



SRI KRISHNA COLLEGE OF TECHNOLOGY

(An Autonomous Institution)

Approved by AICTE | Affiliated to Anna University Chennai|
Accredited by NBA - AICTE| Accredited by NAAC with 'A' Grade
KOVAIPUDUR, COIMBATORE 641042



TASK AND TIME MANAGEMENT

A PROJECT REPORT

Submitted by

GODREIGN ELGIN Y	727822TUAM014
LINGESH T	727822TUAM026
PALLAVI M	727822TUAM034
SANJANA KUMARI S	727822TUAM047

In partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

JULY 2024

BONAFIDE CERTIFICATE

Certified that this project report **TASK AND TIME MANAGEMENT** is the bonafide work of **GODREIGN ELGIN Y 727822TUAM014, LINGESH T 727822TUAM026, PALLAVI M 727822TUAM034, SANJANA KUMARI S 727822TUAM047** who carried out the project work under my supervision.

SIGNATURE

Mrs. SOUNDARYA S

SUPERVISOR

Assistant Professor,

Department of Computer Science
and Engineering (Artificial
Intelligence and Machine
Learning),

Sri Krishna College of
Technology, Coimbatore-641042

SIGNATURE

Dr. SUMA SIRA JACOB

HEAD OF THE DEPARTMENT

Associate Professor,

Department of Computer Science and
Engineering (Artificial Intelligence and
Machine Learning),

Sri Krishna College of
Technology, Coimbatore-641042

Certified that the candidate was examined by me in the Project Work Viva Voce examination held on _____ at Sri Krishna College of Technology, Coimbatore-641042.

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

First and foremost we thank the **Almighty** for being our light and for showering his gracious blessings throughout the course of this project.

We express our gratitude to our beloved Principal, **Dr. M.G. Sumithra**, for providing all facilities.

We are grateful to our beloved Head, Computing Sciences **Dr.T. Senthilnathan**, for her tireless and relentless support.

With the grateful heart, our sincere thanks to our Head of the Department **Dr. Suma Sira Jacob**, Department of Computer Science and Engineering for the motivation and all support to complete the project work.

We thank **Mrs. Soundarya**, Assistant Professor, Department of Computer Science and Engineering, for his motivation and support.

We are thankful to all the **Teaching and Non-Teaching Staff** of Department of Computer Science and Engineering and to all those who have directly and indirectly extended their help to us in completing this project work successfully.

We extend our sincere thanks to our **family members** and our beloved **friends**, who had been strongly supporting us in all our endeavour.

ABSTRACT

The efficient management of academic schedules is a critical challenge for educational institutions, impacting both faculty and students. Traditional timetable generation methods often involve manual effort and are prone to errors and inefficiencies. This project presents an automated timetable generation system designed to optimize the scheduling process, minimize conflicts, and ensure optimal resource utilization. By leveraging advanced algorithms and heuristics, the system dynamically creates timetables that accommodate various constraints, including classroom availability, faculty preferences, and course requirements. The system utilizes a constraint satisfaction approach to handle complex scheduling needs, incorporating features such as conflict resolution, priority-based scheduling, and real-time adjustments. A user-friendly interface allows administrators to input data, configure preferences, and generate timetables with minimal effort. The project aims to enhance the scheduling efficiency, reduce administrative workload, and improve overall educational experience by providing well-organized and conflict-free schedules. Through rigorous testing and validation, the system demonstrates its effectiveness in various academic settings, showcasing its adaptability and robustness in handling diverse scheduling scenarios. This automated solution represents a significant advancement in academic administration, offering a reliable and scalable approach to timetable generation.

TABLE OF CONTENT

CHAPTER.NO	TITLE	PAGE NO
	INTRODUCTION	
1	1.1 PROBLEM STATEMENT 1.2 OBJECTIVE 1.3 OVERVIEW	1
	SYSTEM SPECIFICATION	
2	2.1 VS CODE 2.2 LOCAL STORAGE	2
	PROPOSED SYSTEM	
3	3.1 PROPOSED SYSTEM 3.2 ADVANTAGES	4
4	METHODOLOGIES	6
	IMPLEMENTATION AND RESULT	
5	5.1 LOGIN 5.2 USER REGISTER 5.3 HOME AND ABOUT 5.4 PRICING PAGE 5.5 DASHBOARD AND PRODUCTS 5.6 CODING	9
	BACKEND SYSTEM SPECIFICATION	
6	6.1 SQL 6.2 REST API 6.3 SPRINGBOOT 6.4 SYSTEM ARCHITECTURE 6.5 CODING 6.6 SECURITY AND AUTHENTICATION 6.7 CODING	24
	CONCLUSION	
7	7.1 CONCLUSION 7.2 FUTURE SCOPE	42

LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
4.1	Process-flow Diagram	6
5.1	Login Page	9
5.2	Register Page	10
5.3	Home Page	11
5.4	About Page	11
5.5	Pricing Page	12
5.6	Dashboard Page	13
5.7	Task creation Page	13
5.8	Description Page	13
6.1	Tables in MySQL	24
6.2	Backend System Architecture	28
6.3	Activity Diagram	31
6.4	Use Case Diagram	32

LIST OF ABBREVIATIONS

ABBREVIATION

ACRONYM

HTML

HYPERTEXT MARKUP LANGUAGE

CSS

CASCADING STYLESHEET

JS

JAVASCRIPT

SDLC

SOFTWARE DEVELOPMENT LIFE CYCLE

JWT

JSON WEB TOKENS

CHAPTER 1

INTRODUCTION

This project aims to offer a seamless and efficient solution for individuals and teams looking to manage tasks and optimize their time through an online platform. In this chapter, we will explore the problem statement, provide an overview, and outline the main objectives of the task and time management system.

1.1 PROBLEM STATEMENT

How can we develop a task and time management system that allows individuals and teams to easily navigate through their tasks, manage their schedules efficiently, and collaborate effectively for optimal productivity?

1.2 OBJECTIVE

In the domain of task and time management, individuals and teams often encounter challenges such as disorganized task tracking, ineffective scheduling, and lack of collaboration. To address these issues, we propose the development of a task and time management system. This system will offer clear task organization, efficient scheduling tools, and collaborative features, ensuring a user-friendly and productive experience for all users.

1.3 OVERVIEW

The primary objective of this project is to develop a task and time management system that provides individuals and teams with a user-friendly platform to efficiently manage their tasks and time. By eliminating the inefficiencies of traditional task management methods and improving collaboration, the system aims to enhance productivity, accountability, and user satisfaction across all users.

CHAPTER 2

SYSTEM SPECIFICATION

In this chapter, we are going to see the software that we have used to build the website. This chapter gives you a small description about the software used in the project.

2.1 VS CODE

Visual Studio Code is a source code editor developed by Microsoft for Windows, Linux, and macOS. It includes support for debugging, embedded Git control, syntax highlighting, intelligent code completion, snippets, and code refactoring. It is also customizable, so users can change the editor's theme, keyboard shortcuts, and preferences.

VS Code is an excellent code editor for React projects. It is lightweight, customizable, and has a wide range of features that make it ideal for React development. It has built-in support for JavaScript, JSX, and TypeScript, and enables developers to quickly move between files and view detailed type definitions. It also has a built-in terminal for running tasks. Additionally, VS Code has an extensive library of extensions that allow developers to quickly add features like code snippets, debugging tools, and linting support to their projects.

2.2 LOCAL STORAGE

Task and Time Management is essential for individuals and teams aiming to maximize productivity and efficiency. This field focuses on organizing tasks, setting priorities, scheduling activities, and tracking progress to ensure the optimal use of time and resources. A well-designed task and time management system offers a comprehensive solution to these needs, providing users with tools to streamline their workflow and manage their responsibilities effectively. At its core, a task and time management system allows users to streamline their workflow by offering features such as task creation, assignment, and prioritization. Users can easily define tasks, set deadlines, and categorize

them based on their importance or urgency. This structured approach helps prevent overwhelm and ensures that critical tasks are addressed promptly. Advanced systems often include customizable templates and recurring task features, further enhancing efficiency by automating routine activities. Scheduling tools are a fundamental aspect of these systems, providing users with a clear overview of their commitments and deadlines. Calendar integrations and visual timelines help users plan their activities more effectively, allowing them to allocate time for each task and avoid conflicts. Reminders and notifications keep users informed of upcoming deadlines and important events, reducing the risk of missed tasks and improving overall time management. Collaboration features are integral to modern task and time management systems, particularly for teams working on projects. These systems facilitate seamless communication and coordination by enabling users to assign tasks to team members, track progress, and share updates in real-time. This collaborative approach enhances transparency and accountability, ensuring that everyone is aligned and contributing to the project's success. Moreover, task and time management systems often offer integration with other tools and platforms, creating a unified productivity ecosystem. Integration with project management software, communication tools, and time tracking applications allows users to manage their work more holistically. This connectivity ensures that data flows smoothly between different systems, reducing duplication of effort and improving overall efficiency. Analytics and reporting capabilities are also key features of advanced task and time management systems. These tools provide valuable insights into productivity trends, task completion rates, and team performance. By analyzing this data, users can identify areas for improvement, optimize their workflows, and make informed decisions about resource allocation. In conclusion, task and time management systems are essential for optimizing productivity and managing time effectively. They offer a range of features that streamline task management, enhance scheduling, and facilitate collaboration.

CHAPTER 3

PROPOSED SYSTEM

This chapter gives a small description about the proposed idea behind the development of our website.

3.1 PROPOSED SYSTEM

This system offers a multitude of benefits from various perspectives. The task and time management platform empowers users to efficiently manage their tasks, schedule activities, and collaborate with team members seamlessly. With a user-friendly interface, individuals and teams can create, prioritize, and track tasks with ease, ensuring that all responsibilities are organized and deadlines are met. The system provides a range of scheduling options, including calendar views and reminder notifications, which help users plan their work and stay on top of upcoming deadlines. It supports both individual task management and team collaboration, allowing users to assign tasks, monitor progress, and communicate effectively within the platform. Once tasks are created and scheduled, they are immediately available for tracking and management. The platform facilitates real-time updates and progress monitoring, enabling users to adjust their plans and priorities as needed. This ensures that tasks are completed efficiently and that any potential delays or issues are addressed promptly. By leveraging technology to streamline task management and time optimization, the platform improves both individual and team performance. It allows users to manage their responsibilities effectively, collaborate efficiently, and achieve their goals with greater ease. The result is a more organized, productive, and responsive work environment, where tasks are managed seamlessly, deadlines are met consistently, and collaboration is enhanced.

