

**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

Abstract

Abstract here

Keywords: Keyword 1, keyword 2, keyword 3, keyword 4, etc.

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 1 |
| 1.1 | Background and Rationale | 1 |
| 1.1.1 | The Philippine Stock Exchange (PSE) | 2 |
| 1.1.2 | Economic Relevance and Benefits of Stock Market Investment | 2 |
| 1.1.3 | Benefits of Investing for the Individual | 3 |
| 1.1.4 | Utilization of Machine Learning in Stock Market Trading . | 3 |
| 1.2 | Statement of the Problem | 4 |
| 1.3 | Significance of the Study | 6 |
| 1.4 | Objectives | 9 |
| 1.5 | Scope and Limitations | 10 |
| 2 | Review of Related Works and Literature | 11 |
| 2.1 | Integration of Machine Learning based Trading Algorithms | 11 |
| 2.1.1 | Comparison of Machine Learning and Deep Learning Mod- els in Stock Market Predictions | 12 |

| | | |
|----------|---|-----------|
| 2.2 | Utilization of Dynamic Mode Decomposition (DMD) on the Financial Markets | 13 |
| 2.2.1 | Chronological Utilization of DMD in the Financial Markets | 14 |
| 2.3 | Synthesis | 15 |
| 3 | Materials and Methods | 17 |
| 3.1 | Development Tools and Software Requirements | 17 |
| 3.1.1 | Development Tools | 18 |
| 3.1.2 | Software Requirements | 18 |
| 3.2 | System Diagrams | 21 |
| 3.2.1 | Top-Level Overview Diagram of the alamSYS and Its Inter- actions to External Systems | 21 |
| 3.2.2 | Process Flow Diagram | 23 |
| 3.2.3 | Data-Flow Diagram (DFD) | 31 |
| 3.2.4 | Object Document Mapper (ODM) Diagram | 38 |
| 3.2.5 | Deep Learning Model Diagram | 43 |
| 3.2.6 | ALMACD Development Diagram | 48 |
| 3.2.7 | Docker-Compose Layer Diagram | 50 |
| 3.3 | Hardware Specifications | 51 |
| 3.3.1 | For the Development of the alamSYS, Deep Learning Model, and Mobile-Based Test Application | 51 |
| 3.3.2 | For Deployed System Testing | 52 |

| | | |
|----------|--|-----------|
| 3.3.3 | For the Test Application | 52 |
| 3.4 | Methodology | 52 |
| 3.4.1 | Software Development Process | 53 |
| 3.4.2 | Procedures | 60 |
| 3.5 | Gantt Chart | 63 |
| 3.5.1 | Gantt Chart for Sprint 1 | 64 |
| 3.5.2 | Gantt Chart for Sprint 2 | 64 |
| 3.5.3 | Gantt Chart for Sprint 3 | 64 |
| 3.5.4 | Gantt Chart for Sprint 4 | 65 |
| 3.5.5 | Gantt Chart for Sprint 5 | 65 |
| 3.5.6 | Gantt Chart for Sprint 6 | 66 |
| 3.5.7 | Full Gantt Chart | 66 |
| 4 | Results and Discussions | 67 |
| 4.1 | alamSYS Documentation | 67 |
| 4.1.1 | Documentation for alamPREPROCESSOR | 68 |
| 4.1.2 | Documentation for alamAPI and alamDB | 72 |
| 4.1.3 | Documentation for alamAPP | 85 |
| 4.1.4 | Build and Deployment Guide | 87 |
| 4.2 | alamSYS System Tests Results and Discussions | 91 |
| 4.3 | DMD-LSTM Model Results and Discussions | 98 |
| 4.4 | ALMACD Results and Discussions | 117 |

| | |
|---|------------|
| 4.5 Results and Discussions for the Real World Application of alamSYS | 120 |
| 5 Conclusions and Future Work | 123 |
| 5.1 Summary of Findings | 123 |
| 5.2 The Implications of alamSYS to the Society | 126 |
| 5.3 Recommendations on Future Works | 127 |
| References | 130 |
| A Source Code Repository | 138 |
| B Raw Data | 139 |
| C Project Management Documentation | 140 |
| D Glossary of Terms | 141 |
| E Acknowledgements | 142 |
| F Author's Contact Information | 143 |

List of Figures

| | | |
|------|---|----|
| 3.1 | Top-Level Overview of the alamSYS and Interactions with External Applications/Systems | 22 |
| 3.2 | Full Overview of the Process Flow Diagram for the alamSYS | 24 |
| 3.3 | Overview of the Process Flow Diagram for the Scheduler | 25 |
| 3.4 | Overview of the Process Flow Diagram for the Data Collector | 26 |
| 3.5 | Overview of the Process Flow Diagram for Data Processor | 28 |
| 3.6 | Overview of the Process Flow Diagram for the Deep Learning Model Applicator | 29 |
| 3.7 | Overview of the Process Flow Diagram for the Trading Algorithm Applicator | 30 |
| 3.8 | Overview of the Process Flow Diagram for the Database Updater | 31 |
| 3.9 | Context Diagram of the alamSYS | 32 |
| 3.10 | DFD of Diagram 0 | 34 |
| 3.11 | DFD of Diagram 1 | 35 |
| 3.12 | DFD of Diagram 1.2 | 36 |
| 3.13 | DFD 2: Data-Flow Diagram for the alamSYS | 37 |
| 3.14 | Object Document Mapper for the alamSYS Database | 39 |

| | | |
|------|---|----|
| 3.15 | Sample Buy Collection from the alamSYS Database | 40 |
| 3.16 | Sample Sell Collection from the alamSYS Database | 41 |
| 3.17 | Sample Info Collection from the alamSYS Database | 42 |
| 3.18 | Sample ML Models Info Collection from the alamSYS Database . | 42 |
| 3.19 | Sample Stock Risks Profile Collection from the alamSYS Database | 43 |
| 3.20 | DMD-LSTM Model Development Methodology for alamSYS . . . | 44 |
| 3.21 | ALMACD Development Methodology for alamSYS | 48 |
| 3.22 | Docker-Compose Layer Diagram for the alamSYS | 50 |
| 3.23 | Modified Agile Development Methodology | 53 |
| 3.24 | Gantt Chart for Sprint 1 | 64 |
| 3.25 | Gantt Chart for Sprint 2 | 64 |
| 3.26 | Gantt Chart for Sprint 3 | 65 |
| 3.27 | Gantt Chart for Sprint 4 | 65 |
| 3.28 | Gantt Chart for Sprint 5 | 65 |
| 3.29 | Gantt Chart for Sprint 6 | 66 |
| 3.30 | Full Gantt Chart | 66 |
| 4.1 | alamPREPROCESSOR Docker Container Directory Tree | 71 |
| 4.2 | alamAPI Web-based API Documentation Using Swagger | 73 |
| 4.3 | alamAPI Web-based API Documentation Using Redocs | 74 |
| 4.4 | Sample API Endpoint: Home Request | 75 |
| 4.5 | Sample API Endpoint: Buy (all) Request | 76 |

| | | |
|------|---|-----|
| 4.6 | Sample API Endpoint: Sell (all) Request | 77 |
| 4.7 | Sample API Endpoint: Stocks Info (all) Request | 78 |
| 4.8 | Sample API Endpoint: ML Model Info (all) Request | 79 |
| 4.9 | Sample API Endpoint: Stocks Risks Profile (all) Request | 80 |
| 4.10 | alamAPI End User UML Use Case Diagram | 83 |
| 4.11 | alamAPI Docker Container Directory Tree | 84 |
| 4.12 | MongoDB Compass Database Management of alamDB | 85 |
| 4.13 | alamSYS Required Folder Structure | 89 |
| 4.14 | Deployment Load CPU Utilization of alamAPI and alamDB Over Time | 97 |
| 4.15 | Deployment Load Memory Utilization of alamAPI and alamDB Over Time | 97 |
| 4.16 | Comparison of MAPE Scores for DMD-LSTM Model Training Across Different Window Sizes | 99 |
| 4.17 | Actual vs Predicted Prices on AC for 100 days | 102 |
| 4.18 | Actual vs Predicted Prices for ALI over 100 days | 102 |
| 4.19 | Actual vs Predicted Prices for AP over 100 days | 103 |
| 4.20 | Actual vs Predicted Prices for BDO over 100 days | 104 |
| 4.21 | Actual vs Predicted Prices for BLOOM over 100 days | 104 |
| 4.22 | Actual vs Predicted Prices for FGEN over 100 days | 105 |
| 4.23 | Actual vs Predicted Prices for GLO over 100 days | 106 |
| 4.24 | Actual vs Predicted Prices for ICT over 100 days | 106 |

| | |
|--|-----|
| 4.25 Actual vs Predicted Prices on JGS for 100 days | 107 |
| 4.26 Actual vs Predicted Prices on LTG for 100 days | 108 |
| 4.27 Actual vs Predicted Prices on MEG for 100 days | 108 |
| 4.28 Actual vs Predicted Prices on MER for 100 days | 109 |
| 4.29 Actual vs Predicted Prices on MPI for 100 days | 110 |
| 4.30 Actual vs Predicted Prices on PGOLD for 100 days | 110 |
| 4.31 Actual vs Predicted Prices on PSEI for 100 days | 111 |
| 4.32 Actual vs Predicted Prices on RLC for 100 days | 112 |
| 4.33 Actual vs Predicted Prices on RRHI for 100 days | 112 |
| 4.34 Actual vs Predicted Prices on SMC for 100 days | 113 |
| 4.35 Actual vs Predicted Prices on TEL for 100 days | 114 |
| 4.36 Actual vs Predicted Prices on URC for 100 days | 114 |
| 4.37 MAPE Scores for 1 to 10 (Days) Successive Predictions | 117 |
| 4.38 PSEI Stock Market Price Trend from March 24 to April 12, 2023 | 121 |
| 4.39 Comparison Between the Day-to-day Gains of alamSYS and PSEI | 122 |

List of Tables

| | | |
|------|---|-----|
| 3.1 | Collected Market Data Details | 44 |
| 3.2 | Summary of Sprints and Activities | 54 |
| 4.1 | Summary of Risk Profile for Each Stock in the alamSYS | 81 |
| 4.2 | Idle System Average Resource Usage Statistics | 91 |
| 4.3 | Internal Load Average Resource Usage Statistics | 93 |
| 4.4 | Deployment Load Test Results (Buy Requests) | 95 |
| 4.5 | Deployment Load Test Results (Sell Requests) | 95 |
| 4.6 | CPU and Memory Utilization Statistics of alamAPI and alamDB Under Deployment Load Testing | 96 |
| 4.7 | DMD-LSTM Training Error Metrics Scores for Different Window Sizes | 98 |
| 4.8 | Baseline LSTM Training Error Metrics Scores for Different Window Sizes | 100 |
| 4.9 | DMD-LSTM Cross-Validation Error Metrics Scores | 101 |
| 4.10 | DMD-LSTM Successive Predictions | 116 |
| 4.11 | Optimal Alma Parameters Validation Results | 118 |

Chapter 1

Introduction

1.1 Background and Rationale

The stock market is a type of market that allows businesses to raise capital by selling stock shares to investors. These shares represent a portion of the company's ownership and entitle the holder to a portion of the company's profits as well as voting rights. The stock exchange also serves as a marketplace for investors to buy and sell these shares, allowing for the efficient trading of company ownership. The stock market plays an important role in the growth and development of the economy by allowing companies to raise capital and investors to buy and sell shares (Chen, 2022; The Economic Times, n.d.).

The stock market, contrary to popular belief, is not a form of gambling. It necessitates a significant amount of analytical thinking and risk management, and the returns are determined by supply and demand for a specific stock, rather than false promises or assurances. In other words, rather than being a scam or a gamble, the stock market is a legitimate platform for investing and generating returns (Schwab-Pomerantz, 2021; Adams, 2022; Summers, 2022).

1.1.1 The Philippine Stock Exchange (PSE)

The Philippine Stock Exchange (PSE), Inc. is the country's official stock exchange market. It is a non-stock company founded in 1992 that manages and operates the country's stock market. Individuals who are registered with the PSE can participate in market exchanges (The Philippine Stock Exchange, Inc., n.d.-a).

Furthermore, the Philippine Stock Exchange Index (PSEI) is the main index of the PSE. (PSEI). The PSEI is a market capitalization-weighted price index composed of the PSE's 30 largest and most actively traded companies. These businesses have been pre-selected based on strict criteria such as liquidity and market capitalization. The PSEI is frequently used as a proxy for the overall performance of the Philippine stock market. (Bangko Sentral ng Pilipinas, n.d.) These PSEI companies are often referred to as blue-chip companies because they are typically large, well-established companies with a track record of strong financial performance. In total the PSE has 286 companies listed as of October 2022, offering investors a diverse range of investment opportunities (Fayed, 2022; The Philippine Stock Exchange, Inc., n.d.-b).

1.1.2 Economic Relevance and Benefits of Stock Market Investment

The stock market is widely acknowledged to play an important role in economic growth because it allocates and provides capital to businesses, which drives economic activity and growth. This is evident from the fact that stock market performance is frequently correlated with the gross domestic product (GDP) of the country. (Trade Brains, 2022; Hall, 2022; Bae & Kang, 2017) Furthermore, historical stock price trends can provide insight into broader economic movements (Campbell, 2021).

In a study conducted by Balaba (2017), they discovered that the stock market

has a positive impact on the Philippines' economy. The study's findings showed that as the stock market rose, the unemployment rate fell. This is because the performance of the stock market leads to job creation, which in turn leads to economic growth. This, in turn, drives economic growth. This relationship was observed in the Philippines from 2007 to 2017.

1.1.3 Benefits of Investing for the Individual

Individuals in the Philippines can trade shares of publicly traded companies on the Philippine Stock Exchange. Investing in the stock market can provide several advantages to an individual, including:

- (a) Protects an individual's money from inflation: Inflation in the Philippines was 6.9% as of September 2022 (Trading Economics, n.d.), while savings account deposit interest rates are only 1-3% annually, (Bureau of the Treasury Bangko Sentral ng Pilipinas, n.d.). This means that savings in deposit banks may not keep pace with inflation, potentially reducing an individual's purchasing power over time. (Royal Bank of Canada Direct Investing Inc., n.d.; EdwardJones, n.d.).
- (b) Capital growth opportunities: Investing in the stock market can provide individuals with the opportunity for significant capital growth without the need for direct investment involvement in business operations. This may benefit individuals. Students and working professionals, for example, can increase their capital while remaining focused on their studies or careers. (U.S. Securities and Exchange Commission, n.d.).

1.1.4 Utilization of Machine Learning in Stock Market Trading

In recent years, there has been a surge in interest in the use of machine learning. Learning techniques for predicting stock market movement in the short and long term. As a result, numerous studies and practical applications investigating the use of machine learning in stock market prediction have been conducted.

These efforts aim to improve prediction accuracy and assist investors in making informed decisions (Kumbure, Lohrmann, Luukka, & Porras, 2022; Strader, Rozycki, Root, & Huang, 2020; Soni, Tewari, & Krishnan, 2022; Rea, 2020; Y. Guo, 2022). In this regard, one of the common techniques used is the Long Short-Term Memory (LSTM). LSTM is a deep learning model that is widely used to forecast the stock market. A study by Budiharto (2021) found that LSTM was effective in predicting the Indonesian stock market with 95% accuracy using a short-term data. Which suggests that LSTM can be a useful tool for making short-term stock market forecasts.

The use of Dynamic Mode Decomposition (DMD) for predicting stock market price trends has recently gained traction in the financial industry. DMD is a mathematical method for identifying patterns and trends in large data sets, such as stock market data. It is possible to make more accurate predictions about future stock price movements by applying DMD to stock market data. This can help investors make more informed investment decisions and potentially generate higher returns. However, a study by Lu and Tartakovsky (2020) found that DMD is faster than Proper Orthogonal Decomposition, but it is less accurate.

Other studies have shown that DMD can be effectively applied to the Turkish and Indian stock markets to predict market price trends (Savaş, 2017; Kuttichira, Gopalakrishnan, Menon, & Soman, 2017). These studies show that DMD is simple to implement and can be used as a useful enhancer for making stock market predictions.

1.2 Statement of the Problem

Economic growth in the Philippines is expected to slow in the coming years as a result of the global pandemic, high inflation, and low employment rates (Alegado, Lopez, & Calonzo, 2022; Canto & Romano, 2022; Reuters, 2022).

The lack of free and publicly available stock market predictive systems or tools currently creates a gap in the information available to the public when compared to large private individuals or institutions. These large institutions have the resources to spend a significant amount of money on stock market research, giving them a significant advantage in the investing market. Where, the public is disadvantaged by this lack of access to the same information (Kim, 2022).

Furthermore, the lack of publicly available stock market prediction tools can lead to individuals, particularly first-time investors, making unwise investment decisions, resulting in significant losses and discouragement from investing in the stock market. This is a significant issue because the number of local investors in the Philippine Stock Exchange is already quite small, accounting for only about 1% of the total population. In addition, there has been a significant decline in foreign investment in the Philippines in recent years (Business World, 2022), leading to a corresponding decline in investment volume. As suggested in the study of Balaba (2017), this is expected to have a negative multiplier effect on the country's economic development in the future.

As a result, the creation of a publicly available, simple-to-use, and accurate stock market price trend prediction system could aid in closing the information gap and leveling the playing field for individual investors. This system could help to increase transparency and fairness in the stock market by providing the public with timely and reliable information, resulting in more informed and confident investing decisions and, ultimately, a more stable and prosperous market. Furthermore, such a system could help to increase individual investor participation in the market, resulting in a more diverse and stable market overall. (Statista Research Department, 2022; Commission on Population and Development, 2021).

However, despite the clear and functional benefits of investing in the stock market, many Filipinos remain hesitant to do so for the following reasons:

- (a) The difficulties that come with learning the fundamentals of effective stock

investing.

- (b) The time-consuming nature of technical and fundamental analysis, especially for students and working people on a tight schedule; and
- (c) The increased financial risk associated with stock market volatility, as well as the potential for emotional decision-making to jeopardize investments.

These factors (*along with other external and internal factors not listed above*) contribute to a lack of confidence and understanding among potential investors, making it difficult for them to take advantage of the opportunities offered by the stock market.

As such the development of this system, aims to address the following:

- (a) The lack of free and publicly available stock market prediction systems or tools.
- (b) The time and resources required to study complex traditional market analysis tools, such as fundamental and technical analysis.
- (c) The potential for inaccurate market decisions leading to significant investment losses; and
- (d) The hesitancy of the Filipino public to begin investing in the Philippine stock market.

1.3 Significance of the Study

The significance of this particular problem lies in the developed system to greatly benefit the stock market, individual investors, and the economy as a whole. Contributions of the system to data-driven investing, financial protection and management, and economic development could provide a valuable resource for investors while also promoting financial stability and growth. Furthermore, the creation of publicly accessible data-driven investing tools or systems may enable

more Filipinos to participate in the market and take control of their own financial future. Overall, this special problem has the potential to have a significant impact on the Philippine stock market and economy.

Specifically, this study is significant for the following reasons:

- (a) The development of the alamSYS aims to provide the following benefits to the Filipino people:
 1. Access to simplified yet accurate information – The proposed system could provide Filipino investors with fast, accurate, and relevant information necessary for effective decision making in the stock market. Using a deep learning model such as LSTM, the system could provide users with the two most important pieces of information: which stocks to buy, and which stocks to sell. This simplified investing model could help investors to make informed decisions and navigate the stock market with confidence.
 2. Provide an application interface to facilitate data-driven market decisions – The system could provide users with an intuitive and user-friendly application interface to facilitate data-driven investment decisions, particularly during times when the market is unpredictable or experiencing a downturn. Whereas traditional market analysis tools may not be sufficient to navigate these challenging conditions, the system's forecasting model could provide investors with the insights and guidance they need to make informed and wise decisions. Which would help to promote confidence and stability in the market, even during times of uncertainty.
 3. A platform for accessible stock market investment – The system aims to provide all investors, regardless of their investment knowledge, educational attainment, and societal status, with a platform for participating in the stock market. By offering a simplified yet accurate model for investment decision making, the system could empower users to make informed decisions and invest with confidence. This could help to democratize access to the stock market and promote financial inclusion for all Filipinos.

(b) The development of the alamSYS, aims to provide the following benefits to the future developers or researchers:

1. Extension of functionality to other financial markets – The system can be easily adapted or expanded to address related problems in other financial markets, such as investing in government bonds or personal finance management. This flexibility and versatility could make the system a valuable tool for a wide range of investment and financial management scenarios.
2. Testing of new trading algorithms and other machine learning models – The system provides a platform for introducing and testing new data-driven trading algorithms and machine learning models. This could allow future researchers and developers to continually improve the system and keep it at the forefront of data-driven investing technology.
3. Development of a graphical user interface – To further improve the public accessibility of the system, a user-friendly graphical user interface can be developed as a web or mobile application. This could make the system easy to use and intuitive for all users, regardless of their technical expertise.

(c) The development of the alamSYS could help to stimulate economic recovery and development in the country by increasing the number of local investors. As discussed in previous sections, the benefits of the system could encourage more people to invest in the stock market, leading to a multiplier effect that could benefit the economy in several ways. For instance, the increased participation in the market could lead to the creation of jobs and a lowering of unemployment rates. Additionally, the influx of capital into the market could drive fast developments and innovations in various industries. Finally, the increased consumer spending that results from successful investing, stimulates economic growth as well. Overall, the development of the alamSYS could have a positive and far-reaching impact on the economy of the Philippines.

1.4 Objectives

This special problem aims to develop a system that makes investing easier, more publicly available, data-driven, and more approachable by minimizing both the time required for stock price trend analysis, and potential financial risk by using DMD-LSTM and integrate Arnaud Legoux Moving Average Convergence-Divergence (ALMACD) as a trading algorithm. More specifically, it aims to do following:

- (a) Develop a system called alamSYS that can collect and process stock market data in order to provide comprehensive stock position suggestions using Dynamic Mode - Long Short-Term Memory with Arnaud Legoux Moving Average Convergence-Divergence integration. Specifically, this was done by creating the following:
 1. Develop a Data Preprocessor. Which includes a Data Collector Module (DCM), which collects the end-of-day historical data of a stock from Mondays to Fridays. The data collected is then processed by the Data Processor Module (DPM), which applies the deep learning model and integrate the trading algorithm to the data. Finally, the processed data is given to the Database Updater Module (DUM).
 2. Develop a RESTful API, referred to as alamAPI, using Python's FastAPI for handling the API endpoints.Specifically, the following API endpoints were developed:
 - 2.1 **Home** – This API endpoint outputs a welcome message. Which should inform the user that they have successfully connected to the alamAPI.
 - 2.2 **Stocks to Buy** – This API endpoint outputs a list of suggested stocks to buy based from the current market price and the predicted price up-trend.
 - 2.3 **Stocks to Sell** – This API endpoint outputs a list of suggested stocks to sell based from the current market price and the predicted price down-trend.
 - 2.4 **Stocks Info** - This API endpoint outputs a list of stocks included in the alamSYS and their corresponding information.

- 2.5 **ML Model Info** - This API endpoint outputs a list of the Machine Learning Models used in the alamSYS and their corresponding information.
- 2.6 **Stocks Risks Info** - This API endpoint outputs a list of the stocks included in the alamSYS and their corresponding risks values based on value at risk (%), volatility (%), and drawdown (%).
3. Develop a database called alamDB, that stores the results provided by the DPM and DUM, and other essential data such as stock information, deep learning model information, and stock risks information about the stock market that is needed to be provided.
 - (b) Develop a Stock Market Price Trend Forecasting Deep Learning Models by utilizing the dynamic modes in DMD as an additional input parameter to an LSTM model. Afterwards, integrate the forecasting with ALMACD as a trading algorithm and basis for entry and exit positions.
 - (c) Finally, develop a mobile-based test application, which from hereon maybe referred to as: alamAPP, to showcase the main functionalities of the developed RESTful API. Specifically which stocks to buy and to sell for a given period of time.

1.5 Scope and Limitations

This study was limited only within the companies listed in the Philippine Stock Exchange. Specifically, 20 high volume trade stocks from the year 2021 to 2022 were selected, which are as follows: (1) MEG, (2) JGS, (3) BDO, (4) FGEN, (5) ICT, (6) ALI, (7) SMC, (8) TEL, (9) GLO, (10) BLOOM, (11) RLC, (12) MER, (13) AC, (14) PGOLD, (15) LTG, (16) MPI, (17) AP, (18) RRHI, (19) URC, and (20) PSE Index will be included in the system, instead of the total 286 listed under the Philippine Stock Exchange.

Chapter 2

Review of Related Works and Literature

One of the challenges facing investors in the Philippine Stock Market is the limited availability of resources and tools for making market decisions. In contrast, other countries have begun implementing machine learning techniques for stock market prediction and analysis, which allows for more accurate decision-making and reduces the risk of poor investment outcomes. As a result, these countries are likely to experience better returns on their investments.

In this literature review, the following general topics are reviewed, discussed, and synthesized: (a) Integration of Machine Learning based Trading Algorithms; and (b) Utilization of Dynamic Mode Decomposition on the Financial markets.

2.1 Integration of Machine Learning based Trading Algorithms

Stock market analysis is crucial for effective risk management. This involves using various methods, such as technical and fundamental analysis, to make informed decisions for investors and traders. In recent years, the growth of com-

puting power and resources has led to the increasing use of machine learning techniques for stock market prediction and analysis. These advances help companies better predict upcoming market trends and make more informed decisions.

The integration of machine learning algorithms in the stock market is growing, as investors and traders increasingly rely on fast and accurate market information to reduce potential risks and make better decisions. These algorithms allow for more efficient analysis of market data, leading to more informed decisions and improved investment outcomes (Obthong, Tantisantiwong, Teamwatthanachai, & Wills, 2020).

2.1.1 Comparison of Machine Learning and Deep Learning Models in Stock Market Predictions

To have a better grasp in the accuracy of the different models used in algorithmic trading it is essential that different models are compared against each other.

Combination of Computational Efficient Functional Link Artificial Neural Network (CEFLANN) and Traditional Technical Analysis

This hybrid model combines a classification-based model: CEFLANN and the traditional technical analysis to create a stock trading framework Dash and Dash (2016), which the results show a profit of 24.29%.

Deep Long Short-Term Neural Network (LSTM) with Embedded Layer

In one of the models developed by Pang, Zhou, Wang, Lin, and Chang (2020), it shows that by adding an embedded layer to the LSTM it yields to a stock market price prediction accuracy of 57.2%. However, its accuracy dips to 52.4% when the model is applied to individual stocks.

LSTM with Automatic Encoder

As part of the second model developed by Pang et al. (2020), this model shows a slightly inaccurate stock market prediction, by only having a measured accuracy of 56.9%. However, compared to the first model developed by the group this is 0.1% more effective for individual stocks.

Optimal Deep Learning (ODL)

In the study conducted by Agrawal, Khan, and Shukla (2019) they have created a stock price prediction model using an Optimal Deep Learning (ODL) which combine the concepts of Correlation-Tensor and an Optimal LSTM algorithm. Whereas their results show a mean and highest accuracy of the model as 59.24% and 65.64%.

NMC-BERT-LSTM-DQN-X Algorithm

More recently, a team have applied a combination of three models for forecasting the market trends. Namely, (1) Non-stationary Markov Chain (NMC), (2) Bidirectional Encoder Representations from Transformers (BERT), (3) Long Short-Term Memory (LSTM). Wherein their model shows an accuracy of 61.77%. Furthermore, the team also mentioned that the model produces 29.25% annual return on investment, with a maximum losses rating of -8.29% (Liu, Yan, Guo, & Guo, 2022).

2.2 Utilization of Dynamic Mode Decomposition (DMD) on the Financial Markets

Dynamic Mode Decomposition (DMD) as an emerging data-driven technique which allows spatial-temporal pattern recognition from a complex set of data and was first introduced in the field of fluid mechanics by (SCHMID, 2010).

2.2.1 Chronological Utilization of DMD in the Financial Markets

In (2015) Mann and Kutz proved that DMD can be used as data-driven analytics on the financial market data. Wherein, DMD allows a predictive assessment of the market dynamics, which helps in the capitalization of stock market strategies and decisions to be applied.

