Real-Time Task Collaboration Platform

1. Architecture Explanation

The application follows a client-server architecture with real-time communication support. Frontend: - Built using React (SPA architecture). - Component-based structure. - State managed using React hooks. - REST API communication. - Socket.IO client for real-time updates. Backend: - Node.js + Express server. - REST APIs for CRUD operations. - Socket.IO for real-time sync. - In-memory data storage (extendable to MongoDB). - Event-driven updates. Logical Database Schema: Users(id, name, email, password) Boards(id, name, ownerId) Lists(id, name, boardId) Tasks(id, title, description, listId, assignedUserId) ActivityLogs(id, action, userId, timestamp)

2. API Documentation

GET /boards - Returns all boards. POST /boards - Creates board. - Body: { name } POST /boards/:boardId/lists - Creates list in board. - Body: { name } POST /lists/:listId/tasks - Creates task in list. - Body: { title } WebSocket Event: boardUpdated -> Emits updated board data to all connected clients.

3. Assumptions and Trade-offs

Assumptions: - Basic single-organization usage. - Minimal authentication (extendable to JWT). - Limited concurrency. Trade-offs: - In-memory storage simplifies setup but no persistence. - Minimal UI styling prioritizes functionality. - No RBAC implemented.

4. Demo Credentials

Demo Email: demo@taskapp.com Demo Password: password123 (Note: Authentication placeholder for production-ready JWT implementation.)