IFN666: Web and Mobile Application Development

Assessment 3: Capstone Project

N8844488 Connor McSweeney

Video Link: https://youtu.be/EEpSlN1yG8Y

Contents

Introduction **1**

Purpose & description 1

Completeness and Limitations 2

Technical Difficulties 2

Use of APIs **2**

FMP API 2

Alpha Vantage 2

Use of End Points of the chosen Stock API **2**

FMP API 2

Alpha Vantage 3

Alpha Vantage 3

Modules used 3

Navigation and Layout 4

Technical Description **5**

Architecture 5

Test plan 6

Extensions **8**

User guide **8**

References **9**

Appendices **11**

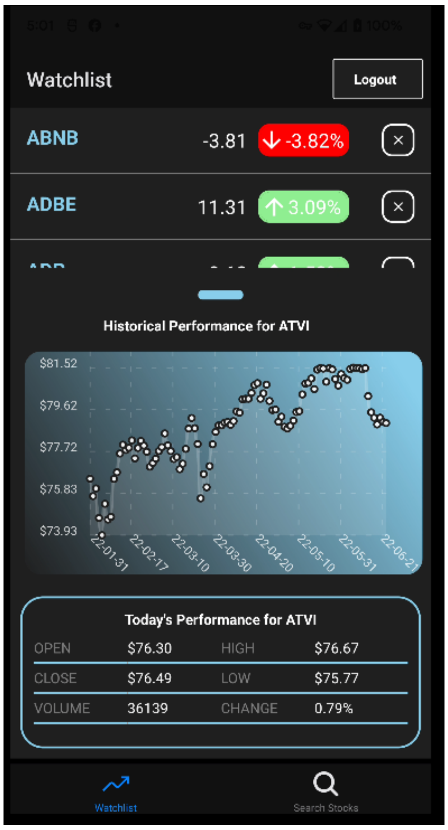
Appendix A – uncomfortable self-checking against CRA 11

Appendix B – Screenshots of test plan results 11

Introduction

Purpose & description

The purpose of this application is to allow users to analyze information relating to stock market performance - specifically, the Nasdaq 100. Using this application, users will be able to navigate, search and select a specific company that they would like to view in greater detail. Users will be able to create their own accounts and store a list of stocks that can be viewed at any point. This list saves to asynchronous storage (for offline access) and the database, so a user can access their account from any device. When viewing a stock, a user is provided with a detailed description of the company's historical performance. (Figure 1).



ABNB	-3.81	-3.82%
ADBE	11.31	3.09%

Today's Performance for ATVI

OPEN	$76.30	HIGH	$76.67
CLOSE	$76.49	LOW	$75.77
VOLUME	36139	CHANGE	0.79%

Figure 1

The application focuses heavily on functionality and system reliability. All functional requirements have been met and error handling has been implemented to ensure that the app is consistently stable and issues can be easily debugged. The implementation of consistent coding practices, standardization, and reusable components during the development of this project ensure that future additional functionalities or adaptations to the code can be made easily and with minimal impact on existing functions.

Completeness and Limitations

The application is fully functional and reaches all API endpoints described in the assessment specification displaying a list of the Nasdaq 100 stocks and historical performance. The data returned from each API request is stored within a global variable with the application. Before making any calls to an API, the application runs a check to confirm that the data does not already exist within the application. However, only the user email, password and watchlist are stored server side. This ensures that when a user receives the most up to date performance data for each stock when they log in. Performance data is presented in both chart and table format, and a search function has been implemented to provide filtering. The application makes use of static and dynamic web pages as well as custom styling. as per the assessment specification. The application is presented in a logical, easy-to-use format with error handling, dynamic loading, a loading screen, and application scaling for different window sizes.

Technical Difficulties

The main technical difficulties faced was the implementation of server-side functionality. Developing the middleware was relatively straightforward, but enabling the transfer of data from the local device to the virtual machine was an arduous task. Further complications arose in attempts to record the device. My laptop does not have the specifications required to run an emulator, so a third party software was used to project my device onto my computer.

