Trello-like Sprint Management Platform
Technical Documentation

***Only For Development & Deployment Teams**

1. Project Overview

Objective:
Build a Trello/Jira-like sprint management platform for agile project management with workspaces, boards, lists, and cards.

Key Features:

- User Authentication (Signup/Login)
- Workspace & Team Management
- Boards, Lists, and Cards (Task Management)
- Drag & Drop Task Reordering
- Comments, Labels, Due Dates
- Role-Based Access (Admin/Member)
- Real-Time Collaboration (Optional)
- Activity Log

2. Technical Specifications

Layer	**Technology**
Frontend	Next.js (React) + TypeScript
Backend	Node.js (Express)
Database	PostgreSQL (AWS RDS)
Auth	JWT + OAuth (Google/GitHub)
Drag & Drop	react-beautiful-dnd or dnd-kit
Realtime	Socket.IO
Deployment	AWS (EC2, ECS, or Lambda)
CI/CD	GitHub Actions + AWS CodePipeline

3. System Architecture

```
Client (React.js)
 ├── Authentication (Login/Signup)
 ├── Workspace & Board Management
 ├── Drag & Drop UI
 └── Real-Time Updates (Socket.IO)
        ⇅
API (Express/Flask)
 ├── Auth Routes (JWT)
 ├── Board/List/Card CRUD
 ├── Role-Based Permissions
 └── WebSocket Handlers (Optional)
        ⇅
Database (MongoDB/PostgreSQL)
 ├── Users, Workspaces, Boards
 └── Lists, Cards, Activity Logs
```

4. Development Phases

Phase 1: Project Setup
Initialize Git repo (GitHub/GitLab).

- Set up Nextjs for frontend.
- Set up Express backend.
- Configure MongoDB Atlas / Supabase (PostgreSQL).

Phase 2: Authentication & User Management
JWT-based login/signup.

- Role-based access (Admin/Member).
- Protected API routes.

Phase 3: Database & API Modeling
MongoDB Schema (Example)

```
// User Model
User: {
  name: String,
  email: String (unique),
  passwordHash: String,
  workspaces: [Workspace._id]
}
```

```
// Workspace Model
Workspace: {
  name: String,
  members: [{ userId: User._id, role: "Admin/Member" }],
  boards: [Board._id]
}

// Board Model
Board: {
  title: String,
  workspaceId: Workspace._id,
  lists: [List._id]
}

// List Model (e.g., "To Do", "In Progress")
List: {
  name: String,
  boardId: Board._id,
  cards: [Card._id],
  position: Number
}

// Card Model (Tasks)
Card: {
  title: String,
  description: String,
  dueDate: Date,
  listId: List._id,
  position: Number,
  labels: [String],
  comments: [{ userId: User._id, text: String }]
}
```

Phase 4: Core Functionality (CRUD + Drag & Drop)
API endpoints for:

- Workspaces (Create, Invite Members)
- Boards (Create, Update, Delete)
- Lists (Reorder, Move Between Boards)
- Cards (Assign Due Dates, Labels, Comments)
- Frontend drag & drop using `react-beautiful-dnd`.

Phase 5: UI/UX Enhancements
- Responsive design (Tailwind CSS / Material UI).
- Dark/Light theme toggle.
- Activity feed (Recent changes).

Phase 6: Real-Time Collaboration
- Implement **Socket.IO** for live updates.
- Notify users when a card is moved/edited.

5. Testing Strategy

Test Type	Tools
Unit Tests	Jest (JS) / PyTest (Python)
API Tests	Postman / Swagger
UI Tests	Cypress / Playwright

6. Deployment Plan

- AWS Deployment Architecture

```
Next.js (Frontend)
 ├── Hosted on AWS Amplify / S3 + CloudFront
 └── Connects to Node.js API
      ⇅
Node.js (Backend)
 ├── Deployed on AWS EC2 / ECS / Lambda (API Gateway)
 └── Connects to PostgreSQL
```

- Deployment Steps
  - ■ Step 1: Set Up AWS InfraNode.jsre
  - ■ Step 2: Deploy Backend (Node.js)
  - ■ Deploy Frontend (Next.js) Configure
  - ■ DNS & HTTPS
  - ■ CI/CD Pipeline (GitHub Actions)

- Database Schema (PostgreSQL)

```
CREATE TABLE users (
  id SERIAL PRIMARY KEY,
  name VARCHAR(100),
  email VARCHAR(100) UNIQUE,
  password_hash VARCHAR(200)
);
```

```
CREATE TABLE workspaces (
  id SERIAL PRIMARY KEY,
  name VARCHAR(100),
  admin_id INT REFERENCES users(id)
);

CREATE TABLE boards (
  id SERIAL PRIMARY KEY,
  title VARCHAR(100),
  workspace_id INT REFERENCES workspaces(id)
);

CREATE TABLE lists (
  id SERIAL PRIMARY KEY,
  name VARCHAR(100),
  board_id INT REFERENCES boards(id),
  position INT
);

CREATE TABLE cards (
  id SERIAL PRIMARY KEY,
  title VARCHAR(200),
  description TEXT,
  due_date TIMESTAMP,
  list_id INT REFERENCES lists(id),
  position INT
);
```

7. Post-Deployment & Maintenance

- Logging & Monitoring:Winston + Sentry.
- Error Tracking:LogRocket (Frontend).
- Feedback Loop: Feature requests → Version updates (v1.1, v2.0).

8. Future Roadmap (Optional Features)

- Notifications (Email/In-App)
- Team Chat (Socket.IO)
- Gantt Chart View (React Gantt)
- Mobile App (React Native)
- Slack/GitHub Integrations (Webhooks)