# 

**CSE300: Software Engineering**

**Monsoon 2022**

**Group 7**

**Group Members**

| **Name** | **Enrolment Number** | **Email-id** |
| --- | --- | --- |
| Sakshi Shah | AU1940213 | sakshi.s5@ahduni.edu.in |
| Astha Patel | AU1940312 | astha.p@ahduni.edu.in |
| Kareena Matwani | AU1940314 | kareena.m@ahduni.edu.in |
| Vraj Parikh | AU1940185 | vraj.p1@ahduni.edu.in |
| Rushil Borad | AU1940179 | rushil.b@ahduni.edu.in |

# **Design Document**

[**Design Document**](#_o362xgqfivv9) **1**

[**1. Introduction**](#_prxp0weaobyi) **3**

[1.1 Purpose of this document](#_9b6uq21w7y4o) 3

[1.2 Document Overview](#_fka50ulxr7vd) 3

[1.3 Identification](#_3jx0xekqq0ox) 3

[1.4 Scope](#_qunxbm4n7d8f) 3

[1.5 References](#_9pi9nu29vjdm) 4

[1.6 Tools and Techniques](#_k6o3vo5kva8i) 4

[1.7 Key Stakeholders](#_hq5bn52e6eiu) 4

[**2. Design Overview**](#_nr5mtsdc7vvl) **5**

[2.1 Background Information.](#_9j7z811hjgrv) 5

[2.2 System Evolution Description](#_8w3dftto0fns) 6

[2.3 Constraints](#_vv9sksqxl60x) 6

[2.4 Risk Analysis](#_fptrb3k15d0w) 6

[2.5. Issues](#_g89i4uocl1at) 8

[2.6. Assumptions](#_ogb51yscbqf4) 8

[2.7 Design Considerations](#_qa9uywb2g7dm) 8

[2.7.1 Goal and Guidelines](#_b3i2oos15s99) 8

[2.7.2 Operational Environment](#_rovde4pc5a6a) 9

[3.1 System Functions](#_trcuv19b3z7r) 9

[3.2 Similar System](#_393ellsbzvde) 10

[3.3 User Problem Statement](#_6h79vj79l3a) 10

[3.4 User Objectives](#_jumbrzvu4yt3) 11

[4. Database Design](#_q8qc8qbwajx1) **11**

[4.1 Conceptual ER Diagram](#_vzhmgmib0qcf) 11

[4.2 Logical ER Diagram](#_hfjg5c1wpww5) 11

[4.3 Physical ER Diagram](#_7pwpghbqejla) 11

[4.4 Modules and Interaction.](#_1az9xceef5ex) 11

[4.5 Data Dictionary](#_n5d90snza48w) 11

[**5. Software Architecture and Architecture Design**](#_tio4pg3g3pj7) **14**

[5.1 Architectural Design](#_3lz70v6mlrj9) 14

[5.1.1. Topology Diagram](#_f983fc3ajpg5) 14

[5.1.2. Website Architecture Diagram](#_8bbc43y20t4) 15

[Link](#_5vvvb18kojf9) 15

[5.2. Software Architecture](#_gupisk6r0pwr) 15

[5.2.1. Software Architecture Diagram](#_d26xt97zf9mv) 15

[5.2.2. Software Element](#_572t2bnkmtsj) 16

[**6. User Interface Design**](#_5tkuft0d7u) **17**

[6.1. User Interface](#_vlkjbdha9ru7) 17

[6.1.1. Wireframes](#_a3cckbb53qqg) 17

[6.1.2. Screen Images](#_8tmiq4g7uzad) 36

[6.1.2. Objects and Actions](#_tvi46f8gbrvp) 58

[6.1.3. Interface Design Rules](#_g7yi8738wk7d) 59

[6.2. Components Available](#_ajmb3zxuwpdl) 60

# 

# **Introduction**

## **1.1 Purpose of this document**

The purpose of this Software Design Document is to provide a description of the design of a complete system to proceed with an understanding of what is to be built and how it is expected to be built. This document also traces back to cover all the listed requirements and analysis.

## **1.2 Document Overview**

Chapter 1: Introduction of the design document

Chapter 2: Design Overview

Chapter 3: Scope of Work

Chapter 4: System Design

Chapter 5: Detailed Design

Chapter 6: Interface Design

Chapter 7: User Interface Design

## **1.3 Identification**

The software will be used on a web browser and for the database design MySQL will be used which will help in generating the required queries.

1.3.1 Title: Task Management System

1.3.2 Version: 1.0.0

1.3.3 Release: 1

## **1.4 Scope**

This document aims to provide detailed design details for developing a task management system keeping in mind the requirements. This document will help developers and maintenance professionals to understand the software. The document includes architectural design and the overall overview of the system from different perspectives. It is intended to capture and convey the significant architectural decisions which have been made on the system. The intended audience of the document is system developers and system designers.

## **1.5 References**

* *Software Engineering: A Practitioner’s Approach, 7/e (McGraw-Hill 2009)*
* *Development Standards, and Guidelines2 General Overview and Design Approach*

## **1.6 Tools and Techniques**

**Tools**

* Miro Online whiteboard
* Whimsical
* VisualParadigm
* Drawio
* Figma
* Xampp

**Languages**

* HTML
* CSS
* PHP
* JSX
* MYSQL

## **1.7 Key Stakeholders**

The key stakeholders of this software are:

1. Employees (primary users)

Employees can use the software to manage their tasks and project effectively in a team. The features like managing team tasks, resources, and deadlines are included on the same platform.

1. Team Leader and Team Managers

The Managers can manage their projects on the platform by sharing relevant information, setting up tasks, tracking progress, and more using the same platform.

1. IT Administrator

The IT administrator can add new users to the software and can also add new projects and tasks and assign them to individual users. The administrator can manage user information, as well as project information on the platform.

# **2. Design Overview**

## **2.1 Background Information.**

There are various platforms like google space, asana, Trello, etc that are used by various students as well as companies today to manage tasks to be done. The Workspace web application will help an organization as a whole from authenticating the information to maintaining progress records of the project can be achieved using this application. This application should be accessible and organization-specific to maintain data integrity. An admin can easily manage projects and tasks.

The statement of needs explains why the system is being developed, what purpose it serves, and why it is necessary. The reason for this web application is to provide an alternative to endless emails and calls among team members and effectively manage tasks. The platform would be a communication link among team members to avoid hustle and confusion.

The solution thus should be easy to access and build a strong communication link to manage the progress of the tasks for each user. Thus, the solution to the requirement can be a web application that connects the admin and users. Admin will manage project information and assign tasks, while the user can view the assigned tasks with a deadline, view needed resources, and can also maintain progress.

## **2.2 System Evolution Description**

This project is on a small scale for demonstration purposes. The system can be further scaled up to a deployment server and the database can be moved to different servers.

In terms of functionalities, a meeting option can be added to the software that would enable users to schedule and have a meeting with the team members using this software itself.

## **2.3 Constraints**

Functionalities

Google and other account-based authentication are not provided which is one of the simplest methods to login in provided by a majority of users. Also, there is no sharing option available that can easily share the project progress and the relevant details.

## **2.4 Risk Analysis**

Inappropriate Project Manager - It results in problems at the very beginning, mainly in the implementation and enforcement of the work. That leads to incorrect change management, risk management, and scope management.

2.4.1 Requirements Analysis

○ The requirements are unclear if they are not understandable by analysts and developers.

○ The functional, as well as non-functional requirements, must be defined precisely involving every relevant stakeholder.

○ The requirements are incomplete if they are missing some of the user needs, constraints and other requirements.

○ The requirement is inconsistent if it contradicts any other requirement in the project.

○ Usually analysts and developers focus on what the system should do and ignore how the system should be (i.e. usability, maintainability, scalability, testability, etc.). Non-functional requirements are essential to project success as much as functional requirements.

○ If the requirements are clear, verifiable, accurate, consistent, complete, and feasible then they are realistic to be put in the requirements document and then implemented.

2.4.2 Design Activity

○ At the time of design verification, it might be discovered that many alternatives to the same design may exist, and choosing the design should depend on the system and its nature.

○ After the selection of a design, it must meet every defined requirement and not contradict any of the requirements.

○ If the system was not decomposed correctly with all the components defined well, the developers may face difficulties in assigning functions and defining objectives for each of the components.

○ Misinterpreted components and their function threatened the allocation activity ad the functionalities are derived from the functional requirements in the requirements document.

○ Sometimes, passing data is not used properly, it is just transferred to another component where it is required. If it is not managed carefully, it can reduce readability and lead to confusion.

2.4.3 Construction Activity

* + A wrong choice in architectural design, would lead to problems in system implementation and integration at a later stage.
  + The choices of architectural design method may affect the choice of programming language and database to be used for the system. A wrong choice of architectural design can lead to choosing a programming language by a developer that is not compatible with the system.
  + If the system is too complex, and the developers do not have enough skills and experience to manage this complexity, then it would result in the incomprehensible design of the system with different difficulties.
  + An inaccurate estimation of available reusable components in the analysis phase can impact the schedule and budget of the project.

2.4.4 Documentation Activity

* + While documenting the requirements, maintainability, and feasibility is to be considered to avoid difficulty to modify the data without rewriting it, in the requirement document.
  + Developers must be involved in the requirements analysis, design, and documentation phase, so that they can easily understand the requirement document at the time of execution. This could lead to a design of the intended software system.
  + The design document must be detailed enough to allow the programmers to work independently. If the components in the design document are not clearly defined; their inputs, outputs, functions, and relationships if not stated properly can lead to confusion.
  + The design document should be written in a common, understandable natural language. It should not be unclear or unreadable by developers.
  + 2.4.5 Coding Activity

2.4.6 Testing Activity

## **2.5. Issues**

| Ref. | Issue |
| --- | --- |
|  | Adding a new user to the project once it is already defined. |
|  | Sharing the Report of the project with other users if required. |

## **2.6. Assumptions**

| Ref. | Assumption |
| --- | --- |
|  | Resources (of different formats) related to the project can be stored and managed as per requirement. |
|  | Scheduling and having an online session can be done with other companion software. |
|  | The admin would have verified the details about assigning the task and who to include in the project before creating or updating details on the portal. |

## **2.7 Design Considerations**

### **2.7.1 Goal and Guidelines**

Architecture: The proposed solution to the problem must satisfy all the functional needs stated by the user and is required to be developed considering all the non-functional requirements. It is supposed to be adaptive to further changes like support for additional features, functionalities, and use cases.

Development Environment: The application development is supposed to stay consistent. Adapting the technological advances this project aims to build the application in the latest versions of technology with backward compatibility for better performance.

Ease of Use: The features of the application must be user-friendly for a strong user experience. Since the application is specific to handling employees' tasks it is supposed to have easy and fast access.

Extensibility: The application is aimed to be extensible to add new features, also providing backward compatibility.

API Enabled: Regional coordination support is a key driver of the project. The application must be API-centric and support an open and published API architecture.

RESTful Framework: The application and underlying architecture must be a REST framework

### **2.7.2 Operational Environment**

The operating environment of the web portal can be custom based on requirements, may it be a university or an organization the website can be used to help the employees maintain their task lists and work information.

The goal of the proposed system includes developing a web application with advance features, the best performance,

Scope of Work

## **3.1 System Functions**

* Admin Side
  + Add, Edit and Delete Project
  + Add, Edit and Delete Task
  + Search for project
  + Update task status
  + Add, update productivity
  + Add, edit , delete events on calendar
  + Download reports
  + Add, edit, delete, search user
* Employee Side
  + View tasks defined
  + Search for project and their deadlines
  + See the project status
  + Search and view the list of the projects
  + View events in the calendar

## **3.2 Similar System**

* + Asana : [Link](https://www.google.com/aclk?sa=l&ai=DChcSEwi_m6Dhoe77AhXFfCsKHWZCDicYABAAGgJzZg&sig=AOD64_2YVUtx7XaZbYkxafcRujGRbAkPsw&q&adurl&ved=2ahUKEwi8mJrhoe77AhVNXWwGHeA1DDMQ0Qx6BAgKEAM)
  + Any.Do : [Link](https://www.dragapp.com/blog/free-task-management-software/#anydo)
  + Trello : [Link](https://www.dragapp.com/blog/free-task-management-software/#trello)
  + Wrike : [Link](https://www.dragapp.com/blog/free-task-management-software/#wrike)
  + Freedcamp : [Link](https://www.dragapp.com/blog/free-task-management-software/#freedcamp)

## 

## **3.3 User Problem Statement**

* + Users have difficulty in managing all these apps or websites with different tasks defined so in our project named Workspace provides all the features at one place.
  + Project managers and Employees can work together and collaborate.
  + Events can be added and deleted which will be shared to both project manager and Employee.

## **3.4 User Objectives**

* + Users can search for the event for each day , see project deadlines each day all at one place
  + Managers can check the progress of the multiple users and also create events and assign to the users and add users according to the convenience inside the project team.

# **4. Database Design**

## 

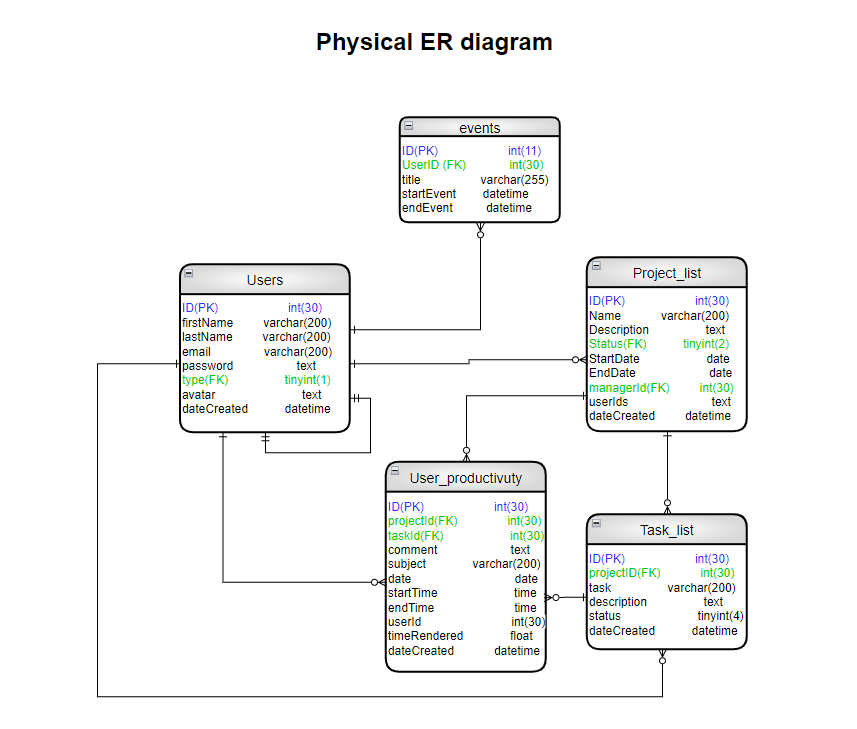
## **4.1 Conceptual ER Diagram**

## 

## **4.2 Logical ER Diagram**

## 

## **4.3 Physical ER Diagram**



## 4.4 Modules and Interaction.

## 4.5 Data Dictionary

4.5.1 Table 1: events

| Element | Description | Data type | Required? | Constraints |
| --- | --- | --- | --- | --- |
| Id | Unique id of each event | int (11) | y | Cannot be null |
| title | Title id for keeping title of the event | varchar(255) | y | Cannot be null |
| start\_event | Start time of the event | datetime | y | Cannot be null |
| end\_event | End time of the event | datetime | y | Cannot be null |

4.5.2 Table 2: project\_list

| Element | Description | Data type | Required? | Constraints |
| --- | --- | --- | --- | --- |
| Id | Unique id of the project | int(30) | y | Cannot be null |
| name | Project name given by admin | varchar(300) | y | Cannot be null |
| description | Details regarding the project | text | y | Can be null |
| status | Status about the project | tinyint(2) | y | Cannot be null |
| start\_date | Start date of the project | date | y | Cannot be null |
| end\_date | End date of the project | date | y | Cannot be null |
| manager\_id | Assigned by (manager name) | int(30) | y | Cannot be null |
| user\_ids | Users in that project | text | y | Cannot be null |
| date\_created | Project creation date | datetime | y | Cannot be null |

4.5.4 Table 3: User

| Element | Description | Data type | Required? | Constraints |
| --- | --- | --- | --- | --- |
| Id | User Id to uniquely identify user | int (30) | y | Cannot be null |
| firstname | User name for display | varchar(200) | y | Cannot be null |
| lastname | User last name | varchar(200) | y | Cannot be null |
| email | User email for login | varchar(200) | y | Cannot be null |
| password | User credentials given by admin to login | text | y | Cannot be null |
| type | The user is admin - employee or project manager | tinyint(1) | y | Cannot be null |
| avatar | Image to text | text | y | Cannot be null |
| date\_created | Date of creation of the admin | datetime | y | Cannot be null |

4.5.5 Table 5: users\_productivity

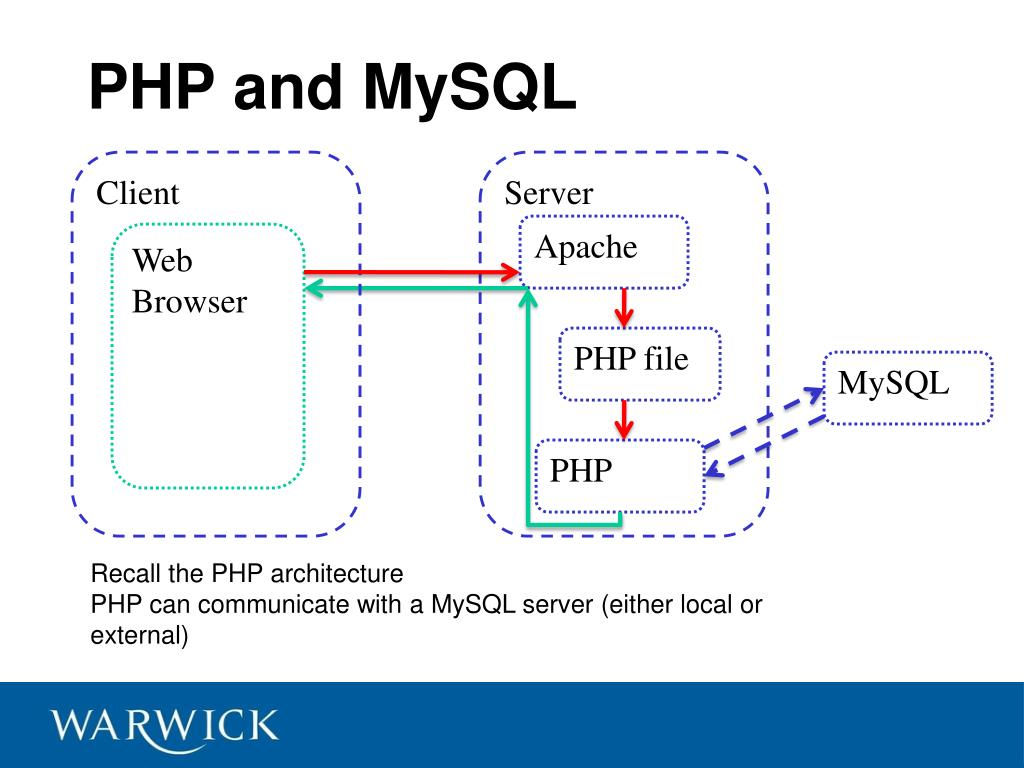
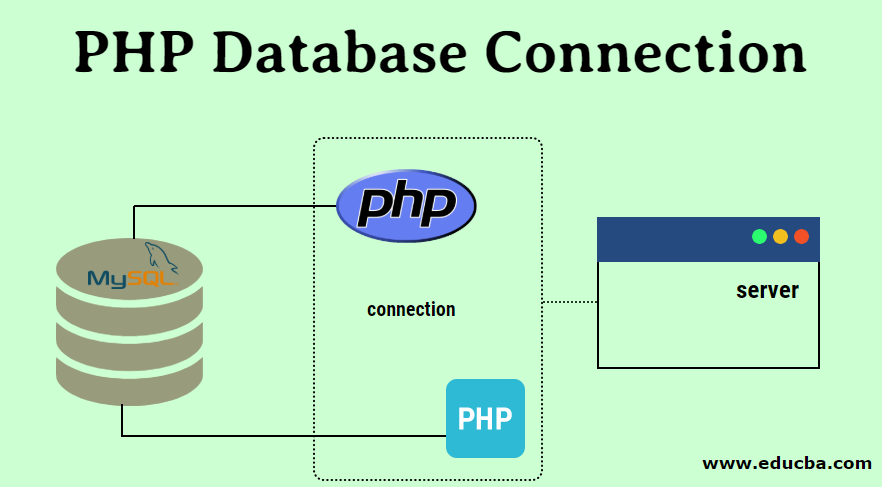
| Element | Description | Data type | Required? | Constraints |
| --- | --- | --- | --- | --- |
| id | Id of the user | int(30) | y | Cannot be null |
| project\_id | Unique of the project id | int(30) | y | Cannot be null |
| task\_id | Unique id of the task | int(30) | y | Cannot be null |
| comment | Comment that the admin gives to the user | text | y | Can be null |
| subject | Project name or progress | varchar(200) | y | Cannot be null |
| start\_time | Start time of task | time | y | Cannot be null |
| end\_time | End time of the task | time | y | Cannot be null |
| user\_id | Unique id of the user | int(30) | y | Cannot be null |
| time\_rendered | Time of the comment | float | y | Cannot be null |
| date\_created | Comment of the data created | datetime | y | Cannot be null |