Use of APIs

FMP API

https://financialmodelingprep.com/api/v3/nasdaq_constituent?apikey=

Alpha Vantage

https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol=${symbol}&apikey=

https://www.alphavantage.co/query?function=GLOBAL_QUOTE&symbol=${stock}&apikey=

Use of End Points of the chosen Stock API

FMP API

https://financialmodelingprep.com/api/v3/nasdaq_constituent?apikey=

This endpoint is used to gather the symbol, name and industry of the Nasdaq-100 companies (only 98 companies are returned in the query as shown above the table). This list of stocks can be filtered via the search bar using either the stock name or symbol as seen in *Figure 2*.

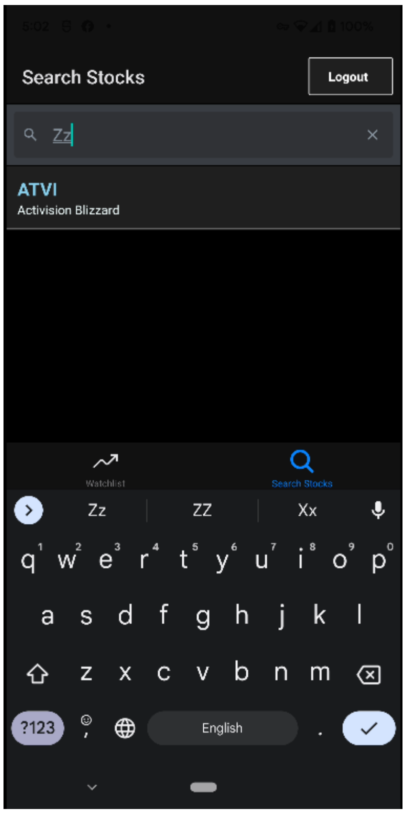


Figure 2

Alpha Vantage

https://www.alphavantage.co/query?function=GLOBAL_QUOTE&symbol=${stock}&apikey=

This endpoint is used to retrieve daily performance data relating to a particular stock. This data is displayed in a table shown in (Figure 1).

Alpha Vantage

https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol=${symbol}&apikey=

This endpoint is used to retrieve the historical performance data of a particular company and is presented in line chart format (Figure 3). The chart axis displays the fortnightly dates as well as the corresponding price.

Modules used

@rneui/themed

Cross platform UI Toolkit used to provide search bar functionality for the stocks screen.

https://www.npmjs.com/package/@rneui/themed/

@react-native-async-storage/async-storag

Provides asynchronous storage for the application.

https://www.npmjs.com/package/@react-native-async-storage/async-storage

@expo/vector-icons

Provides a range of assets that can be used as icons.

https://www.npmjs.com/package/@expo/vector-icons

@react-navigation

Provides navigation functionality for the React-Native application.

https://reactnavigation.org/

react-native-chart-kit

Provides chart building functionality in React-Native applications.

https://www.npmjs.com/package/react-native-chart-kit

react-native-easy-grid

Provides table/grid building functionality in React-Native applications.

https://www.npmjs.com/package/react-native-easy-grid

react-native-gesture-handler

Provides support for touch and gesture based functionality.

https://www.npmjs.com/package/react-native-gesture-handler

expo-status-bar

Used as the status bar within the application.

https://docs.expo.dev/versions/latest/sdk/status-bar/

reanimated-bottom-sheet

Provides functionality for a sliding bottom sheet within the application.

https://www.npmjs.com/package/react-native-bottomsheet-reanimated

Navigation and Layout

The application design was relatively straightforward and did not change beyond small layout alterations. The application incorporates five screens (including the bottom-up sliding screen), and the navigation for these screens is where I start my development - beginning with the search screen.

Once this was implemented and crudely styled, I went about developing the watchlist using similar styling. I then implemented the bottom sliding screen and the chart and table components for this. Finally, I began work on the middleware and the server-side database.

