Backend-Heavy Project Proposal: Task Management API

Project Overview

Create a backend service for a Task Management API that allows users to manage tasks, assign priorities, and set due dates.

The project will focus on building robust backend functionalities, utilizing Node.js and PostgreSQL.

Tech Stack

- Node.js: Backend runtime environment

- Express.js: Web framework for creating APIs

- PostgreSQL: Relational database for structured data

- Sequelize: ORM for PostgreSQL

- Jest: Testing framework for unit tests

- Swagger: API documentation

- Docker: Containerization for deployment

- PDFKit: For generating downloadable reports in PDF format

Project Flow

1. User Registration & Authentication:

- User signup and login endpoints with JWT authentication.

- Password hashing with bcrypt.

2. Task CRUD Operations:

- Endpoints to create, read, update, and delete tasks.

- Tasks have attributes: title, description, priority, due date, and status.

3. Task Filtering and Sorting:

- Retrieve tasks based on priority, status, or due date.

4. Task Report Generation:

- Generate a PDF summary of tasks and their statuses.

5. Testing & Documentation:

- Write unit tests and generate Swagger API documentation.

6. Dockerization:

- Package the application for easy deployment.

Daily Milestones and Prompts

Day 1: Project Setup

- Initialize Node.js project.

- Install dependencies (Express, Sequelize, PostgreSQL, bcrypt, etc.).

- Set up PostgreSQL database and Sequelize models.

- Set up Docker environment.

Prompts:

- Can you help me set up the folder structure for this project?

- How do I configure Sequelize to connect to a PostgreSQL database?

- Can you provide a Dockerfile for this application?

Day 2: User Authentication

- Build user signup and login routes.

- Add JWT-based authentication middleware.

- Test authentication endpoints.

Prompts:

- How do I hash passwords securely using bcrypt?

- Can you help me design the JWT authentication middleware?

- What tests should I include for the authentication module?

Day 3: Task Management API

- Create Sequelize models for tasks.

- Build CRUD endpoints for tasks.

- Test task-related endpoints with Postman.

Prompts:

- Can you provide a Sequelize model for a task with the required attributes?

- How should I structure the routes for CRUD operations?

- What is the best way to test these endpoints?

Day 4: Task Filters and Sorting

- Implement filters for task retrieval by priority, status, and due date.

- Add pagination for task retrieval.

- Write integration tests for these features.

Prompts:

- How do I implement pagination with Sequelize queries?

- Can you help me write a query to filter tasks by priority and due date?

- What are some common test cases for filtering and pagination?

Day 5: Report Generation

- Use PDFKit to generate task summary PDFs.

- Add an endpoint to download task reports.

- Test PDF generation and download functionality.

Prompts:

- Can you provide an example of generating a PDF using PDFKit?

- How do I set up a route to serve PDF files for download?

- What tests should I write for the PDF generation feature?

Day 6: Documentation and Testing

- Write Swagger documentation for all API endpoints.

- Complete unit tests using Jest.

- Ensure code coverage of at least 80%.

Prompts:

- How do I set up Swagger documentation for this project?

- What are the best practices for writing unit tests with Jest?

- Can you help me generate a code coverage report?

Day 7: Finalization and Deployment

- Perform final code reviews and testing.

- Push the Dockerized application to a container registry.

- Deploy the application locally using Docker Compose.

Prompts:

- Can you help me create a Docker Compose file for local deployment?

- How do I perform a final review for code quality?

- What steps should I take to push the Docker image to a registry?

Deliverables

1. Backend Code: Node.js project with structured folder organization.

2. Database Schema: PostgreSQL database with well-defined tables.

3. API Documentation: Swagger documentation hosted on /docs endpoint.

4. Tests: Jest test suite with high code coverage.

5. Dockerized Application: Ready-to-deploy containerized application.

6. PDF Report Feature: Task summary report downloadable as a PDF.

Download PDF

Generated project details are included in the attached PDF. Use this plan to efficiently complete the project within the given timeline.