**Cryptocurrency Prediction Platform Final Report Document**

**By**

**Foster Sandlin**

**Introduction:**

This new system that i'm implementing is an order based product system that provides users with neural network cryptocurrency predictions. The user will place a order for the desired coin from a list on the web portal, after the user has selected a coin they wait the designated amount of time which will be less than 12 hours and then they will receive their prediction on the UI. The UI will contain a list of coins that can be selected as well as a designated chart area which will contain the prediction model chart when the order is finished.

**Problem Statement:**

The problem I wanted to solve or the goal I wanted to fulfill was to create a tool that someone could use to utilize for next day predictions of currencies given a previous day price. With this tool a user wouldn’t have a definite answer obviously as what was to come but they would be able to use this tool for tips and possible guesses in estimating what will come next.

**Background:**

The idea started with making a utility that would allow you to analyze next day prices by using there previous day price and would learn overtime and get better at predicting next day price with feeding it data from years of opening and closing prices on a coin. With this utility in mind I had an idea of making this an application that you would install and run but in this circumstance I integrated the utility within a web platform that would allow people to go and place predictions and wait for the neural network to complete where they can view a set of graphs after explaining what the neural network produced.

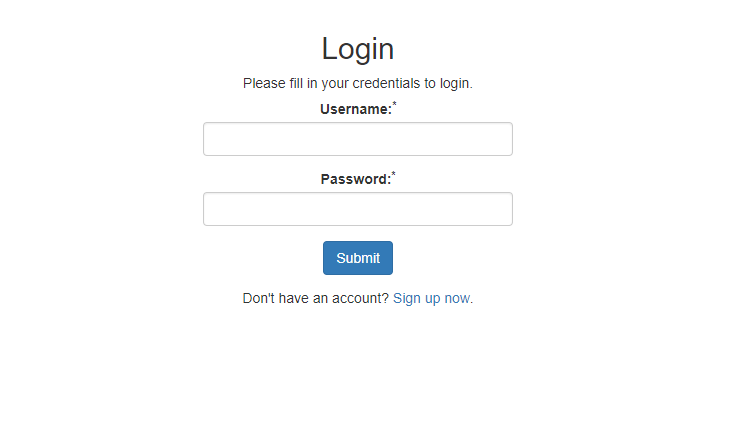
**Documentation with Code:**

The core algorithms of this project happen after you place your order. There are also many smaller algorithms used for preprocessing the data before placing your predictions. Some of the smaller algorithms before i get to the core are verifying account info when logging in and also account creation on the web server. When reaching the home screen there is a “Login/Logout” button on the top right which if your logged in and press this button it logs you out and if you are not logged in, this button will bring you to the login page.



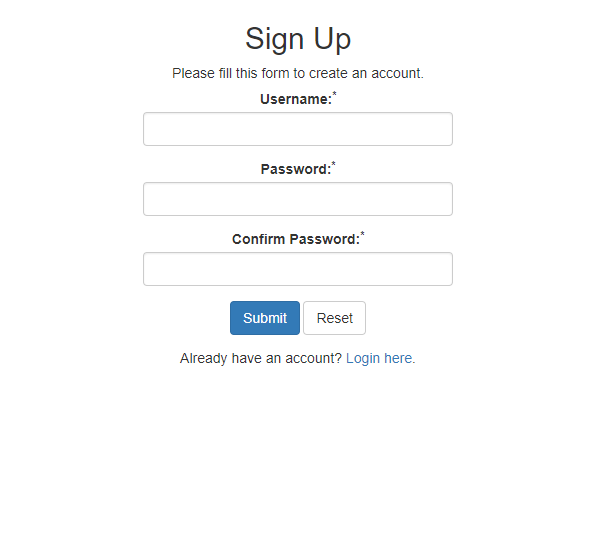
The Login page is presented below:

Fig 1:



If you haven't signed up then the “Sign up now” option allows you to do so here.

