Software Requirements Specification

Version 0.1

08/04/2026

A Taxi Service Similar to Uber

Michael Sorensen

Table of Contents

1.0. Introduction

1.1. Purpose

The purpose of this document is to present a detailed description of the Rideshare System, a taxi-hailing service similar to Uber. 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 respond to external stimuli. This document is intended for both the stakeholders and the developers of the system and will be reviewed and approved by the startup's project stakeholders before design and implementation begin.

1.2. Scope of Project

This software system will be a mobile-based ride-hailing platform that connects Customers who need a ride with Drivers who can provide one. The system is designed to let a Customer request a ride from their current location to a desired destination, see an upfront price estimate, and track the assigned Driver until pickup. The system is designed to let a Driver view nearby ride requests, accept the ones they choose, and be guided through the pickup and drop-off process.

More specifically, the system manages the full lifecycle of a ride: request, price estimation, driver matching, pickup, drop-off, payment, and rating. The mobile applications communicate with a central Rideshare System that maintains records of Customers, Drivers, and Rides in a relational database. The system also integrates with an external Payment Gateway to process fares and tips, and an external GPS/Mapping Service to determine locations and estimate travel times. The algorithm used to estimate travel time between two locations from historical travel data is outside the scope of this document.

1.3. Overview of Document

The next chapter, the Overall Description section, of this document gives an overview of the functionality of the product. It describes the informal requirements and use case diagram, and is used to establish a context for the technical requirements specification presented in Section 3.0.

