“EFFORT TRACKING SYSTEM”

A

Project Report

submitted

in partial fulfillment

for the award of the Degree of

Bachelor of Technology

in Department of Information Technology

Mentor:
Mr Manoj Raman
Assistant Prof.

Submitted By:
Apeksha Pitaliya
21ESKIT019

Department of Information Technology
Swami Keshvanand Institute of Technology, M & G, Jaipur
Rajasthan Technical University, Kota
Session 2024-2025

Swami Keshvanand Institute of Technology, Management & Gramothan, Jaipur

Department of Information Technology

CERTIFICATE

This is to certify that Ms. Apeksha Pitaliya , a student of B.Tech(Information Technology) 8th semester has submitted her Project Report entitled ”EFFORT TRACKING SYSTEM ” under my guidance.

Mentor	**Coordinator**
Mr Manoj Raman	Priyanka Yadav
Assistant Prof.	Assistant Prof.

DECLARATION

We hereby declare that the report of the project entitled ”EFFORT TRACKING SYSTEM ” is a record of an original work done by us at Swami Keshvanand Institute of Technology, Management and Gramothan, Jaipur under the mentorship of ”Mr Manoj Raman ”(Dept. of Information Technology) and coordination of ”Ms Priyanka Yadav ” (Dept.of Information Technology). This project report has been submitted as the proof of original work for the partial fulfillment of the requirement for the award of the degree of Bachelor of Technology (B.Tech) in the Department of Information Technology.It has not been submitted anywhere else, under any other program to the best of our knowledge and belief.

Team Members **Signature**

APEKSHA PITALIYA

21ESKIT019

Acknowledgement

A project of such a vast coverage cannot be realized without help from numerous sources and people in the organization.We take this opportunity to express our gratitude to all those who have been helping us in making this project successful.

We are highly indebted to our faculty mentor Mr. Manoj Raman .He has been a guide, motivator source of inspiration for us to carry out the necessary proceedings for the project to be completed successfully. We also thank our project coordinator Ms. Priyanka Yadav for her co-operation, encouragement, valuable suggestions and critical remarks that galvanized our efforts in the right direction.

We would also like to convey our sincere thanks to Dr. Anil Chaudhary, HOD, Department of Information Technology, for facilitating, motivating and supporting us during each phase of development of the project.Also, we pay our sincere gratitude to all the Faculty Members of Swami Keshvanand Institute of Technology, Management and Gramothan, Jaipur and all our Colleagues for their co-operation and support.

Last but not least we would like to thank all those who have directly or indirectly helped and cooperated in accomplishing this project.

Team Members:

APEKSHA PITALIYA

21ESKIT019

Table of Content

List of Figures

Chapter 1
Introduction

1.1 Problem Statement and Objective

Problem Statement: Traditional HR management systems often rely on manual work or outdated tools, leading to inefficiencies, data errors, and poor user experiences. These issues can slow down organizational processes and affect employee satisfaction. With increasing digital transformation, there's a growing need for smart, automated HR solutions that are accessible and reliable.

Objective: This project aims to develop a web-based HR Management System using Django and React. It focuses on creating a secure, scalable platform for managing employee data, authentication, and HR workflows. The goal is to enhance operational efficiency and provide a responsive, intuitive interface for both administrators and employees.

1.2 Literature Survey/Market Survey/Investigation and Analysis

Various HR management systems exist in the market, such as BambooHR, Workday, and Zoho People, offering features like employee tracking, payroll, and performance management. However, many are expensive, closed-source, or lack customization options for smaller organizations. Literature and market analysis show a growing trend toward cloud-based, modular HR solutions. This project addresses that

gap by offering a flexible, open-source alternative tailored for small to mid-sized enterprises.

1.3 Introduction to Project

This project is a web-based Effort Tracking System developed using Django for the backend and React for the frontend. It is designed to help organizations efficiently manage HR tasks such as employee data management, user authentication, and role-based access. The system aims to digitize traditional HR processes, reduce manual effort, and ensure data accuracy. With a modern, responsive interface and a scalable architecture, the project caters to the growing need for customizable and cost-effective HR solutions in today's digital workplace.

1.4 Proposed Logic / Algorithm / Business Plan / Solution / Device

The proposed solution is a full-stack HR management system that integrates a Django REST API with a React frontend. The backend handles authentication, user roles, and employee data through secure endpoints, while the frontend offers a dynamic and user-friendly interface for interaction. The system follows a modular approach, allowing easy updates and scalability. Business-wise, it can serve as a customizable open-source product for small to mid-sized companies seeking affordable HR automation. The logic ensures role-based access control, ensuring that HR personnel, managers, and employees interact only with relevant data.

1.5 Scope of the Project

This project focuses on building a scalable and user-friendly HR management system tailored for small to mid-sized organizations. It covers key functionalities such as employee registration, profile management, authentication, and role-based access control. The system is designed to be modular, allowing future integration of features like attendance tracking, payroll, and performance evaluation. It supports both administrative and employee-level access, ensuring a smooth workflow. The project serves as a base for further customization and deployment in real-world organizational settings

Chapter 2
Software Requirement Specification

2.1 Overall Description

The Django-React TRAKING System is a web-based Human Resource Management application that streamlines employee administration, attendance tracking, and task management. It integrates a Django backend with a React frontend to offer a responsive and secure interface. The system supports role-based access for admins and employees, ensuring efficient HR operations and data integrity.

2.1.1 Product Perspective

2.1.1.1 System Interfaces

The Tracking system is designed as a client-server architecture with clear separation between the frontend and backend.The frontend, built with React and Vite, interacts with the backend Django REST API over HTTP using secure JSON-based requests. System interfaces include RESTful endpoints for authentication, employee data, and file uploads, enabling smooth communication between modules. Additionally, the backend connects to a SQLite database for data persistence. The architecture allows future integration with external systems such as payroll APIs or third-party authentication providers, making the system modular and extensible.

2.1.1.2 User Interfaces

The Tracking system provides a responsive and intuitive user interface developed using React and styled for a modern web experience. Users interact through a browser-based frontend that includes components such as login forms, dashboards, employee profile pages, and administrative panels. HR admins have access to additional features like user management and data controls, while employees can view and update their own profiles. The interface supports clear navigation, role-based visibility, and validation feedback, ensuring a smooth user experience across devices. The frontend communicates with the Django backend through RESTful APIs for real-time data interaction.