# **5. Software Architecture and Architecture Design**

## **5.1 Architectural Design**

## **5.1.1. Topology Diagram**

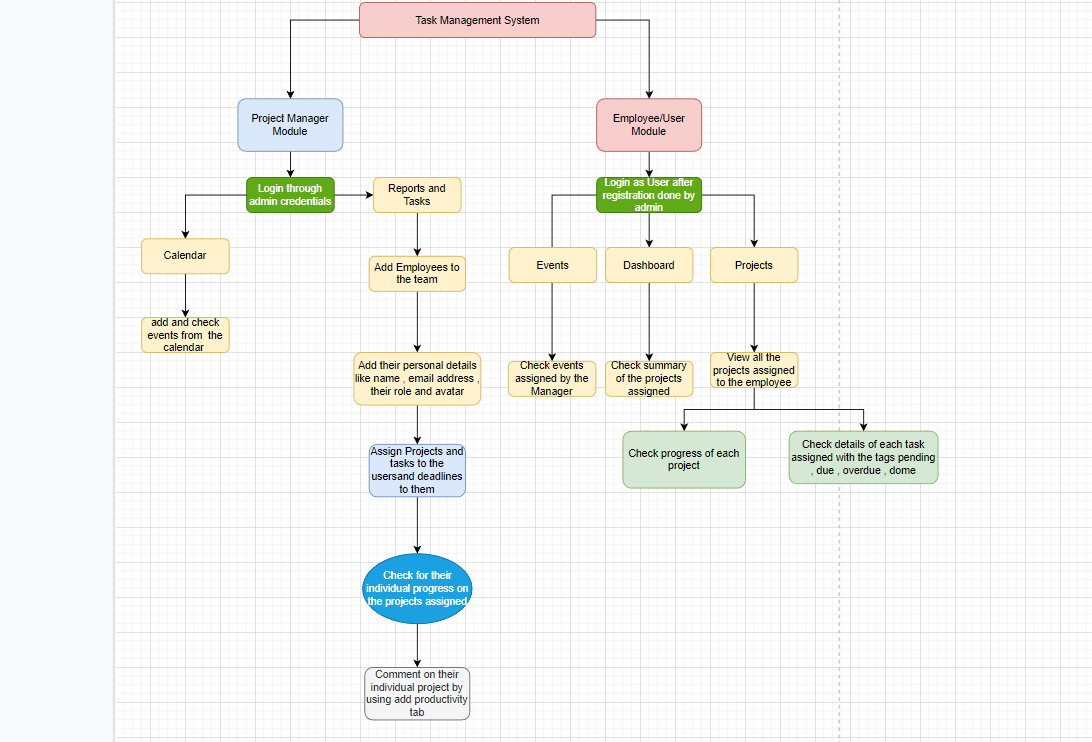
* Topological design represents the communication between the nodes , devices involved , the connection between all the devices are shown here, like a network topology.



[Source](https://www.educba.com/php-database-connection/) [Source](https://www.slideserve.com/alvis/php-mysql)

## 

## **5.1.2. Website Architecture Diagram**



[Link](https://drive.google.com/file/d/1j-aNUKB96XfLdkD3aD5sNsxjB0CTb0AT/view?usp=sharing)

## **5.2. Software Architecture**

## **5.2.1. Software Architecture Diagram**

* The software architecture gives us an idea and creates a visual representation of the physical elements attached to the software system. In between there are general limitations and boundaries of the system , also that software architecture helps in describing patterns used throughout the designing of the software.



## 

## **5.2.2. Software Element**

|  | System | Note |
| --- | --- | --- |
| Frontend/Backend Framework | CSS , JavaScript , PHP | PHP , a scripting language, helps in creating dynamic and interactive based pages for the website. Javascript and CSS were used for making the frontend and user interaction more aesthetic. |
| Design Frames | Figma | The basic prototypes of how the website would look were made using figma which gives the better visualization of how the application will look. |
| Database | MySQL | MySQL helps in maintaining databases very efficiently , also it works well in combination with PHP. |

