

PROJECT MANAGEMENT

MINI PROJECT

Submitted by

MADHIUKSHA S

22ALR045

AKSHARA R K

22ALR001

MADHUMITHA S

22ALR047

in partial fulfillment of the requirements for

the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

DEPARTMENT OF ARTIFICIAL INTELLIGENCE



KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI ERODE – 638 060

NOVEMBER 2024

DEPARTMENT OF ARTIFICIAL INTELLIGENCE
KONGU ENGINEERING COLLEGE
(Autonomous)
PERUNDURAI ERODE – 638 060
NOVEMBER 2024

BONAFIDE CERTIFICATE

This is to certify that the Project report entitled **PROJECT MANAGEMENT** is the bonafide record of the project work done by **MADHIUKSHA S (Register No: 22ALR045), AKSHARA R K (Register No: 22ALR001), MADHUMITHA S (Register No:22ALR047)**, in the partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in **Artificial Intelligence and Machine Learning** of Anna University Chennai during the year 2024-2025.

SUPERVISOR

HEAD OF THE DEPARTMENT

Date:

(Signature with seal)

Submitted for the end semester viva voce examination held on _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

KONGU ENGINEERING COLLEGE**(Autonomous)****PERUNDURAI ERODE – 638 060****NOVEMBER 2024****DECLARATION**

We affirm that the Project Report titled **PROJECT MANAGEMENT** being submitted in partial fulfillment of the requirements for the award of Bachelor of Technology is the original work carried out by us. It has not formed the part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion this or any other candidate.

Date:**MADHIUKSHA S
(22ALR045)****AKSHARA R K
(22AKR001)****MADHUMITHA S
(22ALR047)**

I certify that the declaration made by the above candidate is true to the best of my knowledge.

Date:**Name and Signature of the Supervisor**

ABSTRACT

Project Management project presents the development of a comprehensive project management system using the MERN (MongoDB, Express, React, Node.js) stack, aimed at enhancing task scheduling, collaboration, and visualization for project teams. The system allows users to create and manage projects, assign tasks, set deadlines, and track task status in real-time. It features dynamic role assignment, with users being assigned as creators or members depending on the project. A key component is the task breakdown functionality, where users can divide tasks into smaller, personal sub-tasks that are visible only to them. The task management system is integrated with visual dashboards displaying project status, task progress, and completion statistics using charts. A messaging system is incorporated for seamless communication among team members. The use of MongoDB for data storage ensures scalability, while React and Node.js enable a responsive and efficient user experience. This system simplifies project management for teams and individuals by providing real-time updates, clear task tracking, and a user-friendly interface.

ACKNOWLEDGEMENT

We express our sincere thanks and gratitude to **Thiru. A. K. ILANGO B.Com., M.B.A., LLB.**, our beloved Correspondent, and all other philanthropic trust members of Kongu Vellalar Institute of Technology Trust who have always encouraged us in academic and co-curricular activities.

We are extremely thankful with no words of a formal nature to the dynamic Principal **Dr. V. BALUSAMY, M.Tech., Ph.D.**, for providing the necessary facilities to complete our work.

We would like to express our sincere gratitude to our respected Head of the Department **Dr. C. S. KANIMOZHI SELVI, M.E., Ph.D.**, for providing the necessary facilities.

We extend our thanks to **Ms. P. RAMYA ., M.E.**, Assistant Professor, Artificial Intelligence, Project Coordinator for her encouragement and valuable advice that made us carry out the project work successfully.

We extend our gratitude to our Supervisor **Ms. S. BENIL JENIFFER, M.E.**, Assistant Professor of Artificial Intelligence, for her valuable ideas and suggestions, which have been very helpful in the project. We are grateful to all the faculty members of the Artificial Intelligence Department, for their support.

TABLE OF CONTENTS

CHAPTER No.	TITLE	PAGE No.
	ABSTRACT	IV
	LIST OF FIGURES	VIII
1.	INTRODUCTION	1
	1.1 EXISTING SYSTEM	1
	1.2 SYSTEM STUDY	1
	1.3 OBJECTIVE	2
	1.4 SCOPE	2
2.	GENERAL DESCRIPTION	3
	2.1 PROJECT PERSPECTIVE	3
	2.2 USER CHARACTERISTICS	3
	2.3 DESIGN AND IMPLEMENTATION	4
3.	REQUIREMENTS	5
	3.1 FUNCTIONAL REQUIREMENTS	5
	3.2 NON-FUNCTIONAL REQUIREMENTS	5
	3.3 USER INTERFACES	6
4.	DETAILED DESIGN	10
	4.1 ARCHITECTURAL DESIGN	10
	4.1.1 MODULE CHARACTERISTICS	10
	4.1.1.1 USER MODULE	10

4.1.1.2	USE CASE DIAGRAM	11
4.2	INTERFACE DESIGN	11
4.3	DATABASE DESIGN	12
4.4	OUTPUT DESIGN	12
5.	TESTING	13
5.1	UNIT TESTING	13
5.2	REGRESSION TESTING	13
5.3	VALIDATION TESTING	13
5.4	VERIFICATION TESTING	14
5.5	INTEGRATION TESTING	14
6.	CONCLUSION AND FUTURE SCOPE	15
	APPENDIX 1	16
	APPENDIX 2	49
	REFERENCES	51

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
4.1	FLOW CHART	10
4.2	USE CASE DIAGRAM	11
4.3	DATABASE DESIGN DIAGRAM	12

CHAPTER 1

INTRODUCTION

In a fast-paced world where team collaboration and productivity are essential, a structured project management application is crucial for managing tasks, assigning responsibilities, and tracking progress. The purpose of this application is to facilitate an organized approach to project management, allowing team heads and members to work cohesively toward shared goals. It is designed to support task scheduling, project tracking, and real-time collaboration, ensuring that all team members have clear, accessible information on their responsibilities and deadlines. By providing role-based access, easy task management, and seamless integration between project and schedule dashboards, this application addresses the complexities of managing both individual and collaborative tasks in a digital workspace.

1.1 EXISTING SYSTEM

Project and task management, while vital to many teams, often rely on outdated methods that lack centralization and efficiency. Here's a breakdown of the existing methods and their drawbacks:

- Spreadsheets and Emails:** Many organizations still use spreadsheets and emails for tracking tasks and projects. This approach leads to scattered data and limits real-time collaboration.
- Lack of Role-Based Permissions:** Existing systems may lack clear distinctions between team roles, leading to difficulties in accountability and permissions, which can affect team efficiency.
- Limited Task Scheduling:** Traditional methods often lack detailed scheduling capabilities, making it challenging to break down larger projects into smaller, manageable tasks.
- Low Visibility into Task Status:** Without a centralized dashboard, it becomes hard to monitor the status of various tasks and projects, which can delay decision-making.

This project management application seeks to address these gaps by offering a role-based, centralized, and fully digital workspace that includes task scheduling, status tracking, and prioritization.

1.2 SYSTEM STUDY

1.2.1 Dashboard Management:

Main Dashboard: Provides a quick overview of all tasks, completed and pending, with sort options by date and priority.

Project Dashboard: Designed for team heads to manage ongoing projects, assign tasks to team members, and track project progress.

1.2.2 User Roles and Permissions:

Team Heads: Can create, edit, delete, and assign projects, while having full control over task visibility and status updates.

Team Members: Can view tasks assigned to them, complete tasks, and break down assigned tasks into subtasks for better organization.

1.2.3 Real-Time Data Synchronization:

Updates task and project status across all relevant dashboards, ensuring that users have the latest information on task status.

1.2.4 Motivational Interface:

Quotes and motivational messages on the login page encourage users to engage more positively with their tasks.

1.2.5 Task Completion Metrics:

Enables users to track their progress, providing a sense of accomplishment and aiding in prioritization.

1.3 OBJECTIVE

Provide a streamlined interface for users to add, edit, delete, and complete tasks while tracking progress with clear visual indicators. Allow team heads to create projects, assign tasks, and track progress, while team members focus on individual responsibilities. Keep team members updated on task statuses, project changes, and deadlines, minimizing miscommunication and improving productivity. With motivational quotes, completion tracking, and user-friendly design, increase user engagement and productivity. Implement secure access and data handling practices to ensure that users' information and project data are protected.

1.4 SCOPE

This project targets small to medium-sized teams who require a reliable and collaborative project management solution. By focusing on task and project assignment, role-based access, and effective scheduling, the application supports complex team projects, where efficient collaboration and transparency are key to success. The application is adaptable and can be scaled to support larger teams and more intricate project workflows.

CHAPTER 2

GENERAL DESCRIPTION

2.1 PROJECT PERSPECTIVE

This web-based project management system leverages the MERN stack (MongoDB, Express.js, React.js, and Node.js) to ensure a responsive, scalable, and real-time user experience. Each user's actions—whether adding tasks, creating projects, or updating schedules—are managed with a centralized backend, ensuring that all users' dashboards reflect the latest project statuses. The MERN stack provides a robust foundation that integrates well with cloud-based solutions for scalability, allowing the system to grow as team needs evolve.

2.2 USER CHARACTERISTICS

2.2.1 Team Heads:

Create and assign projects. Set deadlines, track team members' progress, and adjust tasks as needed. Ability to edit, delete, and mark tasks as finished for the entire team.

2.2.2 Team Members:

View tasks assigned to them, mark tasks as read, and complete tasks based on priority and deadlines. Break down their tasks into subtasks for better management.

2.2.3 General Users:

Motivated with positive quotes on the login page to enhance engagement and start each session on a positive note. Can view and manage their individual tasks with sort options.

2.3 DESIGN AND IMPLEMENTATION

2.3.1 User Authentication & Role-Based Permissions:

Implementing secure login with role-based permissions to ensure users only access data relevant to their role.

2.3.2 Real-Time Updates and Synchronization:

Synchronizing data across dashboards in real-time for all users to avoid lag in task updates.