Utilization of DMD for Determining the Cyclic Behavior in the Stock Market (2016)

By utilizing the reproducible Koopman modes it made it possible to have extracted four cyclic variations (also reproducible modes) in the stock market, which were previously unknown and have persisted since the 1870s' global economic crisis (Hua, Roy, McCauley, & Gunaratne, 2016; Williamson, 2015).

Utilization of DMD as part of an Algorithmic Trading Strategies for the Turkish Stock Market (2015 and 2017)

The study of Mann and Kutz (2015) in the utilization of DMD for financial stock market prediction has become the foundation of the study by Savaş (2017) on the algorithmic trading strategies with Dynamic Mode Decomposition for the Turkish Stock Market. Wherein, based on their results they found out that the timing of DMD analysis was not significantly accurate, as such they have used a simple moving average with genetic algorithm to improve the market timing of DMD, which prevents 80% of the false trade signals.

Furthermore, this also shows that DMD is an effective alpha model that is easy to implement and use for any algorithmic trading strategy, and the addition of technical analysis tools can further improve its capabilities, especially on the predictive temporal side of the data.

Utilization of DMD-based Trading Strategy in the Chinese Stock Market (2016)

In the study by Cui and Long (2016), they have found that DMD was able to capture the dynamic patterns of the Chinese Stock Market, especially in a sideway trending market.

Their study also shows that the predictive ability of DMD can effectively model the behavior of the Chinese Stock Market, even if there are no clear trends that can be observed.

Utilization of Adaptive Elastic DMD to Improve Momentum Strategies (2021)

A study by Uchiyama and Nakagawa (2021), using Adaptive Elastic Dynamic Mode Decomposition (AEDMD) shows that they were able to estimate the market trend, and were able to demonstrate that the approach is better than existing momentum strategy which are only based on simple past trends.

2.3 Synthesis

Fast and accurate market information is an essential tool for stock market participants. In recent years, the development of machine learning models for the financial markets, such as stocks, has proven to be increasingly effective in predicting future stock prices and trends. The use of Dynamic Mode Decomposition (DMD) in the stock market has also been shown to be effective in predicting stock price trends. The simplicity and elegance of the Koopman Decomposition Operator make it an ideal basis for the development of a Stock Market Price Trend Forecasting System.

Hence, these studies are crucial for the development of the alamSYS. As it can

provide investors with fast and accurate information about which stocks are likely to go up or down, allowing them to make more informed decisions about buying or selling those stocks.

In addition to the potential benefits for investors and traders, the implementation of machine learning techniques in the stock market can also help improve market efficiency and reduce the risk of market manipulation. By providing a more accurate and comprehensive view of market trends, these techniques can help ensure that prices reflect the true value of stocks and other assets, leading to more stable and fair market conditions.

Chapter 3

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

3.1 Development Tools and Software Requirements

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

3.1.1 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).

3.1.2 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.

3.2 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.

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

Figure 3.1 shows the top-level overview of the alamSYS and its interactions to any third-party or external applications.

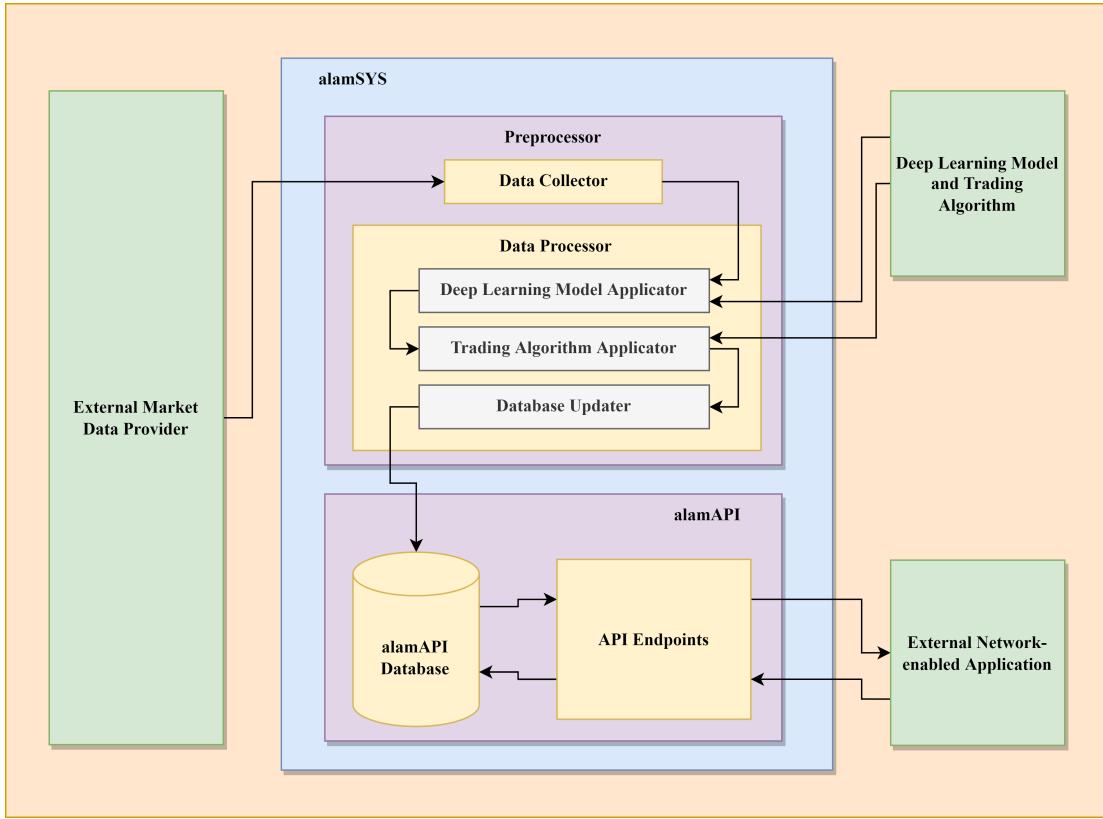


Figure 3.1: Top-Level Overview of the alamSYS and Interactions with External Applications/Systems

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.

3.2.2 Process Flow Diagram

The diagram shown in Figure 3.2 the different processes that the system will undergo once it has been deployed in the server.

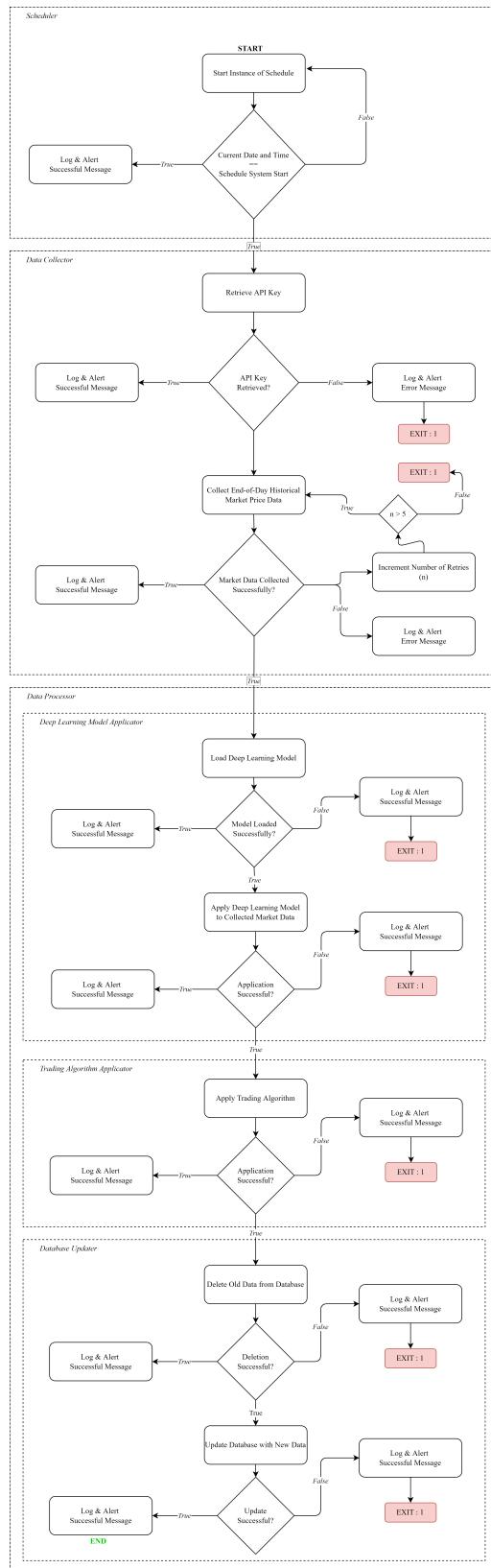


Figure 3.2: Full Overview of the Process Flow Diagram for the alamSYS

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 3.3.

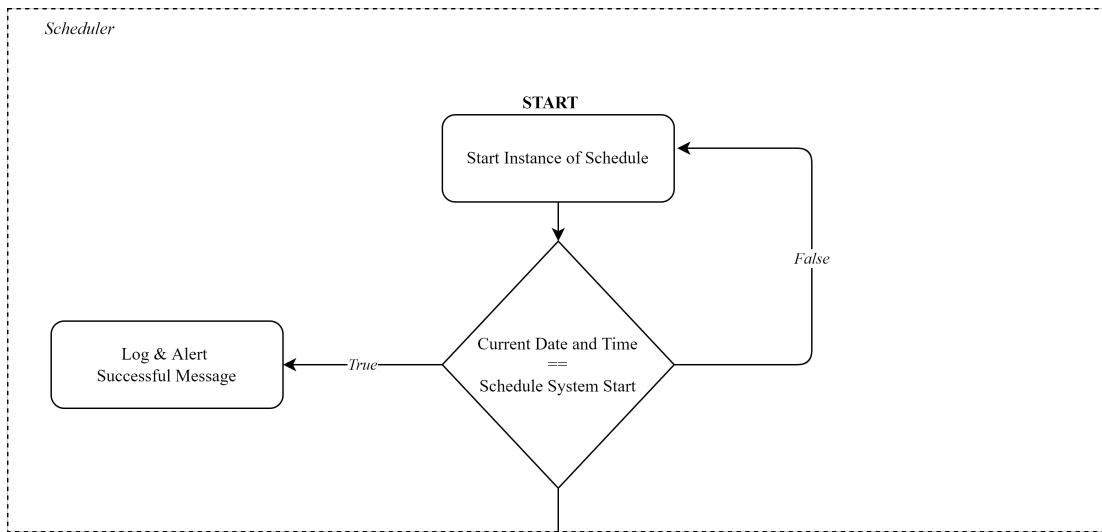


Figure 3.3: Overview of the Process Flow Diagram for the Scheduler

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 3.4 depicts this, as well as the entire data collection process.

The flow of processes discussed above can be observed in Figure 3.4.

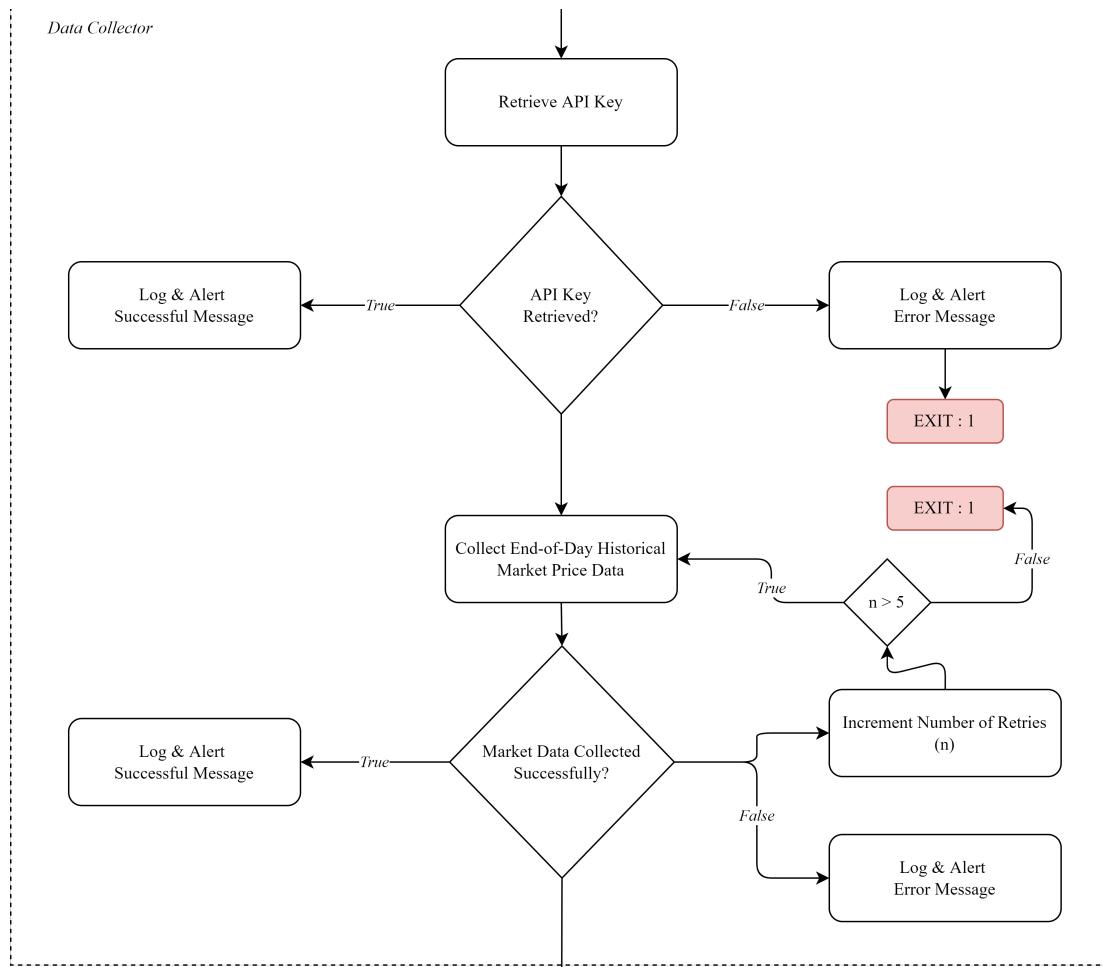


Figure 3.4: Overview of the Process Flow Diagram for the Data Collector

Data Processor

Data Processor is a process that is divided into three subprocesses, as shown in Figure 3.5.

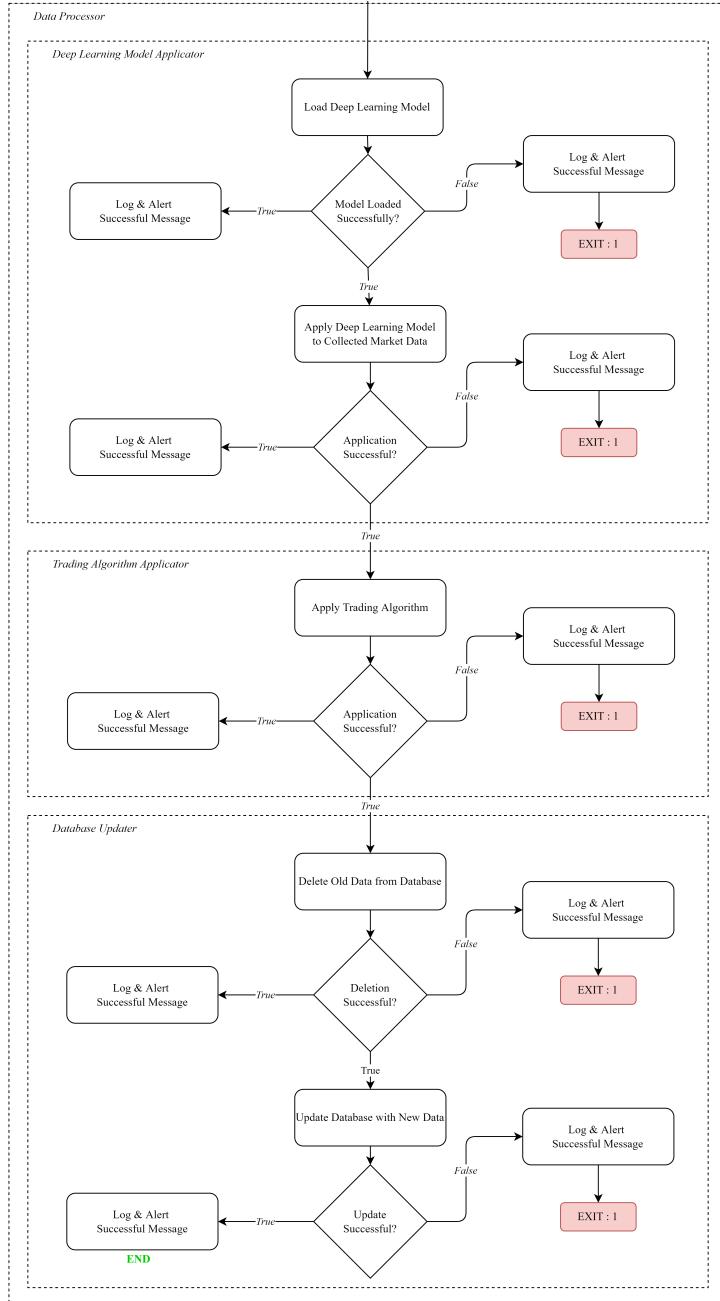


Figure 3.5: Overview of the Process Flow Diagram for Data Processor

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 3.6.

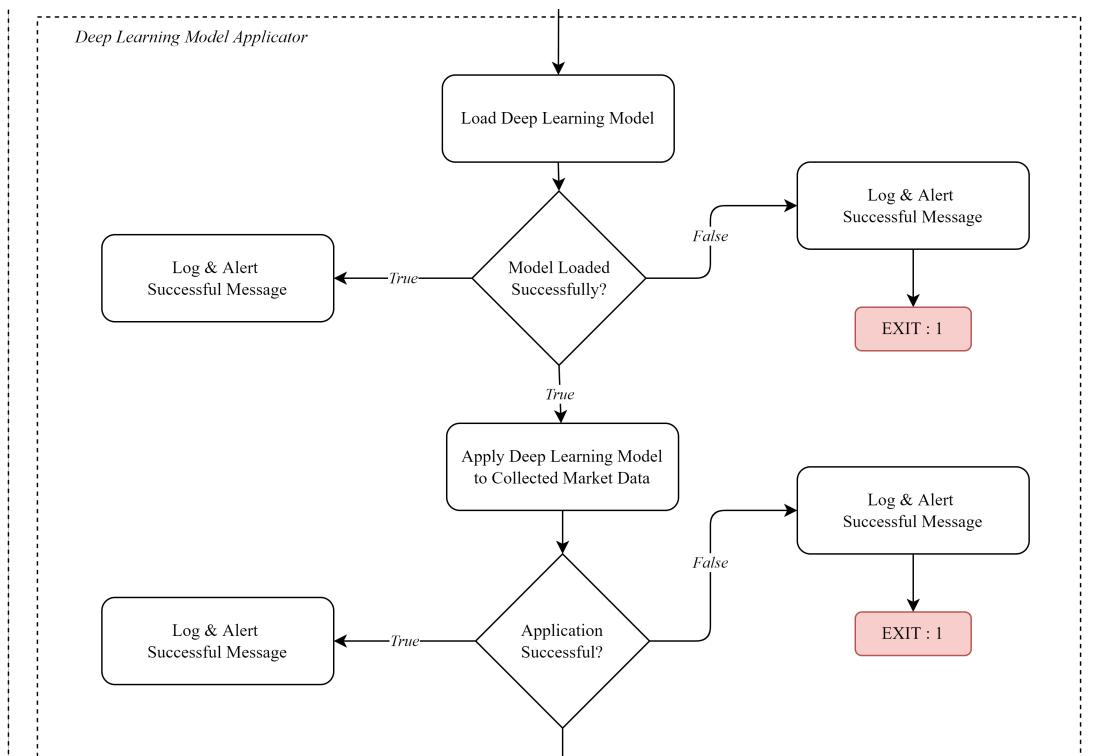


Figure 3.6: Overview of the Process Flow Diagram for the Deep Learning Model Applicator

- (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 3.7.

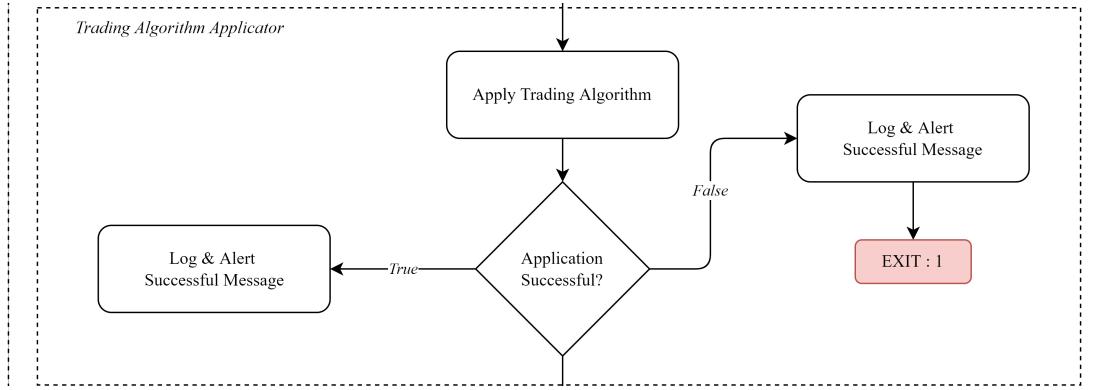


Figure 3.7: Overview of the Process Flow Diagram for the Trading Algorithm Applicator

- (c) Database Updater - This subprocess updates the database with the output data from the trading algorithm applicator. This is done as illustrated in Figure 3.8.

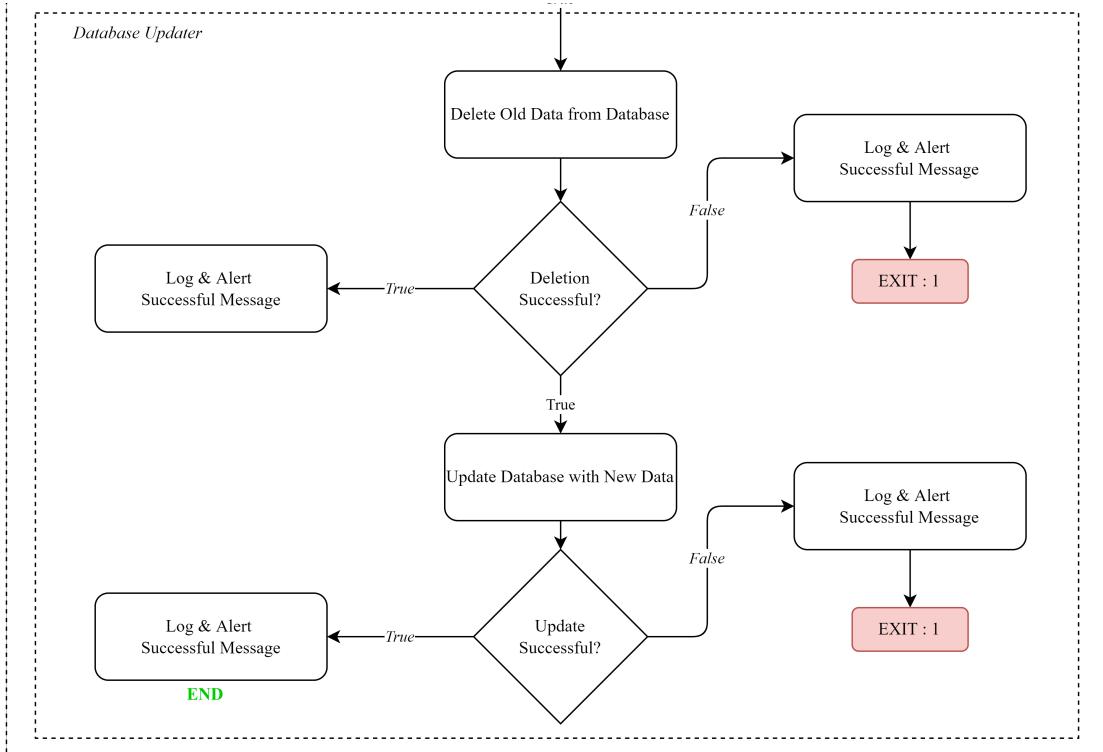


Figure 3.8: Overview of the Process Flow Diagram for the Database Updater

3.2.3 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 3.9.

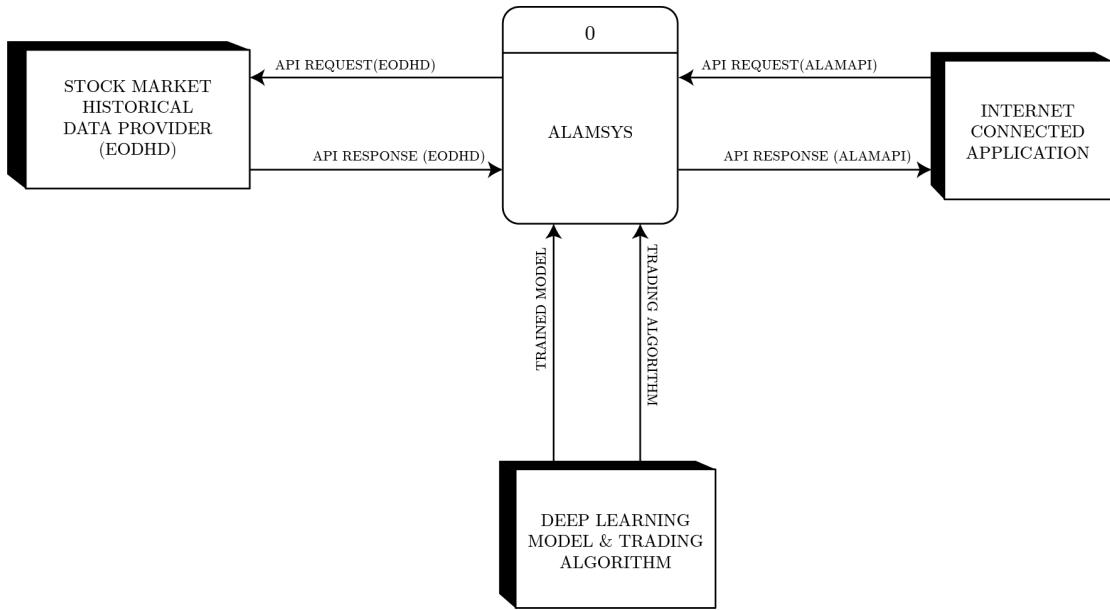


Figure 3.9: Context Diagram of the alamSYS

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

- Stock Market Historical Data Provider - which was provided by EODHD.
- Deep Learning Model & Trading Algorithm - these were developed alongside the alamSYS, specifically the DMD-LSTM Model and ALMACD, respectively.
- 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 3.9 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 3.10.

