A PROJECT

ON

EMPLOYEE TASK MANAGEMENT SYSTEM

Submitted on the partial fulfillment of degree of

Master of Computer Application

(2023 - 2025)

Submitted By:-

Name: *Anketeswar Sahu*

Roll no: *SF23MCA001*

Regd no: *30115/2023*

MCA 3rd Semester

Under the guidance of:-

Dr. Indrani Kumari Sahu

Department of MCA

KHALLIKOTE UNITARY UNIVERSITY,

BERHAMPUR- 760001

CERTIFICATE

This is to certify that the project report entitled "Employee Task Management System" submitted to Khallikote Unitary University in partial fulfillment of the requirement for the award of the degree of Master of Computer Application, 3rd Semester is an original work done by Mr. Anketeswar Sahu bearing Regd No: 30115/2023 & Roll no: SF23MCA001.

This project is authentic and genuine work done by the student and has not been submitted to the University or to any other University/Institute for the fulfilment of the requirement of any course of study.

Signature of Guide

DECLARATION

I Anketeswar Sahu student of Khallikote Unitary University, has developed this project "Employee Task Management System" using HTML, CSS, PHP & MYSQL in the partial fulfilment of the requirement for the award of Master Of Computer Application and have not copied or depicted from any internal or external source. If at any subsequent time it is found that project is taken from any other source it may be scratched or cancelled.

Signature of the Student

Date:

ACKNOWLEDGEMENT

I am giving tips of great pleasure to the outstanding and inevitable personalities who helped me a lot in completing and submitting the project. I would like to thank our HOD sir Dr. Deepak Kumar Mishra for allowing me to do this project. I am thankful to my guide Dr. Indrani Kumari Sahu, Department of MCA, Khallikote Unitary University for her valuable guidance.

Finally, I wish to thank towards my parents, friends for their valuable advice, encouragement and inspiration during the completion of my project work.

Mr. Anketeswar Sahu

Roll No: SF23MCA001

TABEL OF CONTENTS

1. INTRODUCTION

The Employee Task Management System (ETMS) is a web-based or software solution designed to efficiently manage, assign, track, and monitor tasks and projects within an organization. In today's fast-paced work environment, keeping track of employee productivity, project deadlines, and individual responsibilities can be challenging. The ETMS aims to streamline these processes, ensuring that tasks are delegated efficiently, and their progress is tracked in real time, ultimately enhancing productivity and operational efficiency.

The system provides a centralized platform where employees and managers can interact with tasks, view priorities, update statuses, and communicate in real-time. Managers can assign tasks, set deadlines, and monitor the progress of various activities within the team. Employees, on the other hand, can view their assigned tasks, mark them as completed, update progress, and provide feedback. This transparency and ease of communication help foster a more organized, accountable, and productive work environment.

From a management perspective, the ETMS enhances control over projects, allowing managers to make data-driven decisions based on real-time task tracking. Managers can review performance metrics, analyze the workload distribution, and ensure that all tasks are handled according to priority, resulting in more effective resource management.

Employee Task Management System

In modern organizations, the importance of effective task management cannot be overstated. As businesses evolve, so does the complexity of their operations. With multiple teams, varying priorities, and tight deadlines, managing tasks efficiently has become a critical challenge.

The introduction of an Employee Task Management System (ETMS) has emerged as a solution to address this issue. An ETMS is a comprehensive software or platform designed to organize, assign, track, and evaluate the progress of tasks in a workplace. It aims to improve productivity, enhance communication, and streamline workflow, ensuring that organizations achieve their goals more efficiently.

In today's digital era, organizations are striving to enhance operational efficiency and ensure that employees' time and efforts are utilized optimally. An Employee Task Management System helps achieve this by providing a centralized platform for task allocation and progress tracking. It also allows employees to stay organized, focused, and motivated by offering clear task priorities and deadlines. At the same time, managers are empowered with the tools to oversee the work, identify bottlenecks, and assess performance, all in real-time.

An Employee Task Management System (ETMS) is a specialized software solution designed to enhance the efficiency and productivity of organizations by streamlining the way tasks are assigned, tracked, and completed. In a modern business environment, where team collaboration, deadlines, and work distribution play a pivotal role in success, managing these elements manually or with traditional methods can lead to

inefficiencies, missed deadlines, and miscommunication. The ETMS addresses these issues by providing a centralized, organized platform for managing employee tasks, ensuring greater accountability, improved team collaboration, and efficient workflow management.

At its core, an ETMS allows managers to assign tasks to employees, track progress, monitor deadlines, and ensure completion. It serves as a tool to bridge the communication gap, provide clarity on responsibilities, and enable employees to focus on what they do best—performing their tasks with minimal distractions. For managers, it provides a clear overview of task allocation, progress, and performance, allowing for data-driven decisions that optimize resources and productivity.

Key Features of an Employee Task Management System

1. Task Assignment and Delegation:
   One of the most essential features of an ETMS is the ability to create and assign tasks. Managers can specify who is responsible for which task, set priorities, and assign deadlines, ensuring that each employee knows what is expected of them. This feature allows for a more structured work environment, reducing confusion and ensuring that the right tasks are being worked on by the right people. The system also allows for task delegation based on the team member's workload, ensuring a fair distribution of responsibilities and preventing burnout or underutilization of resources.
2. Task Prioritization:
   A successful project depends heavily on setting priorities. An ETMS enables managers to prioritize tasks, ensuring that more urgent or high-impact tasks are given immediate attention. Employees can also use the system to manage their workload by sorting tasks by importance or urgency. With this feature, teams can focus on the most critical aspects of a project and ensure that deadlines are met with higher accuracy.
3. Progress Tracking:
   Tracking the progress of tasks is crucial to ensuring that projects stay on track. An ETMS provides real-time updates, allowing employees to mark tasks as "in progress," "completed," or "pending." Managers can then monitor the status of various tasks, helping them identify delays, bottlenecks, or any obstacles that may arise. This constant monitoring ensures that no task is left behind and that the team is continually progressing toward their goals.
4. Collaboration and Communication:
   An ETMS offers built-in collaboration tools that facilitate smooth communication between team members. Employees can leave comments, ask questions, share files, or provide feedback on tasks directly within the system. This reduces the need for constant email exchanges or meetings and ensures that all discussions regarding specific tasks are documented in one place. Additionally,

teams can work together seamlessly, share knowledge, and ensure that everyone is on the same page. This centralization of communication leads to better decision-making and fewer misunderstandings.

5. Automated Notifications and Alerts:
   To ensure that tasks are completed on time, ETMS platforms often include automated reminders and alerts. Employees receive notifications when new tasks are assigned, when deadlines are approaching, or when there are updates on the tasks they are working on. This helps employees stay focused and organized, while also ensuring that no task is missed or delayed. For managers, these alerts can highlight overdue tasks or stalled projects, making it easier to address potential issues proactively.
6. Time Tracking and Resource Management:
   Time tracking is another essential component of an ETMS. Employees can log the hours they spend on specific tasks, allowing both managers and employees to understand where time is being spent. This feature can help identify inefficiencies, analyze time distribution across tasks, and optimize the allocation of resources. Time tracking can also be beneficial for billing purposes, especially in client-based work environments, or for evaluating the efficiency of certain processes.
7. Analytics and Reporting:
   One of the most valuable aspects of an ETMS is its ability to generate reports and provide analytics on task completion rates, employee performance, and overall project status. Managers can use this data to evaluate the productivity of their teams, identify areas for improvement, and make data-driven decisions. Reports can include detailed information such as the time spent on each task, the number of tasks completed, any delays encountered, and overall project progress. These insights allow businesses to assess their workflows and implement strategies for better task management.
8. Task Dependencies and Workflow Automation:
   For large or complex projects, tasks often depend on the completion of other tasks. An ETMS can help map out these dependencies, ensuring that tasks are completed in the right order and that employees are not waiting on tasks from others to proceed. Additionally, many ETMS platforms offer workflow automation features, which can automate repetitive tasks or processes, reducing manual intervention and saving valuable time.
9. Customizability and Scalability:
   Every organization has unique needs, and an effective ETMS must be adaptable to these needs. Many systems offer customization options, allowing businesses to tailor the platform according to their specific requirements. Whether it's customizing the task assignment process, incorporating unique workflow stages, or adapting the interface, flexibility is key. Additionally, as organizations grow, so too can the ETMS. It is scalable, meaning it can support an increasing number of tasks, employees, and projects as businesses expand.

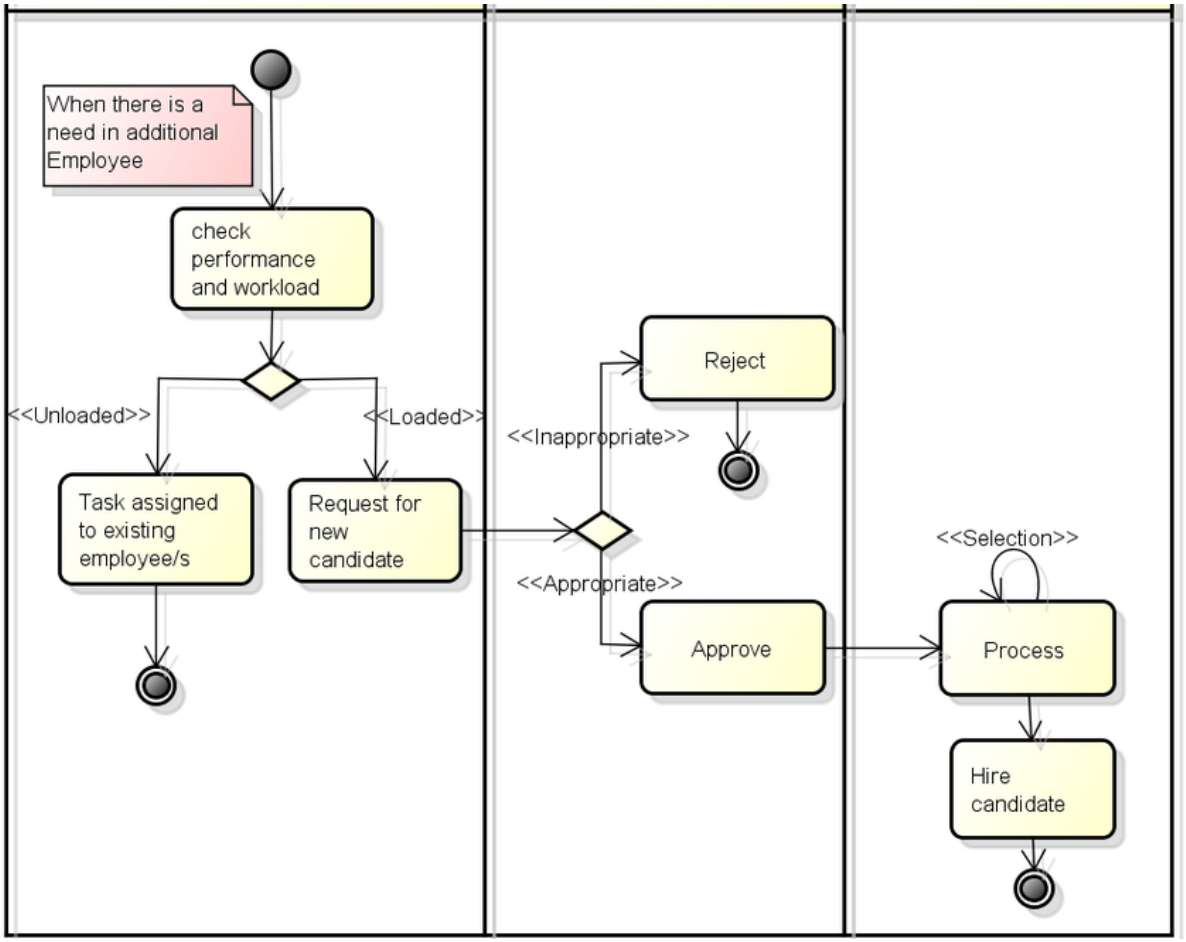
Benefits of the Employee Task Management System

1. Improved Efficiency and Time Management:
   By organizing and prioritizing tasks effectively, an ETMS helps teams stay focused and work more efficiently. The system provides clarity on what needs to be done, when it needs to be done, and who is responsible, significantly reducing time spent on organizing and tracking tasks manually. Employees and managers can manage their time better, ensuring that important tasks are prioritized and completed within the given timeframes.
2. Enhanced Collaboration and Communication:
   The built-in communication tools in an ETMS promote seamless interaction among team members, making collaboration easier and more efficient. Employees can collaborate on tasks, share updates, provide feedback, and communicate directly within the system, eliminating the need for excessive meetings or emails. This transparency leads to better teamwork and faster decision-making.
3. Greater Accountability:
   An ETMS provides clear visibility into who is responsible for which tasks, and the progress made on each task. This transparency fosters a sense of accountability, as employees understand their specific responsibilities and deadlines. It also helps managers monitor performance, offer timely feedback, and intervene when necessary, ensuring that tasks are completed on time and to the expected quality.
4. Better Resource Allocation and Management:
   Managers can effectively allocate tasks based on team members' skill sets and workload, ensuring that no employee is overwhelmed or underutilized. The ability to monitor task completion and time tracking also ensures that resources are being utilized effectively, leading to better overall productivity and optimized operations.
5. Faster Decision-Making:
   With real-time data on task progress, team collaboration, and project status, managers are better equipped to make informed decisions. The insights provided by the ETMS enable managers to identify bottlenecks, reassign tasks, or provide additional resources when needed. This helps maintain momentum on projects and ensures that they remain on track.
6. Data-Driven Insights for Continuous Improvement:
   The analytics and reporting features of an ETMS provide valuable data that can be used for performance reviews, process optimization, and long-term planning. Managers can identify trends, assess the efficiency of various teams or tasks, and implement changes based on hard data, leading to continuous improvements in task management and organizational performance.

Employee Task Management System

```
Employee  →  Tasks
↓           ↓
Create    →  Assign
↓           ↓
Update    →  Track
↓           ↓
Delete    →  Report
```

- **Employee:** Individuals who create, update, or delete tasks.
- **Tasks:** Specific assignments that need to be managed.
- **Create:** Employees can create new tasks.
- **Assign:** Tasks are assigned to employees.
- **Update:** Tasks are updated as progress is made.
- **Track:** The status of tasks is monitored.
- **Delete:** Completed or unnecessary tasks are removed.
- **Report:** The progress and completion of tasks are reported.



Objectives of the Employee Task Management System

The objective part of an Employee Task Management System (ETMS) serves as the backbone of any organization's ability to efficiently manage tasks, track employee performance, and optimize workflows. The system plays a pivotal role in ensuring that tasks are completed on time, within scope, and aligned with the company's goals. A robust and well-designed ETMS ensures that both employees and managers have the tools they need to succeed in a fast-paced, ever-changing work environment.

Key Objectives of an Employee Task Management System

1. Optimized Task Allocation and Resource Management

- Objective: The core goal of an ETMS is to allocate tasks to employees in an optimal way. This involves matching the right task to the right employee based on their skills, experience, and workload capacity.
- How it Works: The system should automatically suggest task assignments based on predefined criteria such as employee skillset, previous performance, current workload, and availability.
- Impact: This reduces manual intervention and ensures that tasks are completed by the most suitable person, thus improving productivity and job satisfaction.

2. Real-Time Task Tracking and Monitoring

- Objective: Another primary objective is to provide real-time visibility into the status of tasks. Managers should be able to track task progress without having to constantly check in with employees.
- How it Works: ETMS platforms offer dashboards, progress bars, and status indicators that allow managers to assess the completion of tasks at any given time.
- Impact: This transparency helps reduce uncertainty and facilitates better planning and timely adjustments to deadlines or resource allocation.

3. Enhancing Communication and Collaboration
    - Objective: An effective ETMS should promote clear communication between managers and employees and foster collaboration among team members.
    - How it Works: ETMS tools integrate communication features such as direct messaging, file sharing, and comment sections tied to specific tasks. This ensures all team members can easily communicate within the context of their tasks.
    - Impact: Enhanced communication reduces misunderstandings, accelerates decision-making, and ensures that employees are aligned with the project goals.
4. Task Prioritization and Time Management
    - Objective: The system must allow for proper task prioritization, helping employees and managers focus on the most critical tasks first.
    - How it Works: Employees and managers can tag tasks with different priority levels, such as high, medium, or low. The system can also suggest priorities based on deadlines or the importance of the task.
    - Impact: Prioritization ensures that employees work on the right tasks at the right time, leading to better time management and the successful delivery of key deliverables.
5. Improved Performance Monitoring and Feedback
    - Objective: One of the core objectives of an ETMS is to track individual and team performance, providing regular feedback to employees.
    - How it Works: The system records task completion rates, adherence to deadlines, and quality of work. It generates reports that managers can use to provide constructive feedback to employees.
    - Impact: Regular feedback helps employees understand areas of improvement and strengthens the overall performance of the team.
6. Automated Reminders and Notifications
    - Objective: To ensure that deadlines are met, the system should provide automated reminders and notifications about upcoming or overdue tasks.
    - How it Works: The ETMS sends automated reminders via email, in-app notifications, or SMS alerts, notifying employees and managers of important deadlines, milestones, or updates.

