PROJECT SHOWCASE / ENGINEERING & PRODUCT PORTFOLIO

Gig Goal Tracker

Technical Case Studies

Prepared for professional review | November 4, 2026 | English edition

Evidence note: Based on prior project discussions, described architecture, selected development/debugging history and portfolio records available at preparation time. This is a portfolio case study, not a fresh source-code audit, external security review, market study or verified commercial due diligence. Planned capabilities and unverified outcomes are labeled accordingly.

Gig Goal Tracker | Technical Case Study

Gig-work earnings productivity | Stage: Actively developed earnings/goal tracking app; advanced mileage/tax reporting scoped or evolving

Technical summary

Gig Goal Tracker is a mobile-friendly web application to help gig workers manage earning targets, trips and work sessions. It is grounded in the developer’s direct use of gig delivery workflows and aims to keep essential daily tracking simple while progressively revealing advanced analysis.

Problem and engineering context

Gig workers often evaluate offers, maintain earnings goals and assemble tax records across fragmented tools. Without a consistent session and trip history, it is difficult to interpret work performance or reconcile mileage.

Technology stack

- Next.js 16, React 19 and TypeScript for application and UI.
- MongoDB / Mongoose for user settings, trip entries, work sessions, earning records and goals.
- Email/password authentication; Google sign-in and account recovery discussed as product evolution.
- Docker deployment on a VPS, with application-specific environment variables and secrets.
- Progressive-disclosure UX to keep the baseline workflow lightweight.

System architecture and component boundaries

- Dashboard: current goal, progress, earning summaries and quick entry actions.
- Settings: workdays, configurable sessions and target windows.
- Transaction records: trip earning, tips, timestamp and optional miles.
- Aggregation: day/week/month/year totals and efficiency ratios using consistent inclusion rules.
- Optional modules: odometer-based mileage and tax exports, isolated from core daily tracking.

Data model and domain contracts

- User preferences: weekly goal, work schedule, break handling and session definitions.
- Trip and earnings record: gross amounts, tips, platform, timestamps and editable metadata.
- Mileage: hybrid design using daily odometer as a primary log and per-trip miles as analytical detail (planned/evolving).
- Derived calculations: dollars/hour, dollars/mile, goal attainment and period totals; ensure time and mileage denominators are correct.
- Reporting: exportable records should distinguish business miles, other miles, tips and source completeness.

Primary application flows

- Choose goal and configure working days/sessions.
- Start work and record trips and tips; edit past entries with an audit-friendly policy.
- Track progress at a glance and review historical summaries.
- Optionally enable advanced mileage/tax tracking when the user needs it.

Engineering challenges and tradeoffs

- Time-zone boundaries and workdays crossing midnight can shift weekly totals.
- Double-counted cash tips or duplicated trips can distort analytics.
- Manual odometer entries require validation for decreases, gaps and inconsistencies.
- Financial summaries must be auditable and should not imply official tax filing or advice.
- Responsive density must preserve simple task entry for workers on the move.

Available evidence and limits

- Project discussions confirm an actively used/developed goal and earnings workflow and current files under iteration.
- A proposed optional odometer/taxes module and future statistics were discussed; do not treat them as finished.
- No independently verified customer count, subscription processing or certified tax integration.

Recommended technical validation / next steps

- Define canonical trip/session schema and date/time boundary rules.
- Implement mileage reconciliation and explicit missing-data indicators.
- Test money arithmetic with integer cents and reproducible aggregation fixtures.
- Create CSV/PDF exports with a clear non-tax-advice disclaimer.
- Private invitation beta, then evaluate whether Stripe subscriptions are justified.

Recruiter interview prompts

- How are weekly boundaries and session crossing-midnight handled?
- How do you prevent duplicated trips and ensure reliable money arithmetic?
- What schema supports odometer reconciliation and export?

Disclosure: This report intentionally distinguishes observed discussions from proposed engineering improvements. Tests, benchmarks, revenue, user adoption and security compliance are not asserted without supporting evidence.