2.1.1.3 Hardware Interfaces

The Tracking system is a web-based application and does not directly interact with hardware components beyond standard computing devices. It is designed to run on any system with internet access, such as desktops, laptops, tablets, or smartphones. The backend server can be hosted on cloud infrastructure or on-premise machines with basic hardware requirements, including a minimum of 2GB RAM, dual-core processor, and sufficient disk space for storing media and database files. Users access the system through standard web browsers without needing specialized hardware.

2.1.1.4 Software Interfaces

The Tracking system consists of a React frontend and a Django backend that communicate through RESTful APIs using JSON. The backend uses Django REST Framework (DRF) for API development and SQLite for database management (can be upgraded to PostgreSQL or MySQL). The frontend is built with React and Vite, and it uses Axios or Fetch for API calls. Standard web browsers like Chrome, Firefox, and Edge are required to access the system. The backend may also interface with third-party authentication or cloud storage services if extended, making the system modular and extensible.

2.1.1.5 Communications Interfaces

The HR system uses HTTP/HTTPS protocols for communication between the frontend (React) and backend (Django REST API). All data is transmitted in JSON format via RESTful API endpoints. User authentication is typically managed using token-based authentication (e.g., JWT or session tokens). The system supports secure communication through HTTPS to protect sensitive HR data. In a networked deployment, the backend may also connect to email servers or external services for notifications, authentication, or future integrations with third-party platforms.

2.1.1.6 Memory Constraints

The Tracking system is lightweight and optimized for deployment on standard web servers or cloud environments. The backend, built with

Django, can operate efficiently with a minimum of 512MB to 1GB RAM for small-scale deployments. The React frontend is served as static files and consumes minimal memory on the client side. As the system scales with more users and employee data (e.g., media uploads), additional memory may be required, particularly for caching, database operations, and concurrent user handling. For optimal performance, it is recommended to use at least 2GB RAM for production environments.

2.1.1.7 Operations

The Tracking system operates as a web-based application, accessible via standard browsers. Users such as HR administrators, managers, and employees log in using secure credentials to perform role-specific tasks. Operations include adding/editing employee profiles, managing user roles, uploading documents, and viewing dashboards. The backend handles data processing, authentication, and business logic, while the frontend provides real-time interaction and feedback. System uptime depends on the hosting environment, and scheduled backups or monitoring tools can be integrated to ensure reliable and continuous operation.

2.1.1.8 Project Functions

The Tracking system offers a set of core functions to streamline human resource operations. Key functionalities include user authentication (login/logout), employee registration and profile management, and role-based access control for administrators and staff. HR admins

can add, update, or remove employee records, while employees can view and edit their own profiles. The system also supports document/media uploads, dashboard views for summary data, and secure data handling through API communication. All functions are designed to be modular and scalable for future enhancements.

2.1.1.9 User Characteristics

The system will be used by three primary types of users: HR staff, administrative staff, and regular employees. Each user has a unique login based on their email and possesses a defined role that determines access levels and available functionalities within the system.

HR Users have access to employee data, leave management, and administrative dashboards.

Administrative Staff have access to backend tools and system configurations via the admin panel.

Regular Employees can manage their personal profiles, view their data, and submit leave requests.

All users share common attributes including:

1. Email (used for login)
2. Username (display name)
3. Password
4. Date Joined
5. Is Active (account status)
6. Is Staff (admin role flag)
7. Is HR (HR role flag)

These characteristics are essential for authentication, authorization, and role-based access control within the system.

2.1.1.10 Constraints

1. **Constraints for Tracking System Authentication Required**: Only authenticated users can access or submit tracking data.
2. **Role-Based Access Control**: Only HR staff can view or modify tracking records of other employees. Immutable Historical Records: Once submitted, tracking logs (e.g., attendance, leave history) cannot be altered by non-HR users.
3. **Data Consistency**: Tracking entries must be timestamped accurately and stored in a secure, centralized database.
4. **Limited Time Editing**: Employees can only edit or submit tracking entries within a defined time window (e.g., same day or 24 hours).

2.1.1.11 Assumption and Dependencies

1. **User Availability**: It is assumed that users (HR, staff, and employees) will have access to stable internet connections and modern web browsers to use the system effectively.
2. **Authentication System**: The system assumes that Django's built-in authentication and custom user model extensions will handle login, session management, and user role assignments reliably.
3. **Third-Party Dependencies**: The system relies on Django for

backend functionality and React for the frontend. Additional dependencies include the Django REST Framework for API services and a relational database (e.g., SQLite/PostgreSQL).

4. **Device Compatibility**: It is assumed that users will access the system primarily from desktop or laptop devices, though mobile compatibility may be partially supported.

5. **Data Integrity**: The system assumes that all input data from users (e.g., leave requests, profile updates) will be validated through frontend and backend checks before being stored.

Chapter 3
System Design Specification

3.1 System Architecture

1. **Layer 1: Client Layer (Frontend)** This is the user interface built in React.js. When a user clicks something (like "Submit"), it sends a request to the backend.
2. **Layer 2: Application Layer (Backend)** The backend (Django) receives the request, does the work (like fetching data from the database), and sends a response back to the frontend.
3. **Layer 3: Data Layer (Database)** The database stores everything – employee info, login credentials, etc. Django queries this database to get or update data as needed.
4. **Deployment (How your project goes live)** Frontend can be hosted on Netlify (it gives you a public URL). Backend can be on localhost for testing.

3.2 Module Decomposition Description

1. **Authentication Module** Purpose: Manages login/logout functionality.

   Features:

   - User registration (optional)

- Secure login with email and password
- Role-based access (Admin/Employee)
- Tech: Django auth system / JWT

2. **User Management Module** Purpose: Allows HR/Admin to manage employee data.

   Features:

   - Add/edit/delete user profiles
   - Assign roles (e.g., HR or Employee)
   - View user lists and profiles

3. **Attendance Tracking Module** Purpose: Tracks when employees check in/out.

   Features:

   - Mark attendance
   - View attendance reports
   - Filter by date, user, or status

4. **Task Management Module** Purpose: Assign and monitor employee tasks.

   Features:

   - Create tasks
   - Assign tasks to employees
   - Track status (e.g., Pending, In Progress, Completed)

5. **Dashboard Module** Purpose: Provides a summary view for HR/Admin.