2.3.3 Scalability:

Designing the system architecture to handle potential growth in the number of projects, tasks, and users.

2.3.4 Data Security:

Encrypting sensitive data (e.g., user passwords) and ensuring database queries are secure to prevent unauthorized access.

2.3.5 Responsive UI/UX Design:

Ensuring a responsive design that functions well on desktops and mobile devices, enhancing accessibility for all users.

CHAPTER 3

REQUIREMENTS

3.1 FUNCTIONAL REQUIREMENTS

3.1.1 User Authentication:

Secure registration and login, with role-based access control that limits specific actions to team heads and members.

3.1.2 Dashboard Features:

Task Dashboard: Allows adding, editing, deleting, and marking tasks as completed, with options to sort by priority and date.

Project Dashboard: For team heads to create and assign projects, add team members via email, and manage project visibility in ongoing and finished project lists.

3.1.3 Project and Task Assignment:

Team heads can assign tasks to members by email, allowing for easy task management and tracking of task completion.

3.1.4 Scheduling and Subtasks:

Allows members to break down tasks into subtasks, adding more detail to complex assignments and ensuring thorough task management.

3.1.5 Real-Time Collaboration:

Offers in-app communication tools for project discussions, enhancing teamwork.

3.2 NON-FUNCTIONAL REQUIREMENTS

3.2.1 Performance:

The system should efficiently handle concurrent tasks and real-time updates, ensuring a smooth user experience without lag.

3.2.2 Scalability:

System architecture should support an increasing number of projects, tasks, and users without affecting performance.

3.2.3 Security:

Data encryption for sensitive data (e.g., passwords).

Implementation of access control mechanisms to limit data access based on user roles.

3.2.4 Reliability and Availability:

Ensure that the application has high uptime and consistent data backups to prevent data loss.

3.2.5 User-Friendliness:

A simple, intuitive interface that minimizes the learning curve, with motivational content to engage users.

3.2.6 Cross-Platform Compatibility:

Responsive design, ensuring compatibility with both desktop and mobile browsers.

3.3 USER INTERFACES

HOME PAGE:

Home Page link to various pages like Login, Register, About us, Contact

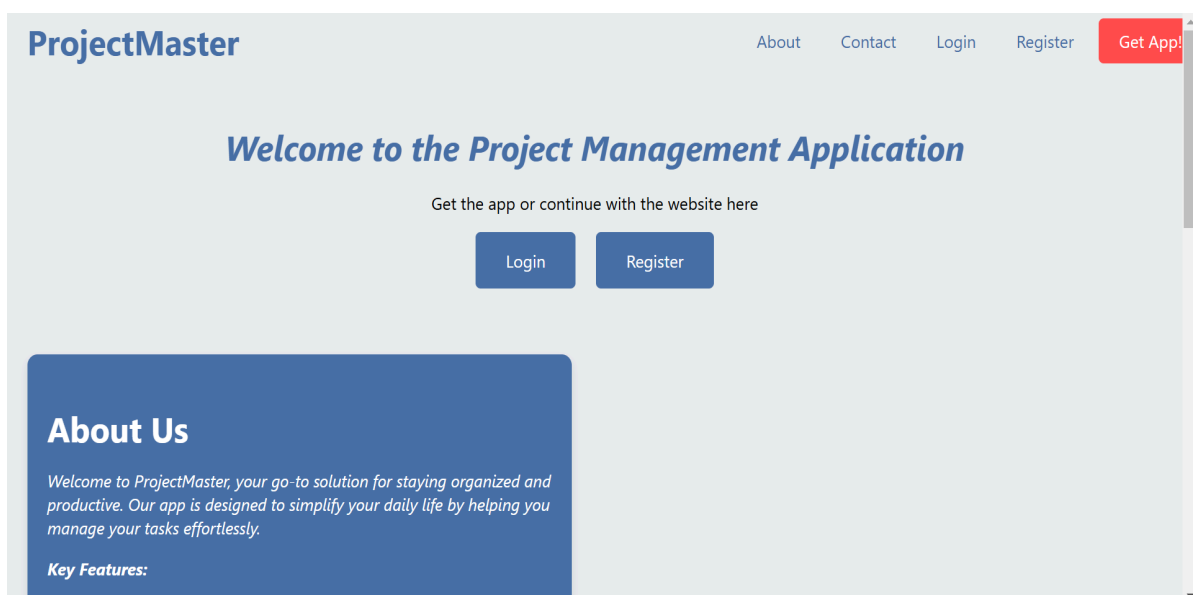
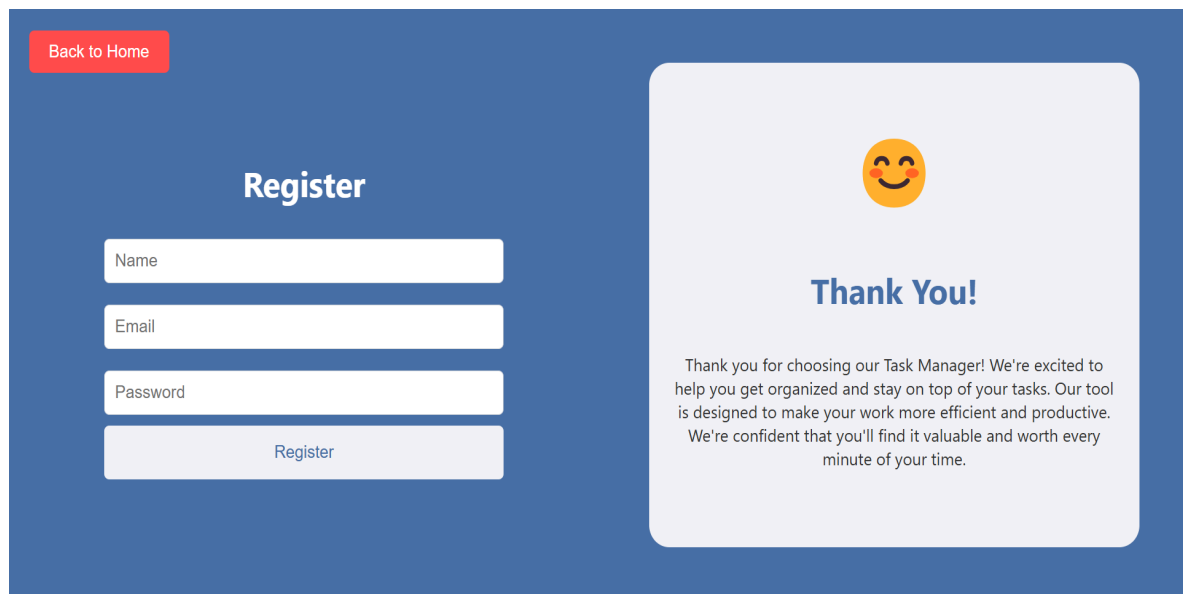


FIGURE 3.3.0

REGISTER PAGE:

Register Page contains field like Name,Email,Password

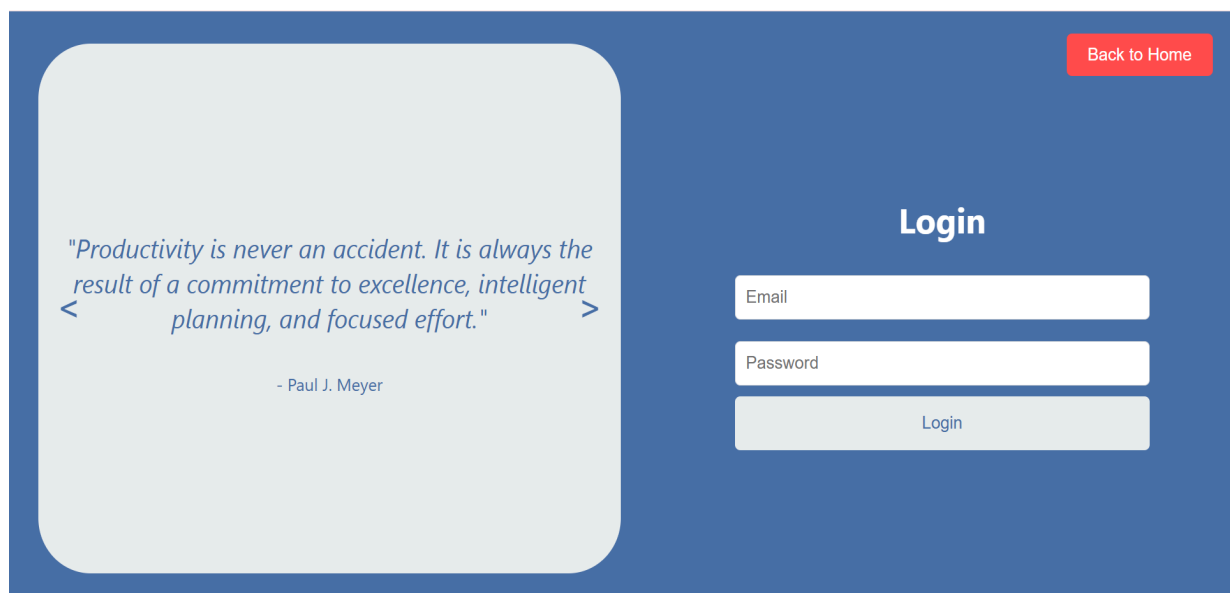


The image shows a web page with a dark blue background. In the top left corner, there is a red button labeled "Back to Home". The main content area is divided into two sections. On the left, under the heading "Register", there are three white input fields labeled "Name", "Email", and "Password", followed by a light blue button labeled "Register". On the right, there is a light blue rounded rectangle containing a yellow smiley face emoji, the text "Thank You!", and a paragraph of text: "Thank you for choosing our Task Manager! We're excited to help you get organized and stay on top of your tasks. Our tool is designed to make your work more efficient and productive. We're confident that you'll find it valuable and worth every minute of your time."