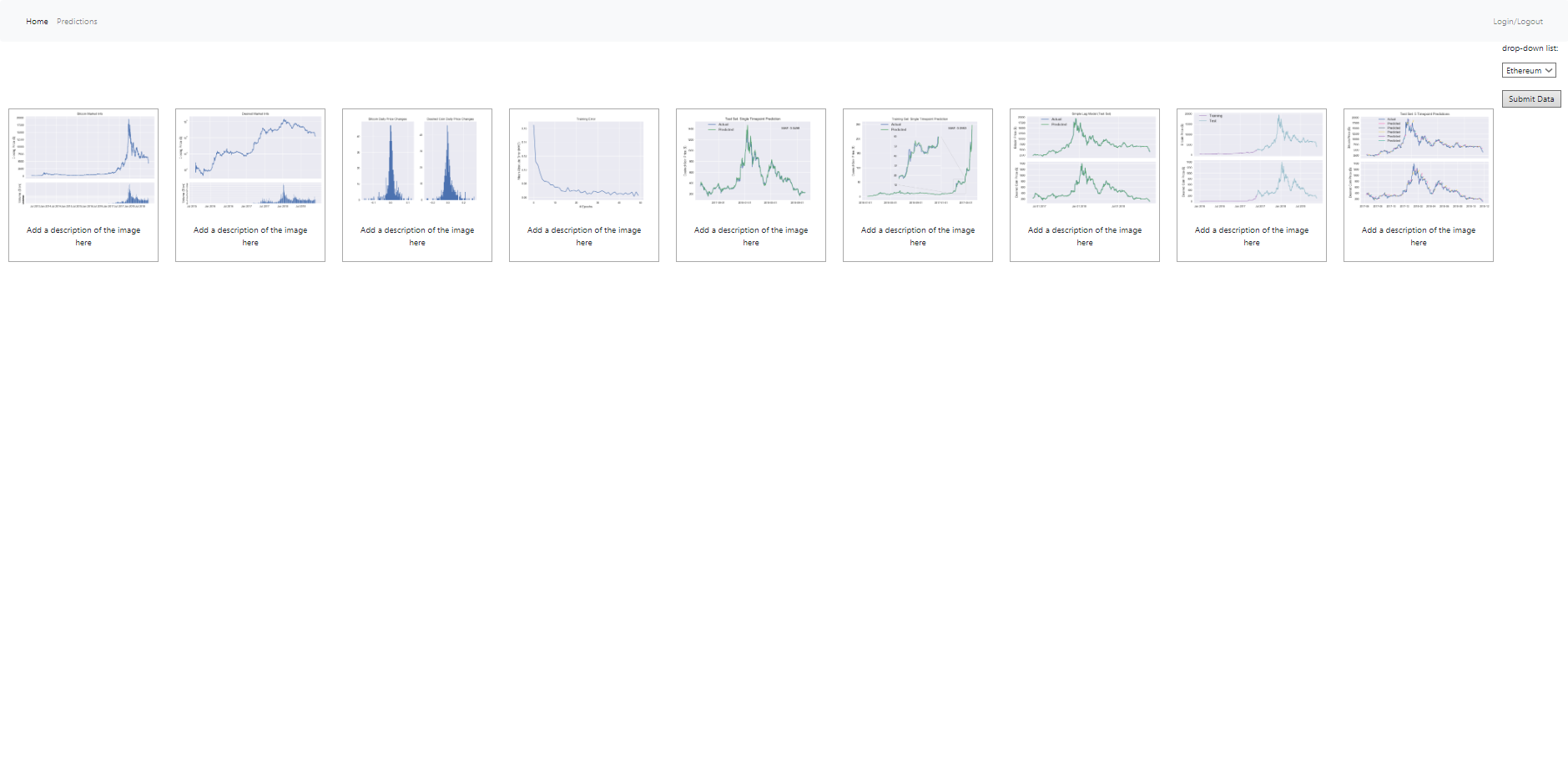
Fig 2:



After you have signed up it will forward you back to the login page where you can now sign in with your new username and password, this page will also tell you if your password does not match and if it isn’t complex enough for a password. After this you are then brought to the main home screen where you are able to view your predictions you place.

Home Screen is presented here:

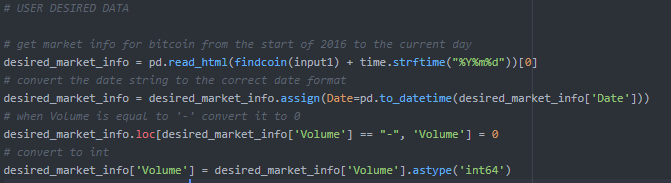
Fig 3:

****

Once on the home screen then you are able to select a coin from the drop-down and click submit to place your order. This then will start the python file which web crawls to a url location that stores large datasets of cryptocurrency history and starts to build the dataset for the prediction. After the web crawler has got its dataset a couple of formating tweaks need to be done before the dataset is ready to be fed into the neural network like normalizing certain columns and replacing certain values that were not numeric to numeric values. The data is collected dynamically through having python read a html url and collect the historical data of the selected coin dating back to 2013 to present.

Here is a picture of the data collection and normalizing and cleaning up certain columns:

Fig 4:



The resulting datasets are then displayed in graph form and then pushed to view for the client to see.The data is then merged with the same dataset but for bitcoin and is put into a train test plot for display for the user so they can visualize what train and test will look like for the dataset.