2.0. Overall Description

2.1 Use case summary

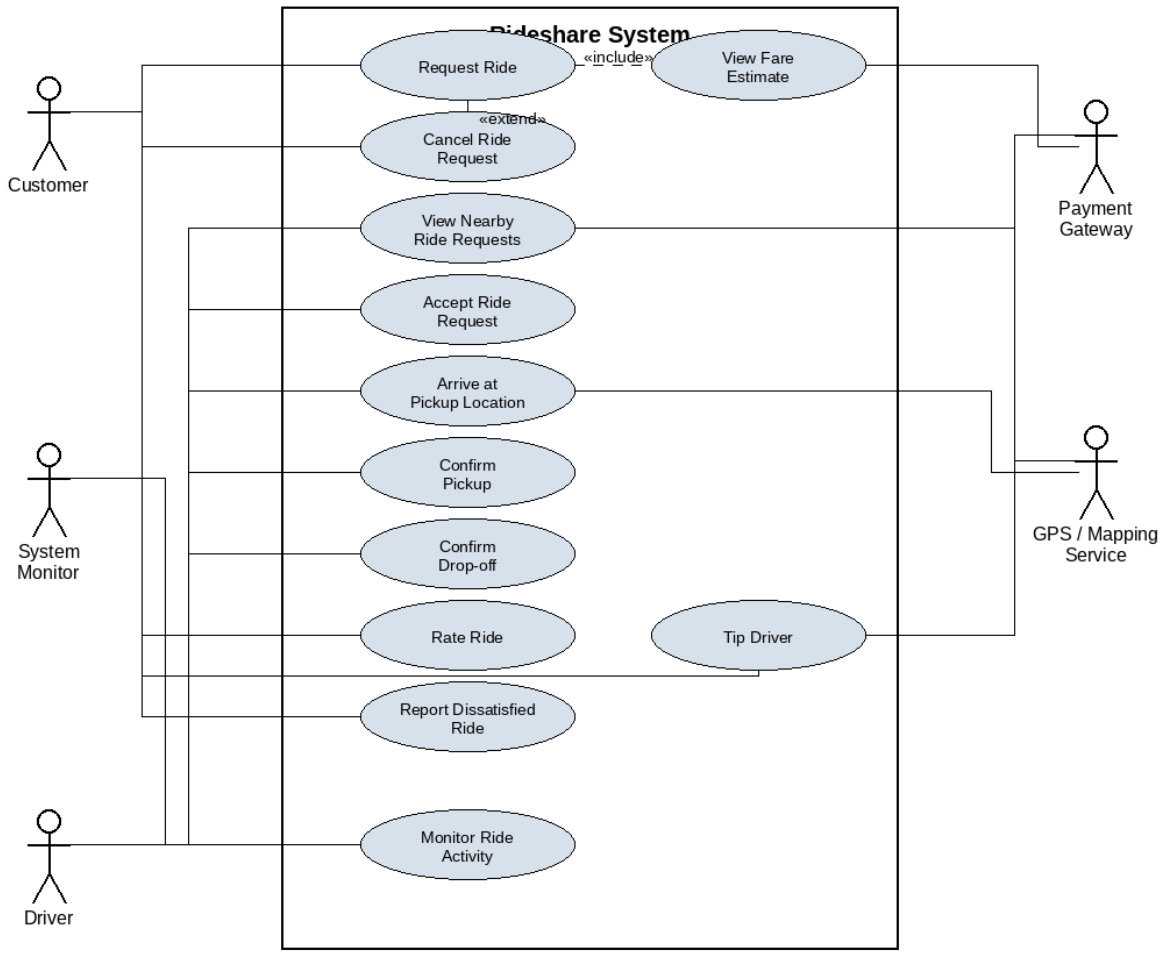

Figure 1 – Use Case Diagram for the Rideshare System

The Rideshare System has two primary actors and two external cooperating systems. The Customer and the Driver each access the system through a mobile application. The Customer requests rides, tracks the Driver's arrival, and rates and tips the Driver after the trip. The Driver views and accepts nearby ride requests, updates the ride status at each step, and rates the Customer after the trip. The system also communicates with an external Payment Gateway to process fares and tips, and with an external GPS/Mapping Service to determine Customer and Driver locations and to calculate distances and estimated arrival times.

2.2 Functional Requirements Specification

This section outlines the use cases for the system's two primary actors, the Customer and the Driver. Some use cases, such as Rate Ride and Report Dissatisfied Ride, are shared by both actors. In addition, the system performs an internal Monitor Ride Activity process (use case 3.2.13) that runs continuously in the background rather than being triggered directly by an actor.

The Ride Lifecycle Process consists of the use cases listed below.

- A Customer opens the app, and the app captures the Customer's current location using GPS.
- The Customer specifies a destination and requests a ride; the system calculates and displays an estimated fare based on distance and time of day.
- Nearby Drivers see the request, including pickup location, destination, and price, and a Driver accepts it.
- The Customer is notified that a Driver is on the way, along with the Driver's information and an estimated time of arrival.
- When the Driver arrives, the Driver marks arrival in the app; if the Customer does not appear within 10 minutes, the Driver may cancel the pickup.
- Once the Customer is in the vehicle, the Driver confirms pickup, drives to the destination, and confirms drop-off.
- After the ride, the Customer and Driver rate each other, the Customer may leave a tip, and either party may report a dissatisfactory ride.

2.2.1 Use Case Groupings

Ride Request consists of use cases

- Request Ride
- Cancel Ride Request
- View Fare Estimate
- Match Driver and Customer

Ride Fulfillment consists of

- View Nearby Ride Requests
- Accept Ride Request
- Arrive at Pickup Location, Confirm Pickup, and Confirm Drop-off

Post-Ride Activity consists of use cases:

- Rate Ride and Tip Driver
- Report Dissatisfied Ride

2.3 User Characteristics

The Customer is expected to be comfortable using a smartphone app, including entering an address, reading a price estimate, and using in-app messaging or calling to reach the Driver if needed. No special technical knowledge is required.

The Driver is expected to be comfortable using a smartphone app while operating a vehicle, including glancing at and accepting ride requests, following in-app turn-by-turn navigation, and updating ride status at each step (arrival, pickup, drop-off). The Driver must hold a valid driver's license, which is verified during account setup.

A Support Representative is expected to be able to review flagged accounts and MonitoredEvent records generated by the Monitor Ride Activity use case (3.2.13) and to take action such as contacting a Customer or Driver, issuing a refund, or suspending an account.

3.0. Requirements Specification

3.1 External Interface Requirements

The Rideshare System links to two external systems: a Payment Gateway and a GPS/Mapping Service. The Payment Gateway processes fare and tip charges to the Customer's stored payment method and disburses payments to the Driver; the system sends it the ride amount, the tip amount, and the Customer's payment token, and receives back a transaction status (approved or declined). The GPS/Mapping Service supplies the Customer's and Driver's current coordinates, calculates the driving distance and estimated travel time between two coordinates, and is used both to compute the fare estimate and to compute the Driver's estimated time of arrival.

