PROJECT TITLE

STOCK ANALYSIS SYSTEM

(BY SEMESTER – 7 OF IV YEAR M.SC. (CA & IT) 2024-25)

SUBMITTED BY:

STUDENT NAME	ROLL NO
PRATIK BANSILAL RANA	4242
DEV BHARATKUMAR VALAND	4271
LAV DIPAKBHAI RATHOD	4281

GROUP ID: - **20**

Date of Submission: **22/12/2024**

SUBMITTED TO:-

K. S. SCHOOL OF BUSINESS MANAGEMENT AND INFORMATION TECHNOLOGY

M.Sc.- Computer Application and Technology

TABLE OF CONTENTS

1. Introduction

❖ Objective of the Project

- The main objective of this project is to analyze historical stock data, calculate useful indicators such as moving averages, visualize trends, and predict future stock prices using linear regression.

2. Problem Statement

❖ Simple Description of the Problem

- Investors often face challenges in understanding historical trends in stock prices and predicting future movements. This project addresses these issues by:

- Fetching historical stock data for a specific company.
- Visualizing stock price trends and moving averages.
- Applying linear regression to predict future stock prices.

3. Basic Concepts

❖ Key AI Terminologies

- **Linear Regression**: A machine learning model that predicts the value of a dependent variable (e.g., stock price) based on an independent variable (e.g., time).
- **A moving average (MA):** is a technical indicator that traders use to smooth out price data and identify trends in a stock's price over time.

❖ Introduction to AI Techniques Used in the Project

1) Linear Regression:

- Predicts stock prices based on historical data.
- Evaluates performance using **Mean Squared Error (MSE)**.

2) Feature Engineering:

- Calculates 20-day, 50-day, and 200-day Moving Averages to identify trends.

3) Data Preprocessing:

- Converts dates into numerical values for regression.
- Handles missing values by dropping incomplete rows.

4) Data Visualization:

- Uses candlestick charts, line plots, and scatter plots to represent trends and predictions.

5) Model Evaluation:

- MSE quantifies the accuracy of stock price predictions.

4. Requirement Analysis

❖ Tools and Technologies Needed

- Python Libraries:
  - **`yfinance`** for stock data retrieval
  - **`pandas`** for data manipulation
  - **`numpy`** for numerical operations
  - **`matplotlib`** and **`mplfinance`** for visualization
  - **`sklearn`** for machine learning (linear regression, metrics, etc.)

❖ Basic System Requirements

- **Python 3.7** or higher
- **IDE** or code editor (e.g., **VSCode**, **Jupyter Notebook**)
- **Stable internet** connection for **fetching stock data**

5. Dataset

❖ Source of Data

- Data is fetched directly from **Yahoo Finance** using the **`yfinance`** library.

❖ Overview of the Data

- The data includes the following fields:
  - **Open**: Opening price of the stock
  - **High**: Highest price during the day
  - **Low**: Lowest price during the day
  - **Close**: Closing price of the stock
  - **Volume**: Number of shares traded

❖ Basic Data Cleaning

- Handling missing values by dropping them.
- Adding additional columns such as moving averages.

6. Proposed Solution

❖ Simple Explanation of the Approach

1. Fetch **historical stock** data for a given **ticker symbol**.
2. Calculate **20-day, 50-day, and 200-day moving averages** for trend analysis.
3. Use **linear regression** to **predict future stock prices** based on historical data.
4. **Visualize results** using various **graphs** to **highlight trends** and **predictions**.

❖ Algorithm or Model Chosen

- **Linear Regression** was chosen for its simplicity and effectiveness.

- **Type**: Supervised Machine Learning Algorithm.
- **Purpose**: Models the relationship between the stock's Date (independent variable) and Close price (dependent variable) to predict future stock prices.
- **Implementation**: Utilized from the **scikit-learn** library.

7. Implementation

❖ Step-by-Step Process

1. **Fetch Stock Data**:

- Use `yfinance` to retrieve historical data for the selected stock.

2. **Data Processing**:

- Add columns for moving averages (20-day, 50-day, 200-day).

3. **Data Preparation**:

- Convert dates to numerical values for regression. Split data into training and testing sets.

4. **Train Model**:

- Fit a linear regression model using training data.

