

**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

<b>1 Materials and Methods</b>	<b>1</b>
1.1 Development Tools and Software Requirements . . . . .	1
1.1.1 Development Tools . . . . .	2
1.1.2 Software Requirements . . . . .	2
1.2 System Diagrams . . . . .	5
1.2.1 Top-Level Overview Diagram of the alamSYS and Its Interactions to External Systems . . . . .	5
1.2.2 Process Flow Diagram . . . . .	7
1.2.3 Data-Flow Diagram (DFD) . . . . .	16
1.2.4 Object Document Mapper (ODM) Diagram . . . . .	22
1.2.5 Machine Learning Model Diagram . . . . .	24
1.2.6 Docker-Compose Layer Diagram . . . . .	24
1.3 Hardware Requirements . . . . .	25
1.3.1 For the Development of the alamSYS, and Training and Testing of the Machine Learning Model . . . . .	25

1.3.2	For the Development of the Test Application and System Testing . . . . .	26
1.4	Methodology . . . . .	26
1.4.1	Software Development Process . . . . .	26
1.4.2	Procedures . . . . .	33
1.5	Gantt Chart . . . . .	34
1.5.1	Gantt Chart for Sprint 1 . . . . .	34
1.5.2	Gantt Chart for Sprint 2 . . . . .	35
1.5.3	Gantt Chart for Sprint 3 . . . . .	35
1.5.4	Gantt Chart for Sprint 4 . . . . .	35
1.5.5	Gantt Chart for Sprint 5 . . . . .	36
1.5.6	Gantt Chart for Sprint 6 . . . . .	36
1.5.7	Full Gantt Chart . . . . .	37
	<b>References</b>	<b>38</b>

# List of Figures

1.1	Top-Level Overview of the alamSYS and Interactions with External Applications/Systems . . . . .	6
1.2	Full Overview of the Process Flow Diagram for the alamSYS . . . . .	8
1.3	Overview of the Process Flow Diagram for the Scheduler . . . . .	9
1.4	Overview of the Process Flow Diagram for the Data Collector . . . . .	11
1.5	Overview of the Process Flow Diagram for Data Processor . . . . .	13
1.6	Overview of the Process Flow Diagram for the Deep Learning Model Applicator . . . . .	14
1.7	Overview of the Process Flow Diagram for the Trading Algorithm Applicator . . . . .	15
1.8	Overview of the Process Flow Diagram for the Database Updater . . . . .	15
1.9	Context Diagram of the alamSYS . . . . .	16
1.10	DFD of Diagram 0 . . . . .	18
1.11	DFD of Diagram 1 . . . . .	19
1.12	DFD of Diagram 1.2 . . . . .	20
1.13	DFD 2: Data-Flow Diagram for the alamSYS . . . . .	21
1.14	Object-Data-Model for the alamSYS . . . . .	22

1.15 Machine Learning Model for the alamSYS . . . . .	24
1.16 Docker-Compose Layer Diagram for the alamSYS . . . . .	25
1.17 Gantt Chart for Sprint 1 . . . . .	34
1.18 Gantt Chart for Sprint 2 . . . . .	35
1.19 Gantt Chart for Sprint 3 . . . . .	35
1.20 Gantt Chart for Sprint 4 . . . . .	36
1.21 Gantt Chart for Sprint 5 . . . . .	36
1.22 Gantt Chart for Sprint 6 . . . . .	36
1.23 Full Gantt Chart . . . . .	37

# **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. (Tiango, 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.

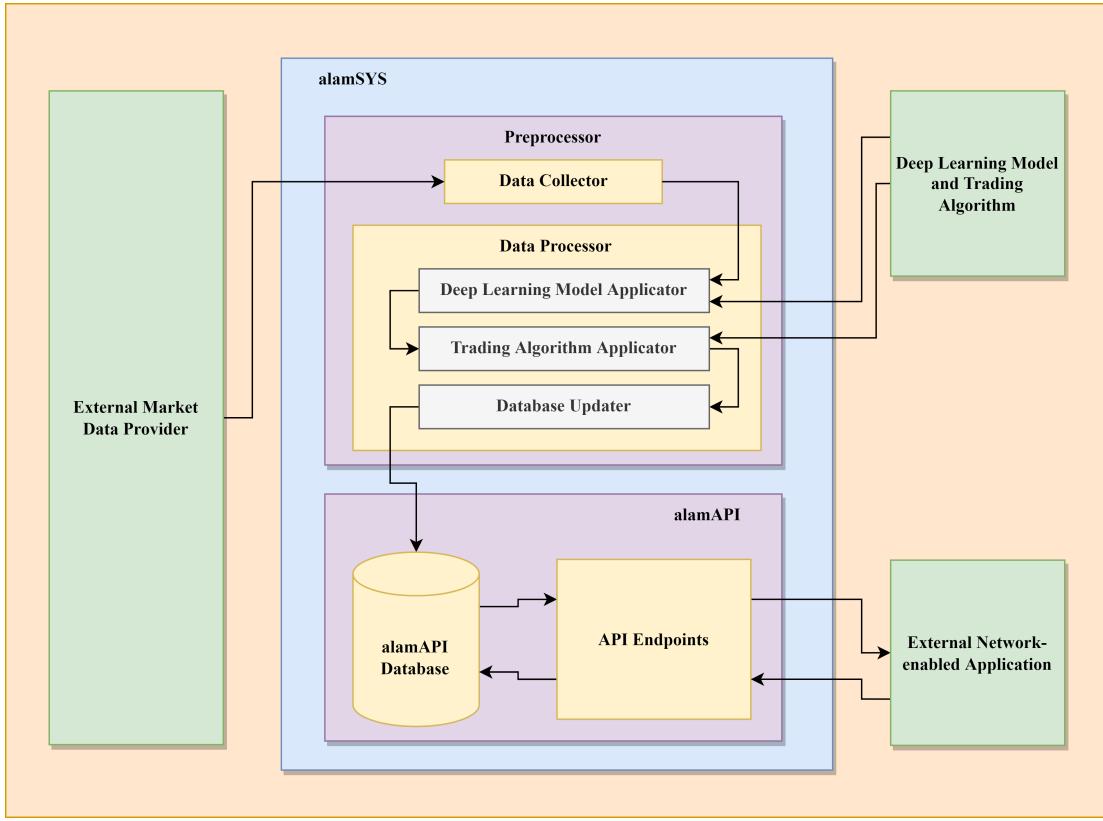


Figure 1.1: Top-Level Overview of the alamSYS and Interactions with External Applications/Systems

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.

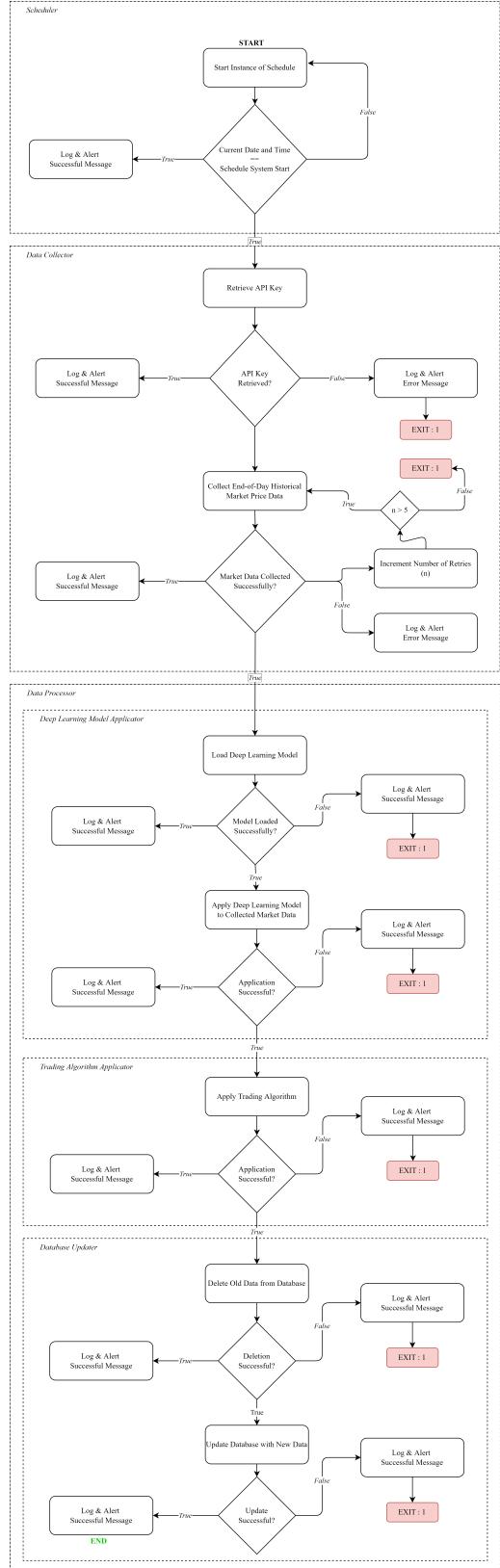


Figure 1.2: Full Overview of the Process Flow Diagram for the alamSYS

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

## Scheduler

Using CRON, a Linux-based scheduler, a scheduled task is provided to the server running the system. Since, the system will be containerized in a Linux System, the scheduler will run once the instance of the Docker Engine is running on the server system, which can be of any operating system. Then, if the current date and time of the contained system matches the scheduled date and time from CRON, it will log that the scheduled task has started, otherwise, it will not do anything and will check again for the current date and time.

