[Description](#_sm4ra97uwo11)

[Intended User](#_aws88pzfmqca)

[Features](#_zheq5430xrpq)

[User Interface Mocks](#_giquerrw6g46)

Login

Signup

Overview

Add Income

Add Expense

Choose Date Range

All Transactions

Categories

Settings

Widget Screen

[Key Considerations](#_gvcvmae8jn8u)

[How will your app handle data persistence?](#_v8my7nhtvz0m)

[Describe any corner cases in the UX.](#_gw69vjn1ico0)

[Describe any libraries you’ll be using and share your reasoning for including them.](#_6yqqubmw5bs)

[Describe how you will implement Google Play Services.](#_qrxg682nywe6)

[Next Steps: Required Tasks](#_v518bncmggeg)

[Task 1: Project Setup](#_8oe8zpk3qsmp)

[Task 2: Implement F](#_rzllsk6uqztx)irebase Auth and Database

Task 3: Implement UI for all screens

[Task 4:](#_kjidlkq4xm3u) Login

[Task 5: S](#_kjidlkq4xm3u)ignUp

Task 6: Overview

[Task 7: A](#_kjidlkq4xm3u)ll Transactions

[Task 8: C](#_kjidlkq4xm3u)ategories

[Task 9: S](#_kjidlkq4xm3u)ettings

[Task 10: I](#_kjidlkq4xm3u)mplement Widget

Task 11: Logout

**GitHub Username**: ChaituPenju

BucksTrack

# Description

BucksTrack is the easiest and most user friendly personal money tracker App.

It basically tracks the money spending based on expenses and income so that user will be able

to stick to a budget and save their money.

The user can enter his/her expenses and income, and can have control over their spendings.

# Intended User

People who want to track their money spendings can use this app.

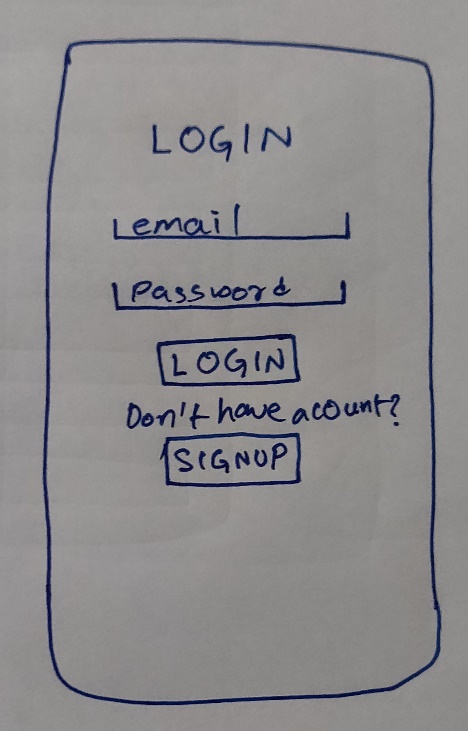
# Features

List the main features of your app. For example:

* Saves money spendings/transaction information
* Track spendings based on date range
* Shows an overview of total Incomes, Expenses and Balance amount left
* Shows category wise income and expenses
* User can see all the transactions from the beginning
* User can filter transactions based on particular week, particular month and chosen date range
* Add Income and Expense separately
* User can be notified of using/updating his income/expenses in the app based on time selected

# User Interface Mocks

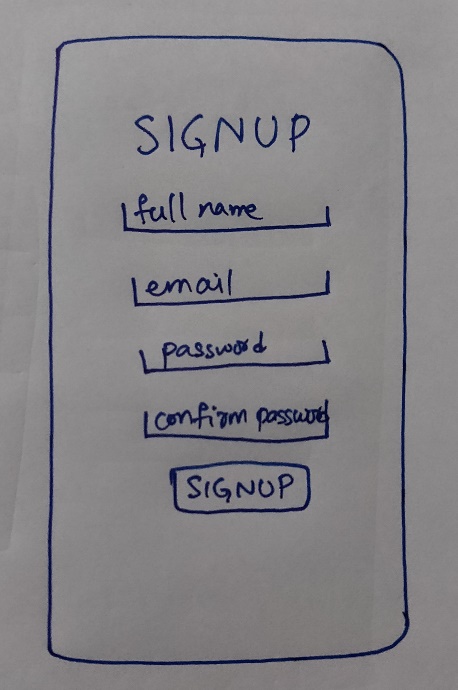
## Login Screen



In the login screen, the user will enter his email id and password and clicks on the button “Login”, the user gets logged into the app, before login the app validates the user input like empty fields, invalid login etc., if the credentials are wrong, the app prompts message via Toast.