FIGURE 3.3.1

LOGIN PAGE:

Login page have field of email and password to move to dashboard page and also contains quotes



The image shows a web page with a dark blue background. In the top right corner, there is a red button labeled "Back to Home". The main content area is divided into two sections. On the left, there is a light blue rounded rectangle containing a quote: "Productivity is never an accident. It is always the result of a commitment to excellence, intelligent planning, and focused effort." attributed to "- Paul J. Meyer". On the right, under the heading "Login", there are two white input fields labeled "Email" and "Password", followed by a light blue button labeled "Login".

FIGURE 3.3.2

DASHBOARD PAGE :

In Dashboard page ,we can add edit delete mark as read the tasks

The screenshot displays a dashboard with three main sections:

- Add Your Tasks:** A form on the left for creating new tasks. It includes fields for 'Task Title', 'Task Description', a priority dropdown (set to 'Low'), and a date picker (set to 'dd-mm-yyyy'). An 'Add Task' button is at the bottom.
- Task List:** A central panel with sorting options ('Sort by Date', 'Sort by Priority'). It lists two tasks:
 - BDA PROJECT:** 'SUPERVISED AND UNSUPERVISED ALGORITHMS 2 USE AND DEPLOY', Priority: High, Date: 2024-11-14. Actions: Edit, Mark as Finished, Delete.
 - SAP:** 'ALL PROOFS 1)NPTEL 2)PSG 3)WDC'.
- Finished Tasks:** A right panel with a 'Project' button and a 'Logout' button. It lists:
 - DA REPORT:** 'DA PROJECT REPORT', Priority: High, Date: 2024-11-12. Actions: Reopen, Delete.
 - NLP VIVA:** 'YOGA MAM', Priority: High.

FIGURE 3.3.3

PROJECT DASHBOARD PAGE:

In project dashboard , we can add delete,edit assign projects

The screenshot displays a project dashboard with three main sections:

- Add New Project:** A form on the left for creating new projects. It includes fields for 'Project Title', 'Project Description', a priority dropdown (set to 'Medium'), a date picker (set to 'dd-mm-yyyy'), and a text field for 'Enter team members' emails, separated by commas'. An 'Add Project' button is at the bottom.
- Ongoing Projects:** A central panel showing project details. It lists:
 - FULL STACK:** 'PROJECT MANAGEMENT', Priority: High, Deadline: 11/13/2024. Actions: Edit, Finished, Delete.
 - Team Head:** madhumithas.22aim@konqu.edu
- Finished Projects:** A right panel showing project details. It lists:
 - DATA ANALYSIS:** 'OTT MEDIA ANALYSIS', Priority: Medium, Deadline: 11/16/2024, Status: Not Started. Actions: Delete, Reopen.

FIGURE 3.3.5

SCHEDULE PROJECT PAGE:

In schedule project page assign task for each team member and further subtask them

Task Schedule for Project

Task Title

Task Description

dd-mm-yyyy

Assignee Email

Add Task

Backend

route connection

Due: 11/12/2024

Assigned to: Keerthana

Mark as Completed Delete Task

FIGURE 3.3.6

DATABASE STORAGE:

In Database,user ,project,task,schedule are stored

Collection	Storage size	Documents	Avg. document size	Indexes	Total index size
schedules	20.48 kB	6	171.00 B	1	36.86 kB
subtasks	20.48 kB	3	130.00 B	1	36.86 kB
tasks	20.48 kB	19	148.00 B	1	36.86 kB
users					

FIGURE 3.3.7

CHAPTER 4

DETAILED DESIGN

4.1 ARCHITECTURAL DESIGN

The architectural design includes various diagrams such as use case diagram, sequence diagram and activity diagram for user module (any one can be team head and team members for each project).

4.1.1 MODULE CHARACTERISTIC

The project contains single modules as follows and their description is given below.

User module

4.1.1.1 User Module

The user module contains a home page, Login Page, Register page, dashboard page, projects page, Schedule page. The user can view all the pages as mentioned in the user module flow FIGURE 4.1

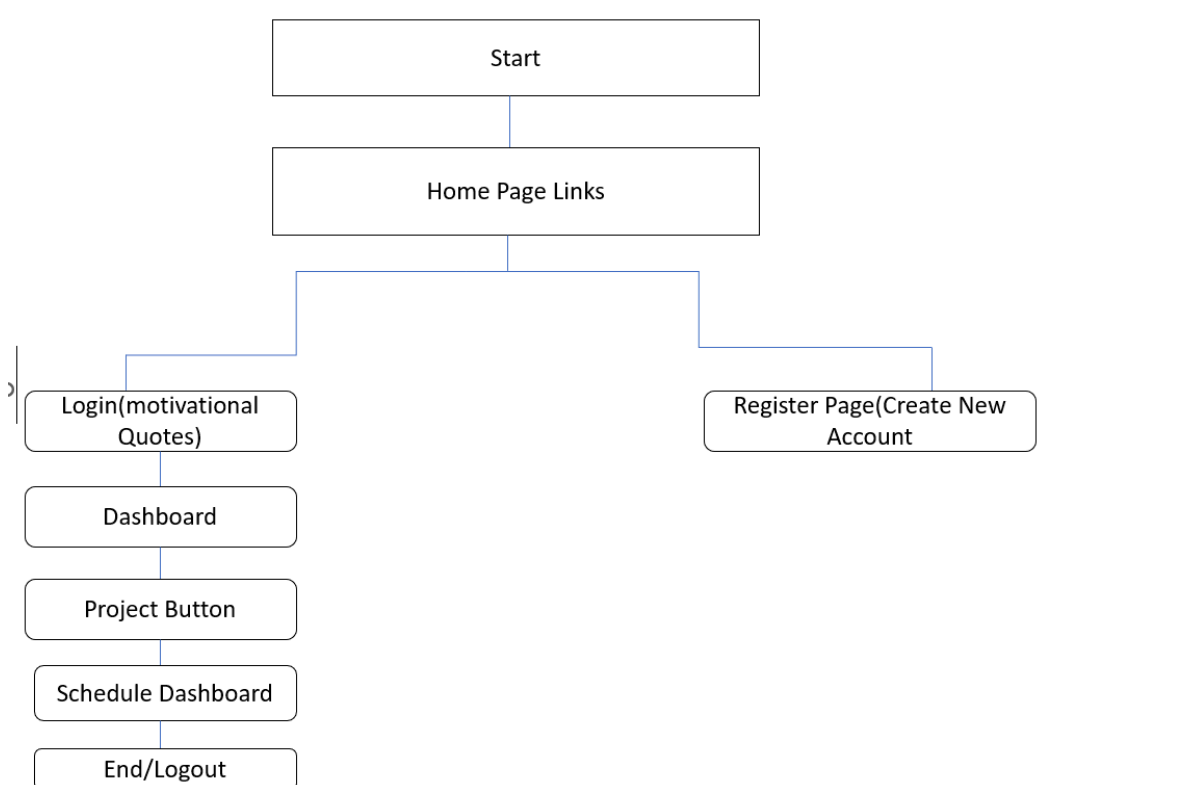


FIGURE 4.1 FLOWCHART

4.1.1.2 USE CASE DIAGRAM

Use case diagrams model the functionality of a system using actors and use cases. Use cases are set of actions, services, and functions that the system needs to perform. A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. The use cases are represented by either circles or ellipses. Due to their simplistic nature, use case diagrams can be a good for understanding project.

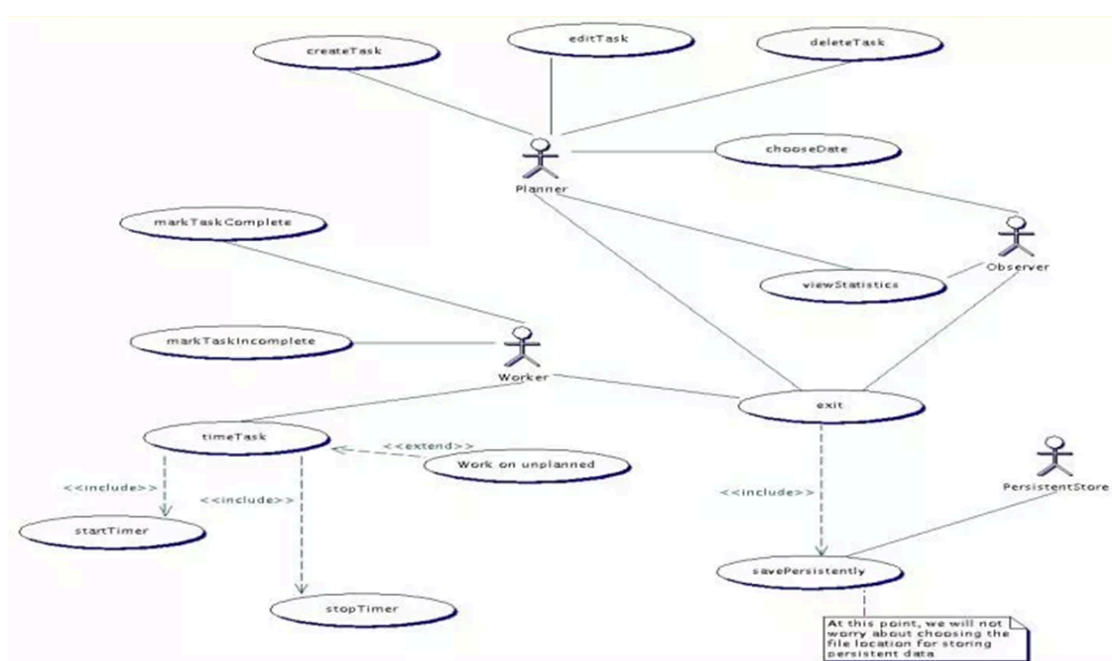


FIGURE 4.2 USE CASE DIAGRAM

4.2 INTERFACE DESIGN

Login Includes fields for username and password, along with a motivational message to boost user engagement. Includes forgotten password options and clear feedback for authentication errors. Shows a comprehensive view of all active tasks and projects, sortable by priority and deadline. Quick-access buttons provide easy navigation for task creation and project management. Provides team heads with management tools for project assignment and progress tracking. Includes visual aids such as progress bars, task lists, and team member statuses. Allows users to view specific task details, update progress, and mark tasks as completed. Enables users to adjust notification preferences, update passwords, and manage personal information.

