AppleStocks

A Simple Application for Apple Stock Quotes Monitoring

Henrique Romão
up202108067@up.pt

Mariana Bessa
up202107946@up.pt

November 29, 2024

Contents

1 Introduction

In this small project we develop a simple and effective mobile device application to display closing stock prices of Apple Inc. The information is displayed according to two factors choosen by the user: time period type — hours or days —; and the number of time periods to showcase — from 2 to 10. The application is divided into two activities, one responsible for getting the user choices, and another responsible for processing the data retrieved from the internet and plotting the processed data. The design is simple, ensuring the application is ease to use, in hopes of increasing acessability and, therefore, the potential number of users.

2 Use Cases

The application supports the following use cases, which are represented in the use case diagram, in Figure 1.

Choose View Options Where the user selects a time interval, Hourly or Daily, and a range, 2–10.

Get Stock Information Where the application connects with the Web API.

See Stocks When the user pushes the "Submit" button and sees the Plot Activity.

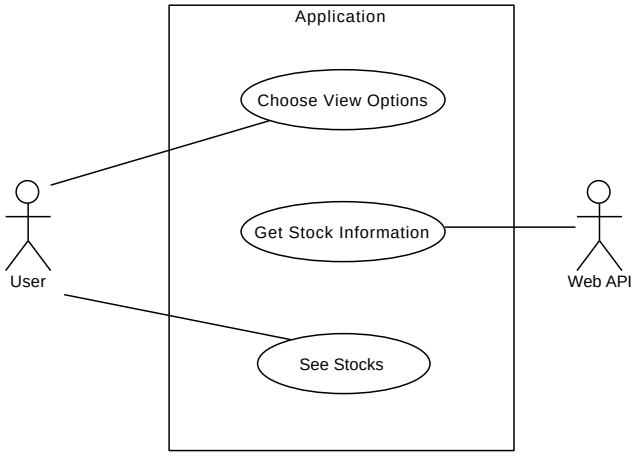


Figure 1: Use Cases Diagram.

3 Architecture

To make the application more user-friendly and the interface less crowded, the application was divided into two activities: the Main Activity, where the user configures their preferences, and the Plot Activity, where the results are displayed. Common to both activities, there is an action bar displaying the name of the app. An overview of the app architecture can be seen in Figure 2.

