Software Requirements Specification

Taxi Service Application

Version 1.0

CST438 Software Design

Alexandra Miller

April 20, 2026

Table of Contents

1.0 Introduction

1.1 Purpose

The purpose of this document is to present a detailed description of the Taxi Service Application. It explains the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate, and how the system will react to external stimuli. This document is intended for both stakeholders and developers of the system and serves as the contractual basis for subsequent design, implementation, and testing activities.

1.2 Scope of Project

The Taxi Service Application is a mobile and server platform that connects customers who need transportation with drivers who provide transportation in their personal vehicles. The system allows a customer to request a ride from a mobile application, pairs the customer with a nearby available driver, tracks the ride through pickup and drop-off, computes and processes the fare, and collects ratings from both parties.

The system provides two mobile applications (one for customers and one for drivers) backed by a central server and relational database. The server handles pairing, fare calculation, payment processing through an external processor, notifications, and monitoring of customer and driver behavior. It integrates with external services for mapping, GPS, and payment but does not itself store credit card numbers.

1.3 Overview of Document

Section 2, Overall Description, gives an overview of the system and its actors through a use case summary and diagram. Section 3, Requirements Specification, provides the external interface requirements, detailed functional requirements as use case descriptions, the logical structure of the data through an entity-relationship diagram and entity tables, and the non-functional requirements that apply to the system as a whole.

2.0 Overall Description

2.1 Use Case Summary

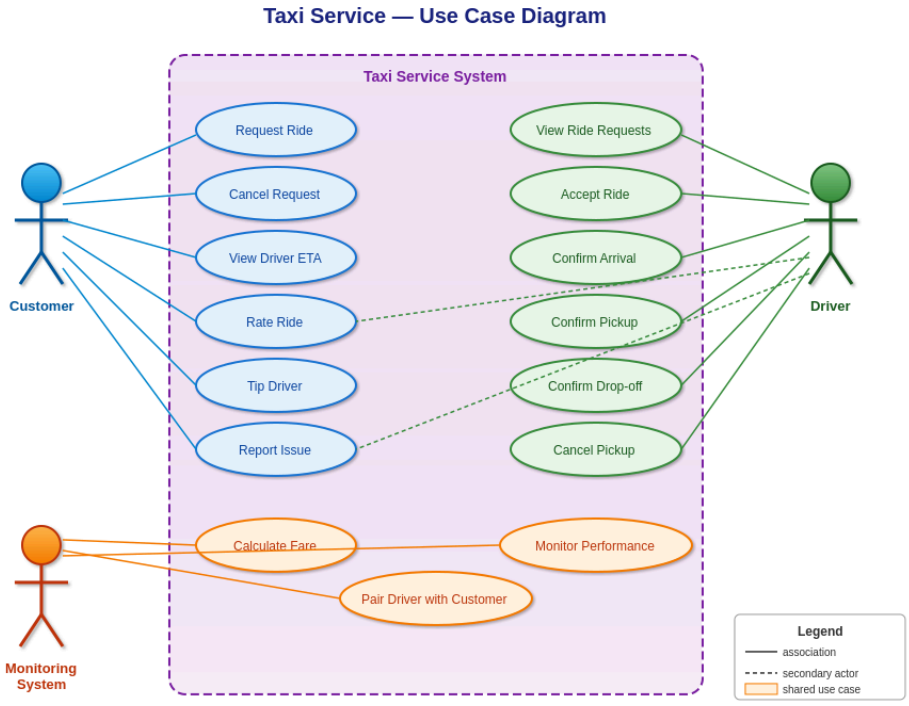

Figure 1 — Use Case Diagram for Taxi Service Application

The Taxi Service Application has two primary live actors -the Customer and the Driver, and one cooperating internal actor, the Monitoring System. The Customer uses the customer mobile application to request rides, view pairing and ETA information, cancel requests, tip, rate rides, and report issues. The Driver uses the driver mobile application to view eligible ride requests in their area, accept rides, confirm arrival and pickup, confirm drop-off, rate rides, and report issues. The Monitoring System is an internal automated actor that runs continuously and evaluates customer and driver behavior against policy thresholds, flagging accounts that require administrative review.