If the user doesn’t have an account, he can click “Signup” button which takes him to signup screen.

## Signup Screen

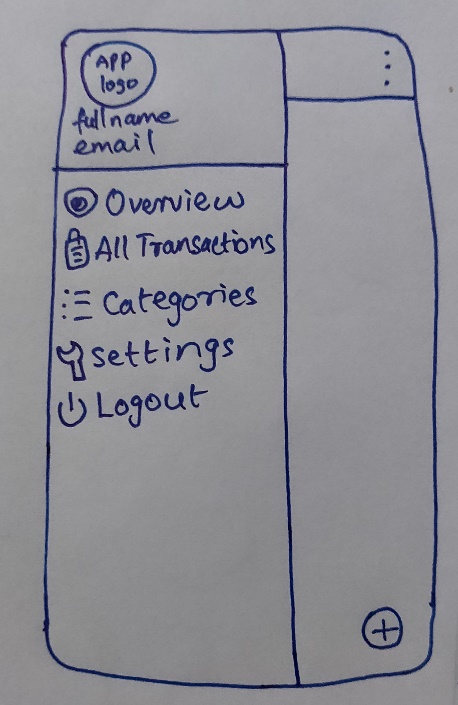
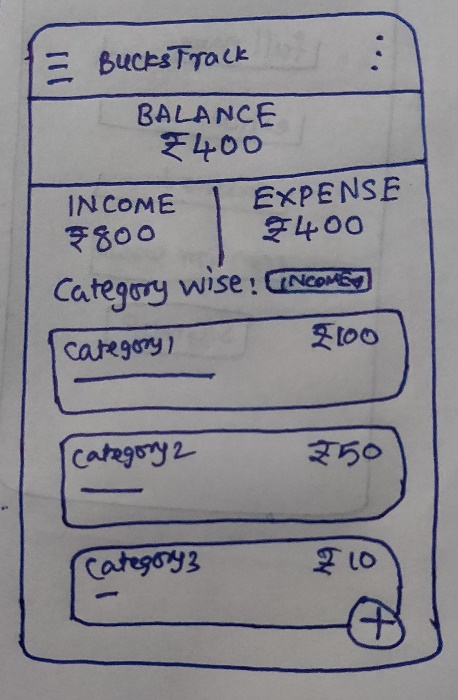


In the signup screen the user enters his full name, email id and password two times to confirm the password, and the app validates the user inputs and when clicked on “Signup” button, creates an account and the app gets logged in.