3.2 ADVANTAGES

CONVENIENCE:

Users can easily manage their tasks and schedules from anywhere with internet access. The system's intuitive interface allows individuals and teams to create, modify, and track tasks without the need for cumbersome manual methods or physical paperwork. This

convenience saves time and effort, especially for those juggling multiple responsibilities and tight deadlines.

ACCESSIBILITY:

The task and time management system provides 24/7 access to task lists, schedules, and project details. Users can view and update their tasks, deadlines, and progress at any time, ensuring that they can plan and adjust their work effectively. This accessibility helps in avoiding last-minute scrambles and ensures that tasks are managed proactively.

COLLABORATION:

The platform enhances team collaboration by enabling task assignment, progress tracking, and real-time communication within the system. Team members can coordinate more effectively, share updates, and provide feedback, which improves overall productivity and ensures that everyone is aligned with project goals.

EFFICIENCY:

By automating task tracking, scheduling, and reminders, the system significantly reduces manual effort and administrative workload. Users benefit from streamlined processes and automated notifications, which help in maintaining deadlines and managing workloads more efficiently. **Insights and Reporting:** The system provides advanced reporting and analytics tools that offer insights into productivity trends, task completion rates, and team performance. These insights help users identify areas for improvement, optimize workflows, and make data-driven decisions to enhance productivity and efficiency.

FLEXIBILITY:

Users have the flexibility to customize their task management experience according to their needs. The system supports various features such as recurring tasks, priority settings, and time tracking

CHAPTER 4

METHODOLOGIES

This chapter gives a small description about how our system works

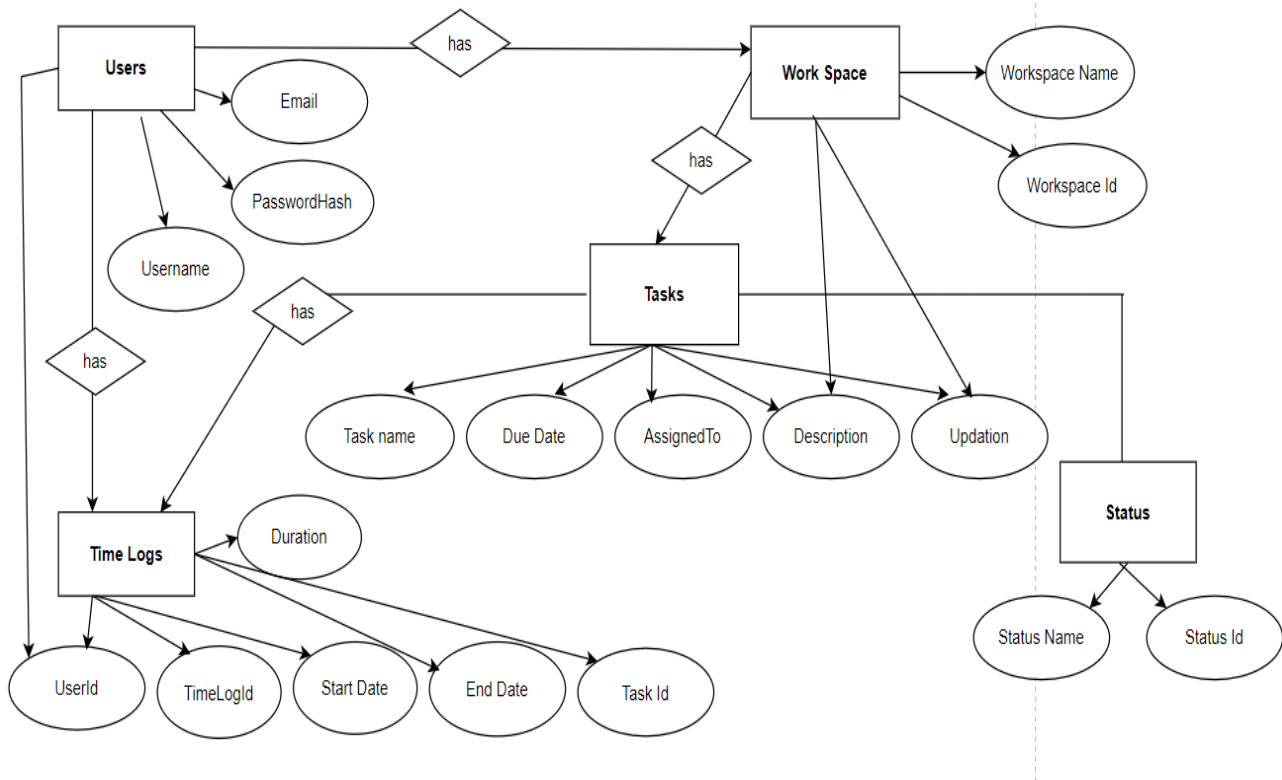


Fig 4.1.Process flow diagram

An task and time management system uses this schema to efficiently organize and schedule tasks, ensuring there are no conflicts and all constraints are met. Here's how each component contributes to the project:

1. USER ENTITY:

- Purpose: Manages user information and their roles (e.g., individual, team member).
- Usage: Helps determine what tasks and projects a user can access and manage.

2. TASK ENTITY:

- Purpose: Manages task details including title, description, deadline, and status.
- Usage: Defines the tasks that need to be completed and tracks their progress.

3. PROJECT ENTITY:

- Purpose: Manages project details including name, description, and associated tasks.
- Usage: Organizes tasks under a common goal or initiative, facilitating better management and tracking..

4. WORKSPACE ENTITY:

- Purpose: Manages workspace details such as workspace key, name, and associated users.
- Usage: Provides a collaborative environment where users can manage tasks and projects collectively.

5. TIMELOG ENTITY:

- Purpose: Manages time tracking details for tasks.
- Usage: Records the amount of time spent on each task, helping users track their productivity.

6. NOTIFICATION ENTITY:

- Purpose: Manages notifications and reminders for tasks and deadlines.
- Usage: Keeps users informed about upcoming deadlines and important updates to tasks and projects.

WORKSPACE GENERATION PROCESS

1. INPUT DATA:

Information about users, tasks, projects, workspaces, and time logs is entered into the system.

2. CONSTRAINT HANDLING:

- The system takes into account constraints such as:
 - Task deadlines.
 - User availability and workload.
 - Project dependencies.
 - Workspace collaboration needs.

3. TASK MANAGEMENT:

- Users can create, update, and delete tasks within projects and workspaces.
- Tasks are assigned to users with set deadlines and priority levels.
- Users can log the time spent on each task, providing detailed insights into task completion and time management.

4. PROJECT MANAGEMENT:

- Projects are created to group related tasks together.
- Users can update project details, track progress, and ensure that tasks are completed on time.

5. OUTPUT:

- The final includes detailed task lists, time logs, and project overviews that can be queried and viewed by users.
- Users can generate reports on their productivity, task completion rates, and project status.

CHAPTER 5

IMPLEMENTATION AND RESULT

This chapter gives a description about the output that we produced by developing the website of our idea

5.1 LOGIN

Log in to access your customized task and time management system. Once logged in, you can effortlessly create, manage, and track your daily, weekly, and monthly tasks. Our intelligent system generates your schedule based on your preferences and requirements, ensuring you stay on top of all your important activities. Whether you're managing personal tasks, professional projects, or a combination of both, our tool streamlines the process and keeps you organized. With features like automated reminders, priority setting, and progress tracking, you'll never miss a deadline or important event.

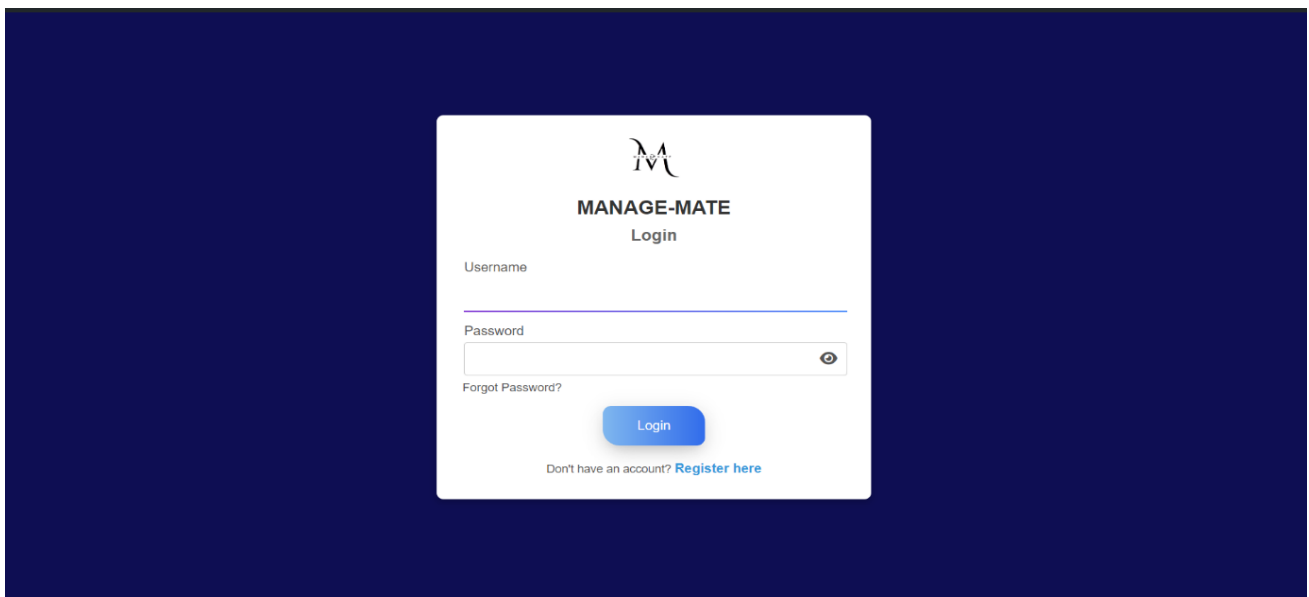


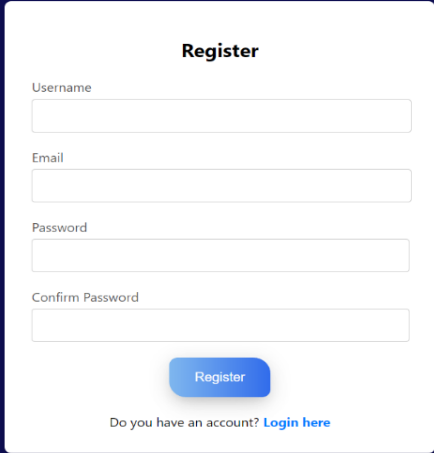
Fig 5.1 LOGIN PAGE

5.2 USER REGISTER

Our Automatic Task Management system simplifies your planning by creating optimal schedules based on your preferences and constraints. By entering your tasks, projects, and available time slots, the system efficiently generates a conflict-free schedule that maximizes your productivity.

