Stock Prediction System (StockBuddy)

Submitted in partial fulfillment of the requirements

for the degree of

B.E. Information Technology

By

Daniel Thatu	**211115**
Aryan Thapliyal	**221262**
Darish Dias	**221263**
Royden Dixera	**221265**

Supervisor

Ms. Vijayalakshmi Balaraju

Assistant Professor

SFIT Library 2024-25

Department of Information Technology

St. Francis Institute of Technology

(Engineering College)

University of Mumbai

2024-2025

CERTIFICATE

This is to certify that the project entitled **"Stock Prediction System (Stock-Buddy)"** is a bonafide work of **Daniel Thatu (211115), Aryan Thapliyal (221262) , Darish Dias (221263) , Royden Dixera (221265)** " submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of Bachelor of Engineering in Information Technology.

Ms. Vijayalakshmi Balaraju
(Project Guide)

Dr. Prachi Raut
(Head Of Department)

Dr. Sincy George
(Principal)

SFIT Library 2024-25

Project Approval Report for B.E.

This project report entitled *"Stock Market Prediction"* by ***Daniel Thatu (211115), Aryan Thapliyal (221262) , Darish Dias (221263) & Royden Dixera (221265)*** is approved for the degree of ***Bachelors of Engineering in Information Technology***.

1. _____
(Vijayalakshmi - B)

2. _____
(Dr. Shachi Natu)

Date: 25/04/25

Place: Mumbai

SFIT Library 2024-25

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Daniel Thatu (211115)

Aryan Thapliyal (221262)

Darish Dias (221263)

Royden Dixera (221265)

Date: 25/04/25

Place: Mumbai

SFIT Library 2024-25

ABSTRACT

This paper presents a Stock Prediction System (StockBuddy) that leverages machine learning techniques to forecast future stock prices. By using historical data, technical indicators, and external factors like news sentiment, the system improves prediction accuracy. Backtesting demonstrates its effectiveness, offering a scalable and adaptable solution for real-time stock forecasting.

SFIT Library 2024-25

Contents

1 Introduction **1**

1.1 Introduction 1

1.2 Problem Formulation 2

1.3 Description 2

1.4 Motivation 2

1.5 Proposed Solution 3

1.6 Scope of the Project 3

2 Literature Review **4**

2.1 Existing Work 4

2.1.1 Literature review related to exiting system/methodology 4

2.1.2 Literature review related to existing algorithms 6

2.2 Gap identified 8

3 System Analysis **9**

3.1 Functional Requirements 9

3.2 Non-Functional Requirements 10

3.3 Specific Requirements 10

4 Analysis Modeling **12**

4.1 Functional Modeling 12

4.1.1 DFD:Level 0 12

4.1.2 DFD:Level 1 13

4.1.3 DFD:Level 2 16

4.2 Timeline Chart 19

5 Design **21**

5.1 System Architecture 21

5.2 Graphical User Interface Design 22

6 Implementation **23**

6.1 Working of the Project 23

6.2 Application Performance 24

6.3 Code 24

SFIT Library 2024-25

7 Testing **31**
7.1 Test Cases 31
7.2 Testing Methods Used 31
7.2.1 Unit Testing 31
7.2.2 Integration Testing 31

8 Results and Discussion **33**

9 Conclusion and Future Scope **35**
9.1 Conclusion 35
9.2 Future Scope 35

10 Acknowledgment **36**

11 Literature Cited **37**

SFIT Library 2024-25

List of Figures

4.1 DFD Level 0. 12
4.2 DFD Level 1. 13
4.3 DFD Level 2. 16
4.4 Timeline for Major Project 19
4.5 Task Breakdown for Major Project 20

5.1 Frontend architecture 21
5.2 Backend architecture 21
5.3 Home 22
5.4 Sign In 22

6.1 HomePage 23
6.2 News Page 23
6.3 Profile Page 23
6.4 Prediction Input 23
6.5 Prediction Result 23
6.6 Prediction History 23
6.7 live stock prices 23

8.1 Models comparison table 33
8.2 Predicted prices from research paper model 33
8.3 Predicted prices from Version1 model 34
8.4 Predicted prices from current version model 34

SFIT Library 2024-25

List of Tables

2.1 Comparison of Methodologies 4
2.2 Digger-Guider Framework 6
2.3 SVM-Based Stock Prediction 6
2.4 A Novel Integrated Approach for Stock Prediction 6
2.5 Stock Market Prediction Using Machine Learning Classifiers and Social Media, News 7
2.6 Stock Prediction Using Machine Learning 7

SFIT Library 2024-25

Chapter 1

Introduction

1.1 Introduction

Stock prediction refers to the process of forecasting the future price movements of publicly traded stocks. This involves using various analytical methods to anticipate how a stock's price will change over time. Predicting stock prices is a challenging task due to the highly dynamic and complex nature of financial markets, where a multitude of factors—ranging from company performance to broader economic conditions—affect price fluctuations.

The primary goal of stock prediction is to provide traders, investors, and financial institutions with insights that can guide their buying and selling decisions. If future stock movements can be accurately predicted, it can lead to substantial profits by capitalizing on market trends, while also helping to mitigate potential risks.

In recent years, Artificial Intelligence (AI) and Machine Learning (ML) have become central to stock prediction efforts due to their ability to process vast amounts of data and identify patterns that are often too complex for traditional methods. AI-powered stock prediction systems can analyze historical data, technical indicators, and external factors, and automatically adjust their predictions based on new inputs.

Key advantages of using AI for stock prediction include:

1. **Data Processing Power**: AI algorithms can quickly process massive datasets, including historical stock prices, trading volumes, and economic indicators, to identify trends and patterns that may signal future price movements.

2. **Predictive Accuracy**: Through machine learning models, AI can improve its predictions over time. By continuously learning from market data, AI systems can become better at recognizing predictive signals.

3. **Real-time Decision Making**: AI systems can make real-time predictions and update recommendations as new information becomes available, such as breaking news, changes in economic data, or fluctuations in stock prices.

4. **Risk Management**: AI can help assess the risks associated with stock investments by analyzing factors like market volatility, economic conditions, and company-specific data. This helps investors make informed decisions that balance potential returns with the risks involved.

SIT Library 2024-25

1.2 Problem Formulation

The primary challenge this project addresses is the difficulty in accurately predicting stock prices due to the volatile nature of the stock market. Existing platforms and models lack advanced features such as integrating diverse data sources, adapting to rapid market changes, and providing real-time interpretability. These limitations make stock prediction less accurate and more computationally demanding.

1.3 Description

Stock prediction systems leverage financial indicators, historical data, and machine learning to forecast price movements, thereby enhancing investment strategies and risk management. By identifying market patterns and incorporating sentiment analysis from news, social media, and financial reports, these systems provide valuable insights for traders.Utilizing time series analysis, regression models, neural networks, and natural language processing (NLP), such systems interpret complex datasets.Sentiment analysis gauges market mood, while technical indicators identify trends and potential reversals. Real-time data feeds enable dynamic adjustments, offering timely insights that assist in decision-making. Automated prediction capabilities reduce human biases and errors, ensuring more objective and data-driven investment decisions. Additionally, integrated risk assessment models evaluate potential losses and returns, supporting balanced portfolio management. The combination of machine learning, big data analytics, and sentiment analysis creates a powerful framework for accurate market forecasts. These advanced systems help investors navigate volatile markets with greater precision, ultimately improving financial planning and decision-making.

SFIT Library 2024-25

1.4 Motivation

The motivation for stock prediction lies in the potential for significant financial gains and improved decision-making. By accurately forecasting stock price movements, investors and traders can better time their buy and sell decisions, potentially leading to increased profits and minimized losses. In highly competitive markets, the ability to predict trends offers a strong competitive advantage.

Stock prediction also plays a crucial role in managing risks. Financial markets are volatile and affected by numerous factors, ranging from economic reports to geopolitical events. Predictive models help investors assess and mitigate risks by providing data-driven insights, allowing for more informed and strategic decision-making.

Additionally, advancements in Artificial Intelligence (AI) and Machine Learning (ML) have motivated the development of more sophisticated stock prediction systems. AI's ability to process large datasets and identify complex patterns that are beyond traditional methods makes it an attractive tool for improving prediction accuracy and real-time market analysis.

