TEAM TASK BOARD

Requirement:

Build a simple task board for a small team to track work.

Description

You’ve joined a 5-person product team at a small company. They currently manage tasks in a messy spreadsheet and a group chat. People miss deadlines because they don’t know what’s urgent, and there’s no shared view of progress. Your job is to give them a clean board they can open in a browser and immediately see who’s doing what, what’s due soon, and where things are stuck. When someone creates a task, they should be able to set a title, description, priority, assignee, and due date. The task appears under “Backlog.” As work starts, they drag it to “In Progress,” then “Review,” then “Done.” If a due date is approaching within 24 hours, the task should show a badge that warns the team it’s at risk. If the due date passes and it’s not done, it shows as overdue. Team members can click a task to read details and leave short comments like “blocked on design” or “waiting for access.” Filters help them narrow to “only my tasks” or “only High priority.” This app doesn’t need fancy animations; it must be reliable, easy to read, and safe. After a page refresh, everything should be there. A new team member should be able to sign up, log in, and use it in minutes.

Frontend:

- Board with 4 columns: Backlog, In Progress, Review, Done.
- Cards show title, priority (Low/Medium/High), assignee, due date, and a status badge (On Track, At Risk, Overdue).
- Click a card to open details (description, comments).
- Filters: by assignee and priority.

Backend:

- User login (email + password).
- APIs to create, read, update, delete tasks; move a task between columns; add comments.
- Badge logic:

    On Track: due date more than 24h away. At Risk: due date within 24h. Overdue: past due and not Done.

Database:

- Users: id, email, passwordHash.
- Tasks: id, title, description, priority, assigneeId, status, dueDate, created, updated.
- Comments: id, taskId, authorId, body, created.

Hosting:

- Deployed frontend and backend with working DB

TEAM TASK BOARD

Features:

The team task board is the tool which helps to manage the task of that particular team efficiently. It stores the user information and also their respective assign task. It has four board such as Backlog, In Progress, Review, Done. Then according to completion of work the developer can drag and drop that task to other board for further action. It makes the easy understand to the developer for the further action. It also simplify the searching by filter and to make the task more efficiently to get the feedback or requirements it has the comment section. It stores all the team members comments for a particular task.

Tech Stack Used:

Frontend : React.js, TailwindCSS

Backend :   Node.js, Express.js

Database : MongoDB

Authentication : Jsonwebtoken, bcrypt

Database Schemas:

Users Collection:

It stores the data of the team members with email , password and give access to the user whose data are save in the database.

Tasks Collection:

It stores all the tasks assigned to a particular user, with a reference to the users collection, so we can know which task is assigned to which user. It have the detailed data of the task which includes the task name, description, priority, assigneeId, status and duedate.

Comments Collection:

It stores all comments made by users on a particular task. Each comment includes fields such as such as taskId, authored, body, created at.

