0. User Story Explanation

Feature: Advertisement Discount Option

User Story: As a budget-conscious rider, I want to watch a 30-60 second advertisement before booking in exchange for a 10-15% discount so that I can reduce my fare when I'm not in a rush.

Explanation: All interviewed users expressed openness to watching ads for discounts. This creates a voluntary way for price-sensitive users to lower costs while generating additional revenue for the platform. The key is making it optional so time-sensitive riders aren't forced to participate.

1. Header

Document: Advertisement Discount Option — Development Specification

Label Prefix (feature): AD (used across modules/components/classes)

Version History

- v1.0 (2025-09-22) — Initial draft

Authors & Roles (never delete anyone; version-specific noted)

- Christy Tseng — Feature Owner (v1.0)

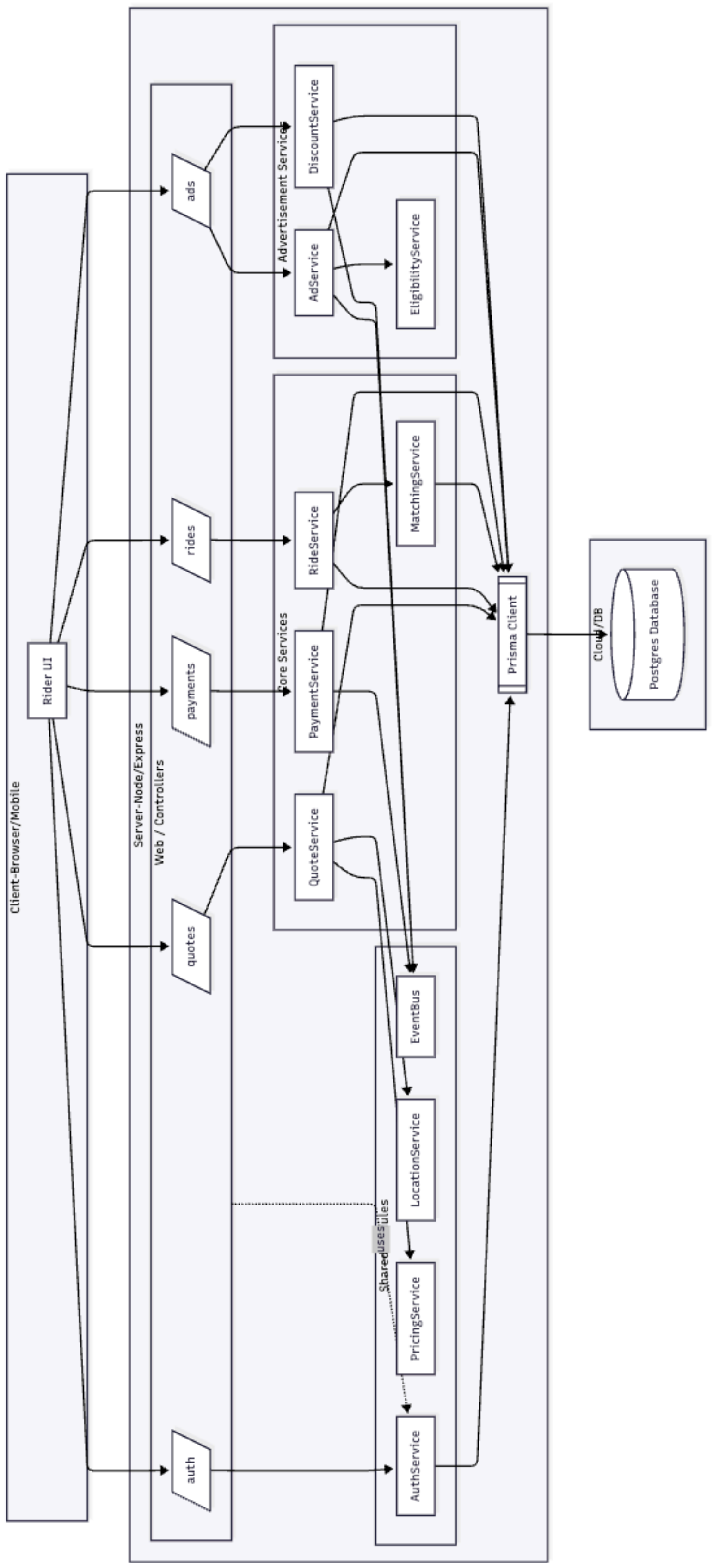
2. Architecture Diagram

Legend

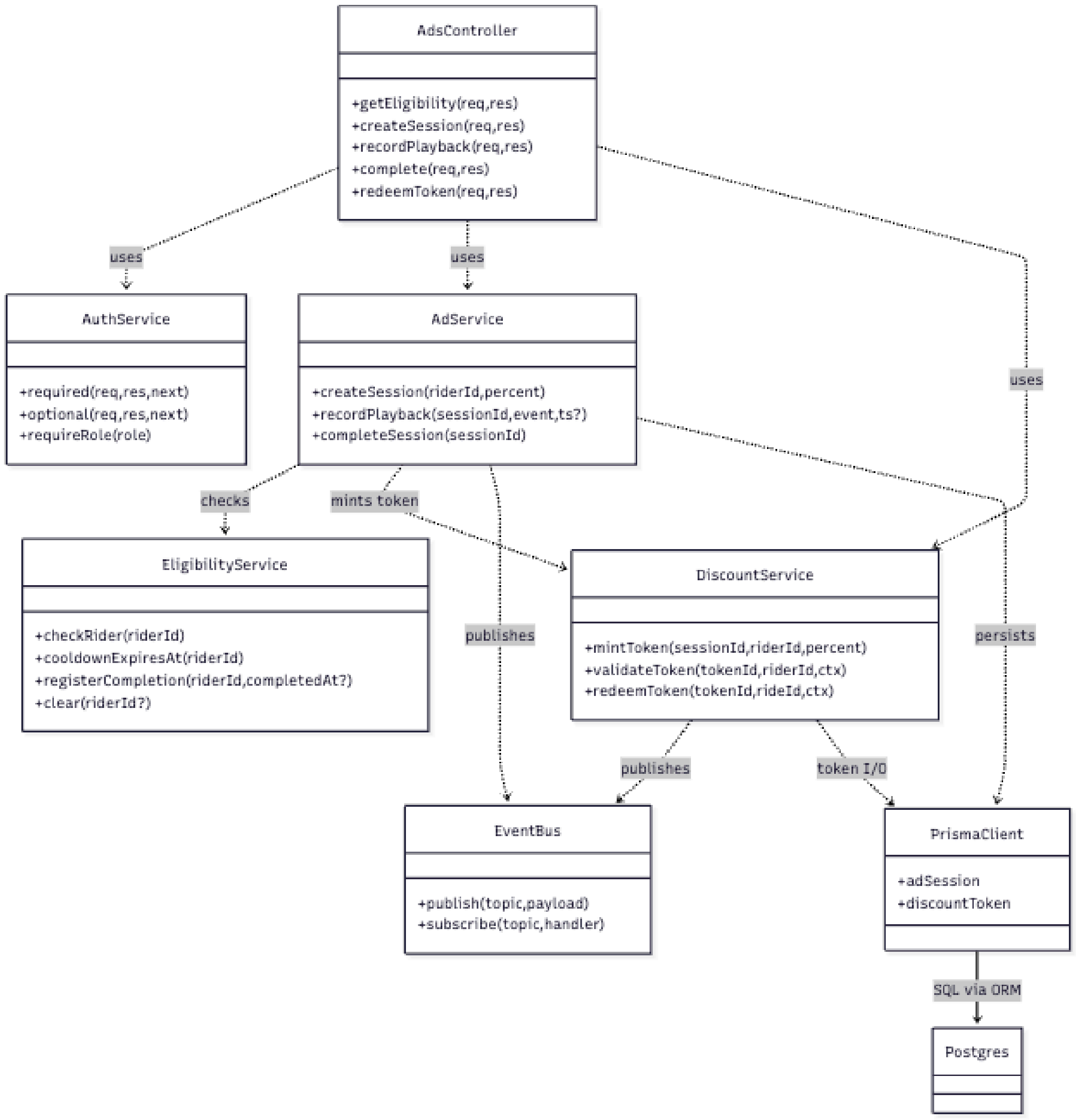
- Client: Rider UI that calls REST endpoints.
- Web/Controllers: Express route handlers for /auth, /quotes, /rides, /payments, /ads.
- Core Services: Core ride domain (quotes, rides, matching, payments).
- Advertisement Services: Ad session lifecycle and discount token issuance/validation.
- Shared Modules: Cross-cutting utilities (Auth, Pricing, Location, EventBus).
- Prisma Client: ORM used by services for all data access.
- Postgres: Primary database storing users, drivers, rides, payments, ad sessions, discount tokens.

Rationale

- Clear separation of concerns: thin controllers, cohesive services, centralized shared utilities.
- Single data access path: all services persist via Prisma to Postgres (simplifies ops, observability, and consistency).
- Ad–core integration through shared modules:
    - DiscountService bridges ad token minting with core quote/ride validation and redemption.
    - PricingService ensures consistent discount math across quote and ride.
    - EventBus enables low-coupling domain notifications (e.g., token minted, ride completed).
- Scalable and evolvable: new endpoints or providers slot into the appropriate service area without altering the fundamental wiring.


Client-Browser/Mobile
Rider UI
Server-Node/Express
Web / Controllers
ads
rides
payments
quotes
auth
Advertisement Services
DiscountService
AdService
EligibilityService
Core Services
RideService
MatchingService
PaymentService
QuoteService
Shared Modules
EventBus
LocationService
PricingService
AuthService
uses
Prisma Client
Cloud/DB
Postgres Database

