An Internship Report on

ERP Management System
&
Webhub Technologies

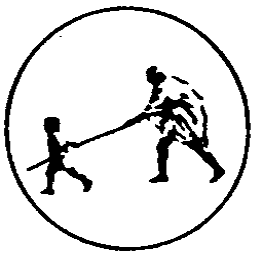
BY

Mr. Sairaj Shrimanwar
Mr. Soham Kotgire
Mr. Prem Myana

Under the Guidance

of

Ms. Savita. S. Wagre



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Mahatma Gandhi Mission's College of Engineering, Nanded (M.S.)

Academic Year 2024-25

An Internship Report on

ERP Management System
&
Webhub Technologies

Submitted to

DR. BABASAHEB AMBEDKAR TECHNOLOGICAL UNIVERSITY, LONERE

in fulfillment of the requirement for the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE & ENGINEERING

By

Mr. Sairaj Shrimanwar
Mr. Soham Kotgire
Mr. Prem Myana

Under the Guidance
of

Ms. Savita S. Wagre

(Department of Computer Science and Engineering)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
MAHATMA GANDHI MISSION'S COLLEGE OF ENGINEERING
NANDED (M.S.)
Academic Year 2024-25

Certificate

This is to certify that the project entitled

"ERP Management System"
&
"Webhub Technologies"

being submitted by ***Mr. Sairaj Shrimanwar, Mr. Soham Kotgire & Mr. Prem Myana*** *to the Dr. Babasaheb Ambedkar Technological University, Lonere , for the award of the degree of Bachelor of Technology in Computer Science and Engineering, is a record of bonafide work carried out by them under my supervision and guidance. The matter contained in this report has not been submitted to any other university or institute for the award of any degree.*

Ms. Savita S. Wagre
Project Guide

Dr. A. M. Rajurkar
H.O.D
Computer Science & Engineering

Dr. G. S. Lathkar
Director
MGM's College of Engg., Nanded

AKNOWLEDGEMENT

We would like to express my deepest gratitude to our project guide, **Ms. Savita S. Wagre** for her invaluable support, guidance, and encouragement throughout the duration of this project. Her profound knowledge and expertise have been instrumental in the successful completion of this work. Her patience and willingness to assist us at every step have greatly enriched our learning experience. Her constructive feedback and insightful suggestions have not only helped us to overcome challenges but also motivated us to strive for excellence.

We gladly take this opportunity to thank **Dr. A. M. Rajurkar** (Head of Computer Science Engineering, MGM's College of Engineering, Nanded).

We are heartily thankful to **Dr. G. S. Lathkar** (Director, MGM's College of Engineering, Nanded) for providing facility during progress of project also for her kindly help, guidance and inspiration. Last but not least we also thankful to all those who help directly or indirectly to develop this project and complete it successfully.

With Deep Reverence,

Sairaj Shrimanwar(203)

Soham Kotgire(229)

Prem Myana(236)

[B.Tech CSE-B]

ABSTRACT

The ERP (Enterprise Resource Planning) Management System is a web-based solution designed to streamline and automate the core administrative, financial, and operational tasks of IT companies and training institutes. Built using the MERN (MongoDB, Express.js, React.js, Node.js) stack, the system provides a centralized platform where different stakeholders—Admins, Project Managers, Developers, and Clients—can interact through role-specific dashboards. The application facilitates real-time project tracking, employee task management, client interaction, financial monitoring, and feedback collection, all within a secure and scalable environment.

The system architecture emphasizes modularity and role-based access control to ensure that each user accesses only the functionalities relevant to their responsibilities. Admins can manage branches, users, tasks, and expenses; Project Managers handle developer assignments, project timelines, and client sessions; Developers update task status, track sessions, and view salaries; Clients can check project progress, submit requirements, book sessions, and provide feedback. Additional features such as attendance tracking, salary automation, referral system, and performance analytics make the system comprehensive and future-ready.

Overall, the ERP Management System not only replaces the need for multiple disconnected tools and manual record-keeping but also empowers organizations to operate more transparently and efficiently. It enhances productivity, reduces communication gaps, and offers a digital backbone that supports business scalability, making it an ideal solution for modern IT-based businesses seeking streamlined management.

TABLE OF CONTENTS

TOPICS	PAGE NO.
INTERNSHIP OFFER LETTER	**I**
INTERNSHIP EXPERIENCE LETTER	**IV**
ACKNOWLEDGEMENT	**VII**
ABSTRACT	**VIII**
TABLE OF CONTENTS	**IX**
LIST OF FIGURES	**XI**
Chapter 1. INTRODUCTION	**1**
1.1 Project Overview	1
1.2 Innovation and Differentiation	3
1.3 Project Objectives	4
1.4 Target Audience	6
1.5 Vision and Mission	6
1.6 Scope of Project	8
1.7 Structure of the Report	9
Chapter 2. LITERATURE REVIEW	**10**
2.1 Introduction	10
2.2 Existing ERP Tools and Technologies	10
2.2.1 Proprietary ERP Systems	10
2.2.2 Open-Source ERP Systems	12
2.2.3 Technology Stack Used in ERP Development	12
2.3 Existing Approaches in ERP System Design	13
2.4 Limitations of Existing ERP Systems	14
Chapter 3. TECHNOLOGIES USED	**15**
3.1 Introduction to Technologies Used	15
3.2 Database Technology	17
3.2.1 Why Choose MongoDB for ERP?	17
3.2.2 Data Storage in MongoDB (ERP Modules)	17
3.2.3 Example Document Structures	18
3.2.4 MongoDB Query Examples	19
3.2.5 Integration with Mongoose (ODM Layer)	19
3.3 Frontend Technologies	20
3.3.1 Why React.js for ERP?	21
3.4 Backend Technologies	22
3.4.1 Why Node.js and Express.js?	22
3.5 Postman – API Testing Tool	24

3.6 Security Technologies 26
3.7 ERP Management System Architecture 27
Chapter 4. METHODOLOGIES USED 31
4.1 Features 31
4.2 Admin Module 31
4.3 Project Manager Module 33
4.4 Developer (Employee) Module 34
4.5 Client Module 35
4.6 Development Process 37
4.6.1 Agile Methodology Adoption 37
4.7 Sprints and User Stories 39
4.8 Challenges Faced 41
4.9 Continuous Integration and Deployment (CI/CD) 43
4.10 Advantages 46
Chapter 5. IMPLEMENTATION DETAILS 48
5.1 Admin Module 48
5.2 Project Manager Module 49
5.3 Developer (Employee) Module 50
5.4 Client Module 52
5.5 MongoDB 53
CONCLUSION 55
REFERENCES 56

LIST OF FIGURES

Figure No.	Name of Figures	Page No.
3.7	Architecture	27
5.1	Admin Home Page	48
5.2	Project Manager Home Page	50
5.3	Employee/Developer Home Page	51
5.4	Customer/Client Home Page	52
5.5	MongoDB Database	53

Chapter 1

INTRODUCTION

The ERP (Enterprise Resource Planning) Management System is a robust web application designed to automate the essential business activities of IT companies and training institutes. With a modular and scalable design, this ERP system brings together various departments and operational units into a single digital platform.

1.1 Project Overview

Its main aim is to streamline workflows, enhance communication among stakeholders, and centralize control over key business functions such as employee management, project tracking, task assignment, attendance monitoring, financial analytics, feedback collection, and client interaction. While ERP systems have traditionally catered to large organizations, today's startups and SMEs require customized ERP solutions to tackle the unique challenges they face as they grow. This project addresses the inefficiencies stemming from manual processes and disjointed software tools by offering an all-in-one platform built on the MERN stack: MongoDB, Express.js, React.js, and Node.js. The system features login portals for four key roles: Admin, Project Manager, Developer (Employee), and Client. Each role comes with specific privileges and a tailored dashboard for their tasks.

For example, the admin oversees the entire organization; the Project Manager tracks developers, assigns tasks, and monitors client sessions; the Developer can check project statuses and update task progress; and the Client can review updates, submit requirements, and give feedback. The application is fully responsive and compatible with all modern web browsers. It eliminates the need for separate tools for HR, project management, and client servicing, making it a strong alternative to costly commercial ERP suites. The application is fully responsive and works across all modern web browsers. It eliminates the need for separate tools for HR, project management, and client servicing, making it a powerful alternative to expensive commercial ERP suites. To create a successful ERP system, we kicked things off with in-depth requirement gathering sessions alongside mock stakeholders. We mapped out data flows using system design diagrams and embraced agile development principles. This approach, which emphasizes continuous iteration and feedback, ensured that the final product

effectively addresses both the functional and non-functional needs of the business.

What really makes this ERP stand out is its clear delineation of user roles and the specific tasks tied to each role. For instance:

- **Admin Role:** This role is all about managing the master data for the entire application. Admins oversee branches, manage all users, add or remove managers and developers, track expenses and profits, generate payroll, and keep attendance logs in check.
- **Project Manager Role:** Acting as the operational lead, the project manager is responsible for delivering projects. They can assign and monitor tasks for developers, interact with students or clients, handle inquiries, calculate salaries, and generate progress and financial reports.
- **Developer/Employee Role:** Developers have a user-friendly dashboard that allows them to see project assignments, track tasks, update their progress, review their salary and attendance, and provide feedback to managers.
- **Client Role:** Clients have a public-facing view where they can check project status updates, book learning sessions, raise questions, submit requirements, give feedback, and refer others.

The ERP Management System not only enhances communication but also automates administrative tasks like payroll generation, expense tracking, and attendance logging through digital records. This automation significantly cuts down on the time spent on manual processes while boosting accuracy and transparency.

With a sleek design interface built using React and Tailwind CSS, the application ensures clarity across various devices and screen sizes. Every component is crafted for optimal usability, making it easy for non-technical stakeholders, especially clients, to navigate and understand their dashboards. Additionally, the integration of MongoDB offers a highly flexible and scalable document database, enhancing the overall functionality of the system.

This system allows for the storage of a variety of data types, such as dynamic feedback forms, project tasks, session details, and financial ledgers. Thanks to its NoSQL architecture, MongoDB is perfect for handling diverse and ever-changing datasets, which are typical in real-world IT organizations.

On the backend, Node.js paired with Express.js provides quick, real-time API responses, role validation, JWT-based session management, and modular routing.

These technical features not only ensure the system is secure and robust but also allow for future scalability. Additionally, error handling and logging mechanisms have been put in place to assist with debugging and maintenance.

To sum it up, the ERP Management System is a forward-thinking solution aimed at digitizing operations, reducing inefficiencies, and centralizing information to boost productivity and decision-making. Its design draws inspiration from actual organizational structures, and its readiness for deployment makes it an asset for automating business processes in the IT service sector.

1.2 Innovation and Differentiation

The ERP Management System introduces several innovations and differentiators that make it suitable for modern IT firms:

- **Modular Role-Based Access:** Every user has a dedicated dashboard and functionalities tailored to their role. This improves usability and ensures data security through RBAC (Role-Based Access Control).
- **Real-Time Updates and Status Tracking:** Project Managers and Clients can see project progress live through status updates and logs submitted by Developers. This creates a transparent and collaborative environment.
- **Integrated Financial Tracking:** Admins and Project Managers can view and manage expenses, calculate profits and losses, and monitor payroll—all in one dashboard.
- **Feedback Loop and Referral System:** Clients can give structured feedback after each phase of project delivery, and a referral module helps generate new leads.
- **Task Assignment and Completion Monitoring:** Tasks can be assigned and monitored dynamically by the Project Manager. Status updates reflect directly on the dashboard, helping identify bottlenecks.

These features differentiate the system from basic task management tools or HR software, positioning it as a fully-fledged ERP solution tailored for IT operations.

