Dissertation

Developing a Progressive Web App

Bradley Birch



Figure 1: PWA (Nath, 2017)

Progressive Web Apps (PWA’s) are new ways to engage users on the web. They allow developers to create app like experiences on a website. This includes offline support, push notification and access to device hardware (Google Developers, 2017). Developers get all of this without having to write a native mobile app or releasing the app on a traditional mobile app store. The aim of this dissertation was to build an offline capable technical analysis tool for financial markets. It leveraged publicly available data sets from the finance firm IG to perform the analysis. During the development I worked with various hardware and software teams to get the project running. The results from this is an application that can perform various technical analysis, such as Moving Average Convergence/Divergence, on a range of financial markets.

Table of Contents

[Introduction 3](#_Toc519246445)

[Aim 3](#_Toc519246446)

[Objectives 3](#_Toc519246447)

[Rationale 4](#_Toc519246448)

[Literature Review 6](#_Toc519246449)

[PWA’s 6](#_Toc519246450)

[Technical Analysis 8](#_Toc519246451)

[Methodology 9](#_Toc519246452)

[Planning 9](#_Toc519246453)

[Software Requirements 11](#_Toc519246454)

[JavaScript and Ember.js 11](#_Toc519246455)

[CSS – Using a Pre-processor 14](#_Toc519246456)

[Data Requirements 14](#_Toc519246457)

[Design 16](#_Toc519246458)

[Software Design 16](#_Toc519246459)

[Data Models 16](#_Toc519246460)

[Look and Feel of the App 16](#_Toc519246461)

[Implementation 16](#_Toc519246462)

[Code 16](#_Toc519246463)

[Testing 17](#_Toc519246464)

[Results and Discussion 17](#_Toc519246465)

[Quality of the App 17](#_Toc519246466)

[Quality of the Code 17](#_Toc519246467)

[Quality of the Development Process 17](#_Toc519246468)

[Conclusion 17](#_Toc519246469)

[Bibliography 17](#_Toc519246470)

[Appendix 19](#_Toc519246471)

# Introduction

## Aim

The aim of this project was to build Progressive Web App (PWA) that performs technical analysis on a range of financial markets for the finance firm IG. The application can gather historical price points for shares, indices, commodities and forex. These data points can then be run though the following calculations:

* Moving Averages
* On-Balance Volume
* Relative Strength Index
* Moving Average Convergence/Divergence
* Covariance
* Back Testing a Strategy

All of these indicators allow traders to be better informed about when to place a trade on market, show how two markets might be related or how a successful a trading strategy might be. The application will need to compute these calculations correctly and display them to a user in an easy to read and understandable manner.

## Objectives

For the application there were high level objectives that the project depended on. These were generated from conversation with client facing staff at IG. These staff interact with IG clients every day and hear first-hand exactly what clients are after.

* Offline capable. The application should have some limited functionality what the user is offline. This can include saving the last searched data points on a market and being able to perform technical analysis on that market.
* Privacy. The calculations should occur on the user’s device and no data should be sent to IG. This ensures transparency between the user and IG.
* Wide range of markets. IG should be able to offer all markets that it offers on its trading platforms so that clients can make fully informed decisions. This includes shares, indices, commodities and forex.
* Highly interactive. Once chart have been generated by the application the user should be able to interact and drill down in to the data set displayed by the chart.
* Explanatory. Indicators are hard to understand. Provide information that is not available on IG’s trading platforms. The application must complement IG current systems and include analysis it does not have elsewhere. Including, Accumulation/Distribution Line, Average Directional Index, Aroon, On-Balance Volume, Relative Strength Index, Moving Average Convergence/Divergence, Covariance, Back Testing a Strategy.
* Mobile. The analysis much work on a mobile device.

## Rationale

This application was built to allow IG clients to perform technical analysis on a mobile device using IG data. Technical analysis is the process of statistically analyising the price movement of markets such as a share, security, forex or commodity. The analysis past trends informs the trader of future trends as they believe that history tends to repeat itself as price movements can become repetitive. Once various analysis has been performed traders can gauge if a market price is likely to rise, fall or stay steady. They can understand if two markets are related and if one changes how likely it is another will change. For example, if the price of oil rises it is likely the shares in air companies will fall. Being able to identify these relationships and trends is incredibly important to traders.

Currently IG has technical analysis tools in various places. Most of the analysis is used in its charting package. Figure 2 shows a chart on IGs main trading platform with a Price Oscillator and MACD indicator.