Fig 5:



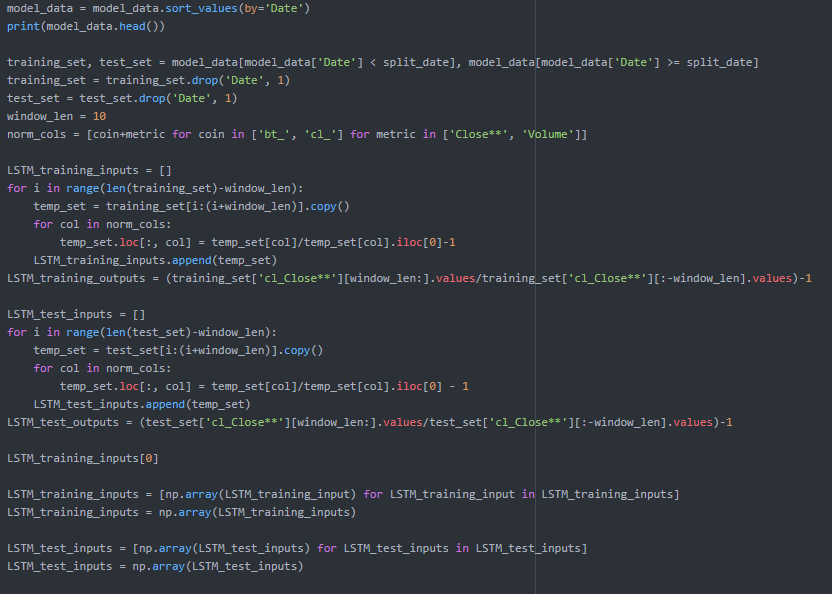
After that then a trivial lag model is displayed with its model predicting next day prices with previous day prices as shown below.

Fig 6:

****

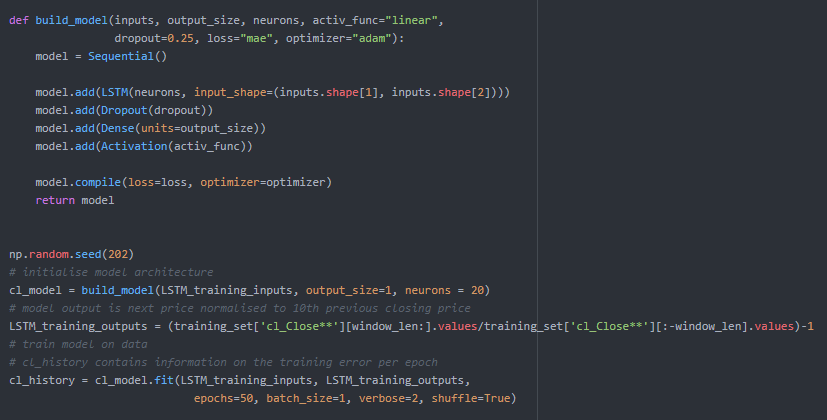
Then another graph is displayed following daily price changes between btc and the users desired coin in a graph that shows the differentiation between day to day prices. Then the data is reversed so that the subsequent rows represent later time points and is then started to be declared into arrays for the LSTM model after the training and test sets have been declared.

Fig 7:

****

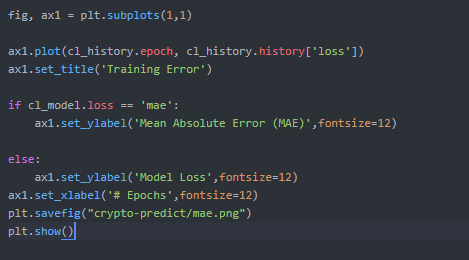
Everything's been declared and set for the model so then the model is built with its parameters at this moment in time.

Fig 8:

****

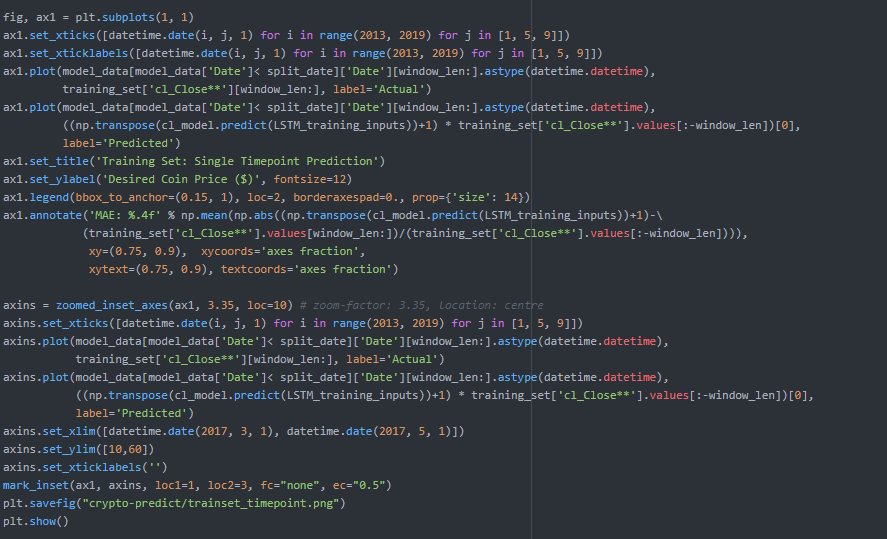
The model is set to report the mean absolute error after each epoch also as seen here.

