

Description

Intended User

Features

User Interface Mocks

Login

Signup

Overview

Add Income

Add Expense

Choose Date Range

All Transactions

Categories

Settings

Widget Screen

Key Considerations

How will your app handle data persistence?

Describe any corner cases in the UX.

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

Describe how you will implement Google Play Services.

Next Steps: Required Tasks

Task 1: Project Setup

Task 2: Implement Firebase Auth and Database

Task 3: Implement UI for all screens

Task 4: Login

Task 5: SignUp

Task 6: Overview

Task 7: All Transactions

Task 8: Categories

Task 9: Settings

Task 10: Implement 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

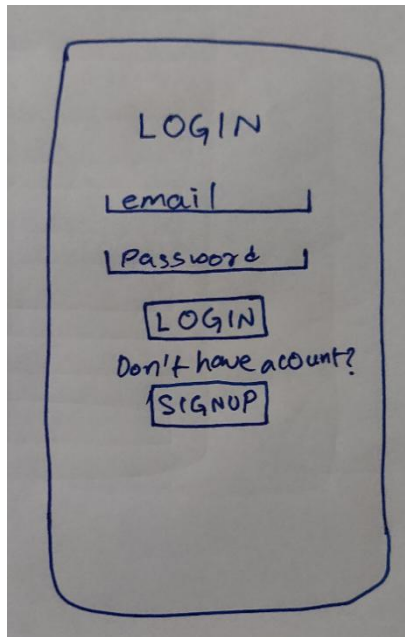
List the main features of your app. For example:

- 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 total 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 see/add/delete his income and expense categories separately.
- User can update his transaction/entry once entered.
- Can select and set his currency type and date format.

User Interface Mocks

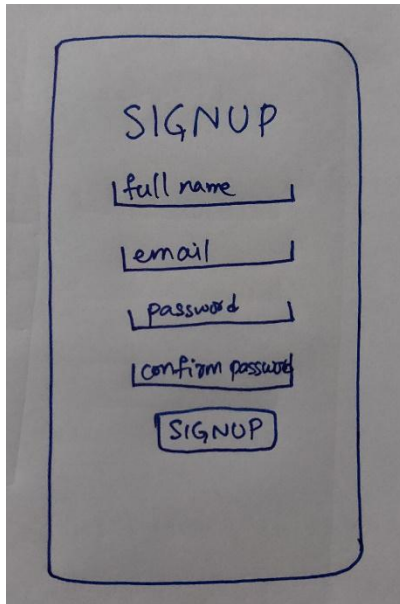
Login Screen



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



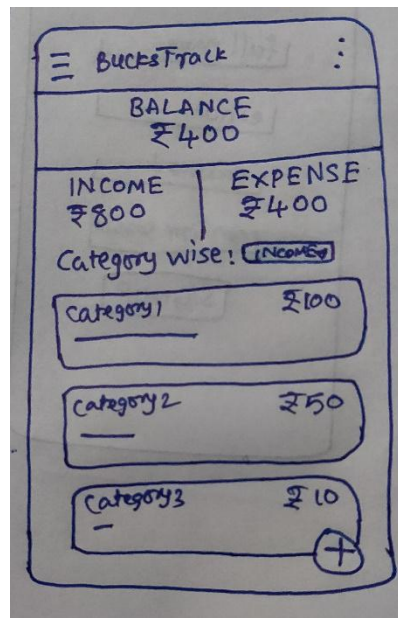
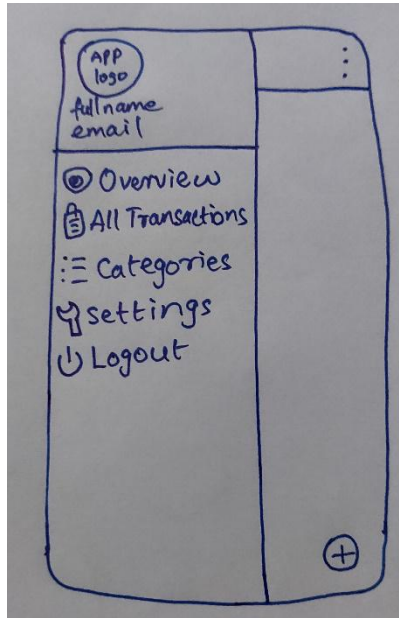
A hand-drawn sketch of a mobile app signup screen. The screen is enclosed in a rounded rectangle. At the top, the word "SIGNUP" is written in capital letters. Below it are four input fields, each represented by a horizontal line with a small vertical tick on the left side. The labels for these fields are "full name", "email", "password", and "confirm password", written in a cursive-like font. At the bottom of the form is a rectangular button labeled "SIGNUP" in capital letters.

