Project Brief: API-Based Products This project requires M.Tech students to develop a comprehensive API-based product, demonstrating a deep understanding of modern API design, implementation, and documentation principles.

Project Objective The objective is to design, implement, and document a robust set of APIs for a chosen software product or service. Students will gain practical experience in building REST and GraphQL APIs, understanding their respective strengths, and deploying a functional backend.

Project Scope & Key Deliverables

1. Product/Service Selection

- * Task: Choose a software product or service that can be powered by APIs.
    - Examples include: a task management application, a social media backend, an e-commerce platform, or a library management system.
- **Deliverable:** A one-page document outlining the selected product/service and its core functionalities.

2. API Design & Implementation

- * Task: Design and implement a set of APIs for your chosen product. Both REST and GraphQL implementations are mandatory.
- * **REST API:** Design endpoints following RESTful principles (e.g., using HTTP verbs, clear resource paths). The implementation should cover CRUD (Create, Read, Update, Delete) operations for key resources.
- * **GraphQL API:** Design a single endpoint with a schema that allows for flexible data querying and manipulation. This should leverage GraphQL's strengths, such as fetching multiple related resources in a single request.

- **Deliverable:** A functional server-side implementation of both API types.

3. API Documentation *

- Task: Document both the REST and GraphQL APIs effectively.
- * **REST API:** Create detailed documentation using the **OpenAPI (Swagger)** specification. This should include schema definitions, example requests/responses, and explanations for each endpoint.

- * **GraphQL API:** Ensure the GraphQL schema is well-defined and self-documenting. The implementation must support **introspection**, allowing developers to explore the schema directly.

4. Demo & Validation

- Task: Demonstrate the functionality of your implemented APIs.
- This can be done via one of the following methods:
- * **UI:** A simple front-end interface that consumes the APIs.
- * **Postman Collections:** A well-organized Postman collection with clear use cases and corresponding requests to validate API functionality.

Deliverable: A live demo showcasing the APIs' capabilities.

Evaluation Criteria (Tentative)

- Implementation Quality (40%): Correctness, efficiency, and robustness of the server-side code.
- API Design (30%): Adherence to RESTful principles, clarity of the GraphQL schema, and effective use of both API styles.
- Documentation (20%): Completeness and quality of the OpenAPI documentation and the self-documenting GraphQL schema.
- Demo (10%): Clarity and effectiveness of the final demo.

Suggested Timeline

- **Week 1-2:** Team formation, product selection, and initial API design.
- **Week 3-6:** Backend implementation of both REST and GraphQL APIs.
- **Week 7:** Documentation of APIs and preparation of the demo.
- **Week 8:** Final report submission and project demonstration.