

City University London

BSc (Hons) Computer Science with Games Technology

Final Year Project Report

Academic year: 2017 – 2018

CRYPTO CURRENCY TRADING PLATFORM

By Fedil Al-Hayawi

Project Supervisor: Andrey Povyakalo

CHAPTER 1

INTRODUCTION

1.1 Report Overview

The website developed in this project aims to provide for a lack in the market, more specifically the Crypto Currency market. The platform offers a way to purchase Crypto Currencies with cash without many of the disadvantages other purchasing options have. It facilitates a trade of two currencies, GBP cash to CC (Crypto Currency) where the user makes the cash payment to a local vendor, such as a convenience store and receives CC short after the vendor confirm receipt of the payment.

The solution outlined in the PDD (Appendix A) was a mobile application, but during development it was modified to a node.js and mongoDB website to provide a platform for 'consumers' to purchase Crypto Currencies. This is especially advantageous as the Bitcoin Core library is commonly used, and very powerful library. In this document I will discuss the process of developing the website's front and back-end, and how it can be used to facilitate CC's purchases with Cash.

1.2 - Description of the Problem

In 2017/2018 there are 3 main ways for the public to purchase CC's. Online platforms that require thorough identification and have high fee's. You can find vendors on platforms like LocalBitcoins who will meet you to exchange cash for CC's. Or via debit/credit card and usually requiring identification in forms of a passport picture, proof of address and more obscure ways, like reading a paragraph stating the intent of the purchase on camera, and sending the video to the vendor. The other way to purchase CC's is via ATM's. ATM's will without doubt become cheaper and more widely available. But currently they are overpriced. For anyone serious about making a profit from crypto, an 8% cut off your funds is not an option.

The platform developed has several advantages in comparison to the options mentioned:

	Price	Requires ID	Availability	Safety
Online Platform (eg. Coinbase, gemini)	↓	↓	↑	↑
Person-To-Person	↑	↓	↑	↓
ATM	↓	↑	↓	↑
Network of local vendors	↑	↑	↑	↑

1.3 – Project Objectives

Objective	Progress review
Develop a website to be used by the public and vendors(admins).	The website developed allows for users to register, login, place and view orders. Automatic emails are sent on creation of an account, and can be resent for the password recovery. The account creation form fields get checked and show the corresponding error to the user. It also checks if the username selected already exists in the Database collection. Passwords are hashed and stored securely in the database.
NodeJS Backend with Express library	<p>The backend uses of the async oriented JavaScript engine NodeJS and the Express library. Using NodeJS was different to other programming languages in the way it uses call-back's for async calls. Express is built on the principle of piped function calls as well.</p> <p>The backend was developed with help of a basic account management system hosted on GitHub. It builds the foundation of usability of the website. The library makes use of Node.js, Express.js, MongoDB, and EmailJS.</p> <p>Additionally, the backend uses CoinDesk API to fetch the current Bitcoin price every couple seconds.</p>
Jade (renamed to Pug) Template Engine	<p>Jade allowed for smooth integration with a basic website template used called 'Massively' by HTML5 UP (Massively, 2018). Jade takes the place of HTML and adds additional features such as easy access to variables passed on by the backend.</p> <p>It does come with some flaws. The language interpreter was incredibly sensitive, and in combination with in-line 'includes', it created a lot of technical difficulties.</p>
Frontend website design and Google Maps API for the vendor map	The website design is modern, but very simple in execution. As a prototype, proof-of-concept, website it only offers the necessities to provide user and vendor interaction. Once the user has logged into their account, they will be redirected to the 'purchase' page. Here the user can search for their address, and locate the vendors highlighted on the map. The 'orders' page shows the user previous orders placed and basic information about it. If the user is logged in with an admin account, they see all orders that have not yet been confirmed. They can also confirm receipt of a payment.
MongoDB (noSQL) Database Backend	Learning to use noSQL databases was an interesting part of the project and I found it to be more intuitive than SQL database solutions.

1.4 – Anticipated Beneficiaries and benefit

The beneficiaries and benefits are:

- The public can purchase CC's cheaply and securely.
- A selected network of vendors will gain an additional source of income
- Wider Crypto Currency adoption.

In order for Bitcoin and other CC's to become more widely accepted, it has to become more available. This platform has the potential to make other purchasing options obsolete and could help with adoption.

1.5 – Assumptions and scope

The scope of the project is limited to accepting orders from a user and awaiting the receipt of payment. The development of the bitcoin integration was left out to allow for more development time with the actual website.

There are several assumptions made:

- The vendors can uniquely identify the order placed by the users.
- The vendors can safely accept cash, without risking loss of funds or fraud.
- The website is not under attack of hackers.
- Vendors will be interested in participating in the network, given a reward in form of commission.

The original aims described in the PDD were too widespread and not feasible. The output of my project is an alternative that fulfils the original need for the platform, but makes cuts to produce a working prototype, above all else.

CHAPTER 2

OUTPUT SUMMARY

2.1 – Crypto currency trading website

The main output of the project was the website developed. It makes use of 5 programming languages, noSQL, and Bitcoin API integration with node.js.

The main ('home') page has the login and register forms. The login will redirect to the 'purchase' page, where users can enter a GBP or BTC value, and display the converted currency. This page includes a google map to enable the users to search for their location, and to show the locations of vendors.

The orders tab differs for regular users and admins. Regular users see a list of all their orders: The amount purchased, the rate, date and whether it has been paid for.

Admins see a list of all open orders in the database, with the ability to confirm orders.

Users can place an order for BTC and locate a vendor on the map to make the payment to. Once the vendors confirm receipt of the payment, it will be released automatically, and should shortly be in the users wallet.

This project uses several libraries. Around 3000 lines of code were written by me, and around 2000 lines of code were reused from libraries and public answer boards such as "StackOverflow". Most of code is in JavaScript.

The website is to be used to purchase Bitcoins using cash. Payment are made to local vendor who accept cash and the system releases the Bitcoins automatically after the payment was made. Partnering vendors will benefit from increased traffic and an additional income stream in form of commission from Bitcoin sales.

CHAPTER 3

LITERATURE REVIEW

3.1 – Node login

Most of the technologies used were dictated by a template called “Node-login”. (Node Login, 2018)

A significant amount of time was invested into learning the libraries used in the backend. Learning node.js was a challenge coming from more traditional languages such as C++ and C#, or Java.

A great resource was “The Complete Node.js Developer Course (2nd Edition)”, a course on Udemy that makes use of Node, Express and MongoDB. (Node.js Udemy, 2017) 3 technologies utilized heavily by the used template. In addition, SocketIO is part of the course. An event-based communication engine with an excellent implementation in JavaScript. Implementing features like auto-scrolling or maps based geolocation and many others is made incredibly ease with SocketIO, and is well explained in the course. It also touches on a topic discussed later on cyber security, or security in general.

Another good resource used a set of teaching modules hosted on npm, package manager. “How-to-npm” is accessed with the command “npm I -g how-to-npm” and bring up a module that guides you through a topic.

Many different resources can be found online to learn node.js. One of the best ways to get a feel for a language is to learn the basics and start writing code. “Node.js Tutorial for Beginners: Learn Node in 1 Hour” is a good crash course, correctly emphasising Node being ideal for I/O-intensive apps, i.e. website with frequent and complex database queries. (Node.js Tutorial, 2018). The video has been uploaded in 2018 and covers some of the newer features of JS and Node.js such as Async, which is a new method of writing asynchronous code that doesn’t require call-backs.

3.2 – noSQL

MongoDB was easy to get started with. Setting up the environment, creating a collection, inserting, looking up and the various other methods are thoroughly explained in “MongoDB: The Definitive Guide”.

(Chodorow, K., 2013) noSQL shows its real strengths with multi-dimensional scaling, because of the flexibility and strength of the query language. For a project of this size, MongoDB can handle most operations with 3 operations. Find() and FindOne(), Update and UpdateOne, and Insert() and InsertOne(). Therefore, I decided use only *MongoDB: The Definitive Guide* to fill the gaps and occasionally look up documentation.

3.3 – Cyber Security

As large amounts of funds might be involved when dealing with Crypto Currencies’, it is important to securely transfer, store and display user data. During implementation of the database I followed several

guides to ensure use of the best practices. The mongoDB docs contains guides on security, including a security checklist that proved itself very useful. (Security Checklist, 2017)

I found out about Salted Challenge Response Authentication Mechanism (SCRAM) using the documentation and implemented it. Passwords stored in the database are ‘salted’, or hashed using a magic number, hidden to users. (MongoDB SCRAM, 2018)

The database was configured using TLS, as per the checklist mentioned. This ensures that messages sent over the network are encrypted. This would be essential in the use case of a large corporation. (MongoDB SSL, 2018)

3.4 – Crypto Currency

As I have a lot of prior knowledge about Blockchain technology, I was able to get started on development after reading the BitCore guide on setting up your own “Full Node. (BitCore Node, 2018) In the end it was decided against BitCore and running a full node, and instead the Bitcoin transactions are made using 3rd party services with API support for broadcasting Bitcoin transactions.

CHAPTER 4

METHOD

This section will describe the work undertaken in the analysis, method, implementation and evaluation stages. Each chapter covers the work undertaken to overcome a certain problem.

