C M A CA

2018 COMPUTING COURSEWORK OCR

Comment to readers:

1 The coursework was done in parallel to the applications development (See Section 2.14 – Time Management). However, the content is purposefully written in the past tense relative to the Evaluation which is written in the present tense.

2 I have gone over each part of my coursework again at the evaluation stage and filled in relevant information that wouldn't have been known at the time. This is to make the project clearer to the reader and prevent contradicting information throughout the project.

1 CONTENTS

2 PLANNING

2.1 PREPARATION FOR INTERVIEW

A client contacted me with a potential idea for an application concerning cryptocurrencies. I said after a consultation and interview I would be able to evaluate whether the project was feasible and possible costs contained.

I initially prepared for the interview with a few questions:

- What would the product entail?
- What timeframe would be ideal?
- Whether they would mind me using it as a project
- Various questions about cost and payment which I will omit for this writeup

2.2 INITIAL INTERVIEW

[Start transcript]

[…]

Me

What do you imagine this product entailing?

Client

So basically, Crypto Exchanges have APIs.

I was wondering if it would be possible to create a desktop app that collates all of these into one manageable portfolio.

I cannot find a windows PC version of any manager out there

and certainly not one that imports using the APIs provided by the exchanges

Me

like information on the current exchange rate?

Client

yeah, and pulls the current amount of stock you hold in each coin

bittrex *[ref 1]* currently have one that I can use on an iOS app

Me

hmm, interesting - I mean it would need to integrate with wallets which would be more complex. Why not just use a website to look up the data?

Client

I have 5 different exchanges

about 10 coins on each,

keeping the value of each and the percentage profit is a nightmare

especially if I'm day trading

I just need a better way of keeping track

Me

Definitely sounds possible from the offset but give me some time to look at the APIs and similar products current available.

[…]

[End Transcript]

References:

1: Bittrex iOS app: https://itunes.apple.com/us/app/b-trex/id1258071406?mt=8

At the end of this interview I started to research available similar products (elaborated later in section 1.5 'Similar product research') additionally available APIs for cryptocurrency access (elaborated later in section 1.7.2 'APIs').

This interview extended to other points, which are mentioned later, for example in Section 2.14.1 'Time management transcript'. However, they are excluded in this section are not relevant currently.

2.3 CLIENT BRIEF

After the initial transcripts the client provided a brief outline of the product to further consolidate my idea of the projects requirements.

A desktop application which allows me to view my current portfolios and balance of bitcoins and various other cryptocurrencies. I would like it to automatically update with the current mean price of the bitcoin to other currencies. I would like it to be customisable, stylish and easy to use. Additionally, I want it integrated with as many different currency exchanges as possible to maximise its usage.

2.4 MINIMUM VIABLE PRODUCT (MVP)

Final draft of the MVP – the minimum requirements my product must fill for it to be considered 'complete'.

1) Desktop based application
    a) Able to be installed and run from an applications directory.
        i) The client is primarily concerned with windows and mac, however cross platform support is preferable going forward.
2) Ability to make a portfolio
    a) Should be intuitive
        i) Should have introduction on first load
    b) Ability to add a wallet/exchange/simple amount of coin
        i) Ability to remove wallet / change simple amount of initial coin
    c) Ability to watch coin gain / fall relative to the initial input
    d) Support for multiple exchanges
3) Ability to view a candle chart of the currency conversion data
    a) Ability to add a chart
    b) Ability to choose which exchange / currency it grabs from
    c) Support for multiple exchanges
4) Persistent storage of user data
    a) Saved to a file somewhere. Not necessarily human readable but reliable.
5) Security
    a) Basic Password on entry
        i) This password is not meant to securely protect the product – instead its main aim is to prevent anyone physically on the computer just being able to immediately see the data.
    b) This program is not meant to be secure by nature – all the data accessible via exchanges / wallet should be read only

Multiple Exchanges e.g.:

- Average
- Binance

- Bitflyer
- Bitfinex
- Bithumb
- Bitsamp
- Bittrex
- Coinnest
- Coinone
- Gdax
- Geminin
- Hitbtc
- Korbit
- Kraken
- Liqui
- Poloniex
- WEX

When I had reached a final draft of my MVP. I showed it to my client and asked if they wished to change any of it. My client was able to clarify point 4.a.i. about the non-necessity of security within this product due to its nature; among some other minor changes the finer points of the specification. Once all changes had been made I asked my client to agree that this specification outlined what a MVP would look like and began the project.

2.5 ADDITIONAL SPECIFICATION

The following is additional ideas suggested by the client and or myself which are not included in the MVP and would be acceptable to not achieve within my application but instead to strive for and possibly implement if time permits.

1) Lazy load of persistent storage
   a) The data from the persistent storage of user data would load 'lazily' allowing the user to interact with the app before the user data is loaded – and then when loaded it replaces the current state (to avoid conflicts)
2) Lazy load of cryptocurrency graphs
   a) The graphs should be loaded asynchronously to the main process so is non-blocking to the UI experience and when it has loaded the data it should add it to the UI.
3) Offline mode
   a) It can work in offline mode using the last data reached when online – and or indicators the program is unable to access the exchanges.
4) Multiple profile support
   a) It can support multiple profiles which in turn each have their own portfolios. This allows for better splitting of the portfolio data.

2.6 SIMILAR PRODUCT RESEARCH

In the aim of achieving my MVP. I looked at many similar products and how they achieved their goals and how they compared to the ones I had been given.

2.6.1 COIN TICKER IPHONE - HTTPS://ITUNES.APPLE.COM/GB/APP/COIN-TICKER-BITCOIN-ALTCOIN/ID636476147

Coin ticker for iPhone provides many of the features like my specification. It allows the adding of portfolios and connection to read only wallet data, so you can accurately track your worth in the currency you desire. It however is restrictive in its configuration. You can customise what cryptocurrencies you want though the format is list based and hard to analyse accurately. Especially as the graphs used have no scales and instead just notions of increases and decreases.

Figure 1 A graph taken from the app showing Poloneix [a cryptocurrency] data

I suspect this is a symptom of it being a mobile app it is hard to contain all this data in an easy to use screen. This is something that I can improve on through the fact that my application will be desktop based.

2.6.2 CRYPTOLIO - HTTPS://GITHUB.COM/LARION/CRYPTOLIO

This product is a terminal based crypto currency portfolio released under an MIT license as open source software by a Github user 'larion'.

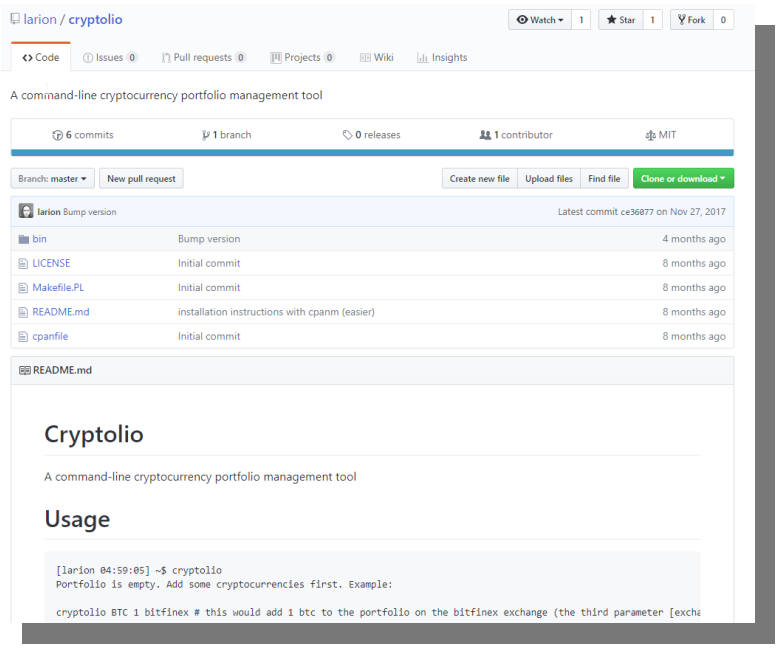


Figure 2 Screen capture of the webpage in which this application is available [CITATION Iar17 \l 2057]

It has many aspects like my MVP specification:

- It can access data from a variety of different exchanges
- Users are able to add their own crypto 'holdings' to it
- It can display this data in a meaningful way to the user

However, it lacks the interface that a GUI based editor or further user settings. That said, it is an important part of my research as it shows the source code behind it as open source software. Therefore, I can examine it and find which APIs it uses. In this case it uses: "https://api.coinmarketcap.com".

2.6.3 CRYPTOCOMPARE - HTTPS://WWW.CRYPTOCOMPARE.COM/

CryptoCompare provides leading of 85 exchanges and various cryptocurrencies, wallets and mining information. Through it you can add your own wallets and data to it and it is able to manage and show you information about your "portfolio". It therefore achieves many features of my specification. It has heavily detailed guides on cryptocurrencies and various areas surrounding it and extensive information on each exchange and cryptocurrency.

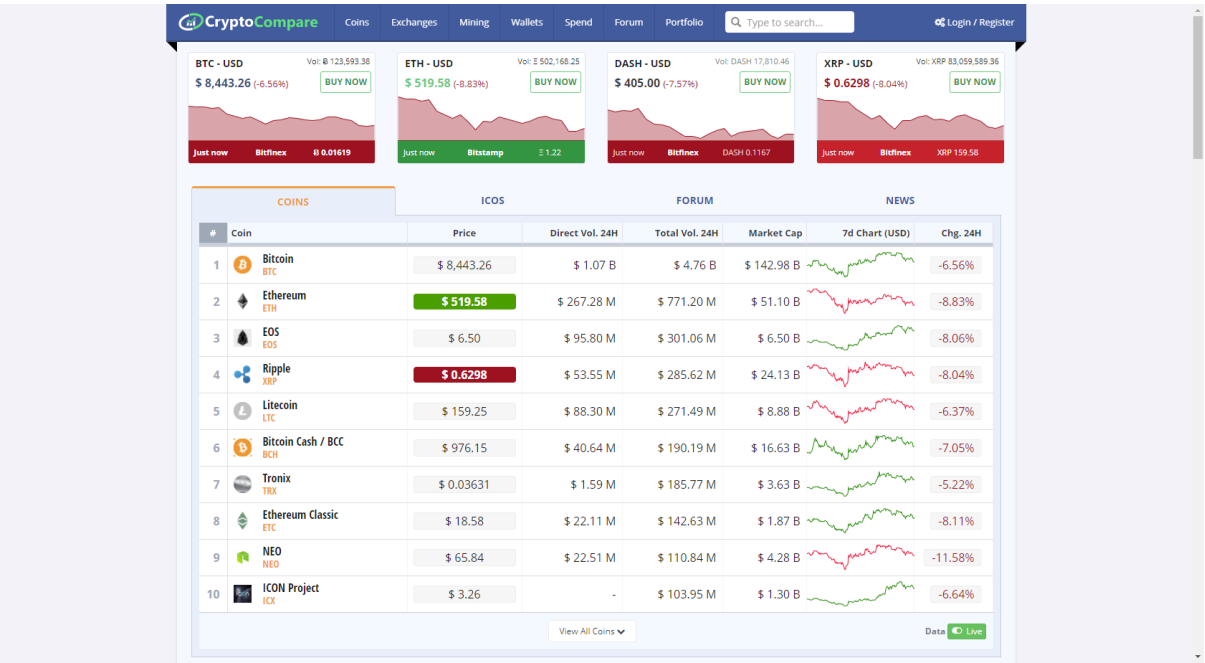


Figure 3 Screen capture of the main page (https://www.cryptocompare.com/)

However, it is a web-based application and would not work on desktop so even though it achieves many points of my MVP it doesn't achieve the central point. This adaption my MVP would require a significant update of the UI.

2.7 GENERAL DEVELOPMENT MODEL

Throughout the development of this application I for a spiral model of development. This allows me to create a very detailed plan to show the work necessary to the coursework requirements and additionally being able to develop the best application possible during the short development window. It also allows me to evaluate my applications performance at the end of the development change.

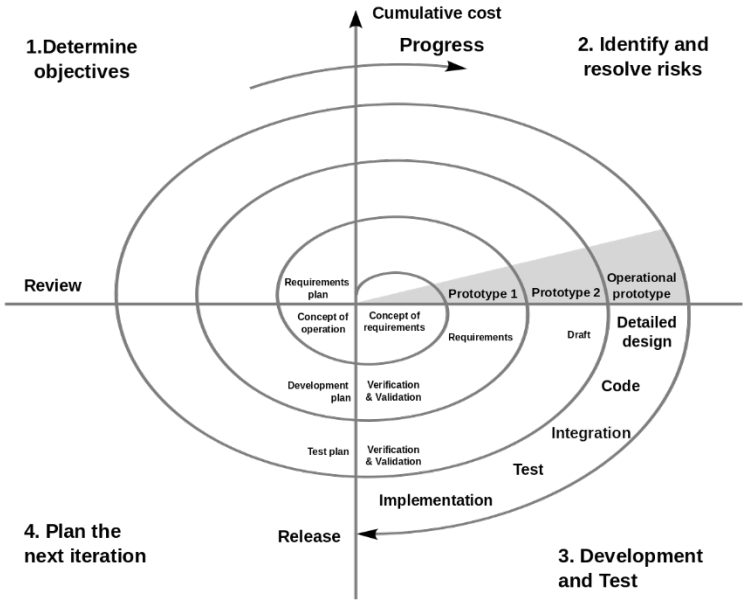


Figure 4 Spiral model development [CITATION Boe04 \l 2057]

2.8 TECHNOLOGIES NEEDED

2.8.1 LANGUAGE CHOICE

There are many languages available that would adequately fit the requirements of the project and or client. Languages such as C# are well known for being able to cope with desktop GUIs very well and are used for a variety of large projects [CITATION Git18 \l 2057]. Java additionally is well known especially with its JavaFX framework. There is additionally a relative newcomer to desktop UI design called ElectronJS [CITATION Ele17 \l 2057]. I have discounted a web-based product purely because the MVP specification the client gives wants a **desktop-based** client. To decide which one was most applicable to this application I compared the pros and cons of each:

2.8.1.1 C# / WPF - HTTPS://DOCS.MICROSOFT.COM/EN-US/DOTNET/FRAMEWORK/WPF/GETTING-STARTED/INTRODUCTION-TO-WPF-IN-VS

This framework is a Windows centric (though cross platform) way of providing enterprise level desktop applications.

2.8.1.1.1 ADVANTAGES

+ Well supported/Much documentation
+ Very well used

2.8.1.1.2 DISADVANTAGES

– Higher learning overhead
– Closed Source
– Restrictive design / structure

2.8.1.2 JAVA / JAVAFX

This is a cross platform approach of providing desktop applications using their prescriptive xml based markup language.

