Project Kit

Title of the Project

Effort Tracking System

Abstract of the Project

The Employee Tracking System is a web-based application designed to streamline human resource management by automating the tracking of employee attendance, leaves, and activity logs. Developed using Django for the backend and React for the frontend, the system enables efficient registration of employees, daily attendance marking, leave management, and report generation. By digitizing core HR functions, the system reduces manual errors, improves transparency, and enhances overall organizational productivity. It provides role-based access for both admins and employees, ensuring secure and efficient data handling across the platform.

Generic keyword:

Employee Tracking, HR Automation, Attendance Management System, Leave Management, Role-Based Access, Human Resource Software, Staff Monitoring, Employee Dashboard, Admin Panel, Activity Logging, Django Backend, React Frontend, Web-Based HRMS, Daily Attendance Logs, Leave Requests, Employee Records, Secure User Authentication, Real-Time Updates, Report Generation, HR Dashboard

Specific Technology keywords:

HTMLCSS React.Js Web Application Database Management

Functional Components:

1. **User Registration and Authentication**
   Enables employees and administrators to securely register, log in, and manage their access using Django-based authentication.
2. **Employee Profile Management**
   Allows users to view, update, and manage personal and employment details such as name, role, contact information, and more.
3. **Attendance Tracking**
   Lets employees mark attendance and enables administrators to monitor employee attendance records over time.
4. **Admin Dashboard**
   Provides administrative users with tools to view employee lists, manage roles, permissions, and monitor HR-related data.
5. **Leave and Absence Management** *(if applicable)*
   Though not clearly shown in the file names, some HR systems include leave tracking. This may be worth confirming in the UI.
6. **Role-Based Access Control**
   Distinguishes between employee and admin users to enforce permission levels across different system areas.
7. **Secure API Communication**
   React frontend interacts securely with the Django REST API using Axios and custom token-based authentication.

8. **Data Visualization (Dashboard)**
   The frontend uses components to display summaries (e.g., dashboards for employees and admins) — improving usability and clarity.
9. **Reusable React Components**
   The frontend is structured with reusable components for layout (`MainWrapper.jsx`, `Sidebar.jsx`), routes, and views.
10. **Security and Data Protection**
    Incorporates user authentication, protected routes, and proper handling of sensitive data on both client and server sides.

Functionality

Users of the system:

Customers:

Role: Access their profiles, view attendance records, update personal information, and interact with available HR features such as dashboards or notifications.

Administrators:

Role:
Manage employee accounts, monitor attendance and activity, configure system settings, generate reports, and ensure overall system security and data integrity.

Core Functionalities:

- **User Registration and Authentication**

Secure login and account access for both employees and administrators using Django's authentication system.

- **Employee Profile Management**

Employees can view and update personal and professional details such as name, contact info, and role.

- **Attendance Management**

Track daily attendance records, allowing employees to log their presence and administrators to monitor attendance trends.

- **Admin Dashboard**

Administrators have access to a comprehensive dashboard to manage employee data, view summaries, and control system settings.

- **Role-Based Access Control**

Ensures that only authorized users can access certain features based on their role (admin vs. employee).

- **Data Management via REST APIs**

React frontend communicates with Django RESTful APIs for seamless data operations across the application.

- **Report Generation**

Allows generation and viewing of HR-related data summaries, such as employee lists or attendance overviews.

- **Reusable and Modular Frontend Components**
Clean React architecture using reusable layout components for routing, dashboards, and views.

- **Secure Communication and Data Protection**
Implements token-based authentication, secure session handling, and backend protections to safeguard user information.

Steps to start-off the project:

The following steps will be helpful to start off the project –

1. Gain a Solid Understanding of the Required Technologies.
2. Research HR Management Needs and User Workflows.
3. Define User Roles (Employees and Administrators).
4. Design a User-Friendly and Intuitive Interface
5. Maintain Consistent and Professional UI/UX

Requirements

Hardware Requirements:

Number	Description	Alternative (if Available)
1.	Minimum requirement – Processor, x86-64 bit CPU	Any modern multi-core processor (Intel i3/i5, AMD Ryzen)
2.	RAM – 4 GB, Disk Space – 5 GB	Recommended: 8 GB RAM, SSD with 10+ G

Software Requirements:

Numbers	Descriptions	Alternatives (if Available)
1.	**Client on Intranet - User Interface**: React frontend, works in browser, platform-independent	Any web browser (Chrome, Firefox, Edge)
2.	**Development Environment**: VS Code, Django (Backend), React (Frontend), SQLite or PostgreSQL, Windows/Linux/macOS	PyCharm, MySQL, Docker, REST API tools (Postman)

Manpower requirements:

2 to 3 students can complete the Django-React HR System project in **4 to 6 months** if they work full time on it.

Milestones and Timelines

Number	Milestone Name	Milestone Description	Timeline (Week No. from Start)	Remarks
1.	Requirements Specification	Define HR system requirements (employee records, leave, roles, etc.). Document them.	2–3	Consider enhancements like role-based access, dashboards, and email notifications.
2.	Technology Familiarization	Understand Django, React, REST APIs, PostgreSQL/SQLite.	4	Focus on setting up sample apps, running the project locally, and reviewingarchitecture.
3.	Database Creation	Design models for Employees, Attendance, Leaves, etc.	5–6	Ensure model relationships and validations are properly set.
4.	High-Level and Detailed Design	Flowcharts for login, user management, leave requests, etc.	7–8	Cover all user interactions admin, manager, employee and include role permissions.
5.	Front-End Implementation	Create login page, employee list UI, and navigation.	9–10	Start writing basic test cases and frontend component testing.
6.	Front-End and Database Integration	Implement API connections, CRUD operations for employee data.	11–12	Ensure React frontend connects properly with Django REST APIs.
7.	Integration Testing	Test functionality: login, add/edit employee, leave approval, etc.	13–14	Use Django tests, React unit tests, and manual testing for flows.
8.	Final Review	Final checks, documentation, and project demo.	15–16	Make sure deployment guide, user manual, and any gaps in features are addressed.

Guidelines and Reference:

- **Object-Oriented Modeling and Design** with UML – *Michael Blaha, James Rumbaugh*

- **Software Engineering** – *Ian Sommerville (Seventh Edition)*
- **ReactJS Documentation** https://react.dev/reference/react
- **Django Documentation** https://docs.djangoproject.com
- **Node.js API Docs** *(relevant if backend services are extended or hosted with Node)* https://nodejs.org/docs/latest/api
- **Wikipedia** – www.wikipedia.com
- **Database Management Systems** – *Raghu Ramakrishnan or Navathe*