Intern Software Engineer Selection Process: Task Assignment

Introduction

Welcome to the intern software engineer selection process for Fluxs. To assess your skills and problem-solving abilities, we have prepared four different tasks using the MERN stack (MongoDB, Express, React, and Node.js). Each task reflects real-world challenges we face at our company, and completing one will demonstrate your proficiency in essential technologies and practices.

You are required to select **one task** from the list below and complete it within **5 days**. We expect you to use your own approach and resources to complete the task. No additional resources or guidance will be provided, as this is also an assessment of your ability to research and work independently.

Task Selection

Please review the two tasks described below, Once you have selected a task, you will have **5 days** to complete and submit it.

Task 1: Task Management with Role-Based Access Control (RBAC)

Overview:
Develop a task management application where users are assigned one of three roles: **Admin**, **Manager**, or **Employee**. Each role has distinct access permissions to manage tasks:

- **Admins**: Full access to the system. They can manage (create, update, delete) all users and tasks.
- **Managers**: Can create and assign tasks to employees within their team. They can also update or delete tasks assigned to their team members but cannot manage other managers or employees outside their team.
- **Employees**: Can view tasks assigned to them and update their task status (e.g., mark as complete, in progress).

The app should enforce these permissions at the API level, ensuring each user can only perform actions allowed by their role.

Day-by-Day Plan:

- **Day 1**: Set up the backend with Node.js, MongoDB, and implement user authentication using JWT.
- **Day 2**: Create role-based access control (RBAC) and task management APIs.
- **Day 3**: Build a frontend in React with different views for Admins, Managers, and Employees.
- **Day 4-5**: Add tests for the APIs and frontend components, validate inputs, and complete documentation.

Key Technologies: Node.js, Express, MongoDB, JWT, React

Task 2: E-commerce Product Catalog with Filtering, Sorting, and Caching

Overview:
Develop an e-commerce product catalog that allows users to view products with advanced features such as filtering, sorting, and caching to improve performance for frequently accessed data.

- **Filtering**: Users should be able to filter products by categories, price ranges, and availability.
- **Sorting**: Products should be sortable by price, popularity, and ratings.
- **Caching**: Use caching to store frequently requested product data, improving response times for repeated requests.

The app should be efficient and optimized for a large dataset of products, ensuring that filtering and sorting operations are fast and responsive.

Day-by-Day Plan:

- **Day 1**: Set up the backend with Node.js and MongoDB, and implement basic CRUD (Create, Read, Update, Delete) operations for managing products.
- **Day 2**: Add filtering and sorting functionalities to the product catalog. Implement caching using Redis for frequently accessed data.
- **Day 3**: Build the frontend in React to display products with search, filtering, and sorting options.
- **Day 4-5**: Perform integration testing to ensure backend and frontend work together seamlessly, finalize the UI, and complete documentation.

Key Technologies: Node.js, Express, MongoDB, Redis, React

Expectations and Deliverables

For each task, you are expected to:

1. **Write well-documented code**: Include comments and a clear structure.
2. **Follow best practices**: Ensure clean code, proper error handling, and scalability.
3. **Testing**: Write unit tests for key components and integration tests for critical flows.
4. **Documentation**: Provide a README file with:
    - Installation instructions
    - A description of your architecture
    - API documentation (Insomnia/Postman)
    - Any assumptions made during development

Submission Guidelines

- You have **5 days** to complete your selected task after confirming your choice.
- Submit your work by pushing it to a GitHub repository and sharing the link with us.
- A link to your hosted application **or** a video link of the application running locally.
- Ensure that all tests are passing and included in your submission.

Selection Criteria

We will evaluate your submission based on the following criteria:

1. **Code Quality**: Clean, modular, and scalable code with proper error handling.
2. **Feature Completeness**: All required features implemented according to the task description.
3. **Testing**: Well-written tests that cover major components and flows.
4. **Documentation**: Clear and thorough documentation, including a README and API documentation.
5. **Timeliness**: Submission of the task within the allotted time.