The consequent processes in this process flow diagram will run after the scheduled task is initiated. Wherein the schedule task will run everyday from Mondays to Fridays, every 5:00 P.M. And the whole process can be seen in the Figure 1.3.

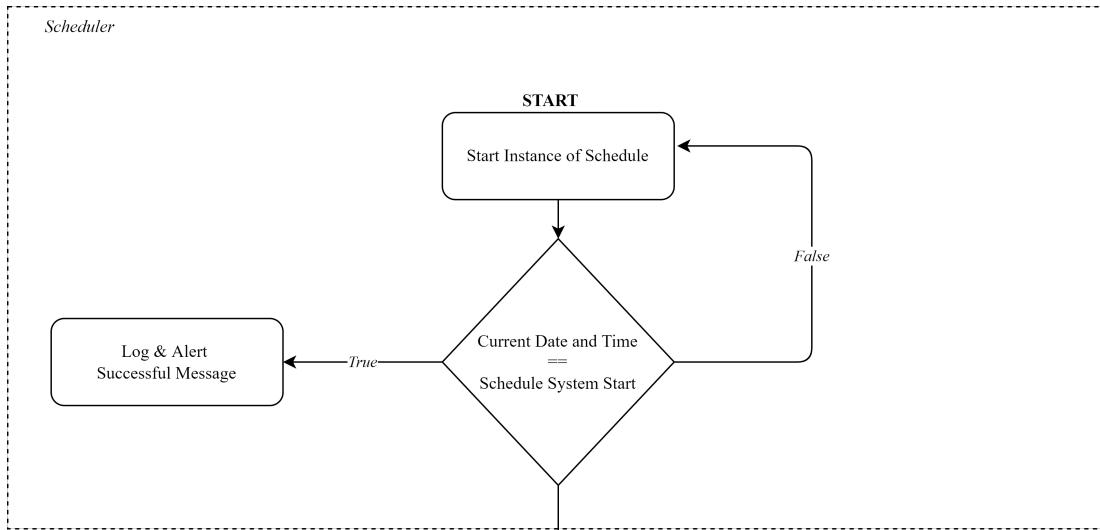


Figure 1.3: Overview of the Process Flow Diagram for the Scheduler

## **Data Collector**

This is the first task that the scheduled activity will do, which is simply to connect to the historical market data provider and collect the historical market data that is updated for the current date.

Wherein, if the collection is successful, it will log that the system has successfully connected and collected the updated historical market data for that day and will proceed to use the collected data to the Machine Learning Model.

Otherwise, it will log the error, and it will check the error message. Wherein, if the error message shows “Error 406” or “Payment Needed”, then the scheduled task will end in this section. This is also the reason why the end of each process ends in logging the activities of the system, so that the maintainers of the system can easily pin-point the problem to be fixed during the actual deployment of the alamSYS. Moreover, if the error is anything else, then the system will retry to collect the data for a maximum of five tries, and if it still encounters an error during the retry window, the session will also end.

The flow of processes discussed above can be observed in Figure 1.4.

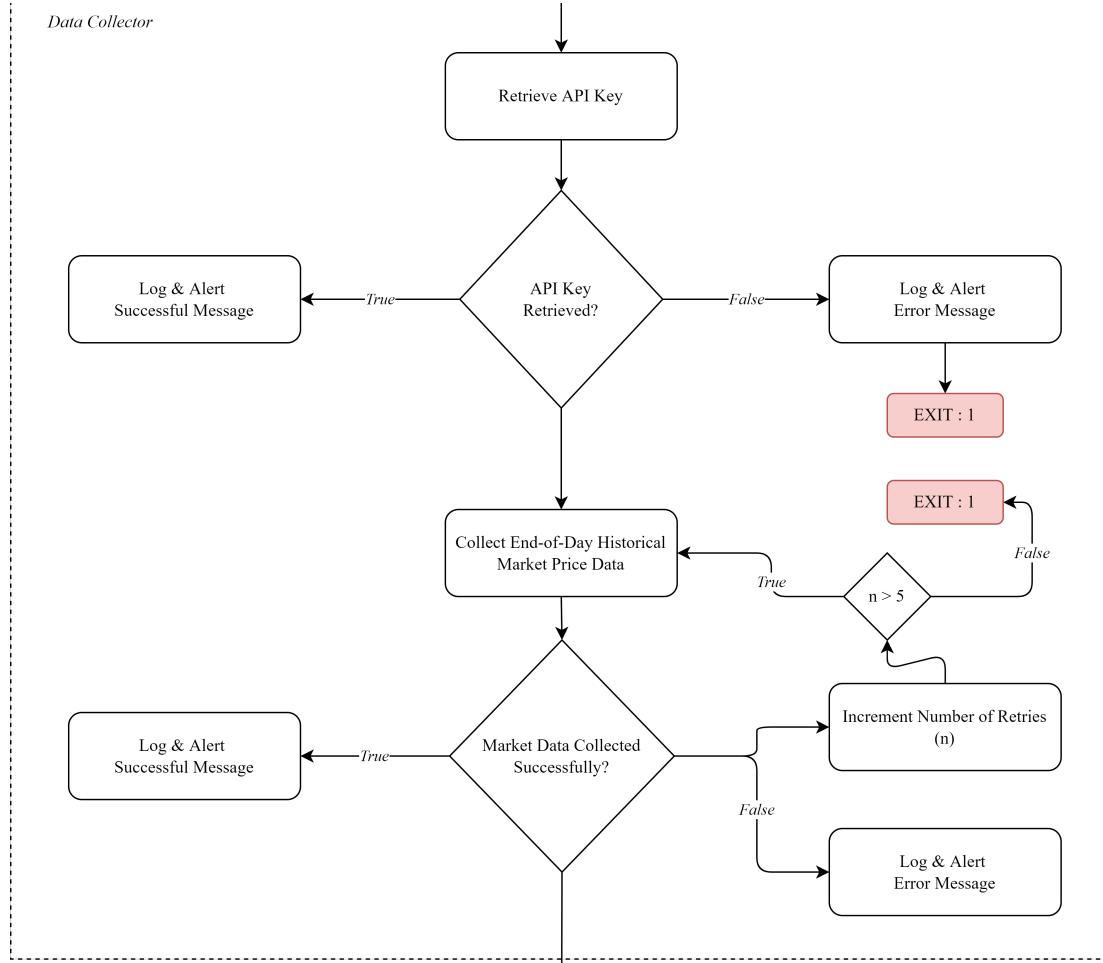


Figure 1.4: Overview of the Process Flow Diagram for the Data Collector

## Machine Learning Model Application

In this process the developed machine learning model/algorithm will use the current historical market data collected to predict the future trend of the stock market and decide whether that stock should be bought or sold for the next market day.

Wherein if the application of the machine learning model is successful, then the system will log the success of the operation and proceeds into updating the database.

Otherwise, it will log the error, and will retry the operation for a maximum of five times. Once after the five retries is unsuccessful, then the system will end the session at this stage.

The flow of processes discussed above can be observed in Figure 1.5.