Figure 2: IG technical Analysis Tools

There are many more charting indicators on the platform and they help traders work out patterns in a single market. You are only to show one market on a single chart and the ability to compare and contrast markets is a challenge. This functionality is restrained to the desktop platforms. You cannot access these charts and therefore these indicators online.

IG also offers metrics such as client sentiment and related markets. This is shown in figure 3. This allows you to see what other traders are doing in regards to a certain market. These figures can be found on a separate location on IG’s website. These metrics can be viewed on a mobile screen.

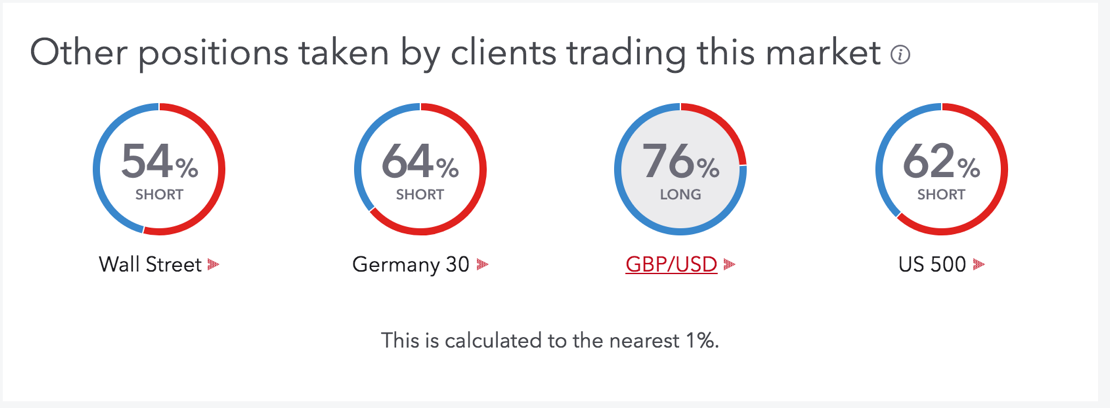


Figure 3: Client Sentiment

Within this project, first, I look to expand upon what IG already offers. I will not be looking at client sentiment instead raw data in a single market or market pairs. I will allow traders to perform flexible analysis on more than one market. I will do it in an informative and clear manner. This will make the app appeal to beginners and pro’s to maximize usage. IG believes that if better analysis can be done by traders they will attract more clients to use their systems. This in turn will have the potential to generate more revenue of IG.

Secondly, I will look to make it a mobile first experience. If you want to use IG products on the go right now you have to use a laptop. The mobile versions of charts are very basic. This is because the desktop charts are so complex it is difficult to port all of their functionality to a mobile device. This project aims to find a sweet spot between functionality and usability of the charts. If traders have more opportunities to use charts and analysis offered by IG than any other competitor then they will stick with IG. Plus, more and more people are using the internet on their mobiles as shown by figure 4. In order to have a slice of this mobile market it is vital to have a strong mobile presence. Brands need responsive apps with progressive enhancement that can work on a variety of mobile devices. If a brand does not have this it is highly likely that another brand will customers will move. This app is meant to keep IG moving with the mobile trend.