## App Navigation Drawer

The app

## Overview Screen

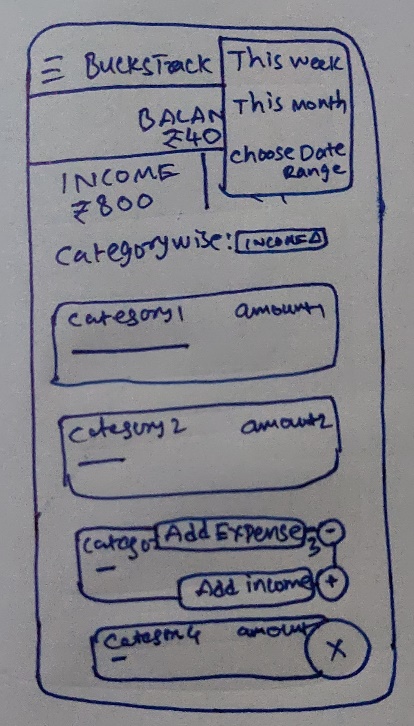
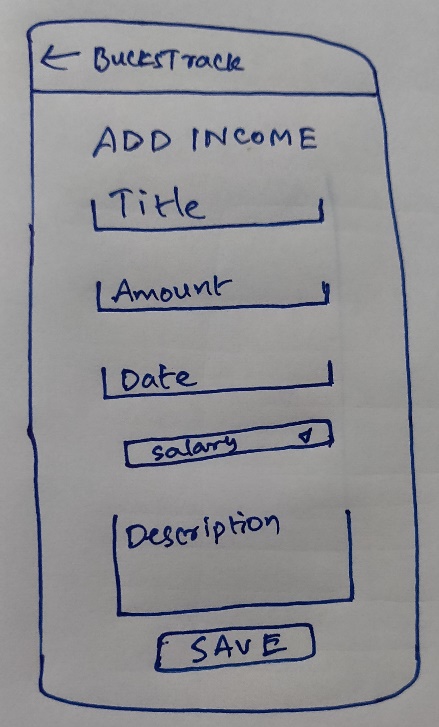
 

Replace the above image with your own mock [ click on the above image, then navigate to Insert → Image… ]

Provide descriptive text for each screen

## Add Income, Expense and Choose Date Range

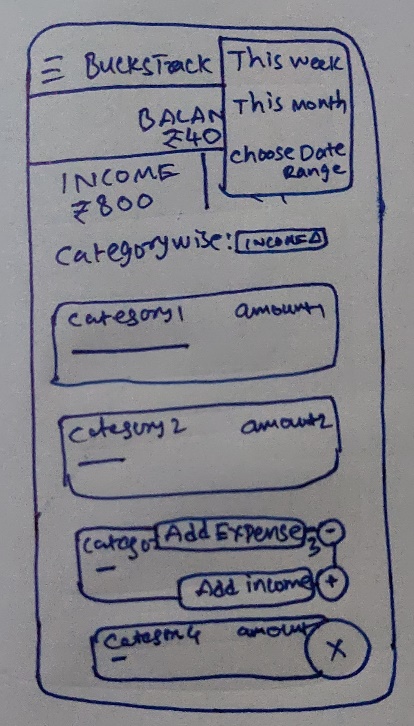
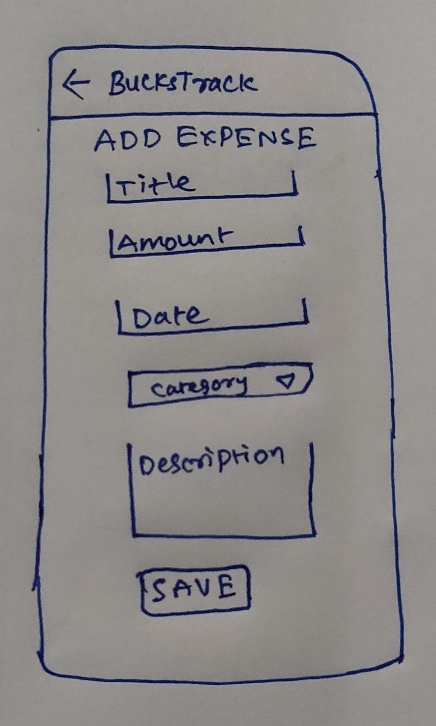
## Add Income Screen

Replace the above image with your own mock [ click on the above image, then navigate to Insert → Image… ]

