TASK AND PROJECT MANAGEMENT

CONTENTS

1.ABSTRACT

Task and Project Management application designed to enhance productivity and organization in dynamic work environments. Functionalities include user authentication, dashboard navigation, task management, and project oversight. Users can securely sign up and log in, accessing personalized dashboards to manage tasks and projects efficiently. Tasks can be added, viewed, and deleted, with support for deadlines and real-time updates. Projects can be created, edited, and deleted, providing users with a structured framework for managing initiatives.

The application prioritizes data persistence using localStorage, ensuring seamless access to tasks and projects across sessions. Security measures include bcrypt encryption for user credentials and adherence to industry best practices. Additionally, the application is designed for scalability, enabling future enhancements and feature additions to meet evolving user needs.

2.INTRODUCTION

In the dynamic landscape of modern work environments, efficient task and project management are indispensable for maintaining productivity and achieving organizational goals. Our Task and Project Management application offers a comprehensive solution to streamline workflows, organize tasks, and oversee projects with ease. Built on a robust technology stack leveraging React.js for the frontend and Express.js for the backend, our application combines powerful features with intuitive user interfaces to enhance productivity and collaboration.

At its core, our application focuses on empowering users to manage their tasks and projects effectively. With a user-friendly interface and seamless navigation, users can easily log in, access their personalized dashboards, and dive into their tasks and projects. The dashboard serves as a centralized hub, providing quick access to essential functionalities and offering insights into ongoing activities.One of the standout features of our application is its robust task management capabilities. Users can effortlessly add, view, and delete tasks, prioritizing them based on deadlines and importance. Tasks are seamlessly integrated with deadlines, allowing users to stay organized and focused on their objectives. Additionally, our application ensures data persistence using localStorage, enabling users to access their tasks across sessions and devices.

In parallel, our application offers comprehensive project management functionalities to help users oversee their projects from inception to completion. Users can create, edit, and delete projects, each with its unique name and description. Projects are displayed in a structured format, providing users with a clear overview of their ongoing initiatives and facilitating collaboration with team members.

Moreover, our application prioritizes security and scalability, implementing robust authentication mechanisms and adhering to industry best practices. User credentials are encrypted using bcrypt, ensuring secure transmission and storage of sensitive information. Additionally, the application is designed with scalability in mind, allowing for future enhancements and feature additions to meet evolving user needs.

In essence, our Task and Project Management application empowers users to optimize their workflows, increase productivity, and achieve their objectives with confidence. Whether managing personal tasks or coordinating complex projects, our application serves as a trusted ally, simplifying the journey towards success in today's fast-paced world.

3.FEATURES

A task and project management system typically encompasses various features aimed at facilitating efficient planning, organization, collaboration, and tracking of tasks and projects. Here are some common features you might find in such systems:

User Authentication:

- Allows users to sign up with a name, email, and password.
- Provides secure login functionality.
- Validates user input on the client-side using React and server-side using Express.js.

Dashboard:

- Presents a centralized dashboard for users after login.
- Includes sidebar navigation for easy access to different sections: Home, Tasks, and Projects.
- Utilizes React hooks for state management, ensuring a responsive and interactive user interface.

Task Management:

- Enables users to add, view, and delete tasks.
- Tasks can be associated with deadlines for better organization.
- Supports real-time updates and rendering using React's virtual DOM.

Project Management:

- Allows users to create, edit, and delete projects.
- Each project can have a name and a description for clear identification.
- Projects are displayed in a list format for easy access and management.

Data Persistence:

- Utilizes localStorage for storing user tasks and projects locally in the browser.
- Ensures data persistence across sessions, providing a seamless user experience.

Responsive Design:

- Implements a responsive design using CSS frameworks like Bootstrap, ensuring compatibility with various screen sizes and devices.

- Ensures optimal user experience across desktop, tablet, and mobile devices.

Server-side Functionality:

- Utilizes Express.js for handling server-side logic and API endpoints.
- Implements RESTful APIs for user authentication, task management, and project management.
- Validates user input and handles errors gracefully to ensure a smooth user experience.