The Request Ride use case sends the Customer's pickup coordinates and destination coordinates to the GPS/Mapping Service and receives back an estimated distance, duration, and route. The Tip Driver use case and the fare-charging step of Confirm Drop-off send a payment amount and payment token to the Payment Gateway and receive back a Boolean approval status.

3.2 Functional Requirements

The logical structure of the data is described in Section 3.3. The Xref entry cross-references each requirement to the corresponding narrative step in Section 2.2.

3.2.1 Request Ride

Use Case Name	Request Ride
Xref	Section 2.2 steps 1-2
Trigger	The Customer opens the mobile app and selects to request a ride.
Precondition	The Customer is logged in and location services are enabled on the Customer's device.
Basic Path	1. The app uses the GPS/Mapping Service to determine the Customer's current location and displays it as the pickup point. 2. The Customer enters or confirms the destination. 3. The system calculates the estimated distance and travel time using the GPS/Mapping Service and computes an estimated fare based on distance and current time of day (see 3.2.3, View Fare Estimate). 4. The system displays the estimated fare to the Customer. 5. The Customer confirms the request. 6. The system adds the request to the pool of open requests visible to nearby Drivers (see 3.2.4, View Nearby Ride Requests).
Alternative Paths	If GPS location cannot be determined, the Customer is prompted to enter a pickup address manually.
Postcondition	An open ride request exists with status “Requested” and is visible to eligible nearby Drivers.
Exception Paths	The Customer may abandon the request before confirming it. If no Driver accepts the request within a configurable timeout, the Customer is notified that no Drivers are currently available.
Other	A Driver is not considered eligible if a BlockedPair record exists between that Driver and the Customer (see 3.2.12, Match Driver and Customer).

3.2.2 Cancel Ride Request

Use Case Name	Cancel Ride Request
Xref	Section 2.2 step 2
Trigger	The Customer selects to cancel a ride request or an accepted ride prior to pickup.
Precondition	The Customer has an open or accepted ride request that has not yet reached pickup.
Basic Path	The Customer selects Cancel from the ride status screen. The system asks the Customer to confirm the cancellation. The Customer confirms. The system updates the ride status to “Cancelled by Customer” and records the cancellation in the MonitoredEvent log (see 3.2.13, Monitor Ride Activity).
Alternative Paths	If a Driver has already been assigned, the system notifies the Driver that the request was cancelled and returns the Driver to the pool of available Drivers.
Postcondition	The ride is marked cancelled and is no longer visible to Drivers.
Exception Paths	If the ride has already reached the pickup stage (the Driver has confirmed pickup), the Customer can no longer cancel through this use case.
Other	None.

3.2.3 View Fare Estimate

Use Case Name	View Fare Estimate
Xref	Section 2.2 step 2
Trigger	Invoked automatically during Request Ride.
Precondition	A pickup location and destination have been provided.
Basic Path	1. The system retrieves the distance and estimated duration between pickup and destination from the GPS/Mapping Service. 2. The system applies the current base rate, per-mile rate, per-minute rate, and a time-of-day multiplier to compute the estimated fare. 3. The estimated fare is returned to the Request Ride use case for display to the Customer.
Alternative Paths	If a surge or discount multiplier is active for the current time and area, it is applied to the calculation and noted to the Customer.
Postcondition	An estimated fare amount has been calculated and displayed.
Exception Paths	If the GPS/Mapping Service is unavailable, a default flat estimate is shown and flagged as approximate.

Other	The actual fare charged after drop-off may differ slightly from the estimate if the actual route or duration differs from the estimate.

3.2.4 View Nearby Ride Requests

Use Case Name	View Nearby Ride Requests
Xref	Section 2.2 step 3
Trigger	The Driver opens the app and is marked available (online).
Precondition	The Driver is online and not currently on an active ride.
Basic Path	1. The system uses the GPS/Mapping Service to determine the Driver's current location. 2. The system creates and presents a list of open ride requests within a configurable radius of the Driver, showing each Customer's pickup location, destination, and estimated price. 3. The list excludes any request from a Customer with whom the Driver has a BlockedPair record (see 3.2.12). 4. The list refreshes as new requests appear or existing requests are cancelled or accepted by other Drivers.
Alternative Paths	If no requests are within radius, the system displays a message indicating no requests are currently available.
Postcondition	The Driver is shown a current list of eligible nearby requests.
Exception Paths	The Driver may go offline at any time, which stops the list from being shown.
Other	None.