# **6. User Interface Design**

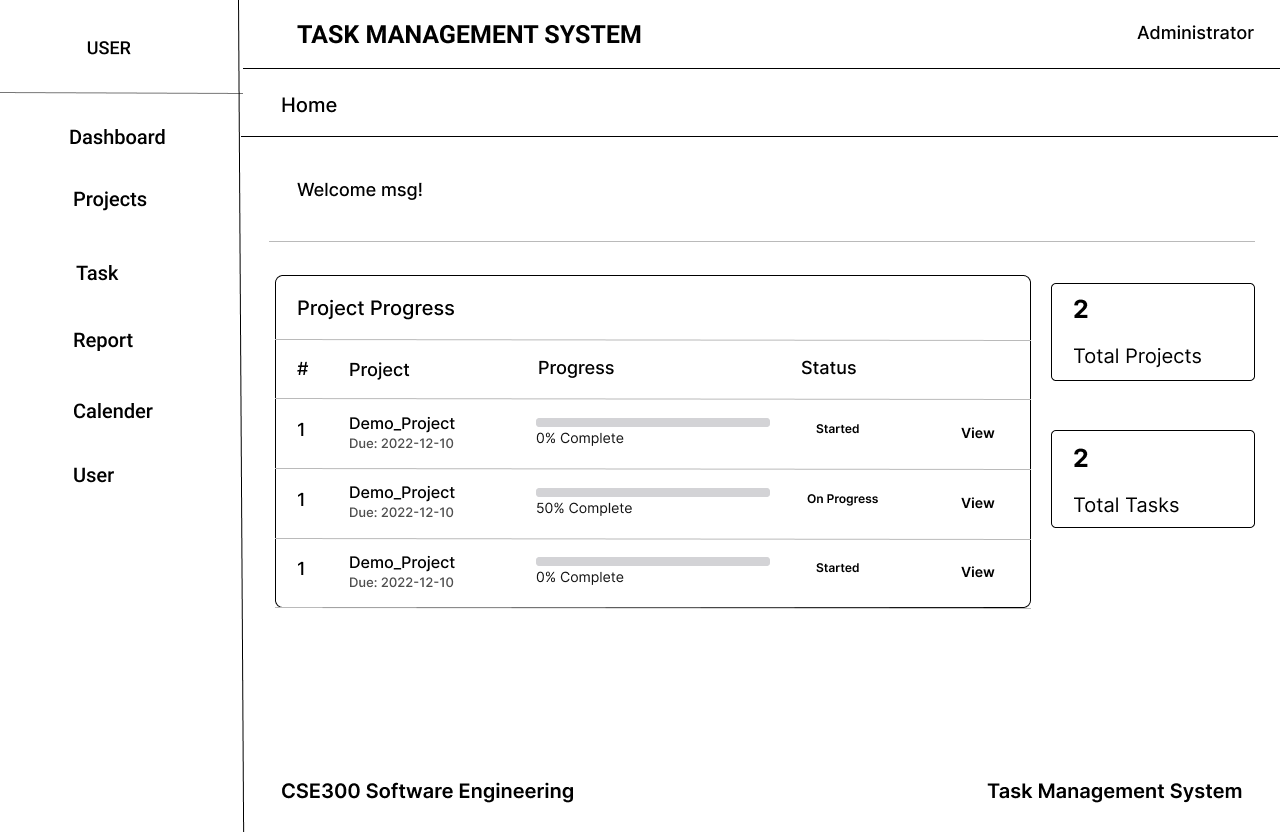
## **6.1. User Interface**

### **6.1.1. Wireframes**

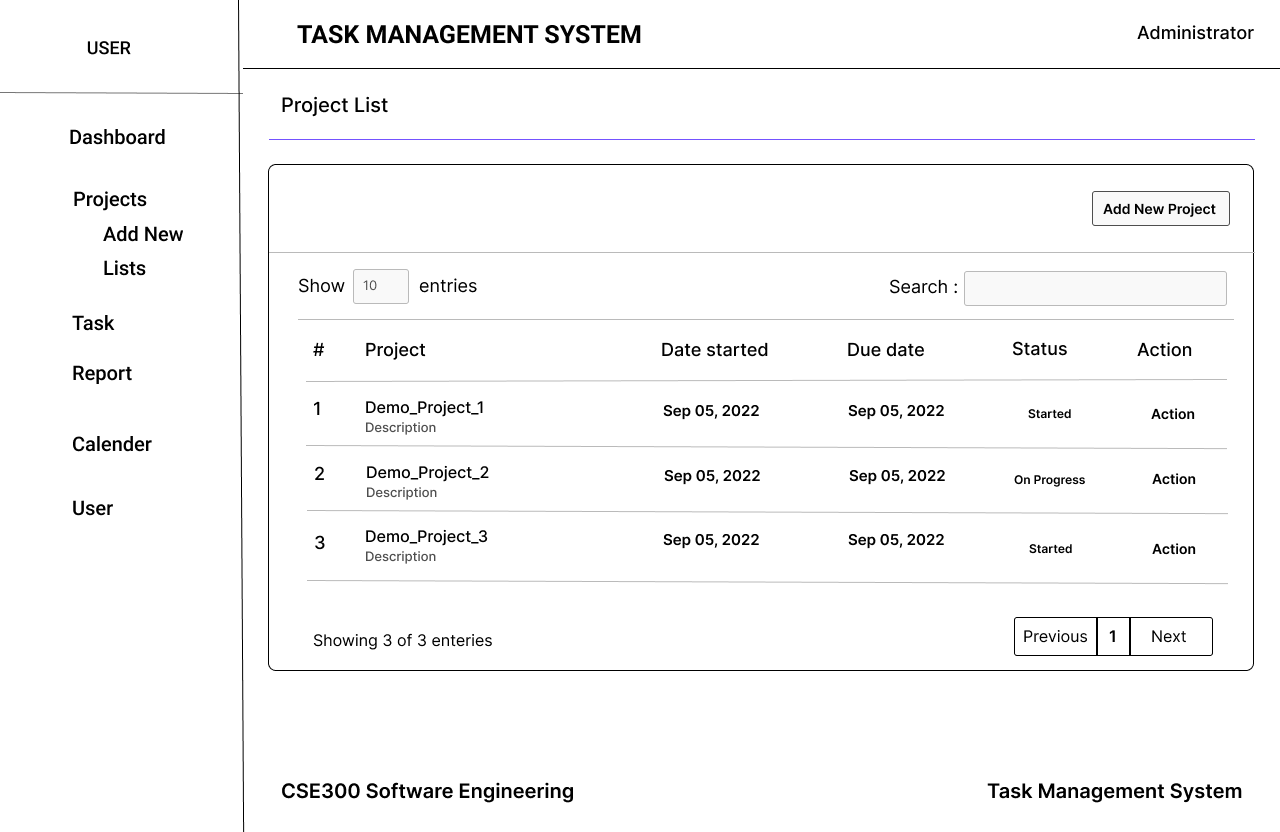
**Login/ Sign up**



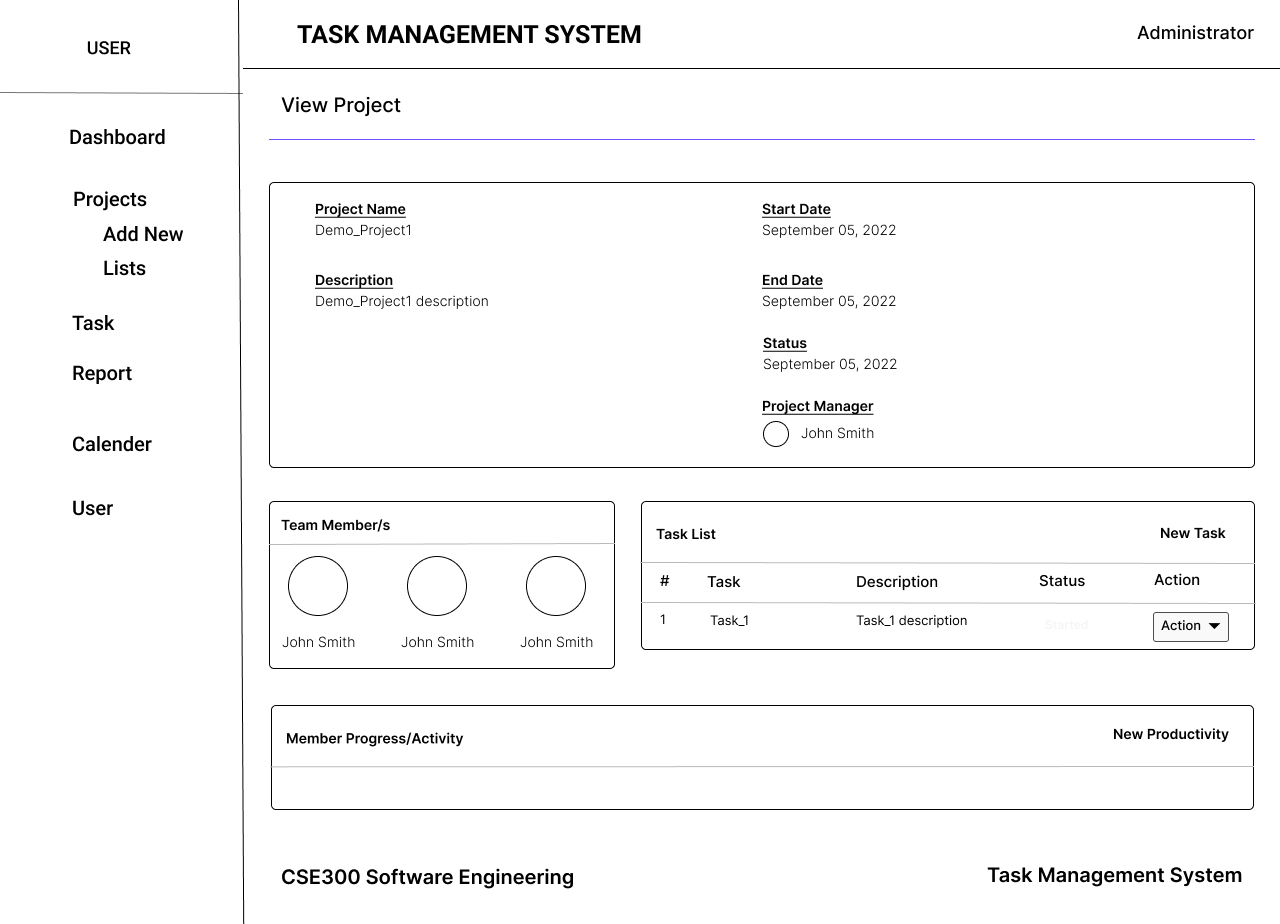
**Dashboard**



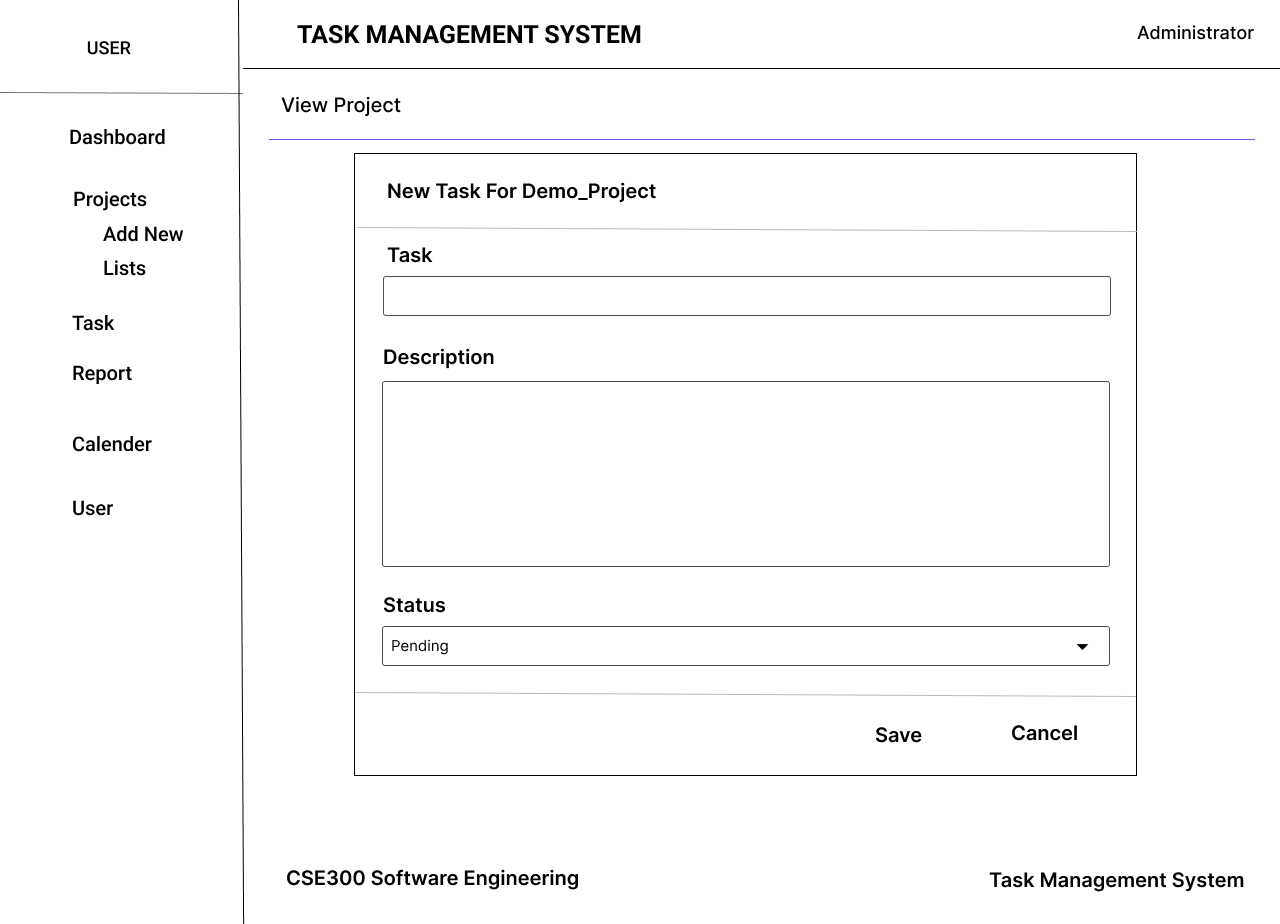
**Project List**



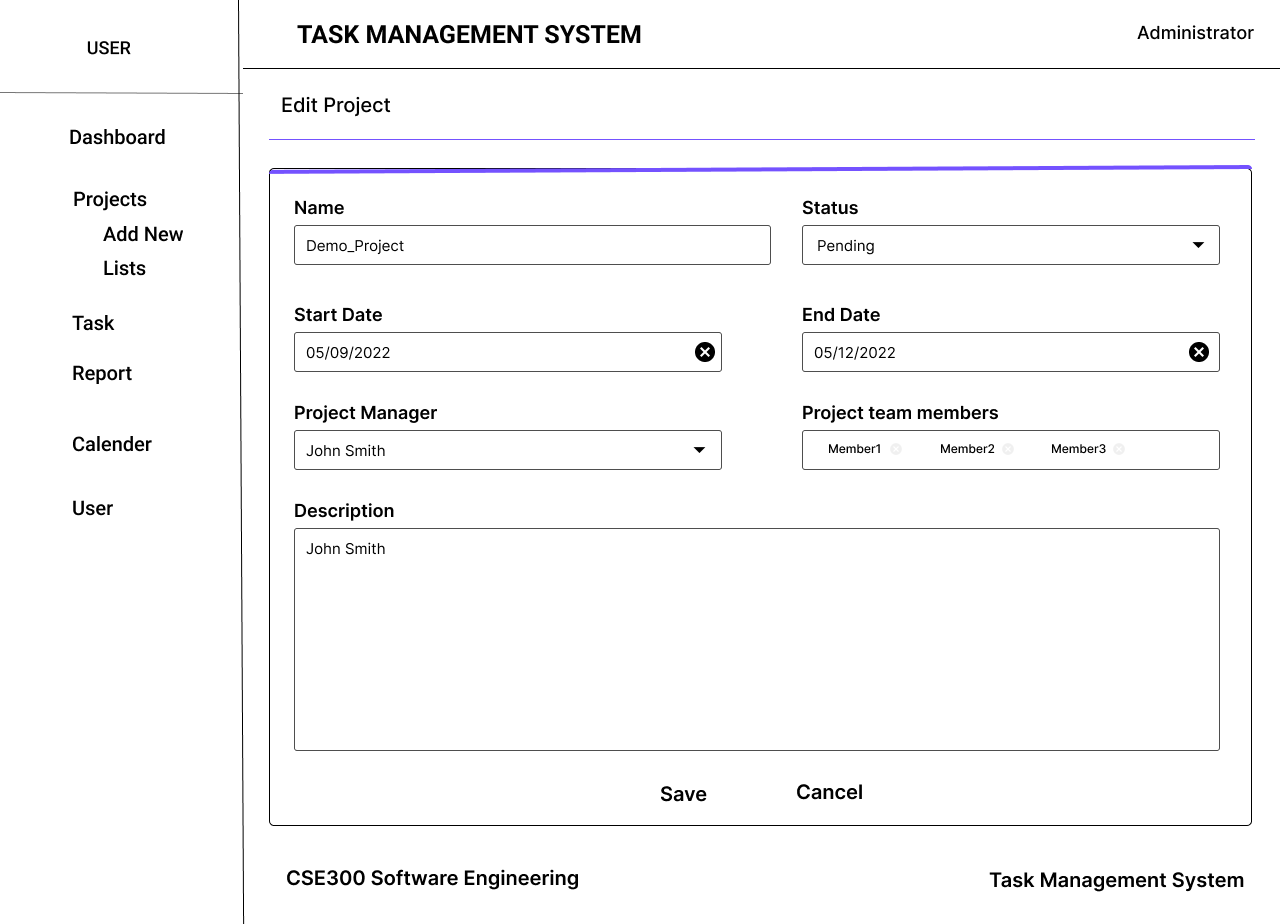
**View Project**



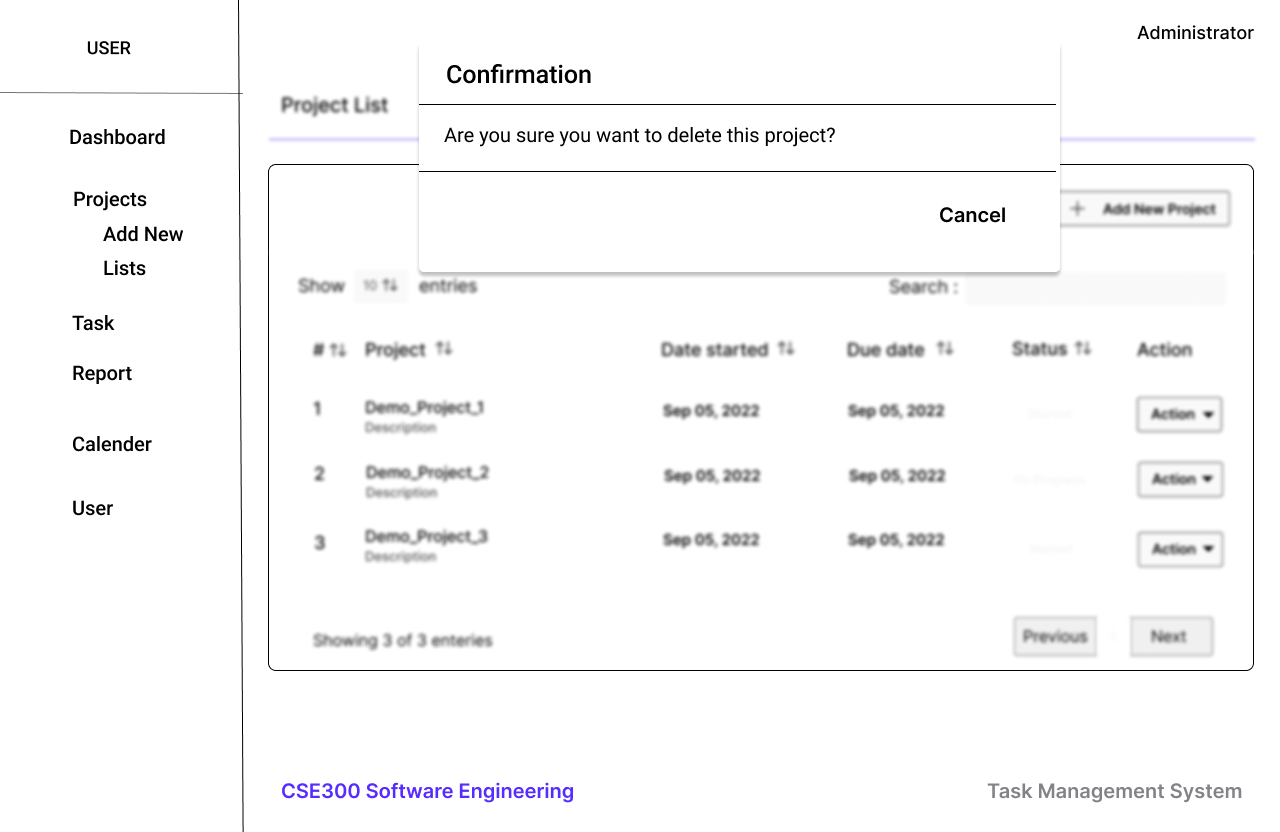
**Add new task**



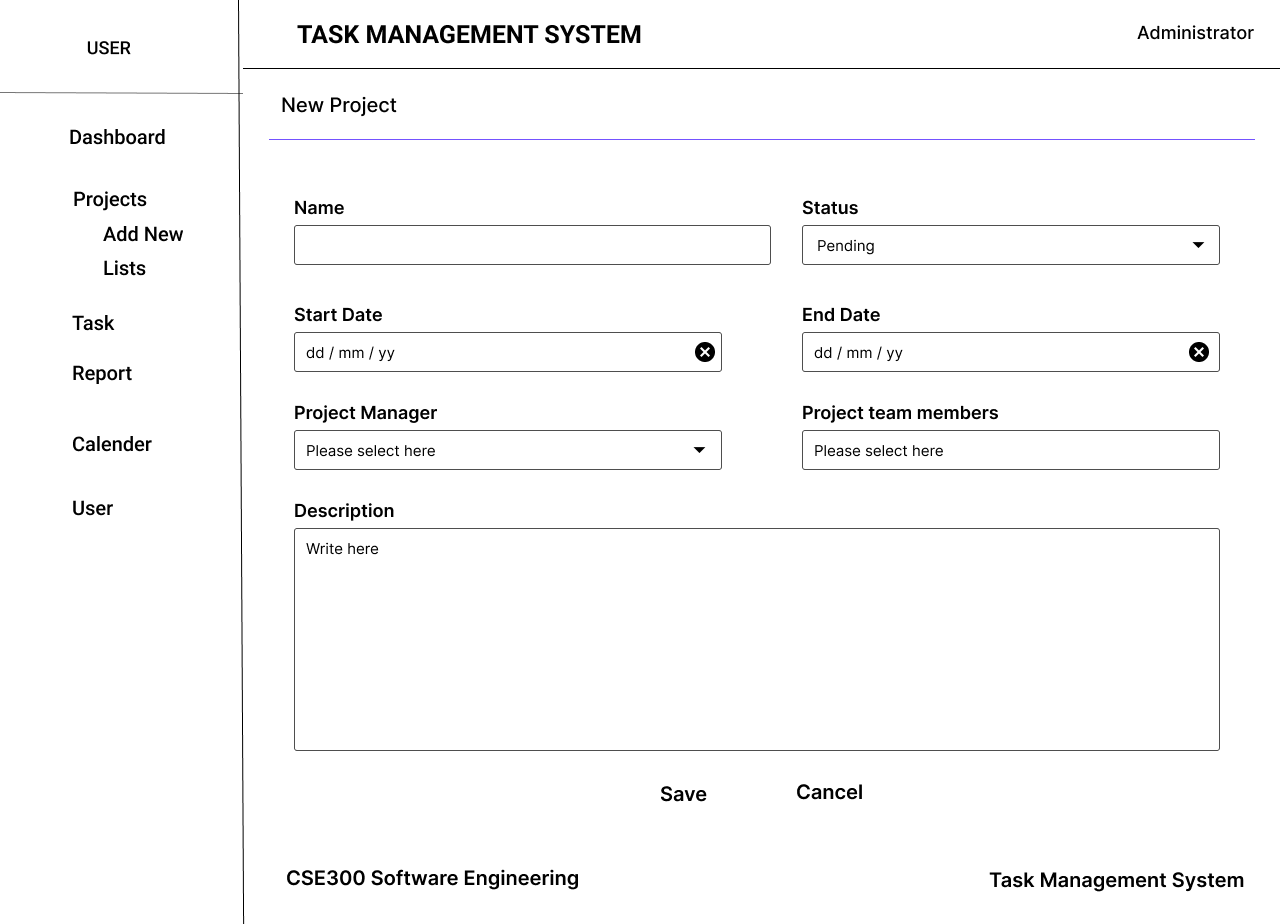
**Edit Project**



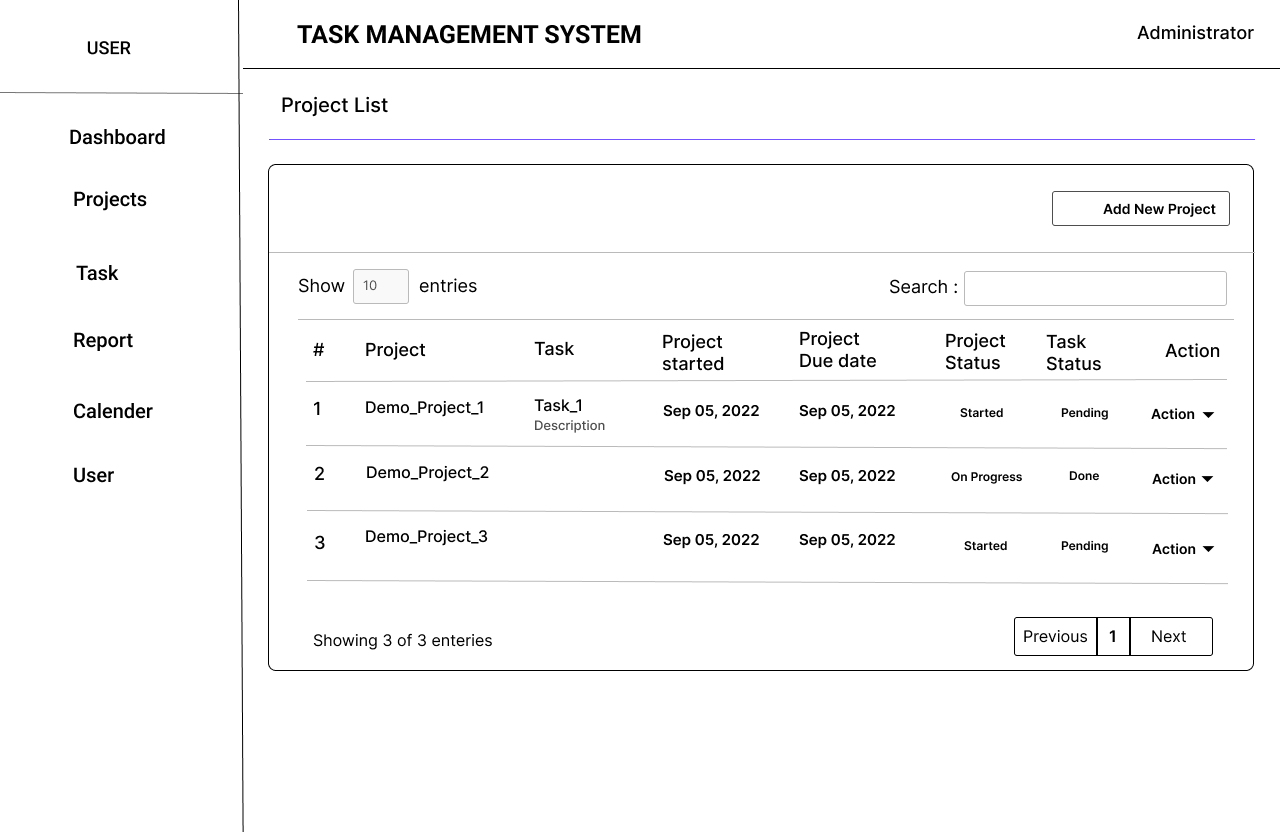
**Delete Project**



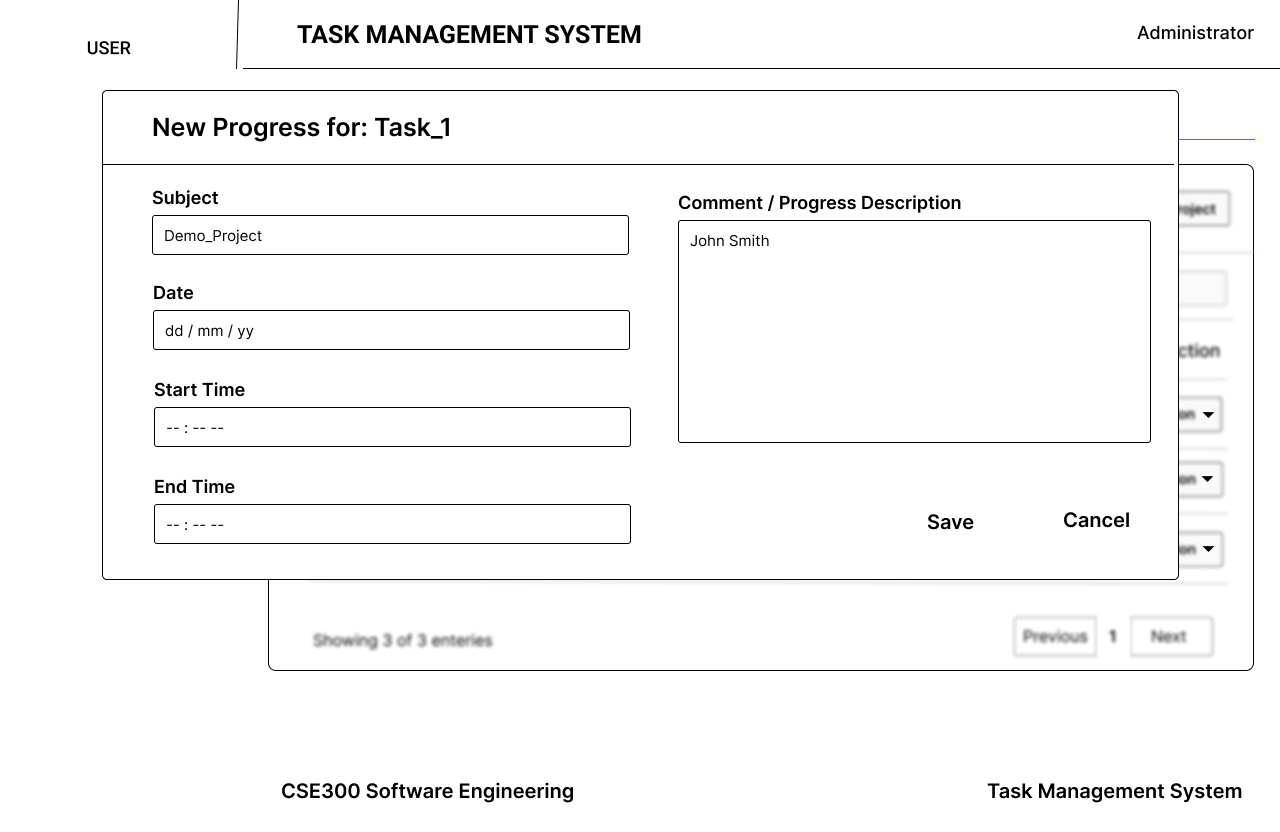
