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)

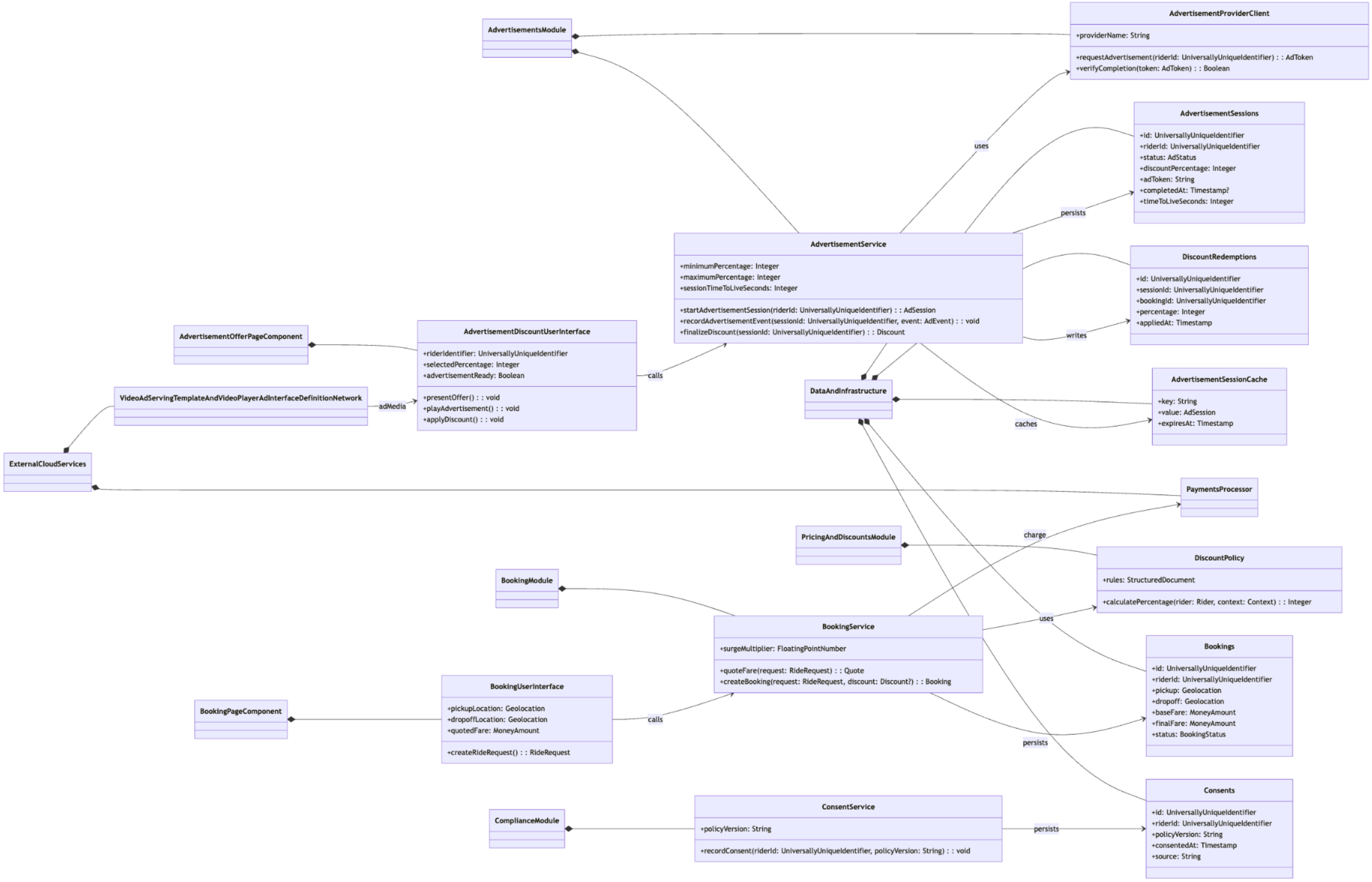
2. Architecture Diagram

Rationale:

- **Separation of concerns:** Client handles UX and ad playback; backend owns verification, discount issuance, booking, consent; external ad network remains a black box. This limits blast radius and simplifies testing.
- **Optional path by design:** Ad flow sits beside (not inside) booking so riders can skip without blocking checkout.
- **Explicit information flows:** Labeled edges (e.g., `verifyCompletion`, `discountPercentage`) make security reviews and contract tests unambiguous.