4.1 – Agile Development Methodology

During development of this project, requirements were constantly changing as new problems would come up. This is primarily why agile approach was used. Development began with very little preparation, as the beginning stages were explorative in nature. For instance, at the inception of the project, C# was used to access the database and handle the Bitcoin transactions. The idea was to run a server client checking the database for outstanding transactions and execute them. C# has a great implementation of mongoDB and it's LINQ query language works flawlessly with the driver for mongoDB and several Bitcoin API's available on the Visual Studio Extensions Store. Though the same can be said about node.js, and given that the backend already uses node.js, it was the right decision to implement everything in one place.

As I have outlined, it's clear how an agile software development approach is advantageous as it emphasises and embraces change.

4.2 – Initial research and analysis

To kickstart the project an appropriate template had to be used. It had to be open source and it in this case under the MIT License. This means it can be used commercially and privately given the copyright stays in the source files. Of course, many other considerations were made when choosing the templates used.

'Node Login' template:

- Well documented source code.
- Rich in features. This includes a cookie manager, session tracking, email sending and receiving, encryption and TLS support.
- Utilizes noSQL database.
- No errors or significant problems on first attempt. The entire code base was up-to-date and following async coding guidelines.

These are some of the criteria when searching for a good starting point.

On the scope of the Crypto Currency market, this project offers an elegant solution, that has only been implemented in North America by a company called 'LibertyX'. This was a starting point in the research made and how the solution provided fits and compares to the one company that offers a similar service.

4.3 – Design

4.3.1 – Agile methodology

Development was incremental and agile. The general method is as follows:

- Define requirements
- Develop, integrate and test
- Deploy, adjust and iterate

4.3.2 – Proof of Concept

The first working outcome was when the ‘Node Login’ template was modified enough to include the design from the HTML5 template ‘Massively’. This was an iterative process, and a slow one at that, as the HTML code had to be translated to the Jade templating language. Nonetheless, it worked out perfectly in the end and made for a good look once the background image was changed.

4.4 – Implementation

4.4.1 – Development environment

Visual studio code was used to write all the code produced. It can be configured extensively and has features such as auto-aligning code. VSC is useful when dealing with cross-language programming. A variable declared on the backend will be highlighted with the correct syntax on the jade front-end. Especially for website development this editor is well suited.

4.4.2 – Programming languages

The technologies used in this project maximise the entry barrier to develop a proof of concept in a short period of time. The programming languages chosen reflect that.

Node.js was the preferred choice for good mongoDB integration and synergy with Jade. The project makes use of various standard and 3rd party libraries, all made accessible with the npm (Node Package Manager), including learning material for node.js. A simple “Git Clone” and “npm install” installs a new module, and a “require ..” call imports it into the application.

4.4.3 – Database

In deciding on a database system security, speed and ease-of-use were taken into consideration. MongoDB with its collections, documents, fields and nested fields gives the programmer the ability to reuse code, across languages. Learning mongoDB queries was intuitive and well works in conjunction with async and event driven programming language node.js.

In comparison with mySQL, it seems SQL queries generally require 1 more step to perform any meaningful operation. It was my first experience with noSQL databases, and it has shown to be very effective.

4.4.6 – Bitcoin API

Many of the complexities of the Bitcoin integration could be avoided using the “bitcoin-transaction” library. (Bitcoin-Transaction, 2018) This is a very light-weight, 1 file library that implements the API’s of CryptoCypher and Blockexplorer to broadcast transactions to the Blockchain.

Reading through the documentation and code helped simplify what would have otherwise been a tedious and time intensive task.

4.5 – Evaluation

4.5.1 – Usability Testing/Use-case diagrams

To manage, track and visualize the necessary use-cases it was decided to make the relevant use-case diagrams. These were made using the basic layout used throughout the course. It contains a brief description of the use case, actors, preconditions, basic flow, alternative flow and postconditions.

The basic flow consists of a bullet pointed list in the format of “- System displays the home page” and “- User submits login form”. Alternative Flow is similar to basic flow but covers other possible paths the user can take. Postconditions briefly explains the desired state as the result. (Appendix C)

4.5.2 – Testing

Testing the website was easy at first, before it became clear that testing on a singular computer, browser and operating system was not reliable.

To overcome these limitations, I set-up virtual environments with different variations and combinations of operation systems browsers and versions of browsers.

- 2 Windows 10 machines with the latest Chrome/Firefox/Opera/Edge browsers.
- 2 Windows 7 machines with latest stable Chrome/Firefox/Opera/Edge browsers.
- 2 MacOS machines with the latest Chrome/Firefox/Safari browsers.

Additionally, tests were performed using a OnePlus 3T Android phone, and emulated a mobile iPhone experience with an online emulator. (AIR iPhone, 2017)

4.5.3 – Cross-platform testing

As described above, there are many different variations of combinations. There are tablets, small outdated phones, laptops, and high-end phones.

Testing was performed as part of the iteration process, but it was not feasible to test everything, after every code change. The method used was based around quickly producing results, and solving multiple compatibility errors simultaneously at the end of every major iteration.

In practice, a full coverage of platforms, browsers and systems was undertaken a few times during development to ensure the majority of the work is in developing new features.

CHAPTER 5

RESULTS

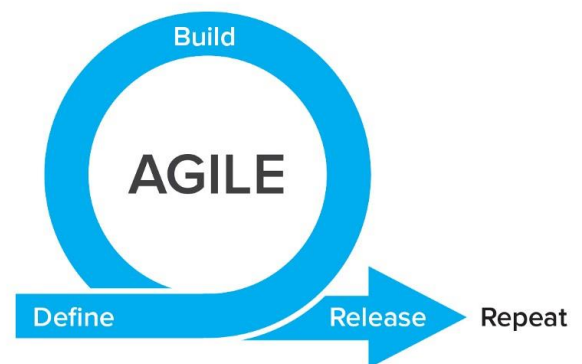
This chapter covers in more depth the results of the project, including the outcomes of methods discussed in Chapter 4 and in the Outputs Summary in Chapter 2.

5.1 – Agile Development Methodology

Development was mostly loose, guided by a 3-step process with frequent redefinitions of requirements. (Agile, 2017)

Define:

- Often a feature will be implemented as the result of finishing another one. In practice, this is done in the code by creating a function or class and defining, with comments, what needs to be achieved, how it will be done, and what the outcome should be.
- Otherwise a feature will be derived from a use-case.
- This process usually occurs every 1-2 days, when a feature or part of a feature is complete.



Build:

- This is where most time will be spent, in order to maximise the output.
- Previous comments are replaced with the code to perform the task.
- Larger tasks are split up into smaller tasks.
- Small sessions of testing are mixed into this process.

Release:

- The new code is pushed and committed to GIT.

5.1.1 – Tools used

Several extensions were used with Visual Studio Code:

Auto Close Tag: This tool automatically adds the closing tag to HTML and XML code.

HTML Snippets: Adds language supports, useful code snippets and colorization.

Blank line at end of file: Self-explanatory. Adds a blank line at the end of every file.

Path Intelligence: Highlights and autocompletes filenames.

Other tools used:

GIMP: Image editing software used for editing the background image and others.

Microsoft Word: To keep track of progress and outstanding work.

Google Keep: Synchronous notes for idea's and thoughts across devices.

5.2 – Analysis and requirements

5.2.1 – The competitor “LibertyX”

Originally known as Liberty Teller, and founded in 2014, the company was a involved in an incubator program. In the early stages, the company made headlines for releasing America's first bitcoin ATM. In December 2014, the cash-to-bitcoin exchange was introduced.

LibertyX allows its users to buy bitcoin in one of the over 19 thousand stores, located in North America.



5.2.2 – High level requirements gathering

The requirements are best documented in the high-level requirements table produced. (Appendix D)

This document covers the vast majority of features required to enable a user to place an order and purchase Bitcoin. The Implementation chapter covers the solutions found to the requirements in detail.

The requirements are documented in this format:

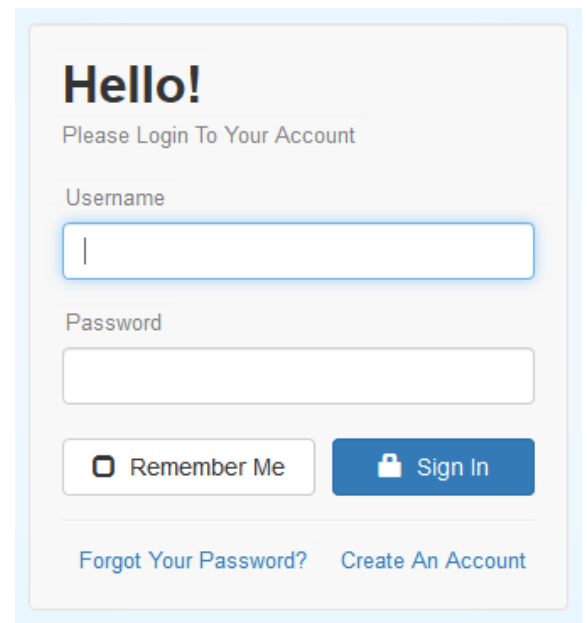
Requirement	Description
Setup MongoDB database	In order to prove that mongoDB is a suitable database for this project, it is the first part to be developed.
Implements node.js template "Node Login"	The Node Login template will form the infrastructure for the website. It is therefore of utmost importance to clean the code, and strip all unnecessary or deprecated code from the code base. The result should be to have a clear understanding of the mechanisms behind the code.
Find suitable HTML 5 template to integrate with the node.js backend	It was decided to opt for a template instead of writing the front-end from scratch. This is due to the project being largely a proof-of-concept, and process should be made quickly.
Bitcoin Blockchain API integration	A suitable node.js implementation of the bitcoin protocol layer must be found. Most likely, this will be the Blockchain " <u>bitcore</u> " library on GitHub.
Bitcoin Price API integration	The backend must fetch the price of 1 BTC every few seconds to ensure that users can't abuse price fluctuations.
Indicate guest/logged-in user with a top bar	Permanently top fixed bar to greet visitors and show a link to the log-in form. It must be responsive and have different content depending on: <ul style="list-style-type: none"> • Is the user logged in? • Is the user an admin? The website should indicate these things.
Securely store and transfer sensitive information	Security guide lines must be followed to ensure the application does not have obvious security leaks. When transferring information between the backend and database, the password must be hashed or in other ways obfuscated. Sensitive information may not be displayed or accessed on or from the front-end.