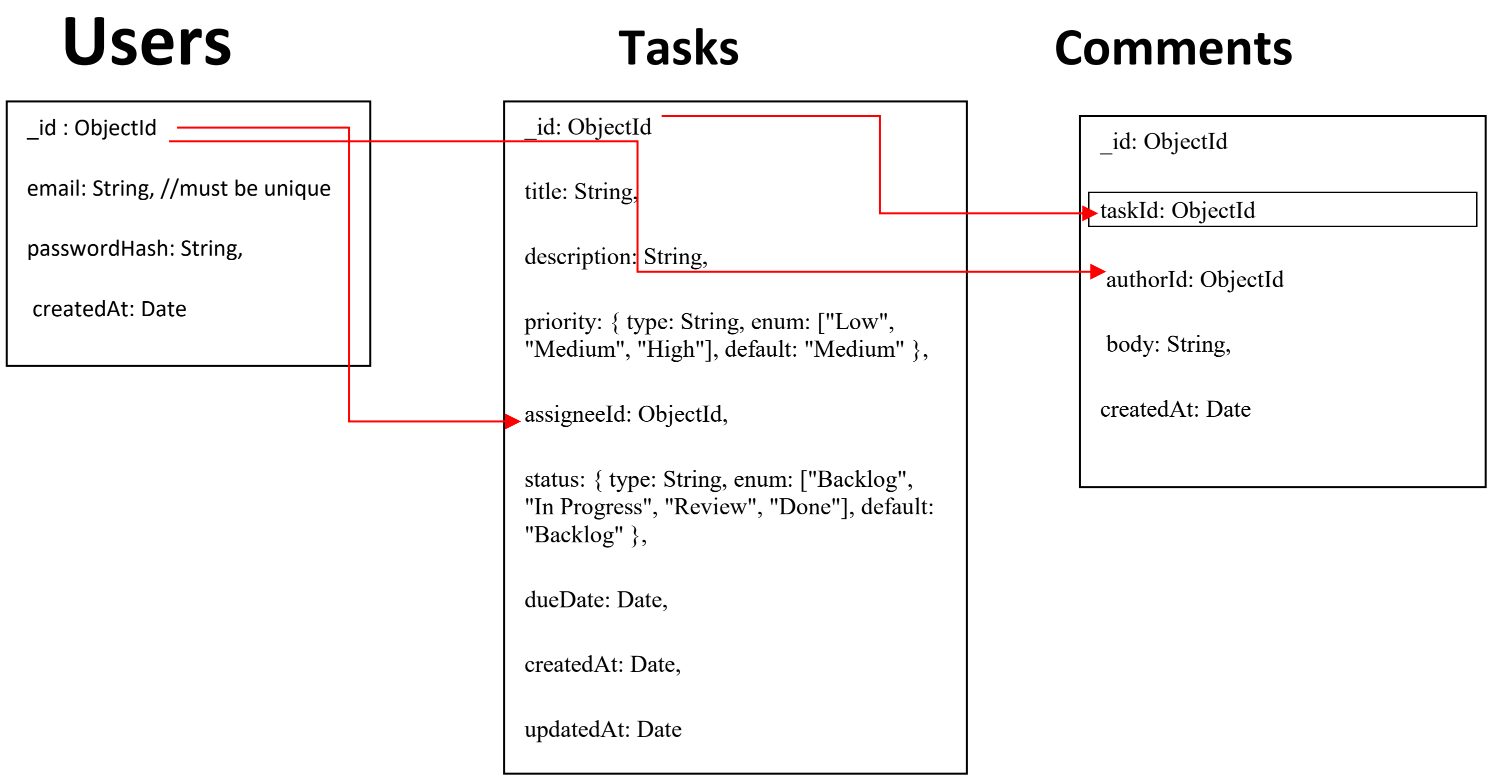
Hosting URL:

Frontend URL : **https://radiant-yeot-e18c90.netlify.app/**

Backend URL :   **https://taskboard-0qzt.onrender.com**

TEAM TASK BOARD

Database Design:

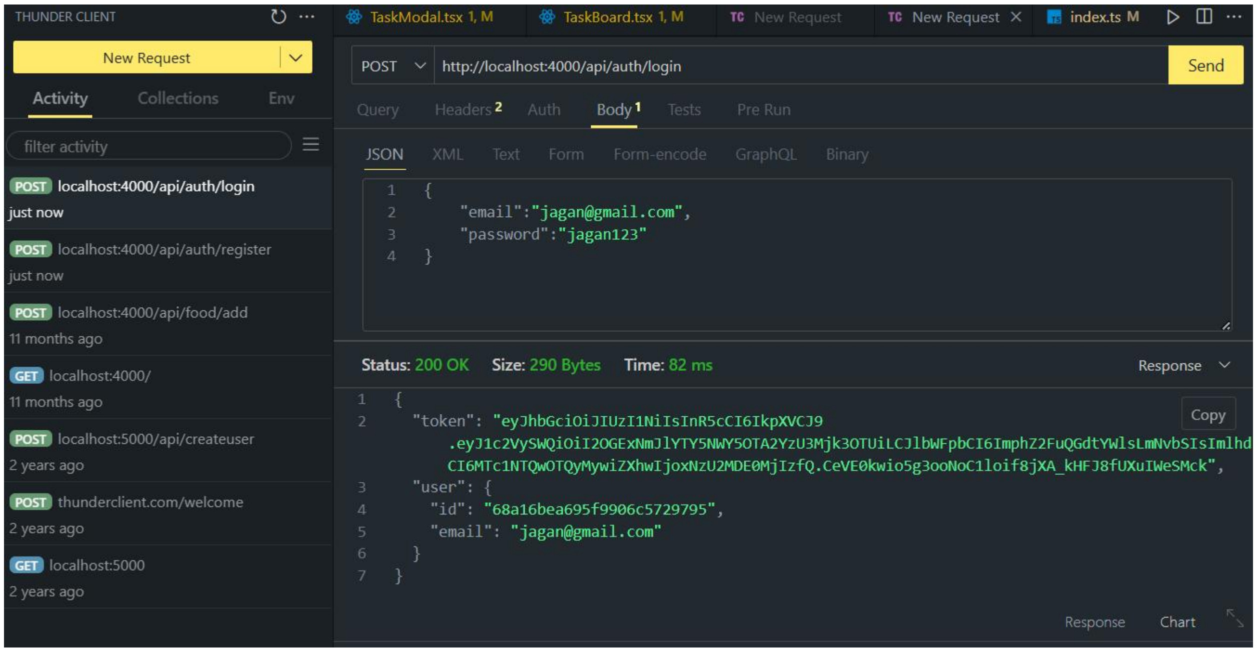


TEAM TASK BOARD

API EndPoints:

localhost End Point: POST : http://localhost:4000/api/auth/login

render End Point: POST : https://taskboard-0qzt.onrender.com/ (Change in endpoint to test in render)


THUNDER CLIENT
New Request
Activity
Collections
Env
filter activity
POST localhost:4000/api/auth/login just now
POST localhost:4000/api/auth/register just now
POST localhost:4000/api/food/add 11 months ago
GET localhost:4000/ 11 months ago
POST localhost:5000/api/createuser 2 years ago
POST thunderclient.com/welcome 2 years ago
GET localhost:5000 2 years ago
TaskModal.tsx 1, M
TaskBoard.tsx 1, M
New Request
New Request
index.ts M
Send
Query
Headers 2
Auth
Body 1
Tests
Pre Run
JSON
XML
Text
Form
Form-encode
GraphQL
Binary
Response
Copy
Chart