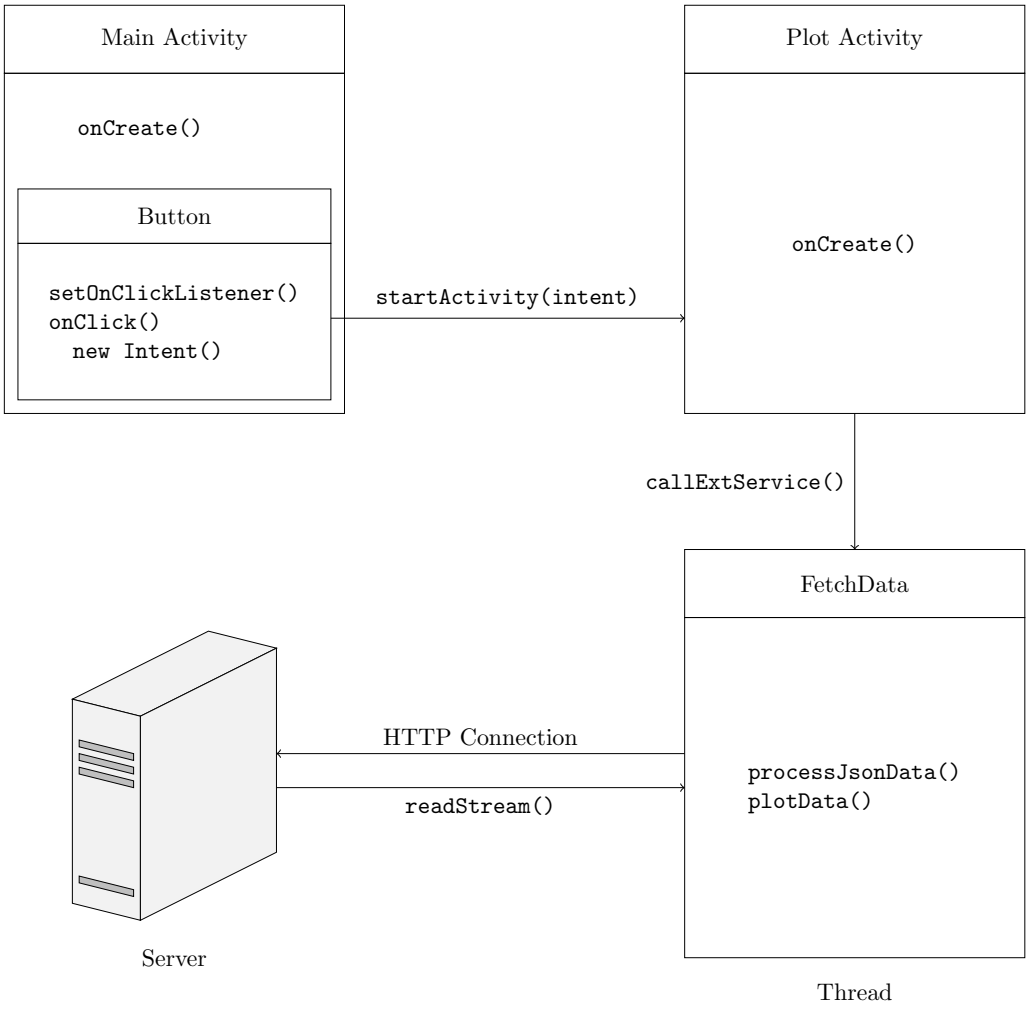


Figure 2: Application Architecture Diagram.

3.1 Main Activity

This activity also includes two `Spinner` objects: one for selecting the type of interval (hours or days) and another for specifying the interval range. These spinners are initialized in the Main Activity code using the `new ArrayAdapter<>()` method, with `setDropDownViewResource()` used to define their layout.

Both spinners have default values: "H" (hours) for the type of interval and "2" for the range. This means that, unless changed by the user, the stock quotes for the last two hours will be displayed by default.

A button is provided to allow the user to proceed, with its listener being implemented within the Main Activity. The `onClick()` method creates an `Intent` to open the Plot Activity, passing the user's spinner selections (as a string and an integer) to the new activity.

3.2 Plot Activity

The plot activity is where, according to the choices made on the Main Activity, the HTTP request is made, retrieving the stock information from the web API, and it is also in this activity that the retrieved information is processed and displayed to the user.

3.2.1 HTTP Request

The first step to retrieve the information from the web API is to read the intent passed from the previous activity, with the time interval type and number. With these two variables, it is possible to craft the URL, which will be used to do the HTTP request. This URL is passed through a function that initiates a new thread, where the rest of the processing will take place. Inside the thread, the processing will be initiated by `FetchData`, a private class that takes a URL and establishes the HTTP connection. The connection status is then assessed, and if it is succesfull, the processing will go on; otherwise, an error message is thrown.

3.2.2 Data Processing

Before the actual processing begins, the input stream received from the HTTP connection is converted into type `String`. With this `String`, function `processJsonData` will create a JSON object to retrieve the closing quote prices and respective time. Redarding the time, it is necessary to convert the dates into float type objects, for the plot function, and so it is necessary to format the dates.

A problem that arises is that, according to the choice of days or hours, the date format will be different, as, in the las case, it will include the hour and timezone. According to the selected time interval type passed through the intent, a different format will be used for parsing the date strings into date objects, which are then converted to floats with the timestamps. The processed information is then passed to the `plotData` function.

3.2.3 Data Plotting

The data is plotted using the LineData widget, part of the MPandroid Chart library, [1]. A widget was used to take advantage of already developed code and shorten the app development period. This particular widget was choosen due to its widespread use, seeming a correct fit for the task in hand.

