### **✅ Day 1: Real-Time TaskBoard App**

**Summary:** A real-time, Trello-style task management board that allows users to create boards, add tasks, and move tasks between columns. Real-time collaboration ensures that multiple users see changes live. Great for showcasing React component architecture, drag-and-drop interfaces, CRUD API logic, and Socket.io for real-time updates.

**Tech Stack:**

* **Frontend:** React, TailwindCSS, React DnD
* **Backend:** Node.js + Express
* **Database:** MongoDB
* **Real-Time:** Socket.io

**Functionality:**

* JWT-authenticated users
* Create/edit/delete boards & tasks
* Drag-and-drop tasks across columns
* Real-time updates across users' browsers
* Persistent task storage in MongoDB

**Deployment:** Frontend on Netlify, Backend on Railway

**Timeline (30-min blocks, 12 hours):**

* 08:00–08:30 – Setup repo, project plan, folder structure
* 08:30–09:00 – Scaffold React + Tailwind, layout base
* 09:00–09:30 – Implement drag-and-drop columns/tasks (React DnD)
* 09:30–10:00 – Build task modals & form
* 10:00–10:30 – Backend boilerplate + connect MongoDB
* 10:30–11:00 – Create models: User, Board, Task
* 11:00–11:30 – Build CRUD endpoints
* 11:30–12:00 – Connect frontend to APIs
* 12:00–12:30 – Add user auth (JWT)
* 12:30–13:00 – Add WebSocket server + Socket.io client
* 13:00–13:30 – Broadcast task updates in real time
* 13:30–14:00 – Final UI polish + UX enhancements
* 14:00–14:30 – README, commit, push
* 14:30–15:00 – Demo video (Loom)
* 15:00–15:30 – Deployment (Netlify + Railway)

### **✅ Day 2: DevQuest – Developer Interview Question Board**

**Summary:** An interactive platform where developers post, solve, and upvote technical interview questions. It functions like a mini Stack Overflow tailored for DSA, system design, and code snippets. This project emphasizes PostgreSQL relational design, voting logic, markdown parsing, and API efficiency.

**Tech Stack:**

* **Frontend:** React, TailwindCSS, Markdown Editor
* **Backend:** Node.js + Express
* **Database:** PostgreSQL
* **Real-Time:** Optional WebSocket notifications for new questions/answers

**Functionality:**

* Auth via JWT
* Users post questions using Markdown
* Upvote/downvote logic with PostgreSQL transactions
* Comment and answer system
* Tagging & search

**Deployment:** Frontend on Netlify, Backend/PostgreSQL on Railway

**Timeline (30-min blocks, 12 hours):**

* 08:00–08:30 – Repo setup, DB schema design
* 08:30–09:00 – Create tables: users, questions, answers, votes
* 09:00–09:30 – Express app + PostgreSQL connection
* 09:30–10:00 – Build auth endpoints (JWT)
* 10:00–10:30 – Create CRUD endpoints for questions/answers
* 10:30–11:00 – Add voting logic (atomic transactions)
* 11:00–11:30 – React scaffolding, base layout
* 11:30–12:00 – Markdown editor + post form
* 12:00–12:30 – Feed view + upvote system
* 12:30–13:00 – Comment system
* 13:00–13:30 – Tag filter/search
* 13:30–14:00 – Optional: WebSocket for new posts alert
* 14:00–14:30 – Final polish, responsiveness
* 14:30–15:00 – Write README, commit
* 15:00–15:30 – Demo video
* 15:30–16:00 – Deploy (Netlify + Railway)

### **✅ Day 3: TutorLive – Real-Time Dev Tutoring Platform**

**Summary:** A live coding and dev tutoring web application that enables real-time chat and collaborative problem-solving. Tutors and students can join sessions, ask coding questions, exchange messages, and write code in a shared editor. This project highlights full-stack integration with real-time collaboration, Go-based backend performance, and MongoDB for session and chat history.

**Tech Stack:**

* **Frontend:** React, Zustand, TailwindCSS, CodeMirror
* **Backend:** Go (Fiber or Chi)
* **Database:** MongoDB
* **Real-Time:** Native WebSocket server in Go

**Functionality:**

* JWT-based user authentication
* Session creation by tutor or student
* Real-time messaging with WebSocket chat
* Shared code editor (with live sync)
* Session chat + code stored in MongoDB
* Optional tutor profile with session history

**Deployment:** Frontend on Netlify, Backend on Fly.io or Railway

**Timeline (30-min blocks, 12 hours):**

* 08:00–08:30 – Repo setup, Go + React scaffold
* 08:30–09:00 – MongoDB schemas: User, Session, Message
* 09:00–09:30 – JWT auth endpoints (login/register)
* 09:30–10:00 – CRUD endpoints for sessions and messages
* 10:00–10:30 – WebSocket server in Go for chat
* 10:30–11:00 – React layout: dashboard, chat, editor
* 11:00–11:30 – Zustand state for user/session context
* 11:30–12:00 – Implement chat UI with real-time sync
* 12:00–12:30 – Integrate live code editor (CodeMirror)
* 12:30–13:00 – Real-time code collaboration (WebSocket)
* 13:00–13:30 – Display session metadata & save chat/code
* 13:30–14:00 – Final polish + responsive design
* 14:00–14:30 – README + GitHub push
* 14:30–15:00 – Record demo walkthrough
* 15:00–15:30 – Deploy on Netlify + Fly.io