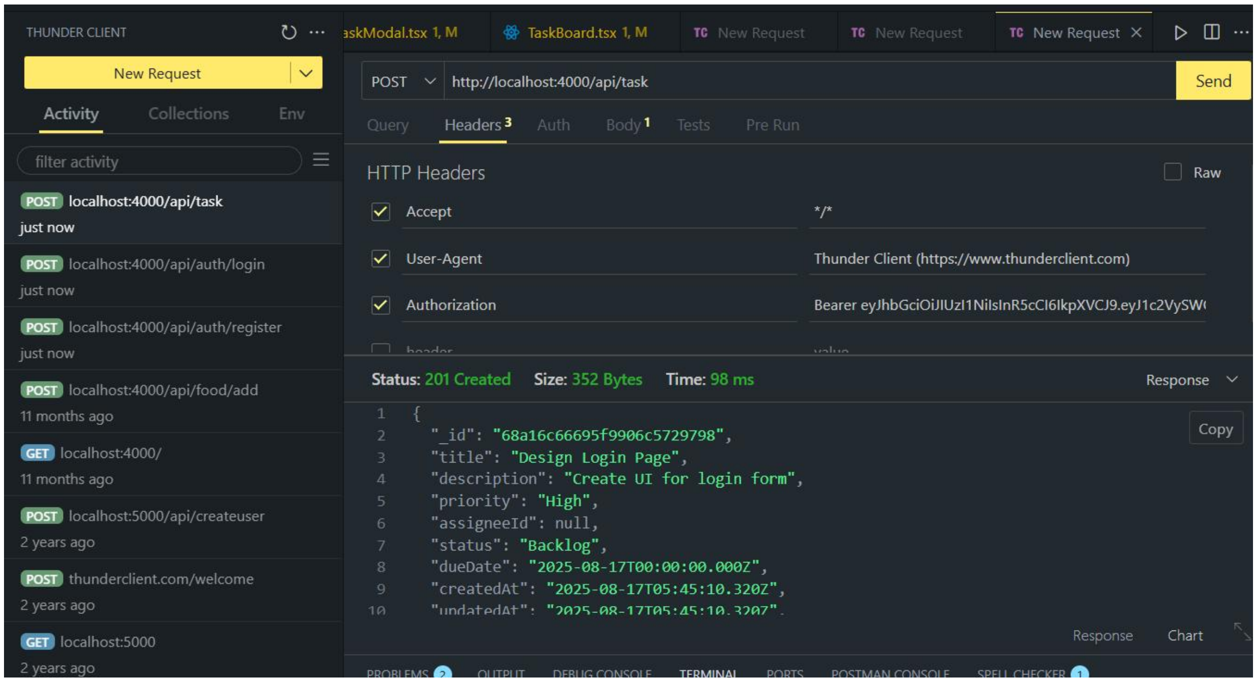
POST http://localhost:4000/api/auth/login

```
{
    "email":"jagan@gmail.com",
    "password":"jagan123"
}
```

