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

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

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

OLARTE, John Markton M.

Nilo C. Araneta
Adviser

June 2023

Contents

List of Figures

List of Tables

Chapter 1

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

1.1 alamSYS Documentation

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

1.1.1 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:

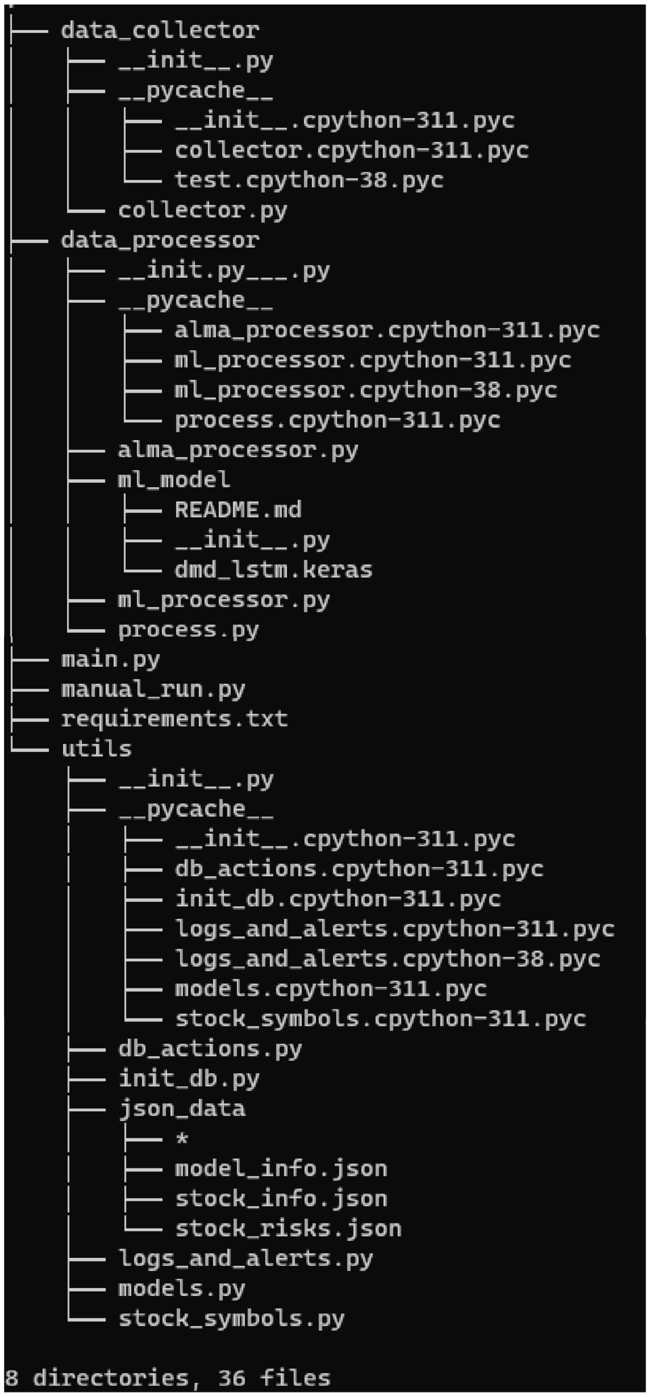
    1. Database model definitions
    2. Database actions such as:
        i. Connecting to the alamDB.
        ii. Disconnecting to the alamDB.
        iii. Purging the Buy Collections.
        iv. Purging the Sell Collections.
        v. Saving the Updated Stocks to Buy JSON file.
        vi. Saving the 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 1.1 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___.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 1.1: 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.

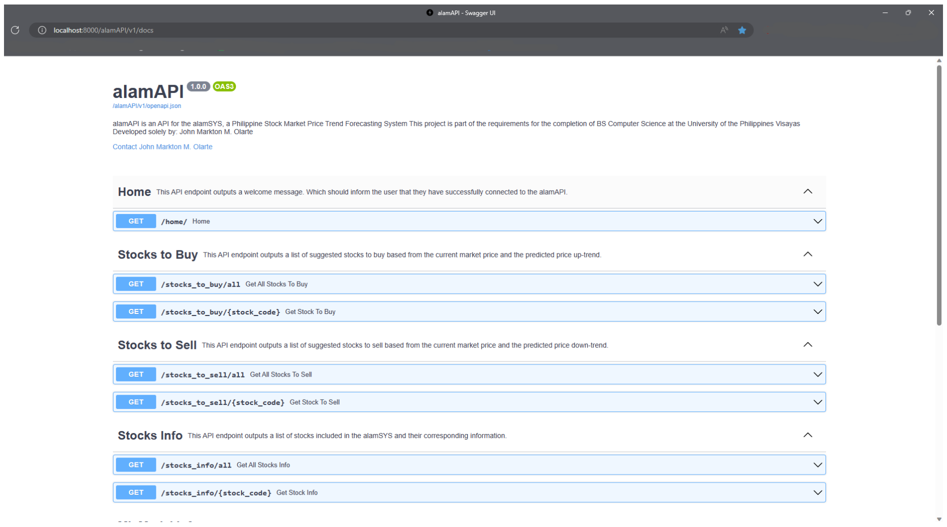
(f) /scheduled_task_logs - This is the directory for all the successful operations ('/success_log.csv') , and all the unsuccessful operations ('/error_log.csv') conducted by the scheduled operation of the alamPREPROCESSOR.

1.1.2 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 1.2. Alternatively, Redocly's alamAPI API documentation can be accessed via this link: *localhost:8000/alamAPI/v1 docs*, as shown in Figure 1.2.



alamAPI - Swagger UI

localhost:8000/alamAPI/v1/docs

alamAPI 1.0.0 OAS3

/alamAPI/v1/openapi.json

alamAPI is an API for the alamSYS, a Philippine Stock Market Price Trend Forecasting System This project is part of the requirements for the completion of BS Computer Science at the University of the Philippines Visayas
Developed solely by: John Markton M. Olarte

Contact John Markton M. Olarte

Home This API endpoint outputs a welcome message. Which should inform the user that they have successfully connected to the alamAPI.

GET `/home/` Home

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.

GET `/stocks_to_buy/all` Get All Stocks To Buy

GET `/stocks_to_buy/{stock_code}` Get Stock To Buy

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.

GET `/stocks_to_sell/all` Get All Stocks To Sell

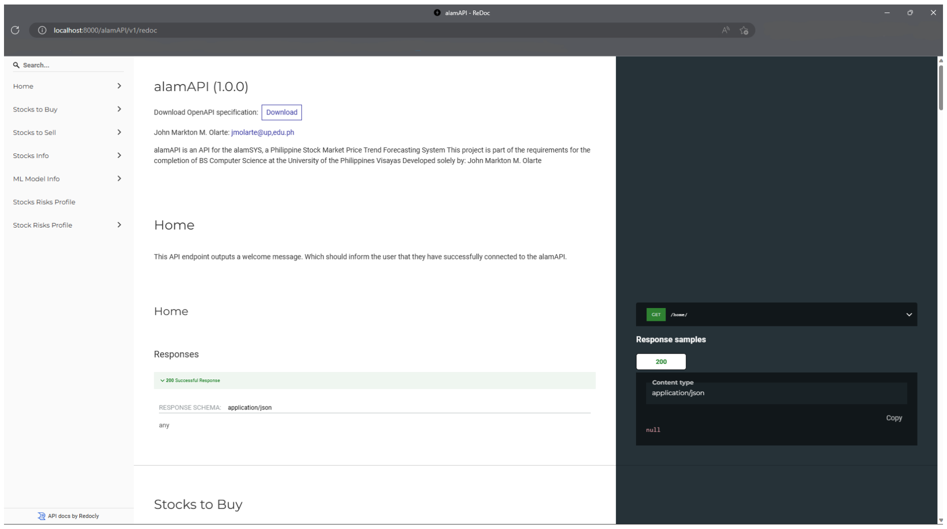
GET `/stocks_to_sell/{stock_code}` Get Stock To Sell

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

GET `/stocks_info/{stock_code}` Get Stock Info

Figure 1.2: alamAPI Web-based API Documentation Using Swagger



alamAPI (1.0.0)

Download OpenAPI specification: Download

John Markton M. Olarte: jmolarte@up,edu.ph

alamAPI is an API for the alamSYS, a Philippine Stock Market Price Trend Forecasting System This project is part of the requirements for the completion of BS Computer Science at the University of the Philippines Visayas Developed solely by: John Markton M. Olarte

Home

This API endpoint outputs a welcome message. Which should inform the user that they have successfully connected to the alamAPI.

Home

Responses

200 Successful Response

RESPONSE SCHEMA: application/json

any

Stocks to Buy

GET /home/

Response samples

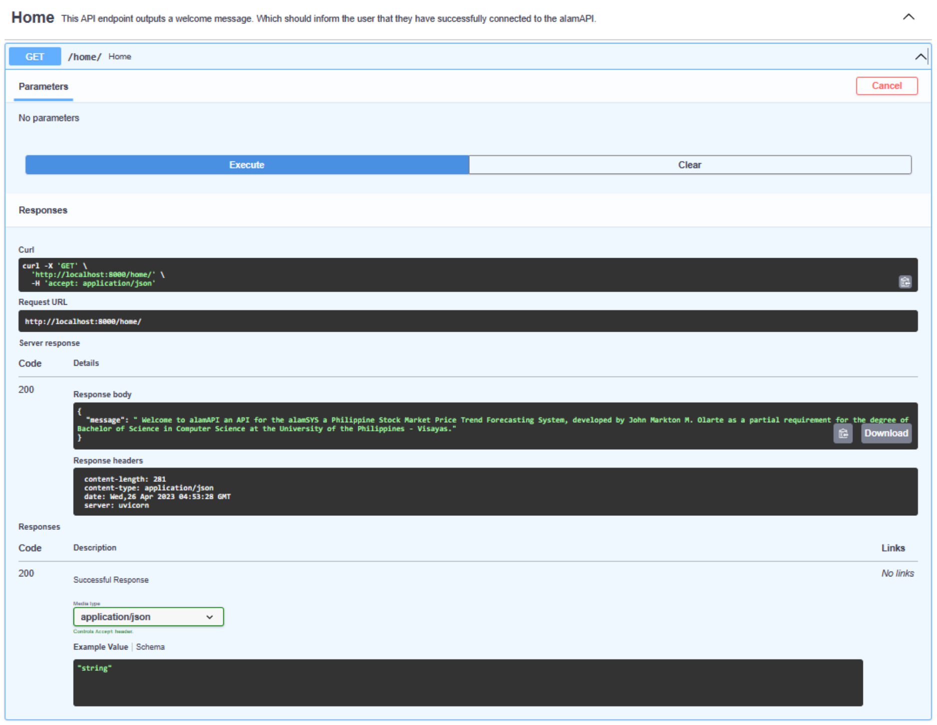
200

Content type
application/json

```
null
```

Figure 1.3: 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 1.4 depicts an example request executed using this API endpoint.