2.8.1.2.1 ADVANTAGES

+ Well-structured language made to fit OOP

2.8.1.2.2 DISADVANTAGES

– Learning overhead with the xml language
– Harder to make look native (cannot naturally embed native UI elements - *easily*)
– Closed Source
– Notoriously bad editor for the UI (however improved recently)
– Java has long compile times which make rapid development harder even with on the run class swapping

2.8.1.3 ELECTRON - HTTPS://ELECTRONJS.ORG/ - [CITATION ELE18 \L 2057]

This framework centres around being completely cross platform and just providing in effect a chromium browser window available to render any modern HTML/CSS/JavaScript. [CITATION Ele17 \l 2057]

2.8.1.3.1 ADVANTAGES

+ Very easy to setup
+ Cross platform
+ Can still access lower level OS features
+ Familiar technologies
+ Open Source (MIT License - [CITATION Git17 \l 2057])

2.8.1.3.2 DISADVANTAGES

– Has large RAM overhead [CITATION Var16 \l 2057]
– Larger file size [CITATION Var16 \l 2057]
– Harder to make look native (cannot naturally embed native UI elements, *easily*)

2.8.1.4 CONCLUSION

In the end I believe ElectronJS is the best choice to be able to build the application the client needs. This is due to its low learning overhead and easy cross-platform compatibility. This will be important as a low learning overhead ensures the best code can be written quickly and efficiently. Additionally, in an age with faster and faster computers, the so-called 'bloat' we get from embedding effectively a chrome browser within our application is mitigated. This is especially true as our application's most intensive task with undoubtedly fetching data from an API – which is unlikely to slow down the whole computer.

2.8.2 APIS

Researching the APIs, I wish to use to get each bit of data such as currency rates/cryptocurrency exchange rates etc. Here's some I have found during the planning stage:

- http://fixer.io/
- https://github.com/ccxt/ccxt

2.8.3 BOILERPLATE COMPARISON

When creating desktop applications with electron there can be a lot of setup such as setting up the electron build process, Hot-module-reloading for fast development and other components. Additionally, it is helpful to use a MVC framework such as ReactJS or Angular to improve development time and prevent bulk in the html

codebase. This in turn presents a problem of managing state in large programs which is generally done through libraries like redux (or MobX) which have direct bindings into Angular or React i.e. react-redux [CITATION rea18 \l 2057].

One well known resource for electron boilerplates is the "awesome-electron" repository which lists tools that use electron, tools for electron, as well as boilerplates: https://github.com/sindresorhus/awesome-electron#boilerplates

It shows a few such as electron-vue, electron-react-boilerplate and others. Though vue and angular both have their own unique boiler plates I am most familiar with ReactJS so I opted for the electron-react-boilerplate (https://github.com/chentsulin/electron-react-boilerplate). It comes with many advantages such as hot module reloading (allowing modules to be swapped out during development). Additionally, FlowJS to prevent static type errors, it also has a built-in electron packager to easily produce my app as an installing item.

2.8.4 NOTE ABOUT FLOWJS

FlowJS is a static type checker for JavaScript built by Facebook [CITATION Fac18 \l 2057]. It allows me to augment my JavaScript code with type blocks as shown:

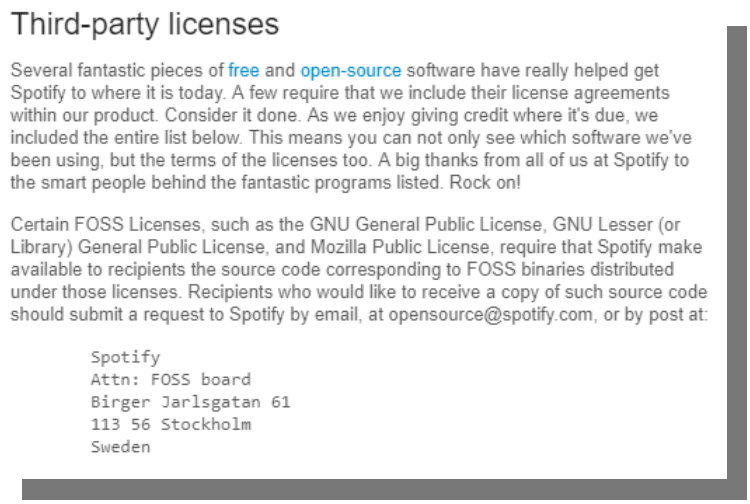
Figure 5 An example of a flow type block

This code defines a type 'actionType' as an object literal which takes two read only keys: 'type' which has a string value and 'payload' which can have any value and is optional.

What should be noted about the introduction of flow into my code is that it is still valid JavaScript code when the flow types are removed. In all senses and purpose, they can just be treated as additional comments to the code. Though to make it extra clear what the actual JavaScript looks like I have automatically generated a `_no-flow-src` folder in my final application's code. This contains all the same files, however all JavaScript files have had the flow notation removed, so just the runnable JavaScript is left.

2.8.5 NOTE ABOUT LICENSES

Throughout my project I will make use of various open source software (OSS). This is commonplace within enterprise software; for example, here is a section on Third party software within the Spotify desktop application (a well-known music streaming software).



Third-party licenses

Several fantastic pieces of free and open-source software have really helped get Spotify to where it is today. A few require that we include their license agreements within our product. Consider it done. As we enjoy giving credit where it's due, we included the entire list below. This means you can not only see which software we've been using, but the terms of the licenses too. A big thanks from all of us at Spotify to the smart people behind the fantastic programs listed. Rock on!

Certain FOSS Licenses, such as the GNU General Public License, GNU Lesser (or Library) General Public License, and Mozilla Public License, require that Spotify make available to recipients the source code corresponding to FOSS binaries distributed under those licenses. Recipients who would like to receive a copy of such source code should submit a request to Spotify by email, at opensource@spotify.com, or by post at:

Spotify
Attn: FOSS board
Birger Jarlsgatan 61
113 56 Stockholm
Sweden

Figure 6 An excerpt from Spotify's desktop application about Third Party Software

However, care must still be taking concerning licenses. Most open source software imposes conditions, though normally light.

For example, one of the most popular licenses: MIT [CITATION Git17 \l 2057] requires the license and copyright notice to be distributed with it in any software. To correspond with these conditions, when building my project, I installed a package called `electron-license` and included it in my build process as so.

```
"generate-licenses": "electron-license build > release/LICENSE",
```

Figure 7 Part of my final package.json – this script is run on building of the project. It compiles all the licenses within all the projects I use and then puts it all in one file – the `LICENSE` file within the release folder.

This then generated a nice license file listing each of the OSS licenses/projects used in the release folder of the project:

```
# Licenses

This project is covered by standard copyright laws.

Below is a list of OSS software used within this product:


This product also includes the following libraries which are covered by the (GPL-2.0 OR MIT) license:

- ua-parser-js

This product also includes the following libraries which are covered by the (OFL-1.1 AND MIT) license:

- font-awesome

This product also includes the following libraries which are covered by the (WTFPL OR MIT) license:

- opener
- path-is-inside

This product also includes the following libraries which are covered by the AFLv2.1 license:

- json-schema

This product also includes the following libraries which are covered by the Apache license:
```

Figure 8 An excerpt from the LICENSE file generated - in total it is more than 1500 lines!

2.8.6 DATA VISUALIZATION FRAMEWORK

In my application my client has requested various data visualizations. These include candle-stick charts [CITATION Wik18 \l 2057]. To visualize these properly in my application without spending a needless amount of time generating my own visualization framework I needed to conclude which framework was best for my use case. It came down to two options in the end.

2.8.6.1 D3.JS - HTTPS://D3JS.ORG/

Advantages:

+ Well supported (by open source community)
+ Many built in graph types
+ Easy to add remove data while the program is running

Disadvantages:

– Large learning curve
– Exposes SVG APIs (would be good but I'm not as familiar with them, therefore again serve to add to the large learning curve).
– Doesn't come with easy export features of the graph

2.8.6.2 PLOT.LY - HTTPS://PLOT.LY/

Advantages:

+ Many built in graph types
+ Well supported (by Plotly Ltd and open source community [CITATION Plo18 \l 2057])
+ Easy built in zoom features/export etc.
+ Almost no learning curve

Disadvantages:

– Less customisability
– Harder to style

2.8.6.3 CONCLUSION

I decided to go with Plot.ly based on a simple advantage/disadvantage analysis. The built-in features and almost no learning curve are very important due to the time constraints of the project, and well worth a sacrifice in customisability.

2.8.7 TESTING FRAMEWORK

To allow me to get real time indications of the products parity with the original specification I had to introduce an automated testing framework into my project. The choice of it was made easy by the boilerplate I had chosen (see 1.8.3).

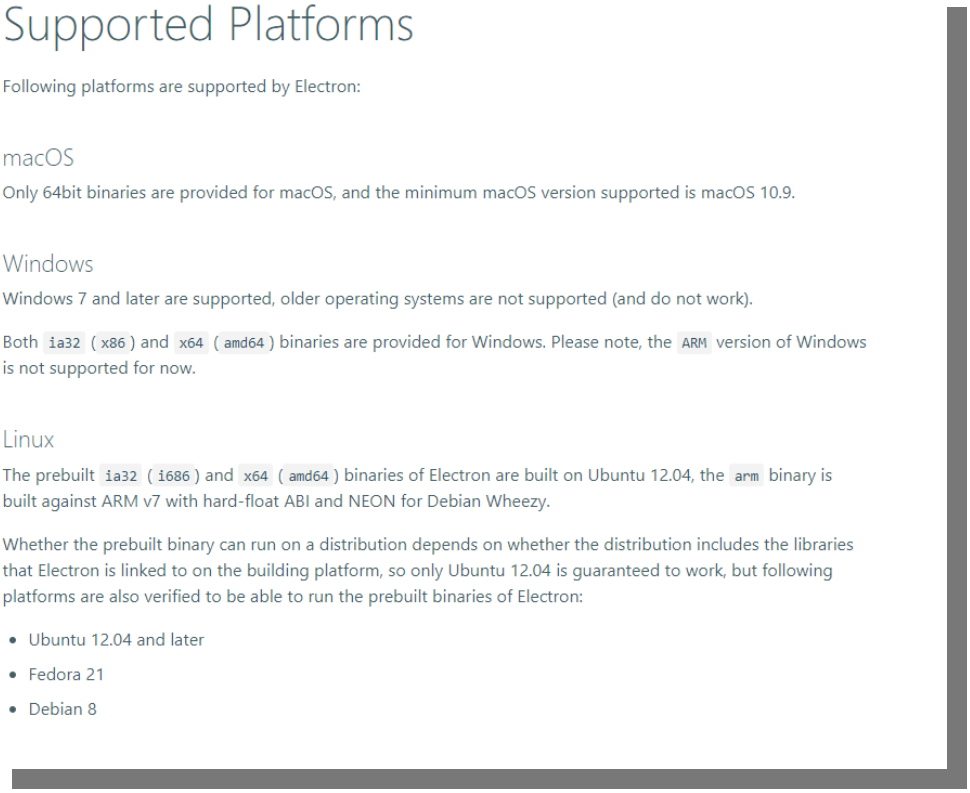
The testing framework I chose was Jest - https://github.com/facebook/jest (with additions such as Enzyme for React testing - https://github.com/airbnb/enzyme).

2.8.8 HARDWARE AND SOFTWARE REQUIREMENTS

The hardware and software requirements are important to analyse especially relative to the client's requirements. From private consultation with the client they have stated how they are using a relative modern

computer with Windows 10. Many those investing in new cryptocurrencies are likely to have more modern computers.

The base requirements for electron are as below:



Supported Platforms

Following platforms are supported by Electron:

macOS

Only 64bit binaries are provided for macOS, and the minimum macOS version supported is macOS 10.9.

Windows

Windows 7 and later are supported, older operating systems are not supported (and do not work).

Both `ia32` (`x86`) and `x64` (`amd64`) binaries are provided for Windows. Please note, the `ARM` version of Windows is not supported for now.

Linux

The prebuilt `ia32` (`i686`) and `x64` (`amd64`) binaries of Electron are built on Ubuntu 12.04, the `arm` binary is built against ARM v7 with hard-float ABI and NEON for Debian Wheezy.

Whether the prebuilt binary can run on a distribution depends on whether the distribution includes the libraries that Electron is linked to on the building platform, so only Ubuntu 12.04 is guaranteed to work, but following platforms are also verified to be able to run the prebuilt binaries of Electron:

- Ubuntu 12.04 and later
- Fedora 21
- Debian 8

Figure 9 Supported systems [CITATION Ele18 \l 2057]

My application would not require any special additional requirements on top of ElectronJS's ones except for possibly an internet connection to fetch the data. However, it would be able to run without it and would have graceful degradation of content [CITATION W3C15 \l 2057] as needed by the MVP.

2.8.9 CONCLUSION

The technologies chosen allowed me to solve my MVP. Each one is tailored to either speed up the development velocity or provide a feature towards the MVP.

However, certain problems in my MVP will still be hard to solve. For example, the persistent data storage I still didn't think had a good solution with the current technologies, so I would need to find an alternative solution to use.

2.9 BASIC LAYOUT DESIGN ✓

I designed a basic overview of what I wanted the app to look like which is shown below.

Figure 10 – A basic design of what the application might look like

Colours used for mock-up:

Area	Colour (#Hex)
Left side bar background	#1C1745
Up arrow left sidebar forecolour	#4ABF40
Down arrow left sidebar forecolour	#BF4240
Padlock left side colour forecolour	#FFE37F
Text colour left sidebar	#D7CDF2
Background colour main area blocks	#D7D7DB

This design is heavily subject to change as the app is pushed through development.

Additionally, I modelled an icon for the application based on the Wikimedia cryptocurrency logo as shown below:

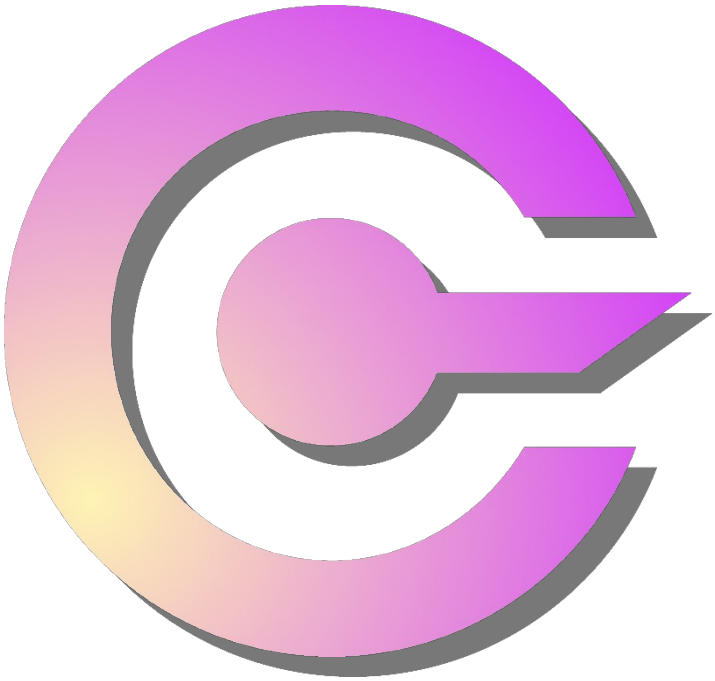


Figure 11 Retouched cryptocurrency logo / New Application logo

Colour Specification for logo:

Area	Colour (#Hex)
Top right side gradient stop	#FF52E5
Bottom left side gradient stop	#F6D242

2.10 DESIGN CONSULTATION WITH CLIENT (TRANSCRIPT)

[Start transcript]

[…]

Me

Please find attached the mock-up design for the final application.

