PROJECT REPORT

on

Stock Market App using Ionic

Submitted to

Amity University Uttar Pradesh in partial fulfilment of

requirements for the award of the Degree of

B.Tech Computer Science 2015-2019

Submitted by

Aayush Jain

A2305215457

Guided by:

Ms. Divya Upadhyay

DECLARATION

I, **Aayush Jain** student of **B.Tech CSE** hereby declare that the project titled "*Stock Market App using Ionic*" which is submitted by me to Amity School of Engineering and Technology, Uttar Pradesh, Noida, in partial fulfillment of requirement for the award of the degree of Bachelor of Technology, has not been previously formed the basis for the award of any degree, diploma or other similar title or recognition.

The Author attests that permission has been obtained for the use of any copy righted material appearing in the Dissertation / Project report other than brief excerpts requiring only proper acknowledgement in scholarly writing and all such use is acknowledged.

Noida

Date

Aayush Jain

(Name and Signature of Student)

CERTIFICATE

This is to certify that **Mr. Aayush Jain**, student of **B.Tech**. in Computer Science and Engineering has carried out the work presented in the project of the Term paper entitled "*Stock Market App using Ionic*" as a part of Second year of Bachelor of Technology from Amity School of Engineering and Technology, Amity University, Noida, Uttar Pradesh under my supervision.

Ms Divya Upadhyay
Assisstant Professor
Amity Institute of Science and Technology, AUUP

TABLE OF CONTENTS

Abstract v

Introduction 6

Methodology 8

Results and Discussion 10

Conclusion & Recommendations 14

Future Scope: 14

References 15

ABSTRACT

Web development technologies have evolved at incredible pace in the past few years. Fast mobile browser rendering speeds and better JavaScript performance have enabled mobile devices to run hybrid apps and deliver performance that is at par with native apps. Ionic is one such framework built on top of AngularJS and Cordova that allows developers to quickly prototype mobile apps.

The following project involved developing a full-fledged Stock Market app in Ionic, and then deploying it on the Android platform. The final app allows a user to see a stock's detailed data, historical graph, and related news. It also has a notes functionality, and a realtime search functionality built in.

INTRODUCTION

About 80% of the consumer's time is spent on mobile apps. An average smartphone user spends about 158 minutes per day on his device. This introduces a huge opportunity for mobile app developers to reach a greater market.

Until recently, there existed three ways to make mobile applications:

1. Native Apps: Platform specific apps can be made by using their respective SDKs and IDEs. For developing iOS apps, one needs to:

    - have a Mac computer
    - download Xcode from the App Store
    - buy the Apple Developer license that costs 99$ per year

    For developing Android apps, one needs to:

    - download the appropriate SDKs
    - buy the Google Developer license which is 25$ one-time registration fee

    Since native apps are written specifically for their respective environments, they need to be written from scratch. This allows the native apps to exploit the hardware and run efficiently. Thus native apps had been considered to be delivering a higher performance.

2. Mobile Websites: Mobile sites are actually the normal sites that we visit on the web browsers. However, these have been designed to adapt to a phone's small screen. This is achieved by adopting the practice of "responsive website design". In such a scenario, only one HTML codebase is needed. The look for the devices is determined by the media queries in the CSS file.
    One advantage of using the mobile site format is that we can update them without

waiting for review and approval from the respective app stores.The disadvantage of these is that mobile sites can't use properly all available phone resources to provide a better experience to the user.

3. Hybrid Apps: A hybrid app is a mobile application, written with the same language that one uses while designing websites with the addition that it contains a WebView which runs the web application inside a native app.
   For years, hybrid apps were looked down upon by native apps for being clunky and having slow rendering speeds. However, in the recent years, thanks to fast mobile browser rendering speeds and improved JavaScript performance, developers can now design, build and deploy hybrid apps with no negative effects on the app performance. Hybrid Apps can be made using various frameworks such as: Xamarin, Onsen, Ionic, React Native, PhoneGap, etc.

This project uses the Ionic Framework to develop the Stock Market Application. Ionic is a brilliant framework that allows web developers to build beautiful and interactive mobile apps using HTML5 and AngularJS. Ionic enables developers to work on just one codebase, and lets them deploy the same code on both iOS and Android platforms, while building the app using web technologies like HTML5, JavaScript, TypeScript, AngularJS and SASS.

