Task Management Application

Overview

The **Task Management Application** is a robust backend solution designed to manage tasks, users, and comments efficiently. It includes modern features like JWT-based authentication, data validation, rate limiting, deduplication, and worker-thread processing for scalability.

Features

Core Functionality

User Management

- Create new users.
- Fetch all users (admin access only).
- JWT-based authentication and authorization.

Task Management

- CRUD operations for tasks.
- Task assignment to specific users.
- Task prioritization and status updates.
- Pagination and filtering for fetching tasks.

Comment Management

- Add comments to tasks.
- Link comments to specific users and tasks.

Advanced Features

Authentication

- Secure login and access management with JWT tokens.

- Role-based authorization (admin, user).

Data Validation

- Middleware to validate incoming request payloads for users, tasks, and comments.

Error Handling

- Centralized middleware for handling application errors.
- Graceful responses with meaningful error messages.

Rate Limiting

- Prevent abuse by limiting request rates per client.

Deduplication

- Ensure unique processing of tasks or requests.

LoggerService

- Logger Service for developers to debug the issues development level

Tech Stack

- **Backend Framework**: Node.js with Express.js
- **Database**: MongoDB (Mongoose ORM)
- **Authentication**: JSON Web Tokens (JWT)
- **Task Processing**: Worker Threads
- **Rate Limiting**: express-rate-limit
- **Environment Configuration**: dotenv

Folder Structure

```
project-root/
|-- config/
|   |-- config.js
|
|-- controllers/
|   |-- commentController.js
|   |-- taskController.js
|   |-- userController.js
|
|-- logs/
|
|-- middlewares/
|   |-- authMiddleware.js
|   |-- authorize.js
|   |-- deduplicator.js
|   |-- errorHandler.js
|   |-- rateLimiter.js
|   |-- validateComment.js
|   |-- validateTask.js
|   |-- validateUser.js
|
|-- models/
|   |-- comment.js
|   |-- task.js
|   |-- user.js
|
|-- routes/
|   |-- commentRoutes.js
|   |-- taskRoutes.js
|   |-- userRoutes.js
|
|-- utils/
|   |-- logger.js
|
|-- workers/
|   |-- taskWorker.js
|
|-- app.js
|-- server.js
```

```
|--. env
|-- package.json
```

API Endpoints

User Endpoints

1)POST /<u>api/users/createNewRegistrationForUser</u> - Register a new user.

<u>User Role Access Token:</u>

```
curl --location 'https://task-management-app-fgw3.onrender.com/api/users/createNewRegistrationForUser' \
--header 'Content-Type: application/json' \
--data-raw '{
   "username":"AkhilAnem123",
   "email": "akhilanem@gmail.com",
   "password": "Anem@1245",
   "role": "user",
   "isActive": true
}
'
```

<u>Admin Role Access Token:</u>

```
curl --location 'https://task-management-app-fgw3.onrender.com/api/users/createNewRegistrationForUser' \
--header 'Content-Type: application/json' \
--data-raw '{
   "username":"Akhil123",
   "email": "akhil1245@gmail.com",
   "password": "Anem@1245",
   "role": "admin",
   "isActive": true
}
```

2)POST <u>/api/users/generateTokenForLoggedInUser</u> - Login and receive a JWT

```
curl --location 'https://task-management-app-fgw3.onrender.com/api/users/generateTokenForLoggedInUser' \
    --header 'Content-Type: application/json' \
    --data-raw '{
```

```
  "email": "akhilanem@gmail.com",
  "password": "Anem@1245"
}'
```

3)GET /api/users/getAllUsers - Fetch all users (admin-only, withOut pagination).

```
curl --location 'https://task-management-app-
fgw3.onrender.com/api/users/getAllUsers' \
--header 'Authorization: Bearer  AUTH_TOKEN'
```

4)GET /api/users/getAllUsersWithPagination - Fetch all users (admin-only, with pagination).

```
curl --location 'https://task-management-app-
fgw3.onrender.com/api/users/getAllUsersWithPagination?page=1&limit=10' \
--header 'Authorization: Bearer  AUTH_TOKEN '
```

Task Endpoints

- **POST /api/tasks/createNewUserTask** - Create a new task.

```
curl --location 'https://task-management-app-
fgw3.onrender.com/api/tasks/createNewUserTask' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer  AUTH_TOKEN' \
--data '{
"title": "Akhil Daily Tasks",
"description": "Implement the external api integration",
"dueDate": "2024-12-31",
"priority": "High",
"status": "Pending",
"assignedTo": "6772ad2c3f35a7175d57e9ad"
}'
```

- **PUT /api/tasks/updateTask/:id** - Update task details.

```
curl --location --request PUT 'https://task-management-app-
fgw3.onrender.com/api/tasks/updateTask/6772d18f764e47ac4173348c' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer  AUTH_TOKEN' \
--data '{
 "status": "In Progress"
```

}'

- **DELETE /api/tasks/deleteTask/:id** - Delete a task.
  curl --location --request DELETE 'https://task-management-app-fgw3.onrender.com/api/tasks/deleteTask/67729a346368c2a1d376f166' \
  --header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY3NzE3ZDI4MzNmODAxZjNiMTk3ODJkNyIsImVtYWlsIjoibmlra2lob2xsaWFAZ21haWwuY29tIiwicm9sZSI6InVzZXIiLCJpYXQiOjE3MzU1MDEyNTgsImV4cCI6MTczNTUwODQ1OH0.7DwmyM0aRmUAoMHQDqz01sSn1Erb521v_jsLop6pnOo'

