How to Use this Template

1. Create a new document, and copy and paste the text from this template into your new document [Select All → Copy → Paste into new document]
2. Name your document file: "**Capstone_Stage1**"
3. Replace the text in green

- Description
- Intended User
- Features
- User Interface Mocks
  - Portfolio List Screen (landing screen)
  - New Portfolio Screen
  - Portfolio Details Screen
  - New Stock Holding Screen
  - Income Tracker Widget
- 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: REST call implementation
  - Task 3: Data Persistence
  - Task 4: Create and Hook up the UI
  - Task 5: Integrate Google Services
  - Task 6: Implement the Home Screen Widget

GitHub Username: marcduby (https://github.com/marcduby)

Income Portfolio Tracker

Description

The Income Portfolio Tracker application will be an investment Android application completely written in Java that will help users track the income generated from their stock or ETF investments. Most investment applications track gains in price, but this application will track the income return of a person's investment portfolio in addition to the portfolio's gains. The intended

user is a value investor that is interested in gauging what yearly cash flow their investment portfolio provides as opposed to just unrealized capital gains.

Development of the application::

- Written completely in Java
- Will use test driven development where possible to limit bugs and speed up turnaround
- Will use stable versions of gradle and its dependencies
- Will use a stable version of Android Studio
- Will use string.xml to store all the application display strings
- Will support accessibility by
    - Providing hints for any input field
    - Providing labels for any UI action widget, like button
    - Have content descriptions set for any dynamic elements
    - Use material Design principles for the UI display and color decisions
- Will support RTL switching in all layouts
- Will be using AsyncTask classes for all REST requests and DB updates
- Will use the Room, LiveData and ViewModel classes to help with the DB persistence interface and activity lifecycle changes

Intended User

The intended user of the Income Portfolio Tracker application is the value investor that is interested in gauging what monthly cash flow their investment portfolio provides in dollar and percentage terms as opposed to just unrealized capital gains.

Features

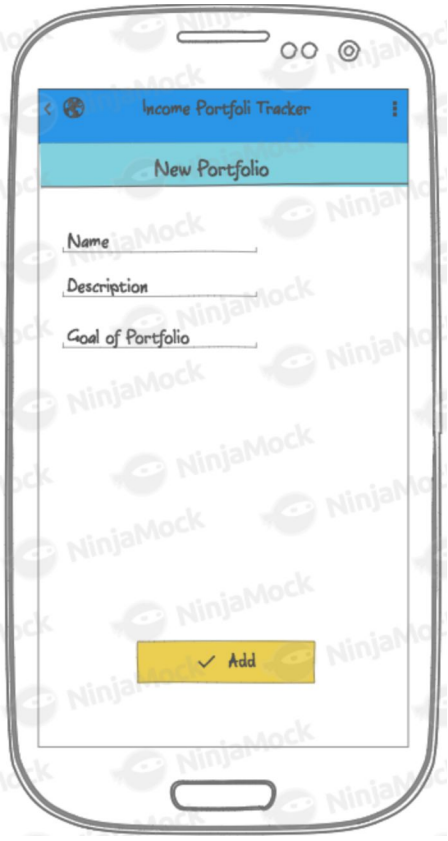
- Lookup any stock listed in the US exchanges by its symbol
- Build one or more portfolios of stock holdings
- Track the value and the yearly yield (percentage and total) of the portfolios based on current value of the investment
- Track the portfolio diversification across industries

User Interface Mocks

Portfolio List Screen (landing screen)

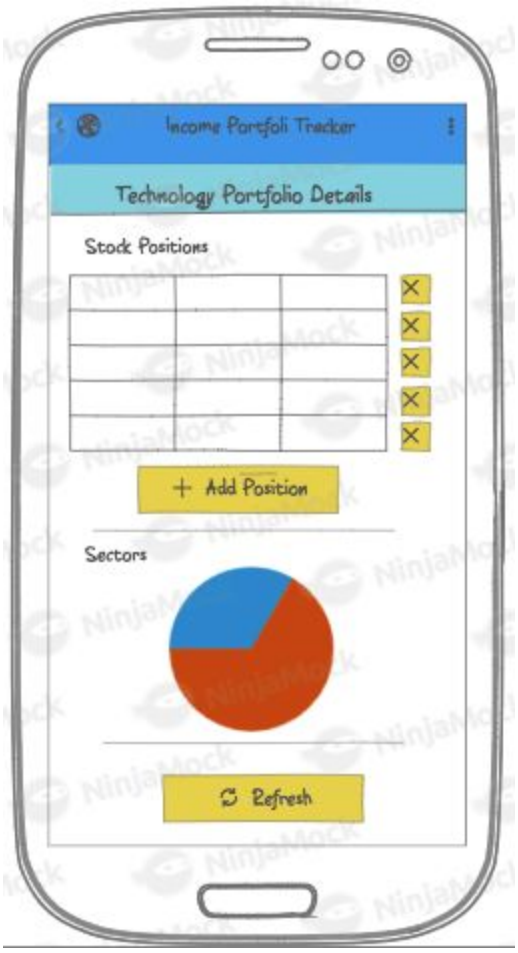
The portfolio list screen is the application landing page; it will list the portfolios saved for the user and will provide an option to create a new one. Clicking on a portfolio in the list will take you to the portfolio details page.

New Portfolio Screen



The new portfolio screen will allow the user to create a new portfolio entity meant to contain multiple stock holdings. The stock holdings will be added in a later screen. Any portfolio added will be persisted in the Android device's SQLlite database.

