alamSYS: Development of Stock Market Price Forecasting System Using Dynamic Mode Decomposition, Long Short-Term Memory with Arnaud Legoux Moving Average Convergence-Divergence Integration

A Special Problem
Presented to
the Faculty of the Division of Physical Sciences and Mathematics
College of Arts and Sciences
University of the Philippines Visayas
Miag-ao, Iloilo

In Partial Fulfillment
of the Requirements for the Degree of
Bachelor of Science in Computer Science by

OLARTE, John Markton M.

Nilo C. Araneta
Adviser

June 2023

Contents

List of Figures

Chapter 1

Materials and Methods

This chapter discusses the materials and methods used for the design and development of the system: alamSYS. Specifically, the following are discussed in this chapter:

(a) Development Tools and Software Requirements

(b) System Diagrams

(c) Hardware Requirements

(d) Methodology

(e) Gantt Chart

1.1 Development Tools and Software Requirements

The development of the alamSYS utilized the following development tools and software requirements:

1.1.1 Development Tools

(a) Visual Studio (VS) Code – This is a highly functional code editor that served as the project's primary development interface.

(b) MongoDB Compass – This is a graphical user interface for developing and managing various MongoDB databases.

(c) GitHub – This serves as the project's code repository and version control system (via git).

1.1.2 Software Requirements

(a) Python (version 3.9.x) – This served as the primary programming language for the development of the various components of alamSYS, with the following libraries specifically used:

- For the development of the API and Database ODM
  - FastAPI (version 0.85.0) – A library that is primarily used to create modern, fast, and high-performance web framework APIs. (Tiangolo, n.d.). Specifically, utilized in the development of the project because of its (1) ease of utilization; (2) fast implementation; (3) high-performance; (4) built-in robust API documentation; and (5) high scalability.
  - mongoengine (version 0.24.2) – A library designed as an Object-Document Mapper that allows Python to connect to and work with MongoDB. (MongoEngine, n.d.) This was used in the alamSYS to connect the API endpoints to the MongoDB database, and vice versa.
  - json (pre-installed) – This is a Python library for converting a Python dictionary to a JSON object and vice versa. This was used in the development of alamSYS for data parsing and conversion from the API to the MongoDB database via an ODM.
  - datetime (pre-installed) – This python library was used for creating a datatime object, which as the name suggests is an object that

contains the date and time information. This was used in the system to keep track of all the processes that occur in the system using date and time logs.

- os (pre-installed) – A Python library that allows the user to perform operating system operations such as creating directories and files, accessing operating system information, and so on. This was used to access the operating system's environment variables as well as to assist with other OS-based functions.

- For the preprocessor (main)
  - schedule (version 1.1.0) – This library allows the user to schedule a function to be executed at a specific date and time. This was used in the system to schedule the processes that occurs in the alamSYS.

- For the preprocessor (data collector)
  - requests (version 2.28.1) – This library allows the user to create web requests to an external or internal servers. This was used to connect and collect the current EOD market data from the third-party market historical data provider: EODHD.
    EODHD – A third-party market fundamental and historical data APIs provider (EODHD, n.d.).

- For the preprocessor (data processor):
  Note that some of these libraries are also used in the development of the DMD-LSTM model.
  - numpy (version 1.23.5) - Utilized for handling large data arrays. This is because, compared to Python's List, numpy is better in terms of performance and memory utilization (Geeks for Geeks, 2022).
  - tensorflow (version 2.11.0) - Utilized for the development of the DMD-LSTM model.
  - matplotlib (version 3.7.0) - Utilized for creating graphical diagrams and plots for the results of the data gathering during the developmental stages of the system, specifically during the development of the DMD-LSTM model.

- pyDMD (version 0.4.0post2301) - This library was used to extract the dynamic modes from the stock market data as an additional training input for the DMD-LSTM model.
- pandas (version 1.5.3) - This library was used to handle the dataframes during the testing period of the alamSYS.

(b) MongoDB – A non-relational (document-based) database, used to hold the necessary data for the alamSYS. Such as stocks info, which stocks to buy or to sell, and the risk profile of each stocks.

(c) Jupyter Notebook – This was used during the training and testing of the DMD-LSTM model.

(e) Docker – A useful tool to creating containers. Containers contains the source code and all its dependencies in one standard unit of software, which can be run in different machines regardless of its difference from the development machine used (Docker, n.d.). As such this was used to create containers for each of the component of alamSYS, to enable it to run in different deployment machines.

(f) Docker-compose – In order to run multiple containers at once, docker-compose was used. This is further discussed in the Container Diagram section of this chapter.

(g) Dart and Flutter - This was used for the development of the mobile-based test application (alamAPP) to showcase how the alamSYS can be used in an actual application. In addition, the following libraries were used:

- http (version 0.13.5) - This library was used to create HTTP requests to the API endpoints of the alamSYS.
- path_provider (version 2.0.13) - This library was used to allow the alamAPP to access the storage of the device, which then allows the application to save the details collected from alamAPI through the http request library.
- syncfusion_flutter_charts (version 20.4.52) - This library was used to show or visualize the predicted graph based on the price predictions given by the alamSYS.
- lottie (version 2.2.0) - This was used to show the loading animation when the alamAPP is waiting for the response from the alamSYS,

as well as animations when the alamAPP failed to connect to the alamSYS through the alamAPI. Overall, this library makes the application more dynamic, interactive, and more user-friendly.

(h) Git - Used as the version control system for the development of the alamSYS.

(i) GitHub - Used as the repository for the alamSYS.

1.2 System Diagrams

In this chapter, the appropriate system diagrams will be shown and discussed. This shall help in the understanding of the system's features, data flow, and processes. Whereas all the diagrams can be viewed in full resolution, using the GitHub repository, provided in the author's note at the title page.

1.2.1 Top-Level Overview Diagram of the alamSYS and Its Interactions to External Systems

Figure 1.1 shows the top-level overview of the alamSYS and its interactions to any third-party or external applications.

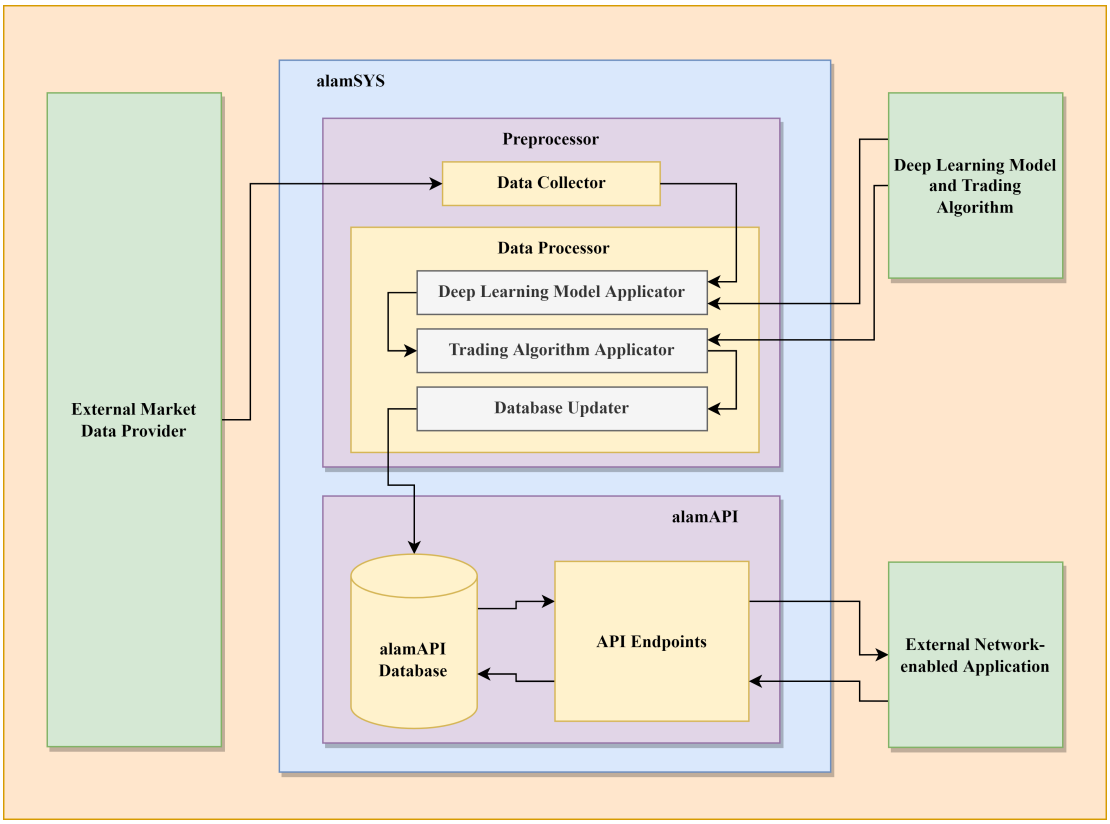

Figure 1.1: Top-Level Overview of the alamSYS and Interactions with External Applications/Systems

As shown from the figure above, the alamSYS is connected to three external entities: (1) External Market Data Provider, which provides the system with the needed historical market data; (2) Machine Learning Model or Trading Algorithm, in the case of this special problem, a machine learning model will be developed and will be utilized by the system, however as previously discussed the system is created to accept any other machine learning model or proprietary trading algorithms that other developers may or want to develop in the future; and (3) External Application, which can be a web-based or mobile-based application, that will utilize and showcase the functionalities provided by the alamSYS, through the API endpoints.

On the middle of the diagram the alamSYS is observed to have three main components, namely, (1) Pre-processor, which is further divided into sub-components:

(a) Data Collector, which collects the data from the external market data provider; (b) Pre-Database Processor, which processes the historical market data collected by applying the developed machine learning model and sending it to the database updater module; (2) Database, which is based on MongoDB, which is a document-based and non-relational database; finally, the database is connected to the (3) API endpoints which processes the request and responses of the system to any external application connected to the API via a network.

1.2.2 Process Flow Diagram

The diagram shown in Figure 1.2 the different processes that the system will undergo once it has been deployed in the server.