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

After successful login the app, the home/base activity will be navigation drawer activity with app logo, full name and email id which are entered by the user.

Overview Screen

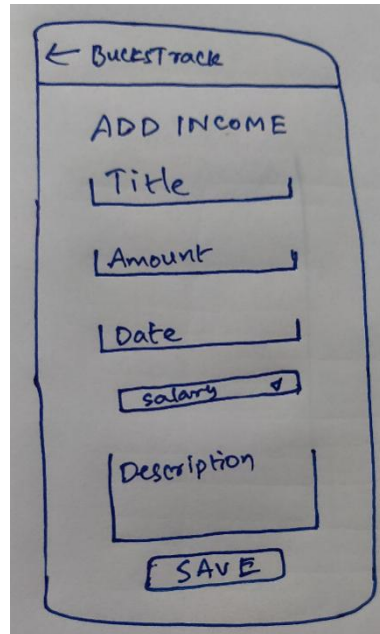
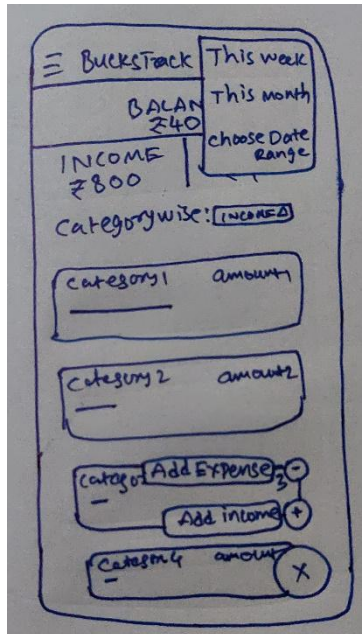


Along with that, we will have an overview activity where the total balance, income and expenses entered by the user are shown. If there is no data, the values show zero. Along with that, we have a spinner with “Income” and “Expense” categories, when you select appropriate option, the category wise total money spendings will be shown. There will also be an Floating action button and options menu on right bottom and top corner respectively.

Add Income, Expense and Choose Date Range

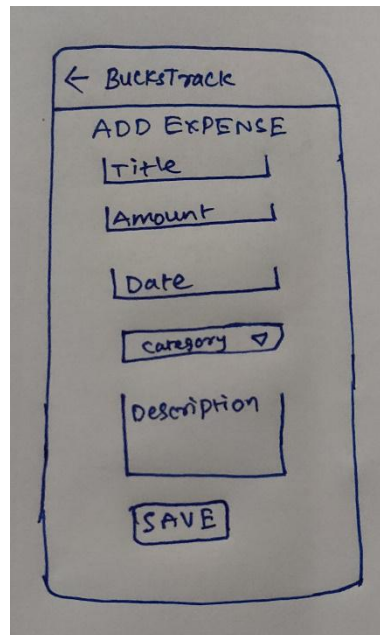
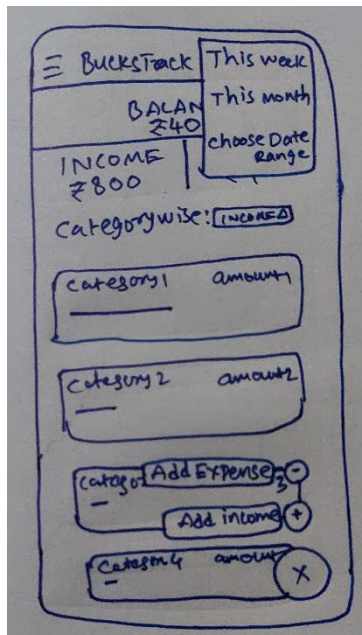
In the overview screen, on clicking of Floating action button(fab), it opens up two fab buttons, one for adding income and another for adding expense. On clicking the menu(three dots) icon on top right corner, opens up 3 options, this week, this month and choose Date Range. Choosing this week or this month option, on selection shows list of transactions of that particular week or month respectively.

