Software Requirements Specification
Taxi Tap by Git It Done

Contents

1 Introduction

Taxi Tap is a mobile platform designed to revolutionize South Africa's minibus taxi industry by digitizing route information, eliminating the need for constant hooting, and creating a semi-structured booking system while preserving the flexibility that makes taxis an essential mode of transport. The system connects passengers and taxi operators through a location-aware mobile application that facilitates taxi requests, communicates passenger locations, and provides real-time vehicle tracking – all without fundamentally changing the existing system's multi-passenger, flexible route nature.

2 User Characteristics

The users of the Taxi Tap system are expected to fit into the following groups:

2.1 Driver User Characteristics

Attribute	**Description**
Familiarity with Mobile Technology	Varies widely; some drivers may be tech-comfortable, while others may struggle with apps.
Access to Reliable Internet and Data	Often limited or inconsistent; drivers operate in areas with poor signal or expensive data costs.
Preferred Language and Communication Style	Preference for local languages (e.g. Zulu, Xhosa, Sesotho).
Attention Capacity While Driving	App must require minimal taps and distractions to ensure safe usage while driving.
Trust and Skepticism Toward New Technology	Some skepticism exists due to concerns about surveillance, manipulation, or job security.
Goals and Incentives for Using the App	Seeking more passengers, quicker pickups, and less idle time, while maintaining their routine.

Table 1: Driver User Characteristics and Considerations

2.2 Passenger User Characteristics

Attribute	**Description**
Digital Literacy	Ranges from students and workers (tech-savvy) to commuters with limited app experience.
Access to Reliable Internet and Data	Frequently encounters low or no connectivity, especially in transit.
Reasons for Using the Platform	Seeks reliable transport, less waiting, and a safer way to locate and use taxis.
Preferred Language and Communication Style	May prefer local languages (e.g. Zulu, Xhosa, Sesotho).
App Usage Context (Where & When)	Often uses the app in crowded, noisy, or busy settings like taxi ranks.
Concerns Around Trust and Safety	Wants to be sure drivers are legitimate and that their location and personal data are protected.
Platform Interaction Needs	Needs to discover taxis, request rides, track driver arrival, and receive ride notifications.

Table 2: Commuter User Characteristics and Considerations

3 User Stories

3.1 Passenger User Stories

User Story	**Acceptance Criteria**	**Definition of Done**
Account Registration & Login	As a passenger, I want to sign up and log in to my account, so that I can securely access and use the Taxi Tap app.	Given that I am on the app's welcome screen, When I choose "Sign Up" or "Log In" and enter valid details, then I should be authenticated and taken to the home screen. Based on my input criteria, I am taken to the home page of Taxi Tap.
View Available Taxis and Routes	As a passenger, I want to view available taxis and their routes on a map, so that I can choose one that matches my travel needs.	Given I am logged in, When I open the home screen, then I should see nearby taxis on a map with route or destination labels. The map displays icons of nearby taxis, including route or destination tags, when available.

Set Pickup and Destination	As a passenger, I want to share my location and set a destination, so that drivers can find me and pick me up efficiently.	Given that I have granted location access, when I enter or select a pickup and destination point, then the app should confirm my trip details and show nearby taxis. Pickup and destination are confirmed and displayed; nearby taxis are suggested based on the selected route.
Book a Seat and Get Confirmation	As a passenger, I want to book a seat on a taxi and receive confirmation, so that I'm guaranteed a spot before the taxi arrives.	Given I've selected a taxi, When I tap "Reserve a seat" and confirm, Then I should receive a booking confirmation and a ride status update. A booking confirmation message appears with the selected taxi details and current ride status.
Track Assigned Taxi in Real-Time	As a passenger, I want to track my assigned taxi in real time, so that I know when and where to expect pickup.	Given my booking is confirmed, When I open the tracking screen, then I should see the taxi's live location and estimated time of arrival. The assigned taxi is visible on the map with a live location marker and updated ETA.
Receiving Alerts When Taxi is Nearby	As a passenger, I want to receive alerts when the taxi is nearby, so that I can be ready at the pickup location.	Given the assigned taxi is approaching, When it is within 500 meters, then I should receive a push notification that it's nearby. A push alert is triggered and received once the taxi enters the defined proximity radius.
See Available Seats	As a passenger, I want to see how many seats are available, so that I can decide whether to book a seat or wait.	Given I view a taxi on the map or booking screen, When I open its details, then I should see the number of available seats.