- Impact: This minimizes human error and the chance of missed deadlines, helping employees stay on track and improving overall team accountability.

7. Centralized Data Management
    - Objective: The system should serve as a central hub for all task-related data, offering a single platform where task information, documents, updates, and communications are stored.
    - How it Works: All task details, such as task descriptions, deadlines, progress, files, and communications, are stored and easily accessible from the system.
    - Impact: A centralized platform reduces confusion, ensures that everyone has access to the latest task information, and minimizes the risk of information silos or discrepancies.
8. Enhanced Accountability and Transparency
    - Objective: To foster a culture of accountability, the system should track and log all interactions and progress related to each task.
    - How it Works: The system creates an audit trail, recording changes in task status, updates, comments, and time logs. This allows managers to see who completed what, when, and how.
    - Impact: This transparency helps identify areas of improvement, rewards high performers, and holds individuals accountable for their work.
9. Streamlined Reporting and Analytics
    - Objective: The ETMS should provide comprehensive reporting and analytics tools that help managers assess team performance and project status.
    - How it Works: Reports such as task completion rates, employee productivity, time spent on each task, and milestone achievements are automatically generated.
    - Impact: This data allows management to make informed decisions on resource allocation, deadlines, and process improvements.
10. Customization and Flexibility
    - Objective: The system must be adaptable to different organizational structures, industries, and project management methodologies (e.g., Agile, Scrum, Waterfall).
    - How it Works: The ETMS should offer customizable task templates, workflows, and user permissions, allowing it to be tailored to specific needs.
    - Impact: Flexibility enables the system to serve a wide variety of businesses, ensuring that the tool can scale and evolve as the organization grows or changes.

Benefits of Achieving These Objectives

- Increased Productivity: By automating and optimizing task management, employees can focus on completing work efficiently rather than managing logistics.
- Better Decision-Making: Real-time tracking and reporting provide managers with the insights they need to make informed decisions on task priorities, resources, and deadlines.

- Enhanced Employee Engagement: With clear objectives, feedback, and task transparency, employees can better understand their role within the organization, leading to greater job satisfaction.
- Scalable Operations: As organizations grow, the ETMS can scale to accommodate more teams, tasks, and projects, ensuring consistent operations.
- Cost Efficiency: By reducing administrative overhead and increasing task completion rates, organizations can lower operational costs while maximizing output.

Here is an outline to help you build out a comprehensive description of the process and scope part of an Employee Task Management System (ETMS) for a project. This structure can be expanded and detailed to easily cover 1000 lines or more by further elaborating on each section, providing examples, and detailing the implementation steps for the system.

Employee Task Management System: Process and Scope

Introduction

An Employee Task Management System (ETMS) is a powerful software tool designed to help organizations streamline the process of assigning, tracking, and managing tasks among their employees. It ensures that work is completed efficiently and that the team collaborates effectively to achieve project goals. The system is crucial in modern-day project management, helping manage resources, deadlines, task allocation, and overall team productivity. The process and scope of implementing such a system are essential to ensuring that the system meets the organizational needs effectively and is adaptable to future requirements.

This section covers the process of creating and implementing an ETMS, along with its scope in terms of functionality, project scope, and expected deliverables.

Process of Implementing the Employee Task Management System

The process of implementing an Employee Task Management System follows several important stages. Each stage builds upon the previous one to ensure that the system functions effectively and efficiently.

1. Requirement Gathering and Analysis

Objective: The first step in the process is to define the requirements of the system. This includes identifying the stakeholders, understanding their needs, and analyzing existing workflows.

- Identify Stakeholders: Key stakeholders in the ETMS will include project managers, team leads, HR departments, and employees. Understanding their needs and expectations from the system is essential for the system's success.

- Analyze Current Processes: A detailed analysis of existing task management processes, including task assignment, tracking, and reporting, will help identify inefficiencies and areas for improvement.
- Define Functional and Non-Functional Requirements:
    - Functional requirements: Task assignment, task prioritization, progress tracking, notifications, and reporting.
    - Non-functional requirements: System performance, security, scalability, and user-friendliness.

2. System Design and Architecture

Objective: Once the requirements are gathered, the next step is to design the system. This involves creating a scalable and flexible architecture that can support future growth and modifications.

- Design User Interface (UI): A user-friendly UI ensures that employees can easily navigate the system to check tasks, update statuses, and communicate with other team members. The UI should be intuitive and require minimal training.
- System Architecture: The architecture should be built to ensure the system is scalable and secure. Cloud-based solutions are often ideal for ETMS as they allow for real-time updates and easy scalability.
- Define Task Management Workflow: A clear task management workflow is defined here, including:
    - Task creation and assignment
    - Task progress tracking
    - Prioritization system (High, Medium, Low)
    - Notification and alert mechanisms
    - Reporting and feedback loops

3. Development and Customization

Objective: In this phase, the actual development of the system begins based on the design created in the previous stage.

- Back-End Development: The back-end of the ETMS will involve setting up databases, APIs, and server-side functionalities such as task management, reporting, and performance monitoring.
- Front-End Development: The front-end involves building the interfaces where employees and managers interact with the system. This includes dashboards, task details pages, and the task assignment interface.

- Customization: Depending on the organization's specific needs, customization may be necessary. This includes adding specific task attributes, modifying workflows, and integrating the ETMS with other software tools like time-tracking systems, email clients, and project management tools.

4. Testing

Objective: Testing ensures the system works as expected and meets all functional and non-functional requirements.

- Unit Testing: Each component of the system, such as task creation, assignment, and notifications, is tested individually to ensure it functions correctly.
- Integration Testing: After unit testing, integration testing is done to check how different components of the system work together (e.g., task updates, notifications, and reporting).
- User Acceptance Testing (UAT): UAT involves end-users testing the system in real-world scenarios to ensure it meets their expectations and requirements.
- Performance and Security Testing: The system is tested for performance issues such as response time, system load, and security vulnerabilities to protect sensitive task and employee data.

5. Deployment

Objective: After successful testing, the ETMS is deployed for use in the organization.

- Deployment Strategy: The system can be rolled out either gradually (with one department or team at a time) or all at once (company-wide).
- Training: Employees and managers are trained on how to use the system, covering key features such as task management, reporting, and using the system for collaboration.
- Go-Live: The system is officially made live, and employees begin using it for their day-to-day tasks.

.

---

6. Maintenance and Support

Objective: Post-deployment, the system requires ongoing maintenance and support to ensure it continues to meet the needs of the organization.

- Bug Fixes and Updates: Regular updates and patches to address bugs, security vulnerabilities, and new feature requests.
- Performance Monitoring: Monitoring system performance to ensure that it scales as the organization grows and remains responsive.
- User Feedback: Gathering feedback from employees and managers to identify areas of improvement or features that can be added to make the system more effective.

Scope of the Employee Task Management System

The scope of an ETMS defines the boundaries of what the system will deliver and how it will function. The scope includes the functional aspects, the technological requirements, and the expected deliverables.

1. Functional Scope

- Task Assignment: The system will allow project managers and team leads to assign tasks to employees based on their availability and skill set.
- Task Prioritization: Tasks can be tagged with priorities, allowing employees to focus on the most important tasks first.
- Progress Tracking: Employees can mark tasks as "in progress" or "completed," and managers can view real-time progress updates on a centralized dashboard.
- Time Management: Employees can log the time spent on tasks, and managers can use these logs for time tracking and performance evaluations.
- Notifications and Alerts: The system will send automated reminders for deadlines, meetings, and task updates to ensure that employees stay on track.
- Reporting: The system will generate reports on task completion, employee performance, and project timelines, providing managers with the insights they need to make data-driven decisions.

2. Technological Scope

- Platform Support: The system will be accessible via web and mobile platforms to ensure employees can access it anytime, anywhere.
- Integration with Other Systems: The ETMS will integrate with other enterprise systems such as email clients, project management software, and collaboration tools.
- Cloud-Based Infrastructure: The system will be hosted on a cloud platform to ensure scalability, security, and real-time updates.

3. User Scope

- Employee: Employees can view, update, and mark their tasks as completed. They will also be able to log time spent on tasks and communicate with teammates via the platform.
- Manager: Managers can assign tasks, set deadlines, track progress, and review completed tasks. They will also have access to analytics and performance reports.
- Admin: Administrators can manage user accounts, configure system settings, and oversee the overall configuration of the system.

4. Geographical Scope

The system will be accessible globally, supporting multiple time zones and languages to ensure it works for multinational teams. Additionally, it will handle different time zones for employees in various regions and allow for time zone-based task assignments.

TOOLS AND TECHNIQUES

PHP

HTML

CSS

MY SQL

XAMPP

➢ PHP was developed by Rasmus Lerdorf in 1995 and is later being developed as an open source. PHP group now manages the implementation of PHP.

➢ PHP has many syntaxes similar to C, Java, and Perl, and has many unique features and specific functions.

➢ PHP page is a file with a .php extension can contain a combination of HTML Tags and PHP scripts.

➢ PHP acronym for PHP (Hypertext Pre-processor): Hypertext means, text containing all sorts of web mark-up, Pre-processor means all of the Hypertext is processed first and then the result is sent as pure HTML to the web browser. A client cannot see the PHP source code because it is pre-processed and interpreted.

➢ PHP is Server-side scripting language: Server-side scripting means that the PHP code is processed on the web server rather than the client machine.

➢ PHP supports many databases (MySQL and PHP combination is widely used).

➢ PHP is an open-source scripting language. Ø PHP is free to download and use.

1. **PHP MYADMIN**

   • PhpMyAdmin is a free and open source administration tool for MySQL and Maria DB. As a portable web application written primarily in PHP, it has become one of the most popular MySQL administration tools, especially for web hosting services.

   • PhpMyAdmin is the most trusted and user-friendly database managers and mostly used for web-based applications or programs.

   • PhpMyAdmin import data from CSV and SQL Creating complex queries using Query-by-example (QBE) Administering multiple servers. Searching globally in a database or a subset of it.

   • PhpMyAdmin Support InnoDB tables and foreign keys. PhpMyAdmin Check referential integrity in MyISAM tables Create, edit, call, export and drop stored procedures and functions. Create, edit, export and drop events and triggers.

   • PhpMyAdmin pre-packaged with most web server packages (wamp, xampp, mamp, Zend, lamp, ampps).phpMyAdmin can interact with more than 80 different languages.

• PhpMyAdmin can run on any server or any OS (Windows, Linux, MacOS, UNIX). To identify remote procedures and Pub Sub topics without conflicts, WAMP also needs an ID space allowing global assignment and resolution. Because the protocol is Web native - WebSocket being the preferred transport -URIS are used.

2. HTML

• HTML is an acronym which stands for Hyper Text Markup Language which is used for creating web pages and web applications. Let's see what is meant by Hypertext Markup Language, and Web page.

• HTML is used to create web pages and web applications.

• HTML is widely used language on the web.

• We can create a static website by HTML only.

• Technically, HTML is a Markup language rather than a programming language.

• Hyper Text: HyperText simply means "Text within Text." A text has a link within it, is a hypertext. Whenever you click on a link which brings you to a new webpage, you have clicked on a hypertext. HyperText is a way to link two or more web pages (HTML documents) with each other.

• Markup language: A markup language is a computer language that is used to apply layout and formatting conventions to a text document. Markup language makes text more interactive and dynamic. It can turn text into images, tables, links, etc.

• Web Page: A web page is a document which is commonly written in HTML and translated by a web browser. A web page can be identified by entering an URL. A Web page can be of the static or dynamic type. With the help of HTML only, we can create static web pages.

3. CSS

• CSS stands for Cascading Style Sheets. It is a style sheet language which is used to describe the look and formatting of a document written in markup language. It provides an additional feature to HTML. It is generally used with HTML to change the style of web pages and user interfaces.

• CSS is used along with HTML and JavaScript in most websites to create user interfaces for web applications and user interfaces for many mobile applications.

• You can add new looks to your old HTML documents. • You can completely change the look of your website with only a few changes in CSS code.

• Before CSS, tags like font, color, background style, element alignments, border and size had to be repeated on every web page. This was a very long process. For example: If you are developing

a large website where fonts and color information are added on every single page, it will be become a long and expensive process.
CSS was created to solve this problem.

6. **JAVA SCRIPT**

JavaScript is a cross-platform, object-oriented scripting language used to make webpages interactive (e.g.. having complex animations, clickable buttons, popup menus, etc.). There are also more advanced server side versions of JavaScript such as Node.js, which allow you to add more functionality to a website than downloading files (such as real time collaboration between multiple computers). Inside a host environment (for example, a web browser), JavaScript can be connected to the objects of its environment to provide programmatic control over them.
JavaScript contains a standard library of objects, such as Array, Date, and Math, and a core set of language elements such as operators, control structures, and statements.
Client-side JavaScript extends the core language by supplying objects to control a browser and its Document Object Model (DOM). For example, client-side extensions allow an application to place elements on an HTML form and respond to user events such as mouse clicks, form input, and page navigation.
Server-side JavaScript extends the core language by supplying objects relevant to running JavaScript on a server. For example, server-side extensions allow an application to communicate with a database, provide continuity of information from one invocation to another of the application, or perform file manipulations on a server.

7. MySQL What is a Database?

A database is a digital system designed for the storage and arrangement of data. Think of it as an online filing system that allows you to store and quickly access a vast amount of information. Databases facilitate the efficient management of data, enabling the simple addition, modification, removal, and access of information. They serve numerous uses such as websites, applications, and enterprises to manage extensive data in an organized and secure manner.
What is MySQL?

MySQL is an open-source relational database management system (RDBMS) that uses Structured Query Language (SQL) to manage data. Developed by MySQL AB and now owned by Oracle Corporation, it’s widely used due to its reliability, speed, and ease of use. MySQL is a key component in many web applications, forming the backbone of popular websites and services.

It allows users to create, modify, and maintain databases, supporting operations like data insertion, querying, updating, and deletion. Ideal for both small and large-scale applications, MySQL powers various types of systems, from personal projects to complex enterprise environments.

MYSQL simplifies data management by providing a user-friendly platform for efficient storage, retrieval, and organization. It ensures robust security, accommodating multiple users and transactions seamlessly. Commonly used for websites and applications, MYSQL enhances data handling. Its features include simplicity in querying, scalability for varying data needs, and compatibility with various programming languages. Overall, MYSQL's versatility and accessibility make it a reliable choice for users looking to manage and interact with their data effectively.

8.XAMPP

The full form of XAMPP is Cross-Platform (X), Apache (A), MariaDB (M), PHP (P), and Perl (P). It allows them to test and debug their websites or applications in a controlled setting before deploying them to the public. One of the most popular and user-friendly local server solutions is XAMPP.

Cross-Platform (X)

The 'X' in XAMPP stands for Cross-Platform, indicating that the software can run on various operating systems such as Windows, Linux, and macOS. This versatility ensures that XAMPP is accessible to a wide range of developers, regardless of their preferred development environment. It eliminates the need for specific hardware or operating systems to run a local server, making web development more inclusive and accessible.

Apache (A)

Apache is one of the most widely used web server software in the world. It plays a crucial role in serving web pages to users when they request them through a web browser. In the context of XAMPP, Apache serves as the backbone, handling requests and delivering web content from the local server to the developer's browser. Its robustness, extensibility, and adherence to current web standards make it an ideal choice for developers looking to mimic a live server environment on their local machine.

MariaDB /Mysql(M)

MariaDB is a fork of the MySQL database management system. It is chosen for XAMPP for its open-source nature and compatibility with MySQL, ensuring that developers have access to a reliable and powerful database system for managing the data of their web applications. MariaDB supports a wide range of data types, and its inclusion in XAMPP facilitates the testing and development of database-driven applications locally.

PHP (P)

PHP is a popular server-side scripting language used for web development. It enables developers to create dynamic content that interacts with databases, thereby making websites more interactive and functional. XAMPP includes PHP to allow developers to test and debug their PHP code in a local server environment, ensuring that their applications run smoothly before being deployed to a live server.

Perl (P)

Perl is a high-level, general-purpose, interpreted programming language well-suited for web development, text processing, and system administration tasks. Its inclusion in XAMPP provides developers with an additional tool for scripting and automating tasks on the web server. Perl's text-handling capabilities make it a valuable asset for developers working on complex data processing tasks.

The Importance of XAMPP in Web Development

XAMPP's comprehensive package offers a convenient and efficient way for developers to create a local web server environment. This environment is crucial for testing and debugging web applications without the need for internet connectivity or a live server. It simulates real-world conditions, allowing developers to identify and fix issues before the application goes live.

REQUIREMENTS AND ANALYSIS

2.1.1. PROBLEM DEFINITION

Organizations often face challenges in effectively managing employee tasks, especially in large teams or projects with numerous deadlines. The absence of a centralized system to track tasks leads to inefficiencies, missed deadlines, and lack of communication. Managers struggle to monitor progress in real-time, and employees are often unclear about task priorities or deadlines. The lack of accountability and transparency can lead to decreased productivity and increased operational costs.