METHODOLOGY

The app required the following tools, softwares, packages and services for its fruitful completion:

1. NodeJS Package Manager: Node enables real-time web applications to employ push technology over websockets. To rephrase that, Node allows client and host applications to establish a two-way communication channel without a sandboxed environment. This technology empowers other frameworks like Ionic to build upon this platform. The NPM command-line utility allows easy installation of required packages for different projects.
2. Ionic: The Ionic framework installed using NPM is what has been used at the core of this project. This allows us to code our application using JS and CSS, and lets us serve them on our localhost.
3. Cordova: Cordova is a mobile application development framework that enables wrapping up of CCS, JS and HTML code depending on the platform of the device.
4. Gulp: Gulp is another npm package that is used for automation of time-consuming and repetitive tasks involved in web-development like minification, concatenation, optimization, linting, etc.
5. Atom: Atom is an open source text editor developed by Github. It is a desktop application built using web technologies, and has support for plugins written in NodeJS.
6. Git and Github: Git is a Version Control System that allows programmers to track changes in files, and also allows collaboration between users. Its project management features include adding commits at major changes, using tags, managing branches and many more.
   Github is a web based Version Control System that allows hosting open-source software projects on the internet. It has support for Git's SCM and DVCS capabilities. It also adds features of its own, such as bug and issue tracking, wikis, adding collaborators, etc.

7. Android SDK: The SDK is used in conjunction of Cordova, that enables converting the Ionic project to an Android based project, and generates an APK.

RESULTS AND DISCUSSION

The app has the following views:

1. **My Stocks List:** This is the first view that the user sees when he first launches the app. It has the list of apps that the user is following, and presents the latest stock data in a user-friendly way allowing him to quickly glance at the data. It also sports a ‘pull-to-refresh’, and a ‘swipe-to-unfollow’ feature.
2. **Stock Details View:** This screen comes into view when the user clicks on a stock ticker. This view has the following components:
    - A red or green header bar with the stock name and current price, representing the stock status.
    - An interactive volume vs date, and price vs date graph with the stock’s historical data.
    - Various stock details such as 52 week high and low, volume, EBITDA, market capitalization, etc with data sourced from Yahoo’s API.
    - A notes section enabling the user to save, access and delete notes.
    - A news section with news related to the stock.
3. **Search View:** This screen pops up when the user wants to search for stocks to follow. It sports realtime search functionality.