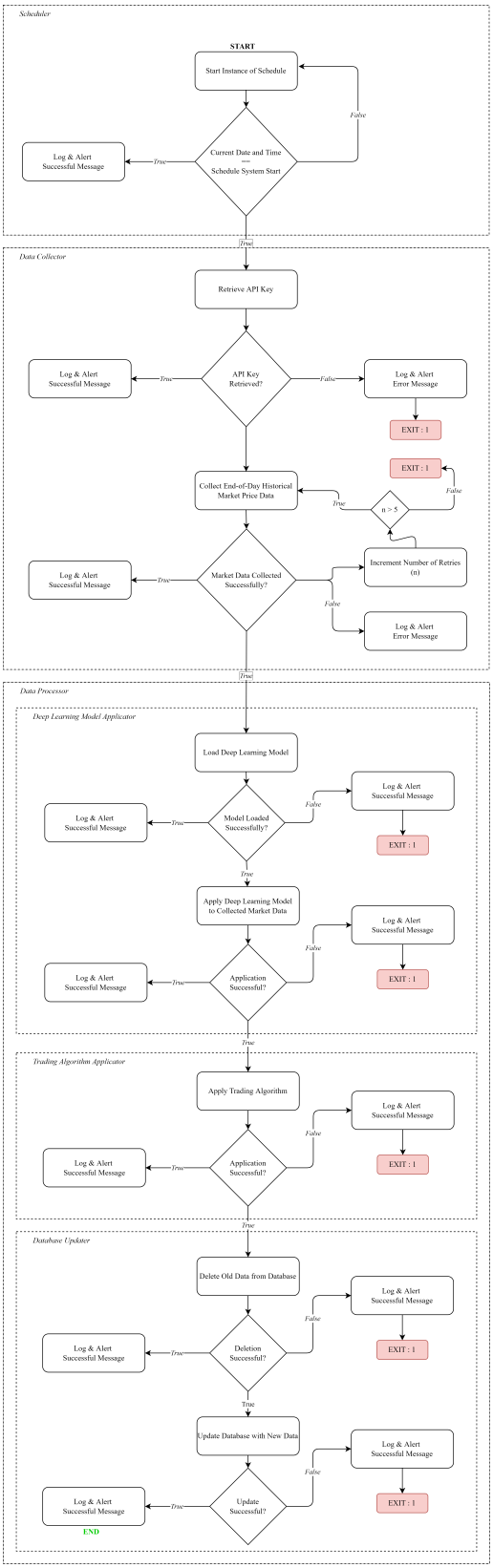

Figure 1.2: Full Overview of the Process Flow Diagram for the alamSYS

To better view and understand the flow of the processes, we can divide the discussions per components in the diagram.

Scheduler

Using Python's Schedule Library, an instance of a scheduled task is initialized upon the startup of the alamSYS. The scheduled task shall execute every Mondays to Fridays at exactly 6 P.M. Where the whole scheduling process is illustrated in Figure 1.3.

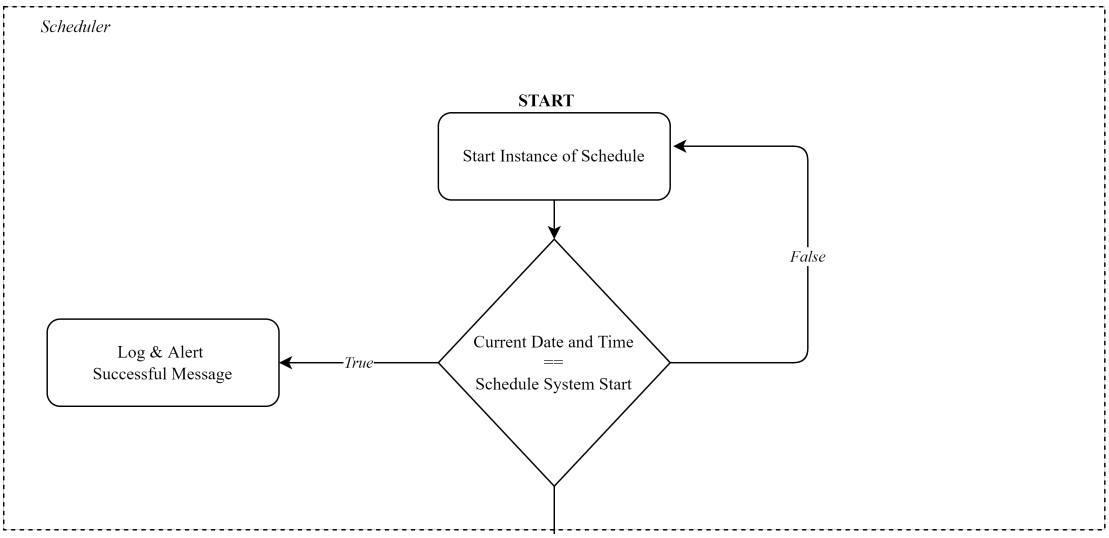

Figure 1.3: Overview of the Process Flow Diagram for the Scheduler

Data Collector

The collection of end-of-day (eod) market data is the first process in the scheduled task.
To collect eod market data, the system first looks for the EODHD API key, which is stored in system variables or provided by the user in the tools directory. If the system is unable to locate an API key, it logs the error and notifies the user before exiting the program.

Once the API key is obtained, the system connects to the EOD market data provider and attempts to collect all market data five times. If it fails to collect data for the fifth time due to errors (i.e. incomplete payments, unstable network, and no established internet connection), the system logs the error, sends an alert to the user, and exits the program.
All successful processes are also logged and sent to the command line interface to notify the user. Figure 1.4 depicts this, as well as the entire data collection process.
The flow of processes discussed above can be observed in Figure 1.4.

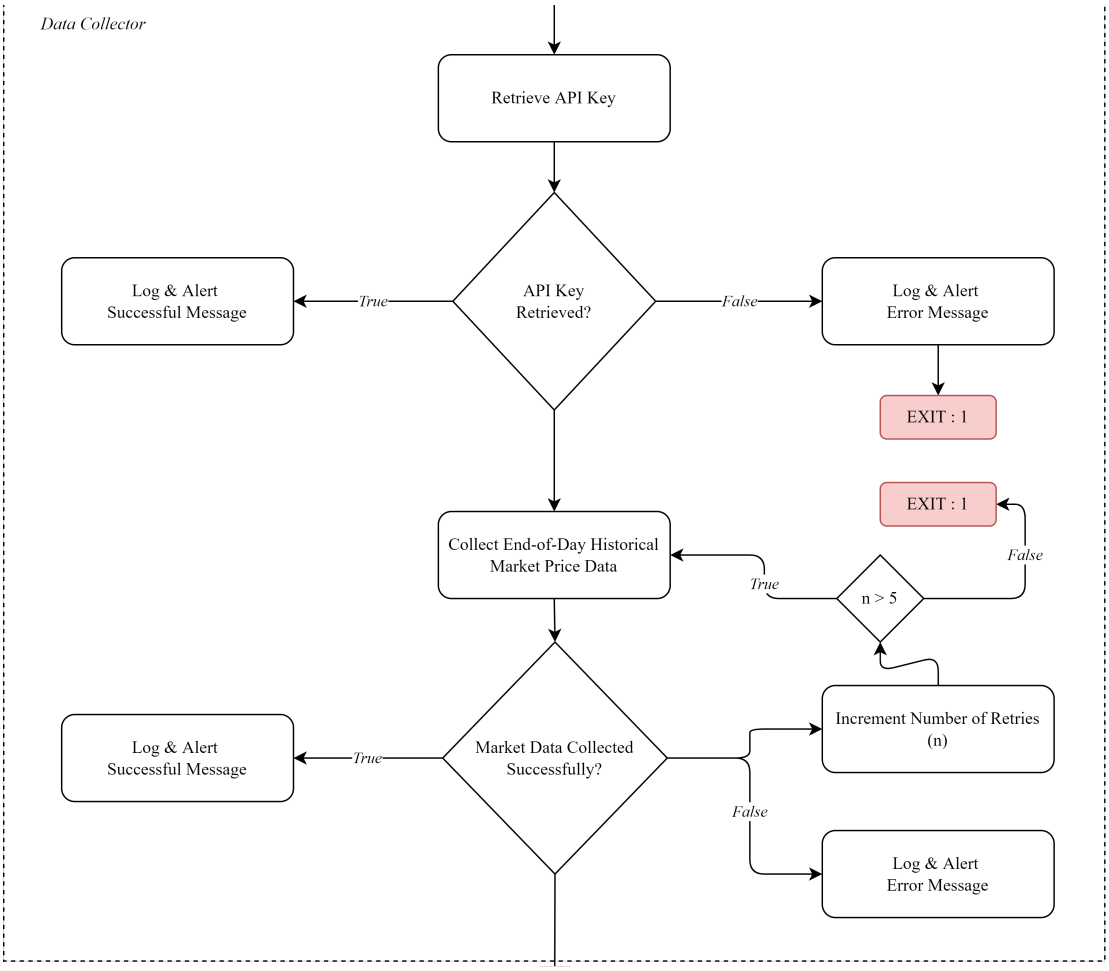

Figure 1.4: Overview of the Process Flow Diagram for the Data Collector

Data Processor

Data Processor is a process that is divided into three subprocesses, as shown in Figure 1.5.

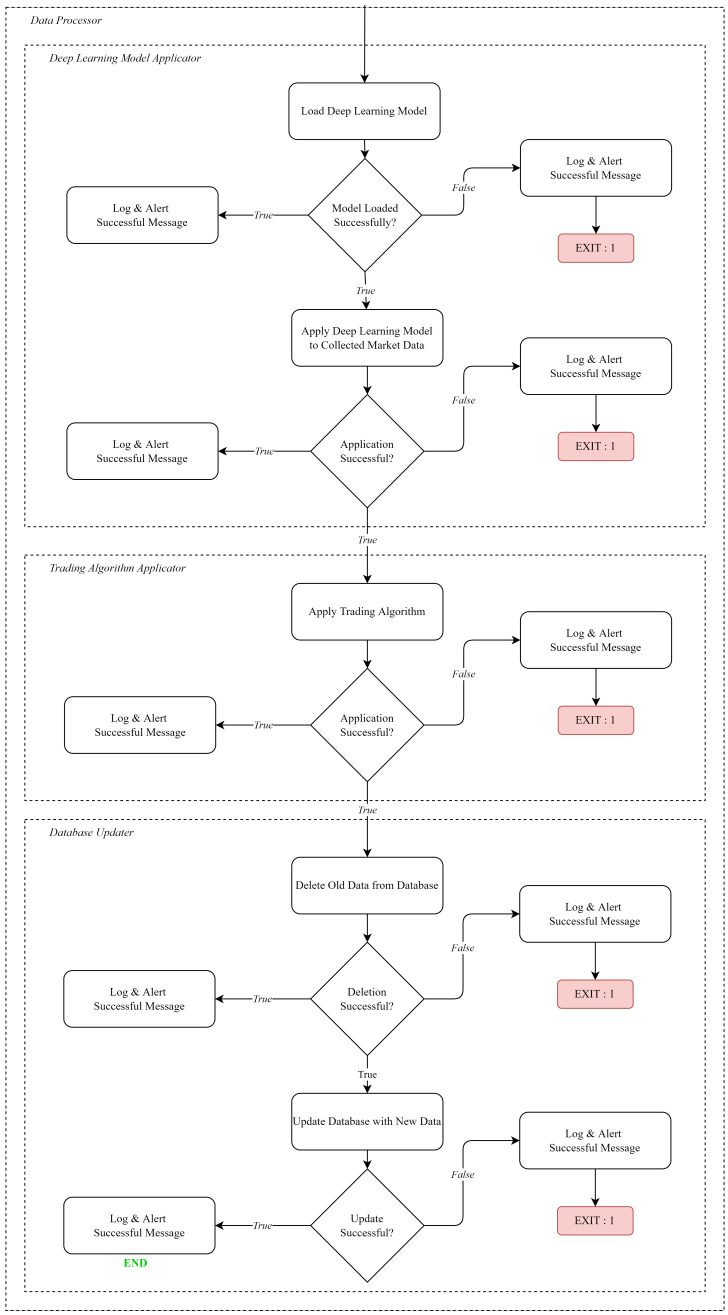

Figure 1.5: Overview of the Process Flow Diagram for Data Processor

Where the three subprocesses are as follows:

(a) Deep Learning Model Applicator - This subprocess applies the deep learning model to the collected eod market data for each stock. Specifically, alamSYS applies the DMD-LSTM model developed as part of this special problem. This is done as illustrated in Figure 1.6.