3.2.5 Accept Ride Request

Use Case Name	Accept Ride Request
Xref	Section 2.2 step 3
Trigger	The Driver selects one of the displayed ride requests.
Precondition	The Driver is viewing the list of nearby ride requests (3.2.4) and is not already assigned to another ride.
Basic Path	1. The Driver selects a request to accept. 2. The system assigns the Driver to the ride and updates the ride status to “Accepted.” 3. The system removes the request from every other Driver's nearby-requests list. 4. The system notifies the Customer that a Driver has been assigned, along with the Driver's name, vehicle information, and estimated time of arrival.
Alternative Paths	If two Drivers attempt to accept the same request at nearly the same time, only the first acceptance received by the system is honored; the second Driver is notified the request is no longer available.
Postcondition	The ride has exactly one assigned Driver and its status is “Accepted.”

Exception Paths	The Driver may not accept a request that has already been accepted by another Driver or cancelled by the Customer.
Other	None.

3.2.6 Arrive at Pickup Location

Use Case Name	Arrive at Pickup Location
Xref	Section 2.2 step 5
Trigger	The Driver reaches the Customer's pickup location.
Precondition	The ride status is “Accepted” and the Driver is en route.
Basic Path	1. The Driver selects “I have arrived” in the app. 2. The system updates the ride status to “Driver Arrived” and notifies the Customer. 3. The system starts a 10-minute wait timer.
Alternative Paths	If the Customer does not appear within 10 minutes of the Driver marking arrival, the Driver may select “Cancel - Customer No-Show,” which updates the ride status to “Cancelled - No Show,” records the event in the MonitoredEvent log, and returns the Driver to available status.
Postcondition	Either the Customer is picked up (see 3.2.7, Confirm Pickup) or the ride is cancelled due to no-show.
Exception Paths	The Customer may still cancel the ride during the wait period (see 3.2.2).
Other	If the Driver's actual arrival time exceeds the estimate shown to the Customer by a configurable margin, the event is recorded for monitoring (see 3.2.13).

3.2.7 Confirm Pickup

Use Case Name	Confirm Pickup
Xref	Section 2.2 step 6
Trigger	The Customer enters the vehicle.
Precondition	The ride status is “Driver Arrived.”
Basic Path	1. The Driver selects “Customer Picked Up” in the app. 2. The system updates the ride status to “In Progress” and records the pickup time. 3. The system begins guiding the Driver toward the destination using the GPS/Mapping Service.
Alternative Paths	None.
Postcondition	The ride status is “In Progress.”
Exception Paths	None.
Other	None.

3.2.8 Confirm Drop-off

Use Case Name	Confirm Drop-off
Xref	Section 2.2 step 6
Trigger	The Driver reaches the destination.
Precondition	The ride status is “In Progress.”
Basic Path	1. The Driver selects “Trip Complete” in the app. 2. The system records the drop-off time and updates the ride status to “Completed.” 3. The system calculates the actual fare based on the actual distance and duration traveled and charges the Customer's stored payment method through the Payment Gateway. 4. The system returns the Driver to available status so the Driver can receive new requests (see 3.2.4).
Alternative Paths	If the Payment Gateway declines the charge, the ride is still marked “Completed” and the Customer is notified to update their payment method; the balance remains due.
Postcondition	The ride status is “Completed” and a Payment record has been created.
Exception Paths	None.
Other	The Driver cannot receive new ride requests again until Confirm Drop-off has been completed.

3.2.9 Rate Ride

Use Case Name	Rate Ride
Xref	Section 2.2 step 7
Trigger	A ride reaches “Completed” status.
Precondition	The ride status is “Completed” and the requesting party has not already rated this ride.
Basic Path	1. The system prompts the Customer to rate the Driver, and separately prompts the Driver to rate the Customer, on a 1-to-5 scale with an optional comment. 2. The rating is stored as a Rating record linked to the ride. The system recalculates the average rating of the party who was rated.
Alternative Paths	If either party rates the ride 2 stars or lower, the system automatically creates a BlockedPair record for that Customer-Driver combination (see 3.2.12) so that they are not matched again.
Postcondition	A Rating record exists for the rating party, and average ratings are updated.
Exception Paths	A party may decline to rate the ride; no Rating record is created for that party in that case.
Other	This use case is invoked independently by the Customer and by the Driver.