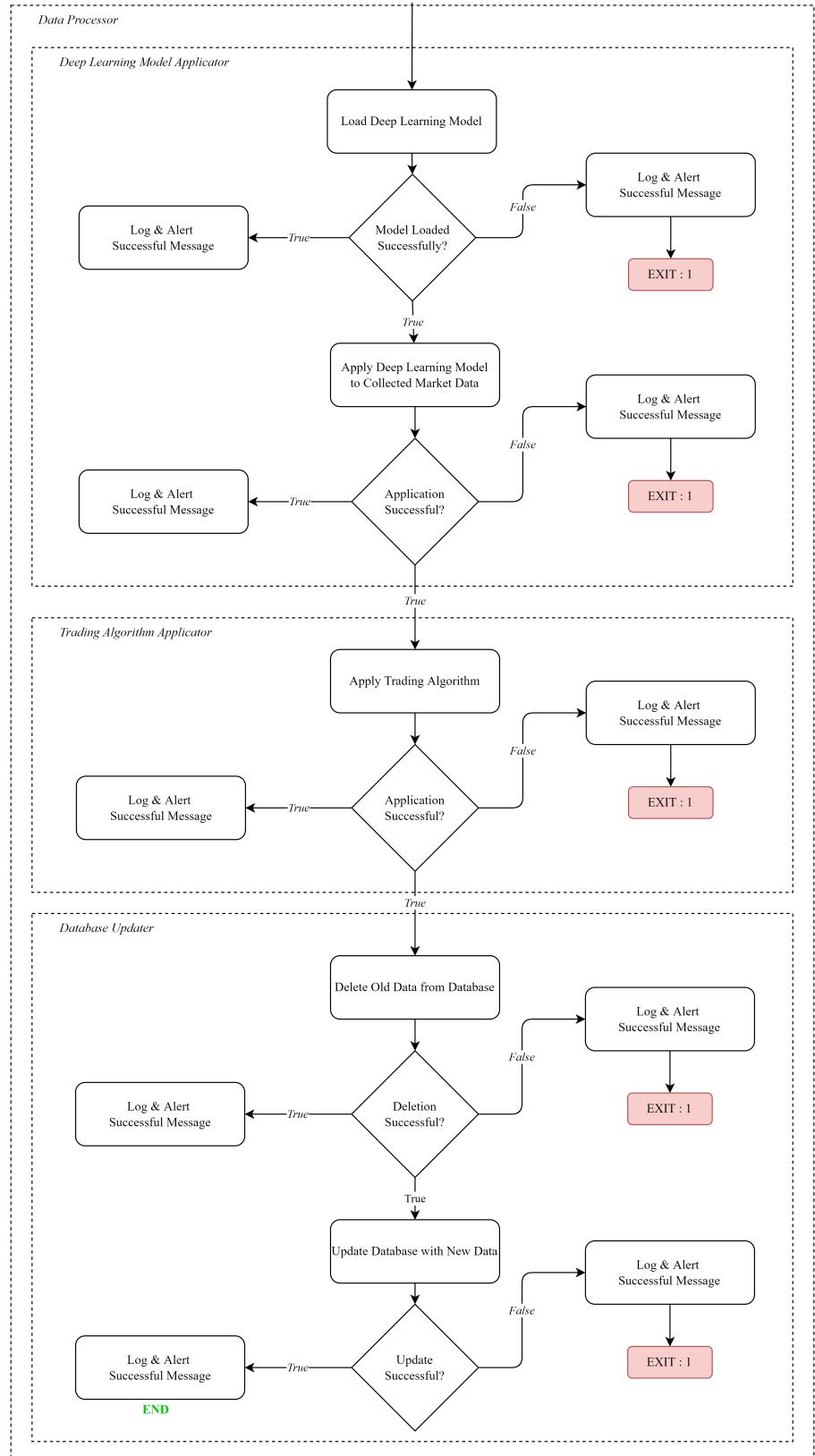


Figure 1.5: Overview of the Process Flow Diagram for Data Processor

## Database Updater

This process flow will purge the old content of the database, and once successful it will update the database with the new documents created from the previous process.

The flow of processes discussed above can be observed in Figure 1.8.

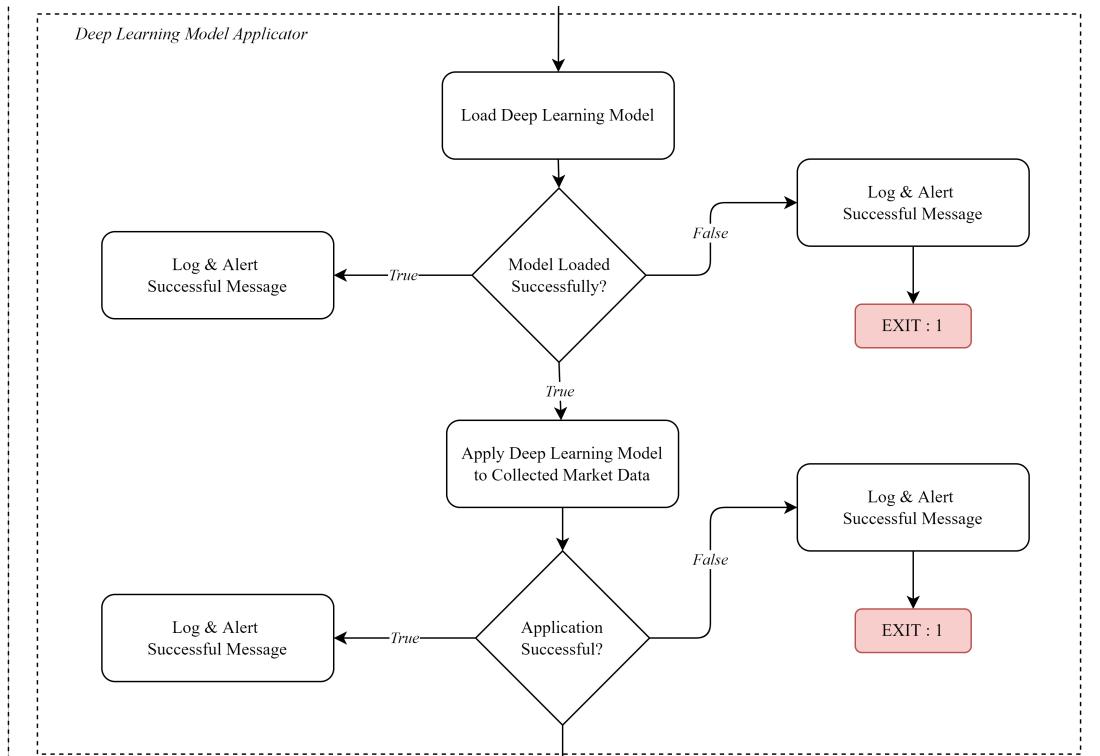


Figure 1.6: Overview of the Process Flow Diagram for the Deep Learning Model Applicator

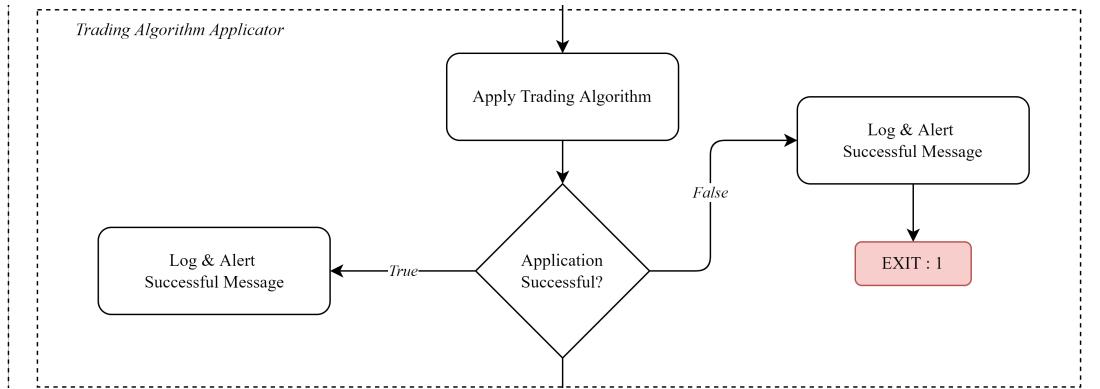


Figure 1.7: Overview of the Process Flow Diagram for the Trading Algorithm Applicator

