Computing Coursework 2018

Planning

Initial transcript

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

hmm like information on the current exchange rate?

Client

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

bittrex currently have one that I can use on an iOS app

Me

hmmm interesting - I mean it would need to integrate with wallets which would be more complex -> though why not just use a website to look up this stuff?

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

hmmm okay

would be interesting to work on - let me just have a look at the apis out there

Client:

Alrighty

Me:

so I just got bittrix on my phone and I see the market you mean - you sure there's no one of these for windows already?

Client:

They provide an API, which I have found only one app that can use it

There's one company called Delta which could potentially be releasing something

Just wondering what your thoughts on the whole situation were

Me:

I saw hmmm - i'll have a look at making a simple PoC and see how long it takes to integrate stuff together - looks like a fun project - and delta looks pretty good - I'm surprised no ones released a desktop version...

Client:

So am I, I would have thought they would release desktop before iOS or apps

Me:

mhm

I guess mobile is such a big market atm?

[...]

Design transcript

[...]

Client Brief

After the initial transcripts the client provided a brief outline of the product:

> *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.*

MVP Plan

A minimum viable product plan – my interpretation of the client's requirements:

1) Desktop based application
2) Ability to make a portfolio
    a) Ability to add a wallet/exchange/simple amount of coin
        i) Ability to remove wallet / change simple amount of initial coin
    b) Ability to watch coin gain / fall relative to the initial input
3) Lookup current exchange rates
    a) Support for multiple exchanges
        i) Average
        ii) Binance
        iii) Bitflyer
        iv) Bitfinex
        v) Bithumb
        vi) Bitsamp
        vii) Bittrex
        viii) Coinnest
        ix) Coinone
        x) Gdax

        xi) Geminin
        xii) Hitbtc
        xiii) Korbit
        xiv) Kraken
        xv) Liqui
        xvi) Poloniex
        xvii) WEX
    b) Allowing changing local currency conversion
4) Security
    a) Google Account based
        i) Two factor auth
        ii) Password
5) Analytics
    a) This is to analyse what actions have been taken in the application.
6) Licensing
    a) The client has suggested he only wishes the application to exist. He would be willing to spend money for it. Though has additionally indicated that it would be fine to sell on. For this requirement I would need to introduce a license server so I can control who is authorised / has paid for the application and who hasn't.

Similar product research

In the aim of making my application the most relevant and to not reinvent invented products. I looked at many similar products across different platforms:

Coin Ticker iPhone - https://itunes.apple.com/gb/app/coin-ticker-bitcoin-altcoin/id636476147?mt=8

Coin ticker for iPhone provides many features

Cryptolio - https://github.com/larion/cryptolio

Terminal based crypto currency portfolio

CryptoCompare - https://www.cryptocompare.com/portfolio/

General development model

Throughout the development of this application I have opted to choose 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.

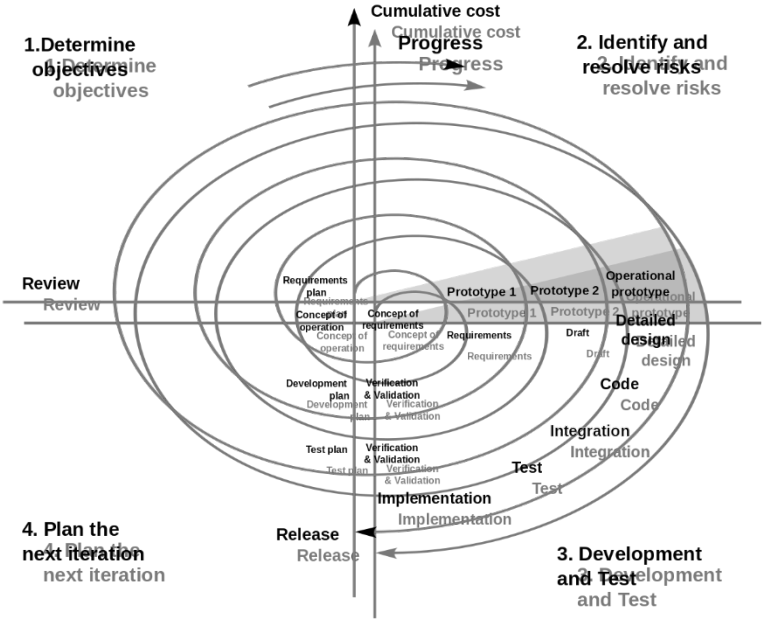


Figure 1 Spiral model development [CITATION Boe04 \l 2057]

Technologies needed

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.

