Homework 9
Stock Trading App

Prof. Marco Papa

Developed and Designed by
Nikhil Chakravartula (Android) & Aamulya Sehgal (iOS)

Contents

1. Objectives

- Become familiar with Java, JSON, Android Lifecycle and Android Studio for Android app development.
- Build a good-looking Android app.
- Learn the essentials of Google's Material design rules for designing Android apps
- Learn to use the Finnhub APIs and the Android SDK.
- Get familiar with third party libraries like Picasso, Glide and Volley.

The objective is to create an Android application as specified in the document below and in the video.

2. Background

2.1 Android Studio

Android Studio is the official Integrated Development Environment (IDE) for Android application development, based on IntelliJ IDEA - a powerful Java IDE. On top of the capabilities you expect from IntelliJ, Android Studio offers:

- Flexible Gradle - based build system.
- Build variants and multiple apk file generation.
- Code templates to help you build common app features.
- Rich layout editor with support for drag and drop theme editing.
- Lint tools to catch performance, usability, version compatibility, and other problems.
- ProGuard and app-signing capabilities.
- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine.

More information about Android Studio can be found at:
http://developer.android.com/tools/studio/index.html

2.2 Android

Android is a mobile operating system initially developed by Android Inc. a firm purchased by Google in 2005. Android is based upon a modified version of the Linux kernel. As of Nov 2018, Android is the number 1 mobile OS, in unit sales, surpassing iOS, while iOS was still the most profitable platform.

The Official Android home page is located at:
http://www.android.com/
The Official Android Developer home page is located at:
http://developer.android.com/

2.3 Amazon Web Services (AWS)

AWS is Amazon's implementation of cloud computing. Included in AWS is Amazon Elastic Compute Cloud (EC2), which delivers scalable, pay-as-you-go compute capacity in the cloud, and AWS Elastic Beanstalk, an even easier way to quickly deploy and manage applications in the AWS cloud. You simply upload your application, and Elastic Beanstalk automatically handles the deployment details of capacity provisioning, load balancing, auto-scaling, and application health

monitoring. Elastic Beanstalk is built using familiar software stacks such as the Apache HTTP Server, PHP, and Python, Passenger for Ruby, IIS for .NET, and Apache Tomcat for Java.

The Amazon Web Services homepage is available at: http://aws.amazon.com/

2.4 Google App Engine (GAE)

Google App Engine applications are easy to create, easy to maintain, and easy to scale as your traffic and data storage needs change. With App Engine, there are no servers to maintain. You simply upload your application and it's ready to go. App Engine applications automatically scale based on incoming traffic. Load balancing, micro services, authorization, SQL and NoSQL databases, memcache, traffic splitting, logging, search, versioning, roll out and roll backs, and security scanning are all supported natively and are highly customizable.

To learn more about GAE support for Node.js visit this page:

https://cloud.google.com/appengine/docs/nodejs

2.5 Microsoft Azure

The Azure cloud platform is more than 200 products and cloud services designed to help you bring new solutions to life—to solve today's challenges and create the future. Build, run, and manage applications across multiple clouds, on-premises, and at the edge, with the tools and frameworks of your choice.

To learn more about the Azure services, visit this page:

https://azure.microsoft.com/en-us/solutions/

To learn more about Azure support for Node.js visit this page:

https://docs.microsoft.com/en-us/azure/devops/pipelines/targets/webapp?view=azure-devops&tabs=yaml#deploy-a-javascript-nodejs-app

3. Prerequisites

This homework requires the use of the following components:

- Download and install Android Studio. Technically, you may use any IDE other than Android Studio such as Eclipse, but the latest SDKs may not be supported with Eclipse. We will not be providing any help on problems arising due to your choice of alternate IDEs.
- You must use the **emulator, preferably Pixel 3A with API 30**. Everything should just work out of the box.
- If you are new to Android Development, Hints are going to be your best friends!

4. High Level Design

This homework is a mobile app version of Homework 6 and Homework 8.
In this exercise, you will develop an Android application, which allows users to search for different stock symbols/tickers and look at the detailed information about them. Additionally, the users can trade with virtual money and create a portfolio. Users can also favorite stock symbols to track their stock prices. The App contains 2 screens: Home screen and the Detailed Stock Information screen. However, the App has multiple features on each of these screens.

This homework contains 9 API calls. There are the calls to the Finnhub APIs for company profile, stock quotes, symbol autocomplete, stock news, recommendation trends, social sentiments, company peers, company earnings, hourly chart data points and historical chart data points. Each of these 9 API calls are the same as Homework #8. So, you can use the same Node.js backend as Homework #8. In case you need to change something in Node, make sure you do not break your Angular assignment (or deploy a separate copy) as the grading for homework will not be finished at least until 1 week later.

We suggest you to use Java. Kotlin is allowed but will not be supported in piazza.

PS: This app has been designed and implemented in a Pixel 3A emulator by using SDK API 30. It is highly recommended that you use the same virtual device and API to ensure consistency.

Demo will be on an emulator using Zoom, no personal devices allowed, see the rules:

https://csci571.com/courseinfo.html#homeworks

5. Implementation

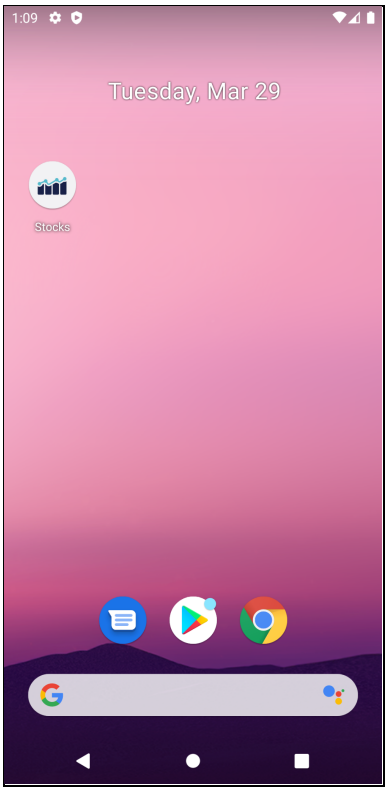

Figure 1.1
App Icon

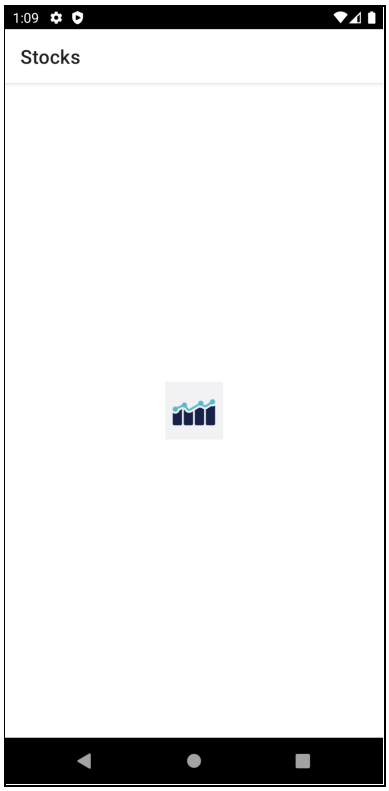

Figure 1.2
Splash Screen

5.1 App Icon and Splash Screen

In order to get the app icon/image, please see the **hints** section.
The app begins with a welcome screen (Figure 1.2) which displays the icon provided in the hint above.

This screen is called Splash Screen and can be implemented using many different methods. The simplest is to create a resource file for the launcher screen and add it as a style to AppTheme.Launcher (see **hints**). Please refer to Figure 1.1 and Figure 1.2.

5.2 Home screen

When you open the app, there will be an initial progress bar while the data is being fetched using volley as shown in Figure 2.1. The home screen will have a toolbar at the top with title “Stocks” and the search icon. Below that, it will show the current date as shown in Figure 2.2.
There are 2 Sections on the home screen:

- ***Portfolio Section*** -
  This section will show:
  - Cash Balance = the money in the wallet currently for the user, which is uninvested cash
  - Net Worth = Cash Balance + total value of all the stocks owned

  This is followed by the list of stocks in the user portfolio with that stock’s:
  - Stock Symbol
  - Market Value = Latest Stock Quote * Number of Shares Owned
  - Change in Price from Total Cost = (Latest Stock Quote - Avg. Purchase Price of the Stock) * Number of Shares Owned
  - Change In Price From Total Cost Percentage = (Change in Price from Total Cost / Total Cost of Stock) * 100, where Total Cost of Stock = (Avg. Purchase Price of the Stock * Number of Shares Owned)
  - Total Shares Owned

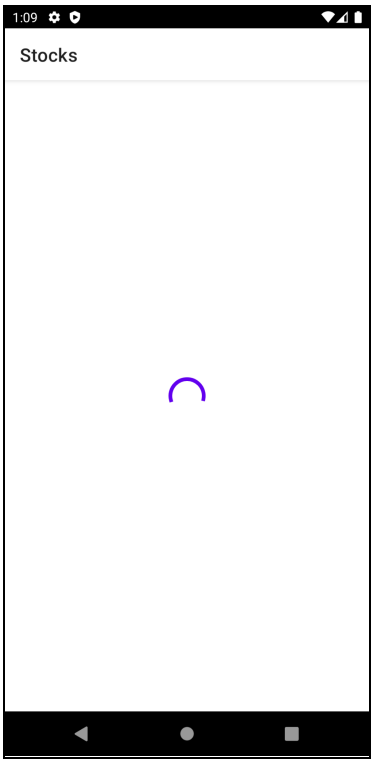

Figure 2.1
Progress Bar

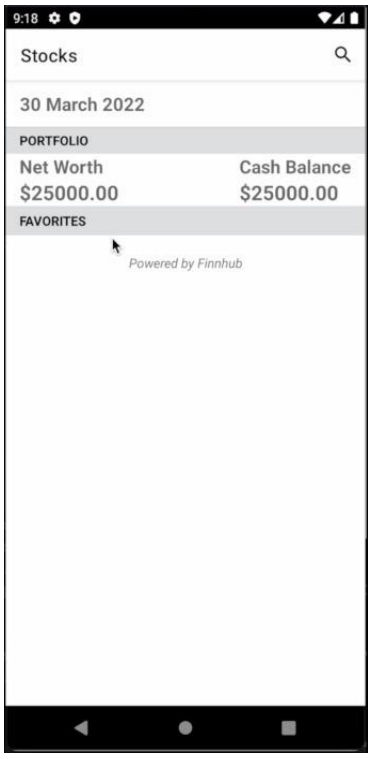

Figure 2.2
Initial Home Screen

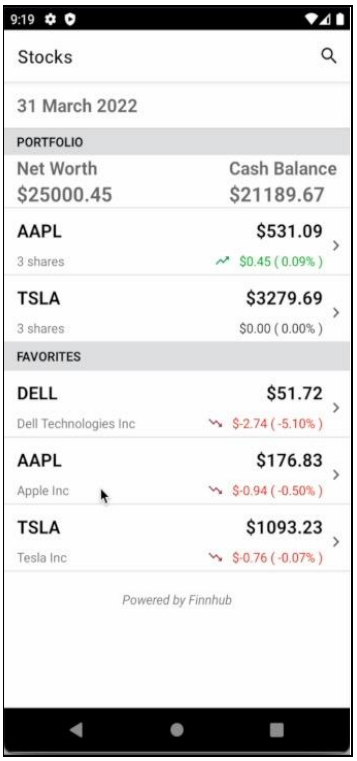

Figure 2.3
Home Screen

<u>If the user starts the app for the first time, they will not have any stocks/shares in the portfolio and an initial pre-loaded amount of $25,000 to trade on the app.</u>