Security:

- Implements secure authentication using bcrypt for hashing passwords and protecting user credentials.
- Ensures that sensitive user data is transmitted securely over HTTPS.
- Implements proper error handling and validation to prevent common security vulnerabilities like SQL injection and cross-site scripting (XSS).

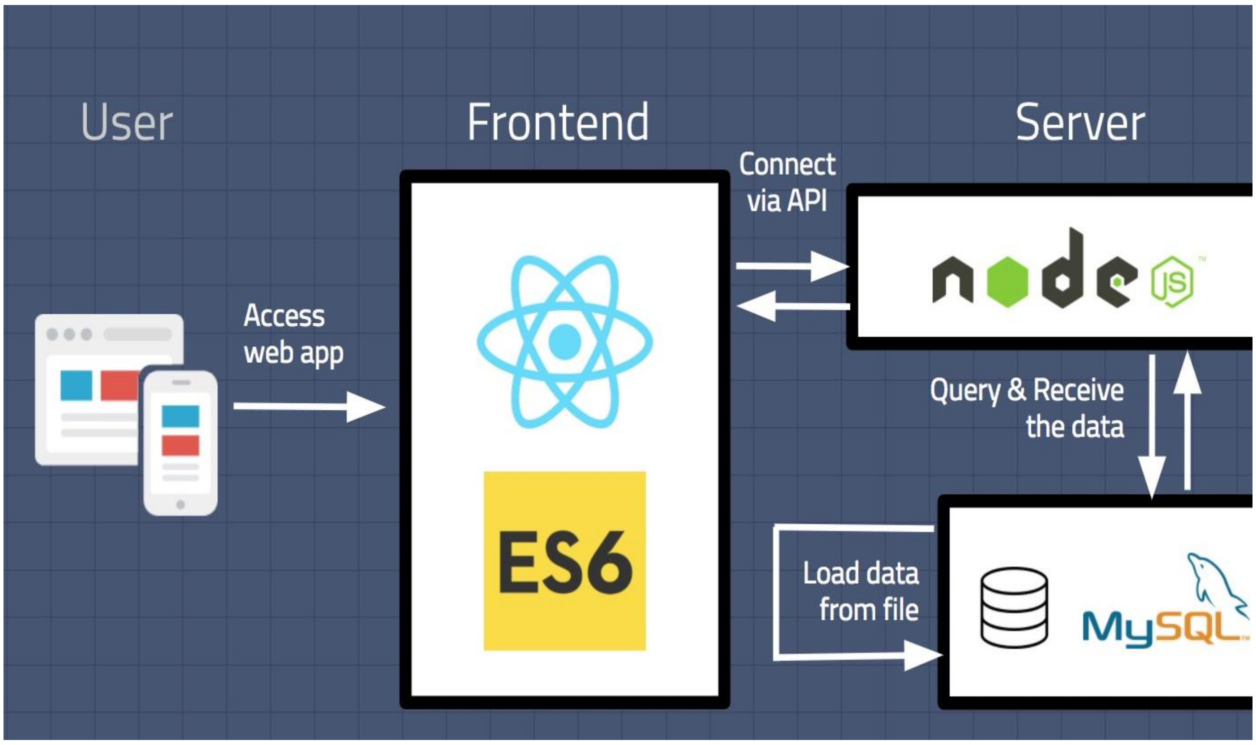
Scalability:

- Designs the application with scalability in mind, allowing for future enhancements and feature additions.
- Follows best practices in code organization and architecture to facilitate maintainability and scalability.

These features collectively provide users with a robust and user-friendly Task and Project Management application, helping them organize their tasks and projects efficiently.

4.ARCHITECTURE OVERVIEW

A task and project management system typically consists of several components that work together to help individuals and teams organize, track, and complete their work efficiently. Here's an architecture overview of such a system:



Frontend Architecture:

- Built using React.js, a popular JavaScript library for building user interfaces.

- Follows a component-based architecture, with each UI element encapsulated within reusable components.
- Utilizes React Router for client-side routing, enabling navigation between different views such as login, signup, dashboard, tasks, and projects.