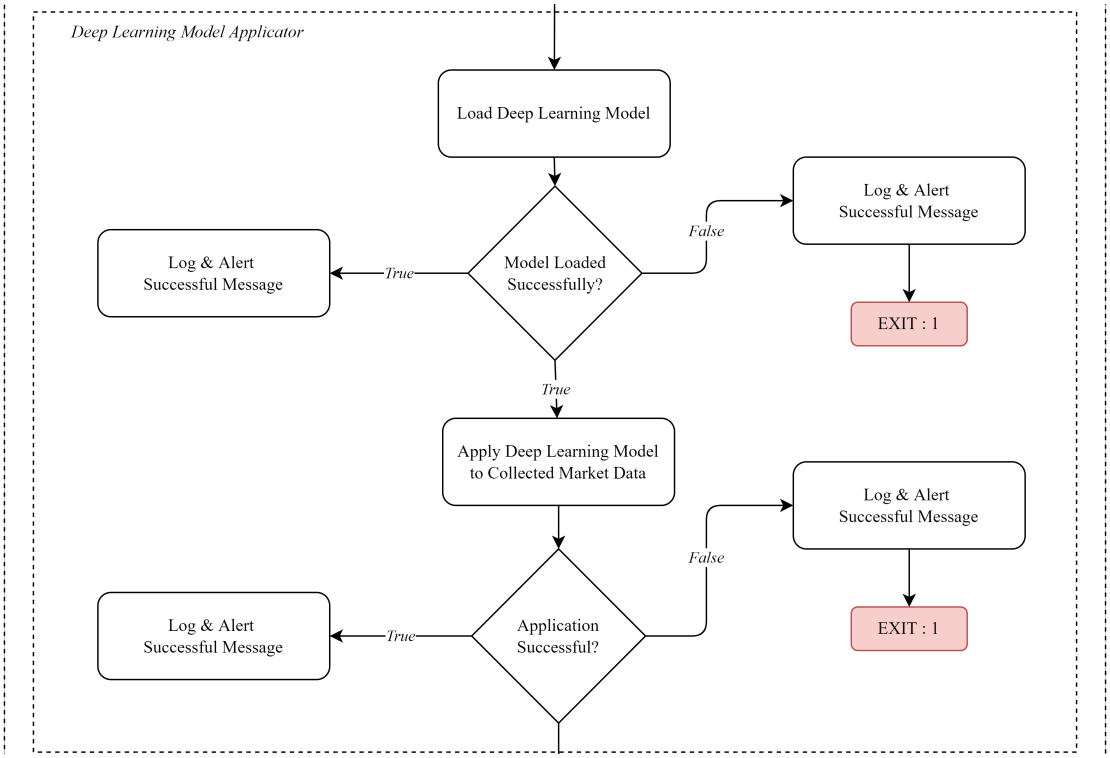

Figure 1.6: Overview of the Process Flow Diagram for the Deep Learning Model Applicator

(b) Trading Algorithm Applicator - This subprocess applies the trading algorithm to the output data from the deep learning model applicator. Specifically, alamSYS applies ALMACD algorithm developed as part of this special problem. This is done as illustrated in Figure 1.7.

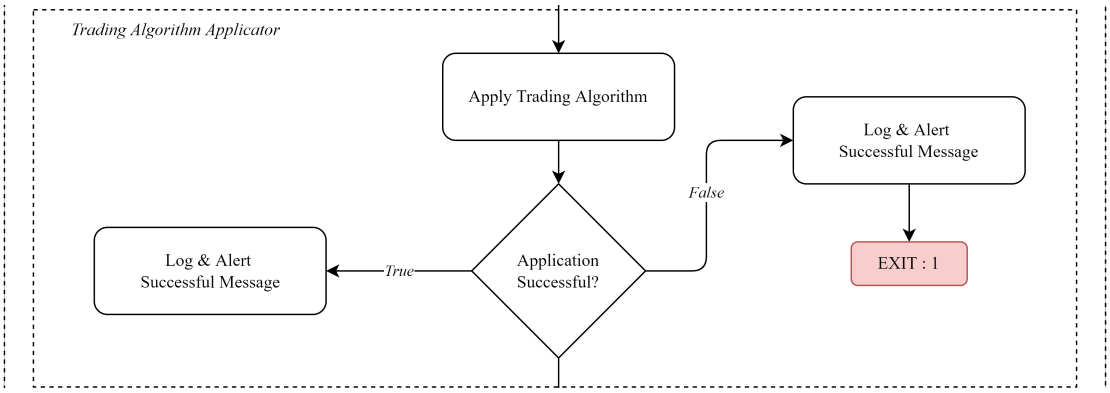

Figure 1.7: Overview of the Process Flow Diagram for the Trading Algorithm Applicator

(c) Database Updater - This subprocess updates the database with the output data from the trading algorithm applicator. This is done as illustrated in Figure 1.8.

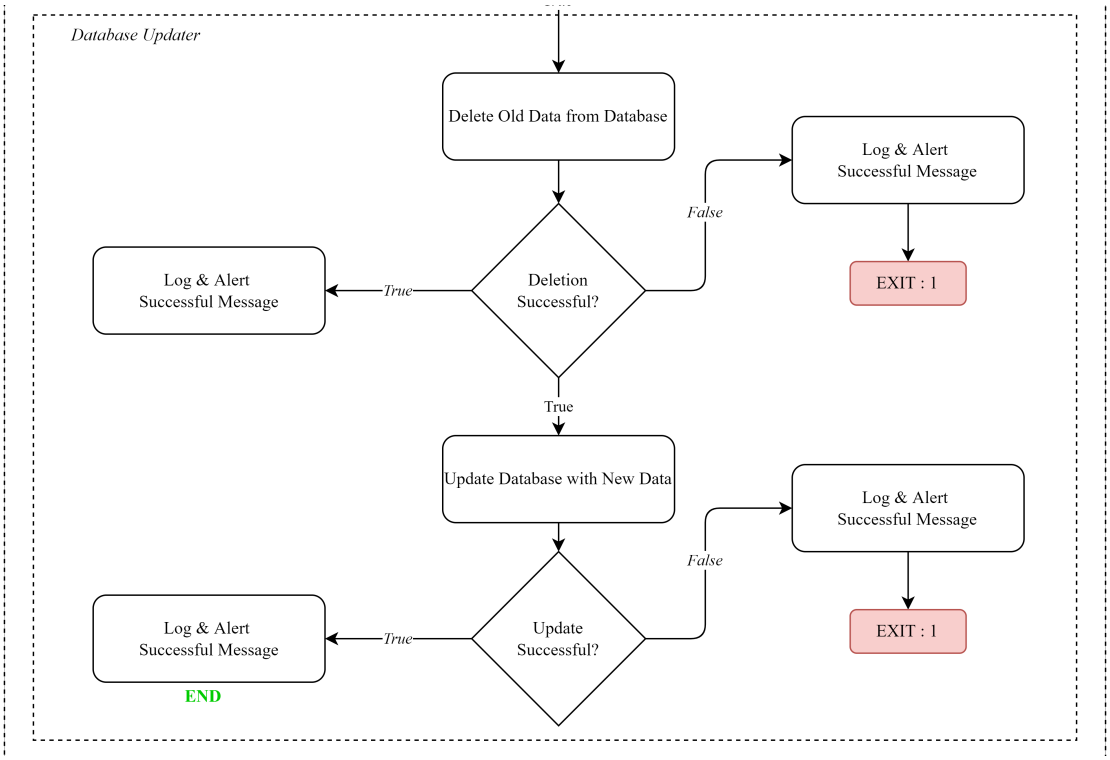

Figure 1.8: Overview of the Process Flow Diagram for the Database Updater

1.2.3 Data-Flow Diagram (DFD)

A data-flow diagram (DFD) helps to understand how processes work and how data flows from one process to the next. This is especially important because it provides an overview of the data's security by demonstrating how it can be accessed. In the case of alamSYS, the only publicly accessible data is the listed stock to buy and sell, as well as other functions as provided in its database and as permitted by the API endpoints.

Furthermore, the DFD paradigm used in the diagrams in this section adheres to the Gane-Sarson DFD symbols, which employ four basic symbols: (1) Entity / External Entity; (2) Data Flow; (3) Process; and (4) Data Store (VisualParadigm, n.d.)

Context Diagram

The overview of the entire process is depicted in a context diagram of the system, labeled process 0, as shown in Figure 1.9.

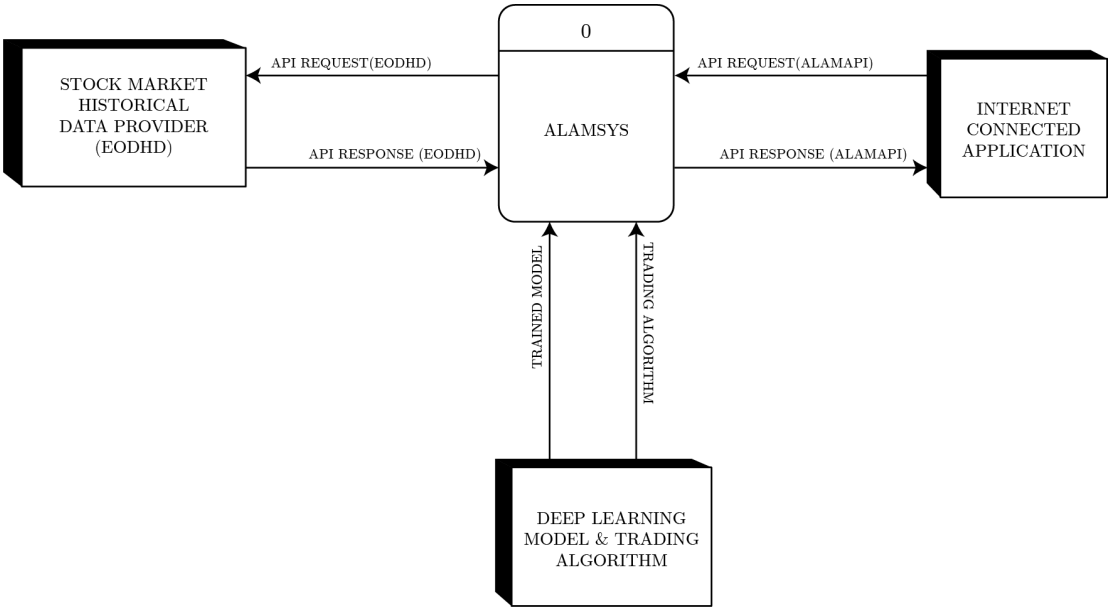

Figure 1.9: Context Diagram of the alamSYS

The diagram above depicts the root process (0), which is the alamSYS itself, and is linked to three external entities:

(a) Stock Market Historical Data Provider - which was provided by EODHD.

(b) Deep Learning Model & Trading Algorithm - these were developed alongside the alamSYS, specifically the DMD-LSTM Model and ALMACD, respectively.

(c) Internet Connected Application - these are any type of applications with internet access. Other applications may include a web-based application, a smart home speaker, and so on.

The data flow lines shown Figure 1.9 include the following:

(a) API Request(EODHD) - This is the request sent to the EODHD API to collect the stock market historical data.

(b) API Response(EODHD) - This is the response received from the EODHD API, which contains the end of day stock market data.

