

FULL STACK PHP DEVELOPER TASKS

Building an Advanced Task Management Application

Overview:

You are assigned the challenging task of building an Advanced Task Management Application using PHP (Laravel) and JavaScript. The application should provide a robust and efficient task management system with real-time collaboration features. Users should be able to create, view, edit, delete, prioritize, assign, and track tasks in real-time. The task is designed to thoroughly test the candidate's proficiency in both front-end and back-end development, requiring complex features and optimizations.

Requirements:

1. Backend - PHP (Laravel)

- Set up a new Laravel project with a scalable architecture.
- Design a database schema for tasks, users, teams (if applicable), and their relationships.
- Implement RESTful APIs with proper authentication for user and task management.
- Utilize Laravel Eloquent's advanced features (e.g., relationships, query optimization) for efficient data retrieval.
- Implement a notification system for real-time updates on task assignments and status changes.

2. Frontend - JavaScript

- Design a dynamic and responsive user interface using Vue.js for the Task Management Application.
- Implement real-time updates for task status changes and assignments using WebSocket technology (e.g., Socket.io).
- Provide an interactive dashboard with drag-and-drop functionality to manage tasks effectively.
- Allow users to filter, sort, and search tasks based on various criteria.

3. User Authentication and Authorization

- Implement a multi-role authentication system (e.g., admin, manager, user) with appropriate permissions.
- Admins should have the ability to manage users, teams, and tasks across the system.

4. Task Collaboration and Comments

- Allow users to collaborate on tasks by adding comments and attachments.
- Implement real-time updates for task comments and attachments.

5. Task Assignment and Team Management

- Enable users to assign tasks to specific team members or individual users.
- Implement team management features, allowing users to create, edit, and delete teams.

6. Performance Optimization

- Optimize database queries, caching, and indexing for enhanced application performance.
- Implement asynchronous processing for long-running tasks to maintain application responsiveness.
- Use Redis or other caching mechanisms for managing real-time updates efficiently.

7. Security and Validation

- Implement robust security measures to protect against common vulnerabilities, such as SQL injection and XSS attacks.
- Apply form validation on both client and server sides to ensure data integrity.

8. Testing and Quality Assurance

- Write comprehensive unit tests, integration tests, and end-to-end tests to ensure the application's reliability.
- Set up continuous integration and continuous deployment (CI/CD) for automated testing and deployment.

9. Localization and Accessibility

- Provide localization support for multiple languages.
- Ensure the application meets accessibility standards (e.g., WCAG) for people with disabilities.

Instructions:

- Use PHP (Laravel) for the backend development and Vue.js for the front end.
- Follow best practices for code organization, readability, and maintainability.
- The user interface should be intuitive, visually appealing, and responsive.

- Performance and scalability are critical factors in evaluating the solution.
- Provide clear instructions on how to set up and run the application locally.
- Include a detailed README file with information about the project's structure, setup, testing, and any additional features.

Submission:

Submit a GitHub repository link containing your project's source code and a live demo link for testing.

The task should not take more than 2 days (48 hours).