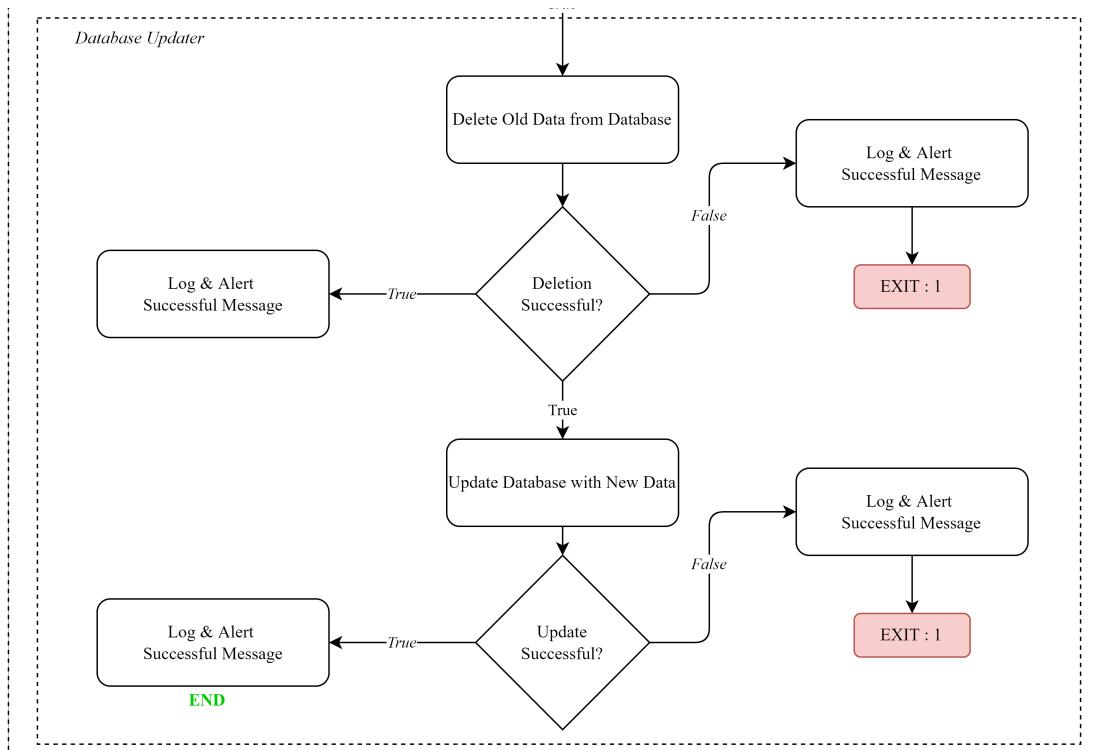


Figure 1.8: Overview of the Process Flow Diagram for the Database Updater

### 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 whole process is shown in a context diagram of the system, labelled as process 0, which can be seen in the provided in Figure 1.9.

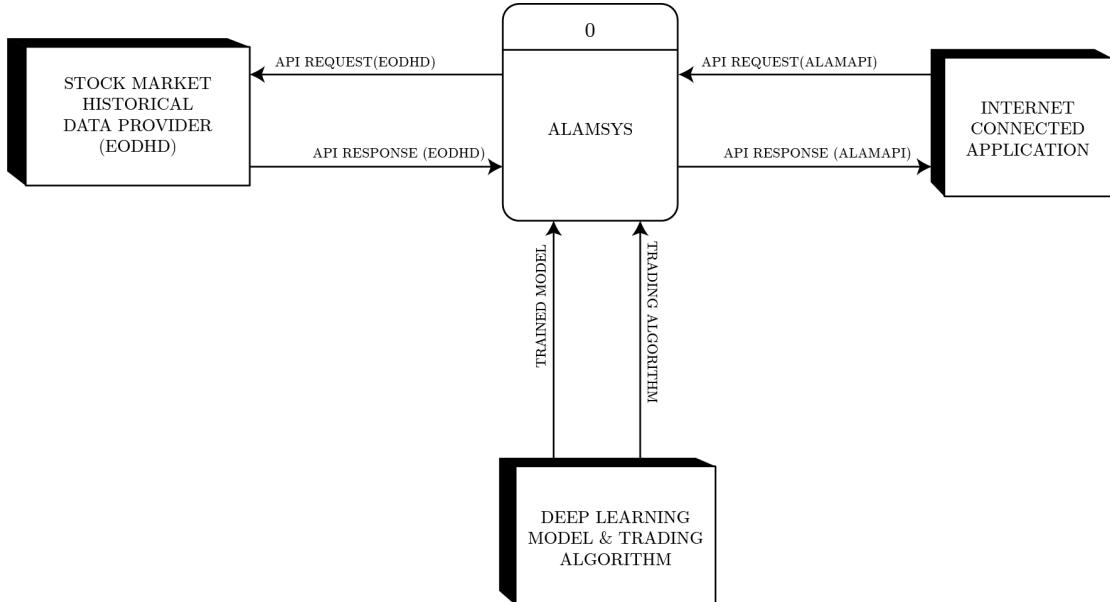


Figure 1.9: Context Diagram of the alamSYS

The above figure shows the root process (process 0), which is the underlying system of the alamSYS which is connected to four external entities: (1) Scheduler, which will be provided by CRON; (2) Stock Market Historical Data Provider, which will be EODHD; (3) Machine Learning Model, which will be developed along side the development of the system; and (4) Third-Party Application, which will also be developed in the conduct of this special problem as the test application for accessing, testing, and showcase of the features of alamSYS.

Moreover, all the necessary data flow lines can also be observed: (1) Scheduled task, which is the committed task on schedule as indicated in the CRON application; (2) API Request: Stock Market Historical Data, which is the request information passed by the root process to the historical market data provider; (3) API Response: Stock Market Historical Data, which is the data passed by the historical market data provider to the process after accepting its request; (4) Trained Model, this is the object class from the Machine Learning model that will be developed and used by the system; (5) Stock Market Historical Data, as the name suggests this is the historical market data which will also be used to train and improve the machine learning model; (6) API Request: Stocks to Buy, which is the data passed from the third-party application to the root process to request for which stocks are in the Buy document of the database; (7) API Response: Stocks to Buy, upon the request of the third-party application, the root process will process the request to the API and sends back the list of stocks to buy to the requester; (8) API Request: Stocks to Sell, which is the data passed from the third-party application to the root process to request for which stocks are in the Sell document of the database; (9) API Response: Stocks to Sell, upon the request of the third-party application, the root process will process the request to the API and sends back the list of stocks to sell to the requester; (10) API Request: Stock Info, which is the data passed from the third-party application to the root process to request for the general information about a particular stock, this will be further discussed in the Object-Document Mapped (ODM) diagram; and (11) API Response: Stock Info, upon the request of the third-party application, the root process will process the request to the API and sends back the information of the stock based on what was requested.

## DFD of Diagram 0

To better understand how each data going in-and-out of the root process, is being processed, it is essential that we look inside the inner workings of the root process, which is shown in the DFD of Diagram 0, as provided in Figure 1.10.

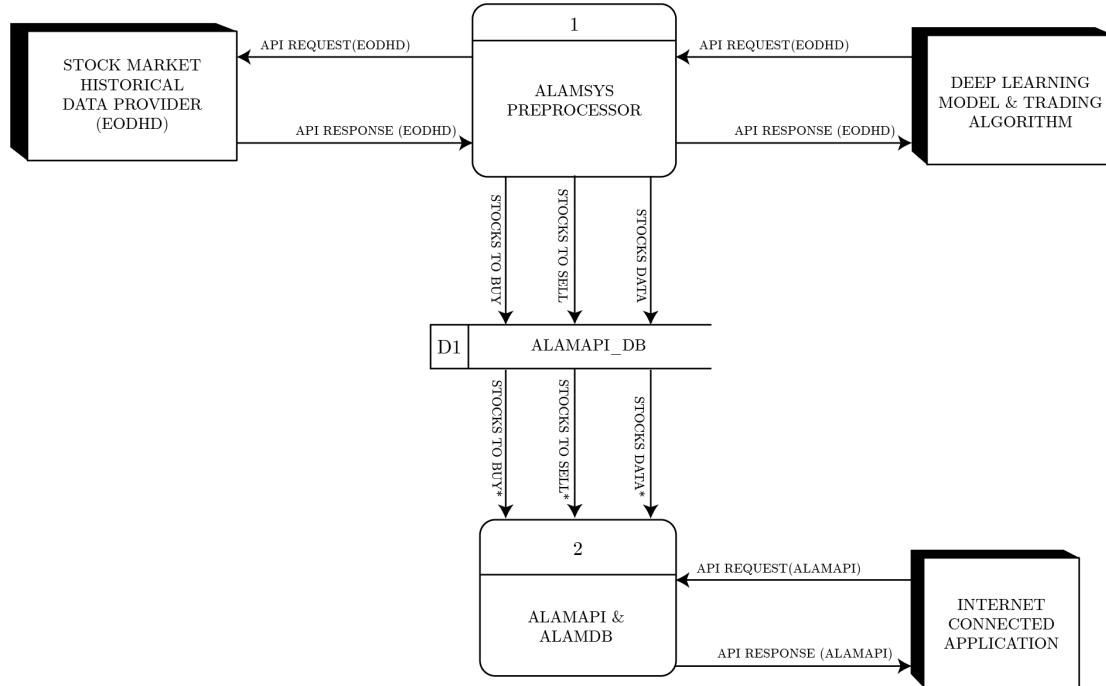


Figure 1.10: DFD of Diagram 0

From the figure above, the root process, has two main processes: (1) alamSYS Preprocessor, which is the system's data processing unit; and (2) alamAPI module, which processes the API requests and responses by utilizing the database of the system.