Add Income Screen



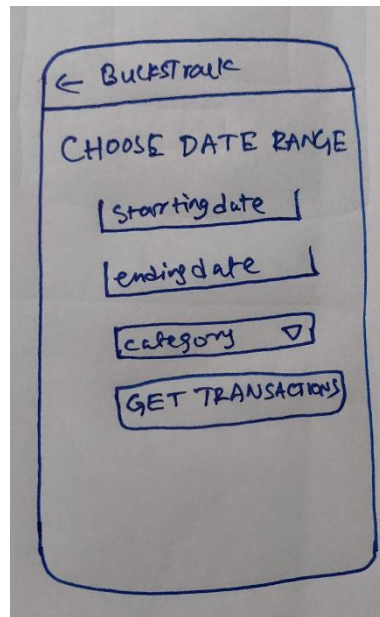
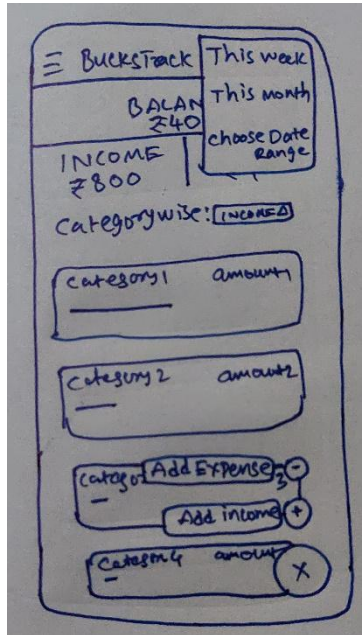
In the add income screen, the user will be asked to fill title, amount, date of transaction, type of income category and description. After validating all input fields, on click of save button, the data is stored in firebase database, and is shown as transaction to the user.

Add Expense Screen



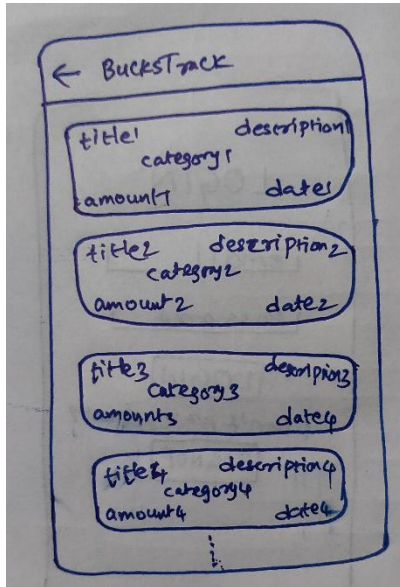
In the add expense activity, the user will be asked to fill title, amount, date of transaction, type of expense category and description. After validating all input fields, on click of save button, the data is stored in firebase database, and is shown as transaction to the user.