(c) API Request(alamAPI) - This are the requests sent by any internet connected application to the alamSYS via the alamAPI.

(d) API Response(alamAPI) - This are the responses sent by the alamSYS to any internet connected application via the alamAPI.
Whereas, in connection to the alamAPI, the following API points may be requested by the internet connected application:

- Home - This API endpoint returns a greeting message. Which should notify the user that they have connected to the alamAPI successfully.
- Stocks to Buy - This API endpoint returns a json data of recommended stocks to buy based on the current market price, the predicted price uptrend, and the entry signal of the trading algorithm in use.
- Stocks to Sell - This API endpoint returns a json data of recommended stocks to sell based on the current market price, the predicted price downtrend, and the exit signal of the trading algorithm in use.
- ML Model Info - This API endpoint returns a json data of the Machine Learning Models used in the alamSYS, as well as their associated information.
- Stock Risks Profile - This API endpoint returns a json data of stocks in the alamSYS as well as their risk values based on value at risk (%), volatility (%), and drawdown (%).

(e) Trained Model - This is the trained model, referring to the DMD-LSTM, that was used to predict the price movement of the stocks in the Philippine Stock Market.

(f) Trading Algorithm - This is the deployed trading algorithm, referring to the ALMACD, that was used to determine the entry and exit signals of the stocks in the Philippine Stock Market.

DFD of Diagram 0

To better understand how each data stream entering and exiting the root process is processed, we must look inside the inner workings of the root process, which is illustrated in the DFD of Diagram 0, as shown in Figure 1.10.

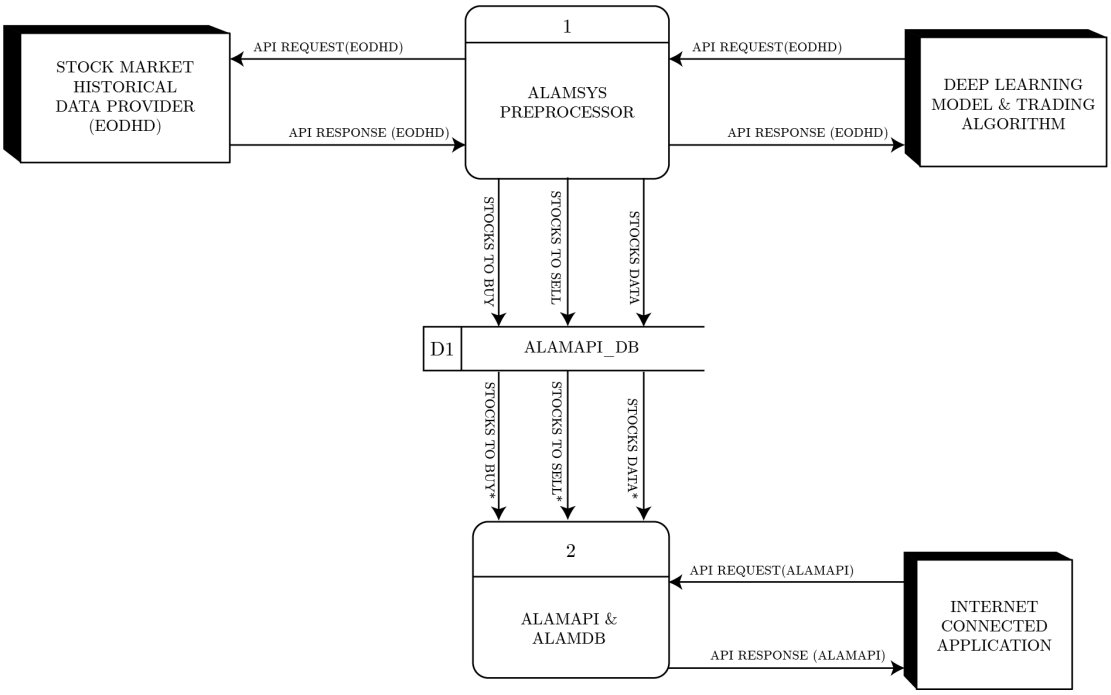

Figure 1.10: DFD of Diagram 0

From the figure above, the root process, has two main processes:

(a) alamSYS Preprocessor - which is the system's stock market data processing unit, which deploys the deep learning model (DMD-LSTM), and the trading algorithm (ALMACD) to predict the price movement of the stocks in the Philippine Stock Market.

(b) alamAPI & alamDB - which is the system's API and database unit, which is responsible for processing the API requests and responses, as well as storing the data of the system.

DFD of Diagram 1

To better understand the internal workings of the Process 1, it is useful to check the DFD of that process, which is illustrated in Figure 1.11

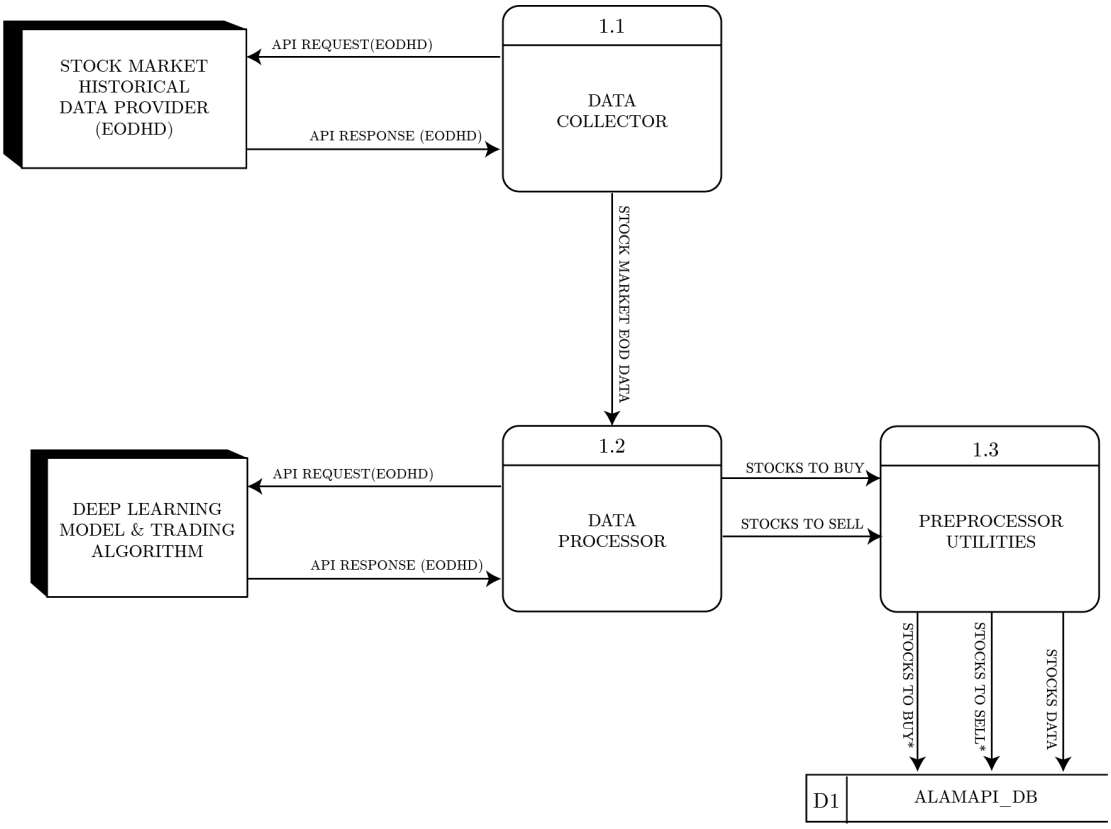

Figure 1.11: DFD of Diagram 1

From the figure shown above, it can be observed that Process 1 is composed of three internal processes, which are as follows:

(a) Data Collector - which is the main process responsible for collecting the historical market data using EODHD End of Day Market Data API.

(b) Data Processor - which is the main process responsible for processing the collected stock market data using the DMD-LSTM Model and the ALMACD Trading Algorithm.

(c) Preprocessor Utilities - these processes contains the utilities used by the data processor, such as the initialization of the database, database related actions, database models, stock symbols, and logs and alerts module.

DFD of Diagram 1.2

This shows the processes inside the process 1.2, which is the data processor.

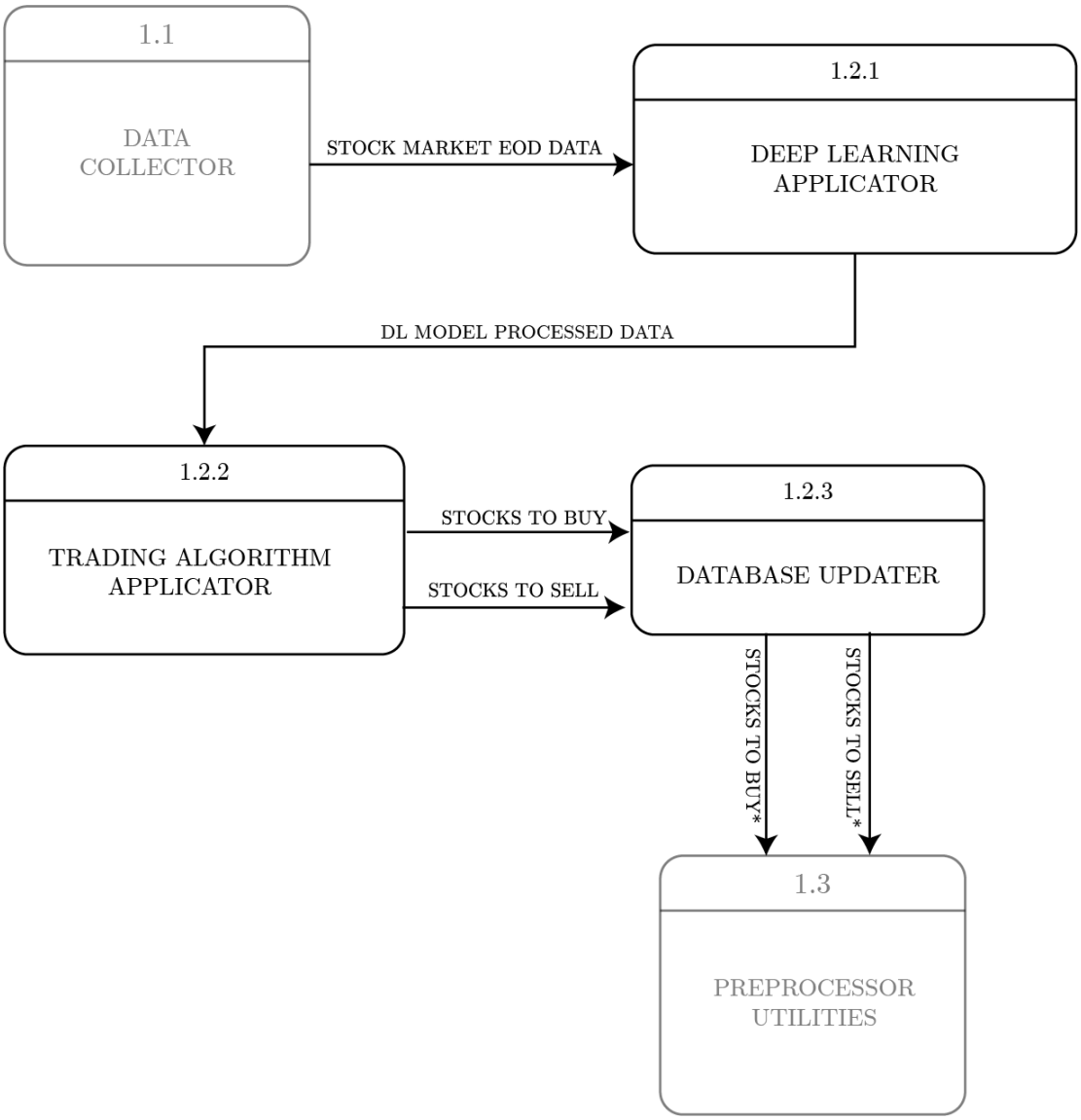

Figure 1.12: DFD of Diagram 1.2

The data processor is further composed of three processes, which are as follows:

(a) Deep Learning Applicator - This subprocess applies the DMD-LSTM model to the collected stock market end-of-day (eod) data.

