­National College of Ireland

Higher Diploma in Science in Mobile Technologies

2014/2015

Witold Zolnowski

x17143853

Crypto Scanner

Technical Report

**Table of Contents**

[Executive Summary 3](#_Toc351559315)

[1 Introduction 3](#_Toc351559316)

[1.1 Background 3](#_Toc351559317)

[1.2 Aims 3](#_Toc351559318)

[1.3 Technologies 3](#_Toc351559319)

[1.4 Structure 3](#_Toc351559320)

[2 System 3](#_Toc351559321)

[2.1 Requirements 3](#_Toc351559322)

[2.1.1 Functional requirements 3](#_Toc351559323)

[2.1.2 Data requirements 3](#_Toc351559324)

[2.1.3 User requirements 3](#_Toc351559325)

[2.1.4 Environmental requirements 3](#_Toc351559326)

[2.1.5 Usability requirements 3](#_Toc351559327)

[2.2 Design and Architecture 3](#_Toc351559328)

[2.3 Implementation 3](#_Toc351559329)

[2.4 Testing 3](#_Toc351559330)

[2.5 Graphical User Interface (GUI) Layout 3](#_Toc351559331)

[2.6 Customer testing 3](#_Toc351559332)

[2.7 Evaluation 3](#_Toc351559333)

[3 Conclusions 3](#_Toc351559334)

[4 Further development or research 3](#_Toc351559335)

[5 References 3](#_Toc351559336)

[6 Appendix 3](#_Toc351559337)

[6.1 Project Proposal 3](#_Toc351559338)

[6.2 Project Plan 3](#_Toc351559339)

[6.3 Requirement Specification 3](#_Toc351559340)

[6.4 Monthly Journal 3](#_Toc351559341)

[6.5 Other Material Used 3](#_Toc351559342)

# Executive Summary

This project aims to solve a problem of multiplicity of cryptocurrency markets and cryptocurrencies, which makes finding arbitrage opportunities manually impractical. Proposed application leverages asynchronous nature of JavaScript language in order to make multiple https requests to cryptocurrency markets in a non-blocking, and hence, efficient manner. Thus obtained market data is fed into a modified version of a Bellman-Ford algorithm in order to identify arbitrage opportunities. Results are presented to a user via GUI built on top of chromium browser, using web technologies. Routes between front, and back-end are managed by Electron package. Initial tests indicate that application works as intended.

# Introduction

This template for technical report is provided for your convenience. It should be seen as a guide rather than an obligatory form. Your individual report might require changes in terms of format or content (i.e., headings) or both.

Print on one side of the paper only (this will be the right hand side when the pages are bound).

In the last couple years cryptocurrency markets have exploded. Cryptos are lauded as new ‘internet’ in terms of impact they will have on our lives. With new technologies, come new opportunities, as well as, dangers. Political, and economical issues aside, it remains to be seen whether cryptocurrencies can deliver on their, sometimes wild and expansive, promises. Majority of cryptocurrencies are still in the incubator state, with very little to offer. That did not stop investors from pumping large amounts of money into currencies that have little to no intrinsic value, in hopes of extraordinary returns, as was the case with the Bitcoin. Because there is no fundamental value behind cryptos, their prices are very volatile. Generally volatility is not desirable to the investors, but every coin [sic] has two sides. Volatility creates price differences between markets that trade the same asset. Quick and often moves in price of an asset creates arbitrage opportunities, meaning that it’s possible to buy asset low at one market, and sell it quickly at a different market for a small profit.

## Background

As of today, there are 1754 cryptocurrencies, and 12591 crypto markets listed on the CoinMarketCap site, which aggregates, and tracks crypto prices. Given the number of markets, and assets being traded on them, it’s impossible to keep track of all the price movements manually. In order to identify the price differences, and arbitrage opportunities, an application is needed. The author of this paper has a personal stake in a crypto space, and has been involved in arbitrage trading for over a year. Therefore, the crypto scanner application, addresses real world need of the author for a tool that can automate a lot of manual labour that was necessary up to now.