Home This API endpoint outputs a welcome message. Which should inform the user that they have successfully connected to the alamAPI.

GET `/home/` Home

Parameters

No parameters

Curl

```
curl -X 'GET' \
  'http://localhost:8000/home/' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:8000/home/
```

Server response

Code Details

200 Response body

```
{
  "message": " Welcome to alamAPI an API for the alamSYS a Philippine Stock Market Price Trend Forecasting System, developed by John Markton M. Olarte as a partial requirement for the degree of Bachelor of Science in Computer Science at the University of the Philippines - Visayas."
}
```

Response headers

```
 content-length: 281
 content-type: application/json
 date: Wed,26 Apr 2023 04:53:28 GMT
 server: uvicorn
```

Responses

Code	Description	Links
200	Successful Response	*No links*

Media type: application/json

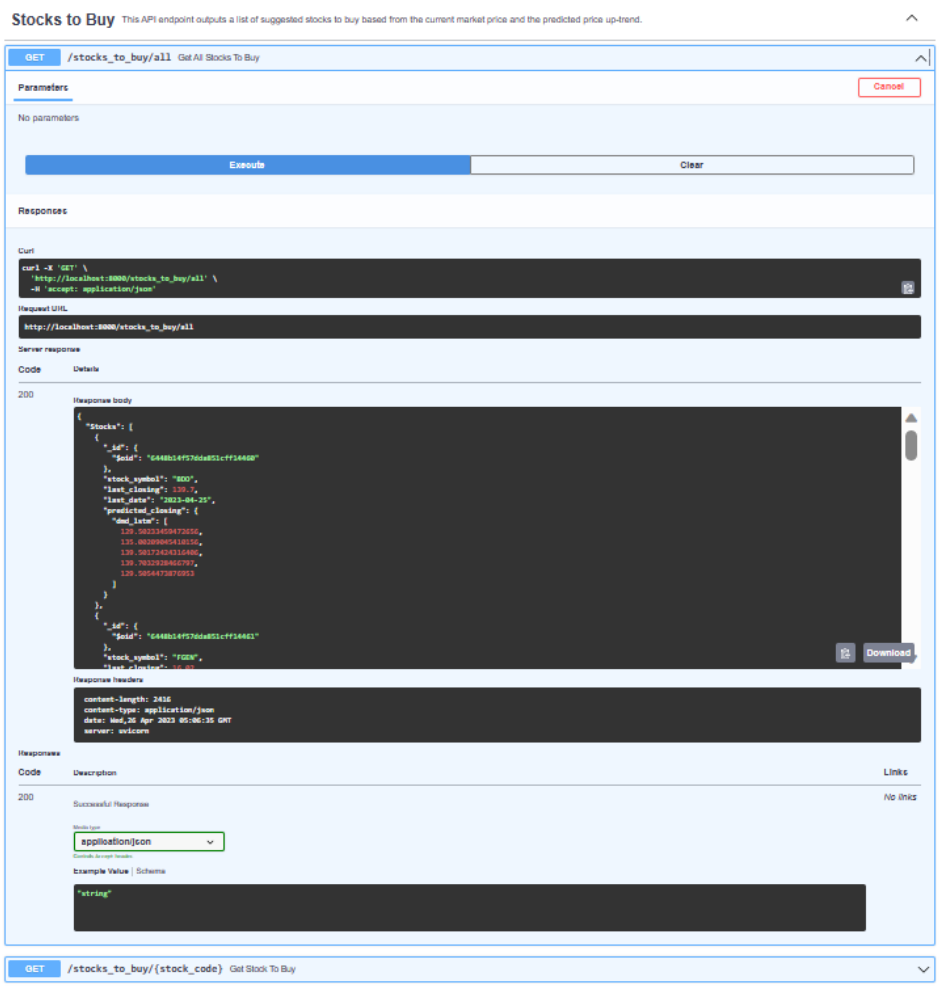
Controls Accept header.

Example Value | Schema

```
"string"
```

Figure 1.4: 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 is shown in Figure 1.5



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.

GET /stocks_to_buy/all Get All Stocks To Buy

Parameters

No parameters

Responses

Curl

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

Request URL

```
http://localhost:8000/stocks_to_buy/all
```

Server response

Code Details

200

Response body

```
{
  "Stocks": [
    {
      "_id": {
        "$oid": "6448b14f57dda851cff14460"
      },
      "stock_symbol": "BDO",
      "last_closing": 139.7,
      "last_date": "2023-04-25",
      "predicted_closing": {
        "dmd_lstm": [
          129.50233459472656,
          135.00209045410156,
          139.50172424316406,
          139.7032928466797,
          129.5054473876953
        ]
      }
    },
    {
      "_id": {
        "$oid": "6448b14f57dda851cff14461"
      },
      "stock_symbol": "FGEN",
      "last_closing": 16.02
```

Response headers

```
content-length: 2416
content-type: application/json
date: Wed,26 Apr 2023 05:06:35 GMT
server: uvicorn
```

Responses

Code Description Links

200 Successful Response No links

Media type: application/json

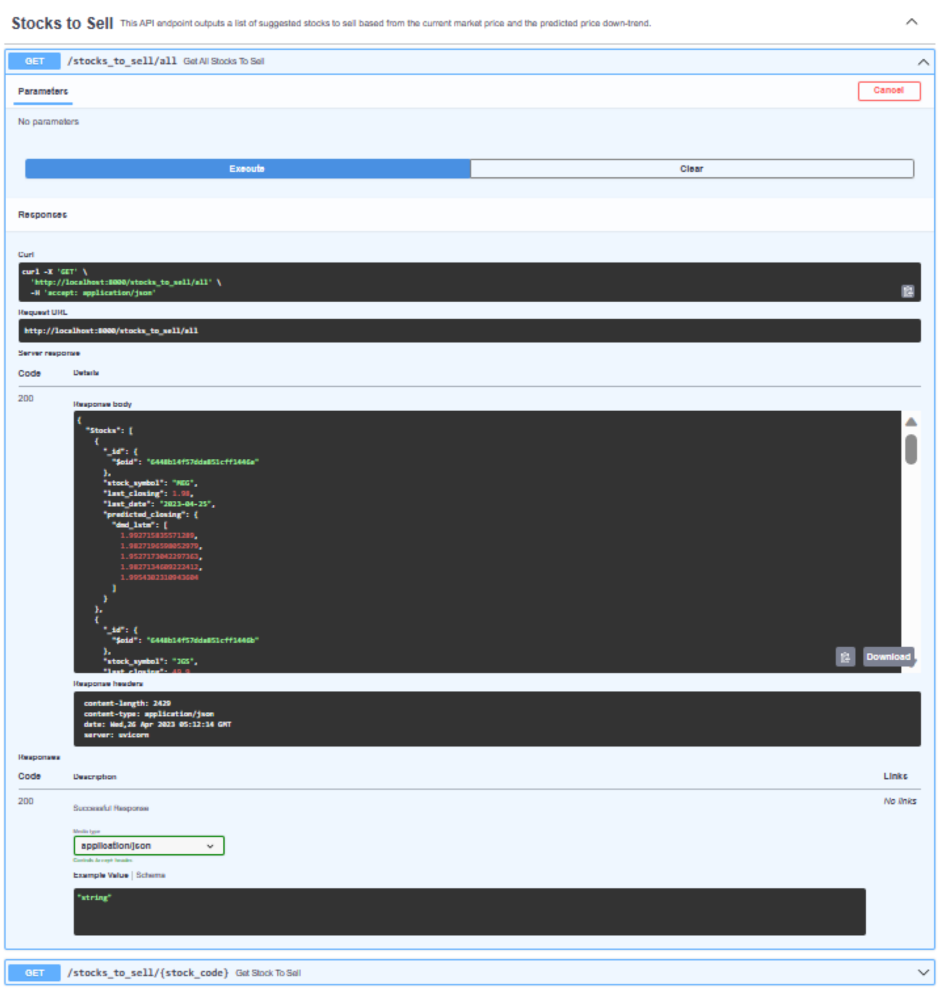
Example Value | Schema

```
"string"
```

GET /stocks_to_buy/{stock_code} Get Stock To Buy

Figure 1.5: 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 1.6 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.
GET
/stocks_to_sell/all
Get All Stocks To Sell
Parameters
Cancel
No parameters
Execute
Clear
Responses
Curl

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

```
http://localhost:8000/stocks_to_sell/all
```

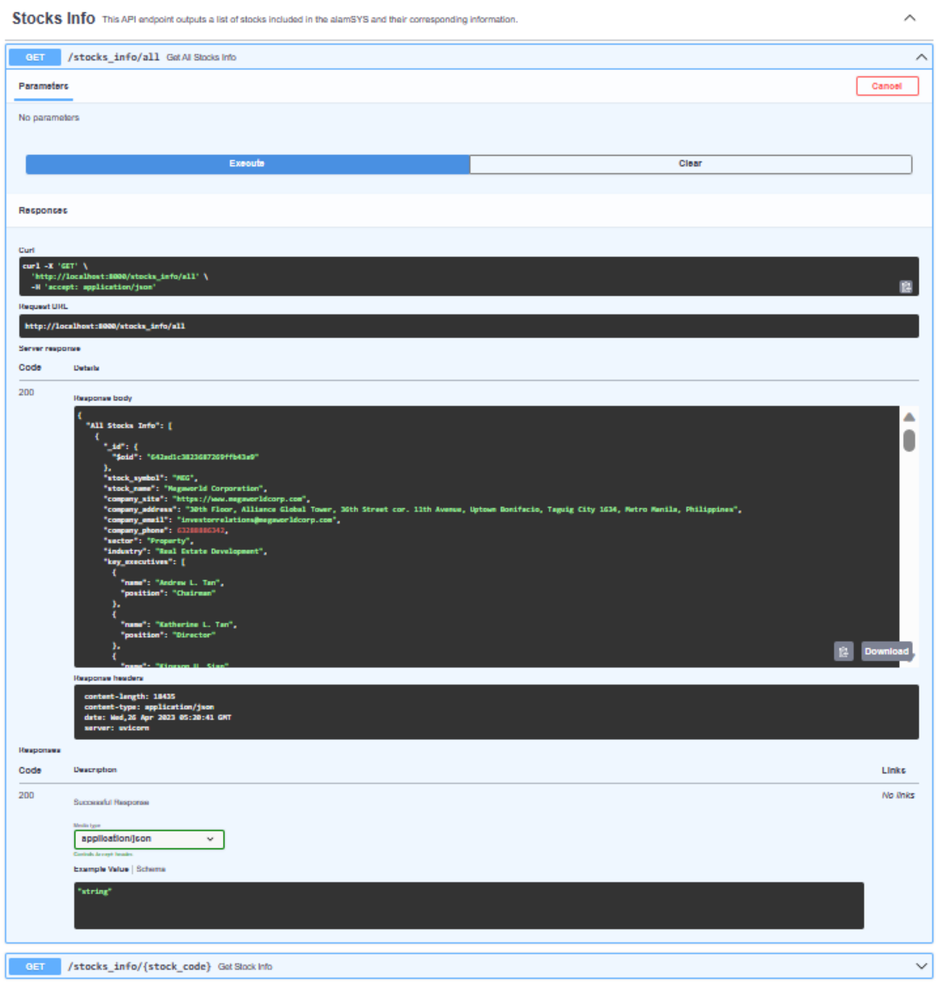
```
{
  "Stocks": [
    {
      "_id": {
        "$oid": "6448b14f57dda851cff1446a"
      },
      "stock_symbol": "MEG",
      "last_closing": 1.98,
      "last_date": "2023-04-25",
      "predicted_closing": {
        "dmd_lstm": [
          1.992715835571289,
          1.9827196598052979,
          1.9527173042297363,
          1.9827134609222412,
          1.9954382310943604
        ]
      }
    },
    {
      "_id": {
        "$oid": "6448b14f57dda851cff1446b"
      },
      "stock_symbol": "JGS",
      "last_closing": 49.9
```

```
content-length: 2429
content-type: application/json
date: Wed,26 Apr 2023 05:12:14 GMT
server: uvicorn
```

```
"string"
```