2.2 Functional Requirements Specification

The Ride Lifecycle consists of the use cases listed below. A Customer requests a ride specifying a destination. The system pairs the Customer with an eligible nearby Driver. The Driver accepts the ride and drives to the pickup location. The Driver confirms arrival; if the Customer is not present within 10 minutes, the Driver may cancel the pickup. The Driver confirms pickup and drives to the destination. The Driver confirms drop-off. Both can rate each other. The Monitoring System watches for patterns that require administrative review.

Customer Use Cases: Request Ride, Cancel Ride Request, View Driver ETA, Rate Ride, Tip Driver, Report Issue.

Driver Use Cases: View Available Ride Requests, Accept Ride, Confirm Arrival at Pickup, Cancel Pickup (after 10-minute wait), Confirm Pickup, Confirm Drop-off, Rate Ride, Report Issue.

System (Shared) Use Cases: Calculate Fare, Pair Driver with Customer, Process Payment, Monitor Performance.

2.3 User Characteristics

The Customer is expected to be familiar with smartphone applications and able to use touchscreen controls, location services, and mobile payment methods. No technical expertise beyond basic mobile app usage is assumed. The Driver is expected to be familiar with smartphone applications, GPS navigation, and mobile payment settlement, and is assumed to hold a valid driver's license and to have completed onboarding including vehicle registration and background verification. The Monitoring System requires no human interaction; it runs continuously on the server and generates flags that are reviewed by administrative staff outside the scope of the mobile applications.

3.0 Requirements Specification

3.1 External Interface Requirements

The Taxi Service Application integrates with the following external systems. The GPS / Mapping Service is used by both mobile applications to determine location, geocode addresses, compute routes, and estimate travel times (REST over HTTPS). The External Payment Processor is used by the server to charge customers and issue driver payouts; only opaque tokens are stored locally. The Push Notification Service delivers asynchronous notifications to mobile devices, with an SMS Gateway as fallback. An Identity Verification Service is used only during driver onboarding to validate license information and conduct background checks; it is not used during normal ride operation.

3.2 Functional Requirements

The logical structure of the data is described in Section 3.3. The Xref entry in each use case cross-references the requirements back to the corresponding use case in Section 2.

3.2.1 Request Ride

Use Case Name	Request Ride
Xref	Section 2 – Request Ride
Trigger	The Customer opens the mobile application and selects the option to request a ride.
Precondition	The Customer is logged in with an account in good standing; the mobile device has active GPS and network connectivity; a valid payment method is on file.
Basic Path	1. The system acquires the Customer's current location by GPS and displays it on the map. 2. The Customer enters or selects a destination address. 3. The system invokes Calculate Fare (3.2.11) and presents the estimated fare. 4. The Customer confirms the request. 5. The system creates a ride record with status REQUESTED and invokes Pair Driver with Customer (3.2.12). 6. The system notifies the Customer that a driver search is in progress.
Alternative Paths	In step 2, if the destination is invalid or cannot be geocoded, the system prompts the Customer to re-enter it. In step 3, if the Customer does not accept the fare, the Customer may cancel before confirming and no ride is created.
Postcondition	A ride request is stored with status REQUESTED and is visible to eligible drivers within the search radius.
Exception Paths	If GPS cannot be acquired within 15 seconds, the Customer may enter the pickup address manually or abandon the request.
Other	The fare estimate is binding unless traffic conditions materially change the route.

3.2.2 Cancel Ride Request

Use Case Name	Cancel Ride Request
Xref	Section 2 – Cancel Request
Trigger	The Customer selects the cancel option on an active ride request.
Precondition	The ride is in status REQUESTED, ACCEPTED, or DRIVER_ARRIVED; it has not reached PICKED_UP.
Basic Path	1. The system displays a confirmation dialog. 2. The Customer confirms. 3. The system updates ride status to CANCELLED_BY_CUSTOMER and records the timestamp. 4. If a driver is already assigned, the system notifies the driver. 5. The system increments the Customer's cancel_count.