5.3 – Design

5.3.1 – Proof of Concept

The foundation built by Node Login paved the way for full integration of the features provided. (Node Login, 2013) The early proof of concept already included the “Forgot Your Password?” feature. It allows users to enter their email address to recover the password by clicking on a link sent to their email that gives instructions.

“Create An Account” links to the signup form which was initially on a different page. The signup form has the functionality of sending a confirmation email to the users email address. The node.js library includes a module called “email-dispatcher.js” to sending emails, that was extensively used throughout.



Hello!
Please Login To Your Account

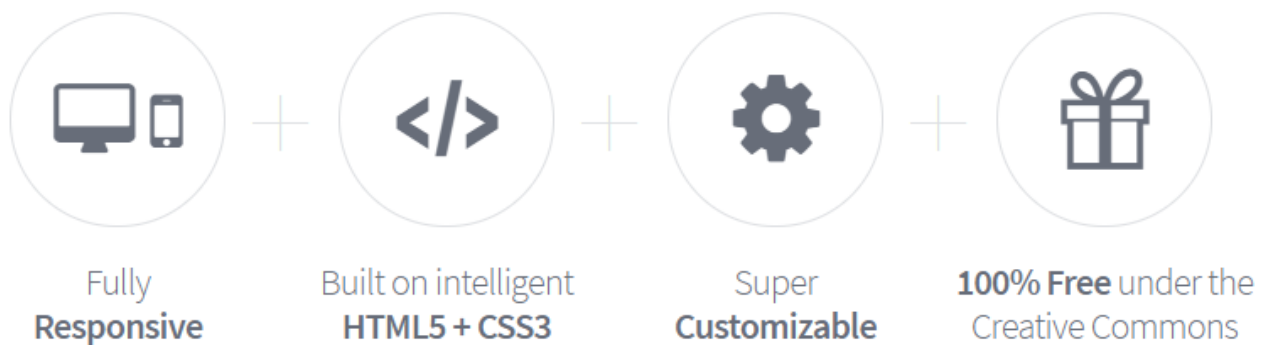
Username

Password

☐ Remember Me

[Forgot Your Password?](#) [Create An Account](#)

5.3.2 – UI Design



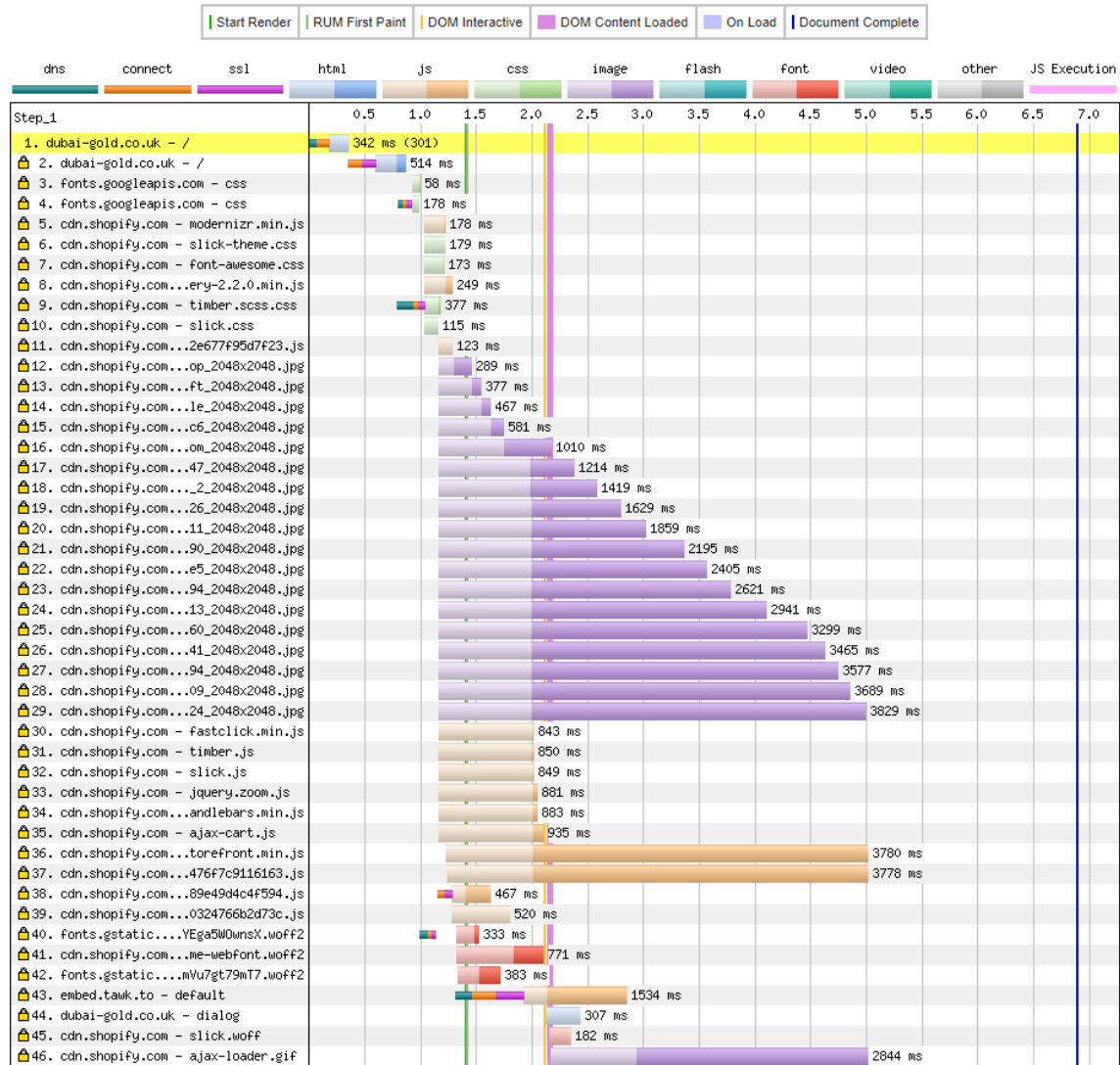
HTML5UP designs were licensed under the Creative Commons Attribution 3.0 meaning:

- Free for personal use
- Free for commercial use
- Free to modify the source files

After vigorous testing it was decided on the Massively template, a fully responsive HTML5 and CSS3 template with an attractive look.

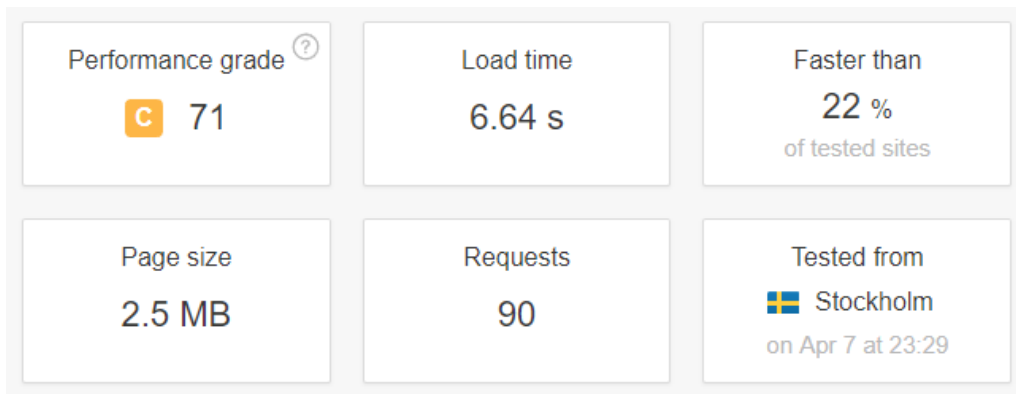
To ensure the website will perform well various tests were executed. Some of the tests are available online and work simply by typing your URL.

- webpagetest.org provides a complete breakdown of the files downloaded by a user.



In this example you can distinguish between the images and other files downloaded and come the correct conclusion that there is an excess of images being downloaded.

- tools.pingdom.com provides a detailed breakdown of possible improvements:



This is the overview giving information on some important information. Load time, page size and Performance grade are the most important for this project.

The performance grade is a Google PageSpeed Insights Score, indicating the performance of the website in comparison to other websites.

Performance insights			
GRADE		SUGGESTION	
F	0	Combine external JavaScript	▼
F	0	Parallelize downloads across hostnames	▼
F	36	Remove query strings from static resources	▼
D	66	Combine external CSS	▼
C	78	Leverage browser caching	▼
B	83	Minimize redirects	▼
A	96	Specify a Vary: Accept-Encoding header	▼
A	96	Minimize DNS lookups	▼
A	96	Serve static content from a cookieless domain	▼
A	96	Specify a cache validator	▼
A	99	Minimize request size	▼
A	100	Avoid bad requests	▼

Each of these points covers an improvement that could be made. Some of the more severe notices is very difficult to fix but can be safely ignored. This is due to modern browsers allowing for errors and alerts without impacting the users experience. For instance, the Google.com search will show errors in the console.

