INTER IIT TECH MEET 14.0
DEVELOPMENT TEAM SELECTION TASK

Submission Deadline: ***Oct 17, 2025 11:59 PM***

1. Problem Statement

A nested commenting system forms the backbone of many interactive digital platforms, enabling structured discussions, knowledge exchange, and community-driven engagement. A well-implemented comment hierarchy ensures clarity, readability, and context retention within conversations.

Your objective is to design and implement a fully functional commenting interface that visually represents comment relationships in a nested structure. Each comment should support replies, and those replies can, in turn, have their own replies, forming multiple hierarchical levels. The interface should remain intuitive and clean, even with deeply nested threads.

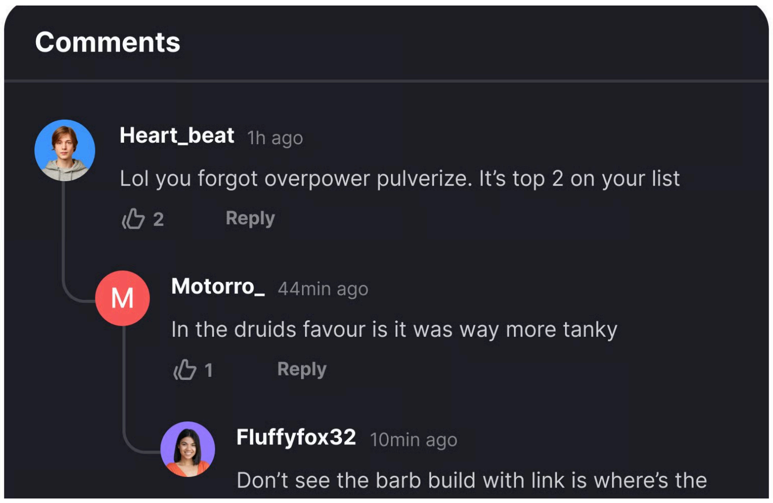
The interface should display a single post (this can be an image, article, or simple placeholder text). Below it, users should be able to see:

- Top-level comments directly under the post.
- Replies to any existing comment, creating a nested thread structure.
- Upvotes for comments to indicate popularity or relevance.
- Collapse and Open the comment thread.

2. Frontend Requirements

1. **Displaying a post or an article:** The post for which the comments would be visible.
2. **Displaying the comments for the article:** In a nested comment thread like structure

    Use this only as a reference point to understand the desired nesting. You are strongly encouraged to design your own interface adhering to other requirements and , focusing on originality and user experience.



Comments

Heart_beat 1h ago
Lol you forgot overpower pulverize. It's top 2 on your list
2 Reply

Motorro_ 44min ago
In the druids favour is it was way more tanky
1 Reply

Fluffyfox32 10min ago
Don't see the barb build with link is where's the

3. **Basic Auth Page:** Create a login page, which can have either some predetermined credentials or a condition on the email domain to login to the web page, to prevent access to restricted content.
4. **UI & Responsiveness:**
    a. Nested replies should visually indent to indicate hierarchy.
    b. Interface should remain readable on desktop, tablet, and mobile screens.
5. **The items in the dataset look like:**
    a) Users: Stores the user details.
    b) Comments: Stores the comment's details.
    c) Data Structure:

        I. User: Represents the hierarchy of the comments.

        - Attributes: id, name, avatar, created_at

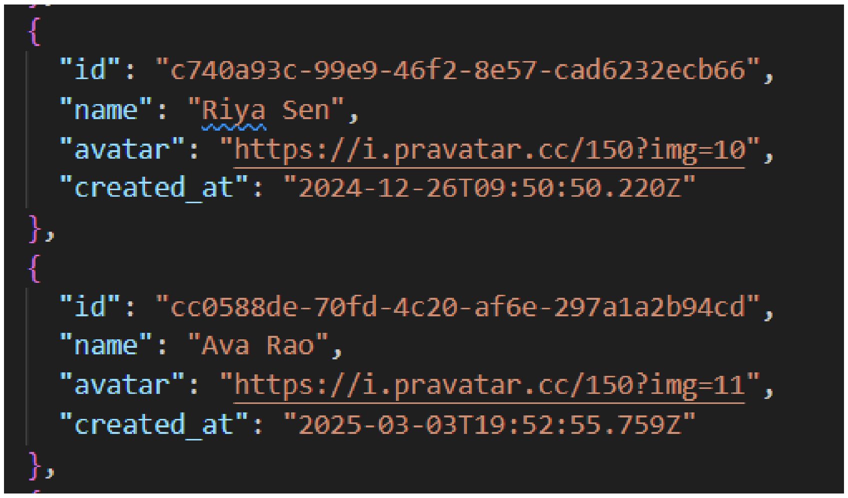
        II. Comments: Represents comment.