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.

Advantages

+ Well supported/Much documentation
+ Very well used

Disadvantages

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

Java / JavaFX

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

Advantages

+ Well-structured language made to fit OOP

Disadvantages

– Learning overhead with the xml language
– Harder to make look native (cannot naturally embed native UI elements)

- 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

Electron - https://electronjs.org/

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]

Advantages

+ Very easy to setup
+ Cross platform
+ Can still access lower level OS features
+ Familiar technologies
+ Open Source

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)

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.

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/

Boilerplate comparison

When creating desktop applications with electron there can be a lot of setup in terms of properly isolating the renderer from the main process (to prevent other programs injecting code). 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. Then there's the problem of managing state in large programs which is generally done through libraries like redux which have direct bindings into Angular or React (see redux-react).

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. 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.

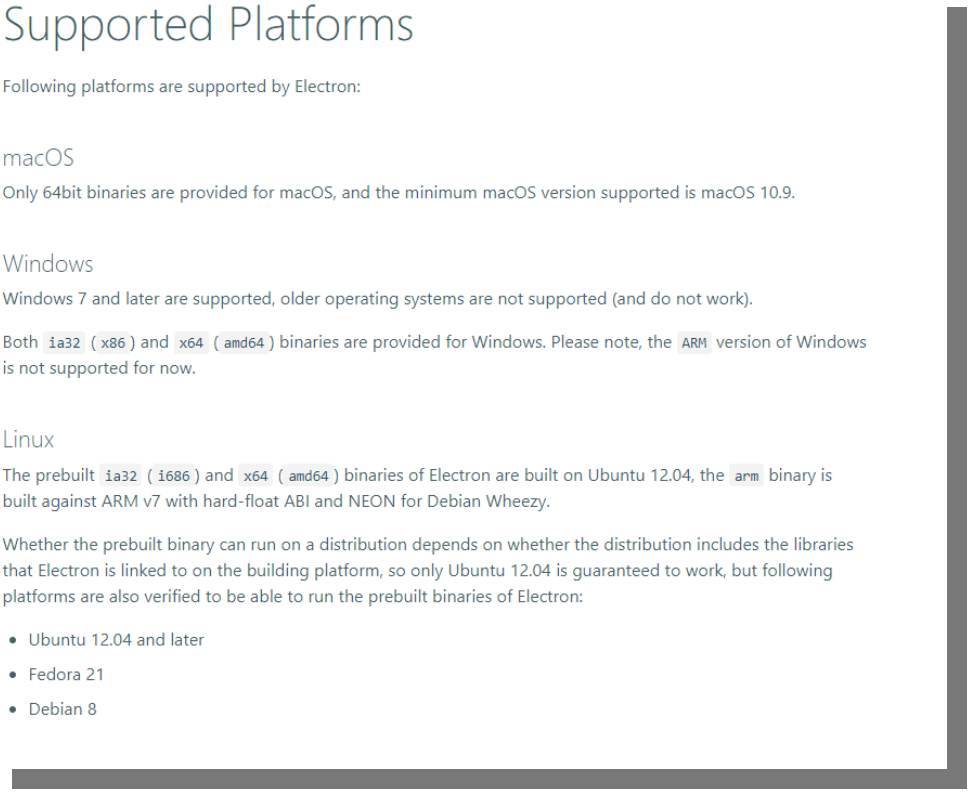
Style choices

Testing framework

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 2 Supported systems [CITATION Ele18 \l 2057]

My application would not require any special additional requirements on top of ElectronJS's ones except for possibly and 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].