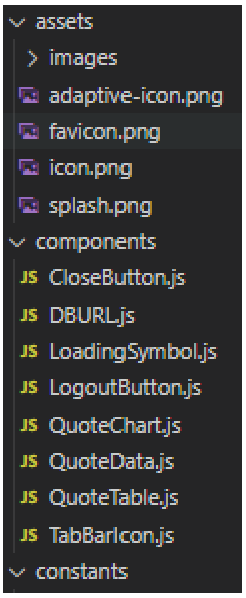
As visual design I focused on creating a highly functional page. I included all functional requirements in my design and spent much time ensuring that the app runs smoothly and handles edge cases well. I implemented error handling and logging where I deemed useful, and incorporated asynchronous functions at points to maximize time efficiency. Additionally, I included loading screens and symbols at points where lagging might occur.

Similar to my previous assessment, my application falls short in its visual aesthetic. I believe that with proper implementation of more sophisticated styling and layouts as well as improved table and chart design, my application could be dramatically improved.

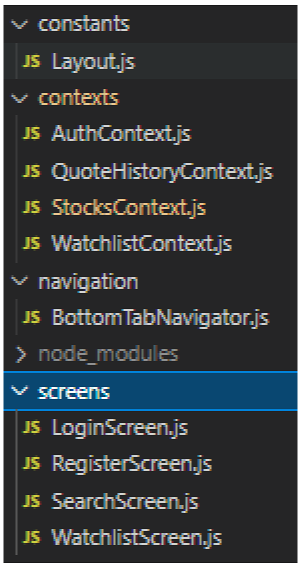
Technical Description

Architecture

That application source code is presented in a logical layout that uses functionality as its distinctive separator. Each distinct group of files (screens, components, images/assets, constants and navigation) are all stored separately. The navigation of the document is handled by a navigation container within the App.js.



```
v assets
  > images
    adaptive-icon.png
    favicon.png
    icon.png
    splash.png
v components
  JS CloseButton.js
  JS DBURL.js
  JS LoadingSymbol.js
  JS LogoutButton.js
  JS QuoteChart.js
  JS QuoteData.js
  JS QuoteTable.js
  JS TabBarIcon.js
v constants
```

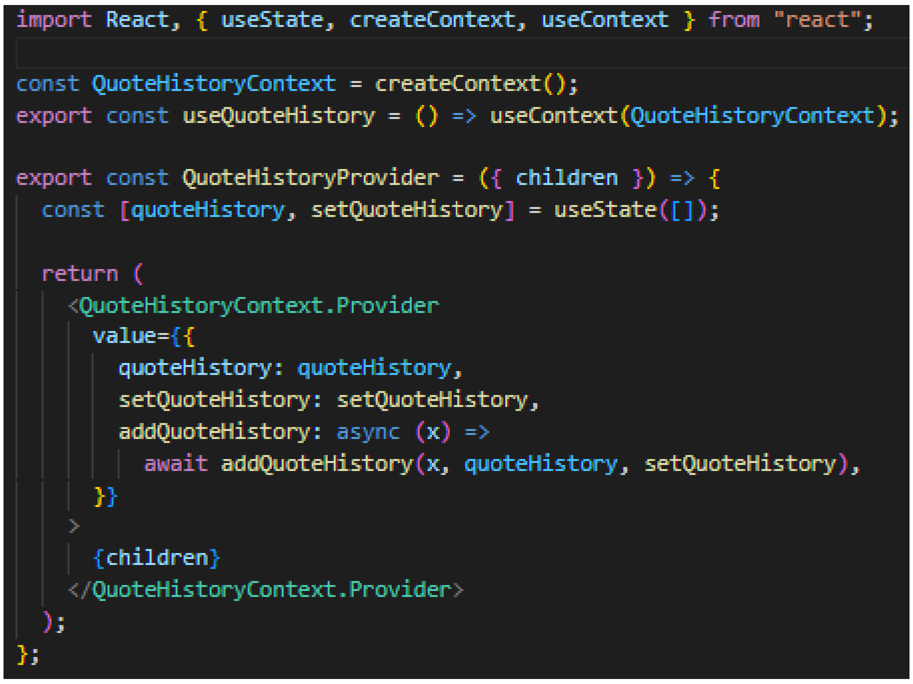


```
v constants
  JS Layout.js
v contexts
  JS AuthContext.js
  JS QuoteHistoryContext.js
  JS StocksContext.js
  JS WatchlistContext.js
v navigation
  JS BottomTabNavigator.js
> node_modules
v screens
  JS LoginScreen.js
  JS RegisterScreen.js
  JS SearchScreen.js
  JS WatchlistScreen.js
```

Figures 3, 4

The screens folder contains all of the viewable pages (excluding the bottom sliding screen), each of which is reachable by the user. These pages include the base layout as well as some necessary functions. All additional logic is either separated into components or housed within the context files.

