🚀 TelemetryTrade – High-Performance Token Trading Interface

TelemetryTrade is a modern, high-performance web trading interface designed to show real-time price movement of crypto tokens. Inspired by the Axiom Trade Token Discovery Table and built to meet the demanding standards of low-latency traders, the project is architected with strict performance, scalability, and accessibility principles in mind.

🏢 Company

TelemetryTrade is your take-home assessment for a frontend developer role in a high-speed trading environment. Focused on UX performance, real-time data, and pixel-perfect design, the app demonstrates your ability to deliver production-grade code with modern tools.

Inspired by startups like **Axiom Trade** and **Eterna**, the goal is to build an institutional-grade trading UI in a limited-time window.

✨ Features

- ☑ Pixel-perfect recreation of Axiom Trade's token discovery table (≤ 2px diff)
- ☑ Fully responsive down to 320px width with mobile-first optimizations
- ☑ Real-time WebSocket price updates and animated row deltas (smooth green/red transitions)
- ☑ Sortable, filterable, and dynamic token table with live hover effects and modals
- ☑ Loading states including skeleton, shimmer, and progressive loading
- ☑ Performant rendering: no layout shifts, <100ms interactions, virtualized rows
- ☑ Visual-regression test compatible (e.g., Percy, Chromatic)
- ☑ Lighthouse ≥ 90 on both mobile and desktop

⚠ Downsides (Limitations)

- ❌ Data is mocked (no real trading)
- ❌ Not production-connected to live chain APIs
- ❌ Charts (e.g. tradingview) are placeholders unless added as a bonus
- ❌ WebSocket server currently runs locally (not yet deployed cross-server for Vercel)

🔥 Extras (Optional Bonus Additions)

- 🎨 Framer motion for smooth animated transitions
- 📊 Token preview charts with Recharts or TradingView widgets
- 🔄 Token state persisted via Zustand or URL params
- ♿ Full keyboard navigation & ARIA roles for accessibility
- 🔍 Search & deep-link filters, user-configurable columns
- 🤖 Deployed WebSocket server (e.g. Fly.io or Railway)

🌐 Web Fundamentals

- **Frontend**: Next.js 14 (App Router), TypeScript (strict mode), Tailwind CSS
- **State Management**: Redux Toolkit + React Query
- **UI Components**: Radix UI / shadcn/ui / Headless UI
- **Streaming**: WebSocket (Socket.io-client mock)
- **Table**: Virtualized row rendering for 10k+ rows with no lag
- **Performance Tools**: Lighthouse, memoization, lazy-loading
- **Testing**: Jest + React Testing Library
- **Architecture**: Atomic Design (Atoms → Molecules → Organisms → Templates)
- **Styling**: Tailwind, utility-first, no inline styles
- **Documentation**: README, code comments, clean commits

```
# Core Libraries to Install
npm install @reduxjs/toolkit react-redux @tanstack/react-query axios socket.io-client @radix-ui/react-popover @radix-ui/react-tooltip tailwindcss @shadcn/ui class-variance-authority clsx framer-motion react-virtual
```

📦 Deliverables Overview

Deliverable	Requirement
🗀 GitHub Repo	Clean commit history, public repo link
🌍 Vercel Deployment	Live running demo of app
🎥 YouTube Demo	1–2 min public walkthrough of features
🛠️ README.md	Architecture, setup steps, tech decisions

☑ All deliverables **required** for completion.

🎬 YouTube Demo Guide

Your demo video should include:

1. App load → skeleton screens
2. Table → hover effects + sorting + tabs
3. Real-time price updates with transitions
4. Mobile view at 320px
5. Quick code walkthrough (optional)
6. Deployed Vercel link in description

Make it **public**, 1–2 min max.

🛠️ Project Roadmap (24–48 Hour Scope)

```
📍 PHASE 1 – Setup & Architecture
1. Initialize Next.js + TS project
2. Configure Tailwind, shadcn/ui, ESLint, Prettier
3. Add folder structure (Atomic Design)
4. Setup Redux Toolkit and React Query

📍 PHASE 2 – UI Construction
5. Build pixel-perfect token table
6. Implement tabs: "New Pairs", "Final Stretch", "Migrated"
7. Add sorting, row hover effects, modal on click
8. Build UI with Radix + Tailwind (≤ 2px diff)

📍 PHASE 3 – Real-Time + Loading States
9. Mock WebSocket server to push price updates
10. Smooth animated deltas on price changes
11. Add skeleton, shimmer, progressive loading

📍 PHASE 4 – Performance & Mobile
12. Virtualized table (without layout shifts)
13. Responsive view for 320px (horizontal scroll)
14. Lighthouse score tuning ≥ 90

📍 PHASE 5 – Deliverables
15. Push code to GitHub (clean commits)
16. Deploy app to Vercel (WebSocket fix)
17. Record 1–2mi YouTube demo (public)
18. Finalize README w/ screenshots + architecture
```

🗂️ Folder Structure (Atomic Architecture)

```
src/
├── app/
│   ├── page.tsx
│   └── layout.tsx
├── components/
│   ├── atoms/         # Buttons, badges, icons
│   ├── molecules/     # Rows, modals, lists
│   ├── organisms/     # Token table, filters
├── features/
│   ├── token-table/   # Redux slice, hooks
├── hooks/             # Custom queries, WS hooks
├── lib/               # Utils, constants, API
├── store/             # Redux config
├── types/             # TypeScript types
└── styles/            # Global styles
```

Development Commands

```
# Install dependencies
npm install

# Run development server
npm run dev

# Run tests
npm run test

# Lint code
npm run lint

# Build for production
npm run build
```

Performance Checklist

- [ ] No layout shifts (CLS = 0)
- [ ] First render < 1.2s on 3G
- [ ] All interactions < 100ms
- [ ] Lighthouse > 90 mobile/desktop
- [ ] JS bundle < 200kb (unused removed)
- [ ] WebSocket client reconnect on fail

Contributing & Support

Clone this repository, install dependencies, and start building! For bug reports or feature suggestions, open an issue on GitHub.

License