Legend:

- **Component / Module grouping:**
  - `AdvertisementOfferPageComponent` and `BookingPageComponent` = one **component box per page** (MVC client).
  - `AdvertisementsModule`, `BookingModule`, `PricingAndDiscountsModule`, `ComplianceModule` = backend service modules.
  - `ExternalCloudServices` groups third-party systems.
  - `DataAndInfrastructure` groups databases and caches.
  - Composition arrows `*--` indicate **containment/ownership** (i.e., what lives inside each component/module "box").
- **Classes (one box per class):**
  Each class lists **Fields** with **types** and **Methods** with **parentheses** `()` (e.g., `startAdvertisementSession()`).
- **Types shown in fields:**
  - Common domain types are written out (e.g., **UniversallyUniqueIdentifier**, **Geolocation**, **MoneyAmount**, **Timestamp**, **StructuredDocument**).
  - If you need examples for rarely used types, add a short note in your doc (e.g., *Geolocation example: latitude 40.44, longitude −79.95*).
- **Arrows between classes:**
  - `-->` = **uses/calls** (information or control flow).
  - `*--` = **composition/ownership** (container→contained, modeling the "box" requirement).
  - Labels on arrows describe the interaction (e.g., `adMedia`, `charge`) when helpful.


AdvertisementsModule
AdvertisementProviderClient
+providerName: String
+requestAdvertisement(riderId: UniversallyUniqueIdentifier) : : AdToken
+verifyCompletion(token: AdToken) : : Boolean
uses
AdvertisementSessions
+id: UniversallyUniqueIdentifier
+riderId: UniversallyUniqueIdentifier
+status: AdStatus
+discountPercentage: Integer
+adToken: String
+completedAt: Timestamp?
+timeToLiveSeconds: Integer
persists
AdvertisementService
+minimumPercentage: Integer
+maximumPercentage: Integer
+sessionTimeToLiveSeconds: Integer
+startAdvertisementSession(riderId: UniversallyUniqueIdentifier) : : AdSession
+recordAdvertisementEvent(sessionId: UniversallyUniqueIdentifier, event: AdEvent) : : void
+finalizeDiscount(sessionId: UniversallyUniqueIdentifier) : : Discount
DiscountRedemptions
+id: UniversallyUniqueIdentifier
+sessionId: UniversallyUniqueIdentifier
+bookingId: UniversallyUniqueIdentifier
+percentage: Integer
+appliedAt: Timestamp
writes
AdvertisementOfferPageComponent
AdvertisementDiscountUserInterface
+riderIdentifier: UniversallyUniqueIdentifier
+selectedPercentage: Integer
+advertisementReady: Boolean
+presentOffer() : : void
+playAdvertisement() : : void
+applyDiscount() : : void
calls
DataAndInfrastructure
AdvertisementSessionCache
+key: String
+value: AdSession
+expiresAt: Timestamp
caches
VideoAdServingTemplateAndVideoPlayerAdInterfaceDefinitionNetwork
adMedia
ExternalCloudServices
PaymentsProcessor
charge
PricingAndDiscountsModule
DiscountPolicy
+rules: StructuredDocument
+calculatePercentage(rider: Rider, context: Context) : : Integer
BookingModule
BookingService
+surgeMultiplier: FloatingPointNumber
+quoteFare(request: RideRequest) : : Quote
+createBooking(request: RideRequest, discount: Discount?) : : Booking
uses
Bookings
+id: UniversallyUniqueIdentifier
+riderId: UniversallyUniqueIdentifier
+pickup: Geolocation
+dropoff: Geolocation
+baseFare: MoneyAmount
+finalFare: MoneyAmount
+status: BookingStatus
BookingPageComponent
BookingUserInterface
+pickupLocation: Geolocation
+dropoffLocation: Geolocation
+quotedFare: MoneyAmount
+createRideRequest() : : RideRequest
calls
persists
Consents
+id: UniversallyUniqueIdentifier
+riderId: UniversallyUniqueIdentifier
+policyVersion: String
+consentedAt: Timestamp
+source: String
ComplianceModule
ConsentService
+policyVersion: String
+recordConsent(riderId: UniversallyUniqueIdentifier, policyVersion: String) : : void
persists