The context files serve as global variables. Each contains a specific set of data that can be consumed by the rest of the application as well as handling the retrieval of that data and related functions. For example, *Figure 6* seen below displays the constant that is accessible to other files within the application. Via the returned ‘Provider’, a file may access certain constants and functions housed within the QuoteHistoryContext file.



```
import React, { useState, createContext, useContext } from "react";

const QuoteHistoryContext = createContext();
export const useQuoteHistory = () => useContext(QuoteHistoryContext);

export const QuoteHistoryProvider = ({ children }) => {
  const [quoteHistory, setQuoteHistory] = useState([]);

  return (
    <QuoteHistoryContext.Provider
      value={{
        quoteHistory: quoteHistory,
        setQuoteHistory: setQuoteHistory,
        addQuoteHistory: async (x) =>
          await addQuoteHistory(x, quoteHistory, setQuoteHistory),
      }}
    >
      {children}
    </QuoteHistoryContext.Provider>
  );
};
```

Figure 5

Finally, the images and assets folders house any images displayed by the application. There is only a single image used in this application as seen in the login and registration screen.

Test plan

Task	Expected Outcome	Result	Screenshot(s)
Checking navigation			
Check nav bar links to both the watchlist and search screens.	Nav bar links both pages.	PASS	Figure 1
Check nav bar links are highlights.	Nav bar links highlight when on set screen.	PASS	AS ABOVE
Check login and registration screens can be navigated between.	Pages can be navigated between using bottom page links.	PASS	Figure 2
Check successful login navigates to watchlist.	Successful login navigates to watchlist.	PASS	Figure 8
Check unsuccessful login.	Unsuccessful login doesn't change screen.	PASS	Figure 9
Check logout return user to login screen.	Logout returns user to logout screen.	PASS	Figure 6
Checking Watchlist			
Watchlist with no items.	Empty list message show	PASS	Figure 10
Watchlist with items.	Stocks shown	PASS	Figure 8
Watchlist scrolling	List with many items can be scrolled	PASS	AS ABOVE
Watchlist with new item	New item appears when added	PASS	AS ABOVE

Watchlist delete item	Item is deleted after removal	PASS	AS ABOVE
Watchlist stock data	Data relating to stock displays with different symbols for positive or negative change	PASS	AS ABOVE
Watchlist bottom sliding screen stock data	Chart and table display correct data	PASS	Figure 1
Watchlist bottom sliding screen positioning	Sliding screen can be moved up and down	PASS	Figure 1
Checking Search			
Search displays all stocks	Search displays all stocks	PASS	Figure 11
Search scrolling	Can scroll up and down	PASS	AS ABOVE
Stock search name	A stocks can be searched by name	PASS	Figure 2
Stock search symbol	A stocks can be searched by symbol	PASS	Figure 12
Stock is can be added to watchlist	Stock appears in watchlist	PASS	Figure 8
Checking Database and Middleware			
Register user	Registering a user displays in DB	PASS	Figure 13, 14
Login	User can login in using data housed in DB	PASS	FIgure 8
Update watchlist	Adding and removing from watchlist updates DB	PASS	Figure 14
Access watchlist	User sees watchlist after logging back in	PASS	FIgure 8
Checking Async Storage			
Update watchlist	Async storage is update when adding or removing from watchlist	PASS	Figure 8
Login token	User can query database using login token	PASS	AS ABOVE
Error handling			
Error logging	Errors are logged where appropriate	PASS	Figure 15
Robustness	App is reliable and fault tolerant	PASS	NA

Note: to avoid having too many images, the tests for window scaling, navbar link highlighting and navbar page linking have been captured with one screenshot for each test type respectively.

