Software Development Year 3  
Professional Practice in IT Documentation

John Groves G00367771

David O’Loughran G00362038

# Abstract

Summary of document and what our application does.

Table of Contents

[Abstract 2](#_Toc69941205)

[Introduction 5](#_Toc69941206)

[System Requirements 6](#_Toc69941207)

[User account 6](#_Toc69941208)

[Access to multiple coins 6](#_Toc69941209)

[Live graph of coins 6](#_Toc69941210)

[Access to market orders and limit orders both buy and sell 6](#_Toc69941211)

[See profits and percentages 6](#_Toc69941212)

[Assets overview of all orders and coins 7](#_Toc69941213)

[Technology Used and Why 8](#_Toc69941214)

[Amazon Relational Database Service 8](#_Toc69941215)

[PostgreSQL 8](#_Toc69941216)

[Node.js 8](#_Toc69941217)

[Passport.js 8](#_Toc69941218)

[bcrypt 9](#_Toc69941219)

[TradingView lightweight charts 9](#_Toc69941220)

[Binance API 9](#_Toc69941221)

[Architecture of the Solution 10](#_Toc69941222)

[Front End 10](#_Toc69941223)

[Database 10](#_Toc69941224)

[Design Methodology Applied 11](#_Toc69941225)

[Software Development Life Cycle 12](#_Toc69941226)

[Features of the Implementation 14](#_Toc69941227)

[Limitations and Known Bugs 15](#_Toc69941228)

[Testing Plans 16](#_Toc69941229)

[Recommendations for Future Development 17](#_Toc69941230)

[Conclusions 18](#_Toc69941231)

# Introduction

This documentation is intended to provide different perspectives on the project. It is intended to provide both user and developer with background information for the system.

The **user is interested in what the project does**, how they interact with it and what the system looks like.  The requirements form part of this section of documentation and will provide detail on the functional components of the system.

The **developer is concerned with how the application is implemented**.  This concentrates on the technical aspects of the user interactions rather than what the system looks like. As part of the development process, the documentation allows the development team to scope out the different technologies that are available to provide a solution to different aspects of system implementation. These technologies can be compared, and decisions made on the merits of each as to whether it is appropriate to use in the development of the new system.

The documentation should provide the following information:

* Architectural Overview of the system.
  + Provide a functional breakdown of the system to be developed.
* Identify and compare the possible technologies, justify the choice of technology to be used in the implementation.
* Database design.
  + Normalise the design for the implementation.
  + Specify data types, field sizes, default values and any constraints on the values to be stored.
* Screen Layout providing a blueprint for how the particular application will look to the end user.

# System Requirements

## User account

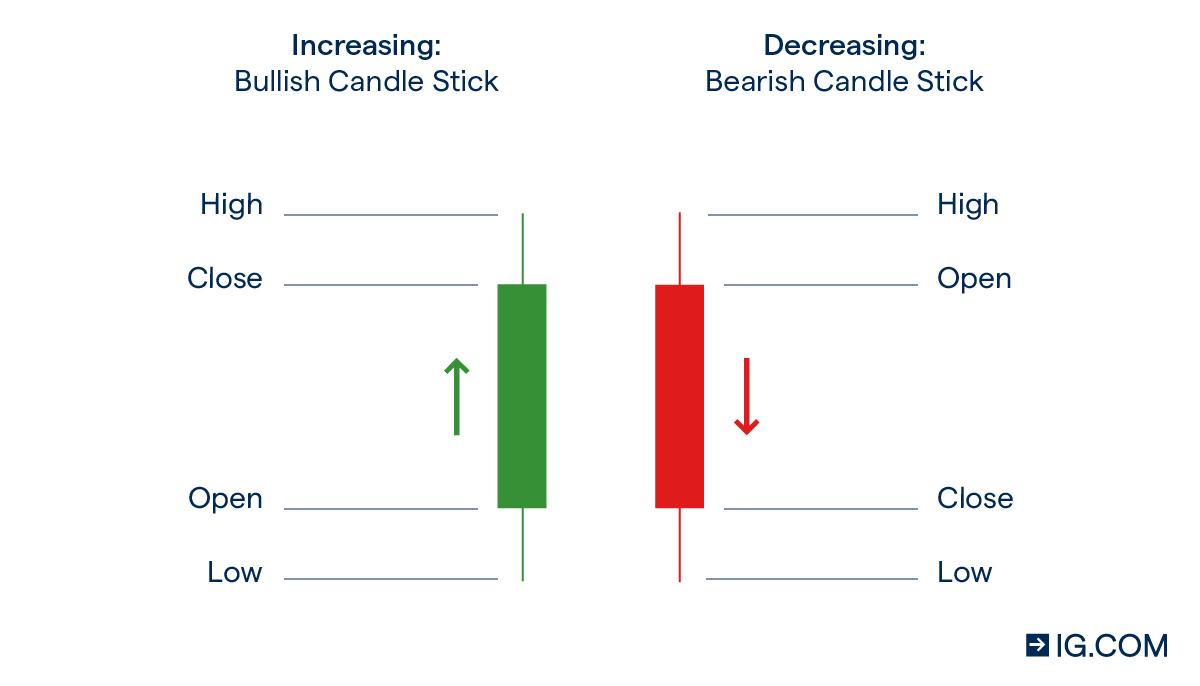
On using our application, the user will be prompt to login or register an account. The user will enter a name, email and password on registering. A user cannot register two accounts under the same email, both entered passwords must match and contain at least 6 characters. Once the user registers an account, they will be able to login under the entered email and password. The user’s password is encrypted before being stored in the database for security purposes.