(b) Trading Algorithm Applicator - The eod data alongside the list of predicted stock prices composes the "DL Model Processed Data", which is sent to this subprocess. This subprocess applies the ALMACD to better determine the entry (buy or hold), and exit (sell) signals for each stocks.

(c) Database Updater - A subprocess responsible for updating the contents of the database, based on the data processed by the Deep Learning Applicator and Trading Algorithm Applicator.

DFD of Diagram 2

Figure 1.13 shows the inner processes of the process 2 (alamAPI & alamDB).

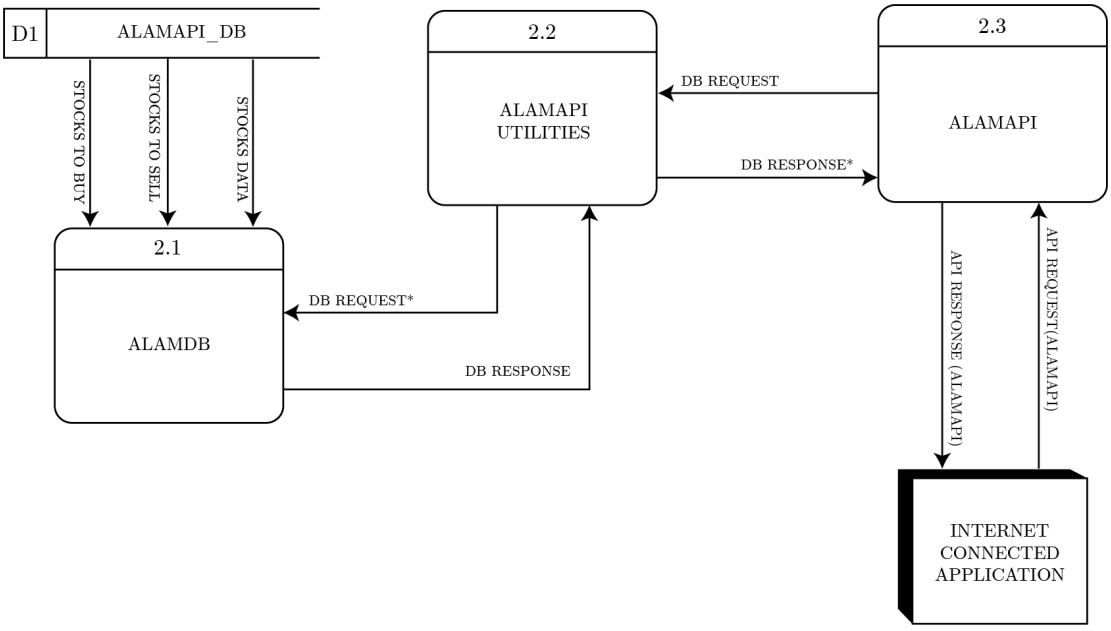

Figure 1.13: DFD 2: Data-Flow Diagram for the alamSYS

The figure above shows three internal processes of the Process 2, namely:

(a) alamDB - This process, processes the database request sent by the alamAPI as requested by the Internet Connected Application through the utilization of alamAPI utilities. It also sends the database response to the alamAPI via the same utilities module.

(b) alamAPI Utilities - This process serves as a mediator of database request and responses between the alamDB and alamAPI.

(c) alamAPI - This process contains all the API endpoints for the alamAPI, which is responsible for processing the requests and API responses from and to the connected users.

1.2.4 Object Document Mapper (ODM) Diagram

Because the system's database is non-relational, an Object Document Mapper (ODM) diagram rather than an Entity Relationship Diagram (ERD) is shown in this section.

The ODM diagram is shown in Figure 1.14:

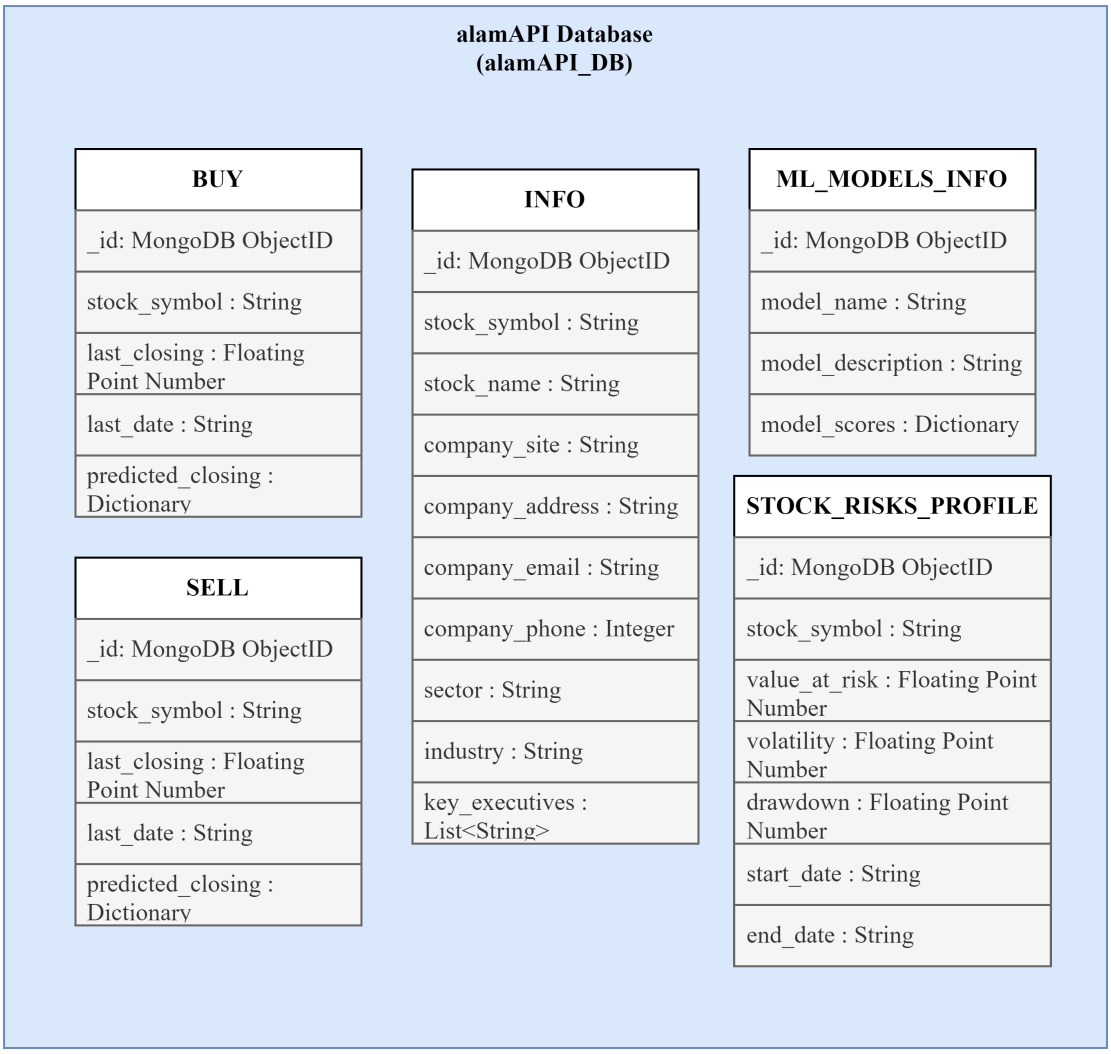

Figure 1.14: Object Document Mapper for the alamSYS Database

As shown from the Figure 1.14, the "alamAPI_DB" is the database name of the system. Where it is composed of five collections namely:

(a) Buy – this collection contains all the stocks that the data processor predicted and classified as a stock to buy. A sample of which is presented in Figure 1.15.

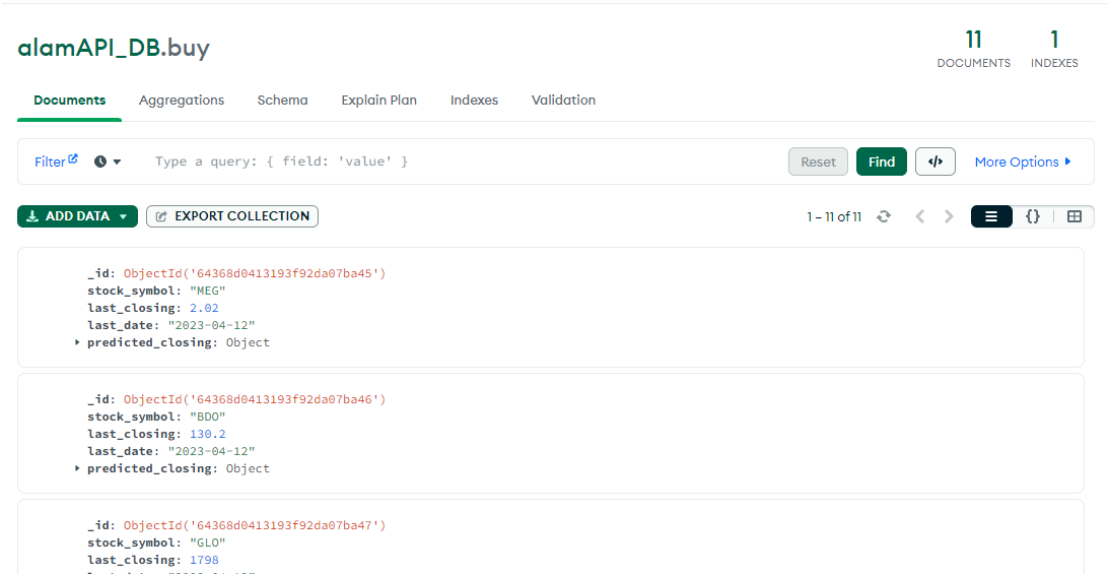

Figure 1.15: Sample Buy Collection from the alamSYS Database

(b) Sell – this collection contains all the stocks that the data processor predicted and classified as a stock to sell. a sample of which is presented in Figure 1.16.

