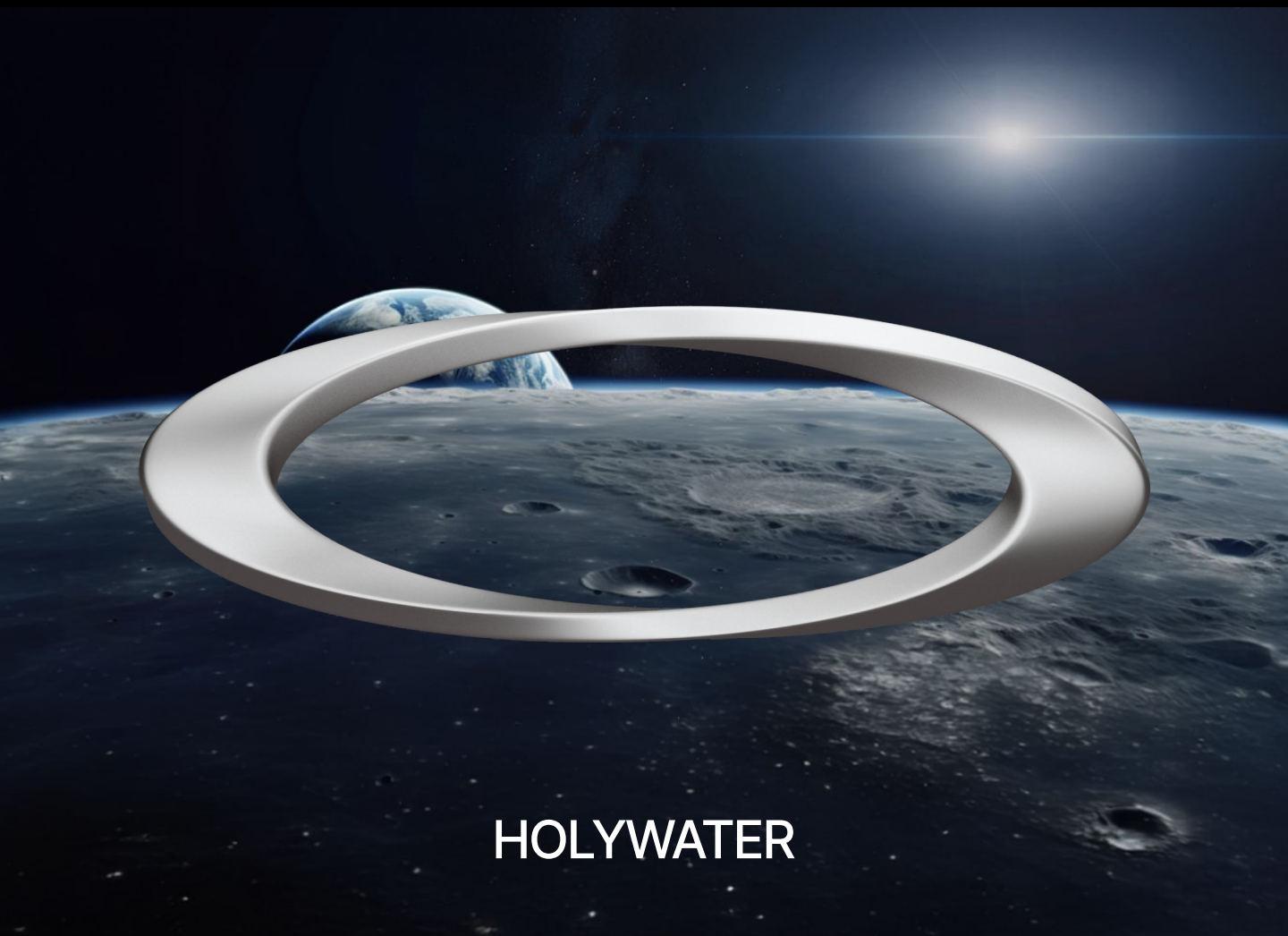
Middle Node.JS Developer

TEST TASK

17 (JAN) 25



Hello!

Thank you for your interest in HOLYWATER!
Our tasks offer exciting challenges and opportunities for growth.

To successfully complete this test task, you will need strong technical skills, problem-solving abilities, and attention to detail.

This is your chance to showcase your expertise and potential.
Good luck!

Overview

→ Develop a Books Content Management System (CMS) using NestJS, GraphQL, PostgreSQL, Redis, DynamoDB, and AWS. The system should allow users to manage book information, handle large datasets efficiently, and ensure high availability and security.

Requirements

1. System Architecture:
— Design a service for the Books CMS using NestJS.
— Use GraphQL to handle data fetching and mutations efficiently.

2. Database Design:
— Utilize PostgreSQL for storing relational data such as book details (title, author, publication date, etc.).
— Use DynamoDB for managing non-relational data such as user activity logs and book reviews.
— Implement caching mechanisms with Redis to enhance data retrieval performance.

NEXT PAGE

3. **Features:**
— CRUD Operations: Implement create, read, update, and delete operations for book entries.
— Search : Implement a robust search functionality for books with filters like author, title, and publication year.
— Sorting and Pagination: Enable sorting by different book attributes and pagination for large result sets.
— User Authentication and Authorization: Secure the system using JWT-based authentication and ACLs (Access Control Lists) to manage user permissions.
— Rate Limiting: Implement rate limiting to prevent abuse of the system.

4. **Scalability and Performance:**
— Design the system to handle high loads, especially focusing on efficient query design and database indexing.
— Ensure the system is scalable horizontally by designing stateless services that can be deployed in a load-balanced environment within AWS.

5. **Security:**
— Ensure secure storage of sensitive data (e.g., user information).
— Implement necessary security headers and proper handling of errors to prevent common vulnerabilities like SQL injection, XSS, etc.

6. **Testing:**
— Write unit tests and integration tests covering major functionalities.
— Document how to run the tests.

7. **Documentation:**
— Provide a README file with setup and deployment instructions.
— Document the API endpoints and provide example queries and mutations.

NEXT PAGE

Submission Guidelines

— Deliver the code repository (preferably on GitLab/GitHub or a similar platform) with complete source code and all relevant files.
— Ensure the code is well-commented and adheres to best coding practices.
— Provide a Postman collection or a similar setup for API testing.

Evaluation Criteria

— Code Quality and Readability: Clean, modular, well-documented code using best practices.
— System Design: Efficient use of patterns and structures suited for high-load environments.
— Database Management: Effective use of relational and non-relational databases, including aspects like schema design and query optimization.
— Security: Implementation of robust security measures.
— Performance and Scalability: System's performance under load and its scalability potential.
— Testing: Coverage and effectiveness of tests.

Please feel free to ask any additional questions! You can provide your answers in any way convenient for you.

Good luck! [You can do it]