1. **API Endpoints:**

   CRUD Operations

   - Create (POST): Add new resources to the system (new book to a library).
   - Read (GET): Retrieve specific resources or collections of resources (get a book by ID, list all book).
   - Update (PUT/PATCH): Modify existing resources (change a book's title,).
   - Delete (DELETE): Remove resources from the system (delete a book).

   Additional Endpoints

   - Depending on the domain, you might require endpoints for search, filtering, sorting, authentication, authorization, or other domain-specific actions.

2. **Request/Response Formats (JSON):**

   - Specify the structure of the JSON data expected in requests and returned in responses.
   - Define any required or optional fields, their data types, and any validation rules.

3. **Data Validation and Error Handling:**

   - Implement input validation to ensure data integrity and security.
   - Handle potential errors gracefully (e.g., invalid input, resource not found) and return appropriate error responses with meaningful messages and status codes.

4. **Data Modeling (SQLAlchemy or Django ORM):**

   - Design a database schema using an ORM like SQLAlchemy or Django's built-in ORM.
   - Define models for each resource type and establish relationships between models (if applicable).

5. **Framework Usage (Flask or Django):**

   - Demonstrate proficiency in the chosen framework by using its features effectively.
   - Leverage routing mechanisms, request handling, template rendering (if applicable), and other relevant features.

6. **Security Considerations:**

   - Sanitize input data to prevent common vulnerabilities like cross-site scripting (XSS) or SQL injection.
   - If applicable, implement authentication and authorization mechanisms to protect sensitive resources.

7. **Testing:**

   - Write unit tests to verify the correctness of individual functions or modules.
   - Write integration tests to ensure that different components of the API work together seamlessly.