Client

That looks brilliant! I really like the styling, though am not too sure about the graphs on the left side.

Me

No I agree, I wasn't sure if it actually added any nuanced information that wasn't already there.

Client

Also they kind of break the flow of the left hand navigation

Me

I'll make a note of the change from the mock-up.

[…]

[End Transcript]

2.11 TESTS NEEDED FOR MVP ✕ ✕ ✕ ✕

Test ID	Test name	Test Description	MVP Spec
1.	Basic Load	The application loads up	1
2.	A UI Exists	The UI is present in the rendered application	1
3.	Installation (windows)	The UI can be installed to an applications directory (windows)	1.a.i
4.	Installation (mac)	The UI can be installed to an applications directory (mac)	1.a.i
5.	On first load displays the portfolio creator	Displays portfolio	2.a.i
6.	Portfolio Creator: Add Base Coin	Allows the user to add a base coin in the portfolio adder	2.a.i
7.	Portfolio Creator: Add Wallet	The user can interact with the portfolio creator (probably through a click) to add a wallet id.	2.a.i
8.	Portfolio Creator: Add Exchange	The user can interact with the portfolio creator (through clicking) to add an exchange id.	2.a.i
9.	Portfolio Creator: Has data from multiple exchanges	It has multiple exchanges loaded in	2.a.i
10.	Main Graph Adder: Exchanges loaded	A list of exchanges is loaded into the main graph adder and displayed to the user (via a select menu etc.)	
11.	Main Graph Adder: User can select and exchange	The user can select an exchange using a drop-down box from the main graph adder	
12.	Main Graph Adder: Symbols loaded	A list of symbols is loaded into the main graph adder and displayed to user (via select menu etc.)	
13.	Main Graph Adder: Form has to be filled in		
E.1.	Lazy load of configs		E.1.a
E.2.	Lazy load data from URLs		E.2.a

Tests beginning with ‘E’ are tests which are part of the additional specification (see Section 2.5 ‘Additional Specification’)

2.12 NAME CHOICE ✓

I discussed with the client what the application should be named, and they were content to leave the name up to me to decide: Choosing a name may seem like a trivial task for anyone. However, it could be argued that the name has an impact on the clients view on the final product. [CITATION Mol16 \l 2057]

Considered names need to reflect the nature of the application which are:

- Modern
- Sleek
- Easy to use
- Secure
- Safe
- Made for Cryptocurrency
- Made to create a Portfolio

2.12.1 CONSIDERATIONS AND COMMENTS

I considered many names though whittled them down to:

- Cryptolio
  - Portmantuas are cliché and non-modern but effective
  - Has a name clash with https://github.com/larion/cryptolio
- Crypto Buddy
  - Overly friendly, doesn't seem secure?
  - Has a name clash with http://www.mycryptobuddy.com/
- BitPortfolio
  - Implies only for bitcoin – or best serves bitcoin.

In the end I decided Cryptolio sounded the best however it had a name clash with a terminal based crypto currency portfolio. So, I decided to change to **Cryptolium**. Which became my final application name.

2.13 PROBLEM SPLITTING/PROJECT DIAGRAM

2.13.1 UI FLOW

It was essential to decide the programs flow before any UI creation was made. Therefore, I made a basic UI flowchart showing the flow of the user interface.

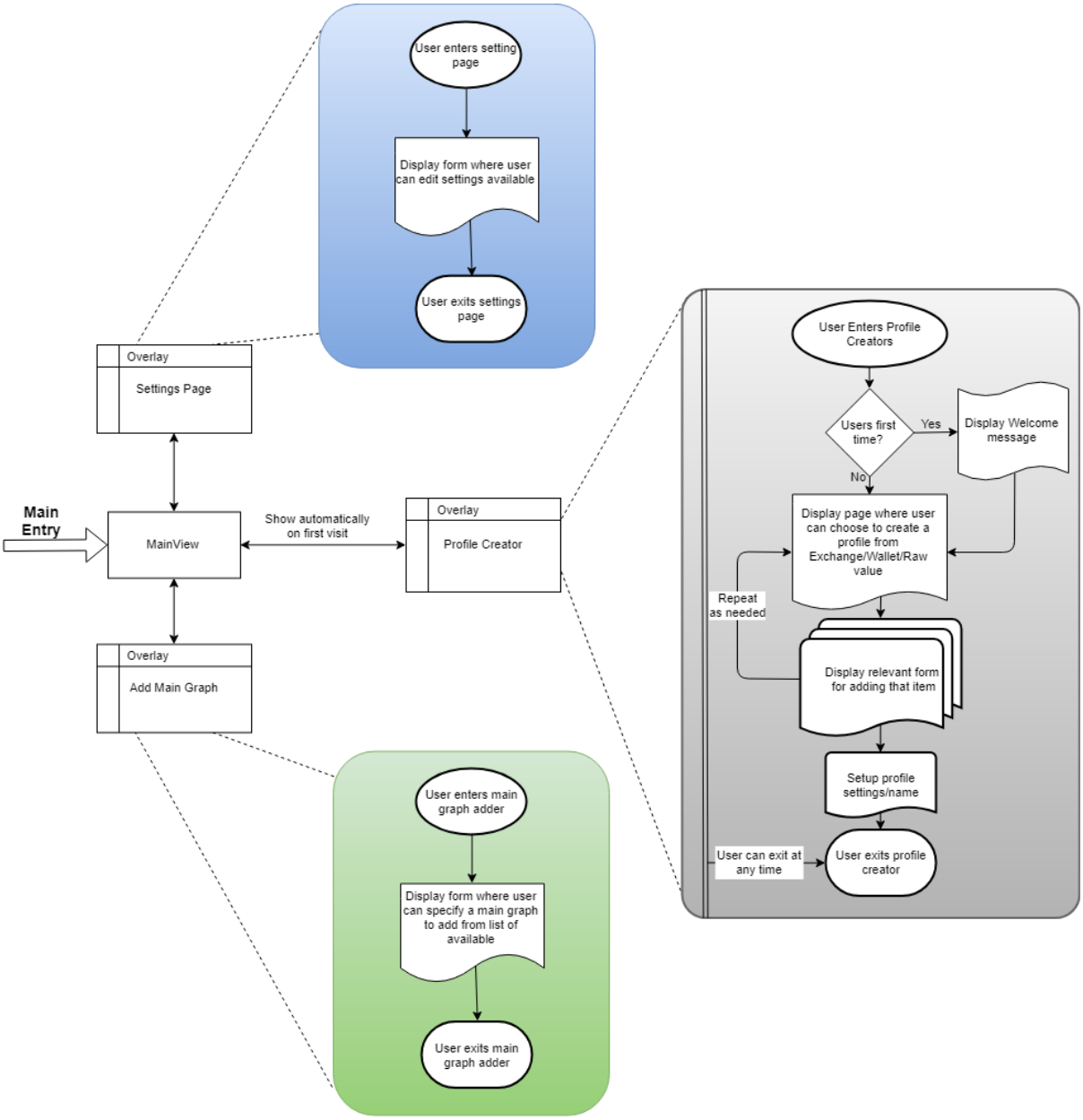


Figure 12 Flowchart showing the basic UI flow for the application

This flow chart shows the intended flow of the UI. The MainView is intended to contain the main graph data / portfolio display. Through triggering via a button or link the overlays, “Add Main Graph,” and “Profile Creator”. The Add Main Graph overlay will allow the user to add a graph to the main view screen. They will get a selection of options on a simple form.

2.13.2 TOTAL PROJECT DIAGRAM

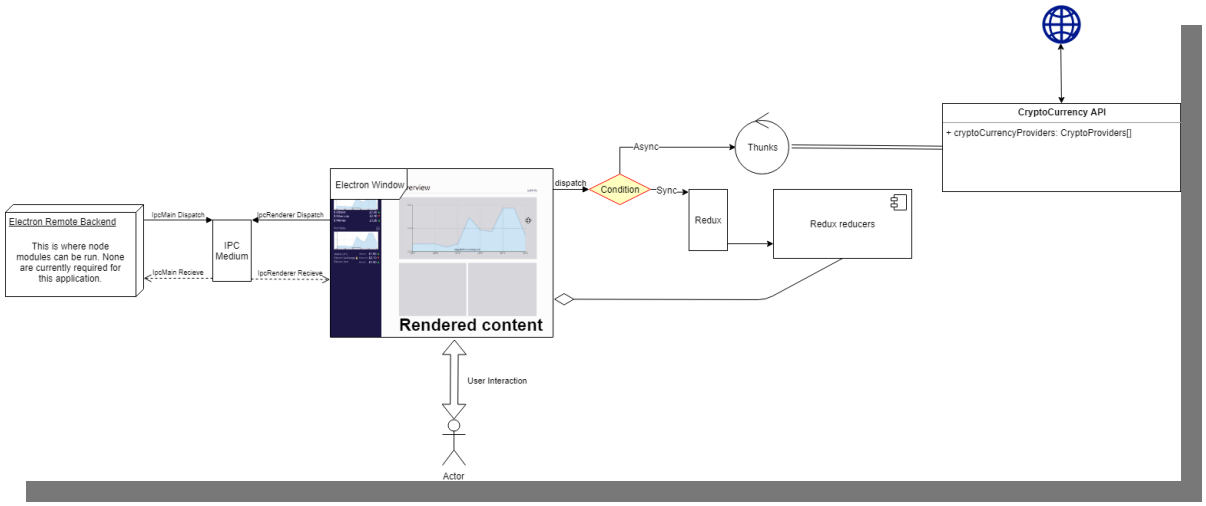


Figure 13 Complete project diagram

2.14 TIME MANAGEMENT

2.14.1 TIME MANAGEMENT TIMESCRIPT WITH CLIENT

[Start transcript]

[…]

Me

What timescale are we talking for the main application?

Client

Ideally running till at latest the end of March to have all the application done and ready for testing.

[…]

[End Transcript]

2.14.2 GANTT DIAGRAM

Following this conversation I was able to make a Gantt chart of the products development.

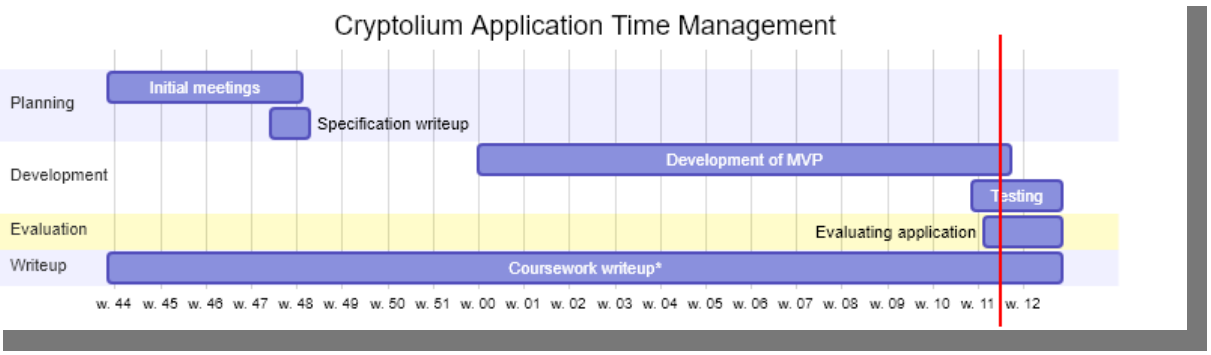


Figure 14 Time management of the application's entire process

Each part should come as little surprise. However, it should be noted [*] that the Coursework writeup is done in parallel to the rest of the project as it makes the coursework more accurate as any issues encountered can be documented as-is.

3 DEVELOPMENT

3.1 TESTING ENVIRONMENT

My automated test environment was set up via Jest and Enzyme (see section 2.8.7 “Testing framework”) modelled after the boilerplate I started with. The tests I plan on running can be divided into two types **automated** and **manual**. Automated tests will make use of Jest and will be ran programmatically on the application build. Most of them will be unit tests – black box testing a certain functionality of a file or process. A few will be end-2-end tests. These will be tests in which the entire application is simulated and interacted with automatically.

Manual tests will all be end-2-end tests in nature as I will be interacting with the finished application.