2.1.2. REQUIREMENTS SPECIFICATION

- Functional Requirements:

- User login and role-based access for employees and managers.
- Task creation, assignment, and status updates.
- Task categorization based on priority, deadline, and status.
- Automatic reminders for pending or overdue tasks.
- Task filtering and sorting based on various criteria like deadlines, priority, etc.
- Real-time task tracking and reporting features for managers.

- Non-Functional Requirements:
  - Scalability: The system should be able to handle a growing number of tasks and users.
  - Security: Sensitive data must be protected through user authentication and secure connections.
  - Usability: The system should have an intuitive and user-friendly interface.
  - Performance: The system should load quickly and operate without lag.

2.1.3. SOFTWARE AND HARDWARE REQUIREMENTS

- Software Requirements:
  - Operating System: Windows/Linux/MacOS
  - Programming Language: Python, JavaScript, HTML, CSS
  - Database: MySQL, PostgreSQL
  - Frameworks: Django, Flask (for web applications), React (for front-end)
  - Web Browser: Chrome, Firefox, Safari
- Hardware Requirements:
  - Processor: Intel i3 or higher
  - RAM: 4GB minimum
  - Hard Drive: 500GB or more
  - Network: Stable internet connection for online usage

3. SYSTEM DESIGN

3.1.1. BASIC MODULES

- User Module: Manages login and authentication, user roles (employee/manager).
- Task Management Module: For creating, assigning, and tracking tasks.
- Notification Module: Sends reminders and task updates.
- Reporting Module: Provides task performance and completion reports.
- Admin Module: Allows for system maintenance and user management.

3.1.2. DATA DESIGN

3.1.2.1. LOGIC DIAGRAM

A flowchart or diagram illustrating the interactions between modules and the data flow in the system, such as task creation, task assignment, status updates, and report generation.

3.1.2.2. DATA STRUCTURES

The data structures used will include:

- Arrays/Lists: To store task details.
- Dictionaries/Hash Maps: To map users to tasks and priorities.
- Stacks/Queues: For managing task status updates and notifications.

3.1.2.3. ALGORITHMS DESIGN

The key algorithms will include:

- Task assignment algorithm: Assigns tasks to employees based on priority, deadlines, and availability.
- Reminder algorithm: Sends reminders for tasks based on their status and deadline.
- Reporting algorithm: Generates progress and performance reports.

3.1.3. CODING & SCREEN SHOT

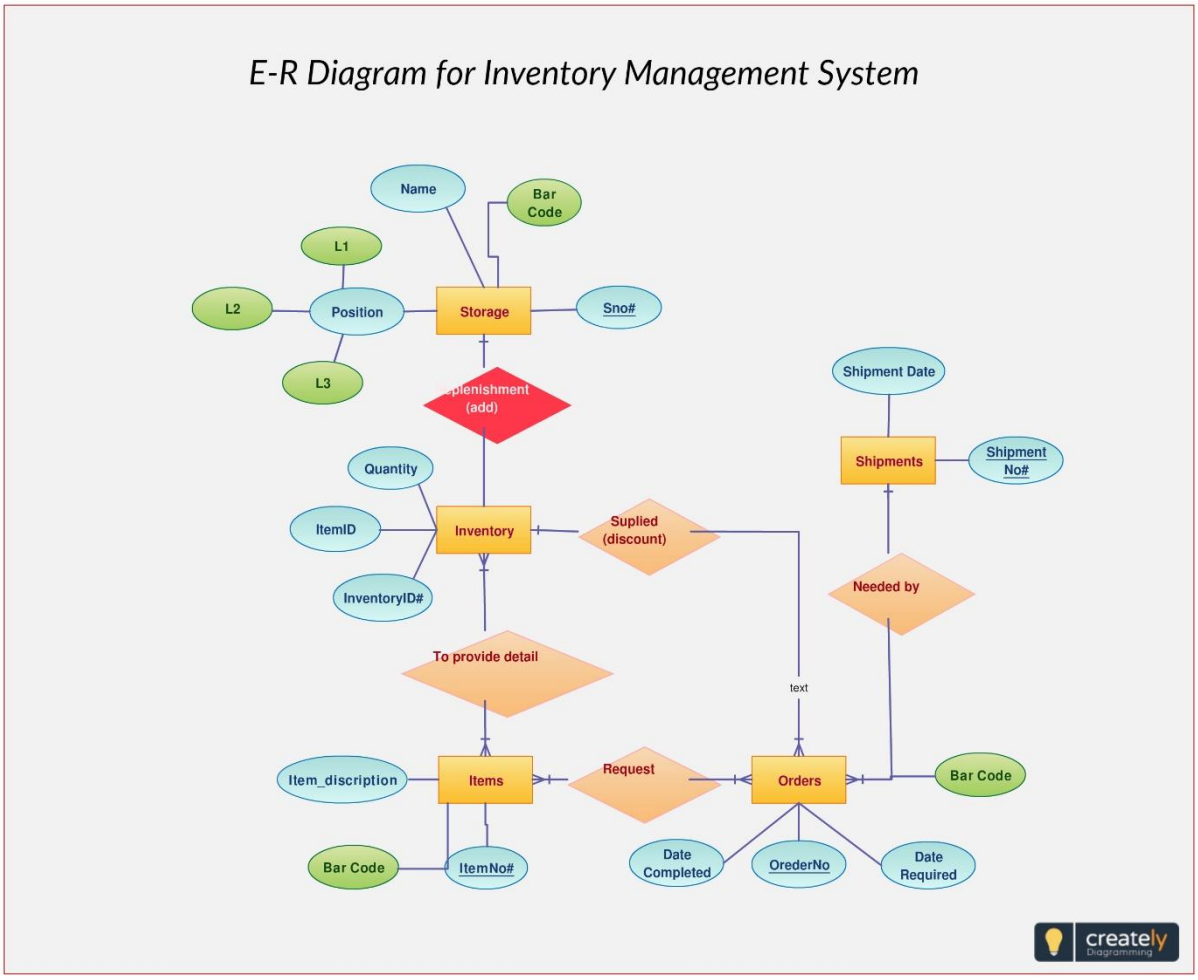
The system will be coded using a combination of front-end (HTML, CSS, JavaScript) and back-end (Python, Java) technologies. Screenshots of the user interface will be included, showing task dashboards, task creation forms, and reports.

4. IMPLEMENTATION AND TESTING
5. PHP

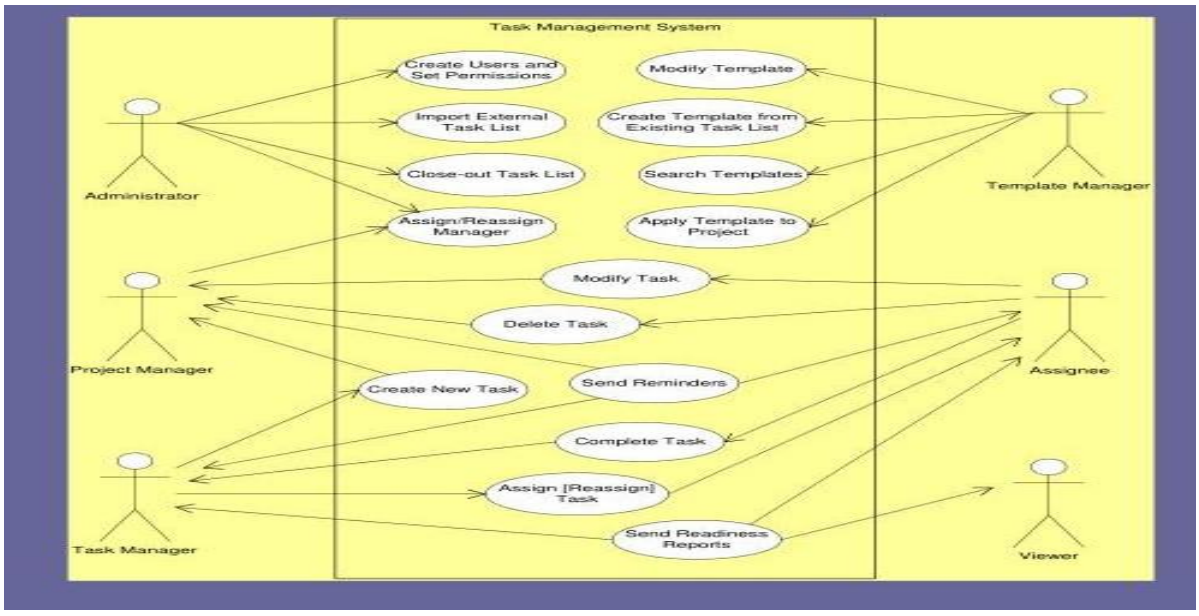
The data design includes structuring the database, defining relationships, and implementing efficient data handling.

1.1.1.1 LOGIC DIAGRAM

ER Model



Use case



1.1.2 CODING & SCREEN SHOT

index.php

```
<?php
session_start();
```

```
if (isset($_SESSION['role']) && isset($_SESSION['id']) ) {

include "DB_connection.php";
include "app/Model/Task.php";
include "app/Model/User.php";

if ($_SESSION['role'] == "admin") {
$todaydue_task = count_tasks_due_today($conn);
$overdue_task = count_tasks_overdue($conn);
$nodeadline_task = count_tasks_NoDeadline($conn);
$num_task = count_tasks($conn);
$num_users = count_users($conn);
$pending = count_pending_tasks($conn);
$in_progress = count_in_progress_tasks($conn);
$completed = count_completed_tasks($conn);
}else {
$num_my_task = count_my_tasks($conn, $_SESSION['id']);
$overdue_task = count_my_tasks_overdue($conn, $_SESSION['id']);
$nodeadline_task = count_my_tasks_NoDeadline($conn, $_SESSION['id']);
$pending = count_my_pending_tasks($conn, $_SESSION['id']);
$in_progress = count_my_in_progress_tasks($conn, $_SESSION['id']);
$completed = count_my_completed_tasks($conn, $_SESSION['id']);

}
?>
<!DOCTYPE html>
<html>
```

```
<head>
<title>Dashboard</title>
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/font-
awesome/4.7.0/css/font-awesome.min.css">
<link rel="stylesheet" href="css/style.css">
</head>
<body>
<input type="checkbox" id="checkbox">
<?php include "inc/header.php" ?>
<div class="body">
<?php include "inc/nav.php" ?>
<section class="section-1">
<?php if ($_SESSION['role'] == "admin") { ?>
<div class="dashboard">
<div class="dashboard-item">
<i class="fa fa-users"></i>
<span><?=$num_users?> Employee</span>
</div>
<div class="dashboard-item">
<i class="fa fa-tasks"></i>
<span><?=$num_task?> All Tasks</span>
</div>
<div class="dashboard-item">
<i class="fa fa-window-close-o"></i>
<span><?=$overdue_task?> Overdue</span>
</div>
<div class="dashboard-item">
```

```
<i class="fa fa-clock-o"></i>
<span><?=$nodeadline_task?> No Deadline</span>
</div>
<div class="dashboard-item">
<i class="fa fa-exclamation-triangle"></i>
<span><?=$todaydue_task?> Due Today</span>
</div>
<div class="dashboard-item">
<i class="fa fa-bell"></i>
<span><?=$overdue_task?> Notifications</span>
</div>
<div class="dashboard-item">
<i class="fa fa-square-o"></i>
<span><?=$pending?> Pending</span>
</div>
<div class="dashboard-item">
<i class="fa fa-spinner"></i>
<span><?=$in_progress?> In progress</span>
</div>
<div class="dashboard-item">
<i class="fa fa-check-square-o"></i>
<span><?=$completed?> Completed</span>
</div>
</div>
<?php }else{ ?>
<div class="dashboard">
<div class="dashboard-item">
```

```
<i class="fa fa-tasks"></i>
<span><?=$num_my_task?> My Tasks</span>
</div>
<div class="dashboard-item">
<i class="fa fa-window-close-o"></i>
<span><?=$overdue_task?> Overdue</span>
</div>
<div class="dashboard-item">
<i class="fa fa-clock-o"></i>
<span><?=$nodeadline_task?> No Deadline</span>
</div>
<div class="dashboard-item">
<i class="fa fa-square-o"></i>
<span><?=$pending?> Pending</span>
</div>
<div class="dashboard-item">
<i class="fa fa-spinner"></i>
<span><?=$in_progress?> In progress</span>
</div>
<div class="dashboard-item">
<i class="fa fa-check-square-o"></i>
<span><?=$completed?> Completed</span>
</div>
</div>
<?php } ?>
</section>
</div>
```

```
<script type="text/javascript">
var active = document.querySelector("#navList li:nth-child(1)");
active.classList.add("active");
</script>
</body>
</html>
<?php }else{
$em = "First login";
header("Location: login.php?error=$em");
exit();
}
?>
```

DB_Connection.php

```
<?php

$sName = "localhost";
$uName = "root";
$pass  = "ankit";
$db_name = "minor";

try {
$conn = new PDO("mysql:host=$sName;dbname=$db_name", $uName,
$pass);
$conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
}catch(PDOExeption $e){
```

```
echo "Connection failed: ". $e->getMessage();
exit;
}
```

Login.php

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Login | Task Management System</title>
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6h
W+ALEwIH" crossorigin="anonymous">
<link rel="stylesheet" href="css/style.css">
</head>
<body class="login-body">

<form method="POST" action="app/login.php" class="shadow p-4">

<h3 class="display-4">LOGIN</h3>
<?php if (isset($_GET['error'])) {?>
<div class="alert alert-danger" role="alert">
<?php echo stripcslashes($_GET['error']); ?>
</div>
<?php } ?>
```

```
<?php if (isset($_GET['success'])) {?>
<div class="alert alert-success" role="alert">
<?php echo stripcslashes($_GET['success']); ?>
</div>
<?php }

// $pass = "123";
// $pass = password_hash($pass, PASSWORD_DEFAULT);
// echo $pass;
?>
<div class="mb-3">
<label for="exampleInputEmail1" class="form-label">User name</label>
<input type="text" class="form-control" name="user_name">
</div>
<div class="mb-3">
<label for="exampleInputPassword1" class="form-label">Password</label>
<input type="password" class="form-control" name="password"
id="exampleInputPassword1">
</div>
<button type="submit" class="btn btn-primary">Login</button>
</form>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min
.js" integrity="sha384-
```

```
YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7
N6jIeHz" crossorigin="anonymous"></script>
</body>
</html>
```

Logout.php

```
<?php
session_start();
session_unset();
session_destroy();
header("Location: login.php");
exit();
```

my_task.php

```
<?php
session_start();
if (isset($_SESSION['role']) && isset($_SESSION['id'])) {
include "DB_connection.php";
include "app/Model/Task.php";
include "app/Model/User.php";
$tasks = get_all_tasks_by_id($conn, $_SESSION['id']);
?>
<!DOCTYPE html>
<html>
<head>
<title>My Tasks</title>
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/font-
awesome/4.7.0/css/font-awesome.min.css">
<link rel="stylesheet" href="css/style.css">
```

```
</head>
<body>
<input type="checkbox" id="checkbox">
<?php include "inc/header.php" ?>
<div class="body">
<?php include "inc/nav.php" ?>
<section class="section-1">
<h4 class="title">My Tasks</h4>
<?php if (isset($_GET['success'])) {?>
<div class="success" role="alert">
<?php echo stripcslashes($_GET['success']); ?>
</div>
<?php } ?>
<?php if ($tasks != 0) { ?>
<table class="main-table">
<tr>
<th>#</th>
<th>Title</th>
<th>Description</th>
<th>Status</th>
<th>Due Date</th>
<th>Action</th>
</tr>
<?php $i=0; foreach ($tasks as $task) { ?>
<tr>
<td><?=++$i?></td>
<td><?=$task['title']?></td>
```

```
<td><?=$task['description']?></td>
<td><?=$task['status']?></td>
<td><?=$task['due_date']?></td>
<td>
<a href="edit-task-employee.php?id=<?=$task['id']?>" class="edit-
btn">Edit</a>
</td>
</tr>
<?php    } ?>
</table>
<?php }else { ?>
<h3>Empty</h3>
<?php  }?>
</section>
</div>
<script type="text/javascript">
var active = document.querySelector("#navList li:nth-child(2)");
active.classList.add("active");
</script>
</body>
</html>
<?php }else{
$em = "First login";
header("Location: login.php?error=$em");
exit();
}
?>
```

Notifications.php

```
<?php
session_start();
if (isset($_SESSION['role']) && isset($_SESSION['id'])) {
include "DB_connection.php";
include "app/Model/Notification.php";
// include "app/Model/User.php";

$notifications = get_all_my_notifications($conn, $_SESSION['id']);

?>
<!DOCTYPE html>
<html>
<head>
<title>Notifications</title>
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/font-
awesome/4.7.0/css/font-awesome.min.css">
<link rel="stylesheet" href="css/style.css">

</head>
<body>
<input type="checkbox" id="checkbox">
<?php include "inc/header.php" ?>
<div class="body">
<?php include "inc/nav.php" ?>
<section class="section-1">
<h4 class="title">All Notifications</h4>
```