With features like real-time adjustments and calendar integration, staying organized has never been easier. Enjoy a seamless task management experience that adapts to your needs and helps you stay on top of your commitments effortlessly.

Register now to get started and experience the convenience of an intelligent task management system designed just for you.



Register

Username

Email

Password

Confirm Password

[Register](#)

Do you have an account? [Login here](#)

Fig 5.2 REGISTER PAGE

5.3 HOME AND ABOUT PAGES

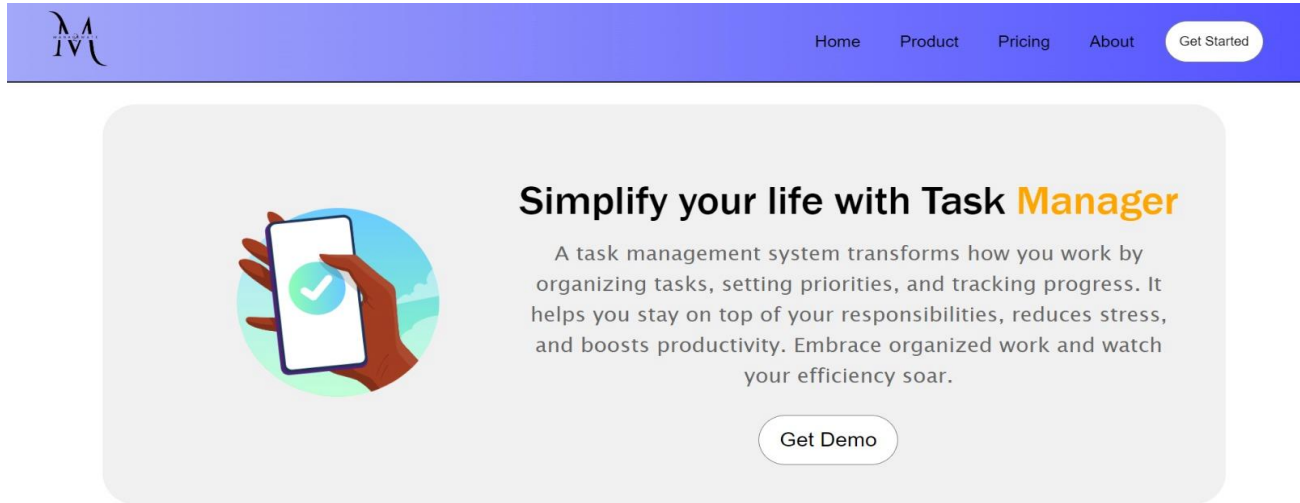


Fig 5.3 HOME PAGE



Welcome to #Manage-Mate.

Join our community of productivity enthusiasts and discover how ManageMate can transform the way you work and play. Let's make managing tasks fun again!

Fig 5.4 ABOUT US PAGE

5.4 PRICING COMPONENT

Explore our range of pricing plans designed to cater to your unique task and time management needs. Whether you're an individual looking for basic task tracking or a team requiring advanced project management features, we have a plan for you. Choose from Free, Standard, Premium, and Enterprise options, each offering a set of features to enhance your productivity and keep you organized. Upgrade at any time to unlock more capabilities and streamline your workflow.

Simple, transparent pricing for every team.

Team size: users

Bill me: Monthly Annually SAVE UP TO 17%

Free	Standard	Premium	Enterprise
Free forever for 10 users \$0	Everything you need to get started \$7.16 per user / month	Align multiple teams \$12.48 per user / month	Advanced analytics, scale and security for enterprises Contact sales
<ul style="list-style-type: none"> ✓ Unlimited goals, projects, tasks, and forms ✓ Backlog, list, board, timeline, calendar, and summary views ✓ Reports and dashboards ✓ 100 automations per site per month ✓ 2 GB of storage ✓ Support from Atlassian Community ✓ Up to 10 users 	<ul style="list-style-type: none"> ✓ Everything from Free plus: ✓ User roles and permissions ✓ External collaboration ✓ Multi-region data residency ✓ 1,700 automations per site per month ✓ 250 GB of storage ✓ 9/5 regional support ✓ Unlimited users 	<ul style="list-style-type: none"> ✓ Everything from Standard plus: ✓ Generate, summarize, and search content with Atlassian Intelligence (AI) ✓ Cross-team planning and dependency management ✓ Customizable approval processes ✓ Per user automation limits (1000 per month) ✓ Unlimited storage ✓ 24/7 support for critical issues ✓ 99.9% uptime SLA 	<ul style="list-style-type: none"> ✓ Everything from Premium plus: ✓ Cross-product insights with Atlassian Analytics and Data Lake ✓ Advanced admin controls and security ✓ Enterprise-grade identity and access management ✓ Unlimited automations ✓ Multiple instances (up to 150) ✓ 24/7 support for all issues ✓ 99.95% uptime SLA
Get it free	Start free trial	Start free trial	Start free trial

Fig 5.5 PRICING PAGE

5.5 DASHBOARD AND PRODUCTS

The Task Dashboard adapts to your workflow, making it easier to manage your time and stay productive. Whether you're juggling personal tasks or managing team projects, our dashboard keeps everything in one place, helping you stay organized and efficient.

In the Products Page, where you can explore the full range of features and tools designed to enhance your productivity and streamline your workflow. Whether you are an individual, a small team, or a large enterprise, our task and time management solutions are tailored to meet your needs.