3.2 SETUP

Directory layout:

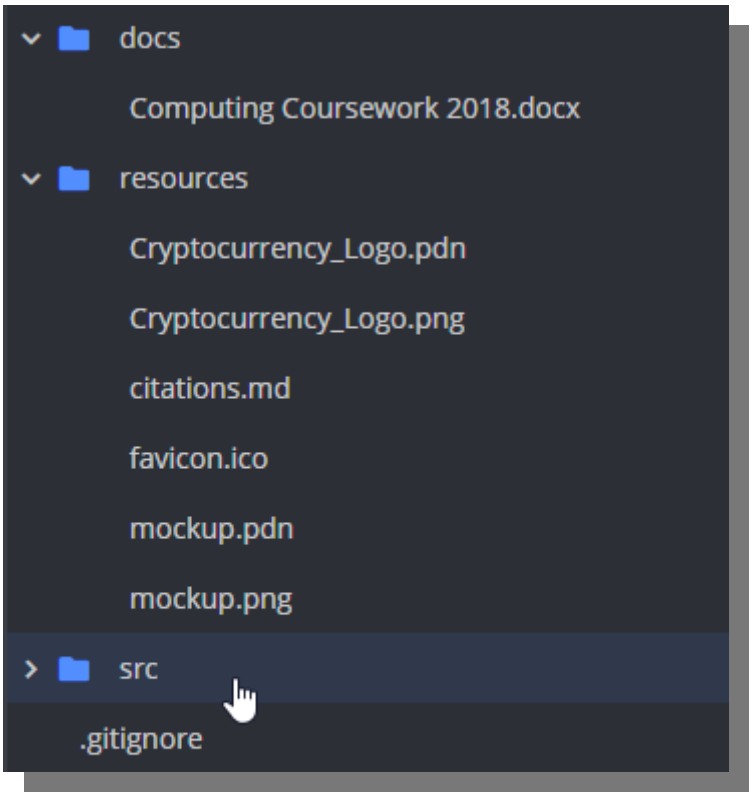


Figure 15 My basic directory layout

@todo src directory

3.2.1 SVN

Early in the project, I introduced a versioning system to better track the progress of the applications development. I created a private (eventually public) GitHub repository to hold the project:

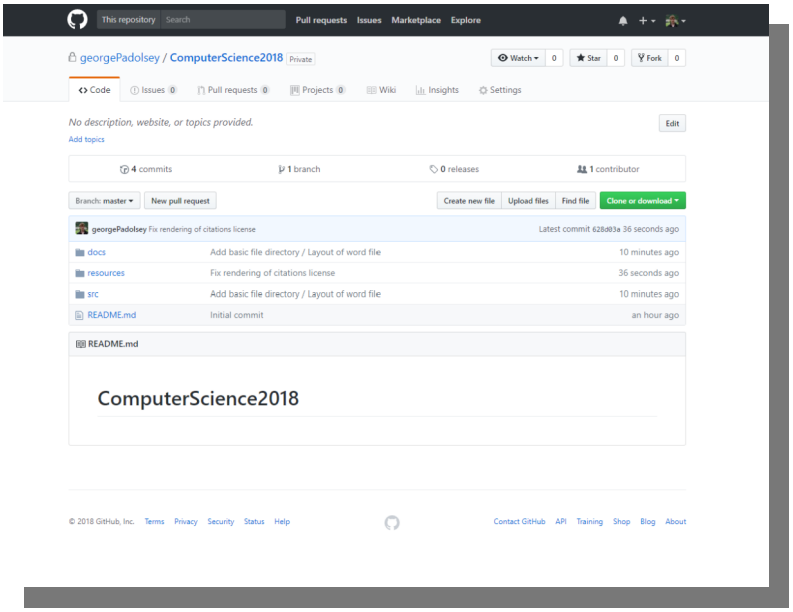


Figure 16 GitHub repository for the application

This also required me to set up a git client on my computer to upload (commit) to the repository. I chose GitKraken due to my familiarity with it:

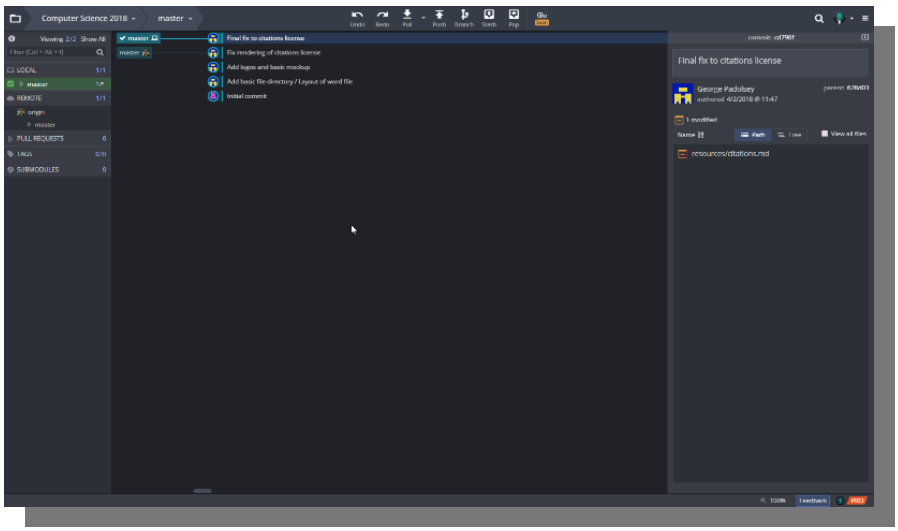


Figure 17 Setting up GitKraken as a version manager

While making the repository I had to setup various metadata files such as a .gitignore file. This file controls which files are committed to the online repository and which are not. For example, we would not want temporary files or library files to be committed to the online repository.

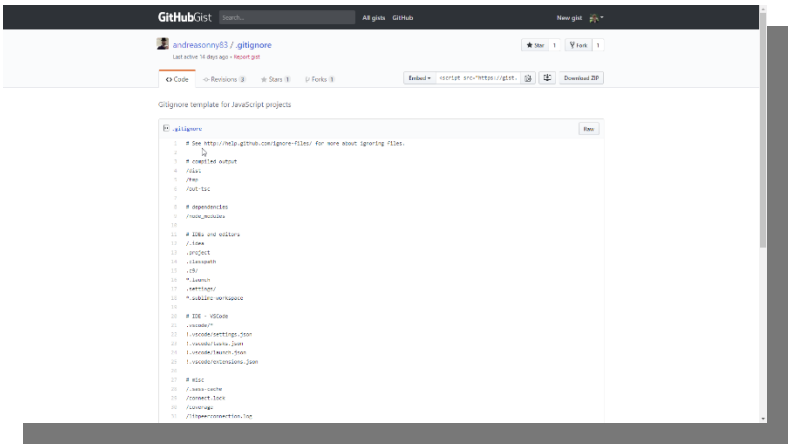


Figure 18 An example .gitignore[1]

3.2.2 GITHUB PROJECT BOARD

It is important to be able to easily see the progress you are making through the development of an app to better inform the client of your deadlines and for the developer to easy see what work needs to be done. To make this easier I employed GitHub recently added project boards which allow me to add 'notes' which I can then mark as in 'To do', 'In progress' or 'Done' depending on their progress which is reflected easily on a nice progress bar.

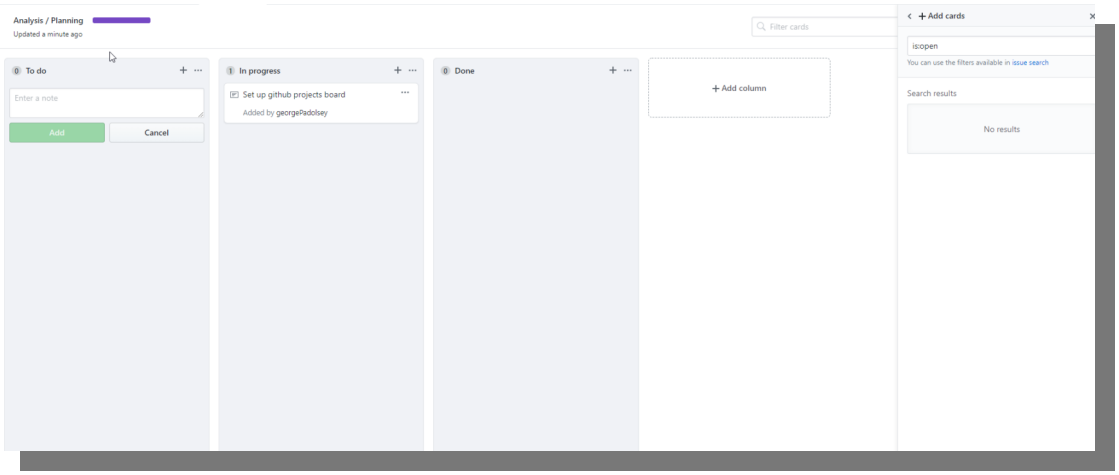


Figure 19 My Github project board for the planning part of the project

3.2.3 BOILERPLATE

I realised I made an error by making the .gitignore before cloning my boilerplate into the repository. When I tried to clone the boilerplate into the folder, it caused an error saying the directory had items in. The resolution to this problem was just deleting the .gitignore file I had made.

```
georg@MegaBiscuitv2 MINGW64 ~/src/Computer Science 2018/src (master)
$ git clone --depth=1 https://github.com/chentsulin/electron-react-boilerplate.git .
```

Figure 20 My original attempt at cloning the repository

[1] https://gist.github.com/andreasonny83/b24e38b7772a3ea362d8e8d238d5a7bc

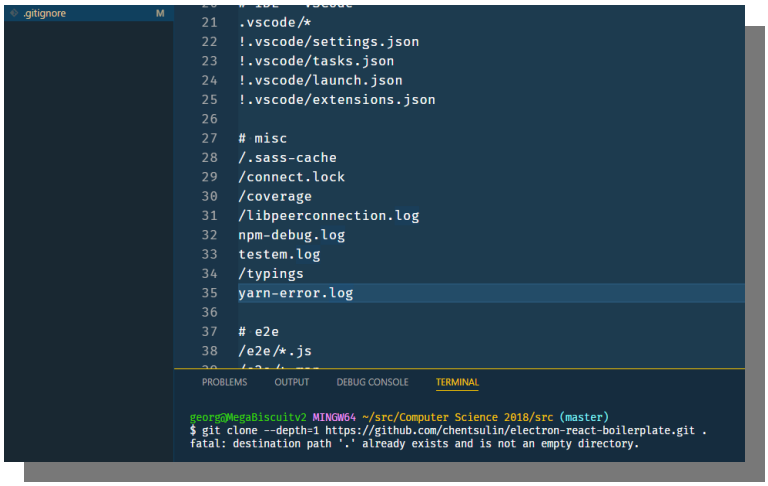


Figure 21 The .gitignore file

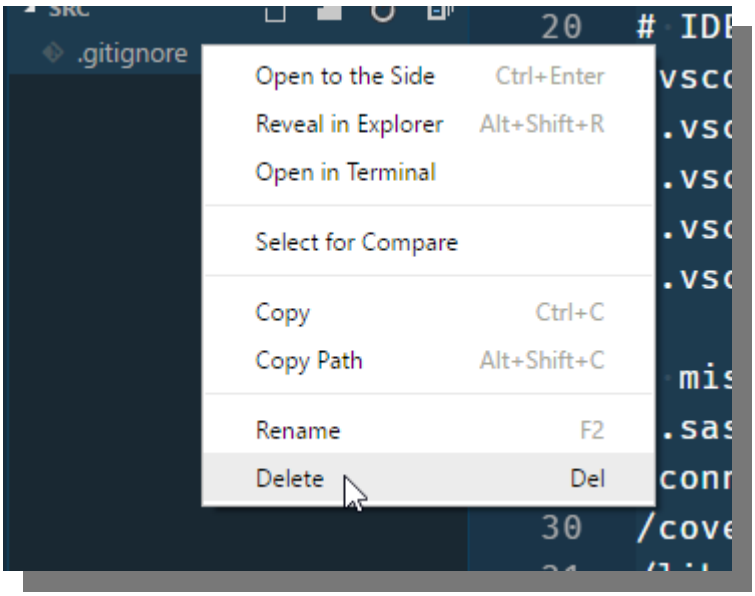


Figure 22 The .gitignore file being deleted.

Finally, I had a fully cloned boilerplate:

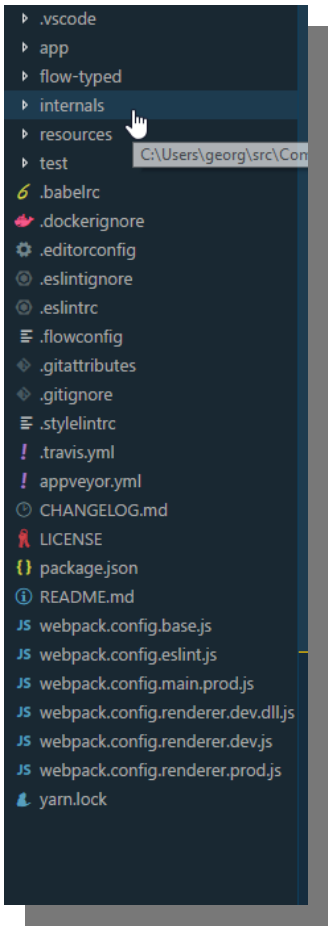


Figure 23 Fully cloned boilerplate

3.2.4 TRAVIS CI

I decided it might be worth setting up continuous integration that would continuously build and test my application after every commit. I was lucky as the boilerplate library had a prebuilt .travis.yml configuration for Travis CI, a CI I had a private plan for allowing me to use it with the repository.

This provided some clear advantages:

- Automated test results without having to use processing power on own computer
- It keeps a log of which commits (changes in code) caused the tests to fail. So, it is easier to track down an error that was introduced at some point.

Therefore, I decided it was worth implementing.

3.2.4.1 IMPLEMENTATION

Unfortunately, when I tried setting it up I got this error:

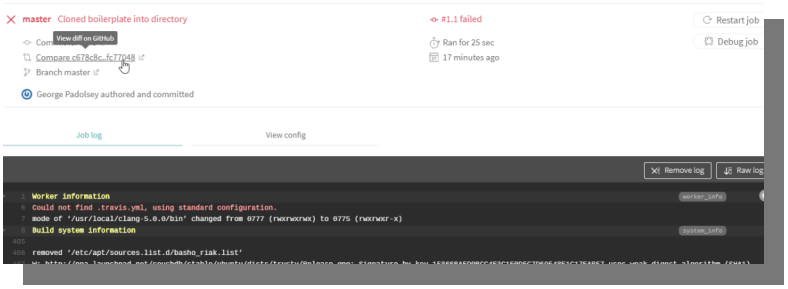


Figure 24 Travis CI error

I quickly identified based on the error message that this was because the .travis.yml was in the src/ folder with the rest of the boilerplate. I moved the .travis.yml to the root directory of the repository and rewrote the scripts within to change directory to the /src directory where the rest of the code is.

```
32  before_script:
33    - cd src/
```

Figure 25 Part of the rewritten .travis.yml

3.2.5 SECURITY CHECKLIST ✓

In preparation for making the application I read up on how to ensure the electron application is made secure. A well-known document[2] on this topic was released by Doyensec, an independent security agency.