Difficulties / Exclusions / unresolved & persistent errors /

All functionalities outlined within the assignment specification have been met and the application does not currently contain any known bugs or output any known errors. Difficulties have been outlined with the technical difficulties section of the report.

Extensions

This application contains a number of customizable and reusable components for further extension of the application. In its current form, it contains only a small amount of market data. With the introduction of more data and more sophisticated modeling tools this website has the potential to become a feature rich market analysis application.

Currently, it contains no user authentication of back-end services. If the application was extended to include a database and fully-realized back-end, the application could house and store user information and preferences and become more personalized to a user's experience. Furthermore, implementing this functionally would reduce the requirement for the application to query external web services for information.

User guide

Upon starting the application you will be directed to the landing page (Figure x). From here the user can either enter their login credentials or navigate to the registration page. Incorrect or invalid entries into the fields provided will alert the user as to what requirements were not satisfied.

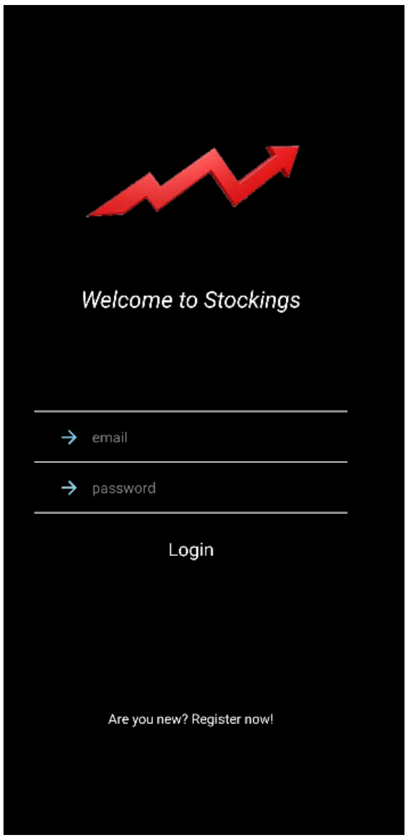


Figure 6

The registration page provides a user with the ability to register their own account. Similar to the login page, the user must provide valid entries into the fields provided to successfully register their account - they will be provided with prompts if an unsuccessful attempt is made.

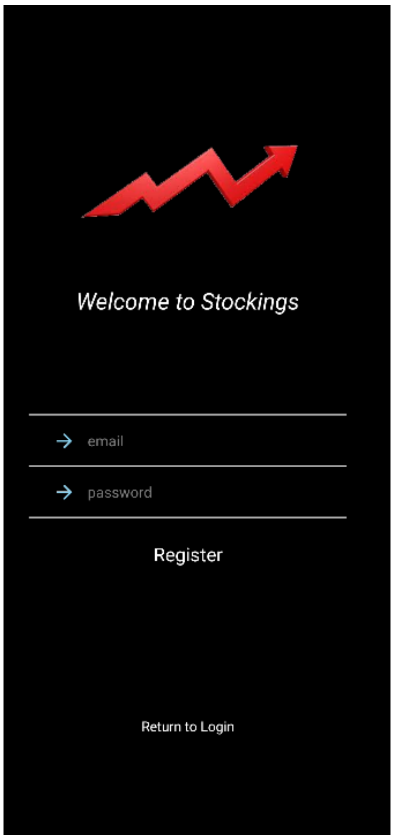
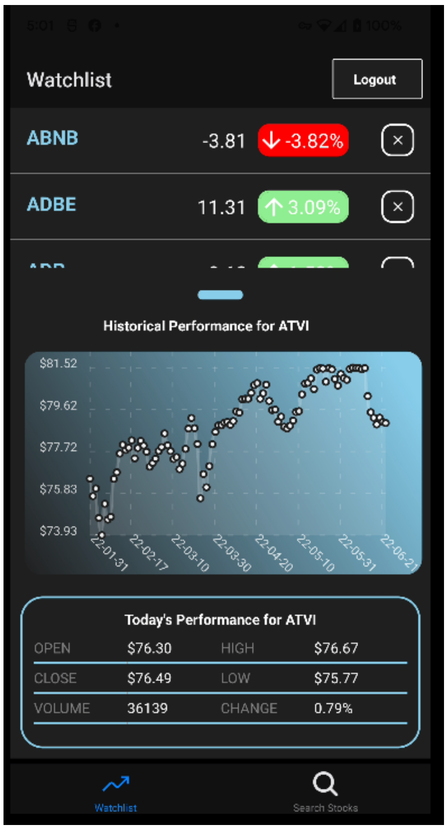