**Add New Project**



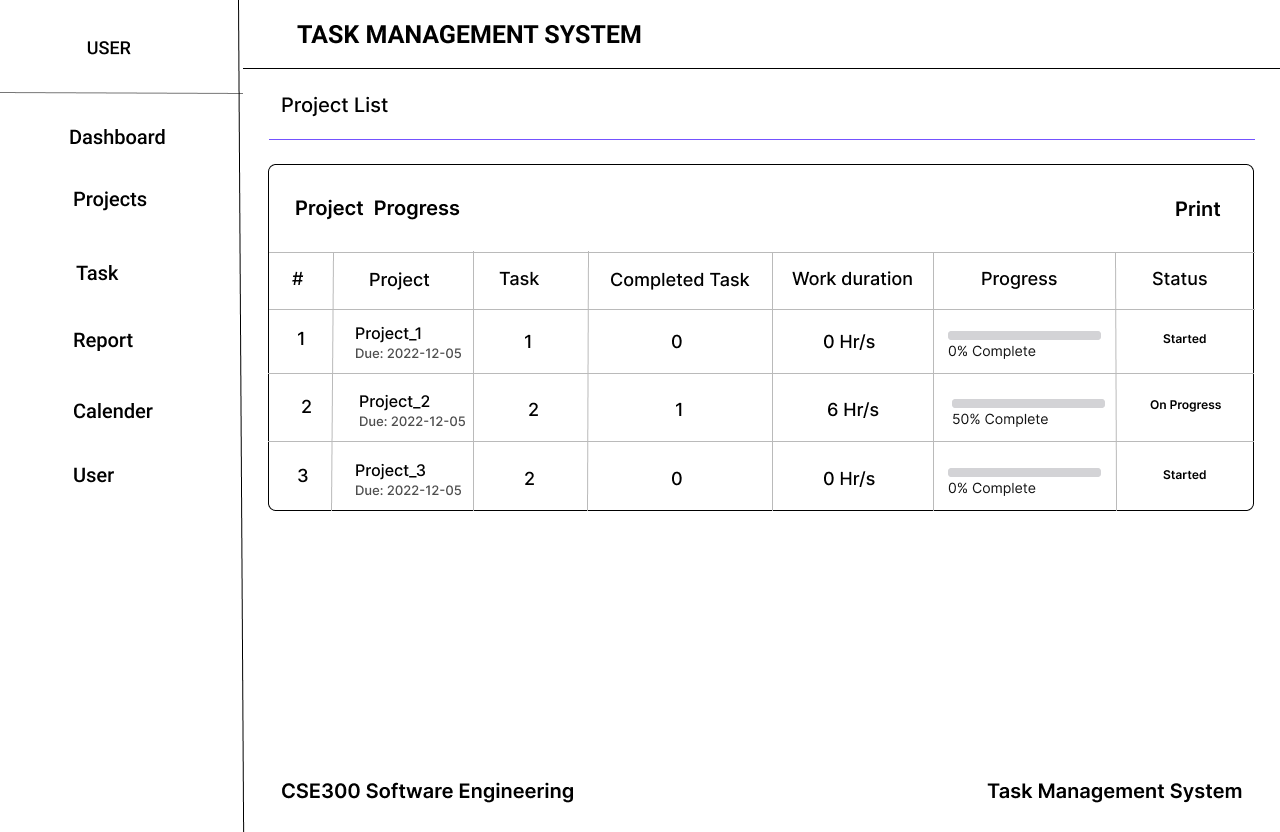
**Task Lists**



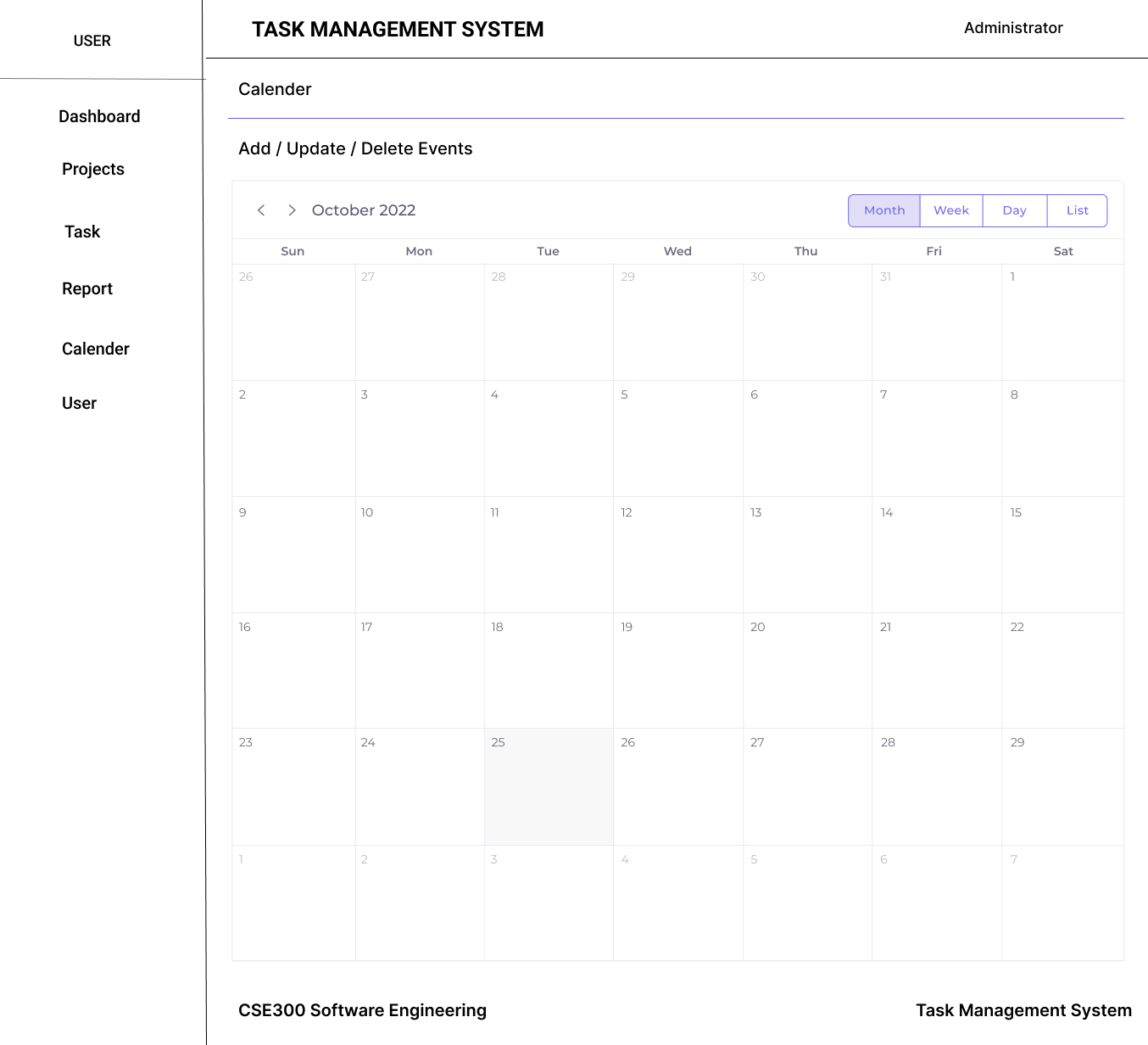
**Add Productivity**



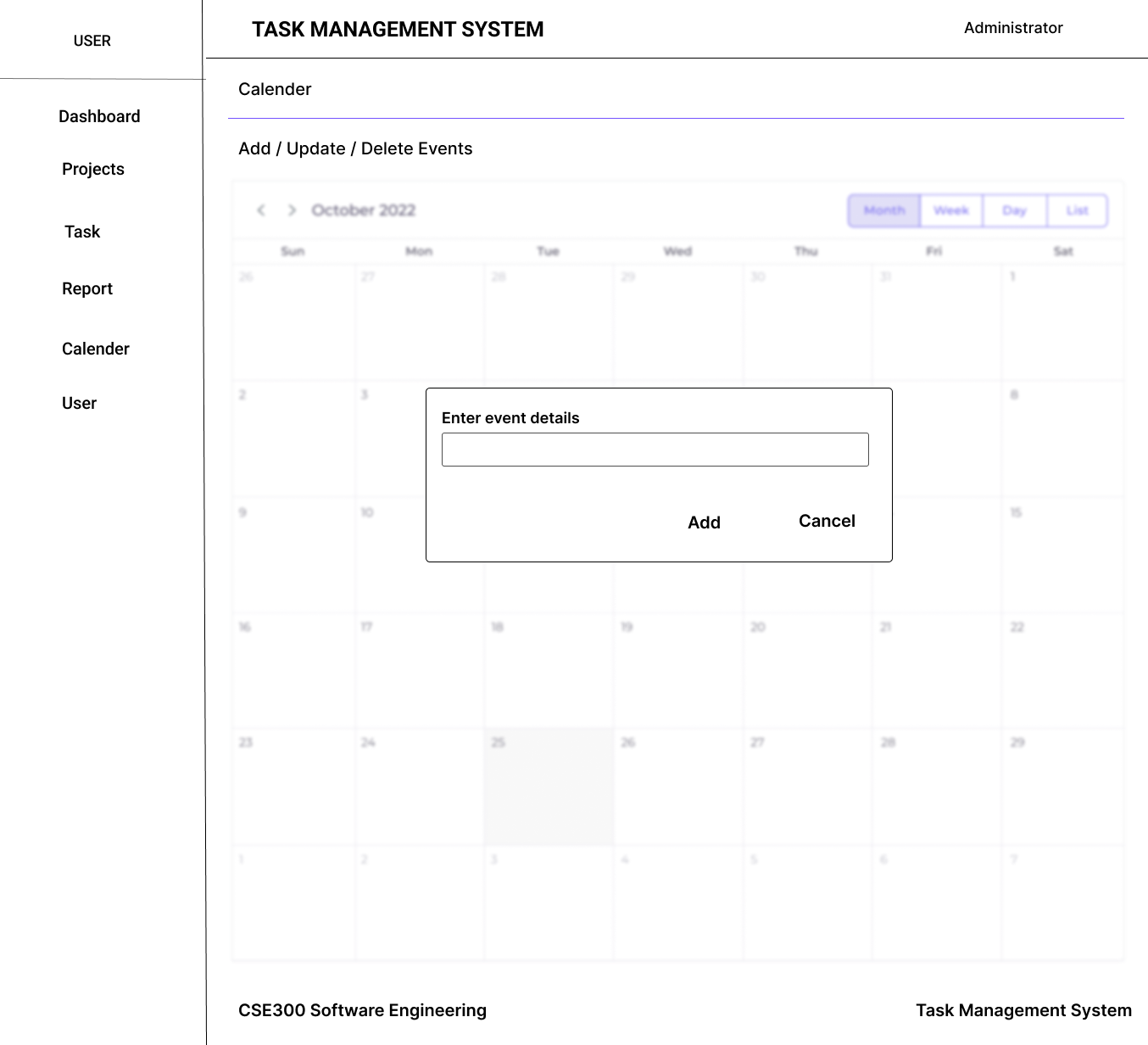
**Reports**



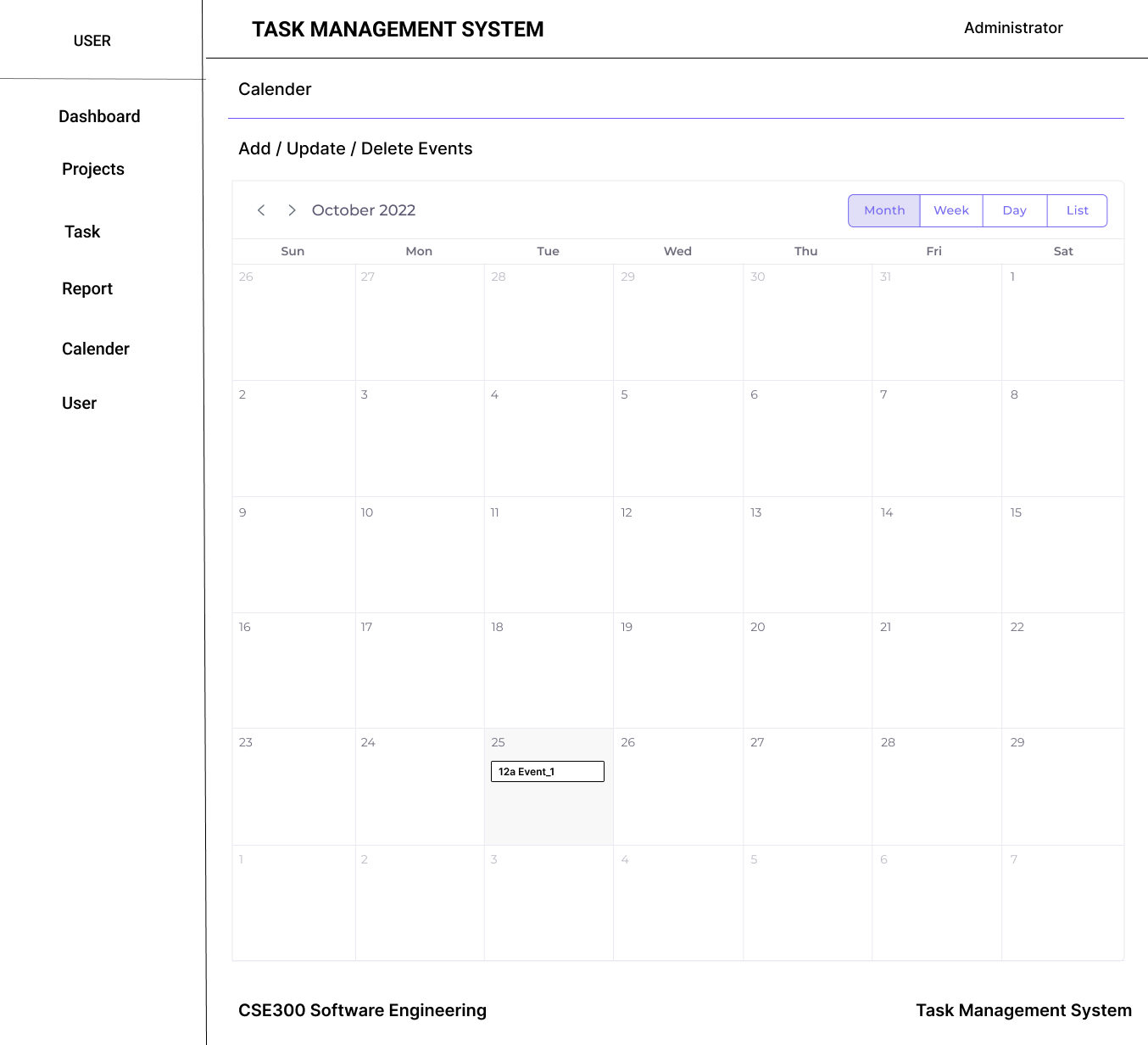
**Calendar**



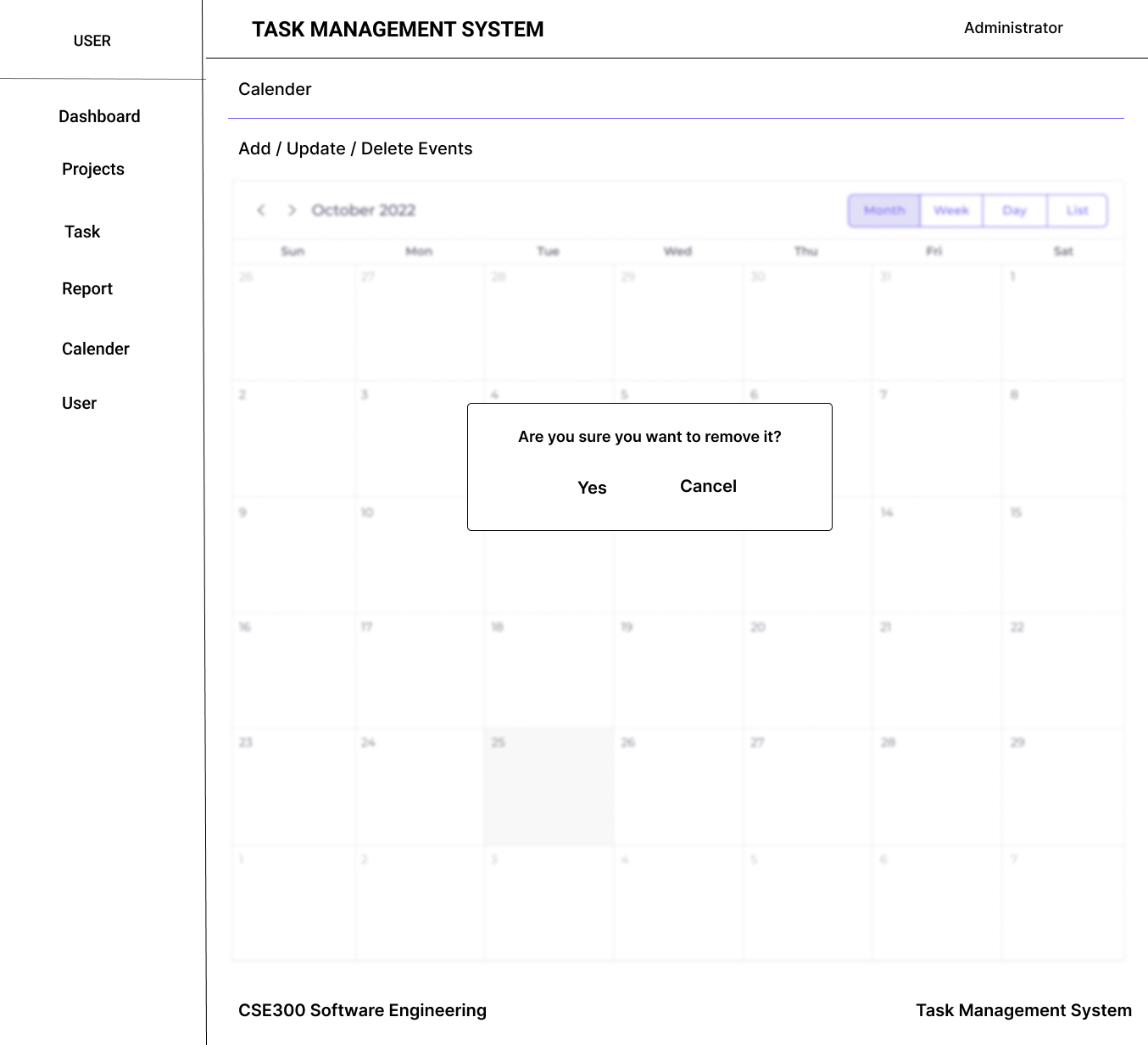
**Add Event**



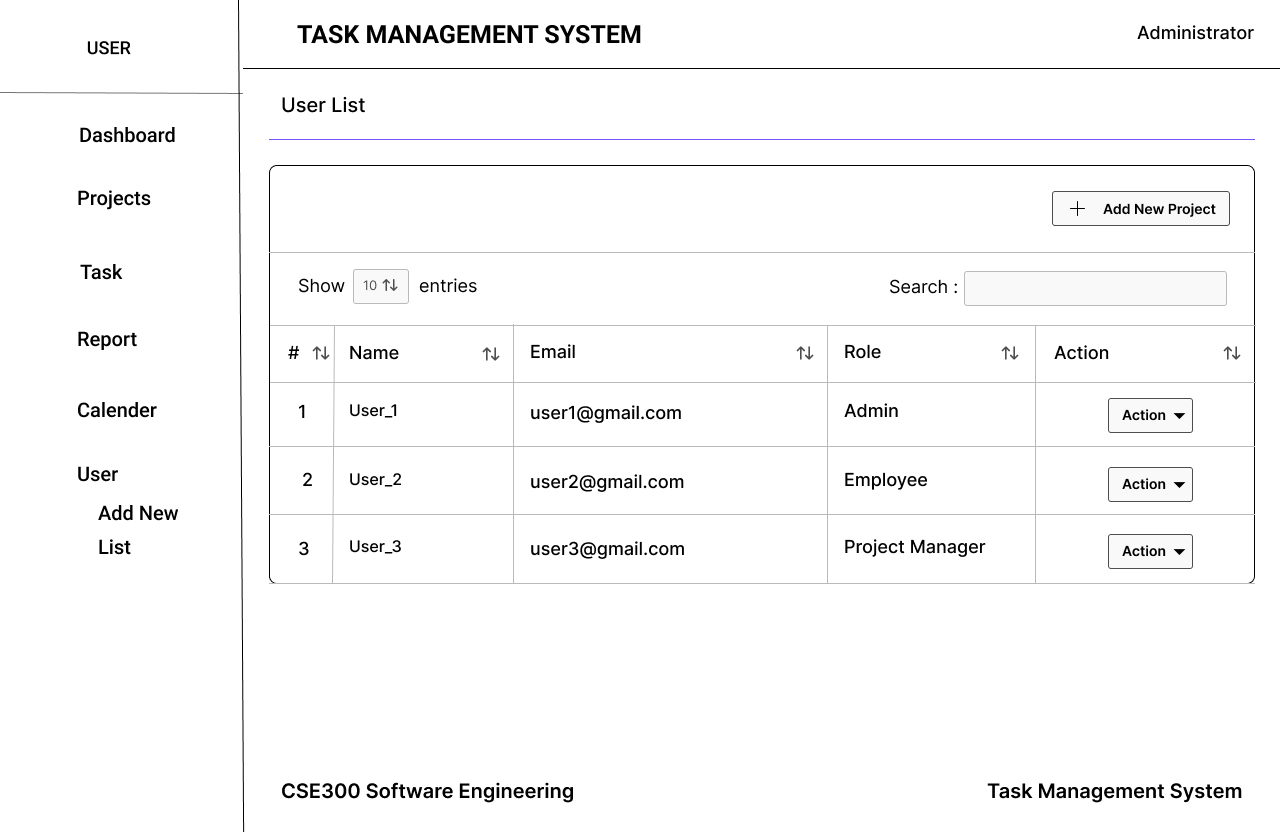
**Event Details**



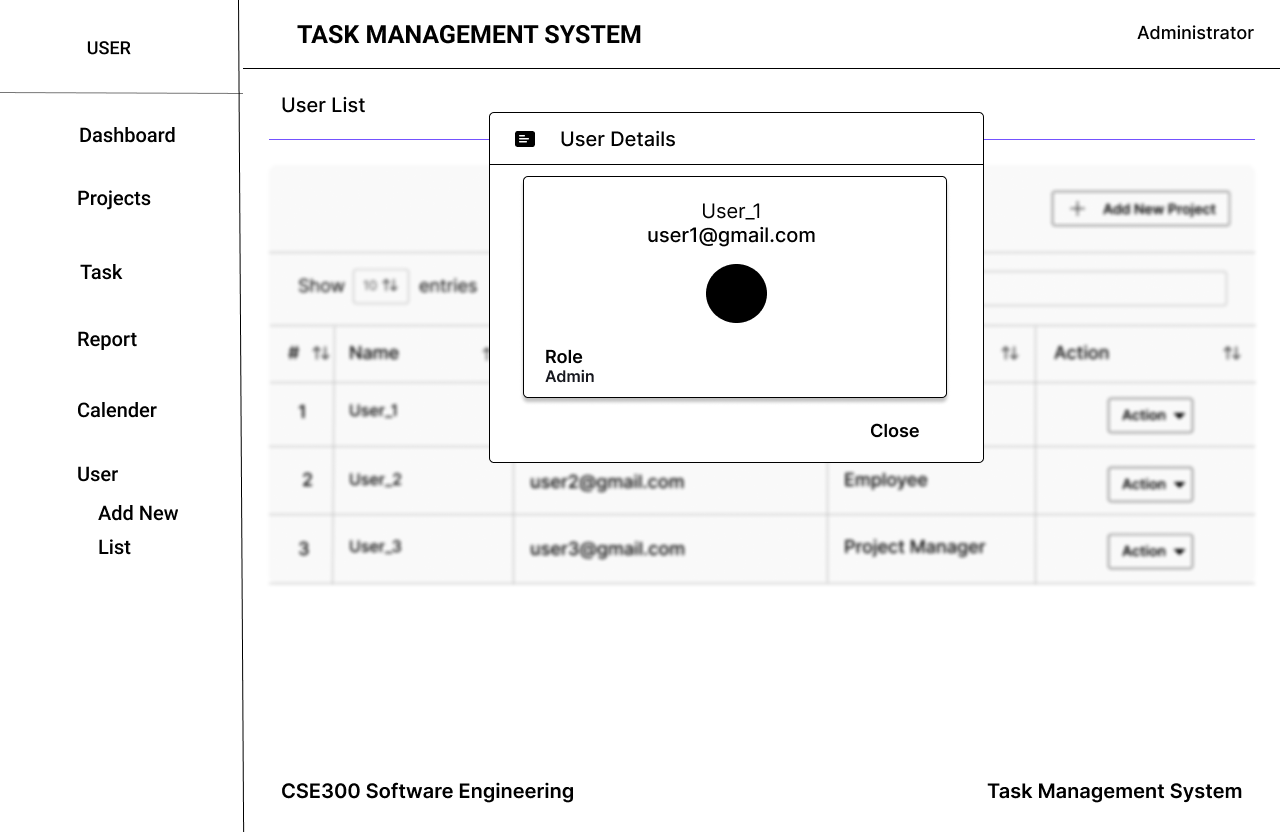
**Delete Event**



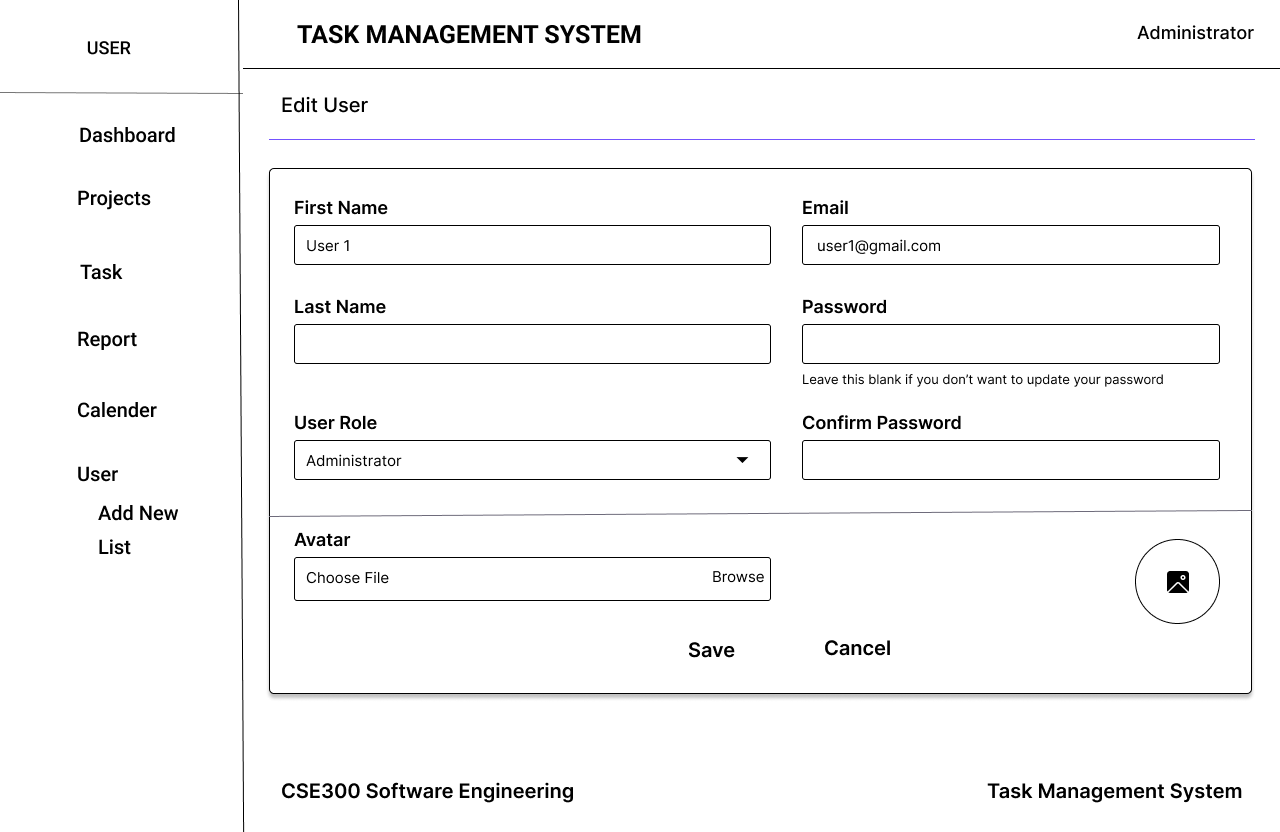
**User List**



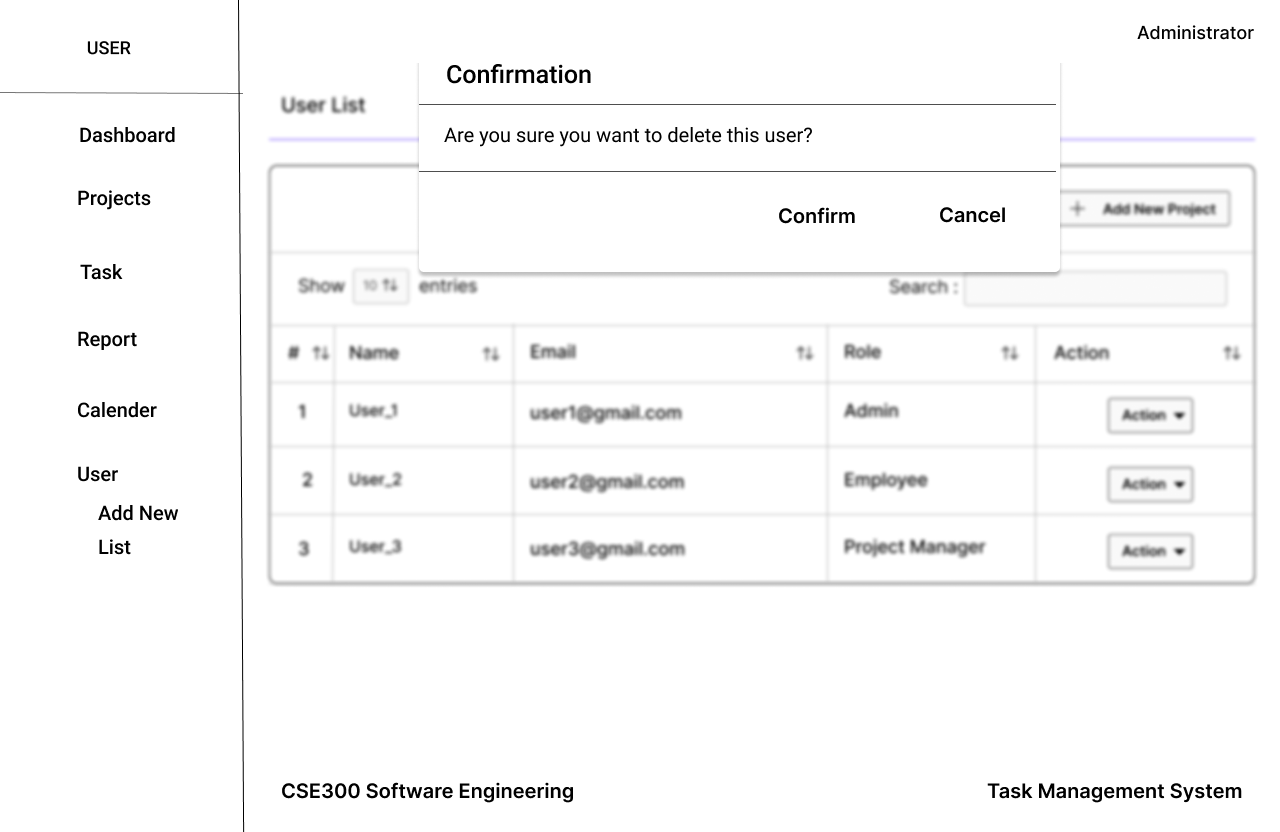