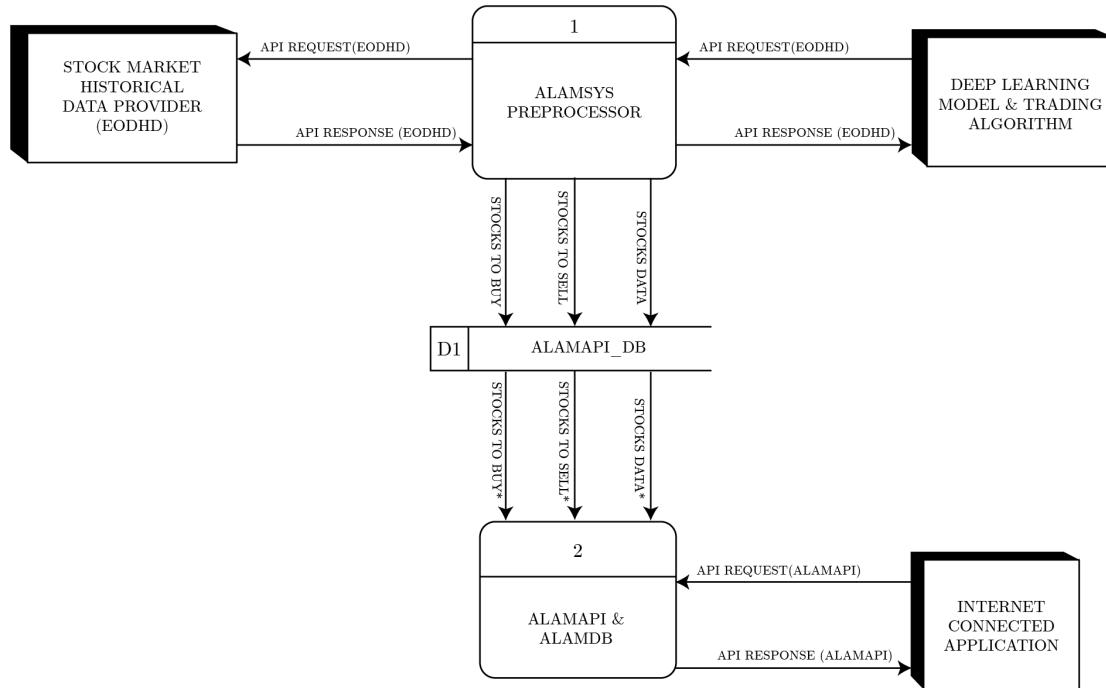


Figure 3.10: DFD of Diagram 0

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

- 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.
- 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 3.11

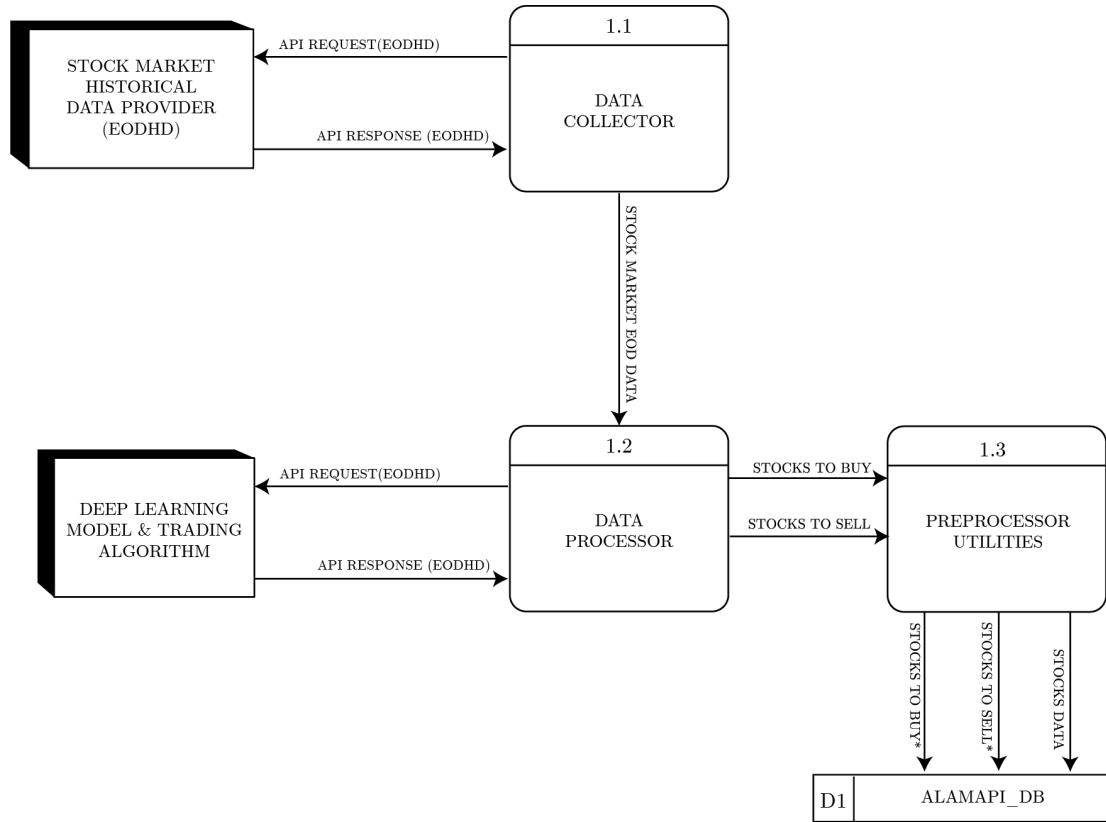


Figure 3.11: DFD of Diagram 1

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

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

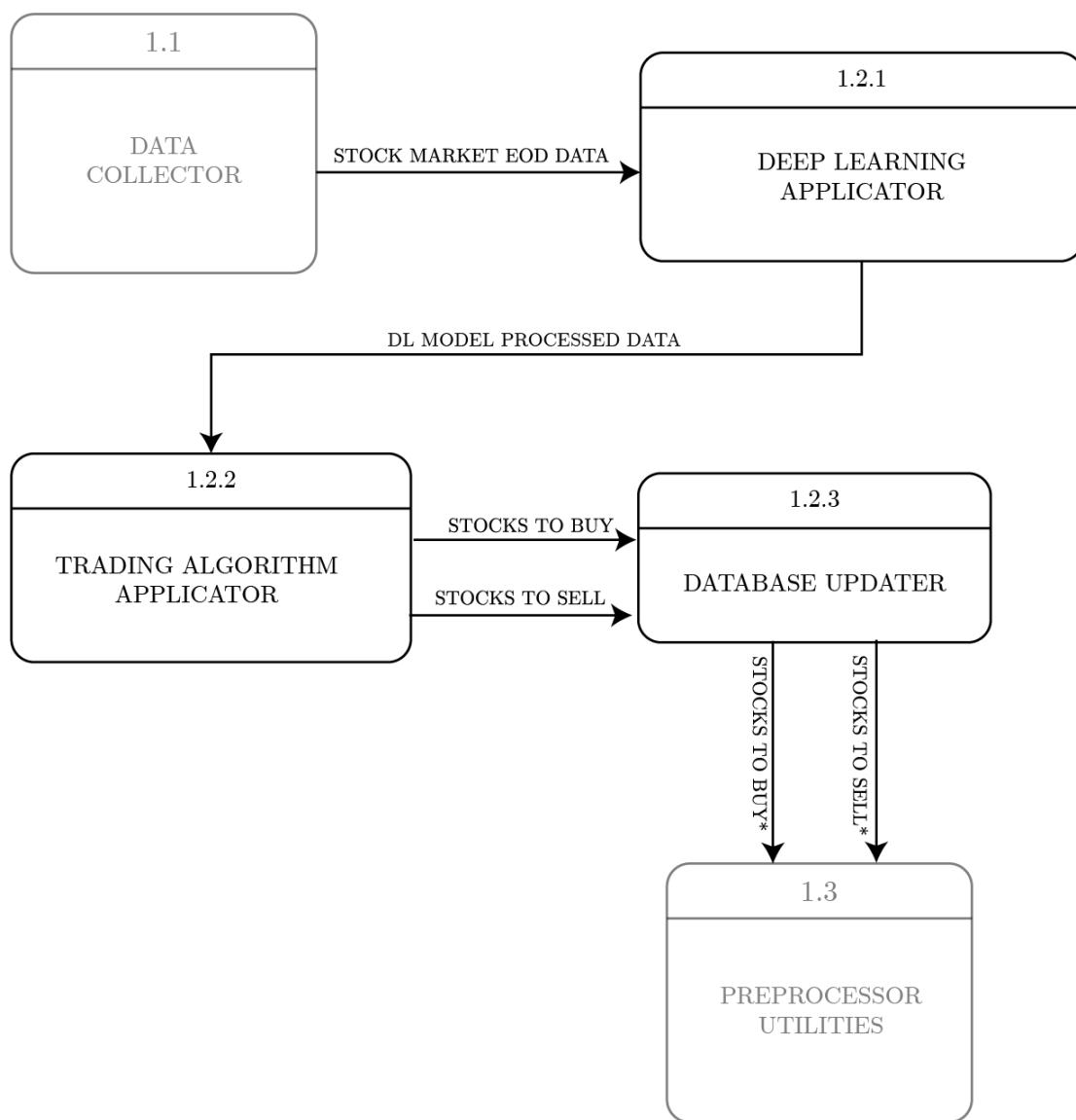


Figure 3.12: DFD of Diagram 1.2

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

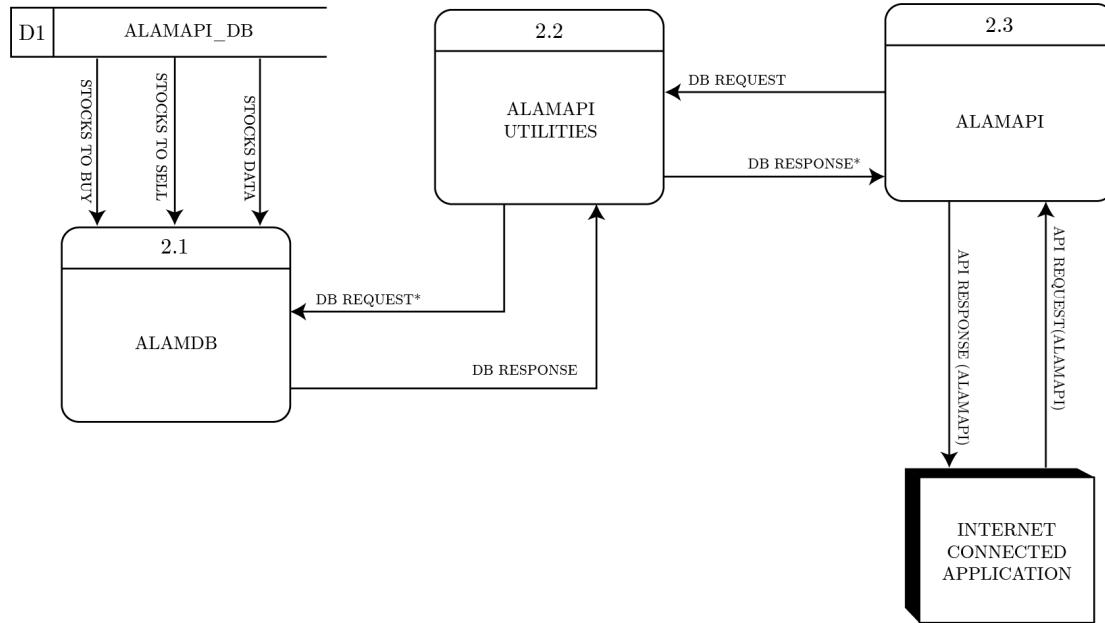


Figure 3.13: DFD 2: Data-Flow Diagram for the alamSYS

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.

3.2.4 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 3.14:

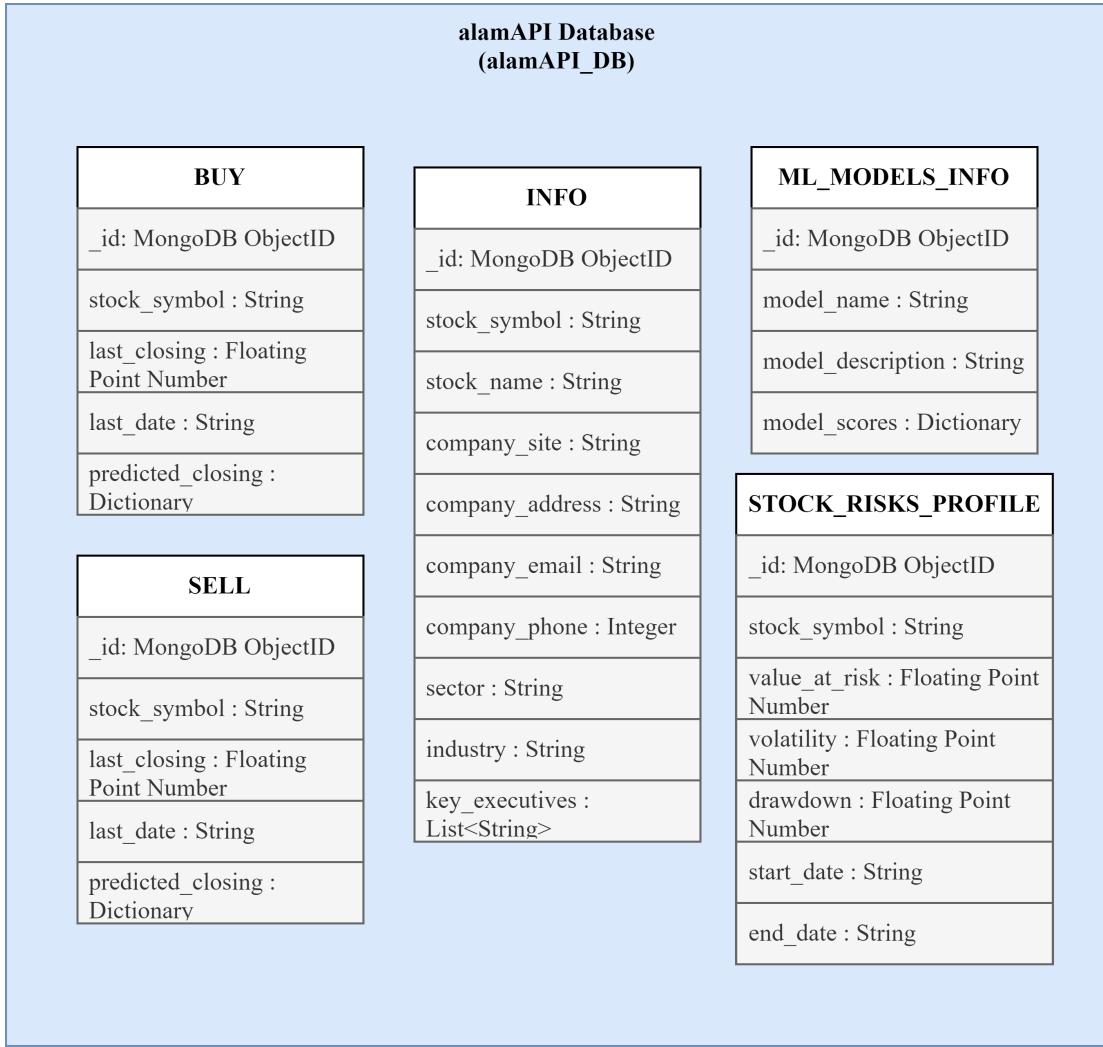


Figure 3.14: Object Document Mapper for the alamSYS Database

As shown from the Figure 3.14, 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 3.15.

The screenshot shows the MongoDB Compass interface for the 'alamAPI_DB.buy' collection. At the top right, it displays '11 DOCUMENTS' and '1 INDEXES'. Below the header, there's a navigation bar with tabs for 'Documents', 'Aggregations', 'Schema', 'Explain Plan', 'Indexes', and 'Validation'. A search bar with the placeholder 'Type a query: { field: 'value' }' is followed by buttons for 'Reset', 'Find', and 'More Options'. Below the search bar are buttons for 'ADD DATA' and 'EXPORT COLLECTION'. The main area contains three document cards, each representing a stock record:

- Document 1:** `_id: ObjectId('64368d0413193f92da07ba45')`, `stock_symbol: "MEG"`, `last_closing: 2.02`, `last_date: "2023-04-12"`, `predicted_closing: Object`
- Document 2:** `_id: ObjectId('64368d0413193f92da07ba46')`, `stock_symbol: "BDO"`, `last_closing: 130.2`, `last_date: "2023-04-12"`, `predicted_closing: Object`
- Document 3:** `_id: ObjectId('64368d0413193f92da07ba47')`, `stock_symbol: "GLO"`, `last_closing: 1798`

Figure 3.15: 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 3.16.

| _id | stock_symbol | last_closing | last_date | predicted_closing |
|--|--------------|--------------|--------------|-------------------|
| <code>_id: ObjectId('64368d0413193f92da07ba50')</code> | JGS | 49.35 | "2023-04-12" | Object |
| <code>_id: ObjectId('64368d0413193f92da07ba51')</code> | FGEM | 16.54 | "2023-04-12" | Object |
| <code>_id: ObjectId('64368d0413193f92da07ba52')</code> | ICT | 211 | "2023-04-12" | Object |

Figure 3.16: 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 3.17.

```

_id: ObjectId('642ad1c3823687269ffb43a9')
stock_symbol: "MEG"
stock_name: "Megaworld Corporation"
company_site: "https://www.megaworldcorp.com"
company_address: "30th Floor, Alliance Global Tower, 36th Street cor. 11th Avenue, Uptow..."
company_email: "investorrelations@megaworldcorp.com"
company_phone: 63288886342
sector: "Property"
industry: "Real Estate Development"
key_executives: Array

_id: ObjectId('642ad1c3823687269ffb43aa')
stock_symbol: "JGS"
stock_name: "JG Summit Holdings, Inc."
company_site: "https://www.jgsummit.com.ph"
company_address: "43/F Robinsons Equitable Tower, ADB Avenue corner Poveda St., Ortigas -"
company_email: "TR@jgsummit.com.ph"

```

Figure 3.17: 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 3.19.

```

_id: ObjectId('642ad1c3823687269ffb43bd')
model_name: "DMD-LSTM"
model_description: "Implemented dynamic modes from Dynamic Mode Decomposition (DMD) to the..."
model_scores: Object
  average_mse: "1993.39569"
  average_rmse: "17.67005"
  average_mae: "12.03009"
  average_mape: "0.035395"

```

Figure 3.18: 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 3.19.

```

_id: ObjectId('642ad1c3823687269ffb43be')
stock_symbol: "MEG"
value_at_risk: -5.365715132
volatility: 3.9499692973
drawdown: 57.2481396806
start_date: "2000-01-03"
end_date: "2023-02-10"

_id: ObjectId('642ad1c3823687269ffb43bf')
stock_symbol: "JGS"
value_at_risk: -4.7623573876
volatility: 3.3610994816
drawdown: 43.1840442168
start_date: "2000-01-03"
end_date: "2023-02-10"

```

Figure 3.19: 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.

3.2.5 Deep Learning Model Diagram

In this section, the process on how the deep learning model was developed is shown in Figure 3.20. Wherein, the process overview is based on the Fine-Tuned Support Vector Regression Model for Stock Predictions by Dash and Dash (2016).

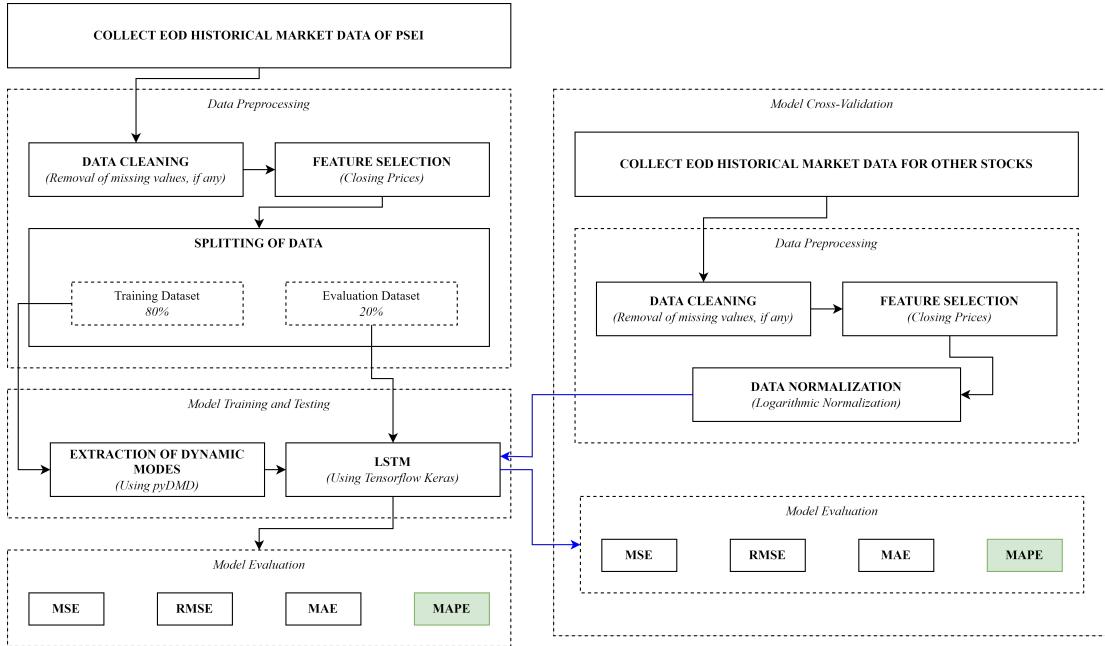


Figure 3.20: DMD-LSTM Model Development Methodology for alamSYS

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 3.1: 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 3.1 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.

3.2.6 ALMACD Development Diagram

This section discusses the development and integration of the Arnaud Legox Moving Average Convergence and Divergence (ALMACD) indicator as a trading algorithm. The ALMACD indicator tells traders when to buy and sell securities based on the convergence and divergence of two ALMA (fast and slow). Specifically, when the fast ALMA crosses above the slow ALMA, the indicator suggests buying the stock. Conversely, when the fast ALMA crosses below the slow ALMA, the indicator suggests selling the stock.

The process of the development and integration of the ALMACD is shown in Figure 3.21.

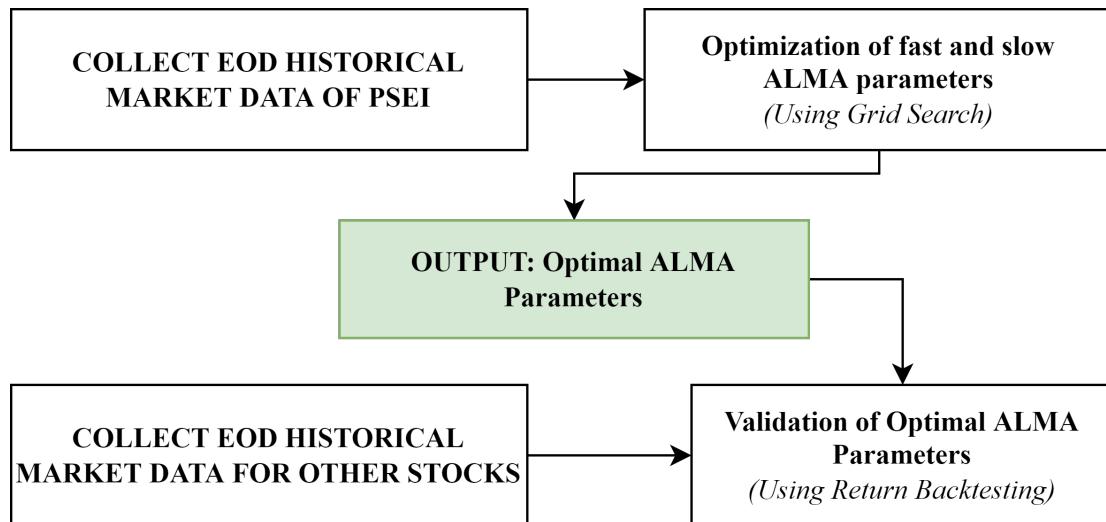


Figure 3.21: ALMACD Development Methodology for alamSYS

Optimization of ALMA Parameters using Grid Search Approach

The ALMACD indicator utilizes two ALMA indicators, one with a fast period and one with a slow period. Where each of them have their own set of optimal parameters, such as their window size, offset, and sigma. In order to find this optimal parameters, a grid search approach was used.

Grid search is a machine learning technique used to fine-tune the hyperparameters of a model. It is a brute force approach that tries every combination of hyperparameters and evaluates the model performance for each combination. The combination with the best performance is then selected as the optimal hyperparameters (Joseph, 2018). However in the case of the optimization of ALMACD, the grid search approach was utilized to find the optimal parameters for the fast and slow alma, such that it yields a favorable performance by computing the expected return of the trading algorithm. Whereas the following parameters were used for the grid search approach:

- For the fast ALMA:

window size: a range from 1 to 10

sigma: a range from 1 to 20

offset: 0.85, 0.90, and 0.95

- For the slow ALMA:

window size: a range from 10 to 20

sigma: a range from 1 to 20

offset: 0.85, 0.90, and 0.95

Validation of ALMACD Parameters Using Return Backtesting Approach

Before integrating the optimal ALMACD parameters to the alamSYS, it was first validated using the data from other stocks (Return Backtesting Approach). Once the returns on the validation stocks were computed to be positive, the ALMACD indicator was then integrated to the alamSYS.

Integration of ALMACD to the alamSYS

The ALMACD indicator was integrated to the alamSYS as a trading algorithm. It was implemented to run after the deep learning model to better identify which stocks to buy and sell.

3.2.7 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 3.22 shows a diagram based on Docker documentation, to better understand the containers and layers of the alamSYS. Note that in the diagram the lowest level is the "Server Infrastructure" and the highest level is the "Docker-Compose" layer.

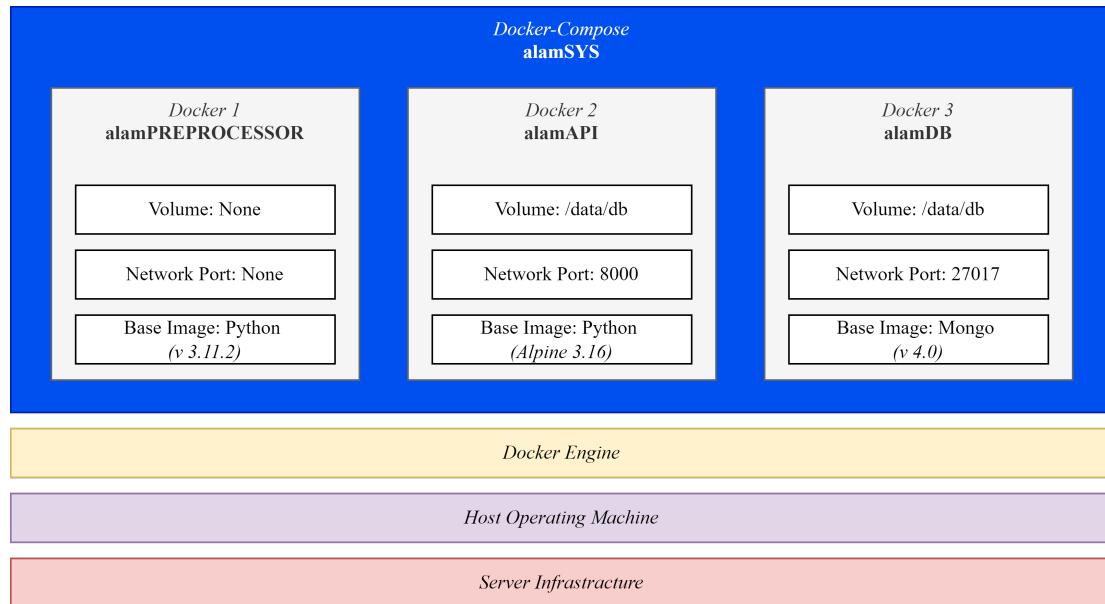


Figure 3.22: Docker-Compose Layer Diagram for the alamSYS

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.

3.3 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.

3.3.1 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

3.3.2 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

3.3.3 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

3.4 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.

3.4.1 Software Development Process

Because of the expected tight time constraints during system development, the author of this paper chose an Agile Software Development Methodology. Agile Sprints were primarily used to manage time efficiently during the software development process (JavaTPoint, n.d.; Milne, 2021). Moreover, a modified Agile Development Methodology as shown in Figure 3.23 was followed in the system development procedures for this special problem.

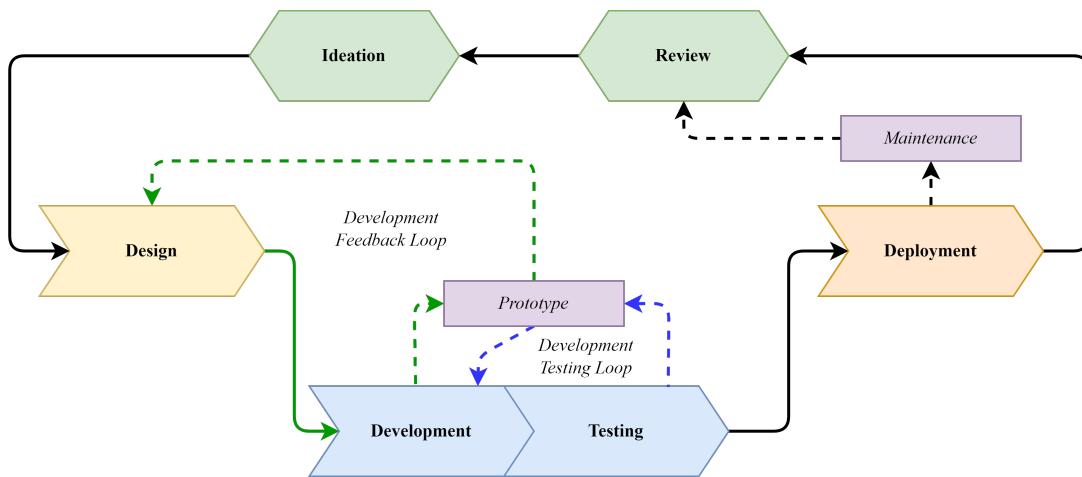


Figure 3.23: Modified Agile Development Methodology

The modified agile development methodology shown above includes additional and specific "feedback loops" such as a development feedback loop that focuses on the continuous design and development process; and a development testing loop that ensures that the software being developed works as intended or better than expected.