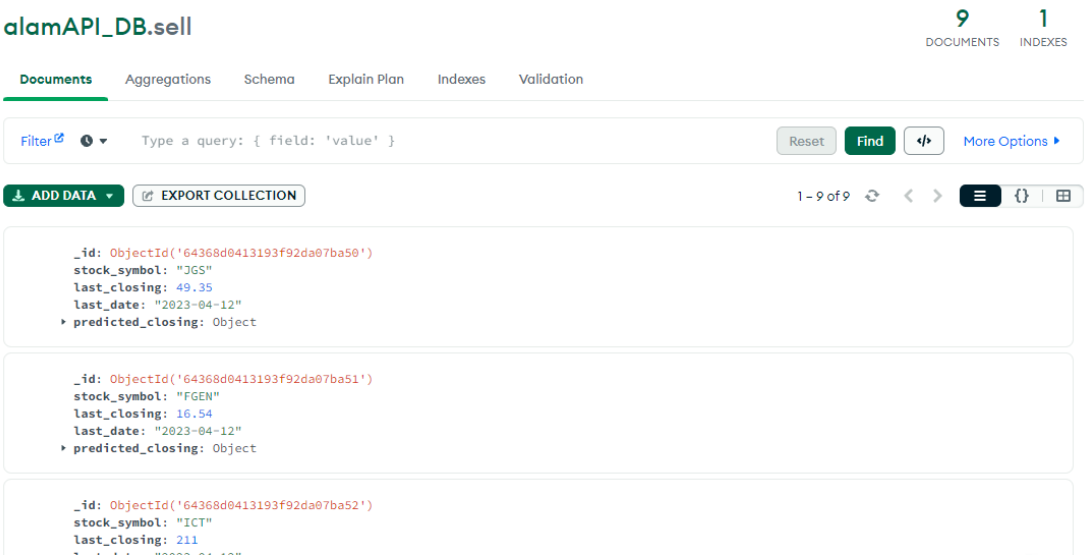

Figure 1.16: Sample Sell Collection from the alamSYS Database

(c) Info – this collection contains the general and relevant information about a stock, or the general company information. Such as the stock symbol, stock name, company site, company address, company email, company phone number, sector, industry, and the company's key executives. Where all of this information are gathered from their official listing accessed in the database of the Philippine Stock Exchange (PSE). A sample of which is presented in Figure 1.17.

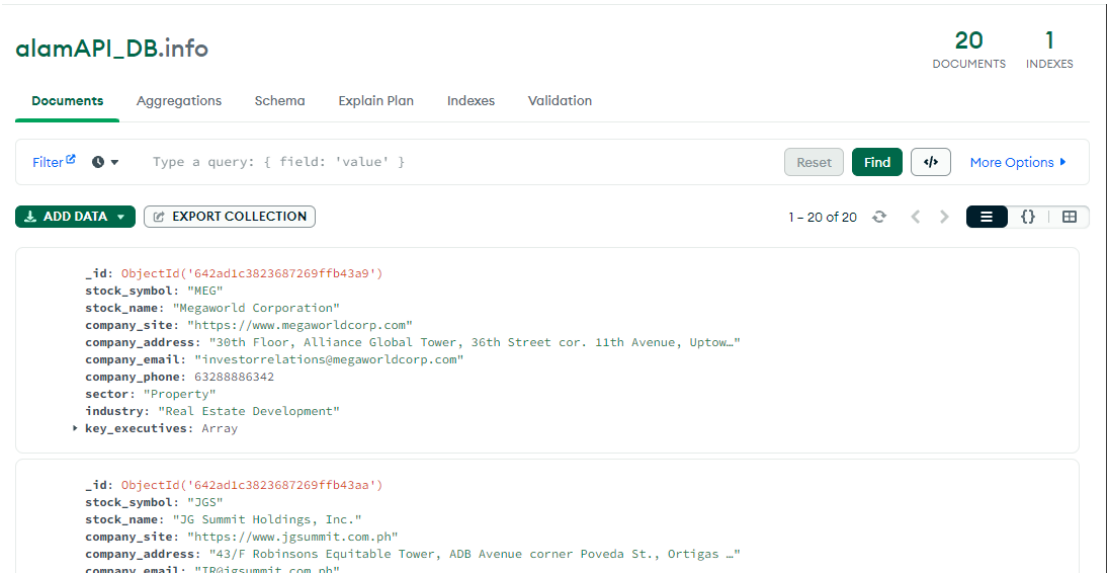

Figure 1.17: Sample Info Collection from the alamSYS Database

(d) Machine Learning (ML) Models Info – this collection contains the details about the Machine Learning Model/s deployed in the system. For the current alamSYS, only one model is deployed, which is the DMD-LSTM model. A sample of which is presented in Figure 1.19.

Figure 1.18: Sample ML Models Info Collection from the alamSYS Database

(e) Stock Risks Profile - this collection contains the details about the risk profiles for each stock. A sample of which is presented in Figure 1.19.

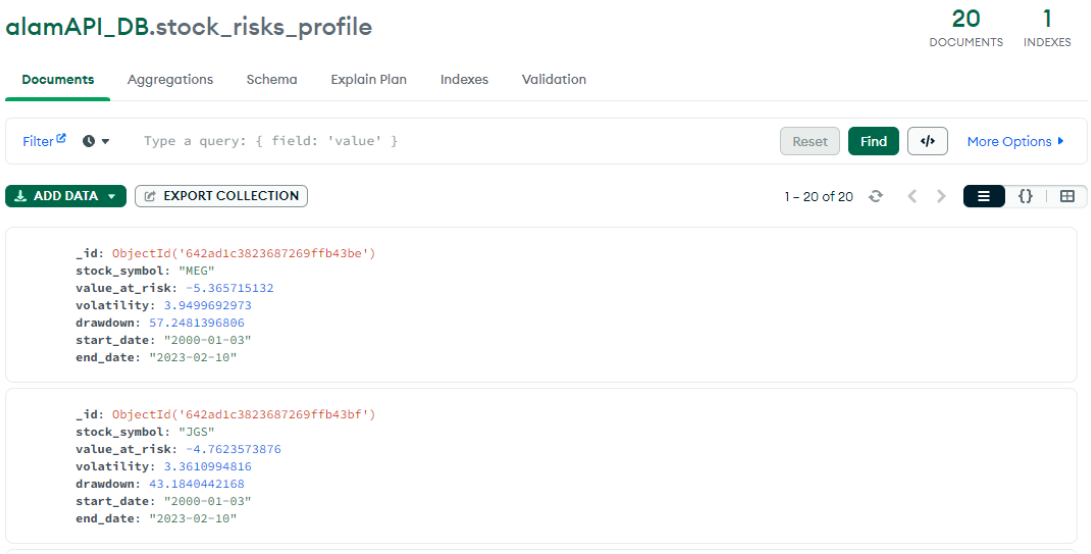

Figure 1.19: Sample Stock Risks Profile Collection from the alamSYS Database

Note that each collection are their own separate entities, hence the database is called non-relational, as the documents are not in any way related to each other.

1.2.5 Deep Learning Model Diagram

In this section, the process on how the deep learning model was developed is shown in Figure 1.20. Wherein, the process overview is based on the Fine-Tuned Support Vector Regression Model for Stock Predictions by Dash and Dash (2016).

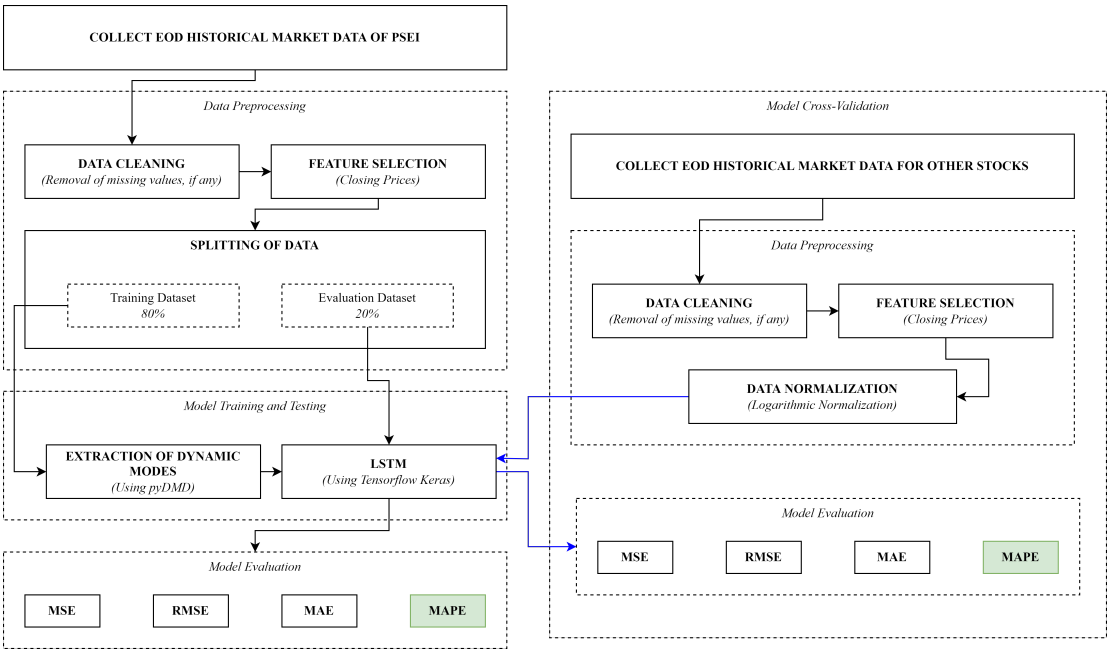

Figure 1.20: DMD-LSTM Model Development Methodology for alamSYS

Data Collection

The market data used to develop the DMD-LSTM model was obtained through EODHD's end-of-day market data API. While PSEI market data was used for model training and testing, market data from other stocks was used for cross-validation of the DMD-LSTM model. The following are the specifics of the stocks gathered:

Table 1.1: Collected Market Data Details

Stock	Data Count	Start Date	End Date
AC	6809	June 27, 1994	February 10, 2023
ALI	6789	June 27. 1994	February 10, 2023
AP	3795	July 16, 2007	February 10, 2023
BDO	5041	May 22, 2002	February 10, 2023
BLOOM	3033	October 30, 2000	February 10, 2023
FGEN	4142	February 02, 2006	February 10, 2023
GLO	6707	January 03, 1995	February 10, 2023

Table 1.1 continued from previous page

Stock	Data Count	Start Date	End Date
ICT	6805	January 03, 1995	February 10, 2023
JGS	6525	June 27, 1994	February 10, 2023
LTG	3774	February 06, 1995	February 10, 2023
MEG	6751	January 03, 1995	February 10, 2023
MER	6799	June 27, 1994	February 10, 2023
MPI	3888	December 18, 2006	February 10, 2023
PGOLD	2762	October 06, 2011	February 10, 2023
PSEI	5675	January 03, 2000	February 10, 2023
RLC	5879	June 27, 1994	February 10, 2023
RRHI	2253	November 11, 2013	February 10, 2023
SMC	6799	June 27. 1994	February 10, 2023
TEL	6814	June 27, 1994	February 10, 2023
URC	6135	June 03, 1995	February 10, 2023

Data Preprocessing

Data preprocessing before model training and testing is composed of three main processes which are as follows:

(a) Data Cleaning - This was done to clean any missing values from the data.

(b) Feature Selection - Closing prices was selected as the main feature of the model.

(c) Splitting of Data - Data was split in the ratio of 80:20 for testing and training data, respectively.

Meanwhile for the data preprocessing for cross-validation, the following processes were done:

(a) Data Cleaning - This was done to clean any missing values from the data.

(b) Feature Selection - Closing prices was selected as the main feature of the model.