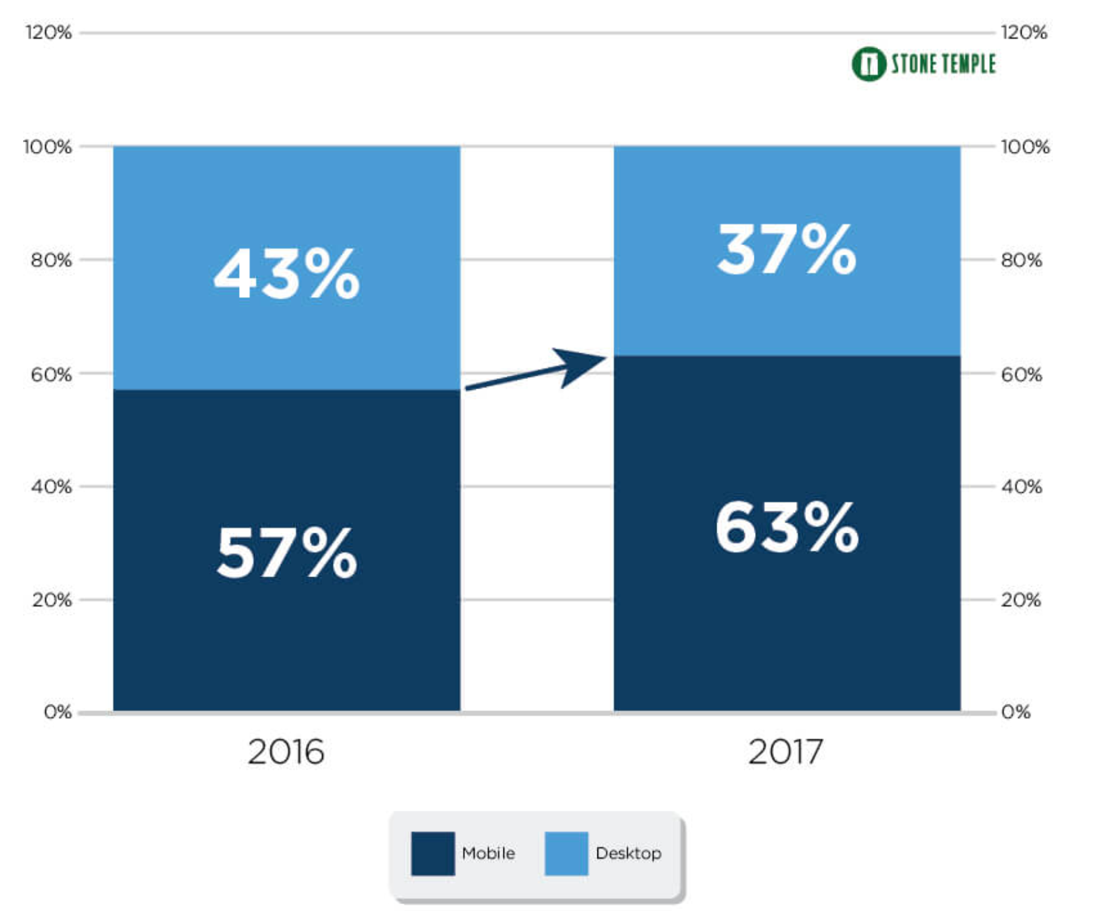


Figure 4: Internet Usage in the US (Enge, 2018)

# Literature Review

In order to fully appreciate and to make the most out of this project and application I conducted a literature review of PWA’s and technical analysis. This research was designed to inform me whether PWA’s are viable alternative to native apps. And if technical analysis is popular enough to warrant the time investment. I conducted this review online and talking to collages at IG.

## PWA’s

PWA’s were designed and created in 2015 by Google as a way to improve upon native mobile apps (Crook, 2017). PWA’s are app-like experiences, they run in a browser but without all of the browser chrome such as the address bar. In this way it feels like you are using a real native app. They are designed to be the best of both worlds from the web and native apps. This means the great features you get on native apps you will also get on PWA’s.

Offline functionality. PWA’s can take advantage of a Service Workers. This is a JavaScript file that runs separate from the main JavaScript thread. A Service Worker can cache static files like HTML pages. But it can also cache content fetched from a server such as news articles or travel updates. These resources can then be served to the user right away without having to go over the network to get them. While this helps with app load times it also means the app can be used offline. This is a real benefit if a user is browsing with a mobile data connection that can often cut out or have a poor signal.  
For my project I will be able to cache recently downloaded market data and I will be able to use it when the user is offline. Since all the calculations will happen on the user’s device there will not be a need to go over the network.

Access to hardware features. PWA’s have access to certain hardware features on a device. These include camera, accelerometer, location and battery level to name a few. These features are included as easy to use API’s in the browser. They are first class citizens in PWA’s and it means making a fully featured app very easy.

Push notifications. Notifications are vital to keeping users engaged with an application. Push notifications remind people to come back to the app and finish what they are doing. For my application this is important because a user could perform another task while my app performs a calculation. After the calculation is finished I would then let the user know they can come back to the app.

However, all of these features listed are what native apps can already do at the moment. Why should I not just build a native app. After my research these are the reasons why I choose to build a PWA (Naylor, 2017).

Works everywhere. PWA’s are built on existing web technologies and in browsers. This means that anyone who has a mobile browser can run a PWA. In 2017 there were 3.77 billion internet users (Kemp, 2017). That gives PWA’s a huge reach considering if I made an iPhone app I could only reach 700 million users (Reisinger, 2017). This maximizes income potential and it saves a lot of time. Since I only have to build one application I can focus on building features and not two apps.