1.3 Project Objectives

The primary objective of this ERP Management System project is to develop a comprehensive, user-friendly, and scalable platform that streamlines and automates the key operational activities of IT companies and educational institutes. This system is intended to facilitate efficient management of administrative, managerial, employee,

and client processes under a single integrated framework.

1. Automation of Core Business Processes

One of the fundamental goals of this ERP system is to automate repetitive and manual business processes, such as attendance tracking, salary processing, project and task management, expense monitoring, and client interactions. Automation reduces human error, saves time, and increases operational efficiency by enabling real-time data updates and workflows.

By eliminating paper-based and siloed systems, the ERP enhances productivity and provides managers and administrators with instant access to critical information, helping them make faster and more informed decisions.

2. Centralized Data Management and Improved Data Accuracy

The ERP system aims to centralize data storage to create a single source of truth accessible to authorized personnel across various departments. Centralized data management minimizes discrepancies and redundancies commonly associated with multiple disconnected systems. It ensures consistency and integrity of data related to employees, projects, clients, finances, and feedback.

Accurate data enables precise reporting, analysis, and forecasting, which are crucial for strategic planning and business growth.

3. Scalability and Flexibility to Adapt to Growing Business Needs

The system is designed with scalability in mind, allowing the addition of new branches, departments, users, and modules as the company grows. The flexible architecture supports customization and easy integration with third-party tools and services to meet evolving business requirements.

This adaptability helps organizations avoid the costs and disruptions associated with switching to new software as their needs change.

4. Enhanced User Experience Across Roles

Different modules are tailored to the unique needs of various user roles such as Admin, Project Manager, Employee/Developer, and Client. The ERP system aims to provide intuitive, role-based dashboards and interfaces to improve user experience and encourage adoption.

For example, project managers can easily track developer progress, budgets, and deadlines, while clients have direct access to project status and communication channels, fostering transparency and collaboration.

5. Real-Time Monitoring and Reporting

A key objective is to provide real-time monitoring of critical metrics such as project status, attendance, salary disbursement, expenses, and client feedback. The system includes robust reporting tools that generate comprehensive reports and visualizations to help management understand performance trends and identify bottlenecks or issues early.

Timely insights support proactive management and continuous improvement.

6. Security and Access Control

The ERP system prioritizes data security and privacy by implementing strict access controls and authentication mechanisms. Role-based access ensures that users can only view or modify information relevant to their responsibilities, preventing unauthorized data exposure.

Additionally, the system will incorporate secure data transmission and storage practices, protecting sensitive information such as salary details and client contracts from breaches or leaks.

7. Integration of Communication and Feedback Channels

To promote continuous feedback and effective communication, the ERP integrates feedback submission features for clients and employees. This helps in capturing actionable insights that can be used to improve project delivery, employee satisfaction, and client relationships.

Seamless communication within the platform reduces dependency on external tools and creates a collaborative work environment.

8. Cost Efficiency and Resource Optimization

By automating administrative tasks and providing comprehensive visibility into resource utilization, the ERP system aims to reduce operational costs and optimize the allocation of manpower and financial resources. This contributes to improved profitability and sustainable growth for the organization.

9. Support for Decision-Making and Strategic Planning

With centralized data, real-time reporting, and analytics capabilities, the ERP system supports better decision-making by providing managers and executives with accurate, timely, and relevant information.

Strategic planning is enhanced by insights into project profitability, employee performance, expense patterns, and client engagement.

10. Compliance and Audit Readiness

The system ensures compliance with organizational policies and relevant industry standards by maintaining detailed records of transactions, attendance, and financial data. This audit trail facilitates internal and external audits, reduces compliance risks, and builds stakeholder confidence.

1.4 Target Audience

The ERP Management System is designed for multiple user groups, each with unique expectations and needs:

- **Administrators:** These users have full control over the system. They manage organizational branches, assign Project Managers, onboard Developers, handle payroll, and track business performance.
- **Project Managers:** Responsible for task distribution, developer performance tracking, client interactions, and handling student or trainee sessions. Their dashboard acts as a control panel for day-to-day management.
- **Employees / Developers:** They can view assigned projects, update task status, view salary details, manage session schedules, and give internal feedback.
- **Clients / Students:** Clients can see project status, request features, pay fees in installments, book sessions, and refer friends. The interface is designed to be user-friendly and informative.

This segmentation ensures that every role within the business ecosystem has dedicated tools to operate efficiently and transparently.

1.5 Vision and Mission of ERP Management System

- **Vision:** To build an intelligent ERP platform that simplifies and enhances the management experience for small-to-mid-sized IT companies and institutes, making enterprise-level automation accessible to all. The long-term vision of this project is to democratize access to advanced enterprise resource planning systems by offering a cloud-based, affordable, and easy-to-use application that can adapt to the needs of dynamic and fast-growing IT businesses. It aims to transform how companies manage operations—from employee workflows to client communications and financial oversight—by delivering a robust, integrated, and intelligent solution.

Furthermore, the vision encompasses creating a paperless environment, reducing redundancies in management, and ensuring that every stakeholder within the organization has seamless access to the tools they need to perform their roles effectively. With continued development and integration of advanced features like AI-based reporting, chatbot assistants, and predictive analytics, the vision is to scale this ERP system into a comprehensive digital backbone for all types of service-oriented organizations.

The system aspires to remove barriers to digital transformation by simplifying the onboarding process, offering training materials, and facilitating easy customization to match specific company needs. In future iterations, it may evolve into a low-code platform, where companies can customize dashboards and workflows without extensive technical expertise. This vision sets the stage for ERP not just as a management tool, but as a growth partner for emerging businesses.

- **Mission:** To deliver a secure, role-based, and customizable ERP solution that enables seamless coordination between Admins, Managers, Developers, and Clients, improving productivity, visibility, and control across the organization.

The mission is centered around empowering businesses through innovation. It focuses on transforming traditional and outdated administrative practices by automating workflows, centralizing data, and reducing operational costs. The ERP system is not just a tool—it's a strategic resource for organizations aiming to stay competitive in a digital-first economy.

Specific mission goals include:

- Designing user-centric interfaces that align with real-world roles and processes.
- Ensuring robust backend infrastructure that supports scalability and real-time performance.
- Delivering high-quality data insights for smarter decision-making.
- Promoting continuous feedback and iterative improvement to evolve the system over time.
- Encouraging ethical development practices including data privacy, transparent reporting, and sustainable deployment.
- Supporting integration with emerging technologies like machine learning for predictive analytics, blockchain for data integrity, and cloud-native services for global scalability.

- Establishing a strong support ecosystem including documentation, community support, and optional managed services.

The ERP system's mission also involves enabling a culture of accountability within organizations—where every task, resource, and outcome is traceable, measurable, and optimizable. The mission further includes advocating digital inclusion by offering accessible interfaces and multi-language support to expand usability in diverse regions.

1.6 Scope of the Project

The scope of the ERP Management System is broad and comprehensive, targeting the specific operational needs of IT companies and training centers.

Functional Scope Includes:

- Secure user authentication and session handling for multiple user roles.
- Admin functionalities to manage branches, employees, tasks, attendance, salaries, expenses, inquiries, and profit/loss reports.
- Project Manager dashboard for managing developers, assigning tasks, session tracking, student admissions, money collections, and feedback.
- Employee interface for task management, session records, salary details, and updating project status.
- Client panel for project progress tracking, requirement submissions, session booking, and referral sharing.
- Feedback modules and analytics dashboards for quality improvement and service monitoring.
- Notification system for alerts related to task updates, salary disbursement, and feedback entries.
- Integrated attendance systems using timestamps or QR code scans for accuracy.
- Performance tracking dashboards using charts and key performance indicators (KPIs).

The project will not cover enterprise-level integrations such as SAP or Oracle-based ERP interconnectivity at this stage. However, the design remains open for API-based connections for future integrations.

1.7 Structure of the Report

The report is organized into the following chapters:

- **Chapter 1:** Introduction (Overview, Innovation, Scope, Objectives)
- **Chapter 2:** Literature Review
- **Chapter 3:** Technologies Used (Frontend, Backend, Database)
- **Chapter 4:** Features and Functionality (Detailed module-wise functionality)
- **Chapter 5:** Implementation Details

  Conclusion

  Reference

Each chapter is structured to provide detailed insight into how the ERP Management System is built, implemented, and evaluated, covering both technical and user-oriented perspectives.

Chapter 2

LITERATURE REVIEW

Enterprise Resource Planning (ERP) systems are the backbone of modern organizational management, offering integrated solutions for managing business processes across departments. As digital transformation has grown across sectors, ERP systems have evolved from traditional desktop software to cloud-based, modular, and scalable platforms. This chapter explores the background of ERP systems, currently available tools and technologies, existing approaches, and the challenges or limitations associated with present-day ERP solutions.

2.1 Introduction

ERP systems unify diverse business processes—such as inventory, human resources, finance, and customer relationship management—into one centralized interface. By integrating these processes, organizations reduce redundancy, improve accuracy, and enable better decision-making. The need for ERP systems continues to grow with increasing business complexity, the shift toward remote work, and the growing demand for real-time data insights.

Enterprise Resource Planning (ERP) systems are the backbone of modern organizational management, offering integrated solutions for managing business processes across departments. As digital transformation has grown across sectors, ERP systems have evolved from traditional desktop software to cloud-based, modular, and scalable platforms. This chapter explores the background of ERP systems, currently available tools and technologies, existing approaches, and the challenges or limitations associated with present-day ERP solutions.

2.2 Existing ERP Tools and Technologies

Several ERP platforms exist in the market today, ranging from large-scale enterprise tools to open-source solutions. Each of these tools aims to centralize business processes, automate operations, and provide analytical insights to enhance decision-making.

2.2.1 Proprietary ERP Systems

- **SAP ERP:** SAP is considered a pioneer in enterprise resource planning. It offers comprehensive modules covering finance, supply chain, production, human resources, and customer services. SAP's strength lies in its scalability and deep

integration features. It supports multi-national operations and complies with various global financial standards. However, SAP requires a large team of trained professionals for installation, customization, and maintenance.

Despite its powerful capabilities, SAP is costly to deploy and maintain, especially for startups and mid-sized firms. The implementation timeline can stretch over several months, involving multiple consultants, software licenses, and hardware investments. Additionally, users often report a steep learning curve due to its complex UI and feature-rich environment.

- **Oracle NetSuite**: A cloud-based ERP solution that offers real-time visibility into operational and financial performance. NetSuite's seamless integration with financial tools, CRM, and e-commerce platforms makes it a solid choice for fast-growing businesses. It provides built-in business intelligence and customizable dashboards.

However, NetSuite may be financially out of reach for small businesses. While its cloud-based nature is an advantage, the software often requires experienced technical personnel to customize it to specific business workflows. Furthermore, the initial setup and subscription costs can be prohibitive for non-enterprise users.

- **Microsoft Dynamics 365:** Dynamics 365 combines ERP and CRM capabilities, offering flexibility, real-time insights, and seamless integration with Microsoft's ecosystem (e.g., Excel, Outlook). It is suitable for both service-based and product-based organizations.

Although Dynamics 365 offers modularity and familiar UI design, it still demands a significant investment in licenses and implementation. Customization of workflows and user training require a dedicated IT team, making it less viable for organizations with limited technical expertise.

- **Tally ERP 9:** Tally is widely used in India for accounting, taxation, and inventory management. It is known for simplicity, localized compliance, and affordability. Tally's offline nature makes it useful for businesses without reliable internet access.

The downside is its limited scalability and lack of advanced modules for project management, HR, and analytics. Tally is more of an accounting solution than a full-fledged ERP, making it insufficient for growing organizations with complex operations.

2.2.2 Open-Source ERP Systems

- **Odoo:** An open-source ERP suite offering modular apps for finance, sales, HR, manufacturing, and inventory. Odoo provides both a community edition and an enterprise version. Its drag-and-drop interface and flexible module system make it highly customizable for developers.