Provide descriptive text for each screen

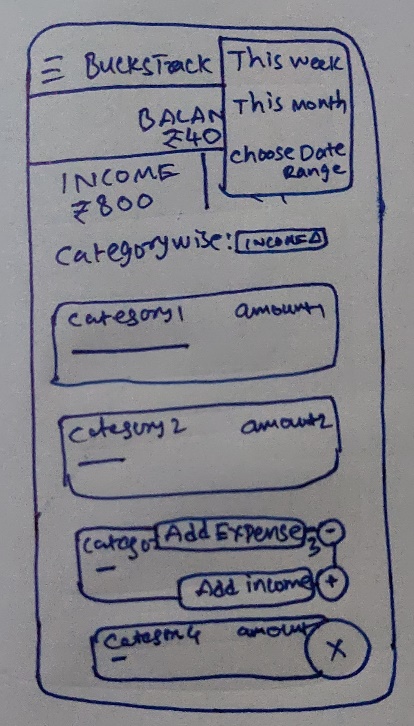
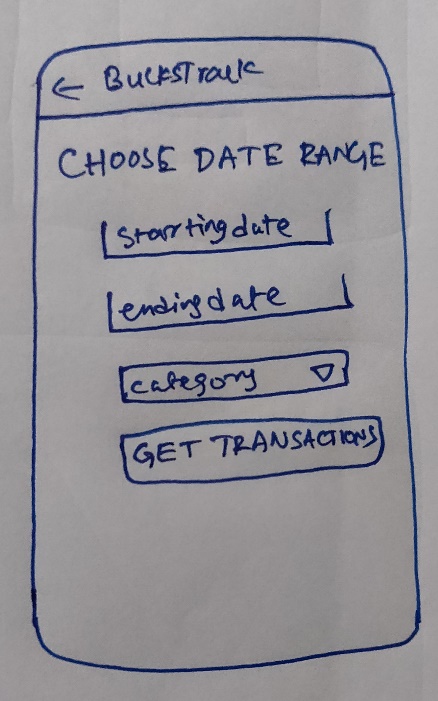
## Add Expense Screen

Replace the above image with your own mock [ click on the above image, then navigate to Insert → Image… ]

Provide descriptive text for each screen

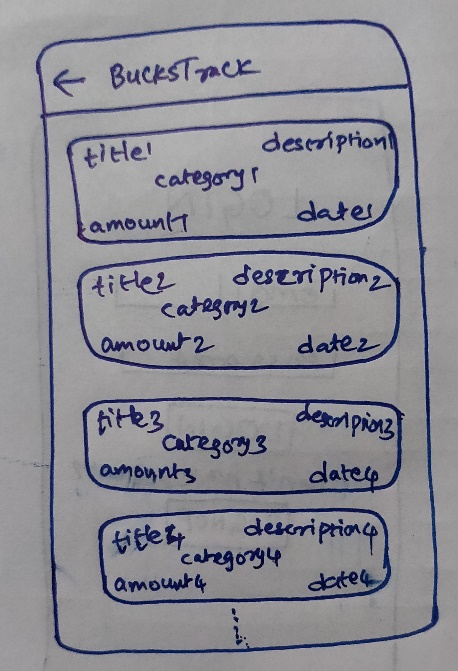
## Choose Date Range Screen

Replace the above image with your own mock [ click on the above image, then navigate to Insert → Image… ]

Provide descriptive text for each screen

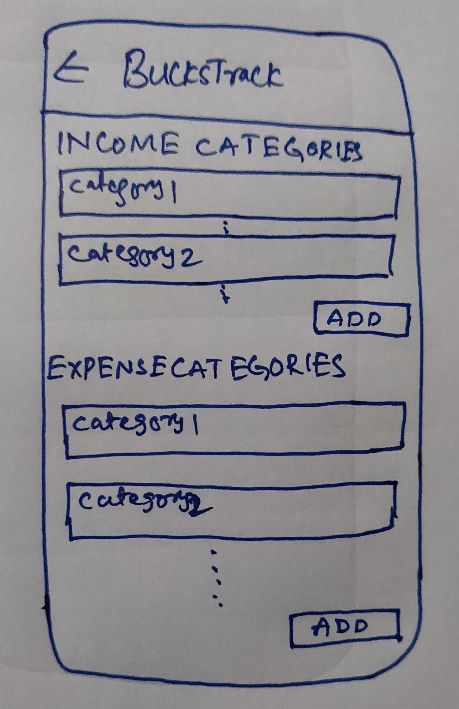
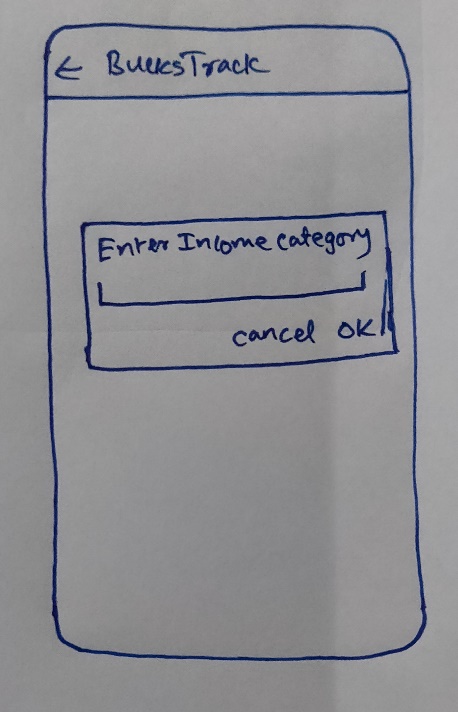
## All Transactions Screen