No App stores. No getting a license or developer account to push the app to the app store. When I am ready to release my application all I have to do is deploy it and it is out there. When I want to update it all of the updates will be available automatically because PWA’s do not store the app on the phone. If there is an internet connection they will always grab the latest version.

Fast load times. According to Google (SHILOVA, 2017) PWA’s very fast. For example; Twitter Lite loads in 3 seconds on a slow connection and Forbes PWA loads in 2.5 seconds on mobile. For finance apps load times and lag are really important. If a market slips and a client misses it because of a poor load time they are going to trade with other people.

From this research it is clear to me that developing a PWA is the right way forward for the type of app that I want. However, it is important to bear in mind the disadvantages of PWA’s and how I can overcome them.

Payments are not integrated as well in PWA’s. Native apps are linked to an app store which is linked to credit or debit card which makes payments easy. PWA’s are not. However, since I won’t be using payments in my application this is not a concern for me.

Low penetration. Does the casual user know what a PWA is or what it does? To me that answer is no. People are much more familiar with native apps. They know where to look for them and they know how to use them. For PWA’s there is no store or central location on where to find them. Subsequently, users may not want to add something to their home screen that they have never seen before and don’t really trust.

No fully functional on iOS. PWA’s are very new and so functionality is not supported across all browsers just yet. One of those is Apple’s Safari browser that uses Webkit under the hood. However, there is work going on that will fix this.

While these are real disadvantages I do not think that they are enough to make the project unviable.

## Technical Analysis

There are a lot of technical indicators that traders can use and I am not going to be able to create them all. It was therefore important to research the most important indicators that traders needed so that I built those first:

* Back Testing. Back Testing is important because if done correctly it can allow traders to hone strategies. To find any glaring flaws in the strategy both technically and theoretically. Back testing implies that what happened in the past will happen in the future. If back testing proves the strategy works it will give traders the faith it will work in the future. (Kuepper, 2018)
* Moving Averages. These allow traders to determine, on average, where stock has been and where it might be going or what resistance levels that it has. Moving average are great because they easy to understand and are often used as a jumping in point to trading. (Mitchell, 2018)
* On-Balance Volume. This is the summary of volume of a stock based off a down day and an up day. It allows you to see longer term upward or downward trends. This is one indicator that looks at volume and not price. It is a good indicator because it takes something quite complicated and makes it much simpler.
* Relative Strength Index. This indicates when a stock has been overbought and is due for a fall. Or when a stock has been oversold and is due for a rise. Typically, you would trade when just as the RSI shows these things about to happen. This is a good indicator because its values are between 0 and 100 meaning you don’t get silly numbers and it is easy to read.
* Moving Average Convergence/Divergence. This indicator shows trends and momentum over time. This is shown by a line above or below zero. Above 0 indicates an up-trend and vice versa (Mitchell, 2018). Again, this is a fairly easy indicator to follow.

These indicators were the most popular that I found and this was confirmed by IG staff. Figure 5 shows an email exchange I had to get verify this.

