Task Management App

Repo Link : **clcik**

Installation Guide

Prerequisites

- Ensure Node.js and npm are installed on your system.

Backend Installation

- Run **npm i** in the root directory to install backend dependencies.

Frontend Installation

- Navigate to the frontend directory using- **cd frontend.**
- Run **npm i** to install frontend dependencies.

Running the Project

Backend

- Start the backend server with **npm run dev.**

Frontend

- Open split terminal
- Navigate to the frontend directory using **cd frontend**.
- Start the frontend server with **npm start.**

Environment Variables

- Create a config.env file in the backend/config directory and include the following variables:

- PORT: Port number for the server.
- DB_URI: Database URI.
- JWT_SECRET: Secret key for JWT authentication.
- JWT_EXPIRE: Expiry time for JWT tokens.
- COOKIE_EXPIRE: Expiry time for cookies.
- SMPT_SERVICE: SMTP service for email functionality.
- SMPT_MAIL: Email for SMTP.
- SMPT_PASSWORD: Password for SMTP.
- SMPT_HOST: Host for SMTP.
- SMPT_PORT: Port for SMTP.
- CLOUDINARY_NAME: Cloudinary account name.
- CLOUDINARY_API_KEY: Cloudinary API key.
- CLOUDINARY_API_SECRET: Cloudinary API secret.

Ensure each field in config.env is filled with the respective information.

Access the Api and Frontend Routes working

To grant admin privileges to a user manually after signup, you can follow these steps:

Signup a User:

- Allow users to register through your app, capturing basic information.
- Store user details in the database, including a unique identifier.

Task Management App
Repo Link : [clcik]

-

Manually Update Role for Admin Access:

- As an admin or through a privileged backend endpoint:
    - Modify the user's role field in the database or user profile.
    - Assign the user an "admin" role manually to grant access to admin routes.
    - This can be done through a backend API endpoint specifically designed for role updates.

Admin Routes Access:

- Create backend routes that are restricted to users with an "admin" role:
    - Implement middleware to check if the user has an "admin" role before accessing these routes.
    - Example: Routes for creating teams, updating teams, assigning tasks, etc., should be accessible only to users with the "admin" role.

Frontend Integration:

- Design frontend components/pages specifically for admin functionalities.
- Implement authorization checks in the frontend to restrict access to admin-only features.
- Show/hide certain UI elements or routes based on the user's role retrieved from the backend.

API Endpoints:

- Ensure the backend API endpoints are appropriately secured and validate user roles before allowing access to admin-specific functionalities.
- Endpoints for creating teams, updating teams, assigning tasks, etc., should have authorization checks to ensure only users with the "admin" role can access them.

API EndPoint and access for user and Admin role

Here's the table format for all the routes with the added base URL `http://localhost:4002/api/v1/`:

HTTP Method	Route	Description	Authorization

Task Management App
Repo Link : clcik

POST	/api/v1/register	Register a new user	None
POST	/api/v1/login	Log in a user	None
GET	/api/v1/logout	Log out the user	None
GET	/api/v1/me	Get user details	Authenticated User
PUT	/api/v1/password/update	Update user password	Authenticated User
PUT	/api/v1/me/update	Update user profile	Authenticated User
GET	/api/v1/admin/users	Get all users (admin only)	Admin
GET	/api/v1/admin/user/:id	Get a single user (admin only)	Admin
PUT	/api/v1/admin/user/:id	Update user role (admin only)	Admin
DELETE	/api/v1/admin/user/:id	Delete a user (admin only)	Admin
GET	/api/v1/teams	Get all teams (admin only)	Admin

Task Management App
Repo Link : [clcik]

POST	/api/v1/teams	Create a team (admin only)	Admin
GET	/api/v1/teams/:id	Get a team by ID (admin only)	Authenticated User
PUT	/api/v1/teams/:id	Update a team by ID (admin only)	Admin
DELETE	/api/v1/teams/:id	Delete a team by ID (admin only)	Admin
GET	/api/v1/user/teams	Get teams by user ID	Authenticated User
POST	/api/v1/tasks	Create a task (admin only)	Admin
GET	/api/v1/tasks	Get all tasks (admin only)	Admin
PUT	/api/v1/tasks/:id	Update a task by ID (admin only)	Admin
DELETE	/api/v1/tasks/:id	Delete a task by ID (admin only)	Admin
GET	/api/v1/user/tasks	Get all tasks for all team members	Authenticated User

Task Management App
Repo Link : clcik

GET	/api/v1/user/tasks/:taskId	Get a task by ID for a user	Authenticated User
PUT	/api/v1/tasks/:taskId/completions	Update completion status for team or individual	Authenticated User
GET	/api/v1/user/notifications	Get all notifications for a user	Authenticated User
PUT	/api/v1/user/notifications/:notificationId	Update notification read status	Authenticated User
GET	/api/v1/taskCompletions	Get all task completions (admin only)	Admin
GET	/api/v1/taskCompletions/:taskCompletionId	Get a task completion by ID (admin only)	Admin

This table summarizes the HTTP methods, routes, descriptions, and required authorization levels for each endpoint in your Express API.

Task Management App
Repo Link :[clcik](#)

For Create Task on post Request
Additionally, on the backend side, we've encountered complexities. Despite this, we've added essential features on the frontend, such as search capability and automated email notifications upon task creation. However, the task creation feature isn't fully integrated on the frontend, so I recommend checking the Postman collection for any other necessary POST requests.

Data send Json:
For Individual team

```
   {
  "title": " Project Report",
  "description": "Finish the final report for the project.",
  "assignmentType": "individual",
  "assignedTo":
   {
     "_id": "656b05fe32d8483a48404522",
     "email":"test@gmail.com"   /// provide real email id for sneding email
   }
  ,
  "dueDate": "2023-12-20T23:59:59.000Z",
 " deadline":"2023-12-31T23:59:59.000Z"
}
```

For Entire team

```
{
  "title": " Project Report",
  "description": "Finish the final report for the project.",
  "assignmentType": "Team",
  "assignedTo":[
   {
     "_id": "656b05fe32d8483a48404522",
     "email":"test@gmail.com"
   },
  {
     "_id": "656b05fe32d8483a48404522",
     "email":"test@gmail.com"
   }
]
  ,
  "dueDate": "2023-12-20T23:59:59.000Z",
 " deadline":"2023-12-31T23:59:59.000Z"
}
```

Task Management App
Repo Link : **clcik**