The problem statements are **optional**. We understand that many applicants may already have unique experiences or projects that showcase their skills, and we value those immensely. If your portfolio includes standout work, you don't need to attempt these problems.

However, if you haven't had the opportunity to work on substantial projects or wish to demonstrate your capabilities in a focused way, these problem statements provide an excellent opportunity to shine. Additionally, any work you do on these problem statements will be owned by you, and you are free to use it in your portfolio or future applications.

Feel free to select any one of the problem statements that aligns best with your skills, comfort level, and background.

Problem Statements

1. Design and Implement a Collaborative Document Management Backend

Your task is to design and implement a backend service for a collaborative document management application. The application should allow users to upload, share, and manage documents.

Requirements:

1. **Core Features:**
    - Users can upload and download documents.
    - Support basic metadata for documents (e.g., title, author, tags, uploaded date).
    - Enable document sharing with other users with different permissions:
        - Read-only
        - Edit
        - Admin
    - Implement document versioning (track changes made to documents).
2. **Scalability and Performance:**
    - Design the system to handle millions of documents and thousands of simultaneous users.

    - Ensure fast retrieval of documents and metadata.
3. **Search and Filters:**
    - Allow searching documents by title, tags, or author.
    - Provide filters for upload date and file size.
4. **Advanced Collaboration (Open-ended):**
    - Suggest additional features to enhance collaboration (e.g., commenting, real-time editing).
    - Justify the trade-offs and technical decisions involved in implementing them.
5. **Deployment:**
    - Provide a deployment-ready solution (Docker, CI/CD pipelines, etc.).

Deliverables:

1. **Codebase:**
    - Implement the core backend functionality.
    - Include tests to validate the correctness of your implementation.
2. **Design Document:**
    - Describe your architecture, design decisions, and trade-offs in a separate file (Markdown or PDF).
3. **Future Enhancements:**
    - Propose at least two additional features and explain how they could be implemented.

---

2. Build and Deploy a Customer Feedback Analysis System

You are tasked with developing a **machine learning pipeline** that analyzes customer feedback data to extract actionable insights. The system should handle data ingestion, preprocessing, model training, and deployment for real-time predictions.

Data sources: you can use any Kaggle datasets like- https://www.kaggle.com/datasets/arhamrumi/amazon-product-reviews

Requirements:

1. **Data Ingestion and Preprocessing:**

- Design a pipeline to handle raw customer feedback (e.g., text reviews) from multiple sources (CSV files, APIs).
- Clean the data by removing duplicates, handling missing values, and normalizing text.
- Identify and implement techniques to handle imbalanced datasets.

2. **Model Development:**
   - Train a machine learning model to classify feedback into predefined categories (e.g., Positive, Negative, Neutral).
   - Explore and justify your choice of model(s) (e.g., traditional ML like Random Forests vs deep learning like BERT).
   - Evaluate your model using appropriate metrics like precision, recall, and F1 score.
3. **Feature Engineering:**
   - Extract meaningful features from the feedback text (e.g., TF-IDF, embeddings).
   - Include exploratory data analysis (EDA) to support your feature selection decisions.
4. **Model Deployment:**
   - Develop an API to serve the model for real-time feedback classification.
   - Ensure the API can handle high concurrency and provides predictions with low latency.
5. **Monitoring and Feedback Loop:**
   - Implement a basic monitoring system to track model performance over time (e.g., data drift, prediction accuracy).
   - Suggest a strategy to retrain the model periodically with new data.

Deliverables:

1. **Codebase:**
   - A complete pipeline for preprocessing, model training, and deployment.
   - Include scripts for unit testing and performance benchmarking.
2. **Documentation:**
   - A detailed report explaining your pipeline, model selection, and trade-offs.
3. **API Endpoint:**
   - Host your model on a cloud platform or provide a Dockerized setup for running the API locally.
4. **Future Plan:**
   - Propose enhancements for handling multilingual feedback or incorporating customer demographic data.

Evaluation Criteria

1. Thought Process and Approach

- **Understanding the Problem**: Have you clearly identified the requirements and constraints of the chosen problem statement?
- **Solution Design**: Does your approach demonstrate a thoughtful design, considering real-world challenges?
- **Reasoning**: Are your decisions for architecture, tools, and algorithms supported by logical explanations?
- **Creativity**: Have you introduced unique or innovative elements in your solution?

2. Implementation and Technical Proficiency

- **Code Quality**: Is your code clean, modular, and well-documented?
- **Pipeline Completeness**:
    - For **Problem 1 (Collaborative Document Management)**: Have you addressed key functionalities like version control, access management, and collaboration features?
    - For **Problem 2 (Customer Feedback Analysis)**: Have you implemented the full ML pipeline, from data preprocessing to deployment?
- **Use of AI/ML (where applicable)**:
    - Did you effectively use AI tools, libraries, or models to solve specific parts of the problem?
    - Is the AI/ML model appropriate for the task and well-integrated into the overall system?

3. Real-World Application and Scalability

- **Practicality**: Does your solution address the problem in a realistic, applicable manner?
- **Scalability and Robustness**:
    - Is the backend/system designed to handle large-scale or dynamic workloads?
    - How well does the solution handle errors, edge cases, or data variations?
- **Performance**: Have you demonstrated or explained how your solution optimizes performance (e.g., response times, model accuracy)?

4. Communication and Presentation

- **Documentation**: Have you provided clear documentation for your solution, explaining its components, usage, and setup?
- **Clarity**: Is your submission organized and easy to understand?

General Guidelines

- **Use of AI Tools**: Using AI tools is encouraged. However, the focus is on **how you integrate and leverage AI** to enhance your solution, not merely on AI-generated outputs. Submissions that rely solely on AI-generated solutions without demonstrating your own thought process or understanding will not be considered for further evaluation.
- **Optionality**: Remember, these assignments are optional. If you have previous projects or experience that already showcase your skills, you can choose to highlight those instead.

Note

This evaluation is designed to assess both technical skills and problem-solving abilities. Whether your expertise lies in backend development or machine learning, the goal is to demonstrate your understanding, creativity, and ability to apply skills effectively.

Please submit your solution by emailing a link to your GitHub repository to **hr@kofukuidealabs.com**. Rest assured, we carefully review every application we receive.

Best of luck!