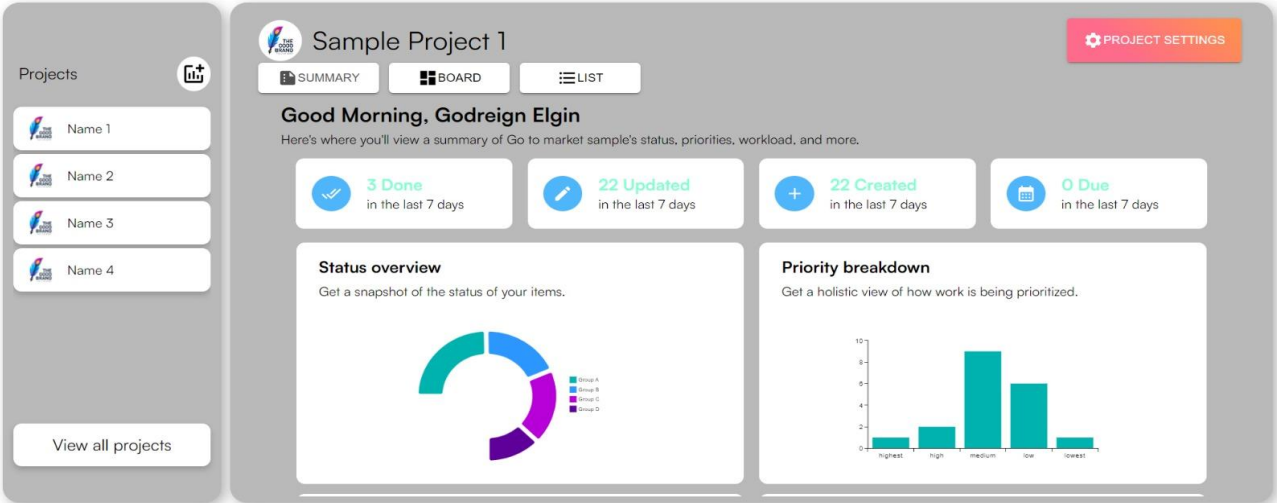


Fig 5.6 DASHBOARD

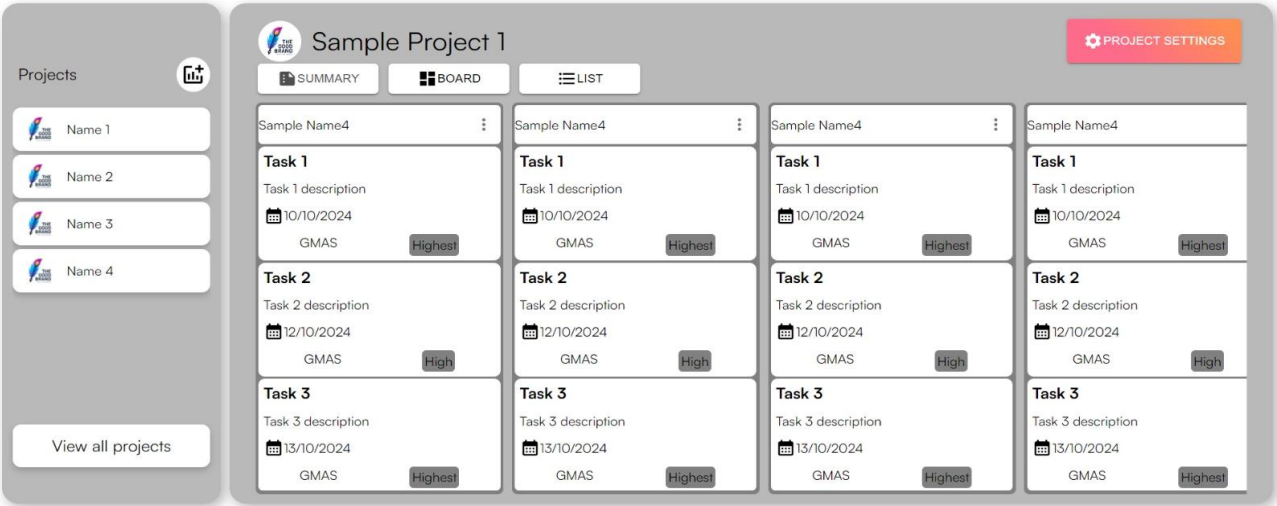


Fig 5.7 TASK CREATION

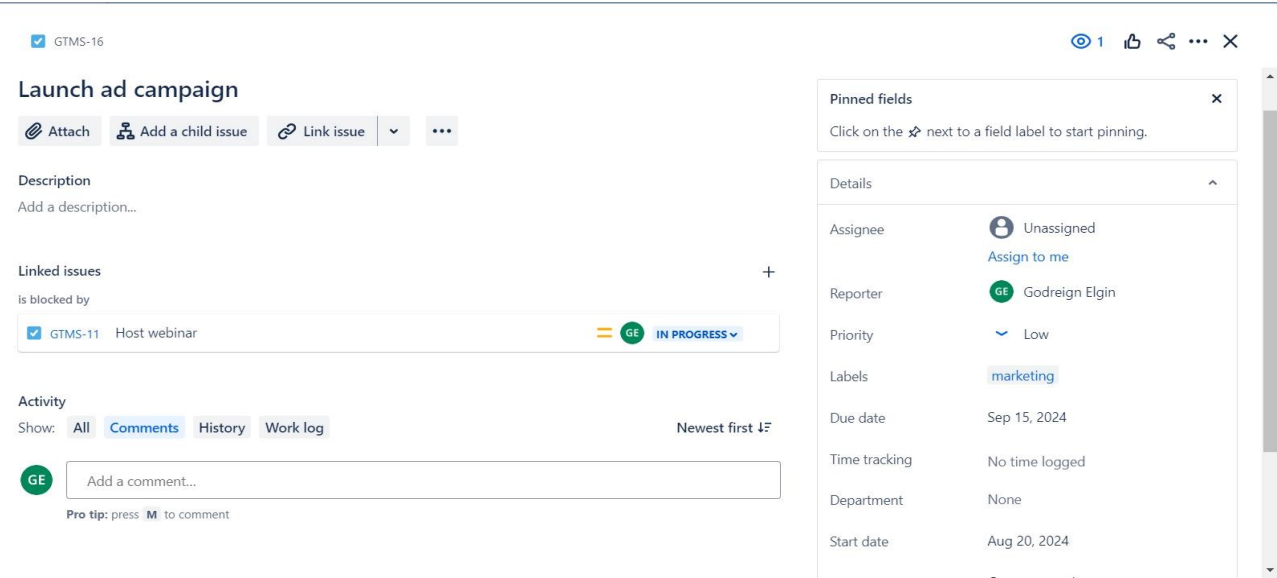


Fig 5.8 DESCRIPTION

5.6 CODING

LOGIN PAGE:

```
function Login1() {
  const [showPassword, setShowPassword] = useState(false);
  const [username, setUsername] = useState("");
  const [password, setPassword] = useState("");
  const [errors, setErrors] = useState({ });
  const [focusedField, setFocusedField] = useState("");
  const handleShowPassword = () => {
    setShowPassword(!showPassword);
  };
  const validateFields = useCallback(() => {
    const newErrors = { };
    const usernameRegex = /^[a-zA-Z0-9]{3,15}$/;
    const passwordRegex = /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)[a-zA-Z\d]{6,20}$/;
    if (!usernameRegex.test(username)) {
      newErrors.username = 'Invalid username or email';
    }
    if (!passwordRegex.test(password)) {
      newErrors.password = 'Invalid password';
    }
    setErrors(newErrors);
  }, [username, password]);
  useEffect(() => {
    validateFields();
  }, [username, password, validateFields]);
  const handleSubmit = (event) => {
    event.preventDefault();
    if (Object.keys(errors).length > 0 || username === "" || password === "") {
      alert('Please correct the errors before submitting. ');
    } else {
      alert(Logging in as ${username});
    }
  };
  return (
    <div className="container">
      <div className="card">
        
        <h1 className="title">MANAGE-MATE</h1>
      </div>
    </div>
  );
}
```

```

<h2 className="subtitle">Login</h2>
<form className="form" onSubmit={handleSubmit}>
  <div className="formGroup">
    <label htmlFor="username" className="label">Username/email</label>
    <input type="text"
      id="username"
      name="username"
      className="input"
      autoComplete="off"
      value={username}
      onChange={(e) => setUsername(e.target.value)}
      onFocus={() => setFocusedField('username')}
      onBlur={() => setFocusedField('')} required />
    {errors.username && focusedField === 'username' && (
      <div className="error-message">
        <FontAwesomeIcon icon={faExclamationCircle} /> {errors.username} </div> )}
    </div>
    <div className="formGroup">
      <label htmlFor="password" className="label">Password</label>
      <div className="passwordWrapper">
        <input type={showPassword ? 'text' : 'password'}
          id="password"
          name="password"
          className="input"
          autoComplete="off"
          value={password}
          onChange={(e) => setPassword(e.target.value)}
          onFocus={() => setFocusedField('password')}
          onBlur={() => setFocusedField('')} required />
        {focusedField === 'password' && (
          <span className="passwordIcon" onClick={handleShowPassword}>
            <FontAwesomeIcon icon={showPassword ? faEyeSlash : faEye} />
          </span> )} </div>
        {errors.password && focusedField === 'password' && (
          <div className="error-message">
            <FontAwesomeIcon icon={faExclamationCircle} /> {errors.password} </div> )}
        <div className="forgotPassword">

```

```

    <Link to="/forgot-password">Forgot Password?</Link> </div>
  </div>
  <center> <button type="submit" className="button">Login</button></center>
</form>
<div className="registerLink">
  <span>Don't have an account? </span>
  <Link to="/register">Register here</Link></div></div></div>): }

```

export default Login1;

PRICING PAGE:

```

const Pricing = () => {
  const [billingCycle, setBillingCycle] = useState('Monthly');
  const [teamSize, setTeamSize] = useState(300);
  const [showMore, setShowMore] = useState(false);
  return (
    <div className="pricing-container">
      <h1>
        <span className="blue-text">Simple, transparent pricing</span>
        <span className="white-text"> for every team.</span>
      </h1>
      <div className="team-size">
        <label htmlFor="team-size">Team size:</label>
        <input
          type="number"
          id="team-size"
          value={teamSize}
          onChange={(e) => setTeamSize(e.target.value)} />
        <span>users</span>
      </div>
      <div className="billing-cycle">
        <label>Bill me:</label>
        <button
          className={billingCycle === 'Monthly' ? 'active' : ''}
          onClick={() => setBillingCycle('Monthly')}
          Monthly
        </button>
        <button

```

```

    className={billingCycle === 'Annually' ? 'active' : ''}
    onClick={() => setBillingCycle('Annually')} >
    Annually
  </button>
  <span>SAVE UP TO 17%</span>
</div>
<div className="plans">
  {[ 'Free', 'Standard', 'Premium', 'Enterprise' ].map((plan, index) => (
    <div key={index} className={`plan ${plan.toLowerCase()}`} >
      <h2>{plan}</h2>
      <p>{plan === 'Free' ? 'Free forever for 10 users' : `Price for ${plan}`}</p>
      <p>${billingCycle === 'Monthly' ? (plan === 'Standard' ? '7.16' : plan === 'Premium' ? '12.48'
: '0') : (plan === 'Standard' ? '6.67' : plan === 'Premium' ? '10.33' : '0')} per user / month</p>
      <ul>
        <li>Features for {plan}</li>
      </ul>
      <Link to="/signup" className="cta-button">Start free trial</Link>
    </div>)))}
</div>
<div className="data-center-card">
  <h2>Pricing for Data Center</h2>
  <p>Scale your teams with Atlassian Data Center.</p>
  {showMore && <p>Additional information about Data Center pricing.</p>}
  <button onClick={() => setShowMore(!showMore)}>{showMore ? 'Show less' : 'Show
more'}</button>
</div>
<div className="features-table">
  <h2>Bring all your teams into Manage-Mate</h2>
  <table>
    <thead>
      <tr>
        <th>Feature</th>
        <th>Free</th>
        <th>Standard</th>
        <th>Premium</th>
        <th>Enterprise</th>
      </tr>

```



```

</thead>
<tbody>
  <tr>
    <td>User limit per site</td>
    <td>10 users</td>
    <td>50,000 users</td>
    <td>50,000 users</td>
    <td>50,000 users</td>
  </tr>
  <tr>
    <td>Storage</td>
    <td>2 GB</td>
    <td>250 GB</td>
    <td>Unlimited</td>
    <td>Unlimited</td>
  </tr>
</tbody>
</table>
</div>
<section className="questions-section">
  <h2>Questions?</h2>
  <p>We have answers. Read our full list of FAQs.</p>
</section>
<footer className="footer-container">
  {[ 'Company', 'PRODUCTS', 'RESOURCES', 'LEARN' ].map((section, index) => (
    <div key={index} className="footer-section">
      <h2>{section}</h2>
      <ul>
        <li><a href="#">Link 1</a></li>
        <li><a href="#">Link 2</a></li>
        <li><a href="#">Link 3</a></li>
      </ul>
    </div>
  ))}
</footer>
</div>):}