- ***Favorites Section*** - This section will show all the stocks that have been marked as "favorite" by the user. This is the same as the watchlist feature of Homework #8. For each favorited stock show:
    - o Stock Symbol
    - o Current price - Latest Stock Quote
    - o Change In Price (Since Last Close)
    - o Change In Price Percentage (Since Last Close)

Additionally, the symbol next to the change in price value should either be trending down, or up based on the change in price value. If the change in price is positive (>0), a trending up symbol should be displayed and the symbol as well as the change values should have green color. If the change price is negative (<0), a trending down symbol should be displayed and the symbol as well as the change values should have red color. If there's no change in price, then there shouldn't be a symbol and the change values should have black color. Please refer to the Hints section for these icons

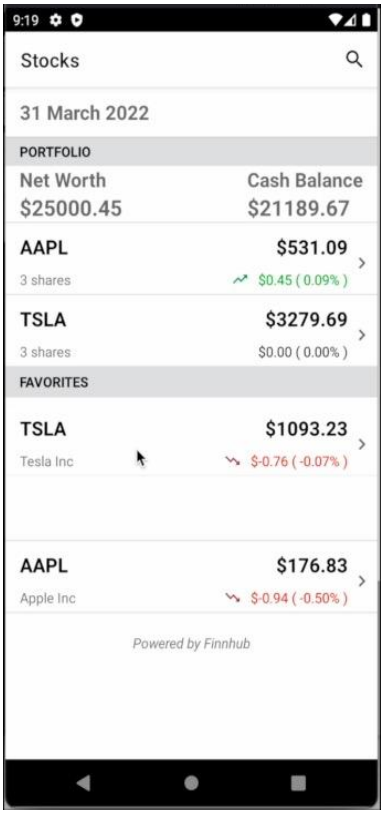

Figure 2.4
Drag Drop

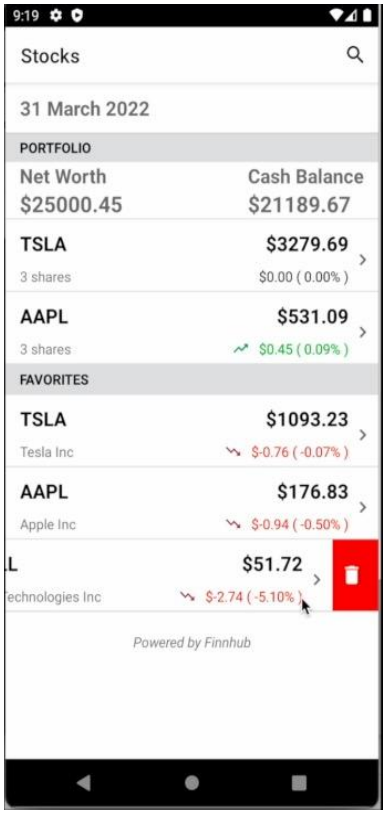

Figure 2.5
Delete Favorite

Each stock listing also has a button on the extreme right, next to the current price field (right arrow (or) a chevron-right button). On clicking the button or the stock listing, the detailed information screen will open for the selected stock. Please refer to the Hints section for the icon.

The home screen view should support two functionalities:

- The **swipe to delete functionality** allows the user to remove/delete the stock from the **favorite section**. On removing a stock from the favorite section, the stock should be removed from the favorite stocks in the local storage and the view. For the delete icon, please refer to the Hints section.
- The **drag and reorder functionality** allows the user to reorder the stocks in **either section**. The user should be able to long press the stock listing and drag it to the new position. The list should be updated accordingly to ensure the new order going forward. The user cannot drag the stock from the favorite section into the portfolio section. The stock can only be dragged and dropped in the same section. If an attempt is made however, then the app should handle it gracefully and not crash. If an item is dragged from the favorite section to the portfolio section, reorder the stock to the top of the favorite section. Similarly, if a stock is dragged from the portfolio section to the favorite section, reorder the stock to the bottom of the portfolio section.

At the bottom of the 2 sections, we have a 'Powered by Finnhub' text in italic. On clicking this text, the App should open the Finnhub homepage in chrome. (Finnhub URL: https://www.finnhub.io).

Note: The price information for each stock should be updated every 15 seconds, same as Homework #8. Additionally, the progress bar should only be shown whenever the home screen is opened initially. While refreshing the stock price every 15 seconds, the spinner SHOULD NOT be displayed. The "Cash Balance" and "Net Worth" values should also be updated based on the latest stock quote. For example, if a user purchased a single stock at $500, and the latest quote price is $501, then Cash Balance = $24500 and Net Worth = $25001.

The home screen has been implemented by using a **RecyclerView** with the **SectionedRecyclerViewAdapter.** Each of the stock listings has been implemented using **ConstraintLayout, TextView, ImageView.**

The **Search** button on the toolbar opens the search bar to type the stock symbol to search. The search bar uses the **autocomplete** functionality

5.3 Search Functionality

- On the top right side, there will be a search button which opens a textbox where the user can enter a keyword to search for a stock symbol. See hints for icon.
- The user is provided with suggestions of keywords using the Finnhub Autocomplete API. (Same as Homework #8)
- When the user taps on a suggestion, it is filled inside the search box and clicking enter/next takes the user to the detailed information screen.
- Before you get the data from your backend server, a progress bar should display on the screen as indicated in the detailed information section.
- The user can only search for valid stock symbols in the search bar. The search should redirect to the detailed information screen only if the user selected one of the autocomplete suggestions to fill the search field.
- Please refer to Figure 3.1.