5.4 – Implementation

This section aims to give an in depth look into how different solutions we're implemented. There will be a discussion about how well the solution satisfies the gathered requirements discussed in 5.2.2 – High level requirements gathering. (Appendix D)

5.4.1 – Development Environment

Visual Studio Code was the preferred code editor for this project. A few key features lead to that decision:

Look and Feel

VSC (Visual Studio Code) offers a variety of themes, with several dark options as well. The editor is fast and doesn't clog when many files are open simultaneously.

A great in-built feature in VSC is code formatting with the Shift + Alt + F shortcut organising and aligning code.

When working with open-source libraries often minified versions of files are distributed. Using the code formatting functionality will make the code readable again.

```
// Purchase //
app.get('/purchase', function (req, res) {
  console.log("GET PURCHASE");
  res.render('purchase', {
    title : 'Control Panel',
    countries : CT,
    udata : req.session.user
  });
});
```

Extensions

VSC hosts a large library of extensions some of which are little tools that make repetitive tasks redundant, highlight and autofill code and generally enhance a programmer's experience.

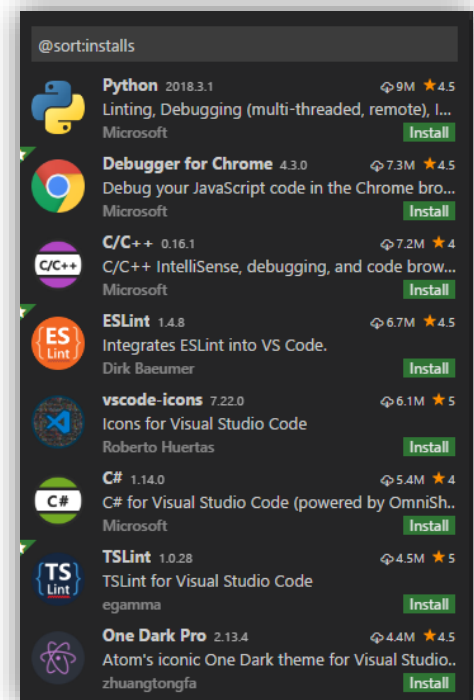
Some especially useful extensions are discussed below:

Auto Close Tag – This extension works with JavaScript CSS and HTML. It helps in placing closing tags automatically and cross-checks class names and definitions. For instance, in the HTML file auto-completion will consider the imported CSS file, and suggest those definitions.

HTML Snippets – In the same spirit as Auto Close Tags, this extension offers shortcuts and templates that are helpful in coding.

One major problem with extensions is that you can become reliant on them, making your work place limited to your personal computer. Ideally, you could program and compile the website with any file editor.

Other extensions used are:



Git History – While this extension can be replaced with TortoiseGit, it does offer a neat view of the commit history. It has similar before-and-after code comparison functionality, and works flawless with TortoiseGit.

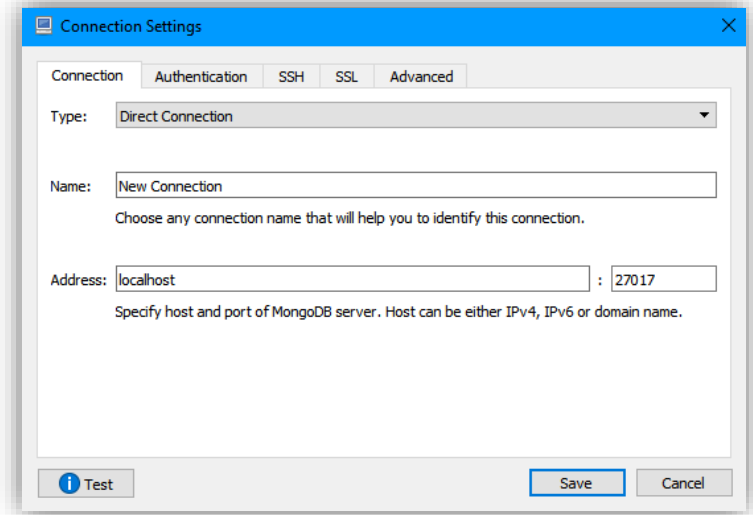
5.4.2 – MongoDB

Robot 3T was the program used to create the database. It is a cross-platform MongoDB management tool, providing a GUI view of the database and allowing you to modify entries.

Creating a database:

Simple tasks are made even easier with this tool. Filling out the Connection Settings setup and clicking Save will create the database for you.

The only pre-requisite to using Robo 3T is to start the mongoDB instance by running “mongod.exe” or in our case “mongos.exe” for the TLS version.



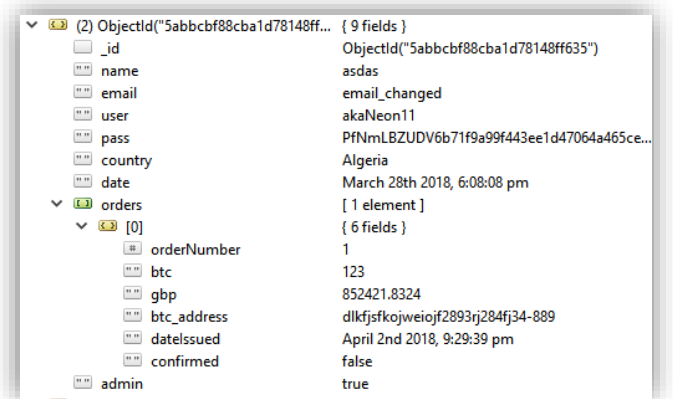
Storing User Information:

The contents stored in the database was highly flexible, but the basics are implemented.

The orders field is an array containing 0, 1 or many orders.

Orders contain the following information:

- orderNumber – A unique order number specific to this user.
- btc – The bitcoin amount purchased.
- gbp – The GBP to be paid.
- btc_address – The Purchase page requires the “Bitcoin Address” field to be filled in, and is stored in this field.
- dateIssued – Record the date the order was placed?
- confirmed – indicated whether or not the order has been paid for.



5.4.3 – Admin privileges

Admins in the systems are the vendors who can confirm receipt of a cash payment. They are presented with this view on the Orders page.

ORDERS [ADMIN]

USERNAME	EMAIL	ORDER NUMBER	BTC	GBP	DATE ISSUED	CONFIRM
akaNeon22	email_changed2	1	1	9754.945	May 4th 2018, 9:26:14 am	<input type="radio"/>

SUBMIT

The admin can see username and email of the user to identify them. Using the confirm radiobox and submitting the form will initiate a Bitcoin transaction to be broadcast to the network.

5.4.3 – Programming languages and technologies

This section outlines how the different programming languages and technologies relate to each other, and how they we’re integrated.

“Api.js” is the entry point to the application. At beginning 18 lines are imports and declarations for other libraries.

This is node.js and not regular JavaScript code. The syntax has significant differences.

App.set('view engine', 'jade');

This line declares that jade files shall be loaded instead of HTML files. The jade files location is

specified for the “views” field of the express variable “app”.

```
var MongoDB = require('mongodb').Db;
```

This line loads the “mongodb” library. The mongoDB integration for node.js supports call-back based interaction.

At the end of jade files, as HTML files, is the list of script required for compilation. For the “Home” page jade file:

```
var http = require('http');
var express = require('express');
var session = require('express-session');
var bodyParser = require('body-parser');
var errorHandler = require('errorhandler');
var cookieParser = require('cookie-parser');
var MongoStore = require('connect-mongo')(session);
var app = express();

app.locals.pretty = true;
app.set('port', process.env.PORT || 3000);
app.set('views', __dirname + '/app/server/views');
app.set('view engine', 'jade');
app.use(cookieParser());
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));
app.use(require('stylus').middleware({ src: __dirname + '/app/public' }));
app.use(express.static(__dirname + '/app/public'));
```

```
// Scripts

block scripts
script(src='/newjs/jquery.min.js')
script(src='/newjs/jquery.form.min.js')
script(src='/newjs/jquery.scrolling.min.js')
script(src='/newjs/jquery.scrollTo.min.js')

script(src='/newjs/bootstrap.min.js')
script(src='/newjs/skel.min.js')
script(src='/newjs/util.js')
script(src='/newjs/main.js')
script(src='/newjs/socket.io.js')
script(src='/js/controllers/loginController.js')
script(src='/js/form-validators/loginValidator.js')
script(src='/js/form-validators/emailValidator.js')
script(src='/js/controllers/signupController.js')
script(src='/js/form-validators/accountValidator.js')
script(src='/js/views/signup.js')
script(src='/js/views/login.js')
script(src='/js/price.js')
```

The minified jQuery file gets imported first in jade files that utilize it. Other libraries used are:

Bootstrap – Used to develop a responsive website, with mobile-first priority. In this project it is used to make adjusting the grid layout of the page easier.

SocketIO – This library enables the backend and front-end to communicate using the Jade templating language to define forms that transmit a name-value list of parameters.

5.4.3 – UI Design

5.4.3.1 Background Image

The background image was made using GIMP, an image processing program. It combines 2 license-free images with a transparency and sharpness filter. The background in the image is slightly blurred and the colours were adjusted in brightness.

