EMPLOYEES TASK MANAGEMENT TOOL

**A PROJECT REPORT**

**Submitted in partial fulfilment of the requirement for the award of the degree**

**of**

**MASTER OF COMPUTER APPLICATIONS (MCA)**

by

**Rupa Kumari**

**23FS20MCA00062**



**Under the guidance of**

**Dr. Arpana sinhal**

**Department of Computer Applications**

**Faculty of Science, Technology and Architecture**

MANIPAL UNIVERSITY JAIPUR, JAIPUR-303007

**Rajasthan, INDIA**

**2025**

**DEPARTMENT OF COMPUTER APPLICATIONS**

MANIPAL UNIVERSITY JAIPUR, JAIPUR – 303 007 (RAJASTHAN), INDIA

Date: 10 May 2025

**CERTIFICATE**

This is to certify that the project titled EMPLOYEE TASK MANAGEMENT TOOL is a record of the Bonafide work done by RUPA KUMARI (23FS20MCA00062) submitted in partial fulfilment of the requirements for the award of the Degree of Master of Computer Applications (MCA) in **Department of Computer Applications** of Manipal University Jaipur, during the academic year 2024-25.

**Dr. Arpana Sinhal**

Project Guide, Dept of Computer Applications

Manipal University Jaipur

**Dr. Shilpa Sharma**

HOD, Dept of Computer Applications

Manipal University Jaipur

(On company letterhead)

Date:

**CERTIFICATE**

This is to certify that the project entitled EMPLOYEE TASK MANAGEMENT TOOL was carried

out by RUPA KUMARI (23FS20MCA00062) at GemsNY IT Solutions, JAIPUR under my guidance

during January 2025 to May 2025.

**Navneet Bhardwaj**

Project Manager,

GemsNY IT Solutions

**ACKNOWLEDGMENTS**

I wish to extend my thanks and appreciation to the **Dean of Faculty of Computer Applications** and the **Director of Manipal University Jaipur** for allowing me to carry out this project.

I sincerely thank **Dr. Shilpa Sharma**, **Head of Department, Computer Applications**, for providing me continuous encouragement and support throughout this project.

I am sincerely thankful to my **internal guide**, **Dr. Arpana Sinhal**, for her guidance, constant support, and helpful insight through the course of this project.

I would like to express my heartfelt gratitude to **Project Coordinator, Dr. Amit Hirawat**, for his valuable suggestions, guidance, and constant support during the project work.

I am very thankful to my **Project Supervisor**, **Navneet Bhardwaj**, for his guidance, continuous support, and patience throughout the development of the entire project.

My thanks also go out the team and mentors of **GemsNY IT Solutions** and **[Ratnesh Gupta]** for their insight and allowing me to work on real-time applications during my internship.

I would like to extend special thanks to the **Laboratory In charge**, the **faculty members**, and **technical staff** of the department for their assistance which made it easier to proceed with the project.

Last but not least, I also thank **my family** and **friends** for constantly motivating me and giving support during the period of developing this project.

Date: 10.05.2025 Rupa Kumari  
Place: Manipal University Jaipur

**ABSTRACT**

In the modern workplace, efficient management of employee tasks has emerged as a critical factor in organizational success. Most manual methods of task tracking typically will miscommunication, delayed updates and lack of responsibility. The Employee Task Management (ETM) Tool was conceived and developed with the first goal of streamlining the processes of assigning, tracking, and completing tasks for an organization.

The development process involved identifying the specific needs of three user roles: Admin, Assigner, and Assignee. The system was developed using a component and modular development structure in Next.js, allowing for extensibility and maintainability. Admins can control everything about the system, such as user management, and monitoring tasks. Assigners can create, assign, and monitor tasks assigned to assignees. Assignees will have access to a customized dashboard to see their own tasks, as well update status in real-time.

The dashboard serves as the central interaction point of the Employee Task Management Tool, organized by task status - Unassigned, In Progress, Pending, and Completed - consumers can easily manage tasks. The dashboard features filtering options for tasks by employees or departments to make it easier for users to focus on single tasks or workgroup teams. The Parent-Child Task feature gives consumer the ability to associate a collection of tasks as part of a related task, thus facilitating the management of complex work projects involving multiple connected tasks.

The user interface (UI) created with Next.js and TypeScript providing a fast and maintainable procedure for interaction. To achieve this, utility-first styling was applied by using Tailwind CSS. Material UI (MUI) was integrated to operate a design system which is modern, consistent, and accessible. The team developed the backend using Express.js, Node.js, and MySQL. I have used MUI to design the entire table of front end of the application by integrating the front end of our design into a working application. I was responsible for fetching data via APIs and displaying it dynamically. I also created communication points between the front end and backend to establish functions for task management and task maintenance.

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Table No** | **Table Title** | **Page No** |
| **5.1** | Comparison with Existing Systems | 36 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No** | **Figure Title** | **Page No** |
| **3.1** | Data Flow Diagram | 11 |
| **3.2** | ER Diagram | 13 |
| **5.1** | Login Page | 29 |
| **5.2** | Admin Dashboard | 29 |
| **5.3** | Assigner Dashboard | 30 |
| **5.4** | Assignee Dashboard | 30 |
| **5.5** | Profile view | 31 |
| **5.6** | Employee’s View, Add and Edit | 31 |
| **5.7** | Add Task | 32 |
| **5.8** | Department’s View, Add and Edit | 32 |
| **5.9** | Permissions | 33 |
| **5.10** | Roles | 33 |
| **5.11** | My Task page | 34 |
| **5.12** | Task Report page | 34 |
| **5.13** | Send Mail to the employees | 35 |
| **5.14** | ToDo list | 35 |

**CONTENTS**

|  |  |
| --- | --- |
| **CONTENT** | **PAGE NUMBER** |
| Acknowledgement | **i** |
| Abstract | **ii** |
| List Of Figures | **iii** |
| List Of Tables | **iii** |
| Introduction  * 1. Introduction to work done/ Motivation (Overview, Applications & Advantages)   2. Project Statement / Objectives of the Project   3. Organization of Report | **1-3** |
| 1. **Background Material**    1. Conceptual Overview (Concepts/ Theory used)    2. Technologies Involved    3. Existing Systems / Literature Review    4. Feasibility Study    5. System Requirements | **4-9** |
| 1. **Methodology**    1. Detailed methodology that will be adopted    2. Data Flow Diagrams (DFD)    3. Entity Relationship Diagram (ERD) | **9-15** |
| 1. **Implementation**    1. Modules    2. Prototype    3. User Roles and Access Levels    4. Testing Strategy / Test Cases    5. Deployment and Hosting | **16-24** |
| 1. **Results And Analysis**    1. Performance Evaluation    2. Screenshots of Working System    3. Comparison with Existing Systems    4. Coding Section | **25-39** |
| 1. **Conclusions & Future Scope**    1. Conclusions    2. Future Scope of Work | **40-42** |
| **REFERENCES** | **43** |

1. **INTRODUCTION**

In the modern business environment, it has become critical to be able to assign and track tasks in any workplace, regardless of the type of work being performed. Companies are no longer simply hoping that employees are doing their work, but have realized that measurable productivity is necessary to hold employees accountable. As managing tasks across multiple teams and departments can often lead to confusion, missed deadlines, and an inability to track employee productivity, many organizations are making the shift to online task management solutions, where all operations may be centralized, tracked in real-time, and open communication is available to all involved.

The Employee Task Management (ETM) Tool, which is an online application, can manage the entire life-cycle of task assignments and tasks performed by employees in an organization. Using ETMS permits supervisors and administrators to assign tasks based on departments, priority, and deadlines. Employees can track assigned tasks and provide updates, including hours worked, expand tasks, or "not start" tasks when applicable, and manage their hours worked. ETM Tool is based on operation roles, so all employees can view, manage, and report on tasks, where their level of access is determined by their role in the organization. By using ETMS, we designed a simple, easy-to-navigate user experience for the employee and structured the application so the employees can be successful in managing their hours assisted by the supervisor.

* 1. ***Introduction to work done/ Motivation (Overview, Applications & Advantages)***

In the evolving landscape of modern nimble organizations, booking your tasks and tracking progress is a must for determining the success of any project. Large corporate offices with multiple departmental projects, small startups managing multiple internal projects, and remote teams trying to collaborate across different time zones (exhibit A: our experience at WSP), share a common challenge with scheduling and completion—ensuring that every task is assigned, tracked, and stays on deadline. This commonality of struggle is the backdrop for why the Employee Task Management (ETM) Tool was created.

The idea was birthed from a very real issue: organizations did not have an effective way to assign tasks, monitor workloads, in as close to real-time as possible. Typically, managers utilized emails, verbal requests, or organized unexpected multi-sheet excel spreadsheets to capture who was doing what (forgetting 'who' was impossible). Obviously, that inefficient method would not capture work in-progress, cause issues with miscommunication, neglect fulfilment of forgotten requests, and create zero accountability. Teams who were successfully working remotely or inter-departments for a task, discussion, or project, were left barely able to coordinate work if we did not formalize an organized structure to guide work. With those struggles at the front of our minds, we imagined how a robust digital employee task management tool could create a to-do list, provide an organized method for tracking progress and completion for all participants—including administrators, managers, and employees—all while providing a similar context for every task.

* 1. ***Project Statement / Objectives of the Project***

Every successful organization operates well with clarity, coordination, and accountability. However, as teams grow, and workflows span departments, it is often difficult to maintain that clarity. This struggle was the genesis of the Employee Task Management (ETM) Tool. When we consulted with our stakeholder, it was clear that a project was necessary to ease, organize, and modernize task assignment and task tracking in ways that had not previously been done in their organization.

The project served as an initial goal to create a digital environment in which all employees, regardless of their department or position, could operate in the same framework. We hoped to make task assignment easy for managers, and simultaneously give employees a structured way to understand their responsibilities, how to update the status of their work, and be accountable—all in real-time.

The project statement was simple and clear:

To design and develop a web-based Employee Task Management Tool that features role-based assignment of work, allows live tracking of status, estimates time requirements, logs progress, and produces performance reports, accomplishing this without redesigning how an organization completes and tracks work.

From this concept, we identified a number of concrete objectives.  First, we wanted the system to be role-based. We wanted types of users, such as administrators, assigners (managers), and assignees (employees) to interact with the system as the community user.  Each user would only see what was relative to their role within the system, providing clarity and security.

Our first goal was to create functionality that allowed managers to assign tasks along with rich metadata that provided context for the tasks.  Our aim was for managers to be able to assign work along with the department, priority, start date, end date, estimated completion time, and notes.  This way, when a task was assigned to an employee, the task had full context.