In this widget the data is passed in an array of entries, and so the function joins the two arrays into a single array of entries with two values, one entry for each dot in the plot. Additionally, the function computes the maximum and minimum closing quote prices, and creates an entry array for each, to be plotted as horizontal lines delimiting those values. Then, each entry array is properly formatted, and sent to the widget to be plotted. It is also necessary to format the x-axis, otherwise the values seen would be the float type objects, which are difficult to interpret and not user-friendly. To this intent, the value is formatted to return a simple date string, and only the first and last values are shown, for a simpler graph analysis. The maximum and minimum values are also shown outside the plot.

4 Interface

The application has a black background with purple accents, as darker colors are more associated with the stock tracking and the stock market. The theme used was `Theme.AppCompat.Light.DarkActionBar`.

The interface is structured using `LinearLayout` objects to arrange the `TextView` elements, `Spinner` objects, and `Submit` button in vertical and horizontal sequences. When the `Submit` button is pressed, the application starts the `Plot Activity` using the `startActivity()` method, passing the selected user inputs as extras, in the intent.

4.1 Main Activity

This activity is design to adapt to both horizontal (landscape) and vertical (portrait) screen orientations, maintaining a consistent and user-friendly layout.

4.1.1 Portrait

Upon opening the application, the user is welcomed by an action bar and two `TextView` elements, where one displays a greeting, and the other provides instructions on how to use the application, as it is shown in Figure 3a. Below these, additional `TextView` elements explain the purpose of each `Spinner` object, which are used for input selection.

The layout for the spinner items and their drop-down lists was custom-designed in the XML files `/layout/spinner_dropdown_item.xml`, which defines the appearance of the spinner drop-down items, and `/layout/spinner_item.xml`, which defines the appearance of the selected spinner items (Figure 3b and Figure 3c). It should be noticed that the spinner items are scrollable. The spinner values are defined in the `strings.xml` file located in `/res/values/`.

Additionally, custom shapes were created to style the `Submit` button and the background of the spinner section. These shapes were implemented in the `/drawable/submit_button.xml` file, which defines the shape of the Submit button, and the `/drawable/round_button.xml` file, which shapes the background of the selection area.

4.1.2 Landscape

In the landscape orientation, a dedicated layout file, `/layout-land`, was created to optimize the user interface for horizontal screens.

This layout maintains the key aspects of the portrait view, with the main adjustment being the rearrangement of the `Spinner` objects: instead of being vertically aligned, they are displayed side by side (horizontally). This design enhances usability by taking full advantage of the horizontal screen space.

The layout is shown in Figure 4a, including the spinner options for interval and range choices (Figure 4b and Figure 4c).