Backend Architecture:

- Developed using Express.js, a minimalist web framework for Node.js.
- Implements RESTful APIs for handling client requests, including user authentication, task management, and project management.
- Utilizes middleware functions for parsing incoming requests, handling CORS (Cross-Origin Resource Sharing), and error handling.

Database Management:

- Integrates with MySQL, a popular open-source relational database management system.
- Utilizes the MySQL database to store user information, including names, emails, passwords, tasks, and projects.

- Executes SQL queries to perform CRUD (Create, Read, Update, Delete) operations on user data, ensuring efficient data management and retrieval.

User Authentication:

- Implements secure user authentication using bcrypt for password hashing and salting.
- Validates user credentials during the login process, ensuring only authenticated users gain access to the application.
- Utilizes JSON Web Tokens (JWT) for session management and user authorization, enabling secure communication between the client and server.

State Management:

- Utilizes React hooks such as useState and useEffect for managing component state and performing side effects.
- Ensures a predictable state management approach, facilitating data flow and synchronization between different components.

Client-Server Communication:

- Implements HTTP requests using the Axios library for sending data between the frontend and backend.
- Ensures secure communication over HTTPS, protecting sensitive user information during transmission.

Data Persistence:

- Utilizes localStorage for storing user tasks and projects locally in the browser.
- Enables data persistence across sessions, allowing users to access their tasks and projects even after refreshing the page or closing the browser.

Scalability and Maintainability:

- Follows best practices in code organization and architecture to ensure scalability and maintainability.
- Separates concerns between frontend and backend components, enabling independent development and scalability of each layer.
- Utilizes modular design principles, facilitating code reuse and extensibility for future enhancements.

4.1Technologies

- **Front-end:** React.js
- **Back-end:** Node.js
- **Database:** MySQL
- **Libraries and Modules**: Axios, React-icons

Our Task and Project Management application utilizes a modern tech stack for seamless functionality. React.js powers the frontend, enabling dynamic user interfaces, while React Router facilitates smooth navigation. CSS and Bootstrap ensure a visually appealing design. Express.js handles server-side logic, with MySQL for efficient data storage. Axios facilitates secure communication between frontend and backend. Together, these technologies create a robust and user-friendly platform for effective task and project management.

5.CODE

#Login.js

```
import React, { useState } from 'react';
import { Link ,useNavigate} from 'react-router-dom';
import Validation from './LoginValidation';
import axios from'axios';
import backgroundImage from './background.jpg';
import './style.css';

function Login() {
  const[values, setValues]=useState({
    email: '',
    password: ''
  })
  const navigate=useNavigate();
  const[errors,setErrors]=useState({})
  const handleInput=(event)=>{
    setValues(prev=>({...prev,[event.target.name]: event.target.value}))
  }
```

```
  const handleSubmit = (event) => {
    event.preventDefault();
    setErrors(Validation(values));
    if (errors.email === "" && errors.password === "") {
      axios.post('http://localhost:8081/login', values, { withCredentials:
true })
       .then(res => {
         navigate('/dashboard');
       })
       .catch(err => {
         console.error("Axios error:", err);
       });
    }
  };
```

```
  return (
    <div className='login-container'style={{ backgroundImage:
`url(${backgroundImage})` }}>
```

```
      <div className='container d-flex justify-content-center align-
items-center vh-100'>
        <div className='bg-white p-3 rounded w-25'>
          <h2>Sign in</h2>
          <form action="" onSubmit={handleSubmit}>
            <div className='mb-3'>
              <label htmlFor="email"><strong>Email</strong></label>
              <input type="email" placeholder='Enter email' name='email'
              onChange={handleInput} className='form-control rounded-0'/>
              {errors.email && <span className='text-
danger'>{errors.email}</span>}
            </div>
            <div className='mb-3'>
              <label htmlFor="password"><strong>Password</strong></label>
              <input type="password" placeholder='Enter password'
name='password'
              onChange={handleInput} className='form-control rounded-0'/>
              {errors.password && <span className='text-
danger'>{errors.password}</span>}
            </div>
            <button type='submit' className='btn btn-success w-100 mb-3
rounded-0'>Log in</button>
            <p>Don't have an existing account</p>
            <Link to="/signup" className='btn btn-default border w-100
bg-light rounded-0 text-decoration-none'>Create Account</Link>
          </form>
        </div>
      </div>
    </div>
  );
}
```

```
export default Login;
```