The Table 3.2 shows the list of sprints and sub-activities that were followed:

Table 3.2: Summary of Sprints and Activities

| Sprint Number | Activities | Completion Date |
|---------------|--|---|
| 1 | <p>Main Activity: Requirement Analysis, System Planning, and Evaluation</p> <p>Sub-Activities:</p> <ul style="list-style-type: none"> • Topic Proposal • Drafted Chapters 1 to 3 for the Special Problem Proposal • System Architecture and User Requirement Analysis | September 15, 2022 to December 9, 2022 |

Table 3.2 continued from previous page

| Sprint Number | Activities | Completion Date |
|---------------|--|--|
| 2 | <p>Main Activity: Development of Initial System Prototype</p> <p>Sub-Activities:</p> <ul style="list-style-type: none"> • Build the different component of the alamSYS as indicated in the top-level overview diagram of the system, the following prototype were initially developed: <ul style="list-style-type: none"> [1.] API endpoints [2.] Database [3.] Preprocessor • Tested the build prototype. | <p>12 Weeks September 30, 2022 to December 23, 2023</p> |

Table 3.2 continued from previous page

| Sprint Number | Activities | Completion Date |
|---------------|---|---|
| 3 | <p>Main Activity: Development and Testing of DMD-LSTM Model</p> <p>Sub-Activities:</p> <ul style="list-style-type: none"> • Stock Market Data until February 10, 2023 was collected. • Development of the Deep Learning Model, following the procedures as previously discussed on section 3.2.5. • Development of Baseline LSTM model for comparison against DMD-LSTM Model. • Deep Learning model training, testing, cross-validation, and evaluation. • Revised of Chapters 1 to 3. | January 1, 2023 to February 21, 2023 |

Table 3.2 continued from previous page

| Sprint Number | Activities | Completion Date |
|---------------|---|-----------------------------------|
| 4 | <p>Main Activity: Integration of DMD-LSTM Model to the alamSYS and System Testing</p> <p>Sub-Activities:</p> <ul style="list-style-type: none"> • Integrated DMD-LSTM to alamSYS. • Initial tests of the system. • Created System Tests and Loggers. • System Testing, Results Logging and Summarization. | January 15, 2023 to March 7, 2023 |

Table 3.2 continued from previous page

| Sprint Number | Activities | Completion Date |
|---------------|---|------------------------------------|
| 5 | <p>Main Activity: System Testing, Analysis, Refactoring, and Deployment</p> <p>Sub-Activities:</p> <ul style="list-style-type: none"> • Development test loop was conducted. • System Refactoring. • System Update and inclusion of additional features. • Results Gathering and Summarization. • Started the development of the test application (for showcasing of the system features). | March 7, 2023 to April 13, 2023 |

Table 3.2 continued from previous page

| Sprint Number | Activities | Completion Date |
|---------------|--|-----------------------------------|
| 6 | <p>Main Activity: Finalization of Paper, System Defense, and Presentation</p> <p>Sub-Activities:</p> <ul style="list-style-type: none"> • Revised Chapter 1 to 3. • Drafted Chapter 4 to 5. • Finalization of the mobile-based test application. • Reviewed the Content of this paper. • Revised and Finalized the special problem paper. • Created presentation slide deck for the presentation of the special problem. | March 14, 2023 to May 12, 2023 |

From Table 3.2, it was shown that it took a total of 34 weeks (from September 15, 2022, to May 12, 2023) to develop the alamSYS and all its components, as well as to write the contents of this paper. This was 5 weeks better than the expected total weeks allotment for the completion of this whole project, as projected in the Gantt Charts shown in section 3.5.

3.4.2 Procedures

This section provides a general overview of the steps conducted in the development of the different components developed in the whole duration of this special problem.

Procedures for the Development of the alamSYS and its Components

The following step-by-step procedures were conducted in the development of the alamSYS and its components (alamAPI, alamDB, and alamPREPROCESSOR):

- (a) User Requirement and Analysis
- (b) System Design and Architecture using DFD
- (c) System Prototyping and Development.
- (d) System Testing and Continuous Development and Integration. Each of the components of the alamSYS were developed into smaller chunks, and each chunks were tested and integrated with the other smaller chunks until a component was functionally developed. Afterwards the same process for the other components were conducted, until a whole working system was developed.
- (e) System Deployment, Review, and Maintenance
- (f) Steps (a) to (e) were repeated until the system was deemed to be stable and ready for deployment within the scopes of this special problem. Moreover, the system shall be maintained even after the submission of this special problem such as the addition of new features.

Procedures for the Development of the DMD-LSTM Model

The following step-by-step procedures were conducted in the development of the DMD-LSTM model:

- (a) Collection of end-of-day Philippine Stock Market Data from the 20 selected stocks from the Philippine Stock Exchange (PSE) for the periods until February 10, 2023.
- (b) Data Preprocessing, Model Training, Testing, Cross-validation, and Evaluation was conducted based on the DMD-LSTM methodology shown in Figure 3.20.
- (c) The best performing DMD-LSTM model was deployed and integrated as part of the alamSYS, specifically used in the alamPREPROCESSOR component.

Procedures for the Development of the ALMACD Trading Algorithm

The following step-by-step procedures were conducted in the development of the ALMACD trading algorithm:

- (a) Collection of end-of-day Philippine Stock Market Data from the 20 selected stocks from the Philippine Stock Exchange (PSE) for the periods until February 10, 2023.
- (b) Using a Grid Search approach, the best ALMA parameters (*i.e.* window size, sigma, and offset) for both slow and fast were determined.
- (c) Using the best ALMA parameters, a trading algorithm was developed which takes in the convergence and divergence of the slow and fast ALMA as basis for entry and exit signals. When the slow ALMA crosses above the fast

ALMA, a buy signal is generated. When the slow ALMA crosses below the fast ALMA, a sell signal is generated.

- (d) The trading algorithm was deployed and integrated as part of the alamSYS, specifically used in the alamPREPROCESSOR component.

Procedures for the Development of Mobile-based Test Application

The following step-by-step procedures were conducted in the development of the mobile-based test application:

- (a) User Interface Design and Development using Figma.
- (b) Creation of Initial Prototype using Flutter.
- (c) Continuation of Feature Development and Testing.
- (d) Integration of HTTP methods with the alamAPI.

Note that the purpose of this application is to showcase the main feature of the alamSYS, as such it is not fit for user deployment. However it is further discussed in the recommendations on how it can be potentially deployed by future developers.

Procedures for the System Testing

The following step-by-step procedures were conducted in the system testing:

- (a) Development of tester application, which are as follows:
 - System Statistics Logger - Using the docker stats stream command, the system CPU, memory, and network utilization stats were recorded.

This logger was used to determine the idle and on load performance of the alamSYS.

- Deployment Tester - This tester application was used to test the deployment reliability of the alamSYS. The test was conducted by deploying the alamSYS in a server while then other computers requests for different functions to the alamSYS over the internet using the HTTP methods. Further the tester also logs the overall response time to process 10, 100, and 1000 requests in a row, as well as the success rate of the requests.
 - Internal System Stress Tester - This tester application was used to test the internal system reliability of the alamSYS. The test was conducted to run the processes of the alamSYS in a loop for 100 consecutive times, and logs the overall response time as well as success rate for each processes.
- (b) Initial testing of the tester applications, to see if they are working as intended
- (c) Actual testing was done using these tester applications, and all results and system logs were recorded.
- (d) The results of the testing were then analyzed, compared, and summarized. Further details regarding this are discussed on Chapter 4.

3.5 Gantt Chart

Based on Table 3.2, the following figures for the Gantt Chart (*created using TeamGantt*) shows the software development schedule for the development of alamSYS. The Gantt Chart was divided into the different sprint to present the project scheduling. Moreover, a zoomed-out view of the whole Gantt Chart, was also be provided at the end of this section.

Note that the schedules in the Gantt Chart were created during the proposal as

such some of the schedules may not coincide with the actual dates indicated in Table 3.2 as changes were made along the way.

3.5.1 Gantt Chart for Sprint 1

Figure 3.24 shows the schedule of activities for Sprint 1.

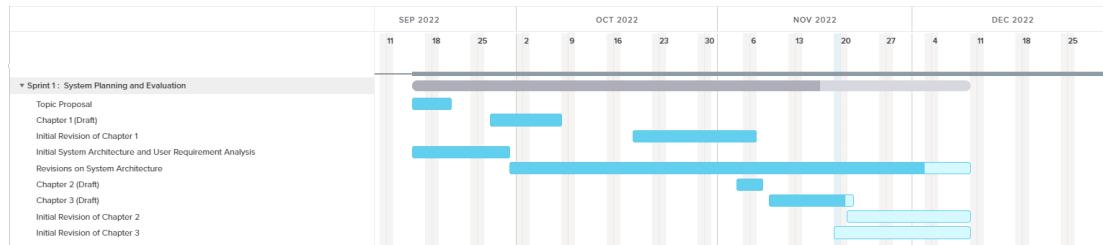


Figure 3.24: Gantt Chart for Sprint 1

3.5.2 Gantt Chart for Sprint 2

Figure 3.25 shows the schedule of activities for Sprint 2.

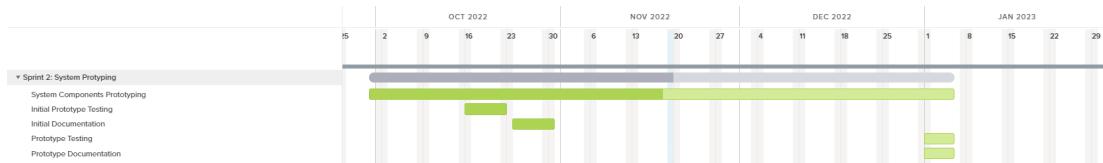


Figure 3.25: Gantt Chart for Sprint 2

3.5.3 Gantt Chart for Sprint 3

Figure 3.26 shows the schedule of activities for Sprint 3.

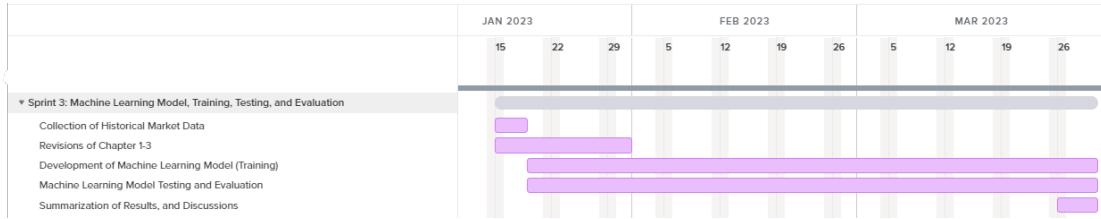


Figure 3.26: Gantt Chart for Sprint 3

3.5.4 Gantt Chart for Sprint 4

Figure 3.27 shows the schedule of activities for Sprint 4.

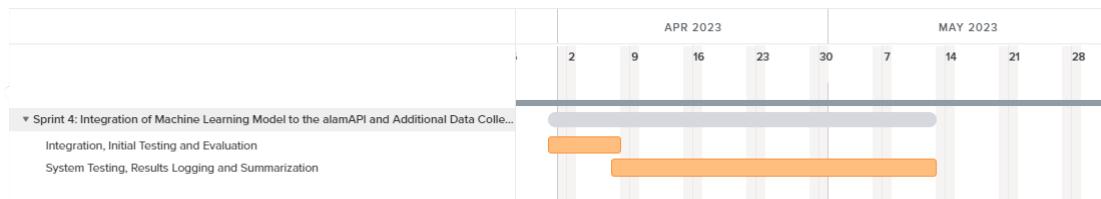


Figure 3.27: Gantt Chart for Sprint 4

3.5.5 Gantt Chart for Sprint 5

Figure 3.28 shows the schedule of activities for Sprint 5.

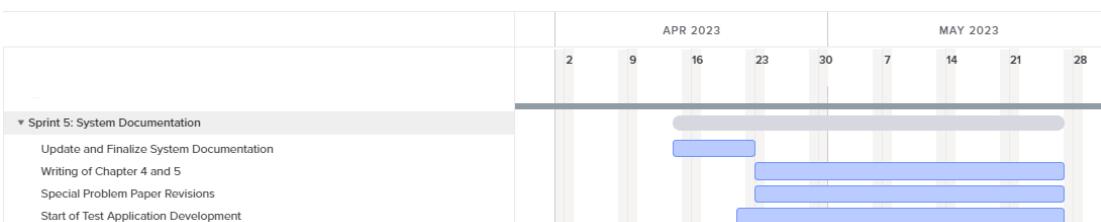


Figure 3.28: Gantt Chart for Sprint 5

3.5.6 Gantt Chart for Sprint 6

Figure 3.29 shows the schedule of activities for the final sprint for the development of alamSYS.

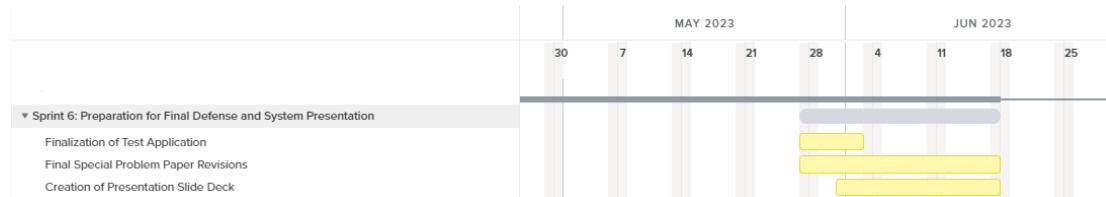


Figure 3.29: Gantt Chart for Sprint 6

3.5.7 Full Gantt Chart

To have an overview of the whole schedule of each Sprints, the full Gantt chart is shown in Figure 3.30.

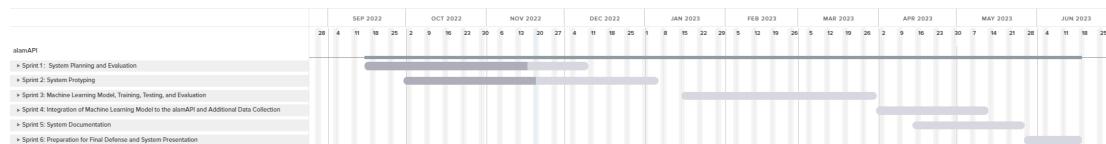


Figure 3.30: Full Gantt Chart

Chapter 4

Results and Discussions

This chapter presents results and discussions from this special problem. Its goal is to provide a comprehensive analysis and interpretation of the data collected for alamSYS's internal and external components. As a result, this chapter is divided into the following sections:

- (a) Documentation for alamSYS
- (b) DMD-LSTM Results and Discussions
- (c) ALMACD Results and Discussions
- (d) alamSYS System Tests Results and Discussions
- (e) Results and Discussions for the Real World Application of alamSYS

4.1 alamSYS Documentation

The goal of this section is to thoroughly document the current state of the alamSYS in order to facilitate meaningful discussions.

4.1.1 Documentation for alamPREPROCESSOR

The documentation section for the alamPREPROCESSOR discusses the currently implemented features, as well as a guide to the file directories to manage these files as effectively as possible for any future updates, maintenance work, and testing required.

alamPREPROCESSOR Features

The current features implemented in the alamPREPROCESSOR are as follows:

- (a) The Data Collector Module (DCM) was integrated into the alamPREPROCESSOR to allow the alamSYS to collect end-of-day historical data from the various stocks on the Philippine Stock Exchange (including the PSEI). EODHD data is collected using the web API.
- (b) The Data Processor Module (DPM) was built into the alamPREPROCESSOR to enable the alamSYS to perform the following tasks:
 1. Apply DMD-LSTM to each stock dataset.
 2. Create a list of stocks to buy and sell by applying the ALMACD to the last 200 days of actual stock data and the five predicted data points from the DMD-LSTM model; and
 3. Update the database with new information about which stocks to buy or sell.
- (c) The alamPREPROCESSOR handles items (a) and (b) automatically. This means that they are set to run at a specific time, which is 6 p.m. every Monday through Friday. Some errors are also handled automatically if they are internally related, which means they are not affected by external factors such as a slow and unstable internet connection, a downtime in the EODHD APIs, and so on.
- (d) In relation to item (c), all system logs are saved in the system's shared docker volume at '/data/db/', as discussed further in the subsequent section.

- (e) The alamPREPROCESSOR is also in charge of adding the following data to the alamDB collection if it does not already exist when the alamSYS starts up:
1. Stock Info Collection
 2. Model Info Collection; and
 3. Stock Risks Profile Collection
- (f) In connection with item (b.3), the alamPREPROCESSOR also saves all historical stock suggestions in the old directories with the current date when they were saved. Furthermore, these data are no longer reflected in the database and are only stored internally in the system.
- (g) Other minor features connected to the utilities directory:
1. Database model definitions
 2. Database actions such as:
 - i. Connecting to the alamDB.
 - ii. Disconnecting to the alamDB.
 - iii. Purging the Buy Collections.
 - iv. Purging the Sell Collections.
 - v. Saving the Updated Stocks to Buy JSON file.
 - vi. Saving the Updated Stocks to Sell JSON file.
 - vii. Saving the Stocks Info Collection from the JSON file.
 - viii. Saving the Model Info Collection from the JSON file; and
 - ix. Saving the Stock Risks Profile Collection from the JSON file.
- (h) Finally, in relation to item (c), system maintainers or administrators can easily restart system critical processes using the system's command line interface provided by docker in cases where errors are not automatically handled by the system. Additionally, they can use the Python program to debug the cause of the error, and they can use vim to change some parts of the source code to fix the error, if ever needed.

alamPREPROCESSOR Docker Management

The management and maintenance of the alamPREPROCESSOR is made possible through the utilization of Docker Container Management tools such as the Docker Desktop, which is a GUI Windows-based software used to maintain and manage docker containers.

Figure 4.1 depicts the tree of files and directories accessible from '/preprocessor/' of the alamPROCESSOR docker container.

```
data_collector
|   __init__.py
|   __pycache__
|   |   __init__.cpython-311.pyc
|   |   collector.cpython-311.pyc
|   |   test.cpython-38.pyc
|   collector.py
data_processor
|   __init__.py
|   __pycache__
|   |   alma_processor.cpython-311.pyc
|   |   ml_processor.cpython-311.pyc
|   |   ml_processor.cpython-38.pyc
|   |   process.cpython-311.pyc
|   alma_processor.py
|   ml_model
|   |   README.md
|   |   __init__.py
|   |   dmd_lstm.keras
|   ml_processor.py
|   process.py
main.py
manual_run.py
requirements.txt
utils
|   __init__.py
|   __pycache__
|   |   __init__.cpython-311.pyc
|   |   db_actions.cpython-311.pyc
|   |   init_db.cpython-311.pyc
|   |   logs_and_alerts.cpython-311.pyc
|   |   logs_and_alerts.cpython-38.pyc
|   |   models.cpython-311.pyc
|   |   stock_symbols.cpython-311.pyc
|   db_actions.py
|   init_db.py
|   json_data
|   |   *
|   |   model_info.json
|   |   stock_info.json
|   |   stock_risks.json
|   logs_and_alerts.py
|   models.py
|   stock_symbols.py

8 directories, 36 files
```

Figure 4.1: alamPREPROCESSOR Docker Container Directory Tree

On the other hand, the alamPREPROCESSOR shares the following directory paths related to error logging and json directories with the other components of the alamSYS via the '/data/db' directory path:

- (a) /data_collector_logs - This is the directory for all the successful operations ('/success_log.csv') , and all the unsuccessful operations ('/error_log.csv') conducted by the data collector.
- (b) /data_processor_logs - This is the directory for all the successful operations ('/success_log.csv') , and all the unsuccessful operations ('/error_log.csv') conducted by the data processor.
- (c) /manual_run_logs - This is the directory for all the successful operations ('/success_log.csv') , and all the unsuccessful operations ('/error_log.csv') conducted by manually running the operations of the alamPREPROCESSOR using the command 'python3 manual_run.py'.
- (d) /preprocessor_utils_logs - This is the directory for all the logged actions conducted by utilities of alamPREPROCESSOR. This is further divided into two directories, which are as follows:
 1. db_actions - Stores logs of successful operations ('/success_log.csv') , and all the unsuccessful operations ('/error_log.csv') from the utilities related to database actions.
 2. init_db - Stores logs of successful operations ('/success_log.csv') , and all the unsuccessful operations ('/error_log.csv') from the utilities related to the initialization of the database.
- (f) /scheduled_task_logs - This is the directory for all the successful operations ('/success_log.csv') , and all the unsuccessful operations ('/error_log.csv') conducted by the scheduled operation of the alamPREPROCESSOR.

4.1.2 Documentation for alamAPI and alamDB

The alamAPI and alamDB documentation section discusses the currently implemented features for the API and database, respectively. This should be used as a guide for any future updates, maintenance, and testing.

alamAPI API Endpoints

This section contains the alamAPI endpoints whose Swagger-based web documentation is accessible via this local link: `localhost:8000/alamAPI/v1/docs`, as shown in Figure 4.2. Alternatively, Redocly's alamAPI API documentation can be accessed via this link: `localhost:8000/alamAPI/v1 docs`, as shown in Figure 4.2.

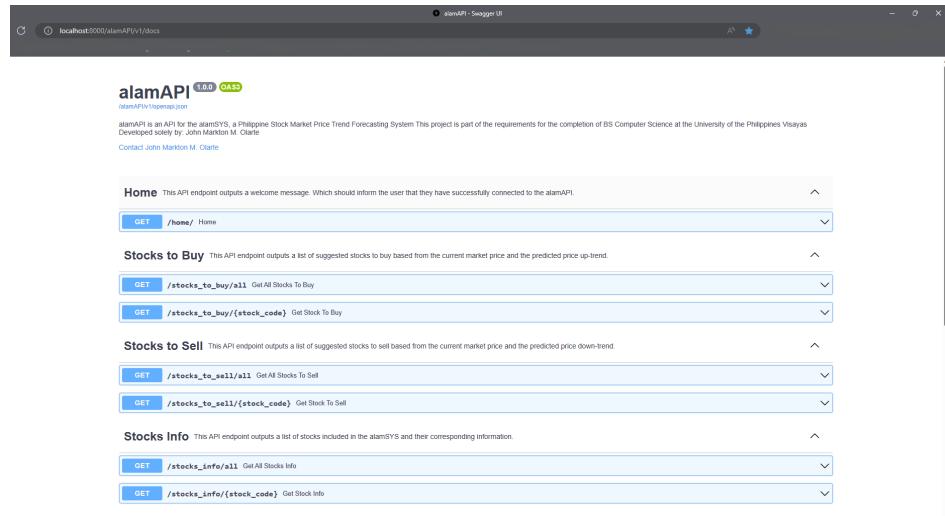


Figure 4.2: alamAPI Web-based API Documentation Using Swagger

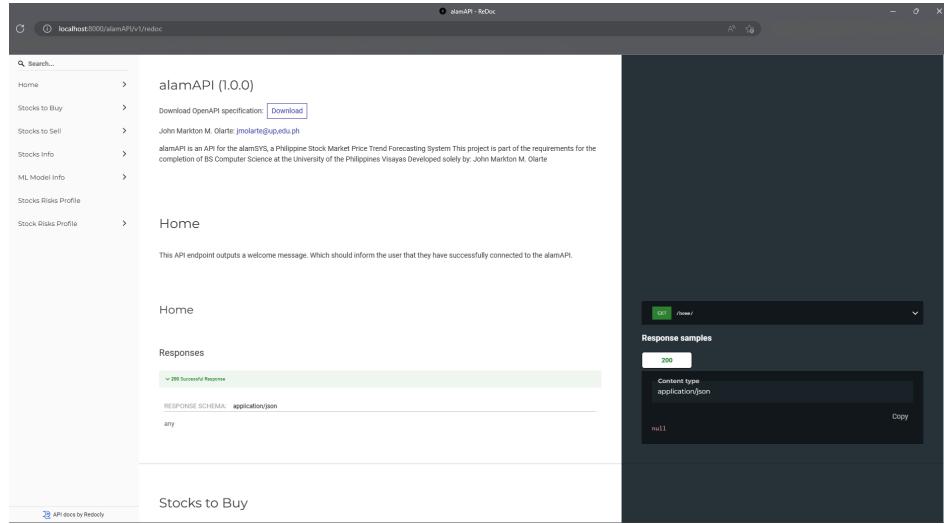


Figure 4.3: alamAPI Web-based API Documentation Using Redocs

Moving on, the Swagger-based documentation was used to show the API Endpoints. To begin, the 'Home' refers to the API endpoint that returns a welcome message informing the API user that they have successfully connected to the alamAPI and that everything is functioning properly. Figure 4.4 depicts an example request executed using this API endpoint.

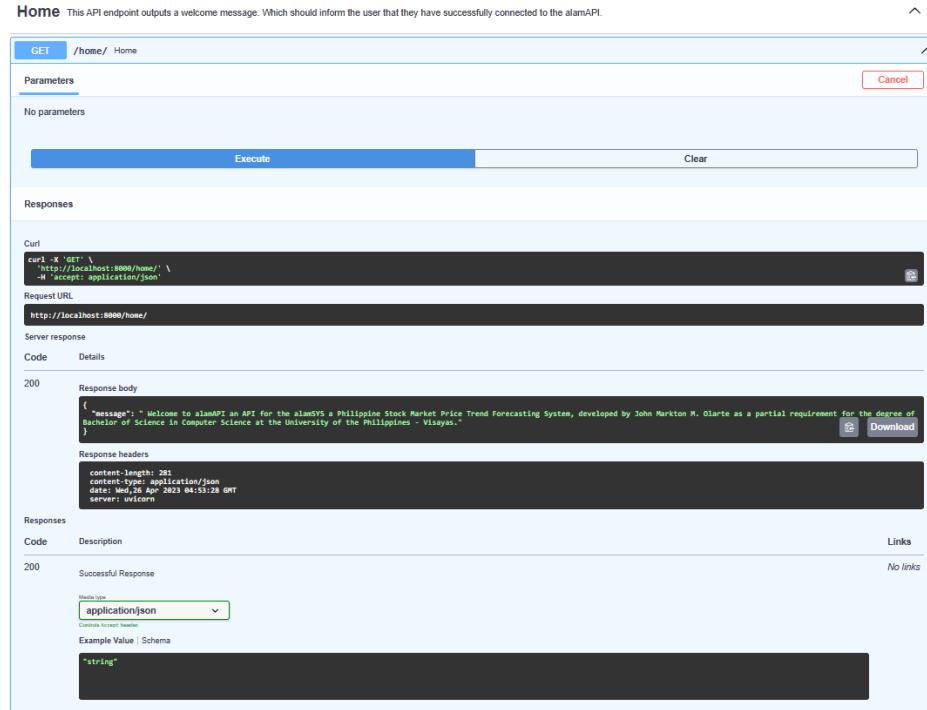


Figure 4.4: Sample API Endpoint: Home Request

Next, API endpoints related to the 'Stock to Buy', which output a list of suggested stocks to buy based on the current market price and the predicted price up-trend. This API endpoint has two uses: users can use the /all path to see all of the suggested stocks to buy, or they can use /stock_code, which requires a specific stock code and returns the details for that stock if it exists in the stocks to buy list, and if it does not, it instead returns a 'Stock not found' alert. A sample execution, specifically for /all is shown in Figure 4.5

The screenshot shows a sample API endpoint for buying stocks. The endpoint is `/stocks_to_buy/all`. The interface includes:

- Curl:** A command-line tool for sending requests to servers. The command shown is:


```
curl -X 'GET' \
        'http://localhost:3000/stocks_to_buy/all' \
        -H 'accept: application/json'
```
- Request URL:** `http://localhost:3000/stocks_to_buy/all`
- Server response:**
 - Code:** 200
 - Response body:**

```
{
  "Stocks": [
    {
      "id": {
        "id": "6448b1e49576da851cff34460"
      },
      "stock_symbol": "GOO",
      "predicted_trend": "Up",
      "last_date": "2023-04-25",
      "predicted_closing": {
        "low": 125.00000000000002,
        "high": 135.00000000000001
      }
    },
    {
      "id": {
        "id": "6448b1e49576da851cff34461"
      },
      "stock_symbol": "TSLA",
      "predicted_trend": "Up"
    }
  ]
}
```
 - Response headers:**

```
content-length: 3436
content-type: application/json
date: Wed, 26 Apr 2023 05:05:30 GMT
server: unicorn
```
- Responses:**
 - Code:** 200
 - Description:** Successful Response
 - Content type:** application/json
 - Example Value | Schema:** "string"

Figure 4.5: Sample API Endpoint: Buy (all) Request

The third set of API endpoints is related to the 'Stock to Sell,' which returns a list of recommended stocks to sell based on the current market price and the predicted price downtrend. Users can use the /all path to see all of the suggested stocks to sell, or they can use /stock_code, which requires a specific stock code and returns the details for that stock if it exists in the stocks to sell list, and a 'Stock not found' alert if it does not. Figure 4.6 shows a sample execution, specifically for /all.