However, Odoo's full potential often remains untapped without significant technical knowledge. The community edition lacks advanced features available in the enterprise version. Organizations also face challenges during version upgrades, which may require complete data migration and redevelopment.

- **ERPNext:** Built on Python and the Frappe framework, ERPNext is user-friendly and cost-effective for small businesses. It covers a wide range of functionalities like accounting, stock management, CRM, and payroll. It also includes RESTful APIs for third-party integration.

While ERPNext is easy to deploy, it still has limitations in UI design and lacks advanced tools for predictive analytics. Smaller teams may struggle with support, especially if they are not familiar with Python development

- **Dolibarr:** Known for combining ERP and CRM features in one lightweight application. It is easy to set up and offers basic modules like invoices, stock, and sales management. Its open-source nature makes it ideal for startups.

On the downside, Dolibarr lacks scalability and is not suitable for complex enterprise needs. The feature set is minimal compared to larger ERP platforms, and its user interface appears outdated when compared to modern web standards.

2.2.3 Technology Stack Used in ERP Development

Modern ERP systems rely on a robust technology stack that ensures scalability, security, and responsiveness. Technologies are chosen based on application size, data complexity, and user roles.

- **Frontend Technologies:** React.js, Angular, and Vue.js are the most popular frontend frameworks used for ERP UI development. They enable the creation of dynamic dashboards, reusable components, and responsive layouts.

React.js in particular is widely adopted for ERP systems due to its component-based architecture and ability to handle real-time data rendering. These frontend frameworks also support integration with chart libraries, responsive design, and state management tools like Redux.

Backend Technologies: Node.js, Django, Laravel, and Spring Boot are commonly used for backend logic and APIs. Node.js with Express.js enables asynchronous, non-blocking operations, ideal for systems with multiple users accessing data simultaneously.

Django (Python-based) and Spring Boot (Java-based) offer ORM-based modeling, built-in authentication, and robust APIs. The choice of backend often depends on the development team's expertise and organizational needs.

- **Databases:** ERP systems use both SQL (MySQL, PostgreSQL) and NoSQL (MongoDB) databases. SQL is suitable for structured data and transactional operations, while MongoDB handles flexible data structures and unstructured documents.

MongoDB is highly scalable and cloud-compatible, making it a go-to solution for modern ERP systems with modular data relationships. Its JSON-like document format is ideal for rapid development and microservices integration.

- **Authentication and Security:** JWT, OAuth 2.0, and HTTPS/SSL encryption are essential for securing user sessions, especially in cloud-based ERPs. RBAC (Role-Based Access Control) is also crucial to restrict access per user role.

Security considerations also include database-level encryption, API access restrictions, and session expiration policies. These layers of protection ensure that ERP data remains confidential and protected from breaches.

- **Hosting & Deployment:** Cloud platforms like AWS, Azure, Heroku, Vercel, and Render provide scalable infrastructure. Docker is often used to containerize ERP apps for easy deployment and environment consistency.

CI/CD tools like GitHub Actions or Jenkins are used to automate testing and deployment. Cloud deployment improves uptime, reduces infrastructure costs, and enables remote accessibility.

2.3 Existing Approaches in ERP System Design

ERP systems have seen significant evolution in architecture and deployment methods. Early ERP platforms were built using monolithic architectures where all components—database, business logic, and UI—were bundled together. While this made deployment relatively straightforward, it also resulted in performance bottlenecks and high maintenance overhead.

Today, the shift is toward modular and microservices-based ERP architectures. These allow each functional unit (e.g., HR, inventory, finance) to be developed, updated, and scaled independently. This makes the system more flexible, easier to maintain, and more aligned with modern DevOps practices. Each module communicates through RESTful APIs or message brokers, allowing better fault isolation and continuous integration.

2.4 Limitations of Existing ERP Systems

Despite their versatility, existing ERP systems have several limitations. Cost is a primary barrier. Enterprise-level ERP solutions require substantial investment not only in software but also in training, infrastructure, and post-implementation support. This makes them inaccessible to startups or small organizations with limited IT budgets.

Another limitation is adaptability. Many ERP tools are rigid, with limited customization options unless supported by skilled developers.

These systems often assume a standard workflow, which may not suit every organization. This results in either inefficient process alignment or the need for expensive third-party customization. In some cases, upgrades or migrations can disrupt existing configurations, leading to data inconsistency and user dissatisfaction.

Chapter 3

TECHNOLOGIES USED

The technologies we chose have a direct effect on the system's performance, scalability, maintainability, and overall user experience. After carefully analyzing the project requirements, the skills of our developers, and the resources at our disposal, we found that the MERN stack was the best fit for building the application.

3.1 Introduction to Technologies Used

When it came to developing the ERP Management System, picking the right technology stack was a crucial first step. What is the MERN Stack? MERN is an acronym for MongoDB, Express.js, React.js, and Node.js — a modern, full-stack JavaScript solution that's perfect for creating dynamic web applications. MongoDB: This is a NoSQL, document-oriented database that stores data in JSON-like BSON documents. Express.js: A lightweight web application framework for Node.js, it's used to create backend APIs. React.js: Developed by Facebook, this JavaScript library helps in building rich, interactive user interfaces. Node.js: This runtime environment allows JavaScript to be executed on the server side. Together, these technologies empower developers to create a scalable, maintainable, and high-performance web application entirely in JavaScript, covering everything from the frontend to the backend and the database. Why MERN Stack for the ERP System? Several important factors led us to choose the MERN stack:

- **Unified Language Across the Stack: -**One of the standout benefits of MERN is that the entire application — from the frontend to the backend and even the database queries — is written in JavaScript. This consistency minimizes context switching for developers, simplifies the development process, and speeds up debugging and iteration cycles.
- **Flexibility and Scalability: -** MongoDB's document-oriented, schema-less design provides incredible flexibility to adapt to changing data structures. This is especially beneficial for ERP systems, where different modules (like attendance, payroll, and projects) require complex and varied data models. Additionally, Node.js's event-driven, non-blocking I/O architecture can handle many concurrent users without slowing down, which is essential for a multi-user enterprise environment.

- **Rich Ecosystem and Community Support: -** the MERN stack is backed by vibrant open-source communities and a wealth of libraries and plugins. This means development teams can tap into pre-built solutions for common challenges like authentication, form validation, and routing. As a result, development speeds up and the overall quality of the code gets a nice boost.
- **Modern, Responsive User Interfaces:** Reacts component-based architecture, developers can whip up dynamic and highly interactive user interfaces that manage rendering and state efficiently. This is crucial for ERP systems, which often require complex forms, dashboards, and real-time updates to provide a seamless user experience.
- **Rapid Development and Prototyping** With the ever-changing needs of ERP systems, the MERN stack shines with its modularity and ease of integration. This allows for quick prototyping and the ability to add features incrementally, making it perfect for an iterative development approach.

  Overview of Each Component in Context
- **MongoDB:** Serves as the main data hub, keeping track of everything from users and employees to projects, attendance records, salaries, tasks, inquiries, and beyond. Its JSON-like document structure fits neatly with the JavaScript logic of the application, making data manipulation a breeze.
- **Express.js:** Helps create RESTful APIs that act as the communication link between the frontend and the database. Express takes care of routing, middleware, and server-side business logic, including authentication and data validation.
- **React.js:** Drives the frontend interface, rendering various views tailored to different roles like Admin, Project Manager, Employee, and Client. React allows for dynamic components such as data tables, charts, forms, and notifications that respond to real-time data changes.
- **Node.js:** Operates the backend server, running JavaScript code outside the browser. Node efficiently handles asynchronous events, manages database connections, and ensures quick API response times, even when the system is under heavy load.

3.2 Database Technology

Introduction to MongoDB

MongoDB is a NoSQL database that stores data in a document format known as BSON (Binary JSON), which gives it a remarkable level of flexibility and scalability. Unlike traditional relational databases like MySQL or PostgreSQL, MongoDB doesn't need predefined schemas. This means it can easily adjust to changes in data structure—something that's crucial for ERP systems that often evolve as businesses expand.

In our ERP Management System, MongoDB acts as the main database, holding all the system data: user records, projects, attendance, salary details, tasks, feedback, and much more.

3.2.1 Why Choose MongoDB for ERP?

ERP systems deal with complex, interconnected, and frequently changing data. Opting for MongoDB instead of relational databases comes with a host of benefits:

Schema Flexibility: MongoDB allows documents (or records) to have varying fields. Since our ERP modules (like Admin, Employee, Client) may need unique data, MongoDB can accommodate them without rigid table structures.

Speed: MongoDB is designed for high-speed read and write operations, ensuring that real-time dashboards and instant data retrieval are seamless.

Horizontal Scaling: It supports sharding, which means large datasets can be split across multiple servers, making it easy to scale as the user base grows.

JSON-Like Structure: With data stored in JSON-like documents, it integrates effortlessly with JavaScript in our MERN stack, minimizing the need for data transformation.

3.2.2 Data Storage in MongoDB (ERP Modules)

MongoDB organizes data using collections (like tables in SQL) and documents (like rows). Here's how various ERP modules are structured:

ERP Module MongoDB Collection Sample Fields in Document

ERP Module	MongoDB Collection	Sample Fields in Document
Admin	admins	name, email, password Hash, created at
Employee/Developer	employees	name, skills, projectId, attendance, salary, tasks
Project Manager	project managers	name, email, assigned Projects, expenses

Projects	projects	title, client Name, assigned Employees, status
Client/Customer	clients	name, contact, feedback, booked Sessions
Tasks	tasks	task Title, assigned To, projectId, due Date
Attendance	attendance	employeeId, date, status (present/absent)
Feedback	feedback	userId, rating, message, role

Each of these collections uses MongoDB's flexible structure to adapt as modules grow and change over time.

3.2.3 Example Document Structures

1. Employee Document (in employees' collection):

```
{
  "_id": "65edc3290b56d1c4a2d0f789",
  "name": "Amit Sharma",
  "email": "amit@example.com",
  "skills": ["React", "Node.js"],
  "assignedProject": "ProjectX",
  "attendance": [
    {"date": "2025-05-01", "status": "Present"},
    {"date": "2025-05-02", "status": "Absent"}
  ],
  "salary": 40000,
  "tasks": ["Task1", "Task2"]
}
```

2. Project Document (in projects collection):

```
{
  "_id": "65edc319abcde1c4bcd12345",
  "title": "ERP Development",
  "clientName": "TechCorp Pvt Ltd",
  "status": "Ongoing",
  "assignedEmployees": ["Amit Sharma", "Neha Verma"],
  "created_at": "2025-04-15"
}
```

These flexible structures allow us to store nested objects (like attendance records or tasks) without the need to normalize data as in traditional SQL databases.

3.2.4 MongoDB Query Examples

MongoDB uses the `find()` method to search documents. Some real queries from our ERP system:

- Fetch all present employees on a given date:

```
db.attendance.find({ "date": "2025-06-01", "status": "Present" })
```

- Get all tasks for a specific employee:

```
db.employees.find({ "name": "Amit Sharma" }, { "tasks": 1 })
```

- Find all active projects:

```
db.projects.find({ "status": "Ongoing" })
```

3.2.5 Integration with Mongoose (ODM Layer)

In our ERP system, we use **Mongoose**, a popular Object Data Modeling (ODM) library for MongoDB and Node.js. Mongoose provides a schema-based solution to model our application data.