As the background image will be loaded first it needs to be small, yet retain most detail and clarity. The image was saved in the 32-bit 95% percent quality JPG format and later compressed using an online image compression tool called “Compress JPEG”. (COMPRESS JPEG, 2017)



5.4.4 – Navigation elements

The 3 tabs with content are Home, Orders and Purchase, and were built as individual features.



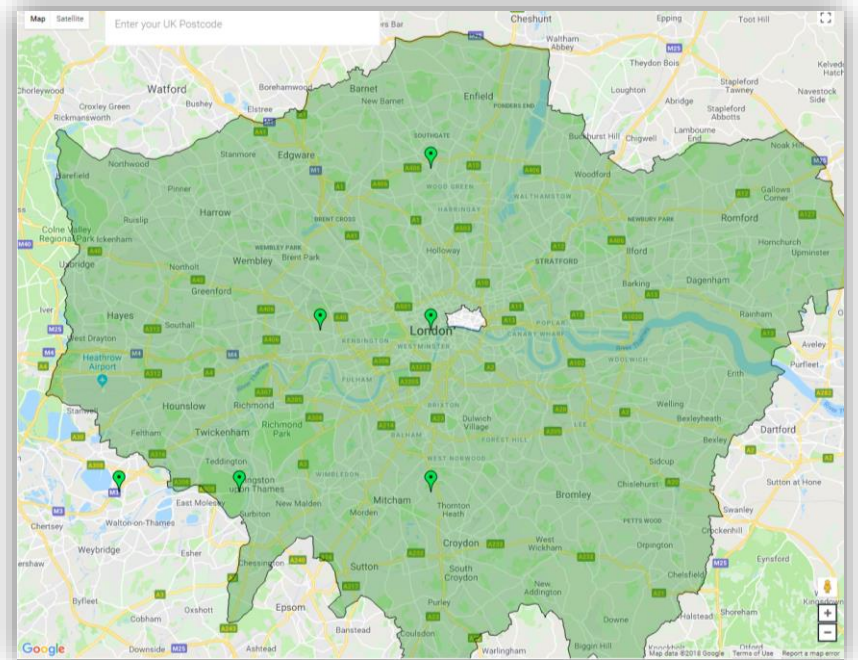
To keep the navigation simple the login form was placed on the Home page, with the “Create An Account” label to hide the login form and reveal the sign-up form. When the user switches to a different tab, only the new elements required will be downloaded, as modern browsers include that functionality. Throughout development this was tested and optimised, by using templates.

5.4.5 – Google Maps API

The map utilized on the purchase page uses the Google Maps API and Google Places API for the location search, with autocomplete functionality. (Google Places API, 2017)

```
map.data.loadGeoJson('../Google/London.json');
```

This line loads a file called “London.json” for the Geographical definitions, in this case the map of London. The json file was obtained using the website “Openstreetmap”. (OpenStreetMap, 2018)



The markers placed are placeholders, but later will constitute a query to the database to obtain the lat/lng (Latitude, Longitude) locations of the vendors. The colour of the highlighted area on the map is set using this call:

```
map.data.setStyle({
  fillColor: 'green',
  strokeWeight: 1
});
```

The Google Maps API delivers all of the features required for the map to be effective for locating an address, the destination and with future improvements, also display the directions from A to B. Using the mouse wheel plus the Control button allows the user can zoom in and out of the map, dragging moves the map and placing the orange figure found on the bottom right of the map activates Google StreetView.

Users can perform a location based search with autocomplete using the top-left search box. After performing a search by pressing Enter or clicking on a search result will zoom into the location found.

5.4.6 – CoinDesk Bitcoin Price API

The backend requires the latest price of 1 BTC in GBP and therefore an API call is made to CoinDesk, who provide a json response when called with the following parameters:

```
host: 'api.coindesk.com',  
path: '/v1/bpi/currentprice.json'
```

A call to setInterval will execute the function continuously and the call is also made when a user connects to the server.

```
// Refresh btc price every time a user connects. This will do for now.  
io.on('connection', function(){  
  updateBTCPrice();  
  console.log("Someone has connected");  
});
```

Once the price has been obtained from CoinDesk it is transmitted to the users client using SocketIO.

```
io.emit('btc_price_change', parsed.bpi.USD.rate);
```

5.4.7 – Bitcoin Blockchain Protocol implementation

Importing the “bitcoin-transaction” library is done through this line:

```
var bitcoinTransaction = require('bitcoin-transaction');
```

The library conveniently implements functions “getBalance()” and “sendTransaction()” which are used to check whether there are enough funds on the systems Bitcoin wallet.

```
// Send bitcoins  
sendBTC(clientWallet, 0.01);
```

“sendBTC()” gets called when an admin confirms receipt of the payment, at which point the system will transfer the bitcoin to the client’s wallet address. The “sendTransaction()” function in the “bitcoin-transaction” library was modified to return the Transaction ID, which is needed to be placed for the client to trace the transaction.

```
Client wallet Address: mkikq29QCizK7qBn25m6tBZPkBnNZmrqT5  
amount 0.01 balance 0.8175  
There are 4 unspent transactions.  
Bitcoins Sent!  
Saving tx_id :462b4dc288d54dc491e419efc5b6fe45b8af4d3f17004c54f36dc39037986ed1 For user: akaNeon22  
Saved tx_id!
```

This is the console output once an admin has confirmed an order. The “tx_id” is saved in the database for that users order.



5.4.8 – Version control GIT

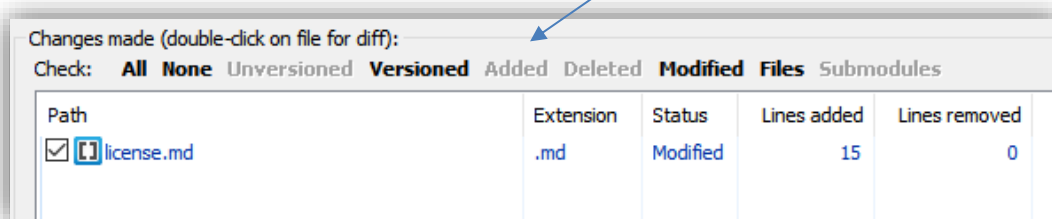
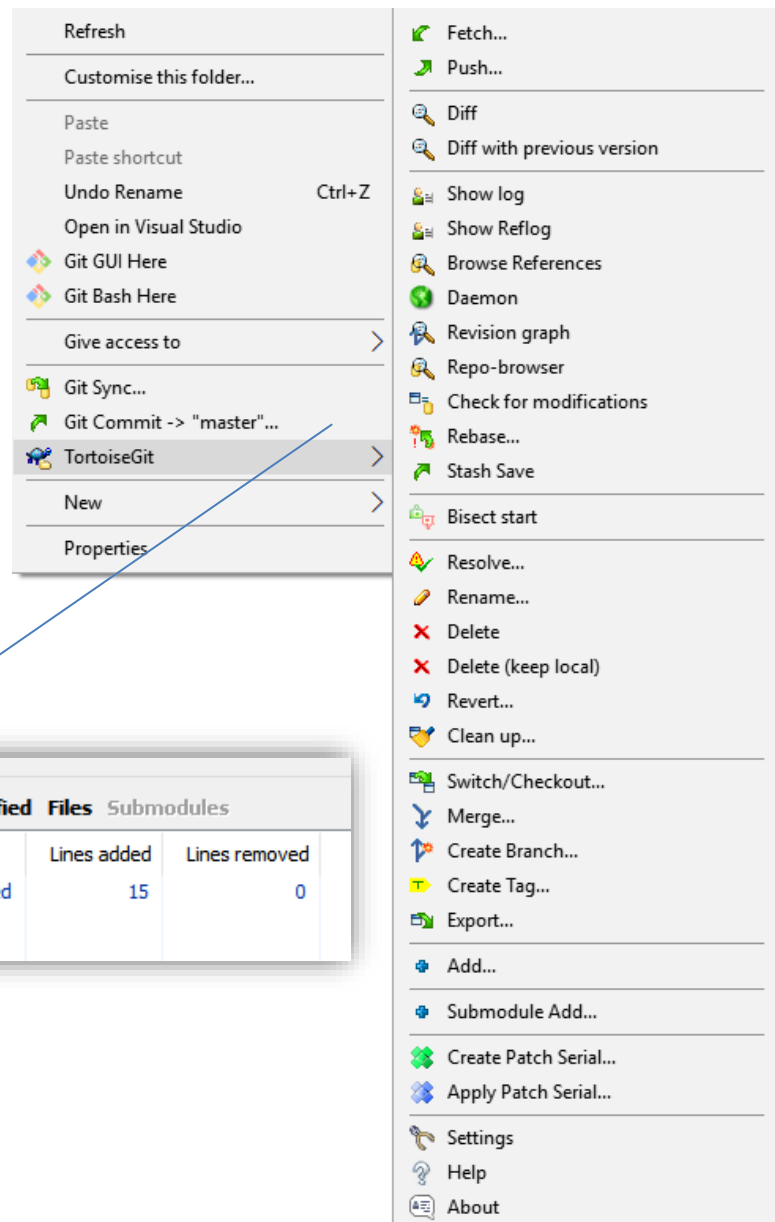
TortoiseGit was the tool used for version control.

It offers a GUI control to GIT, and makes it possible to use all git features without knowing the language behind it.