3. Class Diagram



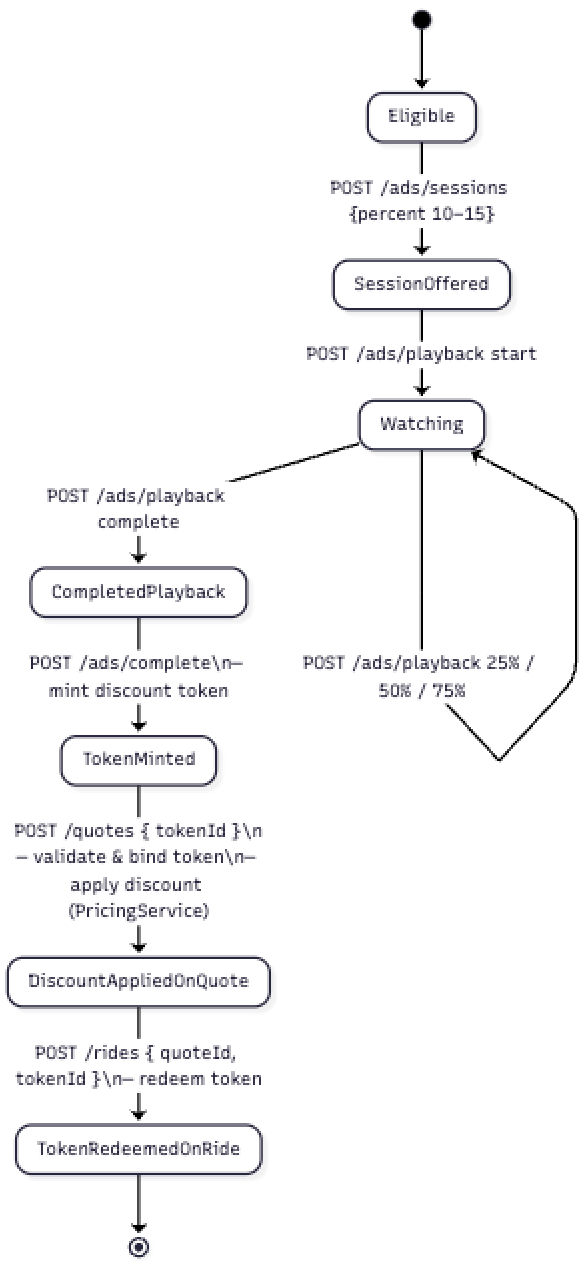
4. List of Classes

- AdsController (Express Router)
  - Purpose: HTTP endpoints for ad eligibility, session, playback, completion, token redeem.
  - Methods: getEligibility, createSession, recordPlayback, complete, redeemToken
  - Depends on: AuthService, AdService, DiscountService
  - Runs on: server
  - Source: backend/src/web/ad.controller.ts:1
- AdService
  - Purpose: Manage ad session lifecycle and trigger token issuance.
  - Methods: createSession(riderId, percent), recordPlayback(sessionId, event, ts?), completeSession(sessionId)
  - Depends on: EligibilityService, DiscountService, EventBus, Prisma (AdSession persistence)
  - Runs on: server
  - Source: backend/src/ad/ad.service.ts:1
- DiscountService
  - Purpose: Issue, validate, and redeem discount tokens bridging ads and discounts.
  - Methods: mintToken(sessionId, riderId, percent), validateToken(tokenId, riderId, ctx), redeemToken(tokenId, rideId, ctx)
  - Depends on: EventBus, Prisma (DiscountToken persistence)
  - Runs on: server
  - Source: backend/src/ad/discount.service.ts:1
- EligibilityService
  - Purpose: Enforce cooldown and daily caps for ad viewing.
  - Methods: checkRider(riderId), cooldownExpiresAt(riderId), registerCompletion(riderId, completedAt?), clear(riderId?)
  - Depends on: Internal in-memory state (no DB)
  - Runs on: server
  - Source: backend/src/ad/eligibility.service.ts:1
- AuthService
  - Purpose: JWT verification middleware and role checks for protected ad routes.
  - Methods: required(req,res,next), optional(req,res,next), requireRole(role)
  - Depends on: jsonwebtoken, env secret
  - Runs on: server
  - Source: backend/src/shared/auth.service.ts:1
- EventBus
  - Purpose: In-process pub/sub for ad lifecycle events (e.g., session completed, token minted/redeemed).
  - Methods: publish(topic, payload), subscribe(topic, handler)
  - Runs on: server
  - Source: backend/src/shared/eventBus.ts:1
- PrismaClient
  - Purpose: Single ORM interface to Postgres for adSession and discountToken entities.

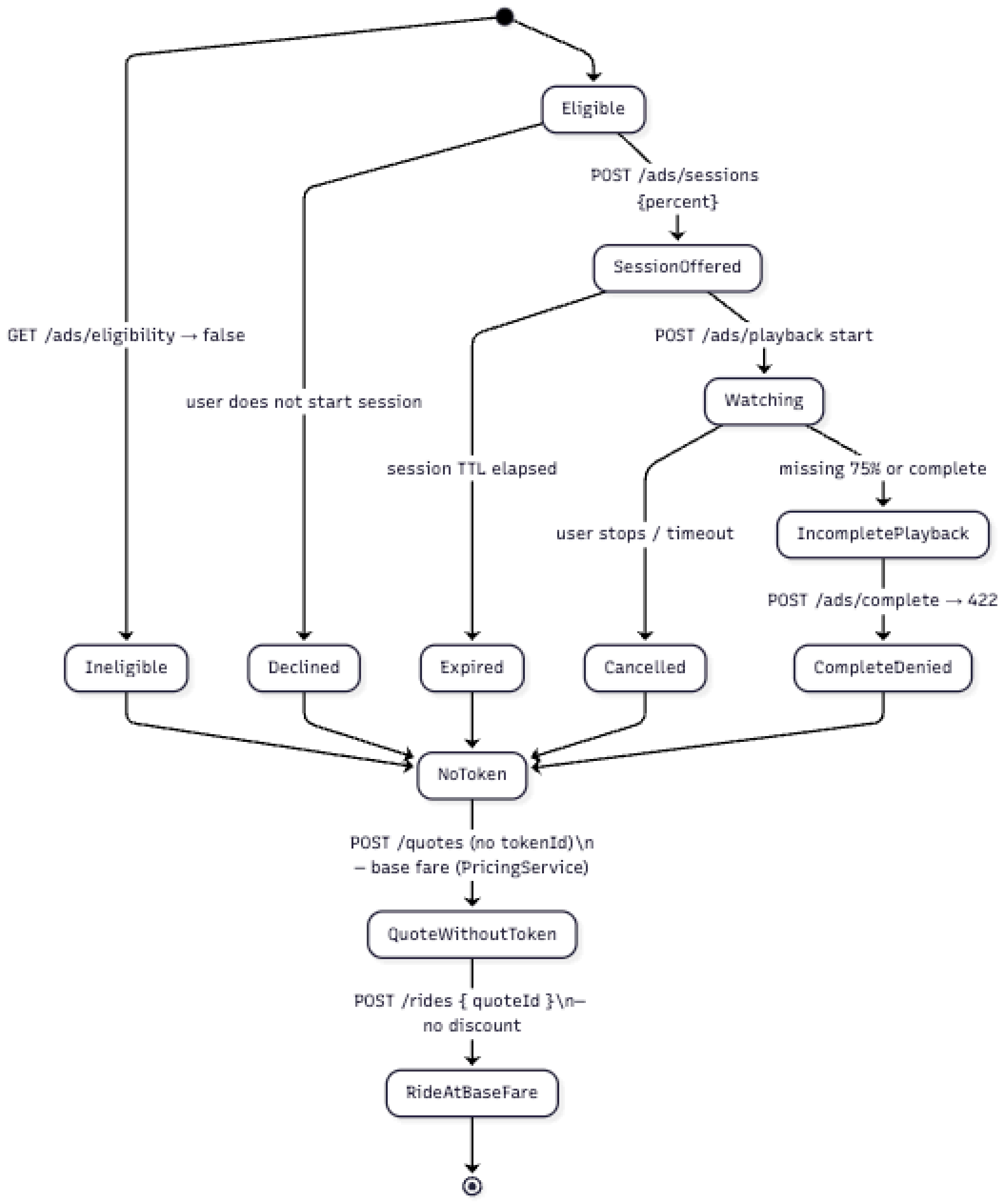
    - Used by: AdService, DiscountService
    - Runs on: server (connects to DB)
    - Source: backend/src/workbench/prisma.ts:1
- Postgres
    - Purpose: Persistent data store for ad sessions and discount tokens.
    - Accessed via: PrismaClient
    - Runs on: cloud/DB environment