Implementing search functionality requires 2 things:

- Implementing a searchable – see hints
- Implementing autocomplete – see hints

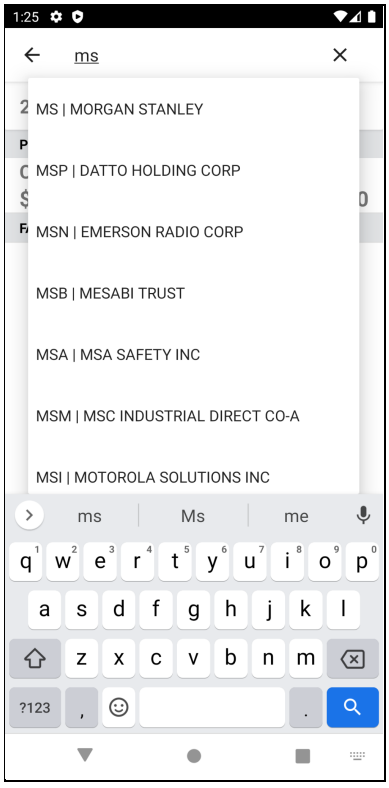

Figure 3.1
Autocomplete

5.4 Detailed Stock Information Screen

On clicking the **chevron-right** button or any stock listing or **searching** for a stock symbol, a progress bar should come up while the details are being fetched. Once the data has been fetched except the chart (since the chart takes longer to load), the progress bar should disappear and information regarding the stock should be available to the user (Figures 4.*).
The top action bar must show the stock ticker/symbol and the back button to go back to the home screen (which has the autosuggest result list that was used for the current search if the user enters the stock detail screen by using the search functionality). The action bar should also contain a favorite icon to add or remove the stock from favorites. The favorite icon will either be filled or bordered depending on whether the stock is marked as favorite or not. Adding/Removing the stock from favorites should also display a toast message as shown in the video. See hints for icons.
Below the action bar, there should be 5 fields: company name, company logo, current price with '$' sign, the change price with '$' sign (the text color should be green or red based on the change price value being positive or negative respectively) and the change percentage with a % sign.
Below these five fields, the hourly and historical charts should be displayed. These charts are similar to Homework #8. The charts should follow a swipeable layout with two tabs.

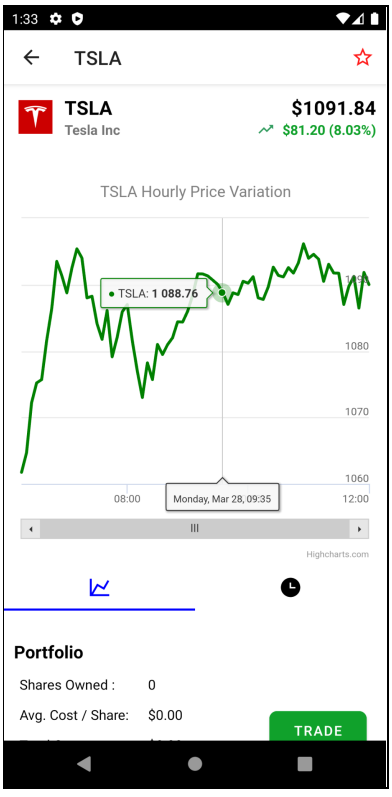

Figure 4.1
Hourly Chart

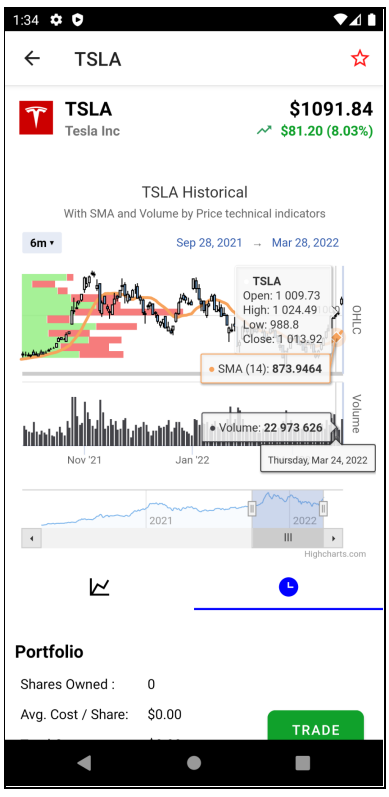

Figure 4.2
Historical Chart

5.4.1 Portfolio Section

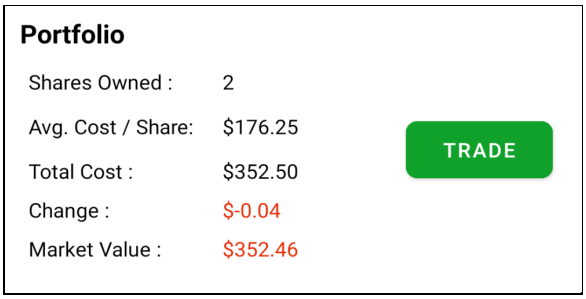

Figure 4.3
Portfolio Detail Section

The **Portfolio section** allows the user to trade the shares of the stock. It contains a left section which shows the number of stocks the user owns, the avg cost per share, total cost of shares owned, change in price of the last quote price to current avg cost and the market value (2 decimals, same for all money related fields) of the stock in the user portfolio.. If the current stock price is greater than the average purchase price of the stock, the Change Value field and the Market Value field should be turned green. On the contrary, if the current stock price is less than the average purchase price of the stock, the values should be turned red. If no change, then values must be in black. The right section contains a customized green trade button.

5.4.2 Stats Section

The **Stats section** shows various stats of the given stock symbol. These are:

- High Price
- Low Price
- Open Price
- Prev. Close

All of these are obtained the same as Homework #8

5.4.3 About Section

The **About section** shows various details about the given stock symbol. These are:

- IPO Start Date
- Industry
- Webpage (link should be clickable and open in Google Chrome)
- Company Peers (as a horizontal scrollable list and should allow clicking a peer symbol to open its detail stock information screen)

All of these are obtained the same as Homework #8.