Fig 9:

****

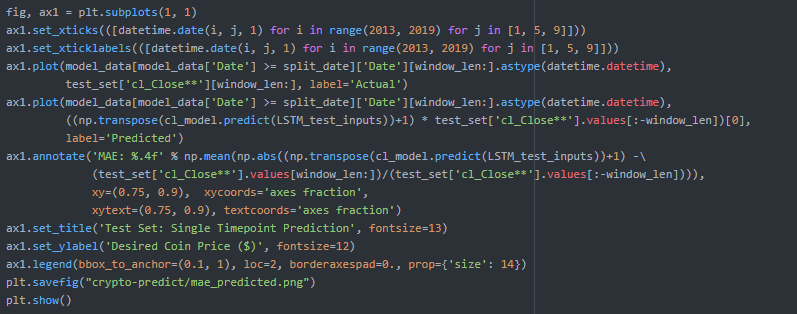
After the model has run it outputs its results and the results are graphed here is training set single time point prediction.

Fig 10:



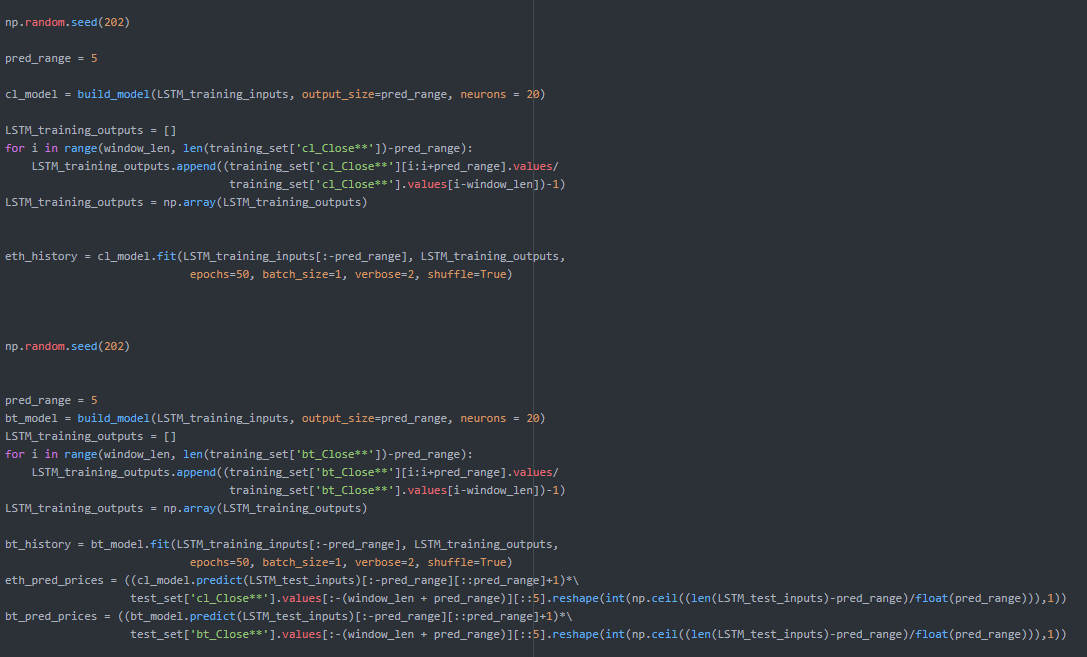
Single time point gets displayed and also predicted mae is displayed too. As seen here.

Fig 11:

****

After single time point is done we then set it our arrays and try prediction for next 5 days, run the model and display our outputted data.

Fig 12: Array and model declaration



Then the outputs are displayed in graph.

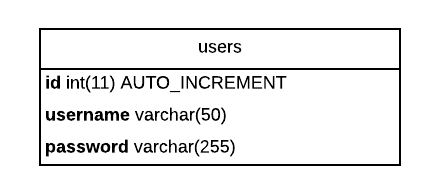
Fig 13:

****

After all graphs have outputed this means that they are being saved and presented on the web server also. So you are able to see the graphs in real time as they are saved to the web server.

