Pratik Shinde
pratik0562001@gmail.com
9730750636

Breaking down the Project

Breaking down the problem and analyzing the requirements step by step.

1) **Database Design:**
   - We need tables for 'Employee' , 'Department', 'EmployeeSalary'.
   - Employee tables should have fields like 'Name', 'Email', 'Address', 'Reporting Manager'.
   - Department table should have fields like Name, Floor.
   - EmployeeSalary table should have fields like Employee, Salary, Start Date, and End Date.

2) **Django Models:**
   - Create Django models for Employee, Department, and EmployeeSalary.
   - Establish relationships between Employee and Department.
   - Use ForeignKey for the Reporting Manager field in the Employee model.
   - Ensure proper indexing and constraints.

3) **Django Admin Customization:**
   - Customize Django admin pages for adding, updating, and deleting employees, departments, and employee salaries.
   - Set up appropriate filters and display options.
   - Utilize Django admin inline models for displaying related entities.

4) Hierarchy Visualization:

- Implement a reporting page for the Department module to display the employee hierarchy.
- Used JavaScript to enhance the dynamic display.
- Allow filtering by date range for viewing historical hierarchies.

5) Department-wise Salary Report:

- Create an admin page for the Employee Salary Management module to add, update, and delete salary entries.
- Develop a reporting page to view department-wise salary costs within a given date range.
- Utilize JavaScript to enhance the reporting page's interactivity.

6) Tech Stack and Libraries:

- Use Python and Django for the backend.
- Choose MySQL for the database, depending on your preference and requirements.
- Leverage Django ORM for database interactions.
- Use Django Admin for basic CRUD operations.
- Integrate JavaScript (jQuery or DataTables) for dynamic and interactive reporting pages.

7) GitHub Repository:

- Set up a GitHub repository to version control your code.
- Regularly commit and push your changes.
- Share the repository link for review.

8) Estimations:

- Break down the tasks further to estimate the time required for each.
- Allocate time for testing, debugging, and documentation.

Summary

Step-by-Step Summary for Creating an Employee Management System with Django

Step 1: Analyse Requirements and Models

- Understand the requirements for the Employee Management System, including Employee, Department, and Employee Salary modules.
- Define models for Employee, Department, and Employee Salary with the necessary attributes and relationships.

Step 2: Set Up Django Project

- Create a new Django project using the command: `django-admin startproject employee_management`.
- Navigate to the project directory using: `cd employee_management`.

Step 3: Create Django App

- Create a new Django app named employee_app using: `python manage.py startapp employee_app`.

Step 4: Define Models

- Define models for Employee, Department, and EmployeeSalary in the 'models.py' file inside the 'employee_app' app.

Step 5: Register Models in Admin Panel

- Register the models in the 'admin.py' file to make them accessible in the Django admin panel.

Step 6: Create Superuser

- Create a superuser account using: 'python manage.py createsuperuser'.
- Provide the necessary details, including username, email, and password.

Step 7: Run Migrations

- Run migrations to apply changes to the database schema: 'python manage.py makemigrations' and 'python manage.py migrate'.

Step 8: Add Sample Data in Admin Panel

- Use the Django admin panel to add sample data for Employee, Department, and Employee Salary.

Step 9: Create Views, Templates, and Forms

- Define views in the 'views.py' file for Employee List, Add Employee, Update Employee, and Delete Employee.
- Create templates in the 'templates' folder for Employee List, Add Employee, Update Employee, and Delete Employee using Bootstrap for styling.
- If needed, create a forms file ('forms.py') for custom form handling.

Step 10: Define URLs

- Define URLs in the ‘urls.py’ file to map views to specific URLs.
- Include these URLs in the main ‘urls.py’ file of the project.

Step 11: Implement Reporting Manager Functionality

- Update the `Employee` model to include a reporting manager with ‘on_delete=models.SET_NULL’ and ‘null=True’.
- Update the ‘EmployeeForm’ to handle reporting manager selection.

Step 12: Run the Development Server

- Run the development server: ‘python manage.py runserver’.
- Access the application in a web browser at ‘http://127.0.0.1:8000/’.

Step 13: Troubleshoot and Debug

- If there are any issues, troubleshoot by checking template paths, views, and template syntax.
- Restart the development server after making changes.

Step 14: Summary and Testing

- Test the Employee Management System by adding, updating, and deleting employees.
- Verify reporting manager functionality.
- Make adjustments as needed based on testing.

Step 15: Code Repository

- Upload the code to a version control system such as GitHub.

Step 16: Final Submission

- Share the repository link with relevant stakeholders for evaluation.