Gramedia Digital
Backend Developer Take-Home Test
(Golang)

Scenario

You are developing a simple RESTful API service for a book management system — similar to what a digital library or bookstore backend might provide. The API will allow users to manage books, authors, and user favorites.

Requirements

1. Core Features

Build a REST API service with the following endpoints:

Endpoint	Method	Description
/books	GET	List all books (with optional pagination, search by title/author, and category filter)
/books/:id	GET	Get details of a specific book
/books	POST	Add a new book
/books/:id	PUT	Update a book
/books/:id	DELETE	Delete a book
/favorites	GET	List user's favorite books
/favorites	POST	Add a book to user's favorites

2. Technical Expectations

- Use Go (Golang) — preferably Go 1.20+
- Use any web framework (or standard net/http) such as Gin, Fiber, or Echo
- Connect to a database (you may use SQLite, PostgreSQL, or MySQL)
- Use GORM, sqlx, or raw SQL (up to your preference)
- Implement clean and modular architecture (layered or hexagonal is a plus)
- Include error handling, logging, and basic validation

- Provide README with setup instructions and explanation of your design choices
- Bonus: Add JWT authentication (optional)

🧮 Free Sample APIs & Tools (for reference or mock data)

- Open Library API — https://openlibrary.org/developers/api
- DummyJSON — https://dummyjson.com/products (for placeholder data)
- Go Playground — https://go.dev/play/ (for snippets testing)
- Postman Echo — https://www.postman-echo.com (for mock requests)

Evaluation Criteria

Category	Description
Code Quality	Clean, readable, idiomatic Go code
Architecture	Proper separation of concerns, maintainable structure
API Design	RESTful conventions, proper response structure
Database Handling	Efficient queries, migrations, and models
Error Handling	Descriptive, consistent error responses
Testing	Unit tests or integration tests where appropriate
Documentation	Clear README, setup steps, and explanation of design

Deliverables

- GitHub repository or zipped project folder.
- A README.md file containing:
  - How to run the project locally (with go run, Docker, or Makefile).
  - Example API requests (using cURL or Postman collection).
  - Explanation of chosen architecture and trade-offs.
  - (Optional) What improvements you would add with more time.

Optional Enhancements (Bonus)

- Add JWT-based user authentication.
- Implement Swagger/OpenAPI documentation.
- Add Dockerfile or docker-compose.
- Integrate unit tests or Postman tests.
- Deploy to a free hosting (e.g., Render, Railway, Fly.io).