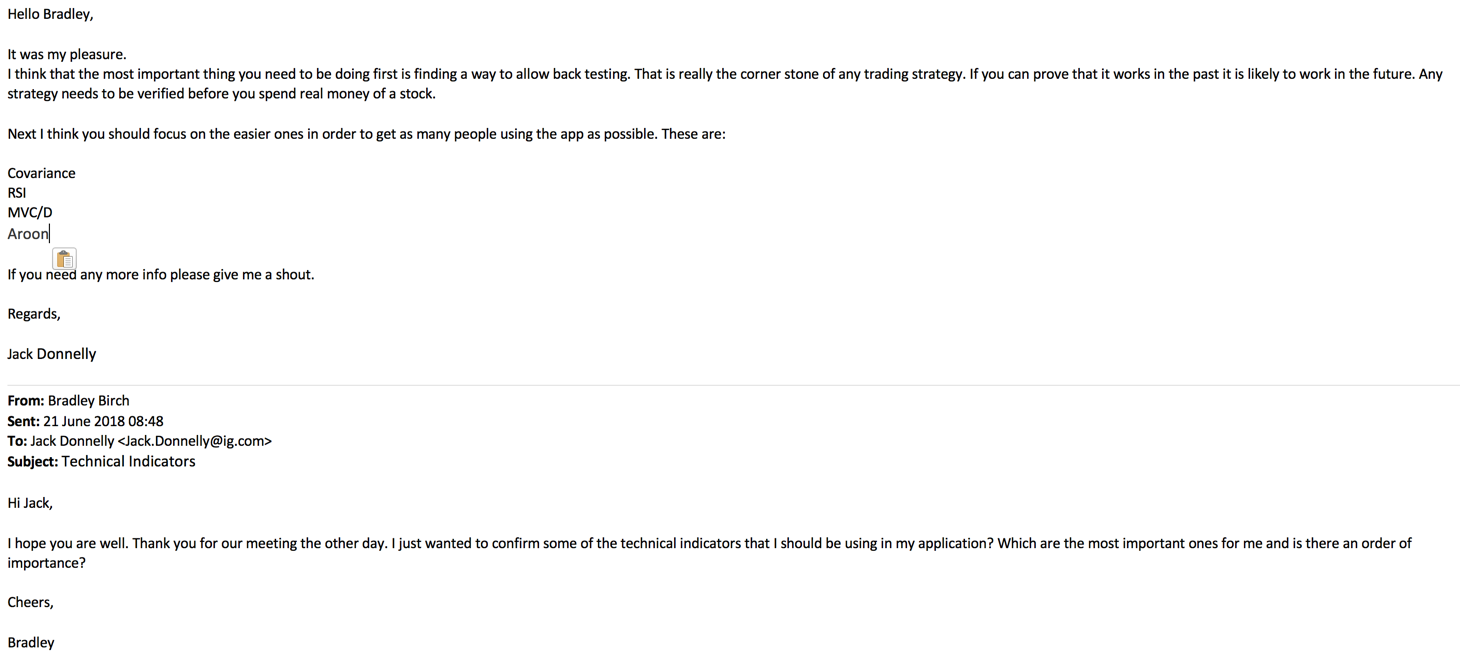


Figure 5: Research on Technical Indicators

# Methodology

Planning for the project involved a lot of different parties and we needed to make sure that everyone was on the same page.

## Planning

During the entirety of the project I decided to use the Agile planning methodology. Agile is an example of a project management process just like the waterfall methodology is. However, I have chosen agile for the way it can anticipate change and adapt much more quickly than other methodologies. Agile is an iterative development process. This means that a cycle of steps occurs many times. Unlike waterfall where it is a linear process. Figure 6 depicts the 6 steps that I took every iteration of the product. Every iteration takes two weeks to implement.

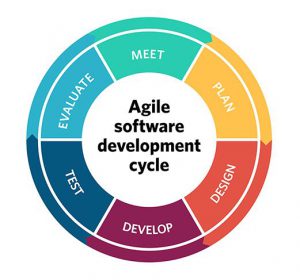


Figure 6: Agile steps (project management, 2017)

I chose agile for a number of reasons. Agile allows for a large amount of change (Segue Technologies, 2015). Going in to my project I was aware that I had high level goals to meet. Nevertheless, I did not know the small details of the project such as presentation or technical details. Agile allows for me to refine and adapt to changing requirements and feedback on the project. By having the opportunity to do this I can get feedback consistently and make sure that the app is the best it can be.

Agile adds a layer of transparency to the project. The business, developers and everyone involved in the project can get a look at where the project is going every two weeks. This can mean that timelines and budgets are more accurate and better reflect the progress of the project. If everyone on the team is on the same page then the team can work better together.

Finally, since agile is a cycle any changes that are made in the two weeks can be shown to users in order to get early feedback and comments. If there were certain features that users do not like then they can be removed or improved on. Agile forces the team to break down the project in to small chunks. Now each member of the team gets to see this small chunk of work being built and can drive its development. This means that every little thing about the app gets attention and can be perfected. Or if there is something wrong with it then it can be changed or removed. This makes the software the best it can be.

## Software Requirements

### JavaScript and Ember.js

The application that I will be building will be built for the web. Subsequently, I have chosen to write the app in JavaScript, the language of the web. JavaScript is the only language natively supported by the web and there is great support for it through every major browser. Writing the app in JavaScript also means I have a choice of JavaScript frameworks to choose from such as React, Ember or Vue.

