# Part A: Letter of Transmittal

14 June 2025

Jay Houston, CEO

HCOMM

1234 Computer Rd

Dallas, TX 75231

Dear Mr. Houston

I am pleased to share this project proposal that introduces a new data product designed to help HCOMM make faster and smarter investment choices. Right now, too many hours and resources are spent on manual research methods. This data product can speed up that process, reduce effort and help to choose the best investments easier.

This proposal will explain the business problem, the benefits of the solution I came up with and how it will be built and used. It also covers costs, impact on HCOMM and other important points. I believe this solution will help HCOMM grow and save money over the long-term.

Sincerely,

DeShawn Houston

DeShawn Houston, Technical Project Director

**Project Proposal**

**Problem Summary**

HCOMM is still using old and slow methods when making investment decisions. These methods take a lot of time and cost the company money in the form of extra work hours. Employees are spending too much time looking up data, checking reports, and doing manual research. This delays decisions and makes it hard to react quickly to market changes. When decisions are slow, the company can miss good chances to invest and make a profit. The process is also tiring for workers and leads to wasted effort. Because of this, HCOMM is not getting the most value from its team or its resources. A new, modern solution is needed to solve these problems.

**Solution**

The new data product will make the investing process much faster and easier. Instead of spending hours gathering and checking data by hand, this tool will collect and organize all the important information in moments. It will also check each stock against proven technical indicators to find the ones that are most worth looking at. This means managers will only see the best options, without being overwhelmed by useless or risky choices. Decision-making will become quicker and simpler. The tool will guide the team to the right choices without guesswork. This can improve results, reduce wasted effort, and give HCOMM a clear edge in a fast-moving market.

**Outline**

The data product will work by gathering important stock data every day. It will sort the information into groups that make sense for investing, such as price movements and technical signals. After sorting, the tool will study the data using math-based indicators like moving averages, price patterns, and more. It will make predictions based on the numbers alone, without using feelings or opinions. These predictions will help decide if a stock is worth buying or selling. The data product is designed to think like an investor who always uses facts and figures. It will be simple to use and easy to understand for the team.

**Data**

The product will use stock market information from the past 10 years provided by the yfinance API. This will include daily prices for when the market opens and closes, as well as the highest and lowest prices for each day. It will also include several technical indicators such as moving averages, price patterns, and volume changes. In addition to these, the tool will check the MACD (moving average convergence/divergence) line, which shows if a stock is trending up or down. Sentiment analysis will also be used, which looks at news opinions about the stock to see if they are mostly positive, negative or neutral. All this data will help the tool make the best predictions possible.

**Objectives and Hypothesis**

The main goal of this project is to predict if a stock should be bought or not. The tool will make this prediction based on how the stock has performed in the past and how it is performing now. The goal is to help HCOMM make decisions faster and more confidently. The hypothesis, or idea being tested, is that if the tool is trained well enough using this data, it will be able to predict if the stock’s closing price will be higher than its opening price for that trading day. If the tool is right most of the time, this will help the company make better trades and earn more money.

**Methodology**

The project will be built using an Agile method. This means the tool will not be made all at once. Instead, small parts of it will be created and tested one by one. After each part is checked and works well, the parts will be put together to make the full product. Each time something new is added, the team will make sure nothing else breaks. This flexible way of building the tool makes sure that any changes or problems can be fixed right away. The project team will be able to adjust their plans without starting over. This helps save time and money while building a stronger tool. Normally, Agile methodology is referenced by applications that are deployed incrementally but this is not the case here. The same adaptability and CI/CD ideas are used but the product will not be deployed until it is fully featured.

**Funding**

The project will need some funding to succeed. A MongoDB Enterprise subscription, which will hold and manage the data, costs about $14,990 per year for each server. The project also needs skilled Python developers, who make about $125,000 each year. In addition, a MongoDB Database Engineer is needed, and they make about $170,000 per year. The computer system itself will also cost money. A cloud-based system could need $20,000 upfront and about $1,000 each month for support and upkeep. These costs are necessary to make sure the product runs smoothly and meets the company’s needs without errors or delays.

**Impact to Stakeholders**

The new data product will help everyone involved in the company. Managers and decision-makers will save time because they will not have to look through pages of reports or search for the right stocks. This means they can make decisions faster and with more confidence. Workers will also spend less time doing research and more time on other useful tasks. Because of this, the company could explore new ways of investing and make more money. Investors and owners may see higher profits because the company can react quickly to market changes and take advantage of good opportunities before they are gone.

