Objective:

The purpose of this assignment is to evaluate your ability to write production-level Python code, build APIs, and develop AI-driven solutions aligned with real-world SaaS applications.

Assignment Tasks:

Dataset:

You have been given a dataset in a separate file named - ***BiztelAI_DS_Dataset_V1***. This dataset consists of the chat transcript between 2 agents regarding articles published in the Washington post and these can be of different types of articles.

Task 1: Data Ingestion and Preprocessing

- Load and process the dataset using **Pandas** and **NumPy**.
- Handle missing values, duplicate records, and incorrect data types.
- Implement **object-oriented programming (OOP)** principles to create a modular data pipeline.
- Convert categorical variables into numerical representations in case applicable.
- Perform basic text preprocessing if applicable (tokenization, stopword removal, lemmatization).

Task 2: Exploratory Data Analysis (EDA)

- Conduct advanced **EDA** to uncover patterns and insights in the dataset.
- Figure out the fields of the data set and summarise the dataset (not the transcripts) at an article and agent wise level. Use your imagination on what could be important in the data to get insights on what agents talk about and how they talk.
- Generate statistical summaries and visualizations using **Matplotlib, Seaborn, or Plotly**.
- Analyze the data fields at each hierarchy and plot distinct data graphs which are applicable.
- You task would be take a particular chat transcript given to you and you have to summarise the entire chat and come up with below things
  - Possible article link that the agents were discussing about
  - Number of messages sent by agent 1 and agent 2
  - Overall sentiments of agent 1
  - Overall sentiment of agent 2

- For performing this above task, you will have to use any available open source light LLM and report accuracies of the above fields.
- You API should take one set of transcript for an article as an input and emit out
  - Possible article link
  - Number of messages by agent 1 and agent 2
  - Overall sentiments of agent 1
  - Overall sentiment of agent 2

Task 3: Building a REST API with FASTAPI/Flask

- Develop a **FASTAPI/Flask REST API** to expose key functionalities of the processed dataset.
- The API should provide:
  - **Endpoint 1:** Fetch and return processed dataset summary.
  - **Endpoint 2:** Perform real-time data transformation (e.g., convert new raw input into preprocessed form).
  - **Endpoint 3:** Allow users to send input and receive processed insights.
- Implement proper **error handling and logging**.
- Optimize the API for **performance and scalability**.

Task 4: Implementing OOP Concepts in Data Processing

- Use **object-oriented programming** to modularize data processing workflows.
- Implement classes and methods to handle different parts of the pipeline (e.g., `DataLoader`, `DataCleaner`, `DataTransformer`).
- Ensure the code is reusable and well-documented.

Task 5: Code Optimization and Performance Tuning

- Optimize Python code for efficiency using vectorized operations in **NumPy/Pandas**.
- Improve API response time by implementing **asynchronous processing**.
- Analyze bottlenecks and propose potential optimizations.

Evaluation Criteria:

- **Production-Ready Code** (Readability, Maintainability, Best Practices)
- **API Development** (Functionality, Performance, Error Handling)
- **Object-Oriented Programming** (Modularity, Reusability)
- **Efficiency & Optimization** (Time Complexity, Memory Usage)
- **Exploratory Data Analysis** (Insights, Visualizations, Feature Engineering)

Submission Guidelines:

- Submit a **GitHub repository** containing:
  - Python scripts/Jupyter notebooks for data processing and analysis.
  - API implementation in **FASTAPI/Flask**.
  - README with instructions to run the project.
  - Documentation explaining insights and methodologies.
- [Optional] Deploy the API and provide a hosted endpoint for testing.

Bonus Points:

- Implement an **authentication mechanism** for the API.
- Optimize data processing using **parallel computing/multiprocessing**.
- Use **Docker** to containerize the API and make it deployment-ready.