   Features:

   - Total employees
   - Daily attendance stats
   - Quick links to important actions

6. **API Module** Purpose: Handles data exchange between frontend and backend.

   Features:

   - REST APIs to fetch, update, or delete data
   - Secured endpoints

7. **Frontend Module** Purpose: Handles the user interface using React.js.

   Features:

   - Dynamic forms and buttons
   - Responsive UI
   - Data display from backend APIs

3.3 High Level Design Diagrams

3.3.1 Use Case Diagram

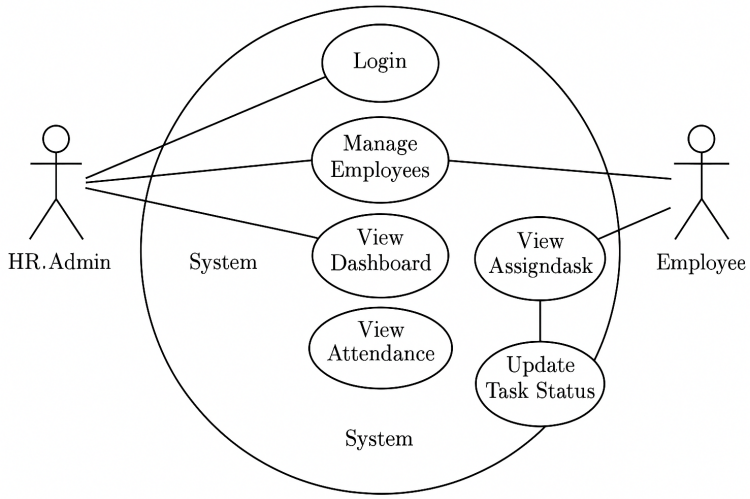

Figure 3.1: USE CASE DIAGRAM

3.3.2 Activity Diagram

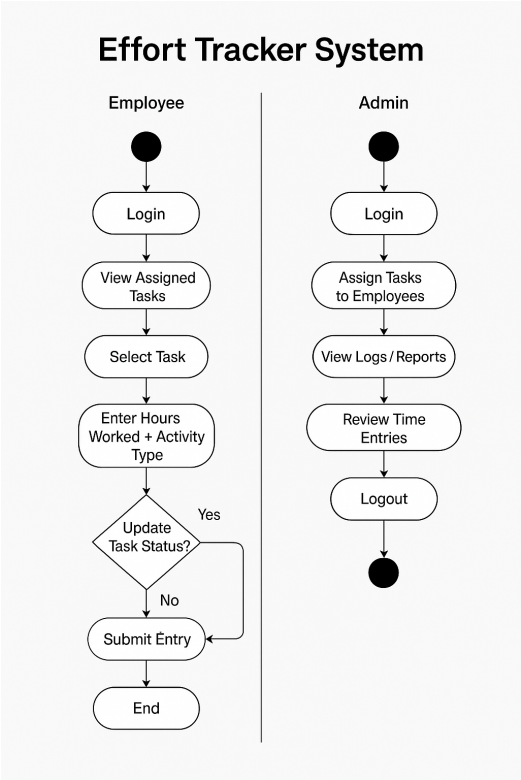

Figure 3.2: ACTIVITY DIAGRAM

3.3.3 Data-Flow Diagram

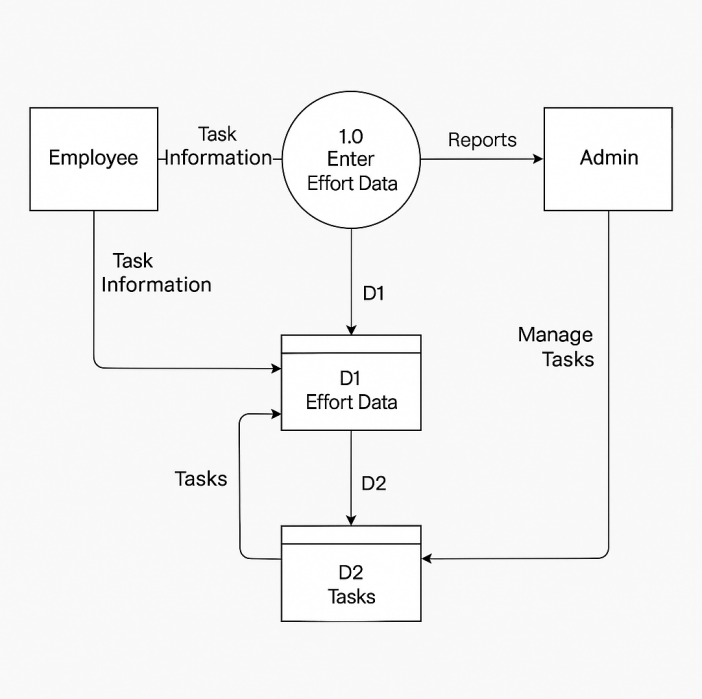

Figure 3.3: DATA FLOW DIAGRAM

3.3.4 Class Diagram

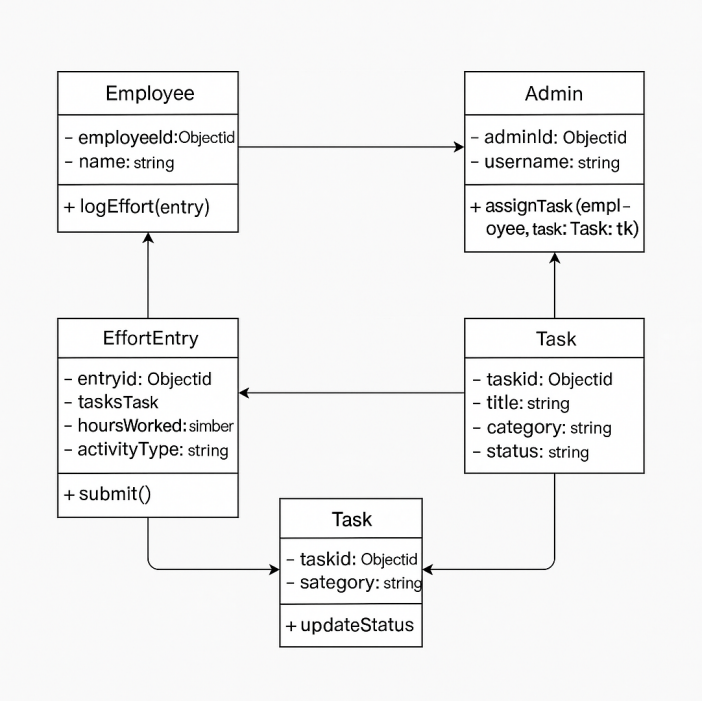

Figure 3.4: CLASS DIAGRAM

Chapter 4
Methodology and Team

4.1 Introduction to Waterfall Framework

The Waterfall framework is one of the earliest and most structured software development life cycle (SDLC) models. It emphasizes a sequential, phase-by-phase development process where each stage must be completed before the next begins. This approach is ideal for projects with well-defined goals, clear requirements, and minimal expected changes making it a suitable choice for the development of the Effort Tracker System.