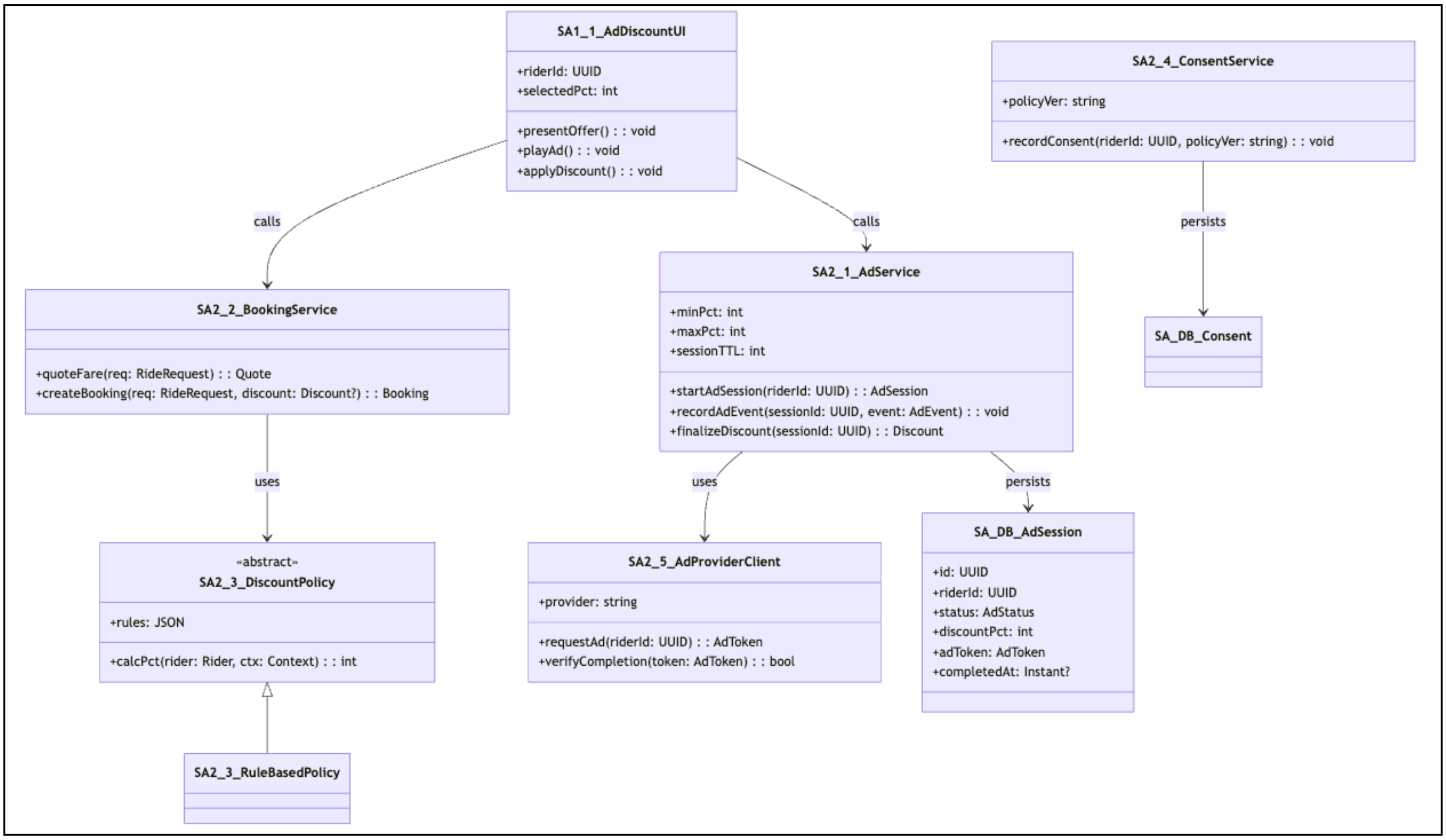
3. Class Diagram

Rationale:

- **One box per class with typed fields & `()` methods:** Enforces clarity and supports code generation/checklists for implementation.
- **Policy object:** `DiscountPolicy` isolates business rules (10–15%) so product can experiment without touching orchestration code.
- **Provider client boundary:** `AdvertisementProviderClient` encapsulates third-party quirks and makes fraud checks testable.