The most important features are:

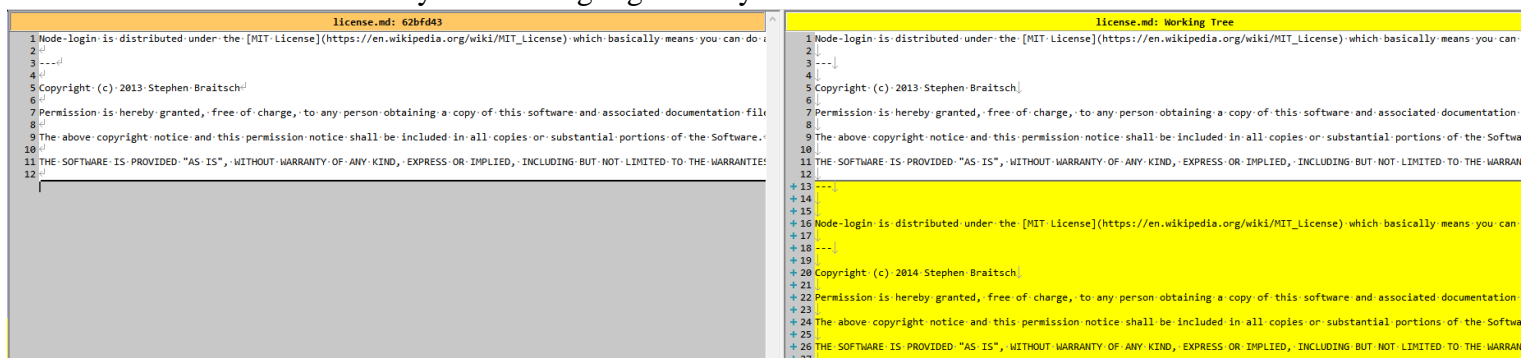
Git Commit -> “master” – Opens a window displaying a list showing all files that we’re changed since the most recent commit. It displays the file name and extension, lines added and removed.

Performing a right-click on a file opens a window with the option to revert all changes in the current version. This feature was very handy in debugging and testing.



Performing a double click on a file opens another powerful tool in TortoiseGit, the “Before-and-After” code viewer depicted in the below figure.

Reminiscent to the Linux development environment, it offers a clear and concise view of all code changes, numbered on the left with symbols to highlight newly added and removed lines of code.



5.4 – Evaluation

5.4.3 – Cross-platform testing

Cross-platform testing is necessary for most website as a significant percentage of users will be using mobile phones, Apple Laptops or Tablets. To accommodate for the continuous and thorough testing necessary, the work-flow follows the guidelines highlighted on the Mozilla documentation.

Initial planning > Development > Testing/discovery > Fixes/iteration

During development there must be a strategy to incorporate testing. Different browsers require specific code, especially for the latest features. In CSS many times 3 lines are needed to cover Firefox, chrome and Internet Explorer:

```
-moz-box-sizing: border-box;
-webkit-box-sizing: border-box;
-box-sizing: border-box;
```

After a feature has been implemented it must be tested on the 3 previously mentioned browsers. For major changes to the code further tests are required. This includes Android and Apple smart phones, tablets and laptops. Additionally, some “Developer” or “Pre-release” browsers were included in the tests, to ensure future waterproofing. These include:

- Firefox Developer
- Chrome Canary
- Opera Developer
- Edge Insider

5.4.1 – Use-case definitions

The planning and development phases relied heavily on the use-case definitions produced. (Appendix C)

Use case definitions are steps taken between an actor and a system to achieve the Postconditions defined.

The system is the collection of all the technologies in the front and back end.

The use cases were defined to ensure development stays on track and produces a proof of concept in time.

Sign Out
Brief description: The user clicks on the sign out button
Actors: User, System
Preconditions: The user is logged in already
Basic flow of events: <ol style="list-style-type: none"> 1. The user visits the home page 2. The user clicks on the sign out button 3. The system logs the user out and redirects to the home page
Alternative Flow: <ol style="list-style-type: none"> a) The user is not already logged in <ol style="list-style-type: none"> 1. The user visits the home page 2. The system hides the sign out button
Postconditions: The user has signed out

CHAPTER 6

CONCLUSIONS AND DISCUSSION

6.1 – Project objectives review

Objective	To what degree was the objective met
I shall develop an Android and iOS application that enables users to buy and sell CC's	This objective was replaced with a responsive website that meets the requirements of the mobile application defined in the PDD. The website allows users to buy Crypto Currencies, though selling is yet an issue. This is due to the fact that transactions can take time on the blockchain to get processed. In a real-world scenario, users would have to wait for the transaction to go through until they can pick up the money from a vendor. Another issue arises when large funds are bought from a user, and a vendor does not have the liquidity to cover the payment.
I shall build a website locating registered vendors, providing information such as opening times and fees	This objective is met with the purchase page and the containing Google map with search functionality and markers highlighting vendor locations
I shall build a network of Vendors	As discovered early on in development this project needed to limit scope on development and proof of concept. Thus, this objective was not met.

6.2 – Use case definitions review

This list will cover the objectives defined in the use case definitions and to what extent they were implemented.

Objective	To what degree was the objective met
Create an account	Implemented on the home page, hidden behind the login form. The registration form can be accessed with a click on the "Create an Account" label on the home page. The form takes the name, email, country, username and password of the user.
Sign in	The sign in form can be found on the home page, but is only visible if the user is not logged in.
Sign out	This feature has not been implemented fully. The code can be found in routes.js, though no button has been assigned for this functionality.

Place order	Users can place an order on the Purchase page by selecting the amount of BTC to be purchased and entering their BTC wallet address. After the vendor confirms receipt of the payment, the BTC will be released.
Confirm order	Users with admin privileges can view all open orders on the Orders page and see the most important information about the order.
View orders	Users can view their own orders on the Orders page. A list displays all information from the date it was placed to the amount of BTC purchased.
Forgotten password	This feature is partly implemented. The form and code are present though are not functional due to lack of time.
Contact Us form submission	The form and code are present but the form does not function as expected.

6.2 – Conclusions

The objective of this project was to produce a presentable prototype of a Crypto Currency trading platform that makes it possible to purchase CC's locally. The project was successful in that the software developed can be used to demonstrate the idea of purchasing CC's with cash and how it is made possible with a 3rd party vendor.

I have made a lot of personal progress during this project. This is due to the development decisions made and all the new technologies used. A lot of development time was taken up by research into CC's, Blockchain transactions, noSQL database management and secure communication channels.

Going forward, the website will be touched up and presented to investors to kickstart development and marketing of the platform.

CHAPTER 7

REFERENCES

- Massively (2018). *Fully responsive HTML5 + CSS3 site*. Retrieved January 19, 2018, from HTML5UP: <https://html5up.net/massively>
- Node Login (2013). *A template for quickly building login systems on top of Node.js*. Retrieved January 28, 2018, from GitHub: <https://github.com/braitsch/node-login>
- Node.js Udemy (2017). *The Complete Node.js Developer Course (2nd Edition)*. Retrieved February 1, 2018, from Udemy: <https://www.udemy.com/the-complete-nodejs-developer-course-2>
- Node.js Tutorial (2018). *Node.js Tutorial for Beginners: Learn Node in 1 Hour*. Retrieved February 26, 2018, from YouTube: https://www.youtube.com/watch?v=TIB_eWDSMt4
- Chodorow, K. (2013). *MongoDB: The Definitive Guide*. Retrieved February 27, 2018
- MongoDB SSL (2018). *Configure mongod and mongos for TLS/SSL*. Retrieved March 11, 2018, from mongodDB: <https://docs.mongodb.com/manual/tutorial/configure-ssl/>
- MongoDB SCRAM (2018). *SCRAM*. Retrieved March 14, 2018, from mongodDB: <https://docs.mongodb.com/manual/core/security-scram/#authentication-scram>
- Agile (2017). *A quick overview to Agile development*. Retrieved March 19, 2018, from Medium: <https://medium.com/@LazaroIbanez/a-quick-overview-to-agile-5c87ffc9e0f2>
- Google Places API (2017). *Autocomplete for Addresses and Search Terms*. Retrieved March 21, 2018, from Google: https://developers.google.com/maps/documentation/javascript/places-autocomplete#places_searchbox
- OpenStreetMap (2018). *London map*. Retrieved March 26, 2018, from OpenStreetMap: <https://www.openstreetmap.org/relation/65606#map=10/51.4887/-0.0460>
- BitCore Node (2018). *Running a Full Node*. Retrieved March 18, 2018, from bitcore.io: <https://bitcore.io/guides/full-node/>
- Bitcoin-Transaction (2017). *bitcoin-transaction*. Retrieved March 18, 2018, from github.com: <https://github.com/Blank101/bitcoin-transaction>
- Security Checklist (2017). *Security Checklist*. Retrieved March 18, 2018, from mongodb.com: <https://docs.mongodb.com/manual/administration/security-checklist/>
- AIR iPhone 3.03 (2017). *AIR iPhone*. Retrieved March 11, 2018, from github.com: <https://air-iphone.informer.com/>

COMPRESS JPEG (2017). *COMPRESS JPEG*. Retrieved March 11, 2018, from compressjpeg.com:
<http://compressjpeg.com/>

CHAPTER 8

GLOSSARY

Term	Meaning
CC	Crypto Currency
Crypto Currency	A digital asset stored on a blockchain
Blockchain	A chain of digital blocks that contain information about the previous block
NoSQL	“Not Only SQL” – Non-relational database
GIT	Version control
API	Application Programming Interface
ATM	Automated teller machine
BTC	Bitcoin
UI	User Interface
GUI	Graphical User Interface

Appendix A – Project Definition Document

PROBLEM DEFINITION DOCUMENT COVER SHEET

- **Contact Details:**

- **Name:** Fedil Al-Hayawi
- **E-Mail:** Fedil@hotmail.de
- **Telephone:** 07454906820

- **Supervisor Contact Details:**

- **Name:** Andrey Povyakalo
- **E-Mail:** a.a.povyakalo@city.ac.uk
- **Telephone:** 02070408247

- **Degree Programme:** Computer Science with Games Technology
- **Project Title:** Crypto Currency trading mobile application making trading accessible locally with help of small business owners
- **Project Proposed By:** Fedil Al-Hayawi (Student)
- **Arrangements For Proprietary Interests:** None, at this stage
- **Proposal Word Count:** 984

PROPOSAL

Problem To Be Solved

The problem addressed with my project is the lack of accessibility to Crypto Currencies (referred to as CC's from here on out) for the general public. With the increasing demand for Bitcoin, Ethereum and other Blockchain-Based digital currencies, and the scarcity of fast, reliable, secure and anonymous means to purchase and sell said currencies, the market has given opportunities to various types of businesses.