In the context of this project, the Waterfall model helped guide the team through each phase: requirement analysis, system design, implementation, testing, and deployment. Since the system's core functions like employee effort tracking, task updates, and administrative controls were defined from the beginning, the model allowed for smooth execution without needing frequent adjustments. It ensured a disciplined workflow, reduced risks during development, and delivered a robust, reliable solution for tracking organizational productivity.

By following the Waterfall model, the development team was able to maintain strong documentation and traceability throughout the project lifecycle. Each phase produced clear deliverables, which helped in monitoring progress and simplifying handovers between design, de-

velopment, and testing teams. This structure also supported easier debugging and validation, as issues could be traced back to specific stages. Overall, the Waterfall methodology contributed to a systematic and disciplined approach, ensuring that the Effort Tracker System met its functional requirements with minimal rework.

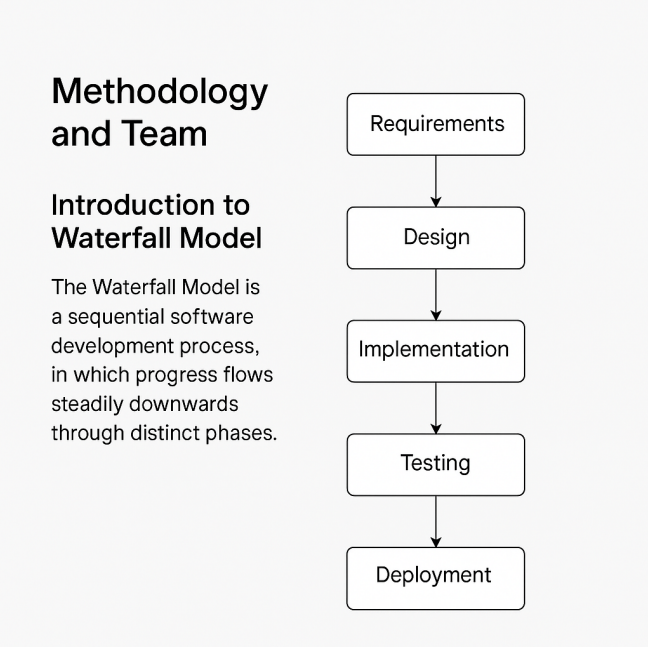

Figure 4.1: WATERFALL MODEL

The sequential phases in Waterfall model are-

1. **Requirement Analysis:** This is the first and most critical phase where all the system's needs are gathered from stakeholders. For the Effort Tracker System, this involved identifying features like user authentication, task assignment, effort logging, admin controls, and report generation. The outcome of this phase is a detailed Software Requirements Specification (SRS) document that guides the entire project.

2. **System Design:** In this phase, the system's architecture and tech-

nical specifications are planned. For your project, this included designing the database schema (e.g., users, efforts, tasks tables), defining API endpoints, and structuring frontend components. High-level and low-level design documents are produced, laying the foundation for development.

3. **Implementation:** With a clear design in place, the actual development begins. In the Effort Tracker System, this meant building the backend with Django (handling APIs, authentication, and logic) and the frontend with React (handling user interactions and UI). Each module (like employee management, time tracking) was coded independently according to the designs.

4. **Integration and Testing:** Once all modules are developed, they are integrated and tested as a complete system. This step includes functional testing, unit testing, integration testing, and bug fixing. For the Effort Tracker System, it involved validating login flows, effort logs, and admin features to ensure everything worked as expected.

5. **Deployment of system:** After successful testing, the system is deployed to a live server or production environment. The Effort Tracker System is made accessible to users through a web browser. Configuration settings, domain setup, and server hosting (local or cloud) are handled during this phase.

6. **Maintenance:** Once deployed, the system may require updates, optimizations, or bug fixes. Maintenance ensures the system re-

mains functional and adapts to any new requirements. In the context of your project, this could involve adding new features like notifications or fixing issues reported by users

Waterfall Model Pros & Cons

Advantages

1. **Easy to Use and Manage** – The linear approach is straightforward and easy to follow.
2. **Well-Structured Phases** – Each phase has specific goals, which helps maintain clarity and focus.
3. **Good Documentation** – Extensive documentation supports future maintenance and team understanding.
4. **Stable Requirements** – Best suited for projects like the Effort Tracker System, where requirements are known upfront.
5. **Progress is Easy to Track** – Milestones are clearly defined at the end of each phase.

Disadvantages

1. **Rigid and Inflexible** – Not ideal if changes are needed after the development has started.
2. **Late Testing** – Errors may not be caught until the final stages of the project.
3. **Not Suitable for Complex or Evolving Projects** – Difficult to handle changing requirements mid-development.

4. **No Early Prototype** – Working software is available only at the end, delaying user feedback.

5. **High Risk for Misinterpretation** – Misunderstood requirements in the early phase can cause major issues later.

4.2 Team Members, Roles & Responsibilities

APEKSHA PITALIYA :

1. **Frontend Developer**

   Designed and developed the user interface using React Ensured responsive and user-friendly layout Integrated frontend with back-end APIs

2. **Backend Developer**

   Developed the backend using Django framework Implemented database models, REST APIs, and user authentication Ensured secure and efficient data handling

3. **Tester / QA Engineer**

   Conducted functional and system testing for all modules Reported bugs and worked with developers to resolve issues Ensured the final product met quality standards

4. **Documentation Lead**

   Prepared SRS, user manuals, and final project documentation Maintained clear and consistent records of each development phase Supported the team with structured project reporting

Chapter 5
Centering System Testing

System Testing was performed to evaluate the complete and integrated Effort Tracker System. This phase ensured that all modules such as employee registration, task assignment, effort logging, and admin reporting worked together seamlessly. The objective was to validate the end-to-end flow of the application and confirm that both frontend and backend components communicated correctly through the defined APIs. The testing was done in an environment similar to production to ensure real-world behavior.