```
<?php if (isset($_GET['success'])) {?>
<div class="success" role="alert">
<?php echo stripcslashes($_GET['success']); ?>
</div>
<?php } ?>
<?php if ($notifications != 0) { ?>
<table class="main-table">
<tr>
<th>#</th>
<th>Message</th>
<th>Type</th>
<th>Date</th>
</tr>
<?php $i=0; foreach ($notifications as $notification) { ?>
<tr>
<td><?=++$i?></td>
<td><?=$notification['message']?></td>
<td><?=$notification['type']?></td>
<td><?=$notification['date']?></td>
</tr>
<?php    } ?>
</table>
<?php }else { ?>
<h3>You have zero notification</h3>
<?php  }?>

</section>
```

```
</div>

<script type="text/javascript">
var active = document.querySelector("#navList li:nth-child(4)");
active.classList.add("active");

</script>

</body>
</html>
<?php }else{
$em = "First login";
header("Location: login.php?error=$em");
exit();
}
?>
```

Profile.php

```
<?php
session_start();
if (isset($_SESSION['role']) && isset($_SESSION['id']) &&
$_SESSION['role'] == "employee") {
include "DB_connection.php";
include "app/Model/User.php";
$user = get_user_by_id($conn, $_SESSION['id']);

?>
<!DOCTYPE html>
```

```
<html>
<head>
<title>Profile</title>
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">
<link rel="stylesheet" href="css/style.css">

</head>
<body>
<input type="checkbox" id="checkbox">
<?php include "inc/header.php" ?>
<div class="body">
<?php include "inc/nav.php" ?>
<section class="section-1">
<h4 class="title">Profile <a href="edit_profile.php">Edit Profile</a></h4>
<table class="main-table" style="max-width: 300px;">
<tr>
<td>Full Name</td>
<td><?=$user['full_name']?></td>
</tr>
<tr>
<td>User name</td>
<td><?=$user['username']?></td>
</tr>
<tr>
<td>Joined At</td>
<td><?=$user['created_at']?></td>
```

```
</tr>
</table>

</section>
</div>

<script type="text/javascript">
var active = document.querySelector("#navList li:nth-child(3)");
active.classList.add("active");
</script>
</body>
</html>
<?php }else{
$em = "First login";
header("Location: login.php?error=$em");
exit();
}
?>
```

Deadline.php

```
<?php
session_start();
if (isset($_SESSION['role']) && isset($_SESSION['id']) &&
$_SESSION['role'] == "admin") {
include "DB_connection.php";
include "app/Model/Task.php";
include "app/Model/User.php";
```

```
$text = "All Task";
if (isset($_GET['due_date']) &&  $_GET['due_date'] == "Due Today") {
$text = "Due Today";
$tasks = get_all_tasks_due_today($conn);
$num_task = count_tasks_due_today($conn);

}else if (isset($_GET['due_date']) &&  $_GET['due_date'] == "Overdue") {
$text = "Overdue";
$tasks = get_all_tasks_overdue($conn);
$num_task = count_tasks_overdue($conn);

}else if (isset($_GET['due_date']) &&  $_GET['due_date'] == "No Deadline")
{
$text = "No Deadline";
$tasks = get_all_tasks_NoDeadline($conn);
$num_task = count_tasks_NoDeadline($conn);

}else{
$tasks = get_all_tasks($conn);
$num_task = count_tasks($conn);
}
$users = get_all_users($conn);



?>
<!DOCTYPE html>
<html>
```

```
<head>
<title>All Tasks</title>
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/font-
awesome/4.7.0/css/font-awesome.min.css">
<link rel="stylesheet" href="css/style.css">

</head>
<body>
<input type="checkbox" id="checkbox">
<?php include "inc/header.php" ?>
<div class="body">
<?php include "inc/nav.php" ?>
<section class="section-1">
<h4 class="title-2">
<a href="create_task.php" class="btn">Create Task</a>
<a href="tasks.php?due_date=Due Today">Due Today</a>
<a href="tasks.php?due_date=Overdue">Overdue</a>
<a href="tasks.php?due_date=No Deadline">No Deadline</a>
<a href="tasks.php">All Tasks</a>

</h4>
<h4 class="title-2"><?=$text?> (<?=$num_task?>)</h4>
<?php if (isset($_GET['success'])) {?>
<div class="success" role="alert">
<?php echo stripcslashes($_GET['success']); ?>
</div>
<?php } ?>
```

```
<?php if ($tasks != 0) { ?>
<table class="main-table">
<tr>
<th>#</th>
<th>Title</th>
<th>Description</th>
<th>Assigned To</th>
<th>Due Date</th>
<th>Status</th>
<th>Action</th>
</tr>
<?php $i=0; foreach ($tasks as $task) { ?>
<tr>
<td><?=++$i?></td>
<td><?=$task['title']?></td>
<td><?=$task['description']?></td>
<td>
<?php
foreach ($users as $user) {
if($user['id'] == $task['assigned_to']){
echo $user['full_name'];
}}?>
</td>
<td><?php if($task['due_date'] == "") echo "No Deadline";
else echo $task['due_date'];
?></td>
<td><?=$task['status']?></td>
```

```
<td>
<a href="edit-task.php?id=<?=$task['id']?>" class="edit-btn">Edit</a>
<a href="delete-task.php?id=<?=$task['id']?>" class="delete-btn">Delete</a>
</td>
</tr>
<?php    } ?>
</table>
<?php }else { ?>
<h3>Empty</h3>
<?php  }?>

</section>
</div>

<script type="text/javascript">
var active = document.querySelector("#navList li:nth-child(4)");
active.classList.add("active");
</script>
</body>
</html>
<?php }else{
$em = "First login";
header("Location: login.php?error=$em");
exit();
}
?>
```

User.php

```
<?php
session_start();
if (isset($_SESSION['role']) && isset($_SESSION['id']) &&
$_SESSION['role'] == "admin") {
include "DB_connection.php";
include "app/Model/User.php";

$users = get_all_users($conn);

?>
<!DOCTYPE html>
<html>
<head>
<title>Manage Users</title>
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/font-
awesome/4.7.0/css/font-awesome.min.css">
<link rel="stylesheet" href="css/style.css">

</head>
<body>
<input type="checkbox" id="checkbox">
<?php include "inc/header.php" ?>
<div class="body">
<?php include "inc/nav.php" ?>
<section class="section-1">
<h4 class="title">Manage Users <a href="add-user.php">Add User</a></h4>
<?php if (isset($_GET['success'])) {?>
```

```
<div class="success" role="alert">
<?php echo stripcslashes($_GET['success']); ?>
</div>
<?php } ?>
<?php if ($users != 0) { ?>
<table class="main-table">
<tr>
<th>#</th>
<th>Full Name</th>
<th>Username</th>
<th>role</th>
<th>Action</th>
</tr>
<?php $i=0; foreach ($users as $user) { ?>
<tr>
<td><?=++$i?></td>
<td><?=$user['full_name']?></td>
<td><?=$user['username']?></td>
<td><?=$user['role']?></td>
<td>
<a href="edit-user.php?id=<?=$user['id']?>" class="edit-btn">Edit</a>
<a href="delete-user.php?id=<?=$user['id']?>" class="delete-btn">Delete</a>
</td>
</tr>
<?php    } ?>
</table>
<?php }else { ?>
```

```
<h3>Empty</h3>
<?php  }?>

</section>
</div>

<script type="text/javascript">
var active = document.querySelector("#navList li:nth-child(2)");
active.classList.add("active");
</script>
</body>
</html>
<?php }else{
$em = "First login";
header("Location: login.php?error=$em");
exit();
}
?>
```

Add_user.php

```
<?php
session_start();
if (isset($_SESSION['role']) && isset($_SESSION['id']) &&
$_SESSION['role'] == "admin") {

?>
<!DOCTYPE html>
<html>
```

```
<head>
<title>Add User</title>
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/font-
awesome/4.7.0/css/font-awesome.min.css">
<link rel="stylesheet" href="css/style.css">

</head>
<body>
<input type="checkbox" id="checkbox">
<?php include "inc/header.php" ?>
<div class="body">
<?php include "inc/nav.php" ?>
<section class="section-1">
<h4 class="title">Add Users <a href="user.php">Users</a></h4>
<form class="form-1"
method="POST"
action="app/add-user.php">
<?php if (isset($_GET['error'])) {?>
<div class="danger" role="alert">
<?php echo stripcslashes($_GET['error']); ?>
</div>
<?php } ?>

<?php if (isset($_GET['success'])) {?>
<div class="success" role="alert">
<?php echo stripcslashes($_GET['success']); ?>
</div>
```

```
<?php } ?>
<div class="input-holder">
<lable>Full Name</lable>
<input type="text" name="full_name" class="input-1" placeholder="Full
Name"><br>
</div>
<div class="input-holder">
<lable>Username</lable>
<input type="text" name="user_name" class="input-1"
placeholder="Username"><br>
</div>
<div class="input-holder">
<lable>Password</lable>
<input type="text" name="password" class="input-1"
placeholder="Password"><br>
</div>

<button class="edit-btn">Add</button>
</form>

</section>
</div>

<script type="text/javascript">
var active = document.querySelector("#navList li:nth-child(2)");
active.classList.add("active");
</script>
```

```
</body>
</html>
<?php }else{
$em = "First login";
header("Location: login.php?error=$em");
exit();
}
?>
```

Update_task_employee.php

```
<?php
session_start();
if (isset($_SESSION['role']) && isset($_SESSION['id'])) {

if (isset($_POST['id']) && isset($_POST['status']) && $_SESSION['role'] ==
'employee') {
include "../DB_connection.php";

function validate_input($data) {
$data = trim($data);
$data = stripslashes($data);
$data = htmlspecialchars($data);
return $data;
}

$status = validate_input($_POST['status']);
$id = validate_input($_POST['id']);
```

```
if (empty($status)) {
$em = "status is required";
header("Location: ../edit-task-employee.php?error=$em&id=$id");
exit();
}else {

include "Model/Task.php";

$data = array($status, $id);
update_task_status($conn, $data);

$em = "Task updated successfully";
header("Location: ../edit-task-employee.php?success=$em&id=$id");
exit();



}
}else {
$em = "Unknown error occurred";
header("Location: ../edit-task-employee.php?error=$em");
exit();
}

}else{
$em = "First login";
header("Location: ../login.php?error=$em");
exit();
```

```
}
```

Update_task.php

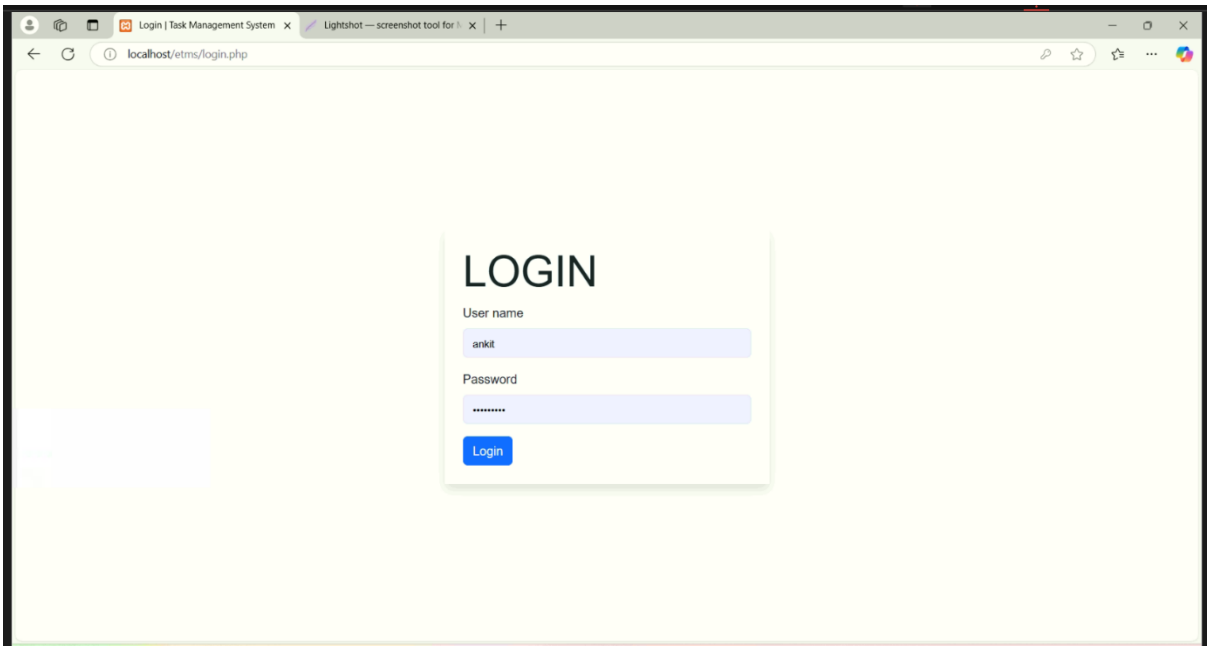
```
<?php
session_start();
if (isset($_SESSION['role']) && isset($_SESSION['id'])) {

if (isset($_POST['id']) && isset($_POST['title']) &&
isset($_POST['description']) && isset($_POST['assigned_to']) &&
$_SESSION['role'] == 'admin'&& isset($_POST['due_date'])) {
include "../DB_connection.php";

function validate_input($data) {
$data = trim($data);
$data = stripslashes($data);
$data = htmlspecialchars($data);
return $data;
}

$title = validate_input($_POST['title']);
$description = validate_input($_POST['description']);
$assigned_to = validate_input($_POST['assigned_to']);
$id = validate_input($_POST['id']);
$due_date = validate_input($_POST['due_date']);

if (empty($title)) {
$em = "Title is required";
header("Location: ../edit-task.php?error=$em&id=$id");
exit();
}else if (empty($description)) {
$em = "Description is required";
header("Location: ../edit-task.php?error=$em&id=$id");
exit();
}else if ($assigned_to == 0) {
$em = "Select User";
header("Location: ../edit-task.php?error=$em&id=$id");
exit();
}else {

include "Model/Task.php";
```

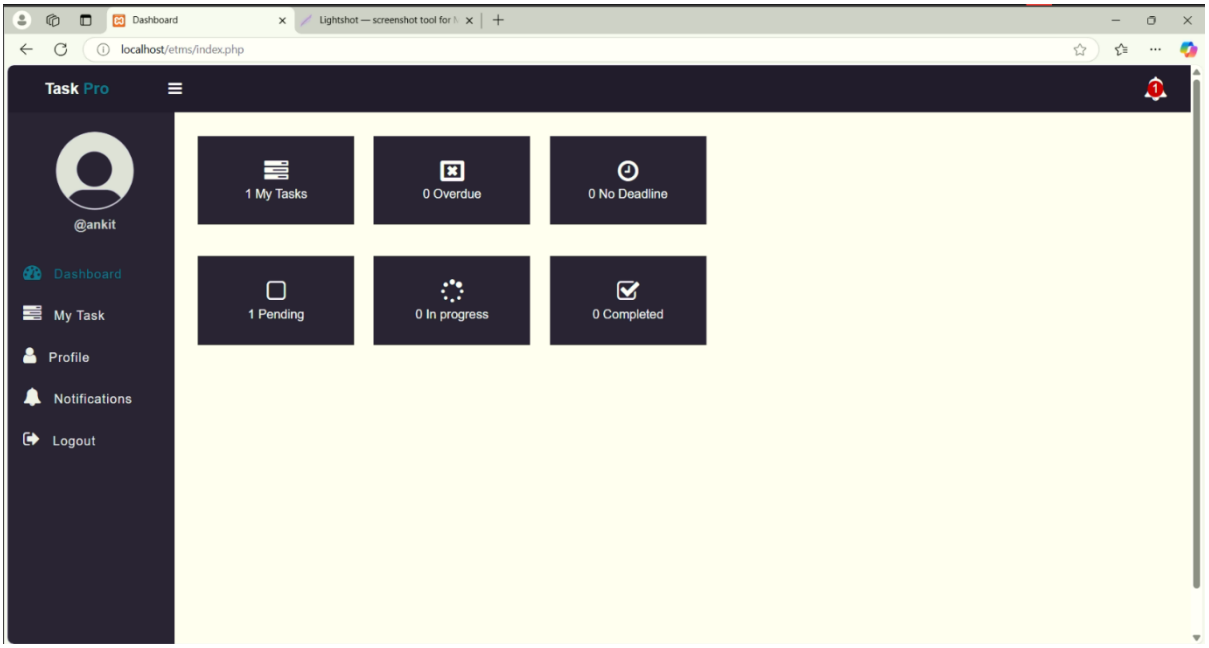
```
$data = array($title, $description, $assigned_to, $due_date, $id);
update_task($conn, $data);

$em = "Task updated successfully";
header("Location: ../edit-task.php?success=$em&id=$id");
exit();



}
}else {
$em = "Unknown error occurred";
header("Location: ../edit-task.php?error=$em");
exit();
}

