Dynamic-Eye
Entry Level Task.

Implementation of a Secure REST API with Database Integration

Outline:

1. Objective:
The objective of this task is to assess the developer's skills and proficiency as a back end developer in Python,Golang,C,C++ programming language. The task involves creating a secure REST API with authentication and integrating it with a database system (MongoDB/MySQL/PostgreSQL).

2. Requirements:
- Programming Language: Any Of Python, Golang, C, C++ ….
- Framework: Any Preferred Framework of the Programmer's choosing. We mainly prefer FastAPI if choosing python language, or Fiber if choosing Golang.
- Database: Choose one from MongoDB, MySQL, or PostgreSQL
- Security: Implement authentication using a token-based approach (e.g., JWT), And/Or Cookie based authentication with encryption.
- API Specification: Follow RESTful principles and design the API endpoints accordingly

3. Task Description:

a. Setup:
- Install the required dependencies (e.g., Python, Flask, fastAPI, Golang, Fiber, Gin, Echo, ...and the chosen database system)
- Initialize a new project directory for the task

b. Database Setup:
- Create a database schema or collection based on the chosen database system
- Define the necessary tables or documents to store data related to the REST API's functionality

c. API Design:
- Identify the required endpoints for the REST API based on the given scenario and the expected functionality
- Design the API endpoints, including the necessary request/response formats and parameters
- Ensure adherence to RESTful principles (e.g., proper HTTP methods, resource naming conventions)

d. Authentication:
- Implement the relevant authentication mechanism (e.g., JWT) to secure the API
- Create an authentication endpoint for user login and token generation
- Include proper error handling for invalid or expired tokens

e. API Implementation:
- Implement the defined API endpoints using the chosen framework.
- Integrate the database system to perform CRUD (Create, Read, Update, Delete) operations
- Include appropriate error handling and validation for input data

f. Testing:
- Write unit tests to verify the functionality of the implemented API endpoints
- Test various scenarios, including positive and negative test cases
- Ensure that the authentication mechanism and database integration are working correctly

g. Documentation:
- Document the API endpoints, their functionality, and the expected request/response formats
- Provide clear instructions on how to set up and run the API locally
- Document any assumptions made during the implementation

Good luck with the task! We look forward to seeing your implementation.