5. State Diagrams

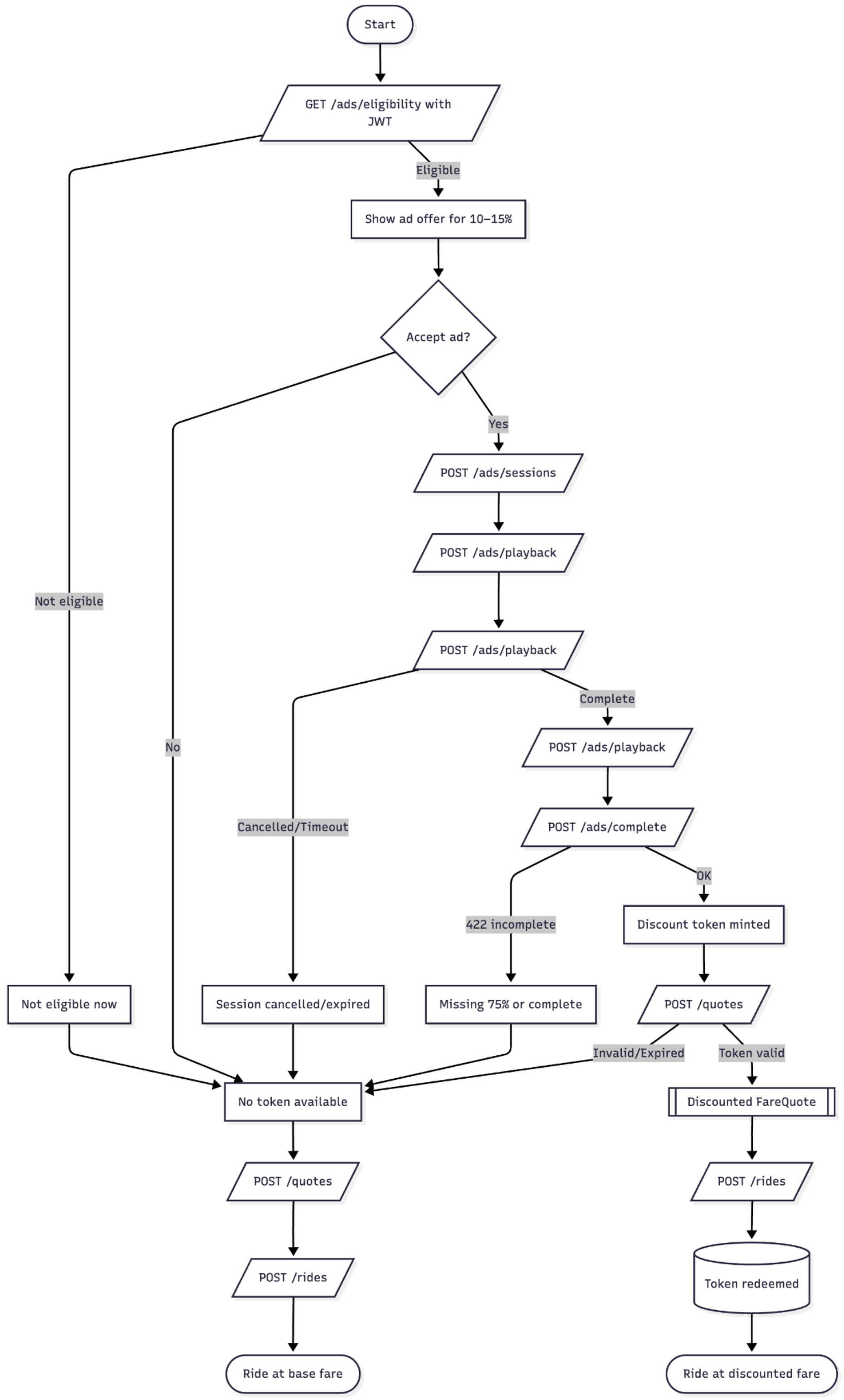
Accepts Ad (Discount Applied)


Eligible
POST /ads/sessions {percent 10–15}
SessionOffered
POST /ads/playback start
Watching
POST /ads/playback complete
POST /ads/playback 25% / 50% / 75%
CompletedPlayback
POST /ads/complete\n– mint discount token
TokenMinted
POST /quotes { tokenId }\n – validate & bind token\n– apply discount (PricingService)
DiscountAppliedOnQuote
POST /rides { quoteId, tokenId }\n– redeem token
TokenRedeemedOnRide

Rejects Ad (No Discount)



6. Flow Chart


Start
GET /ads/eligibility with JWT
Eligible
Not eligible
Show ad offer for 10–15%
Accept ad?
Yes
No
POST /ads/sessions
POST /ads/playback
POST /ads/playback
Complete
Cancelled/Timeout
POST /ads/playback
POST /ads/complete
OK
422 incomplete
Discount token minted
Not eligible now
Session cancelled/expired
Missing 75% or complete
POST /quotes
Invalid/Expired
Token valid
No token available
Discounted FareQuote
POST /quotes
POST /rides
POST /rides
Token redeemed
Ride at base fare
Ride at discounted fare

7. Development Risks and Failures

Failure Modes

1. ADF-1 — Token/Quote Binding Drift

- What: Discount token is validated at quote time but not enforced at ride time (e.g., ride created without the bound token, or with a different token), or binding lost due to state drift.
- Likelihood: Medium
- Impact: High (revenue leakage or failed ride creation causing abandonment)
- Diagnostics (test spec)
    - Unit: Quote with token → attempt ride without token → expect 400 “Discount token required for discounted quote”.
    - Unit: Quote with token A → attempt ride with token B → expect 400 “token does not match quote”.
    - Integration: Chaos test to clear in-memory quote cache before ride; assert ride creation still rejects or revalidates.
    - Observability: Alert on spikes of 400/409 for “token mismatch/required”.
- Recovery
    - Identify affected rides where discountedAmount present but no redeemed token; retro-adjust fares or issue credits.
    - Invalidate any mis-bound tokens and notify impacted riders.
    - Add strict server enforcement (already in RideService) plus an idempotent re-validation at ride creation.
    - Add metrics for quoteId↔tokenId binding success; add canary tests in CI.

2. ADF-2 — Partial Persistence on Completion

- What: Ad session completes; token minted, but DB update linking tokenId to session fails (or vice versa). Repeating completion mints duplicates or returns inconsistent state.
- Likelihood: Medium
- Impact: Medium-High (duplicate tokens, redemption confusion)
- Diagnostics (test spec)
    - Unit: Simulate failure between mintToken() and session update; assert idempotency (second complete returns the original token).
    - Data check: Sessions with status COMPLETED but null tokenId; tokens whose sessionId has no session tokenId linkage.
    - Observability: Alert on repeated completes for same sessionId; count mint vs. complete deltas.
- Recovery
    - Backfill: For COMPLETED sessions without tokenId, link the latest ACTIVE token with matching sessionId (or invalidate extras).
    - Deduplicate: Revoke duplicate tokens (EXPIRED/REVOKED) keeping the first ACTIVE.

- Add transaction or idempotency key: wrap completeSession in a transaction, or lookup-then-mint with upsert by sessionId.