Alternative Paths	In step 2, if the Customer does not confirm, the cancellation is aborted.
Postcondition	Ride is cancelled; assigned driver has been notified and may receive new requests.
Exception Paths	If the pickup has already occurred, cancellation is blocked and the Customer is directed to support.
Other	A cancellation fee may apply per NFR-BIZ-02.

3.2.3 View Available Ride Requests

Use Case Name	View Available Ride Requests
Xref	Section 2 – View Ride Requests
Trigger	The Driver opens the driver application and sets availability to AVAILABLE.
Precondition	The Driver is logged in, not on an active ride, and has GPS and network connectivity.
Basic Path	1. The system obtains the Driver's GPS location. 2. The system queries rides in status REQUESTED whose pickup is within the configurable radius (default 5 miles). 3. Pairs excluded by Pair Driver with Customer (3.2.12) are filtered out. 4. The system displays each request with pickup, destination, distance, estimated fare, and Customer rating. 5. The list refreshes every 10 seconds.
Alternative Paths	If no eligible requests exist, the system shows a message and continues refreshing.
Postcondition	The Driver has seen the current list of eligible requests.
Exception Paths	If the network is lost, the system warns the Driver and pauses refreshing.
Other	Customer identity is not revealed before acceptance; only first name and rating are shown.

3.2.4 Accept Ride

Use Case Name	Accept Ride
Xref	Section 2 – Accept Ride
Trigger	The Driver selects a request from the available list.
Precondition	The ride is still in status REQUESTED.
Basic Path	1. The system verifies the ride is still REQUESTED. 2. The system updates the ride: status=ACCEPTED, driver_id set, estimated pickup time calculated. 3. All other requests are hidden from the Driver. 4. The Customer is notified with driver name, vehicle info, license plate, and ETA. 5. Turn-by-turn navigation to pickup is displayed to the Driver.
Alternative Paths	If the ride was already taken or cancelled, the Driver is returned to the requests list.
Postcondition	The ride is assigned to the Driver; no other requests are shown until completion or cancellation.
Exception Paths	If the database update fails, the ride remains REQUESTED.
Other	Once accepted, the Driver cannot see other requests until the ride completes or is cancelled.

3.2.5 Confirm Arrival at Pickup

Use Case Name	Confirm Arrival at Pickup
Xref	Section 2 – Confirm Arrival
Trigger	The Driver selects "I have arrived" on the driver application.
Precondition	Ride is in status ACCEPTED; Driver GPS is within 300 ft of pickup.
Basic Path	1. The system verifies the Driver is near the pickup location. 2. Status is updated to DRIVER_ARRIVED and arrival timestamp is recorded. 3. The Customer is notified that the driver has arrived. 4. A 10-minute wait timer starts. 5. The Cancel Pickup option (3.2.6) becomes available after 10 minutes if the Customer has not appeared.
Alternative Paths	If the Driver is not near the pickup, the system warns and requires explicit confirmation.
Postcondition	Status is DRIVER_ARRIVED and arrival time is recorded.
Exception Paths	If GPS is unavailable, manual confirmation is permitted but the ride is flagged for monitoring.
Other	The 10-minute threshold is configurable by administration.

3.2.6 Cancel Pickup (Driver)

Use Case Name	Cancel Pickup
Xref	Section 2 – Cancel Pickup
Trigger	The Driver selects Cancel Pickup after the 10-minute wait.
Precondition	Ride is in DRIVER_ARRIVED and 10 minutes have elapsed since arrival.
Basic Path	1. The system verifies the 10-minute threshold. 2. A confirmation dialog is shown. 3. The Driver confirms. 4. Status is updated to CANCELLED_BY_DRIVER_NO_SHOW; Customer is notified; cancel_count is incremented. 5. The event is logged via Monitor Performance (3.2.13). 6. The Driver becomes available for new requests.
Alternative Paths	Before 10 minutes, the option is disabled and the use case does not proceed.
Postcondition	Ride is cancelled with no-show reason; Driver is available.
Exception Paths	The Driver may abandon the dialog and continue waiting.
Other	A no-show fee may apply per NFR-BIZ-02.

