# AP Statistics Project: Complete Technical Guide

Your Physical Touchstone for Project Organization

Robert Colson

June 23, 2025

**Project Overview**

**What it is:** A complex web application for teaching AP Statistics that has evolved from a traditional client-server model to an innovative peer-to-peer (P2P) blockchain-like system called "APStatChain."
**Current State:** The project is a TypeScript monorepo with a modern React frontend and a sophisticated decentralized communication system between students.

## Core Technologies & Languages

### Primary Stack

- **Language:** TypeScript (used everywhere for reliability)
- **Runtime:** Node.js (development server, tests, tools)
- **Database:** SQL files present (PostgreSQL/MySQL likely)
- **Frontend Framework:** React
- **Build Tool:** Vite (fast development server)

### Key Libraries

- **Data Management:** TanStack Query (React Query) - for server data fetching/caching
- **Navigation:** React Router
- **P2P Communication:** PeerJS (WebRTC) - direct browser-to-browser connections
- **Local Storage:** Dexie.js (IndexedDB wrapper) - stores blockchain data in browser
- **Cryptography:** @noble libraries - for hashing and signatures
- **Legacy:** Socket.IO Client (mostly unused now, used internally by PeerJS)

# File Organization

## Top-Level Structure

```
apps/
          web/           # Main React frontend (what students see)
          api/           # Backend server (mostly deprecated/empty)
          web-legacy/    # Old version (reference only)

packages/
          chain-core/    # Core blockchain logic & cryptography
          chain-p2p/     # Peer-to-peer connection management

database/          # SQL schema files
pdfs/             # AP Statistics curriculum materials (by unit)
docs/             # Project documentation
archive/          # Deprecated code (ignore)
logs/             # Development logs (can delete)
```

Listing 1: Project Directory Structure

## Test Organization

- **Unit Tests:** Co-located with source code (`.test.ts` files)
- **E2E Tests:** Should be in `apps/web/e2e/` (may be missing)
- **Mocks:** Centralized in `apps/web/src/mocks/handlers.ts`

# Essential Commands

## Daily Development

- **`npm run dev`** - *Your main command* - starts everything (frontend + backend)
- **`npm run build`** - Creates production-ready builds

## Testing

- **`npm test`** - Runs all unit tests across entire project
- **`npm run test:watch`** - Interactive test mode (re-runs on file save)
- **`npm run test:ui`** - Visual browser-based test interface

## Code Quality

- **`npm run format`** - Auto-formats code with Prettier
- **`npm run lint`** - Checks for code errors/style issues
- **`npm run type-check`** - Validates TypeScript types
- **`npm run clean`** - Nuclear reset (deletes node_modules, dist folders)

### E2E Testing (from web directory)

```
cd apps/web
npx playwright test
```

## Data Flow & Architecture

### The Evolution Story

Your project has migrated from traditional client-server to decentralized P2P:

**OLD WAY (Deprecated)**

- Central API server handled user accounts, progress
- Socket.IO provided real-time updates
- Traditional database-driven architecture

**NEW WAY (Current)**

- Decentralized "APStatChain" system
- Direct browser-to-browser communication
- Local storage in each student's browser

### Current Data Flow (Step-by-Step)

1. **App Startup**
   - Static content loads instantly from `allUnitsData.ts`
   - `BlockchainProvider` creates `BlockchainService` instance
   - `P2PNode` initialized for network identity
2. **Peer Discovery**
   - DNS seeding queries your domain (e.g., `seed.apstatchain.com`)
   - Gets list of active peers (other students)
3. **Connection Setup**
   - PeerJS uses public signaling server for initial handshake
   - WebRTC establishes direct browser-to-browser connection
4. **Data Synchronization**
   - P2P messages: `TX_BROADCAST` (lesson completed), `BLOCK_PROPOSAL`
   - Data validated and stored in IndexedDB via Dexie
   - UI updates reactively

### System Responsibilities

- **Central API:** Legacy/Optional (user accounts, backups)
- **Socket.IO:** Deprecated (replaced by P2P)
- **P2P "APStatChain":** Primary system (real-time student progress sharing)

## Testing Strategy

### Test Types & Tools

- **Unit/Integration:** Vitest (fast, Jest-compatible)
- **End-to-End:** Playwright (browser automation)
- **API Mocking:** MSW (Mock Service Worker)

### Mocking Setup

- **Development:** Add `?msw=true` to URL for mock API responses
- **Unit Tests:** Automatic mocking via `mocks/server.ts`
- **Mock Definitions:** All in `apps/web/src/mocks/handlers.ts`

### Critical E2E Test Scenarios (from test-results analysis)

1. **Authentication Flow:** Username → Login → Dashboard → Navigation
2. **Core Learning Loop:** Complete Mining Puzzle → See Leaderboard Results
3. **Visual Regression:** Screenshot comparison for UI stability
4. **App Updates:** Detect old version, prompt refresh

### Test Environments

- **Frontend Tests:** `happy-dom` (simulated browser)
- **Backend Package Tests:** `node` environment

## Key Dependencies by Category

### Frontend Core

React, TypeScript, Vite, TanStack Query, React Router

### P2P & Blockchain

PeerJS, @noble/* (crypto libraries), Dexie.js (IndexedDB wrapper)

### Testing

Vitest, Playwright, MSW, @testing-library/* (React testing utilities)

### Code Quality

ESLint, Prettier, TypeScript compiler

## Important Notes for Future Development

### What's Working (Don't Break)

- P2P communication system is the primary data flow
- Local IndexedDB storage via Dexie
- Monorepo workspace structure
- Co-located testing strategy

### What's Legacy (Can Ignore/Remove)

- `apps/api` folder (empty/deprecated)
- Direct Socket.IO usage in main app
- Traditional API fetch calls (have fallback logic)

### Potential Issues to Watch

- E2E tests may be missing (config exists but files absent)
- P2P system complexity when adding features
- Browser compatibility for WebRTC/IndexedDB

## Quick Reference Card

**Essential Commands**

**Start Development:** `npm run dev`
**Run Tests:** `npm test` or `npm run test:watch`
**Format Code:** `npm run format`
**Clean Reset:** `npm run clean`

**Key Locations**

**Main App:** `apps/web/src/`
**P2P Logic:** `packages/chain-core/` & `packages/chain-p2p/`
**Tests:** Next to source files (`*.test.ts`)
**Mocks:** `apps/web/src/mocks/handlers.ts`

**Architecture Flow**

**Data Flow:** Static → P2P Discovery → WebRTC → IndexedDB → React UI
**Architecture:** Monorepo → TypeScript → React → P2P Blockchain → Local Storage