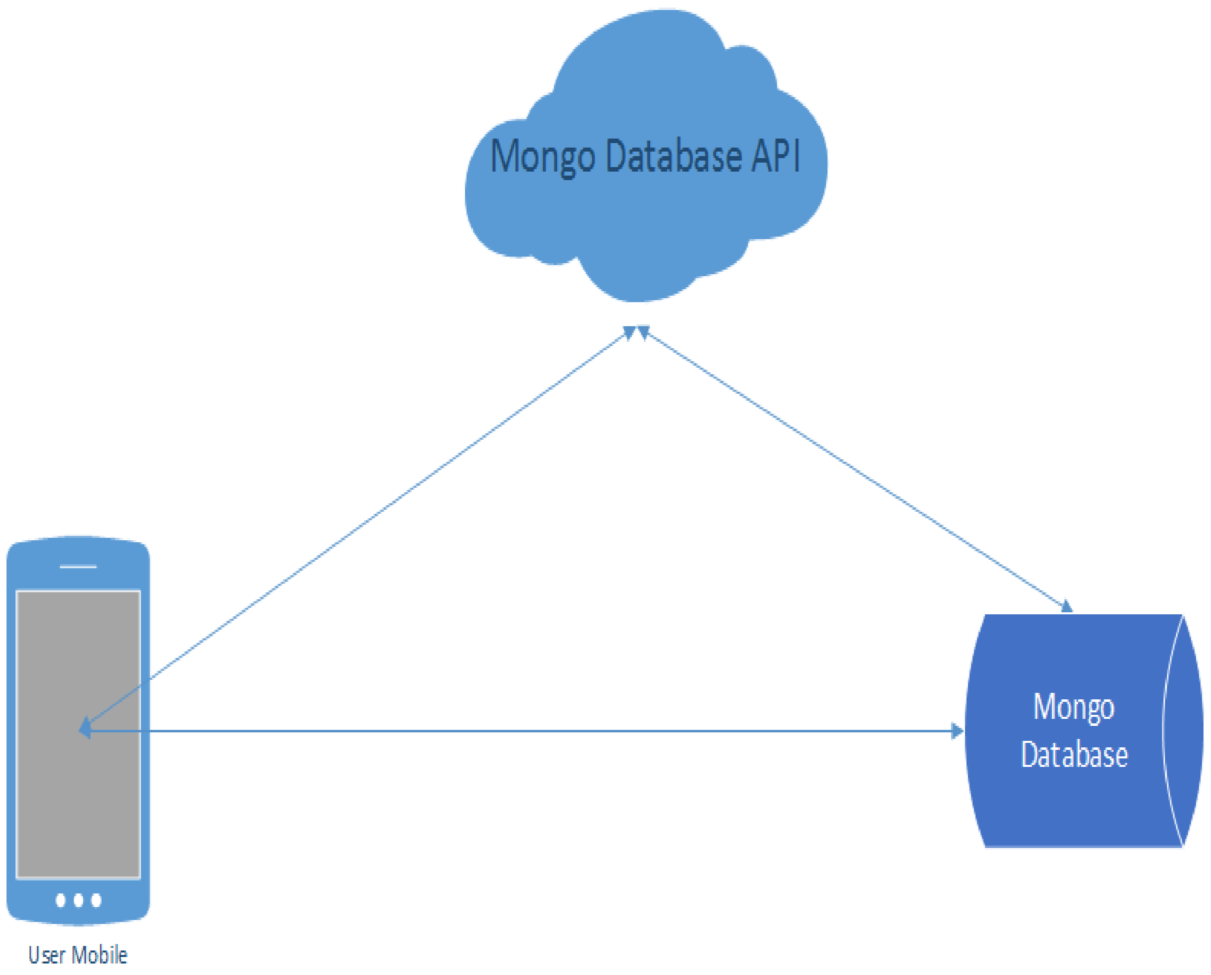
6) Expenses page for Expenses Tracker



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

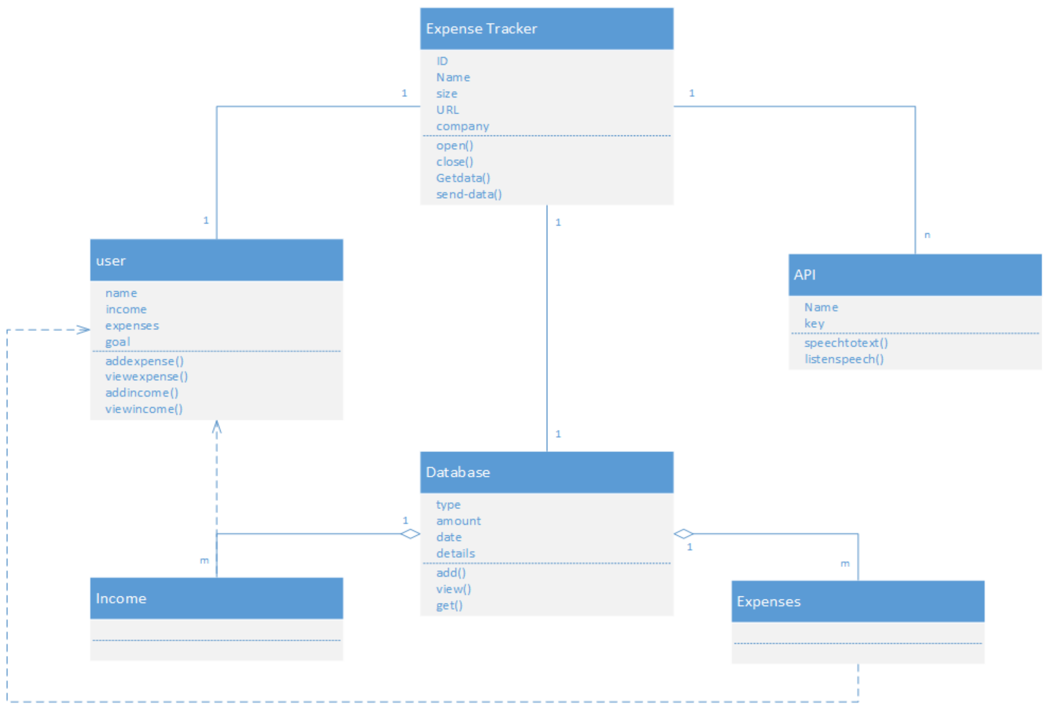
This shows the system architecture of the expense tracker application.



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.

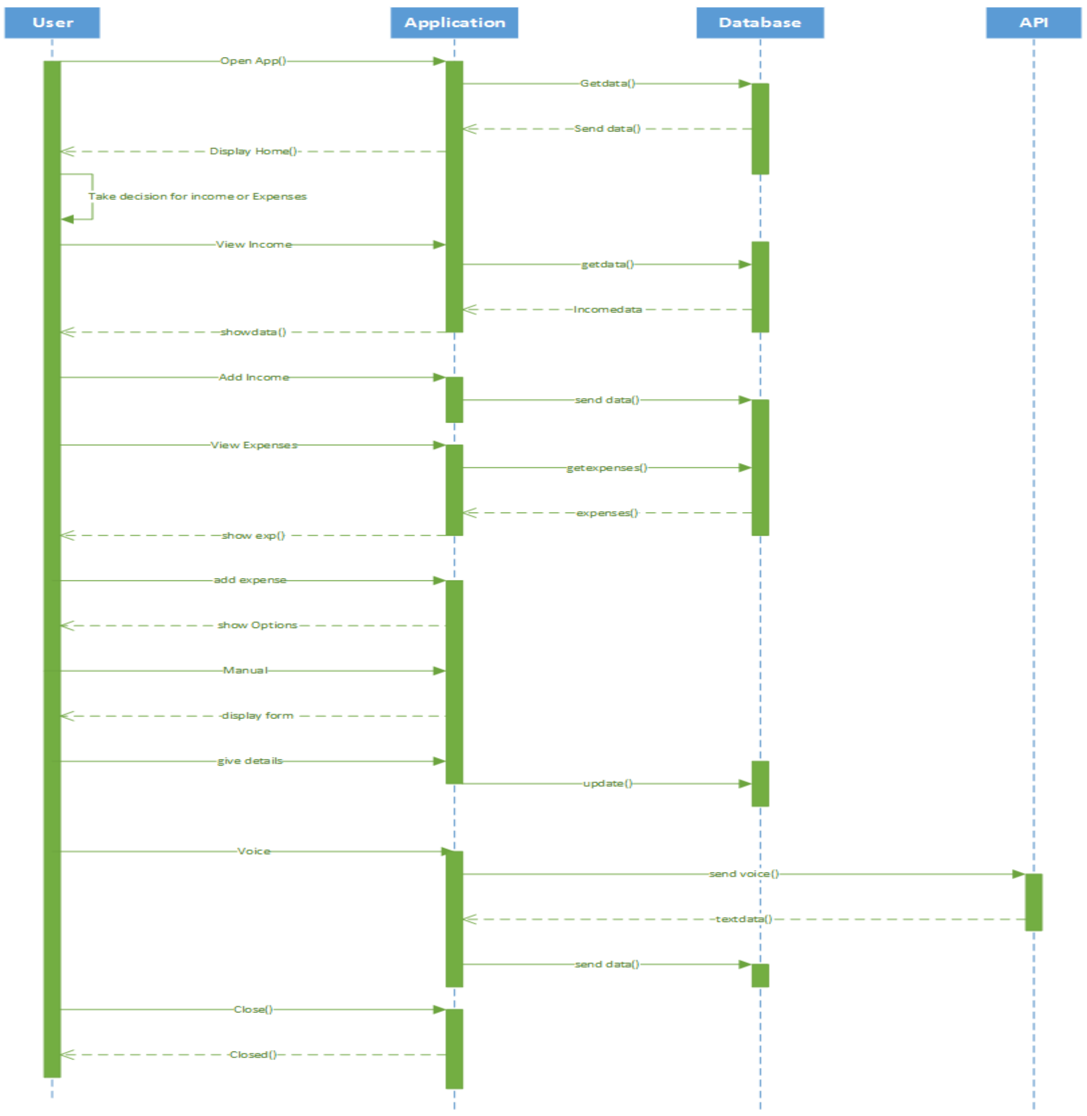
Database is a class in which all the data is stored. Its members are type of entry income or expenses, amount or value of entry, date of entry, details about the entry. Its functions are to add, view, and get the entries to the database and from the database.

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.



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.

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.

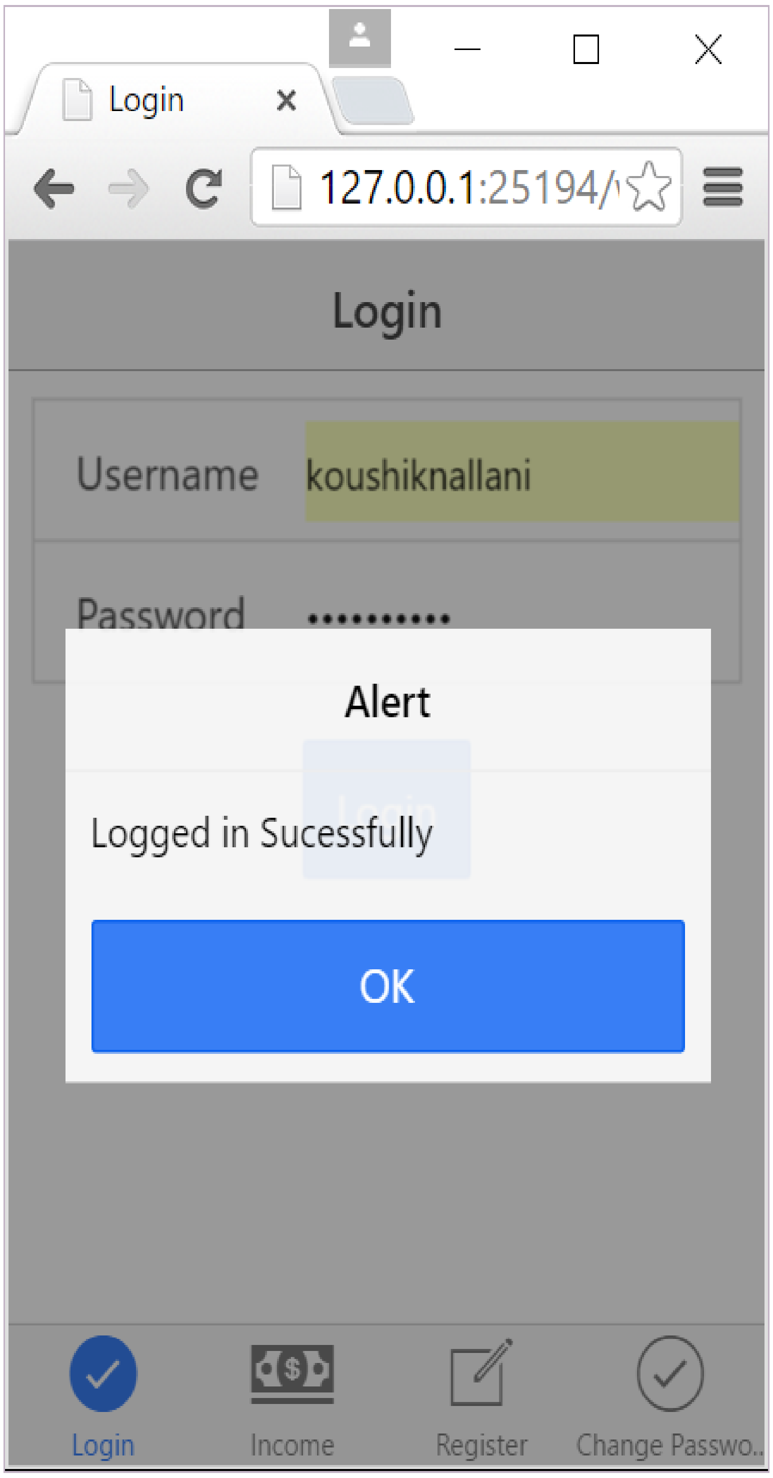
“Expense Tracker” is the database name in which it has a set of collections. The data is stored in the form of JSON data but for the view of developer data can also be viewed in the form of the table created by developer choice.

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.

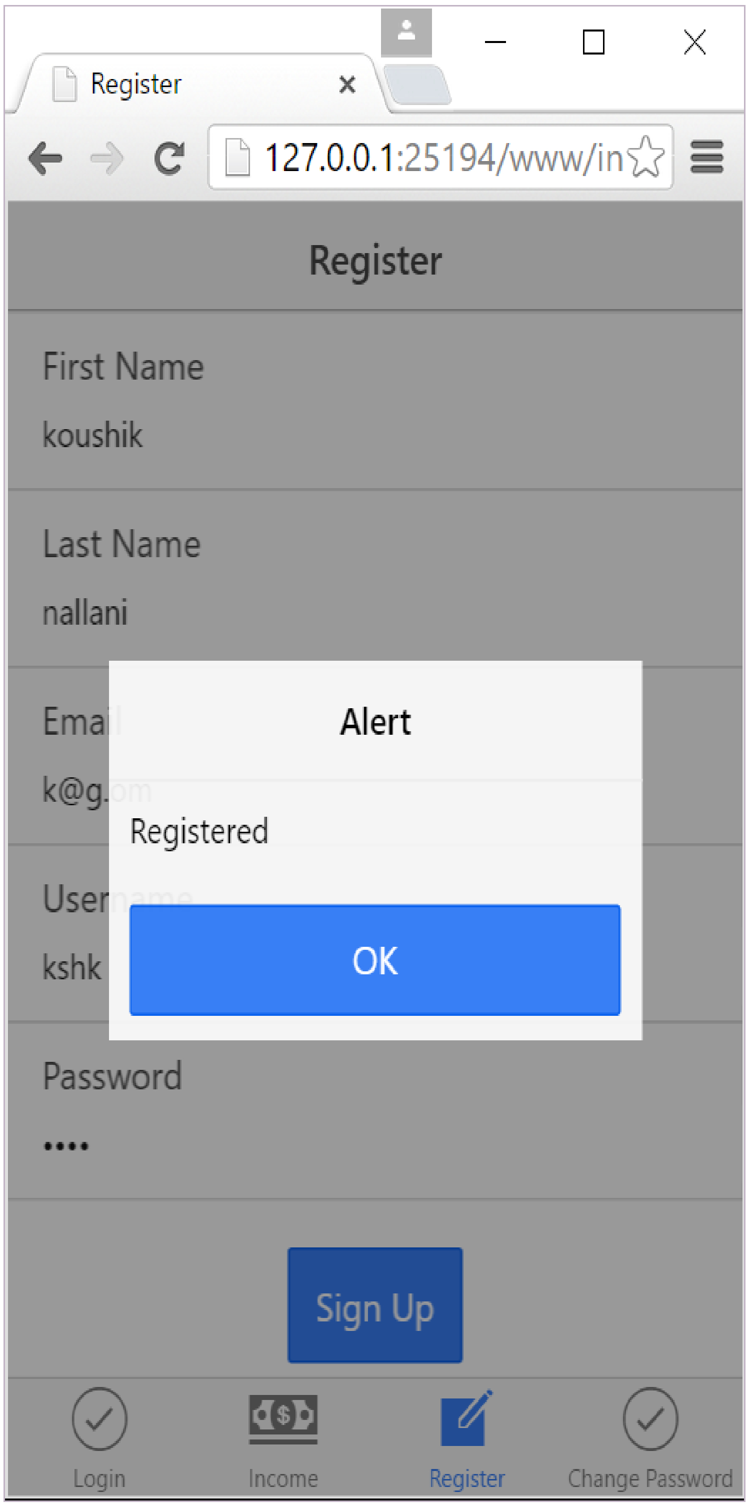
Mobile Client Implementation:

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

••••

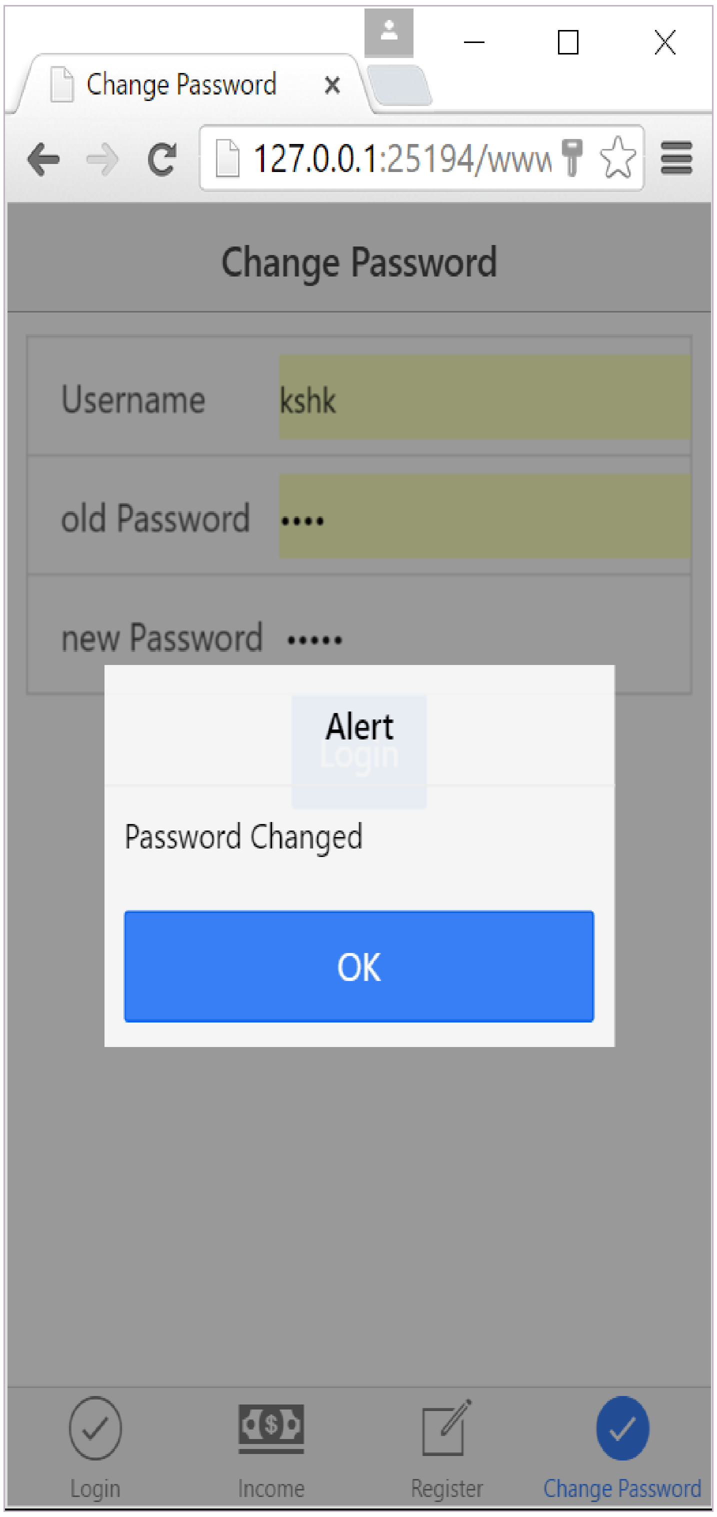
Alert

Registered

OK

Sign Up

Registration



Change Password

Username kshk

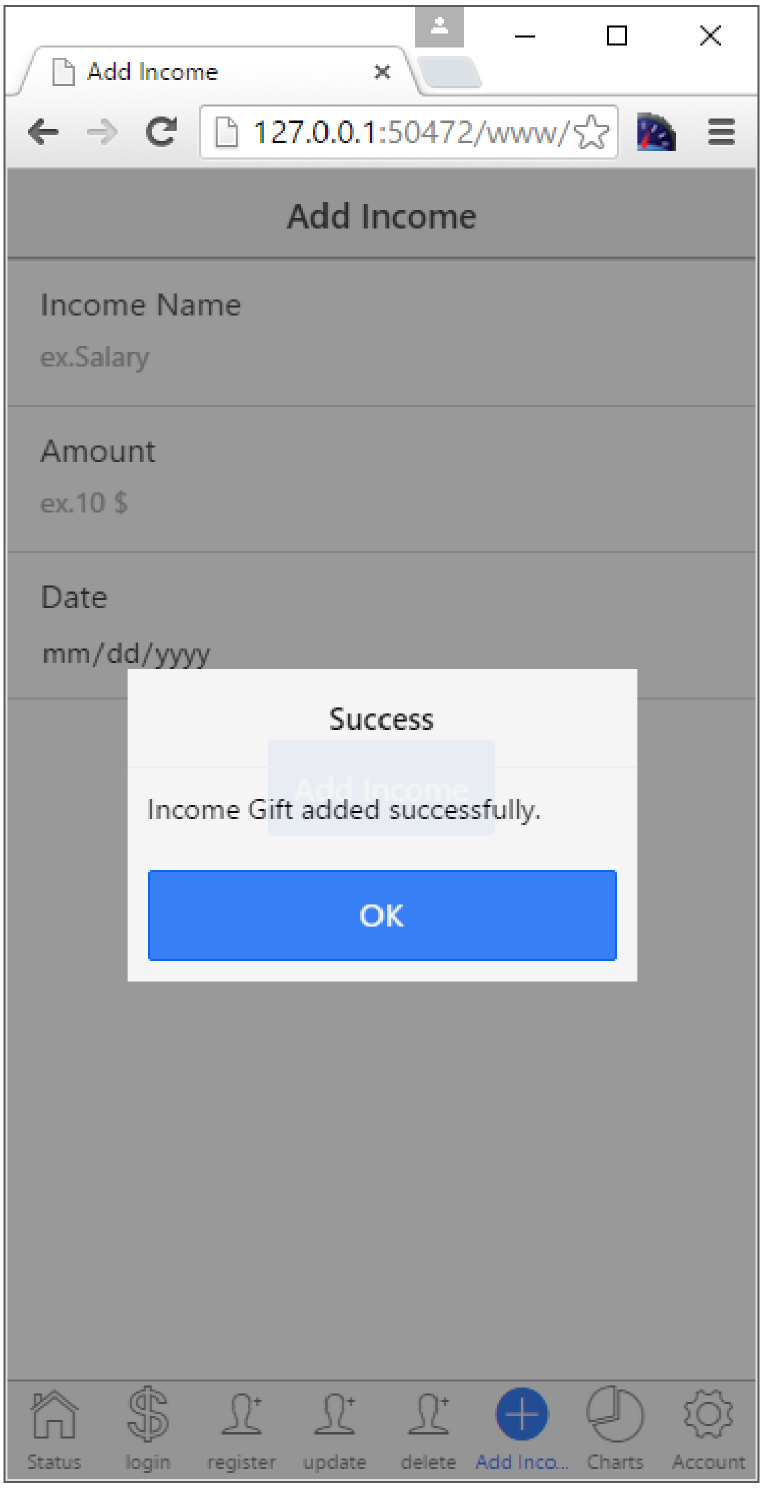
old Password ••••

new Password •••••

Alert

Password Changed

Changing password


Success

Income Gift added successfully.

Adding Income

Deployment

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

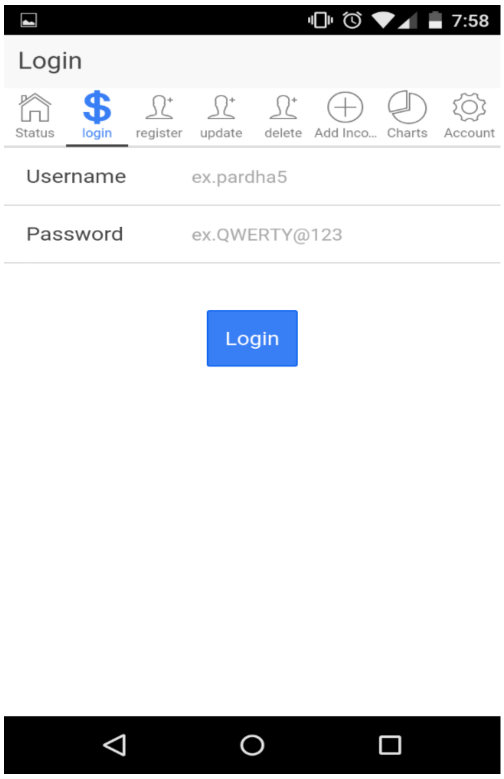
Report

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.