4.3 DATABASE DESIGN

Stores user information (e.g., name, role, email, password) with encrypted passwords and access control flags. Contains project details such as project name, description, start/end dates, assigned team members, and project status. Manages individual tasks, each with fields for task name, project ID, assigned user ID, priority, due date, completion status, and any subtasks. Notification Collection: Logs notifications for task updates, assignments, and other alerts, linked to users and tasks.

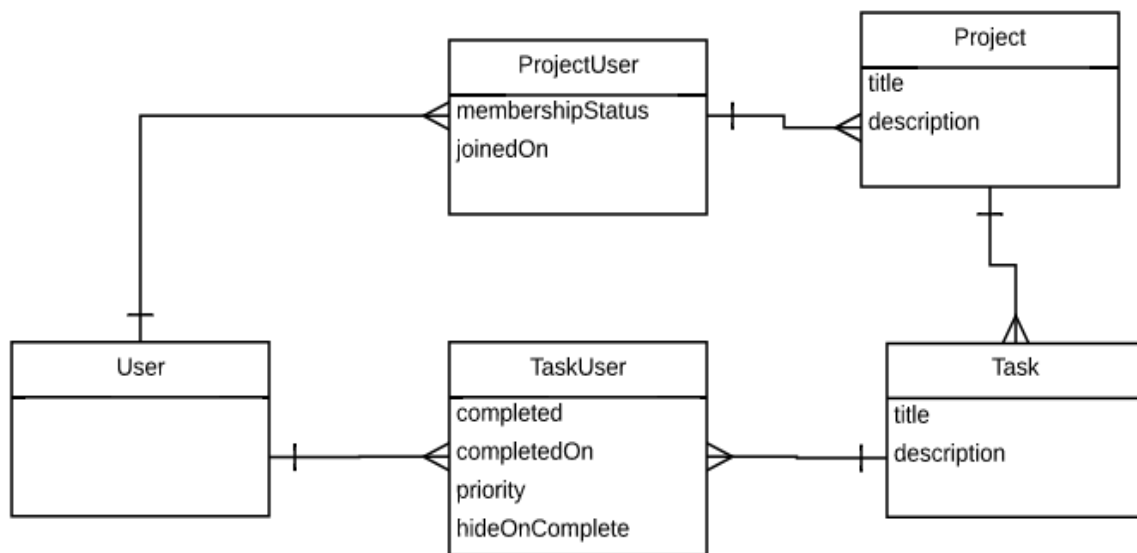


FIGURE 4.3 DATABASE DESIGN DIAGRAM

4.4 OUTPUT DESIGN

Displays a user-friendly report of tasks completed, showing metrics like the percentage of tasks completed, overdue tasks, and upcoming deadlines. Provides visual output of project timelines, completion percentages, and individual team members' contributions. Shows motivational feedback based on completed tasks, deadlines met, and project milestones achieved.

CHAPTER 5

TESTING

5.1 UNIT TESTING

Unit testing is conducted on the smallest individual components to ensure they function as intended in isolation. Key modules for testing include. Verify that login and registration work securely, with accurate role-based access control. Dashboard Functionalities: Check if tasks are displayed, sorted, and filtered correctly on the dashboard for both team heads and team members. Task Assignment: Test the assignment functionality to ensure that tasks are allocated correctly based on user roles, and that assignments are communicated accurately to respective team members. Role-Based Permissions: Confirm that only team heads can create, edit, and assign tasks while team members can only view and update tasks assigned to them. Unit tests are automated where possible to ensure efficient, repeatable testing and quick feedback on each component's reliability. This phase of testing helps detect any errors early in the development process, ensuring each unit's robustness before moving on to integration.

5.2 REGRESSION TESTING

Regression testing is essential whenever updates are made to the application. This process confirms that new code does not interfere with the functionality of existing components. After implementing updates or enhancements, the regression tests are run to verify that: Dashboard Views remain unaffected by recent changes. User Permissions are still accurately enforced following updates to the role-based access system. Task and Project Management Functionalities operate without disruption, maintaining the application's reliability. This approach helps maintain stability and ensures that each update aligns smoothly with the rest of the application.

5.3 VALIDATION TESTING

Validation testing focuses on verifying that the application behaves as expected according to the user's input. Specific validations include Ensuring data inputs, such as email formats,

deadlines, and priority levels, meet predefined criteria to prevent errors. Checking that actions like task creation, editing, and deletion reflect accurately across all dashboards. Ensuring that feedback (e.g., success or error messages) appears for user actions, making interactions intuitive and informative. This phase confirms that each user action aligns with the expected outcome, supporting a smooth and reliable user experience.

5.4 VERIFICATION TESTING

Verification testing ensures that the project meets all specified requirements and functions as expected. This involves:

- Requirement Fulfillment:** Confirming that all outlined requirements, such as role-based access, real-time synchronization, and secure data handling, are met.
- System Integrity:** Ensuring the application's backend operations, such as data storage and retrieval, align with the front-end functionality.
- Real-Time Updates:** Verifying that changes made by team heads (like task assignments) are reflected in real-time for team members.

Verification testing guarantees that the application complies with the design specifications and delivers the intended functionality.

5.5 INTEGRATION TESTING

Integration testing evaluates the interactions between different components to verify they work cohesively. Areas of focus include:

- Data Flow Between Database and Dashboard:** Ensuring smooth data retrieval and storage processes, with real-time updates reflected on the dashboard.
- Synchronization Across User Views:** Confirming that task assignments, updates, and completions are synchronized between team member and team head views.
- Task Assignment Workflow:** Testing that tasks assigned by team heads are properly displayed and actionable by team members.

By testing these integrations, the development team can identify and resolve any issues related to data flow and inter-component compatibility, contributing to a unified and functional application.

CHAPTER 6

CONCLUSION AND FUTURE WORK

This project management application effectively meets the growing need for an intuitive, digital solution that facilitates collaboration, task management, and real-time project tracking. Built using the MERN stack, the application offers a scalable and responsive user interface, making it suitable for small to medium-sized teams that seek enhanced productivity and transparency in their projects. Extending the application to mobile platforms to allow users to manage tasks, view dashboards, and track projects on-the-go, improving accessibility. Implementing advanced analytics tools to offer insights on project progress, team productivity, and task completion rates, empowering team heads to make data-driven decisions. Utilizing machine learning to analyze task dependencies, deadlines, and team workloads to suggest task priorities and enhance productivity. Providing seamless synchronization with popular calendar applications like Google Calendar or Outlook, allowing users to schedule tasks and project deadlines more effectively within their preferred calendar system. Adding functionality for tracking task dependencies, helping teams manage complex projects by visualizing the relationships between tasks.

APPENDIX 1

CODING

Backend:

Server.js:

```
const express = require('express');
const dotenv = require('dotenv');
const connectDB = require('./config/db');
const userRoutes = require('./routes/userRoutes');
const taskRoutes = require('./routes/taskRoutes');
const cors = require('cors');
const projectRoutes = require('./routes/projectRoutes');
const scheduleRoutes = require('./routes/scheduleRoutes');
const subtaskRoutes = require('./routes/subtaskRouter');
dotenv.config();
connectDB();
const app = express();
app.use(express.json());
app.use(cors());
app.use('/api/users', userRoutes);
app.use('/api/tasks', taskRoutes); // Correctly use taskRoutes
app.use('/api', projectRoutes);
app.use('/api/schedules', scheduleRoutes);
app.use('/api', subtaskRoutes);
const PORT = process.env.PORT || 5000;
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

projectRoutes.js:

```
const express = require('express');
const router = express.Router();
const authMiddleware = require('../middleware/authMiddleware'); // Ensure the
path is correct
const User = require('../models/User');
```

```

const Project = require('../models/Project');
const jwt = require('jsonwebtoken');
router.post('/projects', authMiddleware, async (req, res) => {
  console.log("POST /projects request received");
  const { title, description, priority, deadline, teamMembersEmails } = req.body; // Fields
  coming from frontend
  if (!title || !description || !priority || !deadline || !teamMembersEmails) {
    return res.status(400).json({ message: 'Missing required fields' });
  }
  try {
    const teamMembers = await User.find({
      email: { $in: teamMembersEmails }
    }).select('_id');
    if (teamMembers.length !== teamMembersEmails.length) {
      return res.status(404).json({ message: 'One or more email addresses not found' });
    }
    const newProject = new Project({
      title,
      description,
      priority,
      deadline,
      teamMembers,
      createdBy: req.user.id,
      status: 'Not Started', // Ensure that the project is created with "Not Started" status
    });
    await newProject.save();
    const populatedProject = await Project.findById(newProject._id)
      .populate('createdBy', 'email')
      .populate('teamMembers', 'email')
      .exec();
    res.json(populatedProject); // Return the newly created and populated project
  } catch (error) {
    res.status(500).json({ message: 'Error adding project', error: error.message });
  }
});

```

```

    }
  });
  router.get('/projects', authMiddleware, async (req, res) => {
    const { page = 1, limit = 10 } = req.query;
    const userId = req.user.id;
    try {
      18
      const projects = await Project.find({
        $or: [
          { createdBy: userId }, // Projects where the user is the creator
          { teamMembers: userId } // Projects where the user is a team member
        ]
      })
        .populate('createdBy', 'email')
        .populate('teamMembers', 'email')
        .skip((page - 1) * limit)
        .limit(parseInt(limit));
      const totalProjects = await Project.countDocuments({
        $or: [
          { createdBy: userId },
          { teamMembers: userId }
        ]
      });
      res.json({ projects, totalPages: Math.ceil(totalProjects / limit), currentPage: page });
    } catch (error) {
      res.status(500).json({ message: 'Error fetching projects', error: error.message });
    }
  });
  router.put('/projects/:id', authMiddleware, async (req, res) => {
    try {
      const { title, description, deadline, teamMembers } = req.body;
      const project = await Project.findOneAndUpdate(
        { _id: req.params.id, createdBy: req.user.id },

```