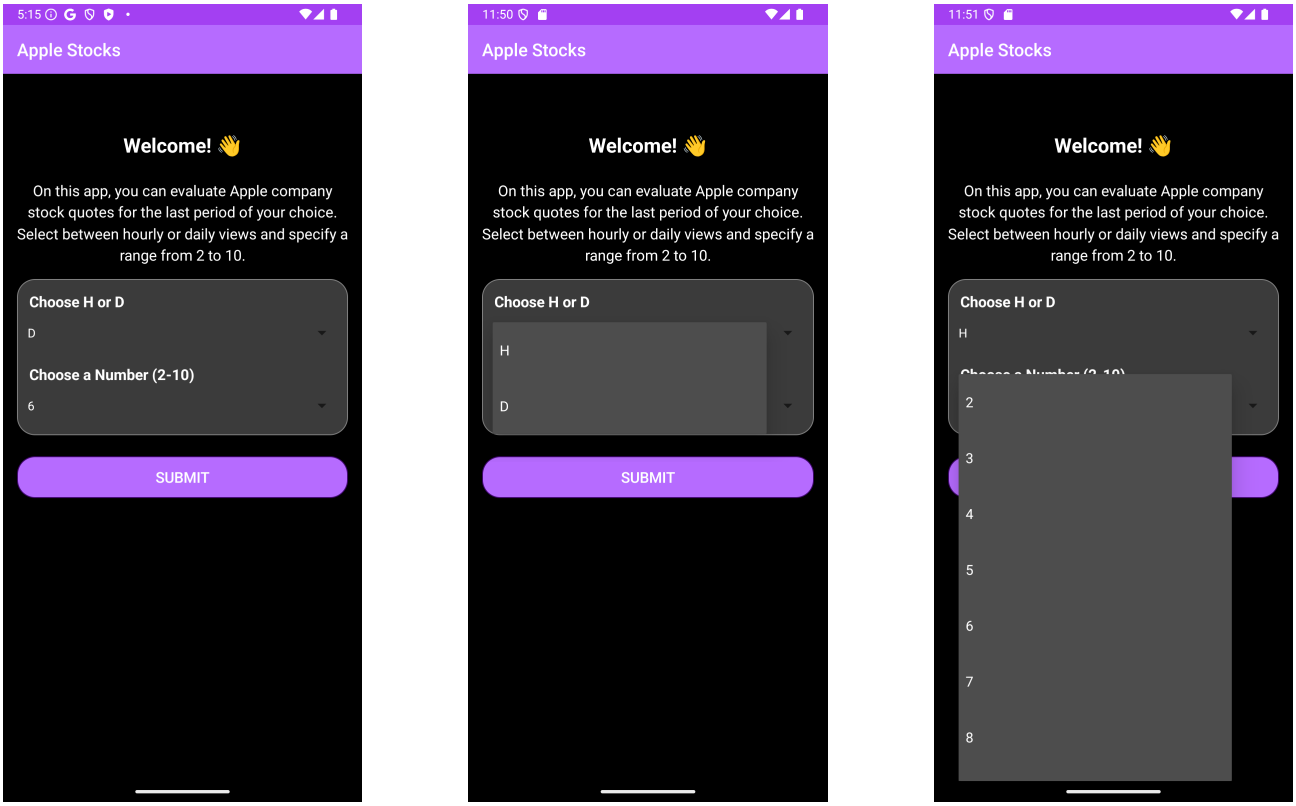
4.2 Plot Activity

The interface for the plot activity can be seen in Figure 5, with the activity still loading the data (Figure 5a) and with all the data loaded (Figure 5b). The activity takes some time loading, but this wait is due to slow connectivity of the web API, having tested the speed for the same request in our personal computers and noticing a slight delay.

To have an indication of the chosen time intervals, the number of time intervals and time interval type are written in the plot title.

4.2.1 Plot

In Figure 5b it is possible to see how the data is plotted, in a simple and clean manner, using a color with a good contrast. The plot is filled with a gradient to ease data visualization, when compared to having no fill, while not



(a) Main Activity. (b) Interval Choice. (c) Range Choice.

Figure 3: Main Activity.

being too tiressome to the eyes, as a solid colorful fill would be. As mentioned, to avoid an excess of information, only the first and last x-axis labels are written, with the title already providing enough information for the correct understanding of the graph.

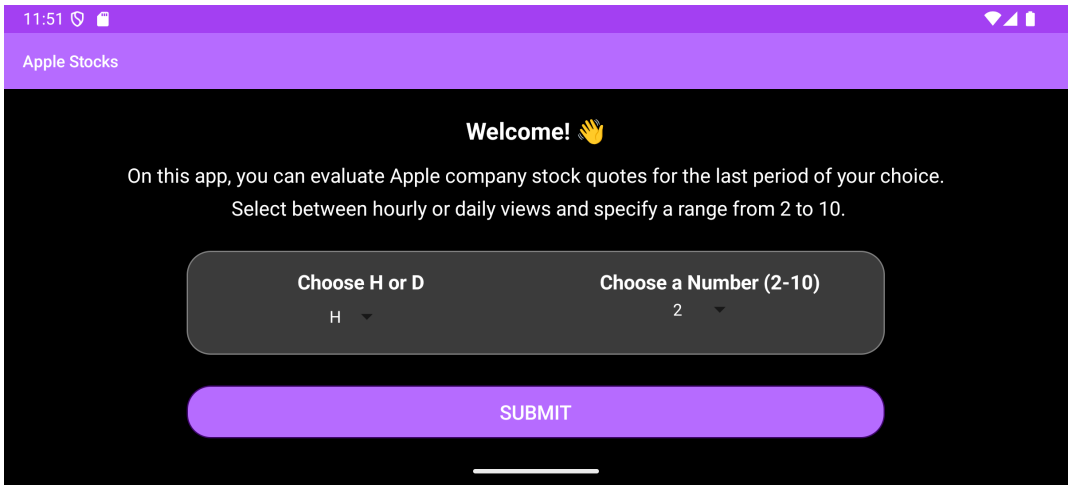
Below the plot, the maximum and minimum close values can be seen, ensuring an exact reading, when compared to a reading done from the plot. It is also possible to note the previoulsy mentioned maximum and minimum bars, in a grey dashed line, that help visualizing the results in a more graphical manner.

