Assignment: Building a Scalable RESTful API in Go

Objective: Develop a scalable RESTful API that handles user management, including user creation, retrieval, update, and deletion. The API should be designed with best practices in mind, showcasing efficient coding, proper error handling, and scalability considerations.

Requirements:

1. **Project Setup:**
    - Use Go modules for dependency management.
    - Include a README.md with setup instructions, design decisions, and any assumptions made.
2. **API Specifications:**
    - Implement the following endpoints:
        - POST /users - Create a new user.
        - GET /users/{id} - Retrieve user details by ID.
        - PUT /users/{id} - Update user information by ID.
        - DELETE /users/{id} - Delete a user by ID.
        - GET /users - List all users with pagination.
3. **Database Integration:**
    - Use a relational database (e.g., PostgreSQL, MySQL).
    - Design a simple schema for user management.
    - Use an ORM (e.g., GORM) or write raw SQL queries.
    - Include migrations with the possibility to migrate up and down.
4. **Error Handling:**
    - Implement comprehensive error handling for different scenarios (e.g., user not found, validation errors).
5. **Testing:**
    - Write unit tests for the core logic.
    - Include integration tests for the API endpoints.
6. **Code Quality:**
    - Follow Go best practices and idiomatic Go code style.
    - Ensure the code is well-documented and maintainable.
7. **Scalability Considerations:**
    - Discuss in the README.md how the API can be scaled horizontally.
    - Design the system to be able to handle 30k requests per minute with a latency of < 100ms.
8. **Security:**
    - Implement basic security features, such as input validation and protection against common vulnerabilities (e.g., SQL injection).
    - Discuss in the README.md how you would handle authentication and authorization if required.

Deliverables:

1. Source code hosted on a Git repository (e.g., GitHub, GitLab).
2. README.md file with:
    - Instructions on how to set up and run the project.
3. Test coverage report.

Presentation:

During the interview, the candidate should present:

1. **Project Walkthrough:** A brief walkthrough of the project structure, key components, and functionalities.
2. **Design and Architectural Decisions:** Discussion on why specific design choices were made, how the application can be scaled, and any trade-offs considered.
3. **Code Review:** A deep dive into a few critical parts of the code to demonstrate their coding style, best practices, and problem-solving skills.
4. **Testing:** Explanation of the testing strategy and demonstration of how tests are executed.
5. **Mentorship Discussion:** How would you mentor and guide less experienced developers on the team, using examples from the project.