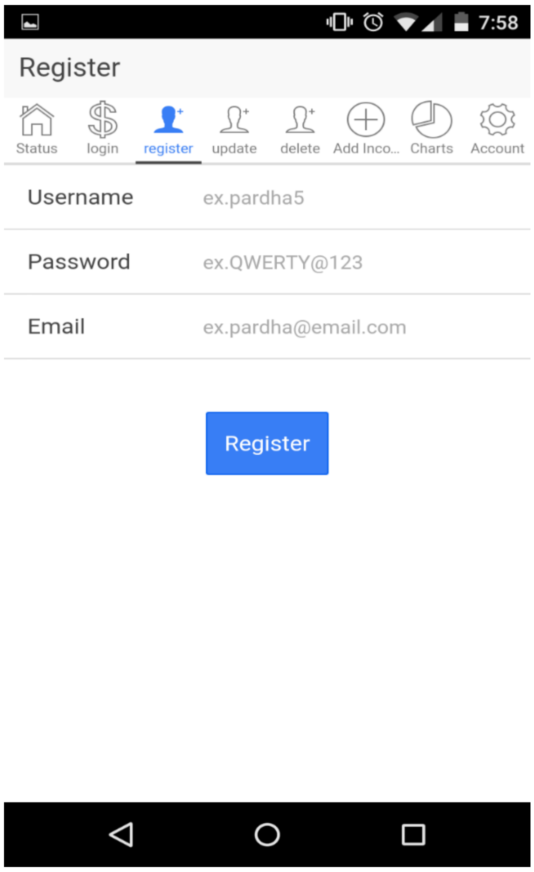
Implementation of the application is done by using ionic frame which is an open source frame work for creating android applications. Ionic frame work uses simple commands to create, build and execute applications. It comes with preloaded components that can be added easily to any application. Total frontend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

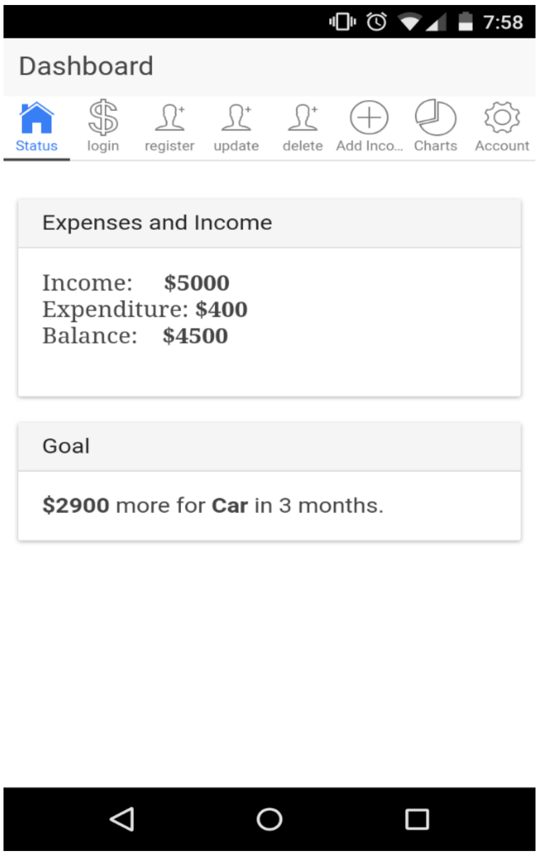
Expense tracker with basic functionalities developed till second increment was uploaded in an android mobile and it is working well. Source code of this project was deployed in github and bluemix.



Login page



Register page



Dashboard

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

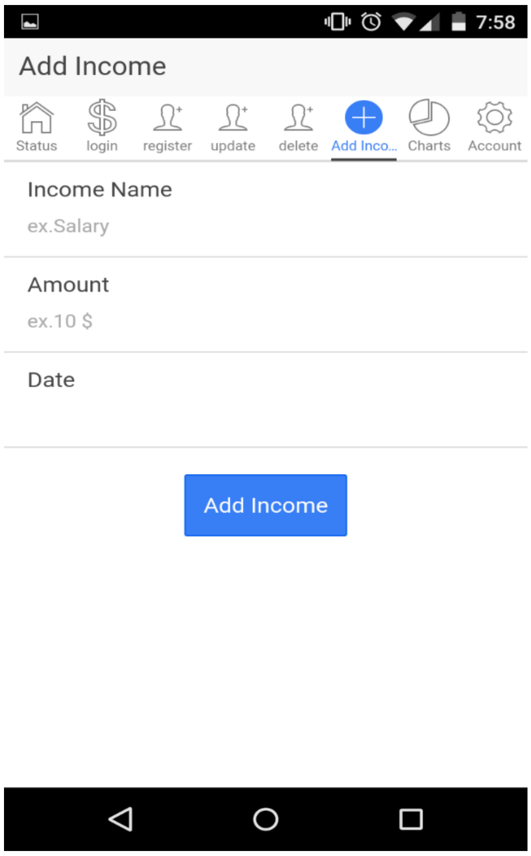
Expenses and Income

Income: **$5000**
Expenditure: **$400**
Balance: **$4500**

Goal

$2900 more for **Car** in 3 months.

Dashboard



Add Income

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

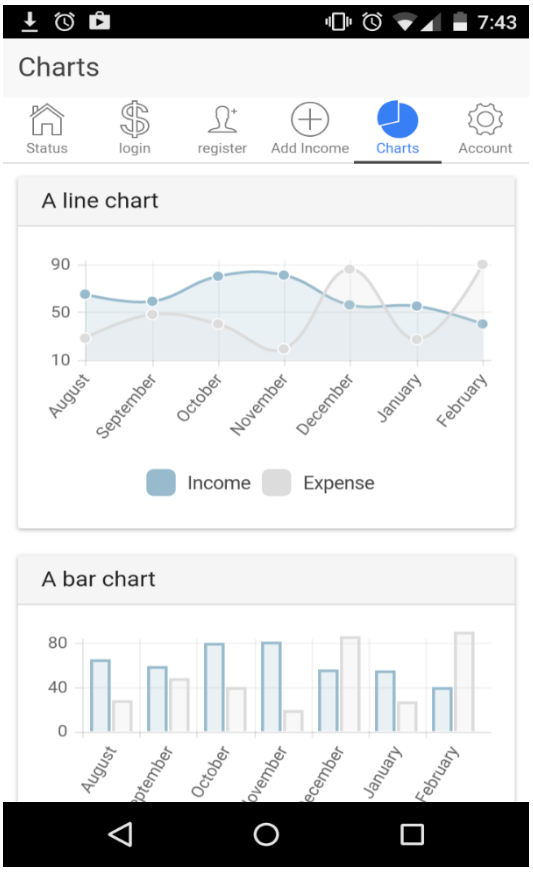
Income Name
ex.Salary

Amount
ex.10 $

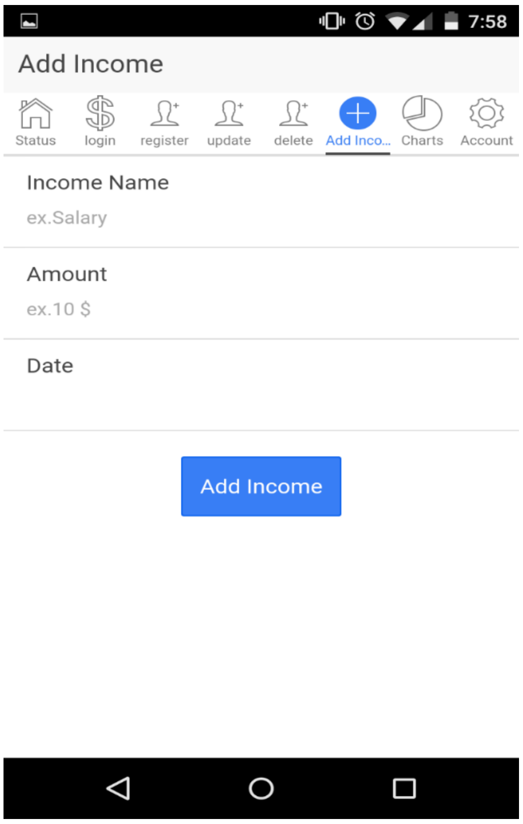
Date

Add Income

Add income page



Charts page

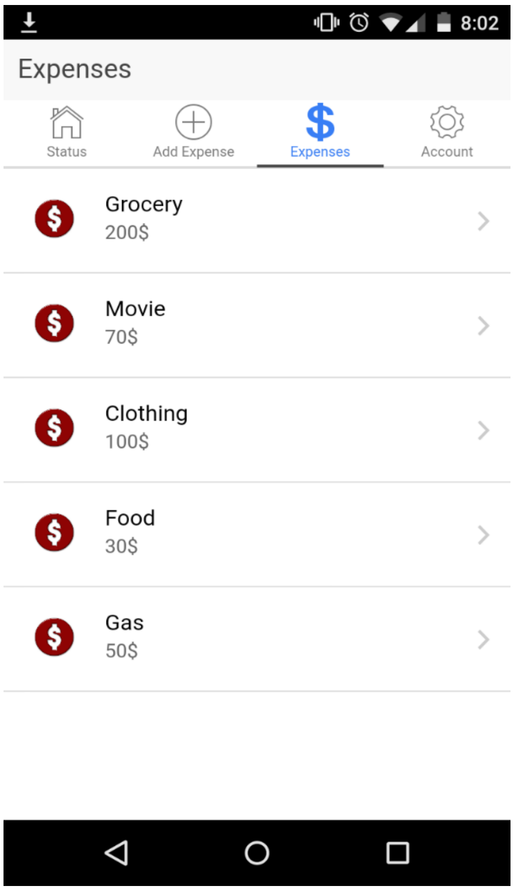


Income Name
ex.Salary

Amount
ex.10 $

Date

Add income page


Expenses

Grocery
200$

Movie
70$

Clothing
100$

Food
30$

Gas
50$

Expense list

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 koye (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.

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

CS 5551 Third Increment Report

Introduction

This document is the summary of work done by us for third increment of our android project expense tracker. In the third increment we have developed the basic working version of expense tracker that interacts with the mongo DB database. Details of the second increment are available in second increment report [1] submitted by us earlier.

For this increment, we mainly focused on enhancing the basic functionalities of the expense tracker integrated with database. These basic functionalities include user interface design, login, registration, adding incomes, adding expenses, charts and goal settings. In addition to these we have also implemented a simple view of incomes, expenses and goals that are added by the user. In this increment we have also included a new tier to our application which is a server that connects our application to the backend database.

The database tier of expense tracker was designed by using Mongo DB [2] a NoSQL database system. Database connectivity was done using servlets. Tables like incomes, expenses and users are implemented as collection and are accessed through a server named Web-Sphere. In future increment this server will be replaced by IBM blue mix.

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.
- Users can add goals.
- Addition of new server tier to interact with database.

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, chart services which is provided by ionic and we are also working on the speech to text api for future increments.

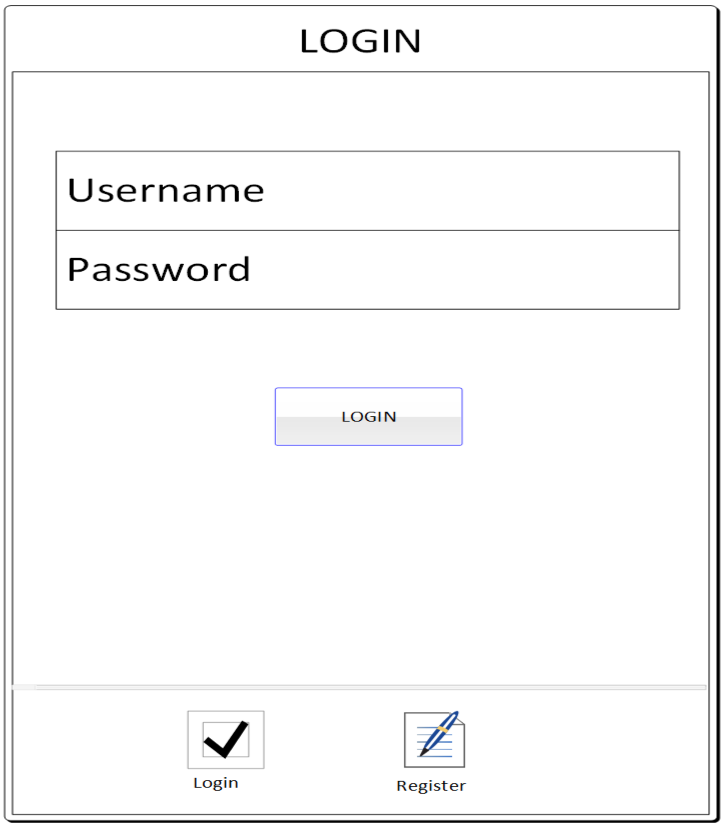
Chart Services [3]: At first during the first increment we tried to use google chart api [4] for developing analytics to our expense tracker application but after introduction of ionic framework, we used chart.js to develop bar and line charts.

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.

Detail Design

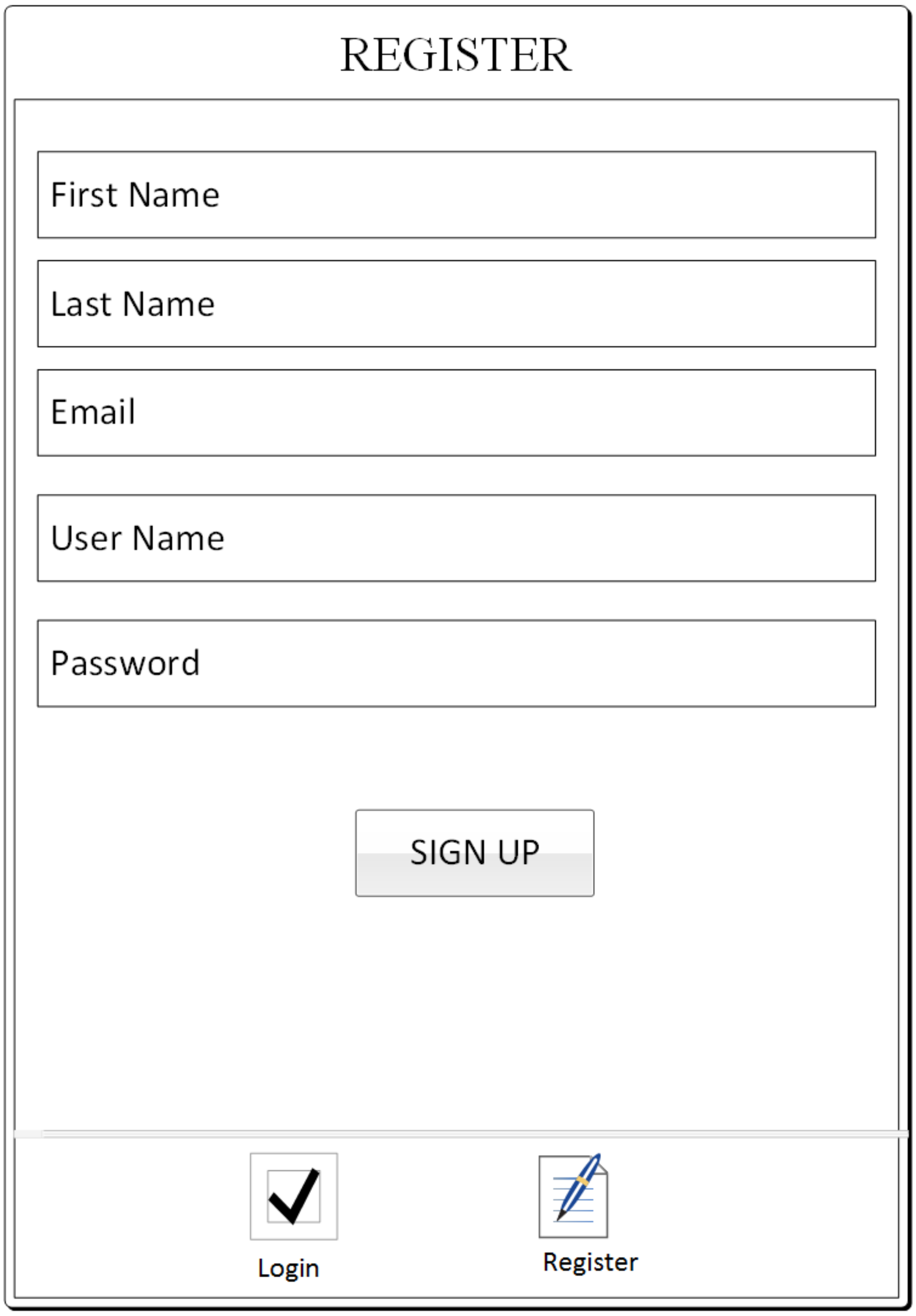
Wireframes

1) Login page for Expenses Tracker



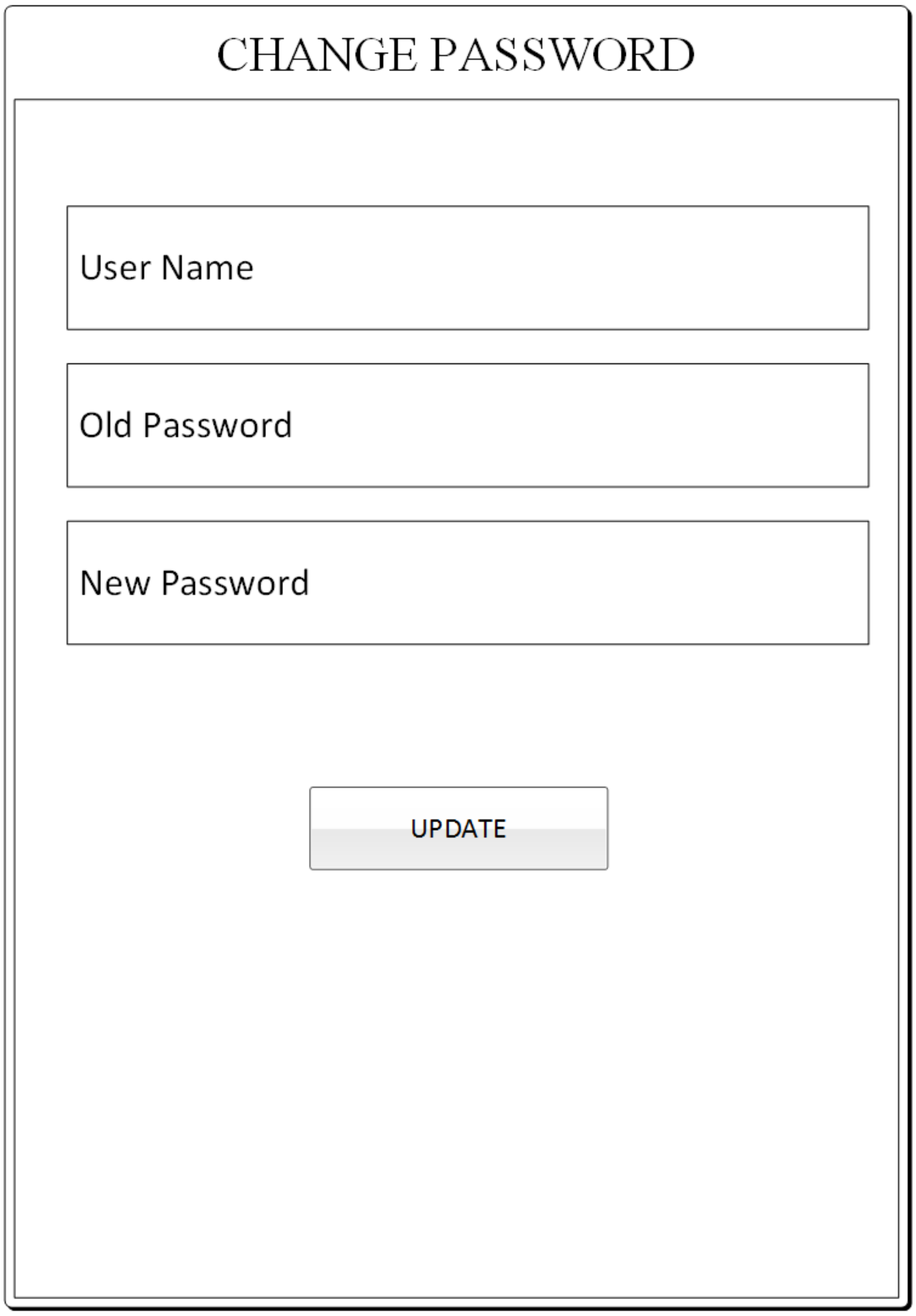
We have designed a basic login page for the users to access their account in a secure way. The details of the users are stored in the mongo lab [6] database.

2) Registration page for Expenses Tracker



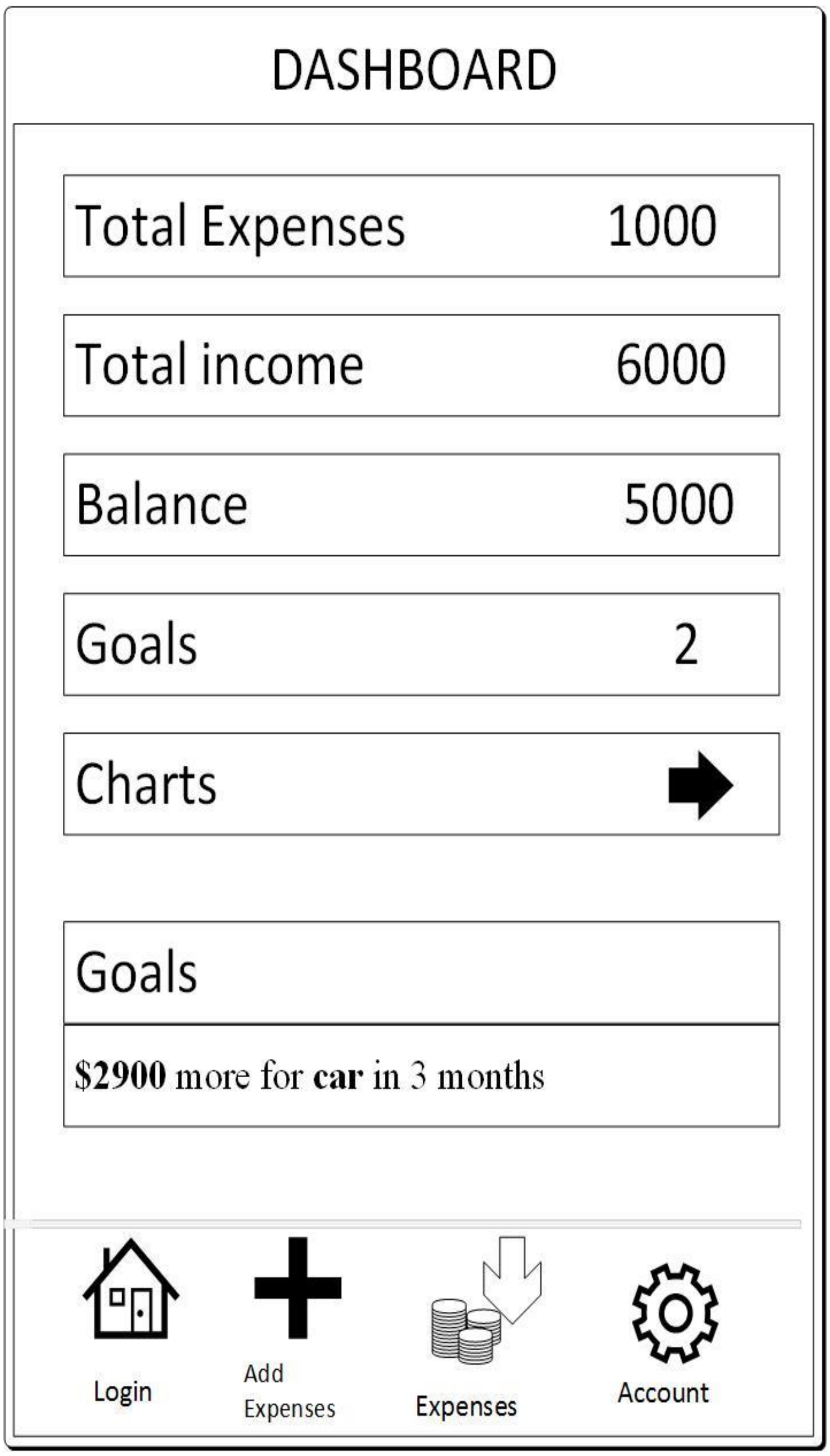
We have designed a basic registration page for the users to create their own account to store all the details of income and daily expenses in a secure way. So that they can access their data where they want.

3) Password update page for Expenses Tracker



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

4) Home or dash board page for Expenses Tracker



DASHBOARD

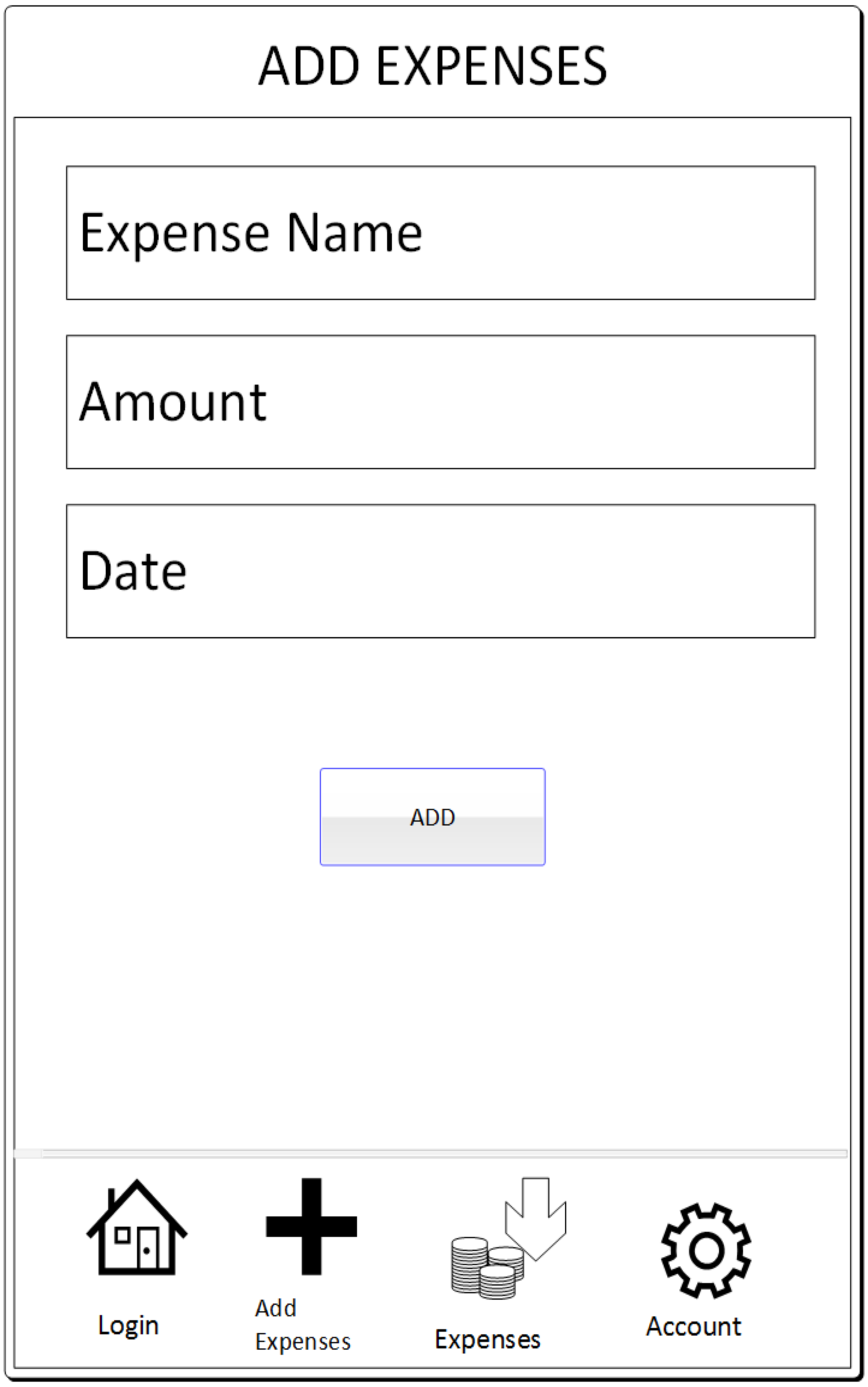
Total Expenses	1000
Total income	6000
Balance	5000
Goals	2
Charts	➡

Goals

$2900 more for **car** in 3 months

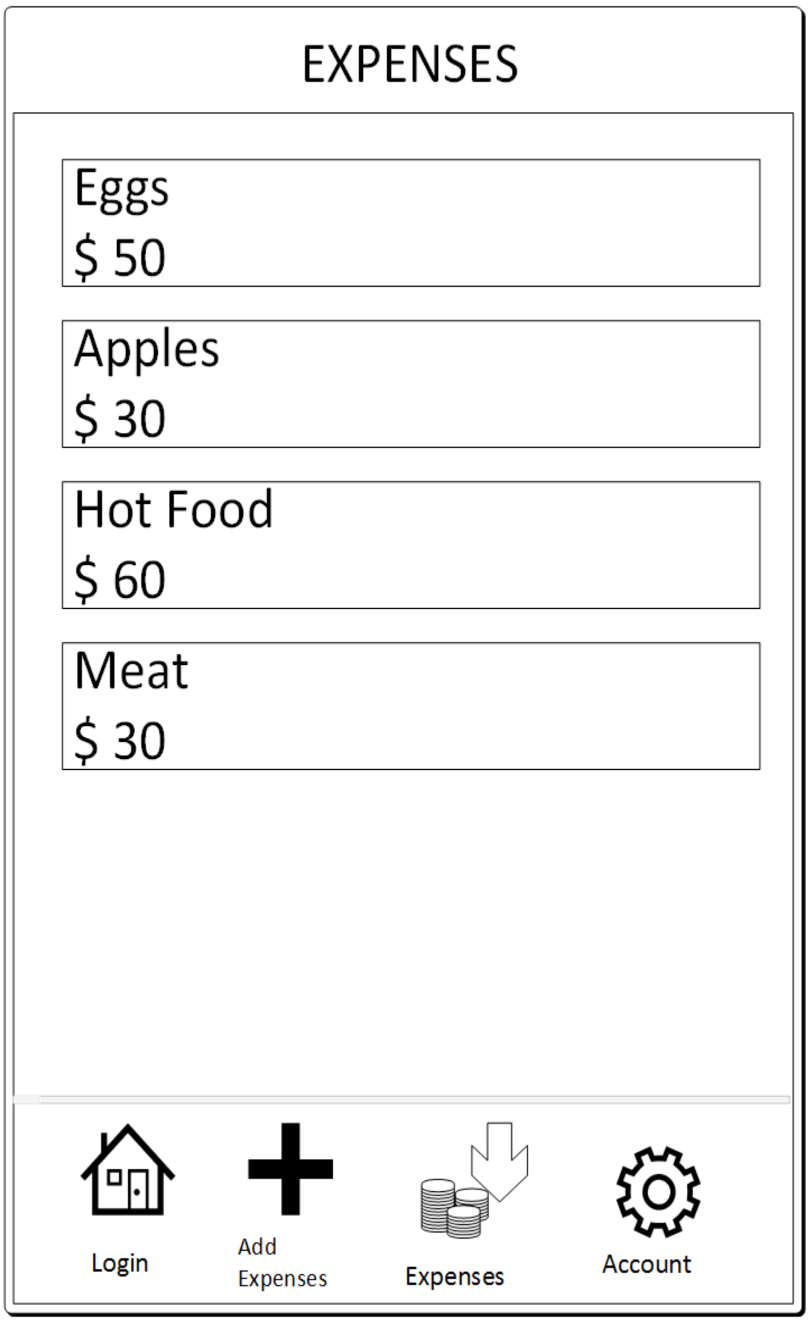
This is the home page where the user can see all is statistics briefly.