## DFD of Diagram 1

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

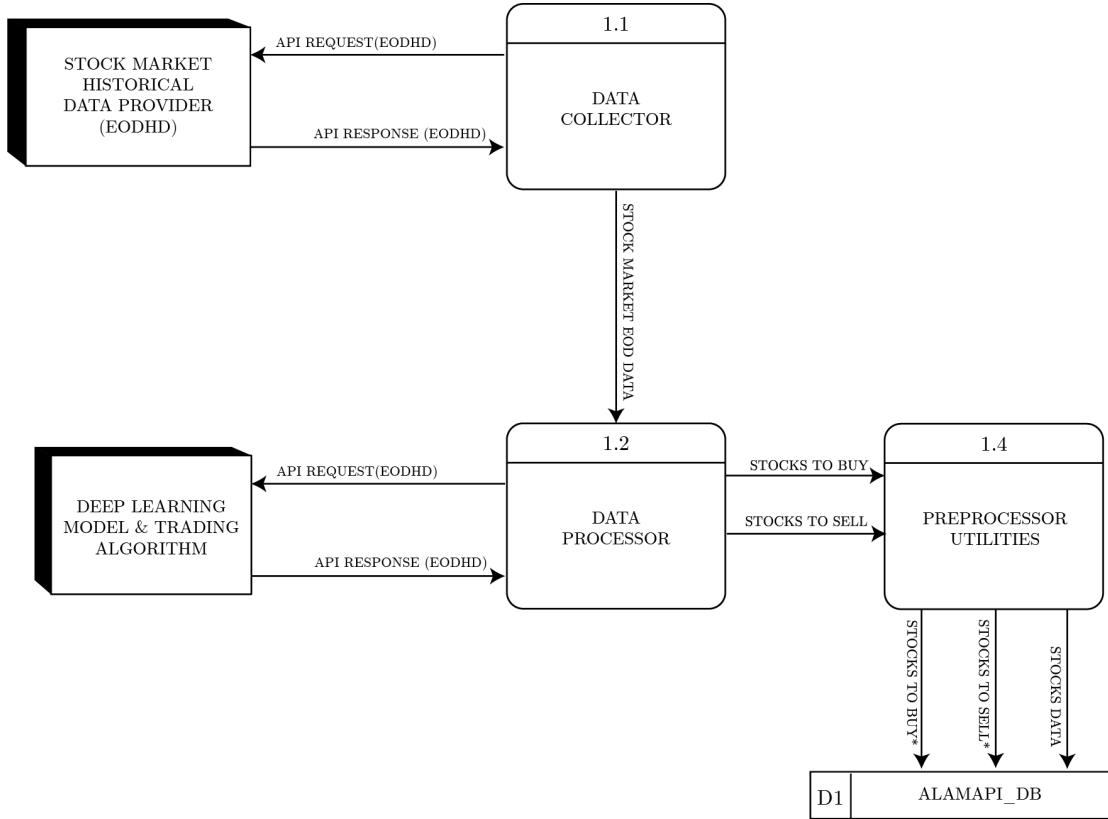


Figure 1.11: DFD of Diagram 1

From the figure shown above, it can be observed that Process 1 is composed of 4 internal processes, namely, (1) Data Collector, which is the main process responsible for collecting the historical market data; (2) Pre-Database Processor, which is the processes that the collected data goes into before being sent to the system's database, the internal processes of this process will be further discuss in the succeeding part of this section; (3) Machine Learning Model Processor, which is the training module or process for the machine learning model that is used to externally train the machine learning model that will be used by the system in the pre-database processor; and (4) Preprocessor Utilities, this will be the processes the process any system utilities such as the database actions, database models, and stores temporary data and system variables.

## DFD of Diagram 1.2

This shows the processes inside the process 1.2: Pre-database Processor.

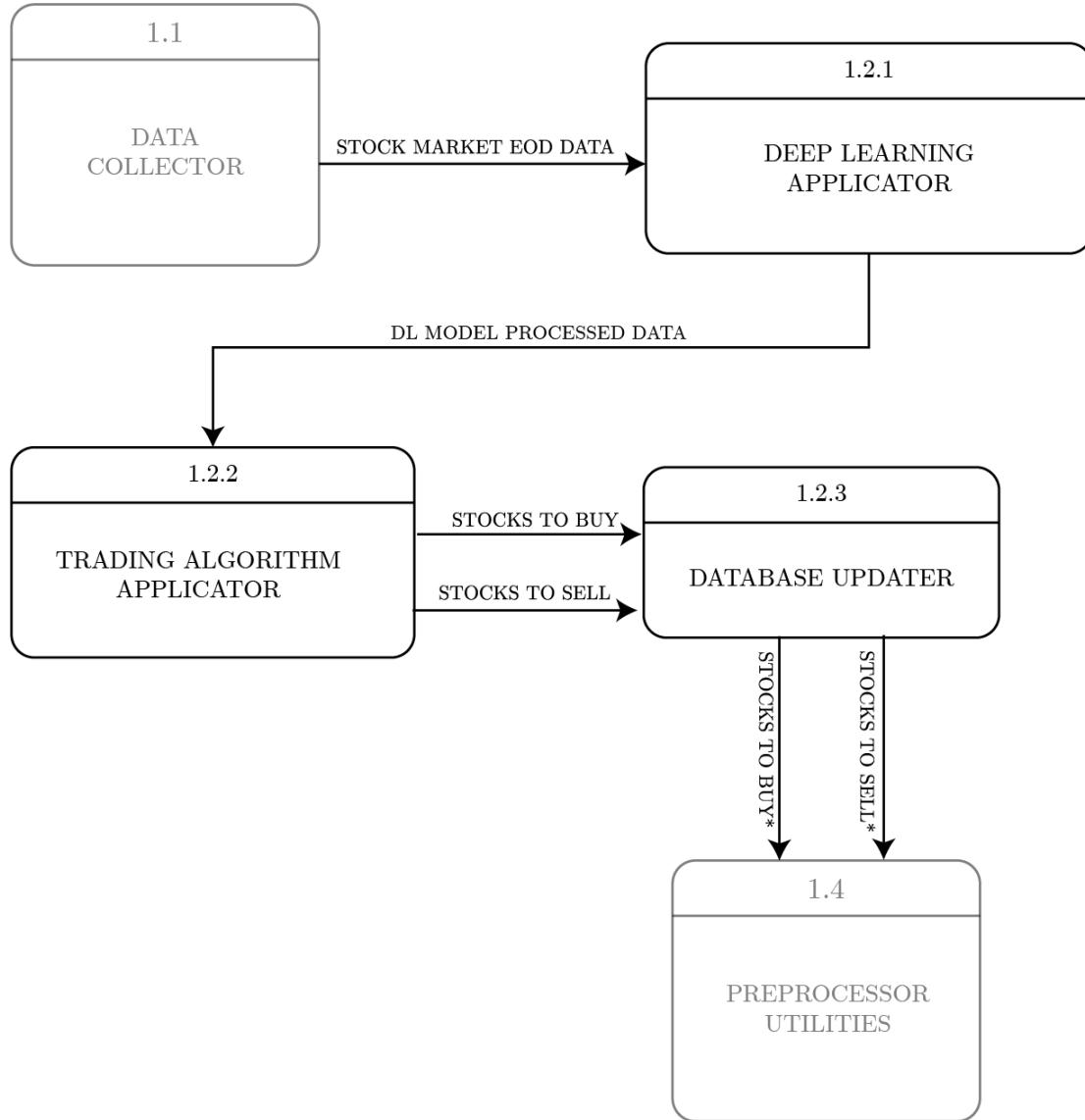


Figure 1.12: DFD of Diagram 1.2

As previously discussed, the pre-database processor consists of processes inside that processes the data before it will be eventually sent to the database of the system. Wherein the process and data flow is shown in the figure above. Namely: (1) Machine Learning Applicator, which applies the trained machine learning model

to the collected data; and (2) Database Updater, which processes the document outputs from the Machine Learning Applicator process, to be used in the database of the system.