4.2.2 Loading Messages

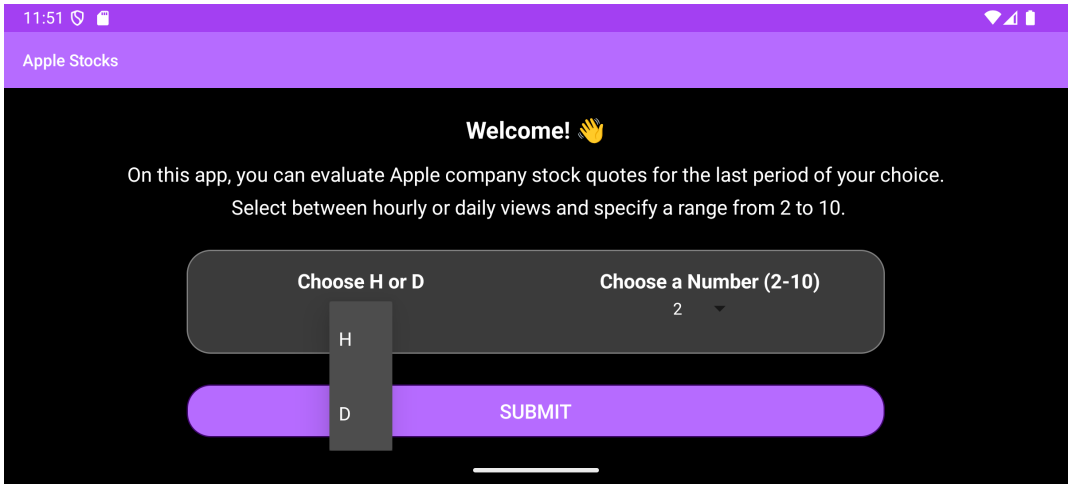
Due to the noticed delay, and in hopes of giving some feedback to the user, the loading messages were added, acting as placeholders for the plot and *extrema* values. The color choosen was yellow, as it is costumary to exemplify warning and loading messages, and the three dots also help to signify the loading process is temporary, making the app more responsive.

4.2.3 Landscape

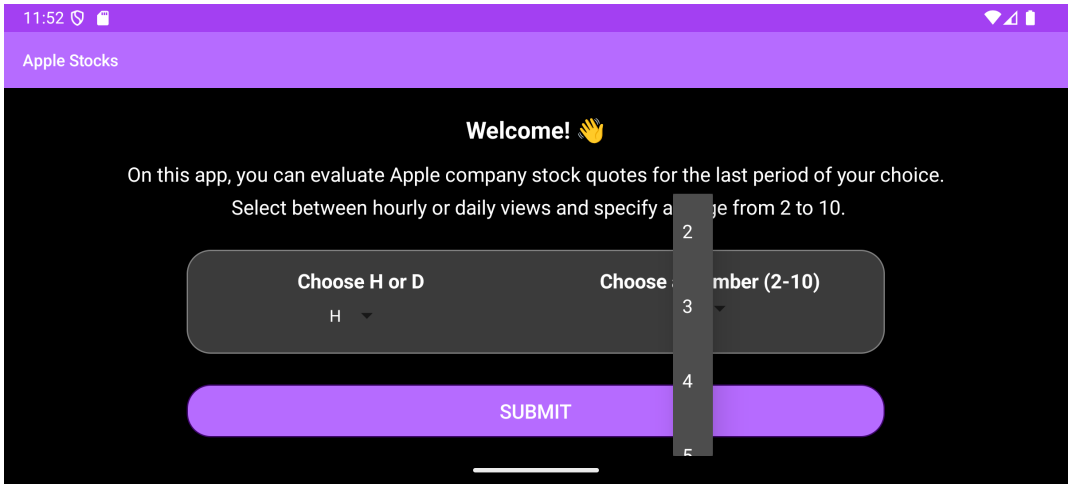
Just as in the main activity, a landscape layout was also designed, altough being very similar to the portrait layout. This similarity, however, is intentional, as the activity has the same function, and so it should not change significantly. The only differences are the placement of the maximum and minimum close values, that are now beside the plot, to ensure a more compact view of the plot, when compared to having the values bellow the plot.