5) Add Expenses page for Expenses Tracker



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.

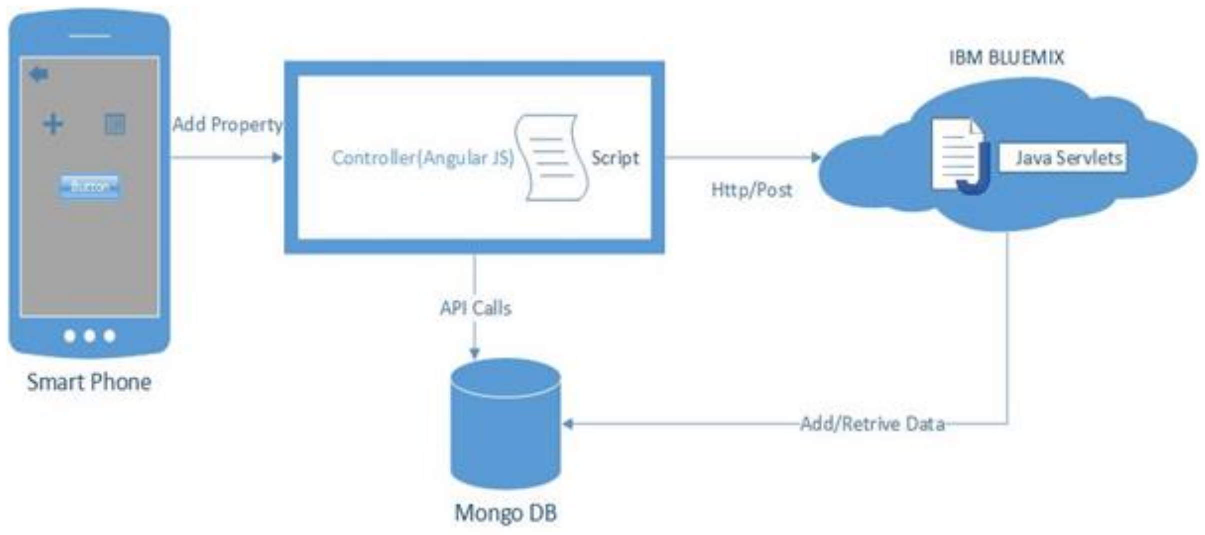
6) Expenses page for Expenses Tracker



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

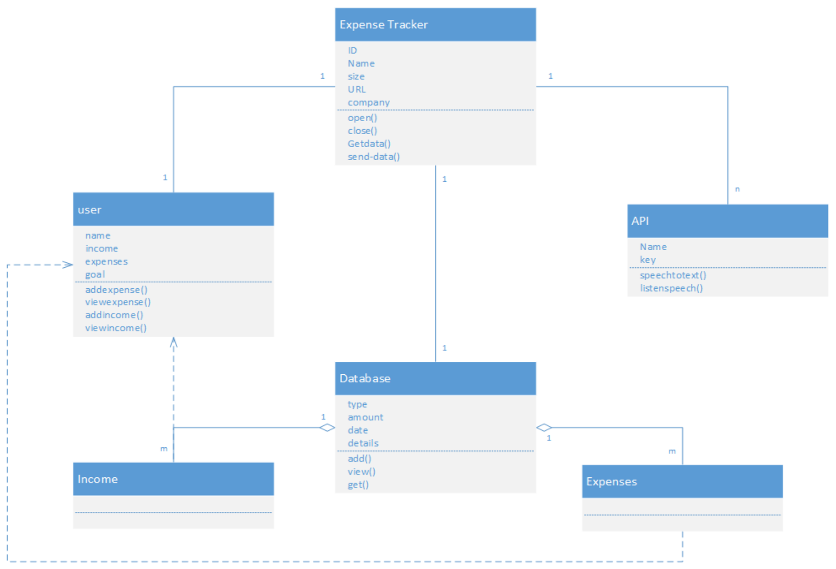
This shows the system architecture of the expense tracker application.



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 controller is being called to ease the process. This request processing is done with the help of a program that runs on server. 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.

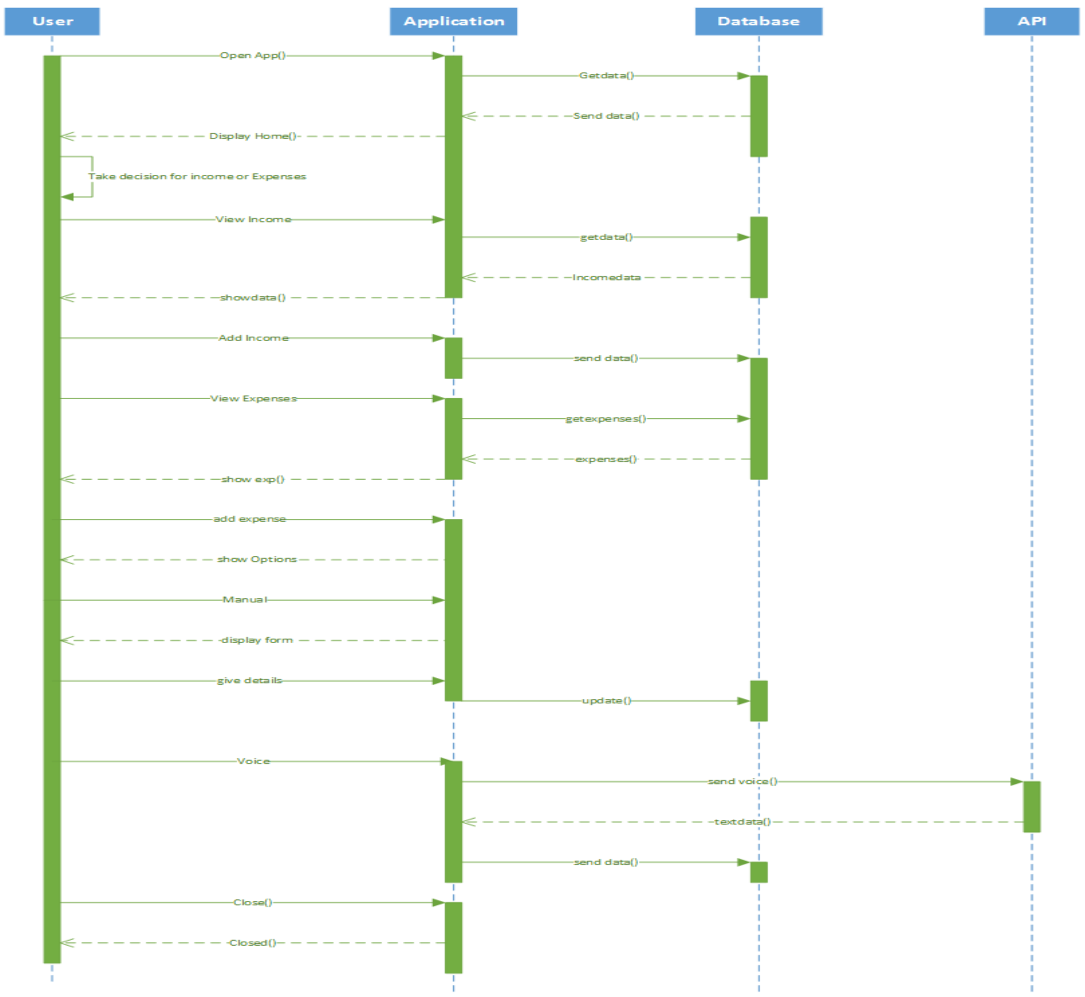
Database is a class in which all the data is stored. Its members are type of entry income or expenses, amount or value of entry, date of entry, details about the entry. Its functions are to add, view, and get the entries to the database and from the database.

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.



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 test cases 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.

Implementation

Server Side Implementation:

For the server side implementation we have used web-sphere that runs on our local machine and processes the request provided by the user. Logic for our application is presented in servlets that are stored in this server and this server is also responsible for interacting with mongo DB for insertion, deletion and updates in the database.

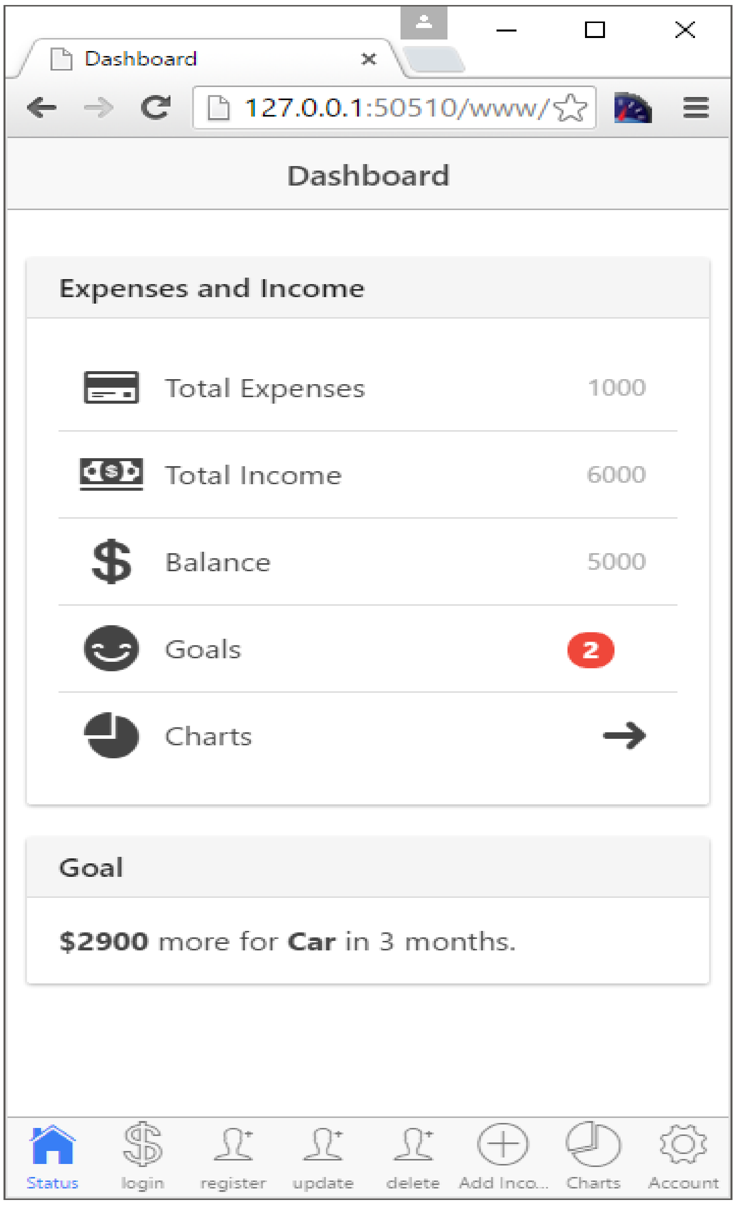
“Expense Tracker” is the database name in which it has a set of collections. The data is stored in the form of JSON data but for the view of developer data can also be viewed in the form of the table created by developer choice.

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 [7] 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.



Dashboard

127.0.0.1:50510/www/

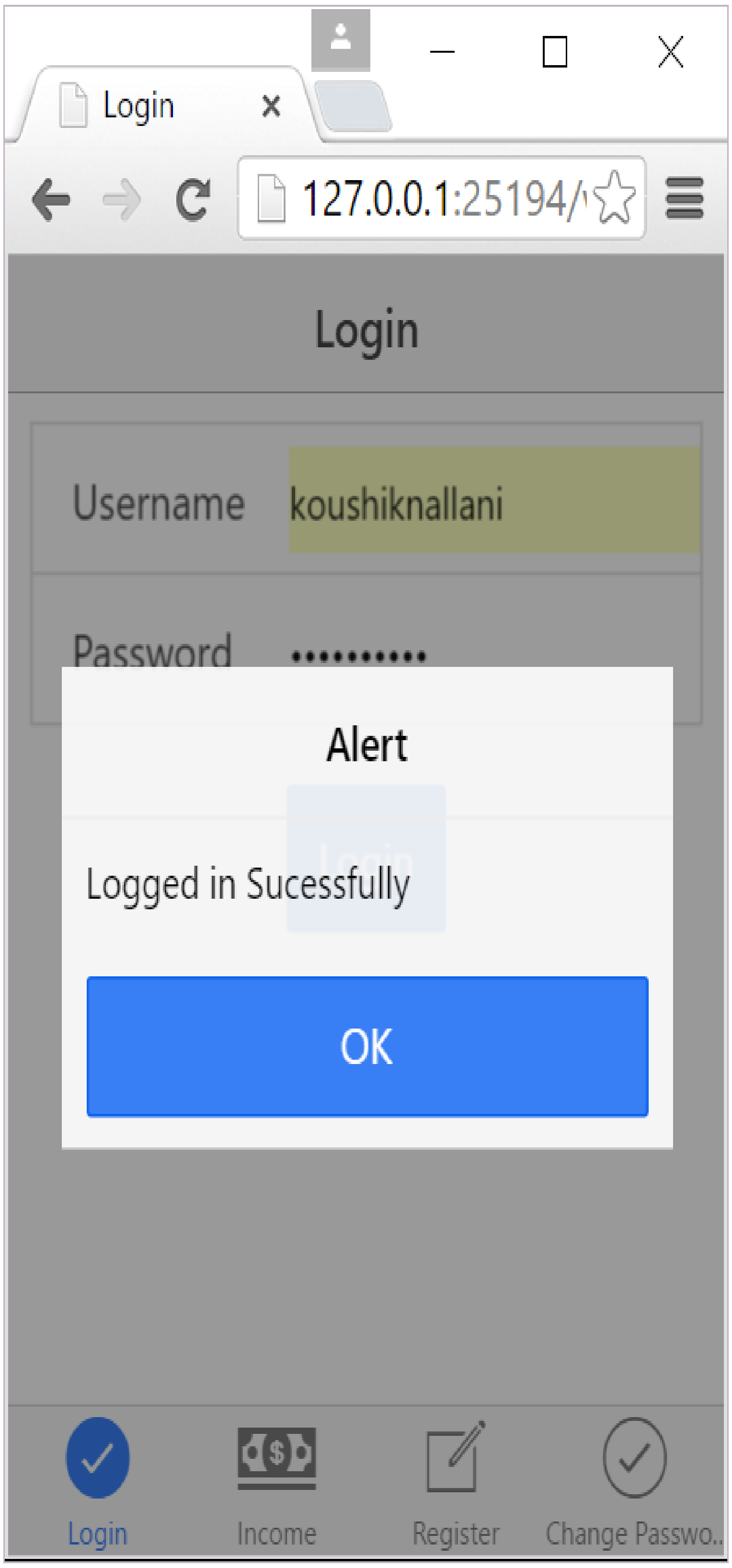
Dashboard

Expenses and Income

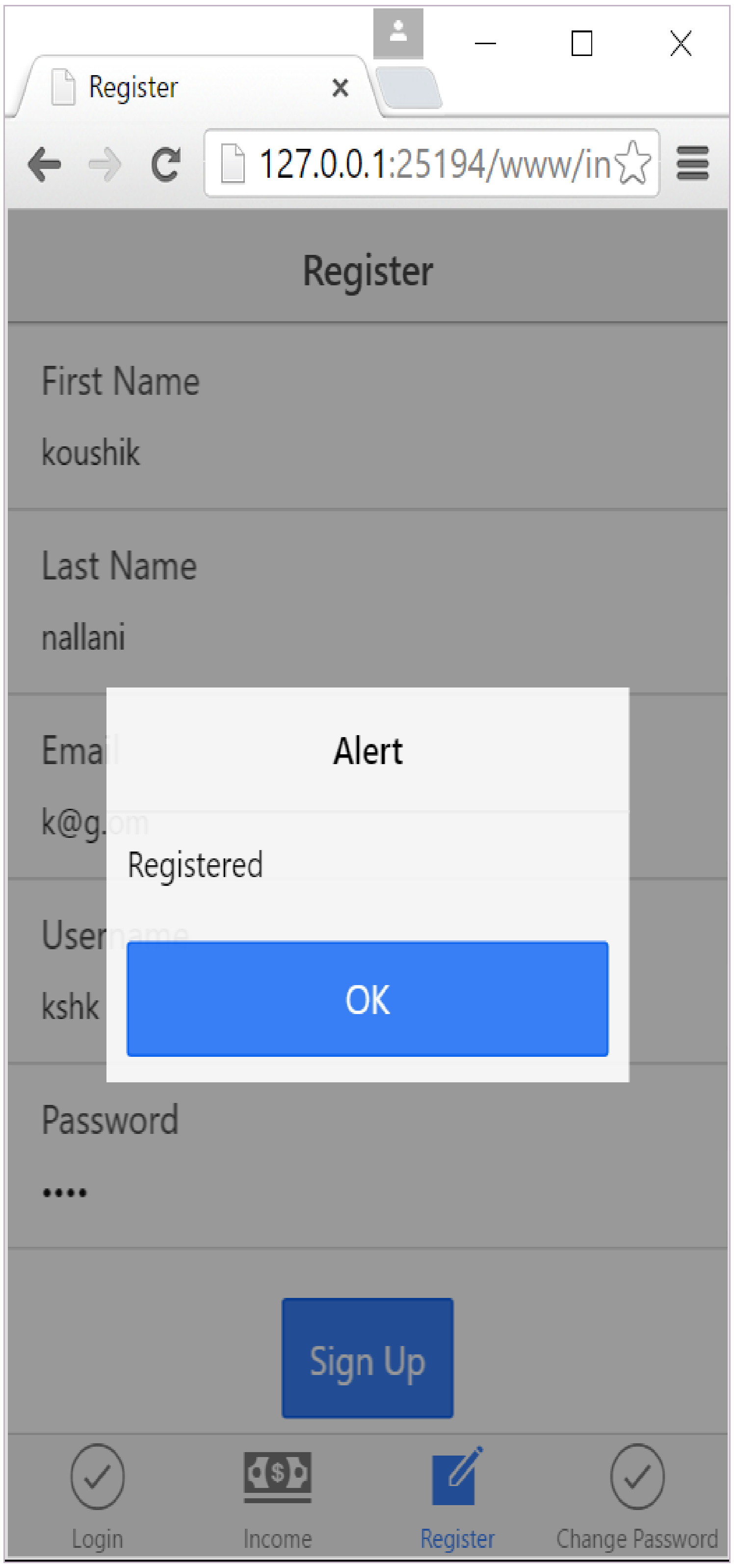
Total Expenses	1000
Total Income	6000
Balance	5000
Goals	2
Charts	→

Goal

$2900 more for **Car** in 3 months.



Login validation



Register

127.0.0.1:25194/www/in

Register

First Name

koushik

Last Name

nallani

Email

k@g.com

Username

kshk

Password

••••

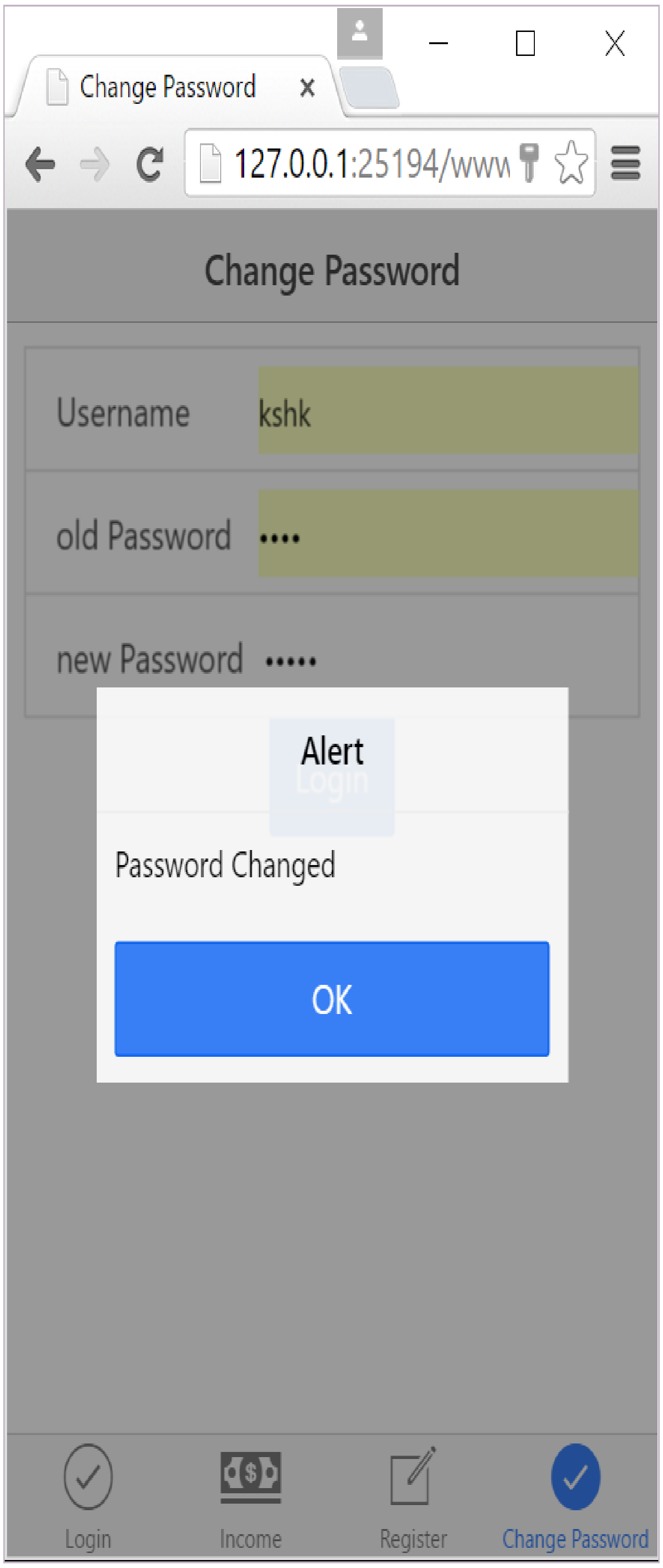
Alert

Registered

OK

Sign Up

Registration


Change Password

127.0.0.1:25194/www

Change Password

Username kshk

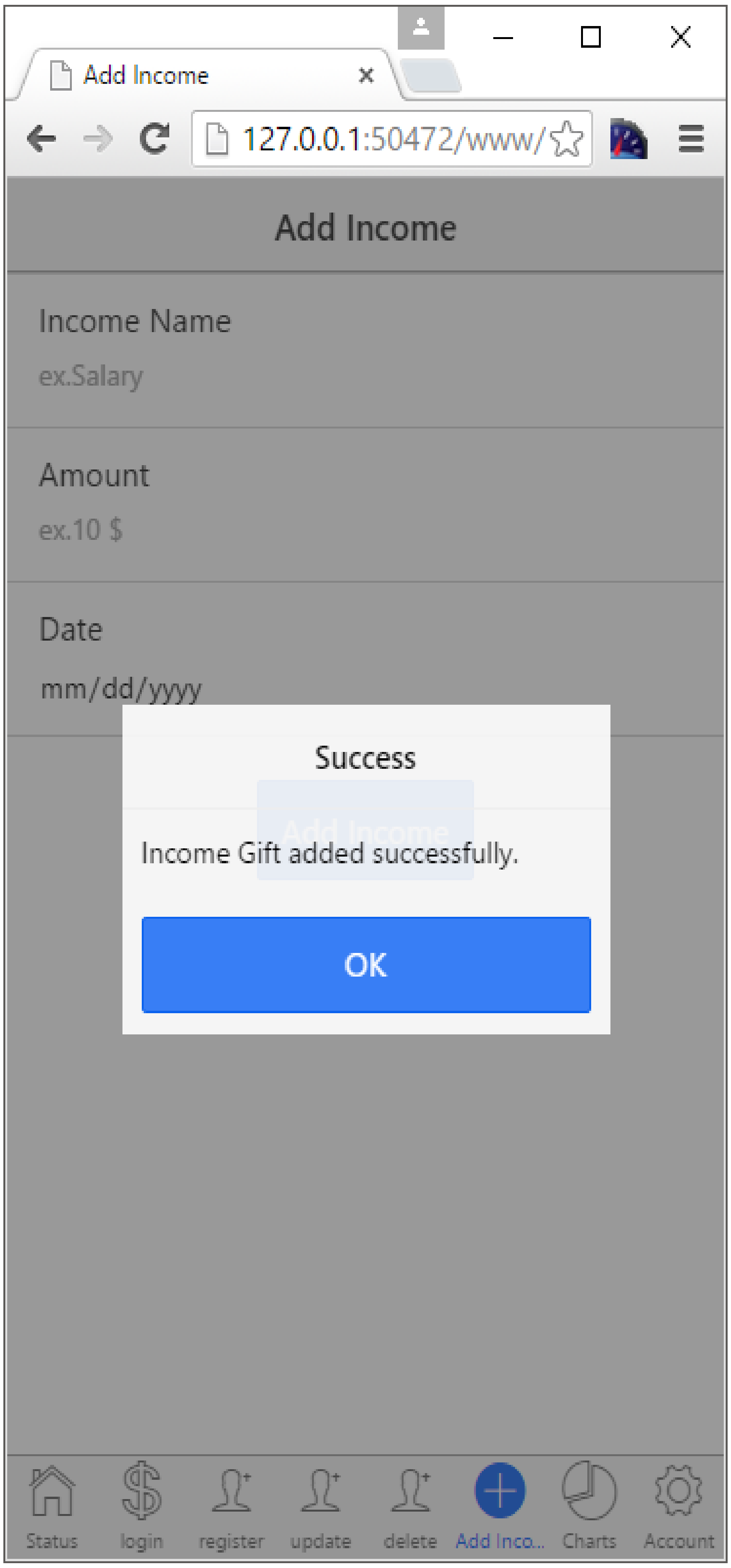
old Password ••••

new Password •••••

Alert

Password Changed

Changing password

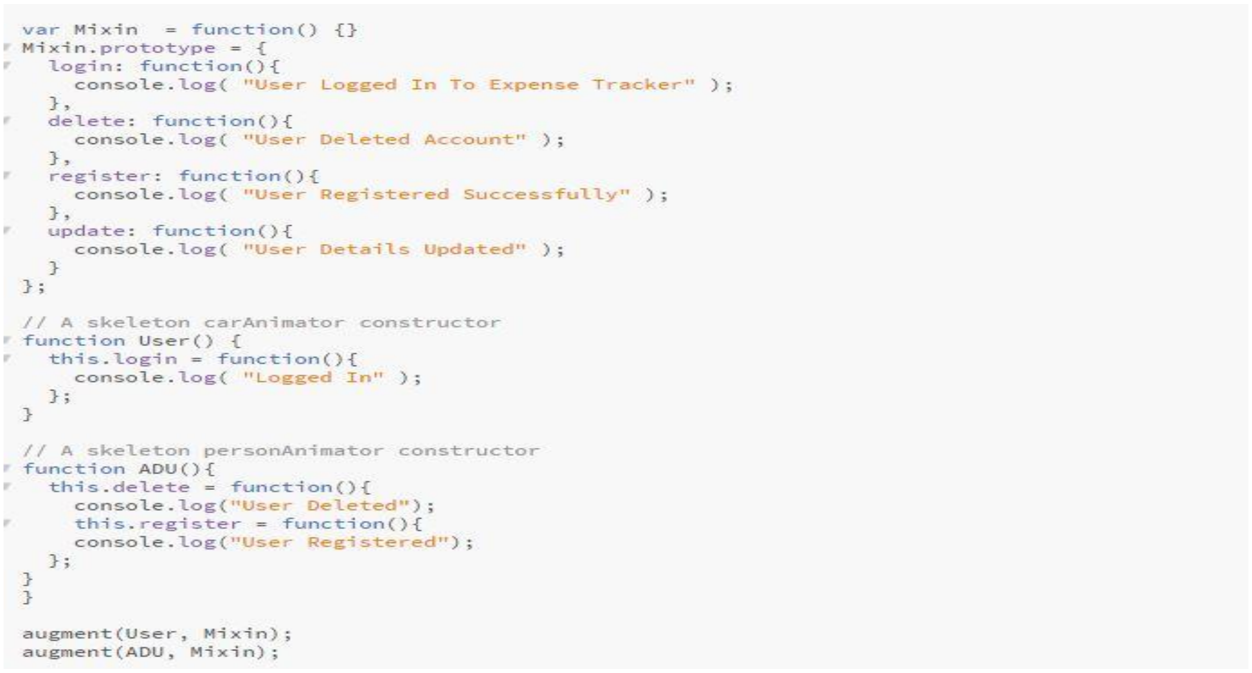


Income Gift added successfully.