For my application I chose to use the Ember framework. I picked Ember for a variety of reasons. First Ember includes two-way data binding. This is useful because it means that if the UI is updated the model, the applications state, will update as well. Similarly, if the model updates the UI will update with it. It means that the state of your view and the controller are the same. Thus, you know at all times the state of your application. Many frameworks have this two-way databinding, such as Angular 1. However, the implementations of the two-way data binding are a bit different. Angular uses a process of Dirty Checking (figure 7). This works by remembering the old and new values and comparing them in a snapshot known as the $digest Loop. When it does this check Angular will check the entire scope. If any values on the $scope have changed it will force every variable to be reassigned as well. Dirty checking will cause performance issues when you have a large amount of scope bindings (~2000) because dirty checking scans the entire scope for changes. While this is not a problem for small or medium apps it could be an issue for larger ones. Ember however, uses observers. Ember will only re-render changed values. Performance wise it’ll mean that in larger apps Ember will perform better.

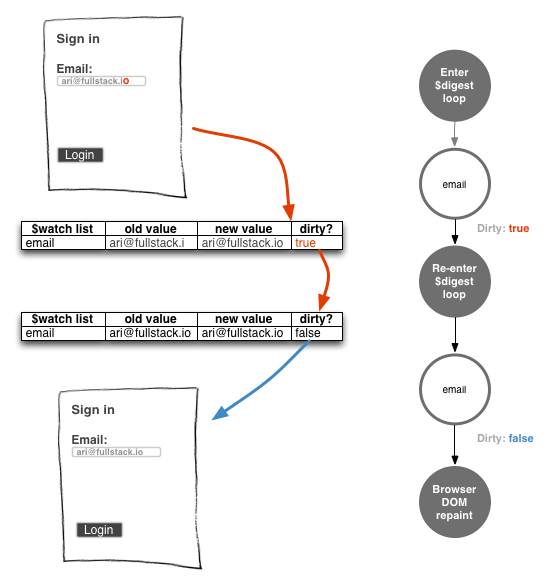


Figure 7: Dirty Checking in Angular - (Lerner, 2016)

Secondly, Ember has a fantastic command line client called Ember CLI. Ember is very strict in how a developer should set up and create their app. This can be good because it enforces a standard which will minimize bugs occurring. And without a standard, developers might be unsure on how files should be structured. When you are using the CLI to create files it will help you and create the basic file structure for you to give you a starting off point. This makes development easier as you do not have to always write boilerplate code in your files.

Next, I prefer the way Ember does its templating (figure 8). Ember uses Handlebar templates. Using an entire library might increase the file sizes of Ember projects. But developers, like me, might prefer handlebars. Handlebars, I think, are more elegant than other templating languages such as using directives in Vue or Angular (figure 9). I think it is easier to spot what is going on with Handlebars.



Figure 8: Handlebar Templates



Figure 9: Directives in Angular

Finally, I have been using Ember for 2 years now. It is a framework that I know and can use effectively. I like the way it is fully featured with built in data layer called Ember Data. This allows for simple manipulation of data where the developer can call simple methods like save and the heavy lifting of persisting the data to the server will be handled by Ember.

### CSS – Using a Pre-processor

CSS is used to style webpages and to make them look appealing to the end user. CSS is a declarative language that can quickly become repetitive and you can lose the structure of your styles. That is why in this project I will be using the SCSS pre-processor. A pre-processor is a scripting language that takes what a developer has written and compiles it down to raw CSS (Kavustuk, 2018).

I have chosen to use a pre-processor because they have built in support for variables. You only need to define a property value once, such as a colour, and you can use the variable name whenever you want to use the colour. Now if you want to change that colour it only has to be changed in one place. In regular CSS this has to be changed in multiple places.

Pre-processors have functions called mixins. Mixins are reusable blocks of styles that can accept arguments and will apply these styles to the element that has invoked the mixin. With Mixins you get to keep you code DRY (Do Not Repeat) and so you have smaller documents.

There are also a handful of smaller reasons I use a pre- processor:

* @extend – Allows you to take a CSS class and attach it to another. Which allows for you to write less code again.
* Loops – SCSS contains for, each and while loops to help pragmatically generate classes.

Overall, I think that using a pre-processor makes CSS much easier to maintain and it allows me to write much less code.

## Data Requirements

Since my app will be data driven I will need to use a reliable and complete data source. Therefore, I will be using IG’s Public API. IG has both a REST API and a streaming API with generous limits.