A focus of our development was on real-time tracking and feedback.  Employees needed to be able to mark their tasks as pending, in-progress, or completed the actual time it took to complete the task, in addition to comments or notes for their manager.  Supervisors would be able to monitor all of this using a clean dashboard that visually represented both department and individual progress.

In addition to the core workflow, the purpose of the system was to produce reports and visual summaries for leadership teams to use to uncover trends in productivity, gaps in performance, and areas that require additional support. With its use of automation to gather and track data as opposed to manual methods, the system aims to reduce error and improve decision making.

Ultimately, the goal was to build more than just a nice task manager - it was to build a digital assistant that connects, cultivates accountability, and increases productivity for the entire organization. Whether small startup or large enterprise, a vision of ETMS was for it to grow and evolve with the needs of the team it supports.

* 1. ***Organization of Report***

The Employee Task Management Tool has traveled a long way from inception to final development, and in order to provide an orderly approach to providing the development process in this final report, each chapter offers a phase of the project throughout its development. In this report, there are separate chapters to portray each stage in the life of the project from the initial idea of the system through to implementation and evaluation. The intent of the approach taken was to take the reader step-by-step through the rationale, planning, designing and developing to completion of this system.

The first chapter, Introduction, introduces the reader to the impetus behind the project, and specifies the problem that the system attempted to address, and the origins of the objectives to determine the attributes and functionality within the system. The full chapter also incorporates a description of the report components that made up the entire report so the reader understands how the project report summaries each of the chapters that proceed in the report.

Next, the Survey of Technologies are descriptions of the tools, frameworks and technologies that we were considering when developing the ETMS, and the tools that were selected as part of the ETMS development process. It also explains the reasoning behind selecting the tools, including React for the front-end, Node.js for the back-end, and certain databases like MongoDB or MySQL for data storage, and how the tools are to build components identified by the plan to provide performance and scale.

Difference between requirement analysis and design the next chapter, Requirement Analysis, looks at the system's functional and non-functional requirements, as well as the needs of the users (roles) across three of the user roles: Admin, Assigner, Assignee, and describes their expected inputs and outputs and boundary constraints. The requirement analysis set a foundation that led to design decisions throughout the other chapters.

System Design outlines the architectural structure used for the ETMS. This chapter provides data flow diagrams (DFDs), entity-relationship (ER) diagrams, and diagrams that break down components of the ETMS and illustrate in detail how the pieces fit together. This section also concludes that development work was conducted with a plan that used foresight.

Next, the Coding chapter specifically explains the technical implementation of the system. Key elements of the codebase will be highlighted with examples to demonstrate how the logic is working, and breakdowns will help demonstrate how the theoretical design was subsequently translated into a usable application. Challenges that were encountered during development implementation will be noted.

Following the coding chapter, the Testing section describes how the system was tested against the requirements previously defined. There are multiple levels of tests that occurred including a variety of levels: unit tests, integration tests, and system level tests and this noted the intended reliability, security, and usability when the platform is in use.

The report then proceeds with Results and Discussion, in which the outcomes of the project will be discussed. There are performance indicators, screenshots of the working application, and user feedback, if applicable. In this section of the report a thorough evaluation of the ETMS would be critically reviewed to identify if the aims of the project were achieved.

Then in the last sections, Conclusion and Future Scope, we can reflect on what has been achieved with the project and consider future improvements, upgrades and features that could be integrated into future versions. These sections demonstrate the limitations of the current system but allow us to remain positive about future development.

The report fulfils this organization structure because it outlines a journey to build the Employee Task Management Tool and allows for reading clarity of readers from a technical review perspective, as well as an academic review perspective, while ensuring the relevant context and insights comply to support the development.

1. **BACKGROUND MATERIAL**

Creating any successful software application is not just a technical function. It is a journey that begins with identifying and understanding the real-world problem; then it enters the realm of theories, focuses on the use of technologies, and confirms it is practical only after a repetitive exploratory explanation. This chapter kick starts the start of the Employee Task Management (ETM) Tool and will cover the underlying concepts, technologies, existing systems, feasibility, and requirements considered in the eventual design and development of a systematics on ETMS.

In a day and age where several teams are involved in common goals or an employees’ roles are shared, employee task handing is necessary, and every organization that has implemented agile and lean will agree that effective task management is a challenge. Most employees today are assigned tasks by way of emails, chat, or verbal instructions. This role cluttering processes can lead to confusion, missed deadlines, and ultimately wasted time and lack of accountability. ETMS identifies this broader context and seeks to improve the task management process in an evolving and increasing digital environment.

* 1. ***Conceptual Overview (Concepts/ Theory used)***

The ETM framework consists of several generally accepted theories and established software development principles. Centered in the framework is the concept of Task Lifecycle Management, which is the backbone of the system. Each task is assigned, tracked, updated, and eventually completed – once the digital system can represent the lifecycle of a task, the task management process is visible and organized.

One supporting theoretical idea employed is Role-Based Access Control (RBAC) model. This model grants every user (Admin, Assigner, and Assignee) access to only what their role requires, limited to the task at hand and limited to competencies associated with his or her role. While it improves security by limiting access, it makes the user's experience easier by removing ambiguity, and clutter in the interface.

The principle of Client-Server Architecture is another principle at play. The front-end (client) communicates with the back-end (server) through API calls. This separation of functions creates the option to develop and scale components of the application independently. The CRUD model (Create, Read, Update, and Delete) is used application-wide to manage task information.

The system adheres to Database Normalization principles to eliminate redundancy, and preserve meaningful relational integrity in the process so that the database elements remain clean and consistent. By aligning all these principles and foundational theories, ETMS is able to deliver a complete, scalable, and user-focused application that will help organizations maintain task management.

* 1. ***Technologies Involved***

When developing a software system, one of the most important decisions is the selection of technologies. The choice of technologies influences the performance, reliability, scalability, maintainability and usability of the system. In order for the Employee Task Management (ETM) Tool to successfully handle and manage tasks through its interface along with real time management of data, it has utilized a cutting-edge scalable web development technology stack.

1. ***Front-End Technologies***

The ETMS front-end is built using React.js/Next.js, which is an open-source JavaScript library for building user interfaces. React has a component-based structure that allows the UI to be modular, maintainable, and reusable. When using React, the UI can be very fast because it uses a virtual DOM and does most of its re-renders intelligently rather than by the traditional method.

To improve the UI, and to keep design consistent and minimal, we used Material-UI (MUI) and Tailwind CSS. Material-UI is a component library that consists of customizable pre-styled components following modern design guidance, and Tailwind CSS is a utility-first CSS framework that is useful for styling sleek and intuitive designs. React Icons were used to provide icons and visual cues to improve clarity and interaction in user dashboards.

***Features Implemented:***

React Hooks (useState, useEffect) for life cycle and state management.

React Router for page navigation.

MUI Table and Inputs for interactive data presentation.

Tailwind for styling flexibility for dark/light mode.

1. ***Back-End Technologies***

The back-end logic and API handling is built using Node.js and Express.js. Node.js is a runtime environment that runs JavaScript on the server side, while Express.js is a lightweight web framework that provides RESTful APIs to communicate with the front end.

This framework provides a way to exchange data between the client and server efficiently, which allows for asynchronous task updates, status changes, and secure role-based access control. Express.js uses middleware, or functions that are invoked, for logging, authentication, and request validation.

***Key Highlights:***

RESTful API structure to allow clean and structured communication.

JSON data interchange means responses can be light and easy to read.

Role-based authentication and access control through middleware.

1. ***Database Technologies***

For data persistence, MySQL is employed. MySQL is a dependable, relational database management system (RDBMS) that allows for structured data to be stored in a manner that upholds integrity and consistency. It can manage tables for employees, tasks, departments, roles, time tracking, and statuses of tasks.

The system contains:

Primary and Foreign keys for managing employee-task-department mappings.

Stored procedures and joins to optimize on the frequent queries.

A normalized schema to reduce redundancy and ensure data integrity.

***Other Tools and Libraries***

* Postman: For testing the API to ensure endpoints responded correctly and with expected
* data.
* Git & GitHub: Version control tools for managing code collaboration and backups.
* VS Code: Development environment with plugins for linting, auto-formatting, and
* Debugging.
* .env: To manage environment variables such as database credentials, api URLs, and secret keys.

1. ***Hosting & Deployment***

The Employee Task Management (ETM) Tool is live and currently available via a

live production link: [***http://15.204.213.25:6001/***](http://15.204.213.25:6001/). The ETMS deployment on a Virtual Private Server (VPS) environment allows for more flexibility, control, and reliability than generally available with shared hosting services.

Both the front-end and back-end of the application are served from a VPS environment and this helps to reduce latency and optimize communication between server and client. The self-hosted platform allows for independent port configurations, environmental settings, and backend optimizations.

* 1. ***Existing Systems / Literature Review***

Prior to struggling to design the ETM Tool, it was important to survey the field of active task management tools and assess the academic and industrial views on digital productivity platforms. A ubiquitous involvement of tools, like Trello, Asana, Jira, have been established in relation to "task tracking" for organizations of all sizes, including start-ups and enterprises. The active employee task management tool using "boards" and "task lists" and "calendars..." mainly address the task management needs of project management and agile roles in departments, but did not address the wants of organizations with internal task delegation within departmental boundaries.

The general pattern was the lack of nuance with regards to roles of employees tied to the organizational hierarchies. Most digital productivity systems assume a flat access structural similar to social media. In a flat structure, all users have fairly equal access to tasks and as a result data, hence visibility is lost in accountability with hierarchical organizations. The excessive lack of consideration of roles from these active task management organizations was a primary motivator behind developing a role sensitive, department aware task management platform, that adheres to how teams are structured in the real world.

Literature regarding employee productivity and organizational digital transformation suggests that task transparency, accountability, and measurable KPIs contribute to maximizing operational efficiency. Journals and white papers consistently highlight the performance improvements that are achievable when systems are designed building on the organization's workflows (not simply based on opportunities for project improvement as it is generally recognized).

The purpose of the ETM Tool is grounded in the discovery of commercial platforms and theoretical research to develop an individualized employee-level task management Tool. It is not a project management tool that informs management about progress or what has been produced - it records department-specific roles, provides a daily living progress log of tasks, and it records how well performance according to key performance indicators, which positions it as an internal system.

* 1. ***Feasibility Study***

Before commencing on the ETM Tool, a preliminary examination of the conditions for feasibility was imperative. A structured feasibility study was undertaken to find out if the system could be successfully designed, implemented, and used in a typical organization. Condition parameters were examined within three categories: technical, operational, and economic feasibility.