3.2.7 Confirm Pickup

Use Case Name	Confirm Pickup
Xref	Section 2 – Confirm Pickup
Trigger	The Driver selects "Customer picked up".
Precondition	Ride is in DRIVER_ARRIVED and the Customer is in the vehicle.
Basic Path	1. Status is updated to PICKED_UP and pickup timestamp is recorded. 2. The fare meter starts and route tracking begins. 3. Turn-by-turn navigation to dropoff is shown.

	4. The Customer app shows "ride in progress".
Alternative Paths	None.
Postcondition	Ride is PICKED_UP; Customer may no longer cancel.
Exception Paths	If Driver location is not at pickup, the system prompts for confirmation.
Other	None.

3.2.8 Confirm Drop-off

Use Case Name	Confirm Drop-off
Xref	Section 2 – Confirm Drop-off
Trigger	The Driver selects "Customer dropped off".
Precondition	Ride is PICKED_UP and the vehicle is at the dropoff location.
Basic Path	1. Status is updated to COMPLETED and dropoff timestamp is recorded. 2. Calculate Fare (3.2.11) computes the final fare using actual distance and time. 3. Process Payment (3.2.14) charges the Customer. 4. Rate Ride (3.2.9) is offered to the Driver. 5. A ride summary and rating prompt are sent to the Customer. 6. The Driver becomes available for new requests.
Alternative Paths	If GPS does not match the recorded dropoff, the ride is flagged for monitoring but still completed.
Postcondition	Ride is COMPLETED; payment processed; ratings solicited.
Exception Paths	If payment fails, a payment-issue flag is raised; Customer is notified.
Other	None.

3.2.9 Rate Ride

Use Case Name	Rate Ride
Xref	Section 2 – Rate Ride
Trigger	Customer or Driver opens the rating screen after ride completion.
Precondition	Ride is COMPLETED; the rater has not yet submitted a rating.
Basic Path	1. A 1-to-5 star rating screen with optional comment is shown. 2. The rater selects a star rating. 3. Ratings of 3 or below mark the ride is_satisfactory = false for that rater. 4. The rater optionally enters a comment and submits. 5. The Rating record is stored; counterparty's avg_rating is updated. 6. Unsatisfactory ratings block this pair in Pair Driver with Customer (3.2.12).
Alternative Paths	A comment is optional; submission proceeds without it.
Postcondition	Rating is stored; pairing blacklist reflects the outcome if unsatisfactory.
Exception Paths	The rater may dismiss the screen and rate later from ride history.
Other	The rating window closes 7 days after ride completion.

3.2.10 Tip Driver

Use Case Name	Tip Driver
Xref	Section 2 – Tip Driver
Trigger	The Customer selects the tip option on a completed ride.
Precondition	Ride is COMPLETED within the last 24 hours; no tip has been applied.
Basic Path	1. Preset tip amounts (15%, 20%, 25%) and a custom option are shown. 2. The Customer selects an amount and confirms. 3. The system charges the tip to the Customer's payment method. 4. The ride tip_amount is updated and the Driver is notified.
Alternative Paths	The Customer may choose "No tip"; the use case ends with no charge.
Postcondition	Tip is recorded on the ride and Driver is notified.
Exception Paths	If payment fails, the tip is not applied and the Customer is prompted to update payment.
Other	Tips are credited per NFR-BIZ-01 settlement schedule.

3.2.11 Calculate Fare

Use Case Name	Calculate Fare
Xref	Section 2 – Calculate Fare
Trigger	Invoked by Request Ride (3.2.1) for estimates and Confirm Drop-off (3.2.8) for final fare.
Precondition	Pickup and dropoff are known; time of day is known; distance is computable.
Basic Path	1. The system loads the regional base fare, per-mile rate, and per-minute rate. 2. A time-of-day multiplier is applied: 1.0 off-peak, 1.5 peak (7-10, 16-19 local), 2.0 late-night (23-5). 3. Fare = (base + distance × per_mile + duration × per_minute) × multiplier. 4. The fare is returned to the caller.
Alternative Paths	A surge multiplier may be applied during surge windows.
Postcondition	A fare amount is returned.
Exception Paths	If regional config cannot load, a default config is used and the ride is flagged.
Other	Final fare may differ from the estimate due to actual distance and duration.