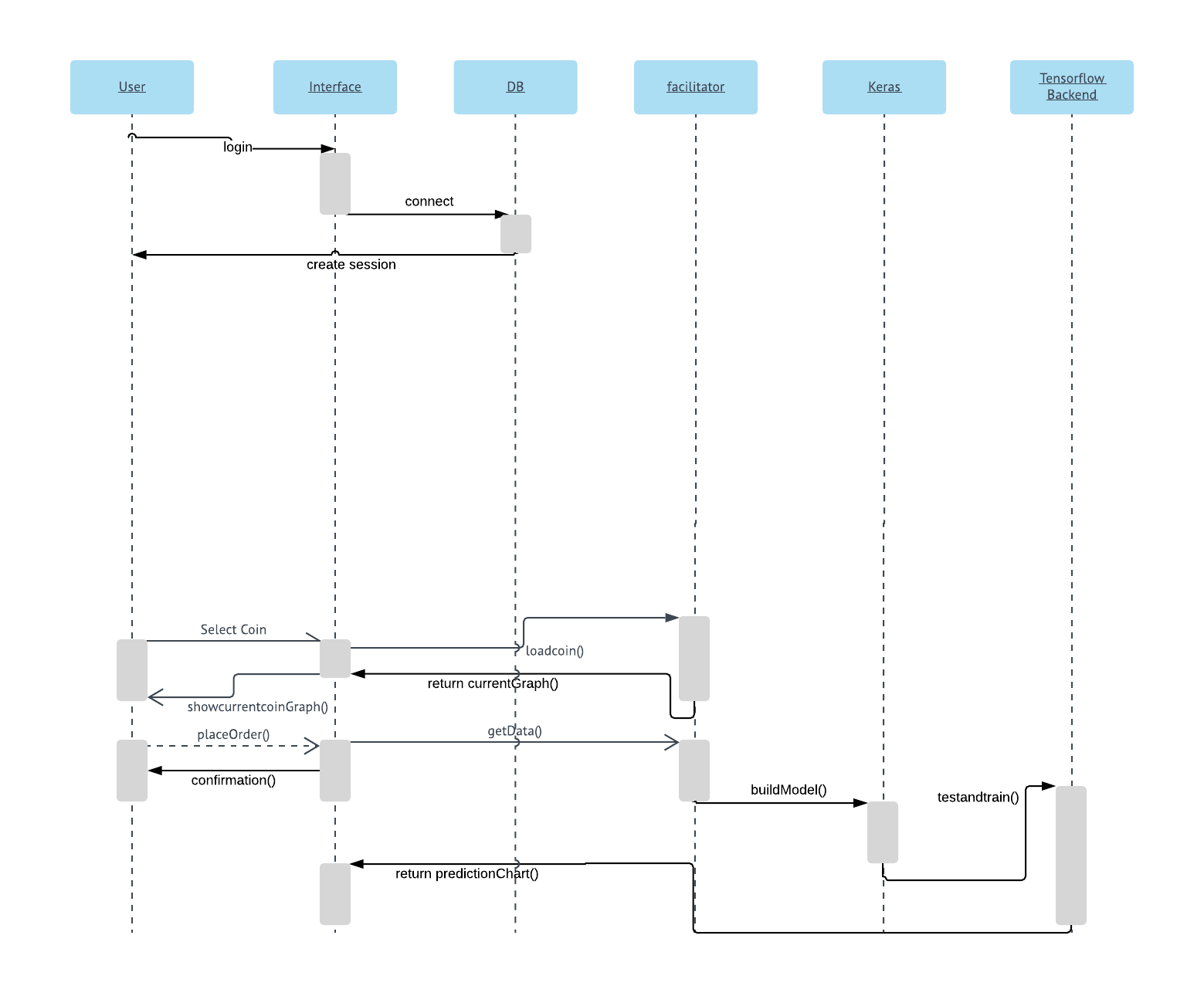
All the graphs you see above in screen shots are made with matplotlib and anything using np as you can see is NumPy which, for example is what is producing the random seeds above. The specific plots generated are plots at first that show the entire historical data visually and then moving on to display visually how much of the data is allocated to train and test and will show daily price changes in your desired coin compared to bitcoin. These plots are helpful to see all of what is going on behind the scenes of the UI and after the informational graphs display test random walk lag model before the LSTM and then following with beginning to train and run the neural network.

**Database:**

The database used is a google cloud platform mysql database which I use a deploy as a service app engine with phpmyadmin and have it pointed at the database to look and manipulate it. The Api’s used are many basic python apis like keras, panda, matploit. The database is just set up as a user login for right now. When you login it checks the user table for your id with right username and password and logs you into the home page. If you haven’t made an account you can make one that inputs your desired username into the database and desired password. These passwords are hashed so the database admins or anyone who has access to the database does not have access to anyone’s passwords. The database table ‘users’ has the following columns as you see above, a 11 digit integer that is generated by mysql which is unique and acts as the index on this table, then is username with max characters 50 and password with max characters 255 both varchar.