From a consumer perspective, there are 3 main ways to purchase CC's:

1. Online Platforms such as: Coinbase, Gemini or Kraken.
2. Person-To-Person Trades facilitated by online platforms such as: LocalBitcoins, LocalEthereum or BitQuick.
3. Crypto Currency ATM's.

Any one of these methods is flawed in many ways. Online platforms have incredibly high fee's, are not anonymous, slow (due to identity checks) and more importantly, not safe. To individuals unfamiliar with the risks involved with using Online Wallets (Containers of CC's), the 2013 "Mt Cox" exchange disaster that ended up in Bitcoin's price falling by 23% within hours, clearly demonstrated the vulnerabilities of such platforms.

The problem with Person-To-Person trades is speed and security. The risks involved with carrying large sums of money and meeting a stranger for a trade are obvious.

As of now, ATM's are rare, expensive (high fee's) and also require identification in some cases. Due to the way ATM's currently work, there is also potential risk. They can be manipulated just like, or even easier than, regular ATM's to trick the user into exposing sensitive information, and of course, loss of money. Another risk is that most of these machines will print out a receipt that can be easily intercepted or stolen. This receipt contains the CC's to be redeemed by the customer and is not obfuscated in any way.

The main competitor is a company called LibertyX. They provide a similar service, though restricted in some ways. Firstly, they are based in the US and only cover a small area. Locally, there is no competition and this project will offer a service that does not currently exist.

Project Objectives

- I shall develop an Android and iOS application that enables users to buy and sell CC's.
 - The application must access a database of Wallets containing the currency to be sold and make it redeemable in some form (Receipt, code or similar).

- The application must be able to verify a payment made by a customer.
- The application must allow users to request and make payments to a bank account.
- I shall build a network of Vendors (owners of small businesses such as Off License shops) using the application.
- I shall build a website locating registered vendors, providing information such as opening times and fees.

Project Beneficiaries

This project aims to provide a means of purchasing and selling CC's locally, avoiding many of the downfalls of other services.

The application will be used by small business owners to sell and buy CC's. They shall be found by customers using a website showing a map with all registered vendors.

Business owners will benefit from exposure as a vendor of CC's.

The application will allow vendors to accept CC's as an additional payment method. This will attract customers naturally.

Work Plan

The main deliverable is the mobile application. Once it has been developed, a business proposal will be written and rehearsed to begin establishing a network of vendors. Once at least a single vendor has expressed interest, the website will be developed, to be used by the public.

Project Timeline				Q4			Q1			Q2		
				Oct	Nov	Dec	Jan	Feb	Mar	Apr	May	Jun
1	Development of Android Application using Java and the Android SDK.	60d	03/11/17	25/01/18								
2	Development of iOS Application using Swift and the iOS SDK.	30d	25/01/18	07/03/18								
3	Writing and rehearsing of business Proposal to potential vendors.	15d	07/03/18	27/03/18								
4	Development of website for vendors using HTML, CSS and JS.	15d	27/03/18	16/04/18								
5	Develop additional features for the applications.	30d	16/04/18	25/05/18								

The development process in tasks 1 and 2 will heavily rely on 3rd party libraries that enable interaction with various Blockchains. Depending on how smoothly development goes, several Blockchains will be included in these tasks. Otherwise they will be included in the “Additional Features” in task 5.

Task 5 will include a number of improvement to usability, visually presentation, and ease of use. Additionally, it will include the payment processing mechanism that allows vendors to pay and get paid by a centralised bank account, accessibly only by system administrators.

Task 3 encapsulates many sub-tasks. These include:

- Finding interested vendors.
- Educating vendors on Blockchain technology.

- Visiting small business owners in person and delivering a convincing business proposal.

Task 4 includes:

- Developing a responsive website.
- Adding a comprehensive map that displays the user's location.
- Adding a registration form for vendors interested in joining the network.

Work Plan

This project heavily relies on healthy state of CC's and requires them to maintain their price. Stakeholder and vendors will be invested and a drop in price can heavily affect liquidity.

As a consumer-based project, it is essential there is demand. Bitcoin has faced many problems in the past. Recently, all CC exchanges were banned in China, causing first a drop in price, and of course more uncertainty in the community.

Such a ban in the UK would make this project redundant for the foreseeable future.

Other risks stem from the nature of Blockchain technology. To name a few:

- Slow transaction speeds. This is a major problem with purchasing as it takes ~30 minutes for 3 confirmations to go through. There must be a mechanism around this problem.
- Rapid price changes. Individuals may try to exploit the system if they identify a price difference between vendors and exchanges.
- Risk of losing funds. All funds will be stored in wallets that are transparent to the public. To gain access, only 1 private key is needed. Attacks to the system could compromise these keys.

There are also risks in the interaction with the vendors such as in the following example:

A customer purchases CC's from a vendor. The vendor will be lending money, in form of CC's, from a centralised bank account, open to all vendors, to pay the customer. The physical money (in £) is now in the vendors hands and they must be trusted to pay it back.

RESEARCH ETHICS CHECKLIST

A.1 If your answer to any of the following questions (1 – 3) is YES, you must apply to an appropriate external ethics committee for approval.		
1.	Does your project require approval from the National Research Ethics Service (NRES)? For example, because you are recruiting current NHS patients or staff? If you are unsure, please check at http://www.hra.nhs.uk/research-community/before-you-apply/determine-which-review-body-approvals-are-required/ .	No
2.	Does your project involve participants who are covered by the Mental Capacity Act? If so, you will need approval from an external ethics committee such as NRES or the Social Care Research Ethics Committee http://www.scie.org.uk/research/ethics-committee/ .	No
3.	Does your project involve participants who are currently under the auspices of the Criminal Justice System? For example, but not limited to, people on remand, prisoners and those on probation? If so, you will need approval from the ethics approval system of the National Offender Management Service.	No
A.2 If your answer to any of the following questions (4 – 11) is YES, you must apply to the City University Senate Research Ethics Committee (SREC) for approval (unless you are applying to an external ethics committee).		
4.	Does your project involve participants who are unable to give informed consent? For example, but not limited to, people who may have a degree of learning disability or mental health problem, that means they are unable to make an informed decision on their own behalf?	No
5.	Is there a risk that your project might lead to disclosures from participants concerning their involvement in illegal activities?	No
6.	Is there a risk that obscene and or illegal material may need to be accessed for your project (including online content and other material)?	No
7.	Does your project involve participants disclosing information about sensitive subjects? For example, but not limited to, health status, sexual behaviour, political behaviour, domestic violence.	No

8.	Does your project involve you travelling to another country outside of the UK, where the Foreign & Commonwealth Office has issued a travel warning? (See http://www.fco.gov.uk/en/)	No
9.	Does your project involve physically invasive or intrusive procedures? For example, these may include, but are not limited to, electrical stimulation, heat, cold or bruising.	No
10.	Does your project involve animals?	No
11.	Does your project involve the administration of drugs, placebos or other substances to study participants?	No
A.3 If your answer to any of the following questions (12 – 18) is YES, you must submit a full application to the Computer Science Research Ethics Committee (CSREC) for approval (unless you are applying to an external ethics committee or the Senate Research Ethics Committee). Your application may be referred to the Senate Research Ethics Committee.		
12.	Does your project involve participants who are under the age of 18?	No
13.	Does your project involve adults who are vulnerable because of their social, psychological or medical circumstances (vulnerable adults)? This includes adults with cognitive and / or learning disabilities, adults with physical disabilities and older people.	No
14.	Does your project involve participants who are recruited because they are staff or students of City University London? For example, students studying on a specific course or module. (If yes, approval is also required from the Head of Department or Programme Director.)	No
15.	Does your project involve intentional deception of participants?	No
16.	Does your project involve participants taking part without their informed consent?	No
17.	Does your project pose a risk to participants or other individuals greater than that in normal working life?	No
18.	Does your project pose a risk to you, the researcher, greater than that in normal working life?	No
A.4 If your answer to the following question (19) is YES and your answer to all questions 1 – 18 is NO, you must complete part B of this form.		

