1. App Overview:

The app will allow users to:

- Create accounts and login (authentication).
- Post discussion threads.
- Comment on threads.
- Like or dislike threads and comments.
- Categorize threads by topics.
- Search threads based on keywords or categories.

2. Protocol (Step-by-Step Process)

1. Setup the Development Environment:

- **Install Node.js**: Install Node.js on your machine for backend development.
- **Setup MongoDB**: Use MongoDB for database storage, locally or on a cloud platform like MongoDB Atlas.
- **Install Required Libraries**: Use npm to install the necessary libraries for each part of the MERN stack.

2. Create the Project Structure:

- Backend (Node.js + Express):
    - **/routes**: Contains routes for API endpoints.
    - **/controllers**: Contains the logic for each route.
    - **/models**: Define Mongoose models for users, threads, and comments.
    - **/middlewares**: Middleware functions for authentication and error handling.
    - **/config**: Store environment variables like DB connection strings and JWT secrets.
- Frontend (React):
    - **/components**: Reusable React components (e.g., Thread, Comment, Navbar).
    - **/pages**: Page components for routing (e.g., Login, Signup, ThreadList).
    - **/services**: API service layer for making HTTP requests to the backend.
    - **/store**: Use Redux (optional) for global state management.

3. Backend Development (Node.js + Express + MongoDB):

- **User Authentication**:
    - Create user schema with Mongoose.
    - Implement JWT-based authentication for login and signup.
    - Protect routes that require authentication using middleware.
- **Thread and Comment Models**:
    - Design the Mongoose schema for Threads (title, description, category, author).
    - Design the schema for Comments (content, author, related thread).
- **API Endpoints**:
    - **User Routes**: Register, Login, Logout.
    - **Thread Routes**: Create, Read, Update, Delete threads.
    - **Comment Routes**: Add and manage comments under a thread.

    - **Like/Dislike**: Implement liking/disliking of threads and comments.
    - **Search**: Endpoint to search threads by title or category.

4. Frontend Development (React + Redux + Material UI):

- **Authentication Pages**:
    - Create forms for user login and signup using React components.
    - Store authentication token using Redux or localStorage.
- **Thread Management**:
    - Create UI for creating and viewing threads.
    - List threads with pagination and filtering by category.
    - Allow users to interact with threads (like/dislike, comment).
- **Thread Search**:
    - Implement search functionality to filter threads by title or category.

5. Integration (Backend & Frontend Communication):

- **Axios**: Use Axios to make API requests from React to Express (for creating threads, adding comments, etc.).
- **JWT**: Manage the authentication token for secure routes.
- **Error Handling**: Show error messages to the user for failed actions (e.g., login failure, form validation).

6. Deployment:

- **Backend**: Deploy the Node.js server to a platform like AWS, Heroku, or DigitalOcean.
- **Frontend**: Deploy the React app on platforms like Vercel, Netlify, or AWS S3.
- **Database**: Host MongoDB using MongoDB Atlas or another cloud service.