Technical Exam for Junior Data Engineer / Junior Data Analyst

Duration: 3 hours

Instructions:

1. **Objective:** Develop a Python application that demonstrates your understanding of core Python fundamentals, object-oriented programming (OOP) principles, problem-solving skills, and ability to interact with a database (MongoDB or MySQL).

2. **Tools & Technologies:**

    - **Programming Language:** Python 3.x
    - **Database:** Choose either MongoDB or MySQL
    - **Database Libraries:**
        - For MongoDB: `pymongo`
        - For MySQL: `PyMySQL` or `mysql-connector-python`
    - **Optional (but a plus):** Use multithreading or multiprocessing
    - **Version Control:** Git (please provide a Git repository link with your code)
3. **Requirements:**

    - Design and implement a modular codebase following OOP best practices.
    - Showcase your problem-solving skills by implementing custom algorithms or data structures where appropriate.
    - Avoid using frameworks or libraries that abstract away core functionalities (e.g., do not use ORMs like SQLAlchemy).
    - Include error handling and input validation.
    - Write clear and concise documentation and comments in your code.
    - Provide a README file with instructions on how to set up and run your application.
    - **Important:** While you may refer to documentation or resources for syntax and library usage, ensure that all code is written by you and reflects your coding style. We are interested in seeing how you approach and solve problems.
4. **Submission:**

    - Upload your code to a public or private Git repository (e.g., GitHub, GitLab).
    - Ensure that all necessary files are committed and pushed to the repository before the deadline.
    - Send an email with the repository link.

The Task:

Build a Task Management Application

You are tasked with creating a command-line **Task Management** application that allows users to manage their daily tasks. The application should support the following functionalities:

1. **Add a new task.**
2. **List all tasks with optional filtering (e.g., by due date, priority, or status).**
3. **Update a task's details.**
4. **Mark a task as completed.**
5. **Delete a task.**

Each task should have the following attributes:

- **Task ID** (unique identifier)
- **Title**
- **Description**
- **Due Date**
- **Priority Level** (e.g., Low, Medium, High)
- **Status** (e.g., Pending, In Progress, Completed)
- **Creation Timestamp**

Additional Requirements:

- **OOP Design:**
    - Create classes that represent the main entities (e.g., `Task`, `TaskManager`).
    - Use encapsulation to protect the internal state of your objects.
    - Implement methods that provide meaningful operations on your classes.
- **Data Structures & Algorithms:**
    - Choose appropriate data structures to store and manage the tasks within your application before persisting to the database.
    - Implement sorting and filtering algorithms for the task listing functionality.
    - Consider edge cases and optimize for efficiency.
- **Database Interaction:**
    - Interact with your chosen database (MongoDB or MySQL or Postgresql) to persist tasks.
    - Design your database schema to efficiently store task data.
    - Write your own queries using the selected database library (`pymongo` or `PyMySQL`).
- **Concurrency (Optional but a Plus):**

- Use multithreading or multiprocessing to handle multiple operations concurrently.
- For example, allow adding tasks while simultaneously performing background operations.

- **Command-Line Interface:**
  - Create an intuitive command-line interface (CLI) that allows users to interact with the application.
  - Provide clear instructions and feedback messages.
  - Handle invalid inputs gracefully.
- **Error Handling & Validation:**
  - Implement input validation for all user inputs.
  - Use try-except blocks to handle potential exceptions.
  - Provide meaningful error messages to the user.
- **Documentation & Code Style:**
  - Follow Python's PEP 8 style guide for your code.
  - Use docstrings to document your classes and methods.
  - Comment your code where necessary to explain complex logic.

Guidelines:

1. **Setup Instructions:** In your README, include clear instructions on how to set up and run your application, including any dependencies that need to be installed.
2. **Database Configuration:**
   - Provide sample configuration files or scripts needed to set up the database schema.
   - Ensure that your application can be easily connected to a new database instance.
3. **Data Persistence:**
   - All tasks should be persisted in the database. Upon restarting the application, the tasks should be loaded from the database.
4. **Time Management:**
   - Be mindful of the time limit. Prioritize core functionalities first before adding optional features.

Good Luck!

We are excited to see how you approach this task and look forward to reviewing your work. If you have any questions or need clarification on the task, please reach out promptly within the allotted time.