

Here's a **complete 2-week plan** to build a **Bug Tracker / Issue Tracker** like Jira using the **MERN stack**. This project is **industry-relevant**, especially for SaaS or enterprise dashboards, and it reflects real team workflows.

---

## Project Overview: Bug Tracker / Issue Tracker

### Goal:

Build a web application where teams can:

- Create & manage projects
  - Report bugs/issues as tickets
  - Assign tickets to team members
  - Move tickets across Kanban statuses (To Do, In Progress, Done)
  - Filter, search, and sort issues
  - Collaborate like in Jira or Linear
-

## □ Tech Stack

### ◆ Frontend

- **React.js** – Component-based UI
- **Tailwind CSS** – Modern responsive styling
- **React DnD or react-beautiful-dnd** – Drag-and-drop Kanban board
- **Axios** – API calls
- **React Router** – Navigation

### ◆ Backend

- **Node.js + Express.js** – REST API
- **MongoDB + Mongoose** – Database
- **JWT + bcrypt** – User auth

### ✂ Extras (Highly Recommended)

- **Context API / Redux** – Global state
  - **Socket.io** – (optional) for real-time collaboration
  - **Helmet + CORS + dotenv** – Security & configuration
  - **Multer (optional)** – File attachments (screenshots)
-

## Use Cases

#	Use Case	Description
1	User Authentication	Users can register/login, JWT auth used to protect routes
2	Project Management	Users can create projects, invite team members
3	Create Issue	Users can create bug reports or feature requests within a project
4	Assign Users	Assign tickets to members of the same project
5	Kanban Board	Drag tickets between “To Do”, “In Progress”, “Done”
6	Comments on Tickets	Team members can collaborate via threaded comments
7	Filter & Search Tickets	By status, priority, assignee, or keyword
8	Edit/Delete Tickets	Update or delete tickets (permission-based)
9	Role-Based Access (Optional)	Admin, manager, developer, viewer permissions
10	Upload Screenshot (Optional)	Attachments to support bug report clarity

---

## 2-Week Development Schedule

### Week 1 – Core Features & Backend

---

#### Day 1: Project Setup

- Setup MERN project folder structure
  - Configure Tailwind in React
  - Setup Express server and connect MongoDB (Atlas)
- 

#### Day 2: Authentication

- User model (name, email, password)
  - Register/Login APIs with bcrypt + JWT
  - Frontend forms and login state management
- 

#### Day 3: Project Management

- MongoDB Project schema (title, description, teamMembers)

- Routes: Create, update, delete, list projects
  - Add/remove members (email invite optional)
  - Show project list in frontend
- 

#### ✓ **Day 4: Ticket Model and Backend APIs**

- Ticket schema: title, description, priority, status, assignee, projectId
  - API: Create, list (by project), update, delete, assign
  - Protect routes with auth middleware
- 

#### ✓ **Day 5: Frontend – Create & Display Tickets**

- Ticket form with fields (title, priority, assignee, etc.)
  - Ticket list UI per project
  - Show ticket metadata (status, assignee, createdAt)
- 

#### ✓ **Day 6: UI Enhancements + Dashboard Layout**

- Sidebar + dashboard layout with Tailwind
  - Project selector dropdown
  - Breadcrumbs and responsive design
- 

#### ✓ **Day 7: Buffer & Testing**

- Fix bugs
  - Test APIs in Postman
  - Save project to GitHub
- 

## **31 Week 2 – Kanban, Filters, Polish, and Deployment**

---

### **✓ Day 8: Kanban Drag-and-Drop**

- Setup `react-beautiful-dnd`
  - Columns: “To Do”, “In Progress”, “Done”
  - Drag ticket to update status
  - Save changes via API
- 

### **✓ Day 9: Comments**

- Create comment schema (ticketId, userId, text, timestamp)
  - Add threaded comment box under each ticket
  - Display comment history
- 

### **✓ Day 10: Filtering & Search**

- Add dropdown filters (status, priority, assignee)
- Add search bar for keyword match

- API support for filtered results
- 

### ✓ Day 11: Edit & Delete Tickets

- Edit ticket modal
  - Delete with confirmation popup
  - Authorization check for user role (basic)
- 

### ✓ Day 12: Deployment

- Deploy backend (Render/Railway)
  - Deploy frontend (Vercel/Netlify)
  - Connect with MongoDB Atlas
  - Secure environment variables
- 

### ✓ Day 13: Polish + ReadMe + Mobile Responsive

- Responsive styles for mobile
  - Add loader/spinner, toast messages
  - Create clean README.md for GitHub
- 




### ✓ Day 14: Final Testing & Video Demo

- Test end-to-end flows

- Record a short walkthrough video
  - Share it on GitHub + LinkedIn
- 

## YouTube Resources

Use these to follow along and adapt features:

1. **Team Project Management App** : [Build & Deploy a MERN Team Project Management App | Google Auth, Roles, Workspaces & Analytics 1/2](#)
  2.  **React DnD Tutorial** – [React Drag And Drop \(dnd-kit\) | Beginners Tutorial](#)
  3.  **MERN Dashboard Project (role-based auth)** – [Code With Ayan](#)
  4.  **JWT Auth MERN Stack** – [PedroTech](#)
  5. **Fullstack/MERN Stack Project Management Application:** [Fullstack/MERN Stack Project Management Application | React.js | Node.js | React Router v7](#)
- 

## Final Deliverables

- Deployed live app
- GitHub repo with README and screenshots
- Responsive UI with JWT auth
- Functional drag-and-drop Kanban

- Ticket creation, filtering, and user assignment
  - Optional: Comments, file upload
- 

## Industry-Level Learnings

Skill	Relevance
Kanban UI	Used in agile tools (Jira, Trello, Asana)
JWT Auth	Core skill in any SaaS product
Role-Based Access	Enterprise-grade applications
Filters/Search	Common in dashboards, CMS
MongoDB Relationships	Many-to-one (tickets -> project), one-to-many (comments)
Real-time features	With Socket.io in extensions