**Workflow and Graphs:**

Fig 14: Original Workflow



**User Interface and Design:**

Fig 15: Original Design

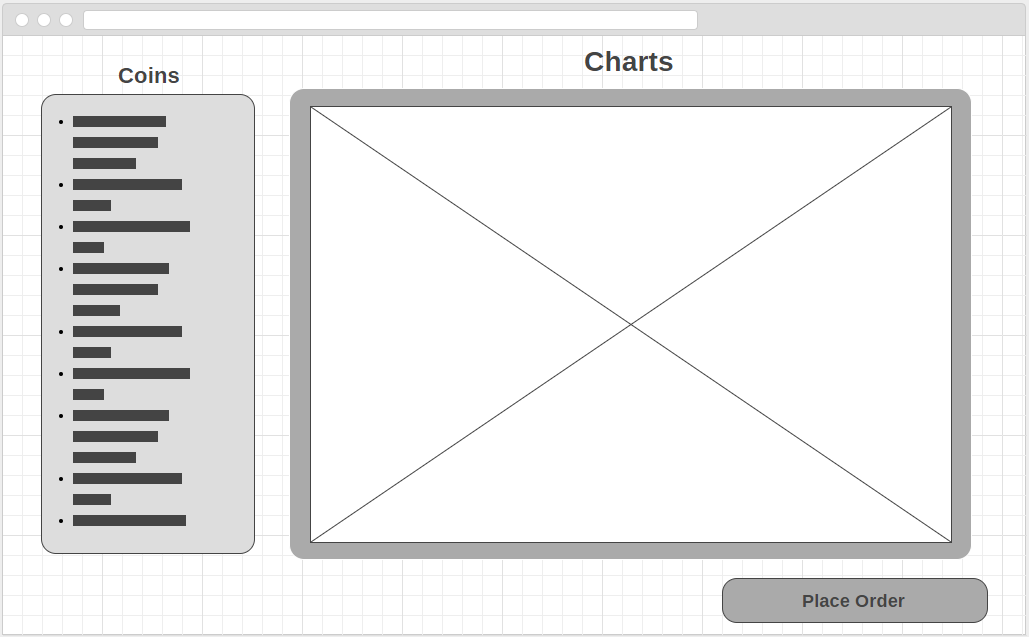
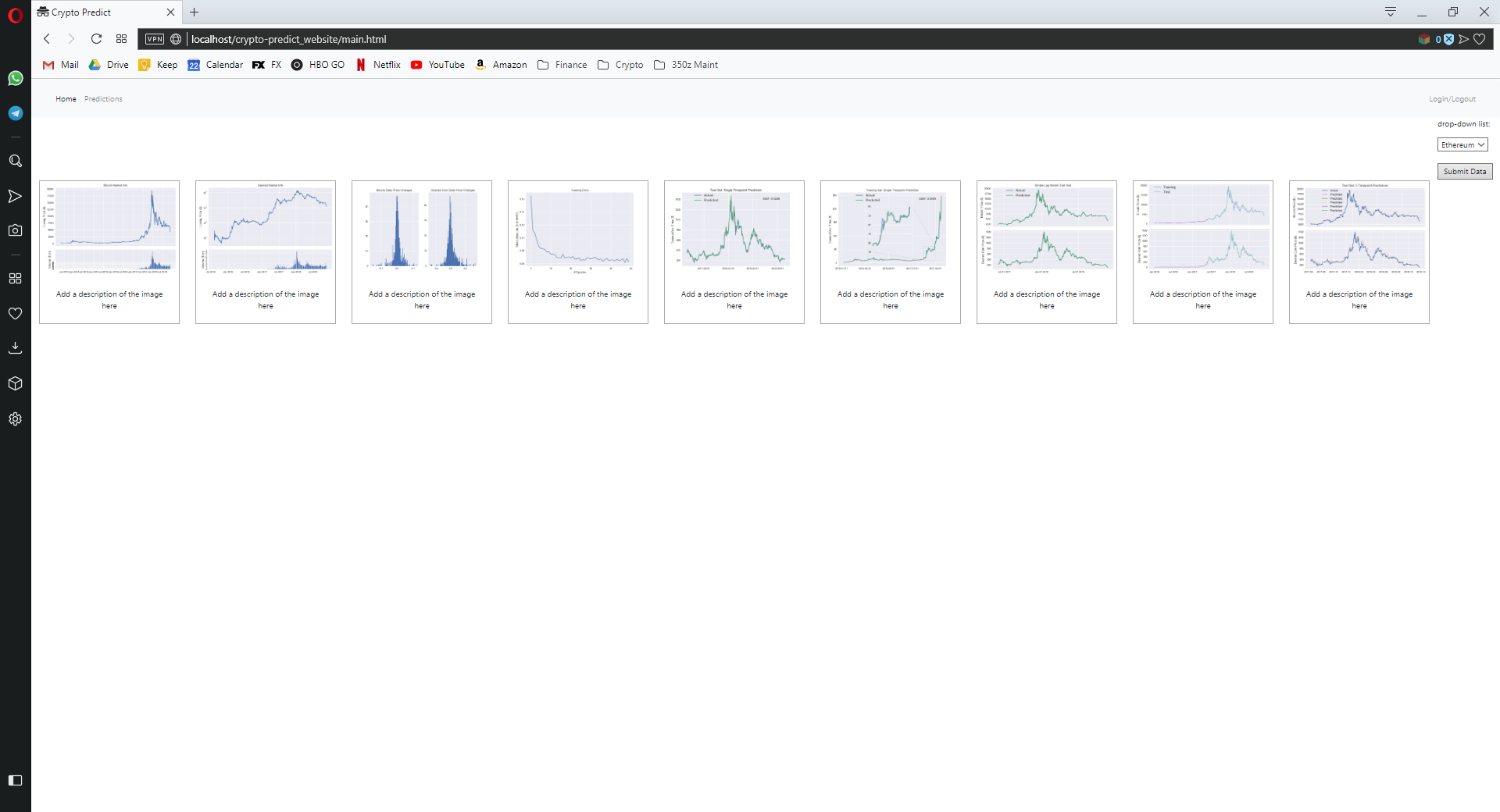
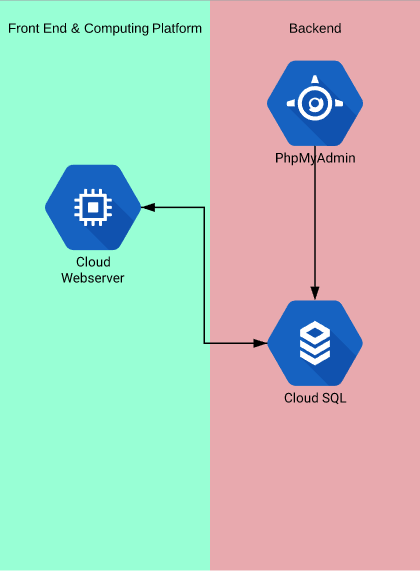