}else{
$em = "First login";
header("Location: ../login.php?error=$em");
exit();
}
```

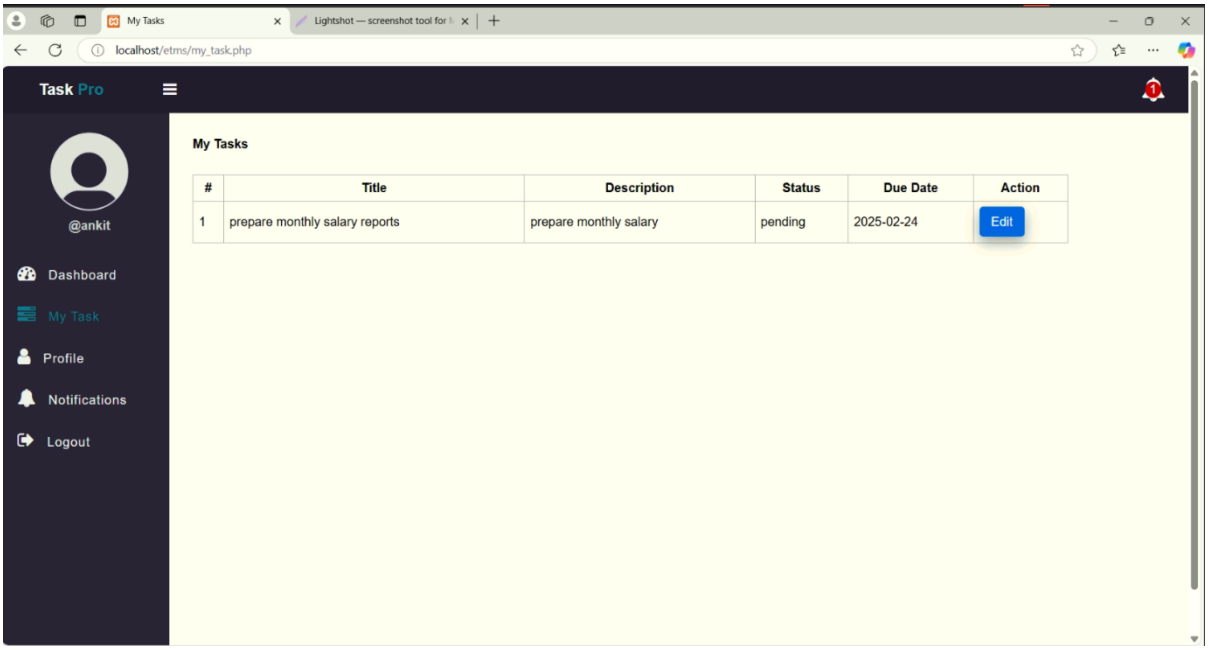
Update_user.php

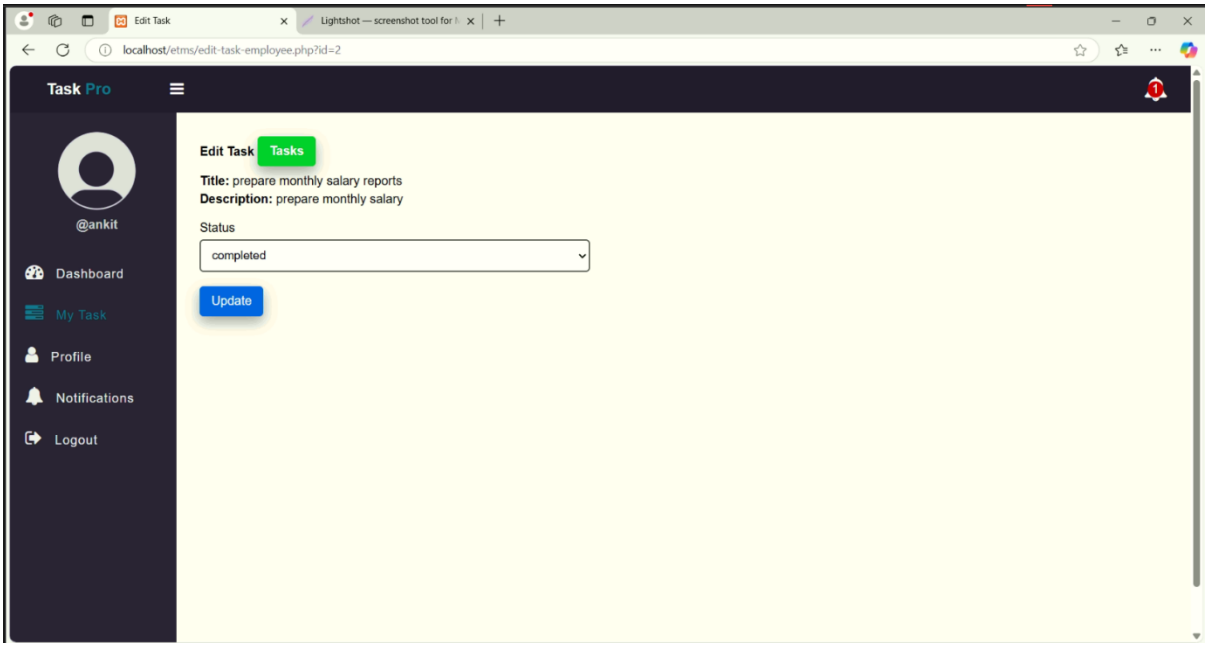
```
<?php
session_start();
if (isset($_SESSION['role']) && isset($_SESSION['id'])) {

if (isset($_POST['user_name']) && isset($_POST['password']) &&
isset($_POST['full_name']) && $_SESSION['role'] == 'admin') {
include "../DB_connection.php";

function validate_input($data) {
$data = trim($data);
$data = stripslashes($data);
$data = htmlspecialchars($data);
return $data;
}

$user_name = validate_input($_POST['user_name']);
$password = validate_input($_POST['password']);
$full_name = validate_input($_POST['full_name']);
$id = validate_input($_POST['id']);
```

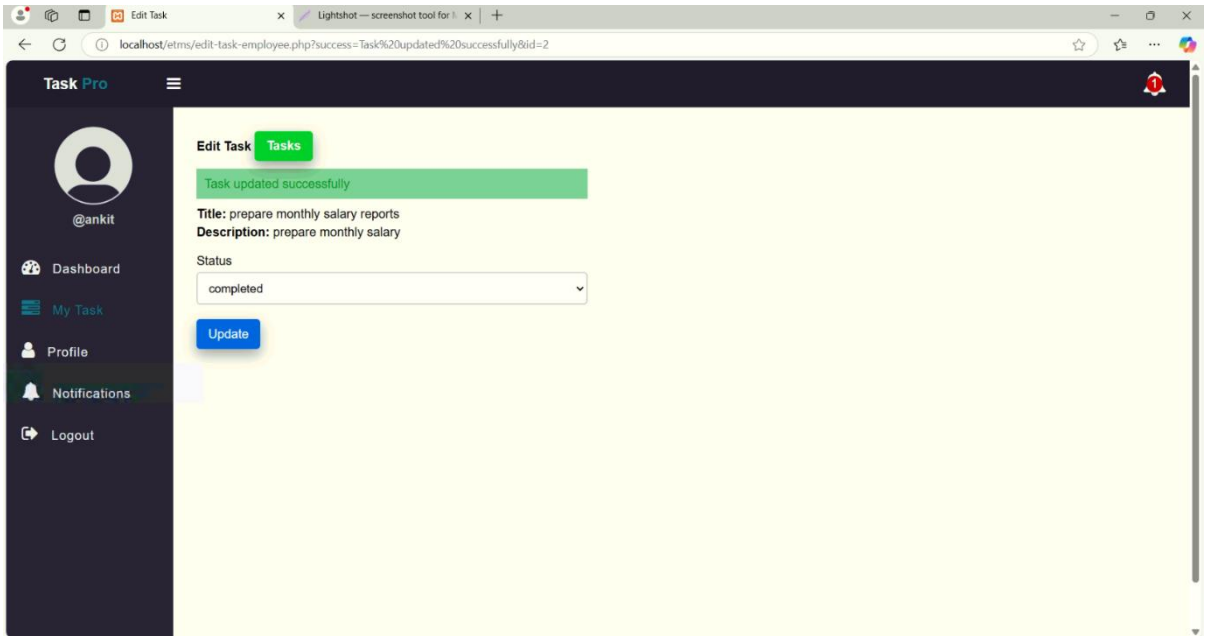
```
if (empty($user_name)) {
$em = "User name is required";
header("Location: ../edit-user.php?error=$em&id=$id");
exit();
}else if (empty($password)) {
$em = "Password is required";
header("Location: ../edit-user.php?error=$em&id=$id");
exit();
}else if (empty($full_name)) {
$em = "Full name is required";
header("Location: ../edit-user.php?error=$em&id=$id");
exit();
}else {

include "Model/User.php";
$password = password_hash($password, PASSWORD_DEFAULT);

$data = array($full_name, $user_name, $password, "employee", $id,
"employee");
update_user($conn, $data);

$em = "User created successfully";
header("Location: ../edit-user.php?success=$em&id=$id");
exit();



}
}else {
$em = "Unknown error occurred";
header("Location: ../edit-user.php?error=$em");
exit();
}