3. ADF-3 — Token TTL/Clock Skew Fallout

- What: Token validated and bound to quote, but expires before ride creation due to short TTL or server/client clock skew; users hit expired errors at redeem.
- Likelihood: Medium-High
- Impact: Medium (conversion loss, user frustration)
- Diagnostics (test spec)
    - Unit: Validate token at quote, wait > TTL, attempt ride → expect 410 EXPIRED; verify clear message and re-quote path.
    - Skew test: Simulate server clock offset; assert grace logic prevents erroneous expiry.
    - Metrics: Alert on spikes of 410 at /ads/token/redeem or ride creation path.
- Recovery
    - Introduce short grace period on redeem after quote binding (e.g., +60–120s).
    - Offer automatic re-quote with applied discount if expiry occurred within grace; otherwise prompt to re-watch ad.
    - Ensure NTP clock sync in infrastructure; lengthen TTL carefully if warranted.

4. ADF-4 — Playback Sequence Bypass/Fraud

- What: Client forges playback events (out-of-order or without sufficient watch) to mint tokens without real viewing; or accepts “complete” without 75% checkpoint.
- Likelihood: Medium
- Impact: High (discount abuse; direct revenue loss)
- Diagnostics (test spec)
    - Unit: Send complete without start/75% → expect 422; send events out of order → expect 422.
    - Fuzz: Randomized event sequences; assert only valid sequence yields token.
    - Anomaly detection: Alert on rapid-fire complete events per riderId, abnormal watch times, or excessive tokens/day.
- Recovery
    - Invalidate suspicious tokens; throttle or block abusive accounts/devices.
    - Strengthen server validation: enforce minimal elapsed time between start→75%→complete; per-rider rate limits; signed session IDs.
    - Add provider callbacks (if used) and verify server-to-server beacons instead of client-only signals.

Ranking (Likelihood, Impact)

- Likelihood (highest → lowest): ADF-3 (Med-High), ADF-1 (Med), ADF-2 (Med), ADF-4 (Med)
- Impact (highest → lowest): ADF-4 (High), ADF-1 (High), ADF-2 (Med-High), ADF-3 (Med)

8. Technology Stack

1. TECH-01 — Node.js (>= 18.x LTS)
  - Used for: Runtime for backend/server and Node test runner.
  - Why: Mature ecosystem, first-class ESM support, stable LTS, wide tooling.
  - URLs: source https://github.com/nodejs/node | author Node.js Foundation/OpenJS | docs https://nodejs.org

2. TECH-02 — TypeScript (5.6.3)
  - Used for: Static typing across backend code.
  - Why: Type safety, IDE support, better refactoring vs plain JS.
  - URLs: source https://github.com/microsoft/TypeScript | author Microsoft | docs https://www.typescriptlang.org

3. TECH-03 — tsx (4.19.2)
  - Used for: TS/ESM dev runner (npm run dev, tests import).
  - Why: Fast startup, zero-config ESM/TS execution vs ts-node.
  - URLs: source https://github.com/privatenumber/tsx | author privatenumber | docs https://github.com/privatenumber/tsx#readme

4. TECH-04 — Express (4.19.2)
  - Used for: HTTP server and routing (controllers).
  - Why: De facto standard, minimalistic, rich middleware ecosystem.
  - URLs: source https://github.com/expressjs/express | author ExpressJS | docs https://expressjs.com

5. TECH-05 — express-async-errors (3.1.1)
  - Used for: Propagate async errors to Express error handler.
  - Why: Simple drop-in vs manual try/catch wrappers.
  - URLs: source https://github.com/davidbanham/express-async-errors | author David Banham | docs README

6. TECH-06 — cors (2.8.5)
  - Used for: CORS headers on API.
  - Why: Lightweight, widely used with Express.
  - URLs: source https://github.com/expressjs/cors | author ExpressJS | docs README

7. TECH-07 — dotenv (16.4.5)
  - Used for: Load environment variables for dev.
  - Why: Standard for .env-based config.
  - URLs: source https://github.com/motdotla/dotenv | author motdotla | docs https://dotenvx.com and README

8. TECH-08 — jsonwebtoken (9.0.2)

- Used for: JWT signing/verification in AuthService.
- Why: Maintained, interoperable, battle-tested.
- URLs: source https://github.com/auth0/node-jsonwebtoken | author Auth0 | docs README

9. TECH-09 — bcryptjs (2.4.3)
- Used for: Password hashing/verification for login.
- Why: Pure JS (no native build) and widely used.
- URLs: source https://github.com/dcodeIO/bcrypt.js | author dcodeIO | docs README

10. TECH-10 — zod (3.23.8)
- Used for: Request validation in controllers.
- Why: TS-first schema + parse, great DX vs Joi/Yup.
- URLs: source https://github.com/colinhacks/zod | author Colin McDonnell (colinhacks) | docs https://zod.dev

11. TECH-11 — Prisma Client (backend ^5.19.1, root ^5.22.0)
- Used for: Type-safe ORM to PostgreSQL.
- Why: Rich schema, migrations, generated types; faster than hand-written SQL for most ops.
- URLs: source https://github.com/prisma/prisma | author Prisma | docs https://www.prisma.io/docs

12. TECH-12 — Prisma CLI (backend ^5.19.1, root ^5.22.0)
- Used for: Generate client, schema pushes/migrations.
- Why: Integrated with Prisma schema/workflows.
- URLs: source https://github.com/prisma/prisma | author Prisma | docs https://www.prisma.io/docs

13. TECH-13 — PostgreSQL (14+)
- Used for: Primary relational database.
- Why: Reliability, strong SQL, robust ecosystem.
- URLs: source https://github.com/postgres/postgres | author PostgreSQL Global Development Group | docs https://www.postgresql.org/docs/

14. TECH-14 — PostGIS (3+)
- Used for: Geospatial types/functions (ST_MakePoint, SRID) in ride storage/queries.
- Why: Native geo support in Postgres; accurate distance/point storage.
- URLs: source https://github.com/postgis/postgis | author PostGIS Project | docs https://postgis.net/documentation/

15. TECH-15 — ESLint (backend ^9.11.0, frontend ^9.36.0)
- Used for: Linting JS/TS code.
- Why: Standard linter, modern rules/plugins.
- URLs: source https://github.com/eslint/eslint | author ESLint Team | docs https://eslint.org

16. TECH-16 — React (19.1.1) + React DOM (19.1.1)
  - Used for: Frontend UI components and rendering.
  - Why: Component model, ecosystem, team familiarity.
  - URLs: source https://github.com/facebook/react | author Meta Open Source | docs https://react.dev and https://react.dev/reference/react-dom