3.2.12 Pair Driver with Customer

Use Case Name	Pair Driver with Customer
Xref	Section 2 – Pair Driver with Customer
Trigger	Invoked by Request Ride (3.2.1) after the ride is created.
Precondition	A ride exists in status REQUESTED and at least one Driver is available.
Basic Path	1. The system queries prior rides between this Customer and any Driver with is_satisfactory = false. 2. An exclusion list of Driver IDs is built from the query. 3. The ride is made visible to Drivers in the search radius except those excluded. 4. If no acceptance occurs within 60 seconds, the radius is expanded by 2 miles, up to 15 miles. 5. When a Driver accepts (3.2.4), pairing is complete.

Alternative Paths	If no acceptance occurs at the maximum radius within 5 minutes, the Customer is notified and may continue waiting or cancel.
Postcondition	A Driver has accepted, or the Customer has been notified that none is available.
Exception Paths	If all nearby Drivers are excluded, the Customer is notified and offered to keep waiting.
Other	The system does not disclose exclusion reasons to either party.

3.2.13 Monitor Performance

Use Case Name	Monitor Performance
Xref	Section 2 – Monitor Performance
Trigger	Runs continuously as a background system process.
Precondition	The system is operational and has access to ride and rating history.
Basic Path	1. Flag Customers with cancel_count > 3 in any rolling 30-day window. 2. Flag Drivers whose actual arrival exceeds estimated arrival by more than 5 minutes on > 20% of recent rides. 3. Flag Drivers whose actual dropoff exceeds estimated dropoff by more than 10 minutes on > 20% of recent rides. 4. Flag accounts with more than 5 unsatisfactory ratings in 30 days. 5. Flagged accounts create Incident records for administrative review.
Alternative Paths	Additional custom rules may be configured by administration.
Postcondition	Problematic accounts have been flagged and Incident records generated.
Exception Paths	If the job fails, an admin alert is raised and the job retries on the next interval.
Other	The job runs every 6 hours.

3.2.14 Process Payment

Use Case Name	Process Payment
Xref	Section 2 – Calculate Fare (post-ride settlement)
Trigger	Invoked by Confirm Drop-off (3.2.8) after final fare is calculated.
Precondition	Ride is COMPLETED; fare is calculated; Customer has a valid payment method.
Basic Path	1. The fare is submitted to the external payment processor via the stored token. 2. On success, a transaction reference is returned. 3. A Payment record is created with status SUCCESS. 4. A receipt is sent to the Customer.
Alternative Paths	On failure, a Payment record is created with status FAILED; the Customer is notified and retry is queued.
Postcondition	A Payment record exists with final status SUCCESS, FAILED, or PENDING_RETRY.
Exception Paths	If the processor is unreachable, the payment is queued as PENDING.
Other	No credit card numbers are stored locally; the processor is PCI-compliant.

3.2.15 Report Issue

Use Case Name	Report Issue

Xref	Section 2 – Report Issue
Trigger	Customer or Driver selects "Report an issue" from a ride record.
Precondition	The reporter has access to the ride; the ride is in progress or completed within 30 days.
Basic Path	1. The system presents categories (safety, cleanliness, route, fare dispute, behavior) and a description field. 2. The reporter selects a category and enters a description. 3. The reporter submits the form. 4. An Incident record is created and linked to the ride. 5. An acknowledgement is sent and the incident is routed to the appropriate queue.
Alternative Paths	Safety-related incidents are routed to a priority queue and acknowledged within 15 minutes.
Postcondition	An Incident record exists and the reporter has been acknowledged.
Exception Paths	The reporter may abandon the form before submission.
Other	Incident records feed into Monitor Performance (3.2.13).

3.3 Logical Structure of the Data

The logical structure of the data stored in the Taxi Service System database is shown in Figure 2 below.