Status: 200 OK Size: 290 Bytes Time: 82 ms

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
      .eyJ1c2VySWQiOiI2OGExNmJlYTY5NWY5OTA2YzU3Mjk3OTUiLCJlbWFpbCI6ImphZ2FuQGdtYWlsLmNvbSIsImlhd
      CI6MTc1NTQwOTQyMywiZXhwIjoxNzU2MDE0MjIzfQ.CeVE0kwio5g3ooNoC1loif8jXA_kHFJ8fUXuIWeSMck",
  "user": {
    "id": "68a16bea695f9906c5729795",
    "email": "jagan@gmail.com"
  }
}
```

POST : http://localhost:4000/api/task


THUNDER CLIENT
New Request
Activity
Collections
Env
filter activity
POST localhost:4000/api/task just now
POST localhost:4000/api/auth/login just now
POST localhost:4000/api/auth/register just now
POST localhost:4000/api/food/add 11 months ago
GET localhost:4000/ 11 months ago
POST localhost:5000/api/createuser 2 years ago
POST thunderclient.com/welcome 2 years ago
GET localhost:5000 2 years ago
askModal.tsx 1, M
TaskBoard.tsx 1, M
New Request
New Request
New Request
Send
Query
Headers 3
Auth
Body 1
Tests
Pre Run
Raw
Response
Copy
Chart
PROBLEMS 2
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
POSTMAN CONSOLE
SPELL CHECKER 1

POST http://localhost:4000/api/task

HTTP Headers

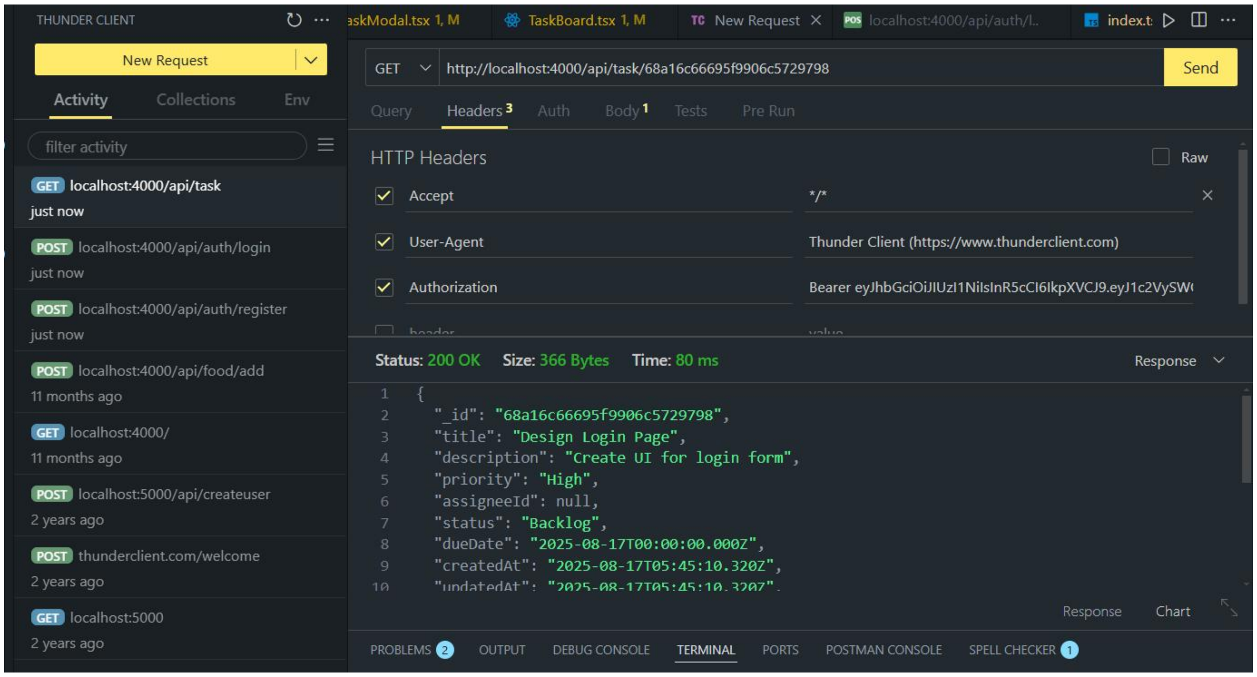
✓ Accept	*/*
✓ User-Agent	Thunder Client (https://www.thunderclient.com)
✓ Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySW
header	value

Status: 201 Created Size: 352 Bytes Time: 98 ms