```

    { title, description, deadline, teamMembers },
    { new: true }
  );
  if (!project) return res.status(404).json({ message: 'Project not found or you do not have
  permission' });
  res.json(project);
} catch (error) {
  res.status(500).json({ message: error.message });
}
});
//19

router.put('/projects/:id/reopen', authMiddleware, async (req, res) => {
  try {
    const project = await Project.findById(req.params.id);
    if (!project) {
      return res.status(404).json({ message: 'Project not found' });
    }
    project.isCompleted = false;
    await project.save();
    const updatedProject = await Project.findById(req.params.id)
      .populate('createdBy', 'email')
      .populate('teamMembers', 'email');
    res.json(updatedProject);
  } catch (error) {
    res.status(500).json({ message: 'Failed to reopen project' });
  }
});

router.put('/projects/:id/status', authMiddleware, async (req, res) => {
  const { status } = req.body;
  try {
    const project = await Project.findOneAndUpdate(
      { _id: req.params.id, teamMembers: req.user.id }, // Ensure only team members can update
      the status

```

```

    { status },
    { new: true }
  );
  if (!project) {
    return res.status(404).json({ message: 'Project not found or you do not have permission' });
  }
  res.json({ message: 'Status updated successfully', project });
} catch (error) {
  res.status(500).json({ message: 'Error updating status', error: error.message });
}
});

router.put('/projects/:id/mark-as-finished', authMiddleware, async (req, res) => {
  try {
    const project = await Project.findById(req.params.id);
    if (!project) {
      //20
      return res.status(404).json({ message: 'Project not found' });
    }
    if (project.createdBy.toString() !== req.user.id.toString()) {
      return res.status(403).json({ message: 'You are not the creator of this project' });
    }
    // Update the project to mark it as finished
    project.isCompleted = true; // Make sure this is set
    // Update the status to "Finished"
    await project.save();
    // Return the updated project with populated fields
    const updatedProject = await Project.findById(req.params.id)
      .populate('createdBy', 'email')
      .populate('teamMembers', 'email');
    res.json({ message: 'Project marked as finished', project: updatedProject });
  } catch (error) {
    res.status(500).json({ message: 'Error marking project as finished' });
  }
}

```

```

});
// Delete project
router.delete('/projects/:id', authMiddleware, async (req, res) => {
  try {
    // Only allow the project creator to delete the project
    const project = await Project.findOneAndDelete({ _id: req.params.id, createdBy:
    req.user.id });
    if (!project) return res.status(404).json({ message: 'Project not found or you do not have
    permission' });
    res.json({ message: 'Project deleted successfully' });
  } catch (error) {
    res.status(500).json({ message: 'Error deleting project' });
  }
});
module.exports = router;

```

taskRoutes.js

```

const express = require('express');
const router = express.Router();
//21
const authMiddleware = require('../middleware/authMiddleware'); // Ensure the path is
correct
const Task = require('../models/Task');
// Create a task
router.post('/', authMiddleware, async (req, res) => {
  const { title, description, priority, date } = req.body;
  try {
    const newTask = new Task({
      user: req.user.id,
      title,
      description,
      Priority,

```

```

    date,
  });
  const task = await newTask.save();
  res.status(201).json(task);
} catch (error) {
  res.status(500).json({ message: error.message });
}
});
// Get all tasks for the logged-in user
router.get('/', authMiddleware, async (req, res) => {
  try {
    const tasks = await Task.find({ user: req.user.id });
    res.json(tasks);
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
});
router.put('/:id', authMiddleware, async (req, res) => {
  try {
    const { title, description, priority, date, completed } = req.body;
    const task = await Task.findOneAndUpdate(
      { _id: req.params.id, user: req.user.id },
      { title, description, priority, date, completed },
      { new: true }
    );
    //22
    if (!task) return res.status(404).json({ message: 'Task not found' });
    res.json(task);
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
});
// Route to schedule a task

```

```

router.put('/:id/schedule', async (req, res) => {
  try {
    const { scheduleDate, scheduleTime } = req.body;
    const task = await Task.findById(req.params.id);
    if (!task) return res.status(404).send('Task not found');
    task.scheduleDate = scheduleDate;
    task.scheduleTime = scheduleTime;
    await task.save();
    res.send(task);
  } catch (error) {
    res.status(500).send(error.message);
  }
});

// Delete a task by ID
router.delete('/:id', authMiddleware, async (req, res) => {
  try {
    const task = await Task.findOneAndDelete({ _id: req.params.id, user: req.user.id });
    if (!task) {
      return res.status(404).json({ message: 'Task not found' });
    }
    res.json({ message: 'Task deleted successfully' });
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
});

module.exports = router;

```

userRoutes.js

```

const express = require('express');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const User = require('../models/User');

```



```

const router = express.Router();
// Register
router.post('/register', async (req, res) => {
  const { name, email, password } = req.body;
  try {
    let user = await User.findOne({ email });
    if (user) return res.status(400).json({ message: 'User already exists' });
    const hashedPassword = await bcrypt.hash(password, 10);
    user = new User({ name, email, password: hashedPassword });
    await user.save();
    res.status(201).json({ message: 'User registered successfully' });
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
});
// Login
router.post('/login', async (req, res) => {
  const { email, password } = req.body;
  try {
    const user = await User.findOne({ email });
    if (!user) return res.status(400).json({ message: 'Invalid credentials' });
    const isMatch = await bcrypt.compare(password, user.password);
    if (!isMatch) return res.status(400).json({ message: 'Invalid credentials' });
    const token = jwt.sign({ userId: user._id }, process.env.JWT_SECRET, { expiresIn: '1h' });
    res.json({ token });
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
});
module.exports = router;

```

Project.js:

```
const mongoose = require('mongoose');
const projectSchema = new mongoose.Schema({
  title: {
    type: String,
    required: true,
  },
  description: {
    type: String,
    required: true,
  },
  priority: {
    type: String,
    enum: ['Low', 'Medium', 'High'],
    default: 'Medium',
  },
  deadline: {
    type: Date,
    required: true,
  },
  teamMembers: [{
    type: mongoose.Schema.Types.ObjectId,
    ref: 'User',
    required: true,
  }], // Array of user references (team members)
  createdBy: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'User',
    required: true,
  }, // User who created the project
  status: { type: String, default: 'Not Started' },
  isCompleted: { type: Boolean, default: false },
});
module.exports = mongoose.model('Project', projectSchema);
```

Task.js:

```
const mongoose = require('mongoose');
25
const taskSchema = new mongoose.Schema({
  user: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },
  title: { type: String, required: true },
  description: { type: String },
  priority: { type: String, enum: ['Low', 'Medium', 'High'], default: 'Low' },
  date: {
    type: Date, // Make sure to use Date type here
    required: true,
  },
  completed: { type: Boolean, default: false },
});
module.exports = mongoose.model('Task', taskSchema);
```

User.js:

```
const mongoose = require('mongoose');
const userSchema = new mongoose.Schema({
  name: { type: String, required: true },
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true },
});
module.exports = mongoose.model('User', userSchema);
```

Frontend:**App.js:**

```
import React from 'react';
import { BrowserRouter, RouterProvider } from 'react-router-dom';
import { AuthProvider } from '../context/AuthContext';
import Home from '../components/Home';
import Register from '../components/Register';
import Login from '../components/Login';
```

```

import Dashboard from './components/Dashboard';
import ProjectDashboard from './components/ProjectDashboard';
import ScheduleProjects from './components/ScheduleProjects';
// Define your routes using createBrowserRouter
const router = createBrowserRouter(
[
26
  { path: "/", element: <Home /> },
  { path: "/register", element: <Register /> },
  { path: "/login", element: <Login /> },
  { path: "/dashboard", element: <Dashboard /> },
  { path: "/project-dashboard", element: <ProjectDashboard /> },
  { path: "/schedule/:projectId", element: <ScheduleProjects /> },
],
{
  future: {
    v7_startTransition: true, // Opt-in to startTransition behavior
    v7_relativeSplatPath: true, // Opt-in to relative splat path behavior
    v7_fetcherPersist: true, // Opt-in to fetcher persistence behavior
    v7_normalizeFormMethod: true, // Opt-in to normalize formMethod casing
    v7_partialHydration: true, // Opt-in to partial hydration behavior
    v7_skipActionErrorRevalidation: true,
    // Opt-in to skip action error revalidation behavior
  },
}
);
function App() {
  return (
    <AuthProvider>
    <RouterProvider router={router} /> { /* Use RouterProvider to provide the router */ }
    </AuthProvider>
  );
}
export default App;

```

Dashboard.js:

```

import React, { useState, useEffect } from 'react';
import axios from 'axios';
import { useAuth } from '../context/AuthContext';
import { useNavigate } from 'react-router-dom';
const Dashboard = () => {
  const { user, logout } = useAuth();
  const navigate = useNavigate();
  27
  const [task, setTask] = useState({
    title: "",
    description: "",
    priority: 'Low',
    date: "",
    status: 'Pending',
  });
  const [tasks, setTasks] = useState([]);
  const [finishedTasks, setFinishedTasks] = useState([]);
  const [editingTask, setEditingTask] = useState(null);
  const [isEditing, setIsEditing] = useState(false);
  const [sortCriterion, setSortCriterion] = useState('priority');
  const [animationClass, setAnimationClass] = useState('fade-in');
  useEffect(() => {
    const fetchTasks = async () => {
      try {
        if (!user || !user.token) return;
        const token = user.token;
        const config = {
          headers: {
            Authorization: `Bearer ${token}`,
          },
        };
      };
      const response = await axios.get('/api/tasks', config);

```