5. **Predict Future Prices**:

- Use the model to predict stock prices for unseen data.

6. **Visualize Results**:

- Generate plots to showcase stock trends, moving averages, and predictions.

❖ Screenshots or Code Snippets

```
import yfinance as yf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import mplfinance as mpf
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

# Fetching stock data
def fetch_stock_data(ticker, period='5y'):
    stock = yf.Ticker(ticker)
    data = stock.history(period=period)
    return data

# Display historical data
def display_data(data):
    print(data.head())
    print(data.describe())

# Calculate moving averages
def add_moving_averages(data):
    data['MA20'] = data['Close'].rolling(window=20).mean()
    data['MA50'] = data['Close'].rolling(window=50).mean()
    data['MA200'] = data['Close'].rolling(window=200).mean()

# Visualize the data with multiple types of graphs
def plot_data(data, ticker):
    # plt.figure(figsize=(18, 10))

    # Plot 1: Open Price
    plt.subplot(2, 2, 1)
    plt.plot(data['Open'], label="Open Price", color="blue")
    # plt.plot(data['Close'], label="Close Price", color="green")
    # plt.plot(data['MA20'], label="20-Day MA", color="red")
    # plt.plot(data['MA50'], label="50-Day MA", color="green")
    # plt.plot(data['MA200'], label="200-Day MA", color="purple")
    plt.xlabel("Years")
    plt.ylabel("Open Prices")
    plt.title(f"{ticker} - Historical Open Price")
    plt.subplots_adjust(wspace=0.4, hspace=0.4)  # Adjust space between plots
    plt.grid()
    plt.legend()

    # Plot 2: Close Price Only
    plt.subplot(2, 2, 2)
    plt.plot(data['Close'], label="Close Price", color="green")
    plt.xlabel("Years")
    plt.ylabel("Close Prices")
```

```
    plt.title(f"{ticker} - Historical Close Price")
    plt.subplots_adjust(wspace=0.4, hspace=0.4)  # Adjust space between plots
    plt.grid()
    plt.legend()

    # Plot 3: Moving Averages Only
    plt.subplot(2, 2, 3)
    plt.plot(data['MA20'], label="20-Day MA", color="red")
    plt.plot(data['MA50'], label="50-Day MA", color="green")
    plt.plot(data['MA200'], label="200-Day MA", color="purple")
    plt.xlabel("Years")
    plt.ylabel("Moving Averages")
    plt.title(f"{ticker} - Moving Averages")
    plt.subplots_adjust(wspace=0.4, hspace=0.4)  # Adjust space between plots
    plt.grid()
    plt.legend()

    # Plot 4: Predictions vs Actual Prices (for Linear Regression)
    plt.subplot(2, 2, 4)
    plt.scatter(X_test, y_test, color='blue', label="Actual Price",s=3)
    plt.plot(X_test, y_pred, color='red', label="Predicted Price")
    plt.xlabel(r"Days since start")
    plt.ylabel("Close Price")
    plt.title("Stock Price Prediction using Linear Regression")
    plt.subplots_adjust(wspace=0.4, hspace=0.4)  # Adjust space between plots
    plt.grid()
    plt.legend(loc="upper left")

    # Plot 5: Candlestick + Volume Chart
    plt.plot(1,1)
    mpf.plot(data[-60:], type='candle', style='charles', volume=True,
title=f"{ticker} - Last 60 Days Candlestick + Volume Chart")

    # plt.subplots_adjust(wspace=0.4, hspace=0.5)  # Adjust space between plots
    plt.show()

# Prepare data for linear regressioncls
def prepare_data(data):
    data = data[['Close']].dropna()
    data['Years'] = (data.index - data.index.min()).days  # Converting Yearss to
numerical values
    X = data['Years'].values.reshape(-1, 1)
    y = data['Close'].values
    return X, y

# Train the linear regression model and predict
def predict_stock_price(X, y):
    global X_test, y_test, y_pred  # For use in the plot_data function
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=0)
    model = LinearRegression()
```

```
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    # Evaluate model performance
    mse = mean_squared_error(y_test, y_pred)
    print("Mean Squared Error:", mse)

    return model

# Predict future price
def predict_future_price(model, days_ahead):
    future_day = np.array([[days_ahead]])
    future_price = model.predict(future_day)
    return future_price[0]

# Main function
def main():
    ticker = input("Enter the stock ticker symbol (e.g., AAPL,TSLA,ADANIENT.NS and
RELIANCE.NS for Apple,Tesla,AdaniEnt. and Reliance): ").upper()
    data = fetch_stock_data(ticker)

    if data.empty:
        print("No data found for the given ticker symbol.")
        return

    display_data(data)
    add_moving_averages(data)
    global X, y  # For use in the plot_data function
    X, y = prepare_data(data)
    model = predict_stock_price(X, y)

    plot_data(data, ticker)

    # Predict price 30 days after the latest data
    latest_day = (data.index[-1] - data.index[0]).days
    predicted_price = predict_future_price(model, latest_day + 30)
    print(f"Predicted price for {ticker} 30 days ahead: $/₹ {predicted_price:.2f}")

# Run the main function
main()
```