```

```
export default Pricing;
```

SUMMARY:

```

import React from 'react'
import '../Styles/Projects/Summary.css';
import DoneAllIcon from '@mui/icons-material/DoneAll';
import EditIcon from '@mui/icons-material/Edit';
import AddIcon from '@mui/icons-material/Add';
import CalendarMonthIcon from '@mui/icons-material/CalendarMonth';
import CalendarMonth from '@mui/icons-material/CalendarMonth';
import { PieChart } from '@mui/x-charts';
import { BarChart } from '@mui/x-charts/BarChart';
import ArrowUpwardIcon from '@mui/icons-material/ArrowUpward';
import ArrowDownwardIcon from '@mui/icons-material/ArrowDownward';
import ArrowForwardIcon from '@mui/icons-material/ArrowForward';
import ArrowBackIcon from '@mui/icons-material/ArrowBack';
import ExpandLessIcon from '@mui/icons-material/ExpandLess';
import HighestPriority from '../SampleJson/HighestPriority.json';
const barData = [
  { label: 'highest', value: 1, icon: <ArrowUpwardIcon style={{ verticalAlign: 'middle' }} /> },
  { label: 'high', value: 2, icon: <ArrowForwardIcon style={{ verticalAlign: 'middle' }} /> },
  { label: 'medium', value: 9, icon: <ExpandLessIcon style={{ verticalAlign: 'middle' }} /> },
  { label: 'low', value: 6, icon: <ArrowBackIcon style={{ verticalAlign: 'middle' }} /> },
  { label: 'lowest', value: 1, icon: <ArrowDownwardIcon style={{ verticalAlign: 'middle' }} /> },
];
const Summary = () => {
  const data = [
    { id: 0, value: 400, label: 'Group A' },
    { id: 1, value: 300, label: 'Group B' },
    { id: 2, value: 300, label: 'Group C' },
    { id: 3, value: 200, label: 'Group D' },
  ];
  const labels = barData.map(item => item.label);
  const values = barData.map(item => item.value);
  return (
    <div className='project-summary-main'>
      <div className='project-summary-main-u1'>
        <div className='summary-main-u1-greet'>Good Morning, Godreign Elgin</div>
        <div className='summary-main-u1-sub-greet'>Here's where you'll view a summary of Go to

```

market sample's status, priorities, workload, and more.</div>

</div>

<div className='project-summary-main-u2'>

<div className='summary-main-u2-card'>

<div className='summary-main-u2-card-left'>

<DoneAllIcon/>

</div>

<div className='summary-main-u2-card-right'>

<div className='summary-main-u2-card-right-u'>3 Done</div>

<div className='summary-main-u2-card-right-l'>in the last 7 days </div>

</div>

</div>

<div className='summary-main-u2-card'>

<div className='summary-main-u2-card-left'>

<EditIcon/>

</div>

<div className='summary-main-u2-card-right'>

<div className='summary-main-u2-card-right-u'>22 Updated</div>

<div className='summary-main-u2-card-right-l'>in the last 7 days </div>

</div>

</div>

<div className='summary-main-u2-card'>

<div className='summary-main-u2-card-left'>

<AddIcon/>

</div>

<div className='summary-main-u2-card-right'>

<div className='summary-main-u2-card-right-u'>22 Created</div>

<div className='summary-main-u2-card-right-l'>in the last 7 days </div>

</div>

</div>

<div className='summary-main-u2-card'>

<div className='summary-main-u2-card-left'>

<CalendarMonth/>

</div>

<div className='summary-main-u2-card-right'>

<div className='summary-main-u2-card-right-u'>0 Due</div>

<div className='summary-main-u2-card-right-l'>in the last 7 days </div>

```

    </div>
  </div>
</div>
<div className='project-summary-main-u3'>
  <div className='summary-main-u3-card'>
    <div className='summary-main-u3-card-u'>
      <div className='summary-main-u3-card-u-up'>
        Status overview
      </div>
      <div className='summary-main-u3-card-u-lo'>
        Get a snapshot of the status of your items.
      </div>
    </div>
    <div className='summary-main-u3-card-l'>
      <PieChart
        series={[
          {
            data: data.map(item => ({ ...item })),
            innerRadius: 130,
            outerRadius: 200,
            paddingAngle: 3,
            cornerRadius: 5,
            startAngle: -90,
            endAngle: 180,
            cx: 150,
            cy: 250,
          }
        ]}
        width={500}
        height={500}
      />
    </div>
  </div>
</div>
<div className='summary-main-u3-card'>
  <div className='summary-main-u3-card-u'>
    <div className='summary-main-u3-card-u-up'>
      Priority breakdown
    </div>
  </div>
</div>

```

```

</div>
<div className='summary-main-u3-card-u-lo'>
  Get a holistic view of how work is being prioritized.
</div>
</div>
<div className='summary-main-u3-card-l'>
  <BarChart
    xAxis={{ scaleType: 'band', data: labels }}
    series={{ data: values }}
    width={500}
    height={300}
    tooltip={{
      render: (params) => {
        const { index } = params[0];
        const item = data[index];

        return (
          <div style={{ display: 'flex', alignItems: 'center' }}>
            {item.icon}
            <span style={{ marginLeft: 5 }}>{item.label}: {item.value}</span>
          </div>
        );
      },
    }}
  />
</div>
</div>
</div>
<div className='project-summary-main-u4'>
  <div className='summary-main-u3-card'>
    <div className='summary-main-u3-card-u'>
      <div className='summary-main-u3-card-u-up'>
        Highest priority
      </div>
      <div className='summary-main-u3-card-u-lo'>
        Get a holistic view of how work is being prioritized.
      </div>
    </div>
  </div>
</div>

```

```

</div>
<div className='summary-main-u4-card-l'>
  {HighestPriority.map((item) => (
    <div className='summary-main-u3-card-l-list-card'>
      <div className='summary-u3-card-l-list-card-left'>
        <div className='summary-u3-card-l-list-card-left-u'>
          {item.taskName} {item.priority}
        </div>
        <div className='summary-u3-card-l-list-card-left-l'>
          {item.taskDesc}
        </div>
      </div>
      <div className='summary-u3-card-l-list-card-right'>
        <div className='summary-u3-card-l-list-card-right-u'>{item.status}</div>
        <div className='summary-u3-card-l-list-card-right-l'>{item.dueDate}</div>
      </div>
    </div>
  ))}
</div>
</div>
<div className='summary-main-u3-card'>
  <div className='summary-main-u3-card-u'>
    <div className='summary-main-u3-card-u-up'>
      Pinned tasks
    </div>
    <div className='summary-main-u3-card-u-lo'>
      Get a holistic view of how work is being prioritized.
    </div>
  </div>
</div>
<div className='summary-main-u4-card-l'>
  {HighestPriority.map((item) => (
    <div className='summary-main-u3-card-l-list-card'>
      <div className='summary-u3-card-l-list-card-left'>
        <div className='summary-u3-card-l-list-card-left-u'>
          {item.taskName} {item.priority}
        </div>
        <div className='summary-u3-card-l-list-card-left-l'>

```

```
        { item.taskDesc }
      </div>
    </div>
    <div className='summary-u3-card-l-list-card-right'>
      <div className='summary-u3-card-l-list-card-right-u'>{ item.status }</div>
      <div className='summary-u3-card-l-list-card-right-l'>{ item.dueDate }</div>
    </div>
  </div>
  ))}
</div>
</div>
</div>
</div>
)
}
export default Summary
```

CHAPTER - 6

BACKEND SYSTEM SPECIFICATION

In this chapter, the content discusses the software employed for constructing the website. This chapter provides a brief description of the software utilized in the project.

6.1 SQL:

```

+-----+
| Tables_in_sample_database |
+-----+
| list_model                 |
| notification_model         |
| roles                      |
| task_model                 |
| user_model                 |
| workspace_model            |
+-----+

```

Fig 6.1 Tables in MySQL

Local storage is a type of web storage for storing data on the client side of a web browser. It allows websites to store data on a user's computer, which can then be accessed by the website again when the user returns. Local storage is a more secure alternative to cookies because it allows websites to store data without having to send it back and forth with each request. It is similar to a database table in that it stores data in columns and rows, except that local storage stores the data in the browser rather than in a database.

Local storage is often used to store user information such as preferences and settings, or to store data that is not meant to be shared with other websites.

It is also used to cache data to improve the performance of a website. Local storage is supported by all modern web browsers, including chrome, Firefox, Safari, and Edge. It is accessible through the browser's JavaScriptAPI. Local storage is a powerful tool for websites to store data on the client side. It is secure, efficient and can be used to store data that does not need to be shared with other websites.

Local Storage is a great way to improve the performance of a website by caching data. Local storage in web browsers allows website data to be stored locally on the user's computer. It

is a way of persistently storing data on the client side, which is not sent to the server with each request. This allows users to store data such as preferences, login information, and form data without needing to send it to a server.

It is typically stored in a browser's cookie file, but it can also be stored in other locations such as HTML5 Local Storage and Indexed. The data stored in local storage is persistent and can be accessed by the website even if the user closes the browser or navigates to another page. It is a great way for websites to store user-specific data, as it is secure, reliable, and fast. It is also a great way for developers to store data that does not need to be sent to the server with each request.

One of the key benefits of using local storage is its reliability. Unlike server-side storage, which can be affected by network outages or other server issues, local storage is stored locally on the user's machine, and so is not affected by these issues. Another advantage of local storage is its speed. Because the data is stored locally, it is accessed quickly, as there is no need to send requests to a server.

6.2 REST API:

A REST API (Representational State Transfer Application Programming Interface) is a popular architectural style for designing networked applications. It is based on a set of principles and constraints that allow for scalability, simplicity, and interoperability between systems.

CLIENT-SERVER:

Separated entities communicate over HTTP or a similar protocol, with distinct responsibilities and the ability to evolve independently.

STATELESS:

Each request from the client to the server must contain all the necessary information to understand and process the request. The server does not maintain any client state between requests.

UNIFORM INTERFACE:

The API exposes a uniform interface, typically using HTTP methods (GET, POST, PUT, DELETE) to perform operations on resources. Resources are identified by URLs (Uniform Resource Locators).

CACHEABLE:

Responses can be cached by the client or intermediaries to improve performance and reduce the load on the server.

LAYERED SYSTEM:

Intermediary servers can be placed between the client and server to provide additional functionality, such as load balancing, caching, or security.

6.3 SPRINGBOOT:

Spring Boot is an open-source Java framework that simplifies the development of standalone, production-ready applications. It offers several advantages for building robust and scalable applications.