## Aims

The aim of this project is to create an application that can obtain data from cryptocurrency markets, without any need to resort to third party API’s. Further, the application should be able to analyse the data, and present in a clear way results of the analysis. Results should indicate what the potential returns are for a given ‘path’, as well as, a number of transactions required to finalize the ‘path’. Finally, the whole process should be performed quickly ie. Sub 30 seconds.

## Technologies

The backbone of the application is **Node.js**, which is a runtime environment for the JavaScript. Normally, JavaScript can only operate within browser environment, which relegated this language to only Front-End applications. With the onset of Node.js, it is possible to use JavaScript in Back-End logic. Because the application is written in only one language, the whole codebase is simplified and more streamlined as a result ( at least in theory ).

Node.js supports ‘packages’ which offer ‘out of the box’ solutions, similar to Java’s .jar files, or Ruby’s gems. The main package used for the development of this application, is the **Electron** package. Due to its size, and impact it has on the way application is developed, it is often referred to as a framework.

Electron is predominantly used for desktop application development. Generally, developing GUI’s for the desktop app’s has steep learning curve, and requires specialized tools. Electron simplifies the whole process by providing GUI environment within a built-in Chromium browser. Browser based GUI is far simpler to work with, because developer can simply use familiar, web based tools, such as HTML /CSS / JS. Further, Electron provides built in channels for communication between Front and Back-End.

Oher significant packages used are: axios, and graph data structures. Former provides simpler, and therefore, more readable syntax for making https requests. Latter package is used for implementing graph building algorithm.

The whole application was developed with the mind to limit the number of dependencies to minimum. Author attempted to achieve this by relying on Node.js’ built in functionalities instead of packages. Same approach has been taken when developing Front-End, where author has decided not to use any of the frameworks, such as Bootstrap or jQuery. Therefore, whole application is written in ‘vanilla’ JavaScript / HTML / CSS.

Application has been developed on Linux Mint operating system, but because it is an Electron app, it can be repackaged to work on any other OS. For example, demo of this application will be presented on a Mac.

# System

## Requirements

### Functional requirements

Primary functional requirement is that the application can analyse data, and identify existing arbitrage opportunities.

### Data requirements

Algorithm must be able to operate on a dataset that’s in JSON format.

### User requirements

User should be able to filter the results section by the number of ‘jumps’ or minimal required return.

### Environmental requirements

Application should work without prior installation of the app itself or any other software (other than the essential OS software).

### Usability requirements

Application should be easy, and intuitive to use. Buttons and input fields should be big, and clearly labelled. Main functionalities should be available on a single page.

## Design and Architecture

System can be best described as two modules. First module performs https requests and fetches data from markets. Once the data is fetched, it is formatted and aggregated. Second module takes the pre-formatted data as an input, and outputs multidigraph ( directed graph with multiple edges between vertices ). Once the graph is created, an algorithm traverses it in order to find negative cycles. If found, results are saved, and passed over to the renderer process ( Front-End ) where they are formatted, and diplayed for user.

Main algorithm used is a modified version of the Bellman-Ford algorithm. It’s main premise is to ‘relax’ all the edges n-1 times ( n being number of nodes ), similar to Dijkstra’s algorithm. Once the ‘relaxation’ is complete, main assumption is that there no further optimization of the paths can be achieved. Therefore, if we ‘relax’ all the edges one more time, and if any of them improves, that means we have a negative cycle present. Negative cycles imply that there is no ‘shortes path’, because negative cycle can be traversed infinite amount of times, each time improving the path. Further, and more importantly to this application, negative cycles signify that if we traverse the negative cycle completely, we will end up with more cryptocurrency than we had before we set out.