Figure 1.6: 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 1.7 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
Cancel
No parameters
Execute
Clear
Responses
Curl

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

```
http://localhost:8000/stocks_info/all
```

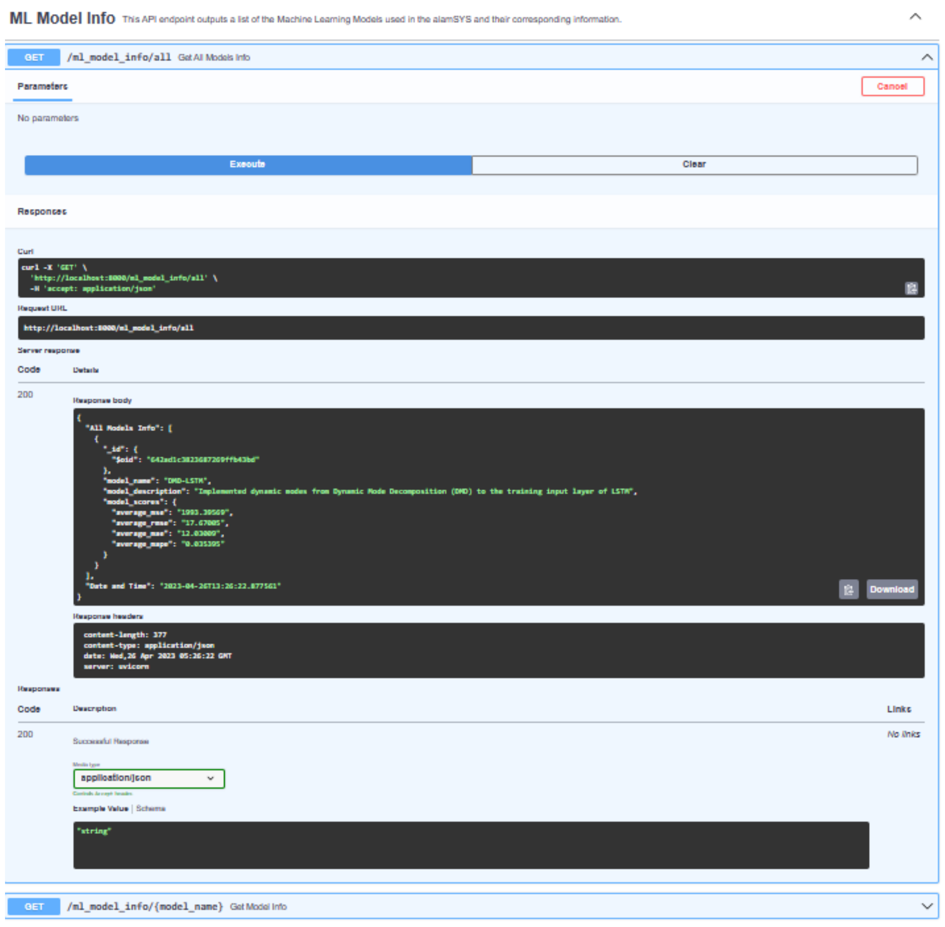
Server response
Code
Details
200
Response body

```
{
  "All Stocks Info": [
    {
      "_id": {
        "$oid": "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, Uptown Bonifacio, Taguig City 1634, Metro Manila, Philippines",
      "company_email": "investorrelations@megaworldcorp.com",
      "company_phone": 63288886342,
      "sector": "Property",
      "industry": "Real Estate Development",
      "key_executives": [
        {
          "name": "Andrew L. Tan",
          "position": "Chairman"
        },
        {
          "name": "Katherine L. Tan",
          "position": "Director"
        },
        {
          "name": "Kingson U. Sian"
```

```
content-length: 18435
content-type: application/json
date: Wed,26 Apr 2023 05:20:41 GMT
server: uvicorn
```

Figure 1.7: 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 1.8 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
Cancel
No parameters
Execute
Clear
Responses
Curl

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

```
http://localhost:8000/ml_model_info/all
```

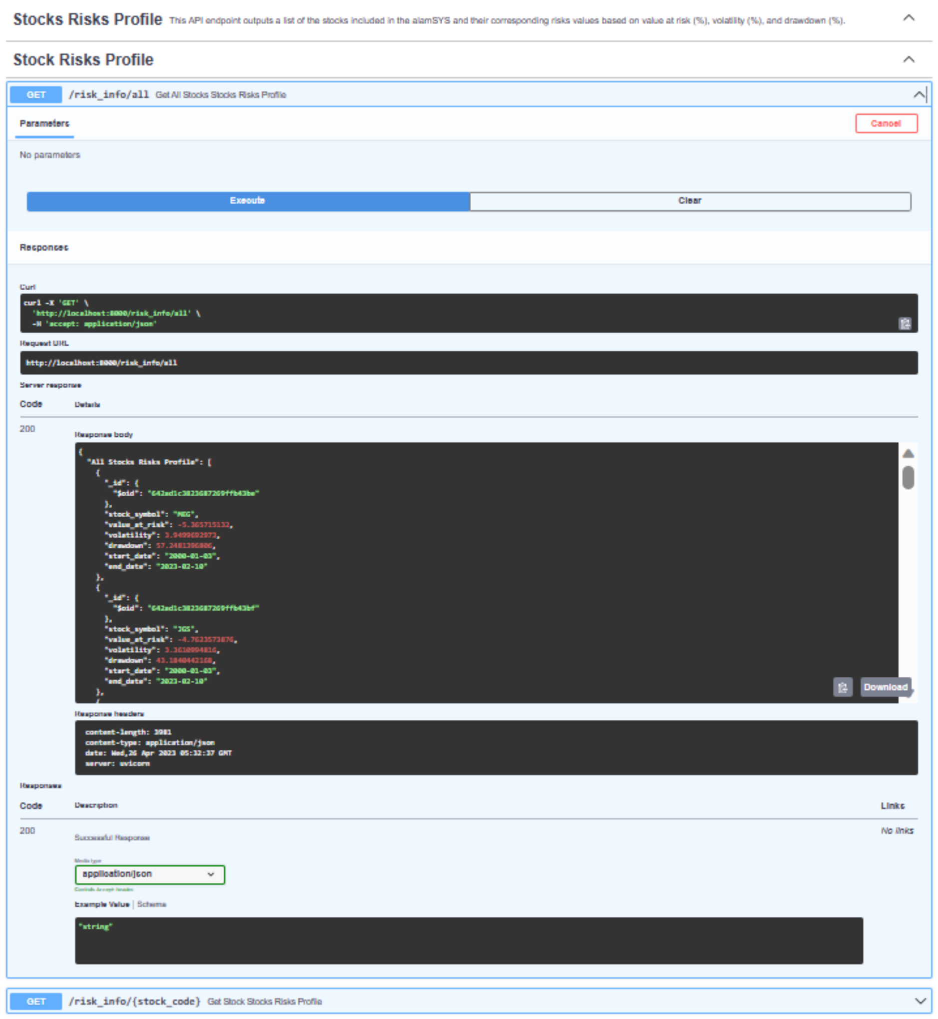
```
{
  "All Models Info": [
    {
      "_id": {
        "$oid": "642ad1c3823687269ffb43bd"
      },
      "model_name": "DMD-LSTM",
      "model_description": "Implemented dynamic modes from Dynamic Mode Decomposition (DMD) to the training input layer of LSTM",
      "model_scores": {
        "average_mse": "1993.39569",
        "average_rmse": "17.67005",
        "average_mae": "12.03009",
        "average_mape": "0.035395"
      }
    }
  ],
  "Date and Time": "2023-04-26T13:26:22.877561"
}
```

```
content-length: 377
content-type: application/json
date: Wed,26 Apr 2023 05:26:22 GMT
server: uvicorn
```

```
"string"
```

Figure 1.8: 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 1.9 depicts a sample execution, specifically for /all.