Key Benefits:

- Define models and data types.
- Add validation, default values, and pre-save hooks.
- Simplifies database operations via methods like .save(), .find(), .updateOne().
- Example Schema:

```
const EmployeeSchema = new mongoose.Schema({
  name: String,
  email: String,
  skills: [String],
  salary: Number,
```

```
  tasks: [String]

});
```

- **Security and Best Practices**

Security is crucial when handling sensitive employee and company data. MongoDB offers several features to enhance security:

- Authentication: MongoDB uses role-based access control (RBAC) to manage what users can do.
- Data Encryption: Data in transit is secured via SSL/TLS. Data at rest can be encrypted using MongoDB Enterprise features.
- NoSQL Injection Protection: By avoiding direct user input in queries and using Mongoose's query builder, injection risks are minimized.
- Regular Backups: Automated backup scripts help preserve data in case of failures.

3.3 Frontend Technologies (React.js, HTML, CSS, JavaScript)

Frontend development is all about the client-side of web applications, focusing on what users see and interact with in their browsers. In our ERP system, the frontend acts as a vital link between the user and the backend logic, making sure that all features are easy to access and neatly organized.

Our ERP platform caters to a variety of users, including Admins, Project Managers, Employees, and Clients, each with their own set of features, dashboards, forms, tables, and interactions. It was essential for us to create a clean, responsive, and role-based user interface to ensure the ERP system is both scalable and user-friendly.

To make this happen, we chose React.js as our main frontend library, complemented by standard technologies like HTML5, CSS3, and JavaScript (ES6+). Together, these tools allowed us to:

- Build a modular and scalable codebase with reusable UI components.
- Provide real-time interactivity through Reactjs state and props system.
- Manage client-side routing and navigation without needing to reload the page.
- Integrate with REST APIs using fetch and axios for smooth data exchange.
- Implement role-based rendering, so each user only sees the features they’re allowed to access.

For instance, when an admin logs in, the interface dynamically displays options for managing users, projects, and finances. In contrast, when a client logs in, the dashboard

only shows project updates and session bookings. This dynamic UI behavior is powered by Reactjs conditional rendering and state management. Additionally, we ensured responsive design using Flexbox, CSS Grid, and media queries, so the application runs smoothly on desktops, tablets, and smartphones. Security was also a priority on the client side: we implemented token-based authentication (JWT) and restricted access to certain routes by wrapping pages with route guards using react-router-dom and protected routes logic. In summary, the frontend of our ERP project is designed to deliver an outstanding user experience (UX) through careful structuring, modern frameworks, and a responsive, intuitive UI tailored to each user's needs.

3.3.1 Why React.js for ERP?

React.js is a JavaScript library developed by Facebook for building user interfaces—especially single-page applications (SPAs) where the page doesn't reload every time the user clicks.

Key Features of React.js used in ERP:

- **Component-Based Architecture**
  Each section of the ERP system (dashboard, project list, attendance form, etc.) is a reusable React component, making the code modular and easier to manage.
- **Virtual DOM**
  React uses a virtual DOM to efficiently update only the parts of the interface that change, improving performance.
- **JSX Syntax**
  JSX allows writing HTML-like syntax inside JavaScript, making UI code easier to understand and debug.
- **Hooks**
  React hooks like useState, useEffect, and useContext were used for state management, side effects, and context sharing (like user roles and authentication).
- **Routing**
  We used react-router-dom to create smooth navigation between ERP modules (e.g., /admin/dashboard, /employee/tasks, etc.) without full page reloads.

HTML5 – The Skeleton

HTML (Hypertext Markup Language) forms the structure of all pages.

Key Features used:

- <header>, <footer>, <section>, <article> for semantic structure.
- <form> tags for login, registration, and task submission.
- <table> for displaying project lists, attendance records.
- <input> fields with different types for forms (date, number, email, etc.).

HTML5 was integrated within React components using JSX.

CSS3 – The Styling Layer

CSS (Cascading Style Sheets) is used to make the ERP visually appealing and user-friendly.

Styling Strategies Used:

- Custom CSS files for each module/component.
- Flexbox and Grid for responsive layouts.
- Media Queries for mobile and tablet responsiveness.
- Color Themes for different user roles (e.g., blue for Admin, green for Project Manager).
- Animations using @keyframes for task status updates or dashboard transitions.

Summary

In conclusion, our ERP system frontend is built using **modern web development** technologies. React.js powers the user interface logic, while HTML and CSS bring structure and design. JavaScript makes everything interactive. Combined, these tools offer a dynamic, responsive, and easy-to-use experience for every role in the system.

3.4 Backend Technologies (Node.js, Express.js, REST APIs)

The backend of the ERP system is responsible for all **server-side operations**, such as database communication, user authentication, session management, business logic, and API handling.

We built the backend using **Node.js** with **Express.js** framework, along with **MongoDB** for the database (covered in 2.2). The backend acts as a **middleware** between the frontend and the database.

3.4.1 Why Node.js and Express.js?

Node.js:

- Built on **Chrome's V8 JavaScript Engine**.
- Allows writing **server-side logic in JavaScript**, aligning perfectly with the React-based frontend.

- Handles **concurrent requests efficiently** using its non-blocking, event-driven model.

Express.js:

- A minimal and flexible Node.js web application framework.
- Helps in building RESTful APIs quickly.
- Supports middleware for handling authentication, error logging, and request validation.

Authentication and Role Management

We implemented **JWT-based (JSON Web Token)** authentication for all user roles.

Flow:

1. User logs in → credentials verified.
2. Server generates JWT token → sent to client.
3. Token is stored in localStorage and used in headers for future API calls.
4. Middleware verifies token and user role.

Role-Based Access Control (RBAC) ensures users only access what they're authorized to.

Middleware Examples

Middlewares used in Express for:

- **Authentication:** Verify JWT tokens.
- **Validation:** Check for empty or malformed inputs.
- **Error Handling:** Catch unexpected server errors.

```
// Auth Middleware Example
const verifyToken = (req, res, next) => {
  const token = req.headers['authorization'];
  if (!token) return res.status(403).send('Token required');
  try {
    const decoded = jwt.verify(token, SECRET_KEY);
    req.user = decoded;
    next();
  } catch (err) {
    return res.status(401).send('Invalid token');
  }
};
```

Summary

The ERP backend system, built using Node.js and Express.js, provides a robust, scalable, and secure foundation for all modules. By following RESTful API principles and using JWT authentication, it ensures high performance, data security, and role-based access. Together with MongoDB and Mongoose, it creates a complete MERN backend stack for enterprise-grade applications.

3.5 Postman – API Testing Tool

Introduction to Postman Postman has become a go-to tool for many when it comes to API development, testing, and debugging. Its user-friendly graphical interface makes it super easy for developers to send HTTP requests to backend servers and see the responses, all without needing a complete frontend setup. For our ERP Management System project, Postman was a crucial asset for testing the RESTful APIs we built using Node.js and Express.js. Even before we wrapped up the frontend integration, Postman helped us ensure that every endpoint was functioning correctly, accurately, and reliably.

Why Postman was Essential in Our ERP System Our ERP project involves various roles like Admin, Manager, Employee, and Client. Each of these roles has its own specific APIs, such as: Admin: /api/admin/createUser, /api/admin/manageProject

Manager: /api/manager/assignTask

Employee: /api/employee/viewProject

Client: /api/client/sendFeedback

Manually testing all these endpoints through the frontend would have taken a lot of time and could lead to UI-related mistakes. With Postman, we were able to: Test backend logic on its own Quickly troubleshoot any failed requests Verify that responses and error codes were correct Easily add authentication tokens in headers Simulate different user roles without hassle

Postman allowed us to:

1. **Test backend logic independently**
2. **Quickly debug failed requests**
3. **Confirm correct responses and error codes**
4. **Add authentication tokens in headers**

5. **Simulate multiple user roles easily**

Core Features of Postman Used in ERP

Request Building (GET, POST, PUT, DELETE)

Postman lets you build any HTTP request easily. For example, to test a **POST request to create a new task**, we:

- Selected POST as method
- Entered http://localhost:5000/api/admin/createTask
- Added a JSON body like:

```
{
  "title": "Fix UI bugs",
  "assignedTo": "emp123",
  "dueDate": "2025-06-10"
}
```

Collections and Environments

We created **collections** for each user role, grouping their respective APIs:

- Admin Collection
- Manager Collection
- Client Collection

We also used **environments** to switch between development and production URLs quickly.

Automated Tests and Assertions

Postman allows you to write test scripts in JavaScript. For example:

```
pm.test("Status is 200", function () {
    pm.response.to.have.status(200);
});

pm.test("Response contains user ID", function () {
```

```
    pm.response.to.have.jsonBody('userId');
});
```

Benefits Gained from Using Postman in the Project

Benefit	Explanation
Faster Debugging	We could test responses immediately, without waiting for frontend integration.
Independent Testing	Developers worked on frontend and backend in parallel. Postman served as the communication bridge.
Error Identification	Helped us spot issues like missing fields, incorrect status codes, or invalid request formats.
Clear Documentation	Postman collections can be exported and shared with team members as API documentation.
Security Testing	Allowed us to simulate unauthorized access by omitting tokens, ensuring that protected routes were secure.

Postman played a **crucial role** in the successful development of our ERP system. It improved our team's productivity, ensured better code quality, and provided an efficient way to test, debug, and document APIs. Its flexibility, powerful features, and easy-to-use interface made it an indispensable tool during the entire development life cycle of the ERP system.

3.6 Security Technologies

When it comes to a modern ERP system—especially one that deals with user management, financial data, session bookings, and employee information—security is absolutely crucial. We can't afford to overlook it. Unauthorized access, data leaks, and the insecure transmission of sensitive information can lead to significant financial and reputational damage for organizations. That's why our ERP system is built with strong, contemporary, and multi-layered security measures to guarantee the confidentiality, integrity, and availability of data.

Here's how we secure our ERP application:

- We use JWT-based authentication to verify user identities.

- We implement role-based access controls (RBAC) to manage authorization.Sensitive data, such as passwords and payment details, is encrypted.
- We ensure data in transit is protected by using HTTPS.

3.7 ERP Management System Architecture

The ERP (Enterprise Resource Planning) Management System is a full-stack web application built using the MERN stack: MongoDB, Express.js, React.js, and Node.js. The system is designed to support multiple user roles (Admin, Project Manager, Developer, and Client) and is modular, scalable, and secure through the use of JWT-based authentication. Below is a breakdown of the system architecture as represented in the image**.**

Our ERP platform caters to a variety of users, including Admins, Project Managers, Employees, and Clients, each with their own set of features, dashboards, forms, tables, and interactions. It was essential for us to create a clean, responsive, and role-based user interface to ensure the ERP system is both scalable and user-friendly.

To make this happen, we chose React.js as our main frontend library, complemented by standard technologies like HTML5, CSS3, and JavaScript (ES6+).

Our ERP Management System is a full-stack web application built using the MERN stack. The system is modular, scalable, and features role-based access control to serve different stakeholders like Admins, Project Managers, Developers, and Clients. The architecture follows a client-server model where the frontend interacts with the backend via secure REST APIs, which in turn communicates with a cloud-hosted MongoDB database. JWT authentication ensures security and access control for each module and user role.