## Implementation

Two main functions of this application are : getPrices(), buildGraph(), and bellmanFord(). First function dispatches and coordinates https requests made to markets, and once the data returns, it formats the data so that it can be aggregated. This function can be found in the util.js file. Second function takes in fetched market data, and constructs a multidigraph. It also adds ‘Root’ node to the graph with directional vertices pointing towards every node. All edges coming from ‘Root’ node have weight of 0, in order to not affect the calculations. Last function controls the arbitrage algorithm.

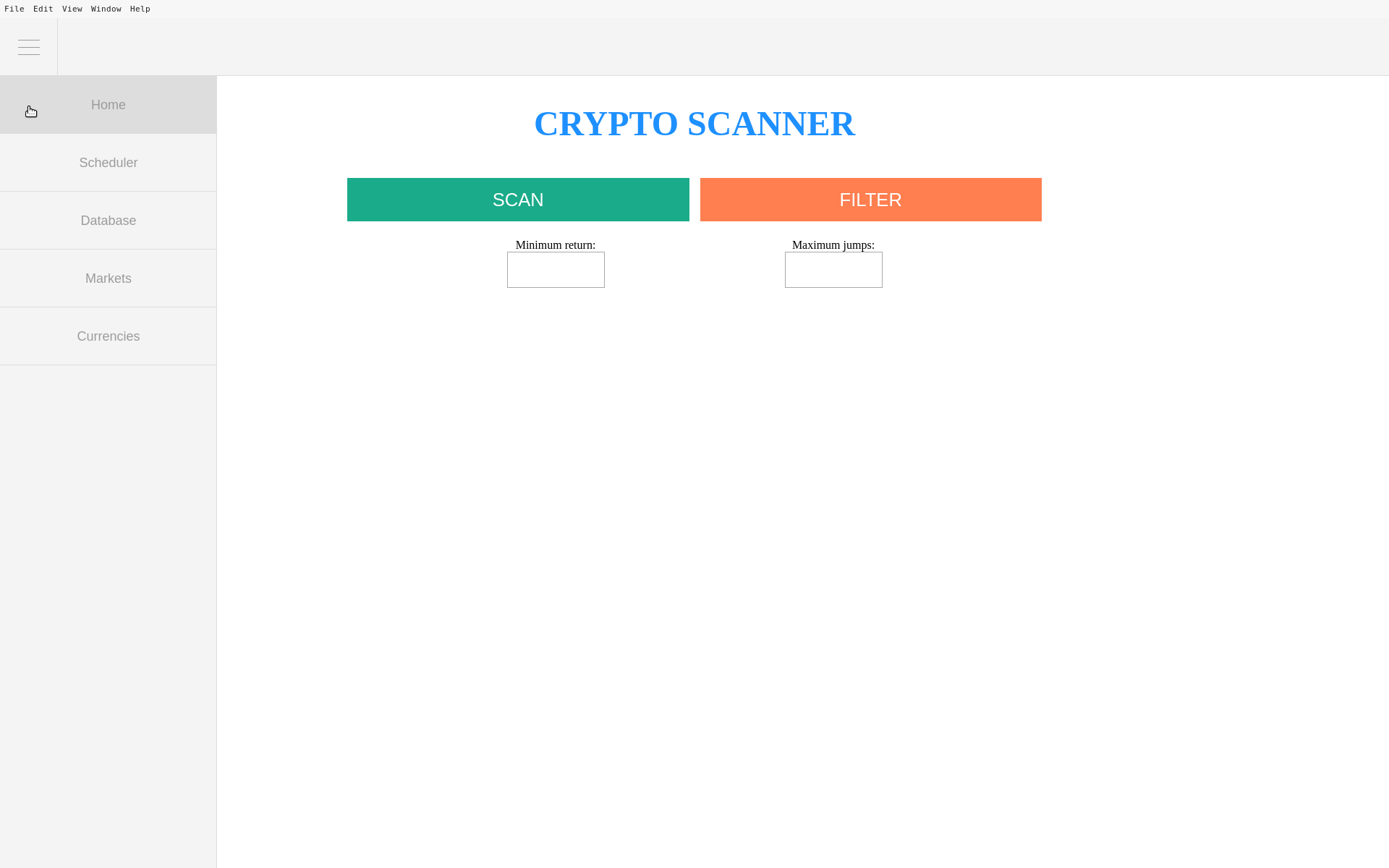
## Testing

Functions throughout the application have been unit tested. Mocha and Chai packages were used to perform the unit tests.