17. TECH-17 — Vite (7.1.7) + @vitejs/plugin-react (5.0.4)
  - Used for: Frontend dev server and build tool.
  - Why: Fast HMR, modern build, simpler than webpack/CRA.
  - URLs: source https://github.com/vitejs/vite and https://github.com/vitejs/vite-plugin-react | author Vite Team (Evan You et al.) | docs https://vitejs.dev and plugin docs

18. TECH-18 — uuid (13.0.0)
  - Used for: Generating IDs on the frontend.
  - Why: Small, widely trusted ID generation.
  - URLs: source https://github.com/uuidjs/uuid | author uuidjs | docs README

19. TECH-19 — DefinitelyTyped type packages
  - Used for: Type definitions in TS (backend and frontend).
  - Why: TS typings for JS libs.
  - Versions: @types/node (^22.18.13), @types/express (^4.17.21), @types/jsonwebtoken (^9.0.6), @types/bcryptjs (^2.4.6), @types/cors (^2.8.17), @types/react (^19.1.16), @types/react-dom (^19.1.9)
  - URLs: source https://github.com/DefinitelyTyped/DefinitelyTyped | author DT maintainers | docs README per package

20. TECH-20 — pnpm (10.19.0)
  - Used for: Package manager (workspace), deterministic installs.
  - Why: Faster installs and disk efficiency vs npm/yarn.
  - URLs: source https://github.com/pnpm/pnpm | author pnpm | docs https://pnpm.io

21. TECH-21 — JSON (Fetch API, browser)
  - Used for: Client–server data exchange.
  - Why: Native, ubiquitous, low overhead for REST.
  - URLs: source https://developer.mozilla.org/docs/Web/API/Fetch_API | author WHATWG/MDN | docs MDN

22. TECH-22 — CORS (HTTP standard)
  - Used for: Cross-origin access between frontend and backend.
  - Why: Required for browser security model.
  - URLs: source https://fetch.spec.whatwg.org/#http-cors-protocol | author WHATWG | docs MDN https://developer.mozilla.org/docs/Web/HTTP/CORS

9. APIs

Ad APIs

- **GET /ads/eligibility**
    - Purpose: Tell a rider if they can start an ad session now (cooldown/daily cap).
    - Auth: Required (Bearer JWT).
    - Params: none.
    - Response: { isEligible: boolean, cooldownEndsAt?: string-ISO }
    - Errors: 401 invalid/missing token.
- **POST /ads/sessions**
    - Purpose: Create an ad session offer for a rider at a given discount percent.
    - Auth: Required.
    - Body: { percent: number } (int, 10–15 inclusive)
    - Response: { sessionId: string-uuid, provider: string, percent: number, expiresAt: string-ISO }
    - Errors: 400 percent out of range; 409 not eligible (cooldown info attached); 401 auth.
- **POST /ads/playback**
    - Purpose: Record playback checkpoints for a session.
    - Auth: Required.
    - Body: { sessionId: string-uuid, event: "start" | "25%" | "50%" | "75%" | "complete", ts?: string-ISO-with-offset }
        - If ts provided, must be valid ISO datetime; otherwise server timestamps.
    - Response: { ok: true }
    - Errors: 400 unsupported event or invalid timestamp; 404 session not found; 410 session expired; 409 cancelled/already completed; 422 sequence invalid (e.g., missing “start”).
- **POST /ads/complete**
    - Purpose: Mark ad session complete and mint a discount token.
    - Auth: Required.
    - Body: { sessionId: string-uuid }
    - Response: { tokenId: string, expiresAt: string-ISO }
    - Errors: 404 session not found; 410 session expired; 409 cancelled; 422 missing required checkpoints (need “start”, “75%”, and “complete” recorded).
- **POST /ads/token/redeem**
    - Purpose: Manually redeem a minted token against a ride (alternative to automatic redeem during ride creation).
    - Auth: Required.
    - Body: { tokenId: string, rideId: string-uuid, quoteId?: string-uuid }
    - Response: { state: "REDEEMED" | "ACTIVE" | "EXPIRED" | "REVOKED" } (on success returns REDEEMED)
    - Errors: 404 token not found; 409 not redeemable/bound to another quote; 403 token not owned by rider; 410 token expired.

Related Core APIs (used to apply the ad discount)

- **POST /quotes**
  - Purpose: Generate a fare quote; apply discount if tokenId provided and valid.
  - Auth: Optional overall, but required if tokenId is provided (must identify rider to validate/bind token).
  - Body: { pickup: {lat:number, lon:number}, dest: {lat:number, lon:number}, tokenId?: string, opts?: { vehicleType?: string, pax?: number } }
  - Response: { id, amount, surge, currency, expiresAt, etaMinutes, discountApplied?: boolean, discountPercent?: number, discountedAmount?: number, discountTokenId?: string }
  - Errors: 400 when token provided without authenticated rider.
- **POST /rides**
  - Purpose: Create a ride from a prior quote; enforces discount token binding and redeems it automatically.
  - Auth: Required.
  - Body: { pickup: {lat,lon}, dest: {lat,lon}, quoteId: string-uuid, tokenId?: string }
  - Behavior: If the quote is discounted, tokenId must be present and match the quote's discountTokenId; token is redeemed on success.
  - Response: Ride object with fare fields (discount applied if valid).
  - Errors: 400 token required/mismatch/invalid association; 403 rider mismatch; 404/expired quote; 410 expired token.

10. Public Interfaces

1. AdsController (Web/Controllers module)

- Public methods
  - Across modules (called by Client via HTTP)
    - getEligibility(req, res)
    - createSession(req, res)
    - recordPlayback(req, res)
    - complete(req, res)
    - redeemToken(req, res)
- Uses from other components
  - From Shared
    - AuthService.required(req,res,next)
  - From Advertisement
    - EligibilityService.checkRider(riderId)
    - AdService.createSession(riderId, percent)
    - AdService.recordPlayback(sessionId, event, ts?)
    - AdService.completeSession(sessionId)
    - DiscountService.redeemToken(tokenId, rideId, { quoteId?, riderId })
- Notes
  - Exposes the only public surface for Ads to the Client.
  - Handles input validation and auth; delegates to services.

