**Introduction**This report outlines the development of a stock price prediction application using a Long Short-Term Memory (LSTM) model. The application fetches historical stock data from Yahoo Finance, processes the data, and predicts future stock prices based on user input. The primary goal of this task was to create an accessible user interface for predicting stock prices, demonstrating the integration of machine learning with web development.**Data Collection**The first step involved fetching historical data for Apple Inc. (AAPL) using the yfinance library. The data was retrieved for the period from January 1, 2020, to January 1, 2024. The closing prices were extracted and reshaped to prepare for normalization. The MinMaxScaler from the sklearn.preprocessing module was utilized to scale the data to a range between 0 and 1, which is crucial for the performance of the LSTM model.

**Code Snippetpython**data = yf.download('AAPL', start='2020-01-01', end='2024-01-01')close\_prices = data['Close'].values.reshape(-1, 1)scaler = MinMaxScaler(feature\_range=(0, 1))scaled\_data = scaler.fit\_transform(close\_prices)

**Data Preparation**To train the LSTM model, the data was prepared by creating sequences of 60 days of historical prices as input features (X) and the corresponding next day's price as the target variable (y). This approach allows the model to learn patterns from the past 60 days to predict the next day's price.

**Code Snippetpython**look\_back = 60X\_train, y\_train = [], []for i in range(look\_back, len(scaled\_data)): X\_train.append(scaled\_data[i-look\_back:i, 0]) y\_train.append(scaled\_data[i, 0])X\_train, y\_train = np.array(X\_train), np.array(y\_train)X\_train = np.reshape(X\_train, (X\_train.shape[0], X\_train.shape[1], 1))

**Model Development**The LSTM model was constructed using the Sequential API from Keras. The model consists of two LSTM layers with 50 units each and a dense layer for output. The model was compiled using the Adam optimizer and mean squared error as the loss function. The model was trained for one epoch with a batch size of one.

**Code Snippetpython**model = Sequential()model.add(LSTM(units=50, return\_sequences=True, input\_shape=(X\_train.shape[1], 1)))model.add(LSTM(units=50))model.add(Dense(1))model.compile(optimizer='adam', loss='mean\_squared\_error')model.fit(X\_train, y\_train, epochs=1, batch\_size=1)

**Prediction and Model Saving**After training, the model was tested using the latest 60 days of scaled data to predict the next price. The predicted price was then inverse-transformed to its original scale. The trained model was saved to a file for future use.

**Code Snippetpython**test\_data = scaled\_data[-look\_back:]test\_data = test\_data.reshape(1, look\_back, 1)predicted\_price = model.predict(test\_data)predicted\_price = scaler.inverse\_transform(predicted\_price)model.save('lstm\_model.h5')

**Web Application Development**A Flask web application was created to provide a user-friendly interface for making predictions. The application includes:A home route that renders the main HTML page.A prediction route that processes user input, fetches the latest stock data, preprocesses it, and returns the prediction.Key FeaturesUser input for stock ticker symbols.Error handling for invalid ticker symbols.Display of prediction results on the webpage.

**Code Snippetpython**@app.route('/predict', methods=['POST'])def predict(): stock\_data = request.form['stock\_data'] ... return render\_template('index.html', prediction=prediction[0][0], error=None)**User Interface**The HTML file created for the application features a clean and responsive design, with styles applied for better usability. The interface includes input validation to ensure the user enters a valid ticker symbol.

**HTML Snippethtml**<form id="stockForm" action="/predict" method="POST"> <label for="ticker">Enter Stock Ticker Symbol:</label> <input type="text" id="ticker" name="stock\_data" placeholder="e.g., AAPL, GOOGL" required> ... <button type="submit">Predict</button></form>

**Conclusion**The stock price prediction task successfully integrated machine learning with web development to create an interactive application. The use of LSTM for time series forecasting proved effective, and the Flask framework allowed for seamless deployment of the model. Future improvements could include enhancing the model's accuracy through hyperparameter tuning and adding more features to the web application, such as historical price charts and analytics.This project serves as a solid foundation for further exploration into financial forecasting and web application development.