## DFD of Diagram 2

The final diagram will show the inner processes of the Process 2: alamAPI module.

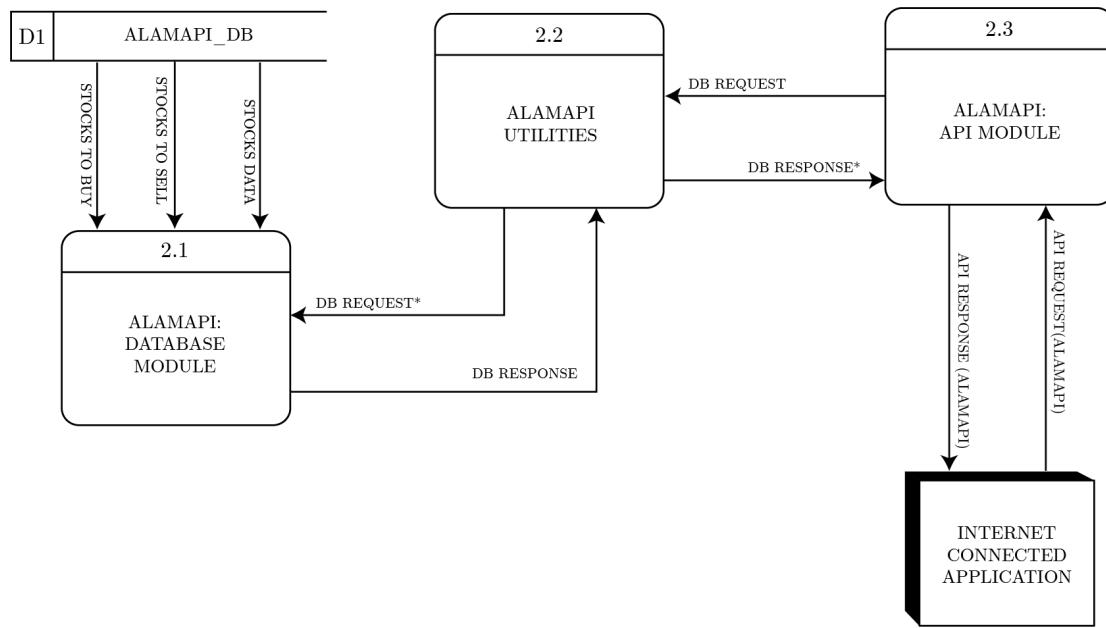


Figure 1.13: DFD 2: Data-Flow Diagram for the alamSYS

The figure above shows two internal processes of the Process 2, namely, (1) alamAPI: Database Module, which the database model that is used to process and connect MongoDB to the Python program; and (2) alamAPI: API Module, which is composed of the API endpoints that processes the requests and response to and from the system, respectively.

#### 1.2.4 Object Document Mapper (ODM) Diagram

Since the database that will be developed for the system will be a non-relational, hence an Object Document Mapper (ODM) diagram is shown in this section rather than an Entity Relationship Diagram (ERD).

The ODM diagram is shown in Figure 1.14:

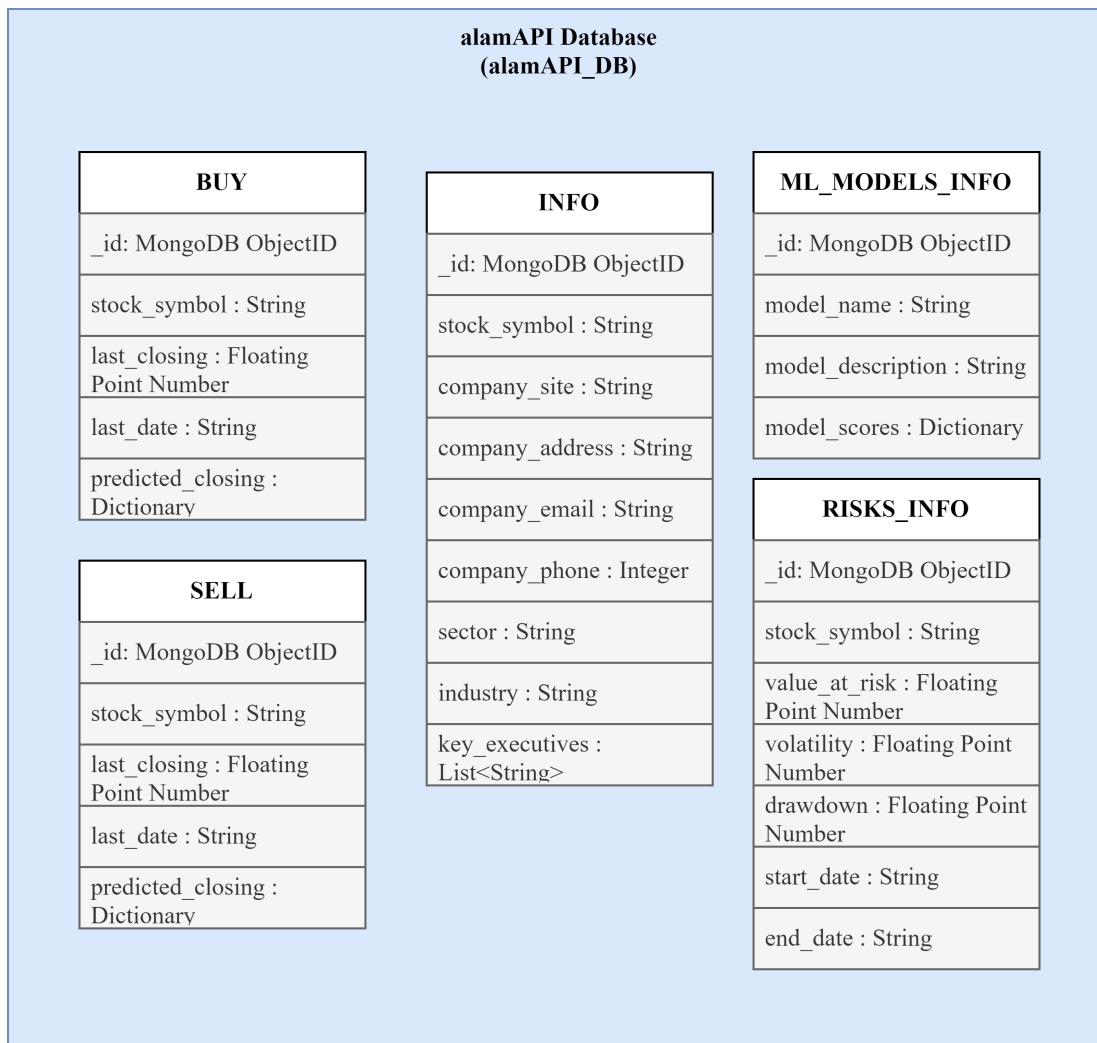


Figure 1.14: Object-Data-Model for the alamSYS

As shown from the Figure 1.14, the "alamAPI\_DB" will be the collection

name of the non-relational database of the system. Wherein it will be composed of three documents with the list items following this convention: “item name”: “item type”. Note that each document are their own separate entities, hence the database is called non-relational, as the documents are not in any way related to each other.

The three documents are as follows:

- (a) Buy – this document will contain all the stocks that the machine learning model predicted and classified as a stock to buy. The diagram shown in Figure 1.14 also tells the path in which this document can be accessed, that is:

$$\text{MongoDBInstance} \rightarrow \text{alamAPI\_DB} \rightarrow \text{Buy}$$

Wherein, information regarding the stocks can be accessed using the stock\_symbol, since the \_id is a private id.

- (b) Sell – this document will contain all the stock that the machine learning model predicted and classifies as a stock to sell. The diagram shown in 1.14 also tells the path in which this document can be accessed, that is:

$$\text{MongoDBInstance} \rightarrow \text{alamAPI\_DB} \rightarrow \text{Sell}$$

Wherein, information regarding the stocks can be accessed using the stock\_symbol, since the \_id is a private id.