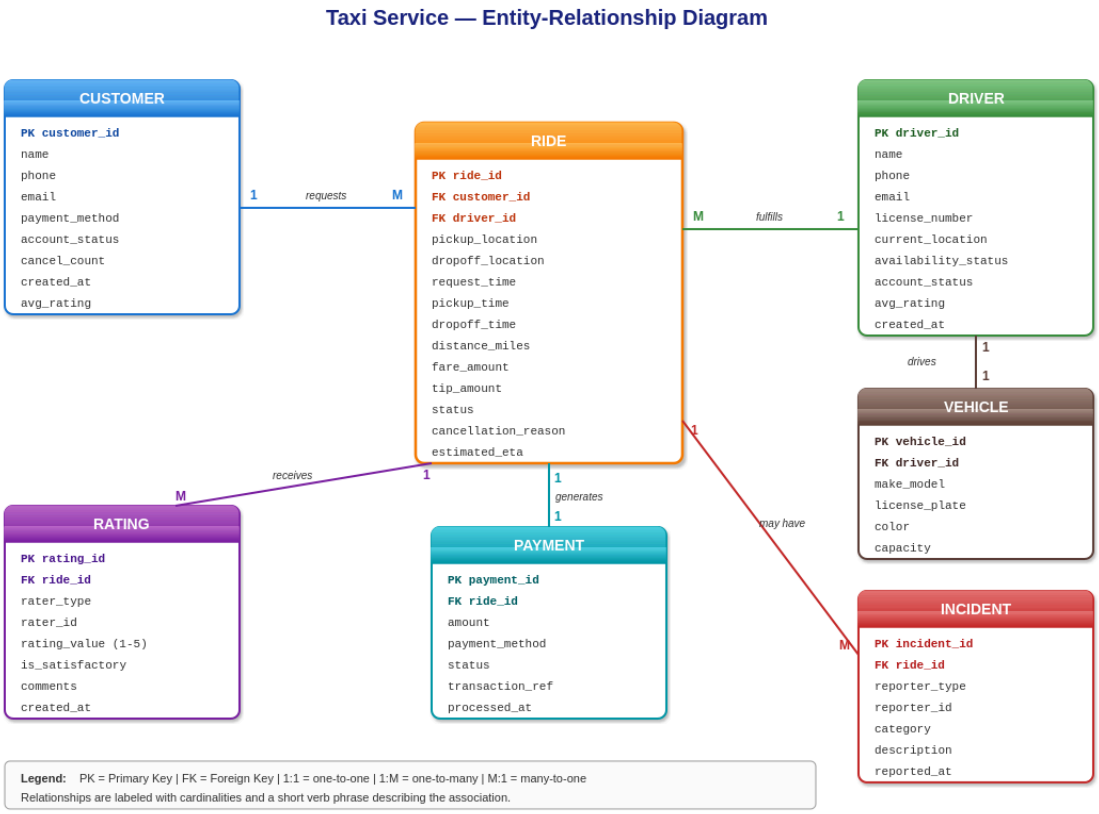

Figure 2 — Entity-Relationship Diagram

The database contains seven principal entities: CUSTOMER, DRIVER, VEHICLE, RIDE, RATING, PAYMENT, and INCIDENT. Primary keys are shown as "PK" in the diagram and entity tables; foreign keys are shown as "FK".

Customer Data Entity

Represents a registered rider. The customer_id is the primary key used throughout the system to link rides, ratings, payments, and incidents to the requesting rider.

Data Item	Type	Description	Comment
customer_id	Integer	Unique identifier	Primary Key, auto-generated
name	Text	Full name of customer	Required
phone	Text	Mobile phone number	Required, used for notifications
email	Text	Email address	Required, unique
payment_method	Text	Token referencing the external payment processor	No card numbers stored locally
account_status	Enum	ACTIVE, SUSPENDED, DEACTIVATED	Default ACTIVE
cancel_count	Integer	Cancellations in rolling 30 days	Used by monitoring
avg_rating	Decimal	Average rating received from drivers	1.0–5.0
created_at	Timestamp	Account creation time	Immutable

Driver Data Entity

Represents a registered driver. The driver_id is the primary key referenced by RIDE (as the fulfilling driver) and by VEHICLE (as the owning driver).

