Employee Management System – Full Stack

Overview:

In this task, you are required to develop an Employee Management System that encompasses various features for managing companies, departments, and employees. The system will allow users to create, read, update, and delete (CRUD) records for each of these entities. Additionally, there will be a workflow for handling the onboarding process of new employees and implementing role-based access control to ensure secure data handling. The system should adhere to the requirements specified below to ensure all features are implemented correctly.

While there are mandatory requirements that must be completed, there are also bonus requirements. The bonus requirements are optional and should only be implemented if you have extra time and it does not compromise the quality of the mandatory requirements. Achieving the bonus requirements is not essential but will be considered a plus.

This task is designed to evaluate your technical skills, problem-solving abilities, and attention to detail. It also assesses your capability to work under stress and tight schedules, reflecting real-world scenarios where meeting deadlines is crucial.

Requirements:

1. Back End:

a. Models:

i. User Accounts

- User Name
- Email Address (Login ID)
- Role

ii. Company:

- Company Name
- Number of Departments
- Number of Employees

iii. Department:

- Company (Select)
- Department Name
- Number of Employees

iv. Employee:

- Company (Select)
- Department (Select)
  - Only departments related to the selected company
- Employee Status
  - Status will be handled through a workflow
- Employee Name
- Email Address
- Mobile Number
- Address
- Designation (Position/Title)
- Hired On
  - Only if hired
- Days Employed
  - Only if hired

b. Validations & Business Logic:

Non-Functional Requirements

i. Validate all required fields are filled
ii. Validate email addresses and mobile numbers are in the correct format.
iii. Automatically calculate the number of departments and employees in the company
iv. Automatically calculate the number of employees in the department
v. Automatically calculate the number of days an employee has been with the company based on the hiring date
vi. Ensure that the Department field in Employee modes only accepts departments related to the selected company.
vii. Handle cascading deletions by ensuring either related records are properly managed or preventing the deletion if necessary.
viii. Handle errors and throw appropriate error codes and messages accordingly

c. **Workflow (Bonus):**
   i. Develop a workflow to model the onboarding process for new employees
   ii. Define stages and allowed transitions:
      - **Stages:** Application Received, Interview Scheduled, Hired, Not Accepted
      - **Transitions:**
         - From **Application Received** to **Interview Scheduled**
         - From **Application Received** to **Not Accepted**
         - From **Interview Scheduled** to **Hired** or **Not Accepted**

d. **Security & Permissions:**
   i. Implement role-based access control to ensure:
      - Only authorized personnel can view and edit data.
      - Different roles have different levels of access (e.g., Admin, Manager, Employee).
   ii. Use secure authentication and authorization mechanisms of your choice. (e.g., Sessions, Tokens).

e. **APIs:**
   i. Create a RESTful API that supports all CRUD operations for all models:
      - **Company**
         - GET: Retrieve a single company or list all companies
      - **Department**
         - GET: Retrieve a single department or list all departments
      - **Employee**
         - POST: Create a new employee
         - GET: Retrieve a single employee or list all employees
         - PATCH: Update an existing employee
         - DELETE: Delete an employee
   ii. Ensure the API handles data securely
   iii. Ensure the API follows RESTful conventions (e.g., using proper HTTP methods like GET, POST, etc.)
   iv. If applicable, provide clear documentation on the API endpoints, parameters and expected responses

f. **Testing (Bonus):**
   i. Include unit tests to validate individual components and functions.
   ii. Include integration tests to ensure different parts of the application work together correctly.

g. **Logging (Bonus):**
   i. Implement logging to track application behavior and capture errors.
   ii. Ensure logs are detailed enough to troubleshoot issues but do not expose sensitive information.