Figure 7

Upon logging in, a user is presented with their watchlist. If the user is logged in for the first time, or has no items stored in their watchlist, the list will be empty. Otherwise, the page will display a list of ‘watched’ stocks with the daily change data visible. The user can remove stocks from their watchlist or select a stock to view more details.



Today's Performance for ATVI

OPEN	$76.30	HIGH	$76.67
CLOSE	$76.49	LOW	$75.77
VOLUME	36139	CHANGE	0.79%

Figure 8

Selecting a stock in the watchlist will prompt a bottom sliding screen to appear. This screen displays the historical performance of the stock, and can be further extended upwards to display additional daily performance data. The top of the screen can be used to adjust the display's positioning. (Refer Figure 1).

The user can reach the search screen using the bottom navigator. Here, the user can search for and select stocks to add to their watchlist. (Refer Figure 2)

A user can select the logout button at the top right of the watchlist or search screen to logout and navigate back to the login screen.

References

API Documentation | Alpha Vantage. (n.d.). Alpha Advantage API Documentation. Retrieved June 1, 2022, from https://www.alphavantage.co/documentation/

Free Stock API and Financial Statements API - FMP API. (n.d.). Financial Modeling Prep. Retrieved June 1, 2022, from https://site.financialmodelingprep.com/developer/docs/

npm: chart.js. (2022, February 12). Chart,Js. Retrieved June 1, 2022, from

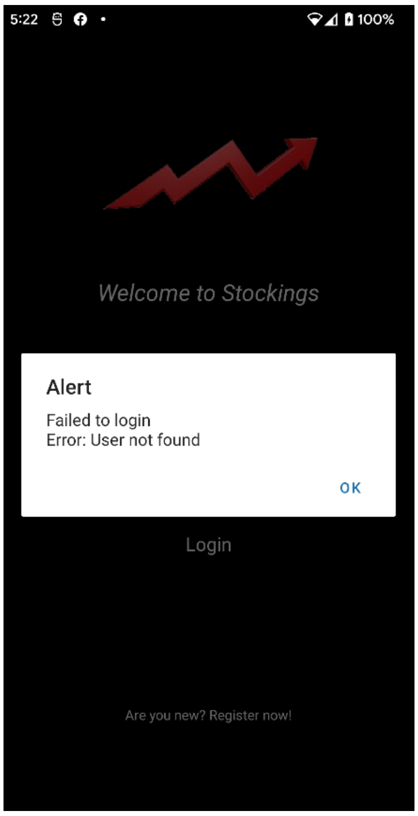
Note: several areas relate directly to my previous assessment, and as such, will closely match the analysis of my assessment 2 submission (e.g. api's used and their purpose, project scope, technical architecture).

Appendices

Appendix A – uncomfortable self-checking against CRA

Marks	Grade level (in 1- 7 scale) my work belongs to (delete the ones not suitable)	Marks I think I should get
Front-end Mobile Application Functionality (20 marks)	7	18 out of 20
Front-end Application Robustness (10 marks)	. 7	9 out of 10
Front-end Application UI Design (10 marks)	7	9 out of 10
Backend functionality, error responses and application reliability. (20 marks)	7	19 out of 20
Application architecture and Code Quality (10 marks)	7	9 out of 10
Back-end Application Architecture and Middleware – DB Connectivity, Logging and Security. (10 marks)	7	9 out of 10
Development Process Code Quality (10 marks)	7	8 out of 10
Report and Reflections (10 marks)	5	7 out of 10
Video demo (10 marks)	7	8 out of 10
Overall Marks (100)		87 out of 100