5.1 Functionality Testing

1. Functionality Testing is a type of black-box testing that validates the software system against the functional requirements specified in the SRS document. For the Effort Tracker System, this phase involved testing all features to ensure that the system behaves as expected when users perform specific actions.

2. Each module of the application was tested for its core functionality. For instance, the Login Module was tested to verify authentication and role-based access (e.g., employee vs admin). The Effort Logging Module was validated to ensure users could accurately input daily work efforts and save them in the database. The Task Assignment Module was tested to confirm that admins

could assign tasks to users and that notifications or updates were reflected correctly in the frontend. Additionally, features like data validation, form submission, error handling, and navigation between pages were rigorously checked.

3. Test cases were designed to include both positive scenarios (valid inputs and expected outcomes) and negative scenarios (invalid data, missing fields, or unauthorized actions). This helped ensure robustness, data integrity, and proper enforcement of business logic. Functionality testing confirmed that the Effort Tracker System was capable of handling real-world operations with accuracy and reliability.

5.2 Performance Testing

Performance testing was carried out to evaluate the responsiveness and stability of the Effort Tracker System under normal and high user loads. The objective was to ensure the system could handle multiple users performing tasks like logging efforts, accessing dashboards, and generating reports simultaneously. Key metrics such as response time, server load, and database query performance were monitored. Simulated traffic was used to replicate real-world scenarios. The system maintained consistent performance, with no major lags or crashes. All critical operations responded within acceptable time limits. This confirmed the system's readiness for multi-user environments and reliable day-to-day performance.

5.3 Usability Testing

Usability testing was conducted to evaluate how user-friendly and intuitive the Effort Tracker System is for end users. Real users, including students and faculty, were asked to perform common tasks such as logging in, submitting daily efforts, and navigating between modules. The system's interface was assessed for ease of navigation, clarity of instructions, and overall user satisfaction. Feedback highlighted that the interface was simple, responsive, and easy to use, even for first-time users. Minor UI adjustments were made based on suggestions. The results confirmed that the system provides a smooth and efficient user experience.

Chapter 6
Test Execution Summary

Test Execution Summary

The Django-React HR Management System was subjected to a series of manual tests to ensure the core functionalities operated as expected. The application was tested across different scenarios and roles, focusing on user experience, data integrity, and access control. Below is a summary of the key test outcomes:

Test Case	Description	Expected Result	Status
HR Login	Login with valid HR credentials	Dashboard should load successfully	Pass
Invalid Login	Attempt login with incorrect credentials	Display error message	Pass
Add New Employee	HR adds a new employee with valid details	Employee appears in the list	Pass
Edit Employee Details	Modify existing employee information	Changes saved and reflected	Pass
Add Attendance	HR selects a date and marks attendance for employees	Attendance saved correctly	Pass
Unauthorized Access	Access protected routes without logging in	Redirected to login page	Pass
Form Validation	Leave required fields empty in forms	Show validation error	Pass
Frontend Responsiveness (Basic)	Access the system on various screen sizes	Layout adjusts appropriately	Pass

