**Project Task: Create a Web Application with CRUD Operations and Advanced Database Features**

**Objective**

Develop a web application with the following requirements:

1. **CRUD functionality** for **5-6 entities**.
2. Utilize an **SQL database** to store most of the entities with a variety of relationships (one-to-one, one-to-many, many-to-many).
3. Implement a **MongoDB database** to handle a single collection for unstructured or semi-structured data.
4. Use **Redis** as a caching layer to optimize performance for frequently accessed data.

**Key Deliverables**

**1. Web Application**

* **Frontend**:
  + Design a user-friendly interface for creating, reading, updating, and deleting records for each entity.
  + Include forms, tables, and modals for CRUD operations.
  + Framework: up to student (can be combined with backend: e.g. MVC)
* **Backend**:
  + Create RESTful APIs endpoints or MVC application to handle CRUD operations for the entities.
  + Framework: (e.g., Node.js, Python Flask/Django, Java Spring Boot, or .NET).

**2. SQL Database**

* **Entities and Relationships**:
  + Model and implement 5-6 entities in the database. Examples:
    1. **Users**
    2. **Products**
    3. **Orders**
    4. **Categories**
    5. **Reviews**
    6. **Payments**
  + Include the following relationships:
    1. **One-to-One**: Example: User ↔ Profile
    2. **One-to-Many**: Example: User ↔ Orders
    3. **Many-to-Many**: Example: Products ↔ Categories (via a join table).
  + Use a relational database management system (e.g., PostgreSQL, MySQL, MS SQL).
* **Features**:
  + Use proper schema design with constraints (primary keys, foreign keys, unique constraints).
  + Add indexing to optimize performance for frequent queries.
  + Write at least 5 complex queries (e.g., joins, aggregations) for testing the relationships.

**3. MongoDB Collection**

* **Unstructured Data**:
  + Design a MongoDB collection for a specific use case (e.g., storing product reviews, user activity logs, or notifications).
  + Define a schema for the collection and integrate it with the web app using an ODM (e.g., Mongoose, Mongo Db Driver).
  + Implement CRUD operations for this collection via the backend APIs.

**4. Redis Cache**

* **Caching Layer**:
  + Implement Redis as a caching solution to improve application performance.
  + Use Redis to cache:
    - Frequently accessed data (e.g., user profiles or product details).
    - Query results from the SQL or MongoDB databases.
  + Implement cache expiration policies (e.g., TTL) and cache invalidation logic.