1.5 Proposed Solution

Integrating several data sources, including historical price patterns, high-frequency trading data, alternative data from social media, and sentiment analysis of news, into a hybrid machine learning framework is one suggested way to enhance stock market prediction. The system can better capture sentiment-driven fluctuations and market complexities by utilizing deep learning models, such as transformer-based models or Long Short-Term Memory (LSTM) networks, and sophisticated noise filtering approaches. Furthermore, overfitting can be minimized and real-time processing improved by maximizing computational efficiency through feature selection and dimensionality reduction. Inconsistencies in alternative data sources will be further addressed by creating a strong sentiment quantification technique and improving data integration tactics, which will result in more precise and trustworthy stock trend forecasts.

1.6 Scope of the Project

This project will develop a platform that integrates advanced natural language processing (NLP) techniques and Long Short-Term Memory (LSTM) networks for stock price prediction. The system will leverage NLP to analyze market sentiment from news and social media, while LSTM will be used to model sequential historical price data. Designed for stock market forecasting, the platform will have the potential to expand into other financial applications, such as portfolio management or risk assessment.

SFIT Library 2024-25

Chapter 2

Literature Review

2.1 Existing Work

2.1.1 Literature review related to exiting system/methodology

Table 2.1: Comparison of Methodologies

Sr. No.	Title of Paper	Review	Analysis/Limitations
[1]	Digger-Guider: High-Frequency Factor Extraction for Stock Trend Prediction	Involves training two interacting models (Digger and Guider) with a mutual distillation process that increases computational and implementation complexity.	1. Handles high-frequency data effectively. 2. Better suited for longer sequences. 3. Testing data used has inconsistent lengths. 4. Supervised learning requires a large dataset. 5. Long sequences can be hard to summarize.
[2]	Support Vector Machine (SVM): A supervised learning model that classifies data points by finding the optimal hyperplane in a high-dimensional space	SVM provides effective classification for linear and non-linear data using kernel functions, achieving high accuracy in stock trend prediction.	1. Robust in high-dimensional data classification. 2. Effective for binary and multi-class tasks. 3. High computational cost for large datasets. 4. Limited interpretability, making classification decisions hard to explain. 5. Struggles with large, noisy datasets in high-frequency trading.

SFIT Library 2024-25

Sr. No.	Title of Paper	Review	Analysis/Limitations
[3]	Modal Decomposition Technology and Machine Learning	Combines modal decomposition with machine learning to detect hidden patterns, enhancing trend prediction by focusing on significant signals.	1. Boosts accuracy by isolating key data signals. 2. Useful for denoising in volatile markets. 3. May need additional preprocessing for complex data. 4. Computationally demanding, hindering real-time applicability. 5. Less suitable for high-frequency, real-time trading.
[4]	Ensemble Methods (Random Forest, NLP, SVM, Regression Tree)	Uses multiple models together to increase prediction accuracy, often surpassing individual models in performance.	1. Provides robust predictions by combining model strengths. 2. Reduces overfitting with model diversity. 3. High complexity, increasing computational demands. 4. Potential interpretability issues due to model combinations. 5. Overfitting risk if ensemble isn't balanced.
[5]	Hybrid Model for Stock Prediction Combining NLP and LSTM	The study explores a hybrid model that combines NLP for sentiment analysis and LSTM for sequential data processing, aiming to enhance stock price predictions by incorporating news sentiment.	1. Effectively integrates textual data with numerical data for predictions. 2. Shows improvement in prediction accuracy over standalone LSTM. 3. Limited scalability for real-time processing due to computational demands. 4. Sentiment analysis accuracy affects overall model performance. 5. The model's performance can be sensitive to changes in market conditions.

SFIT Library 2024-25

2.1.2 Literature review related to existing algorithms

Table 2.2: Digger-Guider Framework

Sr No.	**1**
Algorithm Used	Digger-Guider Framework
Algorithm Complexity	High
Advantages Over Other Algorithms	Handles High-Frequency Data, Noise Reduction
Technical Challenges	Overfitting, High Computational Cost
Need for Implementation in Stock Prediction	High-Frequency Trading, Improved Prediction Accuracy

Table 2.3: SVM-Based Stock Prediction

Sr No.	**2**
Algorithm Used	SVM-Based Stock Prediction
Algorithm Complexity	Moderate
Advantages Over Other Algorithms	Effective in Small Datasets, Global Optimal Solution
Technical Challenges	Kernel Selection, Scalability
Need for Implementation in Stock Prediction	Lower-Frequency Data, Simpler to Implement

Table 2.4: A Novel Integrated Approach for Stock Prediction

Sr No.	**3**
Algorithm Used	Modal Decomposition Technology and Machine Learning
Algorithm Complexity	Moderate to High
Advantages Over Other Algorithms	Signal Processing and Machine Learning Integration
Technical Challenges	Decomposition Mode Selection, High Computational Cost
Need for Implementation in Stock Prediction	Noisy Environments, Capturing Complex Dynamics

SFIT Library 2024-25

Table 2.5: Stock Market Prediction Using Machine Learning Classifiers and Social Media, News

Sr No.	**4**
Algorithm Used	Ensemble Methods (e.g., Random Forest, Gradient Boosting)
Algorithm Complexity	High
Advantages Over Other Algorithms	Improved Accuracy, Reduced Overfitting
Technical Challenges	Hyperparameter Tuning, Overfitting Risk
Need for Implementation in Stock Prediction	Robust Models, Diverse Datasets

Table 2.6: Stock Prediction Using Machine Learning

Sr No.	**5**
Algorithm Used	Neural Networks (e.g., LSTM, CNN), Linear Regression
Algorithm Complexity	High
Advantages Over Other Algorithms	Complex Patterns, Adaptability
Technical Challenges	Computational Power, Overfitting, Interpretability Issues
Need for Implementation in Stock Prediction	Temporal Dependencies, Real-Time Prediction

SFIT Library 2024-25

2.2 Gap identified

[1] Digger-Guider Framework for High-Frequency Factor Extraction Gap: Limited use of multi-modal data integration. While this study handles high-frequency data, it does not incorporate diverse data sources (e.g., social media, news) that could enhance the model's predictive capabilities.

[2] Stock Market Prediction Using Linear Regression and SVM Gap: Use of outdated or incomplete datasets. Linear regression and SVM may lack robustness for real-time or dynamic market data, as they often rely on static data inputs and simpler data structures.

[3] Integrated Approach Based on Modal Decomposition Technology and Machine Learning Gap: Absence of adaptation to changing market dynamics. Modal decomposition aids in noise reduction but does not actively adjust for rapidly shifting trends, making it less adaptable to fluctuating market conditions.

[4] Ensemble Methods Using Social Media and News Gap: Underexplored impact of transaction costs and market impact. Although ensemble methods improve accuracy with additional data inputs, they lack consideration of transaction costs, which can affect profitability in real-world trading.

[5] Neural Networks (LSTM, CNN) and Linear Regression for Stock Prediction Gap: Limited focus on model interpretability and explainability. Neural networks capture complex patterns but are often "black boxes," making it difficult to interpret the model's decision-making process, which is crucial in finance.

This listing directly relates each gap to the corresponding paper, offering a clear view of limitations in each approach.

SFIT Library 2024-25

Chapter 3
System Analysis

3.1 Functional Requirements

- **Performance:** The system should process data and generate predictions in real-time, with response times under 2 seconds for user queries.
- **Scalability:** The architecture must support an increasing number of users and data volume without degradation in performance.
- **Reliability:** The system should maintain an uptime of at least 99.5%, ensuring availability during trading hours.
- **Security:** Implement industry-standard security measures, including data encryption, secure authentication, and protection against unauthorized access.
- **Usability:** The user interface should be intuitive and user-friendly, allowing users to navigate easily and access features with minimal training.
- **Compatibility:** The system should be compatible with major web browsers and devices, ensuring accessibility for all users.
- **Maintainability:** The codebase should be modular and well-documented, allowing for easy updates and maintenance by developers.
- **Backup and Recovery:** Implement regular backup processes and disaster recovery plans to prevent data loss and ensure business continuity.
- **Compliance:** The system must comply with relevant financial regulations and data protection laws (e.g., GDPR, SEC regulations).