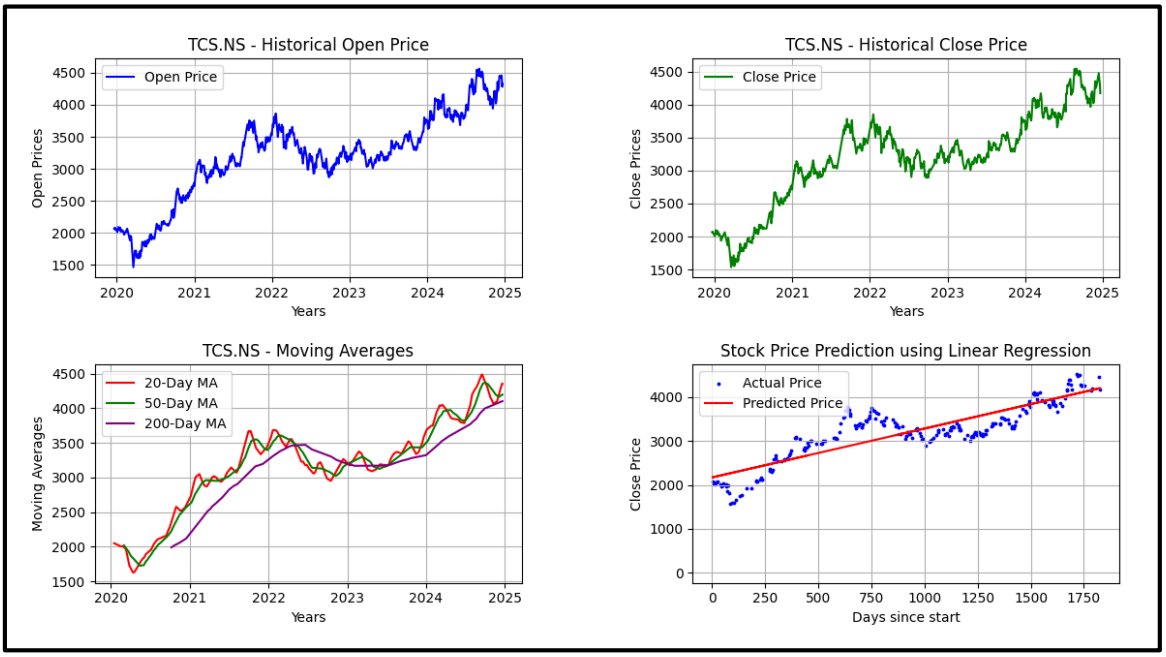
❖ Key Function Example

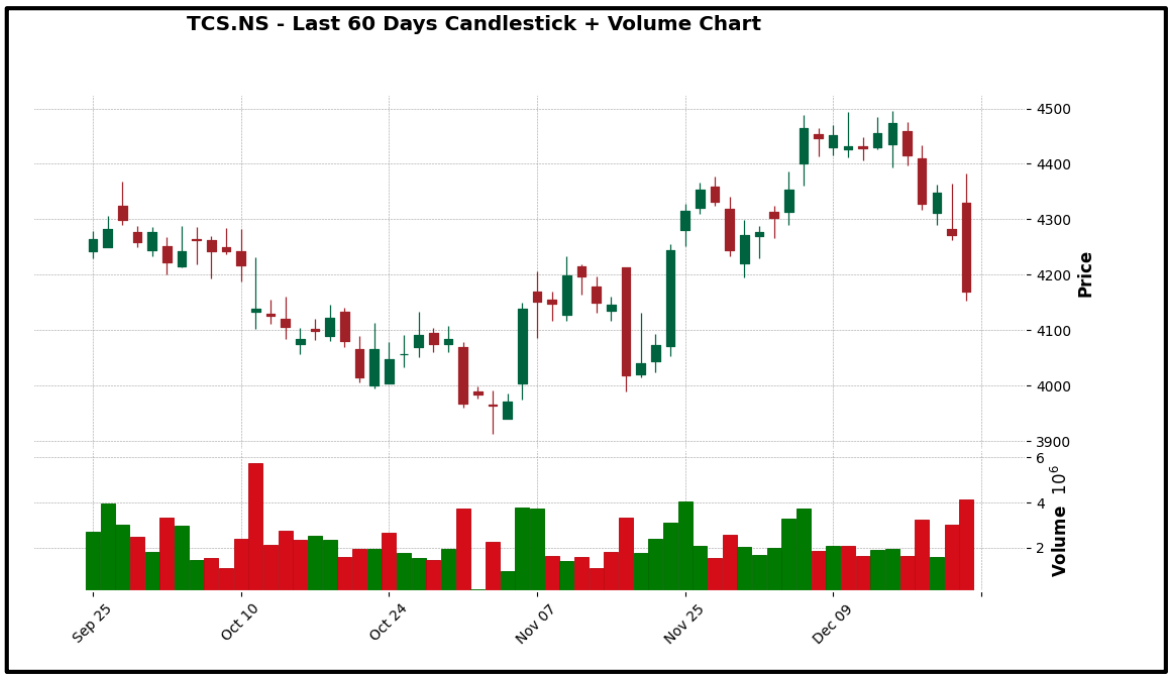
```
def add_moving_averages(data):
    data['MA20'] = data['Close'].rolling(window=20).mean()
    data['MA50'] = data['Close'].rolling(window=50).mean()
    data['MA200'] = data['Close'].rolling(window=200).mean()
```

8. Testing

❖ Simple Test Cases

- Input: **`TCS.NS`** (TCS stock ticker) – Indian stock
  - Output: Graphs showing trends and a predicted price 30 days ahead.