#LoginValidation.js

```
function Validation(values){
    let error={}
    const email_pattern=/^[^\s@]+@[^\s@]+\.[^\s@]+$/
    const password_pattern=/^(?=.*\d)(?=.*[a-z])(?=.*[A-Z])[a-zA-Z0-
9]{8,}$/
    if(values.email===""){
        error.email="Email should not be empty"
    }
    else if(!email_pattern.test(values.email)){
        error.email="Email did not match"
    }else{
        error.email=""
    }

    if(values.password===""){
        error.password="Password should not be empty"
    }
    else if(!password_pattern.test(values.password)){
        error.password="Password did not match"
    }else{
        error.password=""
    }
    return error;
```

```
}
```

```
export default Validation;
```

#signup.js

```
import React, { useState } from 'react';
import { Link, useNavigate } from 'react-router-dom';
import Validation from './SignupValidation';
import axios from 'axios';

function Signup() {
  const [values, setValues] = useState({
    name: '',
    email: '',
    password: ''
  });
```

```
  const navigate = useNavigate();
  const [errors, setErrors] = useState({});
```

```
  const handleInput = (event) => {
    setValues(prev => ({ ...prev, [event.target.name]:
event.target.value }));
  };
```

```
  const handleSubmit = (event) => {
    event.preventDefault();
    setErrors(Validation(values));
    if (Object.values(errors).every(error => error === '')) {
        axios.post('http://localhost:8081/signup', values)
        .then(res => {
          navigate('/');
        })
        .catch(err => {
          console.error('Error:', err);
        });

    }
  };
```

```
  return (
    <div className='d-flex justify-content-center align-items-center bg-
primary vh-100'>
      <div className='bg-white p-3 rounded w-25'>
        <h2>Sign up</h2>
        <form action="" onSubmit={handleSubmit}>
          <div className='mb-3'>
            <label htmlFor='name'><strong>Name</strong></label>
            <input type="name" placeholder='Enter name' name='name'
              onChange={handleInput} className='form-control rounded-0' />
            {errors.name && <span className='text-
danger'>{errors.name}</span>}
          </div>
          <div className='mb-3'>
            <label htmlFor='email'><strong>Email</strong></label>
            <input type="email" placeholder='Enter email' name='email'
              onChange={handleInput} className='form-control rounded-0' />
            {errors.email && <span className='text-
danger'>{errors.email}</span>}
          </div>
          <div className='mb-3'>
```

```
            <label htmlFor='password'><strong>Password</strong></label>
            <input type="password" placeholder='Enter password'
name='password'
              onChange={handleInput} className='form-control rounded-0' />
            {errors.password && <span className='text-
danger'>{errors.password}</span>}
          </div>
          <button type='submit' className='btn btn-success w-100 mb-3
rounded-0'>Sign up</button>
          <p>Already have an account</p>
          <Link to="/" className='btn btn-default border w-100 bg-light
rounded-0 text-decoration-none'>Log in</Link>
        </form>
      </div>
    </div>
  );
}
```

```
export default Signup;
```

#SignupValidation.js

```
function Validation(values){
    let error={}
    const email_pattern=/^[^\s@]+@[^\s@]+\.[^\s@]+$/
    const password_pattern=/^(?=.*\d)(?=.*[a-z])(?=.*[A-Z])[a-zA-Z0-
9]{8,}$/
    if(values.name===""){
        error.name="Name should not empty"
    }
    else{
        error.name=""
    }
    if(values.email===""){
        error.email="Email should not be empty"
    }
    else if(!email_pattern.test(values.email)){
        error.email="Email did not match"
    }else{
        error.email=""
    }

    if(values.password===""){
        error.password="Password should not be empty"
    }
    else if(!password_pattern.test(values.password)){
        error.password="Password did not match"
    }else{
        error.password=""
    }
    return error;
```

```
}
```

```
export default Validation;
```