SFIT Library 2024-25

3.2 Non-Functional Requirements

- **Accessibility:** The platform should be accessible via both web browsers and mobile devices, supporting platforms like Android and iOS.
- **Usability:** The user interface should be intuitive and user-friendly, allowing users to navigate videos and summaries seamlessly.
- **Performance:** The system should summarize videos quickly, and the quizzes should be generated without noticeable delays.
- **Maintainability:** The architecture should be simple and modular so that it can be easily updated and modified for future improvements.
- **Scalability:** The platform should be scalable, supporting a wide range of video types and lengths without a significant drop in performance.
- **Sustainability:** The system should maintain consistent performance over time without the need for frequent manual intervention.
- **User-Friendly Graphical Interface:** The interface should visually guide users through video summaries, quizzes, and phrase searches in a clear and understandable way.

3.3 Specific Requirements

Hardware

- **CPU:** Intel Core i5 or equivalent.
- **GPU:** Integrated Graphics (Minimum 1GB).
- **RAM:** Minimum 8GB.
- **Storage:** Minimum 500MB of free space.
- **Operating System:** Windows 10 or later, macOS 10.14 or later.

Software

- **Programming Languages:** Python for backend NLP and LSTM, JavaScript, Tailwind, Shadcnui for frontend development.
- **Frameworks:** ReactJS for frontend development, Flask, Express for backend server.

SIT Library 2024-25

- **APIs:**
  - Tradingview, Finnhub, Yahoofinance
- **Databases:** MongoDB.
- **Libraries:**
  - NLTK or other NLP libraries for text processing.
  - ReactJS for building the user interface.
- **Browser Compatibility:** Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge.

SFIT Library 2024-25

Chapter 4

Analysis Modeling

4.1 Functional Modeling

4.1.1 DFD:Level 0

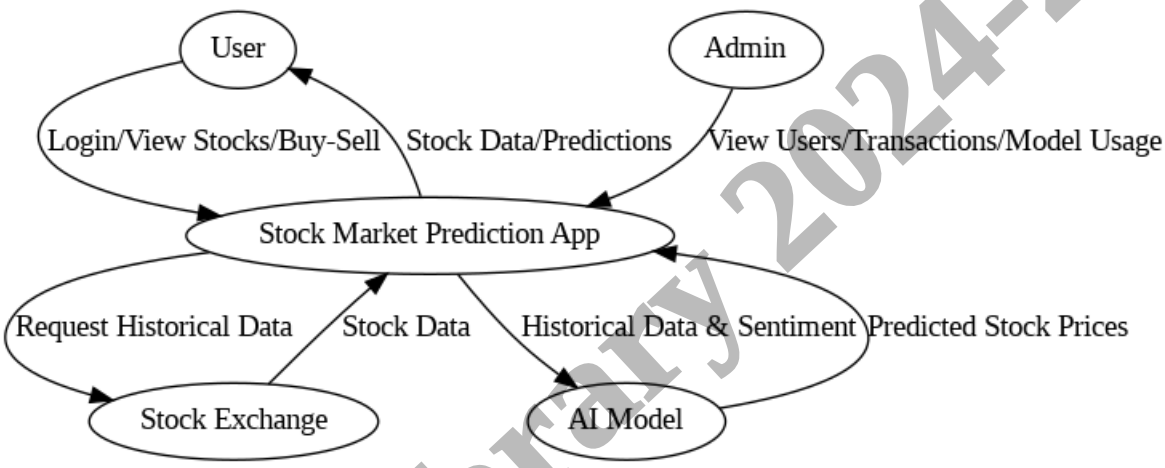


Figure 4.1: DFD Level 0.

In the above DFD, the overview of the data flow diagram is being described where the user can view,buy/sell stocks or predict selected stocks in the application which is fetched by the application.Admin manages/views the application monitoring user activity as well as transactional usage.

SFIT Library 2024-25

4.1.2 DFD:Level 1

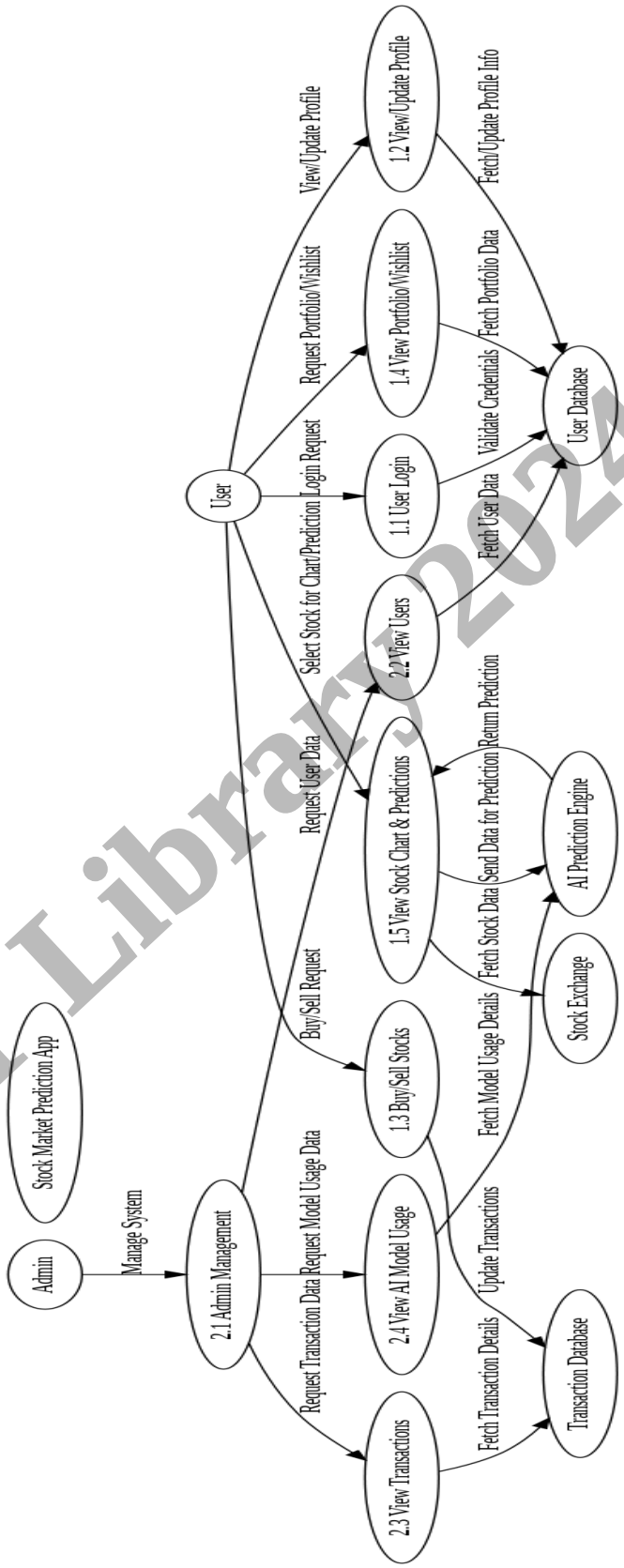


Figure 4.2: DFD Level 1.

SFIT Library 2024-25

DFD Level 1 Explanation

1.1 Stock Market Prediction App

This is the main system that manages the connections and data flow between different processes, including interactions for Admin and User functionalities.

1.2 Admin

Admins manage the system and have access to functions such as viewing transaction data and monitoring AI model usage.

- **View Transactions (2.3)**: The Admin requests transaction details, which are retrieved from the Transaction Database.
- **View AI Model Usage (2.4)**: The Admin can check AI model usage data, with updates sent to track model performance.
- **View Users (2.2)**: The Admin can view details about users by accessing data from the User Database.