```

const allTasks = response.data;
setTasks(allTasks.filter(t => !t.completed));
setFinishedTasks(allTasks.filter(t => t.completed));
} catch (error) {
if (error.response?.status === 401) {
console.error('Unauthorized! Logging out...');
logout();
navigate('/login');
} else {
console.error('Error fetching tasks:', error.response?.data || error.message);
}
}
28
};
fetchTasks();
// Trigger the fade-in animation when the component mounts
setAnimationClass('fade-in');
}, [user, logout, navigate]);
const handleLogout = () => {
logout();
navigate('/');
};
const handleChange = (e) => {
setTask({ ...task, [e.target.name]: e.target.value });
};
const addTask = async () => {
try {
const token = user?.token;
if (!token) throw new Error('No token available');
const config = {
headers: {
Authorization: `Bearer ${token}`,
},
},

```

```

};

const response = await axios.post('/api/tasks', task, config);
setTasks([...tasks, response.data]);
} catch (error) {
  console.error('Error adding task:', error.response?.data || error.message);
  alert(`Failed to add task. Error: ${error.response?.data?.message || error.message}`);
}
};

const onSubmit = (e) => {
  e.preventDefault();
  if (isEditing) {
    updateTask();
  } else {
    addTask();
  }
  setTask({
    title: "",
    description: "",
    priority: 'Low',
    date: "",
    status: 'Pending',
  });
  setIsEditing(false);
  setEditingTask(null);
};

const editTask = (task) => {
  setTask(task);
  setEditingTask(task);
  setIsEditing(true);
};

const updateTask = async () => {
  try {

```

```

const token = user?.token;
if (!token) throw new Error('No token available');
const config = {
  headers: {
    Authorization: `Bearer ${token}`,
  },
};
const response = await axios.put(`/api/tasks/${editingTask._id}`, task, config);
setTasks(tasks.map(t => (t._id === editingTask._id ? response.data : t)));
if (response.data.status === 'Finished') {
  setFinishedTasks([...finishedTasks, response.data]);
} else {
  setFinishedTasks(finishedTasks.filter(t => t._id !== editingTask._id));
}
} catch (error) {
  console.error('Error updating task:', error.response?.data || error.message);
  alert('Failed to update task. Error: ${error.response?.data?.message || error.message}');
}
};

const markAsFinished = async (id) => {
30
  try {
    const token = user?.token;
    if (!token) throw new Error('No token available');
    const config = {
      headers: {
        Authorization: `Bearer ${token}`,
      },
    };
    const updatedTask = tasks.find(t => t._id === id);
    if (!updatedTask) return;
    updatedTask.completed = true;
    const response = await axios.put(`/api/tasks/${id}`, updatedTask, config);

```



```

setTasks(tasks.filter(t => t._id !== id));
setFinishedTasks([...finishedTasks, response.data]);
} catch (error) {
  console.error('Error marking task as finished:', error.response?.data || error.message);
  alert(`Failed to mark task as finished. Error: ${error.response?.data?.message ||
error.message}`);
}
};

const reopenTask = async (id) => {
  try {
    const token = user?.token;
    if (!token) throw new Error('No token available');
    const config = {
      headers: {
        Authorization: `Bearer ${token}`,
      },
    };
    const updatedTask = finishedTasks.find(t => t._id === id);
    if (!updatedTask) return;
    updatedTask.completed = false;
    const response = await axios.put(`/api/tasks/${id}`, updatedTask, config);
    setFinishedTasks(finishedTasks.filter(t => t._id !== id));
    setTasks([...tasks, response.data]);
  } catch (error) {
    console.error('Error reopening task:', error.response?.data || error.message);
31
    alert(`Failed to reopen task. Error: ${error.response?.data?.message || error.message}`);
  }
};

const deleteTask = async (id) => {
  try {
    const token = user?.token;
    if (!token) throw new Error('No token available');

```

```

const config = {
  headers: {
    Authorization: `Bearer ${token}`,
  },
};

await axios.delete(`/api/tasks/${id}`, config);
setTasks(tasks.filter((task) => task._id !== id));
setFinishedTasks(finishedTasks.filter((task) => task._id !== id));
} catch (error) {
  console.error('Error deleting task:', error.response?.data || error.message);
  alert(`Failed to delete task. Error: ${error.response?.data?.message || error.message}`);
}
};

const handleProjectClick = () => {
  navigate('/project-dashboard');
};

const sortedTasks = [...tasks].sort((a, b) => {
  if (sortCriterion === 'priority') {
    return a.priority.localeCompare(b.priority);
  } else if (sortCriterion === 'date') {
    return new Date(a.date) - new Date(b.date);
  }
});

return 0;
});

const formatDateForInput = (date) => {
  if (!date) return '';
  const formattedDate = new Date(date).toISOString().split('T')[0]; // Convert to
  'YYYY-MM-DD'
  return formattedDate;
};

32

return (
  <div style={styles.container} className={animationClass}><style>

```

```

{
  @keyframes slideUp {
    from {
      opacity: 0;
      transform: translateY(100%);
    }
    to {
      opacity: 1;
      transform: translateY(0);
    }
  }
  .slide-up {
    animation: slideUp 0.7s ease-in-out;
  }
}
`
</style>
<div style={styles.addTaskSection}>
  <header style={styles.header}>
    <h1 style={styles.heading}>Add Your Tasks</h1>
  </header>
  <form onSubmit={onSubmit} style={styles.form}>
    <input
      type="text"
      name="title"
      placeholder="Task Title"
      value={task.title}
      onChange={handleChange}
      required
      style={styles.input}
    />
    <textarea
      name="description"
      placeholder="Task Description"

```

```

value={task.description}
//33
onChange={handleChange}
required
style={styles.textarea}
/>
<select
name="priority"
value={task.priority}
onChange={handleChange}
required
style={styles.select}
>
<option value="Low">Low</option>
<option value="Medium">Medium</option>
<option value="High">High</option>
</select>
<input
type="date"
name="date"
value={formatDateForInput(task.date)}
onChange={handleChange}
required
style={styles.input}
/>
<button type="submit" style={styles.button}>{isEditing ? 'Update Task' : 'Add
Task'}</button>
</form>
</div>
<div style={styles.taskContainer}>
<div style={styles.taskListSection}>
<h2 style={styles.subHeading}>Task List</h2>
<div style={styles.sortButtonsContainer}>

```

```

<button onClick={() => setSortCriterion('date')} style={styles.sortButton}>Sort by
Date</button>
<button onClick={() => setSortCriterion('priority')} style={styles.sortButton}>Sort by
Priority</button>
</div>
{tasks.length === 0 ? (
<p style={styles.noTasksMessage}>No tasks yet</p>
34
) : (
<ul style={styles.taskList}>
  {sortedTasks.map((task) => (
    <li key={task._id} style={styles.taskItem}>
      <h3 style={styles.taskTitle}>{task.title}</h3>
      <p style={styles.taskDescription}>{task.description}</p>
      <p style={styles.taskPriority}><strong>Priority:</strong> {task.priority}</p>
      <p style={styles.taskDate}><strong>Date:</strong> {formatDateForInput(task.date)}</p>
      <div style={styles.buttonGroup}>
        <button onClick={() => editTask(task)} style={styles.editButton}>Edit</button>
        <button onClick={() => markAsFinished(task._id)} style={styles.finishButton}>Mark as
        Finished</button>
        <button onClick={() => deleteTask(task._id)}
        style={styles.deleteButton}>Delete</button>
      </div>
    </li>
  ))}
</ul>
)}
</div>
<div style={styles.finishedTasksSection}>
  <header style={styles.finishedTasksHeader}>
    <h2 style={styles.subHeading}>Finished Tasks</h2>
    <button onClick={handleProjectClick} style={styles.projectButton}>
    Project

```

```

</button>
<button onClick={handleLogout} style={styles.logoutButton}>Logout</button>
</header>
{finishedTasks.length === 0 ? (
  <p style={styles.noTasksMessage}>No finished tasks yet</p>
) : (
  <ul style={styles.taskList}>
    {finishedTasks.map((task) => (
      <li key={task._id} style={styles.taskItem}>
        <h3 style={styles.taskTitle}>{task.title}</h3>
        <p style={styles.taskDescription}>{task.description}</p>
        <p style={styles.taskPriority}><strong>Priority:</strong> {task.priority}</p>
35
        <p style={styles.taskDate}><strong>Date:</strong> {formatDateForInput(task.date)}</p>
        <div style={styles.buttonGroup}>
          <button onClick={() => reopenTask(task._id)}
            style={styles.reopenButton}>Reopen</button>
          <button onClick={() => deleteTask(task._id)}
            style={styles.deleteButton}>Delete</button>
        </div>
      </li>
    ))}
  </ul>
)}
</div>
</div>
</div>
);
};
export default Dashboard;

```

Home.js:

```

import React, { useEffect } from 'react';
import { Link } from 'react-router-dom';

const Home = () => {
  useEffect(() => {
    const heroSection = document.querySelector('#hero-section');
    const aboutSection = document.querySelector('#about');
    const appSection = document.querySelector('#get-the-app');
    const contactSection = document.querySelector('#contact');

    const fadeIn = (element) => {
      element.style.opacity = 1;
      element.style.transform = 'translateY(0)';
    };

    fadeIn(heroSection);

    const observer = new IntersectionObserver((entries) => {
      entries.forEach(entry => {
        if (entry.isIntersecting) {
          fadeIn(entry.target);
        }
      });
    });

    observer.observe(aboutSection);
    observer.observe(appSection);
    observer.observe(contactSection);

    return (
      <div style={styles.container}>
        <div className="background-animation"></div> {/* Animated Background */}
        <header style={styles.header}>
          <h1 style={styles.title}>ProjectMaster</h1>
          <div style={styles.headerLinks}>
            <a href="#about" style={styles.headerLink}>About</a>

```

