Software Design Document
DAI Trader

Adam Burich
Bright Hsu
Kayla Savage
Janice Tran
William Puppo
Sandalu Widyalankara

February 1st 2022
Version 1.0.0

SDD Revisions

Date	Description	Revision	Editor
01/19/2022	Created Template	0	Kayla Savage
01/20/2022	Created Table of Contents	1	Kayla Savage
01/25/2022	Created SDD Revision	2	Kayla Savage
01/25/2022	Added Introduction	3	Kayla Savage
01/29/2022	Design Overview	4	Kayla Savage
01/29/2022	User Interface	5	Kayla Savage
01/30/2022	Docker	6	William Puppo
01/31/2022	Machine Learning Algorithm	7	Bright Hsu
02/01/22	Revised Table of Contents	8	Janice Tran
02/01/22	References	9	Janice Tran
02/01/22	Revised Definitions	10	Janice Tran
02/01/22	Data Preprocessing and Model	11	Kayla Savage
02/01/22	System Architecture Diagram	12	Sandalu Widyalankara

Table of Contents

1. Introduction

DAITrader, an investment scenario simulation of the stock market, utilizes machine learning and artificial intelligence in order to provide a stock experience to the user. Additionally, the application produces possible predictions of company projections.

1.1 Purpose

The purpose of this document is to specify the software requirements for DAITrader. It is to clarify and explain the requirements needed in order for the application to function for both programmers and users.

DAITrader provides insight about stocks, particularly, the companies that own them, their trends, and examinations of their projections. This application will feature a GUI, in which the user will be able to interact with and see various simulations.

1.2 Intended Audience

The intended audience for this document are beginners, students, investors, developers, and testers. The application is intended to be beginner friendly.

- User: Intended users can include beginners through intermediate investors and learners.This document is needed as they need to review diagrams, dynamics, and specifications in order to ensure the application is what the developer promised. These types of users will have the same set of functions with different intentions.
- Developer: The developer is one who is tasked with reading, testing, and modifying the application. In order to perform such actions, it is a necessity that the requirements are understood through this document.
- Tester: The tester is one who is tasked with reading, understanding, and testing the application. This document provides application specifics and functionality for that purpose.

1.3 Project Scope

The application DAITrader offers users insight into the simulation of stocks. Furthermore, it allows one to keep track of stock projections provided by the system.

The main purpose is to feature scenarios of stocks to the user in order to provide an experience of various stock projections. This is done with a GUI, database, evaluation system and backend API.

1.4 Definitions, Acronyms and Abbreviations

SRS: Software Requirements.

Schema: A schema is the organization or structure of a database. The activity of data modeling leads to a schema. [1]

Specification Connects: Links this requirement with another

Includes: Has the appropriate constraint in it

Extends: Shows or cancels a constraint effect if the conditions are met.

ToS: Terms of Service agreement defining terms that customer must agree to before use of application.

RSA: Encryption algorithm based on ‘the factoring problem’ that utilizes a public and private key to maintain data obfuscation. [2]

User Stories: Presents a user description based on a potential use of a certain software requirement within the application.

Domain Class Diagram: A diagram featuring the domain requirements for the trajectory of the application procedures.

GUI: Graphical user interface.

API: Application programming interface.

OS: Operating System.

PAAS: Platform as a service.

2. Design Overview

DAITrader is an investment scenario simulation of the stock market. Using machine learning/AI, DAITrader generates predictions of various companies' value. DAITrader will also aim to offer users a finance related educational perspective; it is our hope that users will be able to use DAITrader to learn more about the stock market.