- **GET /api/tasks/getAllTasksWithPagination** - Fetch all tasks (supports pagination and filtering).
  curl --location 'https://task-management-app-fgw3.onrender.com/api/tasks/getAllTasksWithPagination?page=1&limit=10' \
  --header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY3NzJjZjYyNzY0ZTQ3YWM0MTczMzQ4NyIsImVtYWlsIjoiYWtoaWxhbmVtQGdtYWlsLmNvbSIsInJvbGUiOiJ1c2VyIiwiaWF0IjoxNzM1NTc3NTc1LCJleHAiOjE3MzU1ODQ3NzV9.2buUoqUwQKvhO-3X8B9iJwg053QbSo5DuAFrFKhOZVg'

- **GET /api/tasks/getTasksWithCommentsById/:id** - Fetch a specific task by ID.
  curl --location 'https://task-management-app-fgw3.onrender.com/api/tasks/getTasksWithCommentsById/6772d18f764e47ac4173348c?page=2&limit=5' \
  --header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY3NzJjZjYyNzY0ZTQ3YWM0MTczMzQ4NyIsImVtYWlsIjoiYWtoaWxhbmVtQGdtYWlsLmNvbSIsInJvbGUiOiJ1c2VyIiwiaWF0IjoxNzM1NTc3NTc1LCJleHAiOjE3MzU1ODQ3NzV9.2buUoqUwQKvhO-3X8B9iJwg053QbSo5DuAFrFKhOZVg' \
  --header 'Content-Type: application/json'

Comment Endpoints

- **POST /api/comments/addCommentToTask/:id** - Add a new comment to a task.
  curl --location 'https://task-management-app-fgw3.onrender.com/api/comments/addCommentToTask/6772d18f764e47ac4173348c' \
  --header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY3NzJjZjYyNzY0ZTQ3YWM0MTczMzQ4NyIsImVtYWlsIjoiYWtoaWxhbmVtQGdtYWlsLmNvbSIsInJvbGUiOiJ1c2V

```
yIiwiaWF0IjoxNzM1NTc3NTc1LCJleHAiOjE3MzU1ODQ3NzV9.2buUoqUwQKvhO
-3X8B9iJwg053QbSo5DuAFrFKhOZVg' \
--header 'Content-Type: application/json' \
--data '{
  "content": "Write a clean code for worker pool"
}'
```

Installation

Prerequisites

- **Node.js**
- **MongoDB**

Steps

1. Clone the repository:

```
git clone https://github.com/NanduPastham123/task_management_app
cd task-management-app
```

2. Install dependencies:

```
npm install
```

3. Configure environment variables in a `.env` file:

```
PORT=3000
MONGO_URI=<your-mongodb-connection-string>
JWT_SECRET=<your-secret-key>
```

4. Start the application:

```
npm start
```

Usage

Authentication

1. Register a new user using `/api/users/register`.
2. Login to receive a JWT.
3. Pass the JWT in the `Authorization` header for protected routes.

Task and Comment Management

- Use task and comment endpoints to manage tasks and add comments.
- Filter tasks by priority or status, and paginate results.

Deployment Steps for Task Management API

This document outlines the steps to deploy the Task Management API using **Render** and **GitHub**.

Prerequisites

Before deploying, ensure you have:

1. A **GitHub** repository with your Task Management API code.
2. A **Render** account (Sign up here).
3. A **MongoDB** database (e.g., using **MongoDB Atlas**).

Step 1: Prepare Your Project

1.1 Ensure Code Readiness

- Verify your `package.json` file includes all necessary dependencies and scripts. Add a `start` script if not already present:

```
"scripts": {
  "start": "node server.js"
}
```

- Ensure your `.env` file contains:
  - `PORT` (e.g., `3000`)
  - `MONGO_URI` (MongoDB connection string)
  - `JWT_SECRET` (secret key for JWT authentication)

1.2 Push Your Code to GitHub

- If not already done, initialize a Git repository and push the code: `git init`

```
git add .
git commit -m "Initial commit for Render deployment"
git branch -M main
git remote add origin <your-github-repository-url>
git push -u origin main
```

Step 2: Set Up MongoDB

2.1 Create a MongoDB Instance

- Use **MongoDB Atlas** or any other provider to create a database.
- Whitelist your IP or allow all IPs for testing.
- Copy the connection string (e.g., `mongodb+srv://<username>:<password>@cluster.mongodb.net/mydb`).

Step 3: Deploy on Render

3.1 Sign Up/Log In

- Go to Render and sign up or log in.

3.2 Create a New Web Service

1. Click on **New +** > **Web Service**.
2. Connect your GitHub account and select your repository.

3.3 Configure the Service

- **Name**: Enter a name for your service.
- **Branch**: Select the branch to deploy (e.g., `main`).
- **Build Command**: Set to: `npm install`
- **Start Command**: Set to: `npm start`
- **Environment**: Select **Node**.

3.4 Add Environment Variables

- Add the following environment variables:
  - `PORT`: `3000`
  - `MONGO_URI`: Your MongoDB connection string.
  - `JWT_SECRET`: Your secret key for JWT authentication.

3.5 Deploy

- Click **Create Web Service**. Render will build and deploy your application.

Step 4: Test the Deployed API

4.1 Access Your API

- Once deployed, Render provides a URL like `https://your-app-name.onrender.com`.
- Use tools like **Postman** or **cURL** to test the endpoints.

4.2 Verify Functionality

- Test key features like user registration, login, task creation, and comment addition.

Conclusion

By following these steps, your Task Management API will be deployed and accessible for testing via the provided Render URL.