#Dashboard.js

```
import React, { useState, useEffect } from 'react';
import './styles.css';
import { AiOutlineDelete } from 'react-icons/ai';

function Subpage() {
    const [taskText, setTaskText] = useState('');
    const [deadline, setDeadline] = useState('');
    const [tasks, setTasks] = useState([]);
    const [allProject, setProject] = useState([]);
    const [newProject, setNewProject] = useState("");
    const [newDescription, setNewDescription] = useState("");
    const [user] = useState("current_user");
```

```
    const showProfile = () => {
        document.getElementById("Profile").style.display = "block";
        document.getElementById("Tasks").style.display = "none";
        document.getElementById("Project").style.display = "none";
    };
```

```
    const showTasks = () => {
        document.getElementById("Profile").style.display = "none";
        document.getElementById("Tasks").style.display = "block";
        document.getElementById("Project").style.display = "none";
    };
```

```
    const showProject = () => {
        document.getElementById("Profile").style.display = "none";
        document.getElementById("Tasks").style.display = "none";
        document.getElementById("Project").style.display = "block";
    };
```

```
    const handleAddproject = () => {
        let newItem = {
            project: newProject,
            description: newDescription,
            user: user
        };
        let updatedArr = [...allProject]
        updatedArr.push(newItem)
        setProject(updatedArr);
        localStorage.setItem('dolist', JSON.stringify(updatedArr));
        setNewProject("");
        setNewDescription("");
    }
```

```
    const handleDelete = (index) => {
        const updatedProjectList = allProject.filter((item, idx) =>
idx !== index && item.user === user);
        setProject(updatedProjectList);
        localStorage.setItem('dolist',
JSON.stringify(updatedProjectList));
    }
```

```
    useEffect(() => {
        let savedProject = JSON.parse(localStorage.getItem('dolist'));
        if(savedProject){
            setProject(savedProject);
        }
    }, []);
```

```
    const handleTaskTextChange = (event) => {
        setTaskText(event.target.value);
    };
```

```
    const handleDeadlineChange = (event) => {
        setDeadline(event.target.value);
    };
```

```
    const addTask = () => {
        if (taskText.trim() === '' || deadline.trim() === '') {
            alert("You must enter both a task and a deadline.");
            return;
        }

        const currentDate = new Date();
        const newTask = {
            text: taskText,
            deadline: deadline,
            overdue: new Date(deadline) < currentDate,
            user: user
        };

        const updatedTasks = [...tasks, newTask];
        setTasks(updatedTasks);
        setTaskText('');
        setDeadline('');

        localStorage.setItem('tasks', JSON.stringify(updatedTasks));
    };

    const removeTask = (index) => {
        const updatedTasks = tasks.filter((task, idx) => idx !== index &&
task.user === user);
        setTasks(updatedTasks);
        localStorage.setItem('tasks', JSON.stringify(updatedTasks));
    };
```

```
    useEffect(() => {
        const savedTasks = JSON.parse(localStorage.getItem('tasks'));
        if (savedTasks) {
            setTasks(savedTasks);
        }
    }, []);

    const getTotalTasksCount = () => {
        return tasks.filter(task => task.user === user).length;
    };
```

```
    const getTotalProjectsCount = () => {
        return allProject.filter(project => project.user === user).length;
    };
```

```
    return (
        <div className="container">
            <div className="sidebar">
                <p><strong>Dashboard</strong></p>
                <button onClick={showProfile}><b>Home</b></button>
                <button onClick={showTasks}><b>Tasks</b></button>
                <button onClick={showProject}><b>Projects</b></button>
            </div>
            <div className="content">
                <div className="Profile" id="Profile">
                    <div className="box3">
                        <div className="box3-0">
                            <p><b>TASK AND PROJECT MANAGEMENT</b></p>
                        </div>
                        <div className="box3-1">
                            <div className='box3-1-1'>
                                <img src="https://www.wimi-
teamwork.com/static/medias/logiciels-gestion-des-taches-1280x640-1.png"
width={700} height={500} alt=""/>
                            </div>
                            <div className="box3-1-2">
                                <p><b>Tasks and projects are organized
according to your specifications. The total count of tasks and projects
is offered to meet your specific needs.</b></p>
                                <div className='table-container'>
                                    <table cellSpacing="10"
cellPadding="10" className="table1">
                                        <tbody>
                                            <tr>
                                                <td
id="totalTasksTitle"><b>Total Tasks</b></td>
                                                <td
id="totalProjectsTitle"><b>Total Projects</b></td>
                                            </tr>
                                            <tr>
                                                <td
id="totalTasksCount"><b>{getTotalTasksCount()}</b></td>
                                                <td
id="totalProjectsCount"><b>{getTotalProjectsCount()}</b></td>
                                            </tr>
                                        </tbody>
                                    </table>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
                <div className="Tasks" id="Tasks">
                    <div className="box2">
                        <div className="tasks-list">
                            <p><b>Your Tasks</b></p>
                            <div className="text">
                                <input type="text" value={taskText}
onChange={handleTaskTextChange} placeholder="Add your tasks" />
                                <input type="date" value={deadline}
onChange={handleDeadlineChange} placeholder="Set Deadline" />
                                <button onClick={addTask}
id="addbutton">Add</button>
                            </div>
                            <ul>
                                {tasks.map((task, index) => (
                                <li key={index} className={task.overdue ?
'overdue' : ''}>
                                    {task.text}
                                    <span onClick={() =>
removeTask(index)}>&times;</span>
                                </li>
                               ))}
                            </ul>
                        </div>
                    </div>
                </div>
                <div className="Project" id="Project">
                    <p><b>Your Projects</b></p>
                    <div className='wrapper'>
                        <div className='input1'>
                            <label><b>Project</b></label>
                            <input type="text" placeholder="Enter project
name" className='Name'
                             value={newProject}
onChange={(e)=>setNewProject(e.target.value)}/>
                        </div>
                        <div className='input1'>
```

```
                            <label><b>Description</b></label>
                            <input type="text" placeholder="Enter project
details" className='Description'
                             value={newDescription}
onChange={(e)=>setNewDescription(e.target.value)}/>
                        </div>
                        <div className='input1'>
                            <button type='button'
onClick={handleAddproject} className='button2'>Add</button>
                        </div>
                    </div>
                    <div className='list'>
                        {allProject.map((item,index) => (
                            <div className='list-item' key={index}>
                                <div>
                                    <h3>{item.project}</h3>
                                    <p>{item.description}</p>
                                </div>
                                <div>
                                    <AiOutlineDelete className='icon'
onClick={() => handleDelete(index)} title="Delete?" />
                                </div>
                            </div>
                        ))}
                    </div>
                </div>
            </div>
        </div>
    );
}
```

```
export default Subpage;
```