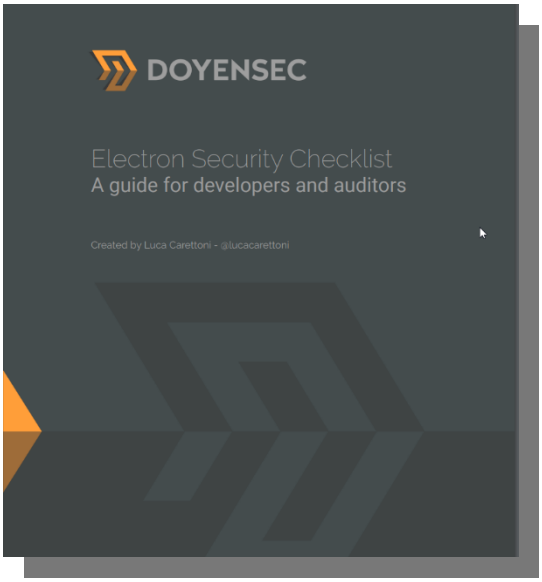


Figure 26 First page of the security checklist

I implemented each of the changes relevant to my application:

[2] https://www.blackhat.com/docs/us-17/thursday/us-17-Carettoni-Electronegativity-A-Study-Of-Electron-Security-wp.pdf

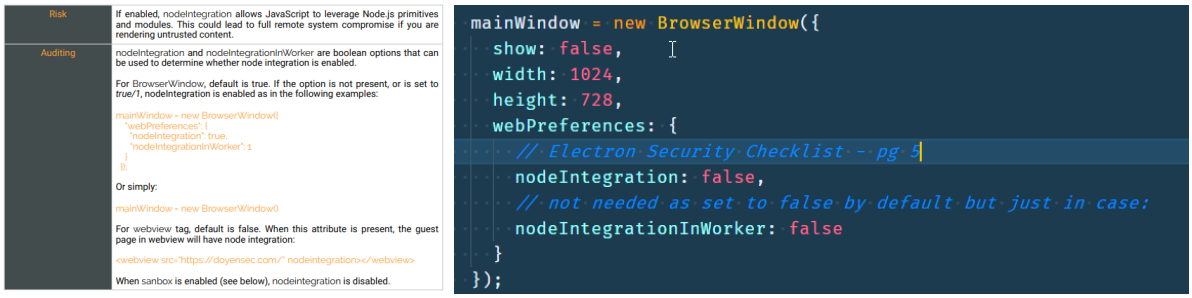


Figure 27 Documentation vs implementation of the checklist

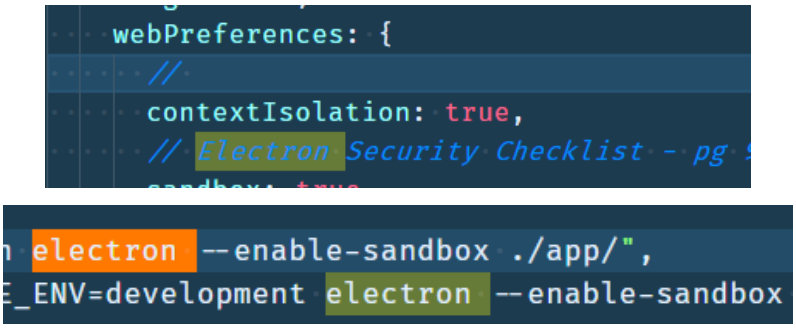


Figure 28 Another example of securing the application – in this case making the build scripts run in sandbox mode [CITATION Ele181 \l 2057]

3.2.6 PACKAGE CHOICE ✓

Throughout the development process decisions had to be made which could not be delegated to the client. These decisions would not impact the client though would impact the developer and development time. For example, the choice of the boilerplate initially was one of those decisions. Repeatedly through the project I decided what was the best way to implement a certain function. For example, I needed a way for user data to persist such as profiles for the app and other configurations. I could roll out my own system for it, however it is such a common problem there are a plethora of opensource packages to choose from. Therefore, I came up with a list and measured each of these advantages and disadvantages between each other:

Package Name	URL	Advantages	Disadvantages	License
Cosmiconfig	https://www.npmjs.com/package/cosmiconfig	+ Multiple formats: can read from a package.json, JSON, YAML, .config.js	– Would need to develop way of getting from electron renderer process to main process to save. – Not electron specific	MIT[CITATION Git17 \l 2057]
Properties	https://www.npmjs.com/package/properties	+ Built in sections + Supports .properties files	– Only supports .properties files (older format) – Would need to develop way of getting from electron renderer process to main process to save. – Not electron specific – API is full of callbacks rather than newer promises [CITATION MDN18 \l 2057] – would complicate my codebase.	MIT[CITATION Git17 \l 2057]
rc	https://www.npmjs.com/package/rc	+ Multiple formats: can read from a package.json, JSON, YAML, .config.js + Built in sections	– Would need to develop way of getting from electron renderer process to main process to save. – Not electron specific	MIT[CITATION Git17 \l 2057] and others
Configstore	https://www.npmjs.com/package/configstore	+ Multiple files + Allows encryption + JSON	– Recommends electron-store – Only JSON format	BSD 2-clause [CITATION Git171 \l 2057]
preferences	https://www.npmjs.com/package/preferences	+ Allows encryption + Multiple files + YAML+JSON	– Would need to develop way of getting from electron renderer process to main process to save.	MIT[CITATION Git17 \l 2057]

			– Not electron specific	
config	https://www.npmjs.com/package/config	+ "Simple to setup"	– Would need to develop way of getting from electron renderer process to main process to save. – Not electron specific – Only JSON	MIT[CITATION Git17 \l 2057]
Electron-store	https://www.npmjs.com/package/electron-store	+ Can use from renderer / main – no need for ipc transport + Electron Specific + Multiple files + Intuitive API	– Only JSON format – Uses electron remote – so may impact security configuration.	MIT[CITATION Git17 \l 2057]
Electron-settings	https://github.com/nathanbuchar/electron-settings	+ Electron specific	– Single file	ISC [CITATION Git172 \l 2057]

To see why licenses the packages are under is important in this process please see Section 2.8.5 ("Note about Licenses").

N.B. This is meant to serve as an example to the type of process I would go through when choosing each of my packages. However, this one will be more documented to show the process in higher detail.

3.2.6.1 CONCLUSION

In the end I decided to go with 'Electron-store' due to it seeming to be the best for my specific electron-based use case. It had an intuitive API and above all seemed to contain everything I needed in my application. Additionally, it had the bonus of supporting basic encryption to stop people changing the config files which I predicted would be a cause of errors early in the program and was later found in my evaluation in Section 4.3 "Troubleshooting".

3.3 REDUX STATE MANAGEMENT

The boilerplate I chose came with a state management system called redux (https://redux.js.org/) packaged with it. The whole idea of this state manager is to be able to easily debug the program using time travel debugging – as well as be able to analyse any state changes.

3.3.1 BASIC OF REDUX

Redux consists of 3 main components:

- Store – a store can be thought of as a giant object (map/dictionary) which can be read almost anywhere in your program. You are not able to write to it or change it any way just by altering values. Technically it is a pseudo-global immutable object.
- Action – An action is a small object with a type that can be dispatched to the store.
    - An action contains a type and a payload. An example action and 'action creator' is shown below:

Redux Example Action

```
// Action Type [Constant]
const SET_NAME = 'SET_NAME';

/**
 * Action Creator to set name of person
 * @param {string} name the name of the person
 * @see https://redux.js.org/basics/actions
 */
function setName(name) {
        return {
                type: SET_NAME,
                payload: name
        }
}

// Usage
store.dispatch(setName('George'));
```

Figure 29 Example redux action creator

- Reducer
    - A reducer is a **pure** function[3] which is registered at the start of a stores 'life'. It is passed dispatched actions from the store and it mutates the state accordingly. For the example above, it might look like:

[3] A pure function is defined as: 1) Always giving the same result given the same arguments. 2) It creates no side effects, mutation of external state etc.

Redux Example Reducer

```
// Action Type [Constant]
const SET_NAME = 'SET_NAME';

/**
 * Default state before any actions have taken place.
 * It is set automatically through es6 default args.
 */
const defaultState = {
  name: 'Bob'
}

/**
 * Pure function which returns a new state as needed
 * given the action and current state provided
 * @param {Object} state the current state
 * @param {Object} action the action provided
 */
function reducer(state = defaultState, action) {
  switch(action.type) {
    case SET_NAME:
      return { ...state, name: action.payload }
    default:
      return state;
  }
}

// Before
store.getState().name; // → Bob

// Usage
store.dispatch(setName('George'));

// After a store's cycle
store.getState().name; // → George
```

Figure 30 Example redux reducer

A basic redux flow can be shown via a simple diagram:

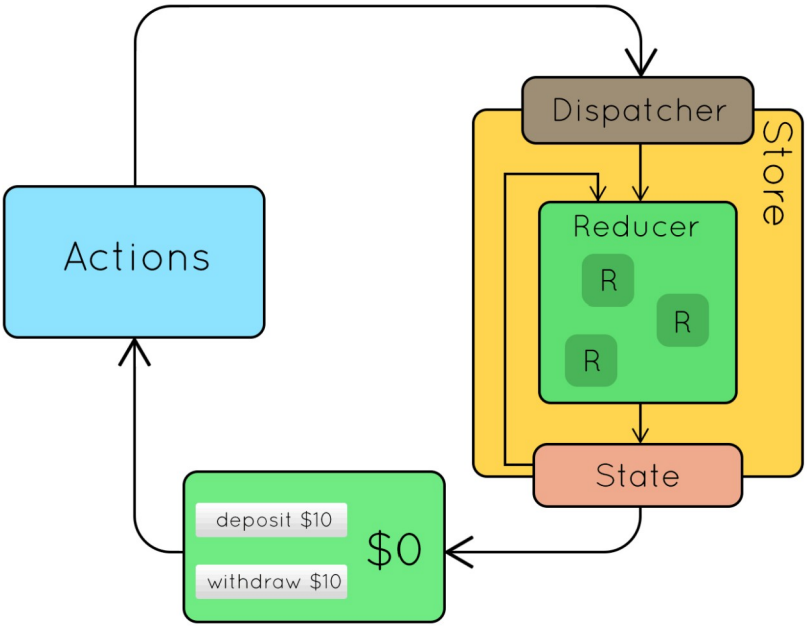


Figure 31 Example redux flow [CITATION Jen16 \l 2057]

3.3.2 TIME TRAVEL DEBUGGING

To best explain time travel debugging I believe this excerpt from Microsofts page on the topic is relevant:

Time Travel Debugging, is a tool that allows you to record an execution of your process running, then replay it later both forwards and backwards. Time Travel Debugging (TTD) can help you debug issues easier by letting you "rewind" your debugger session, instead of having to reproduce the issue until you find the bug. [CITATION Mic18 \l 2057]

Redux developer tools[4] included in my boilerplate are bundled with a timeline as shown below.

[4] https://github.com/gaearon/redux-devtools

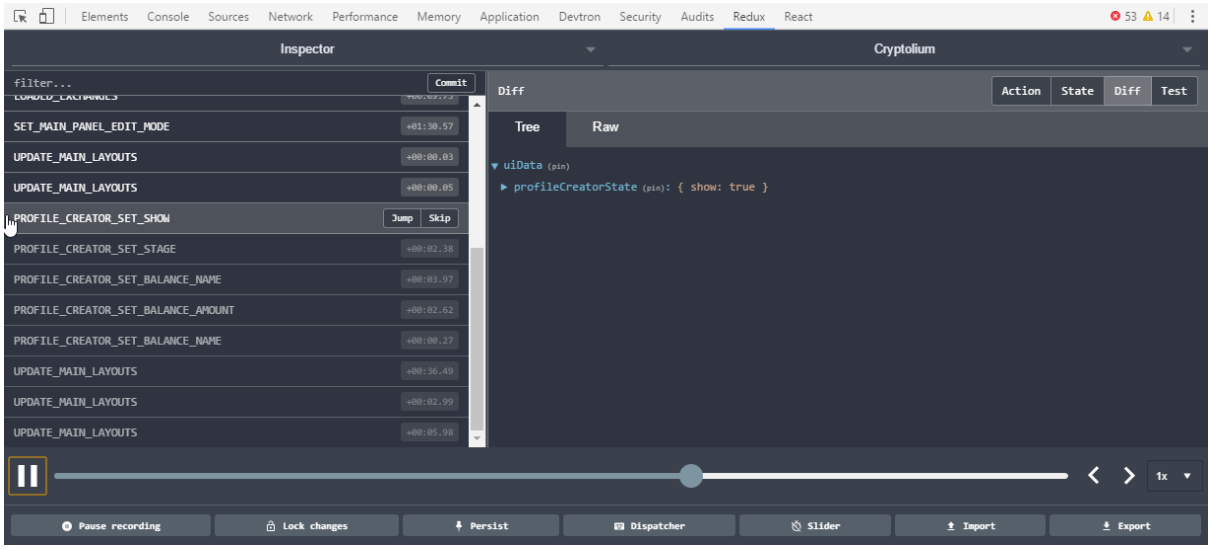


Figure 32 Redux developer tools used during running my application.

I am to move the slider to control where in the timeline I currently am. Additionally, I can skip to a specific part in the timeline as needed. This was extremely important in my development process as allowed me to locate bugs which would normally be much harder to find as I'd have to specifically set up the state of the application to where I thought the bug occurred. Additionally, many state changes in the application would happen so fast it would be hard to observe, and breakpoints could be a pain as they were hit so many times (so required many skips to get to the relevant bug occurring function).

3.3.3 USE OF REDUX

Redux was an essential part of my application and controlled most of the state within my project. Shown below is a graph of a snapshot of the state at one point during the application and a brief explanation.

3.3.4 GRAPH

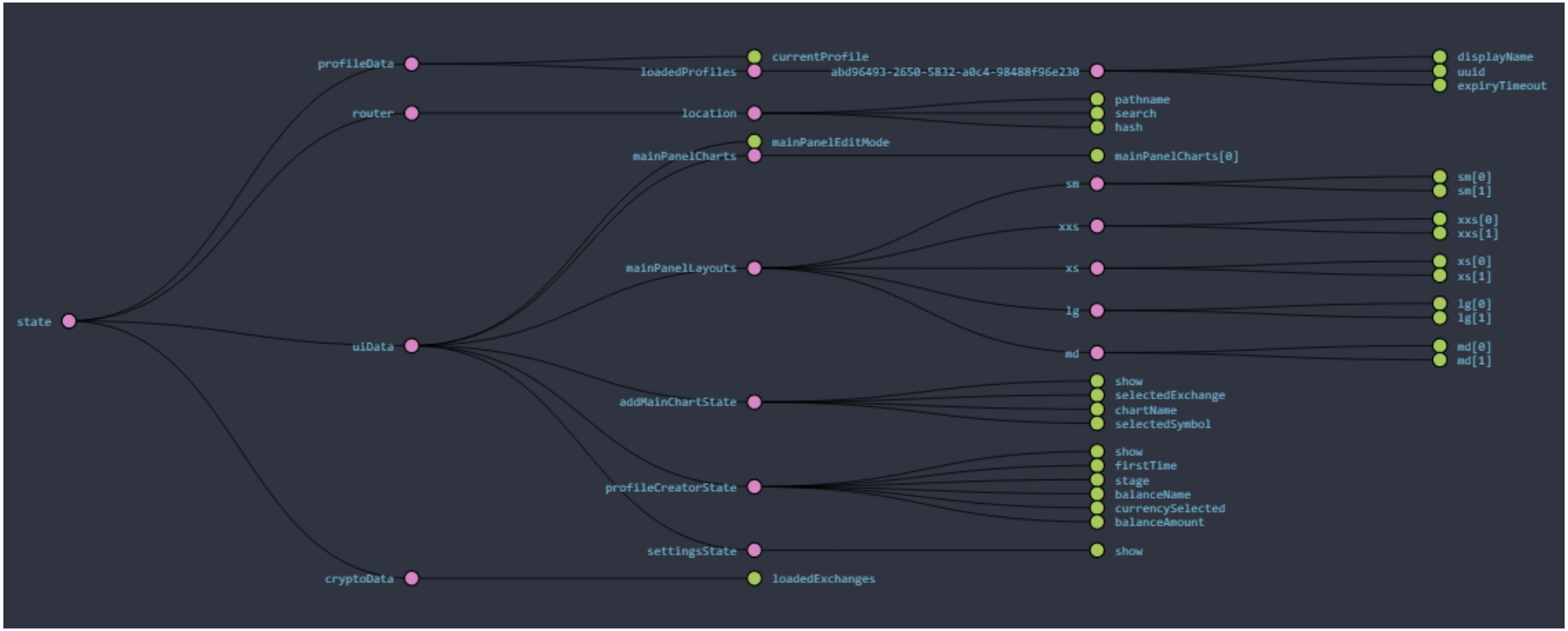


Figure 33 Graph of the redux state while running the application

In this graph we can see the splitting of the data into profileData, router, uiData and cryptoData.

