React.js Issue Tracker Project

I am working in MNC, but not in project. I want to switch with three years experience in react.js. what kind of project use cases can i show in my github ? Give details of two such case such as issue tracker system and ecommerce with system design ideas

That's a great move—having solid projects on GitHub can *massively* boost your chances during interviews, especially with 3 years of React.js experience. Since you're aiming to switch, showcasing projects that simulate **real-world use cases**, follow **best practices**, and include **good system design** will make your profile stand out.

Here are two **detailed project ideas** (with system design suggestions) you can build and push to GitHub:

🚀 1. Issue Tracker System (like Jira-lite)

🌟 Key Features:

- User Authentication (JWT-based)
- Create, assign, and update issues
- Set priorities, tags, and deadlines
- Kanban Board view (To Do, In Progress, Done)
- Team/Project management
- Comment and Activity history

🔧 Tech Stack:

- **Frontend**: React.js (Hooks, Context API/Redux), React Router
- **Backend**: Node.js + Express
- **Database**: MongoDB or PostgreSQL
- **Auth**: JWT + bcrypt
- **State Management**: Redux Toolkit or Zustand (to show variety)
- **UI Library**: Material UI or Tailwind CSS

🧱 **System Design Highlights:**

- **Modular architecture**: Split the frontend into reusable components: `IssueCard`, `BoardColumn`, `CommentBox`, etc.
- **Backend**:
  - Routes: `/api/issues`, `/api/projects`, `/api/users`, `/api/comments`
  - Use MongoDB for fast and flexible issue/document storage
  - Store activity logs as separate collection for each issue
- **Kanban updates**: Use WebSocket or polling to auto-update board in real-time
- **Authorization**: Use middleware for role-based access (e.g., Admin, Developer, Tester)

💡 **Bonus:**

- Add filtering/searching/sorting
- Drag and drop using `react-beautiful-dnd`
- Dark mode toggle

🛒 2. E-Commerce Web App (Full-stack)

🌟 **Key Features:**

- Product listing, categories, filters
- Shopping cart, wishlist, checkout flow
- Order history, payment status
- Admin dashboard (add/edit products, see sales analytics)
- User reviews and ratings
- Optional: Integrate payment gateway (Stripe demo)

🔧 **Tech Stack:**

- **Frontend**: React.js + Redux Toolkit or Context API
- **Backend**: Node.js + Express.js
- **Database**: PostgreSQL or MongoDB (depending on familiarity)

- **Authentication**: JWT for API + session for UI
- **Payments**: Stripe API (test mode)
- **UI**: TailwindCSS or Chakra UI

🧱 **System Design Highlights:**

- **Product service**: `/api/products`, includes search, sort, filter
- **Cart logic** stored in frontend state, sync with backend
- **Order service**: `/api/orders` with payment intent via Stripe
- **Admin module**: Protected routes, stats via charting library (like Chart.js)
- **Modular folder structure**: `components/`, `services/`, `pages/`, `utils/`

💡 **Bonus:**

- Add SSR with Next.js if you want to show performance optimization
- Add GraphQL backend instead of REST if you want to stand out with advanced skills
- Show unit tests (Jest, React Testing Library) to show testing discipline

---

📌 Final Tips for GitHub:

- Include a **README.md** with:
  - Project description
  - Live demo link (use Vercel/Netlify/Render)
  - Tech stack
  - Folder structure diagram
  - Setup instructions
- Use **commits with clear messages**
- Add **issues** and **project boards** in your GitHub repo to simulate real project workflow

---

Would you like me to help you scaffold or design folder structure or APIs for either of these projects? Or maybe help you write the README to make it look polished?