Data Item	Type	Description	Comment
driver_id	Integer	Unique identifier	Primary Key, auto-generated
name	Text	Full name of driver	Required
phone	Text	Mobile phone number	Required
email	Text	Email address	Required, unique
license_number	Text	Driver license number	Validated at onboarding
current_location	Geo	Latitude and longitude	Updated continuously when online
availability_status	Enum	AVAILABLE, ON_RIDE, OFFLINE	Default OFFLINE
account_status	Enum	ACTIVE, SUSPENDED, DEACTIVATED	Default ACTIVE
avg_rating	Decimal	Average rating received from customers	1.0–5.0
created_at	Timestamp	Account creation time	Immutable

Vehicle Data Entity

Represents the vehicle a Driver currently uses. Each Driver has exactly one active Vehicle; driver_id is the foreign key into DRIVER.

Data Item	Type	Description	Comment
vehicle_id	Integer	Unique identifier	Primary Key, auto-generated
driver_id	Integer	Owning driver	Foreign Key → DRIVER
make_model	Text	Make and model	e.g., "Toyota Camry"
license_plate	Text	Registration plate	Required, unique
color	Text	Vehicle color	Displayed to Customer on pairing
capacity	Integer	Passenger seats	Typically 1–6

Ride Data Entity

Represents a single ride request and its lifecycle from request through completion or cancellation. The status attribute captures the current state; timestamps for each state transition are stored in the corresponding fields.

Data Item	Type	Description	Comment
ride_id	Integer	Unique identifier	Primary Key, auto-generated
customer_id	Integer	Requesting customer	Foreign Key → CUSTOMER
driver_id	Integer	Assigned driver	Foreign Key → DRIVER, nullable until pairing
pickup_location	Geo	Pickup latitude/longitude	Required
dropoff_location	Geo	Dropoff latitude/longitude	Required
request_time	Timestamp	When the ride was requested	Required
pickup_time	Timestamp	When pickup was confirmed	Null until pickup
dropoff_time	Timestamp	When drop-off was confirmed	Null until drop-off
distance_miles	Decimal	Actual miles travelled	Populated at drop-off
fare_amount	Decimal	Final fare in local currency	Populated at drop-off
tip_amount	Decimal	Tip applied by customer	Null until tip applied
status	Enum	REQUESTED, ACCEPTED, DRIVER_ARRIVED, PICKED_UP, COMPLETED, CANCELLED_BY_CUSTOMER, CANCELLED_BY_DRIVER_NO_SHOW	Required
cancellation_reason	Text	Reason for cancellation if applicable	Null if not cancelled
estimated_eta	Integer	Estimated minutes to pickup	Set when driver accepts

Rating Data Entity

Represents a rating submitted by a Customer or Driver after a completed ride. The is_satisfactory flag is derived from rating_value and used by Pair Driver with Customer (3.2.12) to prevent re-pairing of parties that had an unsatisfactory experience.

Data Item	Type	Description	Comment
rating_id	Integer	Unique identifier	Primary Key, auto-generated
ride_id	Integer	Associated ride	Foreign Key → RIDE
rater_type	Enum	CUSTOMER or DRIVER	Who submitted the rating
rater_id	Integer	ID of rater in respective table	Interpreted by rater_type
rating_value	Integer	Star rating 1–5	Required
is_satisfactory	Boolean	True if rating_value > 3	Derived, used for pairing logic
comments	Text	Optional free-text comment	Nullable
created_at	Timestamp	When submitted	Required

Payment Data Entity

Represents a payment transaction associated with a Ride. The Taxi Service System stores only an opaque token reference; all actual card data is held by the external payment processor.

Data Item	Type	Description	Comment
payment_id	Integer	Unique identifier	Primary Key, auto-generated
ride_id	Integer	Associated ride	Foreign Key → RIDE
amount	Decimal	Amount charged	Includes fare and tip when settled
payment_method	Text	Token used for the charge	No card numbers stored locally
status	Enum	PENDING, SUCCESS, FAILED, PENDING_RETRY	Required
transaction_ref	Text	Reference from external processor	Null if failed
processed_at	Timestamp	When processed	Null if pending

Incident Data Entity

Represents an issue reported by a Customer, a Driver, or the Monitoring System. Incidents feed into administrative review and aggregate pattern detection.