Conclusion

How my MVP and general product is solvable using the technologies I have chosen

Problems that will be hard to solve

Basic Layout design

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

Figure 3 – A basic design of what the application might look like

Colours used for mockup:

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:

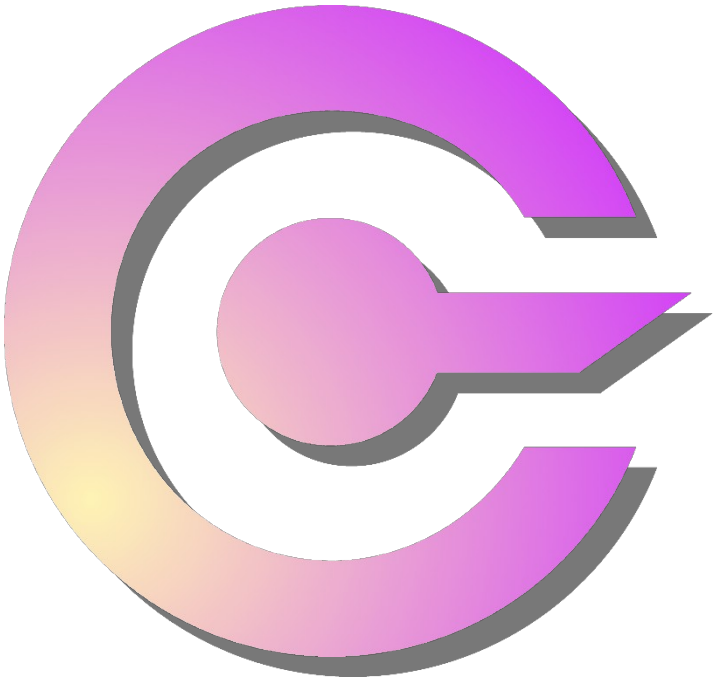


Figure 4 Retouched cryptocurrency logo / New Application logo

Colour Specification for logo:

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

Tests needed for MVP

Test name	Test Description	MVP Spec
Basic Load	The application loads up	
UI Exists	The UI is present in the rendered application	

Name Choice

This may seem like a trivial task for an application. However, it could be argued that the name has an impact on the clients view on the final product.

Considered names need to reflect the nature of the application being:

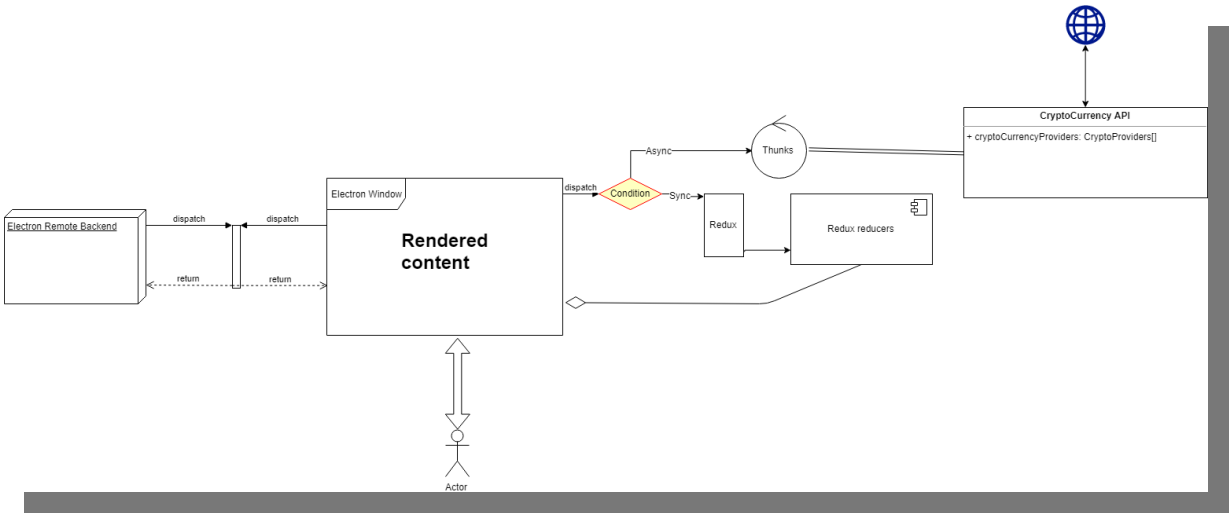
- Modern
- Cryptocurrency
- Portfolio
- Sleek
- Easy to use
- Secure
- Safe

Considered names:

- Cryptolio
    - Portmantuas are cliché and non-modern but effective
    - NAME CLASH: https://github.com/larion/cryptolio
- Crypto Buddy
    - Overly friendly, doesn't seem secure?
    - NAME CLASH: 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 it slightly into Cryptolium. Which makes it sound more professional and as effective.

Problem splitting



Development

Testing

Testing Needed

Evaluation

Testing

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

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

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