2.1 Technologies Used

1. Pandas
2. Tensorflow/Keras
3. Numpy
4. MatplotLib
5. Docker
6. Github
7. Python
8. AWS
9. AlphaVantage for Historical Stock Trade Values and Daily Stock Value

2.2 System Architecture and Operation

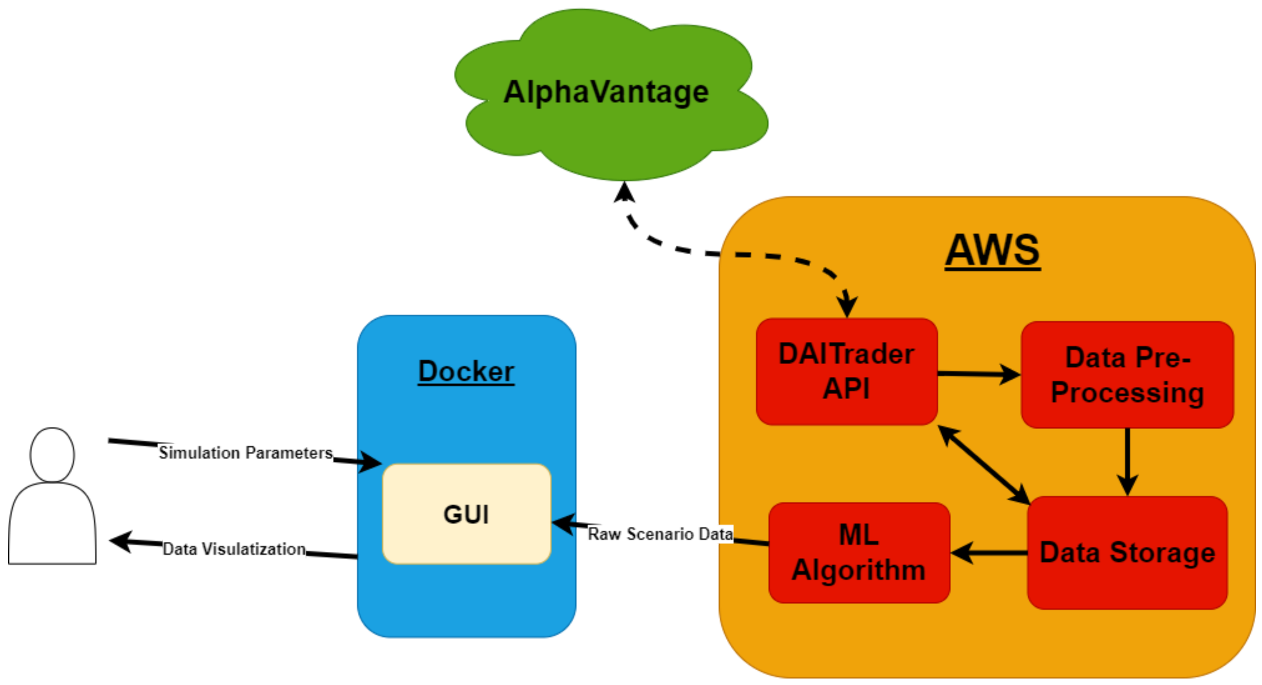


Figure 1 depicts the high-level system architecture. The system will be constructed from a combination of distinct components:

2.3 Design and Implementation Constraints

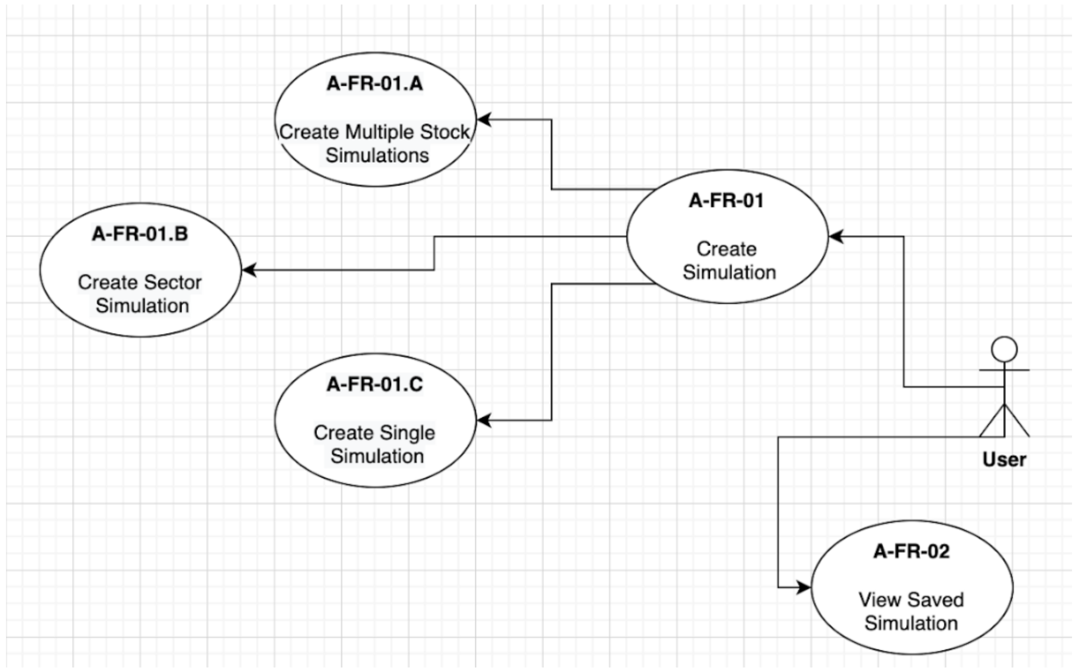
This application is created with the programming language python3 and uses various python libraries for the modules needed for the GUI application, the API, evaluation system and the database. Such libraries include Pandas, Numpy, Tensorflow/Keras, Matplotlib, and Docker.

2.4 User Documentation

Official Website: http://www.cci.drexel.edu/SeniorDesign/2021_2022/DAITRADER/index.html