#server.js

```
const express = require('express');
const mysql = require('mysql');
const cors = require('cors');

const app = express();
const corsOptions = {
    origin: 'http://localhost:3000',
    credentials: true
};
app.use(cors(corsOptions));
app.options('*', cors(corsOptions));
app.use(express.json());
```

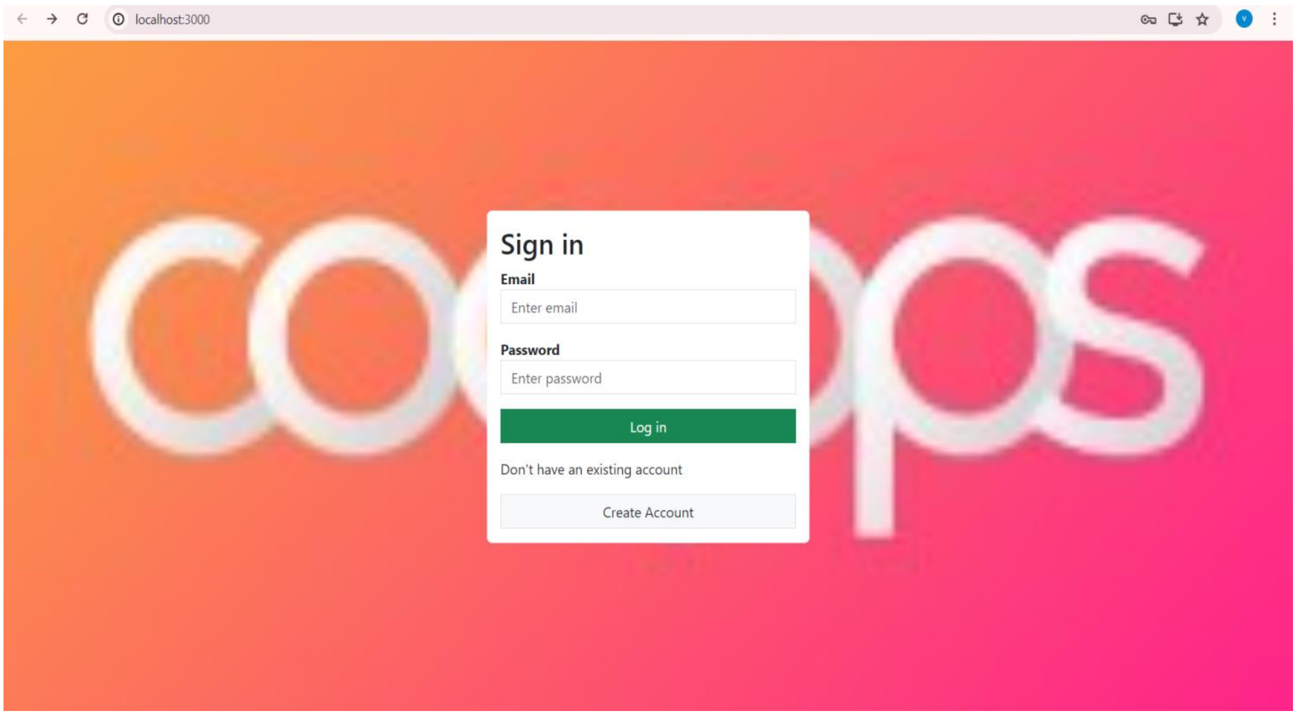
```
const db = mysql.createConnection({
    host: "localhost",
    user: "root",
    password: "",
    database: "signup"
});
```

```
app.post('/signup', (req, res) => {
    const { name, email, password } = req.body;
    const sql = "INSERT INTO login (name, email, password) VALUES
(?, ?, ?)";
    const values = [name, email, password];
    db.query(sql, values, (err, data) => {
        if (err) {
            console.error("Error occurred during signup:", err);