		The number of available seats is clearly shown for each listed or selected taxi.
Rate Completed Trip	As a passenger, I want to rate my trip after completion, so that I can provide feedback to help improve the service.	Given my ride has ended, When I open the app, then I should be prompted to leave a 1–5 star rating and optional comments. The rating form appears automatically after the ride ends, and feedback is successfully submitted to the system.
Use App on Low Bandwidth	As a passenger, I want to use the app on low bandwidth, so that I can still interact with core features in areas with poor connectivity.	Given I have limited internet access, When I open the app, then I should still be able to view saved routes, taxis, and queue a ride request that sends once reconnected. The app functions with cached map data and stores ride requests locally, syncing once connectivity is restored.
View All Available Stops	As a passenger, I should be able to see all the available stops that are there during my trip.	Given I'm on a selected route, when I view route details, then I should see a list or map of all possible stops. A stop list or map view is shown, detailing all available stops along the route.
See Estimated Time to Destination	As a passenger, I should be able to see how long it will take to reach my destination or drop-off spot.	Given I've set a destination, when I view trip details, then I should see the estimated time remaining. ETA is shown dynamically on screen and is updated with real-time traffic and route changes.

Table 3: Commuter user stories

3.2 Driver User Stories

User Story	**Acceptance Criteria**	**Definition of Done**

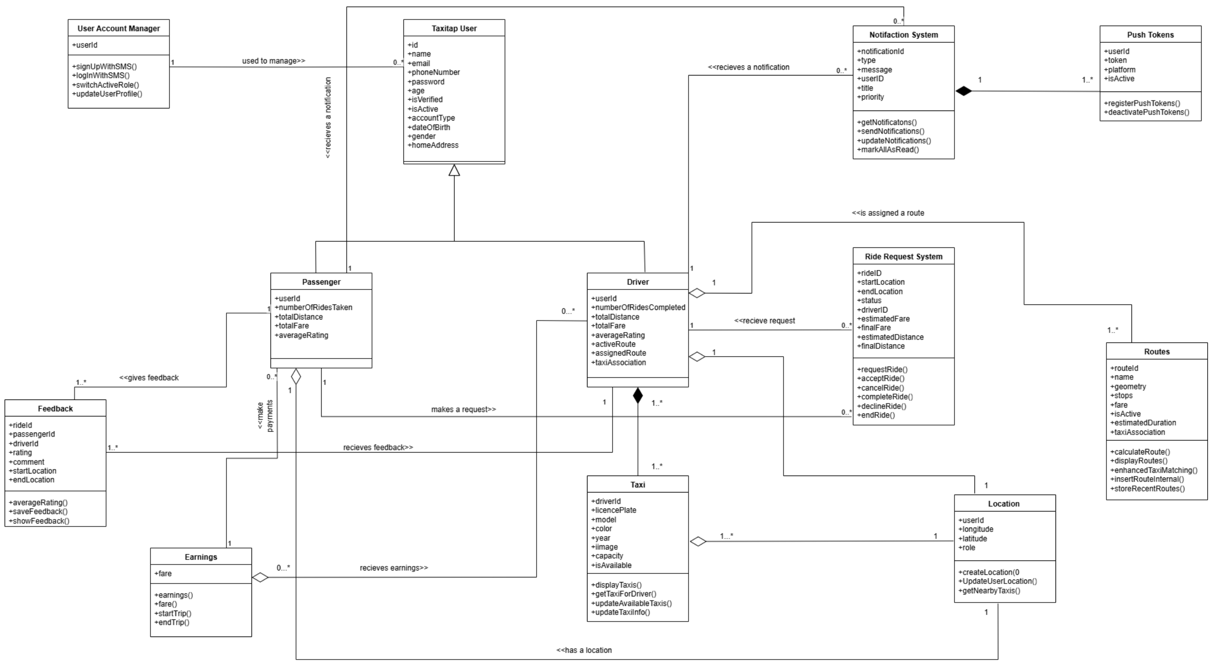
Account Registration & Login	As a driver, I want to sign up and log in to my account, so that I can securely access and use the Taxi Tap app.	Given that I am on the app's welcome screen, when I choose "Sign Up" or "Log In" and enter valid details, then I should be authenticated and taken to the home screen. Based on my input criteria, I am taken to the home page of Taxi Tap
Announce Route & Destination	As a driver, I want to input the route I will be taking and the destination, so that passengers can see if I'm heading in their direction.	Given that I'm logged in, when I set my starting point and destination, then the route is visible to nearby passengers. The route is stored and displayed to the eligible passenger's interface.
Go Online/Offline	As a driver, I want to go online or offline as needed, so that I can control when I am available to receive ride requests.	Given that I'm on the driver dashboard, when I tap "Go Online" or "Go Offline", then my status is updated accordingly and affects request visibility. The driver's online/offline status is reflected, and the passenger can no longer see the taxi on the map.
Receive Ride Requests	As a driver, I want to receive ride requests from nearby passengers, so that I can choose which pickups to accept.	Given that I am online and have an active route, when a passenger requests a ride, then I receive a notification with request details. Ride requests from matching passengers are delivered in real-time to the driver's interface.
Accept or Decline Requests	As a driver, I want to accept or decline a ride request, so that I can manage my route and taxi capacity efficiently.	Given I have received a ride request, when I tap "Accept" or "Decline", then the system updates the request status and notifies the passenger. Accepted rides appear on the active list; declined requests are logged and cleared.