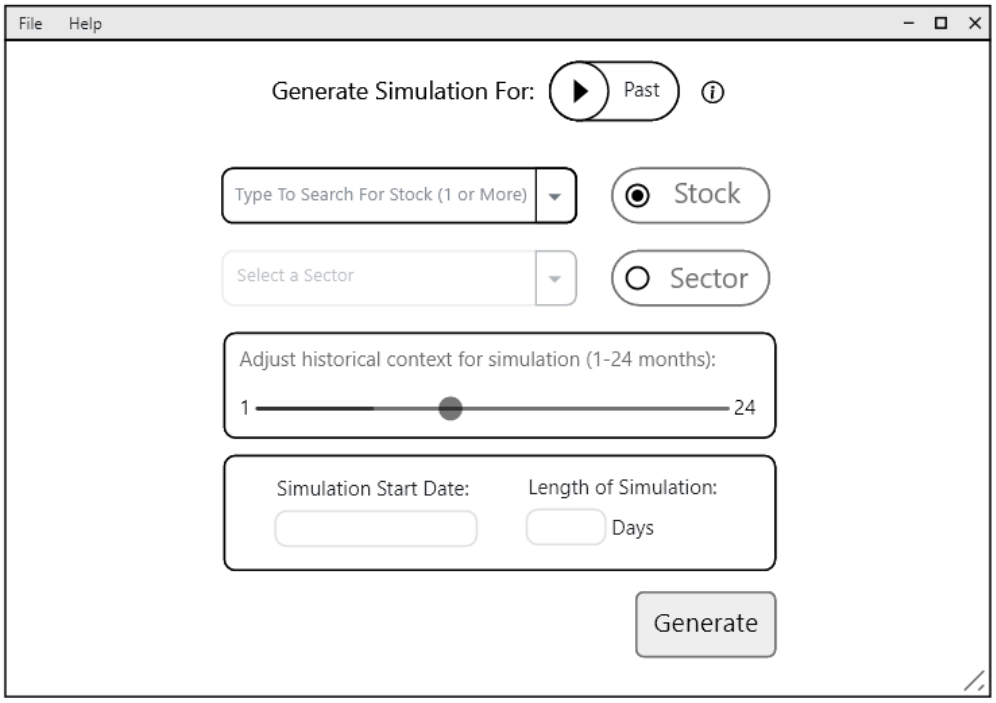
3. User Interface

3.1 Use Case Diagram

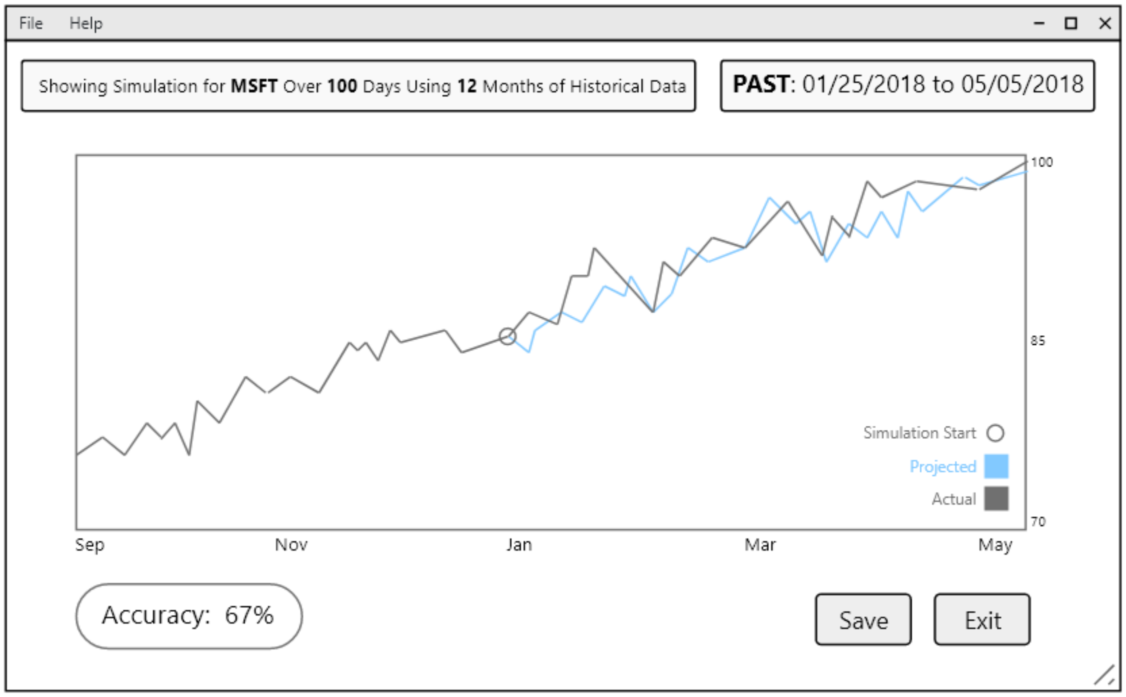


3.2 GUI - Wireframes

3.2A1:



3.2A2:

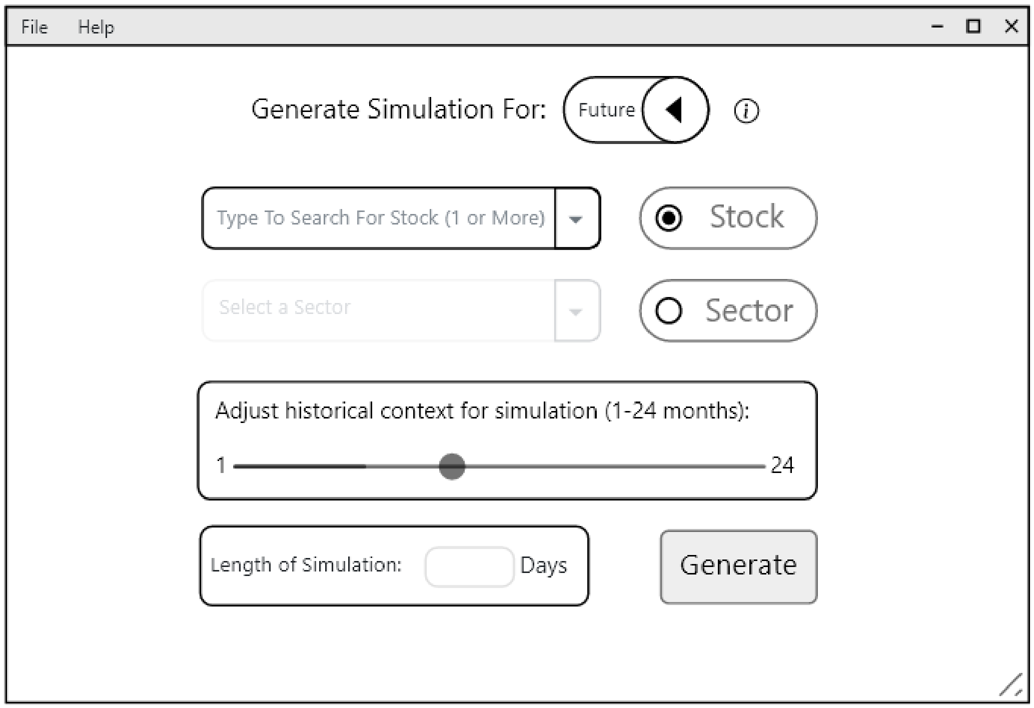


Showing Simulation for **MSFT** Over **100** Days Using **12** Months of Historical Data

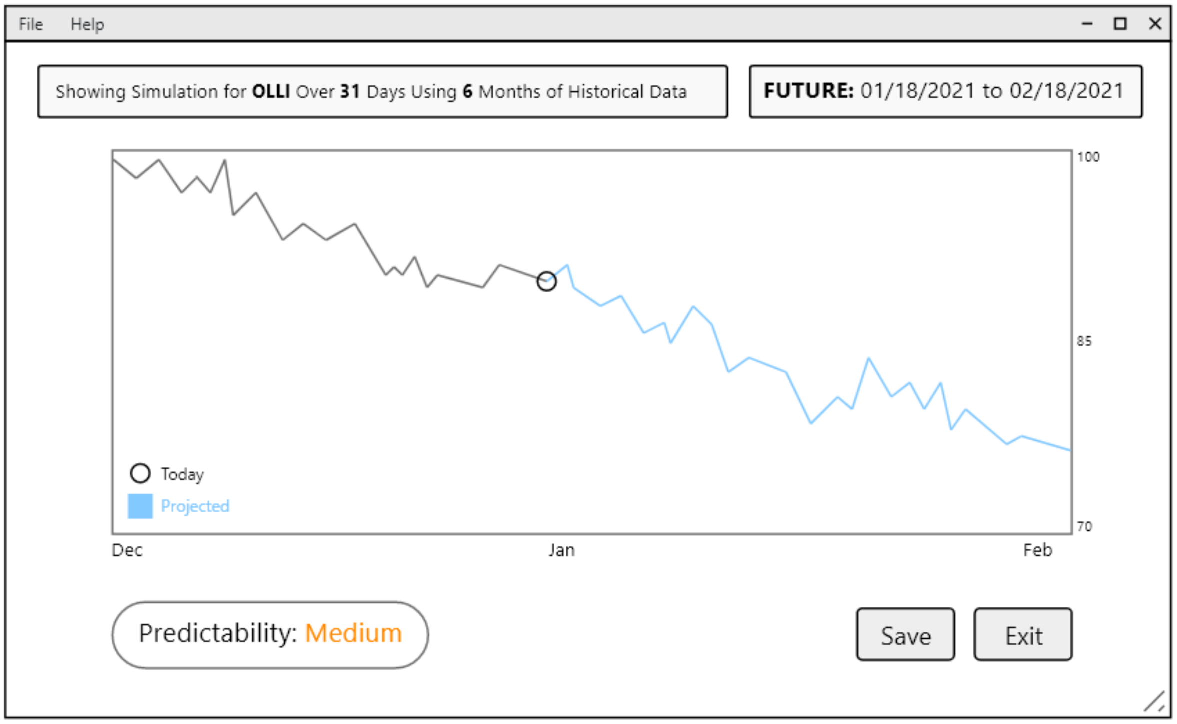
PAST: 01/25/2018 to 05/05/2018

Accuracy: 67%

3.2B1:



3.2B2:


File
Help
Showing Simulation for **OLLI** Over **31** Days Using **6** Months of Historical Data
FUTURE: 01/18/2021 to 02/18/2021
100
85
70
Dec
Jan
Feb
○ Today
🟦 Projected
Predictability: Medium
Save
Exit

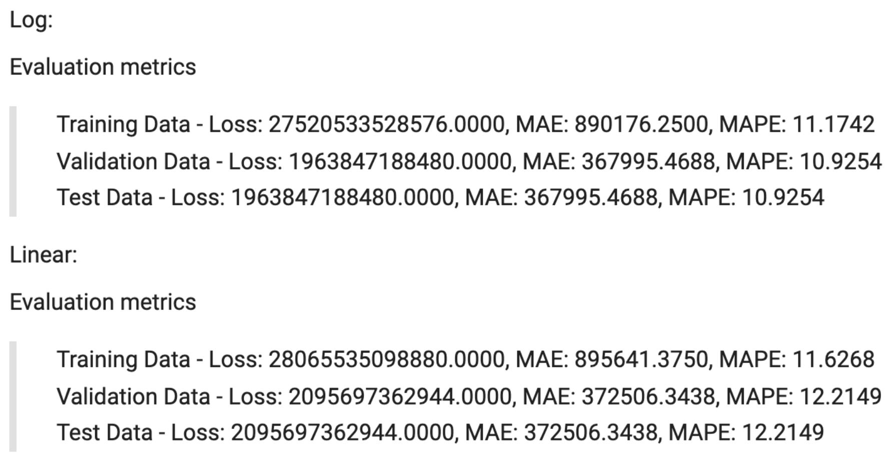
3.2A1: Screen for generating a simulation in the past for selected stocks
3.2A2: Simulation generated as a result of user inputs from 3.2A1
3.2B1: Screen for generating a simulation for the future on selected stocks
3.2B2: Simulation generated as a result of inputs from 3.2B1

4. Machine Learning Algorithm

With a proof of concept finished and showing promising results, this section will discuss the future of the machine learning algorithm.

4.1 Result and Revision

The modified linear regression model produced a decent Mean Absolute Percentage Error or MAPE score of 10, however, there were noticeable outliers and output delay within the model itself. Additionally, the grading metrics of the system techniques were not clear. Mean absolute percentage error (MAPE) creates various accurate predictions that do not rely on time and since stock prediction relies heavily on time based predictions this proposed an issue and having one prediction be extremely off will invalidate any other prediction after that until the model generates a new prediction.



Log:

Evaluation metrics

```
Training Data - Loss: 27520533528576.0000, MAE: 890176.2500, MAPE: 11.1742
Validation Data - Loss: 1963847188480.0000, MAE: 367995.4688, MAPE: 10.9254
Test Data - Loss: 1963847188480.0000, MAE: 367995.4688, MAPE: 10.9254
```

Linear:

Evaluation metrics

```
Training Data - Loss: 28065535098880.0000, MAE: 895641.3750, MAPE: 11.6268
Validation Data - Loss: 2095697362944.0000, MAE: 372506.3438, MAPE: 12.2149
Test Data - Loss: 2095697362944.0000, MAE: 372506.3438, MAPE: 12.2149
```