```

<a href="#contact" style={styles.headerLink}>Contact</a>
<Link to="/login" style={styles.headerLink}>Login</Link>
<Link to="/register" style={styles.headerLink}>Register</Link>
<Link to="/download" style={styles.getAppButton}>Get App!</Link>
</div>
</header>
<div id="hero-section" style={styles.heroSection}>
<h1 style={styles.heading}>Welcome to the Project Management Application</h1>
<p>Get the app or continue with the website here</p>
<div style={styles.buttonContainer}>
<Link to="/login" style={styles.button}>Login</Link>
<Link to="/register" style={styles.button}>Register</Link>
</div>
</div>
{/* Layout Sections */}
<div style={styles.mainContent}>
{/* About Us Section */}
<div id="about" style={styles.leftColumn}>
<div style={styles.section}>
<h2 style={styles.sectionHeading}>About Us</h2>
<p style={styles.sectionContent}>
Welcome to ProjectMaster, your go-to solution for staying organized and productive. Our
app is designed to simplify your daily life by helping you manage your tasks effortlessly.
</p>
37
<p style={styles.sectionContent}>
<strong>Key Features:</strong>
</p>
<ul style={styles.featuresList}>
<li>
<strong>Project Task Addition & Scheduling:</strong>
Users can add tasks and assign them specific timelines (daily, weekly, monthly).

```


Tasks are automatically split across different time periods for better tracking (daily, weekly, and monthly).

Progress Tracking & Visualization:

Real-time charts displaying task progress and completion rates.

Visual breakdown of tasks by time period, giving users insights into their project progress.

Rewards & Deductions:

Users receive rewards for completing tasks on time.

Incomplete tasks result in deductions, motivating users to stay accountable.

Motivational Reminders:

System sends motivational reminders to encourage users to stay focused and complete their tasks.

Comprehensive Dashboard:

An intuitive and user-friendly dashboard showing a clear overview of all tasks, progress, and upcoming deadlines.

Project Analysis:

Performance reports and task completion statistics are available to help users manage their projects more effectively.

<p style={styles.sectionContent}>

38

At ProjectMaster, we believe that staying organized shouldn't be difficult. That's why we've combined simplicity with powerful features to help you get more done, every day.

```

</p>
</div>
</div>
{/* Get the App Section */}
<div id="get-the-app" style={styles.rightColumn}>
  <div style={styles.section}>
    <h2 style={styles.sectionHeading}>Get the App</h2>
    <p style={styles.sectionContent}>
      Download our app today and start managing your projects more effectively. Available on
      all major platforms.
    </p>
    <Link to="/download" style={styles.button}>Download Now</Link>
  </div>
</div>
{/* Contact Us Section */}
<div id="contact" style={styles.leftColumn}>
  <div style={styles.section}>
    <h2 style={styles.sectionHeading}>Contact Us</h2>
    <p style={styles.sectionContent}>
      Have questions or need support? Reach out to us through our contact form or email us
      directly. We're here to help!
    </p>
    <Link to="/contact" style={styles.button}>Contact Us</Link>
  </div>
</div>
</div>
<div style={styles.starsEffect}></div> {/* Twinkling Stars Effect */}
</div>
);
export default Home;

```

ProjectDashboard.js:

```

import React, { useState, useEffect } from 'react';
import axios from 'axios';
39
import { useAuth } from '../context/AuthContext';
import { useNavigate } from 'react-router-dom';
const ProjectDashboard = () => {
  const { user } = useAuth();
  const navigate = useNavigate();
  const [project, setProject] = useState({
    title: "",
    description: "",
    priority: 'Medium',
    deadline: "",
    status: 'Not Started',
    teamMembersEmails: "",
  });
  const [projects, setProjects] = useState([]);
  const [finishedProjects, setFinishedProjects] = useState([]);
  const [isEditing, setIsEditing] = useState(false);
  const [editingProject, setEditingProject] = useState(null);
  const animationClass = 'slide-up';
  useEffect(() => {
    const fetchProjects = async () => {
      try {
        if (!user || !user.token) return;
        const token = user.token;
        const config = {
          headers: {
            Authorization: `Bearer ${token}`,
          },
        };
      }
    };
    // Fetch projects for both creator and assignees

```

```

const response = await axios.get('/api/projects', config);
const allProjects = response.data.projects; // Access 'projects' from response
// Separate into ongoing (not completed) and finished (completed) projects
setProjects(allProjects.filter(project => !project.isCompleted)); // Ongoing projects
setFinishedProjects(allProjects.filter(project => project.isCompleted)); // Finished projects
} catch (error) {
  if (error.response?.status === 401) {
    40
    console.error('Unauthorized! Logging out...');
    navigate('/login');
  } else {
    console.error('Error fetching projects:', error.response?.data || error.message);
  }
};
fetchProjects();
}, [user, navigate]);
const handleProjectChange = (e) => {
  setProject({ ...project, [e.target.name]: e.target.value });
};
const addProject = async () => {
  try {
    const token = user?.token;
    if (!token) throw new Error('No token available');
    const config = { headers: { Authorization: `Bearer ${token}` } };
    const teamMembersEmails = project.teamMembersEmails.split(',').map(email =>
      email.trim());
    const response = await axios.post('/api/projects', {
      ...project,
      teamMembersEmails, // Send team members' emails
      createdBy: user.id, // Send the user id as the creator
    }, config);
    setProjects([...projects, response.data]);
  }
};

```

```

    } catch (error) {
      console.error('Error adding project:', error.response?.data || error.message);
      alert(`Failed to add project. Error: ${error.response?.data?.message || error.message}`);
    }
  };

  const onSubmit = (e) => {
    e.preventDefault();
    if (isEditing) {
      updateProject();
    } else {
      addProject();
    }
  }
}

41
setProject({
  title: "",
  description: "",
  priority: 'Medium',
  deadline: "",
  teamMembersEmails: "",
  createdBy: user.id,
});

setIsEditing(false);
setEditingProject(null);
};

const editProject = (project) => {
  setProject({
    title: project.title,
    description: project.description,
    priority: project.priority,
    deadline: formatDateForInput(project.deadline),
    teamMembersEmails: project.teamMembers.map(member => member.email).join(', '), //
    Map team members to their emails
  });
};

```

```

setEditingProject(project);
setIsEditing(true);
};

const updateProject = async () => {
  try {
    const token = user?.token;
    if (!token) throw new Error('No token available');
    const config = { headers: { Authorization: `Bearer ${token}` } };
    const response = await axios.put(`/api/projects/${editingProject._id}`, project, config);
    setProjects(projects.map(p => (p._id === editingProject._id ? response.data : p)));
  } catch (error) {
    console.error('Error updating project:', error.response?.data || error.message);
    alert(`Failed to update project. Error: ${error.response?.data?.message || error.message}`);
  }
};

const handleDelete = async (projectId, isFinished) => {
  42
  try {
    const token = user?.token;
    if (!token) throw new Error('No token available');
    const config = { headers: { Authorization: `Bearer ${token}` } };
    await axios.delete(`/api/projects/${projectId}`, config);
    if (isFinished) {
      setFinishedProjects(finishedProjects.filter(p => p._id !== projectId));
    } else {
      setProjects(projects.filter(p => p._id !== projectId));
    }
  } catch (error) {
    alert('Error deleting project');
  }
};

const handleReopen = async (projectId) => {
  try {

```

```

const token = user?.token; // Ensure that you have a token
if (!token) throw new Error('No token available'); // Handle missing token
const config = { headers: { Authorization: `Bearer ${token}` } }; // Set token in request
headers
const response = await axios.put(`/api/projects/${projectId}/reopen`, {}, config);
// Handle success
setFinishedProjects(finishedProjects.filter(project => project._id !== projectId));
setProjects([...projects, response.data]);
} catch (error) {
console.error('Error reopening project:', error); // Log error for debugging
alert('Error reopening project');
}
};

const handleProjectClick = (projectId) => {
navigate(`/schedule/${projectId}`); // Navigate to the schedule page for the selected project
};

const formatDateForInput = (date) => {
if (!date) return '';
const formattedDate = new Date(date).toISOString().split('T')[0]; // Convert to
'YYYY-MM-DD'
return formattedDate;
};
43
const markAsFinished = async (projectId) => {
try {
const token = user?.token;
if (!token) throw new Error('No token available');
const config = { headers: { Authorization: `Bearer ${token}` } };
const response = await axios.put(`/api/projects/${projectId}/mark-as-finished`, {}, config);
const updatedProject = response.data.project;
setProjects((prevProjects) => prevProjects.filter((p) => p._id !== projectId));
setFinishedProjects((prevFinished) => [...prevFinished, updatedProject]);
} catch (error) {

```

```

console.error('Error marking project as finished:', error);
alert('Failed to mark project as finished');
}
};
return (
<div style={styles.container} className={animationClass}>
<style>
{
@keyframes slideUp {
from {
opacity: 0;
transform: translateY(100%);
}
to {
opacity: 1;
transform: translateY(0);
}
}
.slide-up {
animation: slideUp 0.7s ease-in-out;
}
`}
</style>
{/* Add Project Section */}
<div style={styles.addProjectSection}>
<header style={styles.header}>
44
<h1 style={styles.heading}>{isEditing ? 'Edit Project' : 'Add New Project'}</h1>
</header>
<form onSubmit={onSubmit} style={styles.form}>
<input
type="text"
name="title"

```