19.	Does your project involve human participants or their identifiable personal data? For example, as interviewees, respondents to a survey or participants in testing.	No
-----	---	-----------

Appendix B – Compiling the website

Step 1: Install node.js

The easiest way to install node.js for windows is through the main website. (<https://nodejs.org/en/>)

Step 2: Setup MongoDB database

Using Robo3T you can setup a mongoDB database with a GUI.

Step 3: Run MongoDB database

To run the mongoDB database run the “mongod.exe” file either by double clicking or running the exe through a command line.

Step 4: Run the server

Running the server is done with the command “node app”, in the root source directory.

Step 5: Access the website

Navigate to localhost:3000 using any browser

Appendix C – Use-case definitions

USE-CASE DEFINITIONS

Create an account
Brief description: The user fills out sign-up form and presses submit
Actors: User, System
Preconditions: The user is not logged in already
Basic flow of events: <ol style="list-style-type: none"> 1. The user visits the home page 2. The system checks if the cookies contain information about this user 3. The user clicks on the “Create An Account” button in the login form 4. The user fills out the sign-up form correctly and clicks the submit button 5. The system creates an entry for this user in the database, logs them in, and redirects to the Purchase page.
Alternative Flow: <ol style="list-style-type: none"> a) The user is already logged in <ol style="list-style-type: none"> 1. The user visits the home page 2. The system hides the login form b) The system finds a cookie containing the users information <ol style="list-style-type: none"> 1. The user visits the home page 2. The system performs a call to login the user automatically and refreshes the page c) The user incorrectly fills out a field <ol style="list-style-type: none"> 1. The user enters their details in the signup form 2. The system performs checks on every field for invalid input 3. The system displays an error message containing instruction to fix the issue
Postconditions: The system adds the users information to the database

Sign in
Brief description: The user fills out login form and presses submit
Actors: User, System
Preconditions: The user is not logged in already
Basic flow of events: <ol style="list-style-type: none"> 1. The user visits the home page 2. The system checks if the cookies contain information about this user 3. The user fills out the login form correctly and clicks the submit button 4. The system will log the user in and load the users data and redirect them to the Purchase page
Alternative Flow: <ol style="list-style-type: none"> a) The user is already logged in <ol style="list-style-type: none"> 1. The user visits the home page 2. The system hides the login form b) The system finds a cookie containing the users information <ol style="list-style-type: none"> 1. The user visits the home page 2. The system performs a call to login the user automatically and refreshes the page c) The user incorrectly fills out a field <ol style="list-style-type: none"> 1. The user enters their details in the login form 2. The system performs checks on every field for invalid input 3. The system displays an error message containing instruction to fix the issue
Postconditions: The user has signed in

Sign Out
Brief description: The user clicks on the sign out button
Actors: User, System
Preconditions: The user is logged in already
Basic flow of events: <ol style="list-style-type: none"> 1. The user visits the home page 2. The user clicks on the sign out button 3. The system logs the user out and redirects to the home page
Alternative Flow: <ol style="list-style-type: none"> a) The user is not already logged in <ol style="list-style-type: none"> 1. The user visits the home page 2. The system hides the sign out button
Postconditions: The user has signed out

Place order
Brief description: The user fills out login form and presses submit
Actors: User, System
Preconditions: The user is logged in already
Basic flow of events: <ol style="list-style-type: none"> 1. The user visits the purchase page 2. The user enters the amount of BTC desired 3. The system converts the entered amount to GBP 4. The user enters their Bitcoin wallet address 5. The user clicks the confirm button 6. The system makes an entry of the order for the user
Alternative Flow: <ol style="list-style-type: none"> a) The user is not already logged in <ol style="list-style-type: none"> 1. The user visits the purchase page 2. The system identifies the user to not be logged in and redirect to the home page b) The user incorrectly fills out a field <ol style="list-style-type: none"> 1. The user enters their details in the login form 2. The system performs checks on every field for invalid input 3. The system displays an error message containing instruction to fix the issue
Postconditions: The user has placed an order

Confirm order
Brief description: The admin selects an order and clicks the submit button
Actors: User/Admin, System
Preconditions: The user logged in is an admin
Basic flow of events: <ol style="list-style-type: none"> 1. The admin visits the Orders Page 2. The system displays all open orders in the database 3. The admin selects the radiobox of an order 4. The admin clicks the submit button 5. The system sends the bitcoin to the user and marks the order as confirmed
Alternative Flow: <ol style="list-style-type: none"> a) The user is not already logged in <ol style="list-style-type: none"> 1. The user visits the orders page 2. The system identifies the user to not be logged in and redirect to the home page b) The user logged in is not an admin <ol style="list-style-type: none"> 1. The system will show a different view for non-admins
Postconditions: The order is confirmed and the Bitcoins are sent to a user

View orders
Brief description: The user views their orders
Actors: User, System
Preconditions: The user is logged in already
Basic flow of events: <ol style="list-style-type: none"> 1. The user visits the orders page 2. The system displays all orders for this user in the database

3. The user is able to see the Bitcoin Transaction ID of orders
Alternative Flow:
a) The user is not already logged in <ol style="list-style-type: none"> 1. The user visits the orders page 2. The system identifies the user to not be logged in and redirect to the home page
Postconditions: The user is shown all their orders

Forgotten password
Brief description: The user sets a new password using their registered email
Actors: User, System
Preconditions: The user is not logged in already
Basic flow of events:
1. The user visits the home page and clicks the “Forgotten Your Password” 2. The system hides the login form and display a different form 3. The user fills out the email field and submits 4. The system sends an email containing instructions to set a new password
Alternative Flow:
a) The user is already logged in <ol style="list-style-type: none"> 1. The user visits the home page 2. The system identifies the user to be logged in hides the login form b) The user enters an incorrect email <ol style="list-style-type: none"> 1. The user visits the home page and clicks the “Forgotten Your Password” 2. The system hides the login form and display a different form 3. The user fills out the email field and submits 4. The system cannot find the email supplied by the user and shows an error message
Postconditions: The user receives an email with instructions

Contact Us form submission
Brief description: The user sends a message through a form in the footer of the website
Actors: User, System
Preconditions: None
Basic flow of events:
1. The user visits any page and fills out the contact us form 2. The user clicks the Send Message button 3. The system checks the contents of the fields submitted 4. The system stores the contents in the database 5. The system displays a “Thank You” message
Alternative Flow:
a) The user incorrectly fills out a field <ol style="list-style-type: none"> 1. The user enters their details in the contact us form 2. The system performs checks on every field for invalid input 3. The system displays an error message containing instruction to fix the issue
Postconditions: The user submits a message

Appendix D – High level requirements

HIGH LEVEL REQUIREMENTS

Abstract

This document will cover all high-level requirements in a table. The purpose of this table is to be a reference to the developer. It is meant to be used together with the use cases and use case diagram.

Requirement	Description
Setup MongoDB database	In order to prove that mongoDB is a suitable database for this project, it is the first part to be developed.
Implements node.js template “Node Login”	The Node Login template will form the infrastructure for the website. It is therefore of utmost importance to clean the code, and strip all unnecessary or deprecated code from the code base. The result should be to have a clear understanding of the mechanisms behind the code.
Find suitable HTML 5 template to integrate with the node.js backend	It was decided to opt for a template instead of writing the front-end from scratch. This is due to the project being largely a proof-of-concept, and process should be made quickly.
Bitcoin Blockchain API integration	A suitable node.js implementation of the bitcoin protocol layer must be found. Most likely, this will be the Blockchain “bitcore” library on GitHub.
Bitcoin Price API integration	The backend must fetch the price of 1 BTC every few seconds to ensure that users can’t abuse price fluctuations.
Indicate guest/logged-in user with a top bar	Permanently top fixed bar to greet visitors and show a link to the log-in form. It must be responsive and have different content depending on: <ul style="list-style-type: none"> • Is the user logged in? • Is the user an admin? The website should indicate these things.
Securely store and transfer sensitive information	Security guide lines must be followed to ensure the application does not have obvious security leaks. When transferring information between the backend and database, the password must be hashed or in other ways obfuscated. Sensitive information may not be displayed or accessed on or from the front-end.

Display and modify orders placed by users	<p>The website shall allow users to place orders on a dedicated page. The form must be able to store orders to the account logged in. If the user is not logged in, placing an order must not be allowed.</p> <p>Admins must have the functionality to accept orders that were paid for. Using checkboxes or radio boxes and a submit button should suffice.</p>
Design/use attractive website design	<p>The website developed is a consumer product and must look and feel professional to gain credibility from users.</p> <p>Dark grey and brown colours should be favoured, with an appropriate background, relating Bitcoin and linking the website to London.</p>
Simplistic UI with social media indicators	<p>Navigation must be intuitive.</p> <p>For the prototype a 3-page site will perform all required tasks without problems.</p> <p>To use social media icons a third-party library must be used.</p>
HTML5 Login/Sign-up form with email retrieval	<p>Login and signup forms will ideally be developed using HTML 5 forms. This means that error checking is built-in and compatible with many platforms and browsers.</p>
Google Maps API integration	<p>A google map must be included on the purchase page of the website.</p> <p>It must highlight the vendors on the map.</p> <p>Preferably, it would display the distance to the user's locations, but for the prototype the following features are necessary:</p> <ul style="list-style-type: none"> • Responsive Map. • Search box for user to locate themselves using Post Code or Address name. • Icons placed at vendors locations • Show information when an icon is clicked.
Host website publicly	<p>The final step is to host all the source files on a public server. A free server may be used for demonstration purposes.</p>