**Ethical and Legal**

This project does not have any major ethical concerns. The data product only uses facts about the stock market that are public and open for anyone to see. It does not collect personal or private data from people or companies. For legal matters, the team must make sure that all the market data being used is allowed for business use. As long as the data is not under special rules or restrictions, it is legal to use it in this way. The team will also check any licenses or rules before adding new data to make sure everything is done properly.

**Expertise**

I have many years of experience writing programs in Python, the main tool for building this product. I have also led projects like this before, managing each part from start to finish. My background includes working with machine learning, which is the technique this tool will use to make predictions. I also have strong skills in cybersecurity, so I can help make sure the data is safe and protected. My knowledge of best practices in machine learning will make sure the product is built the right way. All of this will help the project succeed and provide real value to HCOMM.

# Part B: Project Proposal Plan

## Project Summary

In terms of an investment strategy, the company has a weakness in pipeline efficiency. Hundreds of thousands of trades are made per day across the globe. There is a ton of information and relevant points used, with or without bias, to make decisions on which trades are correct. Right now we have analysts going through this data manually on a day to day basis. This creates a lag in executing trades. By no means are we failing but there is a gap to be filled.

Our customers are all time-sensitive people. They count on our company to help manage their investments in a way that relieves them of time commitments and the stress of doing their own homework on the market. With this new data product, we can provide them with a renewed confidence in our ability to manage their assets. Seeing faster decision-making based on at a minimum the same quality of diligence we have now. Realistically, that quality of inputs will be raised and hopefully provide results that are exponentially more impressive.

The deliverable associated with the design and development of the data product is a webapp that will allow analysts to get initial predictions on successful trades based on the numbers. This product will have 3 major components; user-friendliness, expeditious aggregation and transparency. The GUI will make it easy to get information on the company being scrutinized. With a list of all the publicly traded companies available, it takes one click to make get an initial prediction as I mentioned, the confidence score of the models' prediction and a visualization of current stock performance. Touching on the confidence score for just a moment, this provides a small glimpse into how much extra effort needs to be made to make a decision. That is not a black and white process, each analyst can interpret the range of confidence scores for their own comfort but it is a useful insight into how much faith the data has. Some may see extremes as clear indicators to speed up the final decision process and others may view it as a reason to dig deeper to try and find a "sleeper." Regardless, the data being presented in such a quick manner, speeds up the process from start to finish.

## Data Summary

The data that will be used for this product is all publicly available through a number of avenues. Financial news websites, market data providers like Alpha Vantage and even government entities like the US Census Bureau. We will be pulling this data from Yahoo Finance using a python API named 'yfinance.' This data's validity is tied to it's underlying source, Yahoo. They have a decades-long tenure of financial media coverage and this will suit our needs perfectly from a historical standpoint. We need long-term data to be successful and yfinance/Yahoo Finance can provide that. The historical data will be contained in a MongoDB instance for model training purposes. This will be added to incrementally to retrain and maintain the model going forward. The data will be pulled from the live API when it is time to be used by the model after deployment. This will provide the model with current data in order to make a prediction on.

There are no ethical considerations to take into account since this is all publicly available information. The only legal consideration is to make sure the Acceptable Use Policies for any of the products used are being followed.

## Implementation

The methodology used will be the Agile method. The Waterfall method was also considered due to it's linear nature and ensuring a strict step-by-step progression trail. However, that does not suit this endeavor well. Agile provides the flexibility and circular cohesion to revisit, revise and revamp along the way BEFORE deploying the product for use. The requirements are not well defined in a way that lends itself to strict planning and development. There is a vision and an outline. Agile practices will allow the team to adapt to upcoming challenges and new considerations without derailing timelines having to completely walk back to the necessary phase that needs to be amended.

The first step for implementation is planning the tasks that need to be accomplished. This includes constructing the database, building the frontend, testing and tuning the model, etc. These will be the backlog for the entire project. The planning stage also constitutes timeline expectations. While it would be spectacular to have a next-day shipped product, that is not realistic. Proper consideration needs to be given to each item so the stakeholders can be provided a legitimate target date. Once the planning stage is complete, these tasks will break out into sprints. Simply put, the backlog will be divided up into manageable and mostly cohesive chunks to make the project come to life in a way that makes sense. Our team leads can take charge on this front while consulting the project manager on what will be accomplished, who will be accomplishing it and when it is expected to be completed. Also, providing insight on any setbacks as there inevitably will be. Then the development happens. The developers and engineers can start making a reality out of the idea. Making sure to bring their abstract, 100 percent functional piece of the project is delivered. Testing is next. Unit testing, regression testing, integration testing, system testing, user testing, etc. This is the most important phase as we can "break" our project a thousand times to make sure the end user has the best possible experience. This will be rigorous and repetitive. Which is also why the Agile method works so well for this project. Once things are broken, changes will need to be made.