* 1. ***Technical Feasibility***

In terms of technical feasibility, the ETM Tool was fully possible based on existing open-source technologies that were current in terms of context and widely used. For example, React.js and Next.js were used to create an interactive user interface that was also fast and highly scalable. Similarly, the ETM Tool used Node.js and Nest.js as a platform to deliver API logic for user roles and to organize task management in a secure environment. These technologies also had a large community of support, which would only contribute positively to the development of the ETM Tool, as they both contained an impressive number of libraries, extensions, and tools to streamline development as much as possible. Additionally, the component means that upgrades in the future, such as real-time notifications, calendar applications, or advanced analytics, could be included with minimal disruption. The entire ETM Tool was designed to run on a SQL-based relational database, which is a safe, structured, and competent platform for mapping complex organizational data.

* 1. ***Operational Feasibility***

The ETM Tool was developed with actual organizational use cases in mind. Most organizations start small with employees and scale up into departments, and some sort of hierarchy. The ETM Tool is structured such that it accommodates role-based access and department filtering of tasks, and therefore it could fit nicely with departments and teams, as teams would generally use the ETM Tool

in a way that reflects how they conduct their work. The ETM Tool needed to align with the daily deliverables of admins, managers, and employees without requiring users to fundamentally change their workflows. The workflow structure also allows users to have access based on their role as manager or employee. A clean UI, intuitive task dashboards and little to no technical training needed for day to day use also made onboarding easy. Employees understand fairly quickly what it means to view, update and track their tasks, while managers can track other people’s progress and generate reports quickly. Structuring ETM to align with actual workflows meant that adoption and effective ongoing use wouldn’t be a problem.

* 1. ***Economic Feasibility***

Geared towards financial savings, the ETM Tool presented great financial opportunities. The project was not responsible to pay licensing fees because it used only free and open-source software development on the front end and the back end. There were no on-going costs with proprietary tools and the development server could be used using the organization’s existing infrastructure or a cheap virtual private server. The costs that needed to be spent on hardware or other cloud-hosting enabled this project to be economical. More importantly, the amount of return on investment became clearer when the long-term productivity injection was incorporated into the summary. Essentially, the system provides members of an organization with clear accountability for tasks assigned, performance monitored, and sufficient time logging for an organization to lessen deadlines missed, communications not missed, and overall efficiencies could be enhanced and realized. These operational efficiencies produce actual economic value, and that makes the ETM Tool not only feasible, but strategically feasible.

* 1. ***System Requirements***

Creating a more comprehensive internet-based solution like the ETM Tool requires a thorough understanding of the requirements of the underlying system. Not just technical specifications, but the requirements set up the foundation of the building blocks of an application to follow to guarantee the application runs correctly, stays scalable, and maintains a user satisfying experience across a variety of device types and network connectivity. Criteria can be grouped into 3 broad categories; Hardware, Software, and User requirements.

* 1. ***Hardware Requirements***

To assure running performance for both development and production environments, it was necessary to assess the minimum and recommend hardware resources.

With respect to client-side, end-users to the ETM Tool include employees, manager, and administrator. Users will not have any hardware limitations due to the ETM Tool in a web browser; therefore, hardware requirements are minimal. A computer with an Intel i3 processor or higher and 4 GB of RAM (preferably 8 GB for multitasking) is sufficient to access and use the application. The ETM Tool suggests a display resolution of 1280×720 pixels as minimum requirement to ensure that all application UI updated rendering platforms render properly, particularly for the specific task dashboards. Additionally, users are suggested to have a reasonable broadband internet connectivity rate to receive updates each time the task updates, and interact with the ETM server.

On the server side, where the backend APIs, the database, and business logic are located, there are slightly higher infrastructure requirements. A quad-core processor, 8 GB of RAM (16 GB is recommended for heavy traffic), and an SSD with a minimum of 50 GB of available storage are ideal. Storage requirements could scale with the organization's data size. To add to this, it is important that there is a high-speed Ethernet connection in place to allow low-latency API interactions; a static IP is preferred where consistent deployment is necessary. It is highly advised to have an Uninterruptible Power Supply (UPS) or similar redundancy in either a cloud or on-premise use case to account for unexpected power outages and network interruptions.

* 1. ***Software Requirements***

On the software side, the ETM Tool uses all modern technologies that are both flexible and scalable.

The front-end is built using React.js, with a Next.js framework providing server-side rendering (SSR), routing, and system performance improvements, which lead to fast-load times and SEO compatibility. The UI/system is styled using Tailwind CSS, which provides utility-first classes to allow intuitive layout. There are additional enhancements provided by MUI components and React Icons for a cleaner and more user-friendly experience.

The backend combines Node.js with Nest.js which is a useful and modular framework for structuring and scaling server-side code. The RESTful APIs on the backend provide access to the data logic and validation and the authentication. The system uses one of two SQL-based relational databases (PostgreSQL or MySQL) with an ORM (TypeORM) acting as the intermediary between the application logic and database logic, thereby easily handling relationships between data, migrations, and running queries. All authentication security is done by using JWT (JSON Web Tokens) for authentication and bcrypt for hashing passwords.

The backend and the frontend servers will be hosted in a Linux environment (ideally specifying to be on Ubuntu 20.04 LTS). A web server, like Nginx, is acting as a reverse proxy to serve the frontend and route requests to the APIs. Code source management and team collaboration will be done using Git to manage and host code on GitHub or GitLab. To monitor the production instance and manage the processes, I use PM2 or Docker. To back up the database, a cron job is scheduled regularly to back up the database for data integrity and disaster recovery reasons.

* 1. ***User Requirements***

Because the ETM Tool will have different users—Admins, Assigners, and Assignees—this design allows users will have a unique experience depending on their role while ensuring simplicity and accessibility.

Admins will have the responsibility of managing users, roles, departments, permissions, and reports. Admins will want to have clear and intuitive dashboards and control panels, to allow for a more comprehensive and high-level view of the organization's task loading, assignment, and completion status.

Assigners (usually team leads or managers) will want an interface that allows them to quickly create, edit, and assign tasks. They want the ability to assign deadlines and priorities to tasks, and be able to monitor it in progress over time.

Assignees, or employees, mostly interact with their assigned tasks. Their interface allows them to display pending and completed tasks/statuses (Pending, In Progress, Completed), leave comments, log the actual date of completion, and receive feedback/ratings based on performance.

Because it is web-based, we don't have to add on to any device other than what's reasonable for the end user (as long as they use a modern browser such as Chrome, Firefox, Edge, or Safari).

From a security perspective, the system uses role-based access control (RBAC) to eliminate the ability for users to see and/or interact with data not related to their role; strong activity logging and encrypted authentication models create accountability and protections for users' data.

1. **METHODOLOGY**
   1. ***Detailed methodology that will be adopted***

The creation of the Employee Task Management (ETM) Tool followed a structured and systematic process to create clarity that could be replicated for scalability and efficiencies. The Agile Software Development Model was followed addressing the developing of the ETM, which meant going through iterative progress, reviewing, and being flexible throughout the software lifecycle. This approach allowed us to chunk the entire project into a manageable state, and work on each independent module in an incremental fashion through "sprints".

The very first thing we did was requirement gathering, going through multiple rounds of engaging with end users such as managers, task assigners, and employees, to determine the exact pain that those manual tasks had on productivity of time. This feedback was turned into functional requirements and non-functional requirements. After these were detailed and finalized, then we moved to the design phase of the system. This encompassed planning the database schema, architecture of the application and the user interface flow to create screen wireframes and flowchart diagrams.

Frontend development was accomplished with React.js and Next.js, which allowed us to have both client-side interactivity, and server-side rendering. The backend APIs were developed with Node.js with Nest.js, which provides a modular, scalable structure and good compatibility with SQL databases. The SQL database itself was built with an efficient design for storing tasks, user roles, time logs, and departments, complete with constraints and relationships that ensured data integrity.

Each discrete major module — user authentication, task assignment, task tracking, department-role mapping, and reporting — was designed, developed, and tested in isolation then combined and tested for User Acceptance Testing. Following this process, extensive debugging, performance tuning, and UI fine-tuning rounded out the development process to try and ensure that the system would operate as expected when used in the real world.

* 1. ***Data Flow Diagrams (DFD)***

The Task Reporting System keeps track of tasks, employees, departments, and employee roles. Employee Task Management Tool allows users to create, assign, track, and report on tasks with different roles: Admin, Assigner, and Assignee. The roles are all different, and the data flows through the ETM Tool according to their role and the work and processes they have started. The below story explains how data flows through the different processes in the ETM Tool.

|  |
| --- |
|  |
| **Fig. 3.1** |

***1*. *The Assigner*** - Task Creation and Assignment

The Assigner is responsible for creating and assigning new tasks to Assignees.

1. ***Task Creation:*** The Assigner logs into the dashboard and goes to the Add Task section. The Assigner enters the task details, which include Task Name, Task Description and Priority.

* The data entered consists of the Task Name, Task Description, Priority and Assignee.
* The system will take that data and enter it into the Task Database.
* The task is linked to an employee (an Assignee), so the system will also update the Employee Database.

***Data Flow:***

Assigner enters task details → System stores data in Task Database → Updates to Employee Database.

1. ***Assignment of Tasks:*** Once a task has been created, the Assigner assigns it to an employee within a department (the Assignee). This assignment means the task is viewable to the Assignee under the dashboard in relation to all assigned items to an employee.

***Data Flow:***

Assigner assigns task → System links task to Assignee → Task is displayed on Assignee’s dashboard.

***2. The Assignee:*** Task Progress and Status Updates

It is the responsibility of the Assignee to work on the task and provide status updates.

1. ***Task Viewing:*** After a task has been created and assigned, the assignee (the initiator) will view the task in the My Tasks section. Here, the Assignee will see all details regarding their task; including task name, priority, and assigned date.
2. ***Task Updates:*** After the Assignee has viewed their tasks, and begins working on it, they will update the status of the task (e.g., in-progress or completed) and log time-to-complete the task (estimated time and actual time). The system takes the information and processes these updates in the task database.

Upon successful completion of the task, the system will mark the task as completed.

***Data Flow:***

Assignee updates task status → System updates Task Database with new status → Assignee logs time → System records time in the Task Database.

***3. The Admin's Duties -*** Task Supervision and Reports

It is the admin’s responsibility to supervise all tasks, generate reports, and maintain roles and permission management.

1. ***Role Management:*** The Admin manages the role configuration and user permissions for all users. The admin can create and manage roles such as Admin, Assigner, and Assignee. The admin can define permissions in granular levels based on the role. The role allows the system to enforce permissions so that a user only has access to what is relevant to their roles and permissions.

* Data Flow:
* Admin assigns roles and permissions → System saves in Role and Permissions Database → The system will enforce access restrictions based on roles/permission setup.

1. ***Task Monitoring:*** The Admin can view tasks in all departments, including task progression. The admin is able to view and update task status, progress and view time logs from the Task Database. The admin can generate reports that include completion, time related to task, and to what degree the task is pending.