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 (%).
Stock Risks Profile
GET /risk_info/all Get All Stocks Stocks Risks Profile
Parameters
Cancel
No parameters
Execute
Clear
Responses
Curl

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

```
http://localhost:8000/risk_info/all
```

Server response
Code
Details
200
Response body

```
{
  "All Stocks Risks Profile": [
    {
      "_id": {
        "$oid": "642ad1c3823687269ffb43be"
      },
      "stock_symbol": "MEG",
      "value_at_risk": -5.365715132,
      "volatility": 3.9499692973,
      "drawdown": 57.2481396806,
      "start_date": "2000-01-03",
      "end_date": "2023-02-10"
    },
    {
      "_id": {
        "$oid": "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"
    },
```

```
content-length: 3981
content-type: application/json
date: Wed,26 Apr 2023 05:32:37 GMT
server: uvicorn
```

```
"string"
```

Figure 1.9: Sample API Endpoint: Stocks Risks Profile (all) Request

To go over the following risk values in greater detail. Furthermore, Table 1.1 summarizes the risk values for all stocks included in the alamSYS:

(a) 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 1.1 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 1.1 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 1.1, 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 1.1: 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 1.1 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 1.10 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.*

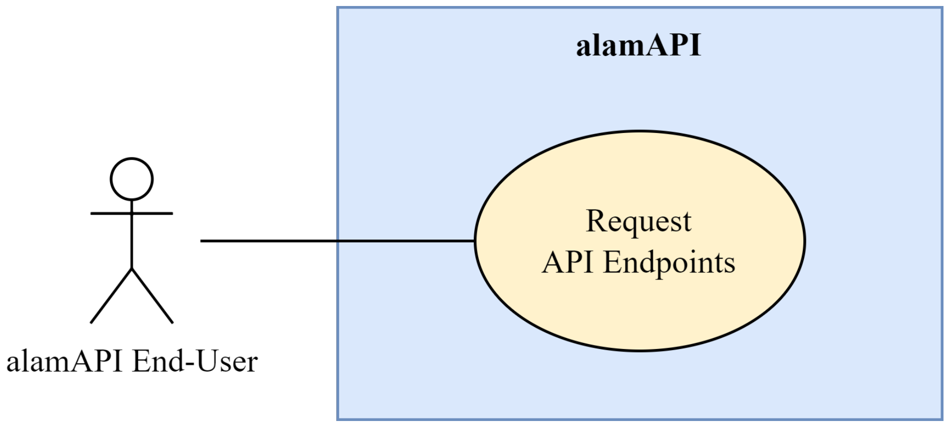
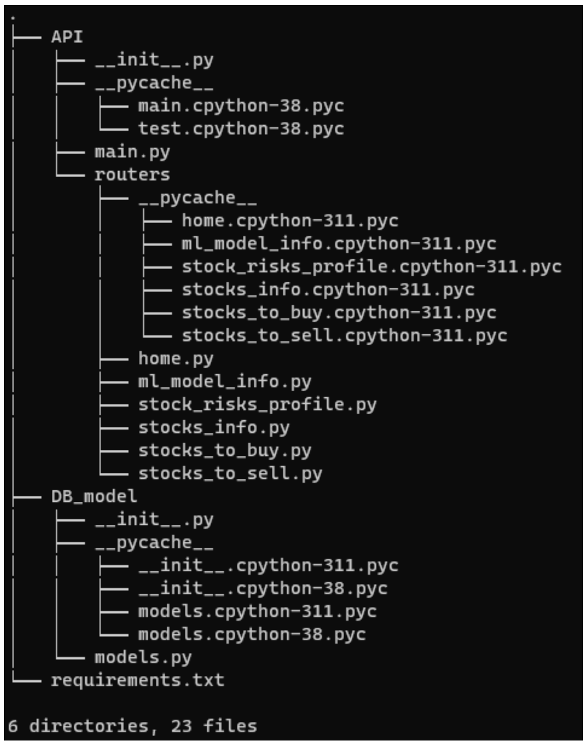


Figure 1.10: alamAPI End User UML Use Case Diagram

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 1.11 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.cpython-38.pyc
│   └── models.py
└── requirements.txt

