Software Design Document

for

Cointrum

Version 1.0

Created by Jordan Emslie

February, 2020

Table of Contents

Introduction

1.1 Purpose

This Software Design Document will give an overview of the processes and specifications we will take over the course of this project to form a functioning Cointrum application. This Document will also help layout the overall scope of the project.

1.2 Scope

The end product of Cointrum will be a Trading bot that will use trading indicators created by a label based Machine Learning Models. In layman's terms it will take examples of what has happened in the past, and use these to “predict” the future. The goal of this project is not to make money, if it were that easy this would have already been created. The goal is to learn about market strategies and Machine learning techniques. There should be an interface for this product that allows, creation of these labels, and the ability to select timeframes in the past to show the software what the label looks like. As well as setting the points where you would need to buy and sell. With these labels you will be able to create a trading map that allows you to use how confident the system needs to be of a label, before making a trade, and how much it will trade based on the confidence level. Also, the users will need to be able to set ‘stopping-points’ where if the price of the stock/currency fluctuates too far from the estimation, then ‘pull out’.

1.3 Overview

Technical Specifications

2.1 Architecture

A large part of development will come organizing how the product will work relative to the backend. Currently the idea is to have a backend that works as a REST API, thus we can choose to create the frontend as a webapp, or extension to browser, then we can do both.

2.1.1 Back-end Framework - Type-Script on a RESTful NodeJS Server (OO)

2.1.2 Front-end Frameworks - Typescript ReactJS

2.1.3 Machine Learning Framework - R.js

Allow us to use simple Machine Learning commands used in R.

2.2 Programming Language

Typescript
React Native, Firebase Functions, and NodeJS are all limited to either Javascript or Typescript. Thus, we are limited to those two options. The main benefits of using Typescript over Javascript is that it supports: Class and Module Support, Type-checking, and relative code completion. Theses additions in a team-development environment means that reading other people's code will be easier to understand, meaning less time wasted figuring out what a component does. As, these extra features that Typescript supports allow you to see attributes and methods linked to the classes/components. Also, Typescript allows you to have strict type-checking, meaning that you can detect and debug type-related errors much faster than with Javascript. We are using the most recent version of Typescript as of the creation of the project, as there is no benefits of using previous versions.

2.3 Database and File Storage

2.3.1 User Store - Firebase Authentication

Firebase Authentication *(Requested by Client)*
The client has requested that we use Firebase Authentication for our User Store. This framework will handle account registration, account info storage, login, login sessions, and email confirmations. The client requested this to be used as it is an easy-to-use framework that has been highly tested, so user credentials are safe and secure. It is completely compatible with React Native, NodeJS, and Firebase Cloud Functions.

2.3.2 Data Store - MongoDB and maybe MySQL (or noSQL w/schema)

MongoDB
Easy for adding and manipulating documents

MySQL

For completing complex queries on data

2.6 Testing Framework

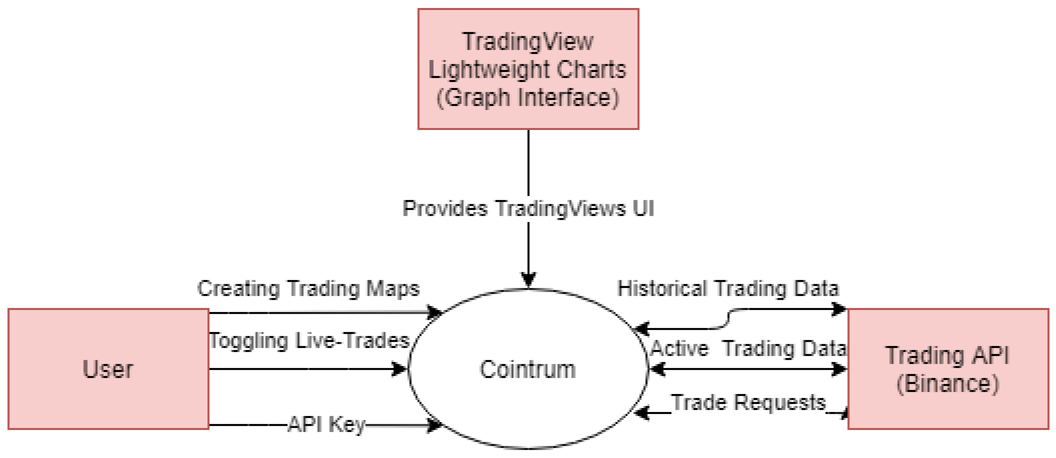
JestJS (Typescript)
For Unit or Integration testing whether it be with React Native, NodeJS, or Firebase Functions we are using JestJS as it supports all three. With this we can complete snapshot testing, assertion testing, code coverage testing and any sort of unit testing. It also supports the ability to easily mock components that you don't want to be affecting the tests. Jest is a highly supported and used framework (used by Facebook, Twitter, etc.)