Figure 4.1 shows the evaluation metrics of the model

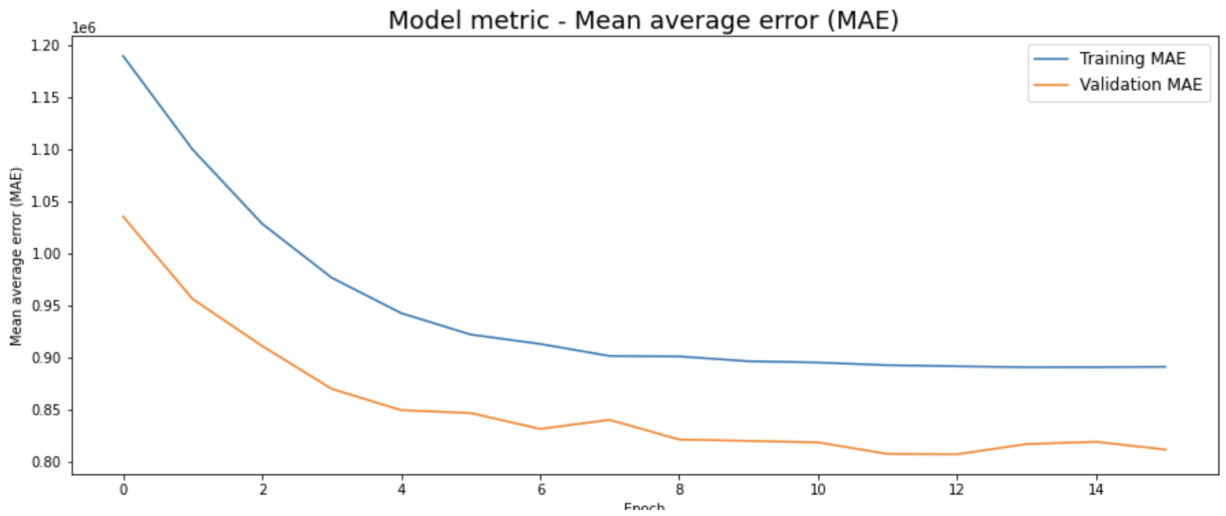


Figure 4.2 depicts the mean average error of the model

The revision of the model plans to tackle this issue, more testing will be needed to try out different methods. This testing will include running an updated dataset with modified preprocessing and cleaning methodologies and custom model evaluation statistics.

4.2 Implementation

Model will be run after preprocessing script is finished, implementation of the model will be done by having the prebuilt model be loaded and tasked to take inputs from the server side of the infrastructure. This will take user inputs from the front end.

5. Data Preprocessing and Database Model

5.1 Data Preprocessing

Combineddata_csv

As described in our resources section, after data is pulled from AlphaVantage API, the data is run through a series of preprocessing steps. This includes, the removal of null and duplicated data values, sorting and the combining of all 500 companies. Combineddata_csv is the collection of all these values into a comma-separated value sheet. These file types are easy to both analyze and visualize.

Attributes

Name	Type	Description
timestamp	Vector <datetime>	Contains a vector of datetime values that depicts the time at which stock price closed
open	Vector <float>	Contains a vector of float values that indicates the price where trading begins
high	Vector <float>	Contains a vector of float values that shows the highest price at which a stock traded during the course of the trading day
low	Vector <float>	Contains a vector of float values that shows the lowest price at which a stock traded during the course of the trading day
close	Vector <float>	Contains a vector of float values that indicates the price where trading ends
volume	Vector <integer>	Contains a vector of integers values that represents the number of shares traded of a particular stock, index, or other investment over a specific period of time
stock_name	Vector<String>	Contains a vector of strings indicating the ticker names of collected stocks

5.2 Database Model

Using PostgreSQL database version 13.3, the database serves a significant function to our project. By controlling, loading and saving the visualization and user information. The database model holds the in-memory data while its contents after running the system is stored on AWS.