6 directories, 23 files
```

Figure 1.11: 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.

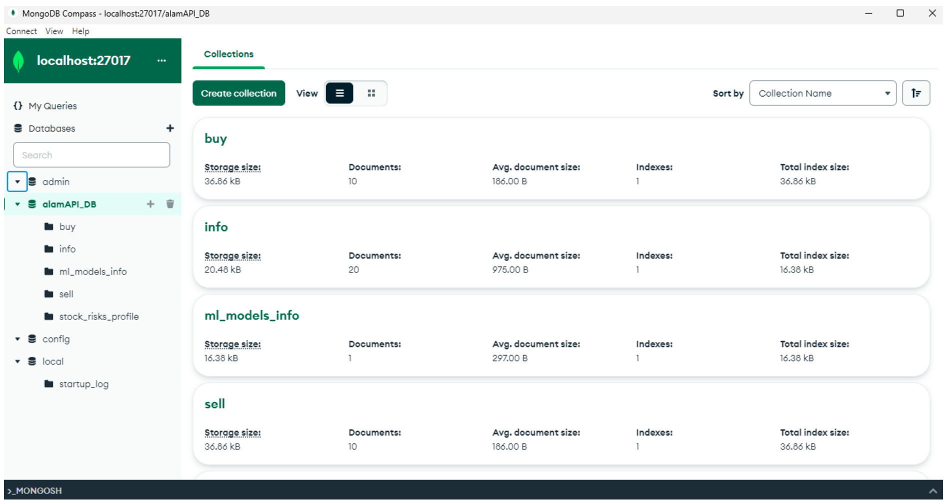
Utilization of the MongoDBCompass is straightforward and is only composed of these few steps:

(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 1.12 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.



	Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
buy	36.86 kB	10	186.00 B	1	36.86 kB
info	20.48 kB	20	975.00 B	1	16.38 kB
ml_models_info	16.38 kB	1	297.00 B	1	16.38 kB
sell	36.86 kB	10	186.00 B	1	36.86 kB

Figure 1.12: MongoDB Compass Database Management of alamDB

1.1.3 Documentation for alamAPP

xxx

User Interface Showcase

xxx

Utilization of alamSYS in the alamAPP

xxx

1.1.4 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 alamPRE-PROCESSOR'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 1.13.

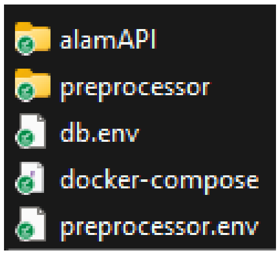


Figure 1.13: alamSYS Required Folder Structure

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 operating 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.

1.2 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 1.2 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 1.2: 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 **??**.

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 1.3: Internal Load Average Resource Usage Statistics

	Data Collector	**Data Processor**	**alamSYS PREPROCESSOR (Data Collector & Data Processor)**
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

Table 1.3 continued from previous page

	Data Collector	**Data Processor**	**alamSYS PREPROCESSOR (Data Collector & Data Processor)**
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 1.3, 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 ailing. 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 1.4 and 1.5. 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 1.4: 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 1.5: 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 1.6 shows the average CPU and memory utilization of alamAPI and alamDB as the load testing as stated above, was being processed.