SIMPLIFIED CONFIGURATION:

Spring Boot eliminates the need for complex XML configuration files by leveraging sensible default configurations and annotations.

EMBEDDED SERVER:

Spring Boot includes an embedded server (e.g., Apache Tomcat, Jetty) that allows developers to create self-contained applications. This eliminates the need for external server installation and configuration, making it easier to package and deploy the application.

DEPENDENCY MANAGEMENT:

Spring Boot incorporates the concept of starter dependencies, which are curated sets of libraries that provide commonly used functionalities. It simplifies dependency management and ensures that all required dependencies are included automatically, reducing configuration issues and potential conflicts.

AUTO-CONFIGURATION:

Spring Boot's auto-configuration feature analyzes the class path and automatically configures the application based on the detected dependencies. It saves developers from writing boilerplate configuration code, resulting in faster development and reduced code clutter.

ACTUATOR:

Spring Boot Actuator provides out-of-the-box monitoring and management endpoints for the application. It offers metrics, health checks, logging, and other management features, making it easier to monitor and manage the application in production environments.

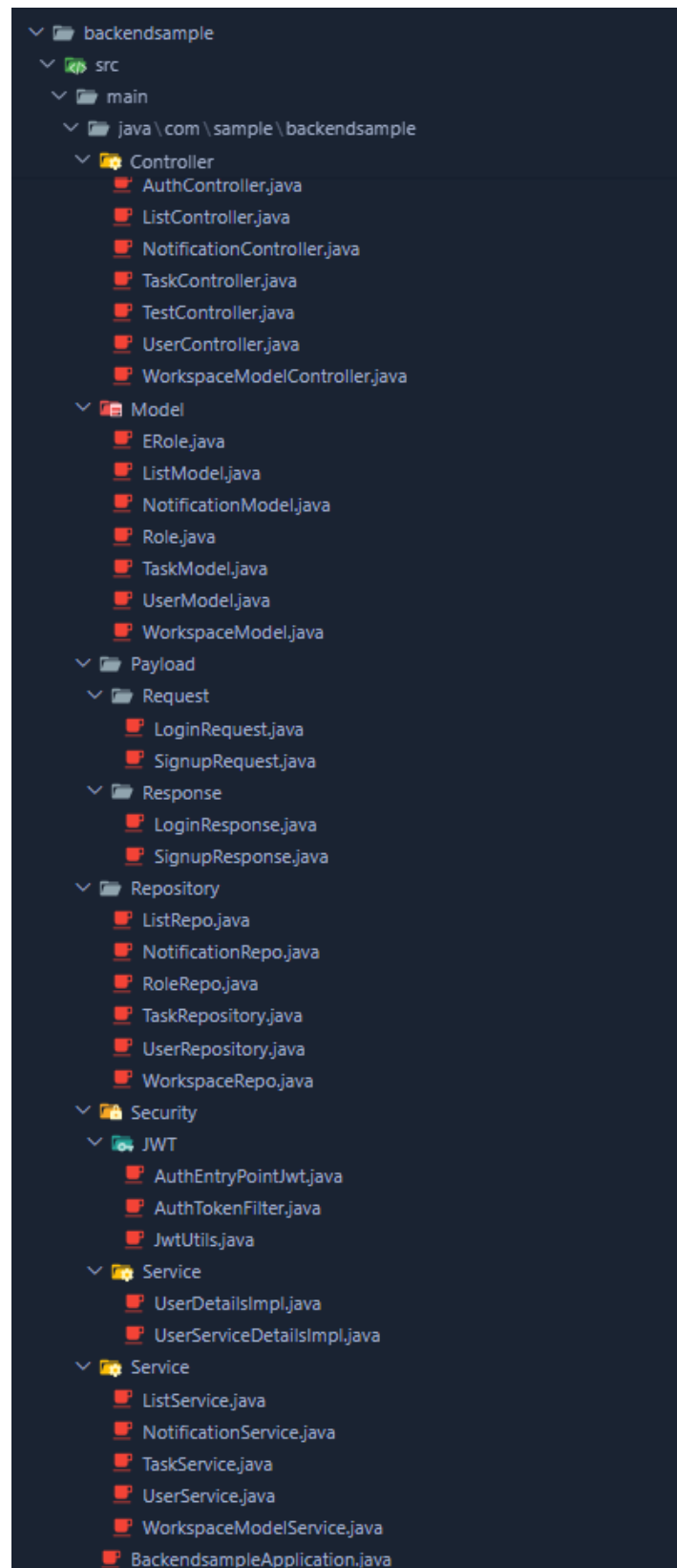
DEVOPS FRIENDLINESS:

Spring Boot's emphasis on simplicity and ease of use makes it DevOps friendly. It supports various deployment options, including traditional servers, cloud platforms, and containerization technologies like Docker. It also provides features for externalized configuration, making it easier to manage different environments.

6.4 SYSTEM ARCHITECTURE

The application adopts a contemporary and scalable three-tier architecture. It comprises the frontend layer, backend layer, and the database layer. Each of these layers fulfills a pivotal role in the application's comprehensive functionality, facilitating seamless communication and efficient data management.

BACKEND



6.2 Backend System Architecture

The backend layer of the Task and Time Management site is built upon the Spring Boot framework, a Java-based solution renowned for its capacity to simplify the development of resilient and scalable web applications. Spring Boot brings to the table a comprehensive array of features and libraries, offering streamlined solutions for managing HTTP requests, data persistence, implementing robust security measures, and seamlessly integrating with external systems. Within the application, the backend takes on the primary responsibility of crafting RESTful APIs. These APIs are meticulously designed to empower CRUD (Create, Read, Update, Delete) operations, catering to the dynamic world of task and time management. Furthermore, the backend is equipped to handle user management and authentication, ensuring a secure and personalized user experience for both administrators and users. In pursuit of enhanced security and modularity, the backend is strategically structured according to the principles of Spring Boot architecture.

SPRING BOOT:

Spring Boot is a Java framework that simplifies the process of building enterprise-grade applications. It provides a robust set of features and conventions for developing backend systems, including dependency management, configuration, and automatic setup. Spring Boot follows the principle of convention over configuration, reducing the amount of boilerplate code required.

REST API:

The backend of the Task and Time Management exposes a RESTful API that allows the frontend to communicate with the server. REST (Representational State Transfer) is an architectural style for designing networked applications. It uses standard HTTP methods (GET, POST, PUT, DELETE) to perform CRUD (Create, Read, Update, Delete) operations on resources. The API endpoints define the URLs and request/response formats for interacting with the system.

CONTROLLER:

In the application, controllers have a crucial role in managing incoming HTTP requests. Controllers map these requests to the appropriate methods within the system. API endpoints are defined by controllers, and they orchestrate the processing logic of incoming requests. Controllers act as the gateway between the frontend and backend, receiving user inputs, validating and processing data, interacting with services, and returning the relevant responses.

SERVICES:

Services within the application encapsulate the essential business logic. Responsible for orchestrating complex operations and facilitating interactions between different system components, these operations encompass data retrieval, validation, transformation, and storage. In this context, services manage task creation, handle user authentication, and other application-specific functionalities, ensuring a seamless user experience.

REPOSITORIES:

In the application, repositories serve as an abstraction layer for interacting with the database. They define the methods required for executing CRUD (Create, Read, Update, Delete) operations and querying the database using SQL or Object-Relational Mapping (ORM) frameworks like Hibernate. These repositories are instrumental in storing and retrieving user and task related data from the database, ensuring the persistence and accessibility of vital information.

DATA TRANSFER OBJECTS (DTOS):

Data Transfer Objects (DTOs) play a pivotal role in enabling data exchange between the frontend and backend layers of the application. These objects define the structure and format of data shared in API requests and responses. DTOs are employed to represent user details, task creations, updation, deletion, and other relevant data that is transferred between the frontend and backend, ensuring seamless communication.

SECURITY:

Security measures are a top priority within the application. Authentication and authorization protocols are diligently implemented to safeguard user data and system integrity. A robust security framework, such as Spring Security, is employed to manage user authentication and access control. It offers features such as user registration, login, password hashing, and role-based permissions, contributing to a secure and reliable application.

This structured and modular approach ensures that the Task and Time Management Site is efficient, secure, and scalable, providing a comprehensive solution for managing user tasks information.