Adding Income

Design Patterns

We have used only one design pattern at present for our application. Mixin design pattern was used to indicate the actions like login, insert, update and delete are performed.



```
var Mixin  = function() {}
Mixin.prototype = {
  login: function(){
    console.log( "User Logged In To Expense Tracker" );
  },
  delete: function(){
    console.log( "User Deleted Account" );
  },
  register: function(){
    console.log( "User Registered Successfully" );
  },
  update: function(){
    console.log( "User Details Updated" );
  }
};

// A skeleton carAnimator constructor
function User() {
  this.login = function(){
    console.log( "Logged In" );
  };
}

// A skeleton personAnimator constructor
function ADU(){
  this.delete = function(){
    console.log("User Deleted");
    this.register = function(){
    console.log("User Registered");
  };
}
}

augment(User, Mixin);
augment(ADU, Mixin);
```

Deployment

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

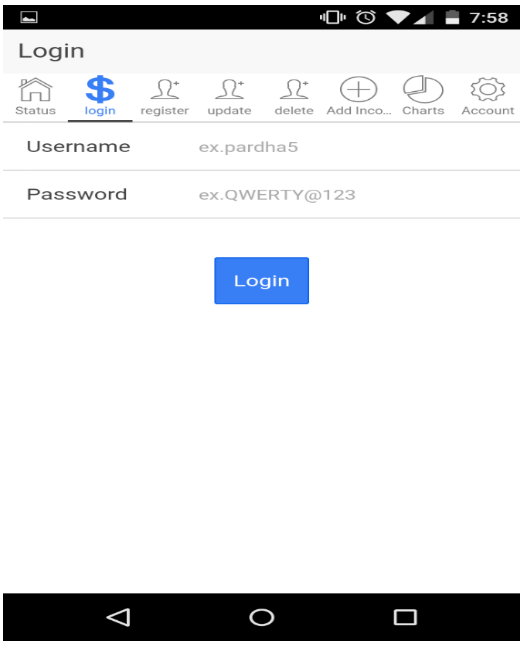
Report

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 server that process the request of the user and stores the data provided by user in database which is running in the background.

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 sore 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.

Application Screen Shots



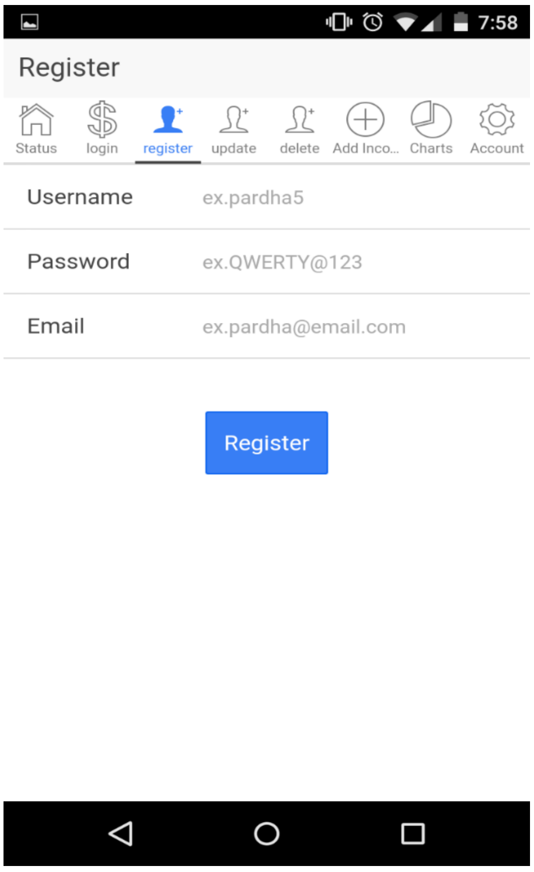
Login

Username ex.pardha5

Password ex.QWERTY@123

Login

Login page



Register

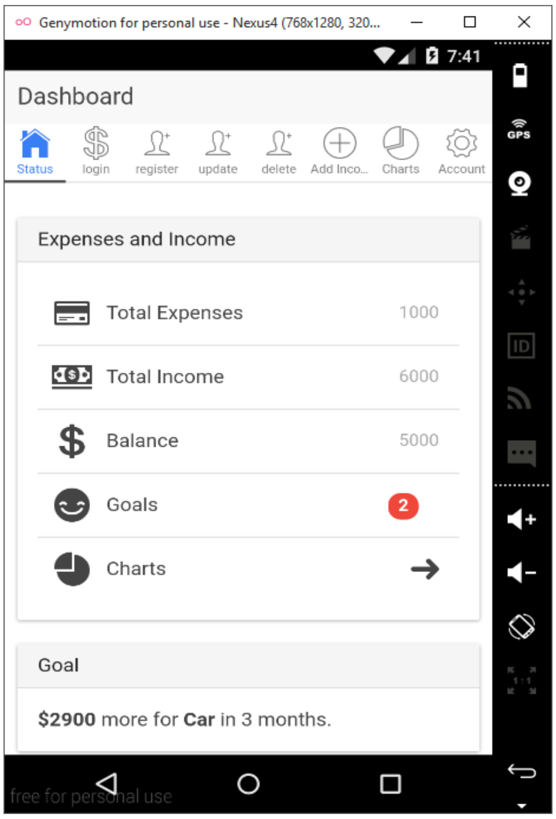
Username ex.pardha5

Password ex.QWERTY@123

Email ex.pardha@email.com

Register

Register page



Dashboard

Expenses and Income

Total Expenses 1000

Total Income 6000

Balance 5000

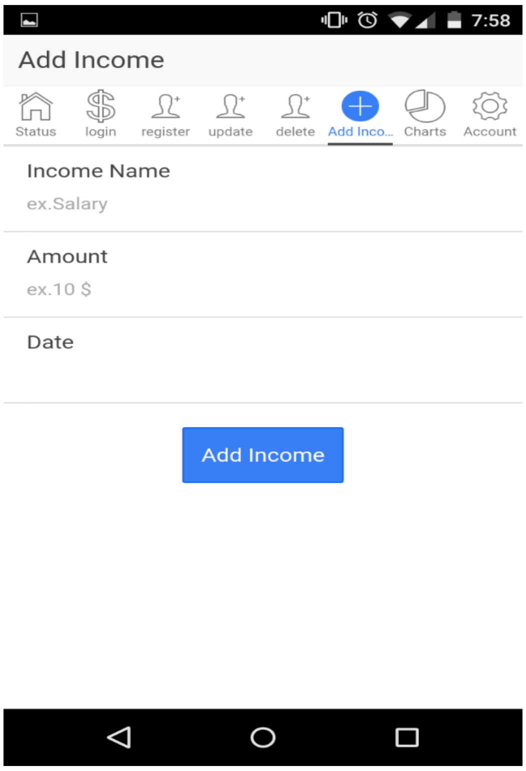
Goals 2

Charts

Goal

$2900 more for **Car** in 3 months.

Dashboard



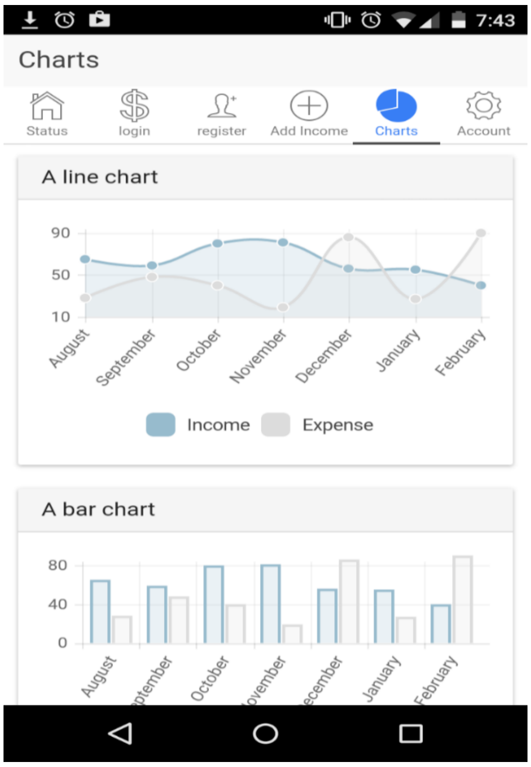
Add Income

Income Name
ex.Salary

Amount
ex.10 $

Date

Add income page

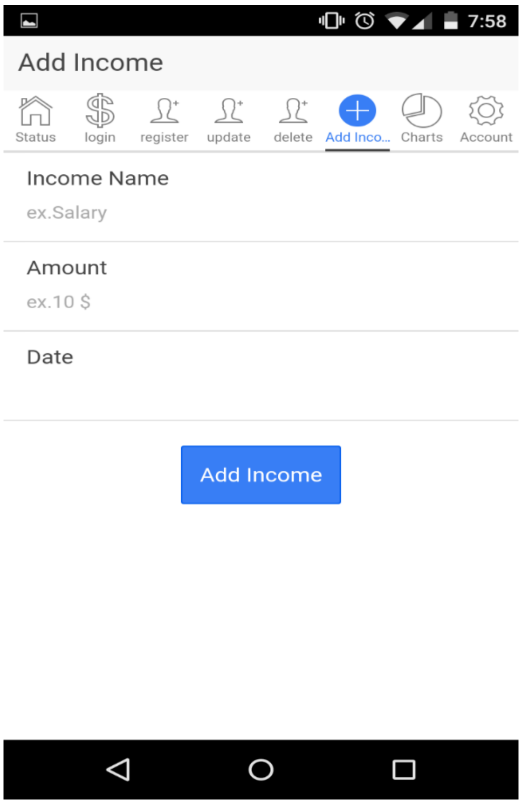

Charts

Status login register Add Income Charts Account

A line chart

A bar chart

Charts page


Add Income

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

Income Name

ex.Salary

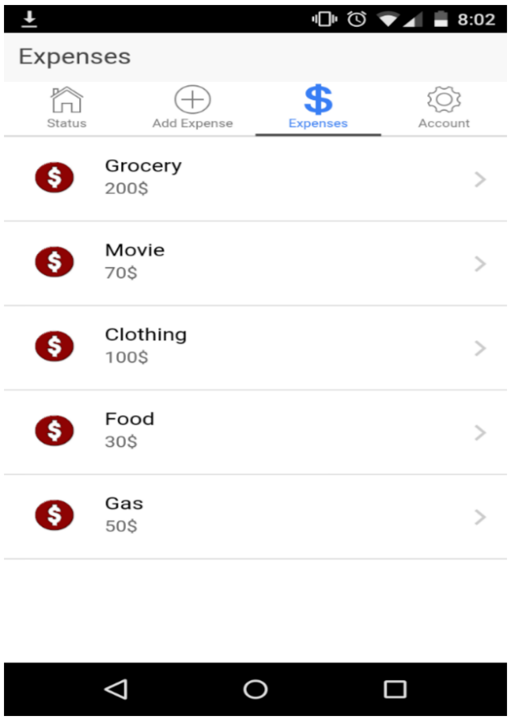
Amount

ex.10 $

Date

Add Income

Add income page



Grocery
200$

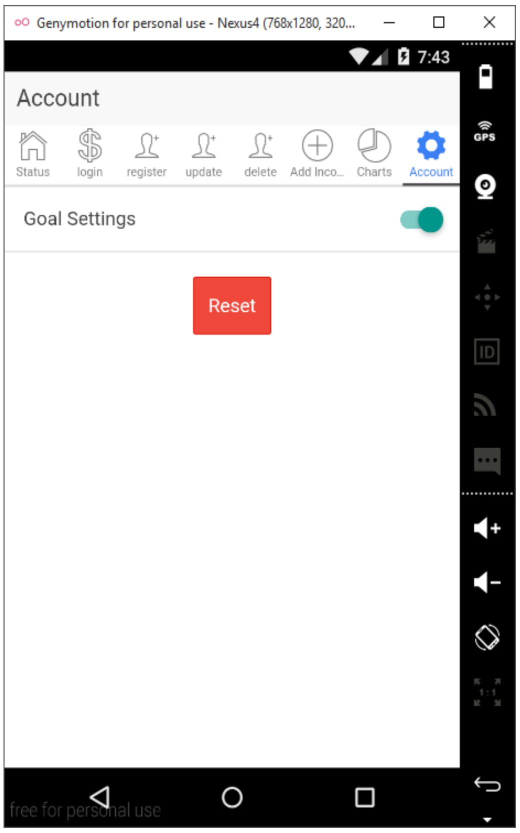
Movie
70$

Clothing
100$

Food
30$

Gas
50$

Expense list



Goal Settings: on

Settings **Tab**

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. A middle ware web server is also added to make MVC architecture.

• Responsibility (Task, Person):

Pardha Saradhi koye (40): Adding Income, View Income, Charts and Goal setting.

Koushik Nallani (48): Login, Registration, Change Password and Goal setting database creation.

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

• Time taken:

15 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.

Other elements of goal setting.

Combining all modules to make one single app.

• Responsibility (Task, Person)

Goal setting other elements, Pardha Saradhi Koye

Voice to text, Koushik Nallani

Integration, Nithin Sai Peram

• Time to be taken (estimated #hours)

30 hours for all.

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

CS 5551 Fourth Increment Report

Introduction

This document is the summary of work done by us for fourth increment of our android project expense tracker. In the fourth increment we have developed a fully working version of expense tracker that interacts with users in getting data and stores them in mongo DB with the help of servlet. This servlet logic to handle data is deployed on to blue mix. Details of the previous increment are available in third increment report [1] submitted by us earlier.

For this increment, we mainly focused on finishing the functionalities that we proposed in this application. These functionalities include user interface design, login, registration, adding incomes, adding expenses, charts, goal settings and displaying the summary of user finances in dashboard. In addition to these we have also implemented a simple view of incomes, expenses and goals that are added by the user. In this increment we have also included a new tier to our application which is a server that connects our application to the backend database.

The database tier of expense tracker was designed by using Mongo DB [2] a NoSQL database system. Database connectivity was done using servlets. Tables like incomes, expenses and users are implemented as collection and are accessed through the servlets or business logic stored in IBM blue mix.

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.
- Users can add goals.
- User can view summary of his finances in the dash board.

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, chart services which is provided by ionic and we are also working on the speech to text api for future increments.

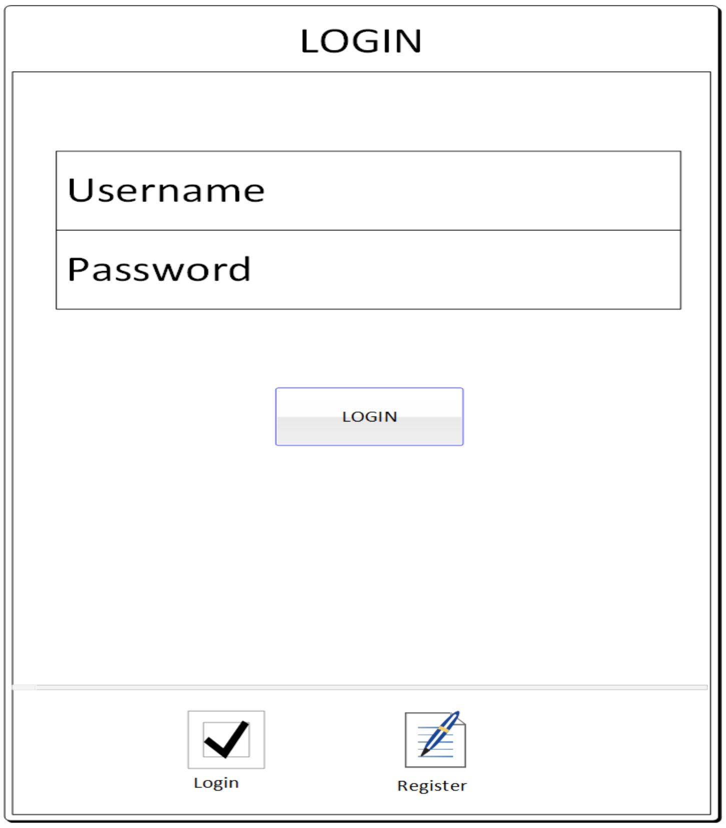
Chart Services [3]: At first during the first increment we tried to use google chart api [4] for developing analytics to our expense tracker application but after introduction of ionic framework, we used chart.js to develop bar and line charts.

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.

Detail Design

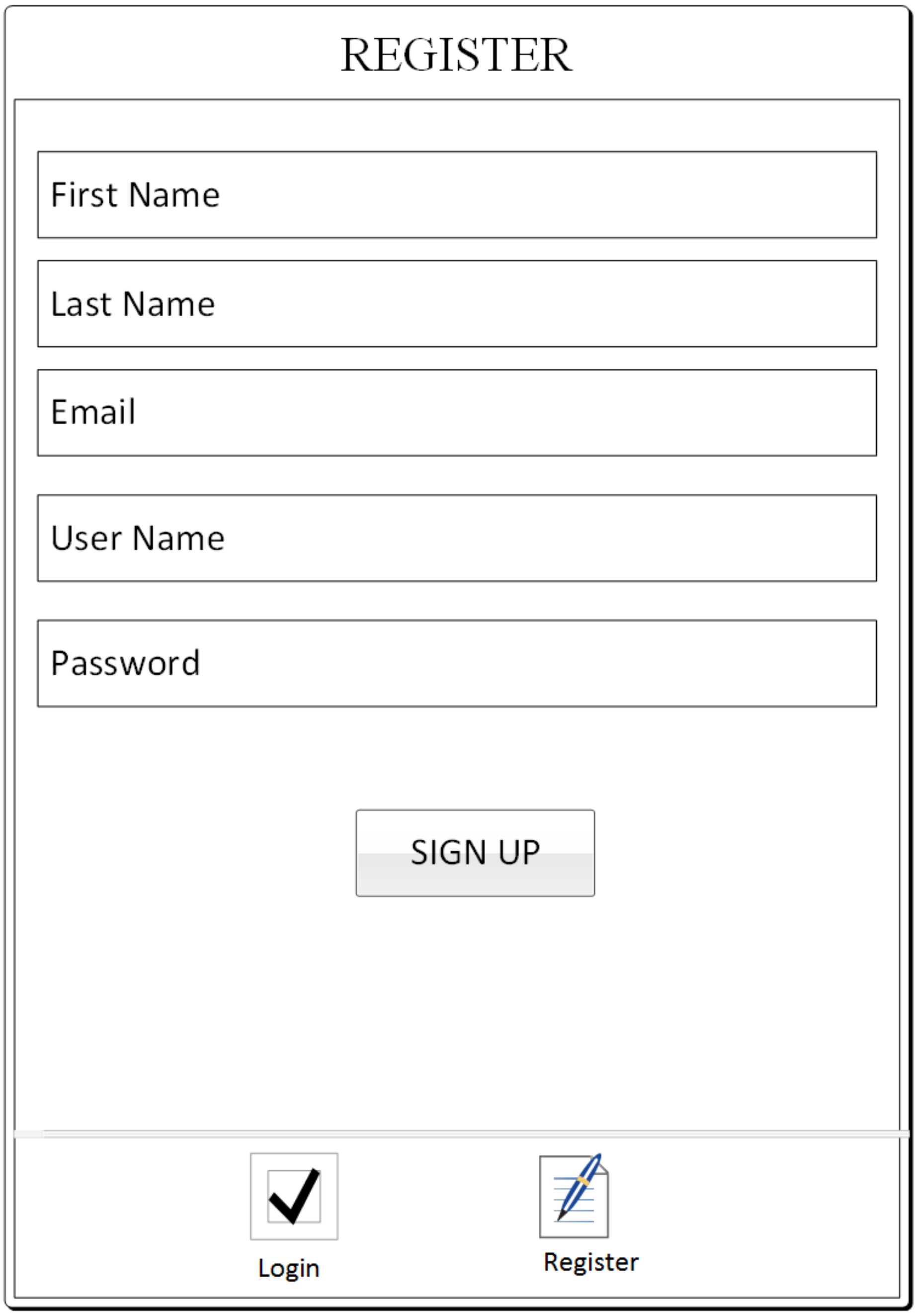
Wireframes

1) Login page for Expenses Tracker



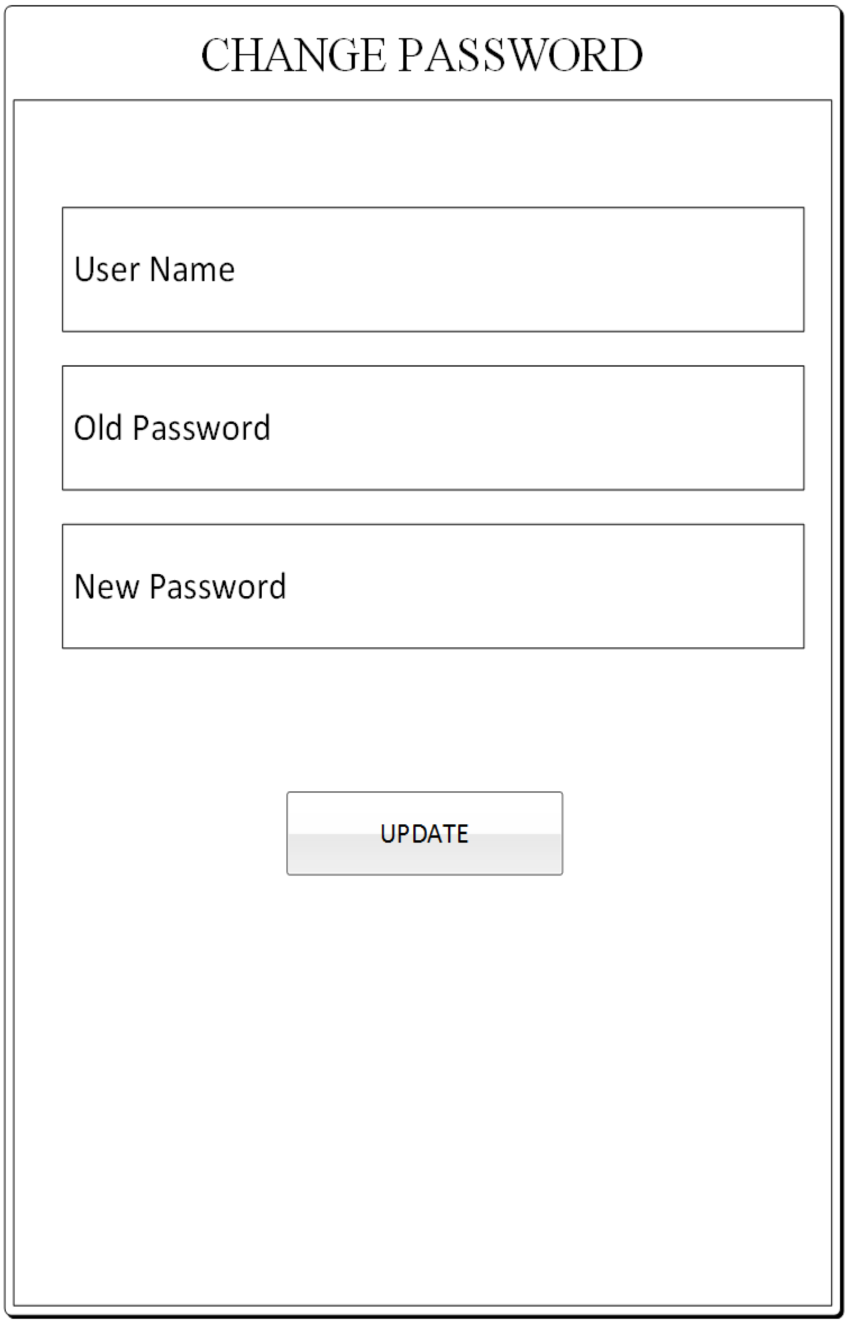
We have designed a basic login page for the users to access their account in a secure way. The details of the users are stored in the mongo lab [6] database.