**View user details**



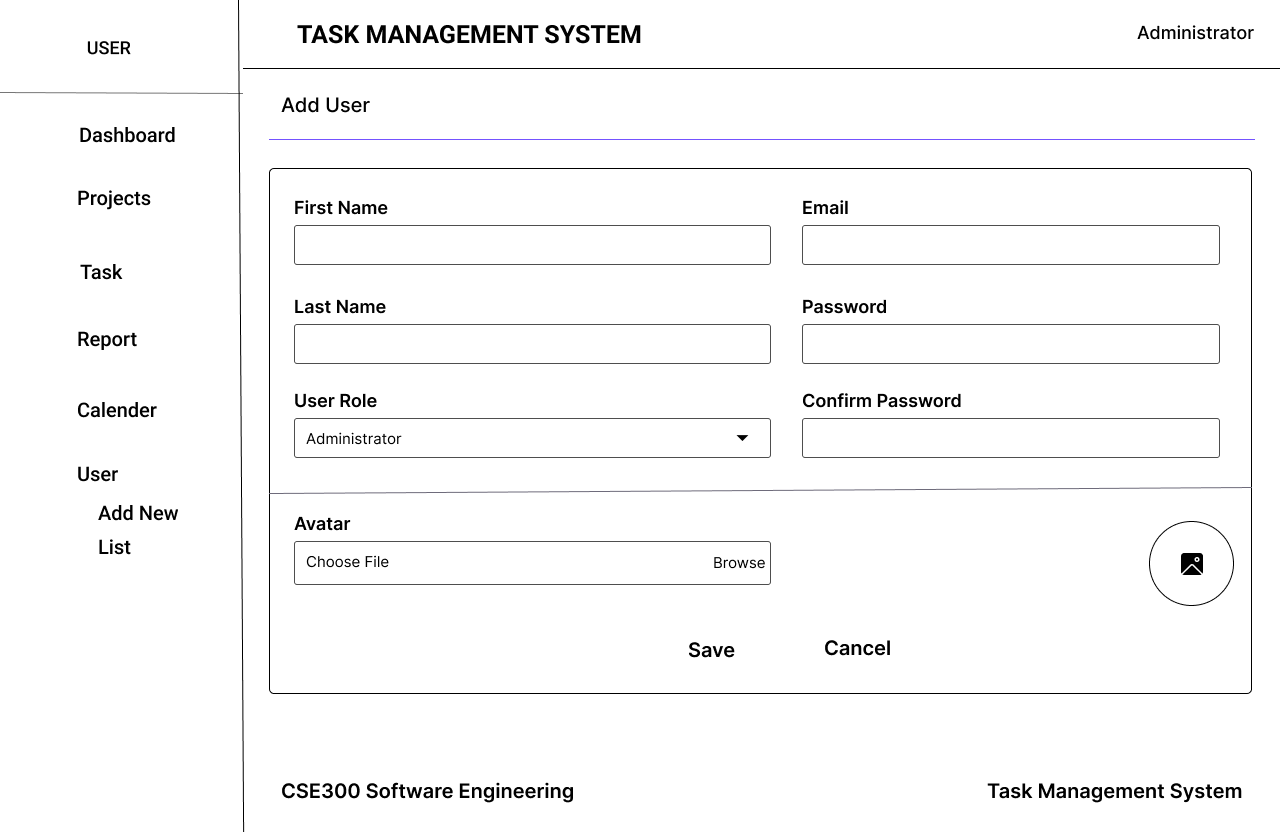
**Edit user details**



**Delete user**



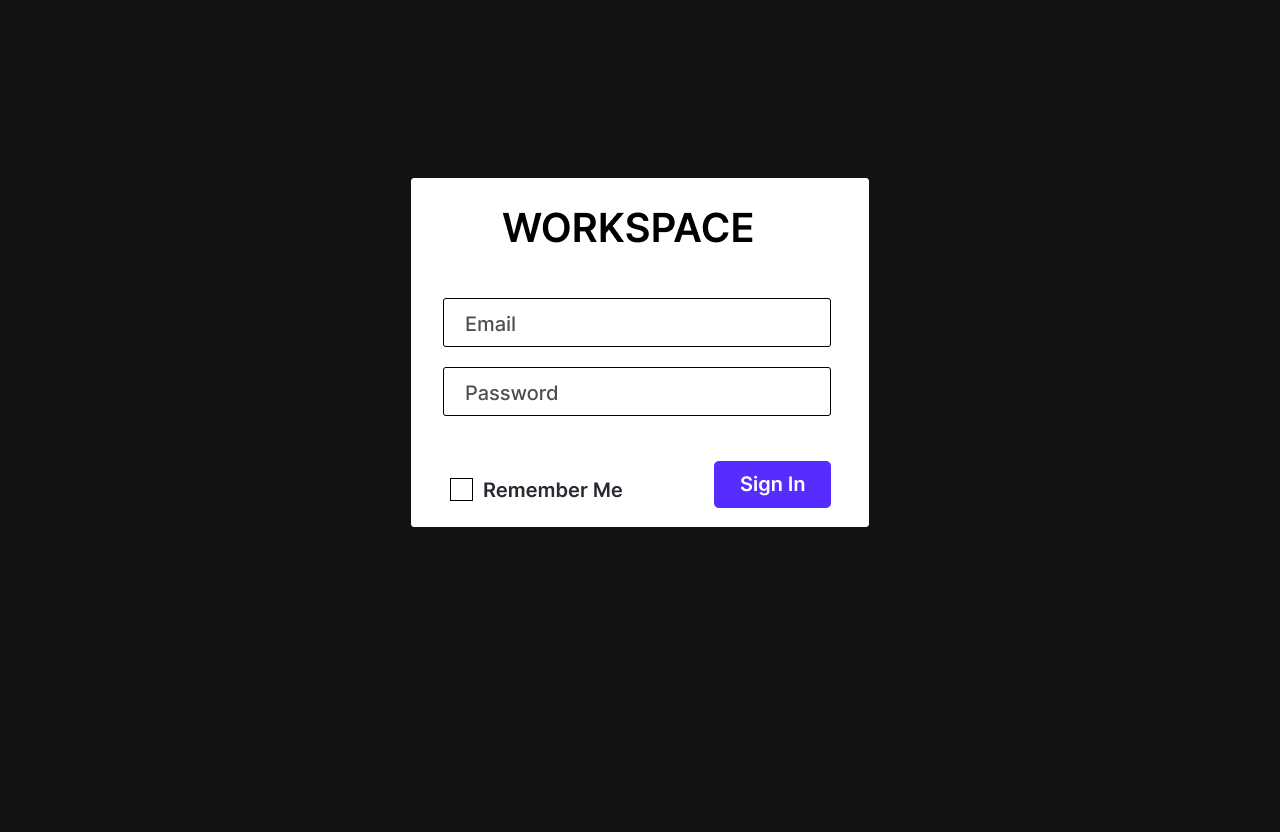
**Add new user**



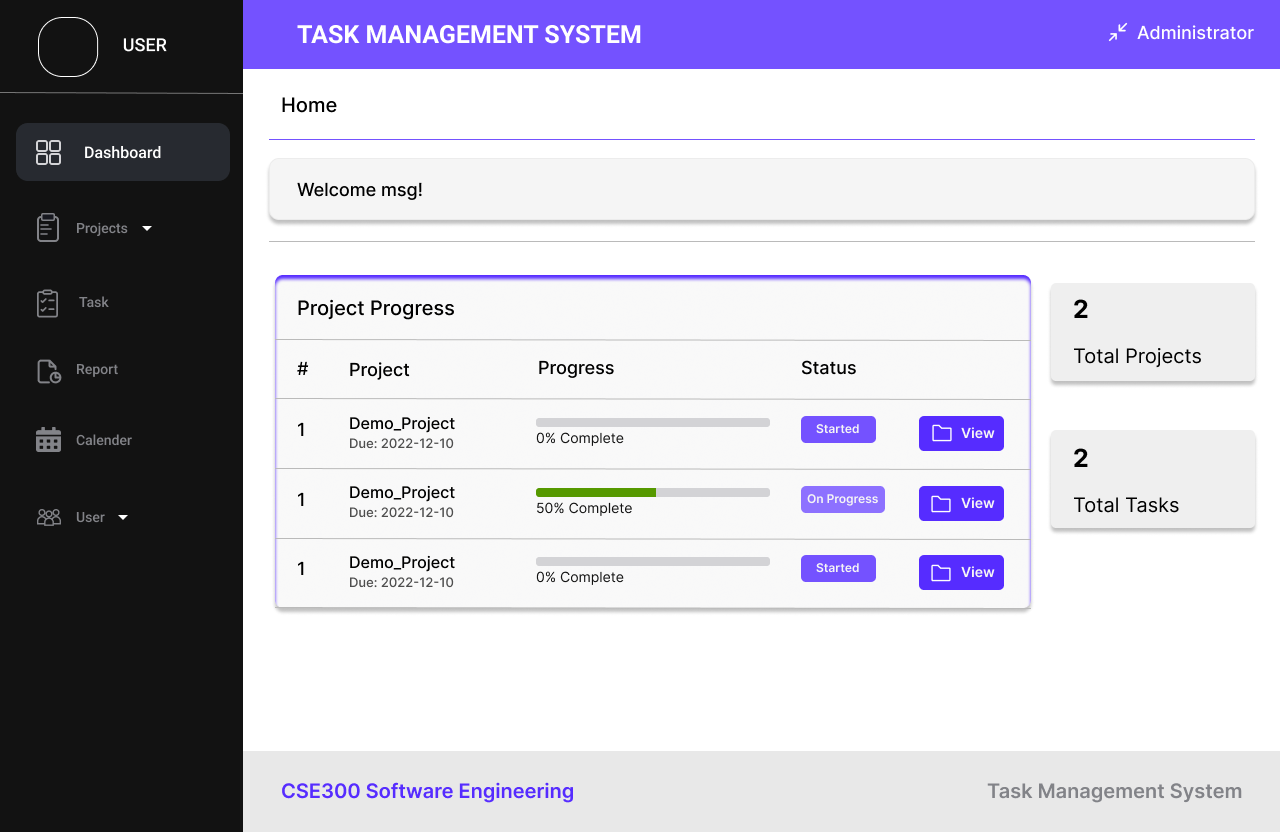
### 6.1.2. Screen Images

The Screen Images for the task management system are displayed below

**Login/Sign In**

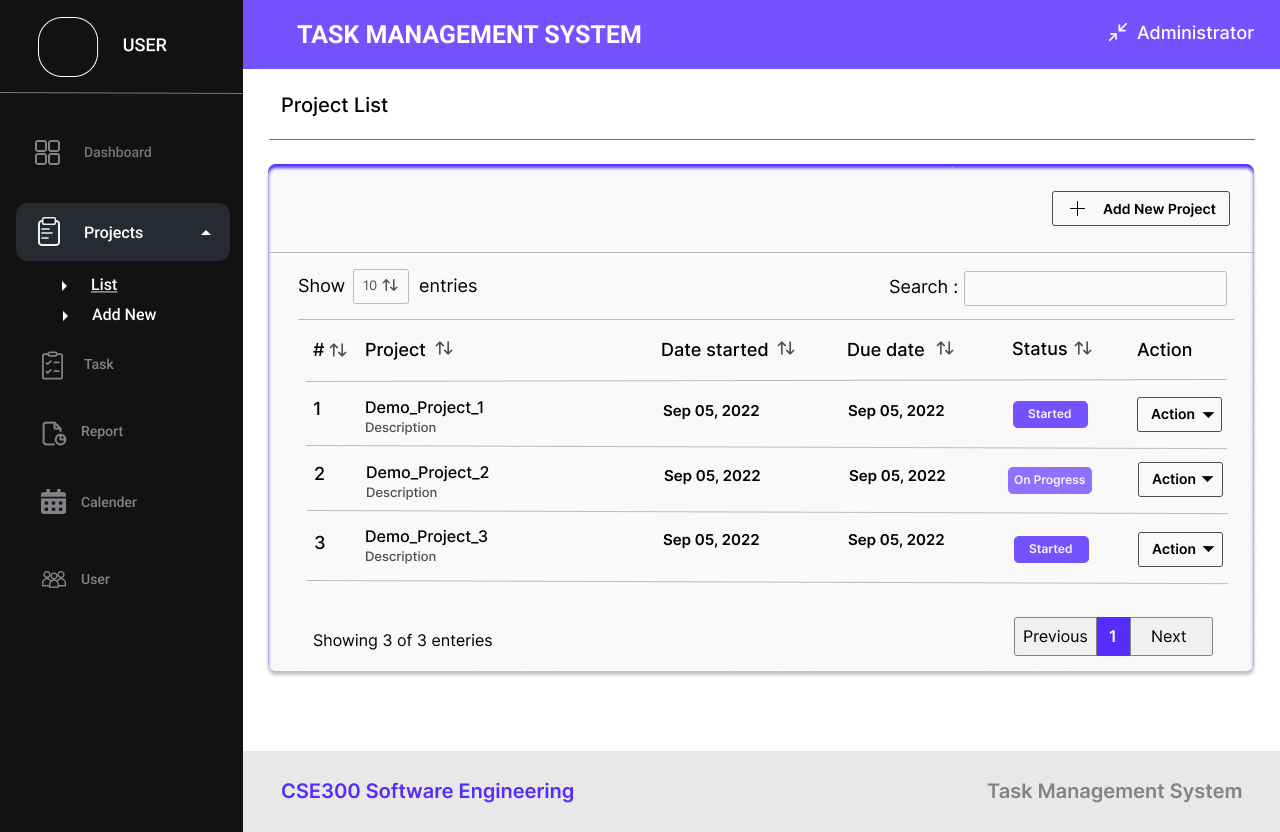
****

**Dashboard**

****

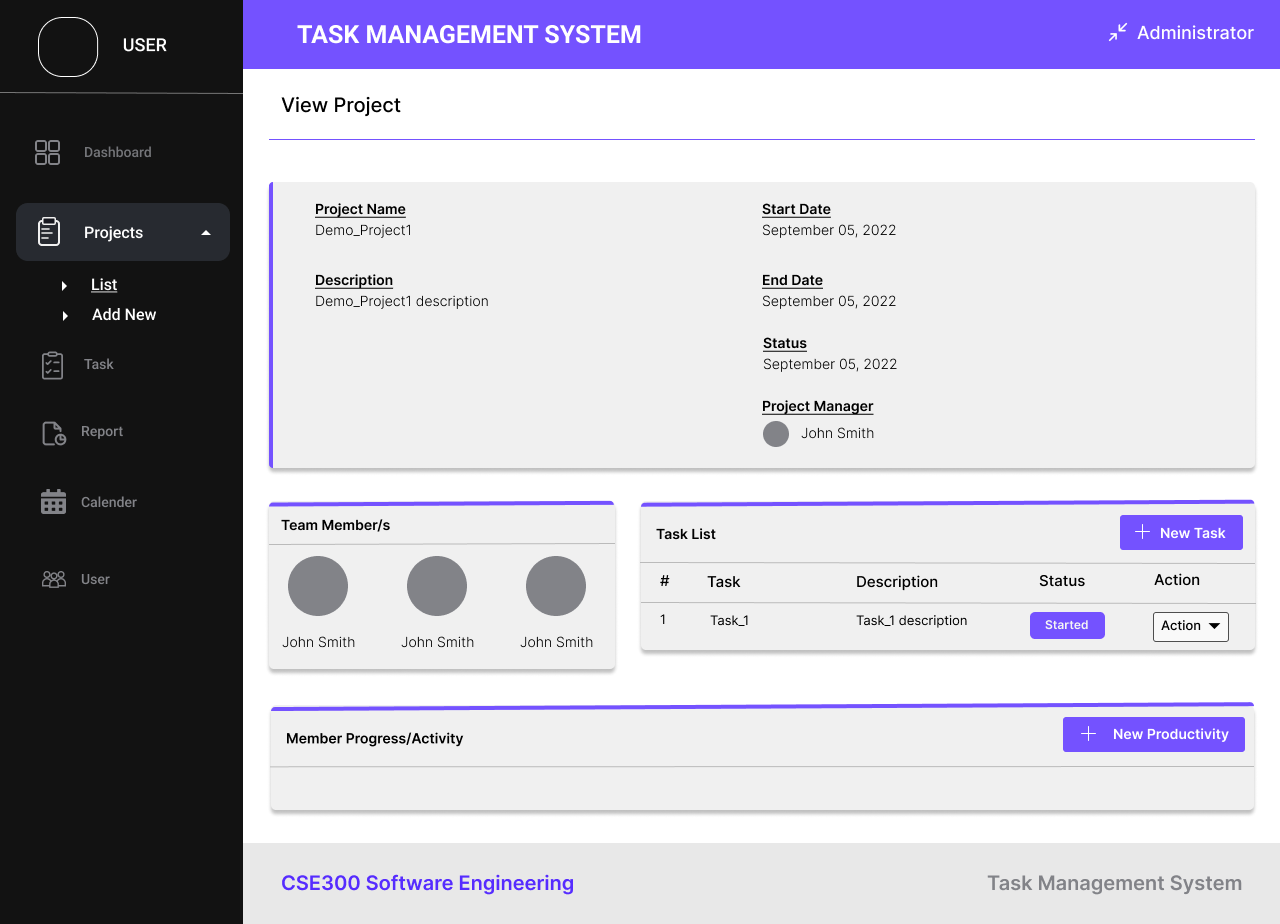
A user would be able to view project progress details along with total projects and total tasks assigned to them on Dashboard.