2. AdService (Advertisement Services module)

- Public methods
  - Within same component (called by classes in Advertisement)
    - Used by AdsController (Web): createSession, recordPlayback, completeSession
  - Across components in the same module
    - Calls to EligibilityService.checkRider/registerCompletion
    - Calls to DiscountService.mintToken
  - Across modules
    - None exposed to other modules (Core doesn't call AdService directly)
- Uses from other components
  - From Shared
    - EventBus.publish(topic, payload) — emits "ads.session.completed"
  - From Persistence (via ORM)
    - PrismaClient.adSession.create/findUnique/update (conceptually; data I/O)
- Notes
  - Owns ad session lifecycle; enforces playback rules.

3. DiscountService (Advertisement Services module)

- Public methods
    - Within same component (called by classes in Advertisement)
        - mintToken(sessionId, riderId, percent) — used by AdService
    - Across components in the same module
        - None (peer services don't invoke DiscountService besides AdService)
    - Across modules (called by non-Ads modules)
        - validateToken(tokenId, riderId, { quoteId? }) — used by QuoteService (Core) to apply discount
        - redeemToken(tokenId, rideId, { quoteId?, riderId? }) — used by RideService (Core) or AdsController route
        - fetch(tokenId) — used by AdService for idempotent completion
- Uses from other components
    - From Shared
        - EventBus.publish(topic, payload) — emits "ads.token.minted" / "ads.token.redeemed"
    - From Persistence (via ORM)
        - PrismaClient.discountToken.create/findUnique/update (conceptual data I/O)
- Notes
    - Bridge between Ads and Core; centralizes token lifecycle and integrity.

4. EligibilityService (Advertisement Services module)

- Public methods
    - Within same component (called by classes in Advertisement)
        - checkRider(riderId) — used by AdsController and AdService
        - registerCompletion(riderId, completedAt?) — used by AdService
        - cooldownExpiresAt(riderId) — sometimes used by controller logic
        - clear(riderId?) — maintenance/testing
    - Across components in the same module
        - None beyond AdService/AdsController
    - Across modules
        - None (Core does not call eligibility)
- Uses from other components
    - None (self-contained in-memory policy)
- Notes
    - Implements daily cap and cooldown windows; stateless API with internal state map.

5. Shared Components (Shared module)

- AuthService
    - Public methods
        - Across modules
            - required(req,res,next) — used by AdsController (and other controllers)
            - optional(req,res,next)
            - requireRole(role)

    - Uses from other components
        - None (relies on jsonwebtoken/env)
    - Notes: Middleware to guard routes and attach req.user.
- EventBus
    - Public methods
        - Across modules
            - publish(topic, payload) — used by AdService, DiscountService (and others)
            - subscribe(topic, handler) — used by any module to react to events
    - Uses from other components
        - None (in-process pub/sub)
    - Notes: Lightweight decoupling for domain events.

11. Data Schemas

🧩 Ad Feature Schema Overview

1. AdSession
Runtime: AdService, AdSessionRepository → AdSessionRecord
Columns:

- id uuid
- riderId uuid (FK → User.id)
- percent int (10–15)
- provider text (e.g., “AcmeAds”)
- status AdStatus (OFFERED|WATCHING|COMPLETED|CANCELLED)
- startedAt, completedAt, expiresAt, createdAt timestamptz
- playbackEvents jsonb ("start", "25%", "50%", "75%", "complete" → ISO timestamps)

One-to-one with DiscountToken via DiscountToken.sessionId; no tokenId in AdSession.
Notes:

- status: Postgres enum
- playbackEvents: fixed-key JSONB

Size: ~324–360 B/row

---

2. DiscountToken
Runtime: DiscountService, AdService, DiscountTokenRepository → DiscountTokenRecord
Columns:

- id text (ULID)
- riderId uuid (FK → User.id)
- percent int
- state TokenState (ACTIVE|REDEEMED|EXPIRED|REVOKED)
- quoteId, redeemedRideId text? (UUID strings, no FK)
- expiresAt, createdAt timestamptz
- sessionId uuid (unique FK → AdSession.id)

Notes:

- id: ULID text (lexicographically sortable)
- sessionId: enforces 1:1 relation with AdSession

Size: ~86–158 B/row

---

3. Enums

Enum	Values	Used By

AdStatus	OFFERED, WATCHING, COMPLETED, CANCELLED	AdSession.status
TokenState	ACTIVE, REDEEMED, EXPIRED, REVOKED	DiscountToken.state

4. Relationships

- AdSession ↔ DiscountToken: 1:1 (DiscountToken.sessionId)
- User ↔ (AdSession, DiscountToken): 1:N via riderId
- Runtime: AdService orchestrates, DiscountService manages tokens.

5. Storage Functions (Optional)

- AdSession JSONB: ≈ 60 + 33n B → ~225 B for n = 5
- DiscountToken: ≈ 86 + 36·hasQuote + 36·hasRedeemed B

12. Security and Privacy

🔒 PII Storage Overview

Temporary PII (in memory)

- **HTTP requests:**
  Includes `email`, `password` (login), `Authorization: Bearer <JWT>` (with `sub` userId), `sessionId`, and `tokenId`.
  Used for authentication, authorization, and ad session/token validation.
  Lives only in Express process memory during the request.
- **Quote & ad session context:**
  Holds `riderId`, pickup/destination `{lat, lon}`, `playbackEvents` timestamps, and optional `tokenId`.
  Used to generate quotes, validate discount eligibility, and bind tokens.
  Stored transiently in in-memory `QuoteStore`; durable playback lives in Postgres.
- **Dev-only in-memory DB:**
  Contains seeded test users (`name, email, passwordHash`), drivers, vehicles, and live driver locations.
  Used only for local testing; never used in production.

Long-Term PII (Postgres)

- **User:** `id, name, email, password(bcrypt), rating, createdAt`
  → For identity, authentication, and account management.
- **Driver:** `id, name, rating, status`
  → For assignments and service operations.
- **Vehicle:** `id, make, model, plate, type, driverId`
  → For regulatory and operational tracking.
- **Ride:** `id, riderId, driverId?, pickup/destination (PostGIS), fare/time fields, discountTokenId?`
  → Core transactional record with essential location data.
- **PaymentIntent:** `id, rideId, amount, status, method?, timestamps`
  → Tracks payments (no card data stored).
- **AdSession:** `id, riderId, percent, provider, status, playbackEvents, timestamps`
  → Records ad viewing for discount eligibility.

- **DiscountToken:** `id (ULID text), riderId, percent, state, quoteId?, expiresAt, redeemedRideId?, sessionId, createdAt`
  → Issues, validates, and redeems ride discounts.