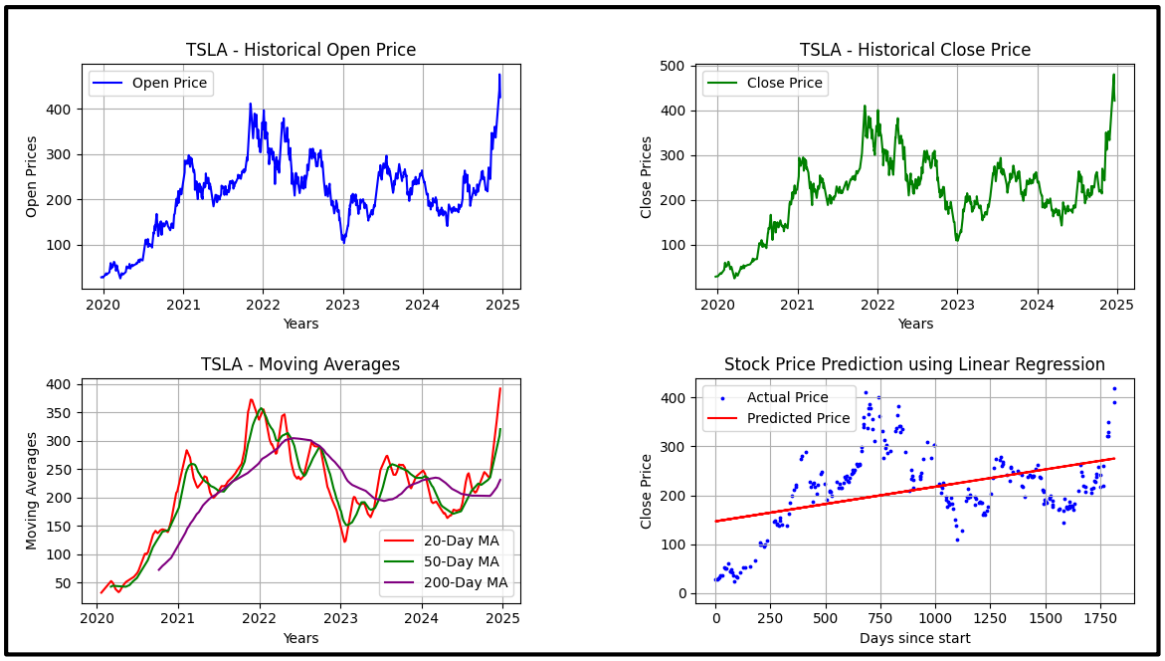


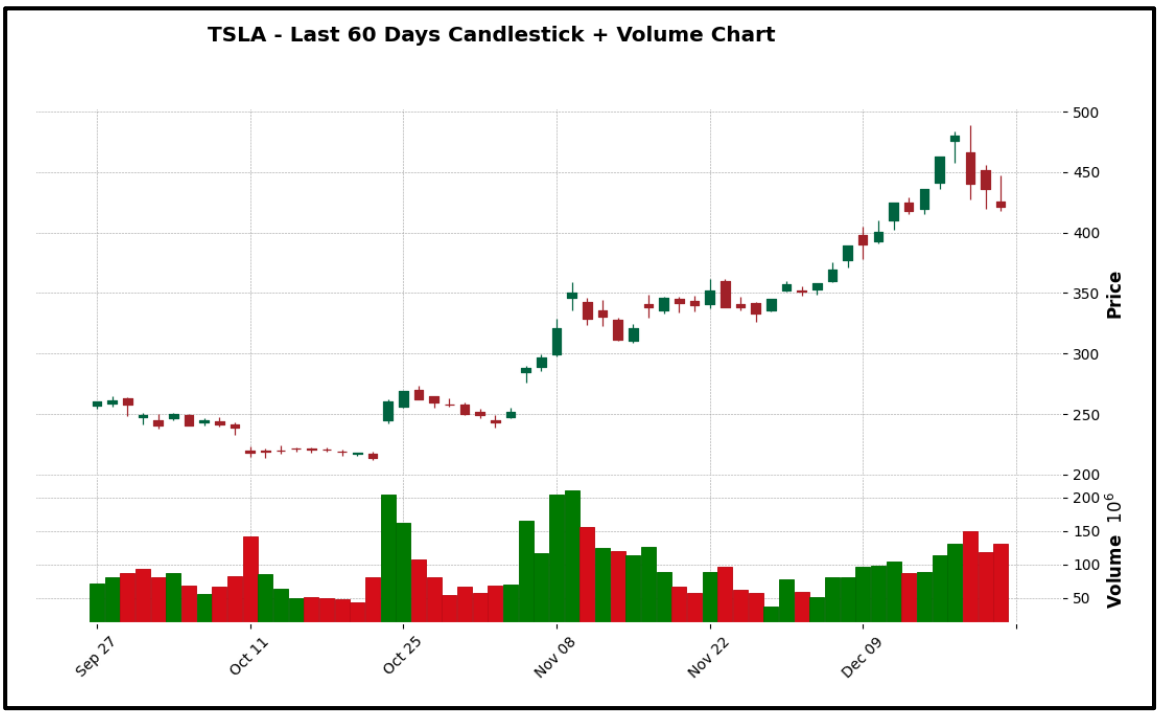


Mean Squared Error: 2876.312885110494

Predicted price for TCS.NS 30 days ahead: ₹ **4231.16**

- Input: **`TSLA`** (**Tesla** stock ticker) – **Global stock**
  - Output: Graphs showing trends and a predicted price 30 days ahead.