Legend:

- **Class box:** Plain name; one box per class.
- **Fields:** `fieldName: Type` (typed; `Type?` = nullable).
- **Methods:** `methodName(params): ReturnType` — **always ends with** `()`; omit return type if `void`.
- **Grouping:** Components/Modules **own** classes via `*--` (composition = “inside the box”).
- **Relations:** `-->` = uses/calls (information/control flow); Service → Data implies persistence/cache.



4. List of Classes

M1 Client

- **SA1.1 AdDiscountUI** — Presents optional ad/discount UX; captures ad events; passes ad tokens to backend.
- **SA1.2 BookingUI** — Collects ride details; shows fare + applied discount.

M2 Backend

- **SA2.1 AdService** — Orchestrates ad sessions, verifies completion, issues discount.
- **SA2.2 BookingService** — Computes quote and books ride with optional discount.
- **SA2.3 DiscountPolicy** — Strategy to calculate discount % (10–15% bounds for this story).
- **SA2.4 ConsentService** — Records rider consent to ad policy.
- **SA2.5 AdProviderClient** — Talks to external ad networks.

M4 Data

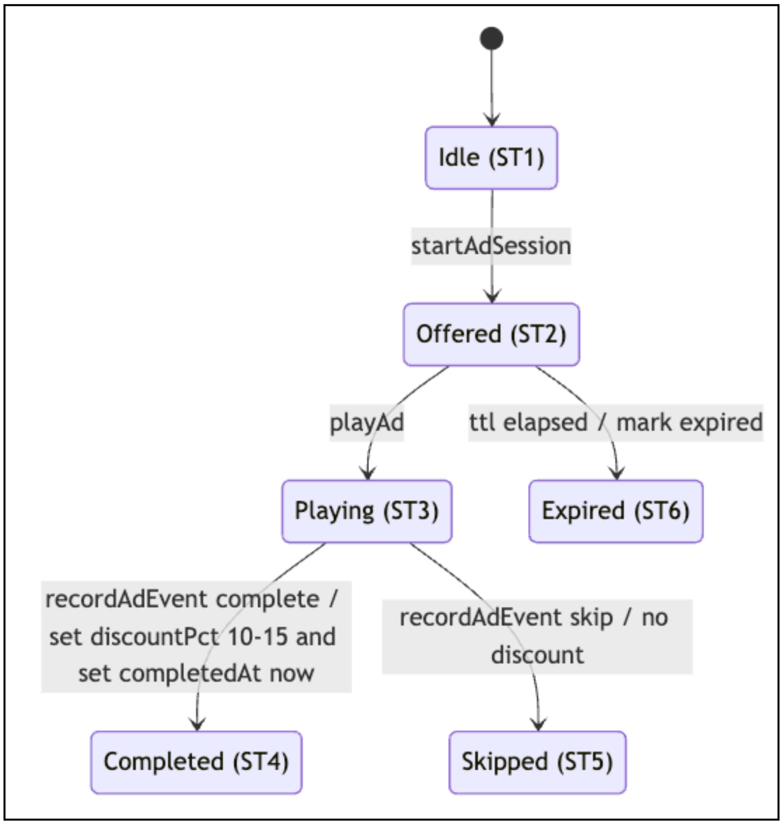
- **SA.DB.AdSession** — Ad session runtime/persisted data (struct).
- **SA.DB.DiscountRedemption** — One-time discount linkage to booking (struct).
- **SA.DB.Consent** — Stored consent record (struct).
- **SA.DB.Booking** — Booking data (struct).

5. State Diagrams

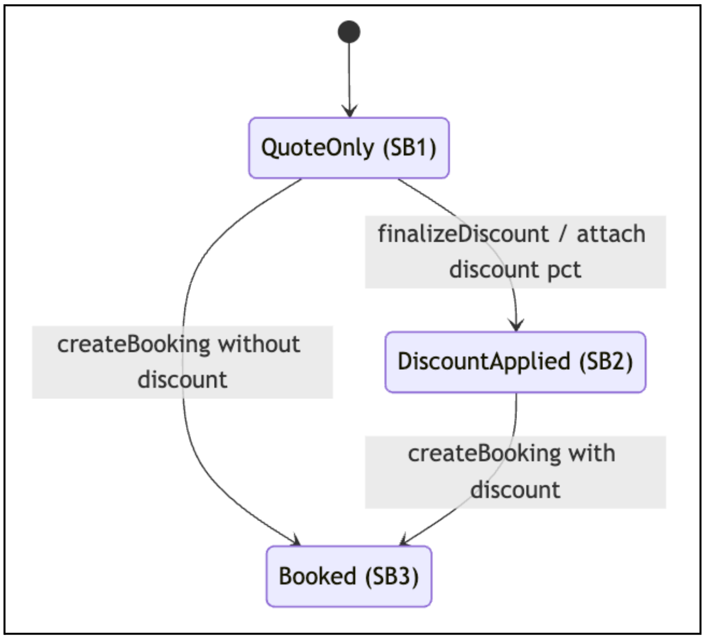
Two focused lifecycles:

- **AdSession:** Idle → Offered → Playing → Completed/Skipped/Expired captures all user/system outcomes and TTL edge cases.
- **Booking:** QuoteOnly → DiscountApplied → Booked cleanly shows discount as a *temporary state*, not a permanent rider attribute.

5a) AdSession lifecycle (label: **ST-AD-1**)


Idle (ST1)
startAdSession
Offered (ST2)
playAd
ttl elapsed / mark expired
Playing (ST3)
Expired (ST6)
recordAdEvent complete / set discountPct 10-15 and set completedAt now
recordAdEvent skip / no discount
Completed (ST4)
Skipped (ST5)

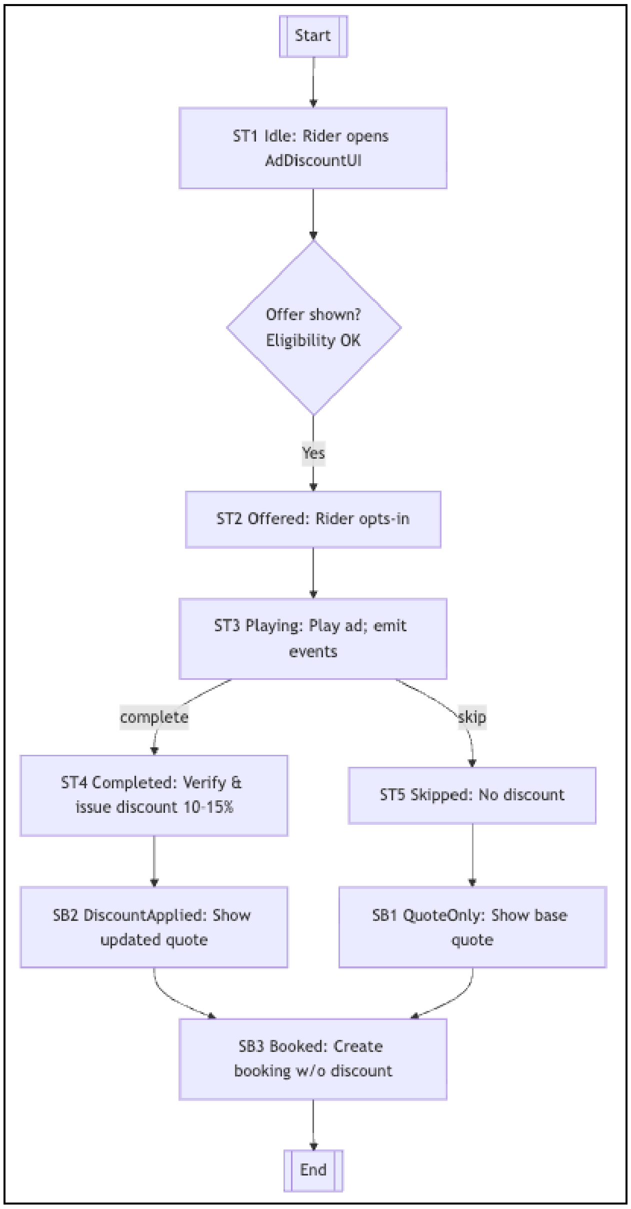
5b) Booking with optional discount (label: **ST-BK-1**)


QuoteOnly (SB1)
finalizeDiscount / attach discount pct
createBooking without discount
DiscountApplied (SB2)
createBooking with discount
Booked (SB3)

6. Flow Chart

Scenario-first view: Shows a happy path (ad completed → discounted quote → booking) and clear fallbacks (skip/expire → base fare).