Table 1.6: 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 1.14 shows the CPU utilization of alamAPI and alamDB over time. And Figure 1.15 shows the memory utilization of alamAPI and alamDB.

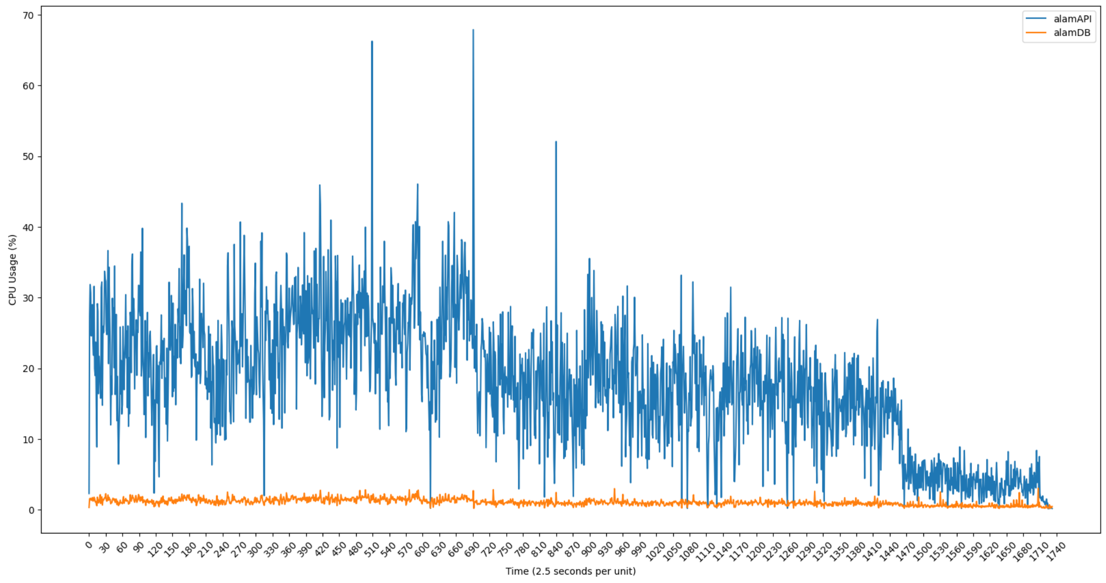


Figure 1.14: Deployment Load CPU Utilization of alamAPI and alamDB Over Time

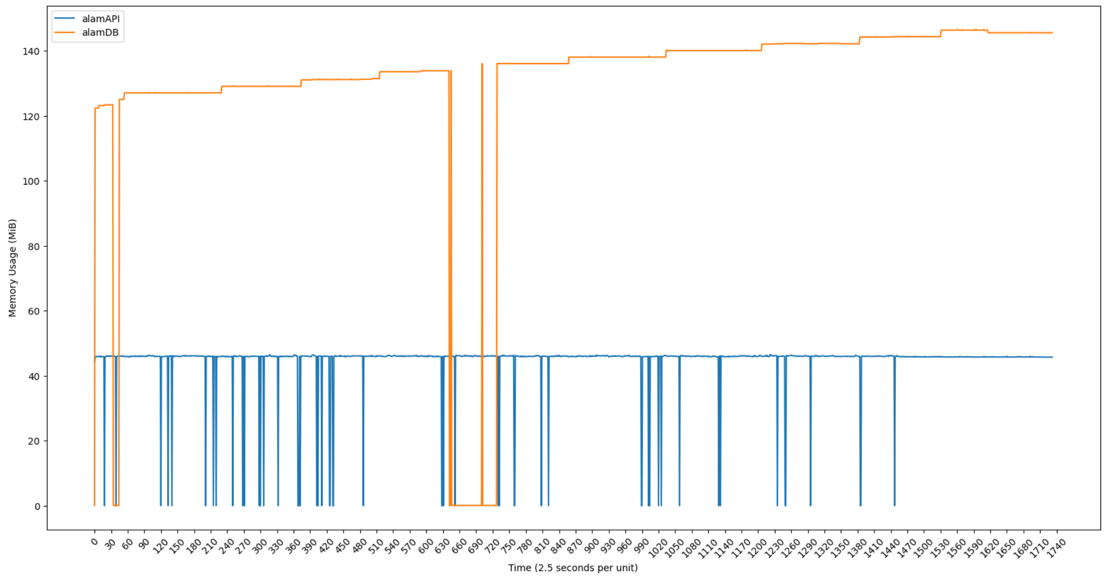


Figure 1.15: Deployment Load Memory Utilization of alamAPI and alamDB Over Time

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.

1.3 DMD-LSTM Model Results and Discussions

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

In Table 1.7 the training error metrics are shown for each of the window sizes tested.

Table 1.7: 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. Morever, we can see the differences from each MAPE score for each window size in the Figure 1.16 shown below.

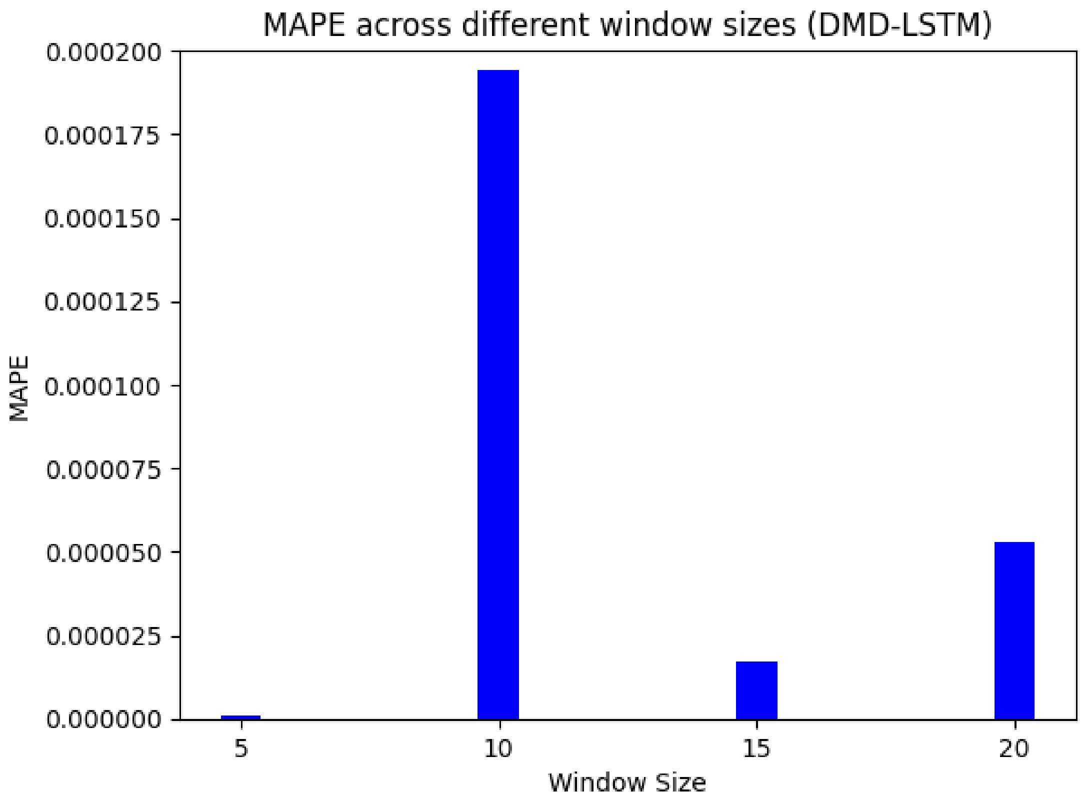


Figure 1.16: Comparison of MAPE Scores for DMD-LSTM Model Training Across Different Window Sizes

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 1.16.

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 1.8: 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 1.9. 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 1.9: 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
LTG	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.

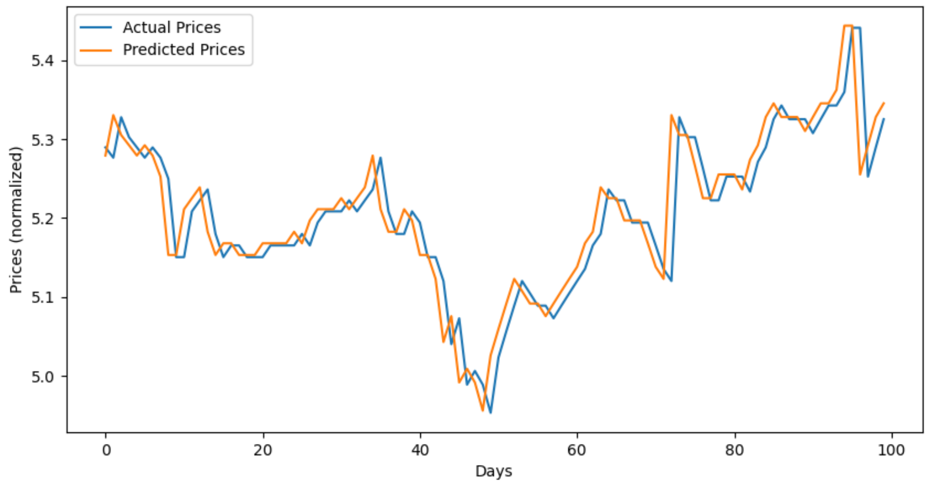


Figure 1.17: Actual vs Predicted Prices on AC for 100 days

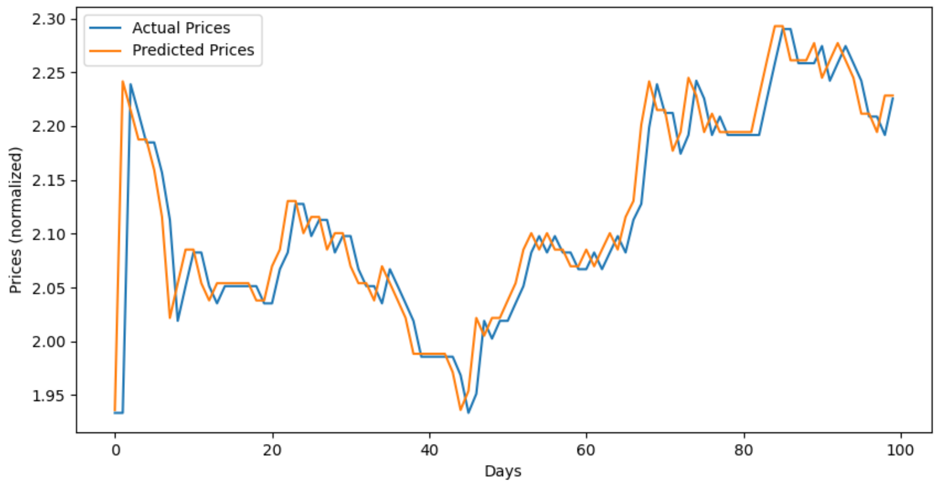


Figure 1.18: Actual vs Predicted Prices for ALI over 100 days

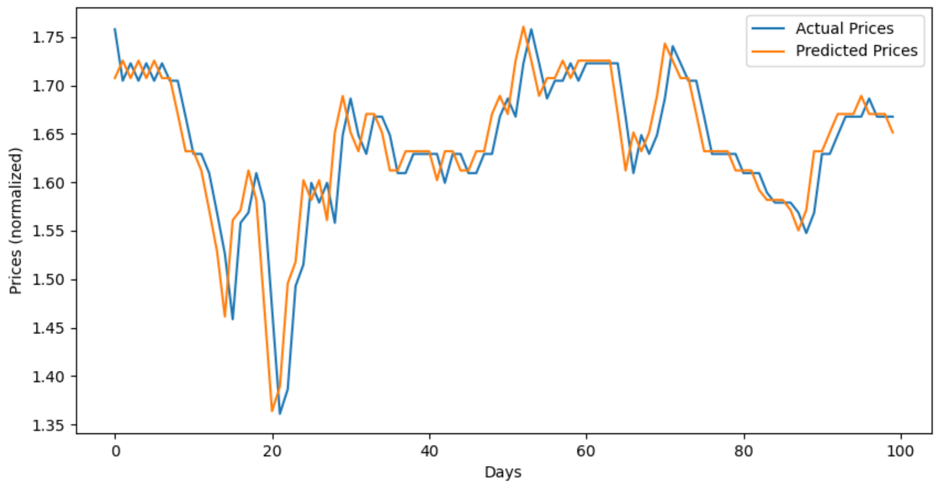


Figure 1.19: Actual vs Predicted Prices for AP over 100 days

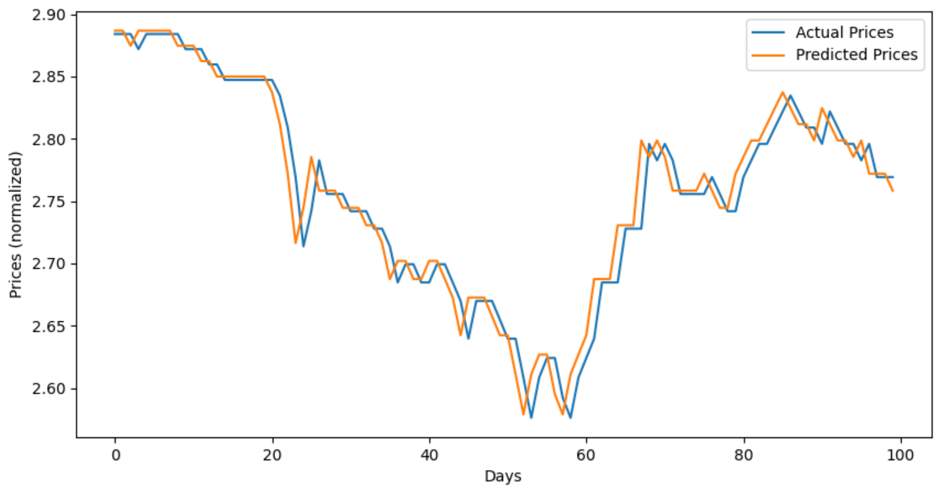


Figure 1.20: Actual vs Predicted Prices for BDO over 100 days

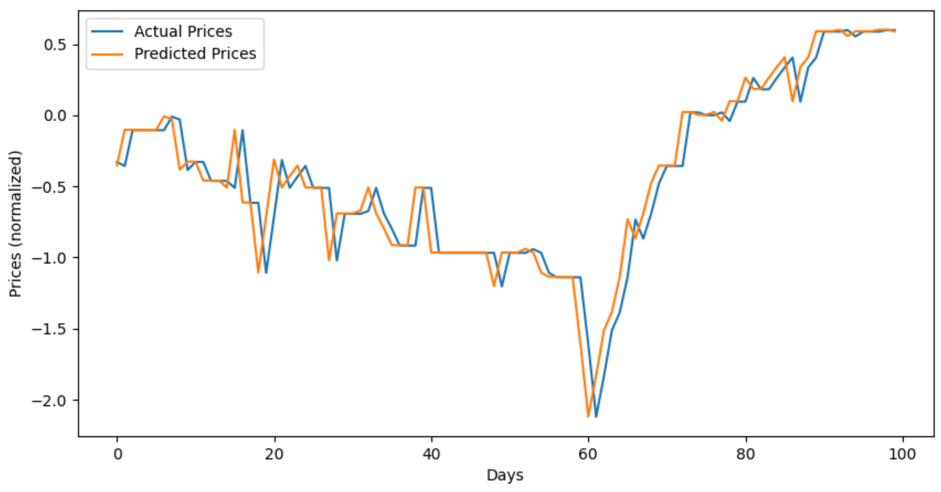