Stocks to Sell This API endpoint outputs a list of suggested stocks to sell based from the current market price and the predicted price down-trend.

The screenshot shows a detailed view of an API endpoint for 'Stocks to Sell'. At the top, there's a blue header bar with the method 'GET' and the URL '/stocks_to_sell/all'. Below this is a 'Parameters' section which says 'No parameters'. There are two buttons: 'Execute' (in blue) and 'Cancel' (in red). Under the 'Responses' section, there's a 'Curl' block containing a command-line example, a 'Request URL' block with 'http://localhost:8000/stocks_to_sell/all', and a 'Server response' block. The 'Server response' block is expanded to show a 200 status code. It includes a 'Response body' section with JSON data and a 'Response headers' section with details like 'Content-Length: 2450', 'Content-Type: application/json', and 'Date: Wed, 26 Apr 2023 05:15:14 GMT'. At the bottom of the 'Responses' section, there's a table for 'Responses' with a single row for code 200. The 'Description' column says 'Successful Response' and the 'Media type' dropdown is set to 'application/json'. The 'Links' column says 'No links'. At the very bottom, there's another blue header bar with 'GET /stocks_to_sell/{stock_code}' and 'Get Stock To Sell'.

Figure 4.6: Sample API Endpoint: Sell (all) Request

The next set of API endpoints is related to the 'Stocks Info', which returns a list of stocks that are part of the alamSYS. This API endpoint has two uses: users can use the /all path to see information from all 20 stocks included in this iteration of the alamSYS, or they can use /stock_code, which requires a specific stock code and returns information from that specific stock if it exists in the alamSYS, and a 'Stock not found' alert if it does not. Figure 4.7 shows a sample execution, specifically for /all.

Stocks Info This API endpoint outputs a list of stocks included in the alamSYS and their corresponding information.

GET /stocks_info/all Get All Stocks Info

Parameters

No parameters

Executes

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8000/stocks_info/all' \
  -H 'Accept: application/json'
```

Request URL

http://localhost:8000/stocks_info/all

Server response

| Code | Description | | | | | | |
|------|---|----------|-------------|-------|-----|---------------------|----------|
| 200 | Response body <pre>{ "All Stocks Info": [{ "id": "642ed1c382687309ff9a43e0", "stock_symbol": "NGG", "stock_name": "Megaworld Corporation", "industry": "Real Estate Development", "company_address": "30th Floor, Alliance Global Tower, 36th Street cor. 11th Avenue, Uptown Bonifacio, Taguig City 1634, Metro Manila, Philippines", "company_email": "investorrelations@megaworldcorp.com", "company_phone": "(632)88880343", "key_executives": [{ "name": "Andrei L. Tan", "position": "Chairman" }, { "name": "Katherine L. Tan", "position": "Director" }, { "name": "Elmer H. Tan" }] }] }</pre> <p>Download</p> Response headers <pre>content-length: 1805 content-type: application/json date: Wed, 26 Apr 2023 05:28:41 GMT server: unicorn</pre> <p>Responses</p> <table border="1"> <thead> <tr> <th>Code</th> <th>Description</th> <th>Links</th> </tr> </thead> <tbody> <tr> <td>200</td> <td>Successful Response</td> <td>No links</td> </tr> </tbody> </table> <p>Media type</p> <p><input checked="" type="button" value="application/json"/> <input type="button" value="text/plain"/> <input type="button" value="application/xml"/></p> <p>Content-type Headers</p> <p>Example Value Schema</p> <p><code>"string"</code></p> <p>GET /stocks_info/(stock_code) Get Stock Info</p> | Code | Description | Links | 200 | Successful Response | No links |
| Code | Description | Links | | | | | |
| 200 | Successful Response | No links | | | | | |

Figure 4.7: Sample API Endpoint: Stocks Info (all) Request

The fifth set of API endpoints is related to the 'ML Model Info', which returns a list of machine learning models that have been deployed in the alamSYS. In particular, in this iteration of the alamSYS, the current deployed deep learning model is DMD-LSTS, as discussed in previous Chapters of this paper. This API endpoint has two uses: users can use the /all path to see information for all models, or they can use /model_name, which requires a model name and returns information for that specific model if it exists in the alamSYS, and a 'Model not found' alert if it does not. Figure 4.8 shows a sample execution, specifically for /all.

ML Model Info This API endpoint outputs a list of the Machine Learning Models used in the alamSYS and their corresponding information.

GET /ml_model_info/all Get All Models Info

Parameters

No parameters

Responses

Curl

```
curl -X GET \ 
http://localhost:8000/ml_model_info/all \
-H "accept: application/json"
```

Request URL

http://localhost:8000/ml_model_info/all

Server response

| Code | Description | | | | | | |
|------|--|----------|-------------|-------|-----|---------------------|----------|
| 200 | Response body <pre>{ "All Models Info": [{ "id": { "model_id": "643ad1c3832687209ffbd3bd" }, "model_name": "DHD-LSTM", "model_description": "Implemented dynamic nodes from Dynamic Node Decomposition (DND) to the training input layer of LSTM", "model_version": "1.0.0", "average_max": "1993.39069", "average_mean": "17.67005", "average_min": "12.87009", "average_std": "9.495505" }], "Date & Time": "2023-04-26T13:26:22.877561" }</pre> <p>Download</p> Response Headers <pre>content-length: 377 content-type: application/json date: Wed, 26 Apr 2023 09:26:22 GRT server: uvicorn</pre> Response <table border="1"> <thead> <tr> <th>Code</th> <th>Description</th> <th>Links</th> </tr> </thead> <tbody> <tr> <td>200</td> <td>Successful Response</td> <td>No links</td> </tr> </tbody> </table> <p>Media type</p> <p>application/json</p> <p>Example Value Schema</p> <pre>"string"</pre> <p>GET /ml_model_info/{model_name} Get Model Info</p> | Code | Description | Links | 200 | Successful Response | No links |
| Code | Description | Links | | | | | |
| 200 | Successful Response | No links | | | | | |

Figure 4.8: Sample API Endpoint: ML Model Info (all) Request

Finally, the final API endpoints related to the 'Stocks Risks Profile', which outputs a list of stocks in alamSYS and their corresponding risk values based on value at risk (%), volatility (%), and drawdown (%). This API endpoint has two uses: users can use the /all path to see the risks profile information for all stocks, or they can use /stock_name, which requires a stock name and returns the risks information for that specific stock if it exists in the alamSYS, and a 'Stock not found' alert if it does not. Figure 4.9 depicts a sample execution, specifically for /all.

The screenshot shows a REST API interface for a 'Stocks Risks Profile' endpoint. At the top, a status bar indicates: 'Stocks Risks Profile This API endpoint outputs a list of the stocks included in the alamSYS and their corresponding risks values based on value at risk (%), volatility (%), and drawdown (%).'. Below this is a main panel titled 'Stock Risks Profile'.

The main panel includes:

- A 'GET /risk_info/all' button to 'Get All Stocks Stocks Risks Profile'.
- A 'Parameters' section with a note 'No parameters'.
- An 'Execute' button and a 'Clear' button.
- A 'Responses' section with tabs for 'Curl', 'Request URL', and 'Server response'.
- In the 'Server response' tab, there's a 'Code' table with a single row for '200 Response body' containing JSON data for two stocks (MSFT and TSLA) with their respective risk metrics.
- A 'Response headers' section showing standard HTTP headers like Content-Type, Content-Length, Date, and Server.
- A 'Responses' table with a single row for '200 Successful Response'.
- A 'Links' section with a note 'No links'.
- At the bottom, another 'GET /risk_info/{stock_code}' button to 'Get Stock Stocks Risk Profile'.

Figure 4.9: Sample API Endpoint: Stocks Risks Profile (all) Request

To go over the following risk values in greater detail. Furthermore, Table 4.1 summarizes the risk values for all stocks included in the alamSYS:

- Value at Risk (VaR) is a statistic that quantifies the extent of potential financial losses within a firm, portfolio, or position over a specific time period, and risk managers use VaR to measure and control the level of risk exposure (Kenton, 2023). The alamSYS calculated the value at risk based on the daily return percentage of each stock from January 03, 2000 to February 02, 2023, that is below 5%. In Table 4.1 it shows a negative VaR, which means that percentile amount is the expected returns below 5% is computed for

that values. For example, PSEI shows -1.88780% VaR which means that it has a 1.88780% chance to loss 5% of its value for the next day, while having 99.11220% chance of gaining above 5% on a daily basis. Take note that the negative symbol in VaR indicates the likelihood of losing in the position.

- (b) Volatility is a statistical indicator of the spread of returns for a specific security or market index. The higher the volatility, in most cases, the riskier the security (Hayes, 2023). It was measured in the alamSYS in terms of the volatility percentage based on standard deviation of daily returns of each stock from January 3, 2000 to February 2, 2023. The data presented in Table 4.1 shows that the LTG stock is the most volatile of the stocks included in the alamSYS, implying that investors or traders should keep a closer eye on LTG than other stocks with volatility ranging from around 1.3% to 7%.
- (c) Drawdown is useful for calculating the historical risk of various investments, comparing fund performance, and monitoring personal trading performance (Mitchell, 2023). It was measured in the alamSYS in terms of the drawdown percentage based on the maximum and minimum values of each stock from data beginning January 03, 2000 until February 02, 2023, essentially to measure the stock's performance during that time period. The overall drawdown values from the alamSYS stocks, as shown in Table 4.1, show a positive upside, but this also indicates the possibility of reversed risk, where a drawdown of 50%, for example, can also mean a potential loss of 50% of an investor's total capital.

Table 4.1: Summary of Risk Profile for Each Stock in the alamSYS

| STOCK | VALUE AT RISK (%) | VOLATILITY (%) | DRAWDOWN (%) |
|-------|-------------------|----------------|--------------|
| MEG | -5.36572 | 3.94997 | 57.24814 |
| JGS | -4.76236 | 3.36110 | 43.18404 |
| BDO | -3.30249 | 2.36405 | 39.67065 |
| FGEN | -3.81908 | 2.55925 | 30.60426 |
| ICT | -4.82705 | 3.52084 | 54.64501 |

Table 4.1 continued from previous page

| STOCK | VALUE AT RISK (%) | VOLATILITY (%) | DRAWDOWN (%) |
|--------------|-------------------|----------------|--------------|
| ALI | -4.39048 | 3.07049 | 56.02617 |
| SMC | -3.40367 | 2.38596 | 45.29878 |
| TEL | -3.69376 | 2.46002 | 44.65912 |
| GLO | -4.12004 | 3.09260 | 54.75806 |
| BLOOM | -5.98500 | 7.06155 | 100.00000 |
| RLC | -4.52936 | 3.41699 | 65.98291 |
| MER | -4.49859 | 3.25474 | 76.00003 |
| AC | -4.29065 | 2.79617 | 46.68883 |
| PGOLD | -3.11492 | 2.13482 | 25.79979 |
| LTG | -6.15322 | 31.22332 | 1667.80691 |
| MPI | -4.05584 | 3.49968 | 81.13252 |
| AP | -3.19764 | 2.18729 | 26.54088 |
| RRHI | -3.03206 | 2.05393 | 24.50967 |
| URC | -4.53239 | 3.19972 | 42.54262 |
| PSEI | -1.88780 | 1.31882 | 30.90366 |

alamAPI Use Case Diagram

Based on the API endpoints discussed in the previous section, the author has illustrated the use case of the alamAPI in this section. Furthermore, the use case diagram in Figure 4.10 adheres to the Unified Modeling Language (UML) definition of a use case diagram as published in Visual Paradigm's (n.d.) guides. *Note that this only pertains to the interaction of potential users in the alamAPI, and not to the whole alamSYS.*

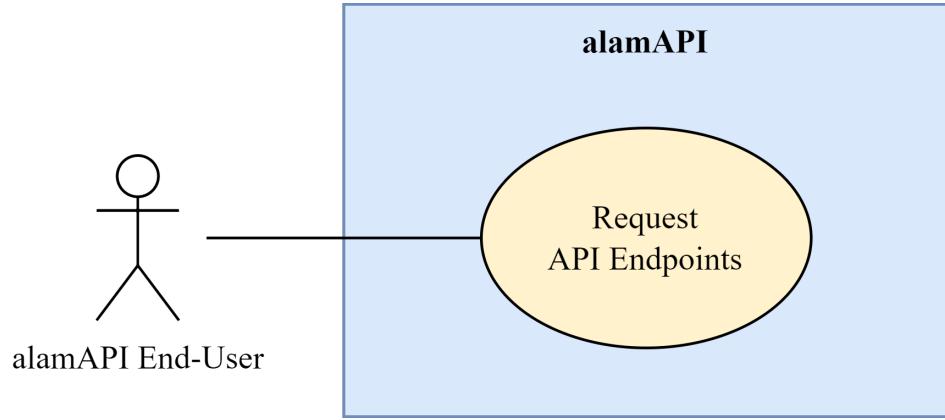


Figure 4.10: alamAPI End User UML Use Case Diagram

The use case diagram above demonstrates how easy and simple it is to use the alamAPI, as end users are only required to create requests from the listed API endpoints.

alamAPI Docker Management

The management and maintenance of the alamAPI container is also made possible through the utilization of Docker Desktop. Figure 4.11 depicts the tree of files and directories accessible from '/alamAPI/' of the alamAPI docker container.

```

.
├── API
│   ├── __init__.py
│   ├── __pycache__
│   │   ├── main.cpython-38.pyc
│   │   └── test.cpython-38.pyc
│   ├── main.py
│   └── routers
│       ├── __pycache__
│       │   ├── home.cpython-311.pyc
│       │   ├── ml_model_info.cpython-311.pyc
│       │   ├── stock_risks_profile.cpython-311.pyc
│       │   ├── stocks_info.cpython-311.pyc
│       │   ├── stocks_to_buy.cpython-311.pyc
│       │   └── stocks_to_sell.cpython-311.pyc
│       ├── home.py
│       ├── ml_model_info.py
│       ├── stock_risks_profile.py
│       ├── stocks_info.py
│       ├── stocks_to_buy.py
│       └── stocks_to_sell.py
└── DB_model
    ├── __init__.py
    ├── __pycache__
    │   ├── __init__.cpython-311.pyc
    │   ├── __init__.cpython-38.pyc
    │   └── models.cpython-311.pyc
    └── models.py
requirements.txt

```

6 directories, 23 files

Figure 4.11: alamAPI Docker Container Directory Tree

On the other hand, alamDB’s docker container management only displays alamDB status updates, whereas direct management employs known mongoDB command line interface commands. However, utilization of CLI commands through docker desktop can be time-consuming for system administrators or system managers, so it is best to use a GUI-based mongoDB management tool, as discussed in the succeeding section.

alamDB MongoDBCompass Management

The use of a GUI-based database management tool for alamDB is critical in maintaining the system’s ease of maintenance. As a result, the author recommends that future system administrators or maintainers use software such as MongoDBCompass, which is available for Windows users.

Utilization of the MongoDBCompass is straightforward and is only composed of these few steps:

- (a) First, indicate a new connection using the default URI: `mongodb://localhost:27017`.
- (b) Once the URI is initiated, press or tap the 'Connect' button to establish connection to the `alamDB`.
- (c) Finally, you have access to the `alamDB`. As shown in Figure 4.12 system users has the access to the following databases:
 - 1. `admin` - which currently have an empty collection of data
 - 2. `alamAPI_DB` - which corresponds to the database of the `alamSYS`
 - 3. `config` - which is also currently empty; and
 - 4. `local` - which contains the `startup_log` collection, and whose data is autogenerated by `mongoDB`.

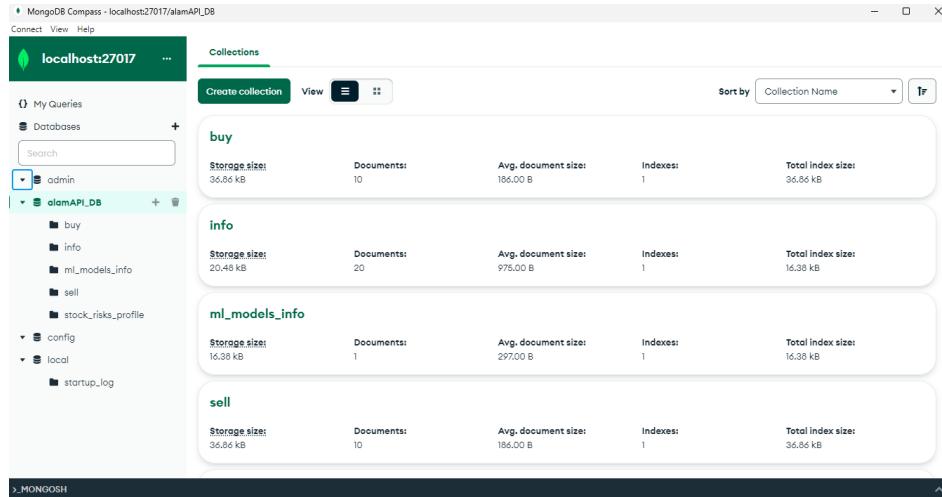


Figure 4.12: MongoDB Compass Database Management of `alamDB`

4.1.3 Documentation for `alamAPP`

The `alamAPP` is a mobile-based test application developed to demonstrate the use of the `alamSYS` to external internet-connected devices, such as a smartphone application. It was created with the Dart-based Flutter SDK. The user interface and code snippets of how the `alamSYS` was used in the `alamAPP` are shown in this section. Furthermore, the full source code for the `alamAPP` is available in the GitHub repository listed in Appendix A of this paper.

User Interface Showcase

This section displays the alamAPP's developed user interfaces. It should be noted that this application only highlights the alamSYS's two most important features, which are its ability to recommend which stocks to buy and which to sell. Additional enhancements to the application are discussed in Chapter 5 of this paper.

xxx

xxx

xxx

xxx

xxx

Overall, the figures depicting the alamAPP's user interface demonstrate that the alamAPI can be used by external applications by connecting via the internet.

Utilization of alamSYS in the alamAPP

The following code snippet written in Dart, was used in the alamAPP's source code to get the details of the stocks to buy from alamAPI. *Note that the [reserved.domain] should be changed to the domain address used on the alamAPI deployment.*

```
Future<List<Stocks>> stocksToBuy() async {
    var url = 'https://[reserved.domain]/stocks_to_buy/all';
    var response = await http.get(Uri.parse(url));
```

```

    var data = jsonDecode(response.body);
    List<Stocks> stocksToBuy = [];
    if (data['Stocks'] != null) {
        data['Stocks'].forEach((v) {
            stocksToBuy.add(new Stocks.fromJson(v));
        });
    }
    return stocksToBuy;
}

```

Meanwhile for the stocks to sell, the following code snippet was used:

```

Future<List<Stocks>> stocksToSell() async {
    var url = 'https://[reserved.domain]/stocks_to_sell/all';
    var response = await http.get(Uri.parse(url));
    var data = jsonDecode(response.body);
    List<Stocks> stocksToSell = [];
    if (data['Stocks'] != null) {
        data['Stocks'].forEach((v) {
            stocksToSell.add(new Stocks.fromJson(v));
        });
    }
    return stocksToSell;
}

```

4.1.4 Build and Deployment Guide

The build and deployment of the alamSYS docker compose are covered in this section. The discussion is divided into two sections. The first section discusses how to build and run the alamSYS docker compose. The second section discusses the current implementation's deployment using a low-cost tunneling service.

Building and Running Docker Compose of alamSYS

The alamSYS is deployed using docker compose containers, which is a docker tool that allows multiple containers to run concurrently. The docker compose

file, on the other hand, is specific to the alamSYS and runs three containers: alamPREPROCESSOR, alamAPI, and alamDB.

Assuming the server device already has docker installed and a copy of the alamSYS source code, which can be accessed via the link in Appendix A, the following '.env' files should be created and added to the alamSYS root folder.

The first '.env' file is the 'db.env' file, which contains the alamDB's environment variables. Wherein, the file contains the following details as indicated in the code snippet below.

```
# MongoDB
MONGO_INITDB_DATABASE="alamAPI_DB"
MONGO_HOST="mongo_db"
MONGO_PORT=27017

# Timezone
TZ="Asia/Hong_Kong"
```

And the other '.env' file is the 'preprocessor.env', which contains the alamPREPROCESSOR's environment variables. Wherein, the file contains the following details as indicated in the code snippet below. Note that the EOD_API_KEY indicated should be changed to the actual API KEY provided in the account of the user and should not be shared to anyone or publicly. Moreover, other than the API KEY, there is nothing to be changed.

```
# Change API KEY to actual API KEY value
EOD_API_KEY=000000000000

# MongoDB
MONGO_INITDB_DATABASE="alamAPI_DB"
MONGO_HOST="mongo_db"
MONGO_PORT=27017

# Timezone
TZ="Asia/Hong_Kong"
```

```
# ml_processor  
MODEL_PATH="model"
```

At the end the folder structure of the alamSYS, should look like the one indicated in Figure 4.13.

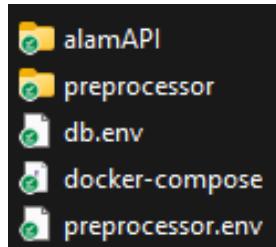


Figure 4.13: alamSYS Required Folder Structure

After creating and adding the necessary '.env' files to the alamSYS root folder. Using the following command, we can now build and run the docker compose command from the terminal:

```
docker-compose build && docker-compose up
```

The aforementioned command creates the alamSYS containers and then runs them all. Furthermore, in subsequent runs, simply run the second part of the command 'docker-compose up' to initialize the alamSYS containers. Only build the containers on the first initialization or if the alamSYS source code has changed.

Deployment using Network Tunnelling Services

In order to minimize the cost of alamSYS deployment, the developer decided to use the LocalXpose service, which is a network tunneling service. It is a reverse proxy that allows you to expose your localhost services to the internet and can be found at <https://localxpose.io/docs/>

LocalXpose's official documentation contains device-specific instructions. Whereas, the deployment instructions in this section are based on the Windows 11 operat-

ing system.

To begin, we must create a reserved domain in the LocalXpose Dashboard, which can be accessed via this link: <https://localxpose.io/dashboard/domain>. In the case of alamSYS, the reserved domain was set to the Asia Pacific (AP) region in order to maintain relatively strong data transfers. Following that, the user must download the binary program for their device's operating system; in the author's case, he downloaded the Windows 64-bit binary file.

The user should launch the command prompt (cmd) from the directory containing the downloaded file. It should be noted that, as of the writing of this paper, the binary file only works with cmd, and using powershell or windows terminal causes errors. To run the binary application loclx, enter the following command into cmd:

```
loclx.exe account login
```

The LocalXPose application would then prompt the user for an access token, which can be obtained at <https://localxpose.io/dashboard/access>. Following that, the user should enter the following command:

```
loclx tunnel http --reserved-domain [reserved.domain] --to  
8000
```

The last command launches the localx tunnel command in http mode with a reserved domain. And the value 8000 at the end indicates that the reserved domain will expose port 8000. This corresponds to the alamAPI's network port, making the alamAPI accessible via any network with an internet connection. It should also be noted that the user should replace '[reserved.domain]' with the actual reserved domain they created in the LocalXpose Dashboard. Finally, the application will now show the status of the connection.

4.2 alamSYS System Tests Results and Discussions

With the development of alamSYS, we must ensure that all of its components are functioning properly. This section focuses on the system's performance while idle and under load, as well as the API's and database's ability to handle multiple requests at a time.

The Table 4.2 shows the CPU and memory utilization of each of the alamSYS components whenever it is not processing any information. Wherein, the data presented below is gathered by logging the system utilization of alamSYS within an hour.

Table 4.2: Idle System Average Resource Usage Statistics

| | alamAPI | alamDB | alamPREPROCESSOR |
|--------------------------|-----------|------------|------------------|
| CPU Utilization (%) | 0.168125 | 0.254313 | 0.009769 |
| Memory Utilization (MiB) | 45.718311 | 166.775377 | 312.798300 |

From the table above, it is shown that the alamSYS as a whole only utilizes 0.432207% of the total CPU power on average when it is on idle. Wherein, the bulk of the CPU power is being used by the database at 0.254313% which is 58.84% of the total average CPU utilization of the alamSYS.

This result actually shows promising idle performance, as an Ubuntu system's normal idle CPU utilization is less than 10%. However, it should be noted that the alamSYS is not completely idle because background tasks such as scheduling, mongoDB processes, and the API must remain active in order to respond to any API queries.

Furthermore, the lower average CPU utilization for alamAPI and alamPREPROCESSOR could be attributed to the linux distribution used as their base image, as previously discussed on 'Docker-Compose Layer Diagram' of Section 3.2.

The CPU utilization for the alamPREPROCESSOR, in particular, was lower than we would have expected given that it is running a schedule checker every second. This indicate that the schedule library utilizes an efficient way to check for schedules. However, it may not be the case for its memory utilization, which is discussed further below.

Moving on, the alamSYS's memory utilization shows that it uses 525.291988 Mebibytes (MiB) or 550.808572 Megabytes (MB) on average. The memory utilization of alamPREPROCESSOR accounts for the majority (59.55%). This demonstrates that, despite having the lowest CPU utilization, the alamPREPROCESSOR consumes more memory than the combination of the alamAPI and alamDB.

Again, as previously discussed, the alamPREPROCESSOR's high average memory utilization is due to the background schedule checking, which in this case is programmed to use more memory than CPU power.

This average utilization result implies that the alamSYS may be able to be deployed on devices with lower specifications. A Raspberry Pi 4, which has a quad core CPU and at least 2GB RAM, is one example (Zwetsloot, 2019). However, idle performance only shows the minimum CPU and memory utilization of the alamSYS components and does not provide a complete picture of its utilization, particularly under load. As a result, we must investigate the system's CPU and memory utilization while under load.

The internal load averages system utilization of the alamSYS' preprocessor is shown in the following table.

Table 4.3: Internal Load Average Resource Usage Statistics

| | Data Collector | Data Processor | PREPROCESSOR (Data Collector & Data Processor) | alamSYS |
|---|----------------|----------------|--|-----------|
| Failure Rate (%) | 0 | 0 | | 0 |
| Success Rate (%) | 100 | 100 | | 100 |
| Average Runtime (s) | 41.72398 | 8.38061 | | 48.30466 |
| Average CPU Utilization (%) | 11.40659 | 92.71117 | | 20.03138 |
| Average Memory Utilization (MiB) | 3.64200 | 57.09545 | | 794.29436 |
| Average Network Utilization (Mb) | 232.73640 | 154 | | 77.27655 |

Before exploring into the results shown in Table 4.3, it is critical to first establish a context for how the data was gathered. The data in the above table was collected while the alamSYS, specifically the alamPREPROCESSOR, was subjected to a stress test load of 100 consecutive data collection and processing. Also, the data of average utilization as indicated for each column are independently gathered from each other.

Based on the table above, each component was capable of processing 100 consecutive processes without failure. This means that the alamSYS, specifically the alamPREPROCESSOR, can run at least 100 times in a row without failing. Also, keep in mind that the alamPREPROCESSOR only collects and processes data once per day, and the developer has included an option for system users or maintainers to manually rerun these processes in cases where the alamPREPROCESSOR fails - for example, due to a lost or slow internet connection, a power

outage, and so on.

Meanwhile, the data processor's average runtime is faster than the data collector's, and it runs on an average of 48.30466 seconds. This was already expected because the data collector needed to connect to the internet and was thus constrained by internet speed. Whereas the average speed during the course of this test was 52.98Mbps, this could imply that the data collector's runtime may be slower or faster depending on the internet speed.

Furthermore, the data processor's average runtime was surprisingly fast given that it needed to apply DMD-LSTM to a total of 20 stocks and calculate the position using the ALMACD given a total of 205 data points. Furthermore, looking at its CPU and memory utilization, it can be seen that it uses more than the data collector, with approximately 87.68% more average CPU power used and approximately 93.63% more average memory used.

In line with the CPU and memory utilization, the average CPU utilization of the alamPREPROCESSOR on load is 99.95% higher than when it is idle. And it uses 60.62% more memory on average.

Lastly, looking at the network utilization of the alamPREPROCESSOR, we can see that the data collector used the most network bandwidth, as expected. However, it may be surprising to see that the data processor uses the network when it should not because it does not need to process or collect data from the internet, but this is still within expectations because network bandwidth utilization also accounts for local network utilization. The data processor of alamPREPROCESSOR, in particular, uses the local network to connect to and update the new data in the alamDB.

The following tables show the system's deployment load results. Specifically, to access the buy and sell collections, as shown in Tables 4.4 and 4.5. Additionally, to have a background on the test that results in the following outcomes. It should

be noted that the test consisted of three instances of the same tester application running on ten different computers. Where each application requests 10, 100, 1000 buy and sell data from the alamDB via the alamAPI, which is tunneled over the internet for server access outside the university's local network using LocalXpose services.

Table 4.4: Deployment Load Test Results (Buy Requests)

| | Number or Requests | | |
|------------------------------------|--------------------|------------|-------------|
| | 10 | 100 | 1000 |
| Success Rate (%) | 100 | 100 | 100 |
| Average Processing Time (s) | 11.905222 | 139.618550 | 1159.773569 |

Table 4.5: Deployment Load Test Results (Sell Requests)

| | Number or Requests | | |
|------------------------------------|--------------------|------------|-------------|
| | 10 | 100 | 1000 |
| Success Rate (%) | 100 | 100 | 100 |
| Average Processing Time (s) | 13.384126 | 130.119867 | 1642.995011 |

As per the tables above, the alamSYS was able to handle all consecutive and simultaneous requests from all external devices. It was able to handle 71,000 requests in approximately one hour and 20 minutes.

Furthermore, the data shows that a request is processed in 1.34 seconds on average. Wherein, the fastest processing time from the data collected was 0.93 seconds. This result is within the expected response time of APIs, where good is defined as 0.1 to one second, and any time between one and two seconds is acceptable, as long as it does not exceed two seconds, which users may perceive

as an interruption in the process (Juviler, 2022).

It is also worth noting that in a separate internal test for the alamSYS, the alamAPI only takes on average 0.0094 seconds to send its response. And that the additional delay in response time on deployment is caused by network-related factors, such as the time it takes the tunneling service to accept the query and send back the system's response. In fact the network related delay contributes 99.30% of the average deployment response time. Hence, it might be useful to deploy the system in a network that further minimizes this additional delay in the response time.

Meanwhile Table 4.6 shows the average CPU and memory utilization of alamAPI and alamDB as the load testing as stated above, was being processed.

Table 4.6: CPU and Memory Utilization Statistics of alamAPI and alamDB Under Deployment Load Testing

| | alamAPI | alamDB |
|--------------------------|-----------|------------|
| CPU Utilization (%) | 17.725949 | 1.070133 |
| Memory Utilization (MiB) | 44.620837 | 129.273305 |

To further visualize the system utilization over time, Figure 4.14 shows the CPU utilization of alamAPI and alamDB over time. And Figure 4.15 shows the memory utilization of alamAPI and alamDB.

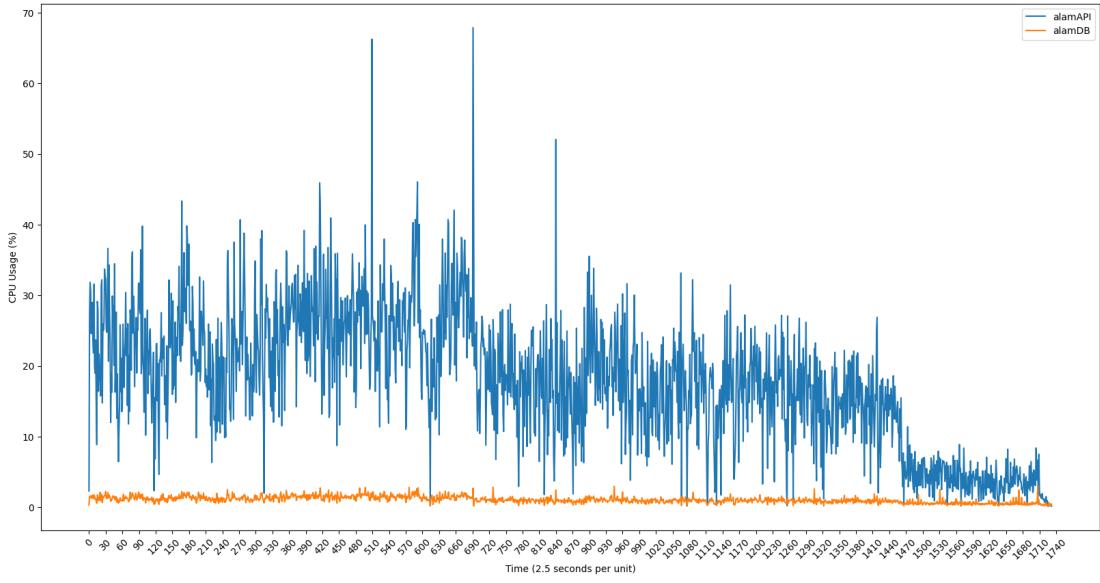


Figure 4.14: Deployment Load CPU Utilization of alamAPI and alamDB Over Time

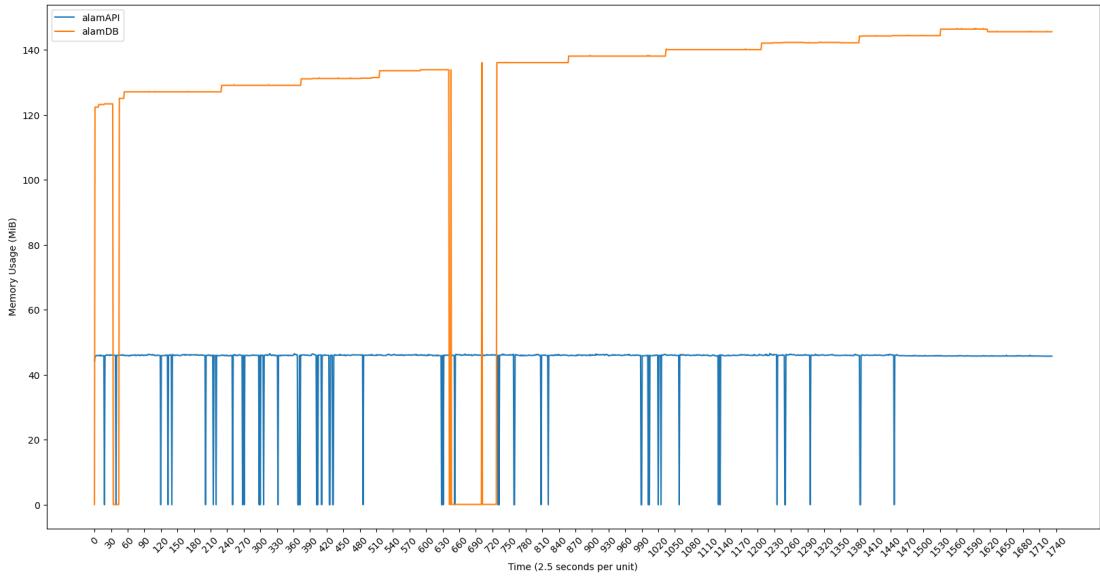


Figure 4.15: Deployment Load Memory Usage of alamAPI and alamDB Over Time

Based on the table and figures above, processing all requests consumes only 17.73% and 1.07% of the CPU for alamAPI and alamDB, respectively. In terms of memory usage, they are 44.62 MiB and 129.27 MiB, respectively.

Furthermore, at around 1740 time units, the CPU utilization for alamAPI was shown to be reduced. This is due to the fact that most tester applications have already completed processing all of the requests that they are programmed to run, reducing the load on the server by half.

4.3 DMD-LSTM Model Results and Discussions

This section presents and discusses the Deep Learning Model's training, testing, and cross-validation results.

In Table 4.7 the training error metrics are shown for each of the window sizes tested.

Table 4.7: DMD-LSTM Training Error Metrics Scores
for Different Window Sizes

| Error Metrics | Window Sizes | | | |
|---------------|-----------------|----------|----------|----------|
| | 5 | 10 | 15 | 20 |
| MSE | 0.000037 | 0.787877 | 0.006917 | 0.057851 |
| RMSE | 0.006106 | 0.887624 | 0.083166 | 0.240522 |
| MAE | 0.004175 | 0.755407 | 0.067645 | 0.202746 |
| MAPE | 0.000001 | 0.000194 | 0.000017 | 0.000053 |

Where it is observed that the best performing model based on having the lowest MAPE score is the DMD-LSTM with a window size of 5. Moreover, we can see the differences from each MAPE score for each window size in the Figure 4.16 shown below.

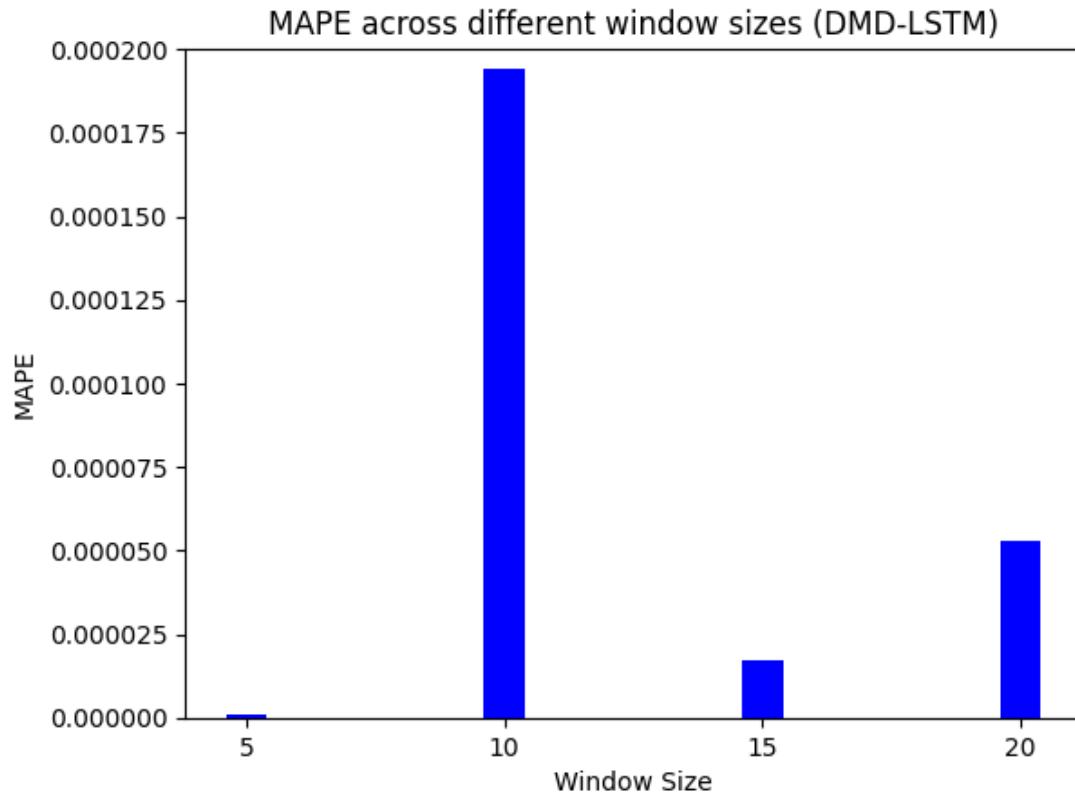


Figure 4.16: Comparison of MAPE Scores for DMD-LSTM Model Training Across Different Window Sizes

The figure above also shows that the MAPE score for window sizes 15 and 20 is higher than the MAPE score for window size 10. MAPE score increases from window size 15 to size 20, indicating that increasing window size may result in a lower performing model.

Furthermore, as previously stated, the window size of 5 results in the best MAPE score being the lowest. Where it outperforms the worst performing model (DMD-LSTM with window size 10) by 0.000193 units. As illustrated clearly in Figure 4.16.

Knowing that the DMD-LSTM model performs as expected based on the training data scores, it is critical that we also examine the training data results from

a baseline LSTM. The baseline LSTM is, as the name implies, a simple LSTM model lacking the DMD component. The table below shows the results of the baseline LSTM training.

Table 4.8: Baseline LSTM Training Error Metrics Scores for Different Window Sizes

| Error Metrics | Window Sizes | | | |
|---------------|--------------|-----------------|-------------|------------|
| | 5 | 10 | 15 | 20 |
| MSE | 2912.840703 | 191.935882 | 1118.183283 | 706.136814 |
| RMSE | 53.970739 | 13.854093 | 33.439248 | 26.573235 |
| MAE | 35.301888 | 9.480864 | 22.099720 | 18.285352 |
| MAPE | 0.009618 | 0.002527 | 0.006024 | 0.005004 |

According to the table above, the baseline LSTM with window size 10 performs the best, with the lowest MAPE score of 0.002527 when compared to the other baseline LSTM models.

However, the DMD-LSTM model with window size 5 outperforms it by 0.002526. As a result, the alamSYS makes use of the DMD-LSTM model, specifically the one with a window size of 5. Where from now on, the DMD-LSTM model refers to the DMD-LSTM model with a window size of 5.

Nonetheless, the DMD-LSTM model's performance is limited to the training dataset from PSEI, and it must be cross-validated using data from other stocks, which includes the PSEI validation dataset. The results of this cross-validation is presented in Table 4.9. It should also be noted that cross-validation uses logarithmic normalization as a data preprocessing technique to make the dataset more normal, which aids in analyzing the model's performance with the given dataset. Normalization techniques, in particular, allow for closer variation within the forecasted data. (S.Gopal Krishna Patro, 2015).

Table 4.9: DMD-LSTM Cross-Validation Error Metrics Scores

| Stocks | MSE | RMSE | MAE | MAPE |
|---------------|------------|-------------|------------|---------------------|
| PSEI | 0.00002 | 0.00419 | 0.00328 | 1.510000e-03 |
| AC | 0.00236 | 0.04856 | 0.03414 | 6.110000e-03 |
| ALI | 0.00255 | 0.05054 | 0.03645 | 1.597000e-02 |
| AP | 0.00129 | 0.03596 | 0.02515 | 9.220000e-03 |
| BDO | 0.00160 | 0.03999 | 0.02799 | 7.250000e-03 |
| BLOOM | 0.01883 | 0.13721 | 0.06901 | 1.052898e+12 |
| FGEN | 0.00224 | 0.04733 | 0.03265 | 1.197000e-02 |
| GLO | 0.00211 | 0.04595 | 0.03149 | 4.680000e-03 |
| ICT | 0.00335 | 0.05785 | 0.03731 | 3.005818e+11 |
| JGS | 0.00331 | 0.05752 | 0.03992 | 2.009923e+11 |
| LTC | 0.01567 | 0.12518 | 0.05858 | 3.583335e+12 |
| MEG | 0.00431 | 0.06565 | 0.04422 | 1.393042e+11 |
| MER | 0.00326 | 0.05708 | 0.03770 | 9.170000e-03 |
| MPI | 0.00273 | 0.05230 | 0.03390 | 2.497000e-02 |
| PGOLD | 0.00149 | 0.03865 | 0.02818 | 7.880000e-03 |
| RLC | 0.00338 | 0.05817 | 0.03978 | 6.922000e-02 |
| RRHI | 0.00131 | 0.03618 | 0.02699 | 6.390000e-03 |
| SMC | 0.00137 | 0.03702 | 0.02317 | 5.690000e-03 |
| TEL | 0.00178 | 0.04214 | 0.03002 | 4.240000e-03 |
| URC | 0.00297 | 0.05447 | 0.03742 | 1.798000e-02 |

As shown in the table above, the chosen DMD-LSTM model performs well across all other stocks, demonstrating that the model is not overfitted to the training dataset. This score additionally suggests that the model works with non-training data.

The figures below show a 100-day worth of predicted prices versus actual prices to better visualize the performance of the DMD-LSTM model for each stock.

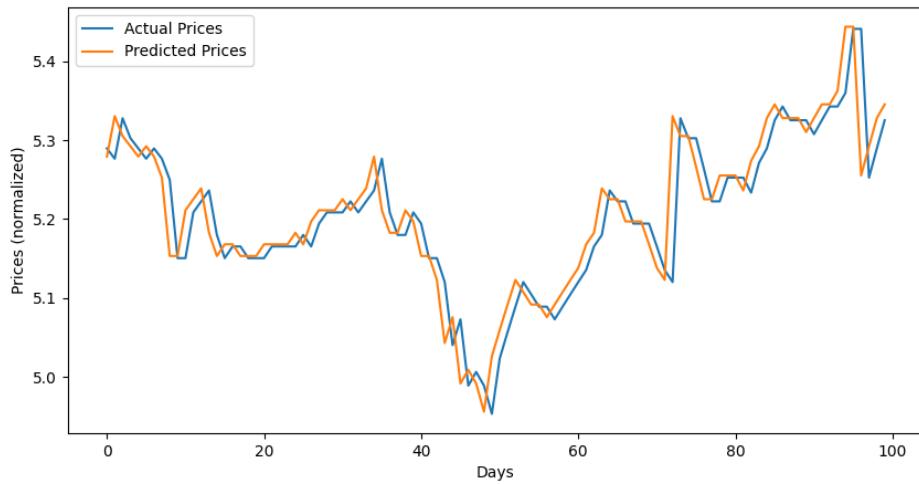


Figure 4.17: Actual vs Predicted Prices on AC for 100 days

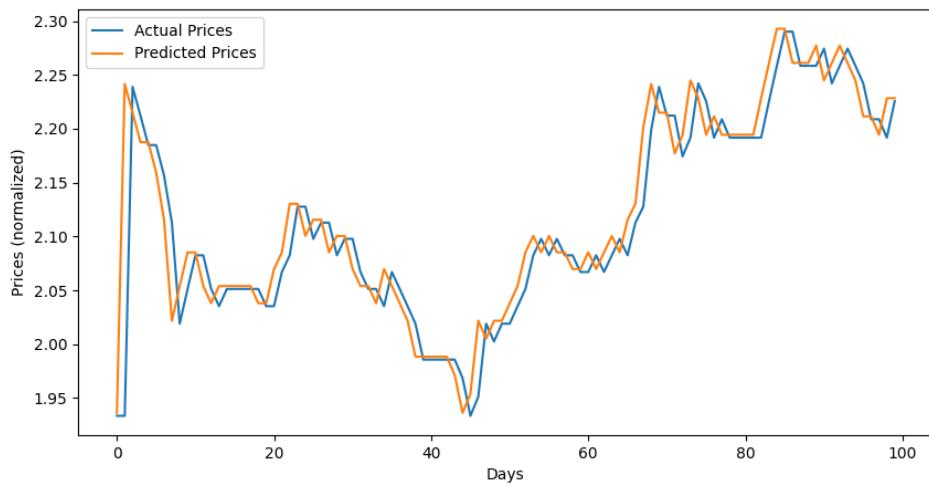


Figure 4.18: Actual vs Predicted Prices for ALI over 100 days

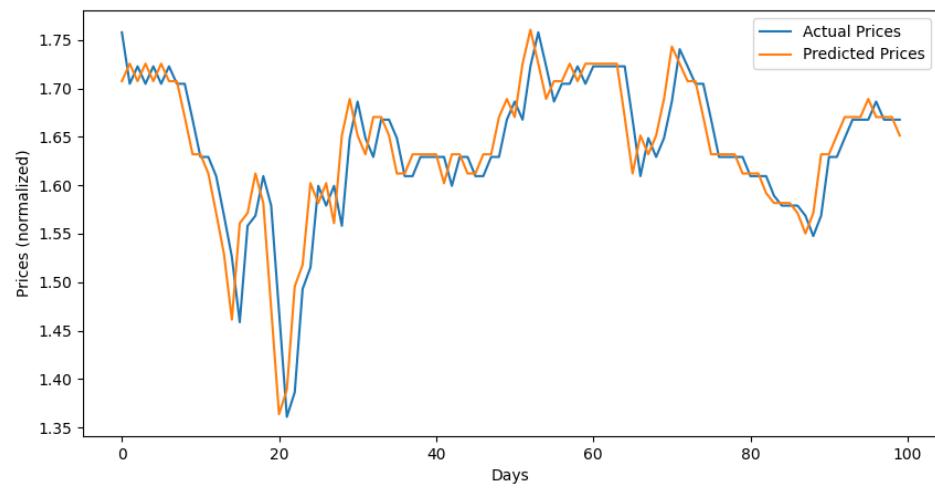


Figure 4.19: Actual vs Predicted Prices for AP over 100 days

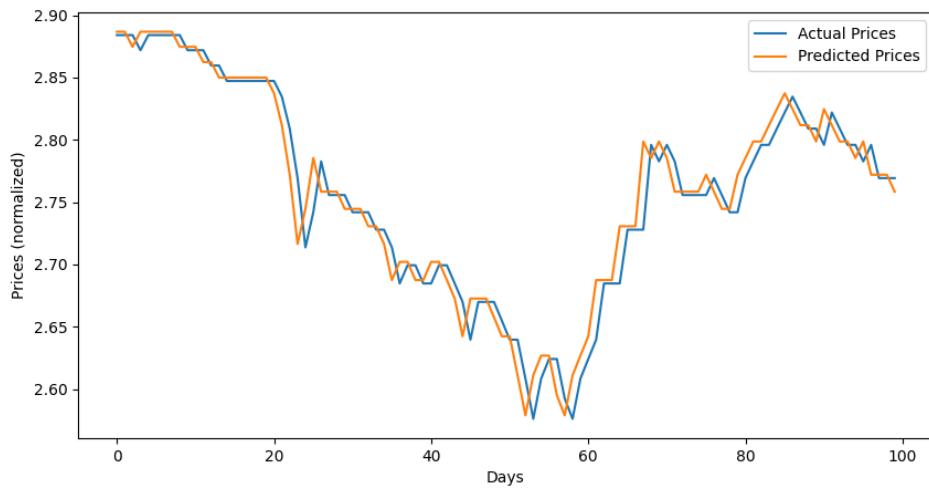


Figure 4.20: Actual vs Predicted Prices for BDO over 100 days

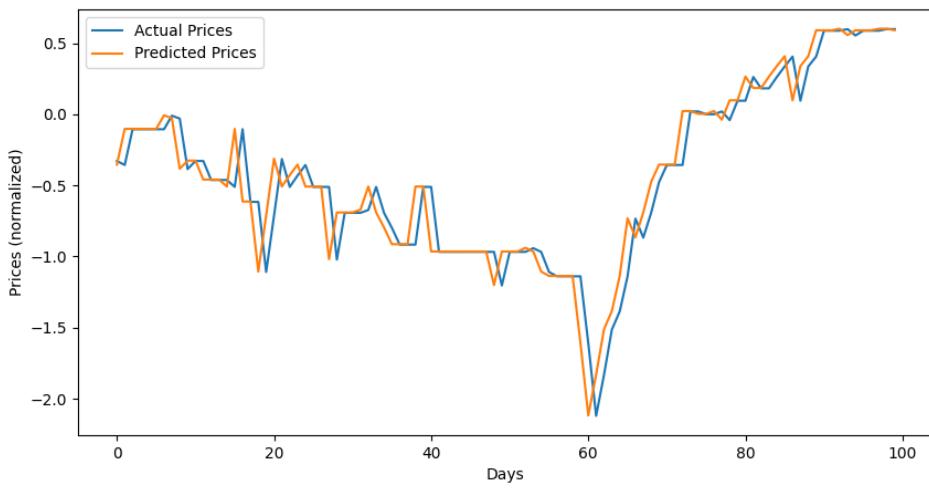


Figure 4.21: Actual vs Predicted Prices for BLOOM over 100 days

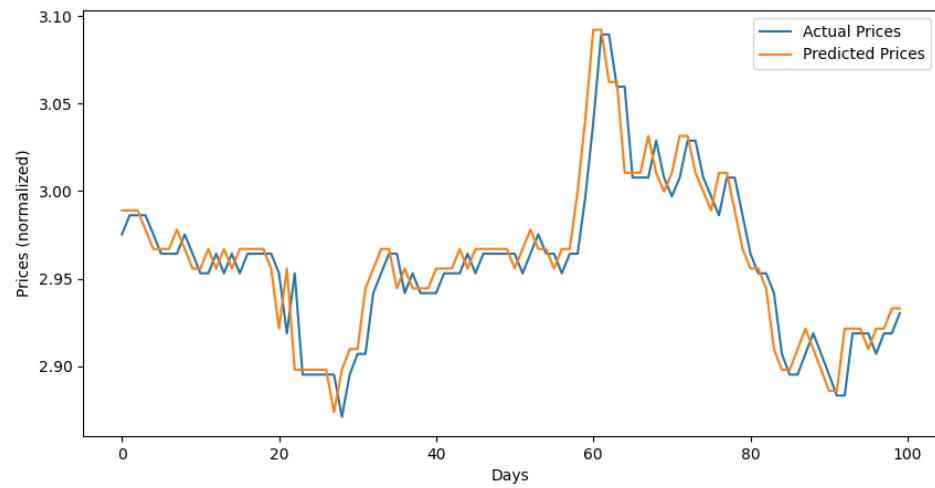


Figure 4.22: Actual vs Predicted Prices for FGEN over 100 days

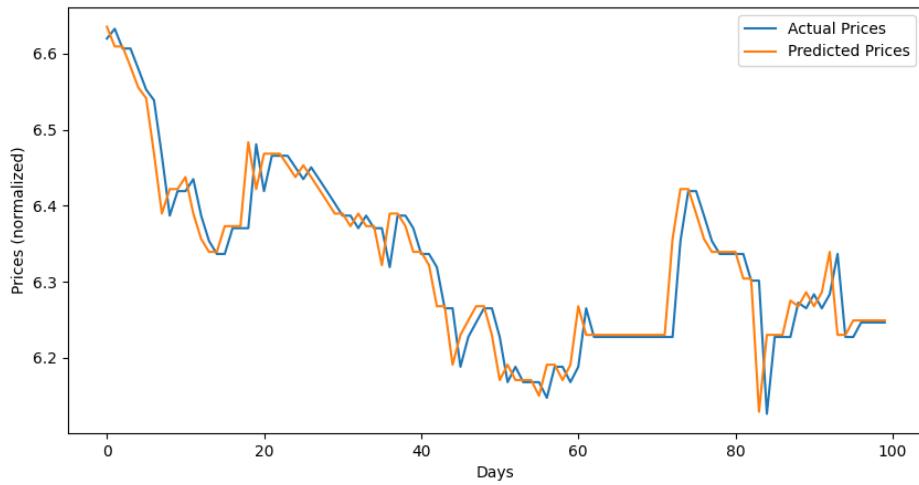


Figure 4.23: Actual vs Predicted Prices for GLO over 100 days

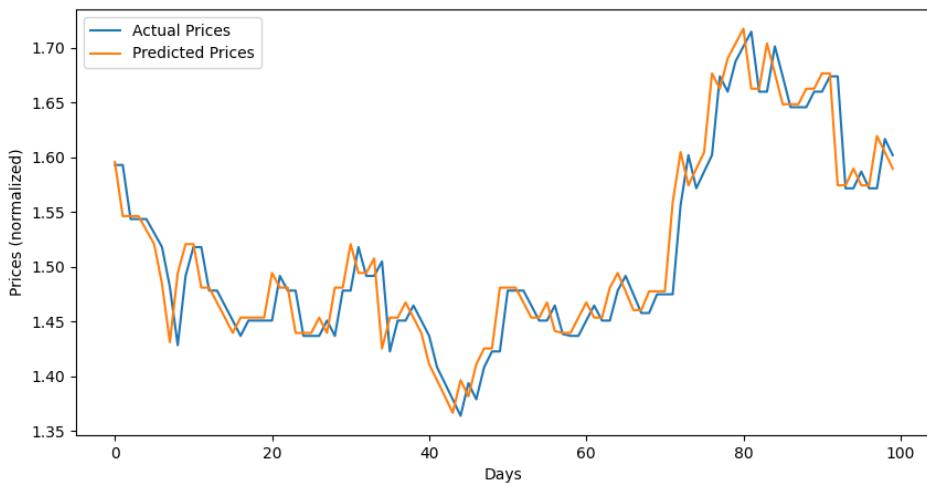


Figure 4.24: Actual vs Predicted Prices for ICT over 100 days

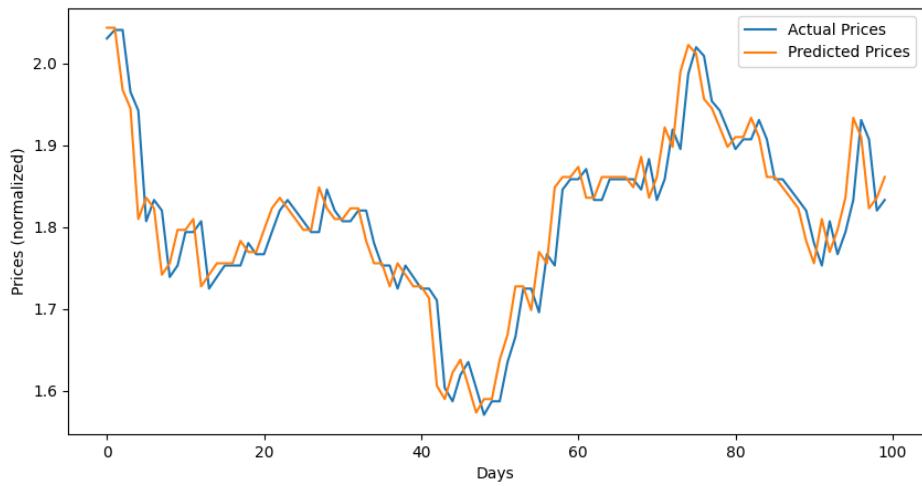


Figure 4.25: Actual vs Predicted Prices on JGS for 100 days

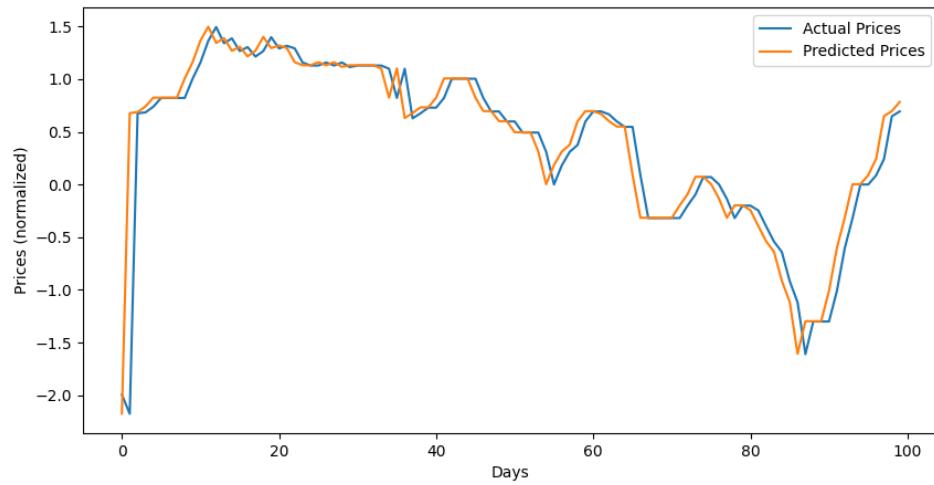


Figure 4.26: Actual vs Predicted Prices on LTG for 100 days

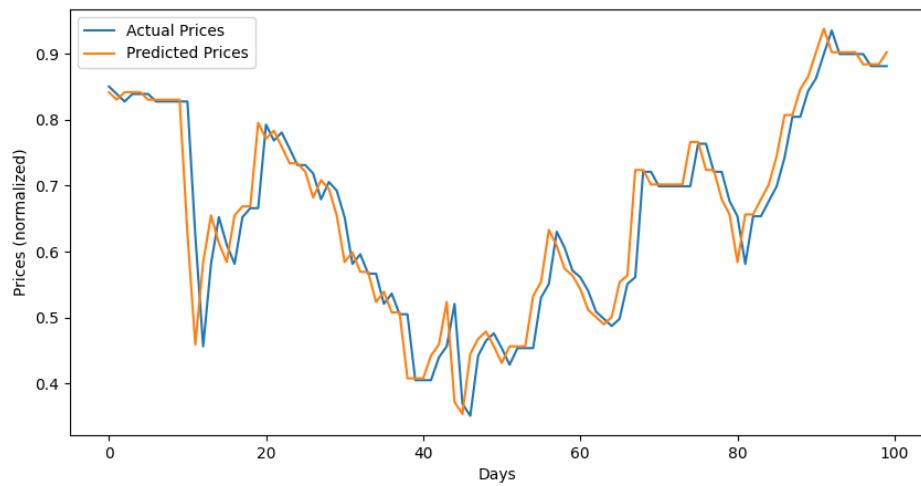


Figure 4.27: Actual vs Predicted Prices on MEG for 100 days

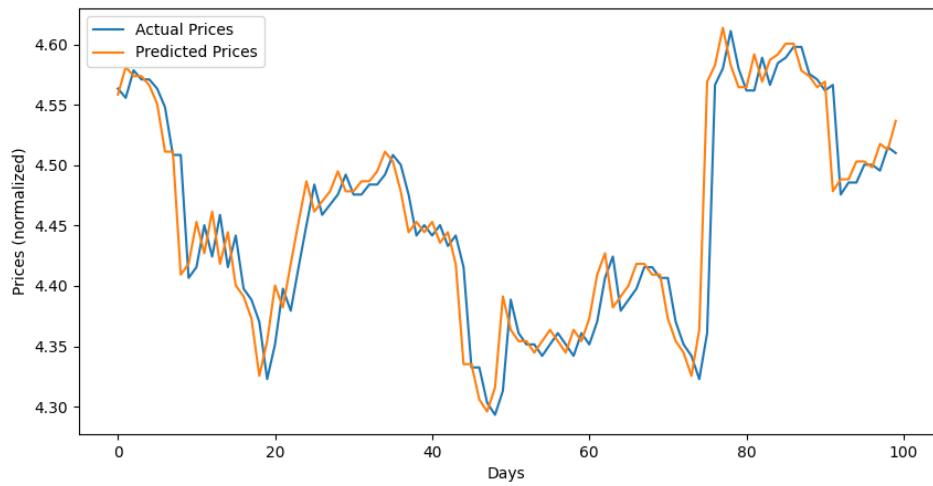


Figure 4.28: Actual vs Predicted Prices on MER for 100 days

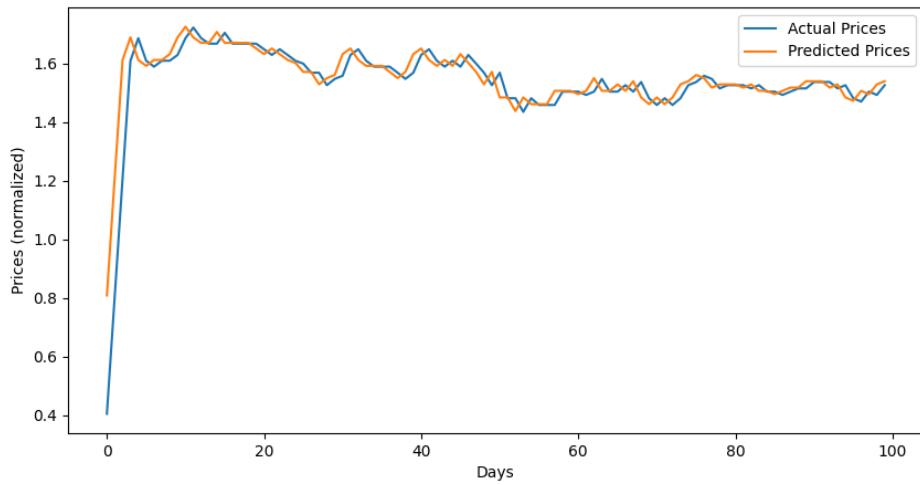


Figure 4.29: Actual vs Predicted Prices on MPI for 100 days

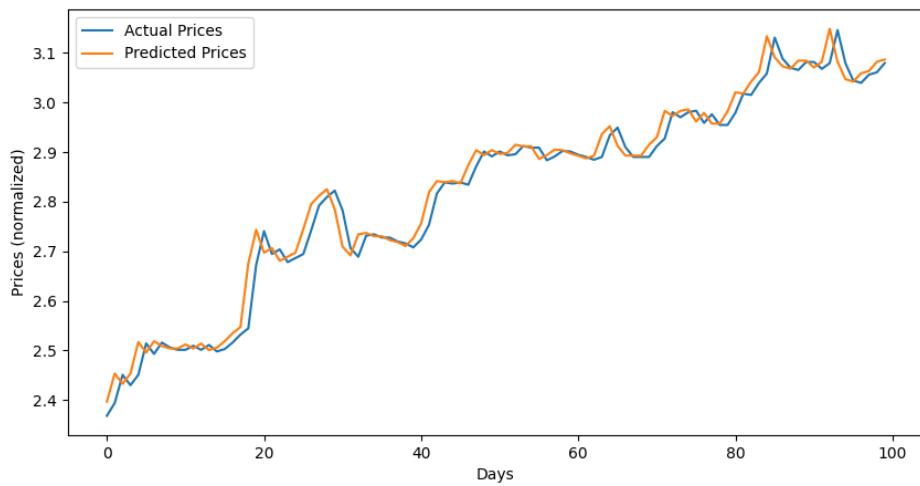


Figure 4.30: Actual vs Predicted Prices on PGOLD for 100 days

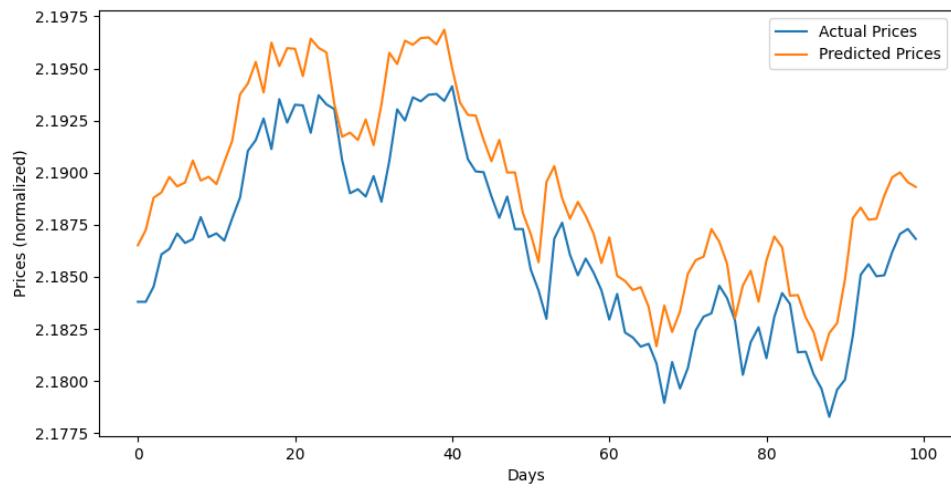


Figure 4.31: Actual vs Predicted Prices on PSEI for 100 days

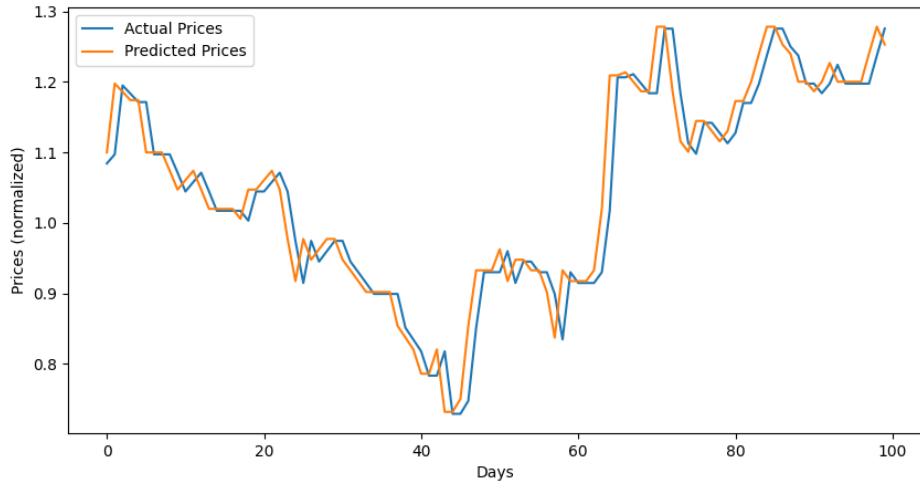


Figure 4.32: Actual vs Predicted Prices on RLC for 100 days

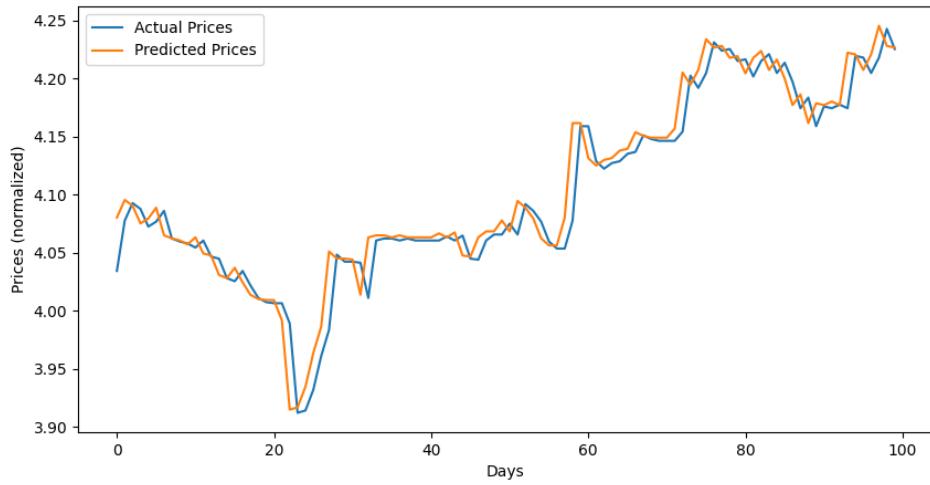


Figure 4.33: Actual vs Predicted Prices on RRHI for 100 days

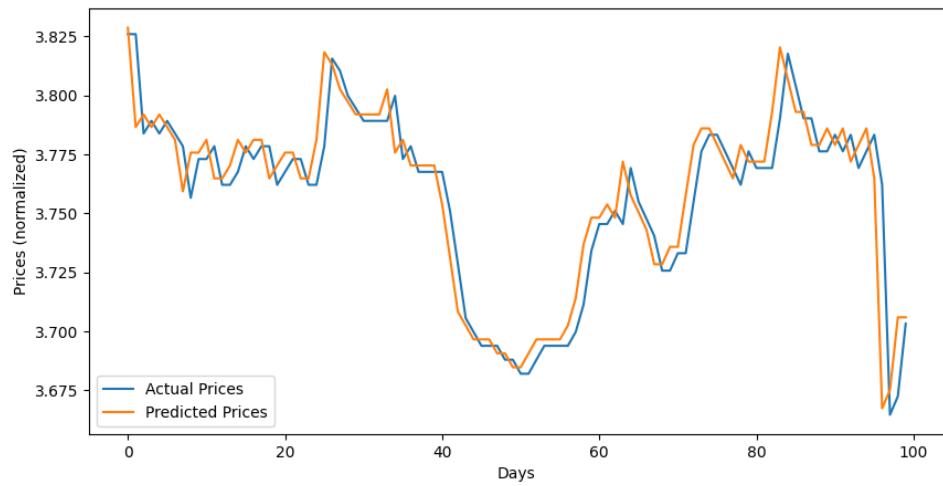


Figure 4.34: Actual vs Predicted Prices on SMC for 100 days

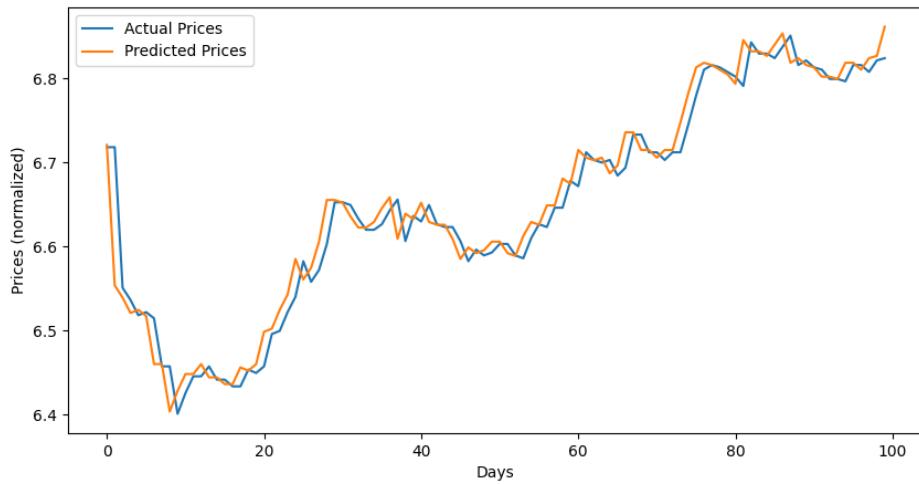


Figure 4.35: Actual vs Predicted Prices on TEL for 100 days

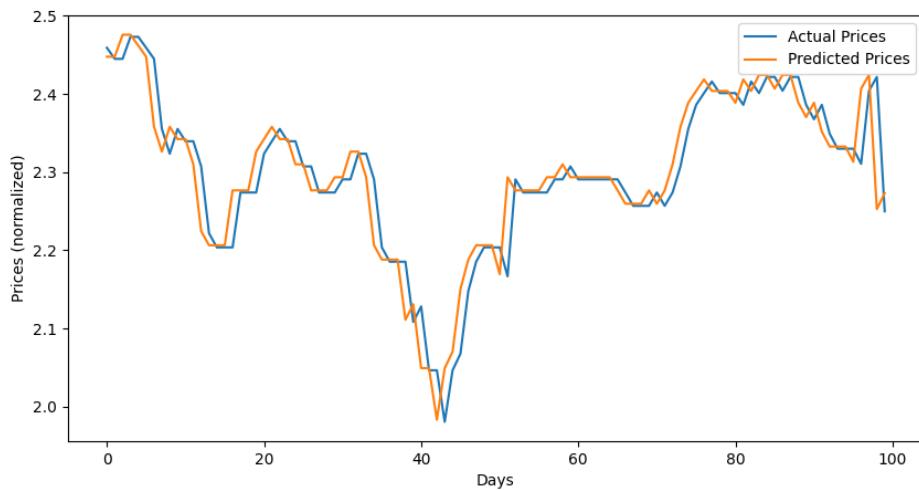


Figure 4.36: Actual vs Predicted Prices on URC for 100 days

The figures above show that the predicted prices follow the actual price trend. In addition, the discrepancy between predicted and actual prices is relatively small, as evidenced by the error metrics scores shown in Table 4.9.