**Project List**

****

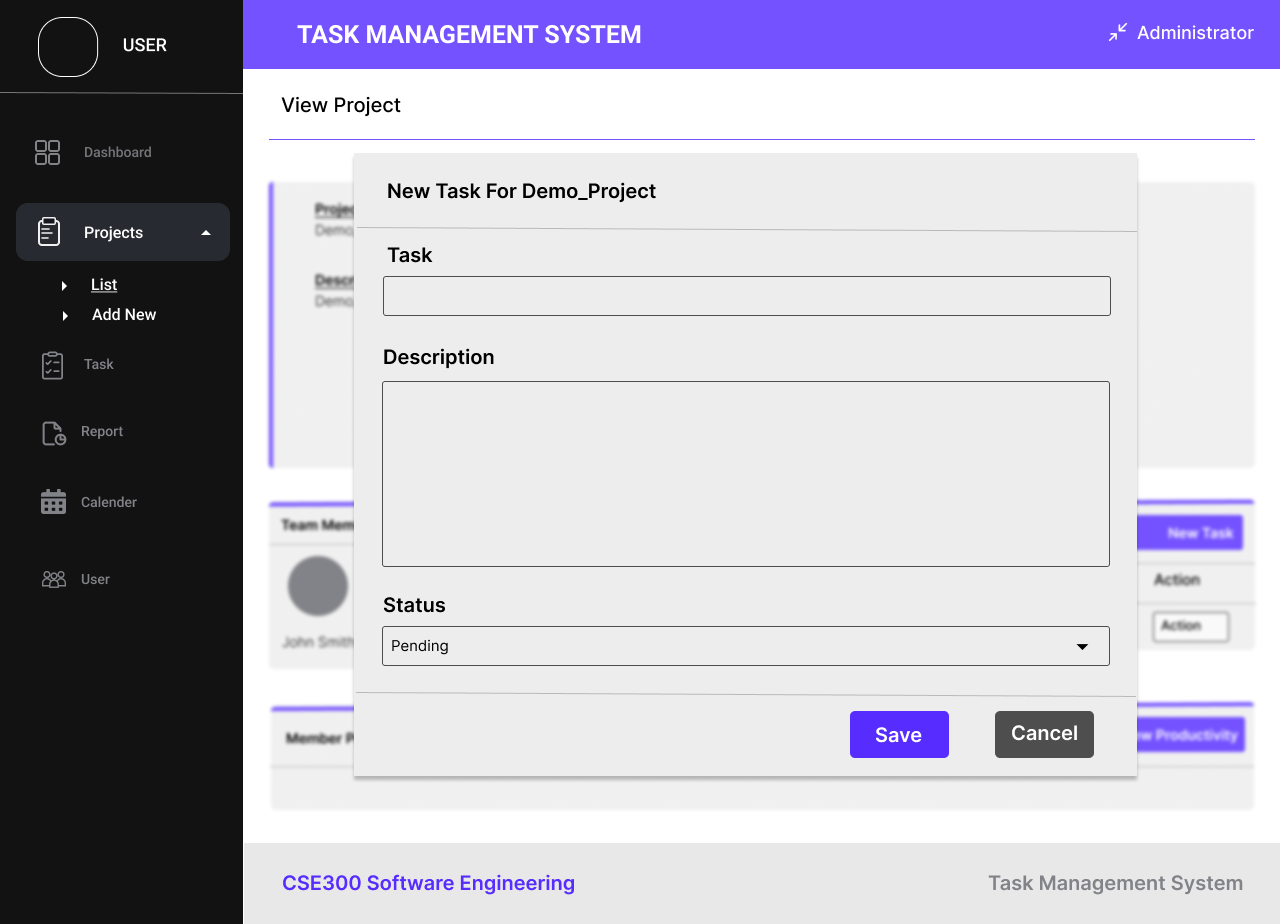
A user would be able to view the project list from the project option which will have details like project name & its description, date started, due date, and status of the project.

**View Project**



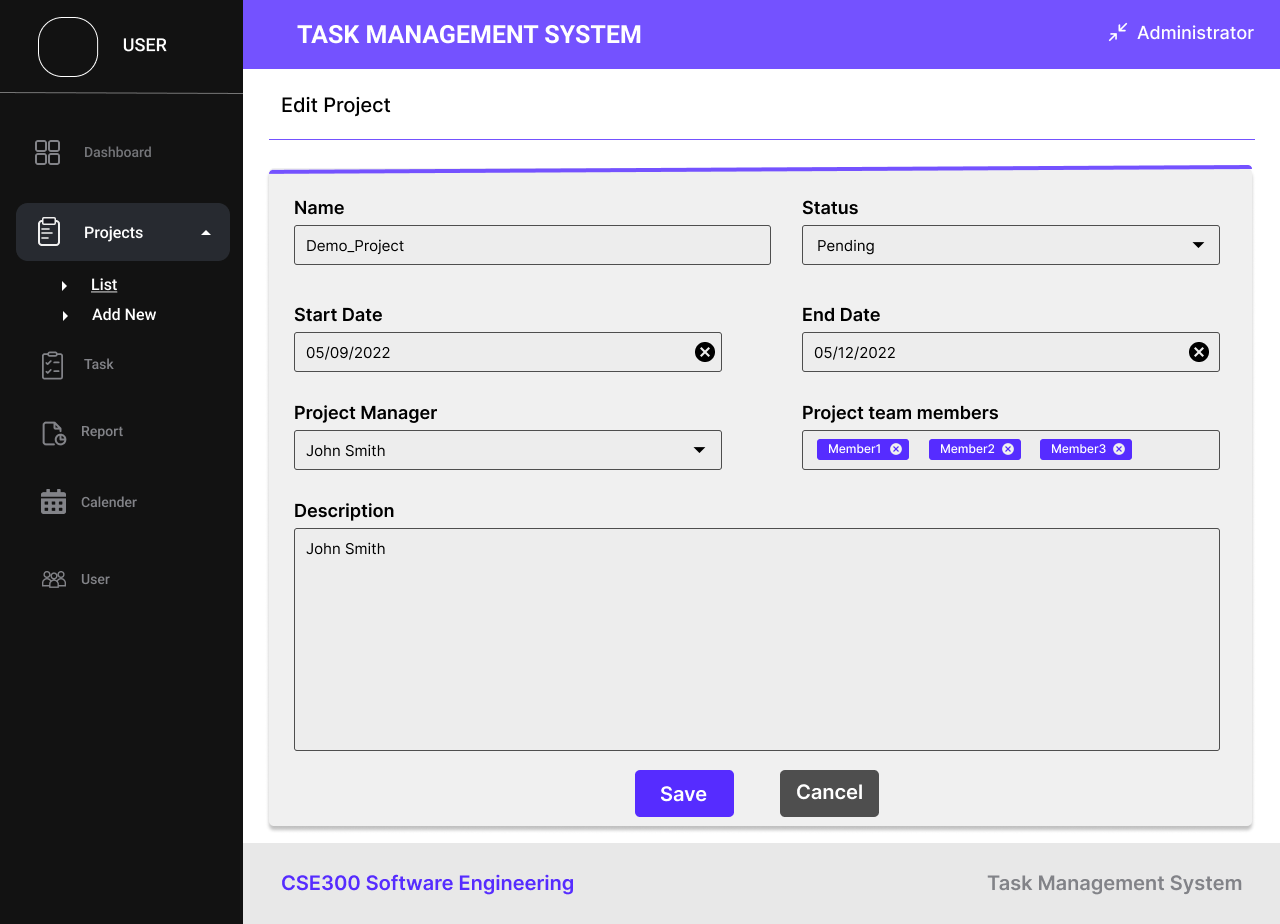
A user would be able to view the project details from the action option which will give a complete overview of project details along with task lists and team members assigned to the ongoing project.

**Add New task**

****

A user would be able to add new tasks from the project section (add new productivity) in which details regarding tasks are required to be added.

**Edit Project**

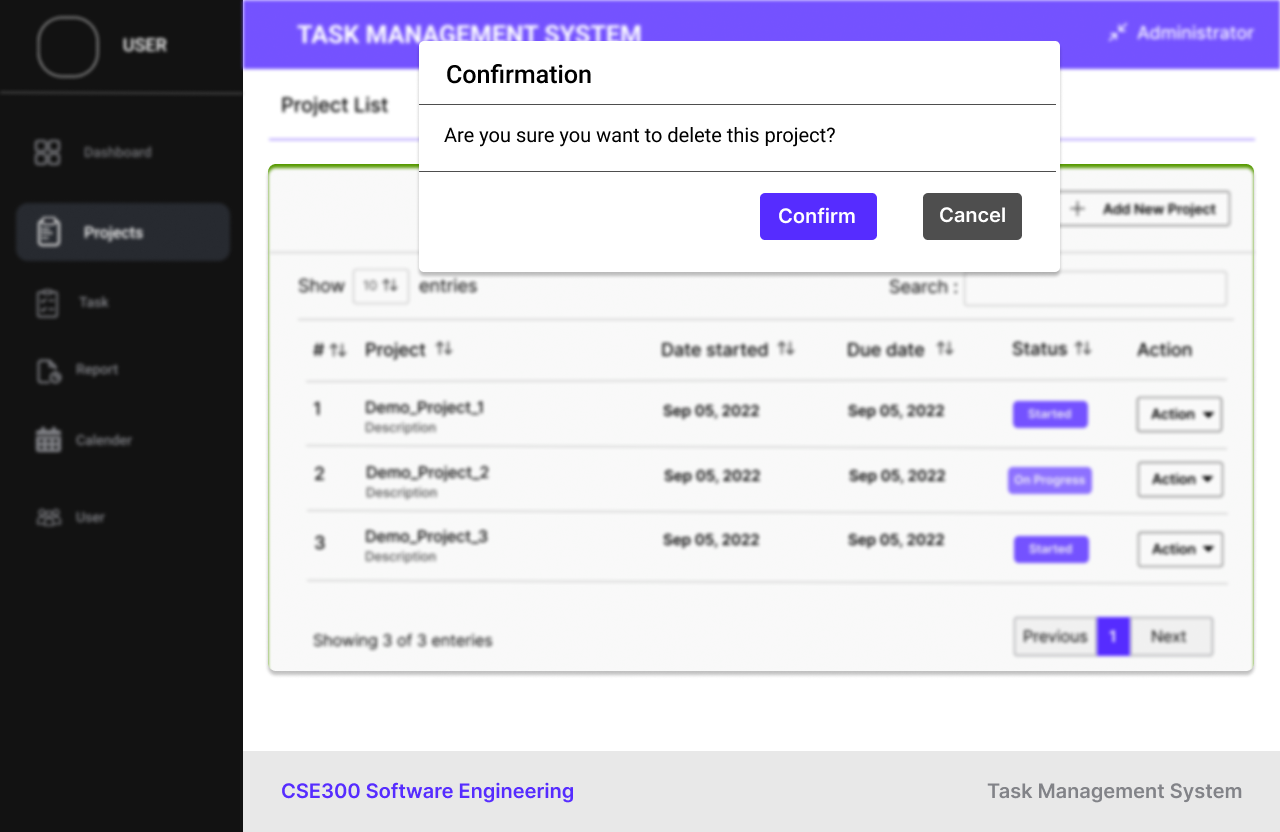


A user would be able to update/edit project details from the action option.

**Project List Action**

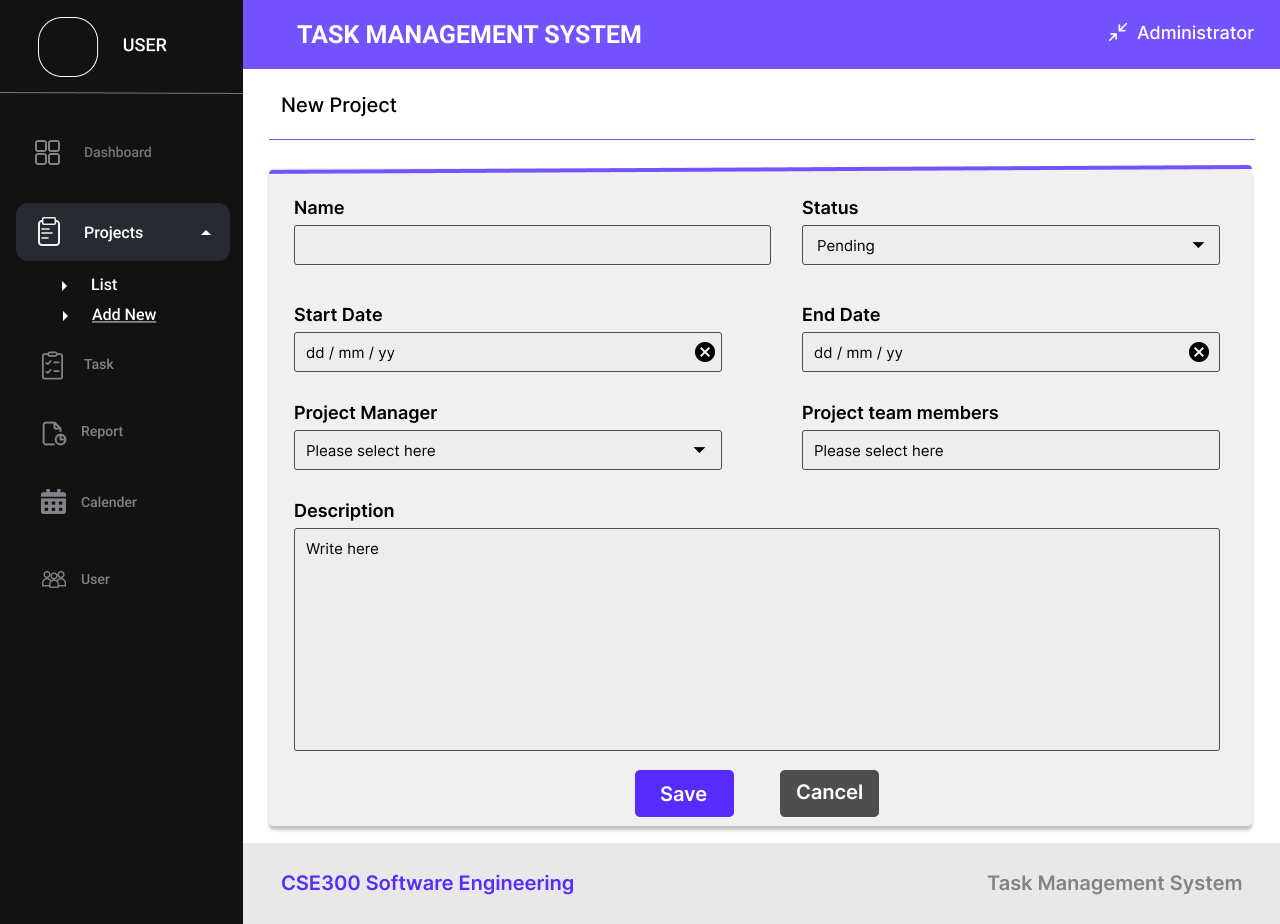
****

**Delete Project**

****

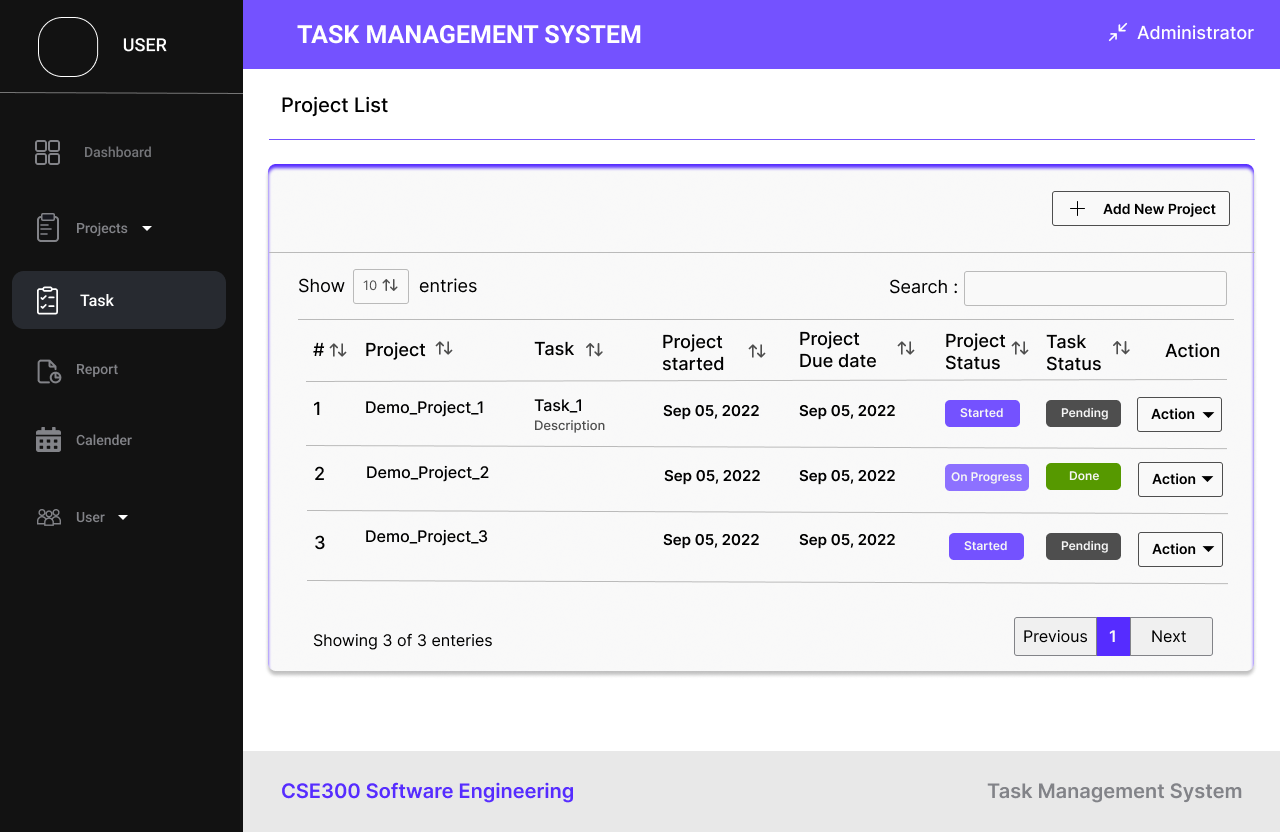
A user would be able to delete the project from the action option which would further ask the confirmation from the user whether to delete the project or not to prevent errors.