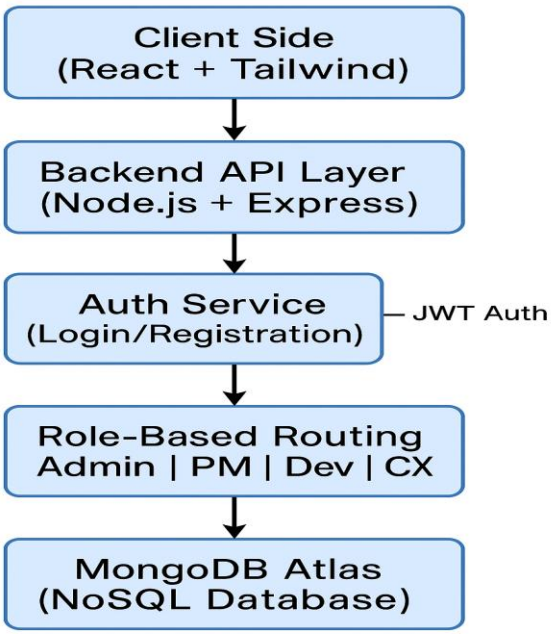


Fig 3.7 Architecture

1. Client Side (React + Tailwind CSS)

This is the frontend layer of the application, where users interact with the system. Built using **React.js** for dynamic, component-based user interfaces and **Tailwind CSS** for rapid styling and responsiveness, this layer:

- Handles routing with React Router
- Communicates with the backend via RESTful APIs
- Implements role-based dashboards (Admin, PM, Developer, Client)
- Offers secure forms, interactive tables, charts, and notifications
- Protects pages using JWT stored in secure cookies or local storage

2. Backend API Layer (Node.js + Express.js)

This layer acts as the bridge between the frontend and the database. Built with **Node.js** and **Express.js**, the backend:

- Exposes RESTful endpoints (e.g., /api/projects, /api/tasks, /api/auth)
- Handles CRUD operations for different modules: users, tasks, projects, attendance, enquiries, etc.
- Implements middleware functions for logging, error handling, and authentication
- Validates user input and performs business logic before interacting with the

database

Example APIs:

- POST /api/login – authenticate user
- GET /api/projects – get all projects based on role
- POST /api/feedback – submit feedback

3. Authentication Service (JWT Auth)

At the core of security lies the **Authentication Service**. When a user logs in, they receive a **JWT (JSON Web Token)** that stores their ID, role, and expiration time. The frontend attaches this token in the headers of each request.

Functions:

- Secure login and registration
- Encrypt passwords using bcrypt
- Middleware to verify and decode tokens
- Session expiry and token refresh handling

Benefits:

- Stateless (no server-side sessions)
- Scalable for microservices or multi-tenant setups
- Enables fine-grained access control by roles

4. Role-Based Routing (Admin | PM | Dev | CX)

Based on the decoded JWT token, each user is routed to a specific dashboard or panel. This layer manages access rights and UI rendering based on roles:

Admin:

- Manage branches, managers, developers
- View projects, enquiries, admissions, expenses
- Control over all modules

Project Manager:

- Assign and monitor developer tasks
- View feedback, attendance, and admissions
- Manage ongoing project lifecycle

Developer:

- View assigned tasks and project requirements
- Update task status and submit work logs
- Access salary details and give feedback

Client (Customer):

- View project status and assigned tasks
- Book sessions, send requirements
- Access fee details, suggestions, referrals

This separation of routes ensures that no user can access unauthorized data or features.

5. MongoDB Atlas (NoSQL Database)

All data is stored in **MongoDB Atlas**, a fully managed cloud NoSQL database. Collections (similar to tables in SQL) are designed to hold documents for:

- Users (with role info)
- Projects and tasks
- Feedback and enquiries
- Attendance and salaries
- Admissions and session bookings

Advantages of MongoDB:

- Schema-less flexibility
- Fast read/write performance
- Easy to scale
- Good fit for object-oriented backend in Node.js

Chapter 4

FEATURES AND FUNCTIONALITY

This chapter dives into the detailed features and functionalities of the ERP Management System. It's crafted to boost operational efficiency by clearly defining user roles—Admin, Project Manager, Developer, and Client. Each module comes equipped with specific tools and workflows tailored to its unique responsibilities.

4.1 Features

The application is designed for seamless integration of organizational processes, ensuring transparency, security, and real-time access to business data. The ERP system adopts a modular design, allowing each role to interact solely with the tools and data pertinent to their duties. This approach enhances clarity, security, and performance throughout the organization.

The system architecture promotes a smooth data flow, fostering strong collaboration among different roles. Built on the MERN stack (MongoDB, Express.js, React.js, Node.js), the ERP system also utilizes cloud-based services and modern web technologies to ensure responsiveness, scalability, and user-friendliness. Each module outlined below mirrors the real-world operational structure of an IT service company or a training institute. Each module is more than just a collection of features—it's a digital environment where users can manage, execute, and analyze their tasks in real time. By clearly segmenting tasks and responsibilities by role, the ERP system enhances efficiency and reduces cross-departmental dependencies. The upcoming sections will delve into each user module, starting with the admin, who holds the highest privileges.

4.2 Admin Module

The admin module is the control center of the ERP Management System. Users with admin privileges are responsible for the holistic operation of the platform, covering organizational, financial, technical, and managerial aspects. The module is designed to give administrators complete visibility and control over the system and users.

Admins initiate the setup of the ERP system, onboard different user roles, and ensure that tasks, resources, and performance metrics are aligned with organizational goals. From managing the database of employees and clients to generating detailed profit/loss statements, the Admin’s role is essential to the digital management of the company.

The admin is also responsible for safeguarding the integrity of the data. Through

powerful dashboards and administrative tools, this module allows in-depth configuration, monitoring, and analytics.

Another critical function is the ability to manage feedback loops and resource allocations in a way that maximizes productivity and minimizes overhead. The Admin is also expected to resolve inter-departmental issues and maintain financial discipline through automated expense and salary management features.

Key Features:

1. **Branch Management:**
    a. Add, update, or remove branches.
    b. Assign managers to branches.
    c. Monitor branch-specific operations and finances.
2. **Manager and Developer Control:**
    a. Add and remove users with specific roles.
    b. Assign developers to managers and projects.
    c. Enable/disable user accounts.
3. **Project Oversight:**
    a. Create new projects and allocate teams.
    b. Set milestones, deadlines, and priorities.
    c. View progress reports and completion status.
4. **Attendance Monitoring:**
    a. Track daily attendance of employees.
    b. Export attendance records.
    c. Generate monthly attendance summaries.
5. **Payroll System:**
    a. Automated salary calculation based on attendance and performance.
    b. Allow salary adjustments and deductions.
    c. Generate pay slips.
6. **Expense Management:**
    a. Log daily, weekly, or monthly expenses.
    b. Categorize expenses by branch or department.
    c. View financial summaries.
7. **Admissions & Inquiries:**
    a. Track incoming leads, admissions, and client conversions.
    b. Assign inquiries to managers.

c. Maintain a searchable inquiry database.

8. **Feedback Review:**
    a. View internal feedback from employees.
    b. Analyze external feedback from clients.
9. **Profit & Loss Reports:**
    a. Real-time analytics for profits and operational losses.
    b. Generate graphical reports using charts.
10. **Administrative Dashboard:**
    a. Graphs and tables displaying system KPIs.
    b. Overview of all modules in one place.

Functionality Summary: Admins function as the system superuser, ensuring operations, finances, and human resources are balanced and tracked efficiently. All reports can be downloaded in CSV or PDF formats for compliance and recordkeeping.

4.3 Project Manager Module

The Project Manager module bridges the gap between administrative oversight and on-the-ground project execution. It empowers managers to control, coordinate, and monitor daily operations, ensuring that client expectations and internal timelines align efficiently.

Managers are tasked with assigning work to developers, keeping track of project timelines, and ensuring timely delivery of milestones. This module allows them to manage both human resources and technical delivery with equal ease, helping to maintain a high standard of productivity.

Apart from technical coordination, Project Managers also interact directly with clients and students. They collect feedback, manage communication, and often function as the first point of contact for session planning and query resolution. Their ability to track performance and spot issues early helps mitigate risk.This module is equipped with financial monitoring tools, session planning features, and reporting mechanisms that allow managers to contribute to both operational and strategic decision-making.

Key Features:

1. **Developer Management:**
    a. Assign developers to specific tasks and projects.
    b. Track performance and task submissions.
    c. Send notifications and task updates.

2. **Project Tracking:**
    a. View real-time progress reports.
    b. Manage project milestones and phase-based tracking.
    c. Upload documentation, source files, and notes.
3. **Session Scheduling:**
    a. Schedule student or client sessions.
    b. Mark attendance and assign mentors.
    c. Generate attendance reports.
4. **Financial Access:**
    a. Record and validate client fee collection.
    b. Approve or raise department-level expenses.
5. **Client Handling:**
    a. View and resolve client-submitted requirements.
    b. Engage in session feedback.
    c. Provide updates to clients via the dashboard.
6. **Performance Analysis:**
    a. Track employee productivity.
    b. Monitor time spent on tasks.
    c. Identify blockers and delays.

Functionality Summary: Managers use this module to maintain smooth project execution and client satisfaction. Their dashboard includes visual indicators of workload distribution, timelines, and client ratings.

4.4 Developer (Employee) Module

The Developer module caters to internal team members responsible for coding, testing, and delivering project components. It is designed to simplify work organization, task tracking, and performance visibility for employees.

Employees use this module daily to interact with their assigned tasks, update progress, track hours worked, and review feedback. The system ensures clarity on project deadlines and simplifies workload management through an intuitive task dashboard.

In addition to task-related features, developers can monitor their attendance records, salary information, and session involvement. This increases transparency and builds trust between management and employees, especially in remote and hybrid work environments.

Developers are also encouraged to contribute feedback on internal processes, helping management improve team engagement and workflow efficiency.

Key Features:

1. **Project Dashboard:**
    a. Overview of assigned projects.
    b. Task progress bars.
    c. Daily check-in system.
2. **Task Status Management:**
    a. Update task progress (In Progress, Completed, Blocked).
    b. Attach screenshots or links.
    c. Comment section for notes.
3. **Time Tracking:**
    a. Log time spent on each task.
    b. Weekly productivity summary.
4. **Salary and Compensation:**
    a. Monthly salary breakdown.
    b. Attendance-linked payments.
    c. Bonus and deduction records.
5. **Feedback Loop:**
    a. Submit feedback about tools, tasks, or managers.
    b. View response status.
6. **Session Panel:**
    a. View booked sessions.
    b. Attendance and mentorship logs.

Functionality Summary: The Developer module empowers employees with independence and accountability. It also improves HR transparency by displaying salary structures and session engagement.

4.5 Client Module

The Client module serves as the communication and engagement interface for customers or students. It simplifies the process of interacting with the company by giving clients visibility into their project status, timelines, deliverables, and billing.

Clients can log in securely to check updates on their projects, submit new requirements, and participate in feedback loops. The module is especially useful for maintaining

customer satisfaction, transparency, and retention.

A major goal of this module is to reduce the need for manual status updates by staff, thereby saving time while improving client experience. Clients can also use this module to book sessions, raise queries, and refer others to the platform.

The interface is designed to be simple, intuitive, and responsive so that non-technical users can easily navigate and utilize its features.

Key Features:

1. **Project Timeline Viewer:**
    a. Graphical view of deliverables and progress.
    b. Phase-wise breakdown (e.g., Design, Development, Testing).
2. **Requirement Submission:**
    a. Submit change requests.
    b. Upload documents and reference links.
    c. Set priorities for requirements.
3. **Feedback Portal:**
    a. Rate sessions and interactions.
    b. Comment on delivery phases.
    c. Suggest service improvements.
4. **Billing and Payments:**
    a. View due and paid invoices.
    b. Installment tracker.
5. **Session Booking:**
    a. Book 1-on-1 sessions with developers or project leads.
    b. Get reminders via email/SMS.
6. **Referral Dashboard:**
    a. Refer contacts via email or phone number.
    b. Track referral bonuses.

- **Functionality Summary:**

Clients can independently manage their interactions with the company without relying on manual updates from managers. The dashboard gives them full visibility and control of their engagements. This extensive feature set empowers every stakeholder in the organization to perform their duties efficiently and effectively. Each feature is designed with scalability, usability, and maintainability in mind, ensuring the system remains

relevant and adaptable to future business needs. The multifunctional structure of the ERP Management System permits each user: administrator, project manager, developer, or client to interact with the platform without difficulty. Regarding each user's role, the system's productivity functions customization on all levels. Not only does individual productivity improve, but organizational productivity is also enhanced. Critical tasks like business processes are monitored, repetitive processes are automated, and milestones are monitored in real time.