Mapping Notes

- Passwords are **bcrypt hashes**, never plaintext.
- Pickup/destination use **Postgres geography**, not addresses.
- `DiscountToken.id` is a **ULID text**, lexicographically sortable.
- `quoteId` and `redeemedRideId` are UUID-like text fields, validated by app logic (no FK).

Approx. Storage Sizes per Row

- User: ~150–200 B
- Driver: ~80–120 B
- Vehicle: ~80–140 B
- Ride: ~160–220 B
- PaymentIntent: ~80–120 B
- AdSession: ~300–350 B
- DiscountToken: ~100–160 B

Security Responsibilities

- **Postgres (Production):**
  - *DBA/SRE* — encryption at rest, backups, access control.
  - *Backend Lead* — schema design, least-privileged roles.
  - *Security Engineer* — vulnerability management, audit policy.
- **Backups/Snapshots:**
  - *SRE* — integrity, encrypted storage, retention.
  - *Security Engineer* — key management, access review.
- **Application Secrets (JWT, DB creds):**
  - *SRE* — secret rotation and scope management.
  - *Security Engineer* — policy and audit.
  - *Backend Lead* — correct usage within services.

Security Officer (Auditing)

Role: Security Officer / Data Protection Officer (DPO)
Responsibilities:
Oversee audits of database and backup access, secret rotation, and PII retention policies. Manage incident response and ensure compliance with data protection laws.
Contact: `<Name> <email>`

Roles to Assign:

- **DBA/SRE:** `<Name> <email>`
- **Backend Lead:** `<Name> <email>`
- **Security Engineer:** `<Name> <email>`
- **Security Officer (DPO):** `<Name> <email>`

13. Risks to Completion

RSK-01 – Ambiguous ad acceptance criteria

- **What:** Unclear playback rules (timing, checkpoints, grace).
- **Impact/Likelihood:** High / Medium
- **Triggers:** QA disagreement, UAT failures, edge-case bugs.
- **Mitigation/Owner:** Finalize and freeze a testable playback spec (timeline + checkpoints + grace); *Product + Backend/Frontend Leads.*

RSK-02 – Backend–frontend contract drift

- **What:** Request/response schema mismatches (timestamps, error codes, token binding).
- **Impact/Likelihood:** Medium / High
- **Triggers:** 4xx spikes, UI parse errors, failing contract tests.
- **Mitigation/Owner:** Typed API client or OpenAPI + CI schema tests; *Frontend + Backend.*

RSK-03 – Data model migration risk (Postgres enums/relations)

- **What:** Enum or 1:1 relation changes causing migration errors or data mismatch.
- **Impact/Likelihood:** High / Medium
- **Triggers:** Prisma migration failures, missing backfills.
- **Mitigation/Owner:** Blue-green or expand/contract migration with backfill scripts; *DBA/SRE + Backend.*

RSK-04 – Idempotency gaps on session completion

- **What:** Token minted but session not updated (or vice versa) under transient failure.
- **Impact/Likelihood:** Medium-High / Medium
- **Triggers:** Duplicate tokens, repeated "complete" calls.
- **Mitigation/Owner:** Make `completeSession` idempotent (lookup-first, upsert by `sessionId`, transaction); *Backend.*

RSK-05 – Token TTL / clock skew churn

- **What:** Token expires between quote and ride creation; inconsistent client/server clocks.
- **Impact/Likelihood:** Medium / Medium-High
- **Triggers:** 410 EXPIRED spikes, user complaints.
- **Mitigation/Owner:** Add short redeem grace window, enforce NTP, support UI re-quote; *Backend + SRE + Frontend.*

RSK-06 – Fraud controls not finalized

- **What:** Weak playback validation allowing discount abuse.
- **Impact/Likelihood:** High / Medium
- **Triggers:** Anomalous mint rates, short watch times.
- **Mitigation/Owner:** Enforce playback sequence + min elapsed time, rate limits, anomaly alerts; *Security + Backend.*

RSK-07 – Memory vs production parity

- **What:** Dev in-memory DB diverges from Postgres behavior (JSONB, geography, indexes).
- **Impact/Likelihood:** Medium / Medium
- **Triggers:** Works in dev, fails in staging/prod.
- **Mitigation/Owner:** Add CI pipeline with seeded Postgres test; *SRE* + *Backend.*

RSK-08 – Performance under load (playback writes)

- **What:** Hot JSONB updates on `AdSession` during heavy campaigns.
- **Impact/Likelihood:** Medium / Medium
- **Triggers:** Latency or lock spikes.
- **Mitigation/Owner:** Batch or append-only writes, partitioning, index tuning; *DBA/SRE* + *Backend.*

RSK-09 – Observability gaps

- **What:** Missing logs/metrics for session lifecycle, token mint/redeem, binding errors.
- **Impact/Likelihood:** Medium / Medium
- **Triggers:** Hard-to-diagnose incidents, long MTTR.
- **Mitigation/Owner:** Add structured logs, metrics, alerts for key flows; *SRE.*

RSK-10 – Compliance / privacy review delays

- **What:** Ad tracking and discounting require policy/legal review.
- **Impact/Likelihood:** High / Medium
- **Triggers:** Launch blocks, late policy changes.
- **Mitigation/Owner:** Early DPO / Security sign-off, retention map, DPIA; *Product* + *Security/Legal.*

RSK-11 – API rate-limit / abuse handling

- **What:** Bots spamming playback/complete endpoints.
- **Impact/Likelihood:** Medium / Medium
- **Triggers:** Traffic spikes, 5xxs, token floods.
- **Mitigation/Owner:** Per-user/IP limits, circuit breakers; *SRE* + *Backend.*

RSK-12 – Frontend UX edge cases

- **What:** Poor handling of expired tokens, ineligible riders, or incomplete playback.
- **Impact/Likelihood:** Medium / Medium
- **Triggers:** Drop-offs in funnel metrics.
- **Mitigation/Owner:** Add clear retry / fallback flows, user messaging; *Frontend* + *Product.*

14. GPT log history

Main one
(codex CLI exported in markdown because codex in vsc cannot provide a link to a chat log like chat does, so this is a work around using an extension called SpecStory)

- https://github.com/yt249/team-code-cruise/blob/main/docs/userstory%203%20log.md

(markdown uploaded to github)

Supporting one for other questions

- https://chatgpt.com/share/69050b46-1f60-8007-83e3-7a7205e05362