High Level Design (HLD) Investment Prediction

Revision Number: 1.1

Last date of revision: 18/05/2024

Akshay Garude

# Document Version Control

|  |  |  |  |
| --- | --- | --- | --- |
| **Date Issued** | **Version** | **Description** | **Author** |
| **13th March 2024** | 1.0 | Abstract, Introduction and general description | Akshay Garude |
| **14th March 2024** | 1.1 | Design detail,and deployment | Akshay Garude |
| **15th March 2024** | 1.2 | Final Version of Complete HLD | Akshay Garude |

**Contents**

[Document Version Control 2](#_bookmark0)

[Abstract 3](#_bookmark1)

1. [Introduction 5](#_bookmark2)
   1. [Why this High-Level Design Document? 5](#_bookmark3)
   2. [Scope 5](#_bookmark4)
2. [General Description 6](#_bookmark5)
   1. [Product Perspective & Problem Statement 6](#_bookmark6)
   2. [Tools used 6](#_bookmark7)
3. [Design Details 7](#_bookmark8)
   1. [Functional Architecture 7](#_bookmark9)
   2. [Optimization 8](#_bookmark10)
4. [KPIs 9](#_bookmark11)
   1. [KPIs (Key Performance Indicators) 9](#_bookmark12)
5. [Deployment 9](#_bookmark13)

# 

# Abstract

This document presents a high-level design for a stock trend prediction and visualization system using LSTM neural networks and Streamlit for a GUI. It includes the architectural framework, functional components, key performance indicators (KPIs), and deployment strategies.

# Introduction

## Why this High-Level Design Document?

The purpose of this High-Level Design (HLD) document is to provide a detailed framework for the development of a stock trend prediction system. This document aims to ensure all design aspects are considered before implementation, detect any potential contradictions, and serve as a reference for how the system's modules interact at a high level.The HLD will:

The HLD will:

* Present detailed design aspects.
* Describe the user interface.
* Define hardware and software interfaces.
* Outline performance requirements.
* Include design features and system architecture.
* List and describe non-functional attributes such as security, reliability, maintainability, portability, reusability, application compatibility, resource utilization, and serviceability.

## 

## 1.2 Scope

The HLD documentation presents the structure of the system, such as the database architecture, application architecture (layers), application flow (Navigation), and technology architecture. The HLD uses non-technical to mildly-technical terms which should be understandable to the administrators of the system.

# General Description

## Product Perspective & Problem Statement

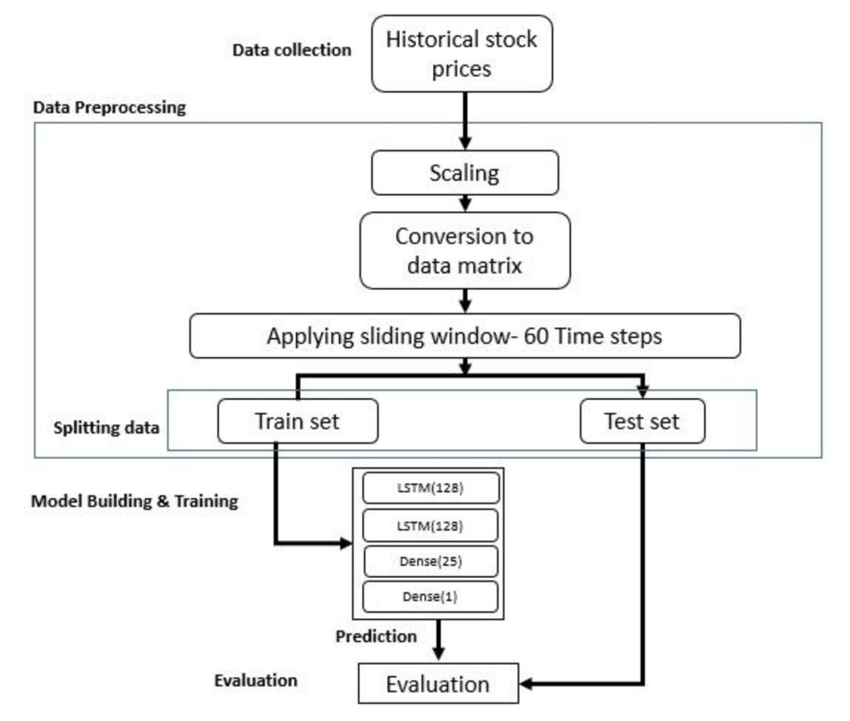
The product aims to predict stock trends and visualize historical and real-time data. The primary goal is to develop an LSTM-based predictive model to forecast stock prices and a Streamlit-based GUI for visualization. This system will help investors and analysts make informed decisions by providing accurate predictions and clear visual representations of stock data.

## Tools used

* Programming Languages and Libraries: Python, Numpy, Pandas, Matplotlib, Scikit-learn, TensorFlow, Keras
* Data Source: Yahoo Finance (yfinance)
* Visualization: Streamlit, Matplotlib
* Hardware: GPU (for training the LSTM model)
* Deployment: Streamlit Cloud, Docker (optional for deployment)

# Design Details

## Functional Architecture





The system comprises two main components: the LSTM model for stock prediction and the Streamlit GUI for data visualization.

**LSTM Model:**

* Data Collection: Fetch historical stock data using yfinance.
* Data Preprocessing: Normalize data, create training/testing datasets.
* Model Training: Build and train the LSTM model using the training dataset.
* Prediction: Use the trained model to make predictions on the test dataset.
* Evaluation: Compare predictions with actual data to evaluate performance.

**Streamlit GUI:**

* User Input: Allow users to input stock ticker symbols.
* Data Fetching: Fetch real-time and historical data for the specified stock.
* Visualization: Display stock prices, moving averages, and prediction results.
* Real-Time Updates: Show current stock price and update visualizations.

**Functional Architecture:**

**A diagram of a process

Description automatically generated**

3 .Optimization

**Data Strategy:** Minimize the number of fields and records. Optimize data extracts by materializing calculations, removing unnecessary columns, and using accelerated views.

**Visualization Optimization:** Use guided analytics to minimize the number of data points in views. Limit the number of filters and prefer include filters over exclude filters. Use continuous date filters and parameters to reduce query load.

# KPIs

## KPIs (Key Performance Indicators)

**Model Accuracy:** Measure the accuracy of the LSTM model's predictions compared actual stock prices.

**Prediction Time:** Evaluate the time taken to make predictions.

**User Engagement:** Track the number of users interacting with the Streamlit GUI.

**System Reliability:** Monitor system uptime and the frequency of failures or errors.

**Data Freshness:** Ensure real-time data is updated accurately and timely.

# Deployment

The system will be deployed using Streamlit, providing flexibility to fit various enterprise architectures. Options include on-premises, cloud, and hosted solutions, allowing seamless integration with existing technology infrastructure.

* **Deployment Steps:**
  1. **Model Training:** Train the LSTM model on a local or cloud-based environment with GPU support.
  2. **Model Deployment:** Save the trained model and load it into the Streamlit application.
  3. **Streamlit Application:** Deploy the Streamlit application to Streamlit Cloud or a Docker container for ease of access and scalability.
  4. **Monitoring and Maintenance:** Implement logging and monitoring to ensure the application runs smoothly and update the model periodically with new data.
* **Example Insights Displayed:**
  1. Historical and real-time stock prices.
  2. Moving averages and trend lines.
  3. Prediction results compared to actual prices.

The dashboard will provide actionable insights and visualizations