Fig 16: Main Home Screen

****

**Infrastructure topology:**

Fig 17: Infrastructure Topology



**State of the Implementation:**

The state of the implementation as follows:

When first landing on the web server there is a login feature. This feature allows you to login with pre existing account or click sign up and will bring you to a page where you are able to create a account. The account creation process has some small features like checking for password complexity and being able to verify password matches. Accounts are stored in a database and you can log in once you have created an account. After logging in you’ll notice that everything has session also which is another must have feature. Once on the website you are able to view pictures of your predictions and that are outputted by the python file in the background. This feature works entirely but at the moment, placing predictions on the website is not fully implemented because of issues with processing power on the cloud. The website allows you to fully view and save your png pictures of the predictions you get which is its primary purpose. Eventually I would like the database to save all of your previous predictions so you could sign in and go back to older pictures but that isn’t fully implemented at the moment. At this moment in time the you can only save images as they show from the immediate query you run. Another feature i added is that the passwords are hashed that you log into your account with which is also a small nice feature for the database. The website as of right now needs a couple more things but everything else is implemented correctly. I right now am having issues running my python file on a cloud vm because of dependency issues but it can be resolved. Like I said before though the getData.py file that does most of the heavy lifting is very process intensive so it would use almost all the resources the webserver has to offer at the moment because it is not set up to operate on a GPU. If i could get it integrated with GPU on google cloud with keras then it would help the efficiency by a lot but it would still cost a similar amount.

**Testing and Evaluation:**

Testing and evaluation of the project was done locally. Testing the implementation of the project was done by me locally through hosting the web server on a localhost on my machine using WAMP. Then through that I was able to test my main python files I developed in Pycharm IDE. The python files took a lot of testing and still will continue to need more. A lot of the testing was done with formatting graphs and also the neural network. The webcrawler that starts the algorithm and goes out to get data from a url is also another processes that needed some testing. This had to be right because i need the data to be formatted exactly how i wanted it so I could normalize certain columns so then the data could be fed into the neural network. Getting the neural network to the point where it wasn’t such an extended amount of time was also something that was tested. After testing I left off that the entire process from putting a order in to getting results only takes minutes and that is okay for me right now. Later I can go back and tweak it more if I would like to change the neural network. Once everything was good to go on my local environment for testing, I pushed my local website to my centos 6 apache web server. Since I made sure the database and configuration settings were all set and I already had my database which was created via google cloud platform, it was very easy to move everything up. Once everything was up on the web server it was just making sure the environment had the same dependencies as my test system like anaconda.