Data Item	Type	Description	Comment
incident_id	Integer	Unique identifier	Primary Key, auto-generated
ride_id	Integer	Associated ride	Foreign Key → RIDE, nullable for system-flagged events
reporter_type	Enum	CUSTOMER, DRIVER, or SYSTEM	Source of the report
reporter_id	Integer	ID of reporter	Null if SYSTEM
category	Text	SAFETY, CLEANLINESS, FARE_DISPUTE, NO_SHOW, BEHAVIOR	Required
description	Text	Free-text description	Required for user reports
reported_at	Timestamp	When reported	Required

3.4 Non-Functional Requirements

The following non-functional requirements apply to the system as a whole rather than to any single use case. They cover performance, security, availability, reliability, usability, maintainability, portability, and business rules, and are written to be specific and measurable.

3.4.1 Performance

NFR-PERF-01: The customer and driver applications shall respond to user interactions within 1.5 seconds under normal load and within 3 seconds during peak load. NFR-PERF-02: The pairing engine shall match a ride request to an eligible driver, when at least one is available within the maximum radius, within 10 seconds in 95% of cases. NFR-PERF-03: The driver application shall refresh the available-requests list at least every 10 seconds while the Driver is available. NFR-PERF-04: Map and location data shall update on both applications at least once every 5 seconds during an active ride. NFR-PERF-05: The system shall support at least 10,000 concurrent active rides without degradation of the above response times.

3.4.2 Security

NFR-SEC-01: All data in transit between mobile applications and the server shall be encrypted using TLS 1.3 or higher. NFR-SEC-02: All personally identifiable information at rest shall be encrypted using AES-256. NFR-SEC-03: The system shall not store credit card numbers or CVV codes; payment handling shall use a PCI-DSS compliant external processor referenced only by opaque token. NFR-SEC-04: Authentication shall require a verified phone number and either a password of at least 12 characters with mixed case, numeric, and symbol, or a federated identity provider. NFR-SEC-05: Session tokens shall expire after 30 days of inactivity and be revocable from account settings. NFR-SEC-06: Authentication, payment, and administrative events shall be logged for at least 12 months. NFR-SEC-07: The system shall comply with applicable data protection regulations including GDPR and CCPA.

3.4.3 Availability and Reliability

NFR-AVAIL-01: The server component shall maintain a minimum uptime of 99.9% over any rolling 30-day window, excluding scheduled maintenance. NFR-AVAIL-02: Scheduled maintenance shall be limited to no more than 2 hours per month and announced at least 48 hours in advance. NFR-AVAIL-03: The system shall tolerate the failure of any single server instance without loss of active rides. NFR-AVAIL-04: Applications shall cache active ride state locally so that brief network outages up to 60 seconds do not disrupt an ongoing ride. NFR-REL-01: Completed ride, rating, payment, and incident records shall be durable. NFR-REL-02: Payment retries shall occur automatically on transient failure up to 3 attempts over 24 hours. NFR-REL-03: The system shall recover from orderly shutdown and restart within 10 minutes with no loss of committed state.

3.4.4 Usability, Maintainability, and Portability

NFR-USAB-01: A new Customer shall be able to complete their first ride request within 5 minutes of first launching the application, assuming valid payment information is available. NFR-USAB-02: Both applications shall support host OS accessibility features, including screen readers and large text. NFR-USAB-03: All user-facing text shall be localised for the regions in which the service operates. NFR-MAINT-01: The server component shall be deployable to any Linux-based hosting environment supporting container orchestration. NFR-MAINT-02: Both applications shall support iOS 15+ and Android 10+. NFR-MAINT-03: Configuration parameters (search radius, 10-minute pickup wait, fare rates, surge thresholds) shall be externally configurable without code changes. NFR-MAINT-04: The system shall expose health-check endpoints and emit structured logs compatible with standard aggregation tools.

3.4.5 Business Rules

NFR-BIZ-01: Drivers shall be paid accumulated fares and tips on a weekly schedule through the external payment processor. NFR-BIZ-02: Cancellation and no-show fees shall be configurable by region and shall not exceed the base fare of the cancelled ride. NFR-BIZ-03: Rides rated 3 stars or below shall be treated as unsatisfactory for future-pairing purposes. NFR-BIZ-04: A Customer whose cancel_count exceeds 3 in any 30-day rolling window shall be flagged for administrative review.