1.3 User

Users interact with various system functions:

- **User Login (1.1)**: Users send a login request, which checks their credentials with the User Database. Successful validation grants them access to the system.
- **View/Update Profile (1.2)**: Users can request to view or update their profile. This involves fetching current profile data or updating the profile data in the User Database.
- **View Portfolio/Wishlist (1.4)**: Users can view their portfolio and wishlist, with data fetched from the User Database.
- **View Stock Chart & Predictions (1.5)**: Users select a stock to view its chart and predictions. The app fetches stock data from the Stock Exchange and then sends the data to the AI Prediction Engine for generating predictions.
- **Buy/Sell Request**: Users can submit buy or sell requests, which interact with both the Transaction Database (to record transactions) and the Stock Exchange (to fetch stock data).

1.4 Transaction Database

Holds transaction details, which are fetched by both Admins (for monitoring transactions) and Users (to execute buy/sell requests).

SFIT Library 2024-25

1.5 User Database

Stores user-related information, such as login credentials, profile data, portfolio, and wishlist. Both Admins and Users access this data based on their respective privileges.

1.6 Stock Exchange

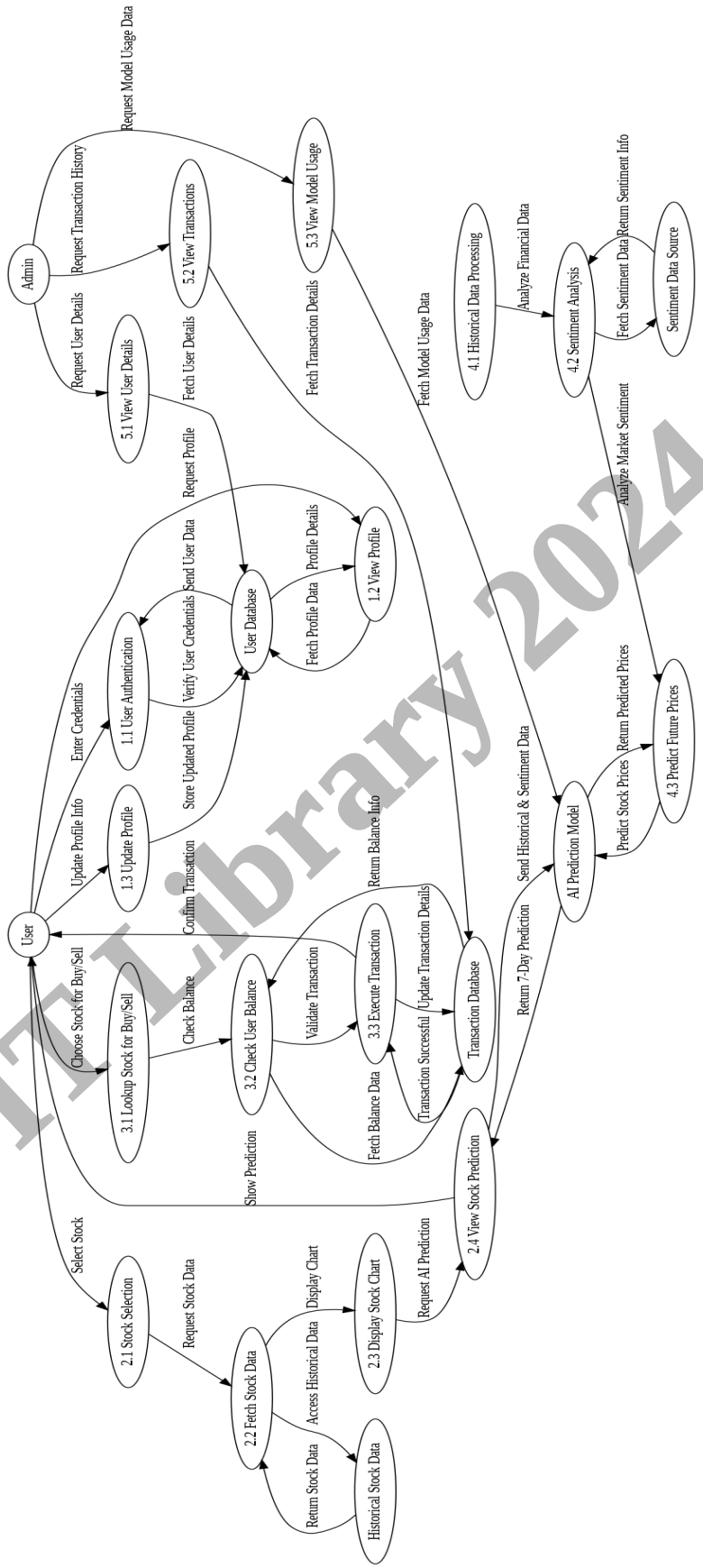
Provides real-time stock data, which the app fetches for stock charting and predictions requested by Users.

1.7 AI Prediction Engine

Receives stock data and returns prediction results, which are then displayed to Users.

SFIT Library 2024-25

4.1.3 DFD:Level 2



SFIT Library 2024-25

Figure 4.3: DFD Level 2.

DFD Level 2 Explanation

2.1 User Authentication (1.1)

Users enter their credentials, which are sent to the User Database for verification. Upon successful validation, users are authenticated.

2.2 View Profile (1.2)

Users can view their profile, and the app retrieves this data from the User Database for display.

2.3 Update Profile (1.3)

Users can make changes to their profile. The updated information is then sent to the User Database to replace the previous data.

2.4 Stock Selection (2.1)

Users select a stock to trade. This selection triggers a request for stock data, which is fetched from Historical Stock Data.

2.5 Fetch Stock Data (2.2)

The app retrieves stock data from Historical Stock Data. This includes returning both historical and real-time stock data for further processing.

2.6 Display Stock Chart (2.3)

With the fetched stock data, the app displays a stock chart. If prediction is requested, the app sends the data to the AI Prediction Model.

2.7 View Stock Prediction (2.4)

Users can view stock predictions generated by the AI Prediction Model. The model analyzes both historical data and sentiment information to forecast stock prices.

2.8 Execute Transaction (3.3)

After a user has selected stocks, they check their balance and validate the transaction. Upon successful validation, transaction details are stored in the Transaction Database.

SFIT Library 2024-25

2.9 View User Details (5.1)

Admins can view user details by sending a request to the User Database, where user profiles and activities are stored.

2.10 View Transactions (5.2)

Admins can view transaction histories by fetching data from the Transaction Database, which keeps all transaction records.

2.11 View Model Usage (5.3)

Admins can monitor the usage of the AI model, with data fetched from Model Usage Data sources.

2.12 Historical Data Processing (4.1)

This node analyzes historical financial data to prepare it for the AI model's prediction processes.

2.13 Sentiment Analysis (4.2)

The app fetches market sentiment data from the Sentiment Data Source and analyzes it. This sentiment data is then used by the AI Prediction Model to improve prediction accuracy.

2.14 Predict Future Prices (4.3)

The AI Prediction Model combines both historical and sentiment data to predict future stock prices, which are then returned to users for decision-making.