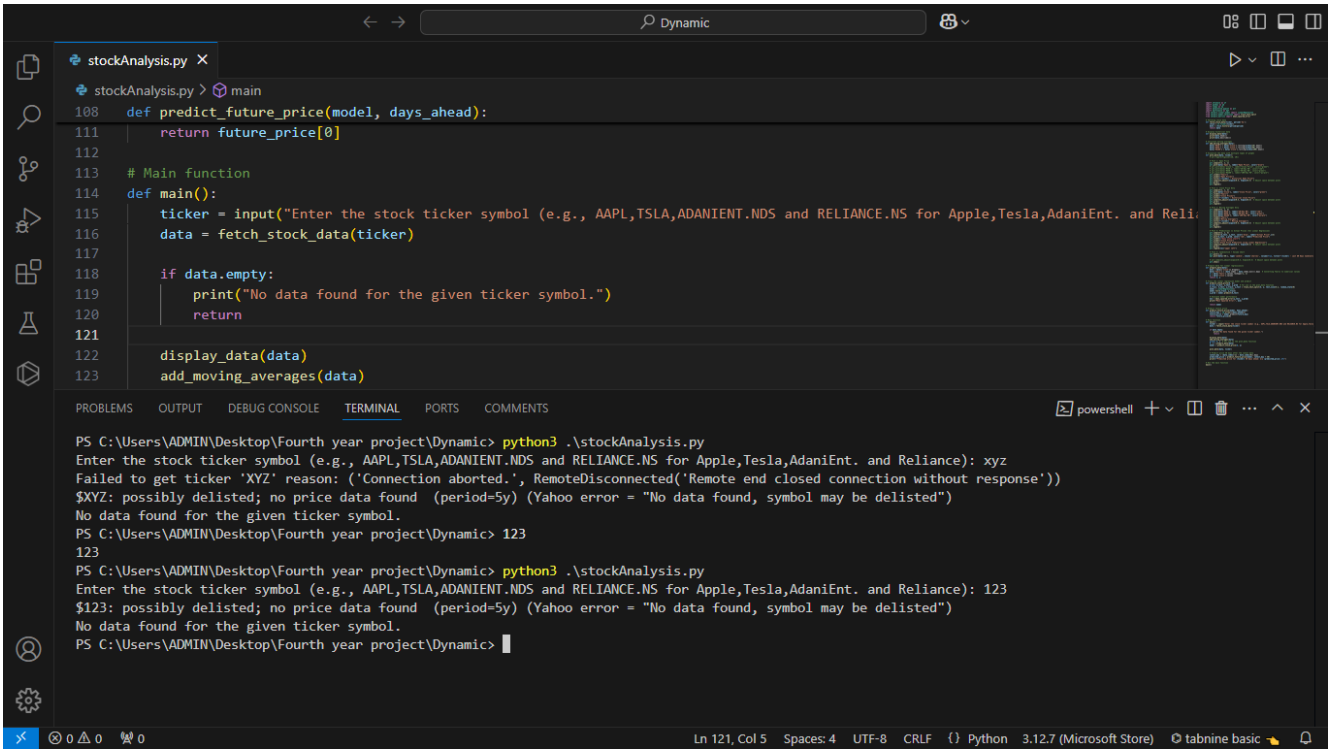




Mean Squared Error: 5380.949385292276

Predicted price for TSLA 30 days ahead: **$273.84**

- Input: Invalid ticker (e.g., **`XYZ`**)
  - Output: Error message indicating no data found.
  - **$XYZ: possibly delisted; no price data found (period=5y) (Yahoo error = "No data found, symbol may be delisted")**
  - **No data found for the given ticker symbol**.



❖ Observations and Results

- The moving averages provided insights into long-term and short-term trends.
- Linear regression predictions showed reasonable accuracy for short-term forecasting.

9. Challenges

❖ Basic Problems Encountered

1. **Handling Missing Data**:
   Some rows in the dataset contained missing values.
   - **Solution**: Removed rows with missing values.
2. **Overfitting in Linear Regression**:
   Linear regression performed poorly on volatile stocks.
   - **Solution**: Added a train-test split to evaluate model performance.

10. Conclusion

❖ What Was Learned?

- **Moving averages** are effective for trend analysis.
- **Linear regression** is a simple yet powerful tool for stock price prediction.

❖ Key Outcomes of the Project

- **Accurate predictions** for **stable** stock trends.
- Improved understanding of stock **data visualization** techniques.

11. References

- API Documentation — yahoofinance documentation
- Scikit-learn Documentation
- Matplotlib and mplfinance Tutorials
- Linear Regression Analysis | Linear Regression in Python | Machine Learning Algorithms | Simplilearn

12. Appendix

❖ Complete Code or GitHub Link

- **https://github.com/pratikrana1612/ai_project**