```
{
  "_id": "68a16c66695f9906c5729798",
  "title": "Design Login Page",
  "description": "Create UI for login form",
  "priority": "High",
  "assigneeId": null,
  "status": "Backlog",
  "dueDate": "2025-08-17T00:00:00.000Z",
  "createdAt": "2025-08-17T05:45:10.320Z",
  "updatedAt": "2025-08-17T05:45:10.320Z",
```

TEAM TASK BOARD

GET : http://localhost:4000/api/task/ 68a16c66695f9906c5729798



THUNDER CLIENT

New Request

Activity Collections Env

filter activity

- GET localhost:4000/api/task — just now
- POST localhost:4000/api/auth/login — just now
- POST localhost:4000/api/auth/register — just now
- POST localhost:4000/api/food/add — 11 months ago
- GET localhost:4000/ — 11 months ago
- POST localhost:5000/api/createuser — 2 years ago
- POST thunderclient.com/welcome — 2 years ago
- GET localhost:5000 — 2 years ago

GET http://localhost:4000/api/task/68a16c66695f9906c5729798 Send

Query Headers 3 Auth Body 1 Tests Pre Run

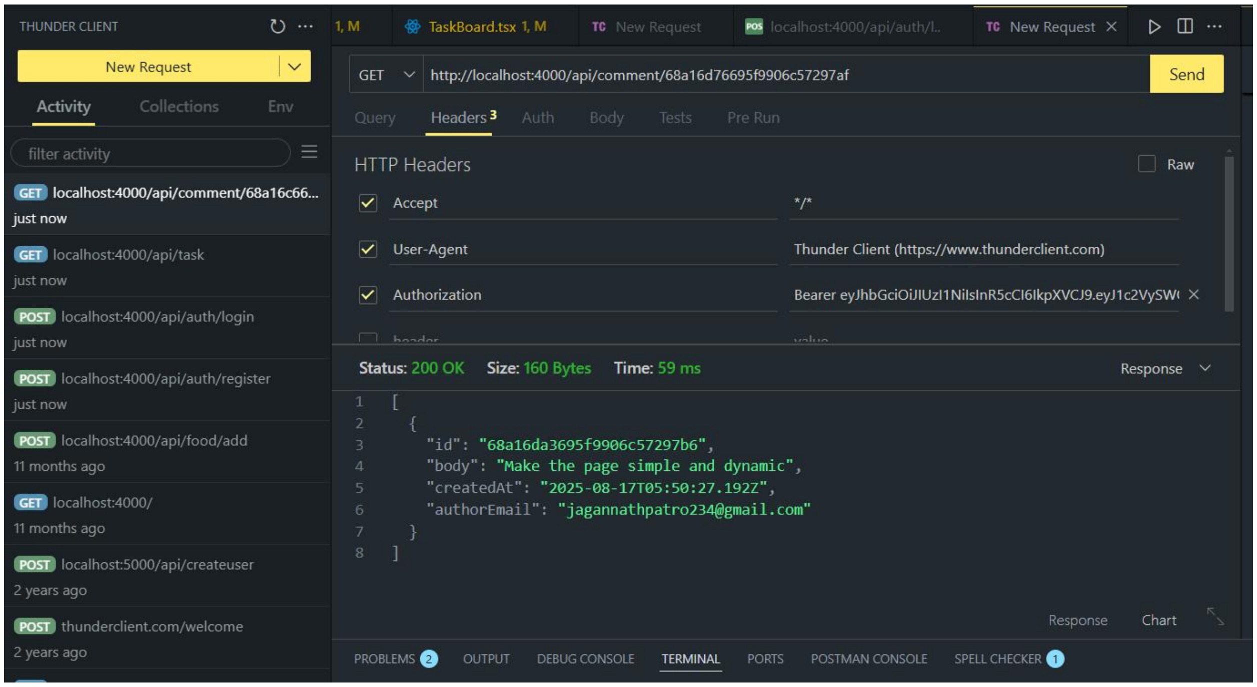
HTTP Headers

Accept	*/*
User-Agent	Thunder Client (https://www.thunderclient.com)
Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySW

Status: 200 OK Size: 366 Bytes Time: 80 ms

