### Low Level Design Document

**Stock Price Prediction System**

**Document Version:** 1.0  
**Last Revised Date:** 2024-05-22  
**Written By:** Akshay Garude

#### DOCUMENT CONTROL

| **Date Issued** | **Version** | **Description** | **Author** |
| --- | --- | --- | --- |
| 2024-05-22 | 1.0 | Initial Version | Akshay Garude |

### Contents

1. Introduction
   * 1.1 Purpose
   * 1.2 Scope
2. System Architecture
3. Code Description
   * 3.1 Data Collection and Preprocessing
   * 3.2 Model Building
   * 3.3 Model Training
   * 3.4 Model Evaluation
4. Deployment
5. Conclusion

### 1. Introduction

#### 1.1 Purpose

The purpose of this document is to provide a detailed overview of the code implementation for a Stock Price Prediction System. It covers the data processing steps, model building, training, and deployment aspects.

#### 1.2 Scope

The scope of this document includes the Python code segments for fetching historical stock data, preprocessing the data, building an LSTM model for stock price prediction, training the model, evaluating model performance, and deploying the model for real-time predictions.

### 2. System Architecture

The system architecture includes the following components:

* Data Collection: Fetching historical stock data using the Yahoo Finance API.
* Data Preprocessing: Preprocessing the data by scaling and splitting into training and testing sets.
* Model Building: Creating an LSTM model architecture for stock price prediction.
* Model Training: Training the LSTM model using historical stock data.
* Model Evaluation: Evaluating the model's performance using test data.
* Deployment: Deploying the trained model for real-time stock price predictions.

### 3. Code Description

#### 3.1 Data Collection and Preprocessing

# Importing required libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import yfinance as yf

from sklearn.preprocessing import MinMaxScaler

# Fetching historical stock data

start = '2010-01-01'

end = '2019-12-31'

df = yf.download('AAPL', start=start, end=end)

# Data preprocessing

scaler = MinMaxScaler(feature\_range=(0, 1))

data\_training = scaler.fit\_transform(df['Close'][0:int(len(df) \* 0.70)])

data\_testing = scaler.transform(df['Close'][int(len(df) \* 0.70):int(len(df))])

#### 3.2 Model Building

from keras.models import Sequential

from keras.layers import Dense, LSTM, Dropout

# Defining the LSTM model architecture

model = Sequential()

model.add(LSTM(units=50, activation='relu', return\_sequences=True, input\_shape=(x\_train.shape[1], 1)))

model.add(Dropout(0.2))

# Add more LSTM layers and dropout as needed

#### 3.3 Model Training

# Compile and train the LSTM model

model.compile(optimizer='adam', loss='mean\_squared\_error')

model.fit(x\_train, y\_train, epochs=50)

#### 3.4 Model Evaluation

# Making predictions using the trained model

y\_predicted = model.predict(x\_test)

# Plotting actual vs predicted prices

plt.figure(figsize=(12, 6))

plt.plot(y\_test, 'b', label='Original Price')

plt.plot(y\_predicted, 'r', label='Predicted Price')

plt.xlabel('Time')

plt.ylabel('Price')

plt.legend()

plt.show()

### 4. Deployment

The trained model can be saved and deployed using frameworks like Flask or integrated into a Streamlit web application for real-time stock price predictions.

A screenshot of a computer screen

Description automatically generated

### 5. Conclusion

This low-level design document provides insights into the code implementation of a Stock Price Prediction System using LSTM neural networks. It covers data preprocessing, model building, training, evaluation, and deployment aspects, enabling developers to understand and replicate the system.