* Data Flow:
* Admin makes report request → System queries task information from the Task Database → Admin views report or exports into Excel or PDF.

1. ***Report Generation:*** The Admin is able to generate reports on task and task progress, those reports will show department progression, time tracking, and department performance. The system will allow the admin to filter based on objects task by date, status, and department.

* Data Flow:
* Admin generates report → System gathers data from Task Database + Employee Database → Admin views or exports report.

***4. The Data Stores -*** Where is the information held?

There are three main data stores with which the system interacts, to hold and retrieve data.

1. ***Task Database:*** This stores all information about the tasks, including task names, task descriptions, task statuses, task assignees, and task time tracking.

* Task Information is created and modified here when new tasks are created and when the Assigner and Assignee log their work.

1. ***Employee Database:*** This stores bit of employee information, including employee names, employee roles, employee departments, and employee contact details. Additionally, this is the place where task assignments in the system are associated with individual employees.

* The Employee Database is updated when tasks are assigned to determine which employee is responsible for the task.

1. ***Role and Permissions Database:*** This generally stores all the roles and permissions to users of the system. Rights for each user (Admin, Assigner, or Assignee), are stored here to define the sections of the system that the user can access.
   1. ***Entity Relationship Diagram (ERD)***

An ERD for an Employee Task Management Tool illustrates the relationship between all entities in an Employee Task Management Tool. Entities in an Employee Task Management Tool include Tasks and references to Employees, Departments, Roles, and Permissions. The diagram is useful in demonstrating task management and related phenomenon that include assignment, tracking, and reporting.

|  |
| --- |
|  |
| **Fig. 3.2** |

***Entities and Their Relationships***

***Task Entity*** – The Core of the System

Within any Employee Task Management Tool, the fundamental entity is the Task. The Task is the central component in an Employee Task Management Tool: it is the essence of the program. The Task digitally represents a real-life task necessary for an employee to complete. Each task has its own identification (ID), priority, goal/purpose, due date, and ownership.

Picture this scenario: you have a dashboard that populates with headings like "Unassigned Tasks", "In Progress", "Completed", etc. Each of those fields is coupled with a record—in your database—of the detailed Task object. Let's examine what constitutes the Task entity and the importance of each component.

***Components of the Task Entity***

***Task ID (Primary Key)***

Each task created will automatically generate a unique identifier. Task IDs give you a reference, filter, and the ability to manage tasks without confusion, despite having similar titles.

***Task Name***

The task name provides a quick and meaningful label for the employee to understand what the task is about—for example, "UI Design" or "API Integration".

***Task Description***

The Assigner or Admin writes a description on what is to be done in this field. This description may include goals, expectations, instructions, or links to source documents.

***Priority***

Every action item in the system has a priority level attached to it to show how urgent it is to act on it. This helps both assigners and assignees plan their workloads in appropriate order and gives ways to let others within the organization know how urgent a task is. Priority is set on a level from 1 to 5, being 5 as the default value (when the assigner does not specify). For example:

***1 – Critical (Highest Priority)***

These actions must be completed immediately – assigned tasks that are priority number one have highest priority to the assignee because they have definite impact on project timeline or imminent deliveries. No time can be lost here.

***2 – High***

These actions are important and are time sensitive. While not critical, they must be scheduled immediately after Critical 1 Assignments.

***3 – Medium***

These actions should be completed within a reasonable time window—standard workflow tasks that are neither urgent nor trivial.

***4 – Low***

These actions have a minor impact—and could possibly be delayed if urgent or important work is required on higher priority work.

***5 – Very Low (Default)***

These tasks are the least time sensitive. They are generally for optional tasks or long-term/future enhancement tasks. Whenever no priority is assigned to a task during creation, it defaults to 5.

The priority system allows Admins and Assigners to communicate urgency to Assignees, and for Assignees to work smarter.

***Start Date***

The official date when the task begins or is assigned. This is useful for planning and tracking project timelines.

***End Date (Expected Completion Date)***

The date by which the task is expected to be finished. It helps in evaluating timeliness and performance.

***Status***  
This field reflects the current progress of the task. The default statuses include:

* ***Pending*** – Assigned but not yet started.
* ***In Progress***– Work has started.
* ***Completed*** – Work is done and finalized.
* ***Disabled*** – work assigned to someone else

***Assigned To*** *(Foreign Key to Employee)*

This links the task to a specific **employee** (the **Assignee**) who is responsible for carrying it out. One employee can handle multiple tasks, but each task is assigned to one primary person.

***Estimated Time:***

Before work begins, the assigner inputs how long (in days, hours, or minutes) they estimate this task will take. For example: *“5d”* means 5 days.

***Actual Time***

This field tracks how much time the employee actually spent on the task, which helps in time tracking, productivity analysis, and billing (if needed).

***Comments***

Both assigners and assignees can add notes, updates, or discussions here—making it easier to track context, changes, or clarifications without needing external communication tools.

1. **IMPLEMENTATION**
   1. ***Modules***

While developing the ETM Tool, the goal was not simply to write some code. The goal was to build a system that models the real-life workflow of assigning tasks, working together, and reporting progress within an organization. In order to reach this goal, the system breaks down into many functional modules, each representing a slice of responsibility. Perhaps the most important aspect of the modules is that they all work closely together such that the user flow of data and functionality is seamless. We will discuss each individual module, the story behind the module development, and their implications within the ETM ecosystem.

* 1. ***User Management Module***

The user functionality lies at the heart of the application. Every organization comprises individuals with different roles — in this case, Admins, Assigners, and Assignees. The User Management Module exists to facilitate the identification and management of the users and their responsibilities.

From the initial log in screen, this module governs all things happening in the application concerning user authentication and authorization. Built using JWT security, and at the backend using Nest.js's AuthGuard, this module allows only the legitimate users to access the application and see only the things they are allowed to see.

The admin can do everything — add users, assign roles, switch roles, etc. Assigners and Assignees, on the other hand, have permissions associated with their roles, which means they have a tightly scoped, secure experience. The modular division of powers represents the start of a role-based and secure workflow in the application.

* 1. ***Task Management Module***

For users, once logged in, they are immediately engaged in managing and acting upon tasks. This is where the Task Management Module comes into play; the physical and figurative center of the ETM Tool.

This module allows Admins and Assigners to create tasks, assign tasks, and check the status of tasks. Each task is a mini universe unto itself -- with a unique task ID (e.g., GNY-007), task description, estimated time to completion, start date, end date, department tag, and a comments/ rating section for the user's feedback after completion.

On the front end, built with React and Next.js, all tasks flow seamlessly using state-managed interfaces and forms for easy user interaction. On the back end, using Nest.js, task data is validated when assigned, stored in the database, and allowed access to only registered users of the ETM Tool. Each Assignee sees only tasks related to them, whereas Admins can view all tasks through the admin user type.

Tasks can be updated, completed, commented on, and reassigned, if necessary, if there is a change in availability; this makes for an excellent user experience on a module that is essentially the engine room of the entire system.

* 1. ***Department & Role Management Module***

Organizations are more than a collective group of people -- they involve a collection of departments working parallel to each other. The Department & Role Management Module was created to reflect this structure.

Administrators use the module to create and manage departments -- "Frontend Development", `"QA", "Content Writing" or any type of department would exist. Each user is associated with a department and each task is also associated with a department. While this creates a better organization structure, it also facilitates the robust filtering capabilities associated with the reporting module.

Roles are what a user has access to do. Admins can increase and/or decrease user roles. Assigners have access to users and tasks only in their department group. Assignees only see their own workload. This level of access and feedback creates accountability, clarity and security.

* 1. ***Reporting & Analytics Module***

A system does not offer enough utility until it is providing insights, this is what the Reporting & Analytics Module aims to do.

Admins and Assigners are able to see comprehensive reports on tasks that can be sorted by department, user, date range, or task status. There's already smart filtering and pagination built into the frontend UI, so there's no concern of bogging down performance when accessing larger datasets.

Need to pull report of completed tasks for the month? Or which department has the most incomplete tasks? You can easily answer both of those questions with this module.

Future enhancements may add features for graphical visualizations, such as bar charts or performance heat maps, to expand the analytics ability of the product.

* 1. ***Comments & Rating Module***

Task assignment is not an assign and forget process. Feedback, communications, and iteration is a key component of successful collaboration. This is where our Comments & Rating Module comes into play.

Once a task is finished, the Assigners or Admins can leave comments and ratings that will assist the assignee in developing an understanding of their performance. This creates a transparent feedback loop and is meant to improve the quality of work executed in future tasks, which in turn is also a history log capturing the narrative of work complete over time.

These comments belong to the task record, providing accountability and an opportunity for continuous improvement within teams.

* 1. ***Mailing & Export Module***

What truly differentiates the ETM Tool from a simple task tracker is the Mailing & Export Module. This module adds power and professionalism to reporting and communication, offering users the ability to quickly send updates, reporting, or task summaries with just a few clicks.

Consider a situation where a team leader needs to send their weekly task status report to their manager and copy a couple of others — the Mailing & Export Module takes care of that extremely easily. A user clicks Send Mail from the Reports page and a lovely modal opens for them to enter the email address (To) area and optional Cc's.

With validation of inputs and on-the-fly tagging, the user cannot make an email entry incorrectly or admit someone whom they need to keep informed. Once the user has sent the email, it will add a formatted task report summary to the email, allowing the user to save time, reduce friction, and communicate effectively.

In addition to this, the same page has the Export as PDF and Download as Excel options. Whether for records, audits, or presentations, these functions allow a user to take their data outside of the application while maintaining the format and layout.

* 1. ***Prototype***

The process of designing the ETM Tool was anchored in continual feedback, iterations, and user-centered design. Before the first version of the entire ETM Tool was promoted to production, a functional prototype was built in order to verify the concepts of task tracking, user roles, and real-time reporting. The prototype was typically a middle step between the extra-polated conceptualization and the operating system. It represented a means for both the developers and the stakeholders to conceptualize how the system (which had not yet been built) could or would operate.

The concept prototype was technically not just an unrealized, prototype wireframe example; instead, it was a prototype built with the same stack as the eventual full-system - front-end with React.js/Next.js and back-end with Node.js with Nest.js and database with SQL. The purpose of the prototype was to simply validate whether the core functionalities of the proposed system could operate; consider example validations such as being able to assign tasks, create dynamic reports, set up email notifications and user access control.

The prototype development began with a standard grouped dashboard layout that was intended to illustrate how the modules would be accessed. On the left navigation bar, we could simply clink navigation links for: Dashboard, Employees, Add Task, Departments, Permissions, Roles, My Tasks, and Reports (in which each would result in a visualization of real-time data).

