**Title: MLOps based Stock Prediction**

**Objective:**

To build an End-to-End Automated Stock prediction with MLOPs principles. The model will be implemented on a production line with abilities of retraining and permanent monitoring of performance.

**Key Requirements:**

**MLOps deployment & Training:**

It is supposed to make an automated pipeline that ingests the data, preprocesses it, trains and validates on it, deploys the model, and monitors.

**Upcoming Stock Forecast Model:**

Leverage high-quality machine-learning to understand past trends and predict future trends of the stock.

Production-Based Automated Enhancing:

Add close-loop feedback in which the model will retrain periodically with new production information. This will make it flexible with regard to evolving market trends.

**Tracking and Monitoring: The Metric:**

Have tools on key performance indicators like performance accuracy, RMSE, latency, data drift and model degradation.

Ensure the transparency and reproducibility of models with the help of experiment tracking software.

**Solution**

This project aims to predict stock prices using machine learning and automate the process using MLOps. The model will be built, trained, deployed, and monitored using tools like Azure, GitHub, and Docker. It will also improve automatically by learning from new data.

**Technologies Used:**

* **Python, Pandas, Scikit-learn** – For data processing and model creation
* **Azure Machine Learning** – For model deployment and automation
* **GitHub + GitHub Actions** – For version control
* **Docker** – For packaging the model into containers
* **ML flow**– For tracking experiments and results
* **Azure Monitor** – For tracking performance and alerts

**Learning Tasks:**

* Learn how to collect stock data from APIs
* Understand how to train models using time-series data
* Explore how MLOps works step by step
* Study how to deploy models and track them

**Implementation Tasks:**

* We have to get the stock data and clean it
* Then Build a simple machine learning model
* Log the results and performance metrics
* Create a Docker container for the model
* Push the model to Azure for deployment
* Set automatic retraining using new data
* Set up monitoring to track accuracy of the model