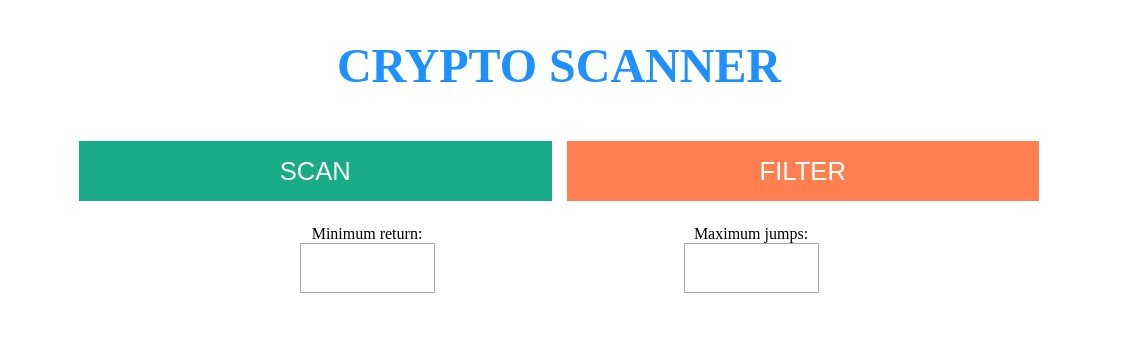
## Graphical User Interface (GUI) Layout

**Main page:**

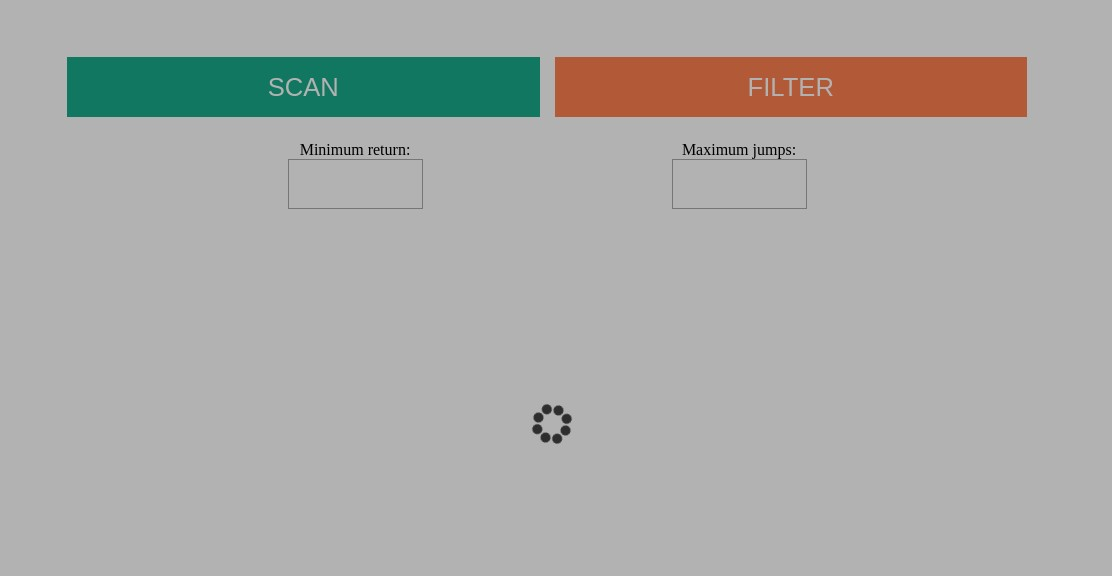
To the left there are links to various sub pages ( to be develped ).