One of the most essential functions presented in the prototype was the Task Report Module. This is the page that allows the user to filter by employee name, task status or date range, and generate valuable reports. Moreover, it serves as a starting point for the advanced features that follow — including creating PDF's, exporting to Excel, and viewing and emailing the reports straight from the interface (as demonstrated in the screenshot below).

The prototype also includes a simulation of a role-based system, so we could also test user privileges — ensuring Admins could assign tasks and see their data across the system, and Assignees could only see tasks assigned to them. This approach helped us improve upon the access logic and security early in the development stage.

We also included UI/UX validation in the prototype. The “Minimal & Clean” design approach wasn't just to make it look convincing — it was to ensure other folks could navigate and understand the interface consistently across devices. We also put it in front of users to hear their feedback about how readable, and collaborative the interface was beyond just navigating through an interface.

In the end, the prototype was more than a mock-up — it was a proof of concept. We were able to reassure ourselves on the direction of development, identify possible bottlenecks early, and give stakeholders a visualization of the system that could highlight both workflow and interaction long before the system was put into full implementation. This series of iterative, prototype processes ultimately helped to develop the final version of the ETM Tool that was more robust, reliable, and met user expectations.

* 1. ***User Roles and Access Levels***

In the ETM Tool, the main concept of the employee task management tool revolves around three important user roles, all having a unique and clear function ensuring assigned tasks move smoothly from beginning to end. The three user roles - Admin, Assigner and Assignee - are more than functional roles; they create a simple sequence of hierarchy to promote clarity, accountability and effectiveness throughout the system.

1. ***Admin: The Ultimate Administrator***

The admin is the top of the hierarchy, controlling every function in the ETM Tool. An Admin is like the leader of an entire organization, controlling everything from users and departments to task assignments and reporting. They have an aerial view of the entire system and operate with an agenda maintaining a high level of performance. Admins need to make sure there are no breakdowns in the system and going even further, they need to make sure every user is correctly operating in the right role, and the organization is on-boarding all tasks and reports correctly for monitoring.

Admins are the constructors of the system. They decide how to grant access, how the department is organized and who has the responsibilities to assign task across their whole user organization. Admins give direction to how the ETM Tool is set up and control all of the options to configure or hotfix the system to keep the flow organized to the right people.

***An Admin's daily responsibilities usually include:***

* Department and Users Management: Admins can add, remove or edit users and organize them into departments.
* Task Assignment Management: Admins can assign tasks to Assigners or Assignees ensuring that all tasks are assigned.
* Report management: Admins can run a report for the entire organization, filter it as useful, and export reports after reviewing them.
* System Settings: Admins assess the system settings and ensure that they meet the needs of the organization, including permissions and workflows.

Typically, this will be an individual in a high-level position such as Head of HR, someone in project management, or a System Administrator who needs full access to administer the platform's overall functionality.

1. ***Assigner: The Connection Between Strategy and Execution***

The Assigner connects strategic goals to tactical actions. You may think of the Assigner as a department head or team leader, the individual that takes the overall vision from Admins or management and assigns the tasks to the execution team. Assigners connect the dots between high-level strategy and the actual execution of work.

With the ETM Tool, Assigners “see everything” that the department or team is doing as they will assign the task to the Assignee and then track it. The Assigner’s job is to assign the right tasks to the right Assignee and to track their progress. The Assigner essentially manages the process by acting as an intermediary between the execution teams and the Administration by ensuring the work is on target, the tasks were clear, and that any reports submitted by the Assignees are expected quality results.

***An Assigner will be responsible for the following:***

* Task assignment: Assigners assign work to their team with each task being well-defined and appointed.
* Task tracking: They monitor the progress of the work to make sure everything is going according to plan.
* Report evaluation: Once Assignees have finished a task, Assigners look through and quality assess the reports, providing feedback where necessary.
* Team-level performance management: They can filter their team's reports to analyze performance for bottleneck processes or areas for improvement.

There is a higher level of access than Assignees but Assigners do not have the ability to manage settings and control the application at a larger scale. These users plan on delegating and tracking team/department tasks with the intention of a smooth-running organization.

1. ***Assignee: The Executor of Tasks***

The Assignee role is the most specific of all. Assignees are the people who actually perform the tasks. They take the tasks assigned to them and get into the details in order to finish the job. After a task is assigned to them, the Assignee can directly access everything they need to know about the task: details, deadlines, and reports—all in the Assignee's personal dashboard.

As for the Assignee, simplicity and clarity is the essence of the role. The Assignee does not worry about managing others or trying to figure out how to set up the system, their only job is to finish what has been assigned to them and update their progress. This allows the Assignee to concentrate on their work, without distractions of other departments, or unnecessary settings on the system-level.

***An Assignee's process looks something like this:***

* Viewing the assignment: For an assignee, they can only see their own tasks. This ensures that they do not get distracted by what other people have on their plates - they can focus on their own tasks.
* Working on assignments: Once they open their task, the assignee is able to see the specific task details, add comments or updates, and track time on each task.
* Completing assignments: When they are done with the task, they will mark it as done, submit a report, and move onto the next task.
* Tracking performance: The assignee can see a summary of their performance with the tasks, this is used for reporting and analysis of the tasks.

Assignees are not able to modify any system settings, they are not able to assign tasks, or view other users' tasks. Their entire role is focused on executing their tasks so there are no questions of efficiency or what their responsibilities are.

* 1. ***Testing Strategy / Test Cases***

Testing is a critical stage in the development of any application, particularly when the application is processing sensitive task-oriented data and multiple roles interact with such data. The aim is to verify that the entire system works together as it was intended, find bugs and errors that might exist within the application, and ultimately improve the quality of the application as a whole.

***Types Of Testing***

In order to ensure testing is thorough and complete, all types of testing will be executed during the development cycle. These include:

* 1. ***Unit Testing***
* Objective: Unit testing covers the unit testing of individual components or units of the application. Each method, function, or class is examined in isolation to confirm everything behaves as expected.
* Tools: Jest, Mocha, React Testing Library.
* Implementation: The unit test for the fetchData function and the data transformation methods are unit tested to determine that it performs the filter and pagination as expected.
  1. ***Integration Testing***
* Objective: This focuses on testing how well the different components of the system work together. This includes the front-end and back-end interaction, and interaction with any third-party services and APIs.
* Tools: Supertest, Cypress, or any API testing framework.
* Implementation: That employee data or task data that is fetched through the back-end API is integrated, or passes into the front-end report components, e.g., ReportTable.
  1. ***Functional Testing:***
* Objective: Focuses on the application’s functionality. The aim is to ensure that the application is doing what it says it should be doing.
* Tools: Cypress or Selenium for end-to-end functional tests.
* Implementation: Functional tests will cover the main user flows: creating a task, assigning a task, reporting a task. For example, an Admin can assign a task to an Assignee, and check to see if the task appears in the task report.
  1. ***Ui/Ux Testing:***
* Objective: Test to see that the UI and experience of the application works well and meets the design objectives. This process checks if the application is easy to use, and if the application maintains the “Minimal & Clean” design approach.
* Tools: Cypress for testing UI visually; Storybook for checking components.
* Implementation: Verifying if components, such as task report tables and user role-specific filters are being displayed as intended.
  1. ***Performance Testing:***
* Objective: Measure how the application performs when under load and ensure it can satisfy several users, keep up with all the complex queries and containing a large dataset of facts without slowing down much.
* Tools: Lighthouse, Google Page Speed Insights, or load testing tools that we build ourselves.
* Implementation: We will look at the performance of the data fetching in the Report Table component and ensure that pagination, filtering, and task updates do not drown the performance of the application.
  1. ***Security Testing:***
* Objective: Helps find weaknesses and security issues with the application and help ensure data protection, and user authentication and authorization.
* Tools: OWASP ZAP, security audits, pen tests.
* Execution: Testing the encryption of user IDs in localStorage, and testing secure user authentication while ensuring that role-based access controls are being properly implemented for Admin, Assigner, and Assignee roles.
  1. ***Regression Testing:***
* Objective: Ensures that recent changes or new features have not broken something that previously worked. Regression testing is an important practice as new code may be added to the system.
* Tools: Jest, Cypress.
* Execution: The regression tests to ensure that the rest of the application is still functioning as intended are executed after implementing a new feature or fixing a bug.
  1. ***Test Cases and Coverage:***

When planning for effective testing, defining test cases for every feature and object is useful for the testing process. Some examples of test cases are:

Giving / Assigning a Task (Functional Test):

* Confirm that an Admin can assign a task to an Assignee.
* Confirm that the assigned task appears in the task list with the Assignee.
* Confirm that task status updates correctly when the task is marked complete.
  1. ***Filtering a Task (Unit Test):***
* Test that the filtering logic correctly filters tasks by department, role, or task status.
* Test that pagination works when filtering many records or datasets.
  1. ***Role Based Access (Security Tests):***
* Confirm that an Assigner cannot view a task assigned to another department.
* Confirm that only Admin users can change task status or assign a new task.

***Bug Tracking and Fixed:***

Bugs/issues are tracked in a bug tracking tool during the testing phase of the project. The bugs/issues are assigned a priority based on severity and severity of the bug. Developers are assigned to fix them based on priority. We use bug tracking tools like Jira, Trello, or GitHub Issues to facilitate this activity.

Continuous Integration/Continuous Deployment (CI/CD):

The testing process can be automated as well through a CI/CD pipeline for the testers to mitigate any bugs or issues. The CI/CD pipeline runs automated tests every time a change is pushed to the code repository - this will help catch bugs early on before they make it to user acceptance testing

* CI Tools - GitHub Actions, Jenkins, CircleCI
* CD Tools - docker for containerization in the CI/CD pipeline; AWS for deployment.

Every time the developers push code, the pipeline is focused on running automated tests that check any new code being added and confirming that nothing new is broken. If the tests are successful, the code is promoted/deployed to the staging environment for further and in-depth testing and review by the developers.

* 1. ***User Acceptance Testing (UAT):***

After the internal testing is completed, the system goes into User Acceptance Testing (UAT). Here, actual users (as applicable Admins, Assigners, Assignees) will be taking the application for a test drive in real-life testing scenarios. This process identifies area/order where there are discrepancies between what was delivered and what was anticipated. Your process/customer and its different users should get a chance to help you identify any gaps in the intended design of the application vs. expected design from the user perspective.

* 1. ***Deployment and Hosting***

After many weeks of planning, designing, coding, and testing, the ETM Tool was finally ready to leave the development desk and go into the real world. But writing code is only half the story — deploying it into a live environment that has real users interacting with it is where the real challenge starts. In this period, it was all about stability, security, and accessibility. We had built the car and now it was time to drive it.