}else{
$em = "First login";
header("Location: ../edit-user.php?error=$em");
exit();
}
```

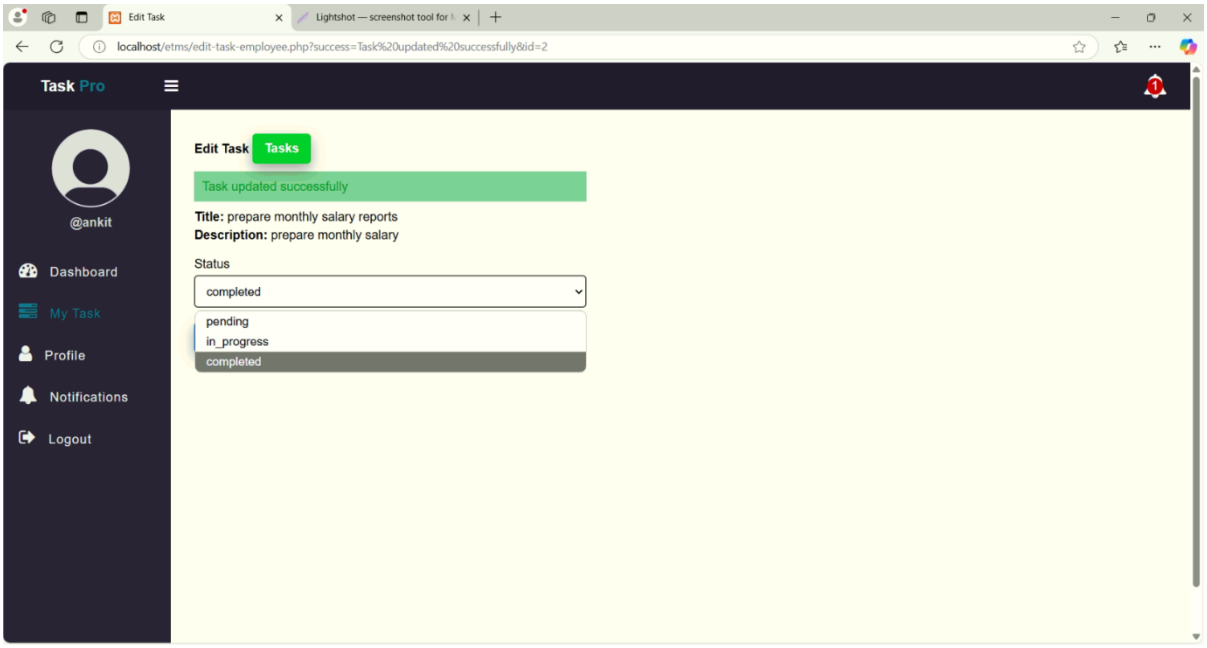

Login | Task Management System
Lightshot — screenshot tool for
localhost/etms/login.php
LOGIN
User name
ankit
Password
Login

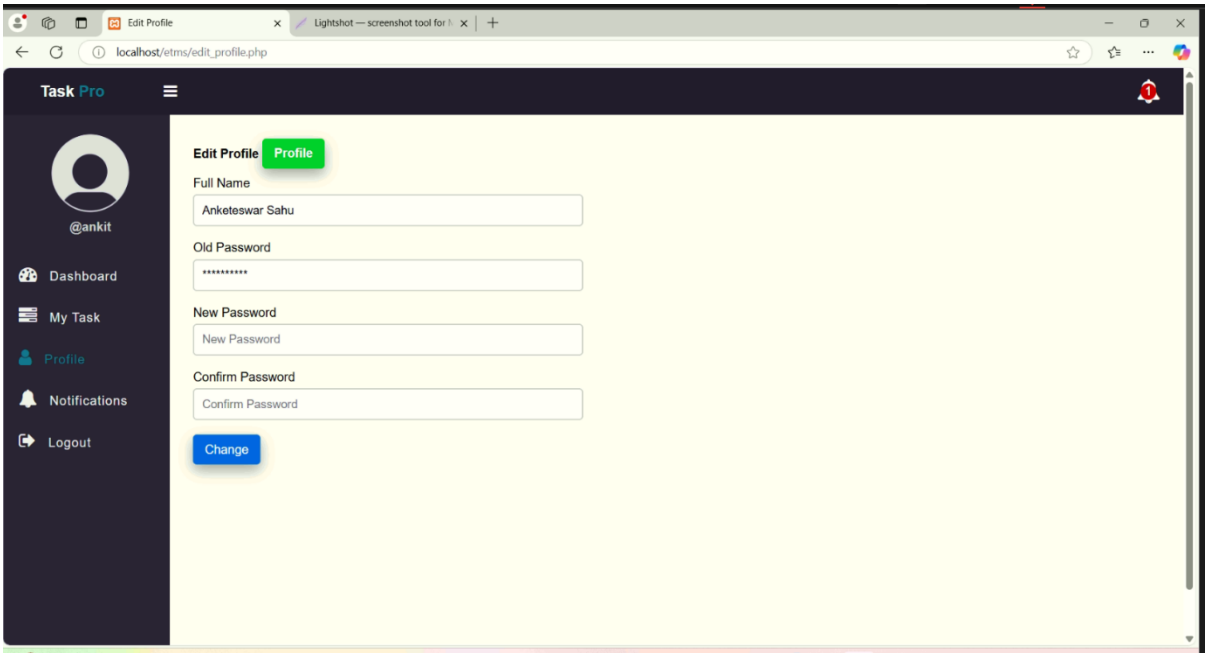

Dashboard
localhost/etms/index.php
Task Pro
@ankit
Dashboard
My Task
Profile
Notifications
Logout
1 My Tasks
0 Overdue
0 No Deadline
1 Pending
0 In progress
0 Completed

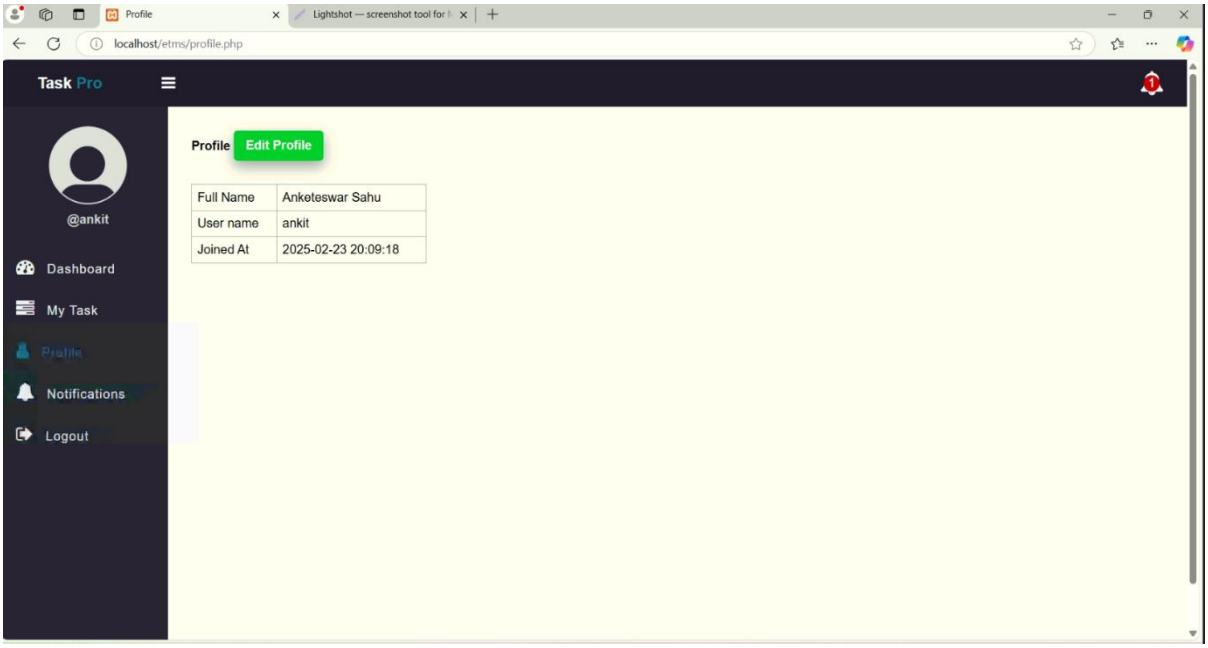

My Tasks
localhost/etms/my_task.php
Task Pro
@ankit
Dashboard
My Task
Profile
Notifications
Logout
My Tasks
Title Description Status Due Date Action
1 prepare monthly salary reports prepare monthly salary pending 2025-02-24 Edit

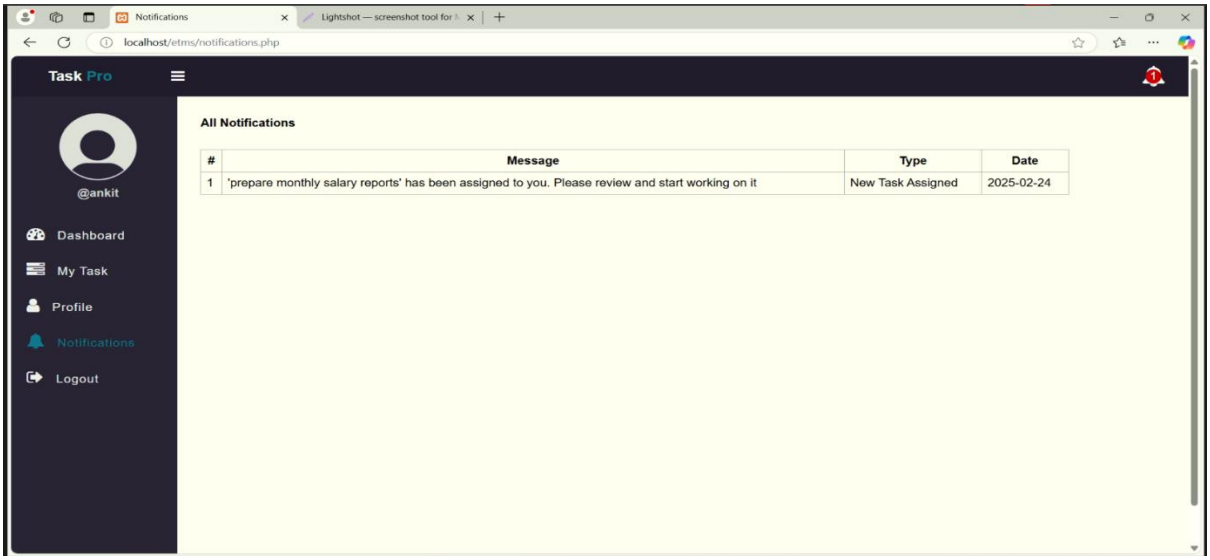

Edit Task
localhost/etms/edit-task-employee.php?id=2
Task Pro
@ankit
Dashboard
My Task
Profile
Notifications
Logout
Edit Task
Tasks
Title: prepare monthly salary reports
Description: prepare monthly salary
Status
completed
Update

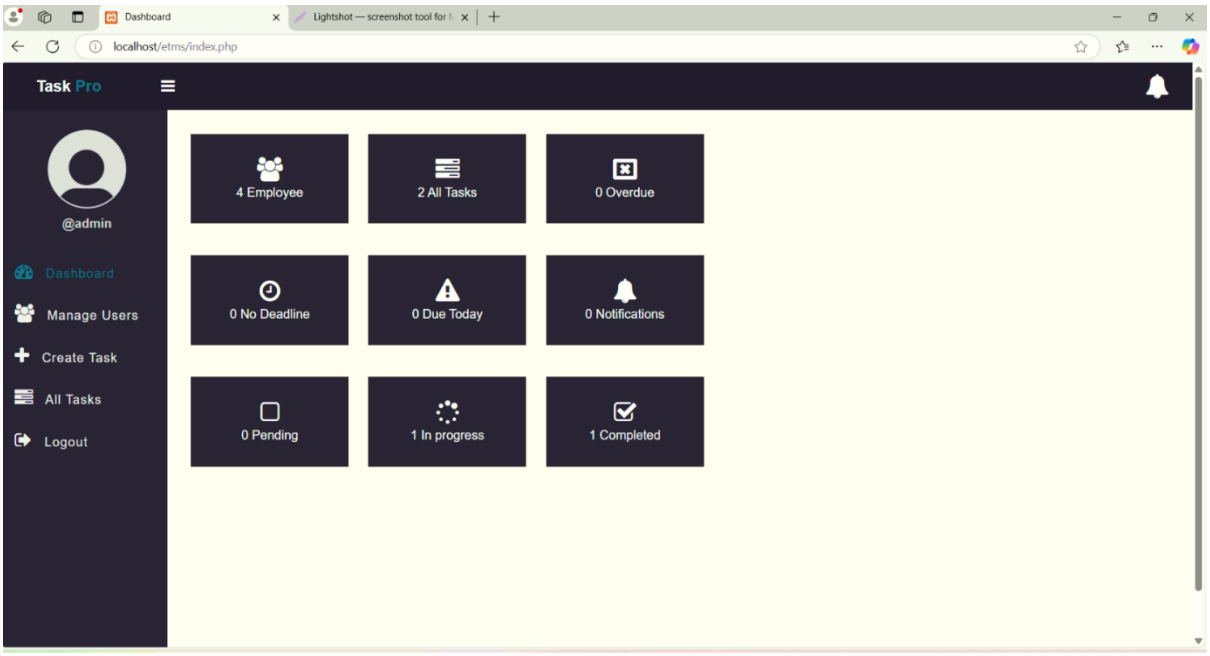

Edit Task
localhost/etms/edit-task-employee.php?success=Task%20updated%20successfully&id=2
Task Pro
@ankit
Dashboard
My Task
Profile
Notifications
Logout
Edit Task
Tasks
Task updated successfully
Title: prepare monthly salary reports
Description: prepare monthly salary
Status
completed
Update

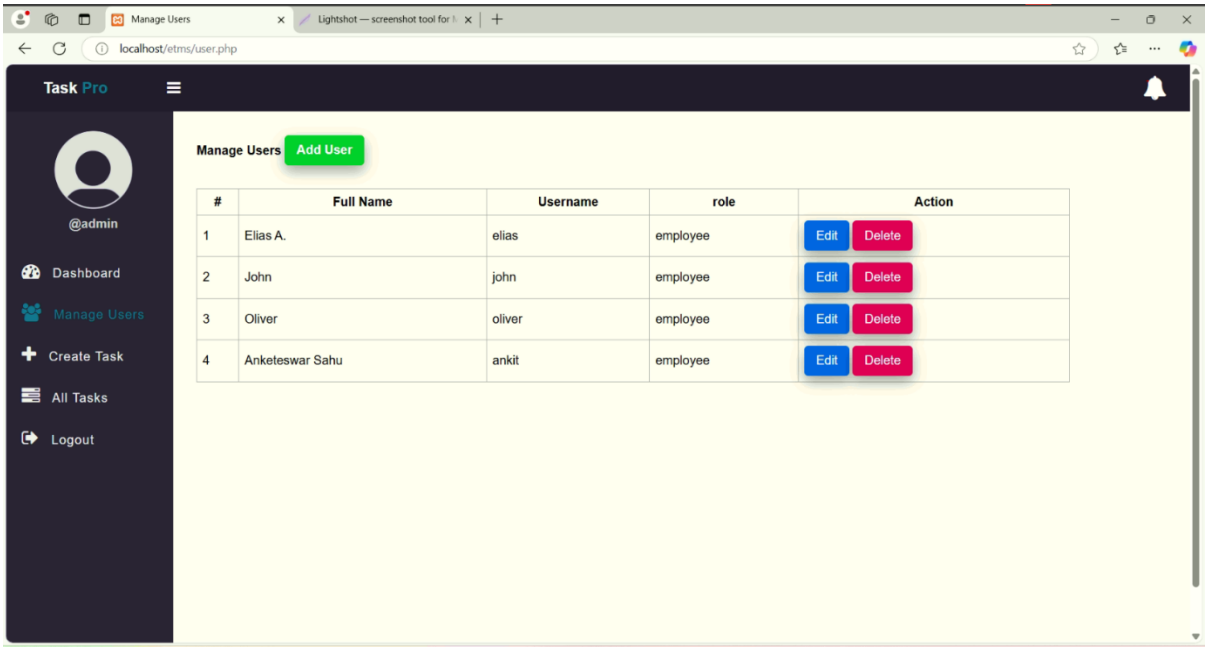

Edit Task
Lightshot — screenshot tool for
localhost/etms/edit-task-employee.php?success=Task%20updated%20successfully&id=2
Task Pro
@ankit
Dashboard
My Task
Profile
Notifications
Logout
Edit Task
Tasks
Task updated successfully
Title: prepare monthly salary reports
Description: prepare monthly salary
Status
completed
pending
in_progress
completed

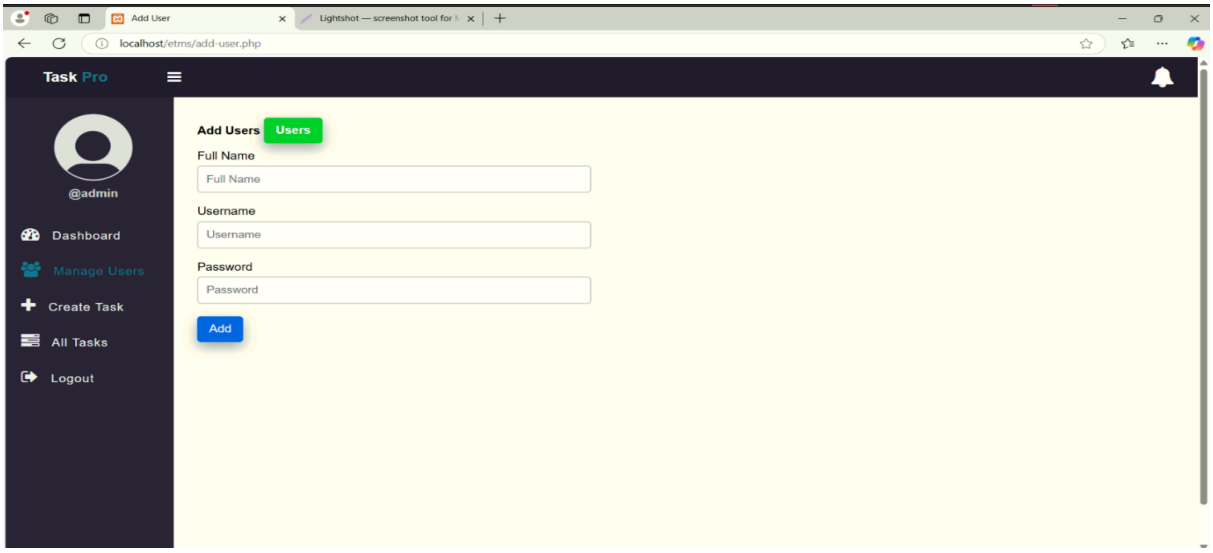

Edit Profile
Lightshot — screenshot tool for
localhost/etms/edit_profile.php
Task Pro
@ankit
Dashboard
My Task
Profile
Notifications
Logout
Edit Profile
Profile
Full Name
Anketeswar Sahu
Old Password
New Password
New Password
Confirm Password
Confirm Password
Change

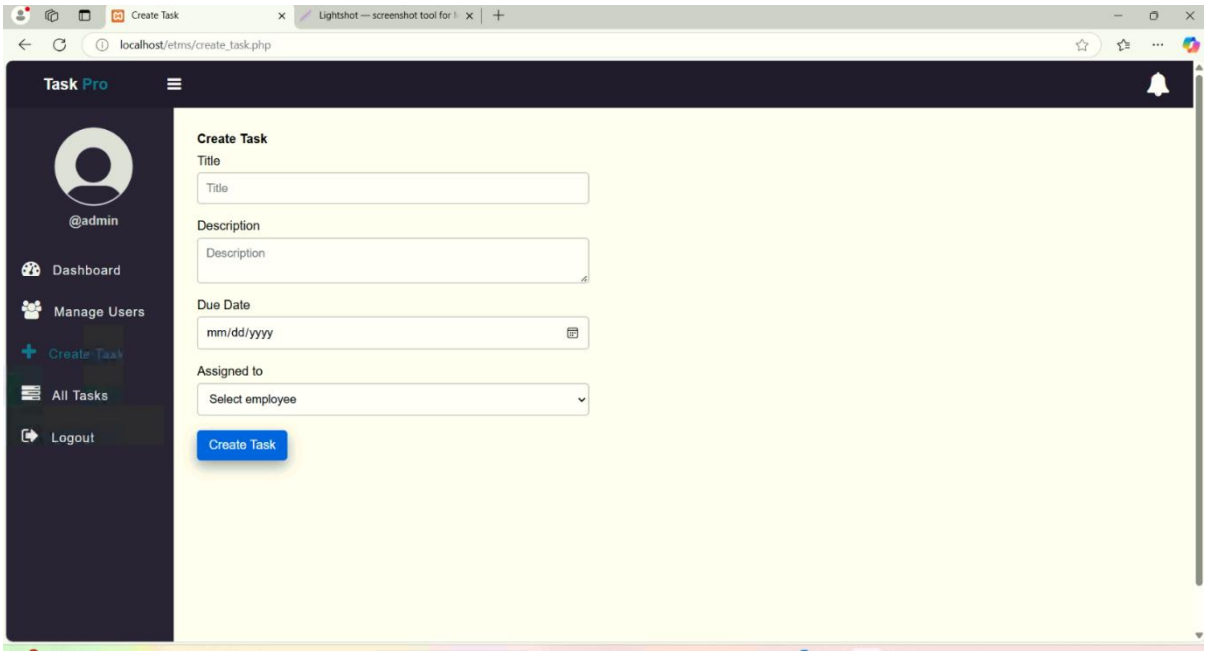

Profile
Lightshot — screenshot tool for
localhost/etms/profile.php
Task Pro
@ankit
Dashboard
My Task
Profile
Notifications
Logout
Profile
Edit Profile
Full Name
Anketeswar Sahu
User name
ankit
Joined At
2025-02-23 20:09:18

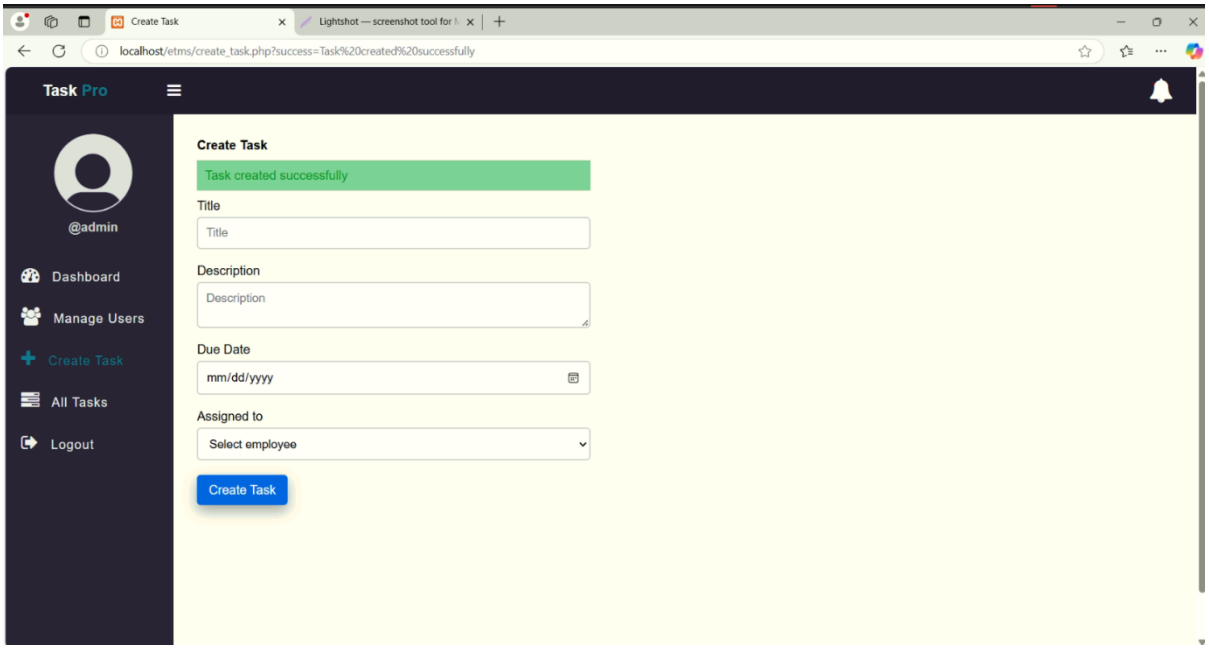

Notifications
Lightshot — screenshot tool for
localhost/etms/notifications.php
Task Pro
@ankit
Dashboard
My Task
Profile
Notifications
Logout
All Notifications
Message Type Date
1 'prepare monthly salary reports' has been assigned to you. Please review and start working on it New Task Assigned 2025-02-24

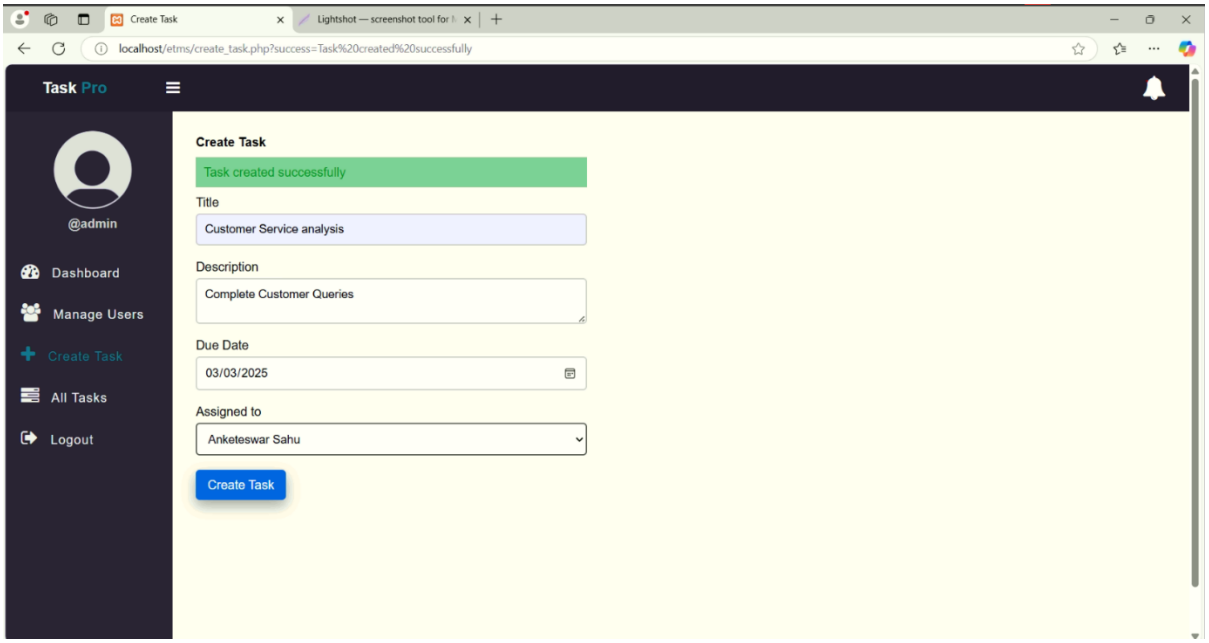

Dashboard
Lightshot — screenshot tool for
localhost/etms/index.php
Task Pro
@admin
Dashboard
Manage Users
Create Task
All Tasks
Logout
4 Employee
2 All Tasks
0 Overdue
0 No Deadline
0 Due Today
0 Notifications
0 Pending
1 In progress
1 Completed

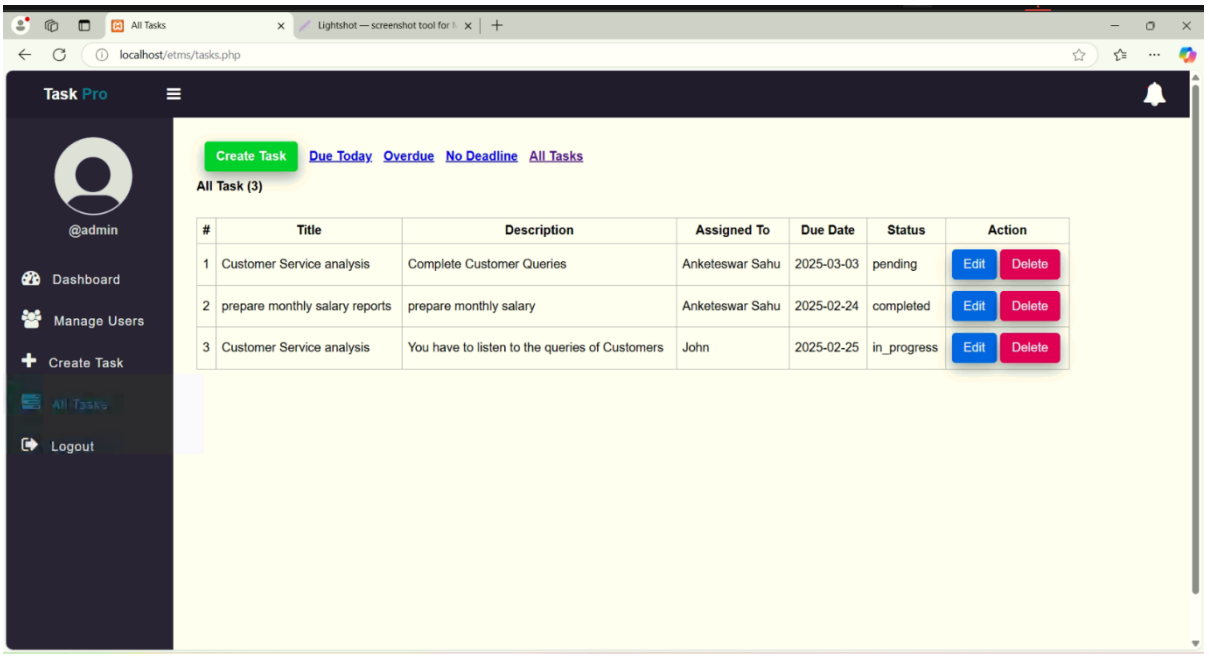

Manage Users
Lightshot — screenshot tool for
localhost/etms/user.php
Task Pro
@admin
Dashboard
Manage Users
Create Task
All Tasks
Logout
Manage Users Add User
Full Name Username role Action
1 Elias A. elias employee Edit Delete
2 John john employee Edit Delete
3 Oliver oliver employee Edit Delete
4 Anketeswar Sahu ankit employee Edit Delete


Add User
Lightshot — screenshot tool for
localhost/etms/add-user.php
Task Pro
@admin
Dashboard
Manage Users
Create Task
All Tasks
Logout
Add Users
Users
Full Name
Full Name
Username
Username
Password
Password
Add


Create Task
Lightshot — screenshot tool for
localhost/etms/create_task.php
Task Pro
@admin
Dashboard
Manage Users
Create Task
All Tasks
Logout
Create Task
Title
Title
Description
Description
Due Date
mm/dd/yyyy
Assigned to
Select employee
Create Task


Create Task
Lightshot — screenshot tool for
localhost/etms/create_task.php?success=Task%20created%20successfully
Task Pro
@admin
Dashboard
Manage Users
Create Task
All Tasks
Logout
Create Task
Task created successfully
Title
Title
Description
Description
Due Date
mm/dd/yyyy
Assigned to
Select employee
Create Task





Implementation and Testing of Employee Task Management System

The Employee Task Management System is designed to help employees manage their tasks efficiently while enabling managers to track the progress and completion of these tasks. The implementation follows a structured approach to ensure proper functioning and usability. The process involves building features for task creation, assignment, progress tracking, and reporting.

1. Implementation

a. System Design

- User Roles:
    - Admin: Can create and assign tasks, manage employees, and view reports.
    - Manager: Can assign tasks to employees, track progress, and approve tasks.

- Employee: Can view, update the status of, and complete assigned tasks.

- Core Features:

1. Task Creation and Assignment:
    - Admin and Manager can create tasks and assign them to employees.
    - Each task includes a title, description, deadline, and priority level.
2. Task Progress:
    - Employees can update the status of their tasks (e.g., "Not Started", "In Progress", "Completed").
    - Managers can track the progress of tasks and reassign tasks if needed.
3. Task Reporting:
    - A report section that shows task statistics, such as completion status and upcoming deadlines.
4. Notifications:
    - Employees are notified when new tasks are assigned or when deadlines are approaching.
5. Authentication & Authorization:
    - Each user (Admin, Manager, Employee) logs into the system with different levels of access control.

b. Technology Stack:

- Backend:
    - Language: Python/JavaScript (Node.js)
    - Framework: Django (Python) / Express.js (Node.js)
    - Database: MySQL/PostgreSQL
    - Authentication: JWT (JSON Web Tokens)
- Frontend:
    - HTML/CSS/JavaScript
    - Framework: React.js or Angular
- Testing:
    - Unit Testing: PyTest (Python), Jest (JavaScript)
    - Integration Testing: Postman, Selenium
    - Database Testing: MySQL Workbench

c. Task Flow Example:

- Admin logs in and creates a task for a specific employee.
- Manager reviews the task assignment, makes changes if necessary, and sets deadlines.
- Employee logs in, views the assigned task, updates the task's progress, and marks it as completed when done.

2. Testing

Testing is crucial to ensure the system performs as expected under different conditions. Below are the types of tests performed on the Employee Task Management System.

a. Unit Testing

- Purpose: Ensure each component (function or class) behaves as expected in isolation.
- Examples:
    1. Task Creation Function: Verify that the system creates tasks with the correct attributes (title, description, due date).
    2. Task Assignment Logic: Ensure that tasks are correctly assigned to employees based on their role and availability.
    3. Status Update Logic: Verify that the status of a task can be updated by the employee and that changes are reflected in the system.

b. Integration Testing

- Purpose: Test how different components of the system work together.
- Examples:
    1. Task Assignment and Notification: After a task is assigned to an employee, the system sends a notification, and the task appears in the employee's task list.
    2. Manager's Approval Workflow: Ensure that after an employee marks a task as completed, the manager can approve the completion and the task moves to the final status.

c. Functional Testing

- Purpose: Ensure that the system meets the functional requirements.
- Examples:
    1. Login and Authentication: Verify that users (Admin, Manager, Employee) can log in and are assigned the correct roles and permissions.
    2. Task Assignment: Test whether tasks can be assigned to the correct employees based on their role and whether they receive notifications.
    3. Task Tracking: Ensure that managers can track employee task progress and receive notifications when tasks are updated.

d. End-to-End Testing

- Purpose: Test the system from a user perspective to ensure everything works seamlessly from login to task completion.
- Example Test Case:
    - Step 1: Admin logs in, creates a task, and assigns it to an employee.
    - Step 2: Employee logs in, views the task, updates its status to "In Progress."
    - Step 3: Manager logs in, tracks the progress, and approves the task completion.

- Expected Outcome: The task is marked as complete, and both the employee and manager can view its status.

e. Performance Testing

- Purpose: Ensure the system can handle a large number of tasks and users without performance degradation.
- Tools: Apache JMeter, LoadRunner
- Example Test Case:
  - Simulate 1000 users logging in and creating tasks simultaneously.
  - Measure response times and ensure the system remains responsive.

f. Security Testing

- Purpose: Ensure the system is secure from unauthorized access and data breaches.
- Examples:
  1. Role-Based Access Control (RBAC): Verify that only users with the proper roles can access certain resources (e.g., only Admins can create tasks, only Managers can approve tasks).
  2. SQL Injection Protection: Ensure input fields are sanitized to prevent SQL injection.
  3. Session Management: Ensure that users are logged out after a certain period of inactivity, and JWT tokens are securely stored.

g. User Acceptance Testing (UAT)

- Purpose: Validate the system's functionality with real users to ensure it meets their needs.
- Process:
  - Invite a group of end-users (e.g., managers, employees) to interact with the system.
  - Gather feedback on usability, and make improvements based on their experiences.
  - Validate if the system's workflow aligns with user expectations.

---

3. Testing Results

- Unit Testing: All individual components functioned as expected. No major issues were identified.
- Integration Testing: The system successfully integrated task creation, assignment, progress tracking, and reporting features.
- Functional Testing: All primary functions (task creation, status updates, notifications) worked as intended.
- End-to-End Testing: All steps in the task management process, from task creation to completion, functioned smoothly.
- Performance Testing: The system handled up to 1000 concurrent users without significant delays.
- Security Testing: All security measures (role-based access, input validation, session timeout) were correctly implemented.

Conclusion

The Employee Task Management System has been successfully implemented and tested to ensure it meets the functional, security, and performance requirements. The system is user-friendly, efficient, and can scale with the number of users and tasks. Continuous monitoring and updates will ensure it remains reliable and secure as the organization grows.

Limitations of the Employee Task Management System

While the Employee Task Management System is designed to improve task tracking and team productivity, it comes with certain limitations that need to be considered during implementation and use. Here are some of the important points regarding the limitations of the project:

---

1. Limited User Roles and Permissions

- Limitation: The system typically supports a limited number of predefined user roles, such as Admin, Manager, and Employee. This can lead to challenges in more complex organizational structures where additional roles (e.g., HR, Team Leads, etc.) may be needed.
- Impact: May require additional customization to accommodate more granular role-based access control for larger teams or different departments.

---

2. Scalability Issues for Large Teams

- Limitation: While the system works well for small to medium-sized teams, handling a large number of users, tasks, and data may lead to performance issues such as slower response times or even crashes under heavy load.
- Impact: As the company grows and the volume of tasks increases, the system may require optimizations or infrastructure improvements (e.g., moving to a more scalable cloud-based architecture or upgrading the database).

---

3. Lack of Advanced Task Management Features

- Limitation: Basic task management features are generally provided, such as task creation, assignment, progress tracking, and completion. However, advanced features like time tracking, Gantt charts, dependencies between tasks, and detailed reporting might be missing.
- Impact: For teams or organizations with more complex project management needs, this could limit the system's functionality and effectiveness in managing larger or multi-phase projects.

---

4. No Integration with Other Tools

- Limitation: The system may not offer out-of-the-box integration with other commonly used project management or productivity tools like Slack, Trello, Google Calendar, Jira, or GitHub.
- Impact: Employees and managers may need to manually update information in the system, leading to inefficiencies. Additional effort would be needed to integrate or sync data with other tools used within the organization.

---

5. Limited Reporting and Analytics

- Limitation: Reporting features may be basic, offering only simple task completion status or deadlines, and may lack advanced analytics like task velocity, productivity trends, or resource allocation optimization.
- Impact: Managers might find it difficult to gain deep insights into team performance or project progress, and making data-driven decisions could become more challenging without these advanced analytics.

---

6. Dependency on Manual Updates

- Limitation: Employees are required to manually update their task statuses and progress. If they forget to update or fail to do so regularly, the system might not provide an accurate reflection of the team's status.