5.4.4 Insights Section

The **Insights section** has following three components:

- Social Sentiments Table
- Recommendation Trends Chart
- Historical EPS Surprises Chart

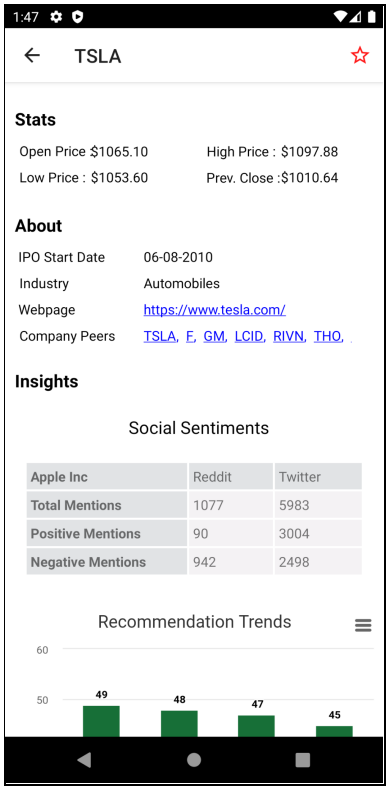

Figure 4.4
Social Sentiments

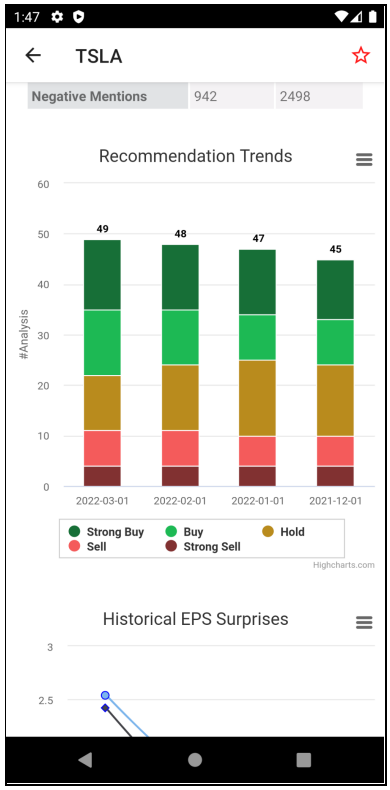

Figure 4.5
Recommendations

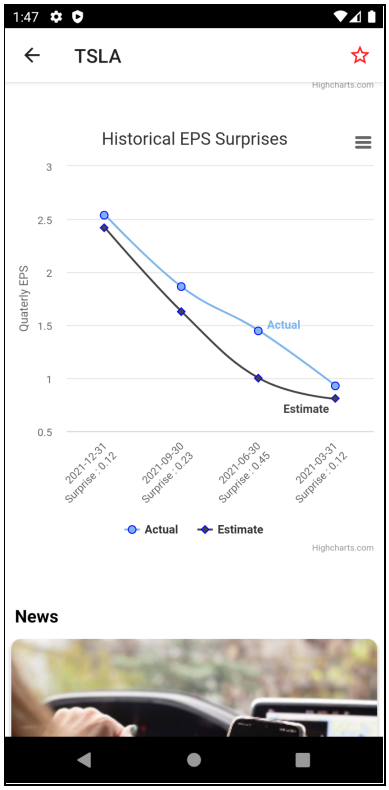

Figure 4.6
EPS

5.4.4.1 Social Sentiments Table

The **Social Sentiments Table** shows information organized in three rows:

- First Row
  - Stock Name Header
  - Reddit Label
  - Twitter Label
- Second Row
  - Total Mentions label
  - Reddit total mentions value
  - Twitter total mentions value
- Third Row
  - Positive Mentions label

    - Reddit positive mentions value
    - Twitter Positive Mentions value
- Fourth Row
    - Negative Mentions label
    - Reddit Negative Mentions value
    - Twitter Negative Mentions value

All of these are obtained in the same way as in Homework #8.

5.4.4.2 Recommendation Trends Chart

The **Recommendation Trends Chart** shows recommendation trends for the stock symbol. You are free to use the Angular HighCharts package or a WebView but must ensure that the functionality is similar to what is given in the video.
All of these are obtained in the same way as in Homework #8.

5.4.4.3 Historical EPS Surprises Chart

The **Historical EPS Surprises Chart** shows company earnings data for the stock symbol. You are free to use the Angular HighCharts package or a WebView but must ensure that the functionality is similar to what is given in the video.

All of these are obtained in the same way as in Homework #8.

5.4.5 News Section

The **News section** displays the news articles related to the given stock symbol. The first article (Top News) has a different format/layout than the rest of the articles in the list.

Each article should contain an article image, the article source, the elapsed time (in seconds, minutes or hours) since the article is published and the article title.

- You can load web images with Picasso. (see hints)
- Images should keep the aspect ratio, be cropped, and have a rounded corner.
- The news section uses **RecyclerView**.

Clicking on a news article opens up a dialog with

- Article Source
- Article Published Date
- Article Title
- Article Description (You may limit description to 10 lines)
- Chrome button (clicking it should open the article in Chrome as shown in Figure 4.9).
- Twitter Button (clicking it should open Twitter in Chrome as shown in Figure 4.10)
- Facebook Button (clicking it should open Facebook in Chrome as shown in Figure 4.11)