View Map & Navigation	As a driver, I want to see a map with passenger pickup and route directions, so that I can navigate efficiently.	Given that I have one or more assigned pickups, When I open the map view, then I should see my location and passenger's location. Live maps with GPS and routing is functional and accurate within the app.
Update Seat Availability	As a driver, I want to update how many seats are available in my taxi, so that passengers can decide whether to book or wait.	Given that I've started a trip or gone online, when I adjust seat count manually, Then, passengers see the updated availability. Seat count updates in real time and is reflected in the passenger's booking screen.
Receive Alerts for New Requests or Updates	As a driver, I want to receive real-time notifications, so that I don't miss ride requests or updates while driving.	Given that I am online, when a new request or important event occurs, then I receive a push notification with the relevant details. Push and in-app alerts trigger correctly and lead to actionable pages.
Work Offline (Partial Functionality)	As a driver, I want to continue using key features even when I'm offline, so that I can operate in areas with poor connectivity.	Given that I am offline or have poor signal, when I open the app, then I should be able to see cached routes and queue ride requests. The app stores critical data locally and syncs changes once reconnected.
Indicate Taxi Association	As a driver, I should be able to indicate which taxi association I am a part of.	Given I am registering or editing my profile, when I select or input my association, then it should be linked to my driver profile. Association name is stored and reflected in the driver's profile and backend database.
Receive Route from Association	As a driver, I should be assigned a route by my taxi association.	Given I belong to a taxi association, when I log in or go online, then the assigned route from the association should appear.

		System pulls assigned route from the association records and displays it on the app.

Table 4: Driver user stories

4 Domain Model



5 Functional Requirements

R1: User Account Management

- R1.1: Users should be able to register as either a driver or a passenger.
- R1.2: Users should be able to update their Profile information.
- R1.3: The system should support role-based access control for passenger and driver interfaces.
- R1.4: Users should be able to reset or change their passwords.
- R1.5: Users should be able to switch between being a driver and a passenger.

R2: Location Services

- R2.1: The system should track driver locations in real-time using GPS.
- R2.2: The system should determine passenger locations for pickup requests.
- R2.3: The system should calculate proximity between taxis and passengers.

- R2.4: The system should send proximity alerts to notify passengers when their requested taxi is approaching.
- R2.5: The system should display estimated time of arrival for approaching taxis.

R3: Taxi Request System

- R3.1: Passengers should be able to request taxi pickups based on their location.
- R3.2: Passengers should be able to see nearby available taxis.
- R3.3: Drivers should be notified of nearby passenger pickup requests.
- R3.4: Drivers should be able to accept or decline pickup requests.
- R3.5: Passengers should be able to specify their destinations.

R4: Route Management

- R4.1: The system should allow drivers to announce their routes.
- R4.2: The system should display taxi routes to passengers.
- R4.3: The system should allow drivers to indicate their destinations.
- R4.4: The system should support flexible drop-off points along routes.
- R4.5: The system should display route information in a visual format suitable for quick comprehension.

R5: Taxi Status Information

- R5.1: The system should display real-time taxi tracking showing vehicle location.
- R5.2: The system should show available seats in approaching taxis.
- R5.3: The system should allow drivers to update their seat availability status.
- R5.4: The system should indicate taxi status (en route, picking up, full, etc.).

R6: Notifications

- R6.1: The system should send push notifications for taxi proximity alerts.
- R6.2: The system should notify passengers when their requested taxi accepts or declines the pickup.
- R6.3: The system should notify drivers of new nearby passenger requests.
- R6.4: The system should provide ETA updates to waiting passengers.
- R6.5: The system should send notifications even with limited connectivity.

R7: Passenger Destination Management

- R7.1: The system should allow passengers to specify their drop-off locations.

R8: User Interface

- R8.1: The system should provide separate interfaces for passengers and drivers.
- R8.2: The system should offer a clean, easy-to-use interface with visual elements.
- R8.3: The system should support multiple South African languages.