- Impact: The system may not provide real-time updates or accurate information, which could lead to misunderstandings about task deadlines or work completion.

---

7. Lack of Mobile Access

- Limitation: If the system is a web-based application without a mobile-friendly interface or mobile app, employees who are on the go may find it inconvenient to access or update their tasks.
- Impact: Limited accessibility may result in delays in task updates or missed deadlines, especially for employees who need to access tasks while away from their desk or office.

---

8. Limited Customization

- Limitation: The system may offer limited customization options, such as predefined task templates, priority levels, or categories. The inability to tailor workflows and task structures to specific business needs might be restrictive.
- Impact: Different teams with unique workflows or task tracking requirements may not be able to fully adapt the system to their needs without major modifications.

---

9. Security Risks with User Data

- Limitation: While the system may implement basic security features (e.g., role-based access), it could still be vulnerable to security threats, particularly if sensitive employee information or task data is not adequately protected.
- Impact: If security measures are not properly implemented, unauthorized users may access confidential data, leading to potential data breaches or misuse.

---

10. Training and User Adoption
    - Limitation: The system might have a learning curve for some employees, especially those who are not familiar with technology or project management systems. This could require additional training efforts and resources.
    - Impact: Without proper training, users may struggle to use the system effectively, leading to suboptimal task management and inefficiency.

---

11. Offline Functionality
    - Limitation: If the system is entirely cloud-based, employees might not be able to access or update their tasks when there is no internet connection.
    - Impact: For remote workers or those in areas with unreliable internet, this could limit the functionality of the system and disrupt their ability to update or track their tasks.

---

12. Task Dependency and Workflow Limitations
    - Limitation: The system may not allow for the creation of task dependencies or the tracking of tasks in a multi-stage workflow. This means employees can only work on tasks in isolation, without linking them to other related tasks.
    - Impact: In complex projects, tasks might be dependent on the completion of others, and the lack of this feature could result in delays or confusion over which tasks need to be completed first.

---

13. Task Reminders and Deadlines
    - Limitation: While notifications can be implemented, some systems may lack robust reminder features or advanced deadline management (e.g., automated rescheduling or priority adjustments based on delays).
    - Impact: Without effective deadline management or reminders, employees might miss critical tasks or deadlines, affecting overall project timelines.

---

14. Limited Feedback Mechanisms
    - Limitation: The system may lack features for feedback or communication between employees and managers, such as comments or real-time chat on tasks.

- Impact: Without an integrated communication feature, employees and managers may need to use separate platforms (e.g., email, chat apps) to discuss tasks, which could lead to fragmented communication.

Future Scope of the Employee Task Management System

The Employee Task Management System is an essential tool for improving productivity, communication, and task tracking within an organization. However, there is always room for growth and enhancement. As businesses evolve, their needs also change, and the system must be adaptable to address these future demands. Below are the important points regarding the future scope of the Employee Task Management System:

---

1. Advanced Analytics and Reporting

- Scope:
    - Integrating advanced reporting and analytics capabilities can provide deeper insights into employee performance, task completion rates, and project efficiency.
    - Features like task completion trends, employee productivity dashboards, and time-to-completion analytics could be added.
- Benefit: This would help managers and decision-makers make data-driven decisions, improve resource allocation, and identify bottlenecks or inefficiencies in the workflow.

---

2. AI and Machine Learning Integration

- Scope:
    - Integrating AI and machine learning algorithms can predict task deadlines, suggest optimal task assignments based on employee performance, or even automatically adjust task priorities.
    - Natural Language Processing (NLP) could be used for more intuitive task creation, where employees or managers describe tasks in plain language, and the system converts them into structured tasks.
- Benefit: This could significantly enhance the system's ability to automatically manage workflows, improve efficiency, and reduce human error in task planning.

---

3. Task Dependencies and Gantt Chart Integration

- Scope:

    - Implementing task dependencies, where tasks are linked based on completion order (e.g., Task B cannot start until Task A is finished), would be beneficial for larger projects.
    - Adding Gantt charts or Kanban boards for visual representation of project timelines and task progress could improve task management and help teams visually track ongoing tasks.
- Benefit: This would be especially useful for complex projects involving multiple interdependent tasks, enabling teams to have better visibility over project milestones.

---

4. Mobile Application and Offline Functionality

- Scope:
    - A mobile version of the system could be developed to provide employees with easy access to tasks and updates on the go. It could include push notifications for updates, task deadlines, and reminders.
    - Additionally, adding offline functionality would allow users to update their task status and make changes when they do not have an internet connection, and sync the data once they are online.
- Benefit: This would improve accessibility for remote workers, field employees, and those who travel frequently, ensuring they stay productive regardless of their location or network availability.

---

5. Integration with Other Productivity Tools

- Scope:
    - The system could be integrated with popular tools such as Slack, Trello, Jira, Google Calendar, and Microsoft Teams. This would allow for seamless data exchange and provide employees with a unified workspace.
    - Calendar Sync: Integrating with Google or Outlook calendars to automatically sync deadlines and meetings could also improve task management.
- Benefit: This would enhance collaboration and reduce the need for users to switch between multiple tools. It would streamline workflow by bringing different productivity tools into one platform.

---

6. Employee Feedback and Collaboration Features

- Scope:
    - Introducing collaboration features such as task comments, file attachments, or real-time discussion threads within the task interface.
    - Additionally, implementing a feedback mechanism where employees can rate the task difficulty, provide feedback on task assignments, or share suggestions for improving the system.

- Benefit: This would improve communication between team members, enhance task clarity, and enable continuous improvement of workflows and task management processes.

---

7. Automated Task Prioritization and Scheduling

- Scope:
    - The system could automatically prioritize tasks based on factors such as deadlines, task complexity, or employee workload. It could use machine learning models to optimize how tasks are scheduled and assigned.
    - Smart Notifications: Based on task deadlines and priority, the system can send personalized reminders to the employees or managers.
- Benefit: Automating task prioritization and scheduling would reduce managerial overhead, minimize task delays, and ensure that the most critical tasks are completed first.

---

8. Customizable Workflows and Task Templates

- Scope:
    - Providing the option to create custom task templates based on specific types of tasks, projects, or departments. This would allow users to set up tasks more efficiently by using pre-defined templates.
    - The system could offer workflow customization where companies can define their own task approval processes, progress tracking steps, and completion criteria.
- Benefit: This would make the system more adaptable to different industries or departments with unique task management needs and workflows.

---

9. Time Tracking and Resource Management

- Scope:
    - Introducing time-tracking features that allow employees to log hours spent on specific tasks, and managers to track the efficiency of the team and allocate resources accordingly.
    - Resource management features could also help in balancing workloads across employees, ensuring that no one is overloaded and resources are utilized effectively.
- Benefit: This feature would provide more transparency and accountability regarding task completion times, and improve resource allocation and planning.

---

10. Multi-Language and Localization Support

- Scope:
    - Expanding the system's usability by providing multi-language support and localization for different regions or countries, making it accessible to global teams.

    - Time zone synchronization could ensure that deadlines and task schedules are appropriately displayed according to each employee's local time zone.
- Benefit: This would make the system more inclusive and user-friendly for organizations with diverse, international teams.

---

11. Cloud-based Deployment and Data Backup

- Scope:
    - Migrating the system to a cloud-based platform for better scalability and reliability, allowing for continuous system updates and global access.
    - Automated backups of task data, ensuring that tasks and project information are not lost due to server crashes or data corruption.
- Benefit: Cloud deployment ensures scalability as the business grows and enhances accessibility and security for users.

---

12. Gamification and Employee Engagement Features

- Scope:
    - Adding gamification elements, such as achievement badges, leaderboards, or task completion milestones, could help keep employees motivated and engaged in their tasks.
    - Reward systems for completing tasks or meeting deadlines could further improve team morale and drive.
- Benefit: This could increase employee engagement, encourage productivity, and foster a positive, competitive work culture.

---

13. Integration with HR and Payroll Systems

- Scope:
    - Integrating the task management system with HR systems to track the time spent on tasks and projects, providing data for performance appraisals or payroll processing.
    - This could also help in employee skill development, identifying strengths and areas for improvement based on task performance.
- Benefit: Streamlining HR and payroll processes while improving performance reviews and employee recognition.

In the context of an employee task management system project, the reference part refers to key concepts, tools, and practices that should be considered when developing the system. Here are some important points that can serve as a reference for such a project:

1. User Roles and Permissions

- Employees: The users who are assigned tasks. Their roles typically include viewing tasks, updating progress, and marking tasks as complete.
- Managers/Team Leads: Users who can assign tasks to employees, monitor task progress, and oversee the completion of tasks.
- Admins: Higher-level users who can configure the system settings, add or remove users, and manage overall system functionality.

2. Task Assignment and Workflow
   - Task Creation: Managers or admins should have the ability to create tasks, assign them to employees, set deadlines, and define priorities.
   - Task Progress Tracking: Employees should be able to update the status of tasks (e.g., "in progress," "completed," "on hold").
   - Task Notifications: Employees and managers should receive notifications (via email or in-app) about task updates, deadlines, and changes.
   - Priority Levels: Tasks should be categorized by priority (e.g., high, medium, low) to help employees focus on critical tasks first.
3. Task Attributes
   - Title/Description: A clear title and detailed description of the task to guide employees.
   - Due Date: Deadline or due date for task completion.
   - Assigned Employee: The user (employee) responsible for the task.
   - Priority: The level of urgency (e.g., high, medium, low).
   - Attachments: Ability to attach documents, links, or files related to the task.
4. Task Status and Tracking
   - Pending/Active/Completed: Tasks can have a status indicating whether they are pending, actively being worked on, or completed.
   - Completion Percentage: Progress can be tracked by percentage or milestones.
   - Comments/Notes: Employees can leave comments for clarification, collaboration, or updates on tasks.
   - Time Tracking: Time spent on tasks may be recorded if required for reporting or billing purposes.
5. Reporting and Analytics
   - Task Reports: Generate reports that show the status of tasks across employees, teams, or departments.
   - Workload Distribution: Visual representation of task distribution across team members, identifying workload balance or overload.