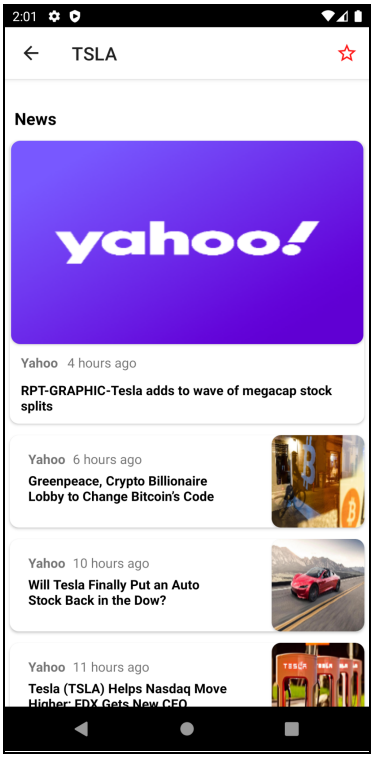

Figure 4.7
News

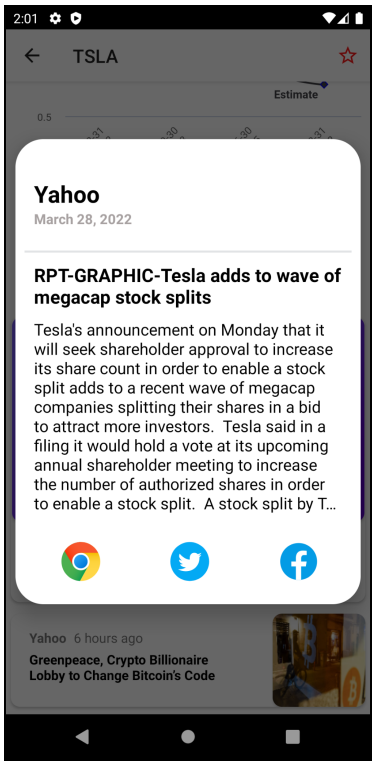

Figure 4.8
News Click

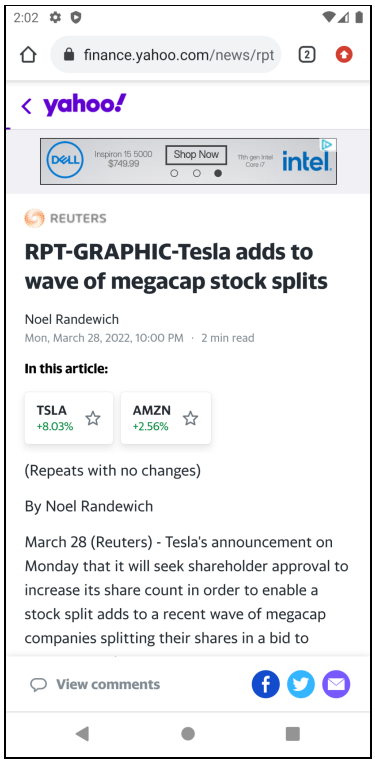

Figure 4.9
Chrome View

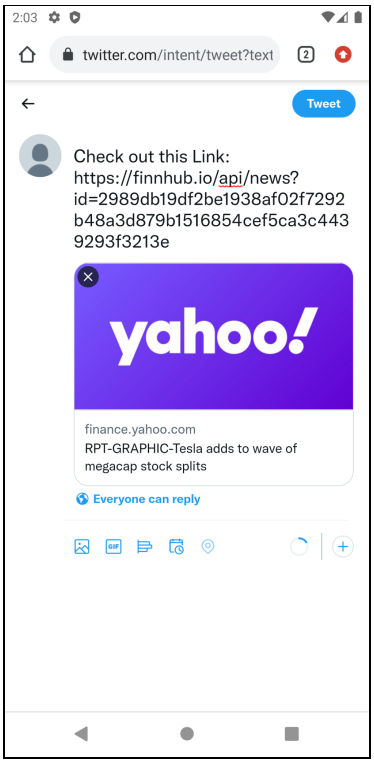

Figure 4.10
Twitter

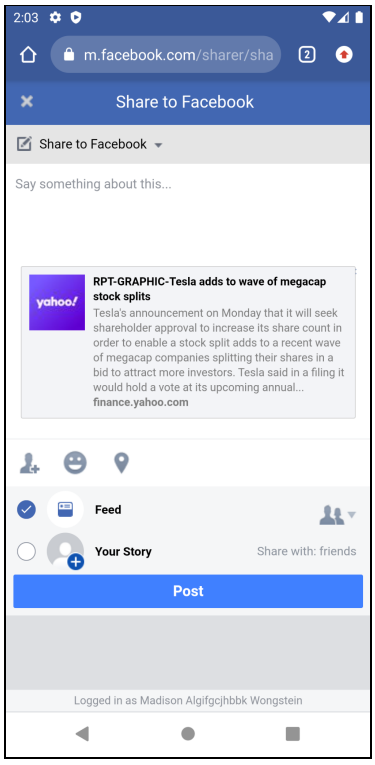

Figure 4.11
Facebook

5.4.6 Trade Dialog

The Trade button in the **Portfolio section** opens a new dialog box for trading. The dialog shows an input box which only accepts numeric input. Below the input field, there is a calculation text box which updates based on the numeric input to display the final price of the trade. The trade dialog also displays the current available amount to trade for the user. The user can either buy or sell the shares. Based on the trade, the amount available to trade will be updated accordingly. There are 5 error conditions to be checked before executing the trade and displaying the trade successful dialog (Figures 4.13 and 4.14). The error conditions are:

- ***User tries to sell more shares than they own*** - The trade dialog box should remain open and a toast message with text 'Not enough shares to sell' should be displayed.
- ***User tries to buy more shares than money available*** **-** The trade dialog box should remain open and a toast message with text 'Not enough money to buy' should be displayed.
- ***User tries to sell zero or negative shares*** **-** The trade dialog box should remain open and a toast message with text 'Cannot sell non-positive shares' should be displayed.
- ***User tries to buy zero or negative shares*** **-** The trade dialog box should remain open and a toast message with text 'Cannot buy non-positive shares' should be displayed.
- ***User enters invalid input like text or punctuations*** **-** The trade dialog box should remain open and a toast message with text 'Please enter a valid amount' should be displayed.