(a) Main Activity.

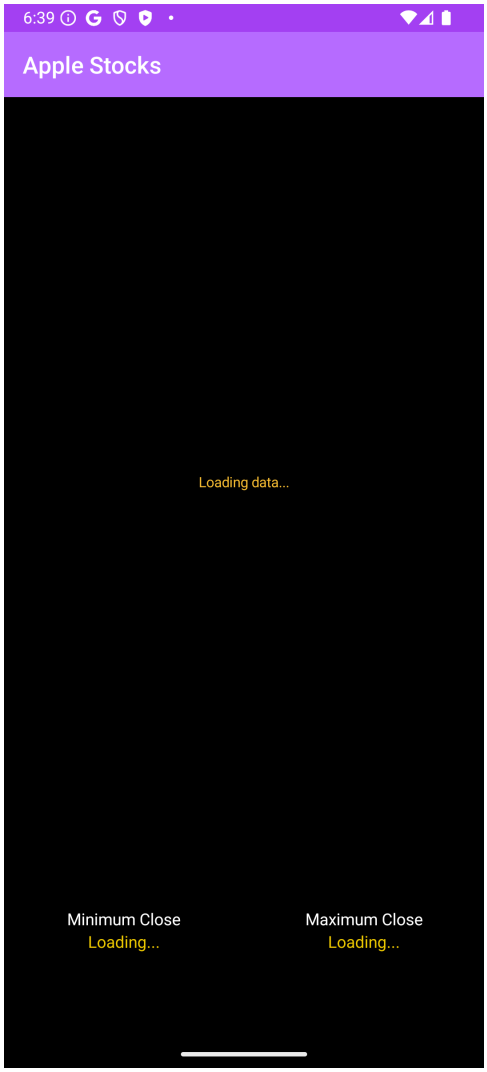


(b) Interval Choice.

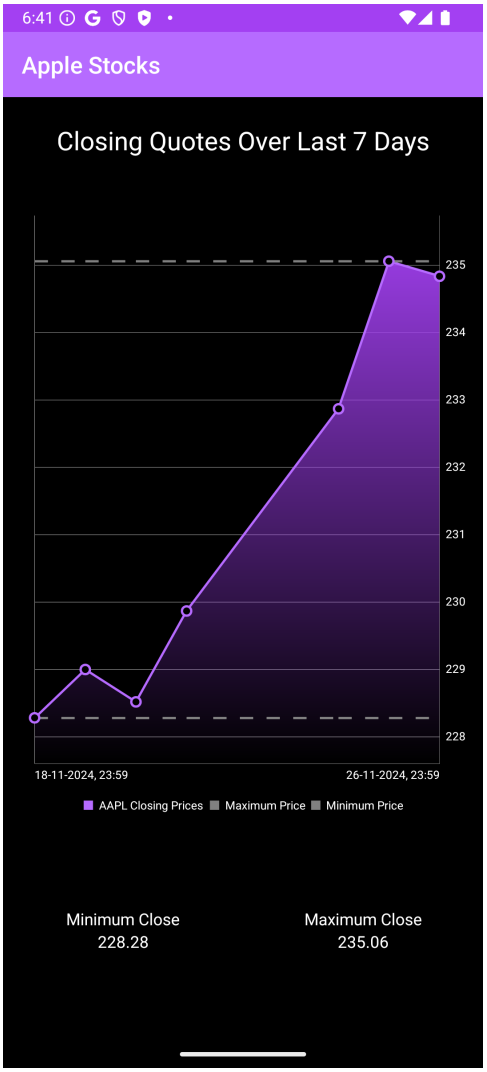


(c) Range Choice.