## 

## Timeline

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Milestone or deliverable** | **Project Dependencies** | **Resources** | **Start and End Date** | **Duration** |
| Requirements | N/A | Stakeholders, End users | 1 July – 14 July | 25 Hours |
| Task Planning | Requirements | Project management team, Developer Leads | 15 July – 21 July | 10 Hours |
| System Design | Requirements | Developer Leads | 22 July – 4 August | 25 Hours |
| Module Development | System Design | Developer Leads and Software Developers | 5 August – 1 September | 100 Hours |
| User Acceptance Testing | Module Development | Software Developers, Developer Leads and End Users | 2 September – 22 September | 50 Hours |
| Deployment | User Acceptance Testing | End Users and Software Developers | 1 October | 5 Hours |

## 

## Evaluation Plan

As mentioned in the implementation plan, testing is the most important phase as we can "break" our project a thousand times to make sure the end user has the best possible experience. Multiple types of testing will be completed in multiple phases throughout the development lifecycle. Each abstract task will undergo it's own unit testing before being adopted. These modules need to be extremely reliable to build confidence in each feature. Once those tests are passed, we can move into integration testing. This type of testing will be completed each time a new module is accepted. Once module 2 joins module 1 in the repository, we have to ensure they mesh well. Can all the function calls be made properly? Is there any missing dependencies in the separate files? So on and so forth. This step will be accompanied by regression testing. Not only will we make sure the pieces work on their own and then together. We need to test to make sure everything still works as intended. If one piece does the math and the other saves the answer, we use regression testing to make sure the order of operations are still accurate and the answer gets saved to the right place. Integration testing can have a calculation done and an answer saved but we need to make sure it is correct. System testing and user testing are the last two pieces to the evaluation puzzle. System testing will ensure we get the intended results once all the pieces are combined. Then we can get the product in front of potential end users for testing to make sure it provides the level of assistance advertised. If we can demonstrate 50% decrease in trade execution time with user satisfaction, we can consider this product viable.

## Costs

|  |  |
| --- | --- |
| Description | Cost |
| Software Developers x 7 | $125,000/year or $60/hr |
| Database Engineer x 2 | $170,000/year or $82/hr |
| MongoDB Cloud License | $14,990/year |
| Azure Cloud License | $20,000/year + $1000/year for support |

# Part D: Post-implementation Report

## Solution Summary

At HCOMM, we have financial analysts who are responsible for making informed, timely and risky decisions on which investments to make. The stakeholders accept that risk based on the performance of the individual. We set out to remove some of the strain on the individual and in turn making that risk even easier to digest. We built an application that give a “Go/No Go” on each stock. The problem we faced had to do with a lack in time efficiency. Each of our analysts had to sift through all of the data on each company for potential investments. That caused delays in executing trades and ultimately cost money to our clients, whether realized or potential losses. Now with the use of and XGBoost-based model, the numerical data can be synthesized immediately. This means only the nuanced data analysis only a human can make needed to be accomplished. We used python and TA-Lib to calculate technical indicators such as the different moving averages, Moving Average Convergence/Divergence, Williams %R and others. We employed Matplotlib to visualize some of the different trends. MongoDB was used as the database for storing all of the information. This choice was made because each document could contain all the data it needed in one entry instead of having to produce a complex schema to hold all the different data types. All of this combines into a Flask application that allows you to select a ticker symbol and have a prediction provided of either “Go” or “No Go.” The confidence score and two charts are provided with each prediction for review as well.

## Data Summary

The data was pulled using the yfinance API. This API pulls market data, relevant news articles and more from Yahoo Finance. An example of the data pulled from yfinance into the MongoDB instance is below.

A screenshot of a computer

AI-generated content may be incorrect.

There were several points that needed to be addressed when using the API. yfinance labels each field with the ticker symbol, e.g., “Open\_NVDA” instead of “Open.” There is also a bunch of data that would not be as useful if added. This means choosing between the correct “yf.Ticker()” and “yf.download()” functions were important. The first allows access to data that includes the news articles required for the sentiment analysis piece. The download function brings in all the historical summary data for entering the numerical values as shown below.

A screenshot of a computer screen

AI-generated content may be incorrect.