Due to its modular architecture, departments can work independently on projects and remain integrated with other teams/subsystems through subordinate mean of control. Managers can track completed deliverables with precision. Developers, on the other hand, can receive assignments and know exactly what is expected of them. Clients are also kept informed without the need to request information. Bridges between departments that do not interact directly have been closed, while strategic objectives and operations are both aligned through a single digital workspace.

The ERP Management System is designed digitally for sustainability and for scalability tailored precisely for IT service companies and training institutes. Its blend of features, user experience, and principles of data security impact the organization's digital transformation process and reinforces operational and growth constructs.

4.6 Development Process

The development process of the ERP Management System followed a structured, agile-based methodology, divided into iterative sprints spanning multiple weeks. At the outset, the project team conducted a comprehensive requirement-gathering phase involving user interviews, process analysis, and research into existing ERP solutions.

4.6.1 Agile Methodology Adoption

1. **New version:**

To keep our development process running smoothly and adapt to changing needs, our team embraced the Agile methodology, specifically the Scrum framework. Agile allowed us to break down the intricate development of the ERP Management System into bite-sized tasks, enabling us to make steady progress through short, focused development cycles known as sprints.

2. **Why Choose Agile?**

ERP systems are typically large and modular, involving various user roles like Admin, Project Manager, Developer, and Client. Given its flexible nature, Agile was the perfect

fit because it encourages:

- Incremental delivery of features.
- Early and ongoing feedback.
- The ability to adapt to shifting requirements.
- Close collaboration among team members.

3. Scrum Practices We Followed

We organized our workflow around essential Scrum practices:

- Sprint Planning Meetings: Held at the beginning of each sprint to set goals and assign tasks.
- Daily Stand-ups: Brief meetings where each team member shared what they accomplished yesterday, what they plan to tackle today, and any challenges they're facing.
- Sprint Reviews: Showcasing completed features to evaluate functionality and gather feedback.
- Sprint Retrospectives:At the end of each sprint, we reflected on what went well, what didn't, and how we could enhance our process for the next sprint.

4. Roles and Responsibilities

Even though we were a small team, we adopted Scrum roles to ensure clarity and accountability:

- Scrum Master: Facilitated meetings and helped remove any roadblocks. (This role rotated among team members.)
- Product Owner: Acted as the voice of the stakeholders by keeping a prioritized backlog of features and clarifying requirements.
- Development Team: All members pitched in with coding, testing, and integration.

5. Tools We Used

To facilitate Agile development, we relied on:

- Trello: For sprint planning and task management using Kanban boards.
- Google Meet: For virtual stand-up meetings and sprint reviews.
- Git & GitHub: For managing source code and collaboration.
- Notion/Docs: For documenting user stories and project details.

6. The benefits we've seen from adopting Agile are pretty impressive:

- We were able to roll out essential ERP modules, like Admin and Project

Management, in no time.

- Our team stayed aligned thanks to a culture of transparency.
- We managed to navigate scope changes smoothly without throwing off our overall progress.
- We also got better at spotting and resolving issues early on, which has really boosted our code quality.

4.7 Sprints and User Stories

In line with the Agile Scrum methodology, we organized our development process into several sprints, each lasting between one to two weeks. Every sprint aimed to deliver a specific set of user stories that highlighted valuable, functional components of the ERP Management System. This iterative approach not only ensured consistent progress but also allowed us to gather feedback, make necessary adjustments, and continuously enhance the application.

1. **Sprint Planning and Execution**

Before kicking off each sprint, we held a Sprint Planning session where we:
- Reviewed the product backlog, which is essentially a list of all the features we wanted.
- Chose high-priority user stories to focus on during the sprint.
- Broke down those user stories into smaller, actionable tasks for development.
- Assigned roles and responsibilities to each team member.

At the end of every sprint, we conducted Sprint Review and Retrospective meetings to showcase the completed work, collect feedback, and discuss ways to improve for the next sprint.

2. **User Stories Format**

We adhered to a standard format for defining user stories:
As a [type of user], I want to [perform some action] so that [achieve some goal].
This approach kept us centered on the user's perspective, enabling us to create features that genuinely addressed real-world problems.
Detect AI-generated content and give it a human touch with our AI Content Detector. Just paste your text, and in a matter of seconds, you'll receive accurate, human-like results!
Here's the text we're analyzing: Sample User Stories by Module
We've put together some essential user stories for various modules:

- **Admin Module:**

    - As an Admin, I want to create and manage user accounts so I can assign roles and permissions.
    - As an Admin, I want to see all registered users so I can keep an eye on their activity and access.
- **Project Manager Module:**
    - As a Project Manager, I want to assign projects to developers to ensure work is distributed effectively.
    - As a Project Manager, I want to monitor project status so I can guarantee timely delivery.
- **Developer/Employee Module:**
    - As a Developer, I want to check my assigned tasks so I can organize my workload.
    - As a Developer, I want to update the status of a project to keep my manager in the loop.
- **Client/Customer Module:**
    - As a Client, I want to track the progress of my project so I can stay informed.
    - As a Client, I want to provide feedback on the project to help make improvements.

Task Breakdown and Tracking

We broke down each user story into specific tasks, such as:

- Frontend UI design (using React.js and Tailwind CSS)
- Backend API development (with Node.js and Express)
- Database schema design (in MongoDB)
- Testing and debugging
- Deployment and review

We utilized Trello to set up a Kanban board with columns like:

- To Do
- In Progress
- In Review

This approach helped us keep track of task progress, minimize confusion, and ensure transparency among team members.

Outcome of Sprint-Based Development

This sprint-based development method enabled us to:

- Deliver functional modules at regular intervals.
- Spot and resolve issues early on.
- Keep all team members aligned with project objectives.
- Make sure every feature was thoroughly tested and reviewed before finalization.

4.8 Challenges Faced

During the development of our ERP Management System, we faced a mix of technical and non-technical hurdles. These challenges taught us valuable lessons about adapting, communicating effectively, and finding practical solutions in real-world situations. Here's a breakdown of the key issues we encountered, along with our approaches to tackle them:

1. Requirement Analysis and Clarity

- Problem:

Understanding what our clients or stakeholders expected and translating those expectations into clear technical requirements was a tough nut to crack. Given the complexity of ERP systems, which include various modules (like Admin, Project Manager, Developer, and Client), each with its own workflows, this was no small feat.

- Solution:

To address this, we organized brainstorming sessions and created detailed user stories and flow diagrams for each module. This process allowed us to pinpoint the exact functionalities we needed to define before diving into the coding phase.

2. Team Coordination and Communication

- Problem:

As a student team, juggling our college schedules while coordinating development tasks proved to be quite challenging. Sometimes, miscommunication led to overlapping work or delays in progress.

- Solution:

We turned to Trello for task management and held weekly stand-up meetings via Google Meet. This approach kept everyone on the same page regarding project updates and outstanding tasks.

3. Version Control and Merge Conflicts

- Problem:

Collaborating on the same codebase often resulted in pesky Git merge conflicts, particularly when it came to updating shared components.

- Solution:

To mitigate this, we adopted a branch-based workflow where each team member worked on their own feature branch. We also made it a point to commit regularly, submit pull requests, and conduct code reviews before merging anything into the main branch.

4. Backend and Frontend Integration

- Problem:

When it came to integrating the frontend (React) with the backend APIs (Node.js + Express), we encountered mismatches in API responses, data formats, and authentication logic.

- Solution:

To streamline this process, we meticulously documented all API contracts and utilized tools like Postman to test endpoints before integrating them into the frontend. This proactive approach significantly reduced miscommunication between the frontend and backend teams.

5. Authentication and Role-Based Access Control

- Problem:

Making sure that our authentication was secure and that each user—whether they were an Admin, Manager, Developer, or Client—could only access their own data turned out to be quite a challenge.

- Solution:

We opted for JWT (JSON Web Tokens) to manage sessions securely and created middleware to check roles and permissions before granting access to any protected routes.

6. Time Constraints

- Problem:

With academic assignments, exams, and internship tasks all overlapping, we found ourselves pressed for time. This meant that some of our sprints had to be either extended or adjusted.

- Solution:

We focused on the essential features for each sprint and kept a flexible sprint backlog, allowing us to push lower-priority tasks to later phases without disrupting our main objectives.

7. Learning Curve with New Tools

- Problem:

Some of our team members were just getting acquainted with technologies like MongoDB, CI/CD pipelines, and Tailwind CSS, which initially put a bit of a brake on our development speed.

- Solution:

We split up the learning tasks and shared tutorials and documentation. Those who became proficient in certain tools took the lead in helping others through peer learning sessions.

4.9 Continuous Integration and Deployment (CI/CD)

To make sure our ERP Management System was built efficiently and rolled out smoothly, we set up a Continuous Integration and Continuous Deployment (CI/CD) pipeline. This CI/CD approach helped us automate our building, testing, and deployment processes, which cut down on manual errors and boosted our development speed and consistency.

- **Continuous Integration (CI)**

Continuous Integration is all about automatically building and testing code every time a team member makes changes to the shared repository. This practice ensures that new code doesn't disrupt existing features and keeps the application ready for deployment.
CI Practices We Implemented:

  - Version Control with GitHub: We pushed all code changes to GitHub, using feature branches for new modules.
  - Pull Requests (PRs): Before merging, we created pull requests for code reviews to ensure quality and functionality.
  - Code Linting and Formatting: We utilized linters (like ESLint) and formatters (like Prettier) to keep our code clean and consistent.
  - Automated Testing: We wrote basic unit and integration tests to validate components before merging them into the main branch.
  - GitHub Actions: We leveraged GitHub Actions to automate testing and linting whenever new code was pushed or a PR was created.

- **Continuous Deployment (CD)** is all about automatically rolling out every change that successfully passes through the CI process into either a staging or production environment.

CD Pipeline Steps:

- Build: Once there's a successful push or merge into the main branch, the codebase gets built automatically.
- Deploy: The application that's been built is then deployed to our cloud hosting environment, like Render, Vercel, or Netlify.
- Environment Variables: We set up secure environment variables in the deployment platform to handle things like database connections, API keys, and JWT secrets.
- Notification: After each successful deployment, team members receive a notification.
- Deployment Frequency: We deploy to a staging environment after every sprint for internal testing, and then to production after the final sprint, once all the core modules are completed and tested.

CI/CD Benefits for the Project

- Faster Feedback Loop: We catch errors and bugs early in the process.
- Reduced Manual Work: Automating the deployment saves time and minimizes the chance of human error.
- Improved Collaboration: Developers can concentrate on coding while the CI/CD pipeline takes care of testing and deployment.
- Reliable Releases: Each release is predictable and thoroughly tested, which leads to better quality and stability.
- Scalable Workflow: This setup makes it easy to add enhancements or grow the team without disrupting the deployment process.

1. Technology Stack Overview

- Our ERP Management System is crafted using the MERN stack:
  - MongoDB: A NoSQL database hosted on MongoDB Atlas, providing cloud storage and scalability.
  - Express.js: A web framework that helps us build RESTful APIs.
  - React.js: The frontend library we use to create the user interface.
  - Node.js: The runtime environment that executes our backend code.

2. Frontend Deployment

- We deployed the React.js frontend on Vercel (or Netlify, depending on your choice).

- This setup allowed for:
    - Continuous deployment directly from GitHub.
    - Automatic rebuilds with every push to the main branch.
    - Support for custom domains (if configured).
    - HTTPS support and an optimized global CDN for quicker load times.

3. Backend Deployment

- The backend, which consists of Node.js and Express, was hosted on Render (or Railway/Heroku).
- This provided:
    - Auto-deployment from GitHub after each push.
    - A runtime environment tailored for REST API servers.
    - Simple configuration for environment variables (like DB URIs, JWT secrets, etc.).
    - Support for logging and monitoring.