**Buttons**: The ‘Scan’ button start the algorithm. Button ‘Filter’ is used to filter the results by minimum return ( expressed in terms of % ), and by maximum allowed ‘jumps’.



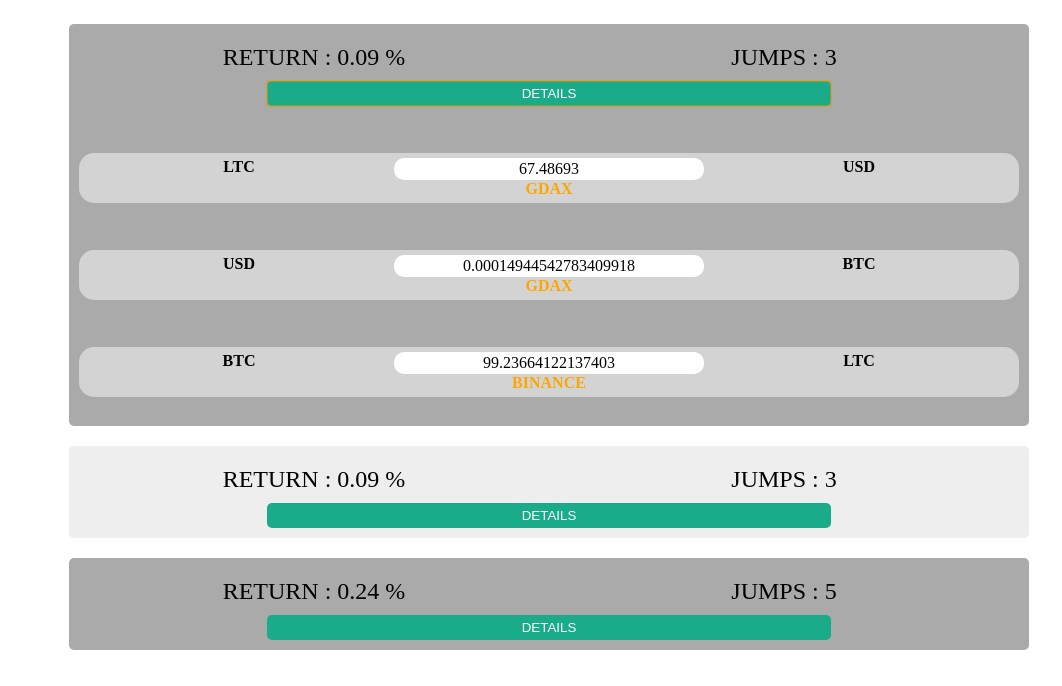
**Loading**: while the scan is being performed the loading screen is display and buttons are deactivated until scan ends.



**Return data**: once the scan ends and arbitrage opportunities are found, they will be displayed beneath the buttons on the main page.



**Results details:** results have button that if clicked displays additional details:



# Conclusions

# Further development or research

With more resources, where could the results of this project lead to?

# References

Students should use the Harvard referencing style.

Please consult the CITE@NCI handbook available in the Library.

# Appendix

Attach all your partial submissions as appendices.

## Project Proposal

## Project Plan

## Requirements Specification

## Project Analysis & Design

## Project Test Plans

## Monthly Journals

### Monthly Journal #1

### Monthly Journal #2

### Monthly Journal #3

## Other Material Used

Any other reference material used in the project for example evaluation surveys etc.

**NOTEs:**

* **The report has to be printed in 2 hard/soft bound copies and submitted to the School Office.**
* **CDs containing code should be glued to the technical report.**
* **Report and code will also be submitted through Moodle.**