ACTIVITY DIAGRAM

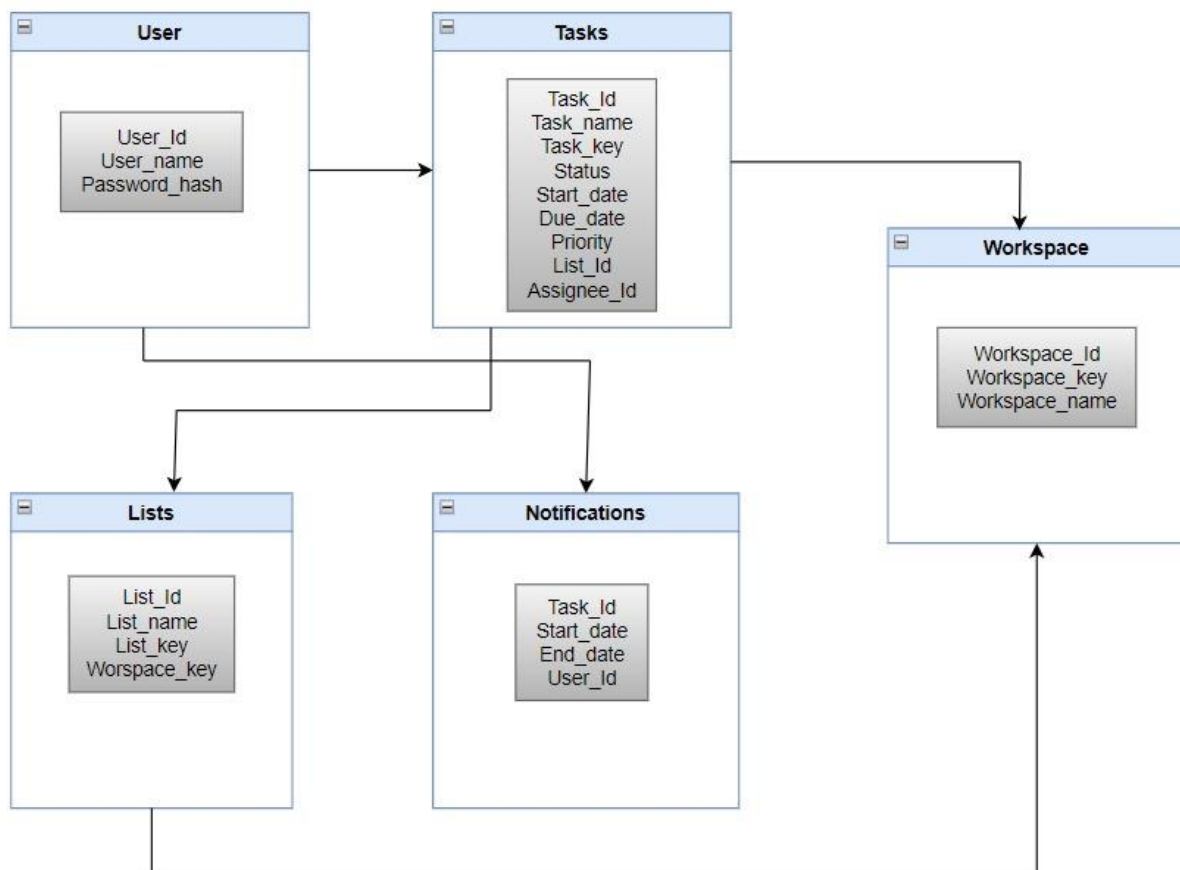


Fig 6.3 Activity Diagram

USE CASE DIAGRAM

A Use Case Diagram is a visual representation of the interactions between users (actors) and the system to achieve specific goals. For a task and time management application, a Use Case Diagram can illustrate how different types of users interact with the system.

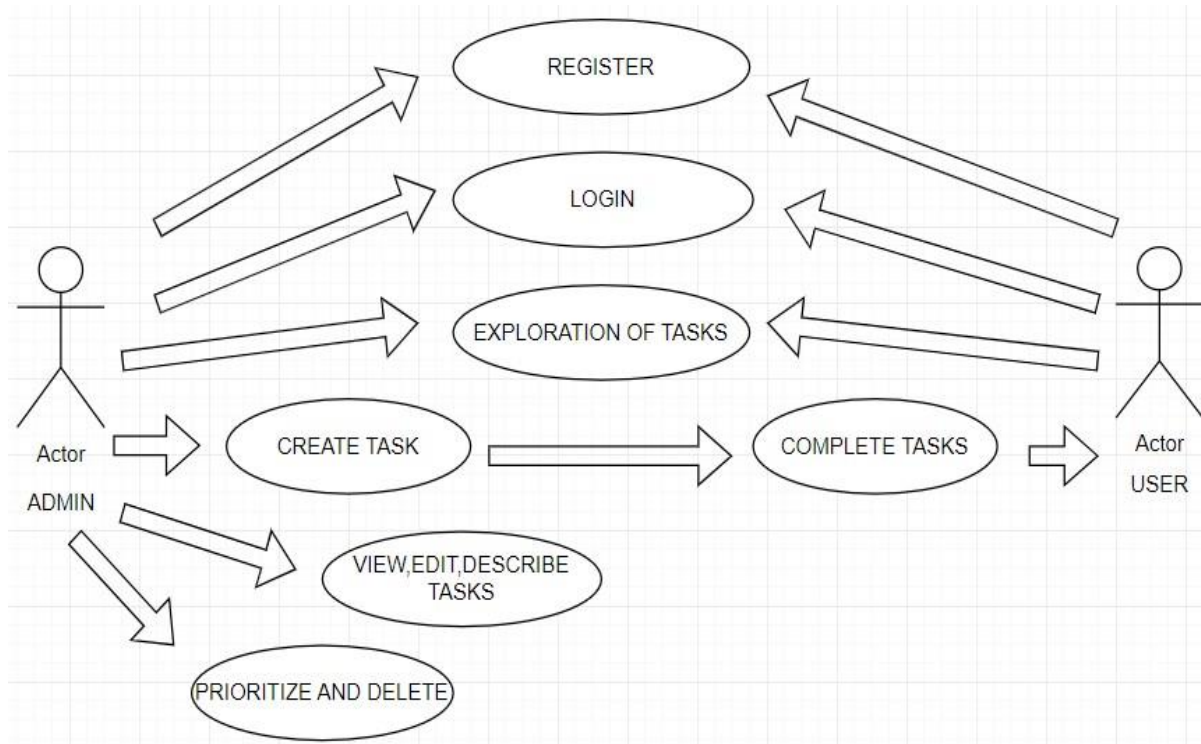


Fig 6.4 Use Case Diagram

6.5 CODING

TASK CONTROLLER.JAVA:

```

package com.sample.backendsample.Controller;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import com.sample.backendsample.Model.TaskModel;
import com.sample.backendsample.Service.TaskService;
import java.util.List;
import java.util.Optional;
@RestController
@RequestMapping("/api/tasks")
public class TaskController {
    @Autowired
    private TaskService taskService;
    @PostMapping
  
```



```

public ResponseEntity<TaskModel> createTask(@RequestBody TaskModel task) {
    TaskModel createdTask = taskService.createTask(task);
    return ResponseEntity.ok(createdTask);
}
@GetMapping
public ResponseEntity<List<TaskModel>> getAllTasks() {
    List<TaskModel> tasks = taskService.getAllTasks();
    return ResponseEntity.ok(tasks);
}
@GetMapping("/{id}")
public ResponseEntity<TaskModel> getTaskById(@PathVariable Long id) {
    Optional<TaskModel> task = taskService.getTaskById(id);
    return task.map(ResponseEntity::ok).orElseGet(() -> ResponseEntity.notFound().build());
}
@GetMapping("/key/{taskKey}")
public ResponseEntity<TaskModel> getTaskByKey(@PathVariable String taskKey) {
    Optional<TaskModel> task = taskService.getTaskByKey(taskKey);
    return task.map(ResponseEntity::ok).orElseGet(() -> ResponseEntity.notFound().build());
}
@GetMapping("/name/{taskName}")
public ResponseEntity<TaskModel> getTaskByName(@PathVariable String taskName) {
    Optional<TaskModel> task = taskService.getTaskByName(taskName);
    return task.map(ResponseEntity::ok).orElseGet(() -> ResponseEntity.notFound().build());
}
@PutMapping("/{id}")
public ResponseEntity<TaskModel> updateTaskById(@PathVariable Long id, @RequestBody
TaskModel taskDetails) {
    Optional<TaskModel> updatedTask = taskService.updateTaskById(id, taskDetails);
    return updatedTask.map(ResponseEntity::ok).orElseGet(() ->
ResponseEntity.notFound().build());
}
@PutMapping("/key/{taskKey}")
public ResponseEntity<TaskModel> updateTaskByKey(@PathVariable String taskKey,
@RequestBody TaskModel taskDetails) {
    Optional<TaskModel> updatedTask = taskService.updateTaskByKey(taskKey, taskDetails);
    return updatedTask.map(ResponseEntity::ok).orElseGet(() ->
ResponseEntity.notFound().build());
}
@PutMapping("/name/{taskName}")
public ResponseEntity<TaskModel> updateTaskByName(@PathVariable String taskName,
@RequestBody TaskModel taskDetails) {
    Optional<TaskModel> updatedTask = taskService.updateTaskByName(taskName, taskDetails);
    return updatedTask.map(ResponseEntity::ok).orElseGet(() ->
ResponseEntity.notFound().build());
}

```

```

    }
    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteTaskById(@PathVariable Long id) {
        taskService.deleteTaskById(id);
        return ResponseEntity.noContent().build();
    }
    @DeleteMapping("/key/{taskKey}")
    public ResponseEntity<Void> deleteTaskByKey(@PathVariable String taskKey) {
        taskService.deleteTaskByKey(taskKey);
        return ResponseEntity.noContent().build();
    }
    @DeleteMapping("/name/{taskName}")
    public ResponseEntity<Void> deleteTaskByName(@PathVariable String taskName) {
        taskService.deleteTaskByName(taskName);
        return ResponseEntity.noContent().build();
    }
}

```

TASK MODEL.JAVA:

```

package com.sample.backendsample.Model;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import java.util.Date;
@Data
@NoArgsConstructor
@AllArgsConstructor
@Entity
public class TaskModel {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long taskId;
    @Column(unique = true)
    private String taskKey;
    @Column(nullable = false, length = 20)
    private String taskName;
}

```

```

@Column(length = 100)
private String taskDesc;
@Column(nullable = false)
private int priority;
@Column(nullable = false)
private String taskStatus;
@Column(nullable = false)
private Date createdDate;
@Column(nullable = false)
private Date dueDate;
private Long assigneeId;
private Long listIdReference;
public static String generateTaskKey(Long id) {
    return String.format("TSK%03d", id);
}
}

```

TASK SERVICE.JAVA

```

package com.sample.backendsample.Service;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.sample.backendsample.Model.TaskModel;
import com.sample.backendsample.Repository.TaskRepository;
import java.util.List;
import java.util.Optional;
@Service
public class TaskService {
    @Autowired
    private TaskRepository taskRepository;
    public TaskModel createTask(TaskModel task) {
        TaskModel savedTask = taskRepository.save(task);
        savedTask.setTaskKey(TaskModel.generateTaskKey(savedTask.getTaskId()));
        return taskRepository.save(savedTask);
    }
    public Optional<TaskModel> getTaskById(Long taskId) {
        return taskRepository.findById(taskId);
    }
    public Optional<TaskModel> getTaskByKey(String taskKey) {
        return taskRepository.findByTaskKey(taskKey);
    }
}