* ***The Server***

We decided to host the ETM Tool on a Virtual Private Server (VPS). Why a VPS? Because it provided us with the control we were looking for — having the option of configuring every aspect of the environment, customizing security and ensuring performance in an environment where we would not have noisy rotten neighbors as with shared hosting.

We provisioned a server running Ubuntu Linux, a stable, tried and tested operating system. It will be our base of operations, the residing place for both the backend and frontend applications to operate and connect as one system.

* ***The Frontend Cometh: Next.js to the Rescue***

The frontend was built using React and Next.js and we wanted it fast, SEO friendly, and production-ready at the same time. We took advantage of Next.js's server-side rendering and static optimization to reduce load times and avoid unnecessary data-fetching.

To host it, we built a production version of the application and hosted it directly through Nginx. Nginx is a high-performance, lightweight web server that can also be used as a reverse proxy. Nginx also handled routing–it would forward requests to the backend when necessary but would serve the static frontend requests.

Lastly, and to be secure, we enabled HTTPS using an SSL certificate from Let's Encrypt. A little green padlock in the browser, and our users could rest assured their data was secure.

* ***The Backend Powering the Application: Nest.js + Node.js***

At the same time, the backend, which was built using Nest.js (a scalable Node.js framework), did all of the heavy lifting, such as connecting to the database, authenticating the user, enforcing role-based restrictions, handling the business logic, and filtering responses for the user based on their respective roles.

To make sure it stayed up and running and functioning properly, we used PM2 (a node.js process manager) to host the backend. PM2 has the added benefit of keeping our app alive (even when crashing, crashing after a server restart, etc.) while making it easy to manage logs and scale.

The backend was also placed behind Nginx, with CORS policies correctly configured so that requests were made from only our trusted frontend.

* ***Database Setup: MySQL as the data backbone***

MySQL is the one that forms the data backbone against which Employee and Task Management (ETM) Tool would work. This is the one which was chosen for all applications for data conserving and retrieval processes. It is known for speed, reliability, and scalability, along with being active in production settings. With this, we were able to derive a schema structure and uphold the integrity and performance of our backendian services.

It was hosted on a virtual private server (VPS), along with our backend and frontend applications, thereby ensuring better access to data. MySQL housed all essential data for users, departments, task details, status updates, time logs, and collaborative comments.

***To make further provisions in favor of performance, security of data, and fault tolerance, the following configurations were made:***

* Secure Access Configuration
* MySQL database access was tightly secured with stronger and complex credentials.
* Database access was allowed for the backend application only with environment-specific credentials.
* Disabling of remote access to avoid any unauthorized connections.
* Blocking of MySQL port (default- 3306) to outside world using server-side firewall rules (UFW/iptables/etc.).

***Role-Based Permission System***

* Application-level roles (Admin, Assigner, Assignee) were mapped into MySQL, giving those users separate accounts based on respective privileges.
* ***For instance:***
* Admin role has access to all tables with full permission for CRUD (Create, Read, Update, Delete) activities.
* Assigners have rights to create/assign tasks but don’t have rights to create/delete users and departments.
* Assignees have read access for tasks assigned to them and can write to time logs/comments.

This privilege separation defended against any user or some unauthorized process from viewing/changing outside of its jurisdiction by mistake or intentional misconduct.

***Daily Automated Backups***

* To protect from data loss, an automated daily backup system using crone jobs was set in place.
* The backup dumps the database using mysqldump into .sql files, with each file timestamped.
* They are compressed, encrypted, and uploaded to a remote secure backup location (e.g., AWS S3/secure FTP server).
* Retention policy kept for keeping daily backups for 07 days and weekly backups for a month.

The countermeasures ensure that in case of hardware failure, data corruption, or accidental deletion, the system can revert to a consistent state with little loss in uptime.

To sum it up, MySQL served as the core handler for all structured data operations and was thereby secured, scalable, and recoverable through proper configuration and best practices.

***Domain Configuration & Routing: Nginx as The Gatekeeper***

Having completed the development of the backend (Nest.js) and frontend (Next.js), the final step was deploying the application to production. The host server is Linux-based Virtual Private Server (VPS).

***To take our application to the Internet, we:***

* Mapped the VPS's public IP to a custom domain name (e.g., etm.example.com)
* Installed and configured Nginx as a reverse proxy server to handle HTTP/HTTPS inbound traffic.

Our Nginx configuration was carefully structured to handle both frontend and backend components efficiently while enforcing security and possible routing.

***Here is how routing and deployment were structured:***

1. ***Enforcing HTTPS with SSL Certificates***

* HTTP requests were all automatically redirected to HTTPS.
* SSL certificates were issued and installed with Let’s Encrypt through Certbot.
* This guarantees that all communications between users and the server would be encrypted with TLS protocols.

1. ***Reverse Proxy for API Requests***

* All the API requests (e.g., etm.example.com/api/...) were proxied by Nginx to the Nest.js backend running on a private port (6000).
* This separation ensured that the backend was protected and that it did not have to directly expose its port in the public.

1. ***Serving Static Frontend Files***

* The frontend (Next.js) application was built down into static assets using next build and next export commands.
* Nginx served these assets straight away—HTML, CSS, JS—from a static folder (e.g., /var/www/etm-frontend).
* This lessened the load of the Node.js server and improved response times for static resources.

1. ***Frontend Routing Handling and 404 Page Handling***

* Nginx was configured to support client-side routing used in single-page applications (SPA).
* Any unmatched routes (e.g., /dashboard, /tasks) were redirected to index.html to be handled by the React-based router.
* Custom 404 error pages were incorporated to give user-friendly feedback on invalid URLs.

1. ***Firewall and Limiting Rate (Optional Improvements)***

* Nginx-basic rate limiting was applied to protect brute force attempts or abuse.
* Only specific ports (80 for HTTP, 443 for HTTPS) were allowed through the firewall.
* Logging was turned on to track any questionable or malicious activity.

***Toward Automation: CI/CD***

Initially we were manually deploying by logging into the server and pulling code through Git, or even worse logging straight into the server, pulling code, restarting. After enough time the system matured to a point that we were looking for ways to have it automated. This forced us to think about our deployment process, and we started using GitHub Actions as a very basic CI/CD pipeline.

From our accomplishments in CI/CD development, whatever usages that were done on the main branch would do the following on every push:

* The server would pull the updated code
* Rebuild the frontend and backend with Docker containers
* Restart using PM2 with zero downtime

This allowed us to reduce manual mistakes, increase deployments and procedural causation of releases.

***The Final Launch***

After all was deployed, and we felt good about things being configured the we wanted, we did one final real user production test. We went through and assigned tasks, we generated the reports, we tested the roles, and it worked just like we had planned.

And finally, we officially went live at: [***http://15.204.213.25:6001/***](http://15.204.213.25:6001/)

It wasn't just a live website; it was a journey from development to delivery. The ETM Tool was live, as more than just code. It was a living being, able to allow teams to manage their work without the issues that they had previously.

1. **RESULTS AND ANALYSIS**

As the ETM Tool transitioned from a development phase to a deployment phase, the true test began - actually using the product in the hands of real users! In a testing environment, all the inputs and controls were controlled; here the complexity of the real world comes into play. Teams across functions began to leverage the platform on a daily basis and we needed the system to validate its reliability, ease of use, and speed.

Certainly, by the first week, we found that Admins were able to not have to think about reporting or managing users, and they could unequivocally provide inputs without delays. Assigners saw a marked improvement in assigning tasks, while Assignees enjoyed the unconfusing common layout and clear task instructions. What was most enlightening was how quickly the system was folded into daily tasks - we already heard ETM referenced in project meetings like it was a reporting function; team leaders were relying on ETM to follow up with requests without having a trail of emails or spreadsheets. It fit.

We monitored key metrics such as speed of response (in API requests from their internet connection), system uptime, and user activity. The platform held steady and saw an average response times of <300ms with 99.8% uptime during daily peak usage. Adoption for the vast majority was smooth as well - 80% of intended users were actively logging in within the first 72 hours of being launched. Did it fit? Users were not overwhelmed. Tasks were clear. Reporting was quicker (I didn’t mention the word report). And for the first time, Admins had a real time view and were no longer reliant on manual status updates.

* 1. ***Performance Evaluation***

After full development and rollout of the ETM Tool, we turned our attention to evaluating its performance in the wild - meaning that we were not just interested in whether it worked (and we had tested some real-world scenarios prior), but how it operated in a functioning, fluid organizational context. In order to be successful, the system needed to work in a variety of contexts - with realistic condition simultaneous use across departments, roles and use cases - rather than in siloed tests.

1. ***Initial Rollout Observations***

The first few days after launch were a near-live system litmus test. Admins started creating departments and users, Assigners logged in to begin delegating tasks, and Assignees got their first tasks in the platform. The experience of live-use allowed us to afford some insight into system capacity with users interacting concurrently. Overall, results were promising - users were able to move around dashboards, assign tasks, and update their task statuses without lag or system crash.

The backend, powered by the Nest.js framework with its architectural integrity, continued to manage all user sessions and role-based queries with reliability. Overall reaction times for the backend APIs averaged 250-300 milliseconds as user activity increased (concurrently) on the server-side. The use of React on the frontend with its virtual DOM promoted rendering performance where possible, and users were generally not inconvenienced when filtering task data and switching between view types.

1. ***Load Testing and Multi-User Support***

To further test the performance of the system we setup simulated stress tests. This consisted of simulating hundreds of users logging on, assigning tasks, pulling reports, and exporting at the same time. The system performed reliably without any downtime or degrading performance. The indexing strategy on our database — especially on user roles, department filters, and task status — played a huge role in performance through load.

We also confirmed the ability to perform complex functions during high demand situations — for example, generating a pdf/excel report on demand, or emailing a batch task summary. Every effort completed within acceptable limits, and we verified the reliability of both our front-end language, as well as our backend services.

1. ***Monitoring, Uptime and Server Optimizing***

Based on our deployed architecture — Dockerized and containerized — it made the scaling and portability effectively streamlined. We used a number of monitoring tools to effectively monitor the usage, memory allocation, as well as server health status in real time. The system performed with over 99.8% of uptime in the first month after deployment. Our restart policies on containers ensured that even under rare edge-case situations that would crash applications the systems recovered within seconds.

We also throttled and applied retry logic on API calls to gracefully octure network disruptions. This helped increase resiliency within the function of critical user actions, like task updates or comment submission are important.

1. ***User Feedback and Iterative Tuning***

Some of the most significant performance measurement came from the end users themselves. Assigners noted that filtering and exporting reports that would previously take minutes in other tools only took a few clicks with our tool! The admins appreciated how seamlessly they could administer multiple departments without using a complicated interface. Even the Assignees, often the least technical users, indicated that the task panel was easy to understand and felt fast.