SFIT Library 2024-25

4.2 Timeline Chart

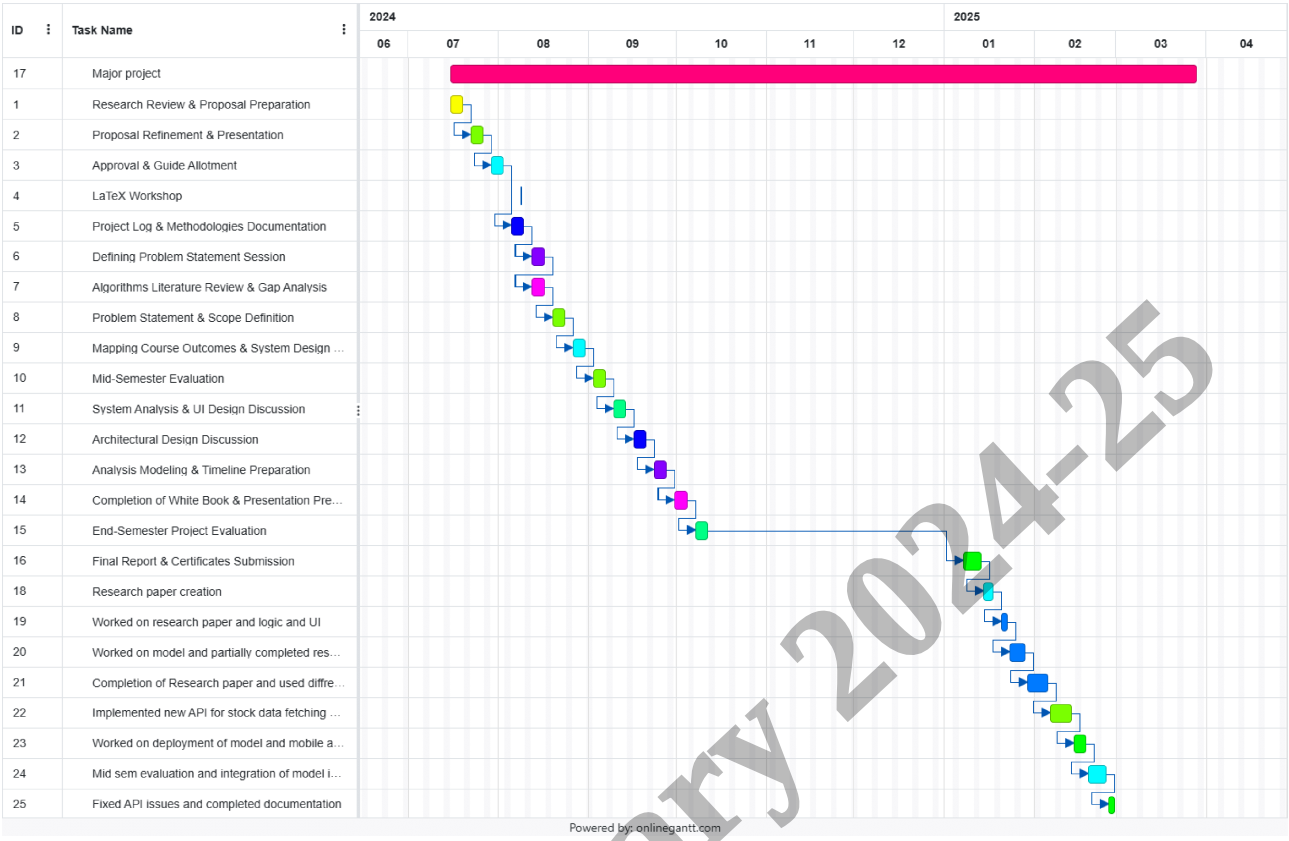


Figure 4.4: Timeline for Major Project

SFIT Library 2024-25

	Name	Start	Finish	Duration
0	Major project	2024-07-15	2025-03-28	185 day
1	Research Review & Proposal Preparation	2024-07-15	2024-07-19	5 day
2	Proposal Refinement & Presentation	2024-07-22	2024-07-26	5 day
3	Approval & Guide Allotment	2024-07-29	2024-08-02	5 day
4	LaTeX Workshop	2024-08-08	2024-08-08	1 day
5	Project Log & Methodologies Documentation	2024-08-05	2024-08-09	5 day
6	Defining Problem Statement Session	2024-08-12	2024-08-16	5 day
7	Algorithms Literature Review & Gap Analysis	2024-08-12	2024-08-16	5 day
8	Problem Statement & Scope Definition	2024-08-19	2024-08-23	5 day
9	Mapping Course Outcomes & System Design Discussion	2024-08-26	2024-08-30	5 day
10	Mid-Semester Evaluation	2024-09-02	2024-09-06	5 day
11	System Analysis & UI Design Discussion	2024-09-09	2024-09-13	5 day
12	Architectural Design Discussion	2024-09-16	2024-09-20	5 day
13	Analysis Modeling & Timeline Preparation	2024-09-23	2024-09-27	5 day
14	Completion of White Book & Presentation Preparation	2024-09-30	2024-10-04	5 day
15	End-Semester Project Evaluation	2024-10-07	2024-10-11	5 day
16	Final Report & Certificates Submission	2025-01-07	2025-01-13	5 day
18	Research paper creation	2025-01-14	2025-01-17	4 day
19	Worked on research paper and logic and UI	2025-01-20	2025-01-22	3 day
20	Worked on model and partially completed research paper	2025-01-23	2025-01-28	4 day
21	Completion of Research paper and used diffrent API for model	2025-01-29	2025-02-05	6 day
22	Implemented new API for stock data fetching and Sentiment analyzer	2025-02-06	2025-02-13	6 day
23	Worked on deployment of model and mobile application	2025-02-14	2025-02-18	3 day
24	Mid sem evaluation and integration of model in project	2025-02-19	2025-02-25	5 day
25	Fixed API issues and completed documentation	2025-02-26	2025-02-28	3 day