2) Registration page for Expenses Tracker



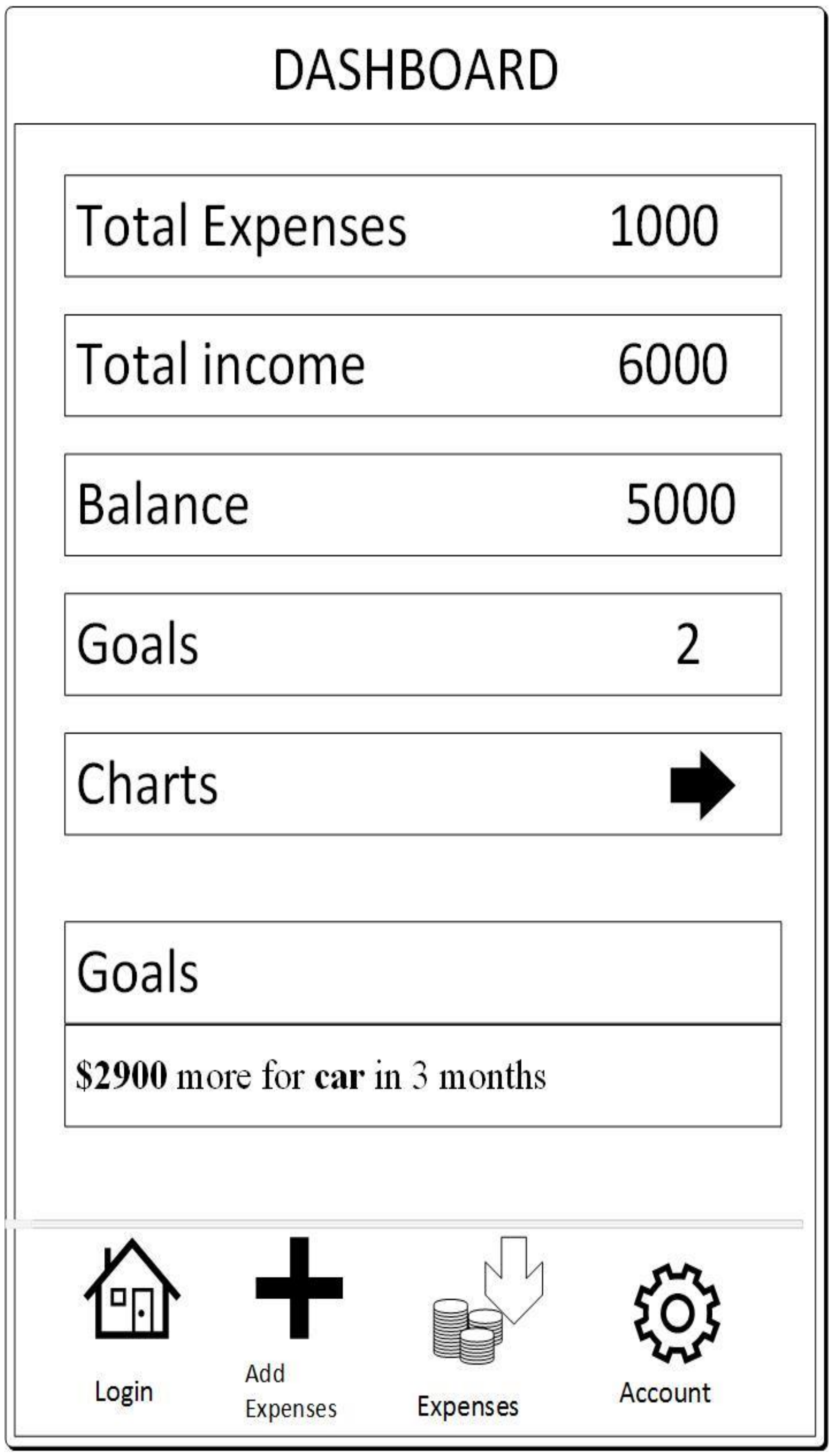
We have designed a basic registration page for the users to create their own account to store all the details of income and daily expenses in a secure way. So that they can access their data where they want.

3) Password update page for Expenses Tracker



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

4) Home or dash board page for Expenses Tracker



DASHBOARD

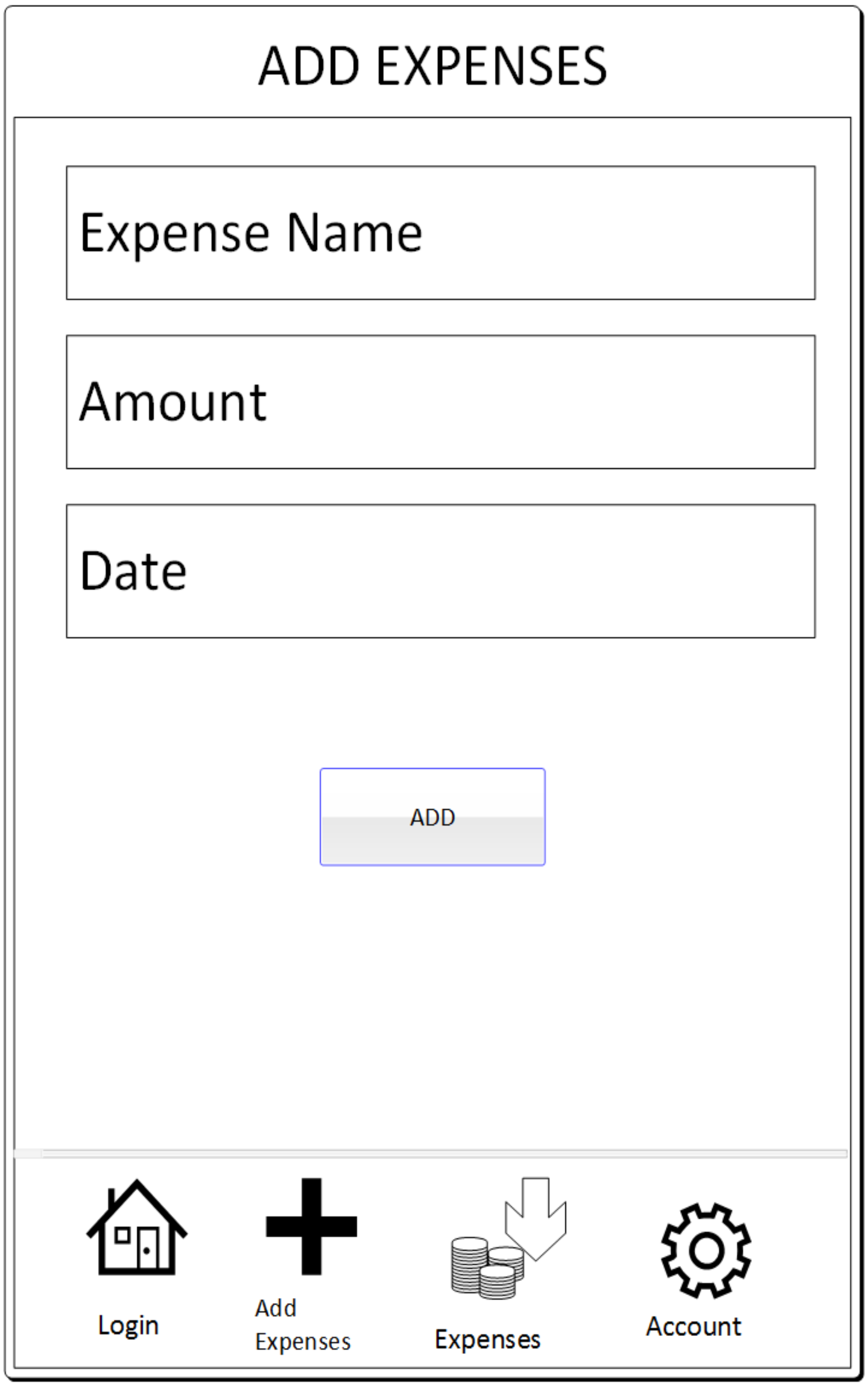
Total Expenses	1000
Total income	6000
Balance	5000
Goals	2
Charts	➡

Goals

$2900 more for **car** in 3 months

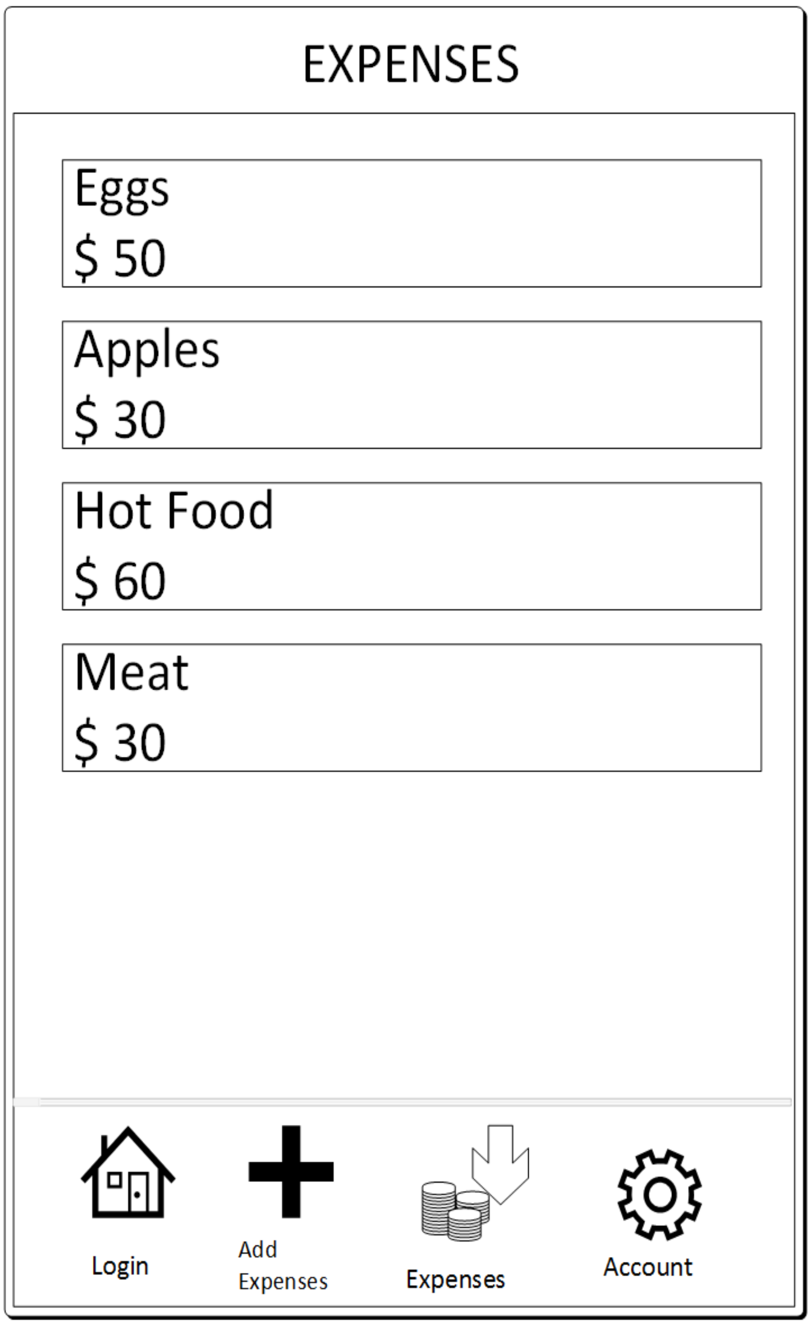
This is the home page where the user can see all is statistics briefly.

5) Add Expenses page for Expenses Tracker



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

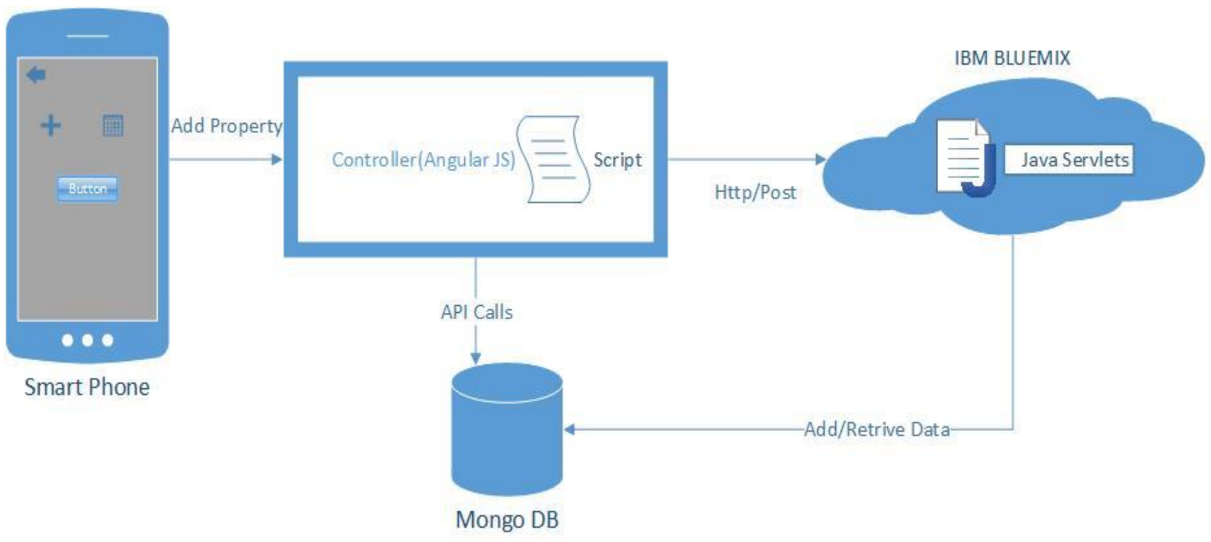
Hot Food
$ 60

Meat
$ 30

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

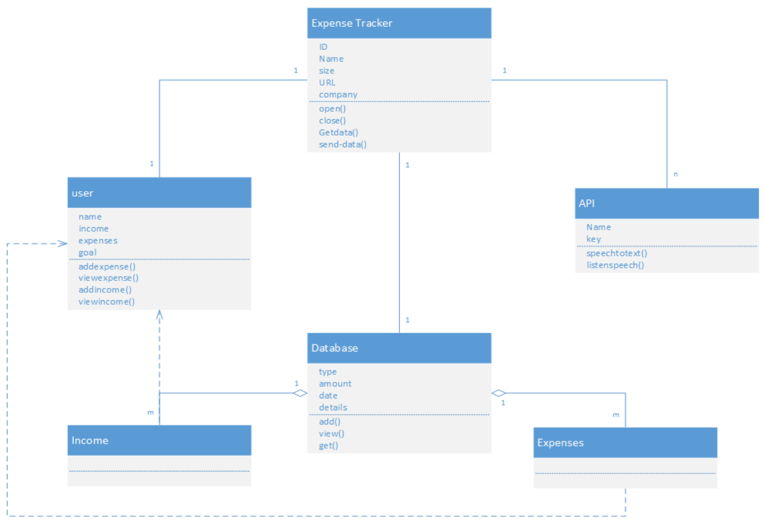
This shows the system architecture of the expense tracker application.



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 controller is being called to ease the process. This request processing is done with the help of a program that runs on server. 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.

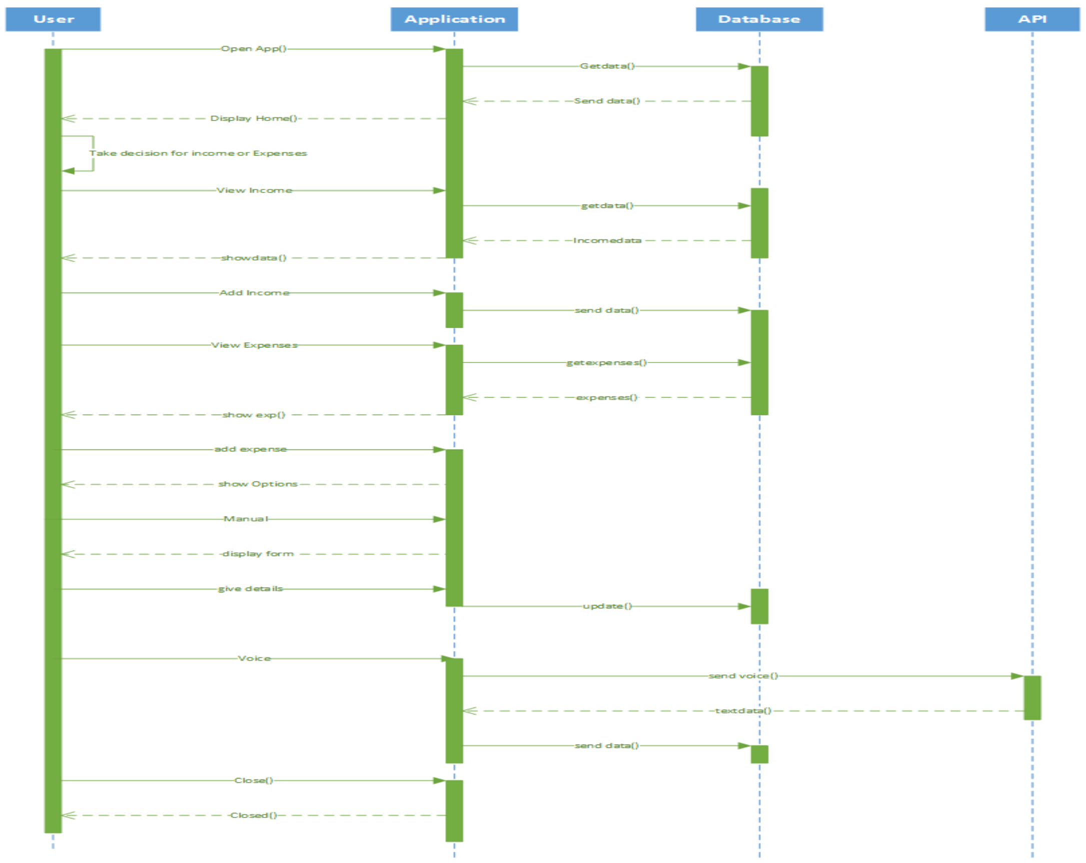
Database is a class in which all the data is stored. Its members are type of entry income or expenses, amount or value of entry, date of entry, details about the entry. Its functions are to add, view, and get the entries to the database and from the database.

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.



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 test cases 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.

Implementation

Server Side Implementation:

For the server side implementation we have used web-sphere that runs on our local machine and processes the request provided by the user. Logic for our application is presented in servlets that are stored in this server and this server is also responsible for interacting with mongo DB for insertion, deletion and updates in the database.

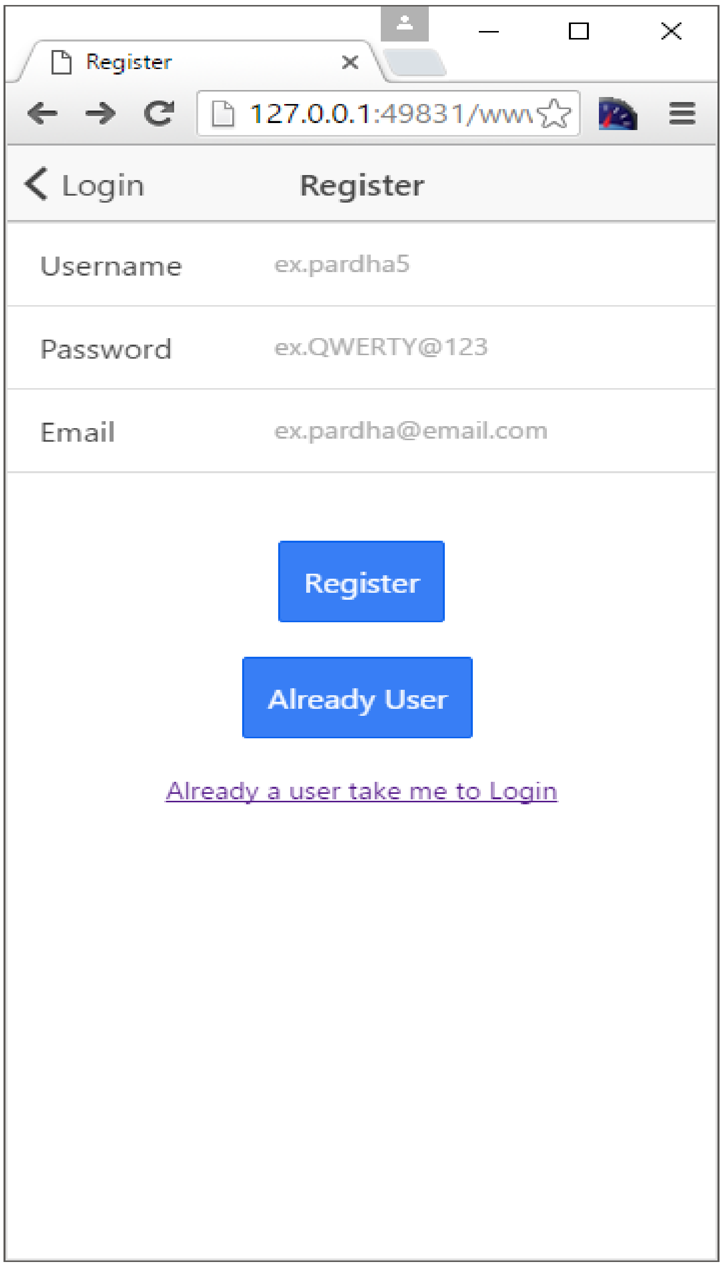
“Expense Tracker” is the database name in which it has a set of collections. The data is stored in the form of JSON data but for the view of developer data can also be viewed in the form of the table created by developer choice.

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.

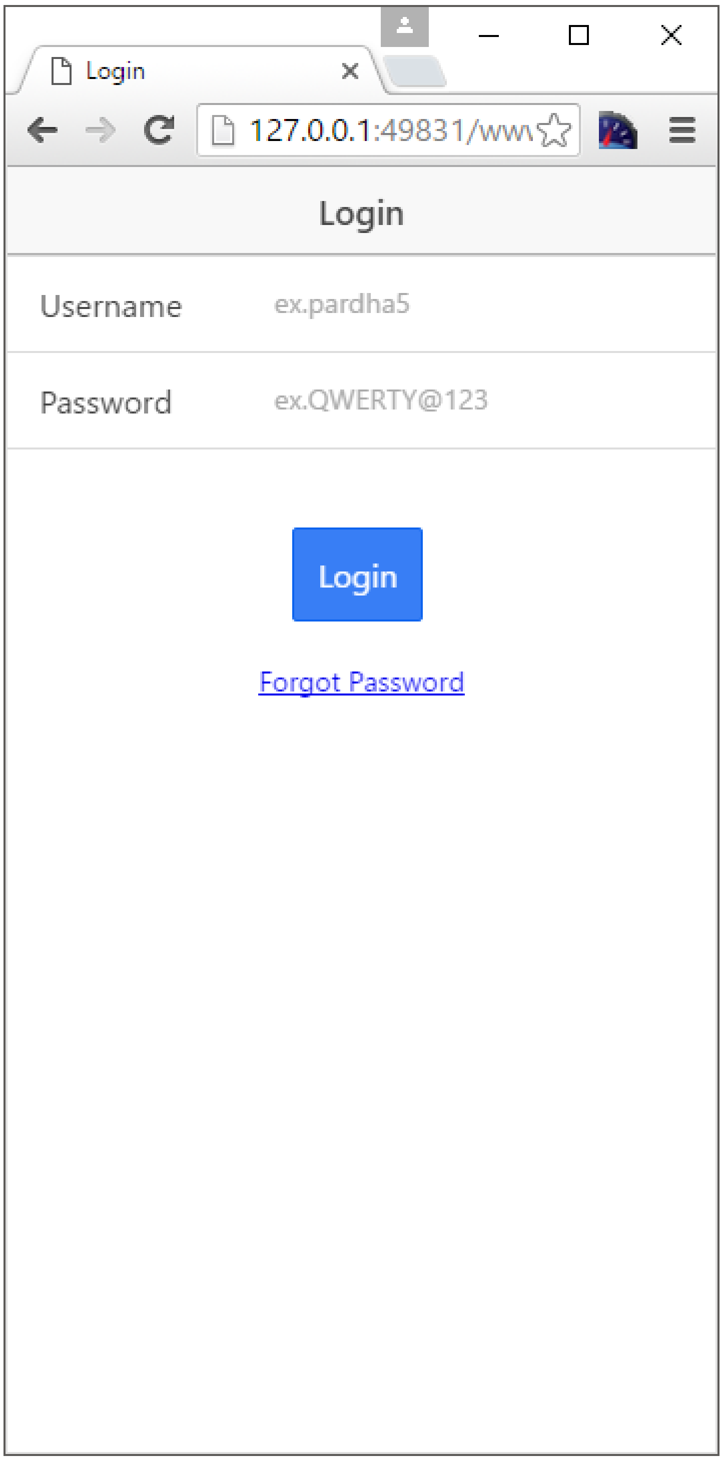
Mobile Client Implementation:

User interface has been designed using ionic framework [7] 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.

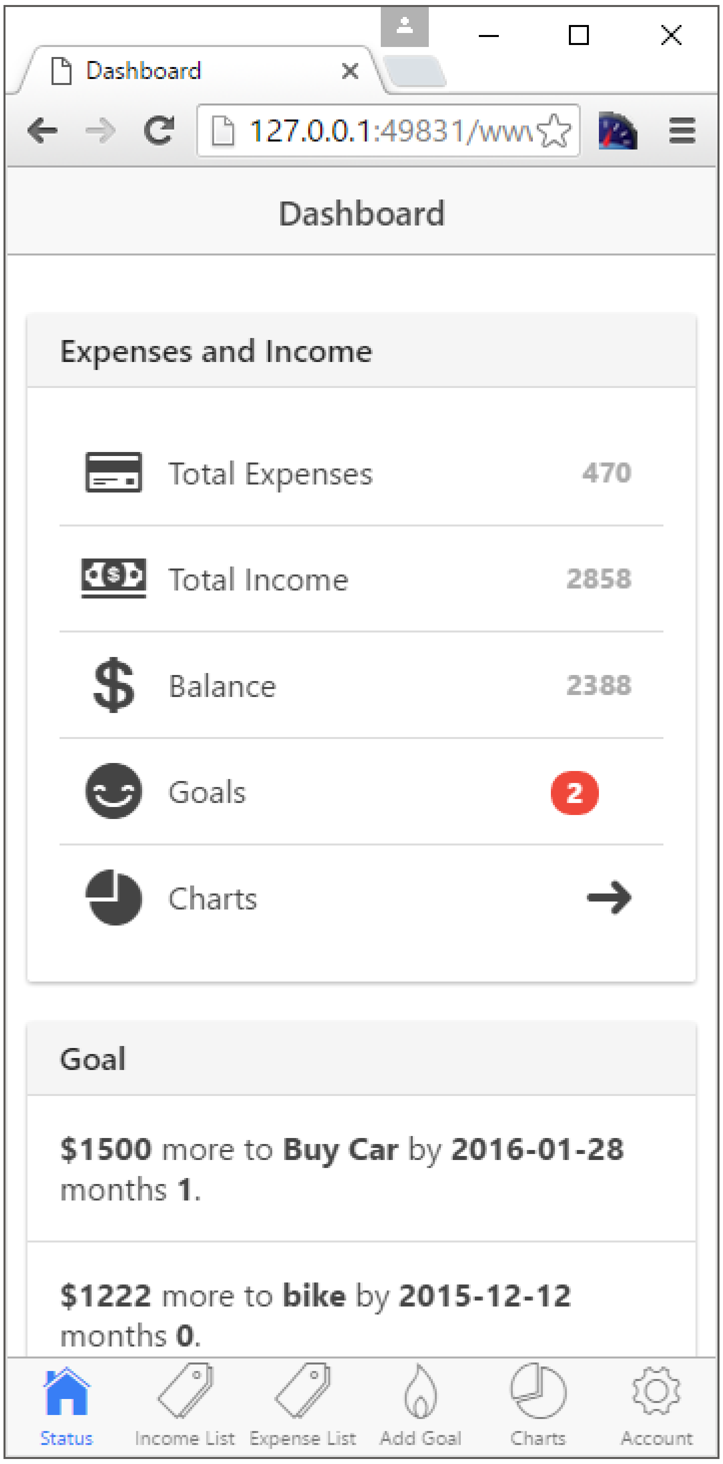
Charts are used to show the statistics of the user. For charts we have used angular-charts.js and charts.js JavaScript files and chart.css for style.