We took early user feedback and made some simple optimizations like reducing image sizes, optimizing font loading and caching access data. These small optimizations decreased the average load time while improving the overall experience further.

* 1. ***Screenshots of Working System***

Materializing the ETM Tool was not only about building user features, it involved building a flawless role-based user experience that put the vision of the project into an implementable interface. As we approached the end of building the application, we documented what brought the application to life with screenshots that documented a live working environment showcasing the key functionality of the application, its design layout, and flow from one role to another.

The screenshots below are visual proof of working with admins, assigners, and assignees within their unique dashboards. Each role interacts uniquely with their dashboard.

1. ***Login Screen - The Starting Point***

The application begins with a clean, secure login page. The login is to be clear and simple so that users navigate easily with their credentials. Once authenticated, they will be directed to their dashboards based on roles that will be unique to their login credentials, allowing for an individualized experience from the very first time they logged in.

|  |
| --- |
|  |
| **Fig. 5.1** |

1. ***Admin Dashboard - Total System Control***

The admin dashboard serves as a control center. All system activity can be accessed and viewed on one screen, including totals for users, departments and task statuses, reports generated. Even important activities like adding new users, viewing task reports or exporting individual department or task data can all be done from this screen in one place.

|  |
| --- |
|  |
| **Fig. 5.2** |

1. ***Assigner Dashboard - Department-Level Task Management***

Assigners work in an area in which they can assign, monitor and review tasks within their teams, and they can't see data and tasks outside of their department. Each department is a unique universe, in which the Assigners can see all tasks for the department, assign new tasks, edit task details and review completion rates, all via a table that intuitively displays the relevant activity.

|  |
| --- |
|  |
| **Fig. 5.3** |

1. ***Assignee Dashboard - Basic and Simple***

Finally, the Assignee dashboard reduces everything to user tasks only. The only things an Assignee will see are the tasks associated with their login identity and will include the task status, due dates, and any comments standards. Each task the Assignee holds can be updated with comments about progress, or they can click to complete the task.

|  |
| --- |
|  |
| ***Fig. 5.4*** |

1. ***Profile Page – Personal Overview***

All users, regardless of role, are able to access and view their profile page. Each user can see details about him or herself including name, email, role, and department to which he or she belongs. Admins may have multiple profiles that they may access and edit, but users may only view their own profile.

|  |
| --- |
|  |
| **Fig. 5.5** |

1. ***Employees – Managing the Workforce***

This section is primarily available to Admins. It provides a full view of all registered users in the system along with their assigned departments and roles. Admins can add, remove, or update employees from this centralized panel, maintaining full control over user management.

|  |
| --- |
|  |
| **Fig. 5.6** |

1. ***Add Task – Initiating the Work Cycle***

Accessible by Admins and Assigners, this feature allows task creation. The form includes fields such as task title, description, assignee selection, department, due date, and priority. It’s built to be simple yet detailed, ensuring the task starts with full clarity.

|  |
| --- |
|  |
| **Fig. 5.7** |

1. ***Departments – Structuring the Organization***

Departments act as the backbone of user categorization. Admins can create, rename, or delete departments here. Assigners are mapped to departments, allowing them to work only within their scope. This ensures organizational clarity and prevents cross-departmental data leakage.

|  |
| --- |
|  |
| **Fig. 5.8** |

1. ***Permissions – Key Level of Control***

For systems that require extensibility, managing permissions are critical. This section provides Admins a way to manage permissions at a granular level in support of each role, extending to the permission of simply viewing information, modifying information, assigning settings, and exporting information. It

Serves as a capstone for access control throughout the ETM Tool.

|  |
| --- |
|  |
| **Fig. 5.9** |

1. ***Roles – Establishing the User Hierarchy***

Admins manage the creation of role definitions here. Each of the defined roles (e.g., Admin, Assigner, Assignee, etc.) is associated with individual permissions. This module allows for expanded future roles as the system allows for an adaptable approach to changing team structures.

|  |
| --- |
|  |
| **Fig. 5.10** |

1. ***My Tasks – Defined Task List for Every Role***

The My Tasks provides a defined view for all users, showing only the tasks assigned to them. No matter the user role, a defined task view is provided to manage responsibilities. Status updates, comments, and time tracking features enable tracking responsibilities as well.

|  |
| --- |
|  |
| **Fig. 5.11** |

1. ***Reports - Data-Driven Performance Tracking***

Reports are the most powerful tools for admins and assigners. The reports can be filtered by department, date range, task status, or by individual users. The data can be previewed in the report, exported to PDF or Excel, or emailed directly from the system. You can monitor your teams in just a few clicks, allowing you to make informed decisions quickly.

|  |
| --- |
|  |
| **Fig. 5.12** |

1. ***Send Mail - Direct Email Functionality***

Send Mail is a feature that allows admins and assigners to send task reports or task updates via email that is pre-populated directly in the system. Users enter the email address for the recipient and CC users if required, preview the message, and send it without leaving the dashboard. It is easy, functional, and combines task email with all other task information in one task entry.

|  |
| --- |
|  |
| **Fig. 5.13** |

1. ***ToDo List – Google Calendar-style Task Scheduling***

The ToDo List module has a familiar Google Calendar-style interface where users can plan, view and manage their tasks as visual timelines and calendars. Tasks are displayed across the calendar in organized days, weeks, and months to directly visualize what is planned when. Whether it's assigning new tasks, or verifying the next set of deadlines, this module effectively setup simple views and structure for Admins, Assigners, and Assignees.

|  |
| --- |
|  |
| **Fig. 5.14** |

* 1. ***Comparison with Existing Systems***

Before the creation of the ETM Tool, many organizations were piecing together their workflows using separate tools: spreadsheets, manual emails, and separate task management apps like Trello, Asana, or Google Sheets. These tools could serve basic purposes for task management, but they lacked the contextual integration with reporting, permissions, departmental roles, and real-time performance information.

***ETM was meant to close these gaps.***

ETM is not a generic platform. Unlike other platforms, ETM offers a hierarchical task level assignment, user access roles, detailed reporting, and integrated features to emailing reports and exporting. Admins no longer need to switch between multiple tools to manage users and track reports, they can do everything in the ETM ecosystem. Assigners use specifically tailored dashboards with in-task data, while Assignees are presented a clean and well-defined set of work to manage and submit.

With more contextual clarity, more centralized control, and more enhanced collaboration, ETM outperforms existing systems that are more closely aligned to enterprise task management needs.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Feature** | **Trello / Asana / Jira** | **ETM Tool** | | Hierarchical User Roles | Limited / Add-ons required | Built-in: Admin, Assigner, Assignee | | Reporting Workflow | Requires custom setup | Core feature, deeply integrated | | Department-Wise Task Separation | Not natively supported | Fully Supported | | Export Reports (PDF, Excel) | Plugins or integrations needed | One-click native export | | Access Control | Partial, mostly via workspaces | Full RBAC using Nest.js | | Self-hosted Deployment Option | SaaS only | Self-hosted and fully customizable | |
| **Table - 5.1** |

* 1. ***Coding Section***

The coding aspect is the core element of the Employee Task Management Tool where the actual logic and layout of the app were implemented. This section will give a detailed explanation of how the development was approached, its technologies, coding structures used and how different functionalities were quickly developed through programming.

1. ***Technology Stack Used***

The application has been developed on top of a custom full-stack JavaScript-based architecture that is fast, scalable, and easy to maintain. You can build a highly efficient web application using this stack with technologies that we're all familiar with - MySQL, Express.js, React.js, Node.js.

***Here is a breakdown of the technology stack used in the project:***

***Frontend:***

* ***React.js****:* React.js is a JavaScript library that builds dynamic user interfaces for component-based Applications. React.js allows you to develop sophisticated user interfaces by breaking each facet of the interface into its components that can be reused. This modularity provides a structure where the code is easier to understand and maintain.
* ***Tailwind CSS****:* Tailwind CSS is a utility-first CSS framework. It helps you customize and build user interfaces without writing custom CSS. It requires a minimalist approach to styling, while providing a catalog of thousands of utility classes that you can combine responsively.
* ***Material UI (MUI)****:* MUI is a widely used React component library built following Google's Material Design. It consists of materials, styles, and example code to create easily customized user interfaces with consistent look and feel. MUI allows for rapid development of rich user interfaces with complex structures, such as buttons, tables, dialogs, and modals that are already structured and can replicate exactly.
* ***React Icons****:* React Icons is a library of customizable icons for React Applications. It also provides a standardized set of icons that you can use in your app to ensure the same uniform design across your application with very little work.

***Key Features:***

* Component-based architecture for reusable UI components.
* TypeScript integration for better type safety, which minimizes runtime errors and enables more reliable code to work with.
* SSR/SSG (Server-Side Rendering / Static Site Generation) with Next.js for SEO benefits and faster page load.
* Tailwind CSS for minimalistic styling and faster styling (and therefore faster development).

***Backend***

* ***Node.js:*** A JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js allows the entire application to be written in JavaScript, and enables the use of the same language for both the front-end and back-end. A significant selling point for Node.js is its performance for I/O-intensive tasks, which allows for real-time and scalable applications.
* ***Express.js:*** A minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. Express.js is used for handling requests and responses, routing, middleware, and building RESTful APIs.
* ***Nest.js:*** A framework for building scalable and maintainable server-side applications using Node.js. Nest.js is built with TypeScript, but organizes server code into modules, controllers, and services. It works well with Express.js and provides additional functionality, such as dependency injection, a robust routing provider, and built-in support for common tasks like validation and authentication.

***Key Features:***

* API Design:
* RESTful APIs: The backend exposes REST APIs to perform CRUD operations and business logic for the frontend.
* JWT Authentication: Used for secure, stateless user authentication using JSON Web Tokens (JWT).
* Modular Design: The backend has a modular design pattern that ensures each feature (e.g., user management, reporting) is encapsulated into modules, making the code easier to maintain and scale.

***Database: MySQL***

* ***MySQL:*** The most popular relational database management system (RDBMS) used to store structured application data. MySQL is an open-source, reliable, high-performance database that facilitates structured SQL queries.
* ***ORM Integration (TypeORM/Prisma):*** MySQL is coupled with an Object-Relational Mapping (ORM) library, such as TypeORM or Prisma, for interacting with the MySQL database. This ORM simplifies database interactions, allowing developers to interact with the database using JavaScript or TypeScript instead of raw querying negotiations.
* ***Key Tables:***
* Users: Stores user information, including email, hashed passwords, and roles.
* Reports: Stores user-generated reports, as well as metadata such as description, user ID, and created date.
* Audit Logs: Stores logs for security and compliance to track changes made by users.