            return res.status(500).json({ error: "An error occurred
during signup." });
        }
        return res.json(data);
    });
});
```

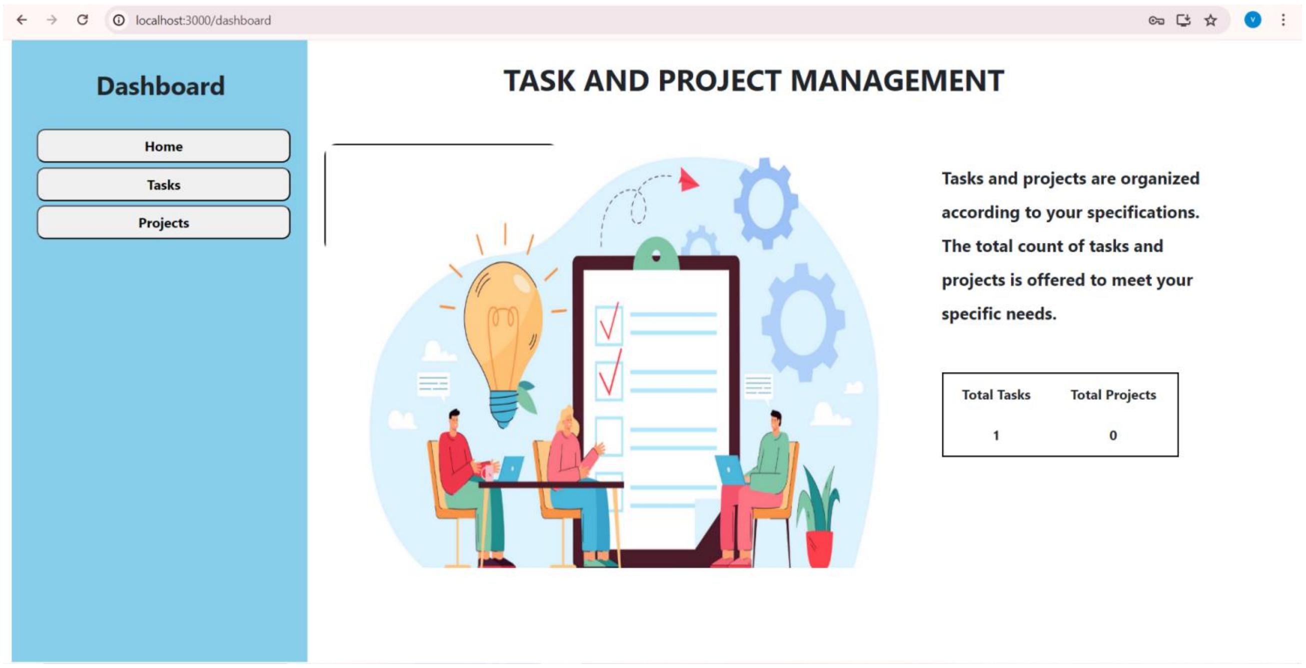
```
app.post('/login', (req, res) => {
    const { email, password } = req.body;
    const sql = "SELECT * FROM login WHERE email = ? AND password = ?";
    db.query(sql, [email, password], (err, data) => {
        if (err) {
            console.error("Error occurred during login:", err);
            return res.status(500).json({ error: "An error occurred
during login." });
        }
        if (data.length > 0) {
            return res.json({ Login:true,user:data[0]});
        }
         else {
            return res.status(401).json({ Login:false });
        }
    });
});
```

```
app.listen(8081, () => {
    console.log("listening");
});
```