(c) Data Normalization - Using logarithmic normalization method, the data was normalized. Logarithmic normalization was utilized to help solved the problem with the data having extreme ranges, which affects the evaluation of the cross-validation. In essence it was used to enable data stability, and increase interpretability (Baeldung, 2022; Tuychiev, 2021; Andrew, 2019).

Model Training and Testing

Using pyDMD the dynamic modes was extracted from the training and testing data, these extracted values alongside the actual closing price data were utilized for the training of an LSTM model using Tensorflow Keras Library.

There are a total of eight model variations trained and tested, which are as follows: (1) Baseline LSTM with window size of 5; (2) Baseline LSTM with window size of 10; (3) Baseline LSTM with window size of 15; (4) Baseline LSTM with window size of 20; (5) DMD-LSTM with window size of 5; (6) DMD-LSTM with window size of 10; (7) DMD-LSTM with window size of 15; and (8) DMD-LSTM with window size of 20. Where the best performing model was used for the cross-validation and was deployed to the system.

Model Evaluation

The DMD-LSTM Model was evaluated using the following error metrics:

(a) Mean Squared Error (MSE) - MSE is a well-known metric for assessing regression models. It calculates the average of the squared differences between predicted and true values. MSE is useful because it penalizes large errors more severely than small errors, which is important in some applications. A lower MSE indicates that the model performed better (Glen, n.d.-a).

(b) Root Mean Squared Error (RMSE) - Another popular metric for evaluating

regression models is the RMSE, which is the square root of the MSE. It, like MSE, computes the average of the differences between predicted and true values. Because it is in the same unit as the target variable, RMSE is easier to interpret. A lower RMSE indicates that the model is performing better (Glen, n.d.-b).

(c) Mean Absolute Error (MAE) - It calculates the average of the absolute differences between predicted and true values. MAE is advantageous because it is more resistant to outliers than MSE and RMSE. A lower MAE indicates that the model is performing better (Secret Data Scientist, 2023).

(d) Mean Absolute Percentage Error (MAPE) - It computes the average percentage difference between predicted and true values. MAPE is useful because it provides a relative measure of error, which makes comparing model performance across different target variable scales easier. A lower MAPE indicates that the model is performing better (Allwright, 2022). Furthermore, this is the primary error metric used to select the final model deployed to alamSYS.

Model Cross-Validation

The model selected for the cross-validation was the DMD-LSTM model with a window size of 5, due to it being the highest performing model based on having the lowest MAPE scores compared to the other models, this is further discussed on Chapter 4 of this paper.

The cross-validation was conducted using the stock market data from the other stocks aside from the training data from PSEI. Where cross-validation of an LSTM model must be done before deployment to assess the model's generalization performance. LSTM models are well-known for their ability to capture long-term dependencies in sequential data, but their performance varies greatly depending on the dataset and model hyperparameters used. And a correctly performed cross-validation helps to provide a more accurate estimate of the model's performance on unseen data, which is critical for ensuring that the model performs well in real-world scenarios (Mellema, 2020; Scherzinger, Roennau, & Dillmann, 2019).

Model Deployment

After determining that the DMD-LSTM model works well with non-training stock market data, it was deployed to the alamSYS as a '.keras' file.

1.2.6 Docker-Compose Layer Diagram

This section illustrates the different layers of the docker-compose containers based on the way it was used in the deployment of the system.

Figure 1.21 shows a diagram based on Docker documentation, to better understand the containers and layers of the alamSYS. textitNote that in the diagram the lowest level is the "Server Infrastructure" and the highest level is the "Docker-Compose" layer.

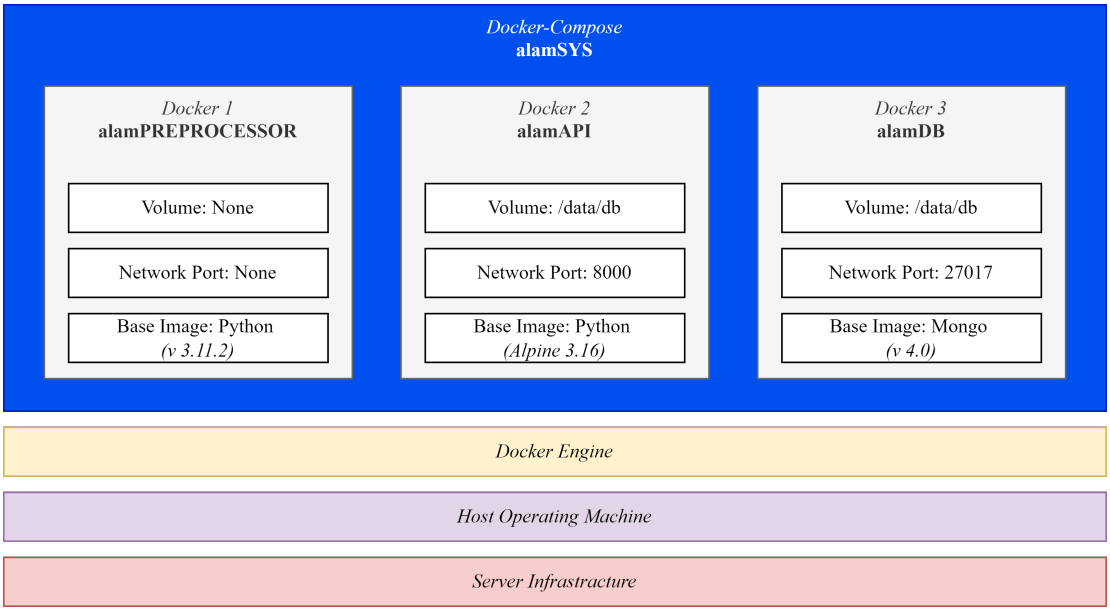

Figure 1.21: Docker-Compose Layer Diagram for the alamSYS

From the figure above, it can be observed that the docker-compose layer is composed of three docker containers, which are as follows:

(a) Docker 1 : alamPREPROCESSOR - This is the docker container that contains the alamPREPROCESSOR application. It uses Python 3.11.2 as its based image. Moreover, it is not attached to any volume or network port.

(b) Docker 2 : alamAPI - This is the docker container that contains the alamAPI application. It uses a Python image as well, but is running on top of an Alpine 3.1 image. This was done as Alpine images runs on minimal resources than other Linux based images. Furthermore this container is connected to the data/db volume, and its network port 8000 is exposed.

(c) Docker 3 : alamDB - This is the docker container that contains the alamDB application which runs on top of the Mongo version 4.0 image. It is also connected to the data/db volume. And its network port 27017 is exposed.

1.3 Hardware Specifications

This section discusses the hardware specification utilized in the development of the different components of the alamSYS, as well as other devices used for testing.

1.3.1 For the Development of the alamSYS, Deep Learning Model, and Mobile-Based Test Application

The development of the alamSYS and its components, as well as for the development and testing of the deep learning model and mobile-based test application, utilized a laptop device with the following hardware specifications:

(a) CPU - Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz to 3.10 GHz. *Note that only the CPU was utilized for the development, however the device has a dedicated GPU: NVIDIA GeForce GTX 1050*

(b) Memory - 16GB DDR4 @2666Mhz

1.3.2 For Deployed System Testing

In order to test the deployment capacity of the alamSYS, other computers were used to connect to the server. In this case the aforementioned laptop device was used as a deployment server. Meanwhile listed below are the specifications of the desktop devices used for the deployment testing. *Note that a total of 10 desktop devices were utilized.*

(a) CPU - Intel Core Pentium

(b) Memory - 4GB DDR4 RAM

1.3.3 For the Test Application

Moreover, the mobile device used to showcase the main features of the alamSYS has the following specifications:

(a) CPU - 8-core 3.2GHz (Snapdragon 870 5G)

(b) Memory - 8GB RAM

1.4 Methodology

This section of the Chapter 3 will be divided into two sections:

(a) Software Development Process, wherein an Agile development will be discussed; and

(b) Procedures, wherein the general procedures of development will be tackled.

1.4.1 Software Development Process

Due to the expected heavy time constraints of the development of the system, the author of this paper decided to follow an Agile Software Development Process, primarily it will be using Agile Sprints for an efficient time management during the whole software development process. Wherein the following are the list of Sprints and sub-activities that will be followed are shown in the Table below:

Table 1.2: Summary of Sprints and Target Activities

Sprint Number	**Target Activities**	**Allotted Time**[1]
1	**Main Activity:** System Planning and Evaluation **Sub-Activities:** • Topic Proposal • Drafting of Chapters 1 to 3 for the Special Problem Proposal • System Architecture and User Requirement Analysis	**12 Weeks** Start: September 15, 2022 End: December 9, 2022

Table 1.2 continued from previous page

Sprint Number	Target Activities	Allotted Time[1]
2	**Main Activity:** System Prototyping **Sub-Activities:** • Build the different component of the alamSYS as indicated in the top-level overview diagram of the system, the following prototype will be developed: [1.] API endpoints [2.] Database [3.] Preprocessor • Testing of the build prototype. This also include creating unit test cases for each component. • Initial Documentations, this will be done inside the GitHub repository.	**12 Weeks** Start: September 30, 2022 End: April 3, 2023

Table 1.2 continued from previous page

Sprint Number	Target Activities	Allotted Time[1]
3	**Main Activity:** Machine Learning Model Training, Testing, and Evaluation **Sub-Activities:** • Collection of Historical Data, outside the Data Collector module of the system. As the full data will be needed for each stock for the training, rather the 200-day only historical data. Whereas the last date on the market data should be January 13, 2023. • Development of the Machine Learning Model. This includes data standardization, data splitting, and data training. • Machine Learning model testing and evaluation. • Revision of Chapter 1-3, in preparation for the final paper submission.	**10 Weeks** Start: January 15, 2023 End: March 30, 2023

Table 1.2 continued from previous page

Sprint Number	Target Activities	Allotted Time[1]
4	**Main Activity:** Integration of Machine Learning Model to the alamSYS and Additional Data Collection **Sub-Activities:** • Testing and Evaluation of alamSYS with the integration of the Machine Learning Model. • System Testing, this will be done to verify the functionality of the whole system, given a test deployment environment. Moreover, this will be done in a span of 4 weeks • Drafting of Chapter 4 and 5	**6 Weeks** Start: March 31, 2023 End: May 12, 2023

Table 1.2 continued from previous page

Sprint Number	Target Activities	Allotted Time[1]
5^2	**Main Activity:** System Documentation **Sub-Activities:** • Updating and Finalization of Documentations included in the GitHub Repository. • Writing of the results, discussions, conclusions, and recommendations for Chapter 4 – 5 • Special problem paper revisions • Start the development of the test application (for showcasing of the system features)	**6 Weeks** Start: April 14, 2023 End: May 26, 2023

Table 1.2 continued from previous page

Sprint Number	Target Activities	Allotted Time[1]
6[2]	**Main Activity:** Preparation for Final Defense and System Presentation **Sub-Activities:** • Finalization of the mobile-based test application • Revisions and Finalization of the special problem paper. • Creation of presentation slide deck for the presentation of the special problem.	**3 Weeks** Start: May 27, 2023 End: June 17, 2023

1. Start and End Dates are based on the University's Academic Calendar and the Schedule provided by the Special Problem Adviser.
2. Sprints 5 and 6 are no longer part of the actual system development but is still included as a basis for the Gantt chart. Moreover, these activities can still be considered as part of the documentation process.