Figure 1.21: Actual vs Predicted Prices for BLOOM over 100 days

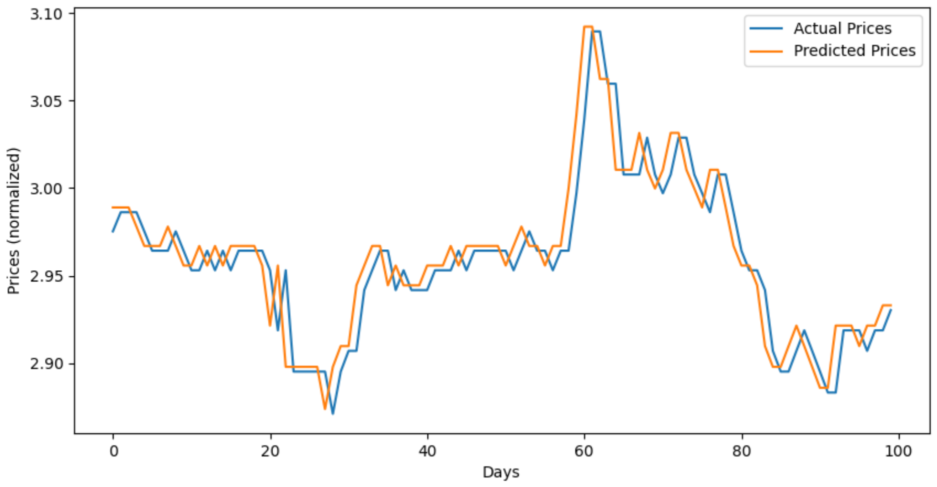


Figure 1.22: Actual vs Predicted Prices for FGEN over 100 days

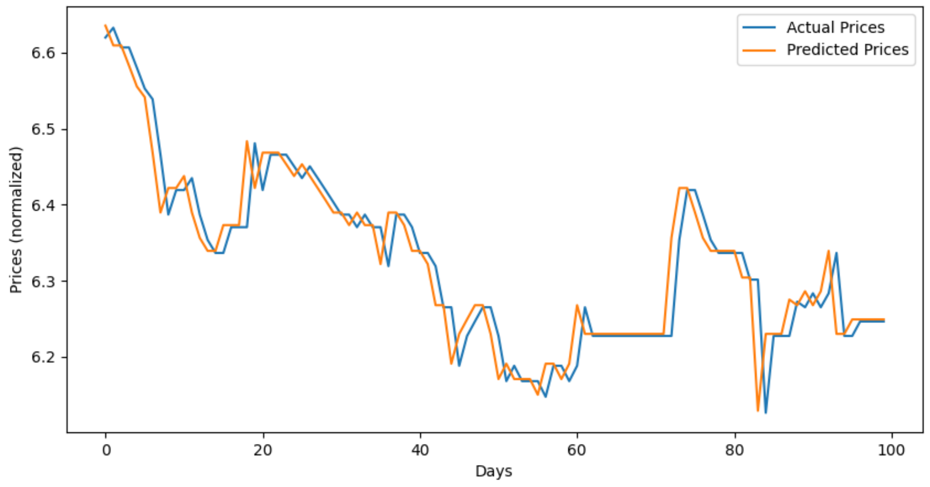


Figure 1.23: Actual vs Predicted Prices for GLO over 100 days

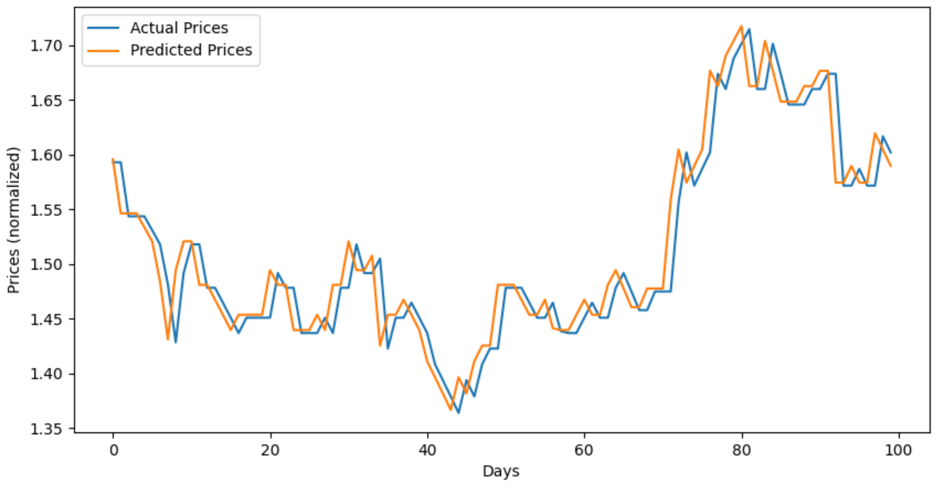


Figure 1.24: Actual vs Predicted Prices for ICT over 100 days

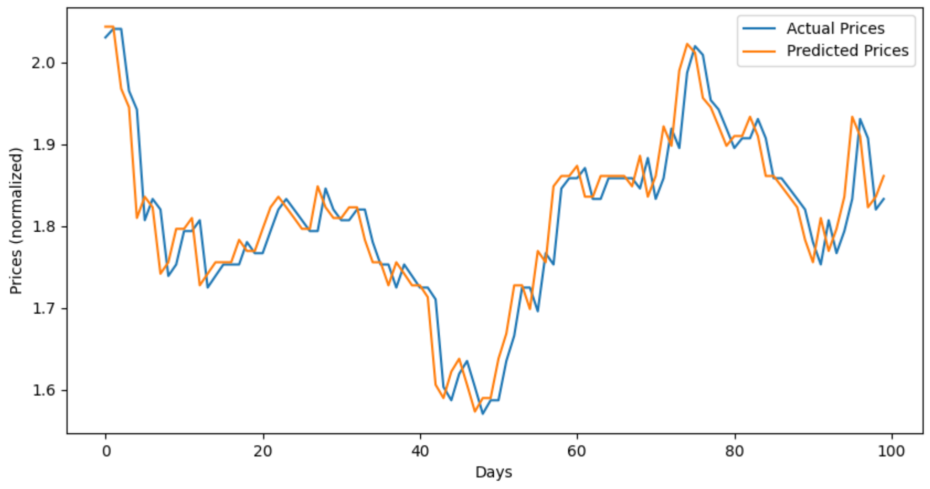


Figure 1.25: Actual vs Predicted Prices on JGS for 100 days

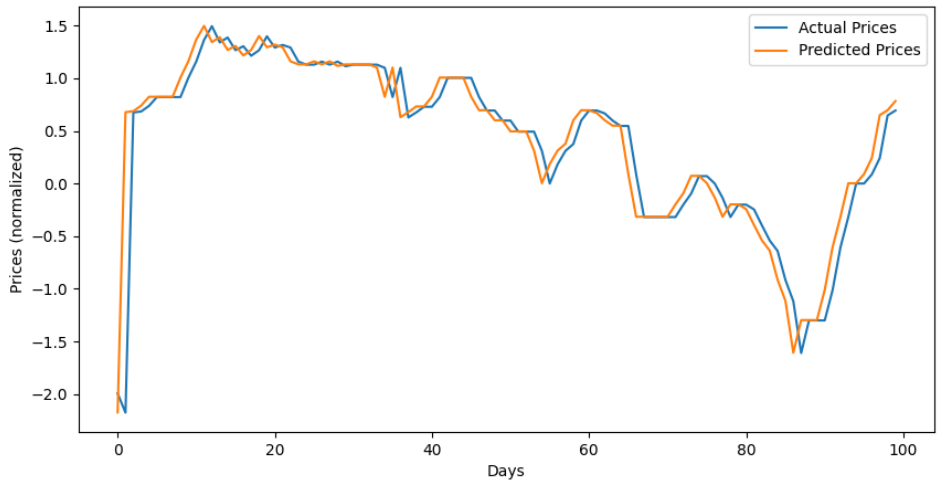


Figure 1.26: Actual vs Predicted Prices on LTG for 100 days

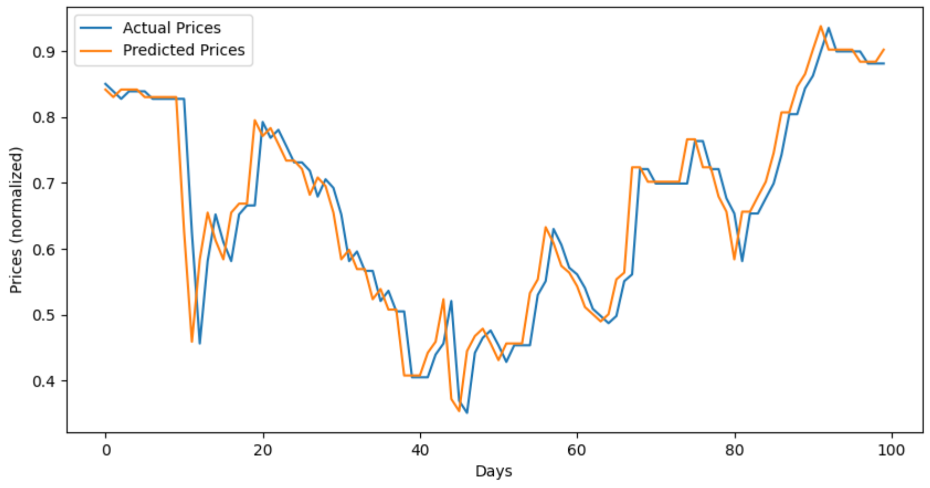


Figure 1.27: Actual vs Predicted Prices on MEG for 100 days

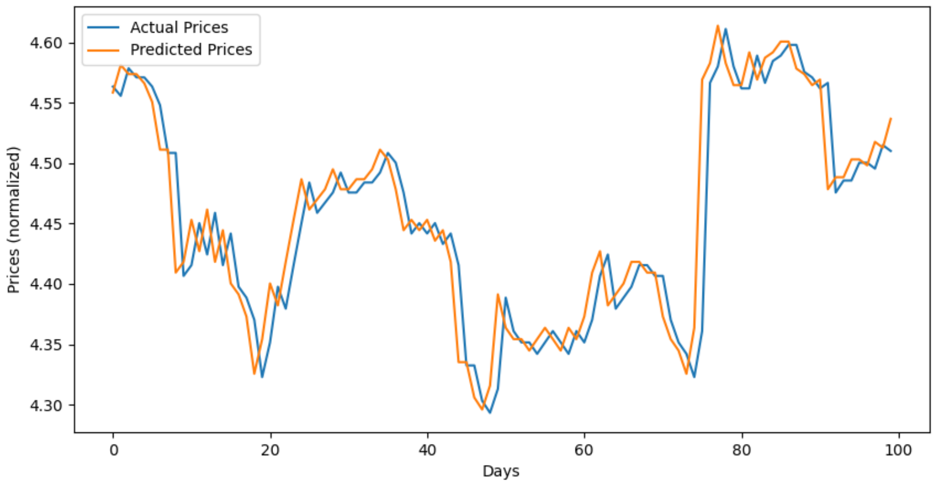


Figure 1.28: Actual vs Predicted Prices on MER for 100 days

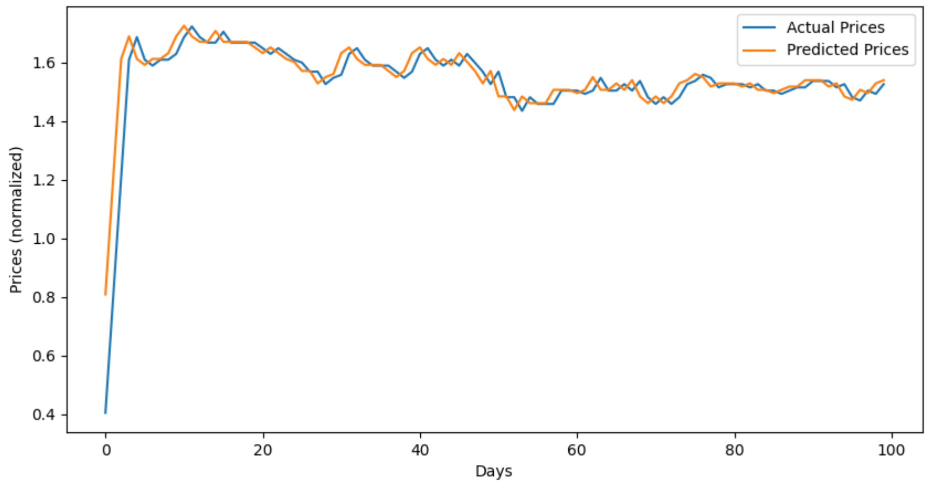


Figure 1.29: Actual vs Predicted Prices on MPI for 100 days

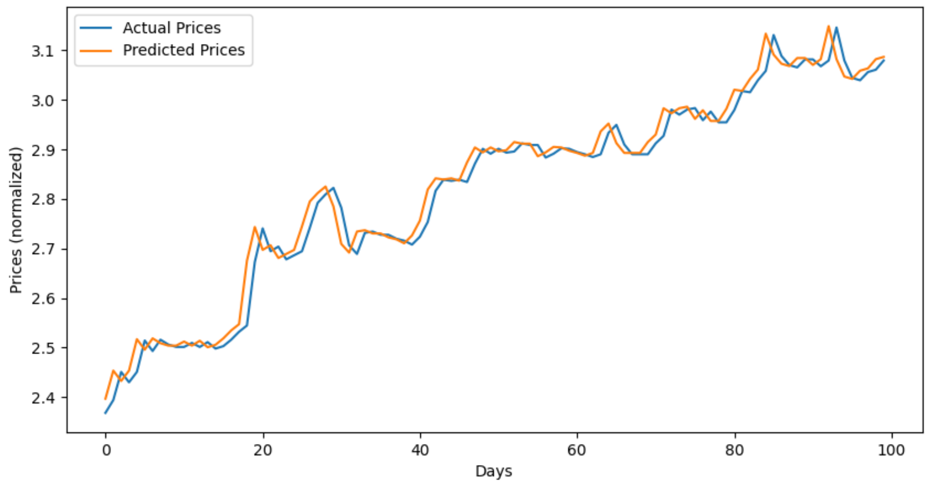


Figure 1.30: Actual vs Predicted Prices on PGOLD for 100 days

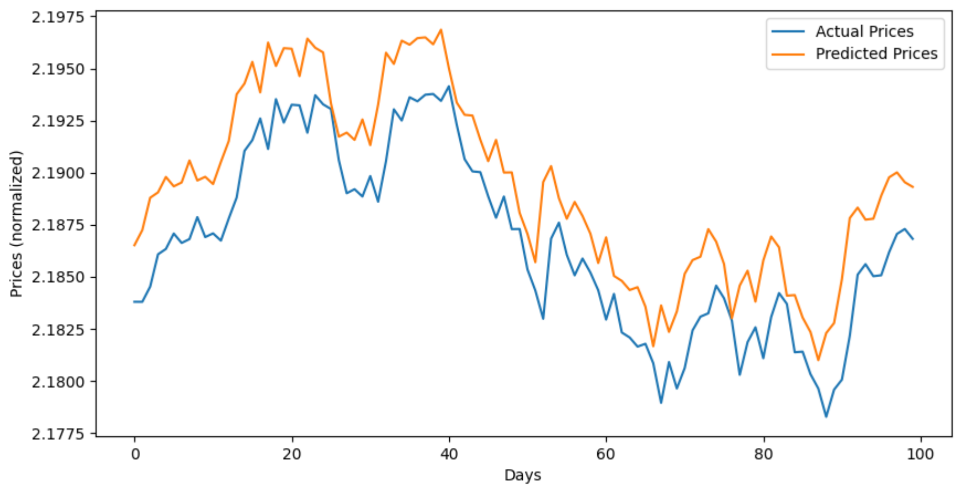


Figure 1.31: Actual vs Predicted Prices on PSEI for 100 days

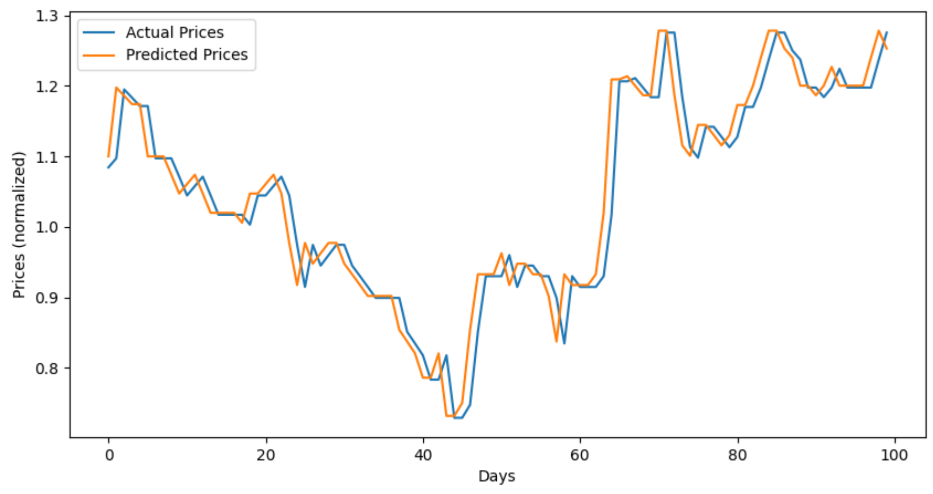


Figure 1.32: Actual vs Predicted Prices on RLC for 100 days

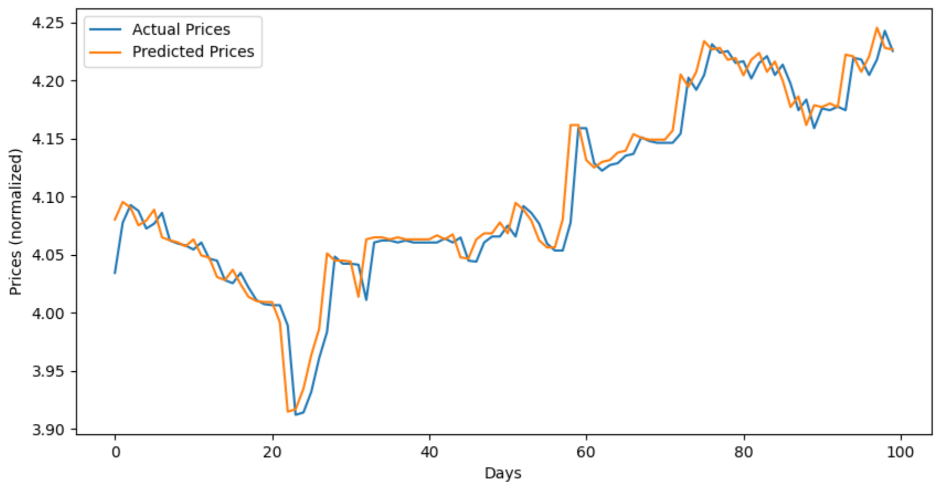


Figure 1.33: Actual vs Predicted Prices on RRHI for 100 days

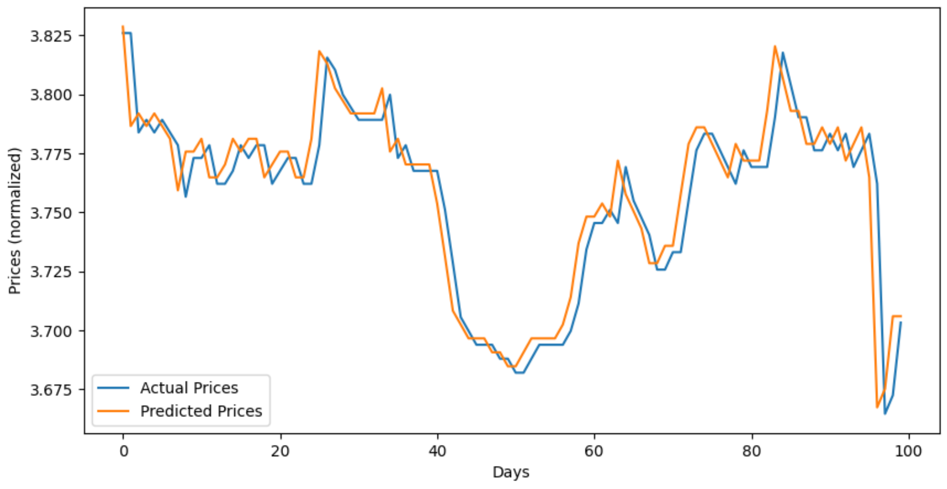


Figure 1.34: Actual vs Predicted Prices on SMC for 100 days

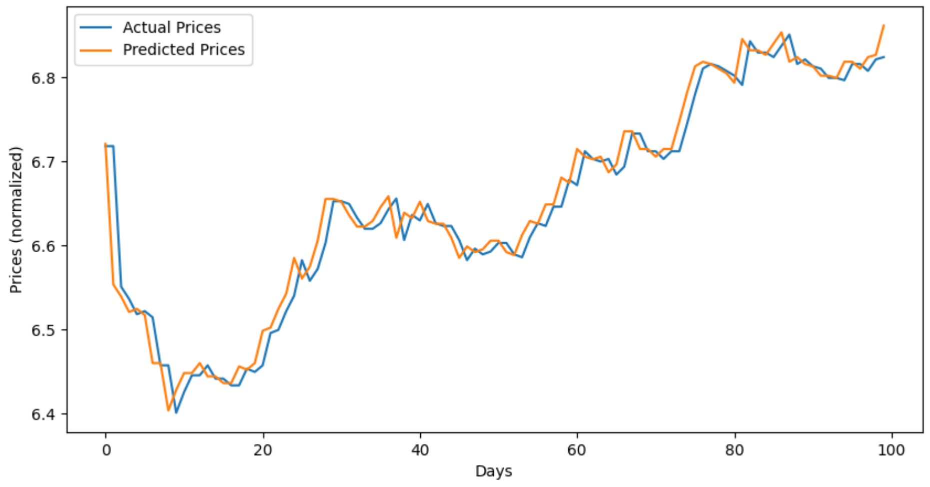


Figure 1.35: Actual vs Predicted Prices on TEL for 100 days

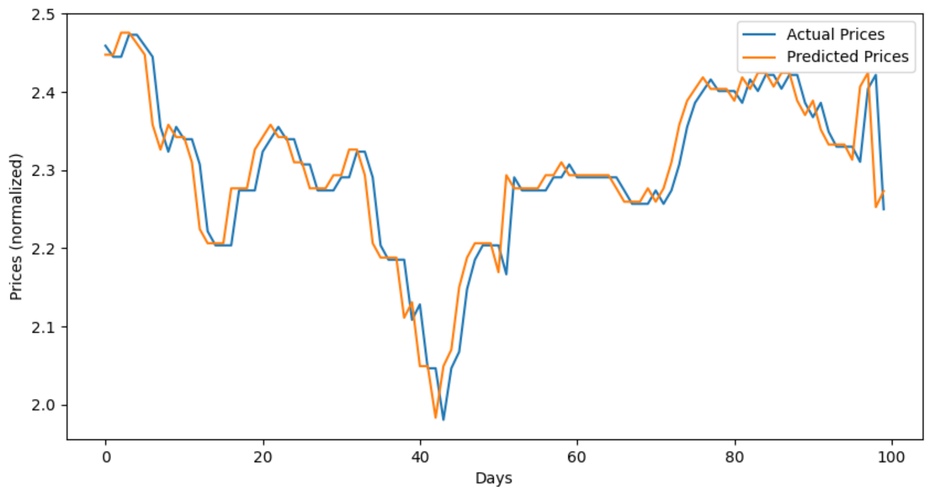


Figure 1.36: Actual vs Predicted Prices on URC for 100 days

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 1.9.

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 1.21, 1.24, 1.25, 1.26, and 1.27, 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 1.10 shows the MAPE scores for the successive predictions of the DMD-LSTM for each days.

Table 1.10: 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 1.10 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 1.37, shown below.