Further testing does need to be done on the neural network though like stated above. Testing the neural network for more viable and optimal solutions could be done also but as it sits right now it does what is intended. A lot of the testing that still is possible to do isn’t entirely necessary but would be good to experiment with. Experimenting with the neural network parameters would be something that could further the proficiency of the platform even more than I already have.

The project does meet the objectives i set out which is to provide a utility tool for predicting next day prices based on previous day price and accomplishes this through training and testing the neural network. This tool of course is not fully accurate because it is impossible to do so with 100% accuracy but it is a useful tool for people to use with there own due diligence.

The system is usable but will continue to need more implementations for ease of use if it ever became a full product. If it became a full product there are also some hurdles for overcoming processing power for multiple orders at the same time but with the right infrastructure i believe it could be possible in time.

**Lessons Learned and Reflection:**

Most of the project went well, I believe this could be done perfectly if I knew a little more about python deep learning. If I could go back and do it again I would have probably picked something I was a little more familiar in. Even though I took datamining last semester it was still a daunting task to complete especially with added features and making it into an actual platform. If I was to do it over I would most likely do it as a research project instead of a platform which would be a little bit less but still have the core components of the project. This would have allowed me to really experiment more on the actual core algorithms rather than working on standing it up as a platform. Even though i'm familiar with setting up web servers and databases getting everything integrated was somewhat a challenge for me the things that went well was linking the infrastructure together and getting everything interlinked. The harder part was working with api’s i'm not as familiar with and getting matploit to act the way wanted considering I haven’t had a great amount of time working with plots besides my semester in data mining. This problem also arises when moving my test system to actual cloud vm. Getting everything setup like anaconda and making sure python scripts are able to be run on the apache web server are important and that's something i have a little less experience with but this was probably my first production type implementation.

When the semester ended I was given a extension on the project and was wondering if i was going to be able to actually complete this platform as it sits now. Since then i have definitely solidified the majority of the platform and really enjoyed working on something that challenges so many skills as a programmer.

**Version 2:**

Well as I already had extra time to complete the project I will touch on what I would do even after this extension because there is always more that can be done. The additional features I would implement would be a more solidified remote processing for the neural network. What I mean by that is i would like to be able to get the neural network processing in the cloud and that would really change this to a serviceable platform. With this you could then add even more features for e commerce type of platform and also be able to look at previous predictions saved on your account through the webserver like I stated earlier. With an implementation like this you could have each account with a payment method or use cryptocurrency as the transaction type and you would be able to exchange these currencies for predictions.

The only problem is that with implementing it like this i would have to solve my data issue with processing the neural network in the cloud which would also technically be a new feature too add. Of course along with the the new features I would clean up the UI also and could add even more features like sms support where it emails you when your order is done. If the processing problem could be solved then you could go into processing the neural network using a cloud computing gpu to run the algorithm. This could potentially be much better than CPU processing and depending on what your price is for the predictions you could have a decent return on investment and would actually be able to make profit on this platform.

**Professional development and Lifelong Learning requirements:**

Over the course of the project I have definitely solidified a lot of my development skills. In the beginning most of my development skills were moderate but I had to use many tutorials because I forgot some key components over the years. Some of key components were things i had to touch up on were just setting things up correctly, along with simple syntax in PHP and CSS. This project helped me solidify those into an actual skill which i'm very grateful for and also learned more about different frameworks and libraries available with these languages. Working with tensorflow and keras was great. Being able to see what these frameworks can do for neural network computing is amazing and this has lead me to see much more applications and use cases that can be applied to neural networks. Like i mentioned before this project really helped me solidify a lot of HTML, CSS, and python. For languages this is really most of the project but I also became much more familiar with debian CLI and cloud computing too working with VIM which I believe to be an extremely valuable tool in the field. Setting up cloud infrastructure was also a backbone of this project and I got to get familiar with Google Cloud Platform which is another useful platform to learn. Many of my resources were just online forums and other peoples experiences, I asked around on the internet and was helped by a few people who have done similar things which definitely helped me gain a better understanding for the task I wanted to do. The biggest challenge of all of it was getting all of the key components to mesh and talk together. Developing the individual pieces was easier to troubleshoot than the small little configuration problems when dealing with the large picture of the project. This is something that I will definitely have to work on and just solidifying my skills as a programmer help with this, even throughout the course of doing this project I can say I have become a much better software developer from analyzing a idea and what it entails for the amount of time and resources needed to complete the idea you have. This is very good for gauging what you can and can't do by yourself, not saying something isn’t possible by yourself but you must be able to analyze the idea and see how long it may take you to implement the idea and this was something i definitely got more familiar with considering i believe the project i took on may have been a little more elaborate than it should have been and i probably could have picked something I was more familiar with but regardless I’m happy I did something difficult, the reward is always more satisfying.