UX ↔ backend alignment: Each box ties to a screen/action and a backend method to prevent gaps between design and implementation.


Start
ST1 Idle: Rider opens AdDiscountUI
Offer shown? Eligibility OK
Yes
ST2 Offered: Rider opts-in
ST3 Playing: Play ad; emit events
complete
skip
ST4 Completed: Verify & issue discount 10-15%
ST5 Skipped: No discount
SB2 DiscountApplied: Show updated quote
SB1 QuoteOnly: Show base quote
SB3 Booked: Create booking w/o discount
End

7. Development Risks and Failures

- **Runtime**
  - **FAIL-RT-1**: RPC to Ad Network fails → show base fare; retry once; log; degrade gracefully.
  - **FAIL-RT-2**: Session TTL expiry mid-playback → mark **Expired**; remove discount; prompt to retry.
  - **FAIL-RT-3**: Overload spikes on AdService → autoscale; circuit-break verification.
- **Connectivity**
  - **FAIL-CN-1**: Mobile offline during ad → pause; resume on reconnect or cancel to base fare.
  - **FAIL-CN-2**: DB connectivity loss → write to cache/queue; reconcile later.
- **Hardware/Config**
  - **FAIL-HW-1**: Node down → multi-AZ, health checks, rolling restarts.
  - **FAIL-CFG-1**: Bad policy config → schema-validated policy; feature flag rollback.
- **Intruder/Security**
  - **FAIL-SEC-1**: Ad-completion fraud → server-side verify token; no client-only trust.
  - **FAIL-SEC-2**: Bot abuse (fake sessions) → rate-limit per rider/device; CAPTCHA on abuse.
  - **FAIL-SEC-3**: Session hijack → short-lived tokens; TLS; HttpOnly/SameSite cookies.

8. Technology Stack

- **TECH-MOB-1 React Native 0.74** — Mobile UI; cross-platform speed; large ecosystem.

- **TECH-WEB-1 TypeScript 5.x** — Static typing across client/server; safety.

- **TECH-BE-1 Node.js 20** — Backend services; async IO; mature tooling.

- **TECH-API-1 OpenAPI 3.1** — Contract-first APIs; codegen.

- **TECH-DB-1 PostgreSQL 16** — Relational integrity for bookings/discounts.

- **TECH-CACHE-1 Redis 7** — Session/TTL cache for ad sessions.

- **TECH-AD-1 VAST/VPAID-compatible Ad SDK** — Standard ad playback/verification.

- **TECH-OBS-1 OpenTelemetry** — Traces/metrics for ad funnels.

- **TECH-SEC-1 JWT/OAuth2** — AuthN/Z for mobile→backend.

Rationale:

- **Type safety end-to-end:** TypeScript on client/server reduces integration bugs in a flow with many edge cases.
- **PostgreSQL + Redis:** Relational integrity for bookings/payments; TTL cache for short-lived ad sessions keeps latency low without overloading the DB.
- **Standards-based ads/telemetry:** VAST/VPAID compatibility and OpenTelemetry help debug funnel drop-offs quickly.

9. APIs

M2.Backend.SA2.1 AdService (Public)

- `startAdSession(riderId: UUID): AdSession` — creates session; returns ad token & TTL.
- `recordAdEvent(sessionId: UUID, event: "play"|"complete"|"skip"): void`
- `finalizeDiscount(sessionId: UUID): Discount?` — returns `{pct:int, expiresAt:Instant}` or `null`.

AdService (Private)

- `_issueDiscount(session: AdSession): Discount`
- `_validateCompletion(token: AdToken): bool`

M2.Backend.SA2.5 AdProviderClient (Private)

- `requestAd(riderId: UUID): AdToken`
- `verifyCompletion(token: AdToken): bool`

M2.Backend.SA2.2 BookingService (Public)

- `quoteFare(req: RideRequest): Quote`
- `createBooking(req: RideRequest, discount: Discount?): Booking`

M2.Backend.SA2.4 ConsentService (Public)

- `recordConsent(riderId: UUID, policyVer: string): void`

M1.Client.SA1.1 AdDiscountUI (Public to app)

- `presentOffer(): void`, `playAd(): void`, `applyDiscount(): void`

10. Public Interfaces

Within the same component (App) — `AdDiscountUI.presentOffer()`, `applyDiscount()`.