Figure 4: Main Activity in Landscape.

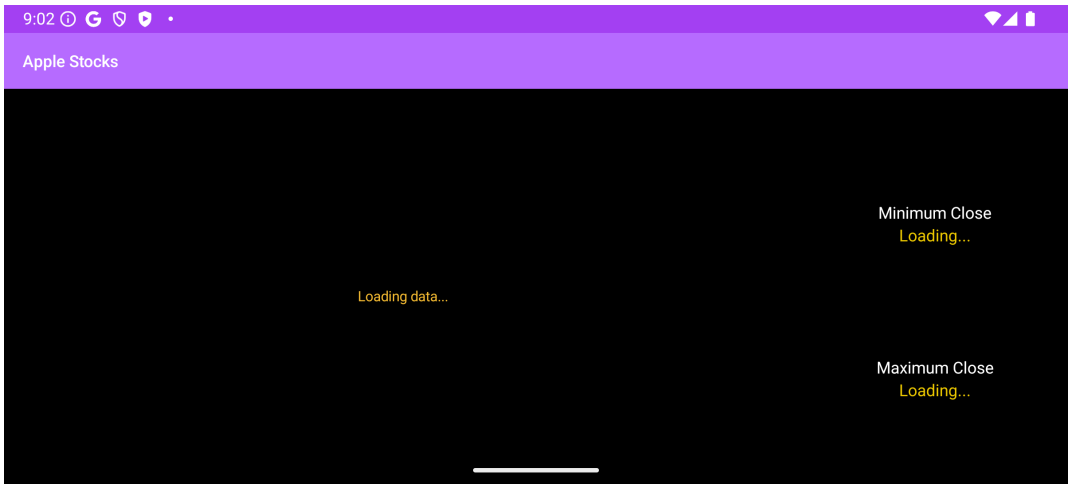


(a) While Loading.



(b) Example for 7 Days.

Figure 5: Plot Activity.



(a) While Loading.

(b) Example for 7 Days.

Figure 6: Plot Activity in Landscape.

5 User Experience

When presenting the interface, some key points for the user experience were already discussed, but the following aspects can be noted.

5.1 Ease of Use

The app provides a straightforward interface with clear instructions for users to configure their preferences. For example, default values in the spinners (e.g., "H" for hours and "2" for the range) simplify the experience for first-time users, allowing quick access to stock quotes without additional configurations. And the choice of spinners also helps the user to have a clear understanding of the choice they are making in regard to the time period and duration.

The layout is simple and intuitive, with the use of clear text labels and structured layouts, which also helps users less familiarized with mobile device use.

5.2 Responsiveness

To address potential delays caused by the API's response time, the app includes loading messages that inform users about the current state (e.g., "Loading..."), as discussed in subsubsection 4.2.2. This feature reduces frustration and assures users that the app is actively working.

In addition, by implementing error messages, the app provides informative feedback to users, maintaining transparency and trust.

5.3 Accessability

High-contrast color themes (black background with purple and gradient accents) are utilized to ensure the interface is visually distinct and easy to read, especially in the context of financial data visualization.

The app employs a clean and minimalistic design for its data plots, avoiding overcrowding by displaying only key points on the x-axis and supplementing graphical information with textual maximum and minimum values. Also, gradient fills enhance readability without causing visual fatigue, and dashed lines for extrema provide a quick graphical reference.

Another aspect, relevant for accessability is the landscape display, which imporves usability across various devices, including different phones and also tablets. The landscape mode can also be seen as a source of responsiveness, as the app adjusts dinamically with the device orientation.

Finally, the app uses a darker theme, which is consistent with similar apps available on the market, aligning user expectations with the industry's visual identity.

References

[1] Phil Jay. Mpandroidchart. `https://github.com/PhilJay/MPAndroidChart`, 2020. Accessed: 2024-11-27.