Figure 4.5: Task Breakdown for Major Project

SFIT Library 2024-25

Chapter 5

Design

5.1 System Architecture

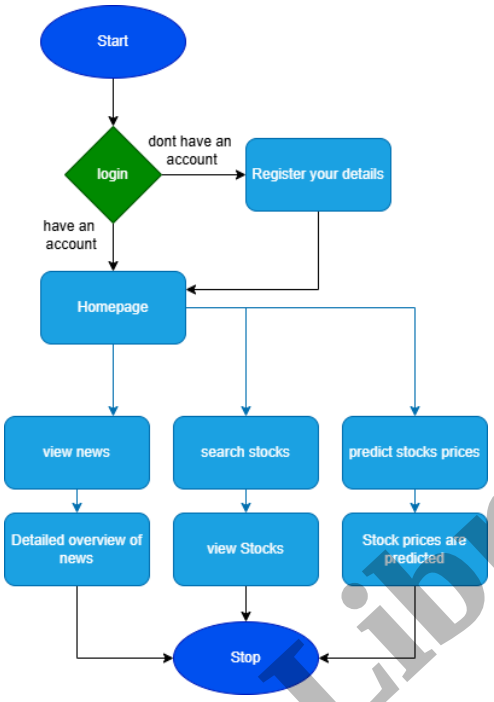


Figure 5.1: Frontend architecture

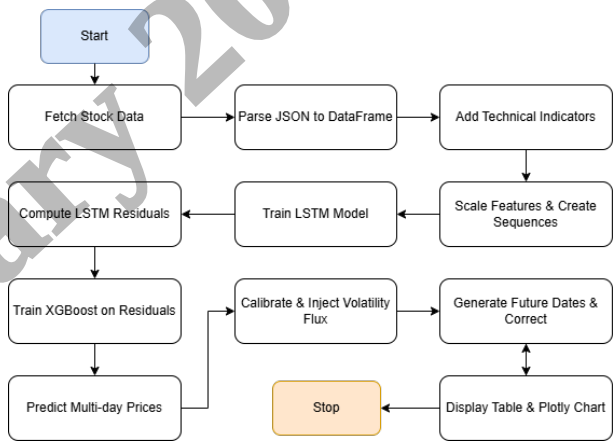


Figure 5.2: Backend architecture

SFIT Library 2024-25

5.2 Graphical User Interface Design

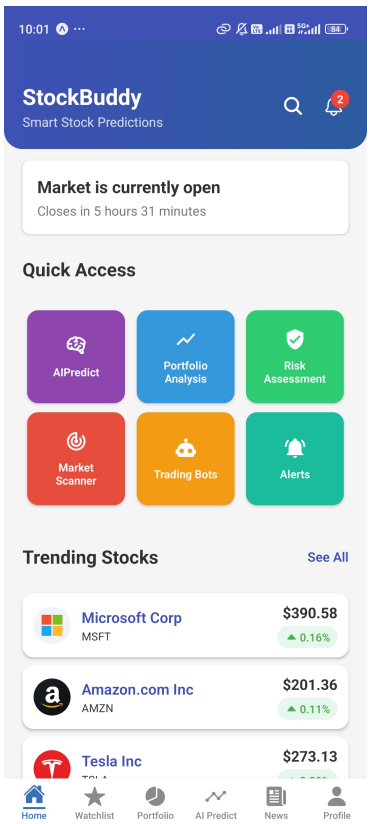


Figure 5.3: Home

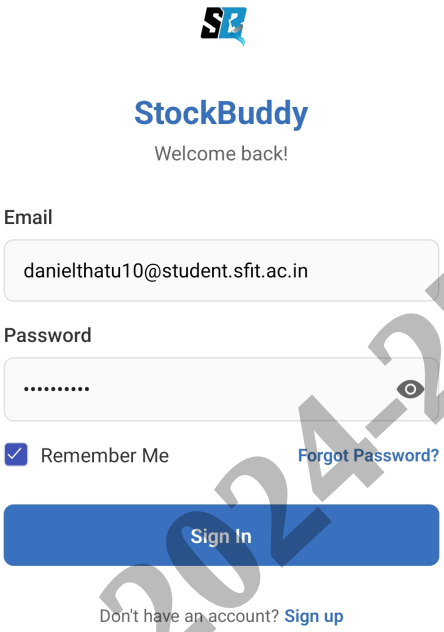


Figure 5.4: Sign In

SFIT Library 2024-25

Chapter 6

Implementation

6.1 Working of the Project

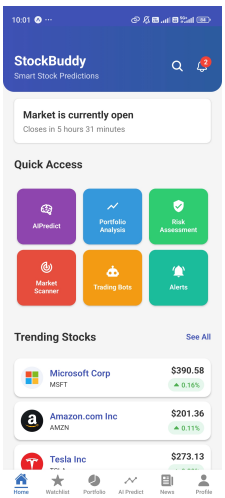


Figure 6.1: HomePage

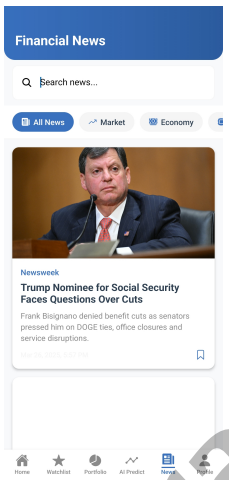


Figure 6.2: News Page

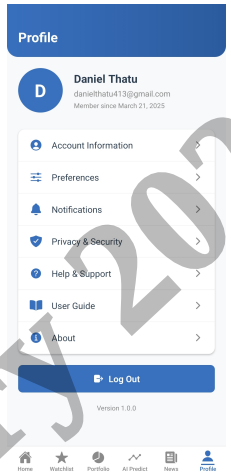


Figure 6.3: Profile Page

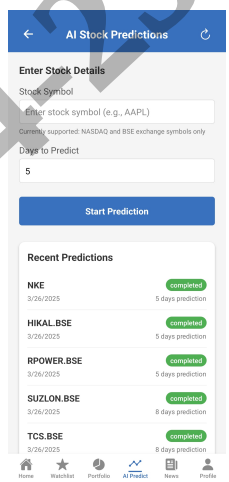


Figure 6.4: Prediction Input

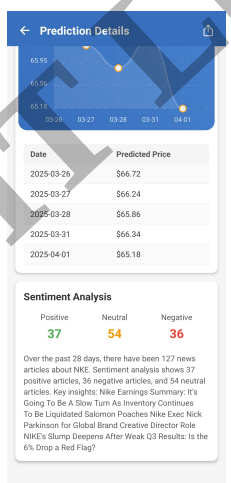


Figure 6.5: Prediction Result

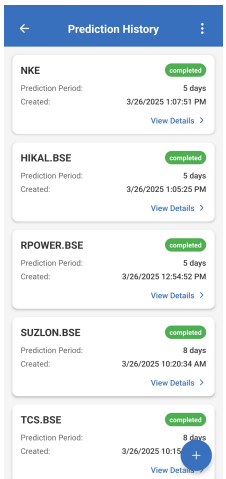


Figure 6.6: Prediction History

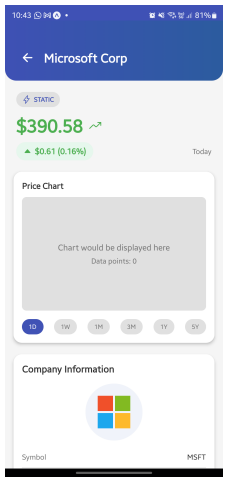


Figure 6.7: live stock prices

SFIT Library 2024-25

6.2 Application Performance

The application demonstrated stable performance under typical usage conditions, with quick response times and smooth navigation across modules. Efficient backend integration ensured reliable data processing and minimal latency during user interactions. The system maintained consistency during multiple concurrent operations, reflecting its scalability and robustness. Overall, the application delivered a responsive, user-friendly experience with optimized resource usage.

6.3 Code

Training Function

```
def train_lstm(X_train, y_train, time_step=30, stop_requested_callback=None):
    """Train an optimized LSTM model for faster yet accurate predictions"""
    n_features = X_train.shape[2]
    X_train = X_train.reshape(X_train.shape[0], time_step, n_features)

    model = Sequential([
        LSTM(64, return_sequences=True, input_shape=(time_step, n_features), recurrent_dropout=0.0),
        LSTM(64, return_sequences=True, recurrent_dropout=0.0),
        Dropout(0.2),
        LSTM(32, return_sequences=True),
        LSTM(32, return_sequences=False),
        Dropout(0.2),
        Dense(16, activation='relu'),
        Dense(16, activation='relu'),
        Dense(1)
    ])

    from tensorflow.keras.optimizers import Adam
    from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping, Callback

    class StopCallback(Callback):
        def on_epoch_end(self, epoch, logs=None):
            if stop_requested_callback and stop_requested_callback():
                self.model.stop_training = True
                print("Training stopped early by user request")

    optimizer = Adam(learning_rate=0.001)
```

SFIT Library 2024-25

```
28      model.compile(optimizer=optimizer, loss='mean_squared_error',
            run_eagerly=True)
29
30      reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.3,
            patience=5, min_lr=0.0001, verbose=1)
31      early_stopping = EarlyStopping(monitor='val_loss', patience=10,
            restore_best_weights=True, verbose=1)
32
33      callbacks = [reduce_lr, early_stopping]
34      if stop_requested_callback:
35          callbacks.append(StopCallback())
36
37      model.fit(X_train, y_train, epochs=30, batch_size=64,
            validation_split=0.2, callbacks=callbacks, verbose=1)
38      return model
39
40  def train_xgboost(X_train, residuals, stop_requested_callback=None):
41      """Train an optimized XGBoost model on LSTM residuals"""
42      if stop_requested_callback and stop_requested_callback():
43          print("XGBoost training cancelled due to stop request")
44          return None
45
46      params = {
47          'objective': 'reg:squarederror',
48          'n_estimators': 300,
49          'learning_rate': 0.1,
50          'max_depth': 6,
51          'subsample': 0.8,
52          'colsample_bytree': 0.8,
53          'min_child_weight': 3,
54          'gamma': 0.2,
55          'reg_alpha': 0.2,
56          'reg_lambda': 1.2,
57          'tree_method': 'hist',
58          'grow_policy': 'lossguide'
59      }
60
61      if stop_requested_callback:
62          class StopCallbackHandler(xgb.callback.TrainingCallback):
63              def after_iteration(self, model, epoch, evals_log):
64                  if stop_requested_callback():
65                      print("XGBoost training stopped by user request")
66                      return True
67                  return False
68
69          xgb_model = xgb.XGBRegressor(**params)
70          xgb_model.set_params(callbacks=[StopCallbackHandler()])
71          xgb_model.fit(X_train, residuals)
```

SFIT Library 2024-25

```
    else:
        xgb_model = xgb.XGBRegressor(**params)
        xgb_model.fit(X_train, residuals, eval_metric=['rmse'],
            early_stopping_rounds=20,
                      verbose=True, eval_set=[(X_train, residuals)])

    return xgb_model
```

Prediction Function

```
def predict_stock_price(lstm_model, xgb_model, data, scaler,
    time_step=30, days_ahead=5, stop_requested_callback=None):
    """Make predictions using both LSTM and XGBoost models with
        proper anchoring to last actual price"""
    # Check if we should stop before starting
    if stop_requested_callback and stop_requested_callback():
        print("Prediction cancelled due to stop request")
        return None

    n_features = data.shape[1]
    temp_input = data[-time_step:].tolist()

    # Get the actual last closing price from the original data (not
        scaled)
    last_actual_close = scaler.inverse_transform(np.array([[data[-1,
        0]]]))[0][0]
    print(f"Using last actual closing price: ${last_actual_close:.2f}
        as base for predictions")

    # Calculate historical volatility (std dev of returns) to use for
        realistic fluctuations
    # Get the original close prices (not scaled)
    original_prices = scaler.inverse_transform(data[:, 0].reshape(-1,
        1))

    # Calculate daily returns
    daily_returns = np.diff(original_prices, axis=0) /
        original_prices[:-1]

    # Calculate the standard deviation of daily returns (volatility)
    volatility = np.std(daily_returns)
    print(f"Historical volatility: {volatility:.4f}")

    # Initialize predictions list to store results
    predictions = []

    # First, predict the actual last price to calibrate the model
    lstm_input = np.array(temp_input[-time_step:]).reshape(1,
        time_step, n_features)
```