3.3.5 DEVIATION FROM PURE FUNCTIONS

In Section 3.3.1 'Basic of Redux' it was mentioned how reducer functions were meant to be pure functions. However, in my development of the project I ended up making the profileReducer and uiReducer non-pure funcitons. This is because they would have the 'side-effect' of saving the data to a file after every action.

Though this breaches one of the design principles of redux, it made it much easier to maintain configuration file state. This is mentioned later in the evaluation.

3.4 CLIENT VIEWING

At this stage in development process I showed the product to my client to gauge their reaction and see whether changes needed to be made to the application to better fit their needs.

3.4.1 CHANGES

3.4.2 VISUALIZATION FRAMEWORK

After consultation with my client about the current visualization software. They claimed the visualization didn't look how they intended and wanted a change in style. I agreed a change needed to be made and considered styling the Plotly graph which they document extensively on their documentation[5]. However, I realised in total the way I'd introduced Plotly into my code was already quite cluttered as I had to expose it as a global variable, so it could be loaded properly, which is generally bad practice. This additional styling and methods required would further serve to make my code harder to maintain. Additionally, it was a hassle to make it fit properly into the grid boxes and decided not to work intermittently (due to resize events or similar not being fired correctly).

I researched any alternatives and discovered a well known open source visualization framework called Victory[6]. It had built in styling which I approved with my client were more to their requirements before implementing it. It is built for React (the framework I'm using), so it was incredibly easy to implement into my code base and I believe made my code far more readable.

[5] https://help.plot.ly/style-your-plots/
[6] https://formidable.com/open-source/victory/

3.4.2.1 CHANGES

```
@@ -1,6 +1,16 @@                                                    Discard Hunk  Stage Hunk
  1   1  // @flow
  2      import React from 'react';
      2  import React, { Component } from 'react';
  3   3  import { connect } from 'react-redux';
      4  import {
      5+ VictoryChart,
      6+   VictoryAxis,
      7+   VictoryCandlestick,
      8+   VictoryTheme,
      9+   VictoryBrushContainer,
     10+   VictoryZoomContainer
     11+ } from 'victory';
     12+ import orderBy from 'lodash/orderBy';
     13
  4  14  import moment from 'moment';
  5  15  import isArray from 'lodash/isArray';
  6  16  import isEqual from 'lodash/isEqual';

@@ -8,59 +18,39 @@ import sizeMe from 'react-sizeme';
  8  18  import CryptoAPI from '../utils/CryptoAPI';
  9  19  import type { CryptoState } from '../_types/Crypte';
 10  20  import type { MainPanelChart } from '../_types/UI';
     21
 11  22  import styles from './styles/OHLCVGraph.scss';
 12  23
 13  24  type Props = {
 14  25    chartData: MainPanelChart,
-15      -  cryptoData: CryptoState
-16      -};
-17      -
-18      -const mapStateToProps = ({ cryptoData }) => ({ cryptoData });
-19      -
-20      -class OHLCVGraph extends React.Component<Props> {
-21      -  gds = [];
-22      -
-23      -  async componentDidMount() {
-24      -    window.addEventListener('resize', () => this.gds.forEach(gd => Plotly.Plots.resize(gd)));
-25      -    if (this.isLoading) return;
-26      -    this.isLoading = true;
-27      -    this.renderGraph();
-28      -    this.isLoading = false;
-29      -  }
-30      -
-31      -  shouldComponentUpdate(nextProps: Props) {
-32      -    if (
-33      -      (nextProps.cryptoData.loadedExchanges !== this.props.cryptoData.loadedExchanges &&
-34      -        !this.loaded) ||
-35      -      !isEqual(nextProps.size, this.props.size)
-36      -    ) {
-37      -      return true;
-38      -    }
-44      -    if (this.isLoading) return;
-45      -    this.isLoading = true;
-46      -    await this.renderGraph();
-47      -    this.isLoading = false;
+26+   cryptoData: CryptoState,
+27+   size: {
+28+     width: number
 29  }
+30+ };
 31
-50      -  loaded: boolean = false;
-51      -  isLoading: boolean = false;
+32+ type State = {
+33+   candleStickData?: Array<{
+34+     x: number,
+35+     open: number,
+36+     close: number,
+37+     high: number,
+38+     low: number
+39+   }>,
+40+   selectedDomain?: any,
+41+   zoomDomain?: any
+42+ };
 43
-53      -  async renderGraph() {
 45
-59      -    if (!this.graphEl) {
-60      -      console.error('[OHLCVGraph] GraphEl is not reffed yet!');
-61      -      return;
-62      -    }
+46+ class OHLCVGraph extends Component<Props, State> {
+47+   state = {
+48+     candleStickData: null,
+49+     zoomDomain: null,
+50+     selectedDomain: null
+51+   };
 52
+53+   async getCandleStickData() {
 54    const exchange = CryptoAPI.getExchange(this.props.chartData.exchangeId);
 55
 56    if (exchange == null) {

@@ -75,88 +65,123 @@ class OHLCVGraph extends React.Component<Props> {
 65      return;
 66    }
 67
-78      -  if (this.loaded) return;
-79      -
-80      -  this.loaded = true;
-81      -  const xVals = [];
-82      -  const closeVals = [];
-83      -  const highVals = [];
-84      -  const lowVals = [];
-85      -  const openVals = [];
-86      -
-87      -  OHLCVdata.forEach(([dateInUTC, openPrice, highestPrice, lowestPrice, closingPrice]) => {
-88      -    xVals.push(moment(dateInUTC).format('YYYY-MM-DD'));
-89      -    closeVals.push(closingPrice);
-90      -    highVals.push(highestPrice);
-91      -    lowVals.push(lowestPrice);
-92      -    openVals.push(openPrice);
-93      -  });
+68+   let dataVals = [];
 69
 70    const { d3 } = window.Plotly;
 71    const WIDTH_IN_PERCENT_OF_PARENT = 100;
 72    const HEIGHT_IN_PERCENT_OF_PARENT = 100;
 73    const gd3 = d3
 74      .select(this.graphEl)
 75      .append('div')
-100     -    .style({
-101     -      position: 'absolute',
-102     -      top: 0,
-103     -      left: 0,
-104     -      width: `${WIDTH_IN_PERCENT_OF_PARENT}%`,
+76+       x: new Date(time),
+77+       open,
+78+       close,
+79+       high,
+80+       low
 81      });
-110     -  let gd;
-111     -  this.gds.push((gd = gd3.node()));
```

Figure 34 Excerpt of the changes in 'OHLCVGraph.js' of moving from Plotly to Victory

3.4.2.2 BEFORE

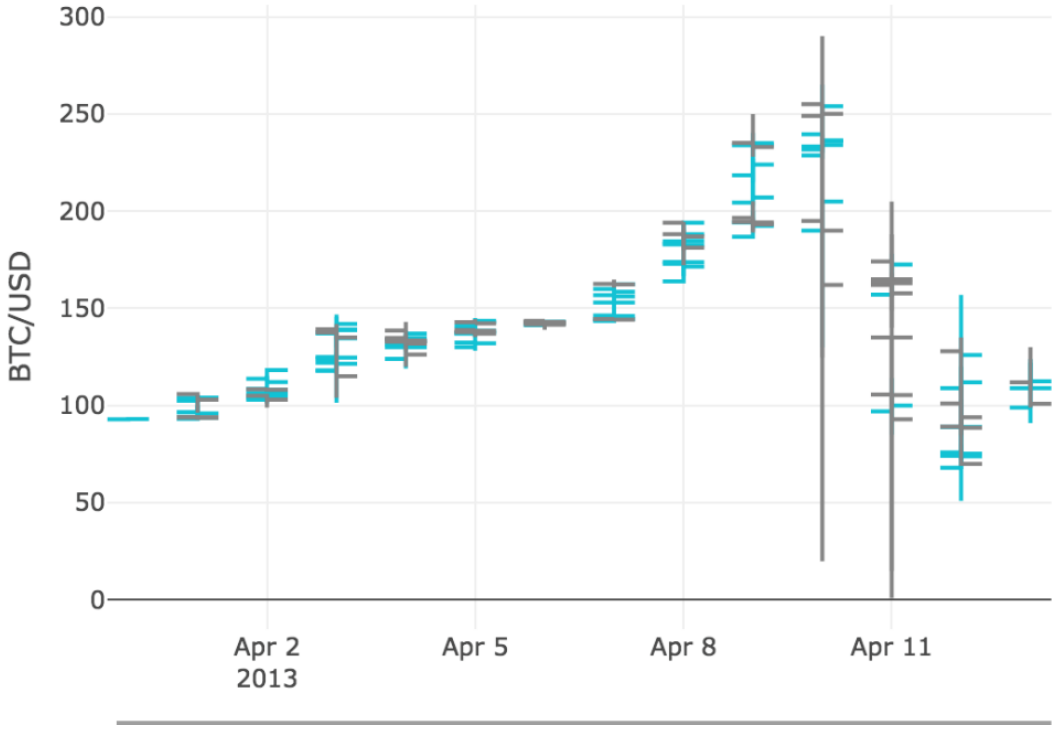


Figure 35 The view of the graph using Plotly

3.4.2.3 AFTER

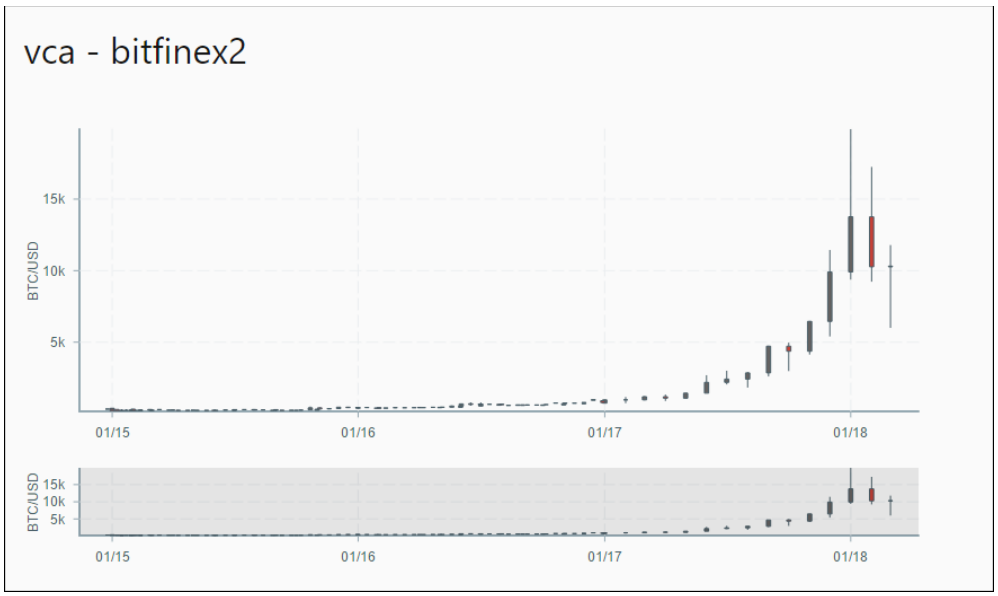


Figure 36 The graph looks with Victory as the data visualization framework

3.5 INTELLIGENT ERROR HANDLING ✓

I realised during the creation of my application that every time I received an error, it would just show a blank screen to the user in the main window and I'd have to open the developer console to view the error. To make it clearer when an error happened I used a new feature of React 16 called 'error boundaries':

*Error boundaries are React components that **catch JavaScript errors anywhere in their child component tree, log those errors, and display a fallback UI** instead of the component tree that crashed. Error boundaries catch errors during rendering, in lifecycle methods, and in constructors of the whole tree below them.*

Figure 37 A quote from the official React dev block article concerning Error Boundaries [CITATION Fac181 \l 2057]

This provided clear advantages:

+ Allowed the user to instantly see an error has happened (instead of a blank screen)
+ It informs them to contact the developer with a brief description of it which encourages bug fixing
+ It provides a better user experience

3.5.1 IMPLEMENTATION

I implemented one error boundary as shown:

```
/**
 * Catch any error within the component stack
 * @param {Error} error - the error thrown
 */
// eslint-disable-next-line class-methods-use-this
componentDidCatch(error: Error) {
  mySwal({
    title: 'Error',
    html: (
      <div style={{ textAlign: 'center' }}>
        <b>Contact the developer with this error</b>
        <br />
        {error.toString()}
      </div>
    ),
    type: 'error'
  });
}
```

Figure 38 An excerpt of my code implementing the error boundary

This error boundary was implemented on my main component – the `Home` component. The mySwal function is Sweet Alert 2[7] with an additional plugin to allow react content[8].