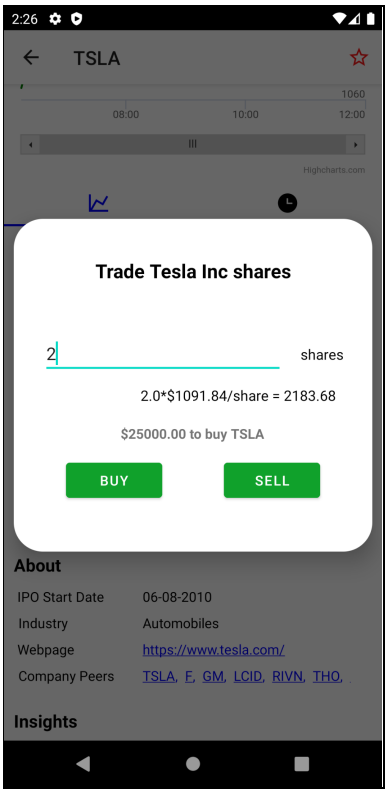

Figure 4.12
Trade Dialog

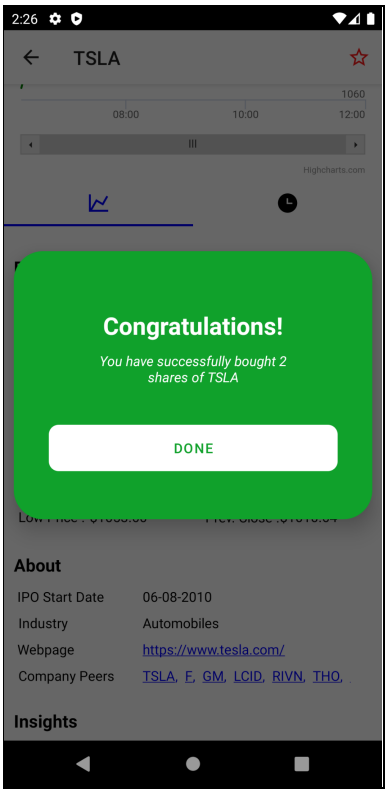

Figure 4.13
Successful Buy

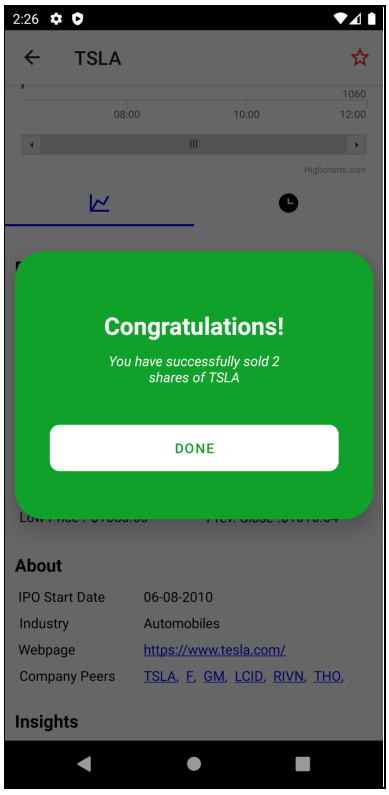

Figure 4.14
Successful sell

After the user clicks on the buy/sell button, a dialog should show the trade success message as shown in Figure 4.11 and Figure 4.12

5.4.7 HighCharts in Android

Recommendation Trends and Historical EPS Surprises charts are implemented using HighCharts Android package. You are free to implement these in a WebView as well, as long as their functionality is similar to what is shown in the video.

The Hourly and Historical charts in the detailed stock information screen uses a *WebView* element to load the HighCharts stock chart. It provides the same features as Homework #8 and the JavaScript code for loading the chart can be reused. To load the chart, the App will load a local HTML file with the necessary JavaScript to request the data from the NodeJS server and display the chart when the data is fetched (See the hints). This choice is made as the HighCharts android package doesn't come with "StockChart" elements.

5.5 Local Storage

You should save both the list of favorite stocks, and your portfolio to local storage. When the app is opened, you should load them, then fetch data from your backend.

We will be using Shared Preferences for storing the app data for this assignment. See hints for more details on the implementation of the Shared Preferences.

5.6 Progress bar

Every time the user has to wait before they can see the data, you must display a progress bar as shown in Figure 4. The progress bar is to be present across the **Home screen and Detailed Stock screen**. One idea would be to display the progress bar by default (where needed) and then hide it as soon as the data is received/view is prepared. See hints for progress bar related ideas and styling.

Note: Based on your implementation, you might need to put progress bars in different places. During the demo, the only place that is allowed to be blank is the WebView that loads HightCharts.

5.7 Summary of detailing and error handling

1. Make sure there are no conditions under which the app crashes
2. Make sure all icons and texts are correctly positioned as in the video/screenshots
3. Make sure the screens and toasts are correctly displayed.
4. Make sure there is a progress bar on initial loading of the home screen, and while loading the stock detail page.
5. Make sure the styling for different features matches the video/screenshots.
6. All API calls are to be made using Node.js backend.

5.8 Additional Info

For things not specified in the document, grading guidelines, the video, or in Piazza, you can make your own decisions. But keep in mind about the following points:

- Always display a proper message and do not crash if an error happens.
- All HTTP requests should be asynchronous and should not block the main UI thread. You can use third party libraries like Volley to achieve this in a simple manner.

6. Implementation Hints

6.1 Icons

The images used in this homework are available in the zip file mentioned in the Piazza post for Homework #9.
Furthermore, please refer to the following websites and search for additional icons.

1. https://materialdesignicons.com
    a.

Icon Description	**Icon name to search for**
Trending Up icon	Trending-up
Trending Down icon	Trending-down
Right arrow button inside Portfolio and Favorites items	Chevron-right
Delete icon during a favorite swipe delete	Delete
Search Icon	Magnify
Favorite Icons	Star, Star-outline
Historical Chart Icon	Clock-time-three
Hourly Chart icon	Chart-line

2. https://icons8.com/icons/set/twitter
    a.