```
{
  "_id": "68a16c66695f9906c5729798",
  "title": "Design Login Page",
  "description": "Create UI for login form",
  "priority": "High",
  "assigneeId": null,
  "status": "Backlog",
  "dueDate": "2025-08-17T00:00:00.000Z",
  "createdAt": "2025-08-17T05:45:10.320Z",
  "updatedAt": "2025-08-17T05:45:10.320Z",
```

GET : http://localhost:4000/api/comment/68a03adb20364ada17c1366f



THUNDER CLIENT

New Request

Activity Collections Env

filter activity

- GET localhost:4000/api/comment/68a16c66... — just now
- GET localhost:4000/api/task — just now
- POST localhost:4000/api/auth/login — just now
- POST localhost:4000/api/auth/register — just now
- POST localhost:4000/api/food/add — 11 months ago
- GET localhost:4000/ — 11 months ago
- POST localhost:5000/api/createuser — 2 years ago
- POST thunderclient.com/welcome — 2 years ago

GET http://localhost:4000/api/comment/68a16d76695f9906c57297af Send

Query Headers 3 Auth Body Tests Pre Run

HTTP Headers

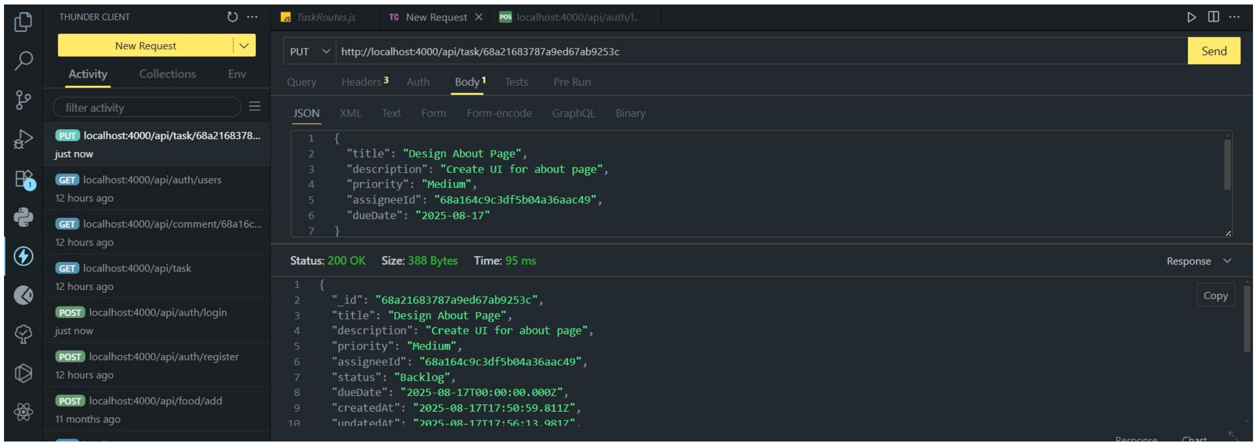
Accept	*/*
User-Agent	Thunder Client (https://www.thunderclient.com)
Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySW

Status: 200 OK Size: 160 Bytes Time: 59 ms

```
[
  {
    "id": "68a16da3695f9906c57297b6",
    "body": "Make the page simple and dynamic",
    "createdAt": "2025-08-17T05:50:27.192Z",
    "authorEmail": "jagannathpatro234@gmail.com"
  }
]
```

TEAM TASK BOARD

PUT : http://localhost:4000/api/task/68a21683787a9ed67ab9253c



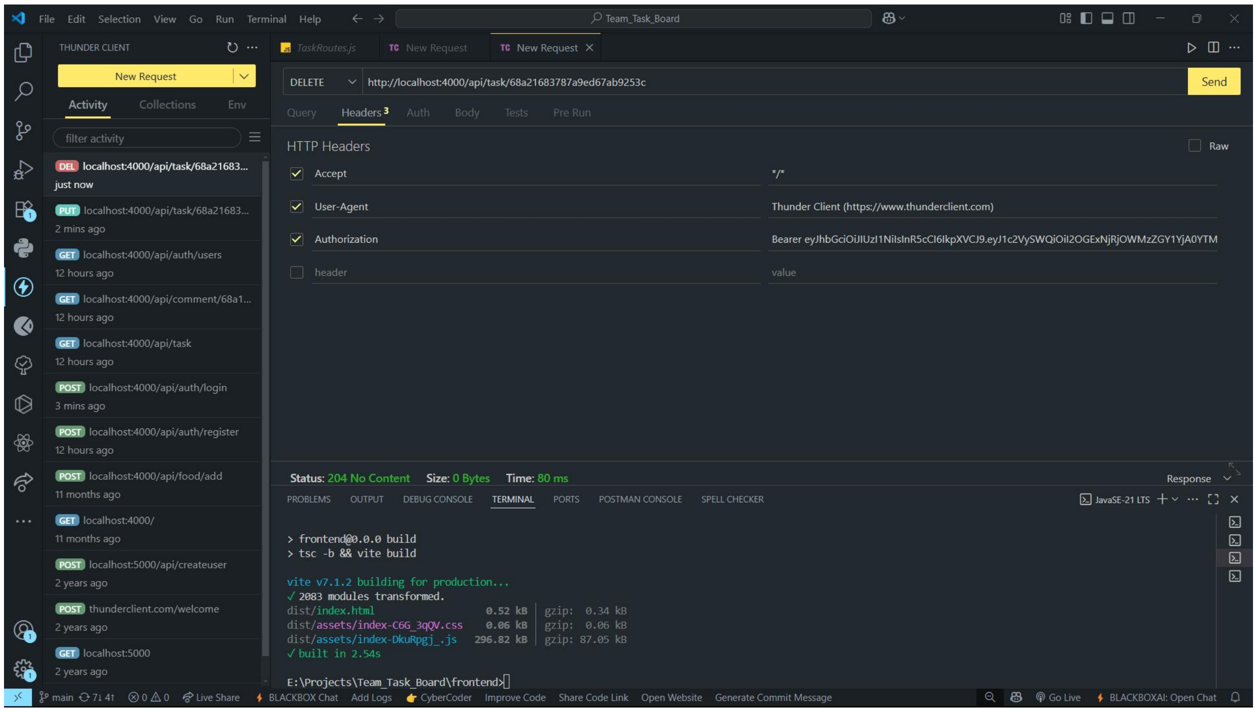
PUT http://localhost:4000/api/task/68a21683787a9ed67ab9253c