It displays an error as shown:

[7] https://sweetalert2.github.io/
[8] https://github.com/sweetalert2/sweetalert2-react-content

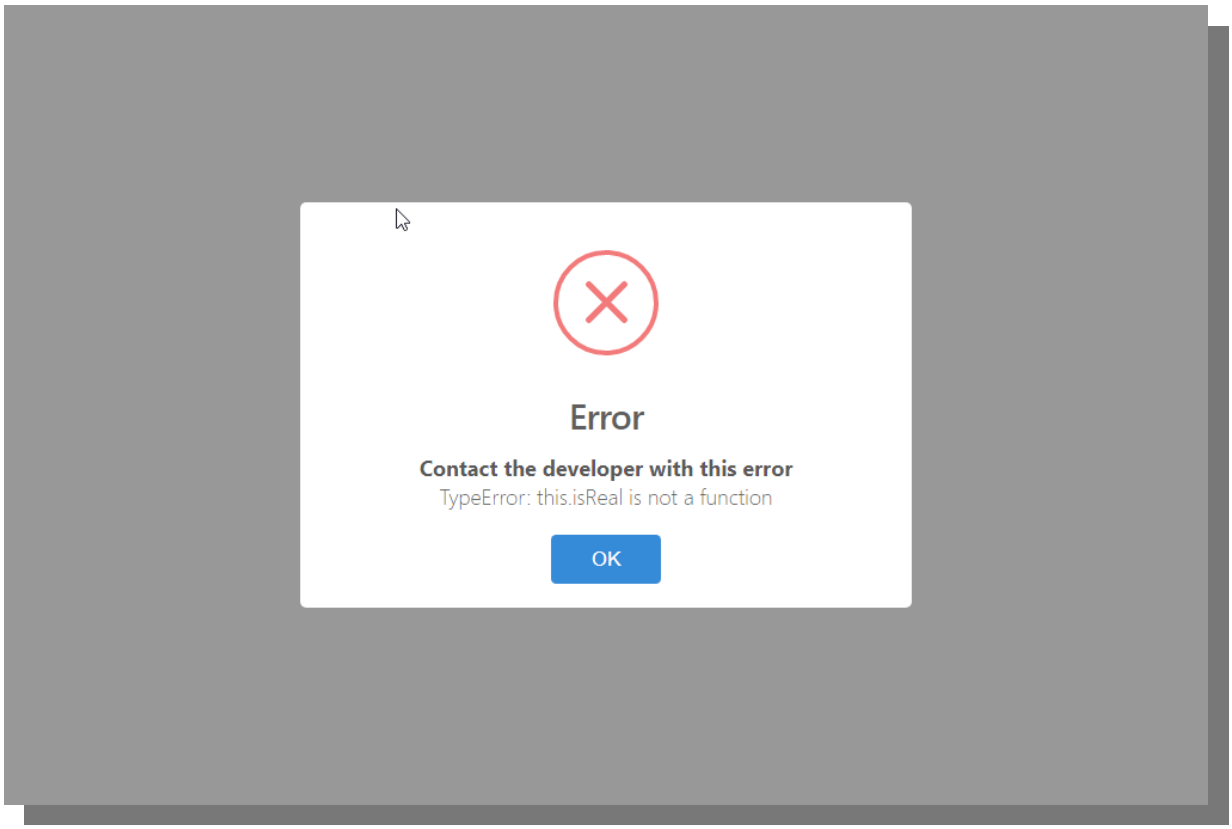


Figure 39 Example error presented using error boundary

In this particular example it allowed me to easily locate the error in my program:

Figure 40 The error in my code

3.6 PROJECT STRUCTURE AT END ✕

- Computer Science 2018
  - assets
    - screenshots
  - docs
  - resources
  - src
    - .vscode
    - app
      - actions
        - types
      - components
      - containers
      - dist
      - enc_keys
      - node_modules
      - reducers
        - ipc
      - store
      - utils
      - _app_node
      - _types
    - dll
    - flow-typed

        - npm
            - @fortawesome
    - internals
        - flow
        - img
        - mocks
        - scripts
    - node_modules
    - release
        - win-unpacked
            - locales
            - resources
    - resources
        - icons
    - test
        - actions
            - __snapshots__
        - components
            - __snapshots__
        - containers
        - e2e
        - reducers
        - utils
- _no-flow-src
    - .vscode
    - app
        - actions
            - types
        - components
        - containers
        - dist
        - enc_keys
        - reducers
            - ipc
        - store
        - utils
        - _app_node
        - _types
    - dll
    - resources
        - icons
    - test
        - actions
            - __snapshots__
        - components
            - __snapshots__
        - containers
        - e2e
        - reducers
        - utils
- _screen_dir
    - app
        - actions
        - components
        - containers

Update with `dirsToLi -d . -I node_modules,.git`

3.7 PROGRAMMING FEATURES ✕

Throughout this project is was necessary to use a variety of programming features. These included but not limited to:

- Iteration
- Conditionals

- Switching / Case conditions
- Enums (Enumerated Types)
- Inheritance / Polymorphism
- Functional programming

For the purposes of this writeup I will provide an example for each of these features to show the variety of programming technique used within this project. N.B. These are just excerpts so I can show some of my reasoning during development, the full source code is linked in Appendix A.

3.7.1 CLASS USE

Throughout my program I made use of classes and various generalisations. For example, to allow the user to edit their name seamlessly as shown

Figure 41 Through the user action of 'clicking' the name is selected, and the user is able to type and replace the text

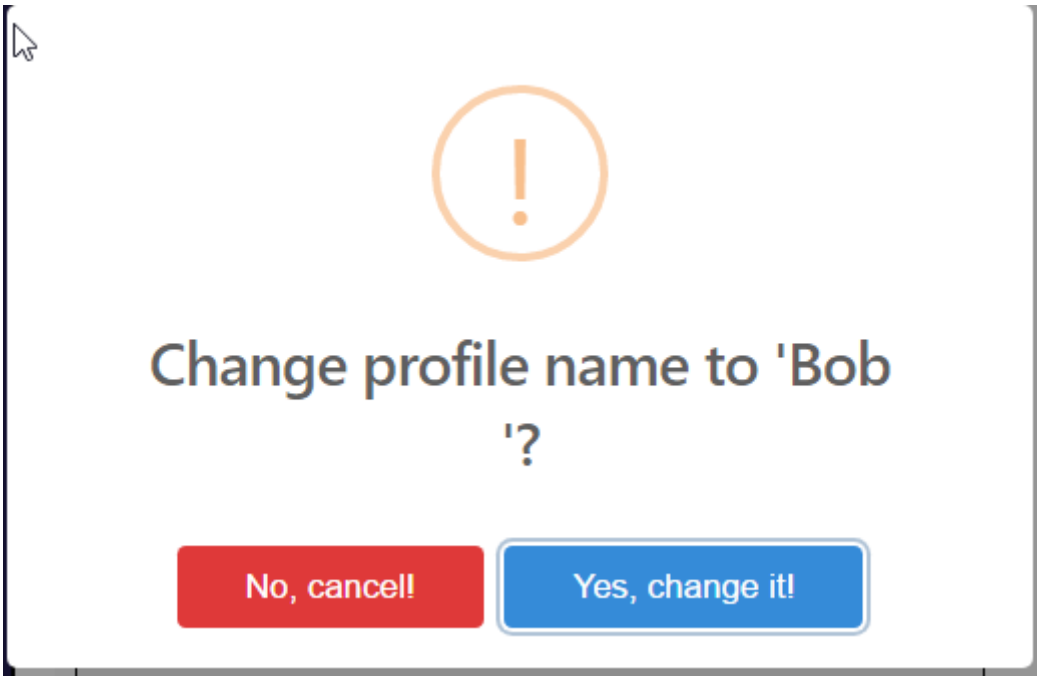


Figure 42 When the user clicks away a dialog box is presented to them confirming their new name choice.

For this process I could have created a specific solution to the problem within the 'SidePanel' component. However, this would serve to clutter the component with extra logic. Additionally, it makes it more unclear, rather than just creating a generalised 'EditableText' component which is made for just this purpose. It also has the benefit that I can reuse it anywhere with little to none extra programming overhead.

This shown class is below and is annotated with comments.

src/app/components/EditableText.js

```
// @flow
import React, { Component } from 'react';

type Props = {
  html: string,
  onChange?: string => void | Promise<void>,
  onBlur?: string => void | Promise<void>,
  disabled?: boolean
};

/**
 * A react component allowing text which can be edited though appear
 * inline.
 */
export default class EditableText extends Component<Props> {
  /**
   * @see https://reactjs.org/docs/react-component.html#shouldcomponentupdate
   */
  shouldComponentUpdate(nextProps: ?Props) {
    return this.el != null && nextProps != null && nextProps.html !== this.el.innerHTML;
  }

  /**
   * @see https://reactjs.org/docs/react-component.html#componentdidupdate
   */
  componentDidUpdate() {
    if (this.el != null && this.props.html !== this.el.innerHTML) {
      this.el.innerHTML = this.props.html;
    }
  }

  // Variable which contains the last known html of the text
  lastHTML: ?string = null;

  // Reference to the element that is holding the text
  el: ?HTMLSpanElement = null;

  /**
   * This method emits the change up to the parent component
   * @param {boolean} blur - whether the event was a blur event or not (a onChange event)
   */
  emitChange(blur: boolean) {
    // if the element hasn't been loaded yet return
    if (!this.el) return;

    const html = this.el.innerHTML;

    // on blur evt
    if (blur && this.props.onBlur != null) {
      this.props.onBlur(html);
    }
    /**
     * I intentionally want it to call both onChange
     * and onBlur, when onBlur
     */

    // on standard change
    if (this.props.onChange != null && html !== this.lastHTML) {
      this.props.onChange(html);
    }
    this.lastHTML = html;
  }

  render() {
    const {
      onChange, html, onBlur, disabled, ...rest
    } = this.props;

    return (
      <span
        {...rest}
        contentEditable={!disabled}
        onInput={() => this.emitChange(false)}
        onBlur={() => this.emitChange(true)}
        /**
         * Here I am purposefully disabling the react/no-danger flag
         * This is because I am aware of the dangers of setting the html this way
         * However it is necessary for this use
         * @see https://reactjs.org/docs/dom-elements.html#dangerouslysetinnerhtml
         */
        // eslint-disable-next-line react/no-danger
        dangerouslySetInnerHTML={{ __html: html }}
        ref={node => (this.el = node)}
      />
    );
  }
}
```

3.7.2 SWITCH STATEMENT USE

For my use of reducers as outlined in the section about Redux State Management (Section 3.3) I had to make use of a switch block. This allowed me to easily and understandably switch between various action types that could be passed to the redux reducer.
In this annotated example, I am switching between action types such as `ADD_MAIN_CHART_SET_SHOW` and `ADD_MAIN_CHART_SET_NAME`. Depending on the action type I update the state variable accordingly.

src/app/reducers/ui/addMainChart.js

```
// @flow
/* eslint no-param-reassign:0 */
/**
 * The merge function supplied by lodash
 * allows us to merge the old state with
 * our new values
 * @see https://lodash.com/docs/4.17.5#merge
 */
import merge from 'lodash/merge';

// Import all the different actionTypes dealt with
import {
  ADD_MAIN_CHART_SET_SHOW,
  ADD_MAIN_CHART_SET_SELECTED_EXCHANGE,
  ADD_MAIN_CHART_SET_SELECTED_SYMBOL,
  ADD_MAIN_CHART_SET_NAME
} from '../../actions/types/addMainChart';

// Types for FlowJS
import type { actionType } from '../../_types/ActionType';
import type { addMainChartState } from '../../_types/UI';

// The default state for the store to take
const defaultMainChartState: addMainChartState = {
  show: false // we don't want to the addMainChartDialog to show initially
};

/**
 * Pure reducer function which takes in the current
 * state of the redux store (or a portion of it)
 * and an action.
 * It then returns a new state modified according
 * to the action
 * @param {?addMainChartState} state the current state
 * @param {actionType} action the action to 'perform' on it
 * @returns Updated State
 */
export default function addMainChartReducer(
  /**
   * This will set the state to the default if undefined
   */
  state: ?addMainChartState = defaultMainChartState,
  action: actionType
) {
  /**
   * We purposefully disable eslint no-param-reassign
   * due to reassign the parameter state throughout this method
   * I don't believe this is a case of bad practice,
   * as the alternative is creating a new redundant variable with
   * an abritary name. This way the meaning of the method is much clearer
   * @see https://eslint.org/docs/rules/no-param-reassign
   */

  /**
   * All actions passed in have a type which can be switched between
   * as needed with the following switch statement
```

```
    * as needed with the following switch statement
    */
  switch (action.type) {
    // Set whether to show to the addMainChartDialog
    case ADD_MAIN_CHART_SET_SHOW:
      /**
       * Here we are replacing the state with a new one
       * which has show set to true/false depending on
       * the action payload
       */
      state = merge({}, state, { show: action.payload });
      break;
    case ADD_MAIN_CHART_SET_SELECTED_EXCHANGE:
      /**
       * Setting the selectedExchange based on action payload
       */
      state = merge({}, state, { selectedExchange: action.payload });
      break;
    case ADD_MAIN_CHART_SET_SELECTED_SYMBOL:
      /**
       * Setting the selectedSymbol based on action payload
       */
      state = merge({}, state, { selectedSymbol: action.payload });
      break;
    case ADD_MAIN_CHART_SET_NAME:
      /**
       * Setting the chartName based on action payload
       */
      state = merge({}, state, { chartName: action.payload });
      break;
    default:
      /**
       * In default case do nothing: this is because other
       * actions types that have nothing to do with this
       * reducer may be passed in.
       */
      break;
  }
  /**
   * Return the state which has been replaced or
   * not changed in anyway
   */
  return state;
}
```

3.8 TROUBLESHOOTING

When testing my program multiple times, I realise most errors occurred due to incorrect default configuration settings, or a setting being set wrong which caused the program to throw many spurious errors. For example:

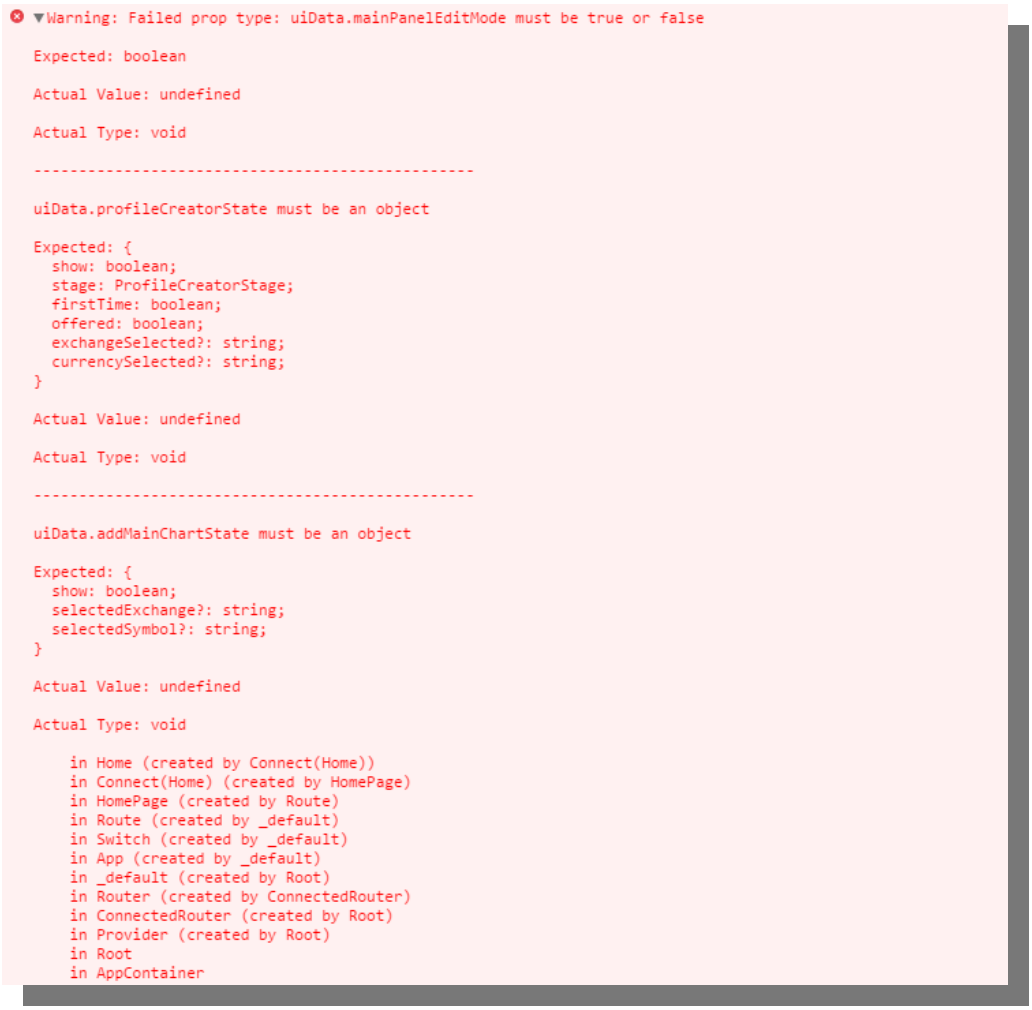


Figure 43 screenshot of an example error caused by a mistake in the configuration file.