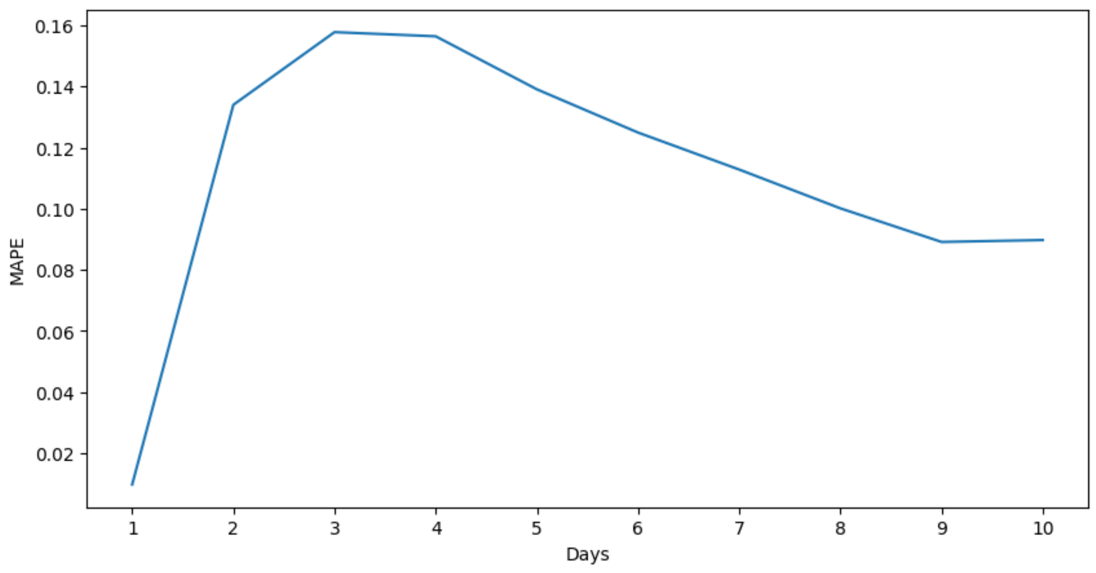


Figure 1.37: MAPE Scores for 1 to 10 (Days) Successive Predictions

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 **??** of this paper.

1.4 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 predefined 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 1.11 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 1.11: 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 1.11 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 alamSYS predictions, as discussed further in the succeeding section.

1.5 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 1.12, 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 1.12: 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 1.38, the positive performance of the alamSYS affirms the findings from Sections 1.3 and 1.4.

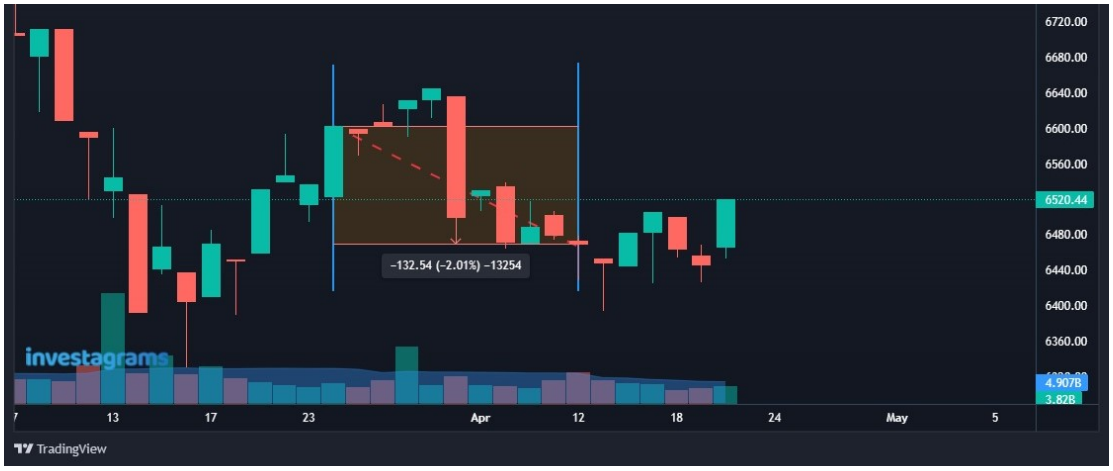


Figure 1.38: PSEI Stock Market Price Trend from March 24 to April 12, 2023

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 1.39 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.

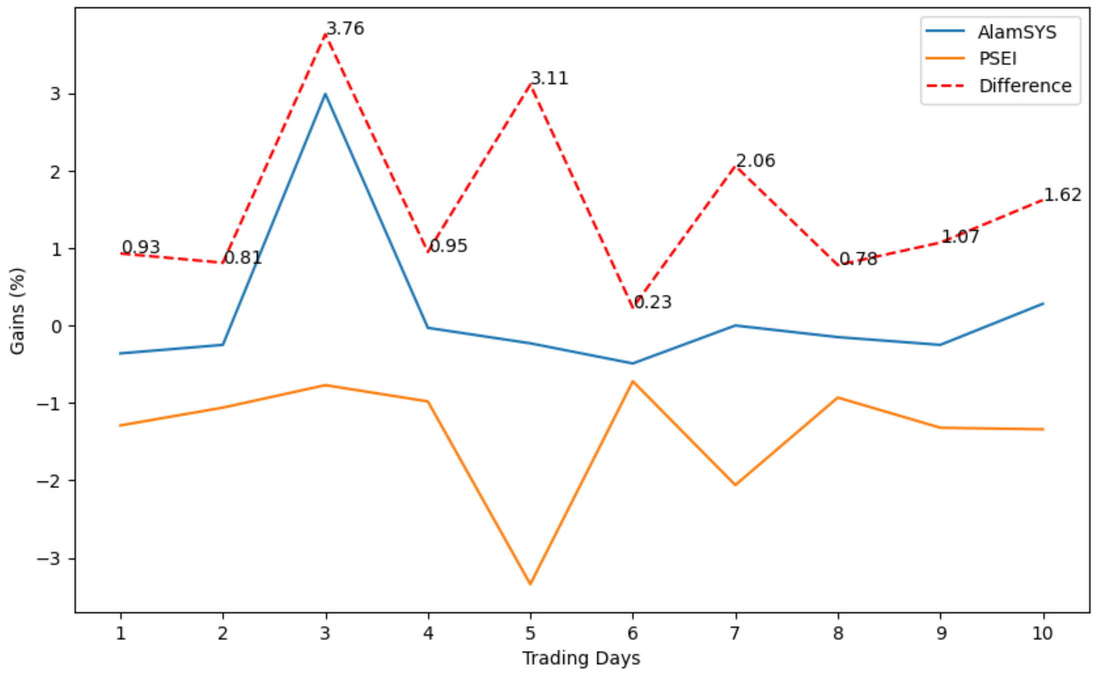


Figure 1.39: Comparison Between the Day-to-day Gains of alamSYS and PSEI

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 1.39.

References

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

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

Mann, J., & Kutz, J. N. (2015, 8). Dynamic mode decomposition for financial trading strategies.

Mitchell, C. (2023). *Drawdown: What it is, risks and examples.* Retrieved April 26, 2023, from https://www.investopedia.com/terms/d/drawdown.asp

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/

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

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/

WallStreetMojo. (n.d.). *Algorithmic trading.* Retrieved April 22, 2023, from https://www.wallstreetmojo.com/algorithmic-trading/

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