3.2.10 Tip Driver

Use Case Name	Tip Driver
Xref	Section 2.2 step 7
Trigger	The Customer selects to add a tip after rating the Driver.
Precondition	The ride status is “Completed.”
Basic Path	1. The system presents suggested tip amounts and a custom amount option. 2. The Customer selects or enters a tip amount and confirms. 3. The system charges the tip to the Customer's stored payment method through the Payment Gateway and updates the ride's Payment record.
Alternative Paths	The Customer may skip tipping.
Postcondition	The Payment record for the ride reflects the tip amount, if any.
Exception Paths	If the Payment Gateway declines the tip charge, the Customer is notified and the tip is not applied.
Other	A tip may be added any time within 14 days of the ride, not only immediately after drop-off.

3.2.11 Report Dissatisfied Ride

Use Case Name	Report Dissatisfied Ride
Xref	Section 2.2 step 7
Trigger	The Customer or Driver selects to report a problem with a completed ride.
Precondition	The ride status is “Completed.”
Basic Path	1. The reporting party selects a reason category (for example, safety concern, rude behavior, unsafe driving, incorrect route, or other) and enters a description. 2. The system creates a MonitoredEvent record of type “Dissatisfaction Reported,” linked to the ride and the reporting party. 3. The system notifies a Support Representative for review.
Alternative Paths	None.
Postcondition	A MonitoredEvent record exists for the report.
Exception Paths	The reporting party may cancel out of the report form without submitting.
Other	A report of this kind, together with a rating of 2 stars or lower, triggers the automatic pairing block described in 3.2.9 and 3.2.12.

3.2.12 Match Driver and Customer

Use Case Name	Match Driver and Customer
Xref	Section 2.2 steps 3-4
Trigger	Invoked whenever ride requests are shown to a Driver (3.2.4) or a Driver's acceptance is evaluated (3.2.5).
Precondition	A BlockedPair table exists recording Customer-Driver combinations that must not be matched.
Basic Path	1. Before a ride request is included in a Driver's nearby-requests list, the system checks whether a BlockedPair record exists for that Customer and Driver. 2. If a BlockedPair record exists, the request is excluded from that Driver's list, and that Driver is excluded from the Customer's potential matches.
Alternative Paths	None.
Postcondition	A Customer and Driver who have a BlockedPair record are never matched to the same ride.
Exception Paths	None.
Other	BlockedPair records are created automatically, as described in 3.2.9, when either party rates a ride 2 stars or lower.

3.2.13 Monitor Ride Activity

Use Case Name	Monitor Ride Activity
Xref	Background section, “system monitors for...”
Trigger	Runs continuously as a background process; it is not directly triggered by a Customer or Driver.
Precondition	The system has ride, rating, and event data to evaluate.
Basic Path	1. The system watches for Customers who repeatedly request and then cancel rides, or who are not present at the pickup location and time. 2. The system watches for Drivers who do not reach the pickup location within the estimated arrival time, or who do not reach the destination within the estimated trip time. 3. The system watches for Customers or Drivers who report a dissatisfactory ride, and for below-threshold ratings. 4. The system watches for other unusual conditions, including an unusually high rate of ride cancellations by a single account, repeated payment failures, and a GPS-reported route that deviates significantly from the expected route. Each detected condition is written to the MonitoredEvent table and, if it exceeds a configured threshold, flagged for review by a Support Representative.
Alternative Paths	None.
Postcondition	MonitoredEvent records exist for all detected conditions, and flagged accounts are queued for review.
Exception Paths	None.
Other	This use case has no human trigger; it is included here because it is a functional requirement of the system, not because it is invoked by an actor action.

3.3. Logical Structure of the Data

The logical structure of the data to be stored in the Rideshare System's relational database is given below. Figure 3 shows the seven entities and their relationships; each is described afterward in terms of its fields, keys, and how it relates to the other entities.