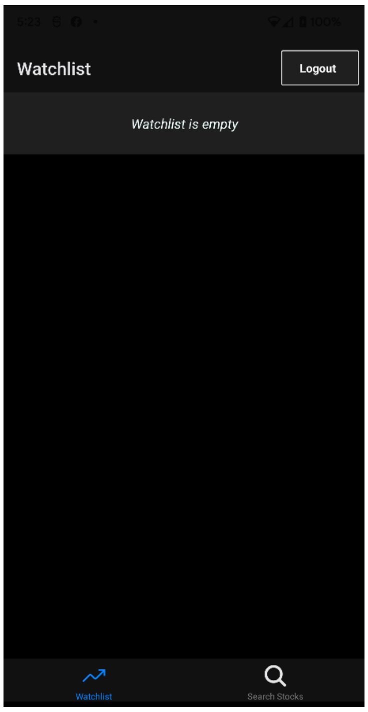
Appendix B – Screenshots of test plan results



Alert

Failed to login
Error: User not found

Figure 9



Watchlist is empty

Figure 10

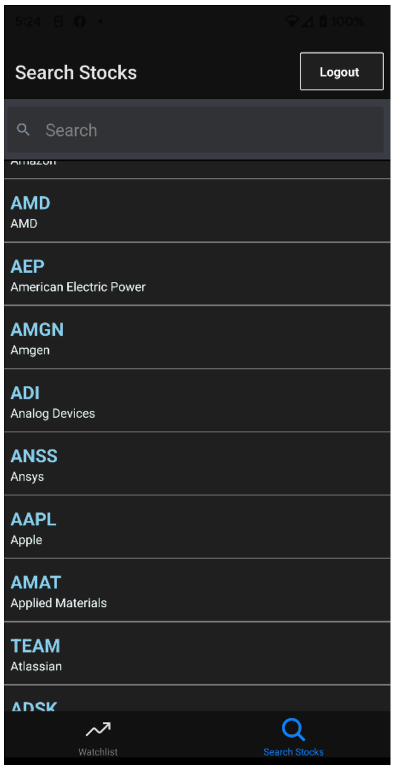


Figure 11

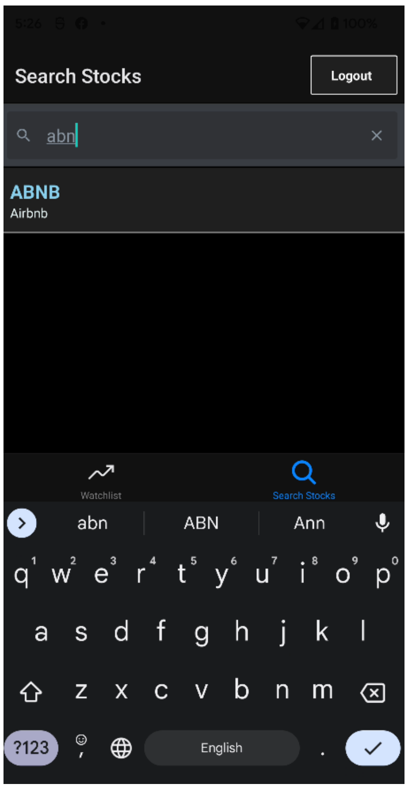
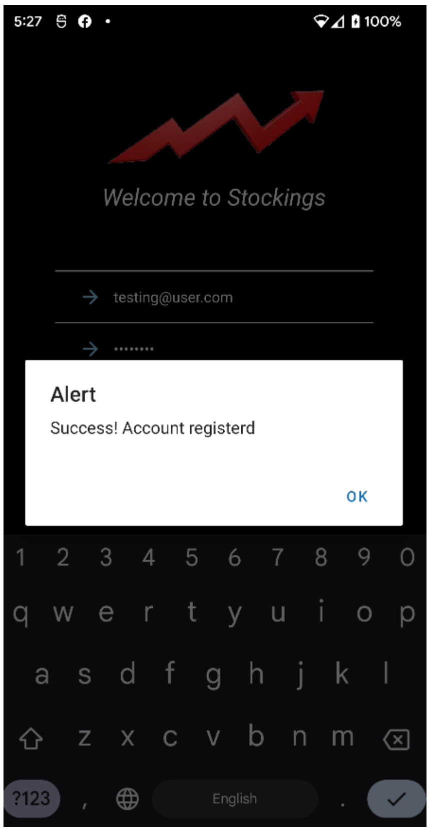