1. ***Frontend Structure and Logic (/frontend)***

Frontend leverages Next.js and TypeScript for type safety and scalability. Below are important folders and what they do:

1. **components/:**

Re-usable UI components -

sideBar/: Navigation Component

profile/: User info card or dropdown

table/: Table UI logic

1. **tasks/:**

Contains all of the task view logic and components -

assignedTask.tsx: Displays task(s) assigned to user

unAssignedTask.tsx: Displays all unassigned task(s)

reportTable.tsx: Displays report data

page.tsx: Parent component for task routing or dashboard

1. **toDoListComponent/:**

Dedicated modular components for Task Tracking -

toDoList.tsx: To-Do's -> lists to do's

clearBtn.tsx: Reset progress on task

AuthGuard.tsx: Route guard to protect from unauthorized access

EditCommentModal.tsx: Modal to edit comment(s) on Task

TaskModal.tsx, Modal.tsx: Modals for creating/viewing Task(s)

sendmail.tsx: Logic to trigger email for report

loader.tsx: Spinner for async actions

textEditor.tsx: Rich text editor for comments

1. **context/:**

Global State -

AuthContext: Provides universal auth and role/context user data throughout app

1. **.env.local**

File that stores environment variables such as base API URL or frontend secrets.

***Authentication Flow***

* The login form produces a call to the backend, which returns jwt token.
* This token is stored in a safe place, and is then used on the frontend to control route access and UI by role.
* The AuthGuard.tsx component prevents users who are unauthorized access to protected routes.

1. ***The Employee Task Management Tool Logic***

Each task,

- Is stored in MySQL with fields like name, status, priority, assignedTo, estimatedTime, comments, and timeLogs.

- Can be created from TaskModal.tsx, parent task logic if applicable.

- It has the states: Pending, In Progress, Disabled, and Completed.

Time tracking allows for manual entries (i.e., 2h, 30m, 1d) with remaining time auto-calculated and updated in the UI.

1. ***Reporting & Filtering Capabilities***

One of the main capabilities of the ETM Tool is the ability to produce dynamic task reports from various filtering options. This capability is implemented on the front end via the reportTable.tsx component, and supported by API endpoints on the back end that query filtered data from the database.

The Report Section provides Admins and Assigners with a way to view all the tasks in the system by department, status, and user. Therefore, the design is such that task-related decisions can be made quickly by providing powerful filters and real-time results for the users.

***Filtering Options***

The report will allow users to filter the task report for the following criteria:

* ***Date:*** Create reports for tasks created or updated on a specific date.
* ***Employee:*** Filter tasks assigned to a specific employee.
* ***Department:*** Reports filtered on the department, i.e., Programming, Marketing.
* ***Task Status:*** Report shows tasks based on status - Pending, In Progress, Completed, and Disabled.
* Users can apply filters singularly or in combination enabling users to filter for very specific task views.

1. ***Export & Action Options***

Once the relevant filters are configured, there are different options to export or share the report via:

***Download PDF:*** The filtered task report (and its relevant data) will be exported into a nicely formatted PDF file.

***Download Excel:*** The filtered data will be exported into an Excel file for further editing or analytics.

***Send mail:*** a built-in feature that allows a user to send a report directly via mail to relevant stakeholders (in this case, their list of recipients' emails).

This feature covers 3 components,

***Frontend component:*** sendmail.tsx opens a modal and captures user input (i.e., recipients(e-mail), comments)

***Backend logic:*** uses nodemailer to process paving paths to send the email (the user has the option to attach the report as a file, or copy and paste it into the body of the e-mail).

This modular approach allows us to offer a feature where users' task data is not only visible, but also actionable.

1. **CONCLUSIONS & FUTURE SCOPE**

***6.1. Conclusions***

As we reached the final phase of our system's development, it was evident the distance the Enterprise Task Management (ETM) had come - from a thought initially scribbled on paper to a working application that allowed Admins, Assigners, and Assignees to utilize an easier way to plan, organize, and manage work tasks. The need was simple: to better organize tasks, the design of the solution would require depth, flexibility, and thoughtfulness.

Along the way in the development process, we encountered the stark realities of other existing systems, including differences in the interpretation of communication, intermittent status reports, and a complete lack of centralized control - as well as other issues inherent in traditional systems. It was essential to recognize and address these issues - and by doing so, we were able to build a system that not only manages tasks but significantly changes the organization of work within teams.

The ETM interfaces with many different roles, all coming together to form one system. The ETM creates a responsive dashboard, task scheduling based on a calendar similar to Google Calendar, reporting capabilities, and email, all with a contemporary user interface/user experience approach. The system helps users track their workflow, provides motivation to complete tasks as scheduled, and facilitates easy and regular updates and measurement of progress.

In some ways, the most useful aspect of this project is the shift in organizational mindset – from a 'reactive' space to a more 'proactive' approach. With advanced filters, real-time panels, data exportation to analytical tools, distribution of email messages, and even playful wall messages, decision-makers can act faster with more confidence. Employees can focus on completing their jobs by having visibility into tasks, deadlines, and progress.

The ETM, in conclusion, has provided an organized, transparent, and intelligence-based approach to work – a necessary evolution for today's organizations.

***6.2. Future Scope of Work***

While the ETM Tool already fulfils its core purpose, we consider it a living platform that grows with its users. The expansion opportunities are plentiful, and we highlight a few key areas of future growth potential below:

1. ***Mobile App for Cross-Platform Capability***

In today's digital world, work is not limited to working at the office desktop. Staff, particularly managers and those working in the field, frequently move around and need access to their tasks and task updates. One of the most meaningful and impactful improvements for growth in the future would be a dedicated mobile app for Android and iOS platforms.

A mobile version of the ETM Tool would allow users to receive notifications in real-time, update task statuses, and access their task lists without being at their desk. Allowing push notifications for deadlines or task assignments will enable users to stay alert and aware of what is due. The mobile app would also allow quick actions (e.g., marking a task as "complete," adding comments, or uploading an image/document attached to a task) to make the platform more engaging and effective. Additionally, from a user perspective, the app could utilize native features such as fingerprint login and/or face recognition to assist users in accessing the platform more securely and efficiently.

1. ***Visual Analytics Dashboard***

The current reporting module provides great data; however, we will need to move towards visual, interactive, and intuitive dashboards. Rather than searching through raw tables and rows, we will have a brightly colored dashboard to display the health of projects, performance of users, workloads by department, and distinction between pending and completed tasks using bar charts, pie charts, line graphs, and heat maps.

A dashboard of this type would allow Admins and Assigners to rapidly identify trends, analyze team performance, or identify loading bottlenecks in real time. Interactive filters (like dragging a date range or hovering over a data point) would not only allow for analysis but would also put the fun in analysis. This makes the system not just a task manager, but a companion for decision-making.

1. ***Real-Time Collaboration Features***

As organizations become increasingly team-oriented, the need for collaborative communication will emerge. While external communication tools like email and messaging services exist to support communication about specific tasks, the ETM Tool can employ a built-in chat or comments feature directly connected to each task. Team members could tag their colleagues, share quick updates, attach files, or even leave a voice note.

Looking ahead, the system would have the option for in-built integration with external tools like Slack, Microsoft Teams, WhatsApp, and others, all while notifying users about the task at hand. The flow of task notifications and discussions would be helpful when sending and receiving updates without switching to different applications. Using existing external communication tools provides reminders, ideas, and nominations for clarifying task discussions, all in one convenient location. Ultimately, all task-related communication would be centralized and organized for clarity and communication context.

1. ***Smart Task Assignment with AI***

Today, assigning tasks is a somewhat manual process, relying on our human judgment to determine who is available or skilled enough. But what if there was a system that aided us in making that decision?

In the future, we will implement an artificial intelligence (AI)-based smart task assignment capability. This would utilize AI to analyze historical data, such as an employee's workload, task performance, average time to completion, and availability, in order to recommend the best person for a new task. The feature could also help identify and flag potential delays, given that there are patterns the system can identify (e.g., certain departments always having late submission deadlines).

Not only would such predictive capabilities help us work more efficiently, they could also help us keep tasks distributed fairly and holistically optimized—especially in larger teams, where monitoring people's workloads becomes a nearly impossible constraint.

1. ***Third Party Integrations***

The potential of any digital system lies in its ability to integrate with other tools that organizations already have in place. In future upgrades, we plan on developing integrations with some of the following platforms: Google Calendar, Microsoft Outlook, Jira, Zoho, and HRMS tools.

This means that we would be able to sync calendars, send automatic reminders of tasks, provide single sign-on (SSO) for a much faster login, and map employee information directly from the HR database. For example, an ETM task that has been planned would be visible in a user's Outlook calendar automatically, or a deadline could be sent out as a reminder via Google notifications. This would enhance the experience for users as well as the usability of integration across departments.

1. ***Audit Trails and Version Control***

As the system becomes essential to teamwork, transparency and traceability will be paramount. Future iterations might offer detailed audit trails that will log every action, when they are taken; whether making, updating, reassigning, or changing the status of a task.

Now, old versions of descriptions, due dates, or notes could be versioned and compared or restored. It would provide protection from inadvertent data loss, as well as present a visual storyline of the impact of a task. These features could be very helpful in compliance environments or larger organizations that have multiple levels of review and approval and documentation.

**REFERENCES**

# Bibliography & References

* [*https://www.npmjs.com/package/react*](https://www.npmjs.com/package/react)
* [*https://mui.com/material-ui/react table/?srsltid=AfmBOoqvHDhrlZJ4J6hkPO4zEkfbigLLkM4G1LD4lVSK50XklmmQXQo-*](https://mui.com/material-ui/react%20table/?srsltid=AfmBOoqvHDhrlZJ4J6hkPO4zEkfbigLLkM4G1LD4lVSK50XklmmQXQo-)
* [*https://nextjs.org/docs/pages/building-your-application/routing/dynamic-routes*](https://nextjs.org/docs/pages/building-your-application/routing/dynamic-routes)
* [*https://nextjs.org/docs/pages/building-your-application/data-fetching/client-side*](https://nextjs.org/docs/pages/building-your-application/data-fetching/client-side)
* [*https://quilljs.com/docs/installation*](https://quilljs.com/docs/installation)
* [*https://tailwindcss.com/docs/colors*](https://tailwindcss.com/docs/colors)
* [*https://codesandbox.io/examples/package/react-big-calendar*](https://codesandbox.io/examples/package/react-big-calendar)
* [*https://mui.com/material-ui/material-icons/*](https://mui.com/material-ui/material-icons/)
* [*https://lucide.dev/icons/categories#math*](https://lucide.dev/icons/categories%23math)
* [*https://react-tooltip.com/docs/getting-started*](https://react-tooltip.com/docs/getting-started)





