Project: Online Forum / Discussion Board

What's This Project About?
Think of this as a mini **Stack Overflow** or **Reddit**, built for students and tech enthusiasts. It's a space where users can ask questions, start discussions, share knowledge, and help each other grow — whether it's fixing a code bug, exploring career paths, or discussing tech trends.

Users can create **threads** (posts), **reply** to others, **vote** on useful answers, and **organize** content using **tags** and **categories**.

Your job? Build a full-stack web application — both the backend (APIs, database, authentication) and frontend (user interface) — that powers this interactive community.

Tech Stack You'll Be Using

Backend:

- **Node.js + Express** – to build the core REST APIs
- **Authentication**:
  → Using **Passport.js** with strategies like **JWT** (token-based) or **session-based login** (cookies + sessions)

Database (Pick One):

- **MongoDB** – perfect for nested replies and flexible schemas
- **PostgreSQL** – great for structured, relational data with foreign keys and join tables

ORM/ODM:

- For MongoDB: **Mongoose**
- For PostgreSQL: **Prisma**, **Sequelize**

Frontend (React.js)

- The user interface will be built using **React**, giving you a smooth, interactive experience

- You'll build pages for:
  → Sign up / Login
  → Home (thread feed)
  → Thread view with nested replies
  → User profile
  → Search and filters

- Use tools like React Router, Context API (or Redux), and Axios/Fetch to handle navigation, state, and API calls

- UI should support real-time feedback and intuitive UX (e.g., upvoting, reply nesting, filtering)

What Features Should It Have?

User Accounts

- Users can register and log in

- Each user has a profile (name, avatar, etc.)

- Passwords are securely hashed

- Authenticated routes (only logged-in users can post/reply/vote)

Thread Creation

- Any logged-in user can start a new **thread**

- Each thread has:
  → Title
  → Description
  → Tags (like “JavaScript”, “Career”)
  → A category (like “Programming”, “Internships”)

Replies (with Nested Comments)

- Users can reply to:
  → Threads
  → Or other replies (creating a nested comment system)

- **MongoDB** handles this with embedded/nested documents

- **PostgreSQL** handles it using a `parent_id` reference in the replies table

---

Voting System

- Users can upvote or downvote both threads and replies

- A user can only vote once per item — no vote spamming

- Helps highlight the most helpful content

Tags & Categories

- Threads can have multiple tags

- Tags help with filtering and discovery

- Categories group threads into broader topics (e.g., “Career”, “Tech”, “Education”)

Search & Filters

Users should be able to:

- Search threads by title or content
- Filter by:
  → Most recent
  → Most upvoted
  → Tags or categories

How the Data Might Be Structured

Option 1: MongoDB (Flexible, Nested)

- `users` – user profile data and auth
- `threads` – each thread has title, content, tags, category, and embedded:
  → `replies` – nested comments
  → `votes` – upvotes/downvotes

Option 2: PostgreSQL (Structured)

- `users` – stores user info
- `threads` – includes title, description, creator_id
- `replies` – links to threads and has `parent_id` for nesting
- `votes` – tracks votes per user per item
- `tags` – list of all tags
- `thread_tags` – join table for thread-tag mapping
- `categories` – groups threads broadly