Registration Page


Login
127.0.0.1:49831/www
Login
Username
ex.pardha5
Password
ex.QWERTY@123
Login
Forgot Password

Login page


Dashboard

127.0.0.1:49831/www

Dashboard

Expenses and Income

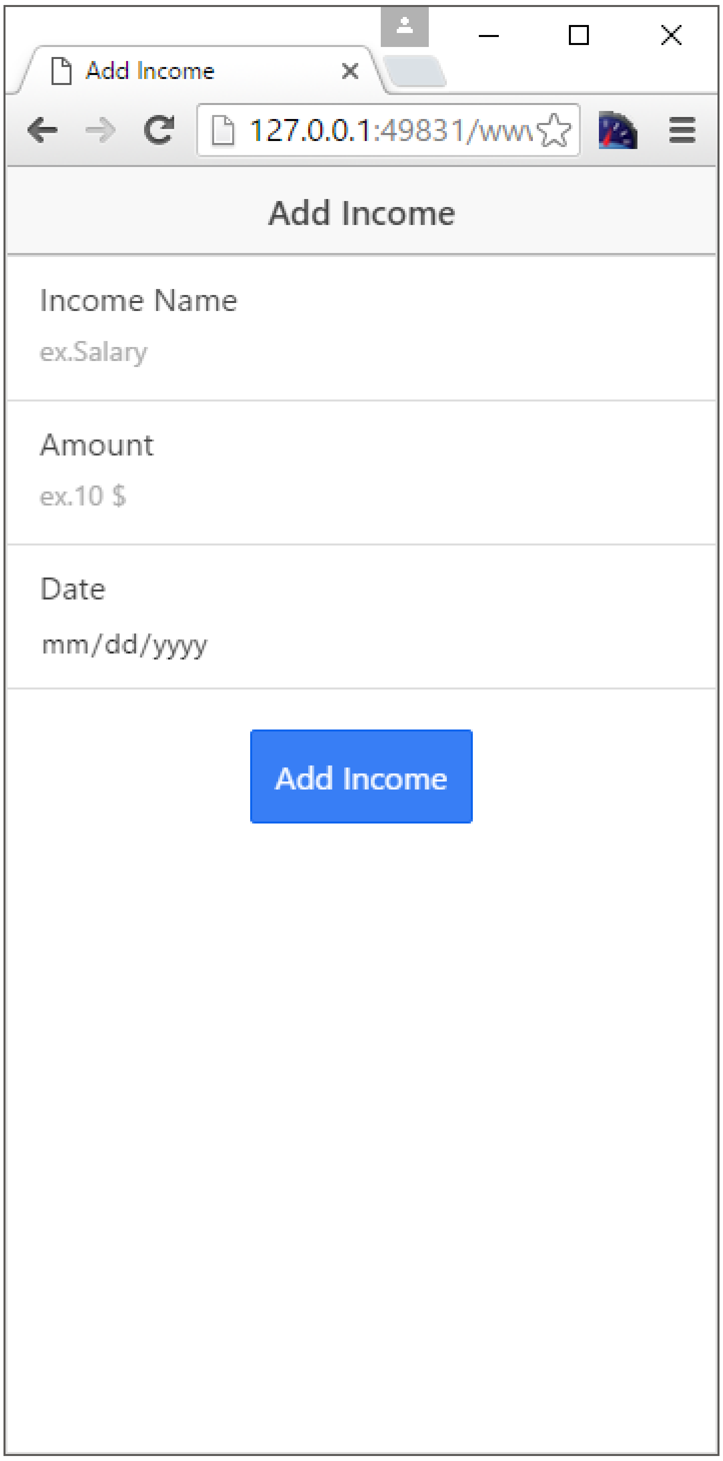
Total Expenses	470
Total Income	2858
Balance	2388
Goals	2
Charts	→

Goal

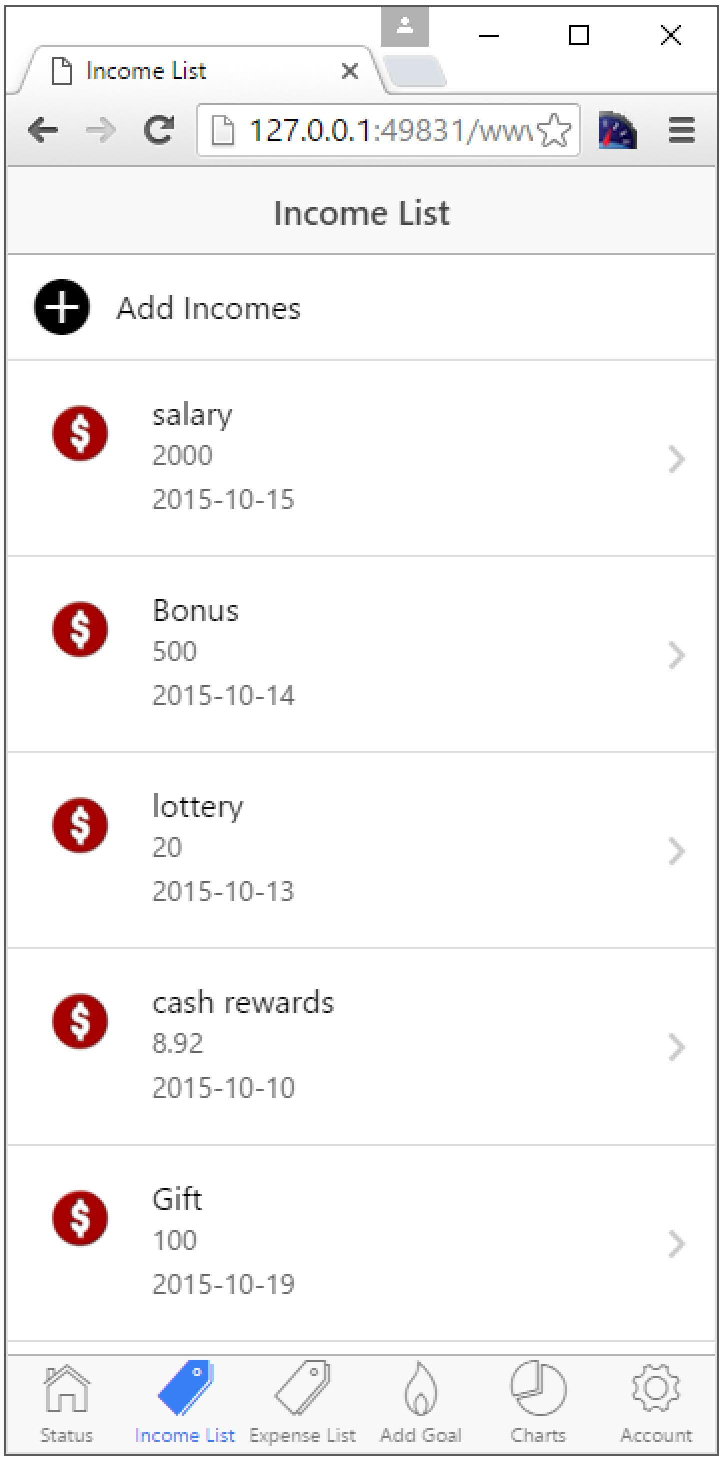
$1500 more to **Buy Car** by **2016-01-28** months **1**.

$1222 more to **bike** by **2015-12-12** months **0**.

Dash Board



Adding Income


Income List

127.0.0.1:49831/www

Income List

Add Incomes

salary
2000
2015-10-15

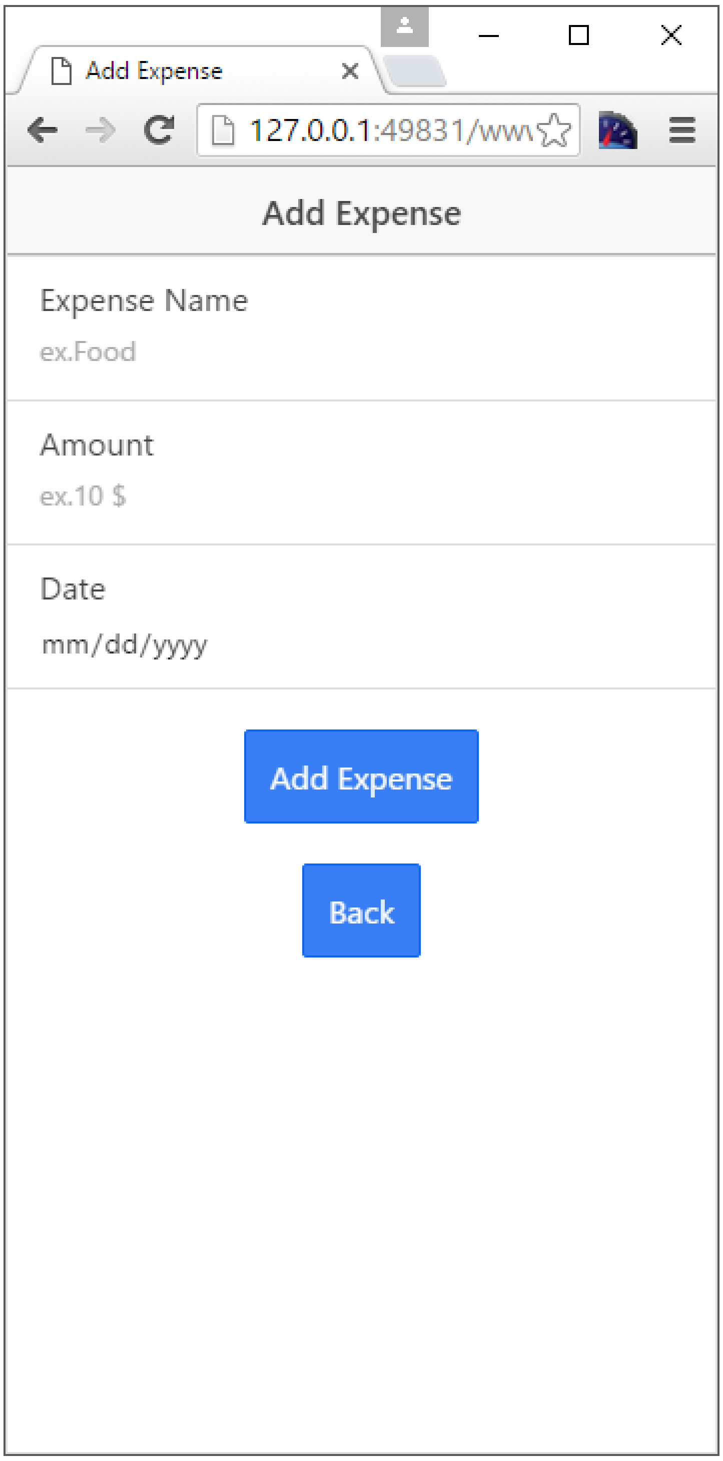
Bonus
500
2015-10-14

lottery
20
2015-10-13

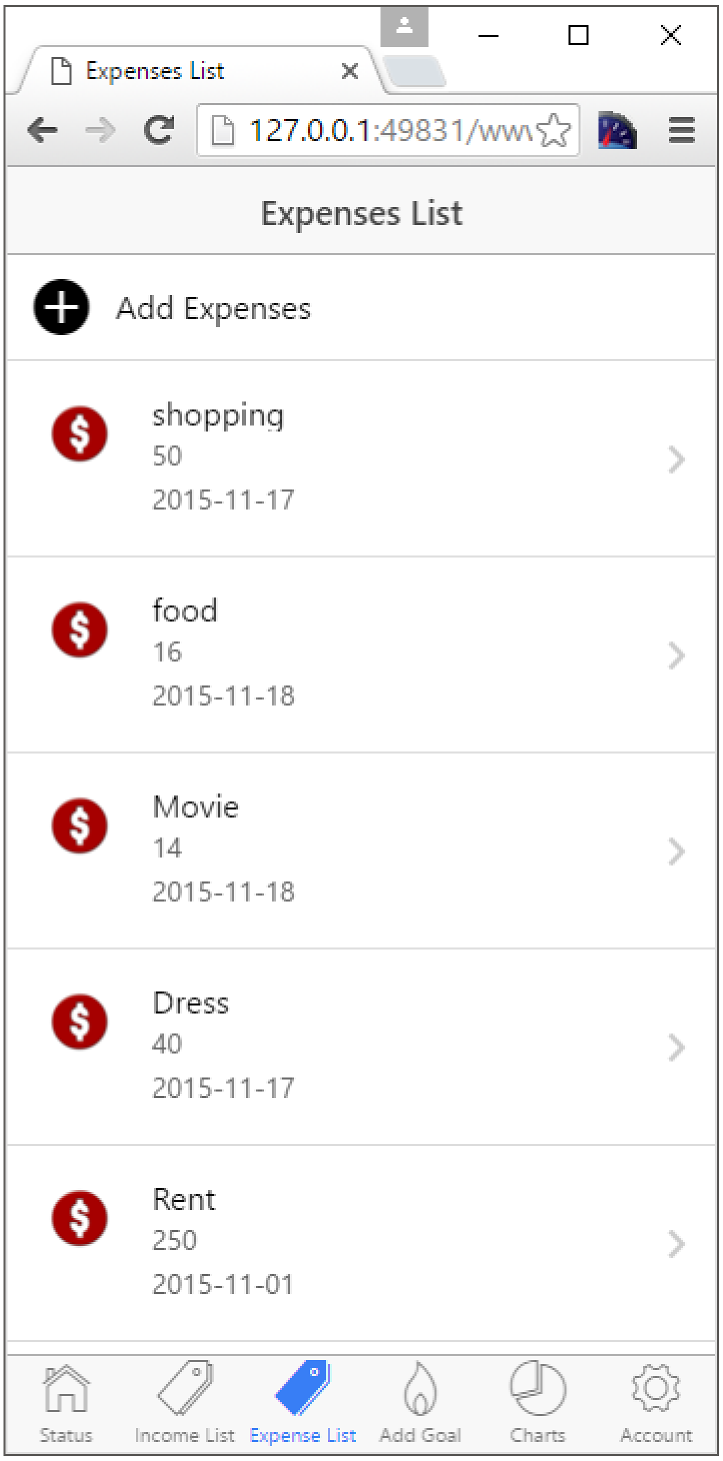
cash rewards
8.92
2015-10-10

Gift
100
2015-10-19

Income List



Adding Expense


Expenses List

127.0.0.1:49831/ww

Expenses List

Add Expenses

shopping
50
2015-11-17

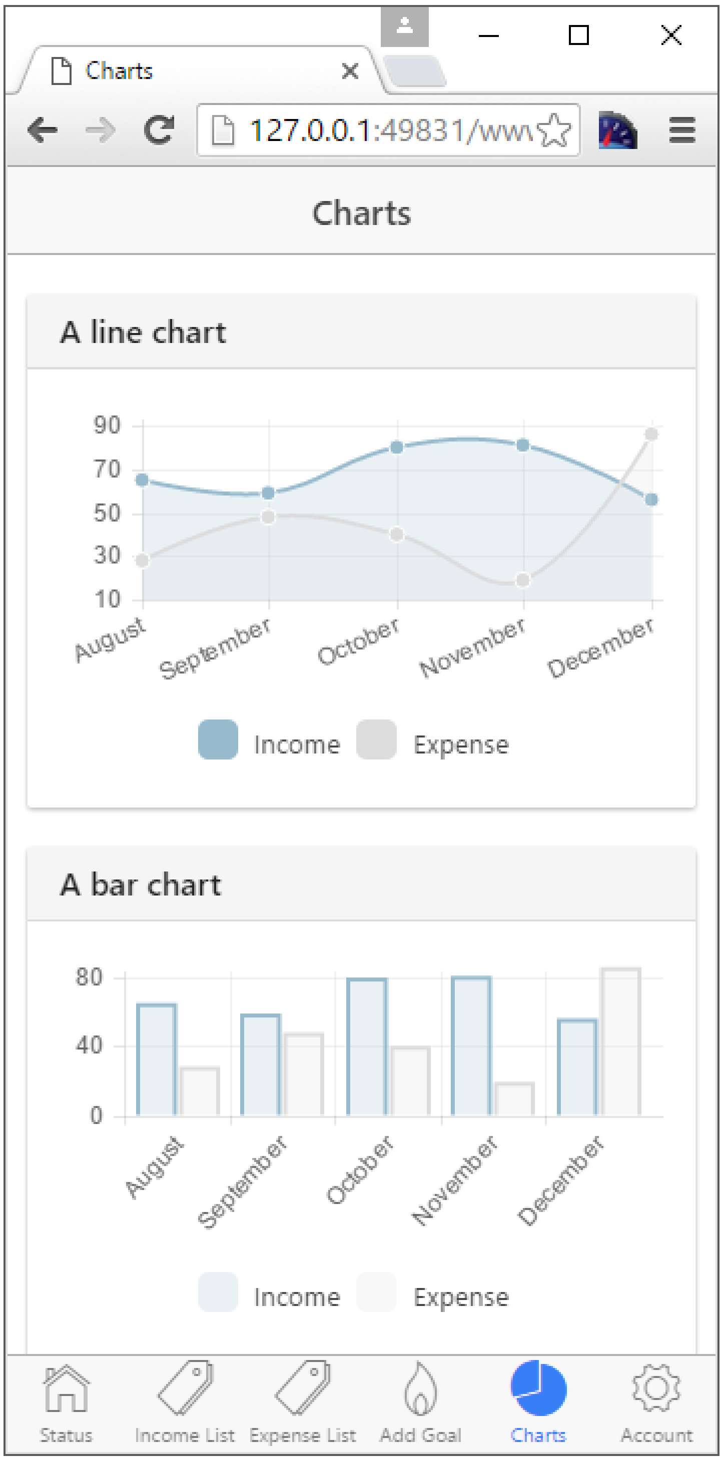
food
16
2015-11-18

Movie
14
2015-11-18

Dress
40
2015-11-17

Rent
250
2015-11-01

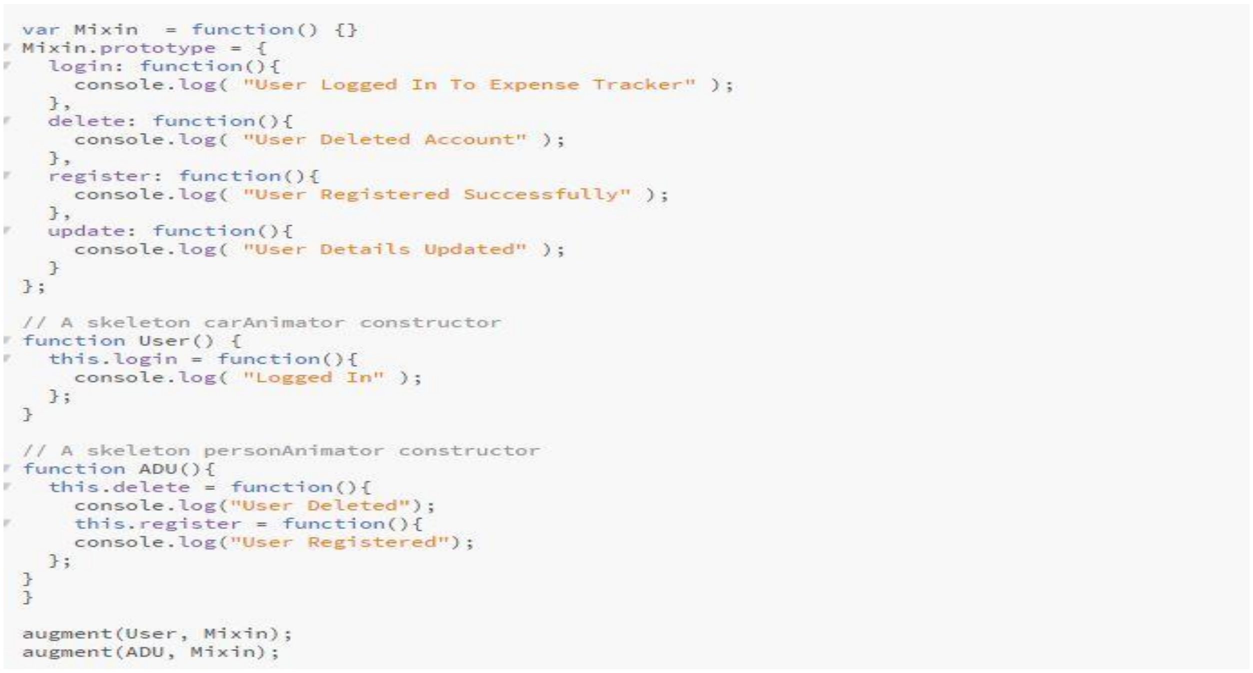
Expense List


Charts
127.0.0.1:49831/www
Charts
A line chart
90
70
50
30
10
August
September
October
November
December
Income
Expense
A bar chart
80
40
0
August
September
October
November
December
Income
Expense
Status
Income List
Expense List
Add Goal
Charts
Account

Charts

Design Patterns

We have used only one design pattern at present for our application. Mixin design pattern was used to indicate the actions like login, insert, update and delete are performed.



```
var Mixin  = function() {}
Mixin.prototype = {
  login: function(){
    console.log( "User Logged In To Expense Tracker" );
  },
  delete: function(){
    console.log( "User Deleted Account" );
  },
  register: function(){
    console.log( "User Registered Successfully" );
  },
  update: function(){
    console.log( "User Details Updated" );
  }
};

// A skeleton carAnimator constructor
function User() {
  this.login = function(){
    console.log( "Logged In" );
  };
}

// A skeleton personAnimator constructor
function ADU(){
  this.delete = function(){
    console.log("User Deleted");
    this.register = function(){
    console.log("User Registered");
  };
}
}

augment(User, Mixin);
augment(ADU, Mixin);
```

Deployment

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

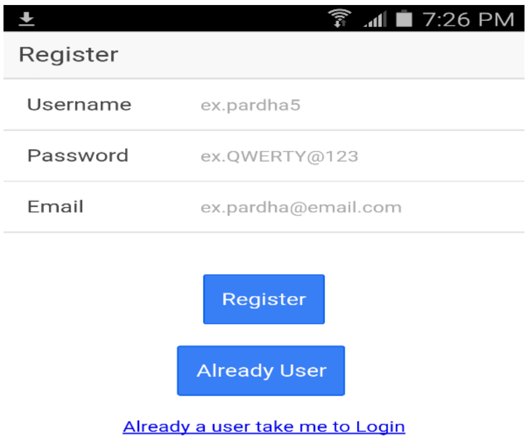
Report

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 server that process the request of the user and stores the data provided by user in database which is running in the background.

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 sore 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.

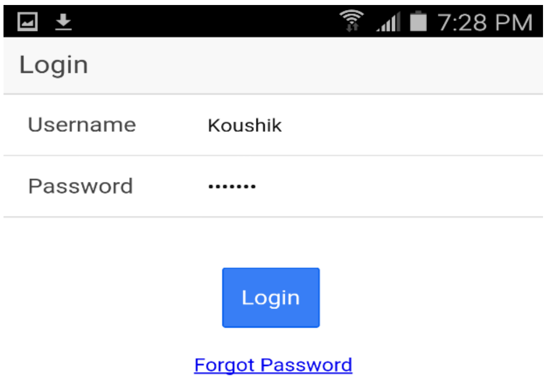
Application Screen Shots


7:26 PM

Register

Username	ex.pardha5
Password	ex.QWERTY@123
Email	ex.pardha@email.com

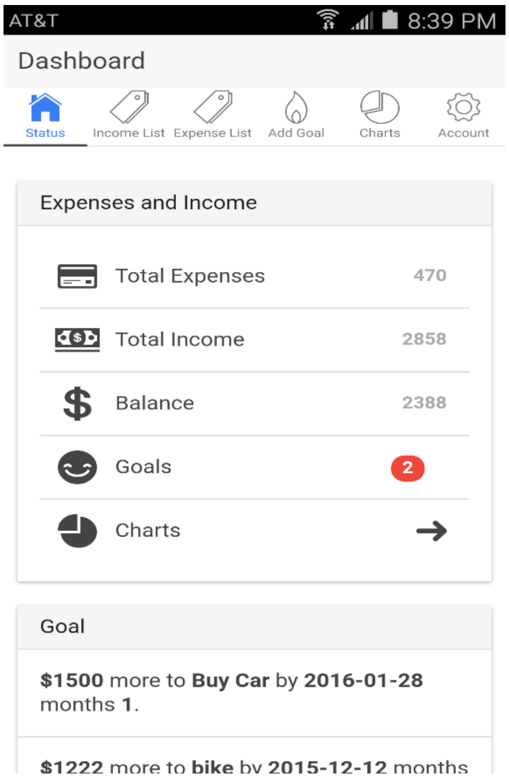
Register page


7:28 PM

Login

Username	Koushik
Password	•••••••

Register page



AT&T 8:39 PM

Dashboard

Status Income List Expense List Add Goal Charts Account

Expenses and Income

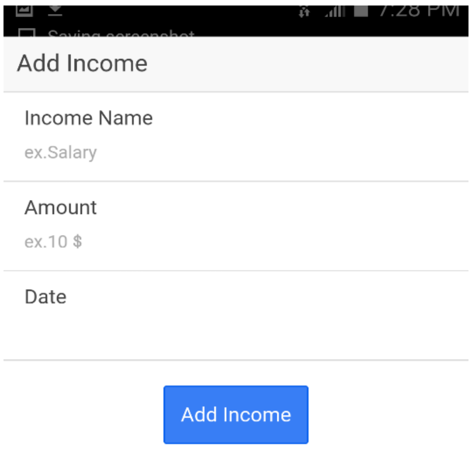
Total Expenses	470
Total Income	2858
Balance	2388
Goals	2
Charts	→

Goal

$1500 more to **Buy Car** by **2016-01-28** months **1**.

$1222 more to **bike** by **2015-12-12** months

Dashboard



7:28 PM

Saving screenshot

Add Income

Income Name

ex.Salary

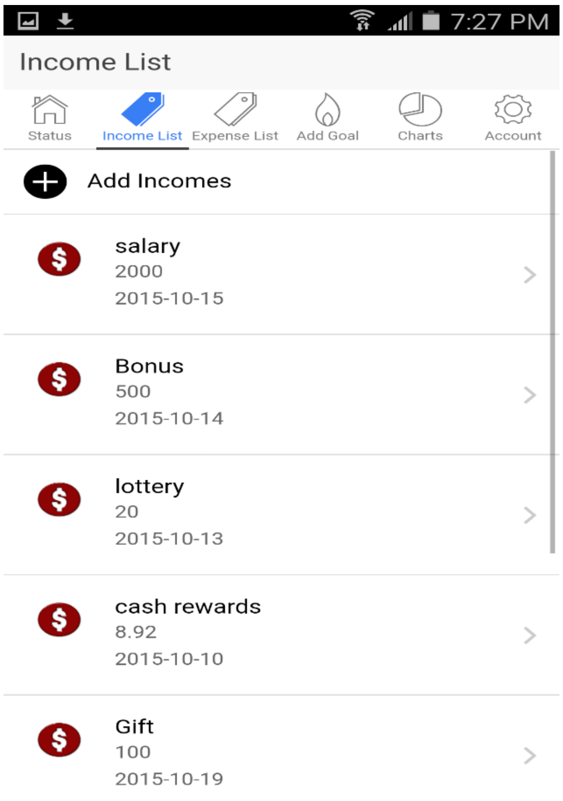
Amount

ex.10 $

Date

Add Income

Add income page



Income List

Add Incomes

salary
2000
2015-10-15

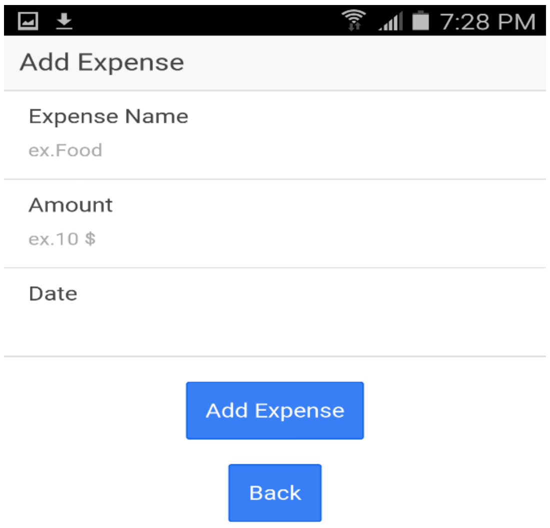
Bonus
500
2015-10-14

lottery
20
2015-10-13

cash rewards
8.92
2015-10-10

Gift
100
2015-10-19

Income List



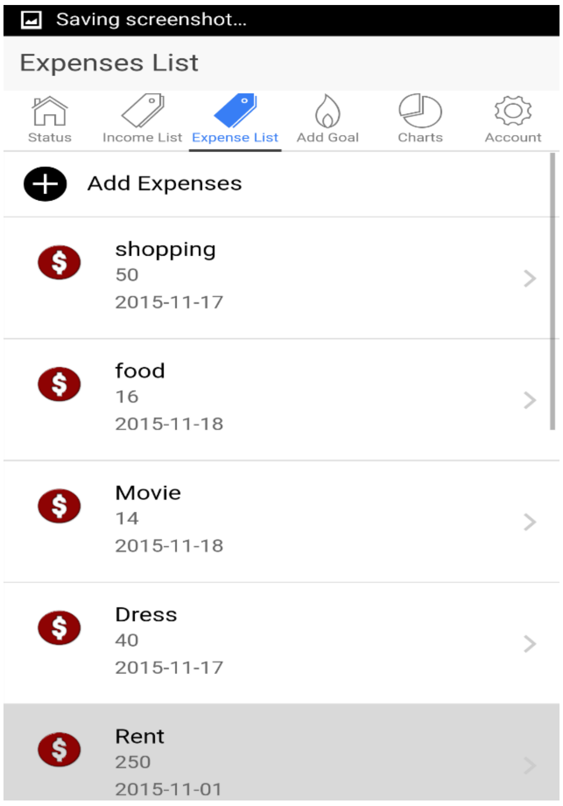
Add Expense

Expense Name
ex.Food

Amount
ex.10 $

Date

Add Expense


Saving screenshot...