4. Database Management

- We utilized MongoDB Atlas, a cloud-hosted version of MongoDB.
- The benefits included:
    - High availability with auto-scaling capabilities.
    - Secure connections through connection strings and IP whitelisting.
    - Easy cluster setup and performance monitoring.
    - Compatibility with Mongoose ORM for smooth database operations.

5. Environment Variables and Secrets Management

To keep things secure and flexible:

- We stored sensitive information like JWT secrets, API keys, and database URIs as environment variables.
- These were set up directly in the dashboards of our deployment platforms (Vercel and Render), rather than being hardcoded into the codebase.

6. Deployment Flow

Here's how we handled the deployment process:

- Code was pushed to GitHub.
- GitHub Actions kicked off the CI process (which included building and testing).
- If the tests were successful:
    - The frontend was automatically deployed to Vercel.

- The backend was automatically deployed to Render.
- MongoDB Atlas stayed continuously connected to the backend.
- We tested the live site after each deployment.

7. Monitoring and Logs

- We kept an eye on backend logs using Render's built-in logging dashboard.
- Errors and deployment statuses were monitored in real-time.

This approach allowed us to quickly spot any bugs or deployment issues and fix them during testing.

4.10 Advantages

The ERP Management System crafted with the MERN stack offers a range of benefits for both the organization and its users. It simplifies project workflows, boosts team collaboration, and centralizes data access. Here are the standout advantages:

1. Centralized Information Management

Every department—be it Admin, Project Manager, Developer, or Client—can tap into a single system for sharing information, which cuts down on the need for various tools and manual communication.

Benefit: Enhances transparency and ensures data consistency across all roles.

2. Real-Time Project Tracking

Project Managers and Clients can monitor the progress of ongoing projects and tasks in real-time, which fosters accountability and speeds up decision-making.

Benefit: Facilitates proactive management and minimizes communication delays.

3. Role-Based Access Control

Users only see the information pertinent to their roles (Admin, Manager, Developer, Client), which bolsters data security and improves the user experience.

Benefit: Lowers the risk of unauthorized data access and safeguards data privacy.

4. Automation of Routine Tasks

The system takes care of several routine tasks like assigning projects, updating statuses, and managing feedback, which lightens the manual workload.

Benefit: Saves time and boosts productivity for team members.

5. Cloud-Based and Accessible Anywhere

Hosted on platforms like Vercel, Render, and MongoDB Atlas, the ERP system can be accessed from any device with an internet connection.

Benefit: Supports remote work and offers 24/7 accessibility.

6. Scalable and Modular Architecture

Built using the MERN stack, the system is both modular and scalable. New features or modules can be integrated without disrupting existing functionality.

Benefit: A future-proof solution that can evolve with the organization's needs.

7. Open Source & Cost-Effective

As a project developed by students, it’s open source and doesn’t come with hefty licensing fees or proprietary tools.

Benefit: Perfect for startups, educational institutions, or small businesses.

Chapter 5

IMPLEMENTATION DETAILS

The application is designed for seamless integration of organizational processes, ensuring transparency, security, and real-time access to business data. The ERP system adopts a modular design, allowing each role to interact solely with the tools and data pertinent to their duties. This approach enhances clarity, security, and performance throughout the organization. The system architecture promotes a smooth data flow, fostering strong collaboration among different roles. Built on the MERN stack (MongoDB, Express.js, React.js, Node.js), the ERP system also utilizes cloud-based services and modern web technologies to ensure responsiveness, scalability, and user-friendliness.

5.1 Admin Module

The admin module is the control center of the ERP Management System. Users with admin privileges are responsible for the holistic operation of the platform, covering organizational, financial, technical, and managerial aspects. The module is designed to give administrators complete visibility and control over the system and users.

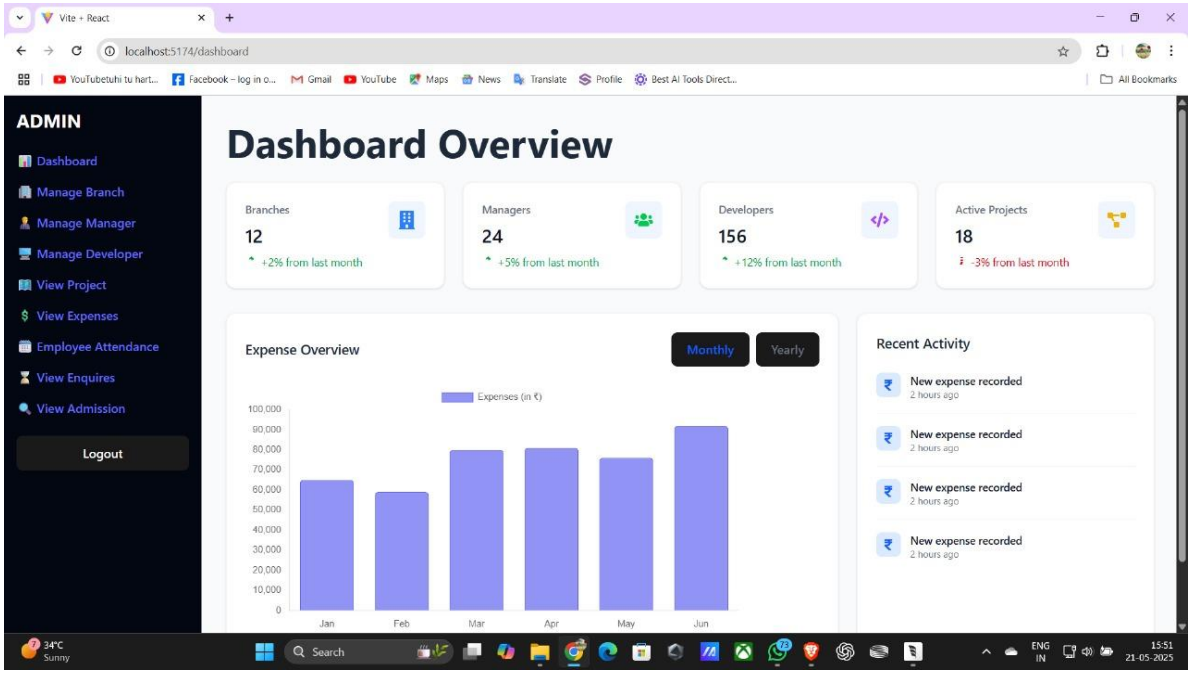


Fig 5.1 Admin Home Page

Admins initiate the setup of the ERP system, onboard different user roles, and ensure that tasks, resources, and performance metrics are aligned with organizational goals. From managing the database of employees and clients to generating detailed profit/loss statements, the Admin's role is essential to the digital management of the company.

The admin is also responsible for safeguarding the integrity of the data. Through powerful dashboards and administrative tools, this module allows in-depth configuration, monitoring, and analytics.

Another critical function is the ability to manage feedback loops and resource allocations in a way that maximizes productivity and minimizes overhead. The Admin is also expected to resolve inter-departmental issues and maintain financial discipline through automated expense and salary management features.

Manage Branch

- **Manage Manager**
- **Manage Developer**
- **Manage Project List**
- **View Project Branch wise Details**
- **Expense Management**
- **View Employee attendance and Salary**
- **View all enquires**
- **View Admissions**
- **Assign and View Developer Task**
- **Check Profit and Loss**
- **View Feedback**

5.2 Project Manager Module

The Project Manager module bridges the gap between administrative oversight and on-the-ground project execution. It empowers managers to control, coordinate, and monitor daily operations, ensuring that client expectations and internal timelines align efficiently.Managers are tasked with assigning work to developers, keeping track of project timelines, and ensuring timely delivery of milestones. This module allows them to manage both human resources and technical delivery with equal ease, helping to maintain a high standard of productivity.

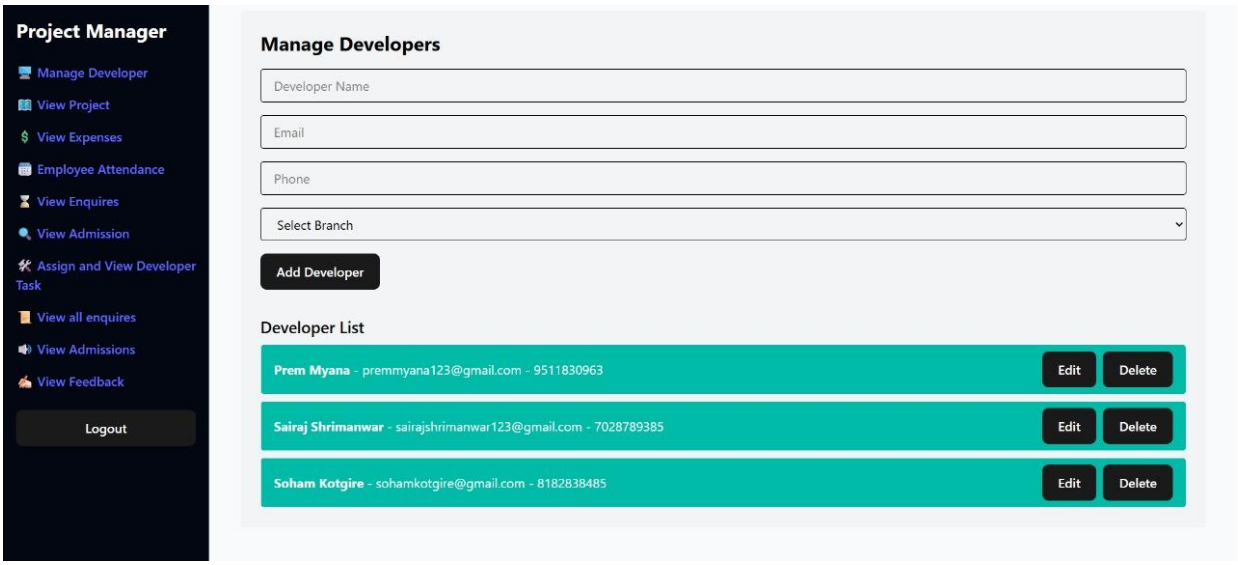


Fig 5.2 Project Manager Home Page

Apart from technical coordination, Project Managers also interact directly with clients and students. They collect feedback, manage communication, and often function as the first point of contact for session planning and query resolution. Their ability to track performance and spot issues early helps mitigate risk.This module is equipped with financial monitoring tools, session planning features, and reporting mechanisms that allow managers to contribute to both operational and strategic decision-making.

- **Manage Developer**
- **View Project**
- **Manage Project List**
- **View Employee Attendance**
- **View Expenses**
- **View enquires**
- **View Admissions**
- **Assign and View Developer Task**
- **View Feedback**

5.3 Developer (Employee) Module

The Developer module caters to internal team members responsible for coding, testing, and delivering project components. It is designed to simplify work organization, task tracking, and performance visibility for employees.