   - Performance Metrics: Insights into individual or team performance, such as completion rates, overdue tasks, and time spent.
6. Integration with Other Tools

- Calendar Integration: Sync tasks with calendar tools (Google Calendar, Outlook, etc.) for better scheduling.
- Email Notifications: Automatic email updates when tasks are assigned, updated, or completed.
- Third-party Tools: Integration with project management tools (Trello, Asana, Jira, etc.), communication tools (Slack, Teams), or time tracking tools.

7. Security and Data Protection
    - Authentication & Authorization: Use of secure login systems, such as single sign-on (SSO), to ensure that users can only access the appropriate parts of the system.
    - Data Encryption: Ensure that data, especially sensitive employee or task data, is encrypted both in transit and at rest.
    - Backup and Recovery: Regular backups of the task data to prevent loss and ensure business continuity in case of system failure.
8. User Interface and Experience (UI/UX)
    - Dashboard: A user-friendly dashboard where employees can view their assigned tasks, deadlines, and priorities at a glance.
    - Task Overview: A clear, concise view of all tasks assigned to the user, with the ability to filter by status, date, or priority.
    - Mobile-Friendly: A responsive design for accessing the task management system from various devices, including mobile phones and tablets.
9. Scalability and Flexibility
    - Scalable System: The system should be designed to scale as the organization grows, with the ability to add more users, teams, and tasks without performance degradation.
    - Customization: The ability to customize task workflows, user roles, and notifications based on specific organizational needs.
10. Feedback and Continuous Improvement
    - User Feedback: Regular collection of feedback from employees and managers to improve the system's usability and functionality.
    - Bug Tracking: A mechanism to log, prioritize, and resolve issues or bugs within the system to maintain smooth operation.

These points provide a broad foundation and reference for developing a comprehensive employee task management system. You may need to adjust these elements based on the specific requirements of your organization and the project’s scope.

For an Employee Task Management System project, the Glossary section typically provides definitions and explanations for key terms used within the system. This ensures clarity and understanding for both developers and users. Below are some important points to include in the glossary of the project:

Glossary for Employee Task Management System

1. Task
    - Definition: A work item or activity that an employee is assigned to complete. It may have attributes like a title, description, priority, due date, and status.
    - Example: "Complete the monthly report."
2. Employee
    - Definition: A user of the system who is assigned tasks. Employees can update their tasks, mark them as completed, or report on their progress.
    - Example: "John Doe is an employee who has several tasks assigned to him."
3. Manager/Team Lead
    - Definition: A user who has the ability to assign tasks to employees, track task progress, and monitor team performance.
    - Example: "Sarah is the manager who assigns tasks to her team members."
4. Admin
    - Definition: A system user with the highest level of control, typically responsible for managing system settings, adding/removing users, and ensuring overall system functionality.
    - Example: "Tom is an admin responsible for managing user accounts and system configurations."
5. Task Assignment
    - Definition: The process of assigning a task to an employee. The task is linked to the employee, and they are responsible for completing it.
    - Example: "The manager assigns the task to Emily with a due date of March 5th."
6. Task Status
    - Definition: The current state of a task in the workflow. Common statuses include "Not Started," "In Progress," and "Completed."
    - Example: "The status of the task is 'In Progress.'"
7. Priority
    - Definition: A level of importance assigned to a task. Tasks can be categorized as low, medium, or high priority based on their urgency or importance.
    - Example: "The manager set the task priority to 'High' because it has a tight deadline."
8. Due Date
    - Definition: The date by which a task is expected to be completed.
    - Example: "The task has a due date of March 10th."
9. Deadline

- Definition: The final time by which a task must be finished. It is typically tied to the due date and may come with time constraints.
- Example: "The task deadline is at 5 PM on March 10th."

10. Progress Update
    - Definition: An employee's update on the status or completion level of a task. This could include updates like "10% Complete" or "90% Complete."
    - Example: "The employee updated the task to show it's 50% complete."
11. Notification
    - Definition: A message sent to users (via email, in-app, or SMS) to inform them about task assignments, changes, updates, or reminders.
    - Example: "The system sent a notification to the employee when the task was assigned."
12. Time Tracking
    - Definition: The process of recording the amount of time an employee spends working on a particular task.
    - Example: "The employee tracked 4 hours of work on the task."
13. Comment/Note
    - Definition: A message or remark left on a task by either the employee working on the task or the manager for clarification or collaboration.
    - Example: "The employee added a comment asking for clarification on the requirements."
14. Completion Percentage
    - Definition: A numerical or visual representation of how much of a task has been completed, often expressed as a percentage (e.g., 50% Complete).
    - Example: "The task's completion percentage is now 75%."
15. Task History
    - Definition: A record of all changes made to a task, including status updates, priority changes, assignments, and comments.
    - Example: "The task history shows that the priority was changed from medium to high."
16. Reporting
    - Definition: The process of generating and reviewing reports related to tasks, such as task completion rates, performance metrics, or workload distribution.
    - Example: "The manager generated a weekly task completion report."
17. Workload
    - Definition: The amount of work or number of tasks assigned to an employee or a team.
    - Example: "The team has a high workload with several pending tasks."
18. Dashboard

- Definition: A user interface that presents an overview of important data, such as assigned tasks, progress, and deadlines.
- Example: "The employee checks their dashboard every morning to see new tasks and deadlines."

19. Attachment
    - Definition: A file or document linked to a task, typically used to provide supporting information or resources necessary for task completion.
    - Example: "The manager attached the project brief to the task."
20. Time Zone
    - Definition: The geographical region in which the system records or tracks time, crucial for managing tasks across different locations.
    - Example: "The task deadline was set according to the employee's local time zone."
21. Escalation
    - Definition: The process of notifying a higher-level manager or admin when a task is overdue or needs immediate attention.
    - Example: "The task was escalated to the manager after the employee missed the deadline."
22. Recurring Task
    - Definition: A task that is repeated regularly, such as daily, weekly, or monthly, with a set schedule for repetition.
    - Example: "The employee has a recurring task to submit a weekly status report every Friday."
23. Dependency
    - Definition: A situation where one task cannot start or be completed until another task is finished.
    - Example: "The report task has a dependency on the data collection task being completed first."
24. Task List
    - Definition: A collection or list of tasks assigned to an individual or a team, typically organized by due date, priority, or project.
    - Example: "The employee checked the task list to prioritize the day's activities."
25. Project
    - Definition: A collection of related tasks grouped together to achieve a specific objective. Projects help organize work and manage tasks at a higher level.
    - Example: "The project involves five tasks, including research, design, and final review."
26. Milestone

- Definition: A significant point or event in a project or task that indicates progress or completion of a phase.
- Example: "The design phase of the project was marked as a milestone."

27. Alert
    - Definition: A notification triggered by specific conditions such as missed deadlines or low task progress.
    - Example: "The system sent an alert when the task exceeded its due date."
28. User Role
    - Definition: A classification of users based on their responsibilities and access levels in the system (e.g., Employee, Manager, Admin).
    - Example: "The user role of 'Manager' allows task assignment and progress monitoring."
29. Kanban Board
    - Definition: A visual board that displays tasks and their progress, typically divided into columns like "To Do," "In Progress," and "Completed."
    - Example: "The employee views the task status using a Kanban board."
30. Burndown Chart
    - Definition: A graphical representation of the completion of tasks over time, often used in agile project management.
    - Example: "The manager uses a burndown chart to track progress on the current sprint."

These glossary terms help define the key concepts in the employee task management system, ensuring that everyone involved understands the terminology used throughout the project.

4.1.1. IMPLEMENTATION APPROACHES

- Agile Methodology: Development will be iterative, with regular updates, feedback, and testing to ensure the system meets the requirements.
- Version Control: GitHub or GitLab for code versioning and collaboration.
- Deployment: The system will be deployed on a cloud server (AWS, Heroku) for easy access and scalability.

4.1.2. TESTING APPROACH

4.1.2.1. UNIT TESTING

Unit testing will be conducted to ensure individual components (task creation, user login, notifications) work as expected.

4.1.2.2. INTEGRATED TESTING

Integrated testing will ensure that all modules work together seamlessly, verifying the flow of data between the front-end and back-end.

---

5. RESULT AND DISCUSSION

The Employee Task Management System is expected to improve task allocation, reduce missed deadlines, and increase productivity. By using a digital solution, organizations will save time, reduce human error, and improve communication across teams.

---

6. CONCLUSION

6.1.1. LIMITATIONS OF THE PROJECT

- Scalability: In the initial phase, the system may not handle extremely large organizations with thousands of users efficiently.
- Internet Dependency: If deployed as a web application, an internet connection is required for full functionality.
- Complexity of Features: Some advanced task management features may be outside the scope of this version.

6.1.2. FUTURE SCOPE OF THE PROJECT

- Mobile App Integration: Developing a mobile application to allow employees to track tasks on the go.
- AI Integration: AI can be used to predict task deadlines and manage workload distribution.
- Multi-language Support: Expanding to support different languages for global organizations.
- Advanced Analytics: Introducing advanced reporting features, such as task prediction and performance forecasting.

---

7. REFERENCES

(List of books, research papers, online resources, and other references used in the project.)

---

8. GLOSSARY

- Task Management: The process of organizing, assigning, and tracking tasks.
- Scalability: The ability of a system to handle growth in terms of users or data.
- Agile Methodology: A development methodology that promotes iterative progress through small, manageable increments.

This structure outlines the Employee Task Management System's core functionality and design.

Here's a concise version of the conclusion for the Employee Task Management System project:

Conclusion Summary

1. System Effectiveness: The system efficiently manages task assignments, tracking, and completion, boosting productivity and reducing manual effort.
2. Improved Collaboration: Enhanced communication and task visibility between employees and managers promote transparency and collaboration.
3. Time Management & Accountability: Task deadlines, progress tracking, and time recording help employees stay on track, ensuring accountability and timely completion.
4. Reporting & Analytics: Provides valuable insights through reporting, helping managers balance workloads and make informed decisions.
5. Scalability & Flexibility: The system is adaptable, able to scale with the organization's growth and customizable to different needs.
6. User Experience: Intuitive UI/UX and mobile accessibility improve usability and foster efficiency, even on the go.
7. Security: Secure authentication, encrypted data storage, and regular backups ensure data integrity and protection.
8. Future Enhancements: Future improvements could include AI for task prioritization, deeper analytics, and more integrations with third-party tools.
9. Lessons Learned: Emphasized user-centered design and continuous improvement to meet evolving task management needs.
10. Final Thoughts: The system is a valuable tool for improving task management, efficiency, and collaboration, with potential for future growth and optimization.

This summary captures the key outcomes and future potential of the Employee Task Management System.