Expenses List

Add Expenses

shopping
50
2015-11-17

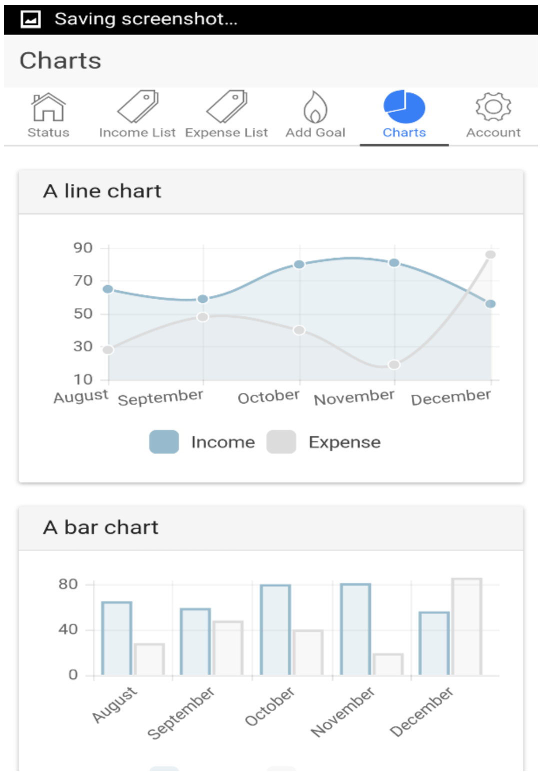
food
16
2015-11-18

Movie
14
2015-11-18

Dress
40
2015-11-17

Rent
250
2015-11-01

Expense **List**


Saving screenshot...

Charts

Chart

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. A middle ware web server is also added to make MVC architecture.

- Responsibility (Task, Person):

Pardha Saradhi koye (40): Charts and Goals.

Koushik Nallani (48): Database design for all collections and Testing.

Nithin Sai Peram (50): Update password, Dash Board.

- Time taken:

15 hours each person.

- Contributions (members/percentage)

Each 33.3 percentage.

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

Implementation package

Index.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="initial-scale=1, maximum-scale=1, user-scalable=no,
width=device-width">
    <title></title>
      <link rel="shortcut icon" href="data:image/x-icon;," type="image/x-icon">
    <link href="lib/ionic/css/ionic.css" rel="stylesheet">
    <link href="css/style.css" rel="stylesheet">
     <link rel="stylesheet" href="css/angular-chart.css">

    <!-- IF using Sass (run gulp sass first), then uncomment below and remove the CSS
includes above
    <link href="css/ionic.app.css" rel="stylesheet">
    -->

    <!-- ionic/angularjs js -->
    <script src="lib/ionic/js/ionic.bundle.js"></script>

    <!-- cordova script (this will be a 404 during development)
    <script src="cordova.js"></script>-->

    <!-- your app's js -->
    <script src="js/app.js"></script>
    <script src="js/controllers.js"></script>
    <script src="js/services.js"></script>
      <script src="js/Chart.js"></script>
       <script src="js/angular-chart.js"></script>
      <script src="js/singleton.js"></script>
    <script src="js/factory.js"></script>
    <script src="js/mixin.js"></script>

      <!--
code by us
-->

  </head>
  <body ng-app="starter">
    <!--
      The nav bar that will be updated as we navigate between views.
    -->
    <ion-nav-bar class="bar-stable">
      <ion-nav-back-button>
      </ion-nav-back-button>
    </ion-nav-bar>
    <!--
      The views will be rendered in the <ion-nav-view> directive below
      Templates are in the /templates folder (but you could also
      have templates inline in this html file if you'd like).
    -->
    <ion-nav-view>

      </ion-nav-view>
  </body>
</html>
```

Templates:

Tab-register.html:

```
<ion-view view-title="Register">
   <ion-content ng-controller="RegisterController">
      <div class="list">
         <label class="item item-input">
            <span class="input-label">Username</span>
            <input type="text" placeholder="ex.pardha5" name="userName" id="userName"
data-ng-model="user.name"/><br>
         </label>
         <label class="item item-input">
            <span class="input-label">Password</span>
            <input type="password" placeholder="ex.QWERTY@123" name="password"
id="password" data-ng-model="user.password"/><br>
         </label>
         <label class="item item-input">
            <span class="input-label">Email</span>
            <input type="text" placeholder="ex.pardha@email.com" name="email" id="email"
data-ng-model="user.email"/>
         </label>
         <center>
            <br><br>
            <button class="button button-positive" id="register" name="register"
ng_click="create(user)">Register
            </button> <br><br>
            <button class="button button-positive" id="takelog" name="takelog"
ng_click="takelog()">Already User
            </button> <br><br>
         </center>
      </div>
   </ion-content>
</ion-view>
```

Tab-Login.html

```
<ion-view view-title="Login">
<ion-content ng-controller="LoginController">
 <div class="list">
  <label class="item item-input">
    <span class="input-label">Username</span>
    <input type="text" placeholder="ex.pardha5" name="username" id="username" data-ng-
model="username"/><br>
  </label>
  <label class="item item-input">
    <span class="input-label">Password</span>
    <input type="password" placeholder="ex.QWERTY@123" name="password"
id="password" data-ng-model="password"/><br>
  </label>
        <center>
        <br><br><button class="button button-positive" id="login" name="login"
ng_click="login(username, password)">
  Login
</button>
     <br><br><a href="./index.html#/update">Forgot Password</a>
     </center>
</div>
    </ion-content>
</ion-view>
```

Tabs:

```
<ion-tabs class="tabs-icon-top tabs-color-active-positive">


  <!-- Dashboard Tab -->
  <ion-tab title="Status" icon-off="ion-ios-home-outline" icon-on="ion-ios-home"
href="#/tab/dash">
    <ion-nav-view name="tab-dash"></ion-nav-view>
  </ion-tab>


   <ion-tab title="Income List" icon-off="ion-social-usd-outline" icon-on="ion-social-usd"
href="#/tab/inview">
    <ion-nav-view name="tab-inview"></ion-nav-view>
  </ion-tab>


    <ion-tab title="Expense List" icon-off="ion-ios-pricetags-outline" icon-on="ion-ios-
pricetags" href="#/tab/exview">
    <ion-nav-view name="tab-exview"></ion-nav-view>
  </ion-tab>


     <ion-tab title="Add Goal" icon-off="ion-ios-flame-outline" icon-on="ion-ios-flame"
href="#/tab/addg">
    <ion-nav-view name="tab-addg"></ion-nav-view>
  </ion-tab>


    <ion-tab title="Charts" icon-off="ion-ios-pie-outline" icon-on="ion-ios-pie"
href="#/tab/charts">
    <ion-nav-view name="tab-charts"></ion-nav-view>
  </ion-tab>




    <!-- Account Tab -->
  <ion-tab title="Account" icon-off="ion-ios-gear-outline" icon-on="ion-ios-gear"
href="#/tab/account">

    <ion-nav-view name="tab-account"></ion-nav-view>

  </ion-tab>

</ion-tabs>
```

Tab-dash:

```
<ion-view view-title="Dashboard">
  <ion-content class="padding" ng-controller="DashController">
    <ion-refresher
    pulling-text="Pull to refresh..."
    on-refresh="doRefresh()">
  </ion-refresher>
      <div class="list card">
      <div class="item item-divider">Expenses and Income</div>
      <div class="item item-body">

          <div class="list">

              <a class="item item-icon-left" href="./index.html#/tab/inview">
    <i class="icon ion-cash"></i>
    Total Income
      <b><span class="item-note" id="tinc">

    </span></b>
  </a>
  <a class="item item-icon-left" href="./index.html#/tab/exview">
    <i class="icon ion-card"></i>
    Total Expenses
      <b><span class="item-note" id="texp">

    </span></b>
  </a>
  <a class="item item-icon-left" href="#">
    <i class="icon ion-social-usd"></i>
    Balance
    <b><span class="item-note" id="tbal">


    </span></b>
  </a>


  <a class="item item-icon-left" href="./index.html#/tab/addg">
    <i class="icon ion-happy"></i>
    Goals
    <span class="badge badge-assertive" id="tgoal"></span>
  </a>


            <a class="item item-icon-left item-icon-right" href="./index.html#/tab/charts">
    <i class="icon ion-pie-graph"></i>
    Charts
    <i class="icon ion-arrow-right-c"></i>
  </a>
       </div>
      </div>
    </div>


    <div class="list card">
      <div class="item item-divider">Goal</div>
      <div class="item item-body" ng-repeat="g in gl">
        <div>
            <b>${{g.amount}}</b> for <b>{{g.name}}</b> by
<b>{{g.date.substring(0,10)}}</b> <b>{{g.time}}</b> months .
        </div>
      </div>
    </div>
  </ion-content>
</ion-view>
```

Tab-inview.html:

```
<ion-view view-title="Income List">
  <ion-content ng-controller="InviewCtrl">
      <ion-refresher
    pulling-text="Pull to refresh..."
    on-refresh="doRefresh()">
  </ion-refresher>
    <a class="item item-icon-left" href="./index.html#/inview/add">
    <i class="icon ion-plus-circled"></i>
    Add Incomes

  </a>
      <ion-list>
      <ion-item class="item-remove-animate item-avatar item-icon-right" ng-repeat="inc in
incomes" type="item-text-wrap" href="#/tab/inview/{{chat.id}}">
        <img src="http://cdn.mysitemyway.com/etc-mysitemyway/icons/legacy-
previews/icons/simple-red-glossy-icons-business/086760-simple-red-glossy-icon-business-
dollar-solid.png">
        <h2>{{inc.name}}</h2>
        <p>{{inc.amount}}</p>
          <p>{{inc.date.substring(0,10)}}</p>
        <i class="icon ion-chevron-right icon-accessory"></i>

        <ion-option-button class="button-assertive" ng-click="remove(chat)">
          Delete
        </ion-option-button>
      </ion-item>
    </ion-list>
  </ion-content>
</ion-view>
```

<u>Tab-exview.html:</u>

```
<ion-view view-title="Expenses List">
  <ion-content ng-controller="ExpenseviewCtrl">
   <ion-refresher
    pulling-text="Pull to refresh..."
    on-refresh="doRefresh()">
  </ion-refresher>
       <a class="item item-icon-left" href="./index.html#/exview/adde">
    <i class="icon ion-plus-circled"></i>
    Add Expenses
  </a>
      <ion-list>
      <ion-item class="item-remove-animate item-avatar item-icon-right" ng-repeat="exp in expenses" type="item-text-wrap" href="#/tab/exview/{{chat.id}}">
        <img src="http://cdn.mysitemyway.com/etc-mysitemyway/icons/legacy-previews/icons/simple-red-glossy-icons-business/086760-simple-red-glossy-icon-business-dollar-solid.png">
        <h2>{{exp.name}}</h2>
        <p>{{exp.amount}}</p>
          <p>{{exp.date.substring(0,10)}}</p>
        <i class="icon ion-chevron-right icon-accessory"></i>


        <ion-option-button class="button-assertive" ng-click="remove(chat)">
          Delete
        </ion-option-button>
      </ion-item>
    </ion-list>
  </ion-content>
</ion-view>
```

Tab-adde.html:

```
<ion-view view-title="Add Expense">
 <ion-content ng-controller="ExpenseController">
    <div></div>
    <div class="list">
  <label class="item item-input item-stacked-label">
    <span class="input-label">Expense Name</span>
    <input type="text" placeholder="ex.Food" name="name" id="name" data-ng-
model="expense.name">
  </label>
  <label class="item item-input item-stacked-label">
    <span class="input-label">Amount</span>
    <input type="text" placeholder="ex.10 $" name="amount" id="amount" data-ng-
model="expense.amount">
  </label>
  <label class="item item-input item-stacked-label">
    <span class="input-label">Date</span>
    <input type="date" placeholder="mm/dd/yy" name="date" id="date" data-ng-
model="expense.date">
  </label>
</div>
    <div><center><button class="button button-positive" id="expense" name="expense"
ng_click="Exadd(expense)">
  Add Expense
</button><br><br>
        <button class="button button-positive" id="back" name="back"
ng_click="exviewback()">
  Back
</button>
        </center></div>
    </ion-content>
     </ion-view>
```

Tab-addg.html:

```
<ion-view view-title="Add Goal">
 <ion-content ng-controller="GoalController">
    <div></div>
    <div class="list">
  <label class="item item-input item-stacked-label">
    <span class="input-label">Goal Name</span>
    <input type="text" placeholder="ex.BuyCar" name="name" id="name" data-ng-
model="goal.name">
  </label>
  <label class="item item-input item-stacked-label">
    <span class="input-label">Goal Amount</span>
    <input type="text" placeholder="ex.10 $" name="amount" id="amount" data-ng-
model="goal.amount">
  </label>
  <label class="item item-input item-stacked-label">
    <span class="input-label">Reach Date</span>
    <input type="date" placeholder="mm/dd/yy" name="date" id="date" data-ng-
model="goal.date">
  </label>
</div>
    <div><center><button class="button button-positive" id="goal" name="goal"
ng_click="Goaladd(goal)">
  Add Goal
</button></center></div>
    </ion-content>
     </ion-view>
```

Tab-add.html:

```
<ion-view view-title="Add Income">
 <ion-content ng-controller="IncomeController">
    <div></div>
    <div class="list">
  <label class="item item-input item-stacked-label">
    <span class="input-label">Income Name</span>
    <input type="text" placeholder="ex.Salary" name="name" id="name" data-ng-
model="income.name">
  </label>
  <label class="item item-input item-stacked-label">
    <span class="input-label">Amount</span>
    <input type="text" placeholder="ex.10 $" name="amount" id="amount" data-ng-
model="income.amount">
  </label>
  <label class="item item-input item-stacked-label">
    <span class="input-label">Date</span>
    <input type="date" placeholder="mm/dd/yy" name="date" id="date" data-ng-
model="income.date">
  </label>
</div>
    <div><center><button class="button button-positive" id="income" name="income"
ng_click="Inadd(income)">
  Add Income
</button></center></div>
    </ion-content>
     </ion-view>
```

Chart.html:

```
<ion-content ng-controller="ExampleController">
    <div class="card">
        <div class="item item-divider">
            A line chart
        </div>
        <div class="item item-text-wrap">
            <canvas id="line" class="chart chart-line" data="data" labels="labels" legend="true"
            series="series" options="{showTooltips: false}"></canvas>
        </div>
    </div>
    <div class="card">
        <div class="item item-divider">
            A bar chart
        </div>
        <div class="item item-text-wrap">
            <canvas id="bar" class="chart chart-bar" data="data" labels="labels" legend="true"
            series="series" options="{showTooltips: false}"></canvas>
        </div>
    </div>
</ion-content>
```

App.js

```
angular.module('starter', ['ionic', 'starter.controllers', 'starter.services', 'chart.js'])


.run(function($ionicPlatform) {
  $ionicPlatform.ready(function() {
    // Hide the accessory bar by default (remove this to show the accessory bar above the
        keyboard
    // for form inputs)
    if (window.cordova && window.cordova.plugins && window.cordova.plugins.Keyboard)
        {
      cordova.plugins.Keyboard.hideKeyboardAccessoryBar(true);
      cordova.plugins.Keyboard.disableScroll(true);


    }
    if (window.StatusBar) {
      // org.apache.cordova.statusbar required
      StatusBar.styleLightContent();
    }
  });
})



.config(function($stateProvider, $urlRouterProvider) {


  // Ionic uses AngularUI Router which uses the concept of states
  // Learn more here: https://github.com/angular-ui/ui-router
  // Set up the various states which the app can be in.
  // Each state's controller can be found in controllers.js
  $stateProvider
  // setup an abstract state for the tabs directive

.state('main',{
  url:'/expense',
      abstract:true,
      template:'<ion-nav-view name="Dashboard"></ion-nav-view>'
  })


  .state('login', {
    url: '/login',
     templateUrl: 'templates/tab-login.html',
        controller: 'LoginCtrl'
     /* views: {
      'tab-login': {

      }
    }*/
  })


  .state('register', {
    url: '/register',
     templateUrl: 'templates/tab-register.html',
        controller: 'RegisterCtrl'
     /* views: {
      'tab-register': {

      }
    }*/
```

```
})

.state('update', {
  url: '/update',
   templateUrl: 'templates/tab-update.html',
      controller: 'UpdateCtrl'
    /*views: {
    'tab-update': {

    }
  }*/
})

.state('delete', {
  url: '/delete',
  templateUrl: 'templates/tab-delete.html',
      controller: 'DeleteCtrl'
    /*views: {
    'tab-delete': {

    }
  }*/
})


// Each tab has its own nav history stack:

 .state('tab', {
  url: '/tab',
  abstract: true,
  templateUrl: 'templates/tabs.html'
})

    .state('tab.dash', {
  url: '/dash',
  views: {
    'tab-dash': {
      templateUrl: 'templates/tab-dash.html',
      controller: 'DashCtrl'
    }
  }
})



     .state('tab.inview', {
  url: '/inview',
  views: {
    'tab-inview': {
      templateUrl: 'templates/tab-inview.html',
      controller: 'InviewCtrl'
    }
  }
})

.state('tab.income-detail', {
    url: '/inview/:chatId',
    views: {
      'tab-inview': {
        templateUrl: 'templates/income-detail.html',
        controller: 'IncomeDetailCtrl'
      }
    }
  })

    .state('add', {
  url: '/inview/add',
  templateUrl: 'templates/tab-add.html',
      controller: 'AddCtrl'
    /*views: {
    'tab-add': {

    }
  }*/
})



.state('tab.exview', {
  url: '/exview',
  views: {
    'tab-exview': {
      templateUrl: 'templates/tab-exview.html',
      controller: 'ExviewCtrl'
    }
  }
})

.state('tab.chat-detail', {
    url: '/exview/:chatId',
    views: {
      'tab-exview': {
        templateUrl: 'templates/chat-detail.html',
        controller: 'ChatDetailCtrl'
      }
    }
  })

.state('adde', {
  url: '/exview/adde',
   templateUrl: 'templates/tab-adde.html',
      controller: 'AddeCtrl'
   /* views: {
    'tab-exview': {


    }
  }*/
})



 .state('tab.addg', {
  url: '/addg',
  views: {
    'tab-addg': {
      templateUrl: 'templates/tab-addg.html',
      controller: 'AddgCtrl'
    }
  }
})
```

```
 .state('tab.charts', {
    url: '/charts',
    views: {
      'tab-charts': {
        templateUrl: 'templates/tab-charts.html',
        controller: 'ExampleController'
      }
    }
  })

.state('tab.account', {
    url: '/account',
    views: {
      'tab-account': {
        templateUrl: 'templates/tab-account.html',
        controller: 'AccountCtrl'
      }
    }
  });




  // if none of the above states are matched, use this as the fallback
  $urlRouterProvider.otherwise('/register');

});
```

Controller.js

```
angular.module('starter.controllers', ['ionic', 'ui.router', 'starter.services'])


.controller('DashCtrl', function($scope) {})


.controller('LoginCtrl', function($scope) {})


.controller('RegisterCtrl', function($scope) {})


.controller('UpdateCtrl', function($scope) {})



.controller('DeleteCtrl', function($scope) {})


.controller('DashController', function($scope,$http) {
  totexp=0;
  totinc=0;
  $scope.doRefresh = function() {
    totexp=0;
  totinc=0;

      $http.get('https://api.mongolab.com/api/1/databases/expensetracker/collections/expens
      e?apiKey=TBtDR-i4rKwMVVXGo4rvWkyPFyIt369K').
  success(function (data) {
    $scope.expenses = data;
    //$state.go($state.current, {}, {reload: true});
   expensed= data;
   //console.log(expensed);
   //console.log(expensed.length);
   //console.log(expensed[1].amount);
   totexp=0;
```

```
 for(var i=0;i<expensed.length;i++)
 {
       totexp=totexp+parseInt(expensed[i].amount);
 }
 console.log(totexp);
 document.getElementById("texp").innerHTML = totexp;

})

        $http.get('https://api.mongolab.com/api/1/databases/expensetracker/collections/incom
        e?apiKey=TBtDR-i4rKwMVVXGo4rvWkyPFyIt369K').
success(function (data) {
    $scope.income = data;
 incomed=data;
 //console.log(tex);
//totinc=0;
 for(var i=0;i<incomed.length;i++)
 {
        totinc=totinc+parseInt(incomed[i].amount);
 }
    document.getElementById("tinc").innerHTML = totinc;
 var bal=0;
 console.log(totexp);
 bal=totinc-totexp;
  document.getElementById("tbal").innerHTML = bal;
 //console.log(bal);
})

        $http.get('https://api.mongolab.com/api/1/databases/expensetracker/collections/goal?a
        piKey=TBtDR-i4rKwMVVXGo4rvWkyPFyIt369K').
```

```
success(function (data) {
    $scope.goal = data;
 var goald=data;
 var len=goald.length;
 document.getElementById("tgoal").innerHTML = len;
 var cdate= new Date();



 //console.log(cdate+" && "+gdate);
 //console.log(cdate.getFullYear());
 //console.log(gdateo.getFullYear());

 //console.log(tmonths);
 var goaltm=goald;
 console.log(goaltm);
 for(var i=0;i<len;i++)
 {
    console.log(goald[i].name);
    var gdate= goald[i].date;
    var gdateo=new Date(gdate);
     var tmonths = (gdateo.getFullYear() - cdate.getFullYear())*12 + (gdateo.getMonth() -
        cdate.getMonth());
     goaltm[i].time=tmonths;
 }
 $scope.gl=goaltm;
})
   $scope.$broadcast('scroll.refreshComplete');


};
```

```
        $http.get('https://api.mongolab.com/api/1/databases/expensetracker/collections/expens
        e?apiKey=TBtDR-i4rKwMVVXGo4rvWkyPFyIt369K').
success(function (data) {
    $scope.expenses = data;
    //$state.go($state.current, {}, {reload: true});
 expensed= data;
 //console.log(expensed);
 //console.log(expensed.length);
 //console.log(expensed[1].amount);
 totexp=0;
 for(var i=0;i<expensed.length;i++)
 {
       totexp=totexp+parseInt(expensed[i].amount);
 }
 console.log(totexp);
 document.getElementById("texp").innerHTML = totexp;

})


        $http.get('https://api.mongolab.com/api/1/databases/expensetracker/collections/incom
        e?apiKey=TBtDR-i4rKwMVVXGo4rvWkyPFyIt369K').
success(function (data) {
    $scope.income = data;
 incomed=data;
 //console.log(tex);
//totinc=0;
 for(var i=0;i<incomed.length;i++)
 {
        totinc=totinc+parseInt(incomed[i].amount);
 }
```

```
    document.getElementById("tinc").innerHTML = totinc;
 var bal=0;
 console.log(totexp);
 bal=totinc-totexp;
  document.getElementById("tbal").innerHTML = bal;
 //console.log(bal);
})


        $http.get('https://api.mongolab.com/api/1/databases/expensetracker/collections/goal?a
        piKey=TBtDR-i4rKwMVVXGo4rvWkyPFyIt369K').
success(function (data) {
    $scope.goal = data;
 var goald=data;
 var len=goald.length;
 document.getElementById("tgoal").innerHTML = len;
 var cdate= new Date();
 var goaltm=goald;
 console.log(goaltm);
 for(var i=0;i<len;i++)
 {
    console.log(goald[i].name);
    var gdate= goald[i].date;
    var gdateo=new Date(gdate);
     var tmonths = (gdateo.getFullYear() - cdate.getFullYear())*12 + (gdateo.getMonth() -
        cdate.getMonth());
     goaltm[i].time=tmonths;
 }
 $scope.gl=goaltm;
})
```

```
})



.controller("ExampleController", function($scope,$http) {


 //
  //var monthval;

        $http.get('https://api.mongolab.com/api/1/databases/expensetracker/collections/incom
        e?apiKey=TBtDR-i4rKwMVVXGo4rvWkyPFyIt369K').
   success(function (data) {
     // $scope.incomes = data;
      chartd=data;
      monthval=[0,0,0,0,0,0,0,0,0,0,0,0,0];
      console.log(monthval[8]+' hey value 8');
      var month=0;
      for(var i=0;i<chartd.length;i++)
      {
      var cdate=chartd[i].date;
      var cdateo= new Date(cdate);
      month=parseInt(cdateo.getMonth());
         console.log(month+ ' inside for')
      if(month==8)
      {
         monthval[month]=monthval[month]+parseInt(chartd[i].amount);
      }
      else if(month==9)
      {
         monthval[month]=monthval[month]+parseInt(chartd[i].amount);
      }
     else if(month==10)
    {
       monthval[month]=monthval[month]+parseInt(chartd[i].amount);
    }
        else if(month==11)
    {
       monthval[month]=monthval[month]+parseInt(chartd[i].amount);
    }
        else if(month==12)
    {
       monthval[month]=monthval[month]+parseInt(chartd[i].amount);
    }
       else{ }
    //console.log(month +' '+i);
}
    console.log(monthval[10]);


       $http.get('https://api.mongolab.com/api/1/databases/expensetracker/collections/expens
       e?apiKey=TBtDR-i4rKwMVVXGo4rvWkyPFyIt369K').
success(function (data) {


        chartde=data;
    monthvale=[0,0,0,0,0,0,0,0,0,0,0,0,0];
    console.log(monthvale[8]+' hey value 8');
    var monthe=0;
    for(var i=0;i<chartde.length;i++)
    {
    var cdatee=chartde[i].date;
    var cdateoe= new Date(cdatee);
    monthe=parseInt(cdateoe.getMonth());
```

```
        console.log(monthe+ ' inside for exp')
    if(monthe==8)
    {
        monthvale[monthe]=monthvale[monthe]+parseInt(chartde[i].amount);
    }
    else if(monthe==9)
    {
        monthvale[monthe]=monthvale[monthe]+parseInt(chartde[i].amount);
    }
     else if(monthe==10)
    {
        monthvale[monthe]=monthvale[monthe]+parseInt(chartde[i].amount);
    }
         else if(monthe==11)
    {
        monthvale[monthe]=monthvale[monthe]+parseInt(chartde[i].amount);
    }
         else if(monthe==12)
    {
        monthvale[monthe]=monthvale[monthe]+parseInt(chartde[i].amount);
    }
        else{ }
    //console.log(month +' '+i);
}