System Architecture

3.1 Data Flow Diagrams

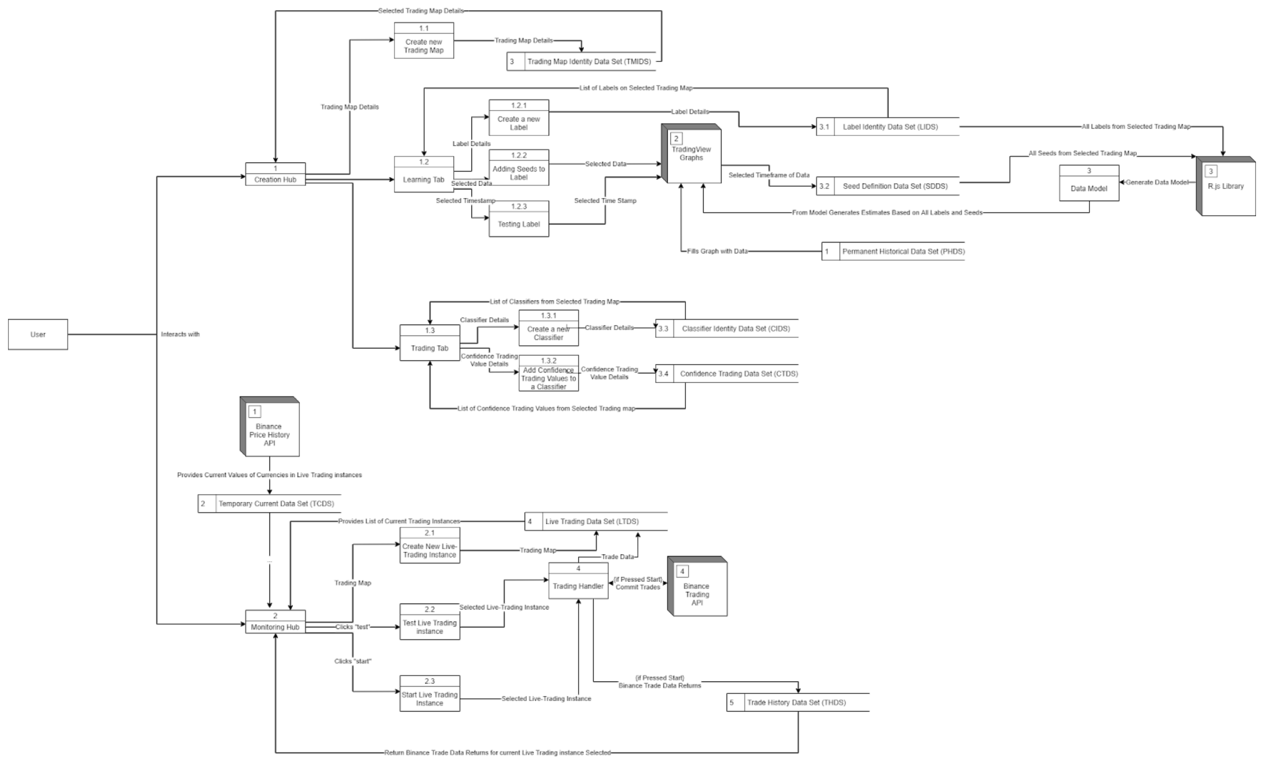
3.1.1 Level 0

The level 0 data flow diagram gives a general view of the app and its main processes. All information that the user sends to Cointrum. And where Cointrum gets/sends data to external sources.



3.1.2 Level 1

The level 1 diagram goes deeper into the application and it's main processes. Description of processes in Work Breakdown structure.


Selected Trading Map Details
1.1
Create new Trading Map
Trading Map Details
3 Trading Map Identity Data Set (TMIDS)
Trading Map Details
List of Labels on Selected Trading Map
1.2.1
Create a new Label
Label Details
3.1 Label Identity Data Set (LIDS)
All Labels from Selected Trading Map
Label Details
1
Creation Hub
1.2
Learning Tab
1.2.2
Adding Seeds to Label
Selected Data
2
TradingView Graphs
Selected Timeframe of Data
3.2 Seed Definition Data Set (SDDS)
All Seeds from Selected Trading Map
3
Data Model
Generate Data Model
3
R.js Library
Selected Data
Selected Timestamp
1.2.3
Testing Label
Selected Time Stamp
From Model Generates Estimates Based on All Labels and Seeds
Fills Graph with Data
1 Permanent Historical Data Set (PHDS)
User
Interacts with
List of Classifiers from Selected Trading Map
Classifier Details
1.3.1
Create a new Classifier
Classifier Details
3.3 Classifier Identity Data Set (CIDS)
1.3
Trading Tab
Confidence Trading Value Details
1.3.2
Add Confidence Trading Values to a Classifier
Confidence Trading Value Details
3.4 Confidence Trading Data Set (CTDS)
List of Confidence Trading Values from Selected Trading map
1
Binance Price History API
Provides Current Values of Currencies in Live Trading instances
2 Temporary Current Data Set (TCDS)
Provides List of Current Trading Instances
4 Live Trading Data Set (LTDS)
2.1
Create New Live-Trading Instance
Trading Map
Trade Data
Trading Map
4
Trading Handler
(if Pressed Start) Commit Trades
4
Binance Trading API
Selected Live-Trading Instance
2
Monitoring Hub
Clicks "test"
2.2
Test Live Trading instance
Clicks "start"
2.3
Start Live Trading Instance
Selected Live-Trading Instance
(if Pressed Start) Binance Trade Data Returns
5 Trade History Data Set (THDS)
Return Binance Trade Data Returns for current Live Trading instance Selected