SFIT Library 2024-25

```
    lstm_pred = lstm_model.predict(lstm_input, verbose=0)[0][0]

    xgb_input = np.array(temp_input[-time_step:]).reshape(1, -1)

    try:
        if xgb_model is not None:
            xgb_pred = xgb_model.predict(xgb_input)[0]
            combined_pred = lstm_pred + xgb_pred
        else:
            combined_pred = lstm_pred
    except Exception as e:
        print(f"Error in XGBoost prediction: {str(e)}")
        combined_pred = lstm_pred

    # Get the model's prediction for the current price
    model_predicted_current = scaler.inverse_transform(np.array([[
        combined_pred]]))[0][0]

    # Calculate correction factor to adjust predictions to actual
        price
    correction_factor = last_actual_close / model_predicted_current
        if model_predicted_current > 0 else 1.0
    print(f"Model calibration: Model predicted ${
        model_predicted_current:.2f} for last known price")
    print(f"Correction factor: {correction_factor:.4f}")

    # Store the previous day's prediction for flux calculation
    prev_day_pred = combined_pred

    # Generate predictions for future days
    for day in range(days_ahead):
        # Check for stop request
        if stop_requested_callback and stop_requested_callback():
            print(f"Prediction stopped at day {day}/{days_ahead}")
            break

        # Prepare input for LSTM (all features)
        lstm_input = np.array(temp_input[-time_step:]).reshape(1,
            time_step, n_features)

        # Get LSTM prediction (for Close price only)
        lstm_pred = lstm_model.predict(lstm_input, verbose=0)[0][0]

        # Prepare input for XGBoost
        xgb_input = np.array(temp_input[-time_step:]).reshape(1, -1)

        # Get XGBoost prediction (adjustment to LSTM prediction)
        try:
```

SFIT Library 2024-25

```
74              if xgb_model is not None:
75                  xgb_pred = xgb_model.predict(xgb_input)[0]
76                  # Combine predictions
77                  combined_pred = lstm_pred + xgb_pred
78              else:
79                  # If XGBoost was stopped, just use LSTM prediction
80                  combined_pred = lstm_pred
81          except Exception as e:
82              print(f"Error in XGBoost prediction: {str(e)}")
83              combined_pred = lstm_pred  # Fallback to LSTM if XGBoost
                    fails
84
85          # Add some natural variation to the prediction based on
                historical volatility
86          # This prevents straight-line predictions by introducing
                realistic flux
87
88          # Base flux on historical volatility and a random factor
89          # Get unscaled values for better flux calculation
90          prev_unscaled = scaler.inverse_transform(np.array([[
                prev_day_pred]]))[0][0]
91          current_unscaled = scaler.inverse_transform(np.array([[
                combined_pred]]))[0][0]
92
93          # Calculate the trend direction and magnitude
94          price_change = current_unscaled - prev_unscaled
95          trend_direction = 1 if price_change >= 0 else -1
96
97          # Generate realistic flux - more volatility for later days (
                compound effect)
98          day_volatility = volatility * (1 + day * 0.1)  # Reduced
                volatility scaling for later days
99
100         # Apply variation - adjust volatility to a realistic range
101         adjusted_volatility = min(day_volatility, 0.015)  # Reduced
                maximum volatility cap to 1.5%
102
103         # Add some random variation that follows historical patterns
                but with direction bias
104         random_factor = np.random.normal(0, adjusted_volatility)
105
106         # Bias the random factor based on trend direction
107         # For upward trends (good): be more conservative with upward
                flux and more generous with downward flux
108         # For downward trends (bad): be more pessimistic with
                downward flux and more conservative with upward flux
109         if trend_direction > 0:  # Upward trend
110             if np.random.random() < 0.7:  # 70% chance of following
```

SFIT Library 2024-25

```
                trend
                # Follow upward trend but conservatively
                flux_factor = abs(random_factor) * trend_direction *
                    0.15  # Reduced upward flux
            else:
                # Stronger counter-trend for upward trends
                flux_factor = -abs(random_factor) * trend_direction *
                    0.3  # Stronger downward flux in uptrend
        else:  # Downward trend
            if np.random.random() < 0.8:  # 80% chance of following
                trend
                # Follow downward trend more strongly
                flux_factor = abs(random_factor) * trend_direction *
                    0.25  # Stronger downward flux
            else:
                # Weaker counter-trend for downward trends
                flux_factor = -abs(random_factor) * trend_direction *
                    0.1  # Weaker upward flux in downtrend

        # Calculate flux amount
        flux_amount = prev_unscaled * flux_factor

        # Apply flux in original scale
        adjusted_unscaled = current_unscaled + flux_amount

        # Convert back to scaled value
        adjusted_pred = scaler.transform(np.array([[adjusted_unscaled
            ]]))[0][0]

        # Create the next day's features with the adjusted prediction
        next_row = temp_input[-1].copy()
        next_row[0] = adjusted_pred  # Update Close price

        # Store for next day's calculation
        prev_day_pred = adjusted_pred

        # Add the adjusted prediction
        predictions.append(adjusted_pred)
        temp_input.append(next_row)

    # If we have no predictions (all steps were stopped), return None
    if not predictions:
        return None

    # Convert predictions back to original scale
    final_predictions = scaler.inverse_transform(np.array(predictions
        ).reshape(-1, 1))

```

SFIT Library 2024-25

```
151     # Apply correction factor to align with actual last closing price
152     corrected_predictions = final_predictions * correction_factor
153
154     print("\nOriginal vs. Corrected Predictions:")
155     for i in range(len(final_predictions)):
156         print(f"Day {i+1}: Original ${final_predictions[i][0]:.2f} ->
            Corrected ${corrected_predictions[i][0]:.2f}")
157
158     return corrected_predictions
```

SFIT Library 2024-25

Chapter 7

Testing

7.1 Test Cases

Test cases were designed to validate each module of the system, including data input, preprocessing, model predictions, and UI behavior. Scenarios such as valid or invalid stock symbol entry, missing data handling, and prediction accuracy thresholds were thoroughly tested.

7.2 Testing Methods Used

To ensure the StockBuddy system functioned accurately, efficiently, and as intended, a combination of software testing methodologies was applied during development. These methods helped identify bugs early, maintain code quality, and confirm that all integrated components worked together seamlessly.

7.2.1 Unit Testing

Unit testing focused on validating individual components of the system in isolation. Key modules such as:

- Data Fetching Layer (handling APIs for stock market data),
- Data Preprocessing Module (scaling, encoding, and formatting for model input),
- XGBoost and LSTM Models (model loading, inference accuracy),
- Utility Functions (error handling, date parsing, etc.)

Were each tested with predefined inputs and expected outputs. This helped in identifying and resolving errors at an early stage, ensuring that each block of code functioned reliably on its own. Python's unittest and pytest frameworks were utilized for writing and automating these tests.

7.2.2 Integration Testing

While unit testing ensured that each module worked individually, integration testing validated that modules interacted correctly as a whole. This was especially critical in StockBuddy, where the workflow involved:

SFIT Library 2024-25

- Triggering predictions from the frontend,
- Fetching real-time or historical data,
- Passing it through preprocessing,
- Feeding it into trained XGBoost and LSTM models,
- Displaying the results accurately in the user interface.

Integration testing was key to simulating real-world user interactions and making sure that the data pipeline, prediction logic, and UI elements were synchronized properly. Any mismatches in data formats, API communication errors, or UI display bugs were identified and resolved during this phase.