```
{
  "title": "Design About Page",
  "description": "Create UI for about page",
  "priority": "Medium",
  "assigneeId": "68a164c9c3df5b04a36aac49",
  "dueDate": "2025-08-17"
}
```

Status: 200 OK Size: 388 Bytes Time: 95 ms

```
{
  "_id": "68a21683787a9ed67ab9253c",
  "title": "Design About Page",
  "description": "Create UI for about page",
  "priority": "Medium",
  "assigneeId": "68a164c9c3df5b04a36aac49",
  "status": "Backlog",
  "dueDate": "2025-08-17T00:00:00.000Z",
  "createdAt": "2025-08-17T17:50:59.811Z",
  "updatedAt": "2025-08-17T17:56:13.981Z",
```

DELETE : http://localhost:4000/api/task/68a21683787a9ed67ab9253c



DELETE http://localhost:4000/api/task/68a21683787a9ed67ab9253c

HTTP Headers

Accept	*/*
User-Agent	Thunder Client (https://www.thunderclient.com)
Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiI2OGExNjRjOWMzZGY1YjA0YTM

Status: 204 No Content Size: 0 Bytes Time: 80 ms

```
> frontend@0.0.0 build
> tsc -b && vite build

vite v7.1.2 building for production...
✓ 2083 modules transformed.
dist/index.html                   0.52 kB │ gzip:  0.34 kB
dist/assets/index-C6G_3qQV.css    0.06 kB │ gzip:  0.06 kB
dist/assets/index-DkuRpgj_.js   296.82 kB │ gzip: 87.05 kB
✓ built in 2.54s

E:\Projects\Team_Task_Board\frontend>
```

TEAM TASK BOARD

Status Code Used:

All API responses return JSON format with appropriate HTTP status codes:

- `200` - Success
- `201` - Created
- `204` - No Content
- `400` - Bad Request
- `401` - Unauthorized
- `403` - Forbidden
- `404` - Not Found
- `500` - Internal Server Error

CORS:

The API allows cross-origin requests from all origins `*` for development. In production, configure CORS to allow only frontend domain.

Security Considerations:

- HTTPS used in production.
- Store JWT tokens securely.
- Implement proper password hashing using bcrypt with salt rounds.
- Add input validation.
- Consider implementing refresh tokens for enhanced security