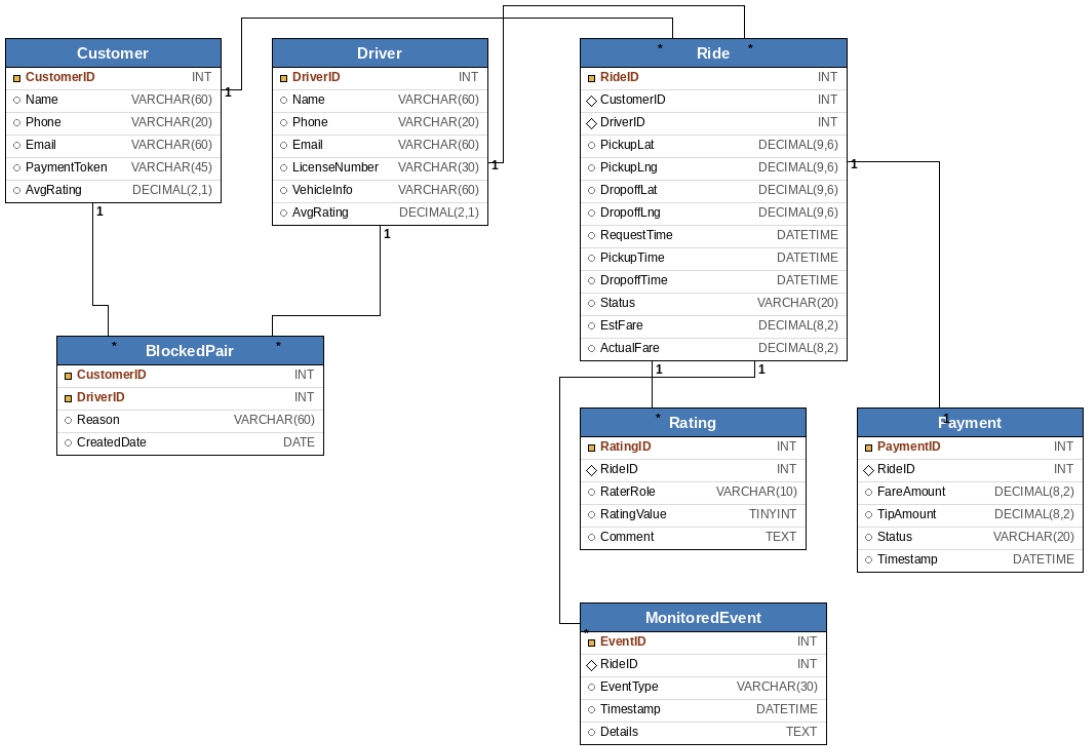

Figure 3 - Logical Structure of the Rideshare Data

The data element descriptions for each of these data entities are as follows. In each table, PK marks the primary key and FK marks a foreign key that references another entity's primary key.

Customer Data Entity

Data Item	Type	Description	Comment
CustomerID (PK)	INT	Unique identifier for the customer	
Name	VARCHAR(60)	Customer's full name	

Phone	VARCHAR(20)	Customer's phone number	Used for driver/customer contact and SMS notifications
Email	VARCHAR(60)	Customer's email address	
PaymentToken	VARCHAR(45)	Reference token for the Customer's stored payment method	Not the actual card number; obtained from the Payment Gateway
AvgRating	DECIMAL(2,1)	Rolling average of ratings received from Drivers	Recalculated after each new Rating row (3.2.9)

Driver Data Entity

Data Item	Type	Description	Comment
DriverID (PK)	INT	Unique identifier for the driver	
Name	VARCHAR(60)	Driver's full name	
Phone	VARCHAR(20)	Driver's phone number	
Email	VARCHAR(60)	Driver's email address	
LicenseNumber	VARCHAR(30)	Driver's license number	Verified during account setup
VehicleInfo	VARCHAR(60)	Make, model, color, and license plate of the Driver's vehicle	Shown to the Customer once a ride is accepted
AvgRating	DECIMAL(2,1)	Rolling average of ratings received from Customers	Recalculated after each new Rating row (3.2.9)

Rating Data Entity

Data Item	Type	Description	Comment
RatingID (PK)	INT	Unique identifier for the rating	
RideID (FK)	INT	The ride being rated	References Ride.RideID
RaterRole	VARCHAR(10)	Whether the rating was given by the Customer or the Driver	Values: Customer, Driver
RatingValue	TINYINT	Star rating from 1 to 5	A value of 2 or lower triggers 3.2.9's automatic BlockedPair rule
Comment	TEXT	Optional free-text comment	