Then, the date needed to be stripped and None values needed to be transformed. Both of these were for compatibility’s sake with MongoDB. Once the data was collected, formatted and archived, it was ready for use. The database will only need to be accessed from that point on for model training and additions for more up-to-date numbers. This will happen periodically so the data costs can be lowered due to still-use.

The “Success” column was added to the database as the label column. The algorithm needed to predict on the “Success” value based on the other data in the documents. That included all the stock’s “simple numbers” like Open, High, Low, Volume. It also had access to the technical indicators based on those numerical values. This would be the RSI, CCI, EMA, SMA, etc. There was also a numerical sentiment score based on the news articles relevant to the symbol and timeline.

## 

## Machine Learning

The method used for the machine learning model was XGBoost. eXtreme Gradient Boost-ing algorithm based on decision trees provided the capability to provide accuracy and speed; two things this project is striving to improve. This ensemble algorithm builds decision trees in repetition with the goal of fixing the errors from the previous. An example of the initial output of the model’s testing is here.A screenshot of a computer program

AI-generated content may be incorrect.

The model was justified in it’s use based on the ease of implementation and immediate positive results. Generally, anything above 70% is acceptable. Even though there are far too many considerations to put a value on what is and is not viable, XGBoost proved to do well on the first iteration of the build. With future tuning and data optimization, this can be increased significantly.

## Validation

XGBoost is a supervised learning algorithm. It requires labels to be placed on the data it is using to train. This is critical to classification predictions, where an output is binary(Go / No Go). The “Success” column was added to the database as the label column. The algorithm needed to predict on the “Success” value based on the other data in the documents. That included all the stock’s “simple numbers” like Open, High, Low, Volume. It also had access to the technical indicators based on those numerical values. This would be the RSI, CCI, EMA, SMA, etc. There was also a numerical sentiment score based on the news articles relevant to the symbol and timeline. This was an appropriate setup due to the requirements of the program. The success columns predicted value translated to the predicted Go / No Go response on the ticker requested. As shown in another section, the initial testing of the model proved to be impressive for a first run.

A screenshot of a computer program

AI-generated content may be incorrect.

Visualizations

The visualizations on the application include different charts and labeled/colored response to help quickly identify some trends and initial impressions on the trades validity. Candlestick charts provide a look into the activity and trend of the price of a stock. Using the Open, High, Low and Close prices for a given time period, it helps analysts with spotting trends and sentiment around that particular company. The Price/Volume chart shows the closing prices and the trading volume for a given day. This is a simple way to get an idea of the magnitude and delta in a company’s value. The prediction and response score give insight into the data products’ analysis on the company in that given moment with the most up-to-date values. The confidence score is a literal representation of how sure the model was about this decision. This information can be used at the analysts discretion. Either choosing to investigate confidence scores closer to 50% or scores with high confidence. Note that this score is for the positive prediction, Go. A .999 confidence score means it is very confident it will be successful while a .001 score means the model was very confident it would be unsuccessful.

The candlestick chart :

A chart with red and green lines

AI-generated content may be incorrect.

The Price/Volume Trend Chart :

A graph with red lines and numbers

AI-generated content may be incorrect.

The prediction and confidence score :

A close up of a sign

AI-generated content may be incorrect.

## User Guide

There are a few specific things to be aware of when installing this program, specifically the use of MongoDB and TA-Lib. For MongoDB, Docker will be the easiest way to start the mongod service. Required for running pymongo library. After installing Docker, simply pull the latest mongodb/mongodb-community-server image and run the container. Technically, you do not need to install a MongoDB gui application. However, this will be helpful to ensure database connections are successfully created instead of running the necessary troubleshooting through the command line.

For TA-Lib, this is much more involved. First, install the MS Visual Studio build tools. <https://visualstudio.microsoft.com/downloads/?q=build+tools> You will need to install the Desktop Development with C++ and .Net Desktop Build Tools modules. After that, TA-Lib will need to be installed via an .exe file. <https://ta-lib.org/install/> has the list of available executables. Preferably this is being done on Windows in which case use the “recommended” executable installer. Once that is done, go into command line and run “pip install TA-Lib”. This should complete the install at which point the import statements should work.

The steps for successfully running the program are below.

Install Python – Choose a version compatible with 3.12

Install TA-Lib (See notes above for installation guide.)

Install Docker -> Pull “mongodb/mongodb-community-server” image -> run container

Install pip

Run ‘pip install -r requirements.txt’ from the project’s root directory

The passphrase is located in the backend.py file to login to the app

After successfully accessing the application, select from the list of tickers available to get responses on that company.