**Add New Project**

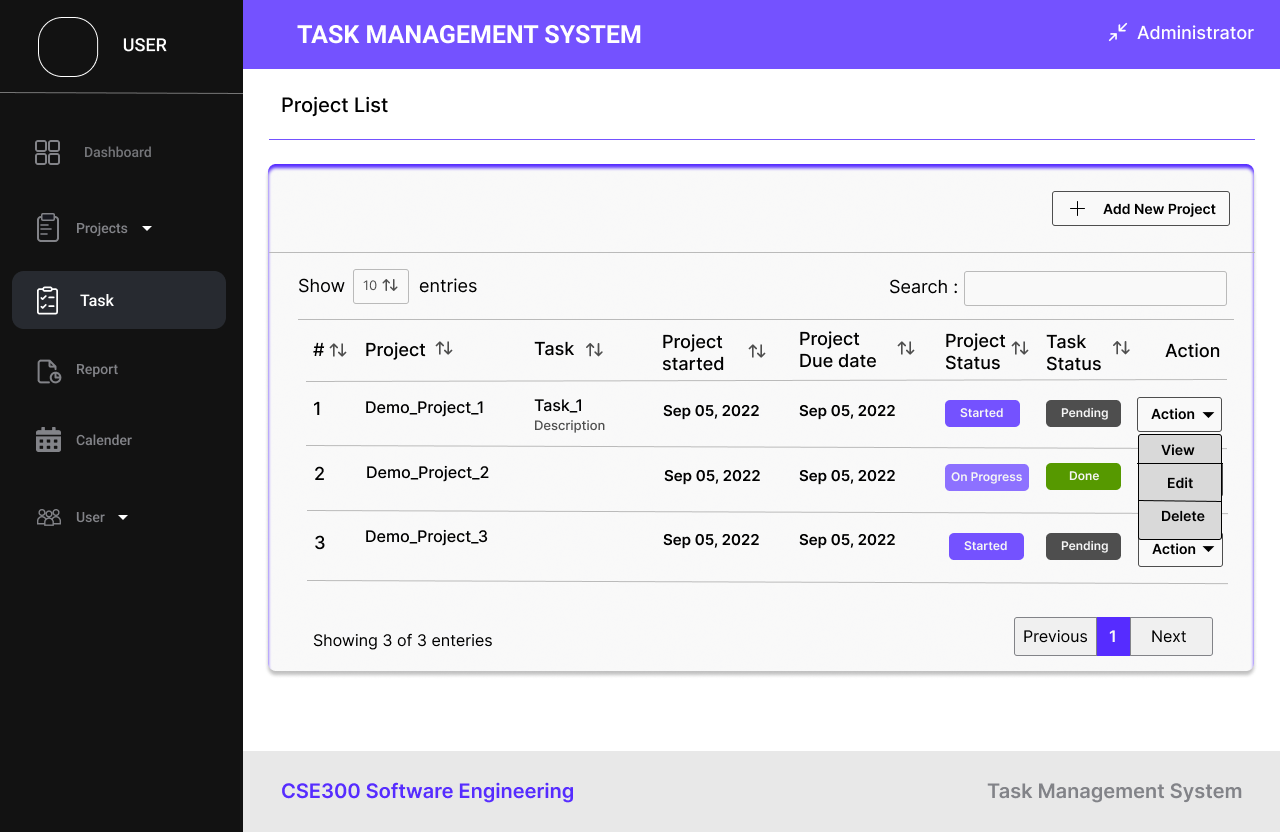
****

A user would be able to add a new project which would further ask for certain details which need to be saved by the user.

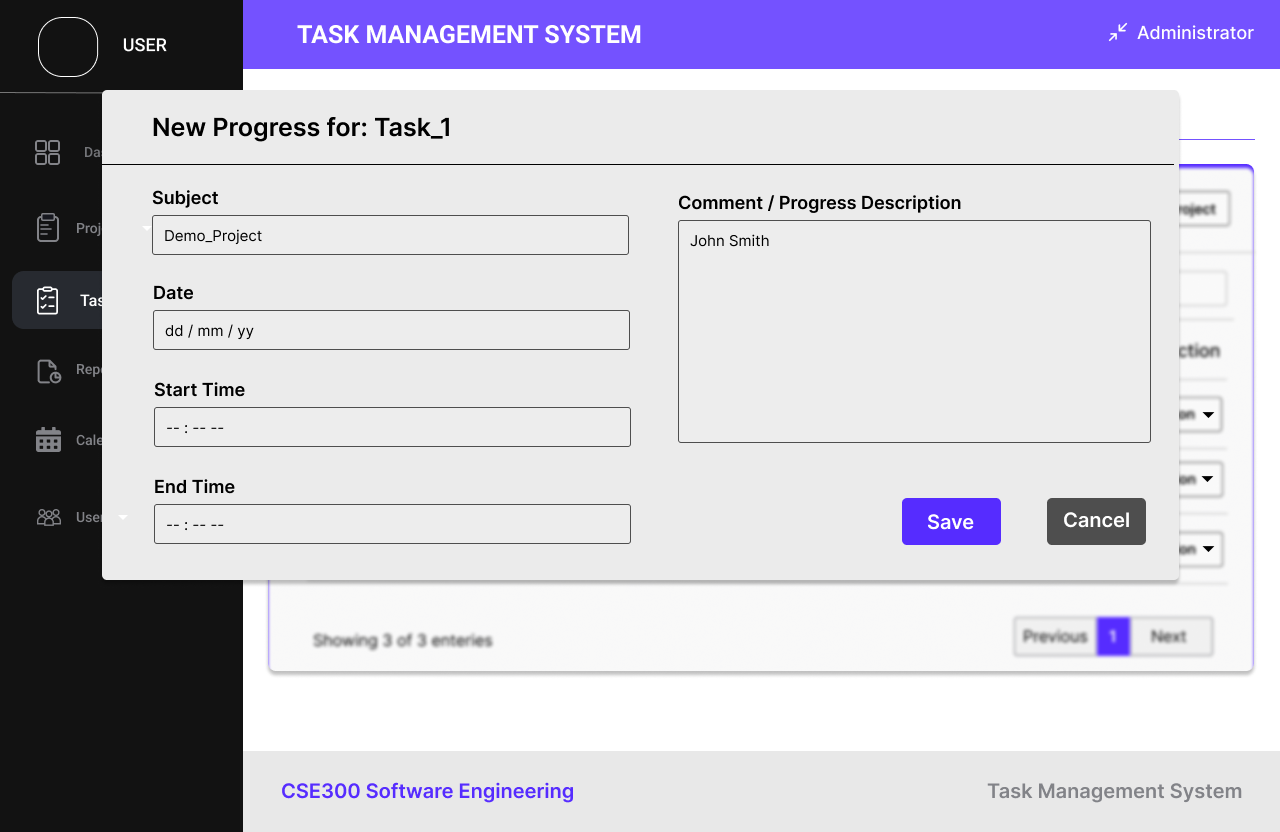
**Task Lists**

****

A user would be able to view the task lists of a project from the task option which would display details like project name, task name and its description, project starting and due date, project status and task status.

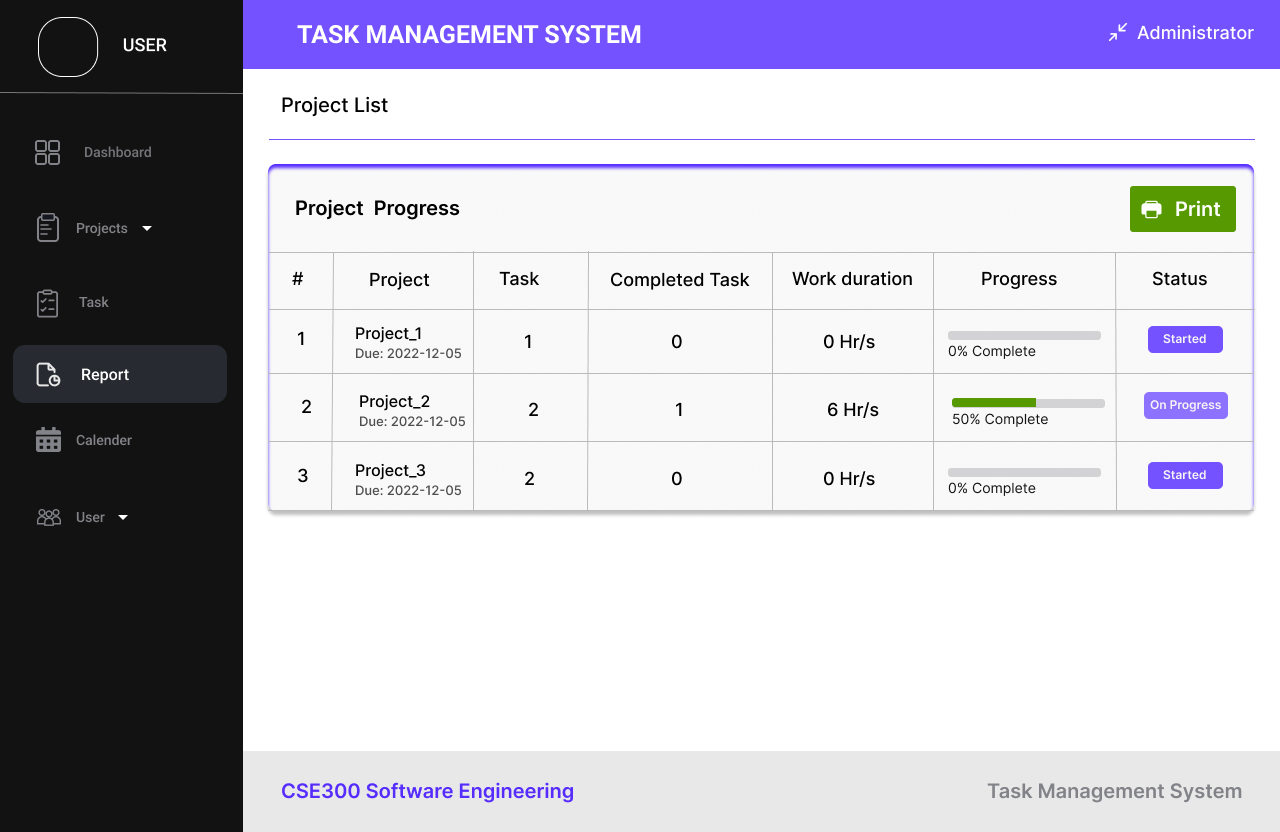
**Task Action Option**

**Add Productivity**

****

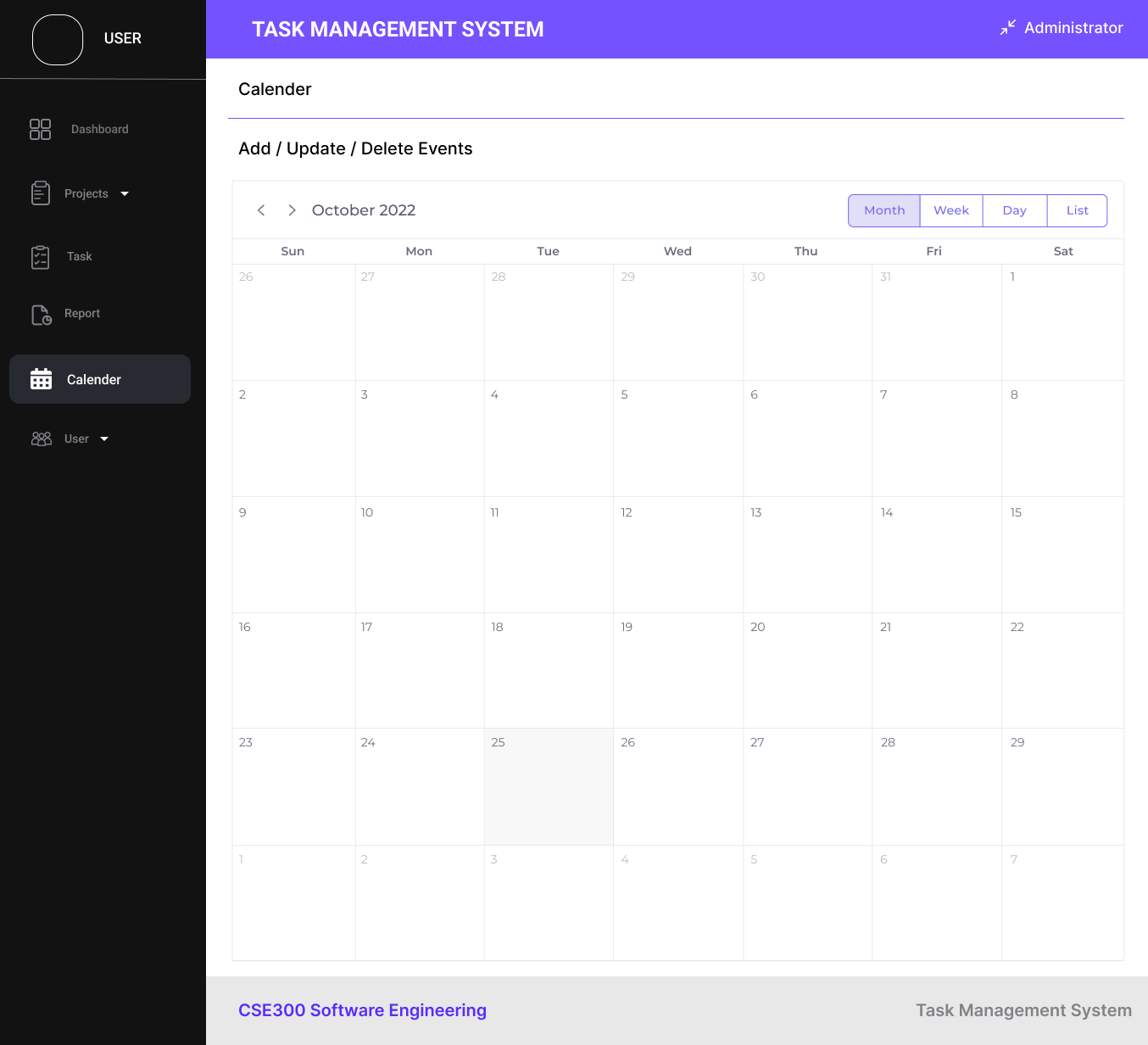
A user would be able to add new productivity/task from the action option .

**Reports**

****

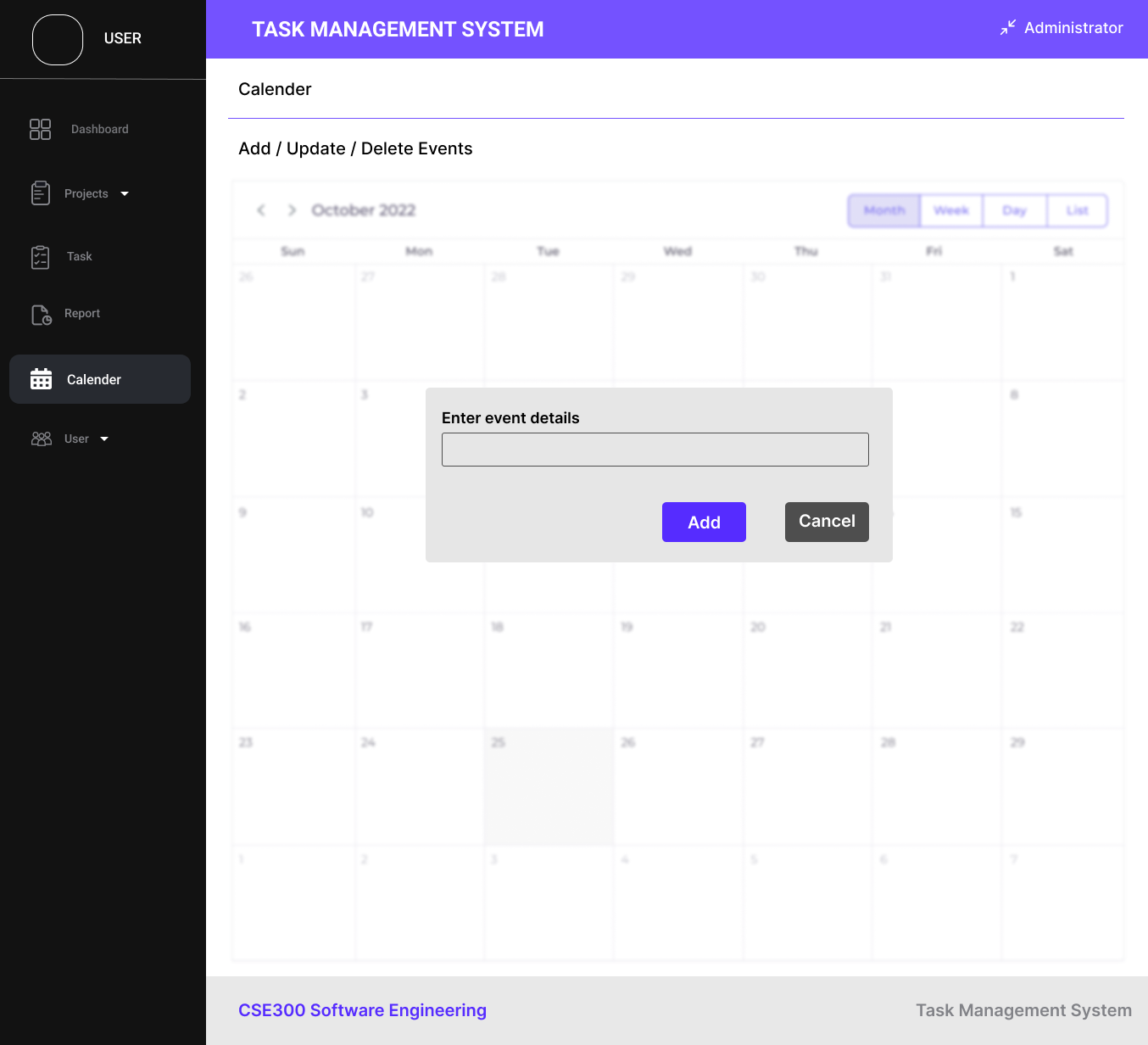
A user would be able to view and print a complete project report from the report option which would display project and task name, total completed tasks, work duration, progress and their status.

**Calendar**

****

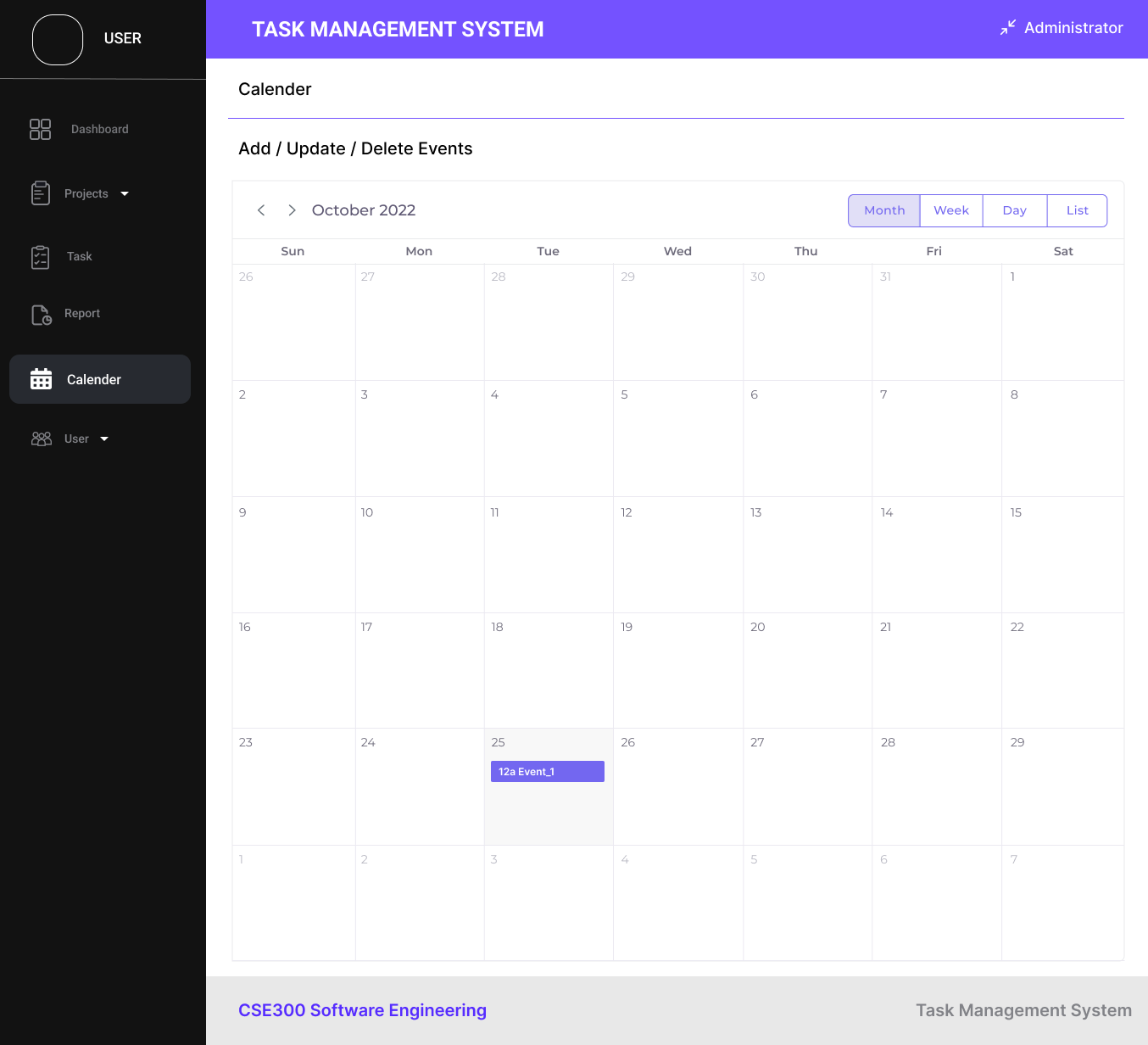
A user would be able to add/update/delete events from the calendar option.