REST Limits:

* Per-app non-trading requests per minute: 60
* Per-account trading requests per minute: 100 (Applies to create/amend position or working order requests)
* Per-account non-trading requests per minute: 30
* Historical price data points per week: 10,000 (Applies to price history endpoints)

Streaming Limits:

* 40 concurrent subscriptions

These limits are more than enough for me to build a fully functional analytics app that will be able to handle different markets and streaming options.

You are also able to request historical data with the following restrictions.

|  |  |
| --- | --- |
| **Resolution** | **Days** |
| **1 Sec** | 4 |
| **1 Min** | 40 |
| **2 Min** | 40 |
| **3 Min** | 40 |
| **5 Min** | 360 |
| **10 Min** | 360 |
| **15 Min** | 360 |
| **30 Min** | 360 |
| **1 Hour** | 360 |
| **2 Hour** | 360 |
| **3 Hour** | 360 |
| **4 Hour** | 360 |
| **1 Day** | 15 years |

These resolutions are great for my application because I can create charts that go back 15 years if the user requires them to. Plus, the granularity of the historical data is perfectly defined to allow me to cater to different trading needs.

# Design

## Software Design

## Data Models

## Look and Feel of the App

# Implementation

## Code

## Testing

# Results and Discussion

## Quality of the App

## Quality of the Code

## Quality of the Development Process

# Conclusion

# Bibliography

Nath, D. (2017). PWA. [image] Available at: https://medium.com/@deepusnath/4-points-to-keep-in-mind-before-introducing-progressive-web-apps-pwa-to-your-team-8dc66bcf6011 [Accessed 14 Jun. 2018].

Google Developers. (2017). Progressive Web Apps | Web | Google Developers. [online] Available at: https://developers.google.com/web/progressive-web-apps/ [Accessed 14 Jun. 2018].

Enge, E. (2018). Mobile vs Desktop Usage in 2018: Mobile widens the gap. [online] Stone Temple. Available at: https://www.stonetemple.com/mobile-vs-desktop-usage-study/ [Accessed 22 Jun. 2018].

Crook, I. (2017). PWAs — Fad or Future? | leanplum. [online] Leanplum. Available at: https://www.leanplum.com/blog/pwa-progressive-web-app/ [Accessed 22 Jun. 2018].

Naylor, I. (2017). Progressive Web Apps vs Native Apps - Who Wins?. [online] AppInstitute. Available at: https://appinstitute.com/pwa-vs-native-apps/ [Accessed 22 Jun. 2018].

Kemp, S. (2017). Digital in 2017: Global Overview - We Are Social. [online] We Are Social. Available at: https://wearesocial.com/special-reports/digital-in-2017-global-overview [Accessed 22 Jun. 2018].

Reisinger, D. (2017). http://fortune.com. [online] Fortune. Available at: http://fortune.com/2017/03/06/apple-iphone-use-worldwide/ [Accessed 22 Jun. 2018].

SHILOVA, M. (2017). Progressive web apps: key benefits, statistics, use cases. [online] Apiumhub. Available at: https://apiumhub.com/tech-blog-barcelona/progressive-web-apps/ [Accessed 22 Jun. 2018].

Kuepper, J. (2018). Backtesting: Interpreting the Past. [online] Investopedia. Available at: https://www.investopedia.com/articles/trading/05/030205.asp [Accessed 22 Jun. 2018].

Mitchell, C. (2018). Trend Trading: The 4 Most Common Indicators. [online] Investopedia. Available at: https://www.investopedia.com/articles/active-trading/041814/four-most-commonlyused-indicators-trend-trading.asp [Accessed 22 Jun. 2018].

project management. (2017). [online] Available at: https://project-management.com/10-key-principles-of-agile-software-development/ [Accessed 6 Jul. 2018].

Segue Technologies. (2015). 8 Benefits of Agile Software Development | Segue Technologies. [online] Available at: https://www.seguetech.com/8-benefits-of-agile-software-development/ [Accessed 6 Jul. 2018].

Lerner, A. (2016). The Digest Loop and $apply |. [online] Ng-book.com. Available at: https://www.ng-book.com/p/The-Digest-Loop-and-apply/ [Accessed 13 Jul. 2018].

Kavustuk, K. (2018). Why we decided to drop LESS for SCSS | The JotForm Blog. [online] Jotform Blog. Available at: https://www.jotform.com/blog/dropping-less-for-scss/ [Accessed 13 Jul. 2018].

# Appendix