From Table 1.2, it is shown that there is a total of 39 weeks; from September 15, 2022, to June 17, 2023, however it must be noted that an additional 1 week was added to each sprint's allotted time to compensate for any unforeseen events during each sprint.

It should also be noted that Sprint 1 and Sprint 2 overlaps as the development

of the prototype will start at Week 3, this will be possible as there will already be an initial system design to be followed, and any changes made during Sprint 1 can easily be adjusted to the creation of the prototype of the system in Sprint 2. This is also the case for Sprints 4 and 5, since their activities overlaps with each other, such that there are things in Sprint 4 that are unsupervised, hence, to better manage the time it is reasonable to start the activities of Sprint 5 along side the later parts of Sprint 4.

Moreover, the full details about the scheduling will be further discussed in the Gantt Chart of this chapter.

1.4.2 Procedures

In this section, the step-by-step procedures that will be followed in line with the development and testing of the system; alamSYS. Whereas the following are the procedures:

(a) Designing of the System Architecture for alamSYS

(b) Designing of Machine Learning Model

(c) Development of System Prototype

(d) Training, Testing, and Evaluation of the Machine Learning Model

(e) Integration of the Machine Learning Model to the alamSYS

(f) Initial testing for alamSYS, this shall also include any debugging, bug fixing, and code refactoring.

(g) Pre-deployment testing, this testing phase includes the following tests that will be done for a one-month continuous system operation:

- Functional Testing, by monitoring the functionality of the alamSYS over 30 days and checking the success and error logs at the end of the given timeframe.

- Stress Testing, by creating ten-million artificial requests to the API everyday for 30 days.

(h) Logging and summarization of results from all the prior tests conducted

(i) Analysis and discussion of test data results.

(j) Code Documentation

(k) Maintenance, which will span beyond the time scope of the special problem.

1.5 Gantt Chart

Based on Table refsummary-sprints, the following figures for the Gantt Chart (created using TeamGantt) shows the software development schedule for the development of alamSYS. The Gantt Chart is divided into the different sprint to present the project scheduling. Moreover, a zoomed-out view of the whole Gantt Chart, will also be provided at the end of this section.

1.5.1 Gantt Chart for Sprint 1

Figure 1.22 shows the schedule of activities for Sprint 1. Wherein, it will start on September 15, 2022, and end on December 9, 2022.

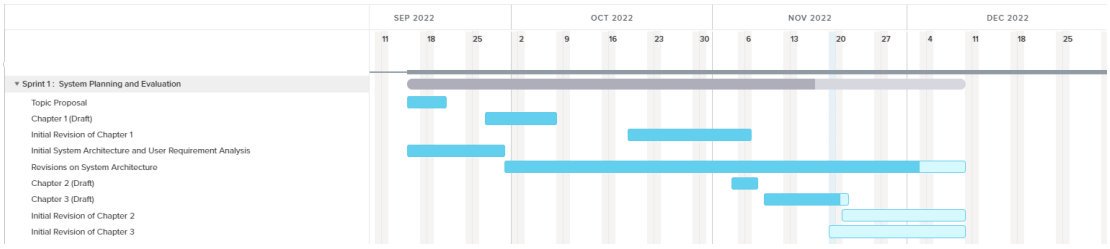

Figure 1.22: Gantt Chart for Sprint 1

1.5.2 Gantt Chart for Sprint 2

Figure 1.23 shows the schedule of activities for Sprint 2. Which will start on September 30, 2022, and end on January 5, 2023.

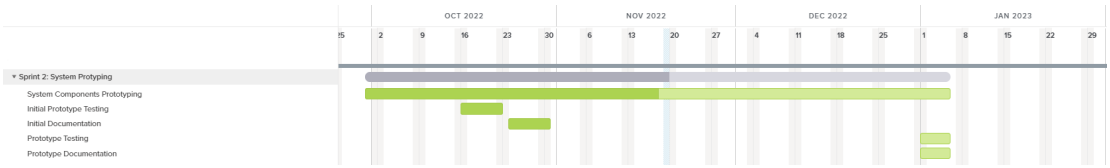

Figure 1.23: Gantt Chart for Sprint 2

1.5.3 Gantt Chart for Sprint 3

Figure 1.24 shows the schedule of activities for Sprint 3. Wherein, it will start on January 15, 2023, and end on March 30, 2023.

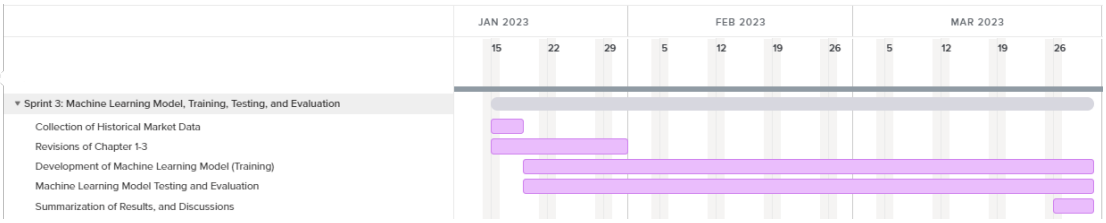

Figure 1.24: Gantt Chart for Sprint 3

1.5.4 Gantt Chart for Sprint 4

Figure 1.25 shows the schedule of activities for Sprint 4. Which will run from March 31, 2023, until May 12, 2023.

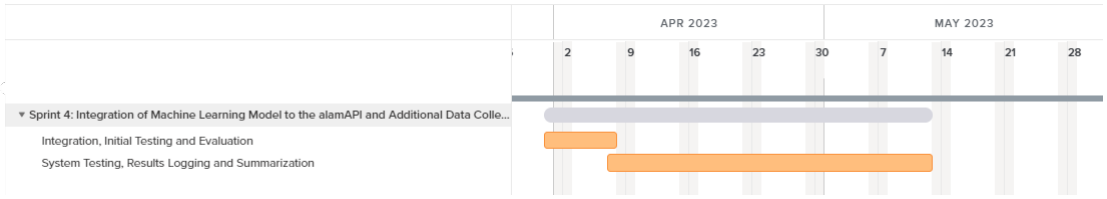

Figure 1.25: Gantt Chart for Sprint 4

1.5.5 Gantt Chart for Sprint 5

Figure 1.26 shows the schedule of activities for Sprint 5. Which will be from April 14, 2023, to May 12, 2023.

Figure 1.26: Gantt Chart for Sprint 5

1.5.6 Gantt Chart for Sprint 6

Figure 1.27 shows the schedule of activities for the final sprint for the development of alamSYS. Which will be done from May 27, 2023, until June 17, 2023. However, it should be noted that the end day may change, depending on the scheduled final defense.

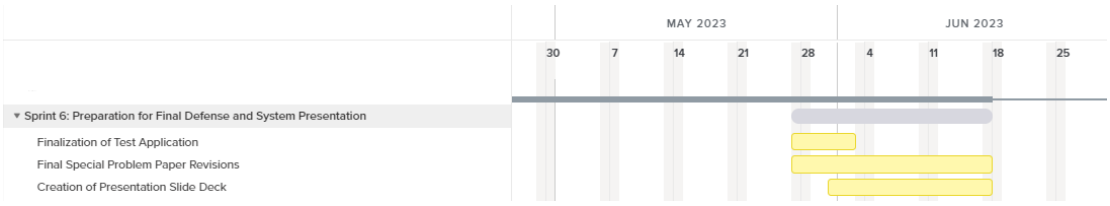

Figure 1.27: Gantt Chart for Sprint 6

1.5.7 Full Gantt Chart

To have an overview of the whole schedule of each Sprints, the full Gantt chart is shown in Figure 1.28.

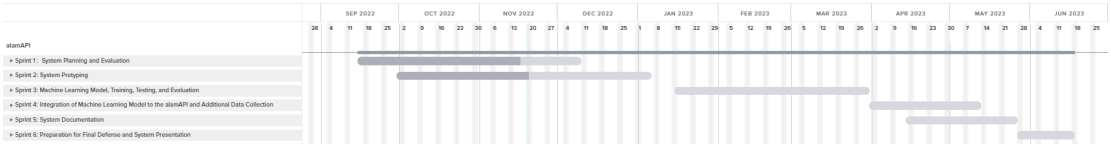

Figure 1.28: Full Gantt Chart

References

Allwright, S. (2022). *What is a good mape score?* Retrieved April 15, 2023, from https://stephenallwright.com/good-mape-score/

Andrew. (2019). *You should (usually) log transform your positive data.* Retrieved April 15, 2023, from https://statmodeling.stat.columbia.edu/2019/08/21/you-should-usually-log-transform-your-positive-data/

Baeldung. (2022). *Normalization inputs for an artificial neural network.* Retrieved April 15, 2023, from https://www.baeldung.com/cs/normalizing-inputs-artificial-neural-network

Dash, R., & Dash, P. K. (2016, 3). A hybrid stock trading framework integrating technical analysis with machine learning techniques. *The Journal of Finance and Data Science*, *2*, 42-57. doi: 10.1016/j.jfds.2016.03.002

Docker. (n.d.). *Use containers to build, share and run your applications. docker + wasm.* Retrieved November 13, 2022, from https://www.docker.com/resources/what-container/

EODHD. (n.d.). *Financial apis documentation. fundamental and historical data apis.* Retrieved November 13, 2022, from https://eodhistoricaldata.com/financial-apis/

Geeks for Geeks. (2022). *Python lists vs numpy arrays.* Retrieved January 13, 2023, from https://www.geeksforgeeks.org/python-lists-vs-numpy-arrays/

Glen, S. (n.d.-a). *Mean squared error: Definition and example.* Retrieved April 15, 2023, from https://www.statisticshowto.com/probability-and-statistics/statistics-definitions/mean-squared-error/

Glen, S. (n.d.-b). *Rmse: Root mean square error.* Retrieved April 15, 2023, from https://www.statisticshowto.com/probability-and-statistics/regression-analysis/rmse-root-mean-square-error/

Mellema, G. R. (2020). Improved active sonar tracking in clutter using integrated feature data. *IEEE Journal of Oceanic Engineering*, *45*(1), 304-318. doi: 10.1109/JOE.2018.2870234

MongoEngine. (n.d.). *Mongoengine documentation.* Retrieved November 13, 2022, from `https://docs.mongoengine.org/`

Scherzinger, S., Roennau, A., & Dillmann, R. (2019). Contact skill imitation learning for robot-independent assembly programming. *CoRR*, *abs/1908.06272*. Retrieved from `http://arxiv.org/abs/1908.06272`

Secret Data Scientist. (2023). *What is mae (mean absolute error)?* Retrieved April 15, 2023, from `https://secretdatascientist.com/mae-mean-absolute-error/`

Tiangolo. (n.d.). *Fastapi documentation.* Retrieved November 13, 2022, from `https://fastapi.tiangolo.com/`

Tuychiev, B. (2021). *How to differentiate between scaling, normalization, and log transformations.* Retrieved April 15, 2023, from `https://towardsdatascience.com/how-to-differentiate-between-scaling-normalization-and-log-transformations-69873d365a94`

VisualParadigm. (n.d.). *Gane-sarson data flow diagram tutorial.* Retrieved November 15, 2022, from `https://online.visual-paradigm.com/knowledge/software-design/gane-sarson-dfd-tutorial/`