I’m a non‑dev. You (Claude) have access to my local folder: ~/Desktop/mivton. This is a Node.js + Express + PostgreSQL app deployed on Railway at https://www.mivton.com. We completed **Phases 1–3** from the “🚀 Mivton Development Plan – Complete Roadmap,” with some items in 3.3 still pending. I need you to: (1) set up a test toolchain, (2) create **Phase 1 (Auth & Sessions)** tests, (3) run them locally, (4) fix failures one by one and commit after each fix, (5) generate a short report. Keep changes within the mivton folder. Do not touch production data.

**Do this now, in order:**

1. **Install dev tooling (locally):**
   * cd ~/Desktop/mivton
   * npm i -D jest supertest @types/jest eslint playwright
   * npx playwright install --with-deps
   * Add package.json scripts:
     + "lint": "eslint ."
     + "test:unit": "jest --runInBand tests/unit"
     + "test:api": "jest --runInBand tests/api"
     + "test:e2e:headless": "playwright test --reporter=list"
     + "test:all": "npm run lint && npm run test:unit && npm run test:api && npm run test:e2e:headless"
2. **Safe local test env:**
   * Create .env.test pointing to a **separate local** Postgres DB (not prod).
   * Add minimal seed & teardown in database/: schema.sql (reuse) + seed\_test.sql + teardown\_test.sql. Include 2 users: userA@example.com, userB@example.com with hashed passwords.
3. **Phase 1 tests (Auth & Sessions):**
   * Create folders: tests/unit, tests/api, tests/e2e.
   * **Unit (Jest):** password hash/verify helpers; validation middleware; DB connector error handling.
   * **API (Supertest):**
     + POST /auth/register (happy path, weak/duplicate email rejection).
     + POST /auth/login (good creds, bad pass, unknown user).
     + Auth‑required route returns 200 when logged in, 401 when not.
   * **E2E (Playwright, local):** visit landing → login page → perform login → redirected dashboard visible; no severe console errors.
4. **Run & fix loop:**
   * Run: npm run test:all.
   * For each failure: reproduce → patch → re‑run specific test → re‑run full suite → **commit** (e.g., fix(auth): handle duplicate email) before moving on.
5. **Output:**
   * Create /test-reports/summary\_phase1.md with: failing tests found, root cause, fix applied, files touched, test links.

**Acceptance checklist (tick before ending chat):**

* Tooling installed; scripts added.
* .env.test + seed/teardown ready.
* Phase 1 unit + API + E2E tests written.
* All Phase 1 critical/high issues fixed; tests green.
* /test-reports/summary\_phase1.md created.