## Access to multiple coins

The user has access to five coins they can trade from. We decided on providing the top three coins being Bitcoin, Ethereum and XRP. And two upcoming coins in this year, Binance Coin and Cardano (ADA).

## Live graph of coins

The user has access to live graphs of the five coins, displaying accurate historical and real time data of each coin. The user can choose between a one minute, four hours, or daily intervals of each candlestick on the graph. Candlesticks are extremely helpful as instead of showing only the price of one coin they have four different price points for each interval. The price points included are the open price, close price, high price and low price for that interval. (leave to Dave for more technical analysis). The data displayed goes back 2 years.



## Access to market orders and limit orders both buy and sell

The user will have access to two different methods of buying and selling coins. Market orders will allow the user to either buy or sell coins at their live price the moment the order is executed. Limit orders on the other hand will allow the user to set which price they would like to buy or sell with the order executing if it hits the specific price or better. For example if the current price is $55000 for 1 Bitcoin and the user executed a limit buy order of $60000, the users trade would instead be for the market value of $55000 as it is the best available price. All limit orders will then be logged to the database for the users account allowing it to be executed even if the price hits while the user is offline.

## See profits and percentages

Maybe

## Assets overview of all orders and coins

The user has access to the asset overview page which displays all data from their wallet. It displays each coin, the amount the user owns, any limit orders that are in place, the current price of that coin and the amount that is worth in USD.

# Technology Used and Why

## Amazon Relational Database Service

We used Amazon RDS to host our PostgreSQL database. Amazon RDS fitted the requirements of the application; it was easy to setup, cost efficient and secure. It provided fast high scalability for calling API’s and for querying to the database. It runs on the same highly reliable infrastructure used by other Amazon Web Services, with no downtime or maintenance.

## PostgreSQL

After extensive research on relational databases we decided on using PostgreSQL. PostgreSQL is a powerful, open source object-relational database system with over 30 years of active development. Our main reason for choosing PostgreSQL was because its performance works best in systems which demand the execution of complex queries, we knew our application would use complex queries to calculate market orders, limit orders and percentages. PostgreSQL was easy to learn, free and the GUI element (pgAdmin) was very useful for maintaining and viewing our database.

## Node.js

After doing research on many different types of front-end web development, we decided on using node.js. The reasons for this was mainly to do with its scalability, its capable to handle a huge number of concurrent connections which was especially important for our application since we would be taking in a lot of cryptocurrency data from different connections. Another big reason for choosing this technology was the simplicity of sharing one language for both server and client side, our application is coded heavily server side and using node.js on our front-end to display the finalized data was very useful. This technology is also lightweight and can reduce the application development time while achieving the same functionality, which helped us get immediate feedback from our production environment.

## Passport.js

Passport is an authentication middleware for Node.js. It is easy to use and implement. With the passport-cookie library it uses session-based authentication to utilize browser cookies to manage logged-in and logged-out users.

## bcrypt

We wanted our application to have some form of extra security for its users. Bcrypt is an easy to use password-hashing function. It is a library for NodeJS which made it easy to install and implement into our application. Each accounts password is hashed 10 times before being stored in the database for extra security.

## TradingView lightweight charts

For our application we wanted to provide our users with a similar experience to trading on a live crypto exchange. Following research into which charting library would be best suited for our application we realized almost every popular exchange utilized different versions of TradingViews charting libraries. They provide a wide range of easily customizable properties while also allowing data of different formats to be easily inserted, modified and displayed within their charts. We decided upon using TradingViews Lightweight Charts as its capabilities fitted our needs while also being free to use for all developers. It allowed us to easily insert historical and live data into the charts from the Binance API. These charts are also very responsive allowing the user to adjust both the time and price scales while providing the ability to zoom in and out on different sections of the chart using their mouse wheel.

## Binance API

For our application we wanted to p

# Architecture of the Solution

## Front End

Talk about each page and layout of them

## Database

Talk about data types / relationship between tables

# Design Methodology Applied

Initial research and plan on project

Worked separately each week and committed at end of said week

Had a projects tab on git but has not been updated

Talk about meetings and how they kept us on track with coding

# Software Development Life Cycle

## Planning

At the start of this project we quickly decided on making a paper trading crypto-exchange application as we both had a large interest in cryptocurrencies. We initially spent the first two weeks doing extensive research of what technologies to use in our application. While researching we kept in mind what database would suit our application best, what front end would work efficiently for displaying large sums of data and how to pull live data of individual coins into our application.

## Requirements

Database: user table for accounts also made master user for us to change table

Host to AWS

Login system with error handling etc

Added session cookies + encrypted password

Added sockets and apis pulling coin data

Set up charts

Added navbar

Database: wallet table linked to user

Focus on queries and math calculations

Added asset overview which displays all coin data from wallet database

Cleaned UI for chart page

Database: trade table set up for limit orders

More maths + query calculations

Completed Limit orders

Added other coins

Fixed any bugs/typos/UI features

# Features of the Implementation

Developer

Note write about technologies that did not work as expected/ wrong approach taken

# Limitations and Known Bugs

# Testing Plans

Test on limit orders

# Recommendations for Future Development

Two-factor Authentication for extra security

Buy/Sell real crypto

Add more coins

Credit card details

Deploy to front end

# Conclusions