Readable Diagram

4 Work Breakdown

4.1 Creation Hub

The Creation Hub is where you will be when setting up your app. It contains two tabs: Learning Tab, and Trading Tab. On the Creation Hub you are not stuck with only having one setup. You are able to save, and open other setups. Each setup is based on one CURRENCY-CURRENCY/STOCK pair. Although, they can be duplicated. And the type of coin can be changed. Note, the seeds of data will be used from the other pair. So you should be werry.

4.1.1 Create new Trading Map

When creating a new Trading Map you will need to fill in some information about what it will be about. First off you need to select the trading pair you will be using. Ex: USDT-BNB (binance).

You then will have to give an API key for the service to have the trades committed (Starting as personal use - so security is kind lax)(Not required, can still to learning without). More advanced settings relative to ML may be added later.

4.1.2 Learning Tab

The Learning Tab is where you will be training the model for your trading bot. This will be creating `labels` of most-likely trading indicators. And, providing seeds with examples of where this `trading indicator` has happened in the past data with the current pair. And, also providing examples of what is not an indicator as a separate label.

4.1.2.1 Create a Label

Creating a Label for an indicator will require a name, and a colour (So it can be distinct from other labels.

4.1.2.2 Creating Seeds (examples) for the Label

When creating seeds for a label, you will first have to select which Label you are adding the seed for. Then, find a time-frame on the trading view chart where the idea of this label has occurred, then select it. Labels should be similar enough that they all follow a pattern, but not similar enough that they are all the exact same. The goal is to cover the spectrum of all possibilities of this Label, Not flood the system into thinking that it needs to be perfect, or that everything is this label. Note: you will have to create at least one label to select sections of the tradingview that are not any indicators. This is so, we can distinguish what is, and what is not a specific label. When creating these seeds, the system will automatically be determining the best buy and sell points during the indicator.

4.1.2.3 Testing Label on Timeframes

Now as you notice, after you've started to select your seeds you can browse through the timeline with the classifier tool, and select certain points to see what it would estimate is going to happen after that point. Then, you can see whether it is following what is actually happening or not.

4.1.3 Trading Tab

The trading tab will be where you take all the Labels that have been learned in the Learning Tab, and set required confidence levels and such that will be required before committing an actual trade for the Trading Map. And, depending on the magnitude of the confidence how much should be invested in a trade. Example: there will be a max amount to trade set. And if it was 90%+ confident then it would trade the max amount. But, the user can set a min amount that requires lets say 60% confidence. Then if it was in between it would calculate the amount between the min and max. Note: if the minimum confidence was never reached, then the trade will never happen.

Another idea is to create a classifier that chains Labels, so maybe whenever one label happens another label is likely to happen. So, we can create a setting to watch for this.

4.1.3.1 Create a Single Label Classifier

Creating a classifier you need to select a label, set name of classifier, colour, and the confidence interval required for this to be activated. Then, you can use the “spreadsheet” to match the min and max trade amounts in the base currency. More settings will be added in the advanced section during development.

4.1.3.1 Create a Multi-Label Classifier

Creating a classifier you need to set name, colour, and the confidence interval required for this to be activated. Then, you can use the “spreadsheet” to match the min and max trade amounts in the base currency.

4.2 Monitoring Hub

On the Monitoring Hub you are able to open Live-Trading instances (tabs like chrome but on bottom). Within each Live-Trading instance you can set one of your trading maps. Then you can click `test` or `start`. Test will be keeping track of what you would be trading (with fees) and tell you how much you would have gained, or lost over a session. Recommended to do this a lot to check and see if your training map is actually something special. Once you think your map is ready, you can click start and the trading will commence. Note, you have a base currency and alt currency set. Whenever you stop the Live-Trading instance, if any life trades are active, it will convert all currency back to the base currency.

4.2.1 Create new Live-Trading instance

4.2.2 Test Current Live-Trading instance

4.2.3 Start Current Live-Trading instance