```

```

public Optional<TaskModel> getTaskByName(String taskName) {
    return taskRepository.findByTaskName(taskName);
}
public List<TaskModel> getAllTasks() {
    return taskRepository.findAll();
}
public Optional<TaskModel> updateTaskById(Long taskId, TaskModel taskDetails) {
    return taskRepository.findById(taskId).map(task -> {
        task.setTaskName(taskDetails.getTaskName());
        task.setTaskDesc(taskDetails.getTaskDesc());
        task.setPriority(taskDetails.getPriority());
        task.setTaskStatus(taskDetails.getTaskStatus());
        task.setCreatedDate(taskDetails.getCreatedDate());
        task.setDueDate(taskDetails.getDueDate());
        task.setAssigneeId(taskDetails.getAssigneeId());
        task.setListIdReference(taskDetails.getListIdReference());
        return taskRepository.save(task);
    });
}
public Optional<TaskModel> updateTaskByKey(String taskKey, TaskModel taskDetails)
{
    return taskRepository.findByTaskKey(taskKey).map(task -> {
        task.setTaskName(taskDetails.getTaskName());
        task.setTaskDesc(taskDetails.getTaskDesc());
        task.setPriority(taskDetails.getPriority());
        task.setTaskStatus(taskDetails.getTaskStatus());
        task.setCreatedDate(taskDetails.getCreatedDate());
        task.setDueDate(taskDetails.getDueDate());
        task.setAssigneeId(taskDetails.getAssigneeId());
        task.setListIdReference(taskDetails.getListIdReference());
        return taskRepository.save(task);
    });
}
public Optional<TaskModel> updateTaskByName(String taskName, TaskModel
taskDetails) {
    return taskRepository.findByTaskName(taskName).map(task -> {
        task.setTaskName(taskDetails.getTaskName());
        task.setTaskDesc(taskDetails.getTaskDesc());
        task.setPriority(taskDetails.getPriority());
        task.setTaskStatus(taskDetails.getTaskStatus());

```

```

        task.setCreatedDate(taskDetails.getCreatedDate());
        task.setDueDate(taskDetails.getDueDate());
        task.setAssigneeId(taskDetails.getAssigneeId());
        task.setListIdReference(taskDetails.getListIdReference());
        return taskRepository.save(task);
    });
}
public void deleteTaskById(Long taskId) {
    taskRepository.deleteById(taskId);
}
public void deleteTaskByKey(String taskKey) {
    taskRepository.deleteByTaskKey(taskKey);
}
public void deleteTaskByName(String taskName) {
    taskRepository.deleteByTaskName(taskName);
}
}

```

6.6 SECURITY AND AUTHENTICATION

Security and authentication lie at the heart of the application's robust infrastructure, ensuring the protection of sensitive data and upholding the integrity of the system.

USER AUTHENTICATION:

User authentication is a fundamental pillar, allowing users, including administrators and employees, to register securely by leveraging email and strong, hashed passwords. A robust login system verifies user credentials and controls access to the application. This ensures that only authorized users can access and manage sensitive tax information.

DATA ENCRYPTION:

Data encryption, both in transit and at rest, is a core component of the security framework. Secure communication channels protect data during interactions between the frontend and backend, while encryption of sensitive data in the database safeguards information in the event of a breach. This dual-layer encryption approach ensures that user data, including personal and financial details, remains confidential and protected.

SESSION MANAGEMENT:

Session management maintains secure user sessions, preventing unauthorized access or data exposure. By implementing secure session tokens, the application ensures that users' sessions are both authenticated and securely managed throughout their interaction with the platform.

SECURITY FRAMEWORKS:

Utilizing security libraries and frameworks, such as Spring Security, enhances the efficiency of authentication and access control. Spring Security provides comprehensive features for securing applications, including user registration, login, password hashing, and role-based permissions, which are essential for protecting sensitive user data and controlling access within the system.

PROTECTION AGAINST SECURITY THREATS:

The application includes protection against common security threats such as cross-site scripting (XSS) and SQL injection. Input validation and sanitization mechanisms are in place to ensure that user inputs are correctly processed and stored, preventing malicious attacks that could compromise the system.

USER AWARENESS AND EDUCATION:

User awareness and education contribute to the overall security posture, ensuring that users are informed and capable of recognizing potential threats. The application provides users with guidelines on creating strong passwords, recognizing phishing attempts, and securely managing their accounts.

REGULAR SECURITY AUDITS:

Regular security audits and vulnerability assessments are conducted to proactively identify and address potential weaknesses. These audits help maintain the application's resilience against emerging security risks and ensure that any vulnerabilities are promptly mitigated.

6.7 CODING

AUTH ENTRY POINT JWT. JAVA:

```

@Component
public class AuthEntryPointJwt implements AuthenticationEntryPoint {
    private static final Logger logger = LoggerFactory.getLogger(AuthEntryPointJwt.class);
    @Override
    public void commence(HttpServletRequest request, HttpServletResponse response,
AuthenticationException authException)
        throws IOException, ServletException {
        logger.error("Unauthorized error: {}", authException.getMessage());
        response.setContentType(MediaType.APPLICATION_JSON_VALUE);
        response.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
        final Map<String, Object> body = new HashMap<>();
        body.put("status", HttpServletResponse.SC_UNAUTHORIZED);
        body.put("error", "Unauthorized");
        body.put("message", authException.getMessage());
        body.put("path", request.getServletPath());
        final ObjectMapper mapper = new ObjectMapper();
        mapper.writeValue(response.getOutputStream(), body);
    }
}

```

AUTH CONTROLLER. JAVA:

```

@CrossOrigin(origins = "*", maxAge = 3600)
@RestController
@RequestMapping("/api/auth")
public class AuthController {
    @Autowired
    AuthenticationManager authenticationManager;
    @Autowired
    UserRepository userRepository;
    @Autowired
    RoleRepository roleRepository;
    @Autowired
    PasswordEncoder encoder;
    @Autowired
    JwtUtils jwtUtils;
    @PostMapping("/signin")

```

```

public ResponseEntity<?> authenticateUser(@Valid @RequestBody LoginRequest
loginRequest) {
    Authentication authentication = authenticationManager
        .authenticate(new
UsernamePasswordAuthenticationToken(loginRequest.getUsername(),
loginRequest.getPassword()));
    SecurityContextHolder.getContext().setAuthentication(authentication);
    UserDetailsImpl userDetails = (UserDetailsImpl) authentication.getPrincipal();
    ResponseCookie jwtCookie = jwtUtils.generateJwtCookie(userDetails);
    List<String> roles = userDetails.getAuthorities().stream()
        .map(item -> item.getAuthority())
        .collect(Collectors.toList());
    return ResponseEntity.ok().header(HttpHeaders.SET_COOKIE, jwtCookie.toString())
        .body(new UserInfosResponse(userDetails.getId(),
            userDetails.getUsername(),
            userDetails.getEmail(),
            roles));
}

@PostMapping("/signup")
public ResponseEntity<?> registerUser(@Valid @RequestBody SignupRequest
signUpRequest) {
    if (userRepository.existsByUsername(signUpRequest.getUsername())) {
        return ResponseEntity.badRequest().body(new MessageResponse("Error: Username is
already taken!"));
    }
    if (userRepository.existsByEmail(signUpRequest.getEmail())) {
        return ResponseEntity.badRequest().body(new MessageResponse("Error: Email is
already in use!"));
    }
    User user = new User(signUpRequest.getUsername(),
        signUpRequest.getEmail(),
        encoder.encode(signUpRequest.getPassword()));
    Set<String> strRoles = signUpRequest.getRole();
    Set<Role> roles = new HashSet<>();
    if (strRoles == null) {
        Role userRole = roleRepository.findByName(ERole.ROLE_USER)
            .orElseThrow(() -> new RuntimeException("Error: Role is not found."));
        roles.add(userRole);
    } else {
        strRoles.forEach(role -> {

```



```

switch (role) {
case "admin":
    Role adminRole = roleRepository.findByName(ERole.ROLE_ADMIN)
        .orElseThrow(() -> new RuntimeException("Error: Role is not found."));
    roles.add(adminRole);
    break;
case "mod":
    Role modRole = roleRepository.findByName(ERole.ROLE_MODERATOR)
        .orElseThrow(() -> new RuntimeException("Error: Role is not found."));
    roles.add(modRole);
    break;
default:
    Role userRole = roleRepository.findByName(ERole.ROLE_USER)
        .orElseThrow(() -> new RuntimeException("Error: Role is not found."));
    roles.add(userRole);
}
});
}
user.setRoles(roles);
userRepository.save(user);
return ResponseEntity.ok(new MessageResponse("User registered successfully!"));
}
@PostMapping("/signout")
public ResponseEntity<?> logoutUser() {
    ResponseCookie cookie = jwtUtils.getCleanJwtCookie();
    return ResponseEntity.ok().header(HttpHeaders.SET_COOKIE, cookie.toString())
        .body(new MessageResponse("You've been signed out!"));
}
}

```

CHAPTER 7

CONCLUSION

This chapter tells about the conclusion that anyone can drive from the project and the learning we learnt by taking over this project.

7.1 CONCLUSION

In conclusion, the proposed task and time management system is designed to benefit individuals and organizations of all sizes, from small teams to large enterprises. This project is scalable to accommodate a diverse range of users and tasks, aiming to streamline task management, optimize scheduling, and enhance collaboration. It simplifies the process of organizing and tracking tasks, provides clear visibility into deadlines and progress, and ensures effective communication and coordination among team members. By promoting transparency, efficiency, and adaptability, the system empowers users to manage their responsibilities more effectively, improve productivity, and achieve their goals with greater ease. With its comprehensive features and real-time updates, this system enhances overall workflow management and supports a more organized and productive work environment.

7.2 FUTURE SCOPE

The proposed task and time management system is designed to benefit individuals and organizations of all sizes, from small teams to large enterprises. Scalable to accommodate a diverse range of users and tasks, this project aims to streamline task management, optimize scheduling, and enhance collaboration. It simplifies organizing and tracking tasks, provides clear visibility into deadlines and progress, and ensures effective communication and coordination among team members. By promoting transparency and efficiency, the system empowers users to manage their responsibilities more effectively, improve productivity, and achieve their goals with greater ease. With comprehensive features and real-time updates, this system enhances overall workflow management and supports a more organized and productive work environment.