**Add Event**

****

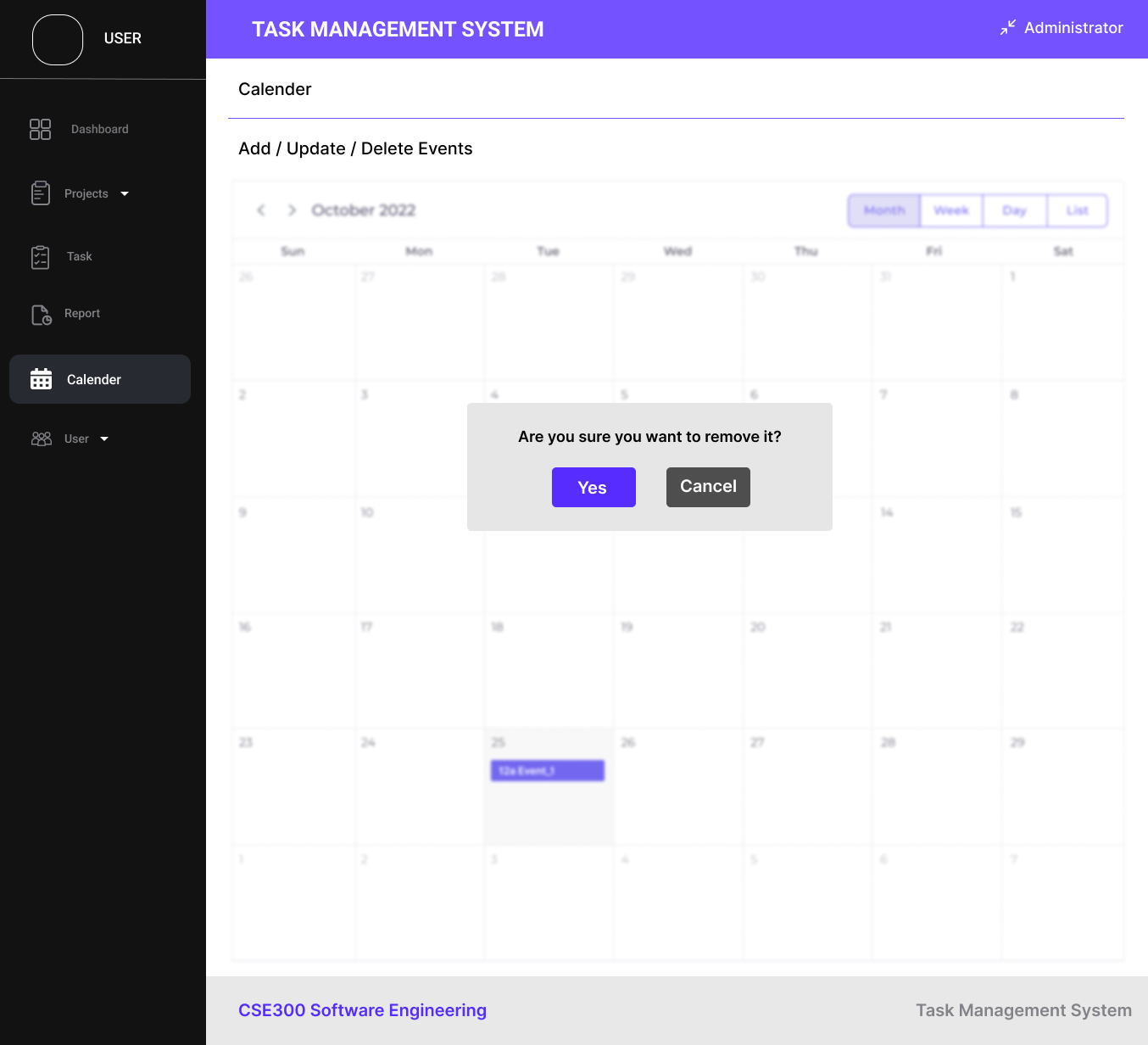
A user would be able to add events by clicking on the date and by entering the event details.

**Event Display**

****

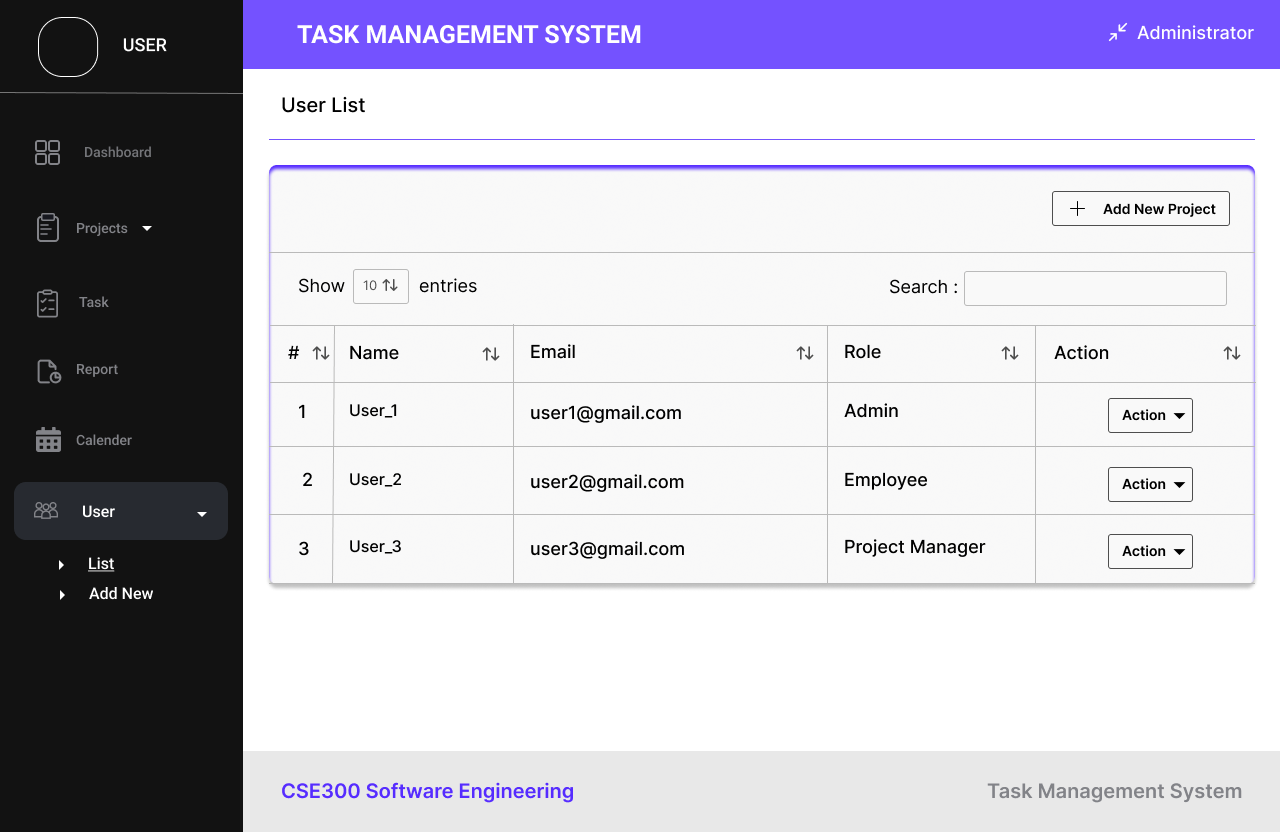
A user would be able to view the added event on a particular date.

**Delete Event**

****

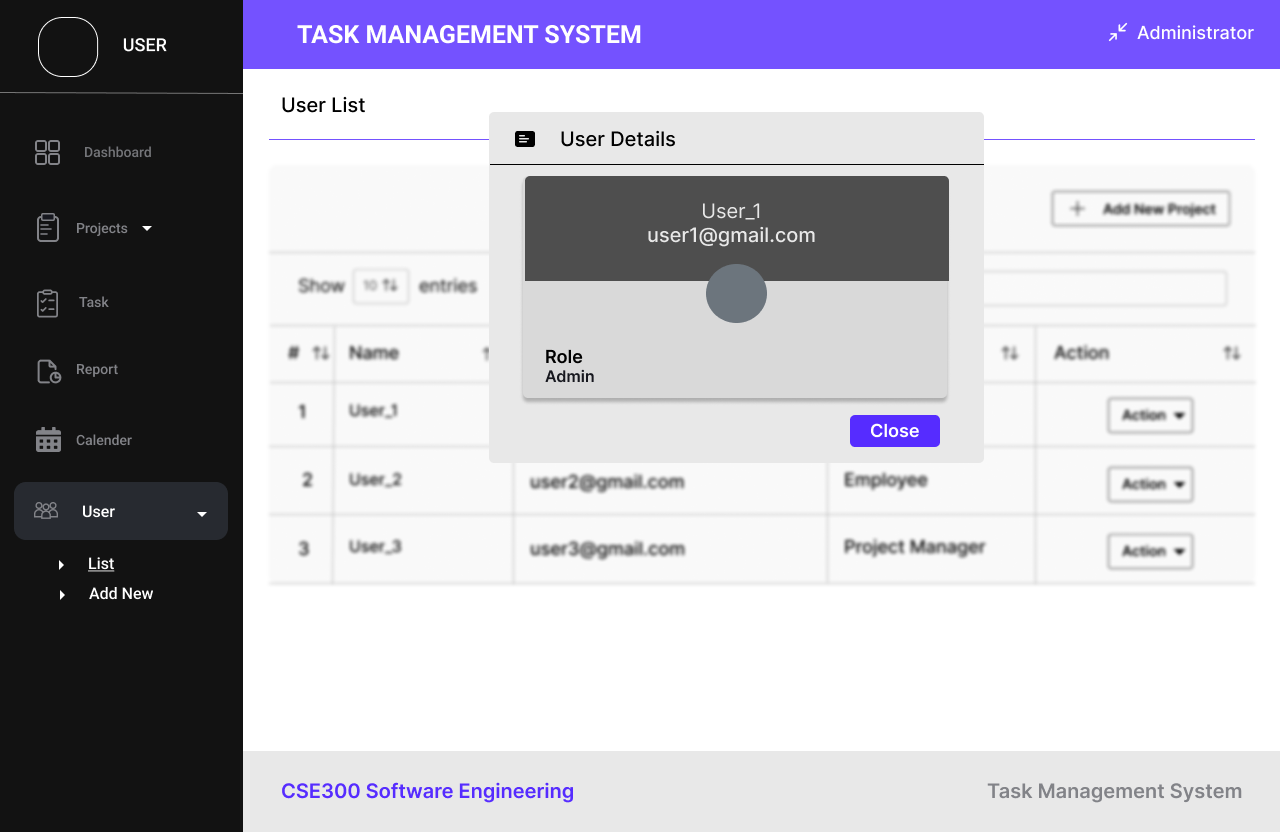
A user would be able to delete an event by clicking on the event which would further ask for confirmation to prevent errors.

**User List**

****

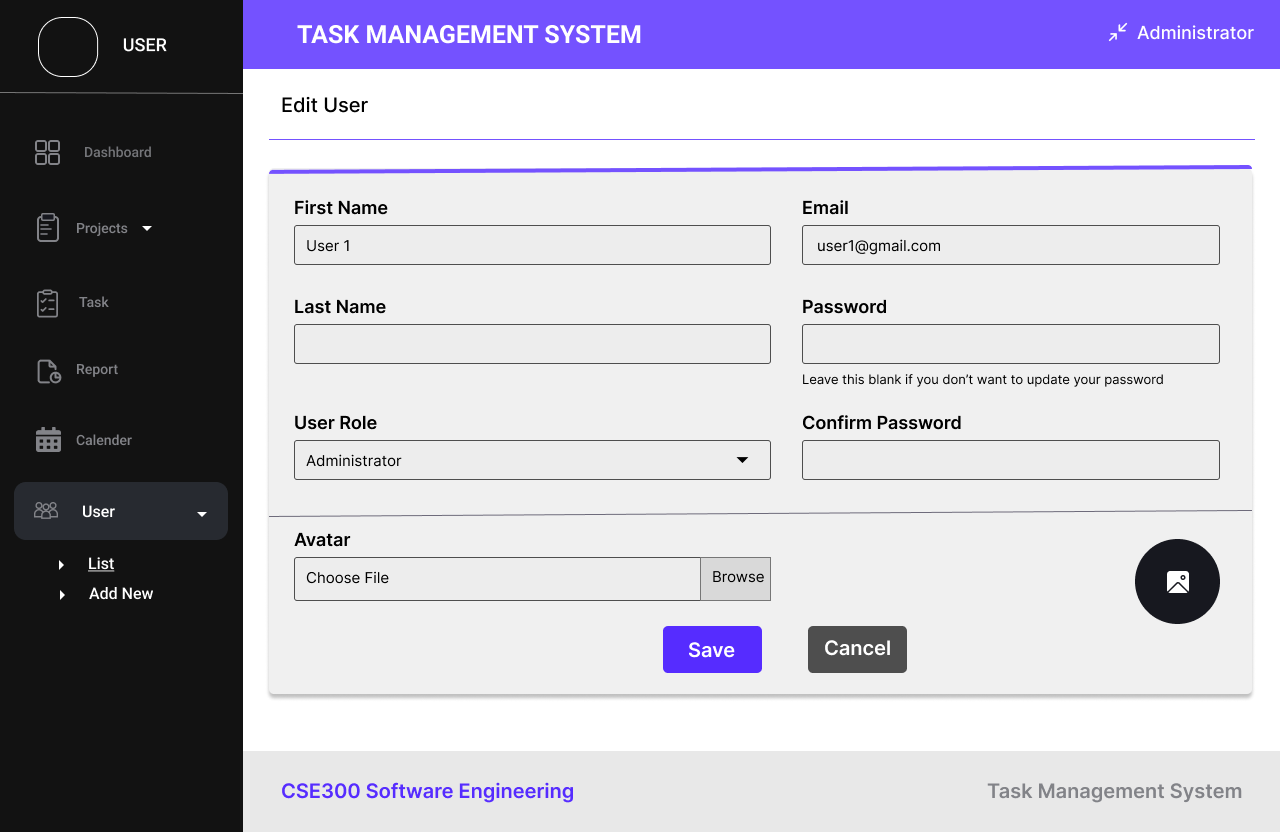
An admin/user would be able to view the list of users along with their email ids and role from the user option.

**View User Details**

****

A user would be able to view the details from the action option.

**Edit User Details**

****

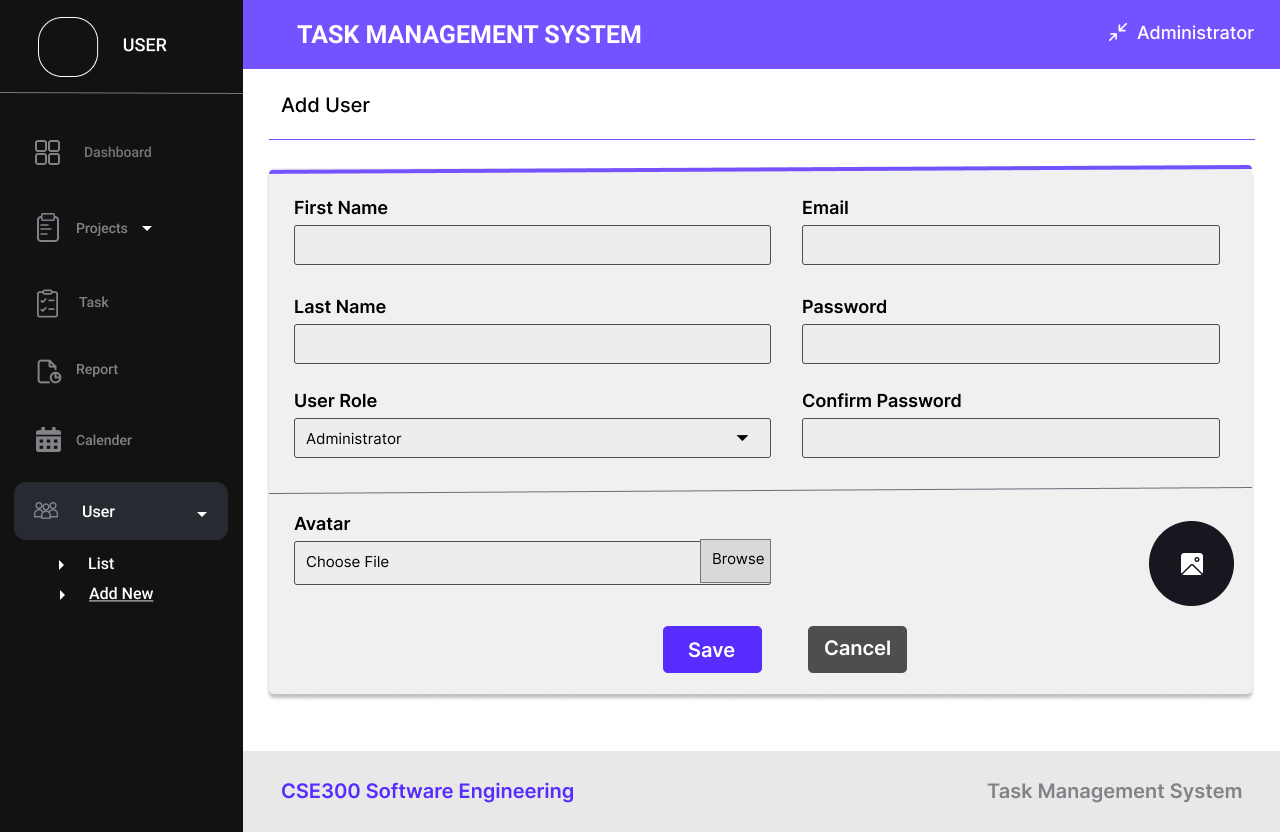
This will allow a user to update/edit their personal details from the action option.

**Delete User**

****

This will allow one to delete a particular user from the action option which would further ask for confirmation to prevent errors.

**Add new user**

****

This will allow one to add new users and their details like first name, last name, email, user role, creating and confirming password and avatar.

### 6.1.2. Objects and Actions

| **Navigation Buttons/Links** | **Activity performed** |
| --- | --- |
| Dashboard | Displays project progress, total projects and task assigned |
| Project → List | Displays list of projects with their detailed information |
| Project → Add New | Allows the user to add new project by entering the required details |
| Tasks | Displays tasks of a project along with their detailed information. |
| Report | Displays project progress with total tasks completed and their work duration |
| Calendar | Allows the user to add, delete and update events |
| User → Add New | Allows to add new user by entering the required details |
| User → List | Displays total users along with their personal information |

### **6.1.3. Interface Design Rules**

The following principles of interface design are considered while developing the user interface:

* **Consistency:** There is consistency in terms of colors, buttons, dialog box, Identical technology in prompts, menus, help, capitalization, layout, fonts etc across the whole software. Consistency has been maintained in similar operations to achieve similar tasks which makes things easy to learn and use.
* **Offer Informative Feedback:** For every user action, there should be system feedback. Sends back information to the user that what action has been done and what has been accomplished. This allows users to continue with the activities.
* **Design Dialogues to Yield Closure:** Sequences of actions are organized into groups with beginning, middle and end.
* **Prevent Errors and constraints:** The system is designed such that the user can not make serious errors. Grayed out menus that are not appropriate and have not allowed alphabetic characters in numeric entry fields to prevent unintentional errors.
* **Permit Easy Reversal of Actions:** As much as possible, all actions are reversible. This feature relieves anxiety in the user and makes the user comfortable as the user knows that my action can be reverted back.
* **Reduce short term memory load:** Humans have limited capacity to store/process information in short term memory .Website locations remain visible. Multiple page displays are consolidated. Sufficient training time has been given for difficult sequences of actions.
* **Visibility:** More visible functions are, more users know what to do next. The design allows users to view all the required options.
* **Affordance:** Refers to an attribute of an object that allows people to know how to use it.

## **6.2. Components Available**

|  | Component | Examples for the website |
| --- | --- | --- |
| Input Controls | Buttons |  |
| Dropdown lists |  |
| Calendar |  |
| Text fields |  |
| Navigational Components | Search Field |  |
| Icons |  |
| Navigation bar |  |
| Information Components | Progress bar |  |