Replace the above image with your own mock [ click on the above image, then navigate to Insert → Image… ]

Provide descriptive text for each screen

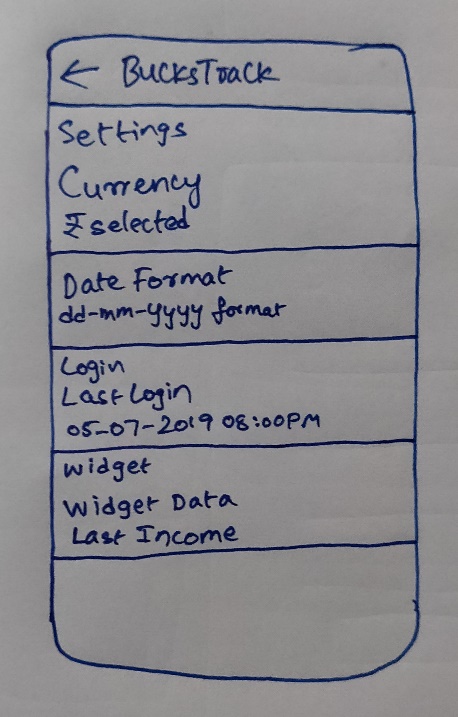
## Categories Screen

Replace the above image with your own mock [ click on the above image, then navigate to Insert → Image… ]

Provide descriptive text for each screen

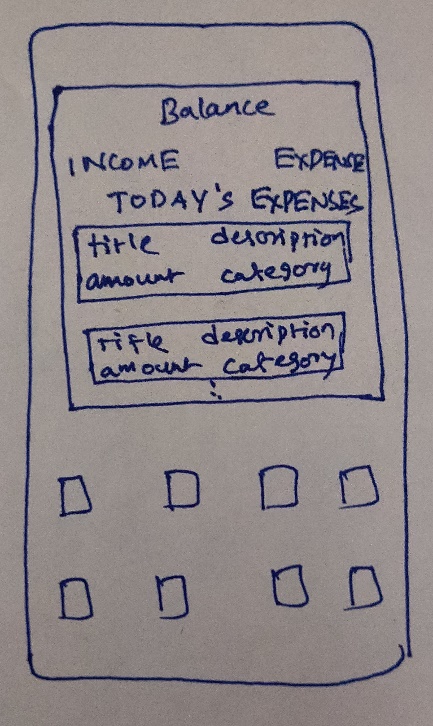
## Settings Screen



Replace the above image with your own mock [ click on the above image, then navigate to Insert → Image… ]

Provide descriptive text for each screen

## Widget Screen



Replace the above image with your own mock [ click on the above image, then navigate to Insert → Image… ]

Provide descriptive text for each screen

## Signup Screen



Replace the above image with your own mock [ click on the above image, then navigate to Insert → Image… ]

Provide descriptive text for each screen

Add as many screens as you need to portray your app’s UI flow.

# Key Considerations

### How will your app handle data persistence?

App handles the data using Firebase Realtime Database.

### Describe any edge or corner cases in the UX.

Swipe items to delete

Navigation Drawer is only available on Overview Activity

### Describe any libraries you’ll be using and share your reasoning for including them.

Clans/FloatingActionButton :

This library is used for material floating action buttons for animations on it.