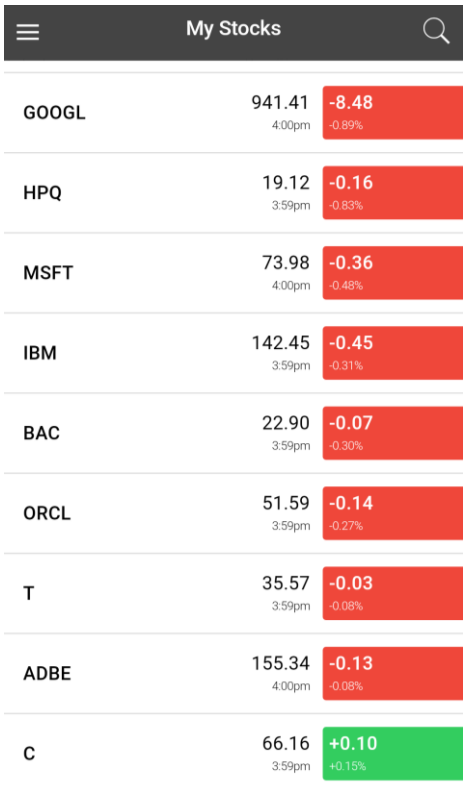

Figure 2: My Stocks View

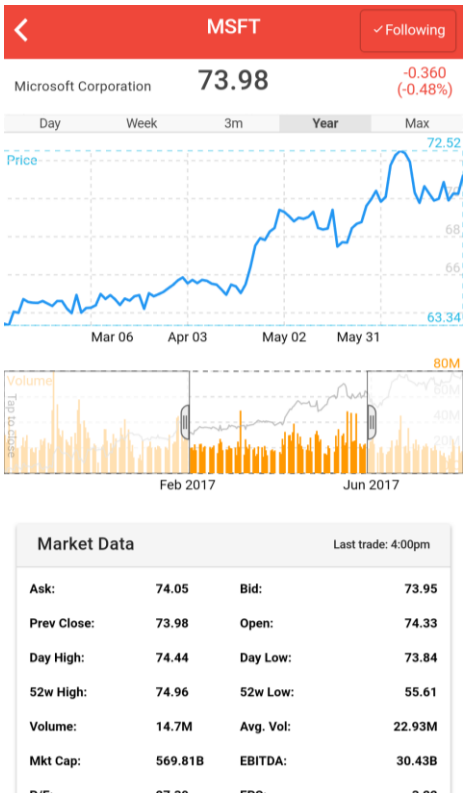

Figure 1: Stock Details View

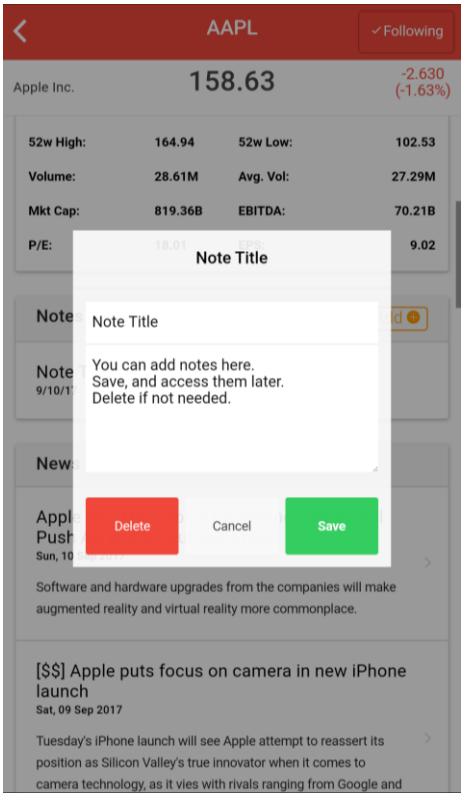

Figure 3: Notes and News Functionality

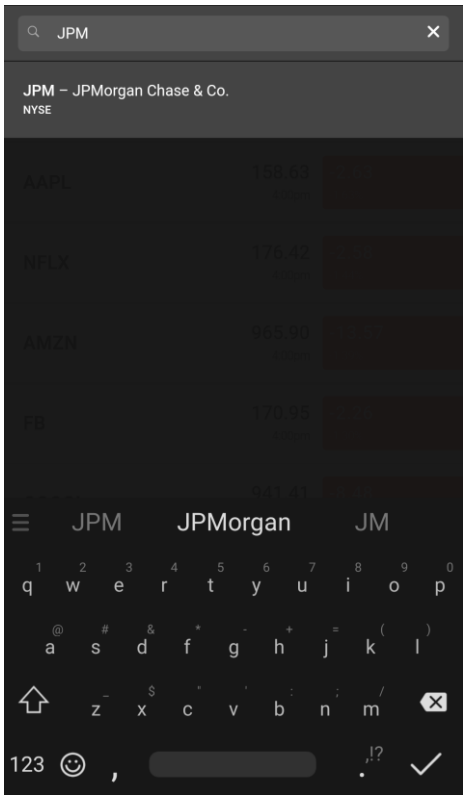

Figure 4: Search Functionality

Some of the components that the app incorporates are:

1. Stock Price Service: This is powered by Google Finance's Stock API, that returns realtime stock price data such as current price, change, change percent, last trade time, etc. Although deprecated in 2012, the API is still publicly accessible, and needs some hacking to run. The obfuscated JSON array is first cleaned up using regex, and then converted back into a JSON object for our use.
2. Stock Details Service: This is powered by Yahoo's Stock API that returns a stock ticker's details such as Market Cap, Ask, Bid, Year High, Year Low, EBITDA, PE Ratio, Average Daily Volume, etc as a JSON object when requested via an http request. The request is routed through the YQL servers, so as to deal with the cross-origin-resource-sharing (CORS) issue.
3. News Service: This is powered by Yahoo Query Language Console, that processes a stock ticker's RSS feed, and returns a JSON object containing news items related to the stock. The JSON object is converted into a JSON array, parsed, and cleaned up using regex to remove and replace incoherent HTML characters.
4. Search Service: This is powered by Yahoo Finance's Autocorrect API, that returns a list of stock ticker symbols when some search input is fed into it. This service is realtime in nature, and returns the stock names as they are being typed.
5. Chart Data Service: This is powered by Quandl's API, pulling in data from its public WIKI database that contains all latest and historical data of various stocks. The data is formatted into a format recognized by the nv-chart library that renders the two graphs in the app. The first graph contains the date versus price data; and the second graph contains the date vs volume data of the stock. The chart has an interactive slider which the user can slide to focus onto a particular time period, and see the graph in a detailed form.
6. Follow Stock Service: This service checks if the stock opened is being followed by the user. If the user is already following it, then an 'unfollow' button is present for his needs. Following or Unfollowing a stock involves the insertion or removal of the corresponding stock ticker symbol from the user's followed stocks array list.
7. Notes Service: This service enables a user to add, access, and delete a note for every stock he follows. This is implemented in the form of a popup on the user's current