Choose Date Range Screen



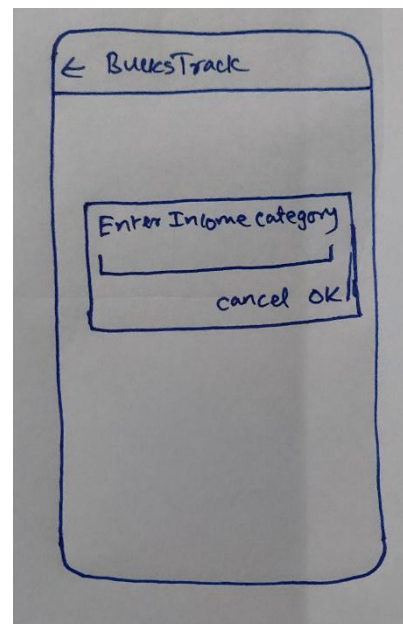
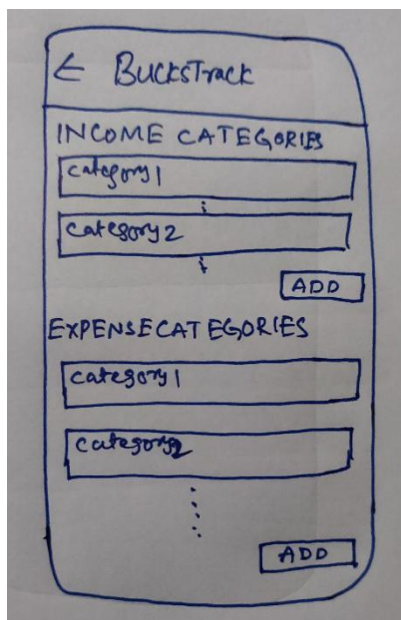
In the choose date range activity, the user will be asked the starting and ending date, and category(both income and expense), where the user can see all the transaction entered by him in between that particular dates.

All Transactions Screen



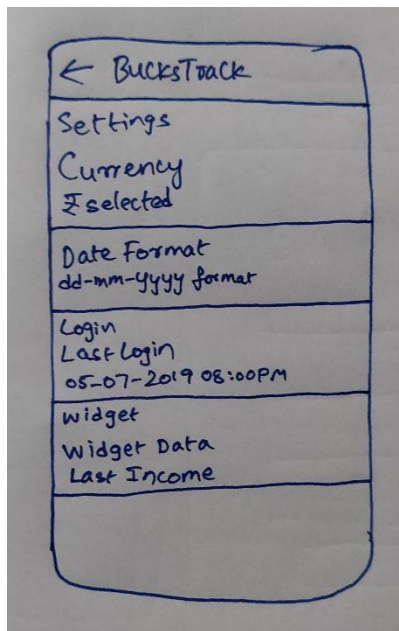
This is the activity where all the list of transactions (income and expense entries) are fetched from the firebase database and are shown to the user sorted by date as a recycler view with card view items containing title on top left, description on top right, amount on bottom left, date on bottom right and category in the middle with different colors(green for income and red for expense) as shown in the mock above. The user can swipe the card to delete the entry. On clicking of particular card, the user can update the data.

Categories Screen



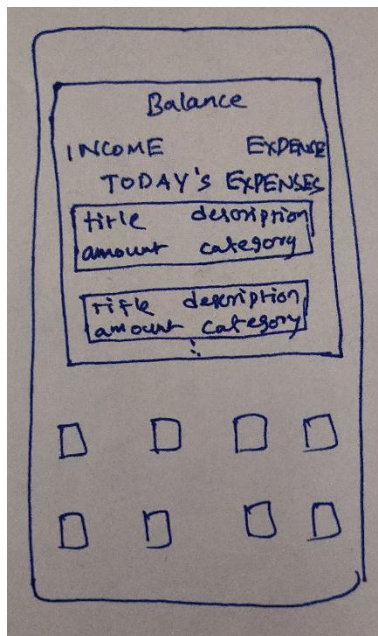
In the categories screen the user is shown a list of two recycler views one with all income categories and other with all expense categories in one scroll view with card layouts. At the list end of each category type, there will be an add button by which on clicking opens up a dialog box asking to enter the income/expense category based on what button clicked, and after entering category and clicking OK, the category will get added to respective category type. The user can swipe them to delete that category.

Settings Screen



In the settings screen the user will be shown his last login date and time for his reference. Also the user can change the currency sybmol from the provided list and can also choose the type of date format as his wish, so that through out the app, the user selected settings will be applied. This is done using a preference screen and fragments. As you can see in the mock image, the user can also select what data to be shown on his widget out of two options(Last Income Transaction/Last Expense Transaction).

Widget Screen



As shown in the above mock image, this is the widget screen where, the balance, income and expenses' total is shown along with user selected last income/last expense transaction in settings screen.

Key Considerations

How will your app handle data persistence?

App handles the data using Firebase Realtime Database. In offline mode, the app uses firebase database cached data that is locally present.

Describe any edge or corner cases in the UX.

Swipe card view items in recycler view 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**”