Table 6.1: Summary of manual tests conducted on the HR Management System

Chapter 7

Project Screen Shots

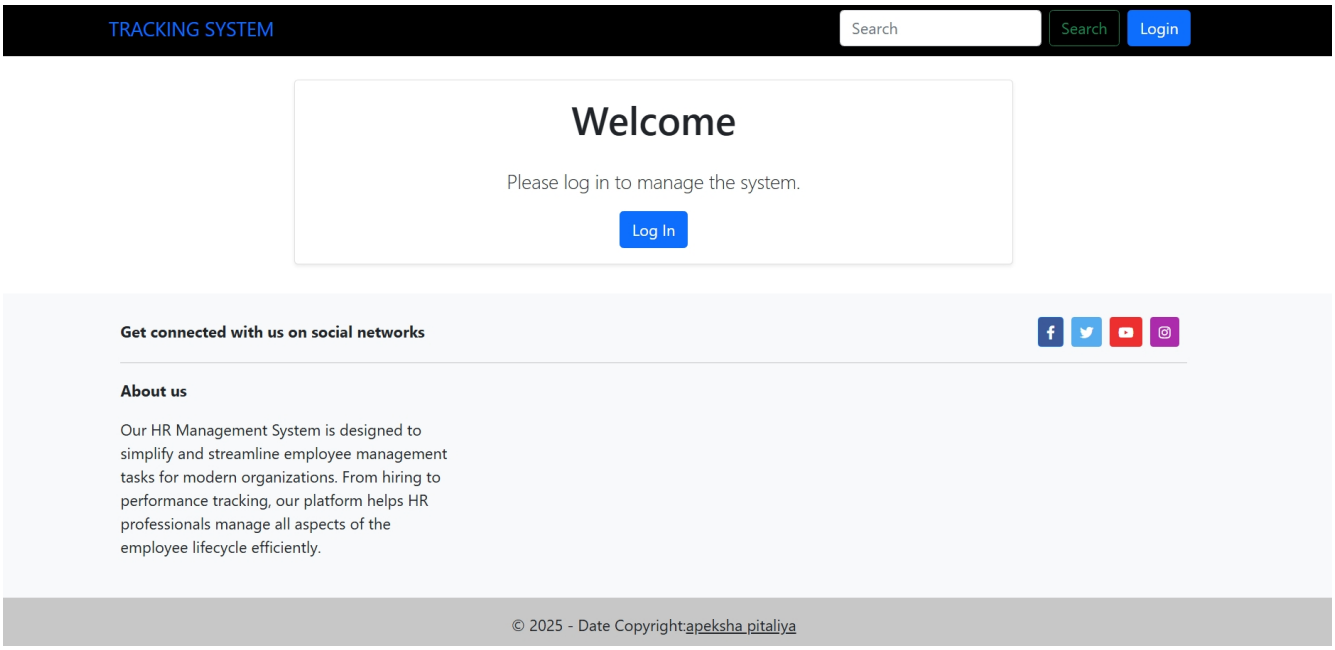

Figure 7.1: WELCOME PAGE

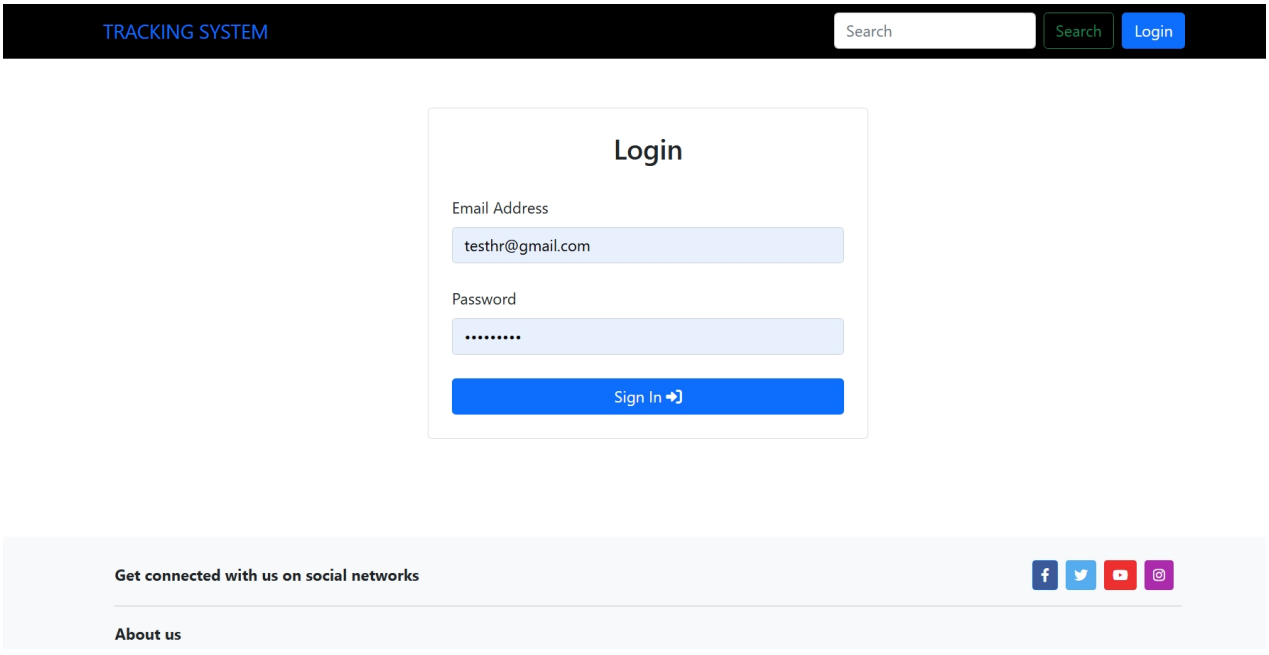

Figure 7.2: LOGIN INTERFACE

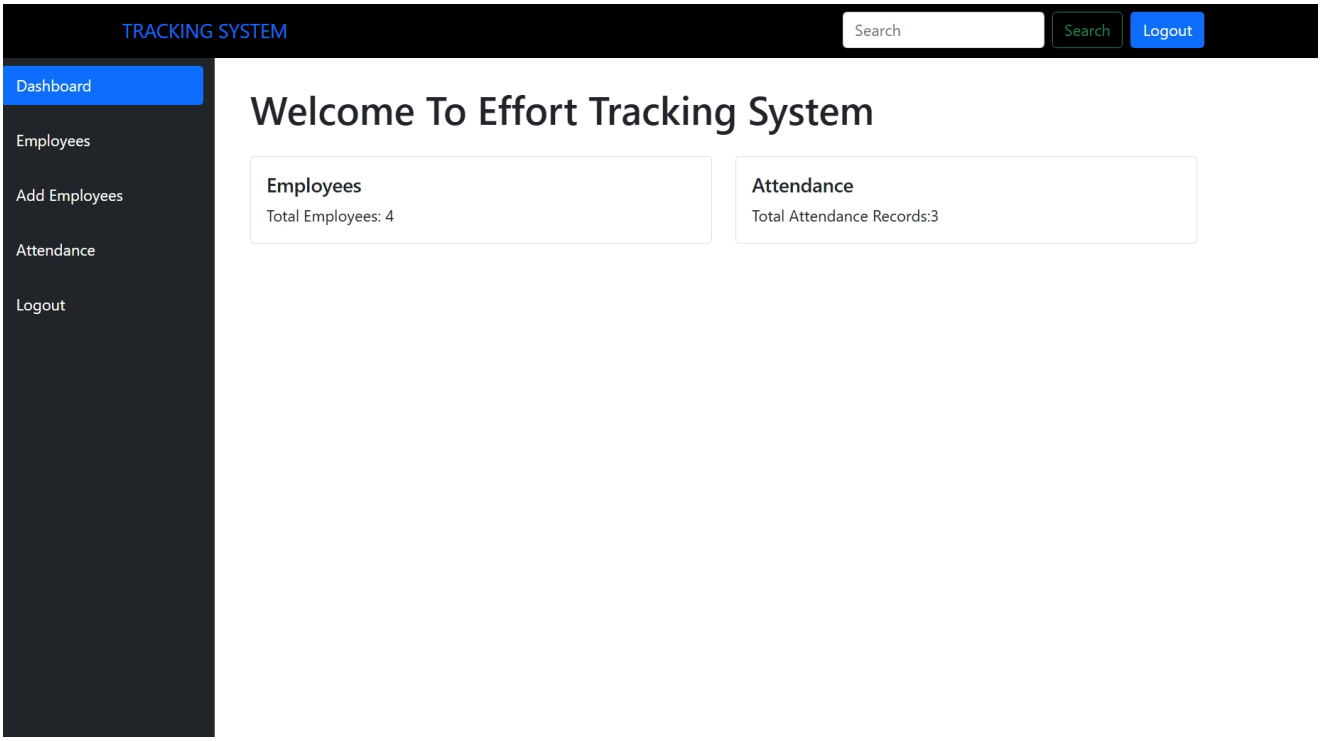

Figure 7.3: DASHBOARD

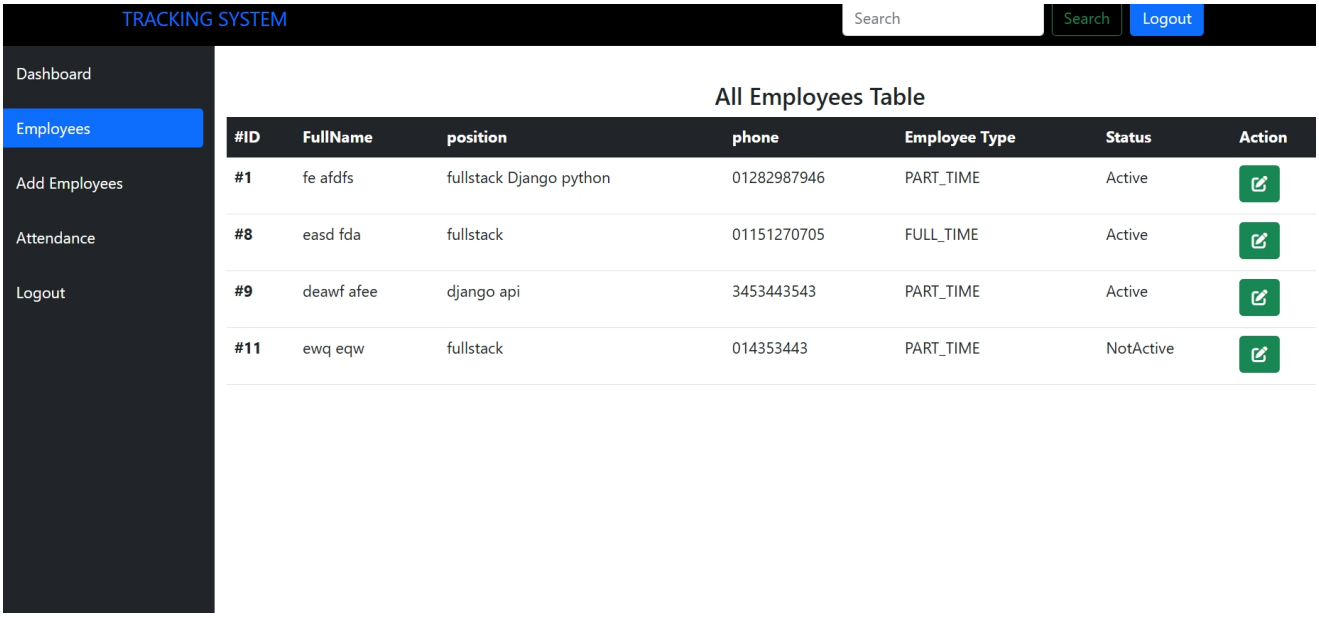

Figure 7.4: EMPLOYEE TABLE

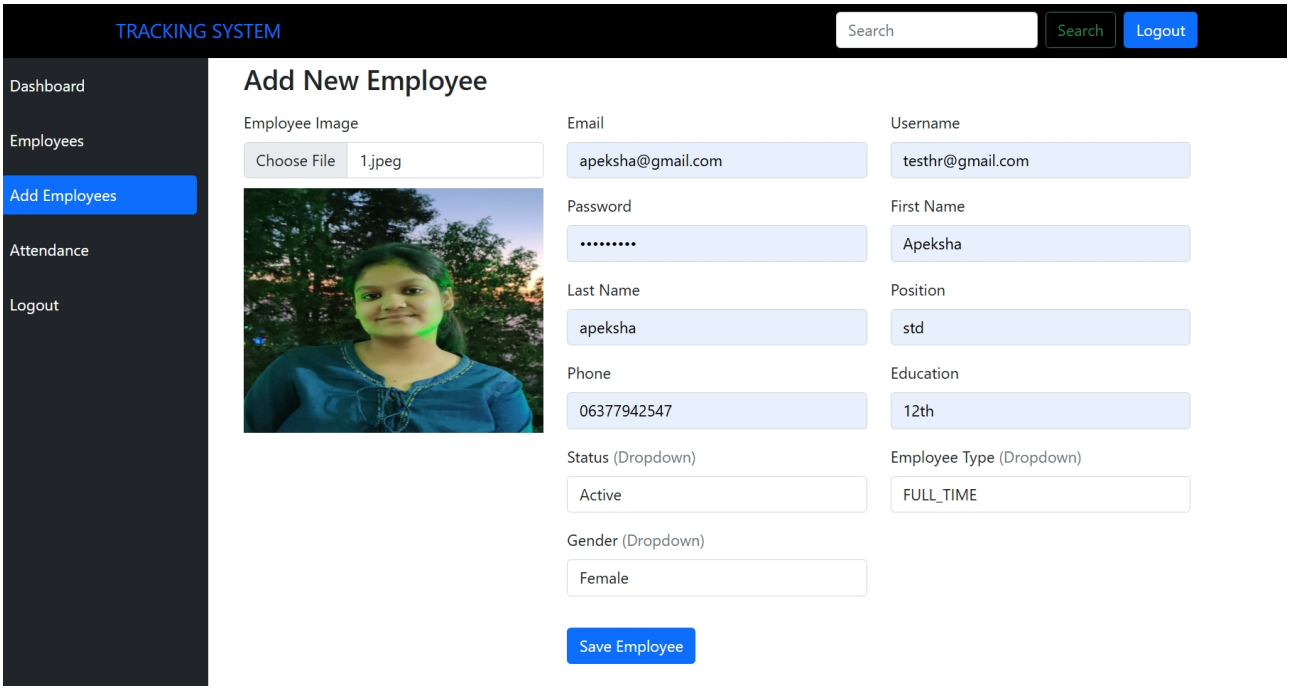

Figure 7.5: ADD NEW EMPLOYEE

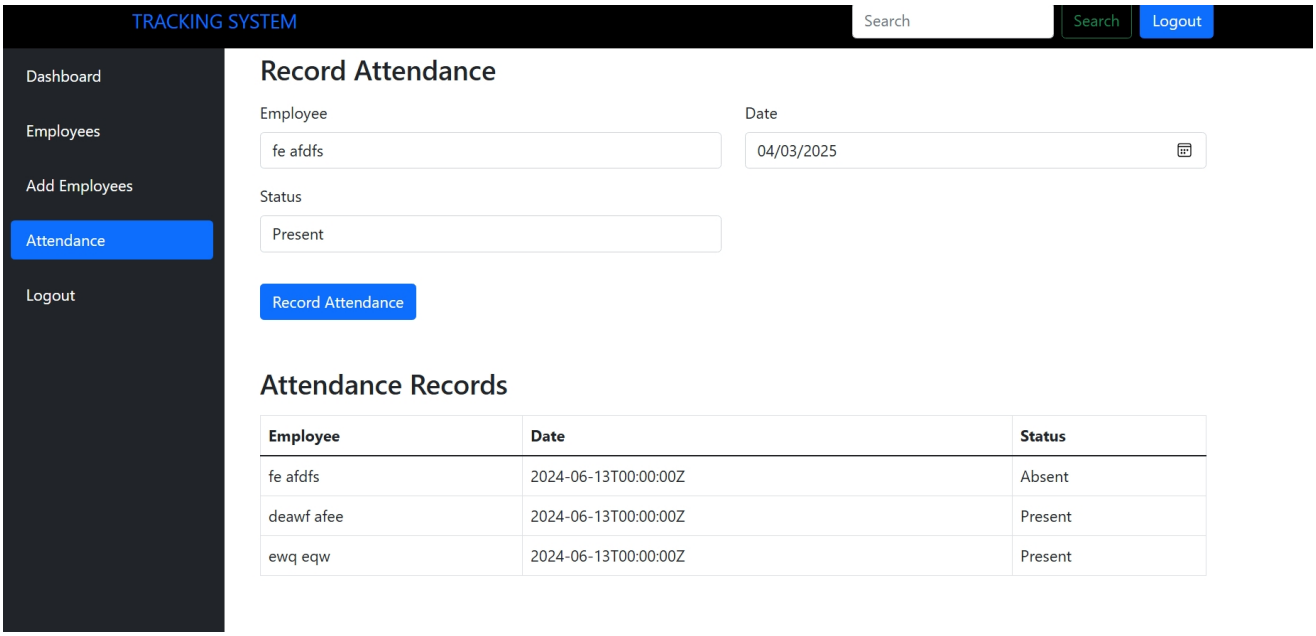

Figure 7.6: RECORD ATTENDANCE

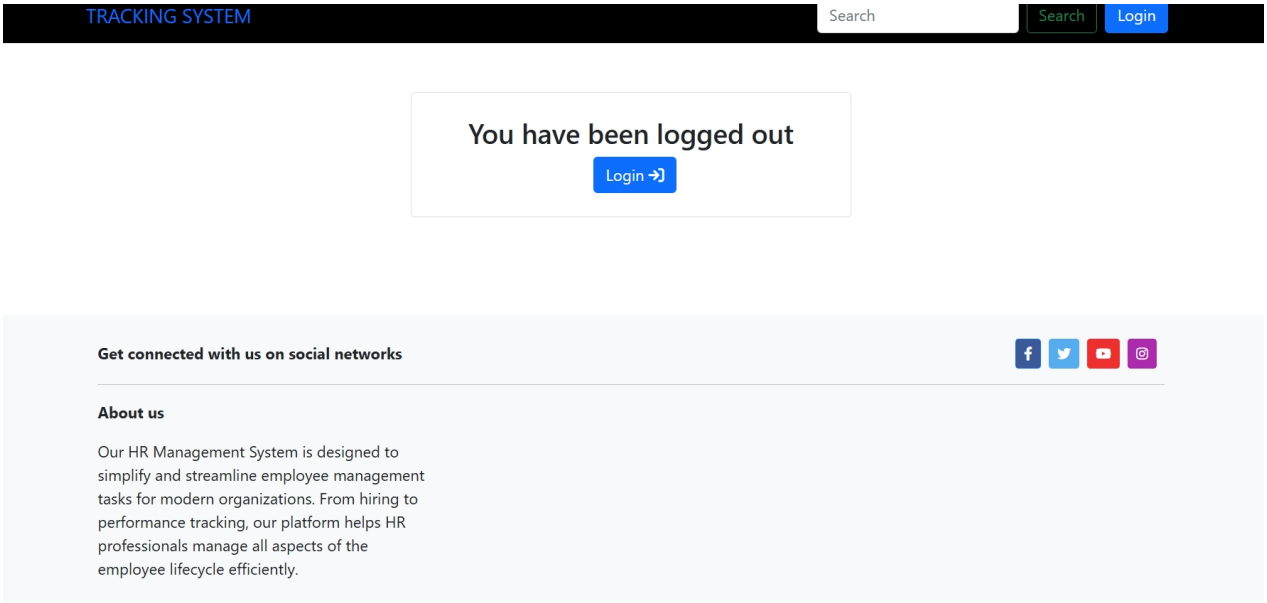

Figure 7.7: LOGOUT

Chapter 8
Project Summary and Conclusions

8.1 Conclusion

Project Summary:

The Django-React HR Management System was subjected to a series of manual tests to ensure the core functionalities operated as expected. The application was tested across different scenarios and roles, focusing on user experience, data integrity, and access control. Below is a summary of the key test outcomes:

Test Case Description Expected Result Status HR Login Login with valid HR credentials Dashboard should load successfully Pass Invalid Login Attempt login with incorrect credentials Display error message Pass Add New Employee HR adds a new employee with valid details Employee appears in the list Pass Edit Employee Details Modify existing employee information Changes saved and reflected Pass Add Attendance HR selects a date and marks attendance for employees Attendance saved correctly Pass Unauthorized Access Access protected routes without logging in Redirected to login page Pass Form Validation Leave required fields empty in forms Show validation error Pass Frontend Responsiveness (Basic) Access the system on various screen sizes Layout adjusts appropriately Pass

Chapter 9
Future Scope

- **Leave Management**: Automate leave requests, approvals, and balance tracking
- **Payroll Integration**: Calculate and manage employee salaries, bonuses, and deductions
- **Role-Based Access Control**: Separate permissions for HR, employees, and admins.
- **Notifications System**: Alerts for birthdays, holidays, attendance issues, etc.

References

[1] *Django Documentation – https://docs.djangoproject.com*

[2] *W3Schools – https://www.w3schools.com/*

[3] *RESTful API Design -https://restfulapi.net*