Icon Description	**Icon name to search for**
Facebook icon	Facebook
Twitter icon	Twitter
Chrome icon	Chrome

You can choose to work with xml/png/svg/jpg versions. We recommend using xml as it is easy to modify colors by setting the Fill Colors.

6.2 Third party libraries

Sometimes using 3rd party libraries can make your implementation much easier and quicker. Some libraries you may have to use are:

6.2.1 Volley HTTP requests

Volley can be helpful with asynchronous http request to load data. You can also use Volley network ImageView to load photos in Google tab. You can learn more about them here:
https://developer.android.com/training/volley/index.html

6.2.2 Picasso

Picasso is a powerful image downloading and caching library for Android.
http://square.github.io/picasso/

6.2.3 Glide

Glide is also a powerful image downloading and caching library for Android. It is similar to Picasso. You can also use Glide to load images..
https://bumptech.github.io/glide/

6.3 Working with action bars and menus

https://developer.android.com/training/appbar/setting-up
https://stackoverflow.com/questions/38195522/what-is-oncreateoptionsmenumenu-menu

6.4 Displaying Progress Bars

https://stackoverflow.com/a/28561589
https://stackoverflow.com/questions/5337613/how-to-change-color-in-circular-progress-bar

6.5 Implementing Splash Screen

There are many ways to implement a splash screen. This blog highlights almost all of them with examples:
https://android.jlelse.eu/the-complete-android-splash-screen-guide-c7db82bce565

6.6 Adding the App Icon

https://dev.to/sfarias051/how-to-create-adaptive-icons-for-android-using-android-studio-459h

6.7 Adding ellipsis to long strings

https://stackoverflow.com/questions/6393487/how-can-i-show-ellipses-on-my-textview-if-it-is-greater-than-the-1-line

6.8 Adding a button to ActionBar

https://developer.android.com/training/appbar/actions
https://stackoverflow.com/questions/12070744/add-back-button-to-action-bar
https://stackoverflow.com/questions/34110565/how-to-add-back-button-on-actionbar-in-android-studio

6.9 Implementing a Recycler view in android

https://developer.android.com/guide/topics/ui/layout/recyclerview
https://stackoverflow.com/a/40217754
https://abhiandroid.com/materialdesign/recyclerview-gridview.html

6.10 Adding Toasts

https://stackoverflow.com/questions/3500197/how-to-display-toast-in-android
https://developer.android.com/guide/topics/ui/notifiers/toasts

6.11 Passing variables to intent

https://stackoverflow.com/questions/2405120/how-to-start-an-intent-by-passing-some-parameters-to-it

6.12 Formatting Text using HTML in TextView

https://stackoverflow.com/a/27462961

6.13 Open Link in browser

https://www.tutorialkart.com/kotlin-android/android-open-url-in-browser-activity/

6.14 Back press behavior on Back button

https://stackoverflow.com/a/27807976

6.15 SearchBar and AutoCompleteTextView

To implement the search functionality, these pages will help:
https://www.youtube.com/watch?v=9OWmnYPX1uc
https://developer.android.com/guide/topics/search/search-dialog

Working with the AutoCompleteTextView to show the suggestions might be a little challenging. This tutorial goes over how it is done to help implement it.
https://www.journaldev.com/9574/android-autocompletetextview-example-tutorial
https://developer.android.com/reference/android/widget/AutoCompleteTextView
In order to link your Search Bar with autocomplete suggestions, these links might help:
https://www.dev2qa.com/android-actionbar-searchview-autocomplete-example/

6.16 To implement favorites and portfolio sections

https://developer.android.com/reference/android/content/SharedPreferences
https://www.journaldev.com/9412/android-shared-preferences-example-tutorial

6.17 Implementing Dialogs

https://mkyong.com/android/android-custom-dialog-example/
https://androidexample.com/Custom_Dialog_-_Android_Example/index.php?view=article_description&aid=88&aaid=111
https://medium.com/@suragch/creating-a-custom-alertdialog-bae919d2e

6.18 Sectioned RecyclerView Adapter

https://github.com/luizgrp/SectionedRecyclerViewAdapter/tree/master/app/src/main/java/io/github/luizgrp/sectionedrecyclerviewadapter/demo/example1
https://github.com/luizgrp/SectionedRecyclerViewAdapter/blob/master/app/src/main/res/layout/section_ex1_header.xml
https://github.com/luizgrp/SectionedRecyclerViewAdapter/blob/master/app/src/main/res/layout/section_ex1_item.xml
https://github.com/luizgrp/SectionedRecyclerViewAdapter/blob/master/app/src/main/res/layout/fragment_ex1.xml

6.19 Rounded buttons/Images

https://stackoverflow.com/a/13872391

6.20 GridView

https://abhiandroid.com/ui/gridview#:~:text=In%20android%20GridView%20is%20a,item%20by%20clicking%20on%20it.

6.21 Using Recyclerview inside ScrollView

https://stackoverflow.com/questions/27083091/recyclerview-inside-scrollview-is-not-working

6.22 Swiping to delete feature in RecyclerView

https://www.journaldev.com/23164/android-recyclerview-swipe-to-delete-undo#code

https://github.com/xabaras/RecyclerViewSwipeDecorator

6.23 Drag and Reorder feature in RecyclerView

https://www.journaldev.com/23208/android-recyclerview-drag-and-drop

6.24 Create Swipeable tabs with ViewPager2 and TabLayout

https://medium.com/busoft/how-to-use-viewpager2-with-tablayout-in-android-eaf5b810ef7c

https://developer.android.com/guide/navigation/navigation-swipe-view-2

7. Files to Submit

You should also ZIP all your source code (without image files and third-party modules) and submit the resulting ZIP file by the end of the demo day.
You will have to submit a video of your assignment demo. Details for that will be sent separately.

****IMPORTANT****

All videos are part of the homework description. All discussions and explanations on Piazza related to this homework are part of the homework description and will be accounted into grading. So please review all Piazza threads before finishing the assignment.