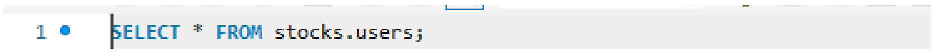
Figure 12



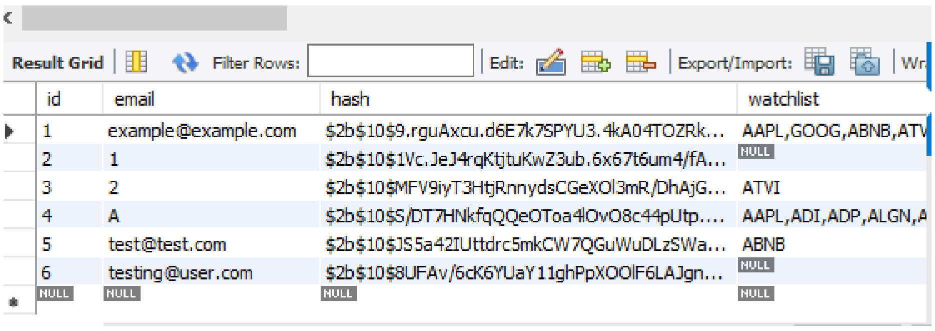
Alert

Success! Account registerd

Figure 13

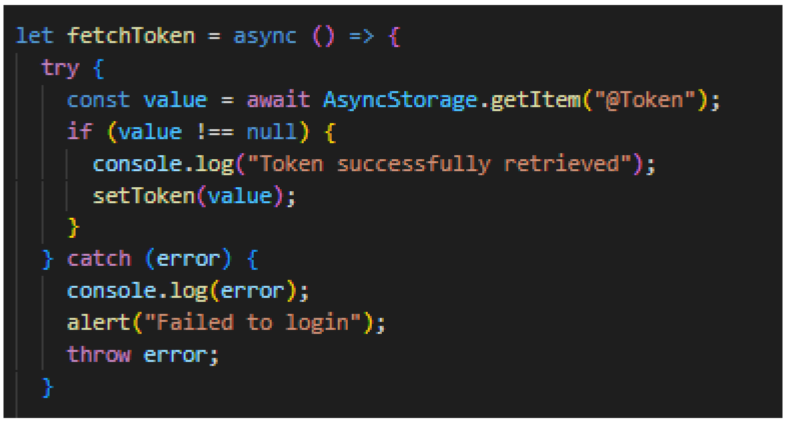


```
SELECT * FROM stocks.users;
```



id	email	hash	watchlist
1	example@example.com	$2b$10$9.rguAxcu.d6E7k7SPYU3.4kA04TOZRk...	AAPL,GOOG,ABNB,ATV
2	1	$2b$10$1Vc.JeJ4rqKtjtuKwZ3ub.6x67t6um4/fA...	NULL
3	2	$2b$10$MFV9iyT3HtjRnnydsCGeXOl3mR/DhAjG...	ATVI
4	A	$2b$10$S/DT7HNkfqQQeOToa4lOvO8c44pUtp....	AAPL,ADI,ADP,ALGN,A
5	test@test.com	$2b$10$JS5a42IUttdrc5mkCW7QGuWuDLzSWa...	ABNB
6	testing@user.com	$2b$10$8UFAv/6cK6YUaY11ghPpXOOlF6LAJgn...	NULL
NULL	NULL	NULL	NULL

Figure 14



```
let fetchToken = async () => {
  try {
    const value = await AsyncStorage.getItem("@Token");
    if (value !== null) {
      console.log("Token successfully retrieved");
      setToken(value);
    }
  } catch (error) {
    console.log(error);
    alert("Failed to login");
    throw error;
  }
```

Figure 15