Task: Build a Go-based REST API for an Instagram-like app

You are tasked with building a Go-based REST API for an Instagram-like app. The API should provide endpoints for users to perform CRUD operations on posts. Posts can include text, images, and videos. Users can like posts and comment on them. Your goal is to create a robust and secure API that ensures proper data validation and storage.

Requirements

1. **Setup and Database:**
    - Create a PostgreSQL database to store user data and posts.
    - Design the necessary tables for users and posts, including relationships for likes and comments.
2. **User Management:**
    - Implement user registration (POST /api/users) and authentication.
    - Users should be able to log in and receive a JWT token for authentication.
3. **Post Management:**
    - Create a RESTful API for posts that supports the following operations:
        - Create a new post (POST /api/posts).
        - Retrieve a specific post by ID (GET /api/posts/{postID}).
        - List all posts (GET /api/posts).
        - Update an existing post (PUT /api/posts/{postID}).
        - Delete a post (DELETE /api/posts/{postID}).
        - Users should be able to add text, images, and videos to their posts.
4. **Documentation:**
    - Document how to set up and run the application.

Optional Requirements

These are good to have bonus requirements and are not mandatory.

1. **Like and Comment:**

- Users should be able to like a post (POST /api/posts/{postID}/like) and see the number of likes.
- Users should be able to comment on a post (POST /api/posts/{postID}/comment).
- Retrieve comments for a post (GET /api/posts/{postID}/comments).

2. **Validation and Security:**
    - Implement validation for user inputs to prevent SQL injection, XSS, and other security vulnerabilities.
    - Implement authentication and authorization for API endpoints. Only the user who created a post should be able to update or delete it.
    - Secure user authentication and token management.
3. **Testing:**
    - Write unit tests to ensure the API endpoints function as expected.
    - Include test cases for both success and error scenarios.
4. **Documentation:**
    - Provide API documentation (e.g., Swagger) explaining the available endpoints.

Notes

- The expected duration for completing this test is approximately 1 week. Please focus on core functionality and avoid unnecessary complexity.
- Candidates should use Go for the backend and PostgreSQL for the database.
- The provided test requirements are high-level. Candidates should feel free to make necessary assumptions about details not explicitly specified, but their assumptions should be documented.
- Candidates should use best practices for Go and database interactions.

Deliverables:

1. The Go source code for your Lambda function.
2. A README file with instructions on how to deploy and test your GO REST APIs.
3. A brief explanation of any design decisions you made.

Evaluation Criteria:

1. Correctness: Does the GO REST APIs perform the required tasks correctly?
2. Code Quality: Is the code well-organized, readable, and maintainable?
3. Error Handling: Does the code handle errors gracefully and provide meaningful error messages?
4. GO Best Practices: Did you follow GO best practices for REST APIs?
5. Documentation: Is the README clear and concise, providing instructions for deploying and testing the GO REST APIs?

Submission:

Please send the GitHub reporsitory of the the deliverables mentioned above and email the GitHub URL to us within 1 week after receiving this email.

Note: This test is designed to assess your ability to work with GO REST APIs, so make sure to follow best practices and provide clean, well-documented code. Good luck!