However, the MAPE scores for BLOOM, ICT, JGS, LTG, and MEG range from ten billion to hundred billion. This outlier in the data is, fortunately, just the result of the applied logarithmic normalization, where some of the data in the datasets of the aforementioned stocks are in the negative range, that influence the calculation of the MAPE scores using the scikit-learn library. Because this library handles the calculation of the MAPE scores, there is no way to fix this bug. Moreover, if we take a look at the graphs of the 100 days prediction versus the actual for the aforementioned stocks in Figures 4.21, 4.24, 4.25, 4.26, and 4.27, respectively, it can still be observed that the model performs well on these stocks.

Not to mention that the other error metrics used show the same performance levels across the different stocks when the DMD-LSTM model is utilized. Meanwhile when the data normalization is removed, the MAPE scores for BLOOM, ICT, JGS, LTG, and MEG become 0.068108, 0.037207, 0.039754, 0.057332, and 0.044411 units, respectively.

Another observation from the graphs comparing actual and predicted prices over 100 days is that the predicted values appear to be higher than the actual prices. This indicates the possibility of loss because the model overestimates its prediction.

The successive predictions for the following day and up to ten days were tested using the price data from PSEI in order to make the system's predictions more useful for actual utilization. Table 4.10 shows the MAPE scores for the successive predictions of the DMD-LSTM for each days.

Table 4.10: DMD-LSTM Successive Predictions

| Successive Days Predicted | Actual and Predicted Data Ratio | MAPE Score |
|---------------------------|---------------------------------|------------|
| 1 | 100% | 0.00973 |
| 2 | 80% | 0.13403 |
| 3 | 60% | 0.15782 |
| 4 | 40% | 0.15646 |
| 5 | 20% | 0.13910 |
| 6 | 0% | 0.12494 |
| 7 | -20% | 0.11283 |
| 8 | -40% | 0.10014 |
| 9 | -60% | 0.08914 |
| 10 | -100% | 0.08976 |

From the table above it must be noted that the ratio values highlighted in red is to demonstrate that, despite the fact that negative ratio values shouldn't exist, doing so simply indicates that the data used to forecast the subsequent price data was overlapping by 2 to 5 times, depending on the ratio, and no longer used any actual data.

Moreover, in the integration of the DMD-LSTM model to the alamSYS, the 5 days successive predictions was utilized. Where it is shown from the Table 4.10 that it still performs well, even if the actual and predicted data ratio is only at 20%. This is also to limit the effect of stock market volatility that might affect the accuracy of the successive predictions of the model.

However, it can also be observed that the MAPE scores for successive days with zero to negative actual and predicted data ratio outperforms the MAPE scores from successive days 2 to 5 as illustrated in Figure 4.37, shown below.

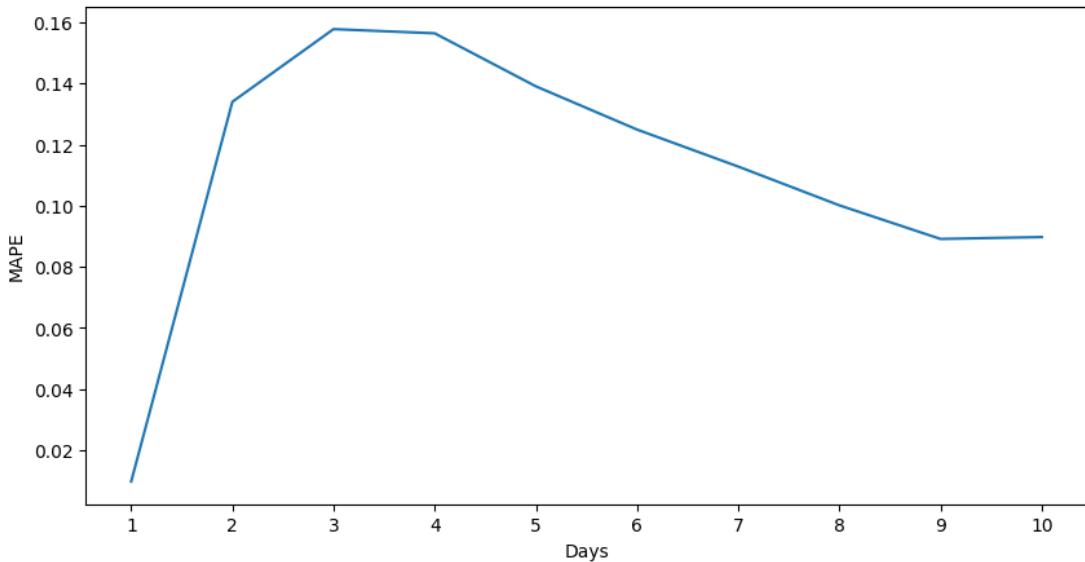


Figure 4.37: MAPE Scores for 1 to 10 (Days) Successive Predictions

Yet, since doing so might result in a poor generalization of data, they were not chosen to be the maximum consecutive days of predictions to be integrated in the alamSYS. As a matter of fact, it could be argued that these data's MAPE scores are overfitted, rendering them unreliable. On the contrary, it might also imply that the model maintains its accuracy for a longer time, even if the majority of the data used are those produced by the model itself. This could be a good thing, and may be attributed to the use of the dynamic modes, as first suggested in the study of Mann and Kutz (2015). In light of these considerations, additional testing is required to establish which of the two claims is true.

Overall, the results from the model training, evaluation, and cross-validation shows that the DMD-LSTM model developed in this special problem performs on par with the other studies that utilizes dynamic modes, as mentioned in Chapter 2 of this paper.

4.4 ALMACD Results and Discussions

The ability to predict consecutive days in the stock market is useless without a trading strategy - which allows risk mitigation and increases the probability of

positive returns over time. Trading strategies, in particular, are based on a pre-defined set of rules and criteria that are used to determine when to buy and sell stocks. (Hayes, 2022).

A variety of algorithmic trading, on the other hand, refers to the use of mathematical and computational techniques to determine the best position to take for a specific set of stocks. Additionally, the possibility of loss due to the influence of human emotion is eliminated. (WallStreetMojo, n.d.).

Whereas, the author used the Arnaud Legoux Moving Average Convergence and Divergence (ALMACD) trading strategy in this special problem and integrated it into the alamSYS as the system's internal trading algorithm. ALAMCD uses predicted prices for the next 5 days, as well as 200 days of actual stock price data, to track the signals and output a simple flag indicating whether to buy or sell that stock at that time.

The compounded expected return after return backtesting is provided for each stock in Table 4.11 using the optimized parameters for the fast and slow ALMA. This was done to validate the potential returns for all stocks, not just the PSEI, from which the best ALMA parameters were derived.

Table 4.11: Optimal Alma Parameters Validation Results

| Stock | Compounded Expected Return |
|--------------|-----------------------------------|
| PSEI | 113966.8500 |
| AC | 20893.1914 |
| ALI | 1072.1418 |
| AP | 690.7100 |
| BDO | 2541.9970 |
| BLOOM | 495.4600 |
| FGEN | 581.0804 |
| GLO | 60538.0035 |
| ICT | 2815.6103 |

Table 4.11 continued from previous page

| Stock | Compounded Expected Return |
|--------------|-----------------------------------|
| JGS | 1569.8650 |
| LTG | 397.2854 |
| MEG | 149.2233 |
| MER | 8586.0306 |
| MPI | 146.0200 |
| PGOLD | 721.2700 |
| RLC | 649.4767 |
| RRHI | 1050.7000 |
| SMC | 2557.0770 |
| TEL | 72070.5000 |
| URC | 3207.5394 |

Based on the table of expected returns above, all stocks are expected to return a positive yield over time when these optimal ALMA parameters are used. It is also worth noting that the expected return is calculated for each unit of stock, which means that if we use the expected compounded return value of MPI at PHP 146.02, which appears to be the lowest - the actual return could be at least PHP 146,020, assuming the minimum board lot required for the stocks is 1000 shares (Pesobility, n.d.).

However, despite the high potential returns, investors should proceed with caution for two reasons. First, the expected return is based on historical price data, which may not follow the trend of future price data, potentially rendering the trading algorithm obsolete (Quantified Strategies, 2023). Second, the return calculation does not account for and compensate for the additional fees associated with buying and selling the stock, which can affect the overall actual returns. Moreover, the author investigated the potential for returns by following the alam-SYS predictions, as discussed further in the succeeding section.

4.5 Results and Discussions for the Real World Application of alamSYS

Following the earlier discussions of the system's theoretical performance in terms of DMD-LSTM model performance and optimal ALMACD's expected returns for each stock. It is critical to test the alamSYS's actual performance in recommending which stocks to buy and sell in a simulated real-world application. It should be noted that this is referred to as 'simulated' because no actual money exchange or utilization occurred, but all calculations are based on real-world fees and data.

The results in Table 4.12, in particular, are based on data collected from March 24, 2023 to April 12, 2023 - a total of 10 trading days. A simple set of rules was followed to standardize the results and minimize the bias that could be introduced:

- (a) When the alamSYS indicated that the stock was to be purchased for that day, buy five times the required boardlot.
- (b) Similarly, if the alamSYS indicates that the stock must be sold, it must be sold at five times the required boardlot, regardless of the calculated return.
- (c) The PSEI will be bought and sold on a daily basis for the baseline comparison; and
- (d) All calculations are done with First Metro Securities' trading calculator.

Table 4.12: Return Performance Comparison Between alamSYS and PSEI

| | Realized Profit (PHP) | Realized Gain (%) |
|---------|-----------------------|-------------------|
| alamSYS | 7,839.75 | 1.51 |
| PSEI | -22,788.90 | -13.810 |

The above table displays the cumulative profit and gains over a 10-day trading period, demonstrating that the realized profit from following the stock recommendations of the alamSYS yields PHP 30,628.65 more than the baseline cumulative

returns of the PSEI. This is because trading the PSEI resulted in a loss of PHP 22,788.90 (a gain of -13.810%), whereas following the trading suggestions of the alamSYS resulted in a profit of PHP 7,839.75 (a gain of 1.51%).

Given that the PSEI was down by around 2% during this period, as shown in Figure 4.38, the positive performance of the alamSYS affirms the findings from Sections 4.3 and 4.4.



Figure 4.38: PSEI Stock Market Price Trend from March 24 to April 12, 2023

Note: The figure above was generated using the chart tool from Investagrams.com

However, the loss in PSEI may be attributed to the fact that no specific trading strategy was followed, but this is still a fair battle between the two systems - as the use of the alamSYS was based solely on its recommendations. Whereas, additional technical analysis on the stock market may result in higher alamSYS gains.

Figure 4.39 depicts the day-to-day gains of both systems to further visualize the performance of alamSYS's suggestions in comparison to the performance of PSEI.

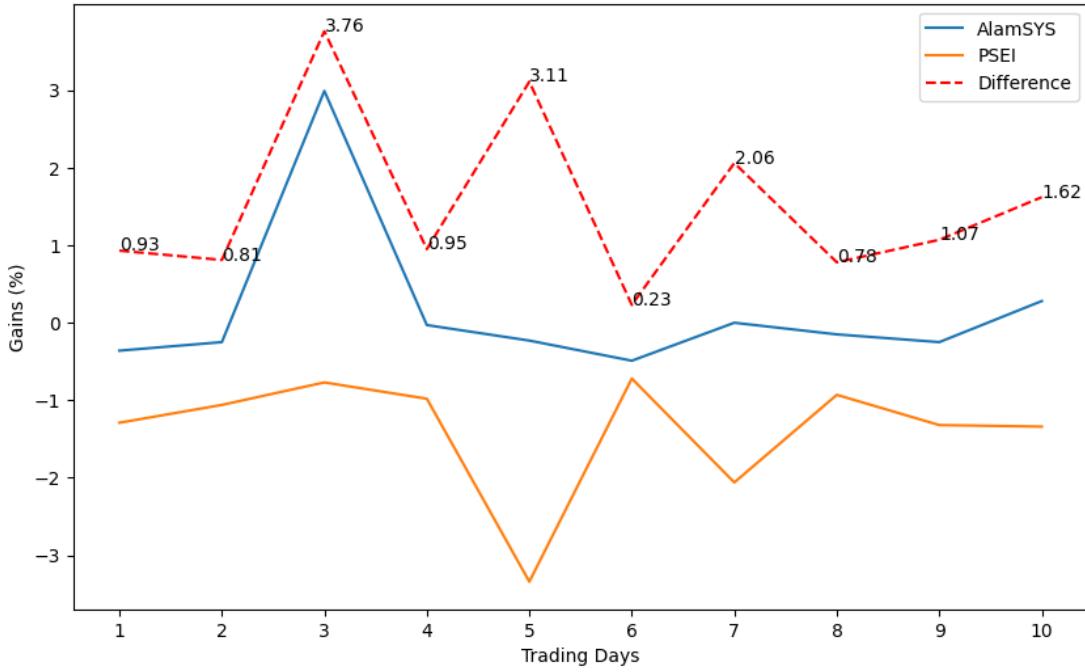


Figure 4.39: Comparison Between the Day-to-day Gains of alamSYS and PSEI

The above figure additionally demonstrates how the alamSYS performs better on a daily basis compared to the PSEI, highlighted by the 'Difference' line. Whereas the alamSYS performed better than the PSEI by 0.93%, 0.81%, 3.76%, 0.95%, 3.11%, 0.23%, 2.06%, 0.78%, 1.07%, and 1.62% over a 10-day period. Also, even though the cumulative return was positive, the alamSYS's day-to-day performance still yielded negative returns. It is worth noting, however, that it was able to reduce the risk of further losses, as can be observed in Figure 4.39.

Chapter 5

Conclusions and Future Work

Chapter 5 presents the findings from this unique problem, as well as recommendations for future research. This chapter specifically covered the following topics:

- (a) Summary of the main findings discussed in Chapter 4 that are relevant to the objectives of this special problem.
- (b) Contributions of the special problem were identified, emphasizing its strengths and limitations.
- (c) Finally, the author provides a roadmap for future works that outlines what needs to be investigated further, additional features that can be added to the system, other external applications or systems that may connect to or be used by the current system, and so on.

5.1 Summary of Findings

The following is a summary of the key findings, in accordance with the objectives outlined in Chapter 2 of this special problem and building on the discussions of the results in Chapter 4.

(a) In connection to the development of the alamSYS:

1. When idle, the alamSYS's average CPU and memory utilization is 0.43% and 525.29 MiB, respectively. Whereas the alamPREPROCESSOR uses the least amount of CPU power, it also uses the most memory due to the scheduling program being active.
2. Both alamAPI and alamPREPROCESSOR use a small amount of CPU power when idle, which is due to the low system requirements of their linux-based base images.
3. The alamPREPROCESSOR was able to process 100 consecutive data collection and processing, indicating a 100 times higher working capacity than expected.
4. The alamPREPROCESSOR's average runtime is 48.30 seconds. Which have been affected by the internet connection speed of 52.98Mbps on average.
5. It also uses an average of 20.03% of the CPU power and 794.29 MiB of memory on load. This means that the alamPREPROCESSOR can be used with low-end devices.
6. When alamPREPROCESSOR is loaded, its CPU and memory utilization are 99.95% and 60.62% higher, respectively, than when it is idle.
7. The average deployed API response time is 1.34 seconds, which is within the acceptable range for avoiding user-perceived interruptions.
8. The delay from the tunneling service accounts for 99.30% of the response time, and the actual response time of the alamAPI is only 0.009 seconds on average; and
9. The average CPU power utilization of alamAPI and alamDB on load is 99.05% and 76.24% higher, respectively, than their idle CPU power utilizations. This allows them to process 71,000 requests in about an hour and 20 minutes. On a different note, onload memory utilization was found to be lower than idle values, but this could simply be due to dips in zero MiB memory utilization over time, as shown in Figure 4.15.

Overall, the findings show that the alamSYS and its three major components (alamPREPROCESSOR, alamAPI, and alamDB) were successfully devel-

oped and integrated. Its idle and on-load performance were guaranteed to be dependable, stable, and efficient.

(b) In connection to the development of DMD-LSTM Model, and ALMACD Trading Algorithm:

1. The DMD-LSTM model with a window size of 5 outperformed the other eight LSTM models that were trained and tested. Where the MSE, RMSE, MAE, and MAPE values are 0.000037, 0.006106, 0.004175, and 0.000001, respectively.
2. Cross-validation of the selected DMD-LSTM model reveals acceptable performance outside of its training data.
3. Up to ten days of consecutive predictions show that the DMD-LSTM model still produces a range of acceptable MAPE scores. However, to avoid the possibility of extrapolation problems affecting the predictions, the alamSYS only integrated up to 5 days of successive predictions. The extrapolation problem on LSTM models were also demonstrated in the studies by Jing, Li, and Peng (2019); and S. Guo, Sun, Pei, and Li (2023).
4. The integration of optimized ALMACD on top of the DMD-LSTM model ensures that the alamSYS recommendations return positive yields for all stocks.
5. The DMD-LSTM and ALMACD applications in alamSYS were tested in real-world trading for a total of 10 trading days, outperforming the PSEI on cumulative return. Furthermore, the alamSYS is capable of mitigating risk and reducing potential losses in its position due to its ability to react quickly on potential price changes attributed to its 5 days in advance predictions and reactive ALMA parameters.

Overall, the combination of DMD-LSTM predictions and ALMACD as a trading algorithm ensures that the alamSYS suggests stock positions that, in theory, yield the highest positive returns while mitigating potential risks associated with market price volatility.

- (c) In connection to the development of the mobile-based test application. The mobile-based application demonstrated the potential of the alamSYS with other external internet-connected devices, such as a smartphone device.

5.2 The Implications of alamSYS to the Society

Based on the findings summary presented in the previous section, the combination of several key components, including the alamSYS, DMD-LSTM model with the optimized ALMACD, and mobile-based test application (alamAPP), has the potential to revolutionize the Philippine financial stock market.

The alamSYS was discovered to be dependable, stable, and efficient, with quick response times and the ability to handle a huge number of queries. While, through cross-validation, the DMD-LSTM model performed within an acceptable error range and proved acceptable performance outside of its training data.

Furthermore, the incorporation of optimized ALMACD confirms that all alamSYS suggestions generate positive returns, limiting potential risks and lowering potential losses. Also, the alamAPP illustrated how the alamSYS might be used with other internet-connected devices. Overall, it positions itself as an accessible stock market investment tool for all Filipinos, with the goal of supporting and empowering investors to make informed decisions and navigate the stock market confidently, even amid difficult market situations.

To put it simply, the sociological implications of a system such as alamSYS have a substantial impact on the potential benefits it could offer to the stock market, individual investors or traders, and the Philippine economy as a whole. The system has the potential to drive economic growth by providing jobs and pushing market advances. The system could boost investor trust and involvement in the stock market by offering credible stock trading advice and limiting potential risks and losses, resulting in a more healthy and prosperous economy.

5.3 Recommendations on Future Works

This section presents a road map for future research based on the insights gained in the preceding sections of this chapter. The roadmap outlines key areas for additional research, potential system enhancements, opportunities for integration with external applications or systems, and other applicable considerations.

(a) In connection to the development of the alamSYS:

1. Automated Model and Trading Algorithm Evaluator and Trainer - Given the possibility of changes in market price trend data over time, the patterns on which the current DMD-LSTM is based may become obsolete. As a result, it may be advantageous to incorporate an additional component that would revalidate the deep learning model as well as the trading algorithm, and if it performs worse than 10% of the current performance, it would automatically rerun the training with the new data until it finds a model and optimal parameters that performs inline or even better than the current one.
2. Alternative System Deployment Methods can be Explored - The tunneling service was found to be responsible for the majority of the delay in the alamSYS response time after deployment. As a result, it may be beneficial to investigate better alternatives, such as deploying the system on a server with a fast and reliable internet connection, though this may necessitate a larger budget. Another low-cost option is to deploy the system on cloud computing platforms such as AWS and Google Cloud.
3. Integration of other Models and Trading Algorithm - Initially, it was planned to integrate multiple combinations of machine learning models and trading algorithms into alamSYS, but due to time constraints, this was not pursued. However, the system was designed with this in mind, and the source code already allows for this feature to be integrated. Future developers, for example, can investigate the integration of ARIMA-based models in alamSYS. This provides users with multiple sources of data-driven information on the market's stocks. Where their decision is not limited to a single piece of information.

4. A Closer Examine on the Stock Suggestions of the System - Because the alamSYS currently only logs its daily predictions, it may be useful for future research to examine the trend of these predictions and try to see the accuracy of this suggestion, as well as potentially add improvements to the system; and
 5. System Maintenance and Additional Features or Improvements to the Current System - The completion of this paper does not indicate the end of the alamSYS's development lifetime; it is expected that the developer, and other developers in the future, will continue to work on improving the system from its initial deployment, using the continuous integration, continuous delivery, and deployment - CI/CD pipeline (RedHat, 2022).
- (b) In connection to the development of DMD-LSTM Model, and ALMACD Trading Algorithm:
1. Solving for the Observed Prediction Offset - The figures comparing predicted and actual price data for each stock revealed a distinct offset pattern. With this in mind, it may be worthwhile to investigate further and add potential additional steps to the predictions to compensate for the offset and improve the model's performance. This could be done during training or as a post-processing technique.
 2. Exploration of Other Machine Learning Models - There are numerous machine learning models that can be investigated for their potential to predict stock market movement. As a result, it is recommended that future developers or researchers be able to devise novel approaches to creating and integrating other models based on the findings and limitations of this special problem.
 3. Redefine the Optimal Fast, and Slow ALMA Parameters - The current optimal ALMA parameters, which do not account for the additional fees for buying and selling stock positions, could possibly not yield to the highest return potentials. As a result, future developers must investigate the possibility of including these fees from the Philippine Stock Exchange in order to better compute the optimal ALMA parameters, and then compare this results on the results of the current optimized ALMA parameters deployed in the alamSYS; and

4. Exploration of Other Trading Strategies and Algorithms - The stock market strategies is not confined on the utilization of moving averages such as Arnaud Legoux Moving Average in creating simple yet effective trading strategies. Hence, it would be beneficial if other trading strategies were explored being integrated alongside the machine learning models to be developed as well.
 - (c) In connection to the development of the mobile-based test application, future developers are encouraged to use other internet-connected or Internet-of-Things (IoT) devices such as smart speakers, smartwatches, smart glasses, and so on. To create an application that makes effective use of the alamSYS. Making certain that it is widely used.

References

- Adams, R. (2022, 7). *Is Investing in Stocks Gambling? No, Investing Isn't Zero Sum.* Retrieved October 07, 2022, from <https://youngandtheinvested.com/is-investing-in-the-stock-market-gambling/>
- Agrawal, M., Khan, D. A. U., & Shukla, D. P. K. (2019, 7). Stock price prediction using technical indicators: A predictive model using optimal deep learning. *International Journal of Recent Technology and Engineering (IJRTE)*, 8, 2297-2305. doi: 10.35940/ijrteB3048.078219
- Alegado, S., Lopez, D. B., & Calonzo, A. (2022, 8). *Philippine economic growth slows to 7.4% as inflation bites.* Retrieved October 06, 2022, from <https://www.bloomberg.com/news/articles/2022-08-09/philippine-economy-expands-7-4-last-quarter-below-estimate>
- 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/>
- Bae, K. H., & Kang, J. (2017). *Does the stock market benefit the economy?* Retrieved from <https://www.efmaefm.org/0EFMSYMPOSIUM/2017/papers/Does%20the%20Stock%20Market%20Benefit%20the%20Economy%20-%20updated.pdf>
- Baeldung. (2022). *Normalization inputs for an artificial neural network.* Retrieved April 15, 2023, from <https://www.baeldung.com/cs/normalizing-inputs-artificial-neural-network>
- Balaba, J. M. (2017). *Does the stock market drive the philippine economy?* Retrieved from <https://www.dlsu.edu.ph/wp-content/uploads/pdf/conferences/research-congress-proceedings/2017/RVREBM/>

RVREBM-I-003.pdf

- Bangko Sentral ng Pilipinas. (n.d.). *Stock market*. Retrieved October 06, 2022, from <https://www.bsp.gov.ph/Statistics/OtherRealSectorAccounts/stocks.pdf>
- Budiharto, W. (2021). Data science approach to stock prices forecasting in indonesia during covid-19 using long short-term memory (LSTM). *Journal of Big Data*, 8(1). Retrieved from <https://doi.org/10.1186/s40537-021-00430-0> doi: 10.1186/s40537-021-00430-0
- Bureau of the Treasury Bangko Sentral ng Pilipinas. (n.d.). *19 selected domestic interest rates weighted average in percent per annum for periods indicated*. Retrieved October 06, 2022, from www.bsp.gov.ph/Statistics/Financial%20System%20Accounts/tab19_dir.aspx
- Business World. (2022, 5). *Psei sinks further as net foreign selling surges*. Retrieved October 06, 2022, from <https://www.bworldonline.com/stock-market/2022/05/24/450559/psei-sinks-further-as-net-foreign-selling-surges/>
- Campbell, G. (2021, 5). *Does the stock market reflect the economy?* Retrieved October 07, 2022, from <https://www.economicsobservatory.com/does-the-stock-market-reflect-the-economy>
- Canto, J., & Romano, K. (2022, 4). *Philippines economic outlook 2022*. Retrieved October 06, 2022, from <https://www.mckinsey.com/featured-insights/asia-pacific/philippines-economic-outlook-2022>
- Chen, J. (2022, 7). *What is the stock market, what does it do, and how does it work?* Retrieved October 06, 2022, from <https://www.investopedia.com/terms/s/stockmarket.asp>
- Commission on Population and Development. (2021). *Popcom: Number of filipinos in 2021 estimated at 110.8 million, sizes of families trending lower at 4 members*. Retrieved October 07, 2022, from <https://popcom.gov.ph/popcom-number-of-filipinos-in-2021-estimated-at-110-8-million-sizes-of-families-trending-lower-at-4-members>.
- Cui, L. X., & Long, W. (2016, 11). Trading strategy based on dynamic mode decomposition: Tested in chinese stock market. *Physica A: Statistical Mechanics and its Applications*, 461, 498-508. doi: 10.1016/j.physa.2016.06.046
- 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/>
- EdwardJones. (n.d.). *Why invest in stocks?* Retrieved September 19, 2022, from <https://www.edwardjones.com/us-en/market-news-insights/guidance-perspective/benefits-investing-stock>
- EODHD. (n.d.). *Financial apis documentation. fundamental and historical data apis*. Retrieved November 13, 2022, from <https://eodhistoricaldata.com/financial-apis/>
- Fayed, A. (2022, 4). *A list of 14 blue chip companies in the philippines*. Retrieved October 07, 2022, from <https://adamfayed.com/a-list-of-14-blue-chip-companies-in-the-philippines/>
- 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/>
- Guo, S., Sun, N., Pei, Y., & Li, Q. (2023). 3d-unet-lstm: A deep learning-based radar echo extrapolation model for convective nowcasting. *Remote Sensing*, 15(6). Retrieved from <https://www.mdpi.com/2072-4292/15/6/1529> doi: 10.3390/rs15061529
- Guo, Y. (2022). *Stock price prediction using machine learning*. Retrieved from <https://www.diva-portal.org/smash/get/diva2:1672304/FULLTEXT01.pdf>
- Hall, M. (2022, 9). *How the stock market affects gdp*. Retrieved October 07, 2022, from <https://www.investopedia.com/ask/answers/033015/how-does-stock-market-affect-gross-domestic-product-gdp.asp>
- Hayes, A. (2022). *What is a trading strategy? how to develop one*. Retrieved April 22, 2023, from <https://www.investopedia.com/terms/t/trading-strategy.asp>

- Hayes, A. (2023). *Volatility: Meaning in finance and how it works with stocks*. Retrieved April 26, 2023, from <https://www.investopedia.com/terms/v/volatility.asp>
- Hua, J.-C., Roy, S., McCauley, J. L., & Gunaratne, G. H. (2016, 4). Using dynamic mode decomposition to extract cyclic behavior in the stock market. *Physica A: Statistical Mechanics and its Applications*, 448, 172-180. doi: 10.1016/j.physa.2015.12.059
- JavaTPoint. (n.d.). *Agile model*. Retrieved April 16, 2023, from <https://www.javatpoint.com/software-engineering-agile-model>
- Jing, J., Li, Q., & Peng, X. (2019). Mlc-lstm: Exploiting the spatiotemporal correlation between multi-level weather radar echoes for echo sequence extrapolation. *Sensors*, 19(18). Retrieved from <https://www.mdpi.com/1424-8220/19/18/3988> doi: 10.3390/s19183988
- Joseph, R. (2018). *Grid search for model tuning*. Retrieved April 17, 2023, from <https://towardsdatascience.com/grid-search-for-model-tuning-3319b259367e>
- Juviler, J. (2022). *Api response time, explained in 1000 words or less*. Retrieved April 22, 2023, from <https://blog.hubspot.com/website/api-response-time>
- Kenton, W. (2023). *Understanding value at risk (var) and how it's computed*. Retrieved April 26, 2023, from <https://www.investopedia.com/terms/v/var.asp>
- Kim, T. (2022, 7). *Alternative data for hedge funds: The competitive edge to investing*. Retrieved November 04, 2022, from <https://www.similarweb.com/corp/blog/investor/asset-research/hedge-funds-use-alternative-data>
- Kumbure, M. M., Lohrmann, C., Luukka, P., & Porras, J. (2022, 7). Machine learning techniques and data for stock market forecasting: A literature review. *Expert Systems with Applications*, 197, 116659. doi: 10.1016/j.eswa.2022.116659
- Kuttichira, D. P., Gopalakrishnan, E. A., Menon, V. K., & Soman, K. P. (2017, 9). Stock price prediction using dynamic mode decomposition. In (p. 55-60). IEEE. doi: 10.1109/ICACCI.2017.8125816
- Liu, C., Yan, J., Guo, F., & Guo, M. (2022, 8). Forecasting the market with machine learning algorithms: An application of nmc-bert-lstm-dqn-x algo-

- rithm in quantitative trading. *ACM Transactions on Knowledge Discovery from Data*, 16, 1-22. doi: 10.1145/3488378
- Lu, H., & Tartakovsky, D. M. (2020, 1). Prediction accuracy of dynamic mode decomposition. *SIAM Journal on Scientific Computing*, 42, A1639-A1662. doi: 10.1137/19M1259948
- Mann, J., & Kutz, J. N. (2015, 8). Dynamic mode decomposition for financial trading strategies.
- 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
- Milne, A. (2021). *The agile development methodology explained*. Retrieved April 16, 2023, from <https://www.netsolutions.com/insights/agile-development-methodology/>
- Mitchell, C. (2023). *Drawdown: What it is, risks and examples*. Retrieved April 26, 2023, from <https://www.investopedia.com/terms/d/drawdown.asp>
- MongoEngine. (n.d.). *Mongoengine documentation*. Retrieved November 13, 2022, from <https://docs.mongoengine.org/>
- Obthong, M., Tantisantiwong, N., Jeamwatthanachai, W., & Wills, G. (2020). A survey on machine learning for stock price prediction: Algorithms and techniques. In (p. 63-71). SCITEPRESS - Science and Technology Publications. doi: 10.5220/0009340700630071
- Pang, X., Zhou, Y., Wang, P., Lin, W., & Chang, V. (2020, 3). An innovative neural network approach for stock market prediction. *The Journal of Supercomputing*, 76, 2098-2118. doi: 10.1007/s11227-017-2228-y
- Pesobility. (n.d.). *Board lot of philippine stock exchange*. Retrieved April 22, 2023, from <https://www.pesobility.com/ref/boardlot>
- Quantified Strategies. (2023). *Why trading strategies are not working (how to know when trading systems stopped performing?)*. Retrieved April 22, 2023, from <https://www.quantifiedstrategies.com/why-trading-strategies-are-not-working/>
- Rea, L. (2020, 8). *Predicting future stock market trends with python and machine learning*. Retrieved October 05, 2022, from <https://towardsdatascience.com/predicting-future-stock-market-trends-with-python-machine-learning-2bf3f1633b3c>
- RedHat. (2022). *What is ci/cd?* Retrieved April 23, 2023, from <https://>

- www.redhat.com/en/topics/devops/what-is-ci-cd
- Reuters. (2022, 9). *Philippine economy to grow more slowly than previously thought in 2022, imf says.* Retrieved October 06, 2022, from <https://www.reuters.com/markets/asia/phippines-economy-seen-growing-65-2022-50-2023-imf-2022-09-26/>
- Royal Bank of Canada Direct Investing Inc. (n.d.). Key benefits of investing in stocks. *Royal Bank of Canada Direct Investing Inc..* Retrieved from www6.royalbank.com/en/di/hubs/investing-academy/chapter/key-benefits-of-investing-in-stocks/jv7atg13/jv7atg1j
- Savaş, M. C. (2017). *Algorithmic trading strategies using dynamic mode decomposition: Applied to turkish stock market.* Retrieved from <https://etd.lib.metu.edu.tr/upload/12621107/index.pdf>
- 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>
- SCHMID, P. J. (2010, 8). Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics, 656*, 5-28. doi: 10.1017/S0022112010001217
- Schwab-Pomerantz, C. (2021, 2). *Is investing in the stock market gambling?* Retrieved October 07, 2022, from <https://www.schwab.com/learn/story/is-investing-stock-market-gambling>
- Secret Data Scientist. (2023). *What is mae (mean absolute error)?* Retrieved April 15, 2023, from <https://secretdatascientist.com/mae-mean-absolute-error/>
- S.Gopal Krishna Patro, K. K. s. (2015). Normalization: A preprocessing stage. *International Journal of Advanced Research in Science, Engineering and Technology, 205.* Retrieved from 10.17148/IARJSET.2015.2305
- Soni, P., Tewari, Y., & Krishnan, D. (2022, 1). Machine learning approaches in stock price prediction: A systematic review. *Journal of Physics: Conference Series, 2161*, 012065. doi: 10.1088/1742-6596/2161/1/012065
- Statista Research Department. (2022). *Number of stock market accounts philippines 2021.* Retrieved October 07, 2022, from www.statista.com/statistics/1194840/philippines-number-of-stock-market-accounts-by-type
- Strader, T. J., Rozycski, J. J., Root, T. H., & Huang, Y.-H. (2020). *Machine*

- learning stock market prediction studies: Review and research directions* (Vol. 28). Retrieved from <https://scholarworks.lib.csusb.edu/jitim/vol28/iss4/3>
- Summers, B. D. (2022, 6). *Investing vs. gambling understanding risk-adjusted*. Retrieved October 07, 2022, from <https://www.forbes.com/sites/forbesfinancecouncil/2022/06/01/investing-vs-gambling-understanding-risk-adjusted-performance/?sh=1ca2a49957d9>
- The Economic Times. (n.d.). *What is 'stock market'*. Retrieved October 05, 2022, from <https://economictimes.indiatimes.com/definition/stock-market>
- The Philippine Stock Exchange, Inc. (n.d.-a). *Company information*. Retrieved October 05, 2022, from https://edge.pse.com.ph/companyInformation/form.do?cmpy_id=478
- The Philippine Stock Exchange, Inc. (n.d.-b). *Company list*. Retrieved October 05, 2022, from <https://edge.pse.com.ph/companyDirectory/form.do>
- Tiangolo. (n.d.). *Fastapi documentation*. Retrieved November 13, 2022, from <https://fastapi.tiangolo.com/>
- Trade Brains. (2022, 7). *How does the stock market affect the economy?* Retrieved October 07, 2022, from <https://tradebrains.in/how-stock-market-affect-the-economy/>
- Trading Economics. (n.d.). *Philippines inflation rate*. Retrieved October 07, 2022, from <https://tradingeconomics.com/philippines/inflation-cpi>
- 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>
- Uchiyama, Y., & Nakagawa, K. (2021, 7). Improving momentum strategies using adaptive elastic dynamic mode decomposition. In (p. 388-393). IEEE. doi: 10.1109/IIAI-AAI53430.2021.00068
- U.S. Securities and Exchange Commission. (n.d.). *What are stocks?* Retrieved October 07, 2022, from www.investor.gov/introduction-investing/investing-basics/investment-products/stocks
- Visual Paradigm. (n.d.). *What is use case diagram?* Retrieved April 26, 2023, from <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/>

- 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/>
- WallStreetMojo. (n.d.). *Algorithmic trading*. Retrieved April 22, 2023, from <https://www.wallstreetmojo.com/algorithmic-trading/>
- Williamson, J. G. (2015). *Handbook of the economics of international migration*. Retrieved December 15, 2022, from <https://www.sciencedirect.com/topics/economics-econometrics-and-finance/global-economic-crisis>
- Zwetsloot, R. (2019). *Raspberry pi 4 specs and benchmarks*. Retrieved April 22, 2023, from <https://magpi.raspberrypi.com/articles/raspberry-pi-4-specs-benchmarks>

Appendix A

Source Code Repository

xxx

Appendix B

Raw Data

xxx

Appendix C

Project Management Documentation

xxx

Appendix D

Glossary of Terms

xxx

Appendix E

Acknowledgements

xxx

Appendix F

Author's Contact Information

xxxx