- (c) Info – this document will contain the general and relevant information about a stock, or the general company information. The diagram shown in 1.14 also tells the path in which this document can be accessed, that is:

$$\text{MongoDBInstance} \rightarrow \text{alamAPI\_DB} \rightarrow \text{Info}$$

Wherein, information regarding the stocks can be accessed using the stock\_symbol, since the \_id is a private id.

- (d) Machine Learning (ML) Models Info – this document will contain the details about the Machine Learning Models deployed in the system. The diagram

shown in 1.14 also tells the path in which this document can be accessed, that is:

$$\text{MongoDBInstance} \rightarrow \text{alamAPI\_DB} \rightarrow \text{ML\_Models\_Info}$$

Wherein, information regarding the ML models can be accessed using the model\_name.

### 1.2.5 Machine Learning Model Diagram

In this section, the process on how the machine learning model will be developed is shown in Figure 1.15. Wherein, the process overview is based on the Fine-Tuned Support Vector Regression Model for Stock Predictions by Dash and Dash (2016).

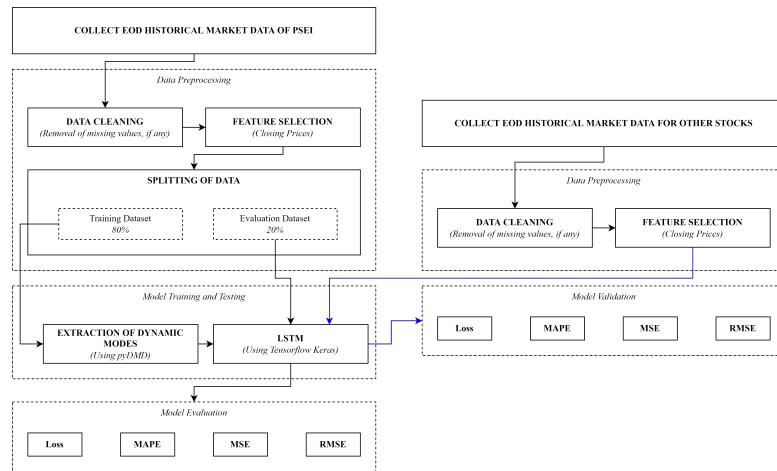


Figure 1.15: Machine Learning Model for the alamSYS

### 1.2.6 Docker-Compose Layer Diagram

In this section, the different layers of the docker-compose containers are shown based on the way it will be used in the deployment of the system. Moreover, Figure 1.16 based on the provided diagram in the Docker documentation, regarding containers. Note that in the diagram shown below, the lowest level is the "Server Infrastructure" and the highest level are the three Docker instances.

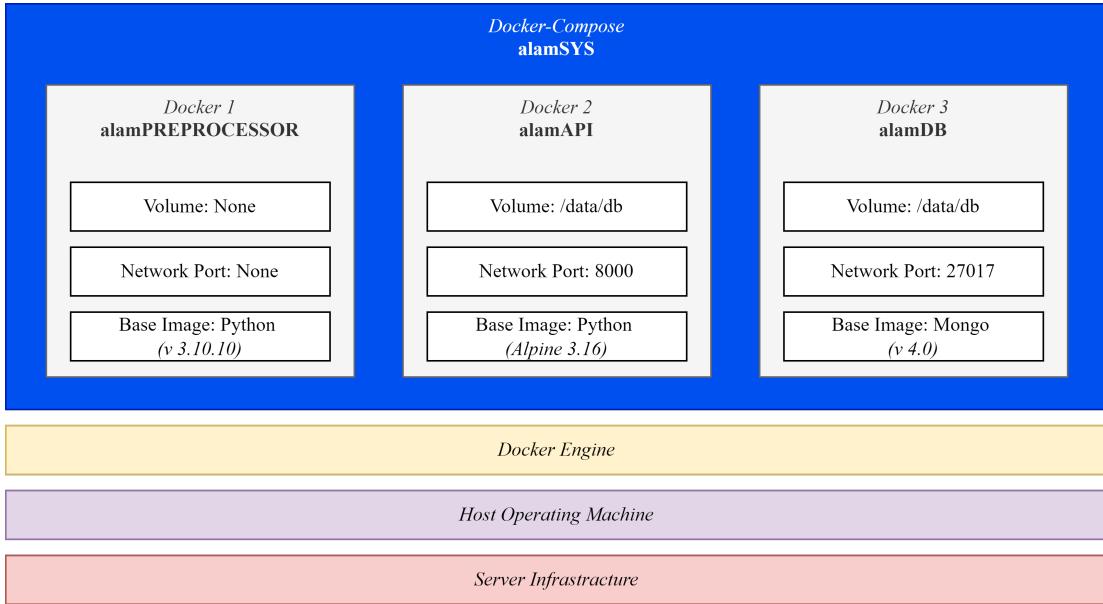


Figure 1.16: Docker-Compose Layer Diagram for the alamSYS

## 1.3 Hardware Requirements

In this section, the hardware requirements will be discussed.

### 1.3.1 For the Development of the alamSYS, and Training and Testing of the Machine Learning Model

To develop the alamSYS and the underlying system that it utilizes, the project developer would be needing a laptop with at least the following minimum requirements:

- (a) A desktop class 4-core CPU running on 2GHz (minimum).
- (b) 16 GB of Random Access Memory (RAM). This is to ensure that multiple instances of programs can run efficiently in the system.
- (c) An up-to-date GPU with CUDA cores, this will be used specifically for faster training and testing of the machine learning model. Although if this is not available then a more powerful CPU maybe required.

### **1.3.2 For the Development of the Test Application and System Testing**

- (a) A device that can connect to a network such as a smartphone (preferably an Android smartphone: as the test application that will be developed will run on Android devices). Moreover, the device specifications does not matter, as long as it can run a browser or the developed Android test application.

## **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 1.1: Summary of Sprints and Target Activities

Sprint Number	Target Activities	Allotted Time <sup>1</sup>
1	<p><b>Main Activity:</b> System Planning and Evaluation</p> <p><b>Sub-Activities:</b></p> <ul style="list-style-type: none"> <li>• Topic Proposal</li> <li>• Drafting of Chapters 1 to 3 for the Special Problem Proposal</li> <li>• System Architecture and User Requirement Analysis</li> </ul>	<p><b>12 Weeks</b></p> <p>Start: September 15, 2022</p> <p>End: December 9, 2022</p>

**Table 1.1 continued from previous page**

Sprint Number	Target Activities	Allotted Time <sup>1</sup>
2	<p><b>Main Activity:</b> System Prototyping</p> <p><b>Sub-Activities:</b></p> <ul style="list-style-type: none"> <li>• Build the different component of the alamSYS as indicated in the top-level overview diagram of the system, the following prototype will be developed:           <ul style="list-style-type: none"> <li>[1.] API endpoints</li> <li>[2.] Database</li> <li>[3.] Preprocessor</li> </ul> </li> <li>• Testing of the build prototype. This also include creating unit test cases for each component.</li> <li>• Initial Documentations, this will be done inside the GitHub repository.</li> </ul>	<p><b>12 Weeks</b></p> <p>Start: September 30, 2022</p> <p>End: April 3, 2023</p>

**Table 1.1 continued from previous page**

Sprint Number	Target Activities	Allotted Time <sup>1</sup>
3	<p><b>Main Activity:</b> Machine Learning Model Training, Testing, and Evaluation</p> <p><b>Sub-Activities:</b></p> <ul style="list-style-type: none"> <li>• 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 than the 200-day only historical data. Whereas the last date on the market data should be January 13, 2023.</li> <li>• Development of the Machine Learning Model. This includes data standardization, data splitting, and data training.</li> <li>• Machine Learning model testing and evaluation.</li> <li>• Revision of Chapter 1-3, in preparation for the final paper submission.</li> </ul>	<p><b>10 Weeks</b></p> <p>Start: January 15, 2023</p> <p>End: March 30, 2023</p>

**Table 1.1 continued from previous page**

Sprint Number	Target Activities	Allotted Time <sup>1</sup>
4	<p><b>Main Activity:</b> Integration of Machine Learning Model to the alamSYS and Additional Data Collection</p> <p><b>Sub-Activities:</b></p> <ul style="list-style-type: none"> <li>• Testing and Evaluation of alamSYS with the integration of the Machine Learning Model.</li> <li>• 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</li> <li>• Drafting of Chapter 4 and 5</li> </ul>	<p><b>6 Weeks</b></p> <p>Start: March 31, 2023</p> <p>End: May 12, 2023</p>