Across components in the same module (Backend) — `AdService.finalizeDiscount()` used by `BookingService`.

Across modules

- **Client→Backend:** `startAdSession`, `recordAdEvent`, `finalizeDiscount`, `quoteFare`, `createBooking`.
- **Backend→Ad Network:** `AdProviderClient.requestAd/verifyCompletion`.
- **Backend→Payments:** `Payments.charge(bookingId, amount)`.

Multi-interface access: Mobile SDK (TypeScript) and REST (JSON). Example REST:
`POST /v1/ad-sessions` → `201 {sessionId, token, ttl}`;
`POST /v1/ad-sessions/{id}/events {event}` → `204`;
`POST /v1/bookings {req, discountSessionId?}` → `201 {bookingId}`.

11. Data Schemas

DB1 AdSessions (owned by `AdService`)

- `id UUID PK`, `rider_id UUID`, `status ENUM('Idle','Offered','Playing','Completed','Skipped','Expired')`, `discount_pct INT NULL`, `ad_token TEXT`, `completed_at TIMESTAMPTZ NULL`, `ttl_sec INT`
- **Estimate:** ~200B/row + token (~200–400B)

DB2 DiscountRedemptions (owned by `AdService`/`BookingService`)

- `id UUID PK`, `session_id UUID FK`, `booking_id UUID FK`, `pct INT`, `applied_at TIMESTAMPTZ`
- **Estimate:** ~120B/row

DB3 Bookings (owned by `BookingService`)

- `id UUID PK`, `rider_id UUID`, `pickup GEOGRAPHY`, `dropoff GEOGRAPHY`, `base_fare MONEY`, `final_fare MONEY`, `status ENUM('Quoted','Booked','Completed','Cancelled')`
- **Estimate:** ~160B/row (+ GEO ~48B)

DB4 Consents (owned by `ConsentService`)

- `id UUID PK`, `rider_id UUID`, `policy_ver TEXT`, `consented_at TIMESTAMPTZ`, `source TEXT`
- **Estimate:** ~100B/row

Rationale:

- **Session-centric model:** `AdvertisementSessions` captures discount eligibility and proof of completion; `DiscountRedemptions` ties the discount to exactly one booking to prevent reuse.
- **Ownership annotated:** Each table's "owner" service reduces ambiguous writes and simplifies access control.
- **Sizing estimates:** Early guardrails for capacity planning (campaign spikes) and GDPR/DSR cost.

12. Security and Privacy

PII (temporary): auth token, geolocation (pickup/dropoff), payment token (non-PAN), device ID. Kept only for session/booking.

- **Why needed:** quoting/booking, fraud prevention, payment authorization.
- **Ingress:** Mobile → `BookingService.quoteFare/createBooking`, Mobile → `AdService.startAdSession`.
- **Flow/Use:** `AdService` uses `riderId` + device to request/verify ad; `BookingService` uses discount to compute `final_fare`; `ConsentService` records `policyVer`.
- **Egress/Disposal:** Ad tokens never logged; discount redemptions retained per tax/audit policy; geolocation redacted after completion (e.g., rounded or deleted per retention policy).
- **Protection:** TLS 1.3, JWT access tokens, short-lived ad session tokens, Redis TTL; access control on DB tables; memory-safety via TS types; audit logs; rate limits.

PII (long-term): booking history minimal set; consent records retained with version.

13. Risks to Completion

- **Ad SDK Compliance (TECH-AD-1):** Medium — integration and server-side verification; SDK updates may break flows. Have fallback (no-ad path), track upgrade criteria, rely on vendor docs/support.
- **Fraud Resistance:** Medium-High — must robustly verify completion; add telemetry and anomaly detection.
- **Policy/Consent UX:** Low-Medium — ensure clear opt-in and revocation; A/B test copy; legal review for minors/regions.
- **Concurrency/TTL Edge Cases:** Medium — expiry during booking; enforce idempotency keys; chaos test.
- **Perf/Overload:** Medium — ad calls add latency; prefetch ads and isolate with circuit breakers.
- **Data Retention & Privacy:** Medium — define deletion windows for geo & ad tokens; document DSRs.
- **Testing:** Medium — scenario/state-based tests mirror #5/#6; contract tests for Ad Network & Payments.

14. GPT log history

https://chatgpt.com/share/68d30441-9140-8007-aa8f-fec765c82b21