        - Attributes: id, text, upvotes, created_at, user_id

    d) **Links to download the dataset is here:** [Dataset]

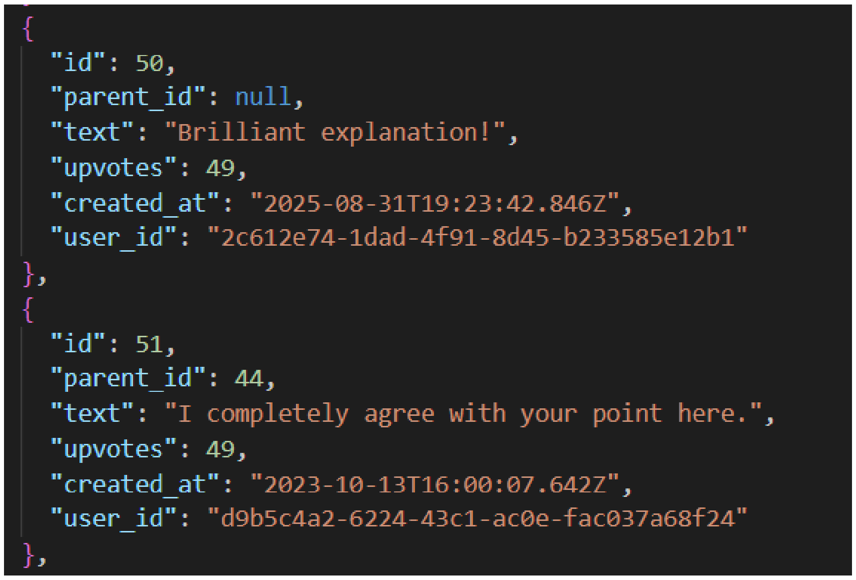
e) Use a simple JSON structure as provided below:

Users Example:



```
{
  "id": "c740a93c-99e9-46f2-8e57-cad6232ecb66",
  "name": "Riya Sen",
  "avatar": "https://i.pravatar.cc/150?img=10",
  "created_at": "2024-12-26T09:50:50.220Z"
},
{
  "id": "cc0588de-70fd-4c20-af6e-297a1a2b94cd",
  "name": "Ava Rao",
  "avatar": "https://i.pravatar.cc/150?img=11",
  "created_at": "2025-03-03T19:52:55.759Z"
},
```

Comments Example:



```
{
  "id": 50,
  "parent_id": null,
  "text": "Brilliant explanation!",
  "upvotes": 49,
  "created_at": "2025-08-31T19:23:42.846Z",
  "user_id": "2c612e74-1dad-4f91-8d45-b233585e12b1"
},
{
  "id": 51,
  "parent_id": 44,
  "text": "I completely agree with your point here.",
  "upvotes": 49,
  "created_at": "2023-10-13T16:00:07.642Z",
  "user_id": "d9b5c4a2-6224-43c1-ac0e-fac037a68f24"
},
```

3. Backend Requirements

1. **API Endpoints:** Use your preferred backend stack to expose simple REST endpoints according to the data given to you and document them.
2. **Database:** Use databases (SQLite, PostgreSQL, or MongoDB) to store locations and items. Use the database schema as given above for the comments, decide on the user schema yourself.
3. **Authentication:** Setup basic email password authentication for registering a user, JWT or Firebase Authentication

4. Containerization and Deployment (Must Try)

1. **Dockerization:** You are encouraged to containerize your application using Docker for ease of deployment.
2. **Deployment:** Deploy it to cloud platforms such as Render, Railway, or Fly.io for the backend, and Netlify or Vercel for the frontend. Alternatively you can try to deploy from scratch using a VM from Azure. You get free credits on your student ID; look at the resources for the GitHub Student Developer Pack.

5. Bonus Tasks (Optional)

1. **Sort and Order the Comments:**
   a. Allow the user to sort the comments based on, user popularity or number of replies, or most upvotes, etc, try to think of more yourself.
2. **Admin Privileges:**
   a. You can add an admin user, that would be allowed to delete the comments of others as well.
3. Try to use your creativity, to make the UI more useful from the user's perspective, you can use modern UI from other sites that use comments, to take inspiration, and come up with things yourself.

5. Tech Stack

Although there is no strict regulation on the tech stack required, the following are preferred:

- **Frontend:** Use a Modern Framework like NextJS, ReactJS, Angular, Vue, etc.
- **Backend:** Keep it as simple as possible, Ex Stacks you can use: Express, FastAPI, Flask
- **Database:** MongoDB, PostgreSQL, MySQL, etc.
- Try to implement **animations** and transitions to ensure smooth user experience, you can use libraries like Framer motion or GSAP for it.

6. Submission Guidelines And Evaluation Criteria

- **GitHub Repository Submission**: Submit the GitHub repository link, make sure it is public.
- **README File:** briefly explain your approach, setup instructions, and features implemented.
- **Hosted Links Submission:** You can get a free domain from *freedomain.one*, and submit the hosted url.
- **Database:**
  - For SQLite, provide an SQL dump.
  - For MongoDB, provide a `.env` file with connection details or hardcode the URL.
- **Partially completed submissions are accepted and encouraged. We value creativity, initiative, and thoughtful problem-solving over mere completion. Even if your solution isn't fully functional, demonstrating clear intent, structured thinking, and genuine effort in tackling the problem is highly appreciated.**
- Evaluation:
  - Responsive UI, and UI/UX Design
  - Deployment
  - Code quality
- **Deadline:** The deadline for the task submission is: **Oct 17, 2025 11:59 PM**

7. Resources

- Free domains for your live deployments: freedomain.one, duckdns.org
- Custom data generation for more extensive testing application: Faker.js
- Dockerization: Docker Tutorial, Docker Curriculum
- GitHub Student Developer Pack: https://education.github.com/pack
- Deploy from scratch: Deploying a web-application using Nginx server and Reverse Proxy 🌻 | by Rajani Ekunde | Medium, Web-Application Deployment Using Nginx Web Server

8. Queries

For any queries, send an email to dev.interiittech14.0@gmail.com or fill the Form