2. Front End:

a. User Interface (UI):

i. Develop a user interface to interact with all system features:

- **Login Page**
  - A landing page where users can log in using their credentials.
  - Implement authentication to secure the application.
- **Company Management**
  - **List Companies Page:** Display a list of all companies with options to view, edit, and delete each company.
  - **View Company Page:** View detailed information of a single company.
- **Department Management**
  - **List Departments Page:** Display a list of all departments with options to view, edit, and delete each department.
  - **View Department Page:** View detailed information of a single department.
- **Employee Management**
  - **List Employees Page:** Display a list of all employees with options to view, edit, and delete each employee.
  - **Create Employee Page:** Form to create a new employee.
  - **Edit Employee Page:** Form to edit an existing employee.
  - **View Employee Page:** View detailed information of a single employee.
- **User Account Management (Bonus):**
  - **Edit User Account Page:** Form to edit the user account details.
  - **View User Account Page:** View detailed information of the user account.
- **Summary Dashboard (Bonus):**
  - A landing page shown by default after successful logins to show simple analytics as a summary of the organization (e.g., Number of Companies & Employees).

ii. Ensure the UI is simple, organized, clear and clean.

iii. Ensure the UI provides navigation between different sections of the system.

b. Validations:

i. Validate all required fields are filled

ii. Validate email addresses and mobile numbers are in the correct format.

iii. Ensure that the Department field in Employee page only shows departments related to the selected company.

iv. Handle errors and show appropriate & user-friendly messages

c. **Employee Report (Bonus):**
    i. Display a report as a table to show detailed information about hired employees, including:
        - Employee Name
        - Email Address
        - Mobile Number
        - Position
        - Hired On
        - Days Employed
        - Company Name
        - Department Name

3. API Integration:

a. Integrate the frontend application with the backend API to facilitate data exchange.
b. Implement the features in the frontend by utilizing the provided APIs
c. Ensure that authentication and authorization is always handled throughout user activity
d. Handle all server-side thrown errors from API responses and provide appropriate & user-friendly messages as a feedback to users
e. Ensure smooth user experience by managing loading states and displaying relevant messages.

Task Submission:

Applicants are required to submit their tasks via a dedicated public GitHub repository through the submission form shared with you in the email. Follow the steps below to ensure a seamless delivery:

1. **GitHub Repository Creation:**
    - Create a public GitHub repository
2. **Repository Structure:**
    - Organize your repository with clear folder structure for different aspects of the task
3. **Code Submission:**
    - Commit your application code to the designated repository, ensuring that it includes all necessary files and directories.
4. **Documentation:**
    - Include a detailed README.md file containing the following:
        1. Comprehensive documentation detailing your approach, implementation details and any considerations made during the development process.
        2. Instructions on how to set up and run your application. Include any prerequisites, installation steps, and additional information required for a smooth evaluation.
        3. Clearly indicate the completion of each task (e.g., Check List). Mention any assumptions or considerations made during the implementation.
        4. If applicable, specify the security measures implemented, especially concerning role-based access control and data protection.
        5. If applicable, provide clear documentation on the API endpoints, parameters and expected responses or instructions on how to access the API documentation externally.
5. **Bonus Requirements:**
    - If you choose to implement any of the bonus requirements, ensure that it does not affect the quality and completion of the mandatory requirements. The bonus requirements are optional and should only be done if you have additional time and wish to showcase extra skills. Achieving these will be considered a plus during the evaluation.
6. **Deadline & Completion:**
    - Even if you are unable to complete all the requirements, it is crucial to submit your task by the specified deadline. Completion of all requirements is important, but submitting on time is a crucial part of the evaluation process and not completing all requirements does not mean failing. Make sure to indicate which parts are done and which are not in the README file to help us understand your progress.
7. **Submission:**
    - Share the GitHub repository link through the submission form shared with you in the email.

Task Evaluation Criteria:

All submitted tasks will be evaluated based on the following criteria

1. Code quality, modularity, structure, cleanness and adherence to best practices
2. API design and documentation
3. User interface and user experience
4. Functionality and accuracy of the data visualizations
5. Responsiveness and performance
6. Security measures and data validation
7. Functionality and completeness of the requirements