     $scope.labels = ["August", "September", "October", "November", "December"];
$scope.series = ['Income', 'Expense'];
$scope.data = [
    [monthval[7], monthval[8], monthval[9], monthval[10], monthval[11]],
        [monthvale[7], monthvale[8], monthvale[9], monthvale[10], monthvale[11]]
    ];


         })


    })
})




.controller('InviewCtrl', function($scope,$http) {



            $http.get('https://api.mongolab.com/api/1/databases/expensetracker/collections/incom
            e?apiKey=TBtDR-i4rKwMVVXGo4rvWkyPFyIt369K').
    success(function (data) {
        $scope.incomes = data;
    })
  $scope.remove = function(chat) {
    //Chats.remove(chat);
  };
    $scope.doRefresh = function() {

            $http.get('https://api.mongolab.com/api/1/databases/expensetracker/collections/incom
            e?apiKey=TBtDR-i4rKwMVVXGo4rvWkyPFyIt369K').
    success(function (data) {
        $scope.incomes = data;
    })
       $scope.$broadcast('scroll.refreshComplete');


  };
})
```

```
.controller('IncomeDetailCtrl', function($scope, $stateParams, Chats) {
  $scope.chat = Chats.get($stateParams.chatId);
})

    .controller('AddCtrl', function($scope) {})

 .controller('AddeCtrl', function($scope) {})

.controller('ExviewCtrl', function($scope, $state) {})

.controller('ExpenseviewCtrl', function($scope,$http, $state,$rootScope) {

            $http.get('https://api.mongolab.com/api/1/databases/expensetracker/collections/expens
            e?apiKey=TBtDR-i4rKwMVVXGo4rvWkyPFyIt369K').
    success(function (data) {
        $scope.expenses = data;
        //$state.go($state.current, {}, {reload: true});
    })
  $scope.remove = function(chat) {
    //Chats.remove(chat);
  };
    $scope.doRefresh = function() {

            $http.get('https://api.mongolab.com/api/1/databases/expensetracker/collections/expens
            e?apiKey=TBtDR-i4rKwMVVXGo4rvWkyPFyIt369K').
    success(function (data) {
        $scope.expenses = data;
        //$state.go($state.current, {}, {reload: true});


    })
```

```
       $scope.$broadcast('scroll.refreshComplete');

  };

})




.controller('AddgCtrl', function($scope) {})




    .controller('AccountCtrl', function($scope) {
  $scope.settings = {
    enableFriends: true
  };
})

.controller('AccountController', function($scope, $state) {
   $scope.takelog = function() {
  $state.go('login');
   }
})

.controller('UpdateController', function ($scope, $http, $httpParamSerializerJQLike,
            $ionicPopup) {

    //$scope.pageClass = 'update';
    $scope.update = function(username, oldpass, newpass) {

    $http({
        method: 'PUT',
        url :
           'https://api.mongolab.com/api/1/databases/expensetracker/collections/users?q={"name
           ":"'+username+'","password":"'+oldpass+'"}&apiKey=TBtDR-
           i4rKwMVVXGo4rvWkyPFyIt369K',
        data: JSON.stringify( { "$set" : { "password" : newpass } } ),
        contentType: "application/json"
    }).success(function(data) {
        console.log(data);
        var alertPopup = $ionicPopup.alert({
                title: 'Update Done !',
                okText: 'Check Details'
        });
        alertPopup.then(function() {

           //window.location.assign("");
        });
    })
    }
})

.controller('DeleteController', function ($scope, MongoRESTService) {
   /* $scope.Inadd = function (data) {
      var id = MongoRESTService.incomeadd(data);
      console.log(id);

    }*/
  })

 .controller('LoginController', function ($scope, $http, MongoRESTService, $state) {
    $scope.login = function (username, password) {
      //console.log(username);
        //alert(username+" "+password);
      var result = MongoRESTService.login(username, password, function (result) {
        console.log(result+' in controller')
        if (result.length = 1) {
          $state.go('tab.dash');
        } else {
          alert('Sorry, check your credentials.')
        }


      });
    }
  })
.controller('RegisterController', function ($scope, MongoRESTService,$state) {
    $scope.create = function (data) {
        var usr=data;
        if(usr.name==null || usr.password==null || usr.email==null)
        {
            alert('Please enter all details');
        }
        else
        {
        var id = MongoRESTService.register(data);


        }
            //console.log(id);


    }
    $scope.takelog = function() {
  $state.go('login');
```

```
}

})

.controller('IncomeController', function ($scope, MongoRESTService) {
    $scope.Inadd = function (data) {
      var id = MongoRESTService.incomeadd(data);



    }
})

.controller('ExpenseController', function ($scope, MongoRESTService, $state,$rootScope) {
    $scope.Exadd = function (data) {
      var id = MongoRESTService.expenseadd(data);
      //console.log(id);

    }
    $scope.exviewback = function() {
  $state.go('tab.exview');
    }
})

.controller('GoalController', function ($scope, MongoRESTService) {
    $scope.Goaladd = function (data) {
      var id = MongoRESTService.goaladd(data);
      //console.log(id);

    }
});
```

Services.js

```
var services = angular.module("starter.services", []);


var url = "http://expense.eu-gb.mybluemix.net/";
services.factory('MongoRESTService', function($http, $ionicPopup, $state) {
  return {
   login: function(username, password, callback) {


    var res = $http.get(url+"user"+"?name="+username+"&password="+password);
    console.log(res+' this is res')


    res.success(function(data, status, headers, config) {
     console.log(data+' data');
      var lr=data;
      console.log(lr+'lr data');
      var cmp='[object Object]';
      if(lr==cmp)
     {
      var alertPopup = $ionicPopup.alert({
              title: 'Success',
              template: 'Login Success.'
              });
        callback(data);
     }
     else
     {
         var alertPopup = $ionicPopup.alert({
              title: 'Failed',
              template: 'Login Failed.'
              });
    }
},

register: function(user) {
  var res = $http.post(url+"user", user);
  res.success(function(data, status, headers, config) {
    var alertPopup = $ionicPopup.alert({
              title: 'Success',
              template: 'user registered successfully.'
              });
      $state.go('login');

  });
  res.error(function(data, status, headers, config) {

  });
},

incomeadd: function(income) {
  //console.log(income);
  var res = $http.post(url+"income", income);
    //var userdata=user;
    //console.log(name);
  res.success(function(data, status, headers, config) {
    //console.log(data);
     var alertPopup = $ionicPopup.alert({
              title: 'Success',
              template: 'Income Added Successfully.'
              });
      //window.location = "/www/index.html#/tab/add";
      $state.go('tab.inview');
  });
  res.error(function(data, status, headers, config) {
   // console.log(data);
  });
},

  expenseadd: function(expense) {
  //console.log(expense);
  var res = $http.post(url+"expense", expense);
    //var userdata=user;
    //console.log(name);
  res.success(function(data, status, headers, config) {
    //console.log(data);
     var alertPopup = $ionicPopup.alert({
             title: 'Success',
             template: 'Expense Added Successfully.'
             });
      //window.location = "/www/index.html#/tab/adde";
      $state.go('tab.exview');
      reload:true
  });
  res.error(function(data, status, headers, config) {
    //console.log(data);
  });
},


  goaladd: function(goal) {
  var res = $http.post(url+"goal", goal);
```

```
          res.success(function(data, status, headers, config) {
           var alertPopup = $ionicPopup.alert({
                    title: 'Success',
                    template: 'Goal Added Successfully.'
                    });
          });
        res.error(function(data, status, headers, config) {
         });
      }


    }
});
```

User Manual:

Introduction:

In a busy day to day life, unknowingly a man spends lot of money for various needs. Sometimes these expenses may exceed his/her monthly budget or income. In order to organize cash flow and financial management there is a necessity of a personal expense tracking application that stores your daily expenses and provides a clear view of your finances.

So in order to track all our expenses and incomes we have developed Expense Tracker, which is used maintain history of all records. Also we have added charts so that user can easily observe all his records and compare with previous months.

Usage of System:

Initially application is to be downloaded while installing application request internet connectivity for the application. After installation user will see registration page, user need to register if he doesn't have an account. After registration user should login into the application. After a successful login dashboard is displayed, initially the dashboard values are 0 as the user is new. User can add an expense or income by opening incomes/expenses list and clicking on add. User can also see charts.

Constraints:

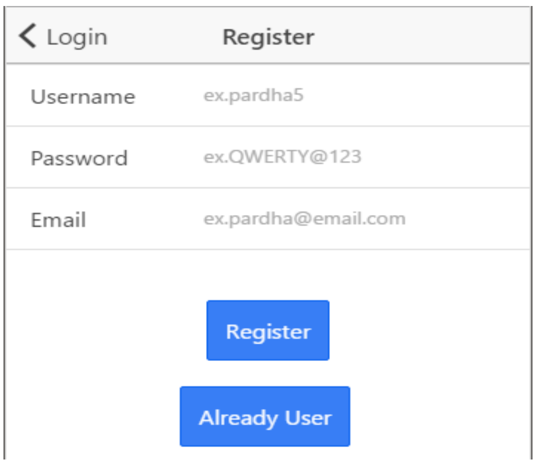
User must have internet connectivity to use the application.

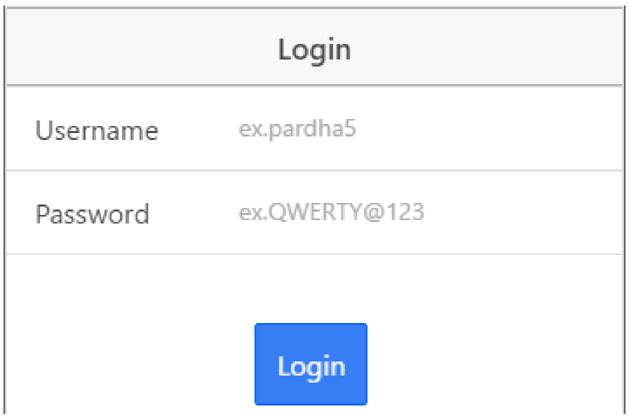
Error Handling:

If user mobile internet connectivity is disconnected operations cannot be performed in the application, so if any operation is performed without connectivity it displays an error for Internet Connectivity.

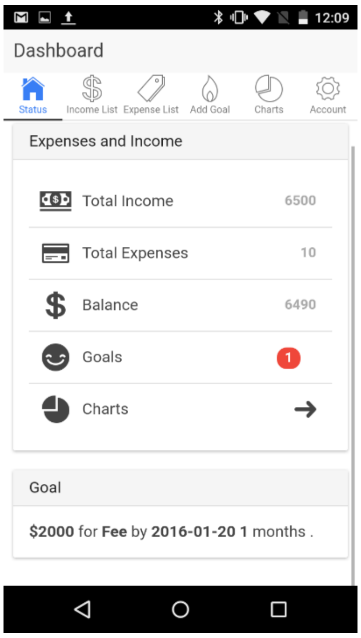
Sample Interaction:

When the application is opened by the user initially registration page is shown, if user doesn't have an account user need to register using following page. After registration user can login using the below page. After successful login an alert is displayed showing the dashboard.





Dashboard display total expenses, total incomes, calculated balance and goals. All goals of user can be seen bottom of the dashboard.


12:09

Dashboard

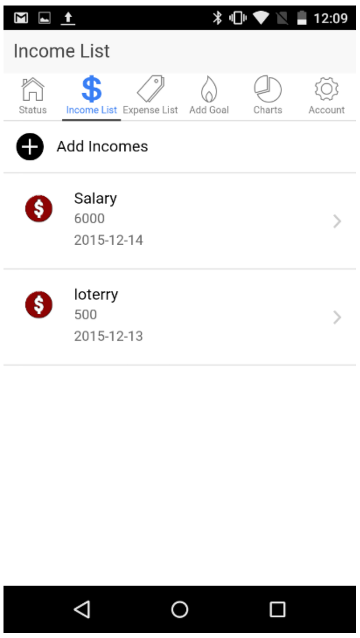
Status | Income List | Expense List | Add Goal | Charts | Account

Expenses and Income

Total Income	6500
Total Expenses	10
Balance	6490
Goals	1
Charts	→

Goal

$2000 for **Fee** by **2016-01-20** **1** months .


12:09

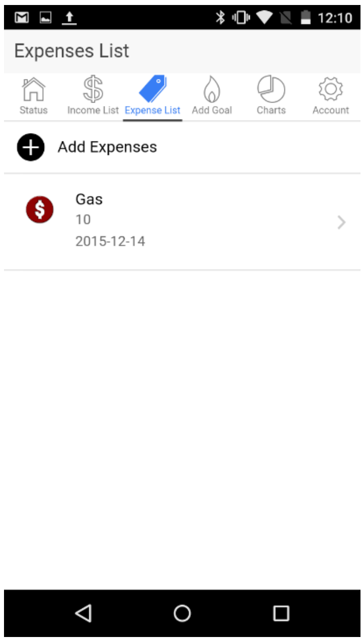
Income List

Status | Income List | Expense List | Add Goal | Charts | Account

Add Incomes

Salary
6000
2015-12-14

loterry
500
2015-12-13


12:10

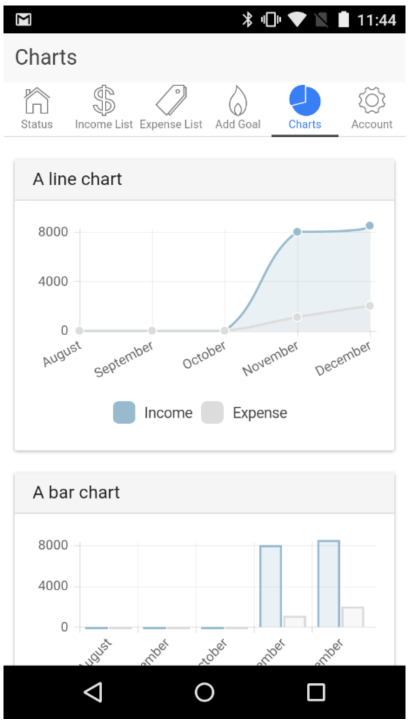
Expenses List

Status | Income List | Expense List | Add Goal | Charts | Account

Add Expenses

Gas
10
2015-12-14

If user click on Total income/Expense user can observe Income/Expense list page which shows all the records of respective type as a list. If user want to add expense/income user should press Add Expense/Income button on the top of the list.

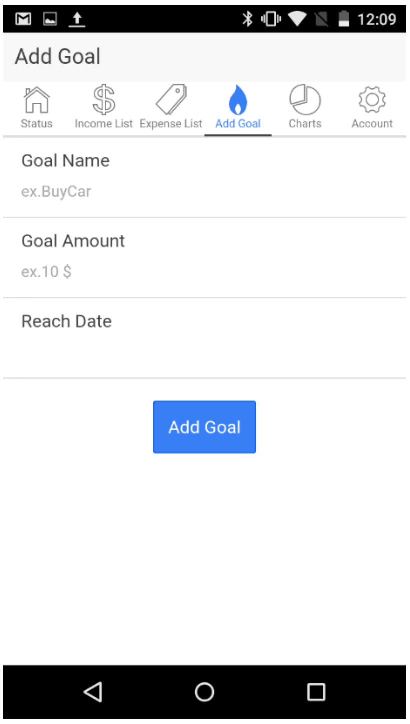

11:44

Charts

Status | Income List | Expense List | Add Goal | Charts | Account

A line chart

A bar chart


12:09

Add Goal

Status | Income List | Expense List | Add Goal | Charts | Account

Goal Name
ex.BuyCar

Goal Amount
ex.10 $

Reach Date

Add Goal

In the above screens user can observe the chart in which past 4 months of expenses is shown in the charts. To add goals, goals tab is used to add goal. Finally user can logout using logout button in account tab.

Project Management:

Project management has played the major part in the success of our project. Any project should have a plan to keep track of all the things that are implemented and to be implemented. Project plan for our project is so much important because we learnt many new technologies in this course and each time we have to implement those technologies what we learnt. For every increment we used to change some part of the project as we have a new technology that replaces the old ones.

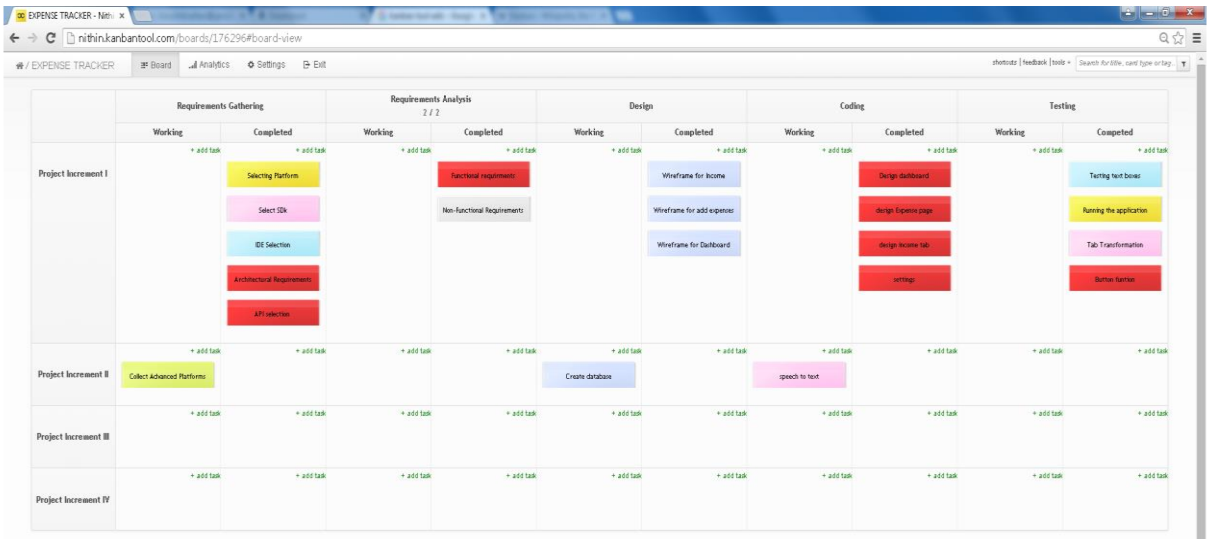
From the beginning we followed agile process model for our project. Agile process model is the highly efficient process model that is currently followed by many software practitioners to develop huge projects because it works on an incremental basis and gives developers a scope to change the requirements at each increments. It is because of this model we were able to provide a working piece of our project at each and every increment of our project. This agile process also gives the developers the ability to divide the project in to small parts and assign each part to individual developer. At the end we combined all the works done by each member and submitted it as an increment.

We used Kanban tool to divide and share project work among our team mates because it is easy view the tasks that are completed by different members in the project. It can be operated by the all the members in the project and it is very easy to understand the tasks completed in the project. It is the best tool to be used in project to analysis the present status of the project and also to now the status of individual work done by the project members. Kanban tool also provides us a facility to add tasks for next phase while we are working on the current phase.

In first increment we focused mainly on the requirements gathering, analysis and designing the user interface so we equally divided work among ourselves and started doing it as per the time allocated for it. From second increment to the end of the project for every increment we followed only one strategy, which was to gather all the requirements for that increment and to divide them among ourselves according to our interests. Each phase work was divided equally among all the members in the group. Project management for each phase will be found in detail at the end of each increment report in this document.

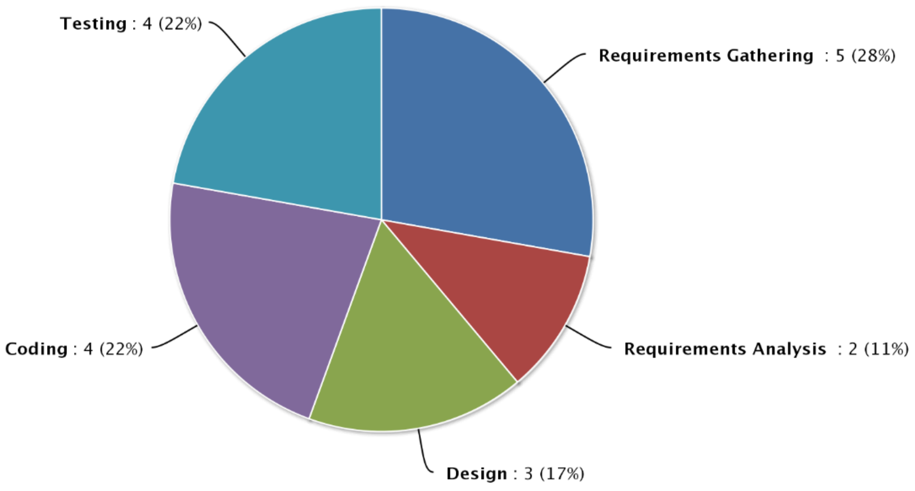
At first working as a team and dividing works among members has been tough but as the project went on it became easy for us. Every member in the group identified their task in each increment and completed them as per the given schedule. There are some situations where we lost to keep up with time but we tried to complete it by working extra hours. All tasks for each phase are added to the Kanban tool task board. Each and every member of the team are assigned equal number of tasks to complete with in a time limit. At the end all the work done by every member of the team is integrated during increment submission. We considered this project as a real project so we cannot think of any other different method.

Below we have provided the screenshots of work shared by us using Kanban tool task board panel and analysis of works to be done on the pie charts.

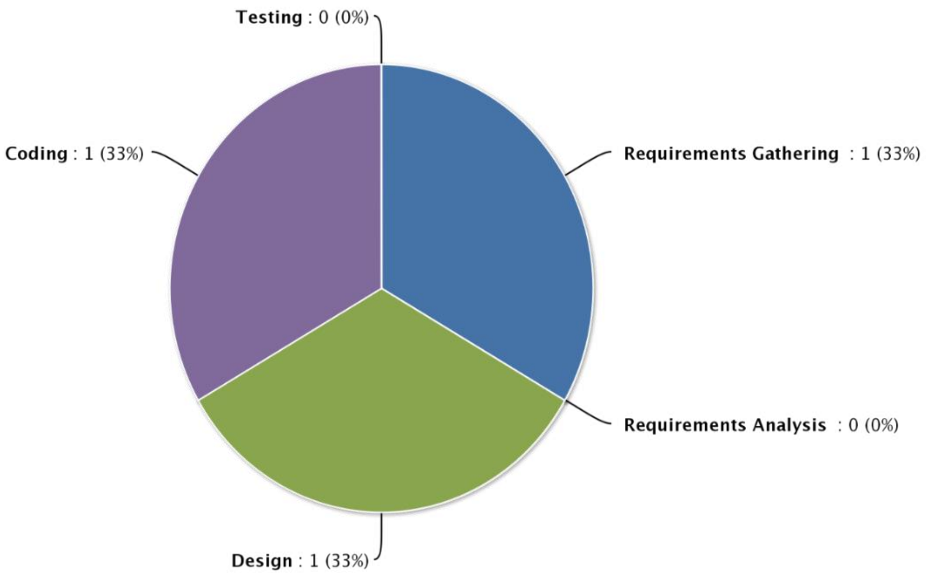

EXPENSE TRACKER - Nithi
nithin.kanbantool.com/boards/176296#board-view
/ EXPENSE TRACKER
Board
Analytics
Settings
Exit
shortcuts | feedback | tools
Search for title, card type or tag...
+ add task

	Requirements Gathering		Requirements Analysis 2 / 2		Design		Coding		Testing	
	Working	Completed	Working	Completed	Working	Completed	Working	Completed	Working	Competed
Project Increment I		Selecting Platform Select SDk IDE Selection Architectural Requirements API selection		Functional requirments Non-Functional Requirements		Wireframe for Income Wireframe for add expenses Wireframe for Dashboard		Design dashboard design Expense page design income tab settings		Testing text boxes Running the application Tab Transformation Button funtion
Project Increment II	Collect Advanced Platforms				Create database		speech to text			
Project Increment III										
Project Increment IV										

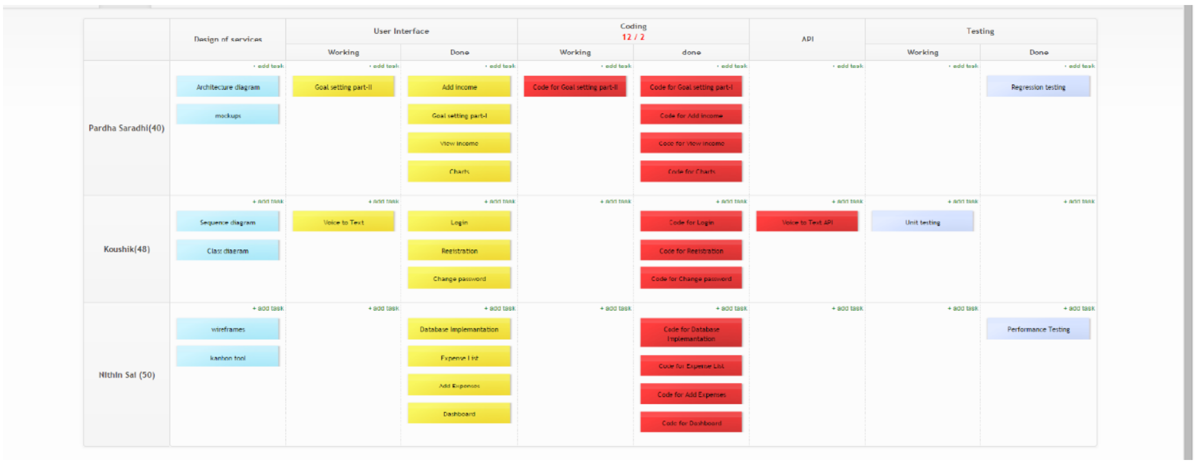
Task board for increment I and II



Pie chart for increment I work

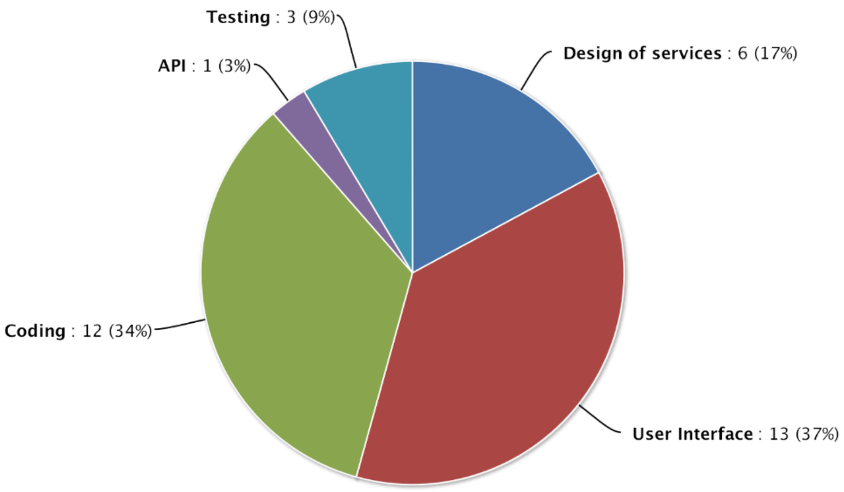


Pie chart for increment II work



	Design of services	User Interface		Coding 12 / 2		API	Testing	
		Working	Done	Working	done		Working	Done
Pardha Saradhi(40)	+ add task Architecture diagram mockups	+ add task Goal setting part-II	+ add task Add income Goal setting part-I View income Charts	+ add task Code for Goal setting part-II	+ add task Code for Goal setting part-I Code for Add income Code for View income Code for Charts	+ add task	+ add task	+ add task Regression testing
Koushik(48)	+ add task Sequence diagram Class diagram	+ add task Voice to Text	+ add task Login Registration Change password	+ add task	+ add task Code for Login Code for Registration Code for Change password	+ add task Voice to Text API	+ add task Unit testing	+ add task
Nithin Sai (50)	+ add task wireframes kanban tool	+ add task	+ add task Database Implemantation Expense List Add Expenses Dashboard	+ add task	+ add task Code for Database Implemantation Code for Expense List Code for Add Expenses Code for Dashboard	+ add task	+ add task	+ add task Performance Testing

Task Board for increment III



Pie chart for increment III