**Table 1.1 continued from previous page**

Sprint Number	Target Activities	Allotted Time <sup>1</sup>
5 <sup>2</sup>	<p><b>Main Activity:</b> System Documentation</p> <p><b>Sub-Activities:</b></p> <ul style="list-style-type: none"> <li>• Updating and Finalization of Documentations included in the GitHub Repository.</li> <li>• Writing of the results, discussions, conclusions, and recommendations for Chapter 4 – 5</li> <li>• Special problem paper revisions</li> <li>• Start the development of the test application (for showcasing of the system features)</li> </ul>	<p><b>6 Weeks</b></p> <p>Start: April 14, 2023</p> <p>End: May 26, 2023</p>

**Table 1.1 continued from previous page**

Sprint Number	Target Activities	Allotted Time <sup>1</sup>
6 <sup>2</sup>	<p><b>Main Activity:</b> Preparation for Final Defense and System Presentation</p> <p><b>Sub-Activities:</b></p> <ul style="list-style-type: none"> <li>• Finalization of the mobile-based test application</li> <li>• Revisions and Finalization of the special problem paper.</li> <li>• Creation of presentation slide deck for the presentation of the special problem.</li> </ul>	<p><b>3 Weeks</b></p> <p>Start: May 27, 2023</p> <p>End: June 17, 2023</p>

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 1.1, 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 1.17 shows the schedule of activities for Sprint 1. Wherein, it will start on September 15, 2022, and end on December 9, 2022.

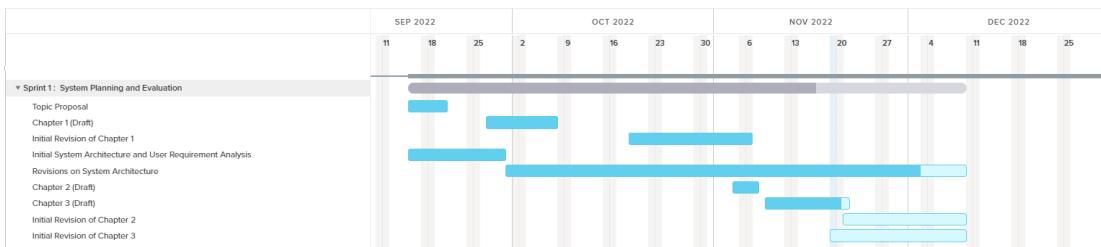


Figure 1.17: Gantt Chart for Sprint 1

### 1.5.2 Gantt Chart for Sprint 2

Figure 1.18 shows the schedule of activities for Sprint 2. Which will start on September 30, 2022, and end on January 5, 2023.

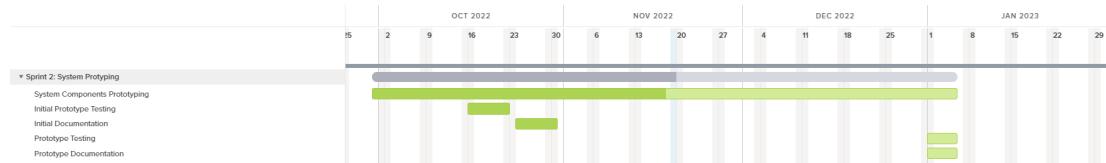


Figure 1.18: Gantt Chart for Sprint 2

### 1.5.3 Gantt Chart for Sprint 3

Figure 1.19 shows the schedule of activities for Sprint 3. Wherein, it will start on January 15, 2023, and end on March 30, 2023.

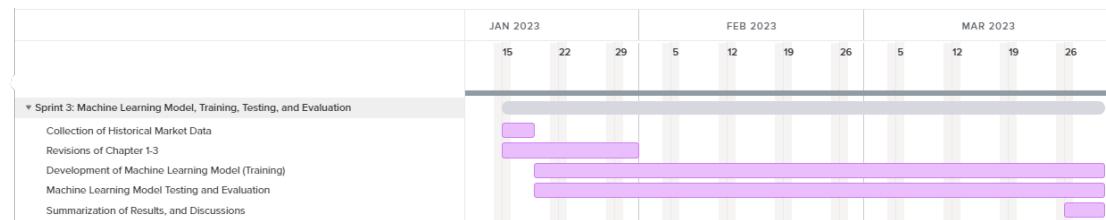


Figure 1.19: Gantt Chart for Sprint 3

### 1.5.4 Gantt Chart for Sprint 4

Figure 1.20 shows the schedule of activities for Sprint 4. Which will run from March 31, 2023, until May 12, 2023.

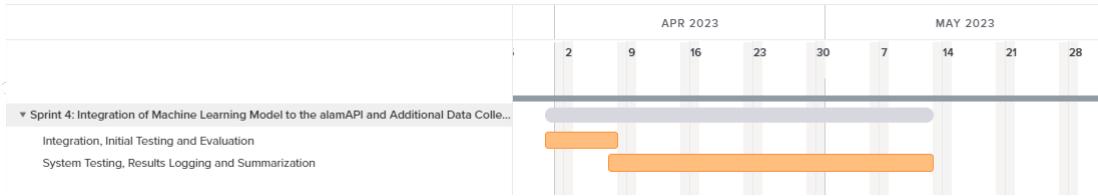


Figure 1.20: Gantt Chart for Sprint 4

### 1.5.5 Gantt Chart for Sprint 5

Figure 1.21 shows the schedule of activities for Sprint 5. Which will be from April 14, 2023, to May 12, 2023.

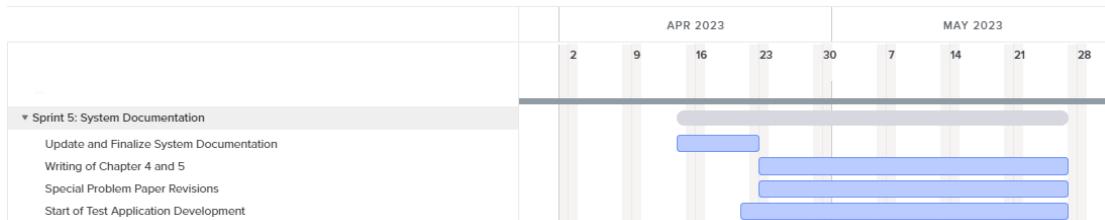


Figure 1.21: Gantt Chart for Sprint 5

### 1.5.6 Gantt Chart for Sprint 6

Figure 1.22 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.

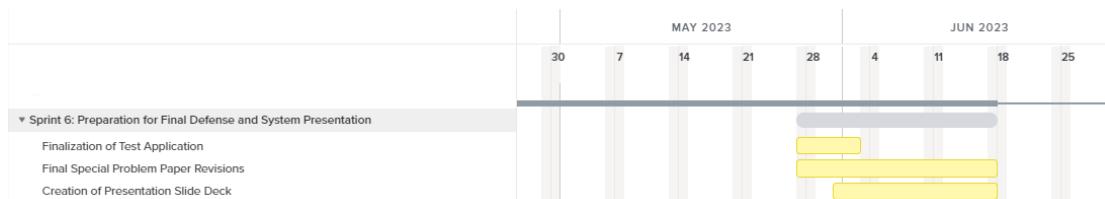


Figure 1.22: Gantt Chart for Sprint 6

### 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 1.23.

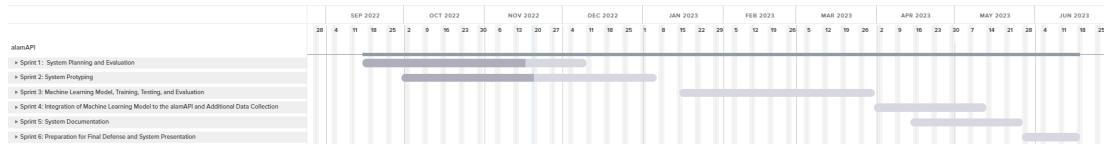


Figure 1.23: Full Gantt Chart

# References

- 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/>
- MongoEngine. (n.d.). *Mongoengine documentation.* Retrieved November 13, 2022, from <https://docs.mongoengine.org/>
- Tiangolo. (n.d.). *Fastapi documentation.* Retrieved November 13, 2022, from <https://fastapi.tiangolo.com/>
- 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/>