```

placeholder="Project Title"
value={project.title || ""} // Ensure it never changes to undefined
onChange={handleProjectChange}
required
style={styles.input}
/>
<textarea
name="description"
placeholder="Project Description"
value={project.description || ""}
onChange={handleProjectChange}
required
style={styles.textarea}
/>
<select
name="priority"
value={project.priority || 'Low'} // Default to 'Low' if undefined
onChange={handleProjectChange}
required
style={styles.select}
>
<option value="Low">Low</option>
<option value="Medium">Medium</option>
<option value="High">High</option>
</select>
<input
type="date"
name="deadline"
value={formatDateForInput(project.deadline) || ""} // Ensure it's never undefined
onChange={handleProjectChange}
required
style={styles.input}

```

```

/>
<input
  type="text"
  name="teamMembersEmails"
  placeholder="Enter team members' emails, separated by commas"
  value={project.teamMembersEmails || ""} // Default to empty string if undefined
  onChange={handleProjectChange}
  required
  style={styles.input}
/>

<button type="submit" style={styles.button}>
  {isEditing ? 'Update Project' : 'Add Project'}
</button>
</form>
</div>

{/* Project List */}
<div style={styles.projectContainer}>
  <div style={styles.projectListSection}>
    <h2 style={styles.subHeading}>Ongoing Projects</h2>
    <ul style={styles.projectList}>
      {projects.map((project) => (
        <li key={project._id} onClick={() => handleProjectClick(project._id)}
          style={styles.projectItem}>
            {console.log('Created By:', project?.createdBy)}
            <p>
              <b>Team Head:</b> {project?.createdBy?.email || 'No Email Available'}
            </p>
            <p>
              <b>Team Members:</b> {project?.teamMembers && project.teamMembers.length > 0
                ? project.teamMembers.map(member => member.email).join(', ') // Map to emails and join
                : 'Not Assigned'}
            </p>

```

```

<h3 style={styles.projectTitle}>{project.title}</h3>
<p style={styles.projectDescription}>{project.description}</p>
<p style={styles.projectPriority}>Priority: {project.priority}</p>
<p style={styles.projectDeadline}>Deadline: {new
Date(project.deadline).toLocaleDateString()}</p>
<button onClick={(e) => {e.stopPropagation(); editProject(project)}}
style={styles.editButton}>Edit</button>
<button onClick={(e) => { e.stopPropagation(); markAsFinished(project._id)}}
style={styles.finishButton}>Finished</button>
<button onClick={(e) => {e.stopPropagation(); handleDelete(project._id, false)}}
style={styles.deleteButton}>Delete</button>
<div style={styles.finishedProjectsSection}>
<h2 style={styles.subHeading}>Finished Projects</h2>
<ul style={styles.projectList}>
{finishedProjects.map((project) => (
<li key={project._id} style={styles.projectItem}>
<h3 style={styles.projectTitle}>{project.title}</h3>
<p style={styles.projectDescription}>{project.description}</p>
<p style={styles.projectPriority}>Priority: {project.priority}</p>
<p style={styles.projectDeadline}>Deadline: {new
Date(project.deadline).toLocaleDateString()}</p>
<p>Status: {project.status}</p>
<button onClick={() => handleDelete(project._id, true)}
style={styles.deleteButton}>Delete</button>
<button onClick={() => handleReopen(project._id)}
style={styles.reopenButton}>Reopen</button>
</li>
))}
</ul>
</div>
</div>
</div>
));
export default ProjectDashboard;

```

APPENDIX 2

SNAPSHOTS

FIGURE A2.0 Task page

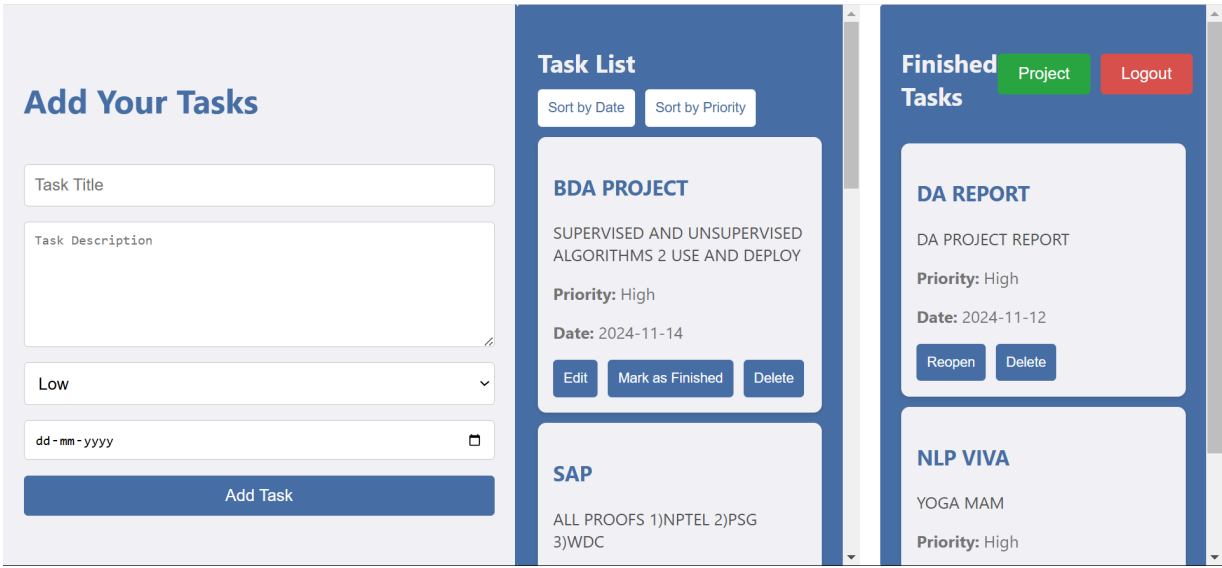


FIGURE A2.1 Project Page

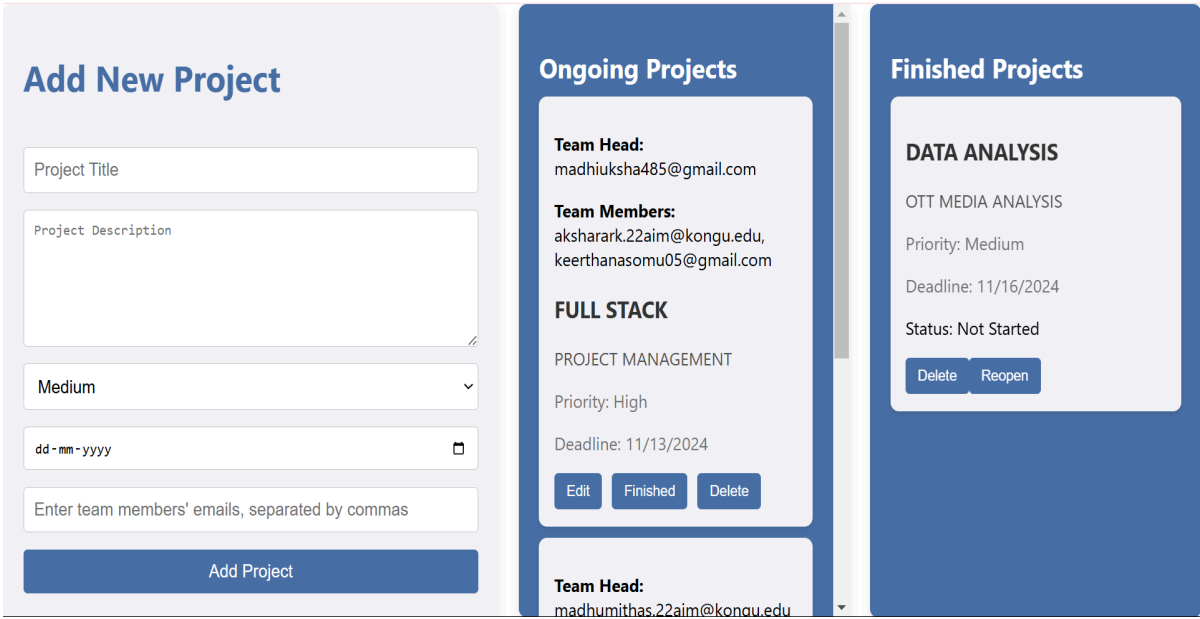


FIGURE A2.2 Schedule Page

Task Schedule for Project

Task Title

Task Description

dd-mm-yyyy

Assignee Email

Add Task

Backend

route connection

Due: 11/12/2024

Assigned to: Keerthana

Mark as Completed

Delete Task

REFERENCES

- [1] A. Sharma, M. Patel, "Full-Stack Project Management Application Development Using the MERN Stack," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 10, no. 3, March 2022. ISSN: 2320-9801.
- [2] K. Gupta, P. Desai, "Task Scheduling and Visualization in Project Management with MERN Stack," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 11, no. 5, May 2022, pp. 95-102. ISSN: 2277-128X.
- [3] D. Mehta, S. Rao, "Implementing Collaborative Project Management Systems Using React and Node.js," *International Journal of Engineering Research & Technology (IJERT)*, vol. 10, issue 4, April 2023. ISSN: 2278-0181.
- [4] P. Verma, R. Jain, "Role-Based Access Control in Full-Stack MERN Applications for Project Management," *Journal of Emerging Technologies and Innovative Research (JETIR)*, vol. 9, issue 2, Feb. 2023, pp. 85-90. ISSN: 2349-5162.
- [5] S. Gupta, M. Singh, "Integrating MongoDB for Large-Scale Project Management Systems in the MERN Stack," *International Journal of Engineering Trends and Technology (IJETT)*, vol. 69, issue 3, March 2023, pp. 45-50. ISSN: 2231-5381.
- [6] A. Kumar, R. Sahu, "Project Task Tracking and Visualization in Full-Stack MERN Applications," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 11, issue 1, Jan. 2024, pp. 78-84. ISSN: 2277-3878.
- [7] P. Singh, V. Sharma, "Data-Driven Project Management Dashboards with MERN Stack," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 11, issue 6, June 2023, pp. 100-108. ISSN: 2278-1021.