6. OUTPUT



/



TASK AND PROJECT MANAGEMENT

Tasks and projects are organized according to your specifications. The total count of tasks and projects is offered to meet your specific needs.

Total Tasks	Total Projects
1	0

7.FUTURE ENHANCEMENT

Collaborative Features:

- Enable collaboration on tasks and projects, allowing multiple users to work together on shared initiatives.
- Implement real-time collaboration features such as chat, comments, and notifications to facilitate communication and coordination among team members.

User Roles and Permissions:

- Implement user roles (e.g., admin, manager, team member) with different levels of access permissions.
- Allow administrators to assign roles and permissions to users, controlling access to sensitive features and data.

Advanced Task Management:

- Introduce advanced task management features such as task prioritization, categorization, and tagging.
- Implement task dependencies and subtasks to

define relationships between tasks and track progress more effectively.

Mobile Application:

- Develop a native mobile application for iOS and Android platforms, offering users the flexibility to manage tasks and projects on the go.
- Ensure seamless synchronization with the web application, allowing users to access their data across devices.

Localization and Internationalization:

- Support multiple languages and locales to cater to a diverse user base across different regions and cultures.
- Implement localization features to adapt the application's interface and content based on the user's preferred language and locale settings.
- Provide users with the ability to restore their data from backups, ensuring continuity of operations and preserving critical information.

Task Automation:

- Introduce task automation capabilities using workflows, templates, and triggers.

- Enable users to automate repetitive tasks, schedule recurring tasks, and define custom workflows to streamline their workflows and increase efficiency.

Integration with Version Control Systems:

- Integrate with version control systems such as Git or SVN to manage code repositories and track changes made to project files.
- Allow developers to link tasks and projects to specific branches or commits, providing visibility into code changes related to project development.

8.CONCLUSION

In conclusion, the Task and Project Management application offers a robust platform for individuals and teams to efficiently organize their tasks and oversee projects. Through the seamless integration users can easily navigate through a user-friendly interface, managing tasks, and projects with ease. The application's emphasis on security, scalability, and maintainability ensures a reliable and adaptable solution for modern work environments.

Furthermore, the Task and Project Management application sets itself apart by prioritizing user experience and productivity. By providing features such as user authentication, task management, and project oversight, the application empowers users to optimize their workflows and achieve their objectives effectively. With a centralized dashboard and intuitive functionalities, the application stands as a valuable asset, simplifying task management and project coordination for individuals and teams alike.