Ride Data Entity

Data Item	Type	Description	Comment

RideID (PK)	INT	Unique identifier for the ride	
CustomerID (FK)	INT	The Customer who requested the ride	References Customer.CustomerID
DriverID (FK)	INT	The Driver assigned to the ride	Null until a Driver accepts the request (3.2.5)
PickupLat, PickupLng	DECIMAL(9,6)	Pickup location coordinates	Captured by GPS in 3.2.1
DropoffLat, DropoffLng	DECIMAL(9,6)	Destination coordinates	Entered by the Customer in 3.2.1
RequestTime	DATETIME	When the ride was requested	
PickupTime	DATETIME	When the Driver confirmed pickup	Set in 3.2.7
DropoffTime	DATETIME	When the Driver confirmed drop-off	Set in 3.2.8
Status	VARCHAR(20)	Current state of the ride	Requested, Accepted, Driver Arrived, In Progress, Completed, Cancelled by Customer, Cancelled - No Show
EstFare	DECIMAL(8,2)	Fare estimate shown before the ride (3.2.3)	
ActualFare	DECIMAL(8,2)	Fare actually charged after drop-off (3.2.8)	

Payment Data Entity

Data Item	Type	Description	Comment
PaymentID (PK)	INT	Unique identifier for the payment record	
RideID (FK)	INT	The ride this payment is for	References Ride.RideID; one Payment row per Ride
FareAmount	DECIMAL(8,2)	Amount charged for the ride itself	Charged in 3.2.8
TipAmount	DECIMAL(8,2)	Amount charged as a tip	Added in 3.2.10; may be $0.00
Status	VARCHAR(20)	Payment Gateway transaction status	Approved, Declined, Pending
Timestamp	DATETIME	When the payment was processed	

Two additional entities, MonitoredEvent and BlockedPair, are also part of the data model shown in Figure 3 but are described here in narrative form rather than as separate tables.

The MonitoredEvent entity has the following fields: EventID (primary key), RideID (foreign key to Ride), EventType (for example, “Cancelled by Customer,” “Driver Late,” “Customer No-Show,” or “Dissatisfaction Reported”), Timestamp, and Details. Each row records one condition detected by the Monitor Ride Activity use case (3.2.13) for a specific ride. A single Ride may have zero or more MonitoredEvent rows.

The BlockedPair entity has the following fields: CustomerID and DriverID, which together form a composite primary key and are also foreign keys to Customer and Driver respectively, plus Reason and CreatedDate. A row in this table means the given Customer and Driver must never be matched to the same ride again (see 3.2.9 and 3.2.12). A single Customer may be blocked from zero or more Drivers, and a single Driver may be blocked from zero or more Customers.

3.4. Nonfunctional requirements

Performance: The system shall respond to ride requests, fare estimate calculations, and nearby-request list updates within 2 seconds under normal load and within 5 seconds under peak load. The mobile apps shall refresh a Driver's location on the Customer's tracking screen at least once every 5 seconds while a ride is in progress.

Availability: The system shall maintain a minimum uptime of 99.9% over any given month, since Customers and Drivers depend on the app being reachable at all times of day. The system shall support at least 10,000 concurrent active users without degradation in response time.

Security: All data transmitted between the mobile apps, the Rideshare System, the Payment Gateway, and the GPS/Mapping Service shall be encrypted using TLS 1.3. Customer payment method tokens shall never be stored in plain text, and the system shall comply with applicable data protection regulations, including GDPR and PCI-DSS for payment data.

Usability: A first-time Customer shall be able to complete a ride request, from opening the app to receiving a fare estimate, in no more than 3 taps after entering a destination. The Driver app shall present an incoming ride request with pickup location, destination, and price within 1 second of the request becoming visible, so that a Driver can accept or ignore it quickly while driving.

Maintainability: The system shall be built from modular components (Customer service, Driver service, Ride matching service, Payment integration, Monitoring service) so that any one component can be updated or redeployed independently. Each component shall maintain automated test coverage of at least 80% and shall be accompanied by up-to-date API documentation.

Portability: The Customer and Driver mobile applications shall run on both iOS 16+ and Android 12+ without platform-specific business logic, and the backend services shall be deployable to any standard container runtime (for example, Docker/Kubernetes) without modification.

Reliability: In the event of a Payment Gateway or GPS/Mapping Service outage, the system shall degrade gracefully: rides already in progress shall not be cancelled solely due to the outage, and the system shall queue and retry payment charges for up to 24 hours before flagging the ride for manual reconciliation.