screen so that he can quickly add or read a note. The note is stored in the app's cache, and is removed only when the user deletes the note.

8. Date Service: This is used internally by the chart data service to access the current date, and a one-year ago date. This is also used to convert the date to milliseconds for the nv-chart library.
9. Shrink Number Service: This is used internally throughout the app, so as to represent large numbers by rounding them off to the nearest thousand, million or billion, and add the decimal in the appropriate place. This is used to make the app look aesthetically more pleasing to the eye.
10. Character Service: This is used internally in the news section of the app, to add an ellipsis(…) at the end of long paragraphs so as to reduce the amount of text on the screen. This is implemented so that the news story description does not get too long, and breaks with an ellipsis at 200 characters.
11. Cache Service: There exist separate cache services for the stock price service, stock details service, notes cache service, followed stocks cache service, and chart data cache service. This has been implemented using Angular's Cache Factory library. Each separate cache service has been implemented so as to save the user's internet data usage, and also prevent unnecessary load on respective API servers. Using a cache service saves time, as the app does not have to wait for external servers to return the required data; instead it is loaded from cache memory. Each service also exists with different expiry periods for the stored data, so as to keep it fresh. This improves the load time of the app.
12. In-App Browser Service: This has been implemented using Cordova's plugin to aid the user quickly visit a news story's webpage without leaving the app. The browser has been configured to not store any cookies or cache, so as to not take unnecessary space on the user's device. This allows the app to have a smaller footprint.

CONCLUSION & RECOMMENDATIONS

The StocksApp Android application has successfully achieved all the targets it had set at the time of its conception, and is now fully operational with no bugs. It has a size of 3.47MBs. The application has been built for the Android platform using the Android SDK, and Cordova. The final release version of the application is designed for Ahead-Of-Time compilation for faster load times on modern devices. Additionally, all JavaScript and CSS files have been minimized into one file for lagless performance. The JavaScript files have been further optimized using Ionic CLI for a snappy response. The entire source code for the application has been uploaded on Github for others to learn, and collaborate.

Future Scope:

Since this project involved the use of Ionic, we can use this same codebase to build the app for Apple's iOS Platform. However, this requires an Apple Developer License worth $99/year, and a MacBook. Besides, all Apple devices come bundled with a pre-installed Stocks App, thereby killing the need for other similar apps. Nevertheless, if the market does dictate a need for the StocksApp application to appear in iPhone's App Stores, then surely it would be worth considering.

Another feature worth considering is the user's ability to save his followed stocks on the cloud, letting him login on any device, and see his portfolio. However this involves setting up additional servers for the app, and host the users' data which turns out to be a costly option for an app that chooses to not display annoying ads, thereby generating no revenue.

REFERENCES

1. **Git Documentation - Reference**. (2017).
   Retrieved from *https://git-scm.com/docs*

2. **Guide to AngularJS Documentation**. (2017).
   Retrieved from *https://docs.angularjs.org/guide*

3. **Ionic Documentation - Ionic Framework**. (2017).
   Retrieved from *https://ionicframework.com/docs/v1*

4. **Quandl Documentation**. (2017).
   Retrieved from *https://docs.quandl.com/docs/in-depth-usage*

5. **Starting your app - Ionic Framework**. (2017).
   Retrieved from *http://ionicframework.com/docs/v1/guide/starting.html*

6. **YQL - Yahoo Developer Network**. (2017).
   Retrieved from *https://developer.yahoo.com/yql*