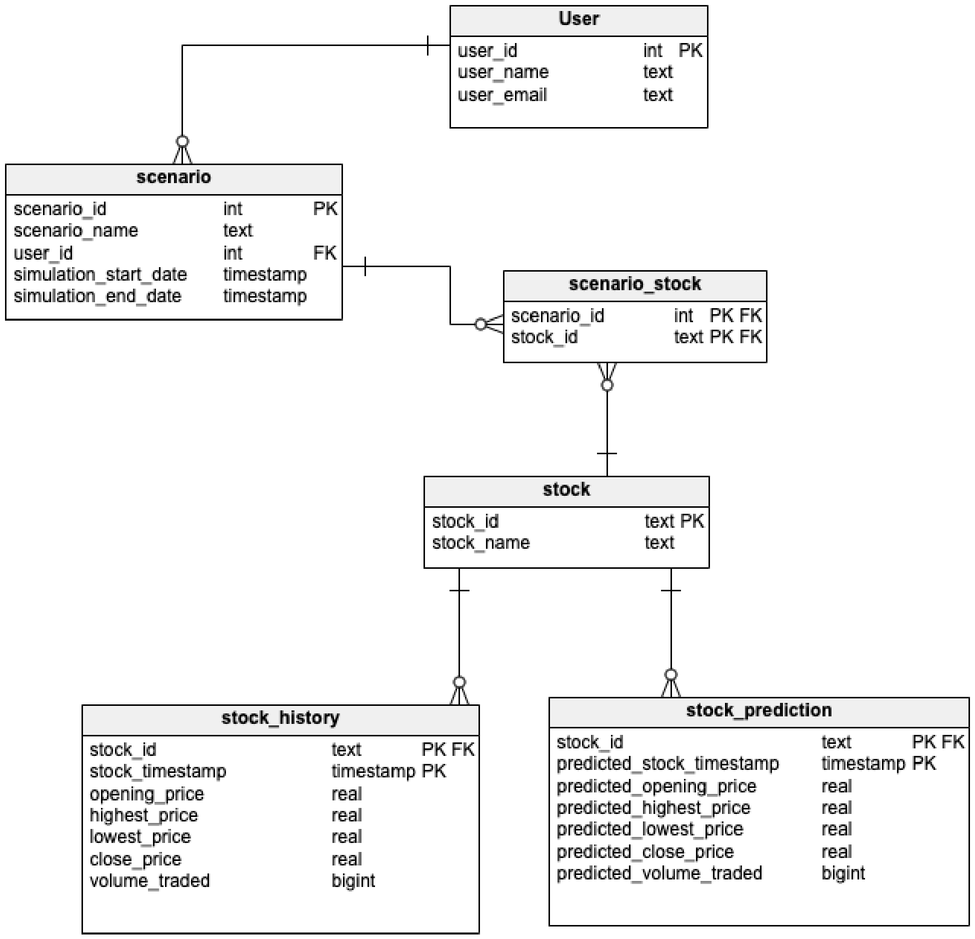


Figure 4 shows the database schema with all appropriate entity relationships.

6. Amazon Web Services (AWS)

We will use an EC2 AWS server to implement our system. Thus far we've set up a server with NGINX for our API, Docker, Docker-Compose, and acquired a domain name to point API requests to. This EC2 server will run the script that utilizes the AlphaVantageAPI to download and host stock data. The data will then be processed and handed to the model to generate a prediction. That prediction will be sent back to the client.

7. Docker

7.1 Concept

This application will be containerized using Docker PAAS in order to distribute the app to any device running Docker PAAS. This removes the need to develop for specific hardware or software and allows flexibility of distribution.

7.2 Implementation

Only the endpoint application will be containerized with no local data persistence. This will be done by writing a dockerfile in order to compile the docker image of the application.

The dockerfile will be distributed through the project's web portal after a user has been registered to the user's device at which time they will access the application by running the dockerfile.

7.3 Timeline

02/06/22 - 02/12/22 : Create Dockerfile template.

02/13/22 - 02/19/22 : Create Dockerfile for application at current state.

02/20/22 - 02/26/22 : Test Dockerfile and image on various endpoints running Docker PAAS.

02/27/22 - 03/12/22 : Refine and deploy.

8. References

[1] Directorate, OECD Statistics. *OECD Glossary of Statistical Terms - Schema Definition*,https://stats.oecd.org/glossary/detail.asp?ID=6262#:~:text=In%20computer%20programming%2C%20a%20schema,a%20formal%20text%2Doriented%20description.

[2] “What Is the RSA Algorithm?” Educative, https://www.educative.io/edpresso/what-is-the-rsa-algorithm.