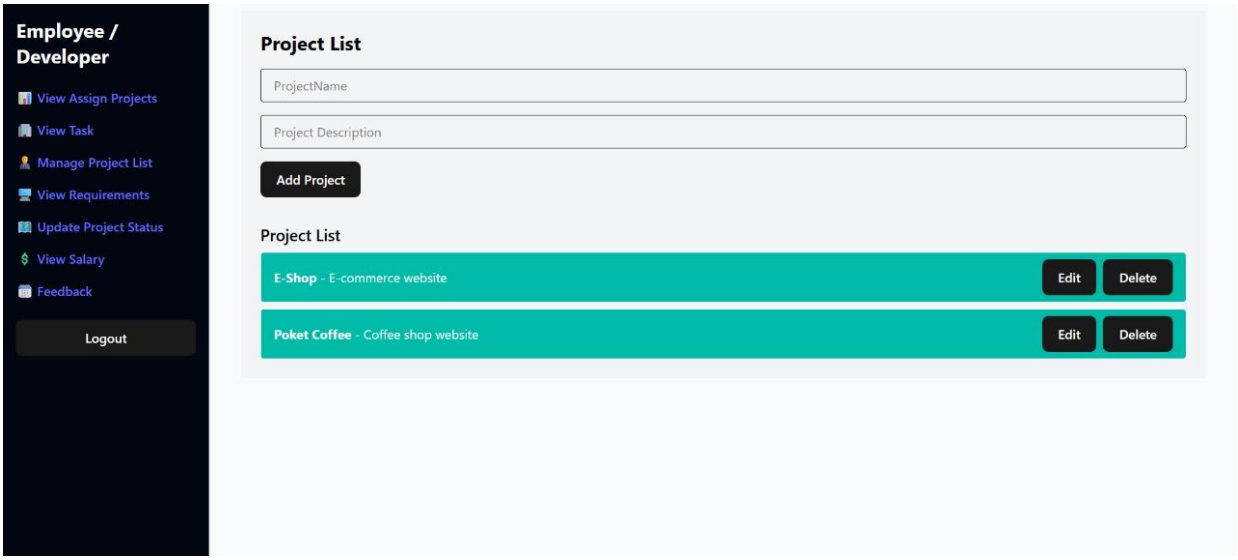


Fig 5.3 Employee/Developer Home Page

Employees use this module daily to interact with their assigned tasks, update progress, track hours worked, and review feedback. The system ensures clarity on project deadlines and simplifies workload management through an intuitive task dashboard.

In addition to task-related features, developers can monitor their attendance records, salary information, and session involvement. This increases transparency and builds trust between management and employees, especially in remote and hybrid work environments.

Developers are also encouraged to contribute feedback on internal processes, helping management improve team engagement and workflow efficiency.

- **View Assigned Projects**
- **Manage Project List**
- **View Task**
- **View Requirements**
- **Update Project status**
- **View Salary**
- **View Feedback**

5.4 Client Module

The Client module serves as the communication and engagement interface for customers or students. It simplifies the process of interacting with the company by giving clients visibility into their project status, timelines, deliverables, and billing.

Clients can log in securely to check updates on their projects, submit new requirements, and participate in feedback loops. The module is especially useful for maintaining customer satisfaction, transparency, and retention.

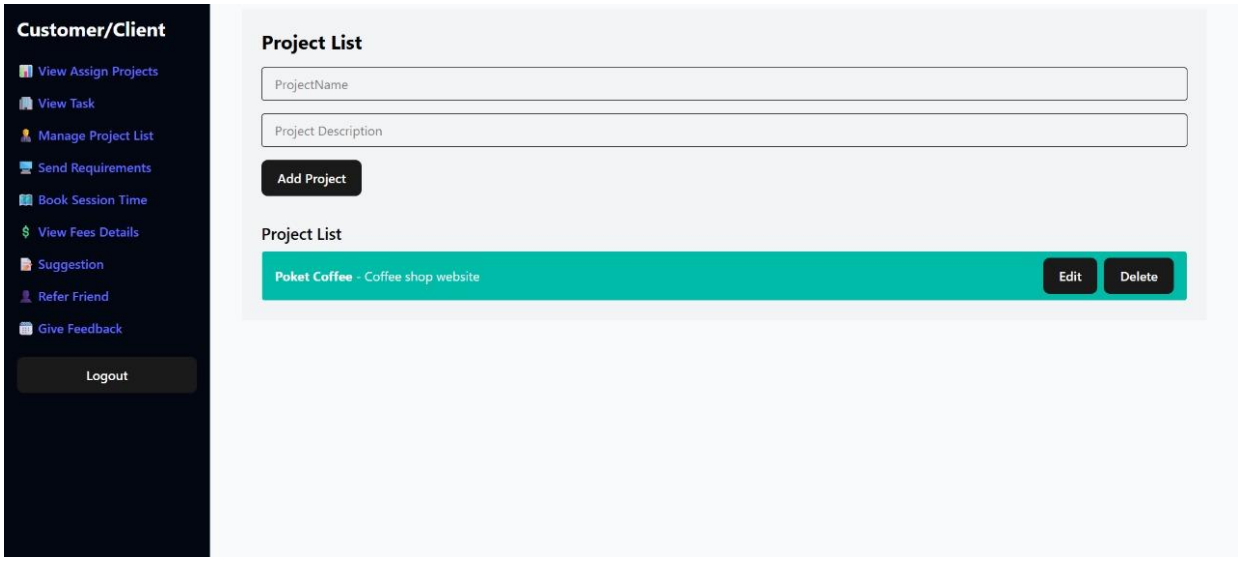


Fig 5.4 Client/Customer Home Page

A major goal of this module is to reduce the need for manual status updates by staff, thereby saving time while improving client experience. Clients can also use this module to book sessions, raise queries, and refer others to the platform.

The interface is designed to be simple, intuitive, and responsive so that non-technical users can easily navigate and utilize its features.

- **Manage Project List**
- **View Assigned Project**
- **Send Requirements**
- **Book Session Time**
- **View Tasks**
- **View Fees details**
- **Suggestion**
- **Refer Friend**

- **Give Feedback**

5.5 Mongo Db

Use of MongoDB in the ERP Management System

MongoDB was chosen as the database for our ERP Management System due to its flexibility, scalability, and ease of integration with the Node.js backend. As a NoSQL database, MongoDB stores data in the form of JSON-like documents, which aligns well with the structure of modern web applications. This made it ideal for handling the dynamic and varied data generated by different user roles such as Admin, Project Manager, Developer, and Client in our ERP system.

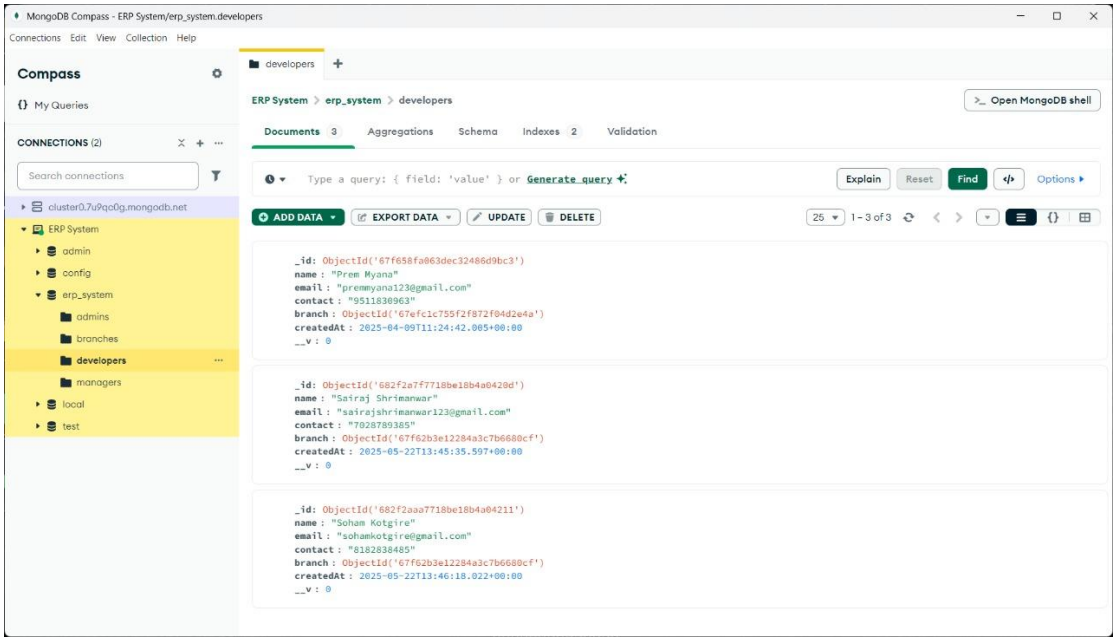


Fig 5.5 MongoDb Database

One of the major benefits of using MongoDB is its schema-less architecture. Unlike relational databases, MongoDB does not require predefined schemas, allowing us to create and modify data models on the fly during development. This was particularly useful when we iteratively refined features and adjusted user roles and permissions throughout the project. Collections such as users, projects, tasks, and feedback were created to organize our data logically, and each document was designed to include only relevant fields, which helped keep the database lean and efficient.

We used MongoDB Atlas, the cloud-hosted version of MongoDB, to deploy and manage our database. Atlas provided us with a secure, scalable, and highly available environment that eliminated the need for manual installation and server maintenance. Its web dashboard allowed us to easily configure clusters, manage access, monitor database performance, and back up our data. Furthermore, it provided integration with

our backend through a secure connection string, which was stored as an environment variable to maintain security.

The integration between MongoDB and our Node.js backend was done using Mongoose, an Object Data Modeling (ODM) library. Mongoose helped us define clear models and validation rules for each collection, ensuring that the data remained consistent even without a rigid schema. It also provided helpful features like population (joining related documents) and middleware support for actions like hashing passwords or validating tokens before saving user data.

Performance-wise, MongoDB proved to be very responsive even when handling simultaneous operations such as task updates, project creation, and role-based queries. As the ERP system involved multiple modules operating in parallel (Admin managing users, Project Managers assigning tasks, Developers updating statuses, etc.), the non-blocking nature of MongoDB's I/O operations complemented our asynchronous backend structure. Queries were optimized using indexed fields, and data fetching was made efficient through pagination and filtering.

In conclusion, MongoDB played a critical role in the success of our ERP Management System by providing a reliable, flexible, and scalable data storage solution. Its document-oriented structure, combined with cloud hosting via MongoDB Atlas and integration through Mongoose, ensured smooth development and deployment of database-related features. The ability to adapt our data models easily, coupled with robust performance and security, made MongoDB the most suitable choice for this project.

CONCLUSION

The development of our ERP Management System was a comprehensive and rewarding experience that blended technical innovation with effective teamwork. Utilizing the MERN stack. MongoDB, Express.js, React.js, and Node.js. we built a full-stack application tailored for four distinct user roles: Admin, Project Manager, Developer, and Client. Each role was equipped with customized access and features to support their specific tasks, from authentication to project and task management. By following Agile methodology and using tools like GitHub, we maintained structured workflows, iterative progress, and strong team coordination throughout the project.

Despite encountering various challenges such as integration issues and tight testing schedules, these obstacles became valuable learning moments. We focused on building secure, reusable components, implemented JWT-based authentication, and managed role-based access control. The deployment process included using Vercel, Render, and MongoDB Atlas, which gave us real-world exposure to cloud services and CI/CD practices. Overall, the project not only met its technical goals but also bridged the gap between academic learning and practical application, giving us deeper insights into enterprise-level architecture and collaborative software development.

REFERENCES

[1] M. Gupta, "MongoDB Atlas: Cloud Database for Modern Applications," Accessed: 28 May 2025.

[2] A. Sharma, "Express.js – Fast, Unopinionated Web Framework for Node.js," Accessed: 27 May 2025.

[3] K. Reddy, "React.js: A JavaScript Library for Building User Interfaces," Accessed: 30 May 2025.

[4] T. Desai, "Node.js: JavaScript Runtime Built on Chrome's V8 Engine," Accessed: 25 May 2025.

[5] R. Verma, "GitHub: Collaborative Platform for Code Hosting and Version Control," Accessed: 26 May 2025.

[6] P. Nair, "Vercel: Frontend Cloud Platform for Deploying React Apps," Accessed: 1 June 2025.

[7] S. Mehta, "Render: Backend Hosting for Web Services and APIs," Accessed: 29 May 2025.

[8] L. Das, "Agile Methodology in Software Development: Principles and Practices," Accessed: 31 May 2025.

[9] J. Thomas, "Continuous Integration and Deployment (CI/CD) for Web Projects," Accessed: 2 June 2025.

[10] D. Rao, "Role-Based Access Control in Web Applications," Accessed: 3 June 2025.