42cc Full-Stack Next.js Developer Test Assignment

🔧 The Task

Build a modern web app for remote teams where each team member can announce what they're working on and when they plan to finish. The app should:

- Be responsive and mobile-friendly
- Provide an intuitive UI and smooth UX
- Include core team expectation management features

This test assignment is designed to touch on many real-world Next.js + React concepts including:

- Server Actions and API routes
- SSR, Server Components, and Client Components
- State management (e.g., using useState, useContext, or global solutions)
- Form creation and validation (both client-side and server-side)
- TypeScript usage and type safety
- Clean architecture and developer experience

We understand that without LLMs this may take several days. But with smart use of AI, it can be completed in a few focused hours. The goal is not just to finish the task but to:

- Follow a well-defined process
- Write and iterate on meaningful prompts
- Review and improve AI-generated code where necessary
- Build with care, readability, and attention to real-world use cases

---

📅 Functional Requirements

1. **User Authentication:**
    - Sign up with email and password
    - Sign in/out (with Clerk)
2. **Expectations Dashboard:**
    - Show all team members' current expectations
    - Each expectation must include:
        - Author name
        - Task/expectation title
        - Time of creation
        - Estimated completion time
3. **Manage Expectations:**
    - Add new expectation (only one active expectation per user)
    - Edit/delete own expectation
    - Mark expectation as done
4. **History:**
    - Show full history of expectations (including done time)

🚀 Tech Stack Requirements

- **Framework:** Next.js 15 (App Router)
- **Backend:** Supabase (database, storage, can be used for auth)
- **ORM:** Drizzle ORM
- **UI:** Shadcn UI, Radix UI, Tailwind CSS
- **Testing:** Vitest / Playwright / Cypress (use TDD for backend logic)
- **Auth:** Clerk (or Supabase Auth)
- **Deployment:** Vercel or similar (demo URL required)

✏️ Process Requirements

1. **TDD First:**
   - Every backend feature (API routes, server actions) **must follow TDD**:
     - First commit: Test
     - Second commit: Implementation
2. **Prompts & AI Use:**
   - You are encouraged to use AI (ChatGPT/Claude, GitHub Copilot, V0/Loveable, Cursor, etc.)
   - Save and commit any AI prompt you use as a comment or a text file in the repo (e.g., `/prompts/add_expectation.txt`)
   - Review the generated code, improve where necessary, and explain why changes were made (via comments or PR notes)
3. **Branching & PRs:**
   - Each ticket must be implemented in a separate branch
   - Branch naming convention: `feature/ticket-<number>-<short-description>` (e.g., `feature/ticket-3-show-expectations`)
   - Create a Pull Request (PR) for each ticket
   - When ready, submit the PR for review to @jardev
4. **Peer Review & Interview Steps:**
   - Code will be reviewed by 2 engineers
   - After passing reviews, you will meet with a PM
   - Final interview with the CEO

📆 Tickets

✉️ Ticket #1: Project Bootstrap & Landing Page

- Create a new project with Next.js 15
- Set up Supabase + Clerk + Tailwind + Shadcn UI + Drizzle ORM
- Add landing page with:
  - App intro
  - Sign-up button
- Deploy to a public staging environment (e.g., Vercel)
- Push to GitHub with CI/CD enabled

🔐 Ticket #2: Authentication

- Implement Clerk authentication (email/password)
- Only authenticated users can access the dashboard

- Add demo user (share credentials)

📊 Ticket #3: Expectations List

- Fetch all current expectations from all users
- Show who expects what and when
- Sort by expected completion time

➕ Ticket #4: Add & Modify My Expectation

- Add new expectation (replaces current one)
- Edit/delete current expectation

✅ Ticket #5: Mark as Done & View History

- Mark current expectation as done
- Show all past expectations
- Include timestamps: created, finished

🔗 Deliverables

- GitHub repository link (with clear commit history showing TDD pattern)
- Live URL (e.g., Vercel)
- Login credentials for demo user
- Brief summary of what was used/generated with AI
- List of prompts used (stored in `/prompts/` directory)

📃 Review Criteria

1. **Correctness:** Meets all functional requirements
2. **Code Quality:** Clean, modular, readable, well-documented, migrations, etc.
3. **Tests:** TDD adherence, meaningful test coverage
4. **Use of AI:** Smart use of LLMs and documented prompts
5. **UX/UI:** Clean and usable interface
6. **Velocity:** How much was completed in a reasonable time
7. **Process Adherence:** Git commit structure, TDD flow, PR discipline, prompt documentation and code reviews

---

Good luck, and have fun building!

42 Coffee Cups Team