Portfolio Details Screen

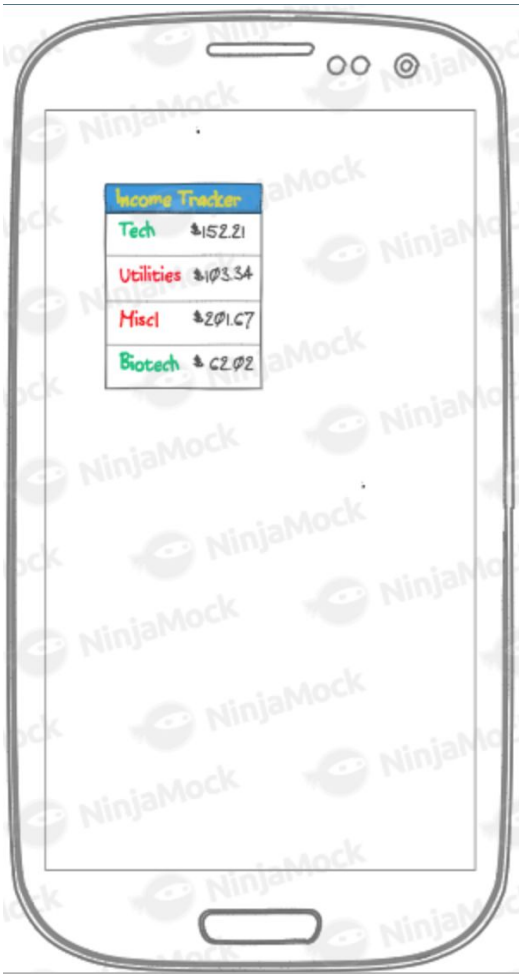


The portfolio details screen is the central screen of the application, as it will provide the user a view of the yearly cash flow generated by the portfolio as well as the sector diversification of the portfolio. A list of the holdings will also offer a quick current value analysis of each stock holding.

New Stock Holding Screen

The new stock holding screen will allow a user to add a stock position to a portfolio. The symbol entry will be validated with the web service to verify its validity, and any market price and company information (name, symbol, industry sector, current price,, dividend) about the stock will be pulled down from the web service and stored in the device's SQLlite DB. The user will fill out the number of shares they purchased and at what cost basis (price).

Income Tracker Widget



The Income Portfolio Tracker application widget will give the user a view of each of the portfolio's expected yearly income in dollar terms, as well as whether the portfolio is up or down from the price the stock holding positions were purchased. Clicking on the widget will bring up the Income Portfolio Tracker application.

Key Considerations

How will your app handle data persistence?

The Income Portfolio Tracker application will handle persistence on the device using Android Architecture Components, specifically the Room library.

Describe any edge or corner cases in the UX.

Potential edge cases to handle:

- Inputs not formatted properly; the application will be based on floating point number entries, so checks before data persistence will have to be done. I'll also explore if keyboard restrictions for those inputs is possible.
- Stock symbols checks will happen as new stock positions are added when current price data is pulled from the server will have to be done to avoid NullPointerExceptions.
- The main data structure will be the portfolio, so should the user delete all their portfolios, the application will have to handle future stock position additions accordingly, by creating a portfolio first.

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

- Room for data persistence; this makes SQLlite database persistence much simpler and error free.
- The MPAndroidChart library will be used to create the charts of the details pages.
- Butterknife for dependency injection of views; this makes for simpler and more readable code.

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

- The Google Firebase Analytics service will be used for tracking of the most used elements of the application. These reports from the analytics data will help inform future features to use and also track performance bottlenecks to address.
- The Google AdMob service will also be used, to show advertisements on the home screen and the portfolio detail screen.
- The Iex Trading (https://iextrading.com/developer/docs/#stocks) REST service will be used to retrieve stock market data; it provides feeds on current market data and dividend payouts needed for the required cash flow analysis of the application. AsyncTask classes will be used to run the web service calls off of the main UI thread.

Next Steps: Required Tasks

The tasks described below will be broken down not by screen functionality but by ground up implementation. The tasks are:

1. Project setup and JSON parsing
    - Create the project and add in all the necessary libraries for the REST calls; implement the JSON parsing using static inputs and unit test
2. REST call implementation
    - Implement the REST calls and integrate with the JSON parsing and all associated integration/unit tests
3. Data Persistence
    - Implement all the data persistence with Room, as well as any integration tests
4. Create and hook up the UI
    - Create the views and integrate with the REST and persistence
5. Google Services
    - Add in the Google Analytics and AdMob services and test
6. Implement the Home Screen widget
    - Design and implement the home screen widget

Task 1: Project Setup and JSON parsing

- Create the application project and set up the github for it
- Implement the JSON parsing and the data objects to use
- Unit test the JSON parsing with static JSON inputs

Task 2: REST call implementation

- Implement the REST calls for the stock information
- Integrate the REST calls with the JSON parsing
- Write integration tests for the REST to Java POJO interface

Task 3: Data Persistence

- Add in the Room library
- Design the persistence classes; ideally reuse the JSON parsing POJOs
- Write integration tests for the adding and deleting of objects

Task 4: Create and Hook up the UI

- Create the activities and layouts in the order detailed above
- Hook up the activities with the REST calls and Room persistence
- Write Espresso tests for integration tests

Task 5: Integrate Google Services

- Hook in the the Firebase Analytics SDK
- Add in Google Ad services

Task 6: Implement the Home Screen Widget

- Design the Home Screen widget UI
- Hook in the portfolio model classes to the screen widget display

---

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**"