Firebase Authentication :

This library is used for authenticating user while login and logout.

Firebase Database :

This library is used for storing the data in a firebase realtime database in json format.

### Describe how you will implement Google Play Services or other external services.

Describe which Services you will use and how.

# Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and break them down into tangible technical tasks that you can complete one at a time until you have a finished app.

## Task 1: Project Setup

* Create project BucksTrack in Android Studio
* Create required activities, fragments xml and java files
* Setup gradle dependencies using the stable version
* Configure the libraries used in the project

## Task 2: Implement Firebase Auth and Database

* Signin/Signup to Google Firebase Console and create the firebase android project there
* Register the app with firebase by submitting the SHA key and app ID
* Add firebase configuration file(google-services.json) and add google services plugin in build.gradle
* Add firebase SDKs firebase auth and firebase realtime database to the app gradle file

## Task 3: Implement UI for Each Activity and Fragment

* Create a basic navigation drawer layout and link all the activities using intents
* Build UI for All the Activities and Fragments, use appropriate icons
* Build the proper navigation within the app

## Task 4: Login

* Create appropriate EditTexts, TextViews and Button for getting username and password
* Add validation in code to avoid user submitting empty/wrong information
* Add functionality in java code of the activity to get credentials
* Add code for login the user into the app using Firebase Authentication LoginWithEmailAndPassword method

## Task 5: SignUp

* Create appropriate EditTexts, TextViews and Button for getting fullname, username password and confirm password
* Add validation in code to avoid user submitting empty/wrong information
* Add functionality in java code of the activity to get the user data
* Add code for signup the user into the app using Firebase Authentication SignupWithEmailAndPassword
* Initialize user data for the app and save user data using firebase realtime database

## Task 6: Overview

* Create the dashboard like main/home activity with total income amount, total expenses amount and balance amount
* Add two options in spinner “income category wise” and “expense category wise” to make a list of selected categories wise total amounts
* On click of FAB on bottom right corner, it opens up the two fab options, Add Income and Add Expense, intent to corresponding activity on option selection

**Add Income Activity**

* Create appropriate EditTexts, TextViews and Button for getting title, amount, date, category type and description
* For date edittext, add DatePickerDialog and update the selected date
* On clicking save button, the data gets saved in firebase realtime database
* Increment the income total in overview activity and calculate balance amount.

**Add Expense Activity**

* Create appropriate EditTexts, TextViews and Button for getting title, amount, date, category type and description
* For date edittext, add DatePickerDialog and update the selected date
* On clicking save button, the data gets saved in firebase realtime database
* Increment the expense total in overview activity and calculate balance amount.

## Task 7: All Transactions

* Set the recyclerview item as card view and populate it with data income and expense data from firebase database.
* Create the adapter and populate the recyclerview of this activity with the items
* Write code for Update functionality on click of an item
* Implement deletion of transaction on swiping left/right

## Task 8: Categories

* Set the values on the recyclerview item xml
* Populate the categories activities with recyclerview adapter and item created above
* Set update category on click of item and swipe to delete feature

## Task 9: Settings

* Use Preferences Screen and Fragment API for settings activity
* Add option to select currency symbol
* Add option to select date display format
* Write code to show Last Login info in settings activity
* Adding option to select what should be shown on widget

## Task 10: Implement Widget

* Set the data values on the widget from firebase database
* Set the data based on user selection from settings screen

## Task 11: Logout

* Create logout functionality using firebase authentication
* On choosing logout option, the user is logged out of the application

Add as many tasks as you need to complete your app.

**Submission Instructions**

* After you’ve completed all the sections, download this document as a PDF [ File → Download as PDF ]
  + Make sure the PDF is named “**Capstone\_Stage1.pdf**”
* Submit the PDF as a zip or in a GitHub project repo using the project submission portal

If using GitHub:

* Create a new GitHub repo for the capstone. Name it “**Capstone Project**”
* Add this document to your repo. Make sure it’s named “**Capstone\_Stage1.pdf**”