SFIT Library 2024-25

Chapter 8

Results and Discussion

The different model versions—Research Paper, Original(Version 1), and Optimized(Current Version)—represent various stages of model refinement for predicting AAPL and TSLA stock prices. The Original model had lower accuracy (AAPL: 67.51%, TSLA: 63.08%) and higher RMSE, indicating less precise predictions. The Research Paper model improved performance significantly by increasing accuracy (AAPL: 68.44%, TSLA: 71.32%) and reducing RMSE slightly. However, it uses more data and longer processing time. The Optimized model further enhanced accuracy (AAPL: 72.04%, TSLA: 74.02%) while achieving the lowest RMSE, demonstrating better prediction quality. Additionally, the optimized model dramatically reduced computation time, making it the most efficient among the three.

	Stock	Accuracy(%)	RMSE	Data Size (Rows, Features)	Samples	Time Taken
Research Paper[5]	AAPL	68.44	9.2	(2516, 11)	2456	8min 45sec
Version 1	AAPL	67.51	8.57	(5031, 11)	5031	5min 30sec
Current Version	AAPL	72.04	7.43	(81, 11)	28	1min 32sec

Figure 8.1: Models comparison table

The Research Paper model improved performance using a dataset of 2516 rows and 11 features, with 2456 samples for prediction. Accuracy increased (AAPL: 68.44%, TSLA: 71.32%), though RMSE remained similar (AAPL: 9.20, TSLA: 27.83). However, it required significantly longer computation time—8 minutes and 45 seconds for AAPL and 8 minutes and 34 seconds for TSLA.

Stock	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10
AAPL	223.49	223.54	224.3	223.91	224.57	225.4	226.3	227.31	227.31	229.47
TSLA	273.03	272.33	273.53	273.61	273.91	274.33	274.31	274.28	275.04	275.73

Figure 8.2: Predicted prices from research paper model

The Original model(Version 1) used a dataset of 5031 rows and 11 features, with 28 samples for prediction. It had the lowest accuracy (AAPL: 67.51%, TSLA: 63.08%) and higher RMSE

SFIT Library 2024-25

(AAPL: 8.57, TSLA: 27.92). The model took 5 minutes and 30 seconds for AAPL and 5 minutes and 07 seconds for TSLA to compute.

Stock	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10
AAPL	222.13	223.03	223.31	223.71	223.5	225.04	225.67	226.46	227.31	227.61
TSLA	272.77	272.97	273.05	273.3	273.9	273.42	274.46	274.24	275.42	275.59

Figure 8.3: Predicted prices from Version1 model

The Optimized model(Current version) drastically improved efficiency by reducing the dataset size to 81 rows and 11 features, maintaining 28 samples. It achieved the highest accuracy (AAPL: 72.04%, TSLA: 74.02%) and the lowest RMSE (AAPL: 7.43, TSLA: 26.79). It also significantly reduced processing time to 1 minute and 32 seconds for AAPL and 1 minute and 04 seconds for TSLA, making it the most effective version.

Stock	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10
AAPL	223.49	223.15	224.24	225.33	223.08	224.45	225.21	225.08	223.57	228.47
TSLA	273.53	274.46	274.51	275.06	274.68	274.3	272.57	273.81	273.04	272.58

Figure 8.4: Predicted prices from current version model

SFIT Library 2024-25

Chapter 9

Conclusion and Future Scope

9.1 Conclusion

Stock prediction systems represent a significant advancement in the field of financial forecasting, combining historical data analysis, machine learning algorithms, and sentiment analysis to deliver more accurate and reliable predictions. By leveraging these technologies, investors can make more informed decisions, mitigate risks, and optimize their investment strategies in an increasingly complex and volatile market environment. The integration of real-time data and automated analysis reduces human error and bias, providing a more objective basis for decision-making.

While no system can guarantee absolute accuracy due to the inherently unpredictable nature of financial markets, these advanced tools enhance the ability to anticipate market trends and fluctuations. As technology continues to evolve, further improvements in data processing, model accuracy, and real-time analytics will likely make stock prediction systems even more integral to modern investment practices. Ultimately, these systems offer valuable insights that support strategic financial planning and contribute to more robust risk management in both individual and institutional investment contexts.

9.2 Future Scope

For future study, the integration of real-time stock prices and financial news using live data feeds can enhance the system's ability to make accurate, dynamic predictions in response to market fluctuations. Another possible direction for future study is to improve the model's architecture by implementing faster and more efficient logic, such as transformer-based or hybrid deep learning models. Additionally, conducting a deeper financial sentiment analysis using domain-specific models and incorporating diverse sources like news headlines and social media content may lead to more reliable and context-aware stock market predictions.

SIT Library 2024-25

Chapter 10

Acknowledgment

We would like to thank our guide, Prof. Balaraju Vijayalakshmi, for her continuous guidance and support throughout this project. We also appreciate the Department of Information Technology at St. Francis Institute of Technology for providing the resources to carry out our work. This project was a team effort—Daniel and Royden worked on the model and app development, while Aryan and Darish contributed to testing and documentation.

SFIT Library 2024-25

Chapter 11

Literature Cited

[1] Y. Liu, et al., "Digger-Guider: High-Frequency Factor Extraction for Stock Trend Prediction," *IEEE Transactions on Knowledge and Data Engineering*, 2024, DOI: 10.1109/TKDE.2024.3424475.

[2] B. Panwar, G. Dhuriya, and P. Joshi, "Stock Market Prediction Using Linear Regression and SVM," in *2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*, 2021, DOI: 10.1109/ICACITE51222.2021.9404733.

[3] S. Mutalib and L. Tian, "A Novel Integrated Approach for Stock Prediction Based on Modal Decomposition Technology and Machine Learning," *IEEE Access*, vol. 12, pp. 123456-123467, 2024, DOI: 10.1109/ACCESS.2024.3425727.

[4] W. Khan, M. A. Ghazanfar, M. A. Azam, A. Karami, K. H. Alyoubi, and A. S. Alfakeeh, "Stock Market Prediction Using Machine Learning Classifiers and Social Media, News," *Springer-Verlag*, part of Springer Nature, 2020.

[5] S. Singh, S. Gutta, and A. Hadaegh, "Stock Prediction Using Machine Learning," *WSEAS Transactions on Systems*, vol. 9, 2021, DOI: 10.37394/232018.

[6] S. Mohan, S. Mullapudi, S. Sammeta, P. Vijayvergia, and D. C. Anastasiu, "Stock Price Prediction Using News Sentiment Analysis," *2019 IEEE Fifth International Conference on Big Data Computing Service and Applications (BigDataService)*, Newark, CA, USA, 2019.

[7] R. Batra and S. M. Daudpota, "Integrating StockTwits with Sentiment Analysis for Better Prediction of Stock Price Movement," *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, Sukkur, Pakistan, 2018.

[8] D. G. Takale, "Enhancing financial sentiment analysis: A deep dive into natural language processing for market prediction," *J. Computer Networks and Virtualization*, vol. 2, no. 2, pp. 1-10, Jul.-Dec. 2024, doi: 10.48001/JoCNV.

[9] Z. Dong, X. Fan, and Z. Peng, "FNSPID: A comprehensive financial news dataset in time series," *Proc. ACM Conf. (Conference'17)*, Jul. 2017, pp. 1-11, 2024, doi: 10.1145.

[10] K. Raut, P. Kasture, C. Gosavi, and T. Deshpande, "Stock market prediction using Alpha Vantage API and machine learning algorithm," *Int. Res. J. Eng. Technol. (IRJET)*, vol. 9, no. 5, pp. 451-456, May 2022.

SFIT Library 2024-25

[11] S. Hossain and G. Kaur, ”Stock Market Prediction: XGBoost and LSTM Comparative Analysis,” *2024 3rd International Conference on Artificial Intelligence For Internet of Things (AIIoT)*, Vellore, India, 2024, pp. 1-6, doi: 10.1109/AIIoT58432.2024.10574794.

[12] J. Soni and K. Mathur, ”Sentiment Analysis of News Headlines for Stock Market Prediction using VADER,” *2023 3rd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, Bengaluru, India, 2023, pp. 1215-1222, doi: 10.1109/ICIMIA60377.2023.10426095.

SFIT Library 2024-25