In this error we can see how 'uiData' had an unknown property 'mainPanelEditMode'. This is because I hadn't set up the default config to contain it – so it didn't get set till it was used. This error was just a warning, so it was able to continue the program without the user noticing. However, it presents a significant problem in having configuration files. Due to the time restrictions of the application I am unable to build a full fledge configuration handler which can cope with missed keys and provide the user meaningful messages. Therefore, as part of my troubleshooting advice I am forced to recommend that if an error occurs, the program should be stopped, the configuration files deleted, and the programs restarted so they can be 'regenerated' properly. This is reflected in my evaluation.

3.9 TEST RESULTS ✗

3.9.1 RESULTS TABLE ✕

3.9.2 CLIENT SIGNOFF

The client mentioned how the password security test didn't pass and I talked to him about it. Even though it was part of our original MVP, we agreed it was not essential and even though easily implemented was not needed at this time. Apart from that the client was happy with the results of the rest of the tests and was content signing off this project as complete.

4 EVALUATION ✕

This application was a challenge to develop though I am glad the client was happy with the result. However, many features of this application and its development process must be evaluated to ascertain where any improvements to the process can be improved.

4.1 FLOWJS SLOWDOWNS

Through the production of my project I encountered various FlowJS problems. These were problems due to me explicitly having to describe a FlowJS type for even the most menial methods. Though I know this prevented many static bugs, it also slowed down the rate of development actively as almost double the amount of code needed to be written. In future I don't think FlowJS is beneficial for a project like this unless it is treated as more optional (as needed) rather than for every single file like I ended up doing.

4.2 REDUX DESIGN DEVIATION

I mentioned in section 3.3.5 "Deviation from Pure Functions" that I did not follow the common design principle of having pure functions as reducers within my redux-based application. I explained in that section how this was to make it easier to keep configuration file state. Evaluating that decision now I realise that was most likely in error. Even though in this application it had no down-sides, if the client wishes me to extend the application or hand it over to another developer at some point it may have unintended consequences as they won't be expecting the side effects from these reducers.

The correct or recommended way to have saving to a configuration file from a redux state is probably via either Middleware[9] or Thunks[10]. Knowing this now I would implement it as such.

4.3 SECURITY PROBLEMS

In this section it may be worth referring to Section 2.3.6, 'Package Choice', and Section 2.3.5, 'Security Checklist'. In my Security Checklist section, I showed how I added a variety of security measures to secure my application. Unfortunately, due to the package choice I made – which require the electron 'remote'[11] object to be accessible. Therefore, many of the security measures introduced never made it into the final source code as they stopped the application from working. An example error message caused is shown:

[9] https://redux.js.org/advanced/middleware
[10] https://github.com/gaearon/redux-thunk
[11] https://github.com/electron/electron/blob/master/docs/api/remote.md

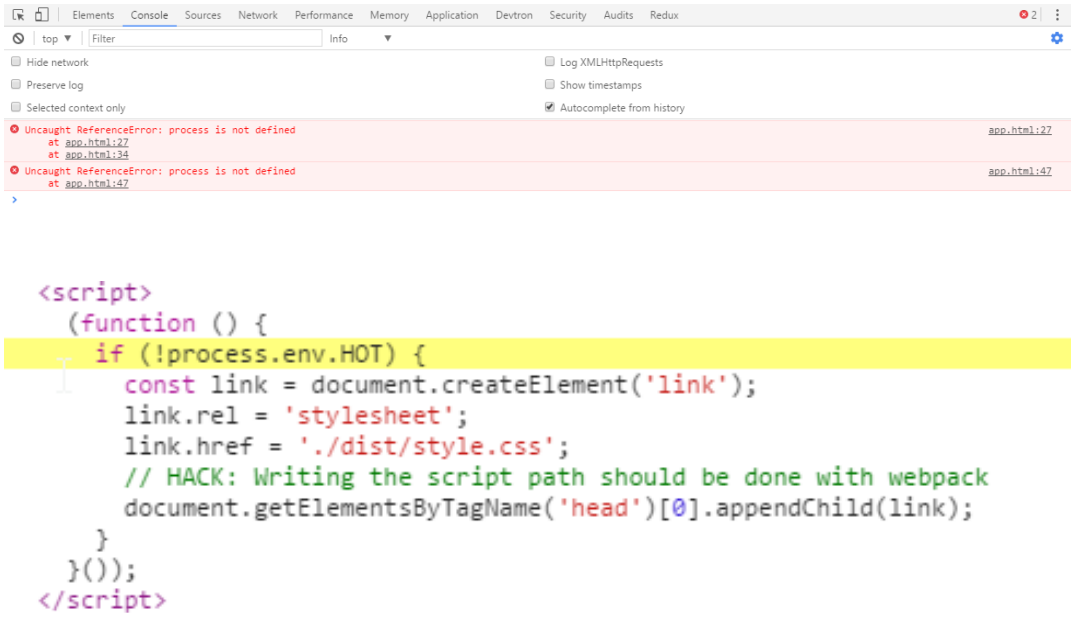


Figure 44 Error and cause (process is undefined when in sandbox mode as node constants like process are not exposed [CITATION Ele182 \l 2057])

Knowing this now I realise I was too presumptive to include the security enhancements so early on without knowing the impact they'd have on the overall application. Therefore, with the benefit of hindsight I would keep this in mind especially for future projects using ElectronJS.

4.4 TROUBLESHOOTING

Shown in Section 3.8 – Troubleshooting, we can see that many problems occurred purely due to my configuration files and possible desynchronization, incorrect values and other problems presented by them. This meant I was forced to recommend regenerating them if a problem occurred in the program. In future projects, ones with more permissive timing schedules. I would instead write an error correction system for the configuration files, to replace incorrect values and add in values which don't exist. Additionally, provide a message to the user that the configuration files were at fault for the error and it wasn't just a general program fault.

5 CONCLUSION -

In conclusion, I believe generally this project has been a success due to fulfilling all the required MVP goals (and ones that weren't eventually required were discussed with the client). There is room for improvement as shown in my evaluation. However, in general this project achieved its goals.

5.1 MAINTANANCE

My application requires little maintenance generally. The most likely part of needing maintained is cryptocurrency exchanges closing or changing their APIs. Luckily though as I am using the ccxt[12] package as my interface to interact with them, it is likely all that would require me to do is update the version used in my program and repackage it to give to the user. The distribution of the updated packaged application to the user could be done through the client.

5.2 SIMILAR PRODUCTS

Through the creation of this product it came to my attention that a similar product was just realised by the name of "Cointracker" [CITATION Nin18 \l 2057]. I believe my project is significantly different however I contacted my client concerning it. They assured me that they still wished the project to be completed as they believe they will still be able to seek a market for the product.

Additionally, at the very end of the development a product that my client theorised would be released was by Delta[13]. This again had similar specifications, though again the client was fine with my continued development on our own version.

I believe this shows the time relevance of products like this and how if they are needed they will be created by the market. Additionally, it is increasingly easy to unknowingly compete in development with other similar projects, which just shows how relevant ideas like development velocity and planning time are.

[12] https://github.com/ccxt/ccxt
[13] https://getdelta.io/

6 APPENDIX A - SOURCE CODE

Due to the size of my project (~17000 lines), it seems infeasible to include every single program file as a picture or formatted text within this document. Therefore, the best compromise is to place some of the program files, which best demonstrate the style / programming techniques used (as shown in Section 3.7) within the project and light commentary on which and provide a link to an online repository with all the project files on. Additionally, it contains a very overt README specifying how to run the program if one wishes.

Online repository link: https://github.com/georgePadolsey/ComputerScience2018

N.B. I can guarantee this link will be valid till 2020 at the least.

7 APPENDIX B - RUNNING THE APPLICATION ✓

The following is from the `HOW_TO_RUN.md` in the main GitHub repository[14] specified in Appendix A:

How to run

Release mode

To view the 'finished' application as an executable file which can be run on Windows/Mac/Linux platforms visit the releases page[1] on the GitHub repo.

Choose the latest version and appropriate file for your platform. (i.e. `.exe` for Windows etc.)

Development mode

Requirements

To be able to run this on your computer in development mode. You must have:

- npm[2] installed (Recommended installing with nvm[3])
- OPTIONAL/RECOMMENDED: yarn[4] installed

Steps

To view the 'finished' application in `development` mode where code can be change and debugging tools are available either:

- Visit the GitHub releases page[1] and download the latest source code (the zip file)
- Clone the GitHub repository[6]. To see a tutorial for how to do this go to the Github Help page[7]

Then run the following commands from the main directory:

- `cd src` to switch to the src directory
- `yarn install` or `npm install` to install all the dependencies
- `yarn run dev` or `npm run dev` to run the program in development mode. It should open the application in a new window after a few seconds.

[14] https://github.com/georgePadolsey/ComputerScience2018

Links

1. Application releases https://github.com/georgePadolsey/ComputerScience2018/releases
2. NPM https://www.npmjs.com/
3. Node Version Manager https://github.com/creationix/nvm
4. Yarn https://yarnpkg.com/en/
5. Github repo https://github.com/georgePadolsey/ComputerScience2018
6. GitHub help page for cloning a repository https://help.github.com/articles/cloning-a-repository/

8 BIBLIOGRAPHY

Boehm, 2004. *File:Spiral model (Boehm, 1988).svg - Wikimedia Commons.* [Online]
Available at: https://commons.wikimedia.org/wiki/File:Spiral_model_(Boehm,_1988).svg
[Accessed 04 02 2018].

Electron contributors, 2018. *electron/sandbox-option.md at master · electron/electron.* [Online]
Available at: https://github.com/electron/electron/blob/master/docs/api/sandbox-option.md
[Accessed 19 March 2018].

ElectronJS, 2017. *Electron | Build cross platform desktop apps with JavaScript, HTML, and CSS..* [Online]
Available at: https://electronjs.org/

ElectronJS, 2018. *sandbox Option | Electron.* [Online]
Available at: https://electronjs.org/docs/api/sandbox-option
[Accessed 27 March 2018].

ElectronJS, 2018. *Supported Platforms | Electron.* [Online]
Available at: https://electronjs.org/docs/tutorial/supported-platforms
[Accessed 04 02 2018].

Facebook Inc., 2018. *Error Handling in React 16 - React Blog.* [Online]
Available at: https://reactjs.org/blog/2017/07/26/error-handling-in-react-16.html
[Accessed 14 March 2018].

Facebook Inc., 2018. *Flow: A Static Type Checker for JavaScript.* [Online]
Available at: https://flow.org/
[Accessed 13 March 2018].

Github Inc, 2018. *Trending C# repositories on GitHub today.* [Online]
Available at: https://github.com/trending/c%23
[Accessed 04 02 2018].

Github, Inc, 2018. *BSD 2-Clause "Simplified" License | Choose a License.* [Online]
Available at: https://choosealicense.com/licenses/bsd-2-clause/
[Accessed 7 March 2018].

Github, Inc, 2018. *ISC License | Choose a License.* [Online]
Available at: https://choosealicense.com/licenses/isc/
[Accessed 7 March 2018].

Github, Inc, 2018. *MIT License | Choose a License.* [Online]
Available at: https://choosealicense.com/licenses/mit/
[Accessed 7 March 2018].

larion, 2017. *larion/cryptolio: A command-line cryptocurrency portfolio management tool.* [Online]
Available at: https://github.com/larion/cryptolio
[Accessed 15 March 2018].

MDN, 2018. *Promise - JavaScript | MDN.* [Online]
Available at: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise
[Accessed 19 March 2018].

Nin Finance, Inc, 2018. *CoinTracker · Cryptocurrency Portfolio & Tax Manager.* [Online]
Available at: https://www.cointracker.io/
[Accessed 7 March 2018].

Plotly Ltd, 2018. *plotly.js | JavaScript Graphing Library.* [Online]
Available at: https://plot.ly/javascript/
[Accessed 21 March 2018].

reactjs, 2018. *reactjs/react-redux: Official React bindings for Redux.* [Online]
Available at: https://github.com/reactjs/react-redux
[Accessed 13 March 2018].

Reynolds, M., 2016. *Why Brand Names Are So Important, According To Science | HuffPost.* [Online]
Available at: https://www.huffingtonpost.com/molly-reynolds/why-brand-names-are-so-im_b_11930994.html
[Accessed March 21 2018].

Terpil, J., 2016. *Redux. From twitter hype to production by Jenya Terpil.* [Online]
Available at: https://slides.com/jenyaterpil/redux-from-twitter-hype-to-production#/
[Accessed 26 March 2018].

Various, 2016. *Ask HN: Why / Why Not Use Electron? | Hacker News.* [Online]
Available at: https://news.ycombinator.com/item?id=12119278
[Accessed 04 02 2017].

W3C, 2015. *Graceful degradation versus progressive enhancement - W3C Wiki.* [Online]
Available at: https://www.w3.org/wiki/Graceful_degradation_versus_progressive_enhancement
[Accessed 04 02 2018].

Wikipedia, 2018. *Candlestick chart - Wikipedia.* [Online]
Available at: https://en.wikipedia.org/wiki/Candlestick_chart
[Accessed 13 March 2018].