Class: CS 370

Semester: Spring 2022

Assignment: D1

**Crypto Bot**

Project Type

* Development

Members

Ernest Duckworth

* Sophmore Software Engineering

Samuel White

* Sophmore Software Engineering

Sam McKay

* Junior Human-Centered Computing

Hardware Details

**Hardware List**

* Raspberry Pi 4 Kit (4gbs)($150)
  + Internet connection
  + Display capabilities (HDMI)
  + Power Connection
  + Mini-HDMI cables
  + Fan
  + Case
* Sensor(s)
  + Laser Fingerprint scanner ($30)
    - With USB to TTR connection for easy use while
* Required Additional Components
  + Power Source
  + Ethernet Cable
  + Display (personal is fine for now)
* Total Expenses: ~$200
  + Websites Ordering From
    - [Amazon](https://www.amazon.com/s?k=optical+fingerprint+sensor&tag=754u-20&ref=nb_sb_noss_2) Ordered: 2/25/2022
    - [CanaKit](https://www.canakit.com/raspberry-pi/pi-4-kits) Ordered: 2/25/2022

**Hardware Explanation**

* Raspberry PI 4 (4gs)
  + The Raspberry PI 4 with 4 gigs of ram is our best option
    - 4 cores for our multi-threading
    - 4gs of ram (allow us a larger data set per coin)
    - Easy integration for sensors
      * Fingerprint Scanner
* Fingerprint Scanner
  + The fingerprint scanner returns a hash value for every person's fingerprint
    - This allows us to encrypt everyone's data with a different hash value
    - This way the only person who can see encrypted data is the person with the fingerprint
      * Allow us to store sensitive data locally
* Additional parts
  + Fan for keeping the PI as cool as it can while running the program
  + Case + Kit to keep PI safe so the group does not have to buy multiple through the time frame of the project

Software Details

**Basic Info**

Language:

* Python 3+

Libraries:

* Pandas
* Pandas-ta
* Matplotlib
* Ccxt

**Models**

* *CoinApi* 
  + Connection to the coin market through CCXT
* *CoinData*
  + Individual coin candle and indicator data
* *CoinScanner*
  + Constantly reads and fills dict or array of CoinData
* *Analyzer*
  + As CoinScanner reads analyzer will determine what should be done
* *Trader*
  + *PaperTrader*
    - Trade fake money in the real market
  + *MarketTrader*
    - Trade real money in the real market
* *Logger*
  + STDOUT msg printing, record.txt, Discord messages
* *Bot*
  + Utilizes all of the models

**Threads**

*Main:*

* On entry, it will make a user sign in or create an account
  + Creating an account will require the user to either enter their coin base API keys or skip entering their information and run paper trading only
  + Set validated information in config
* From here they can select a list of coins they are interested in
  + Have a default list for easy access
    - Can filter information out of the default list
  + Can add their own coins
    - We will have to add a test to ensure user-entered coins exist
* Now Reader can be created with their selected Coins
  + Trading style can be determined as well (paper trading or coin base)
* From here the main process will wait for keyboard arguments from \*client\*
  + Keyboard arguments could be
    - Account details
      * Total in account
      * Total in various coins
      * Possible percentage data depending on API capabilities
    - Certain coin information display
      * Display coin in the various time frames (1m,5m,10m,30m,1h, etc)
      * Display coin with indicators
    - Display top watch
      * Coins with the highest `Analyzer` rating
    - Quit/SignOut
      * Close the threads and exit the program
    - switcher
      * Switch to new user
        + Signout && sign-in
    - Etc…

*Scanner:*

* The bot will scrub the current market fetching new candles for selected \*\*Coins\*\* as they are released
  + Fetches are grabbed by their timestamp
    - In the case of a fetch failure
      * Wait and try to connect after a small amount of time
      * Keep running count of consecutive failed fetches
        + If it reaches over 5 in a row end program with an error
  + This is done in a forever process(single thread dedicated to running this process alone)
  + Will be done sequential loading with all coins
    - Coins will not be loaded concurrently (on multiple threads)
  + Once all coins have been downloaded the reader will execute the `Analyzer`
    - And go back to trying to fetch new candle data
  + Maintain the `dataFrame` size, don't allow any coin to have over x amount of candle info

*Analyzer:*

* Will be created by `Reader`
* On entry, it will begin analyzing the data of all the coins in the data stream
* Algorithm
  + Use various trading strategies to create a weight for if a coin should be bought or not
  + Can be on various strategies
    - Indicators
    - Speculative
      * support/resistance
    - Etc
  + As weights are determined keep a sorted list of which coins have the highest weights
  + Depending on strategy
    - Create a thread to execute buy orders on hit cases
    - Wait until the end of the analyzer to create a single thread that executes multiple buy orders
      * In this case, we could not create a second thread and just pass the process into a buy/series where it executes a query of buy/sells
        + On each successful buy/sell another thread could be created for `Notifier`
* Close thread and return control back to `Reader`

*Buyer:*

* Will be created by `Analyzer`
* On entry, it will begin
  + A: Executing a single order
  + B: Executing a handful of orders
* Determine which platform is being used
  + `Coinbase`
  + `PaperTrader`
* Execute the trades
* Close thread and return back to `Analyzer`

*Logger (optional):*

* Will be created by `Analyzer` after successful execution of `Buyer`
* Connects to the discord server
* Prints buy/sell the message
  + Also prints message to STDOUT (optionally)
  + Appends the message to the end of logging file (undetermined file name)

Further Details

**Thread Status**

* The max amount of threads that this program may be running at a single time will be \*\*4\*\*
  + Most of the time it will only be running \*\*2\*\*
* Since the `Main` will have to communicate with `Reader` for a lot of its processes
  + There will have to be a 2-way communication process between the 2 so that `Main` can request information from `Reader` and `Reader` can return the requested information
* `Reader` will not necessarily need communication between the buyer just be able to tell if it returned with a success or failure message(determine if buy/sell was successful)
* `Main` will have to be able to access its own API to the market so data can be fetched for the user(does not come out of our backend unless the client wants the same candlesticks that are used in the backend \*\*1m\*\* by default)
* `Logger` could be sending out multiple notifications to multiple different platforms if possible

**Justification**

Our overall goal is to create a self-contained crypto bot that can operate solely on its own, it will have the capability of storing multiple ‘user’ accounts. Users will be authenticated through an optical fingerprint scanner. We are going to develop the project with the intention of the program being stored on a Raspberry Pi 4 (4gs). We chose the Raspberry Pi 4 (4gs) because it has quad-cores and 4gs of ram which will be more than enough to run the crypto bot and comes with WIFI and Bluetooth capabilities. We didn’t go lower or higher with our selection of the ram mainly because of funds and availability restrictions and as we continue to develop we didn’t want to run into an issue with memory if we got a lower amount of ram (1-2gs). Currently, our plan with the software plan is to develop it in Python so that we have access to tons of libraries that are already set up. For example, the python library ‘ccxt’ is an amazing library that allows us to access several different crypto markets API. We intend to research a little further on what language/library we are going to use before the raspberry Pi arrives (hopefully next week or so). All in all, we expect that the project is going to cost around $250.

**Back of Napkin Drawing**

