

Neighborhood Library Service – Take-Home Test

Overview

A small neighborhood library wants a new service to manage its members, books, and lending operations. You must build this service using:

- **Python** for the server implementation
 - **gRPC** with **Protocol Buffers** for the service interface
 - **PostgreSQL** as the data store
-

Functional Requirements

- The library needs to track:
 - **Books** (title, author, etc.)
 - **Members** (name, contact info, etc.)
 - **Borrowing/Returning** operations (who borrowed what, when, etc.)

At a minimum, your service should allow the library staff to:

1. **Create/Update** records for books and members.
2. **Record** when a member borrows a book.
3. **Record** when a borrowed book is returned.
4. **Query** or list borrowed books in some fashion (e.g., all books a certain member has out).

You may include more functionality if you see fit (e.g., handling overdue books, tracking fines, due dates, etc.), but the above points are the core requirements.

Your Task

1. Design the Database Schema

- Decide on the tables/entities you'll need, how they relate, and what fields/columns belong in each.
- You'll store this data in PostgreSQL.

2. Define the gRPC Service

- Create one or more `.proto` files with the necessary Protobuf message types and service definitions.
- Determine which RPC methods (and their request/response messages) are needed to support the core library operations.

3. Implement the Python gRPC Server

- Write Python code to spin up a gRPC server.
- In each RPC method, interact with your PostgreSQL database (create/read/update records, etc.).

4. Frontend

- Create a Reactjs application and Integrate with the backend services.
- Use NodeJS as an API Gateway for integration with gRPC.
- Showcase a minimal UI design. Integrate any components, libraries as needed to showcase aesthetics.
- Design screens that have integration with at least 3 different API's. For eg. List books, create book, borrow book..

5. Documentation/Delivery

- Provide a short **README** detailing how to:
 - Set up the database (scripts or Docker instructions, etc.).
 - Generate or compile the `.proto` files (if needed).
 - Run the Python server (command-line steps, required libraries, environment variables).

- Include any necessary files or folders so another developer can easily build, run, and test your service.
-

Evaluation Criteria

1. Schema Design

- Clarity, normalization, and use of relationships in PostgreSQL.

2. Service Interface

- How intuitive and well-structured your gRPC endpoints and Protobuf messages are.

3. Code Quality

- Organization, readability, and maintainability of your Python server code.

4. Documentation

- Ease of setup and clear explanation of how to test the service.
-

Tip (Optional)

- Consider adding **error handling** for cases like trying to borrow a book that's already checked out.
- Implementing **validation** (e.g., ensuring valid inputs for new records).
- Show how a client might call your service (e.g., providing a sample script).

Goal: Implement a robust yet concise gRPC-based service in Python, backed by PostgreSQL, that meets the library's borrowing/returning requirements. Be creative with the design, but ensure you cover the core use cases above. Good luck!