R9: Rating and Feedback

- R9.1: Passengers should be able to rate drivers/taxis.
- R9.2: The system should collect feedback on routes and service.

R10: Fare Management

- R10.1: The system should calculate fare estimates based on route and distance.
- R10.2: The system should support cash payment options.
- R10.3: The system should track payment status for trips.

R11: Taxi Identification

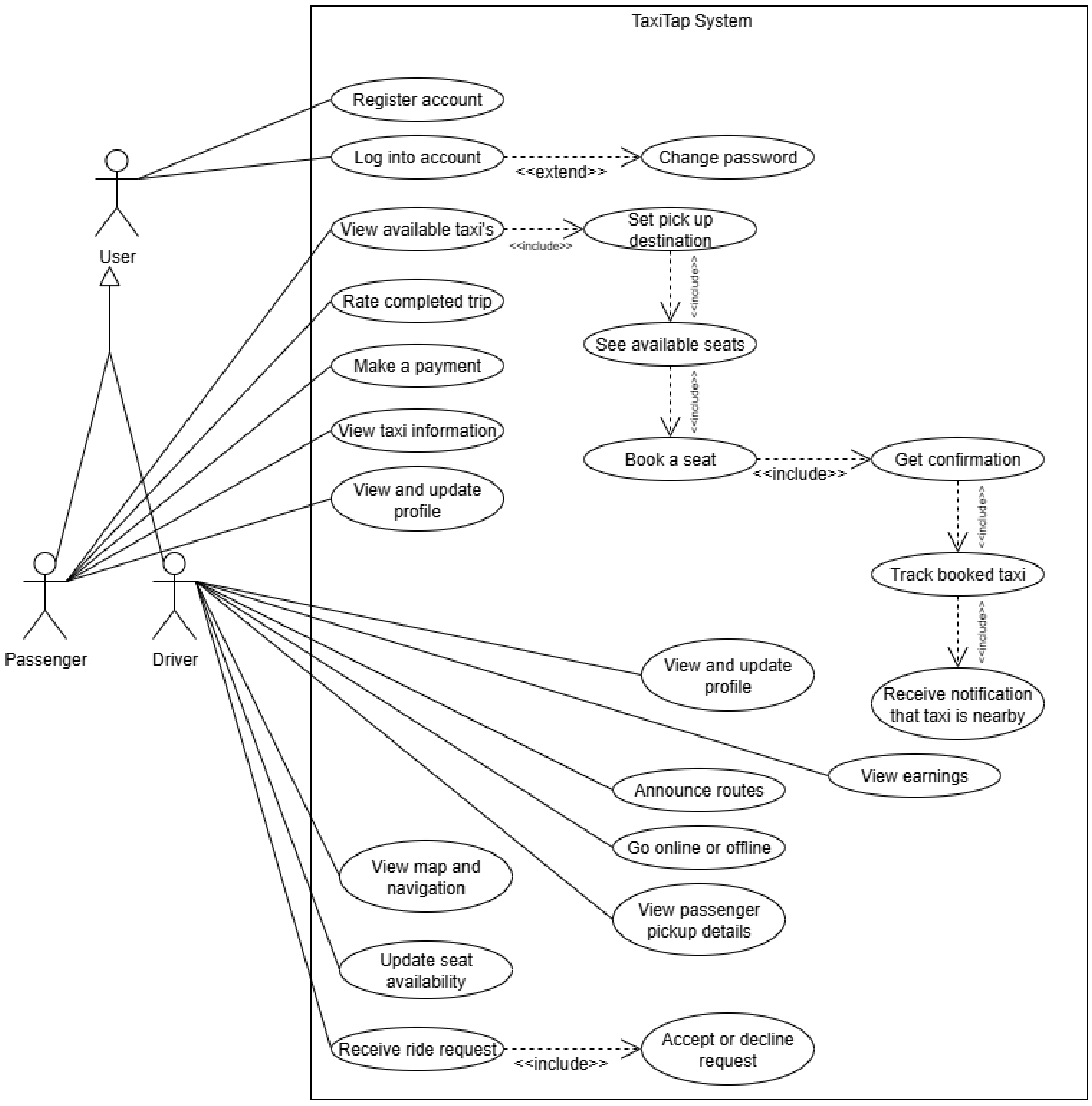
- R11.1: The system should provide unique identifiers for each taxi.
- R11.2: The system should support pin-based taxi identification and verification.
- R11.3: The system should display taxi information (registration, operator) to passengers.

R12: Safety Features

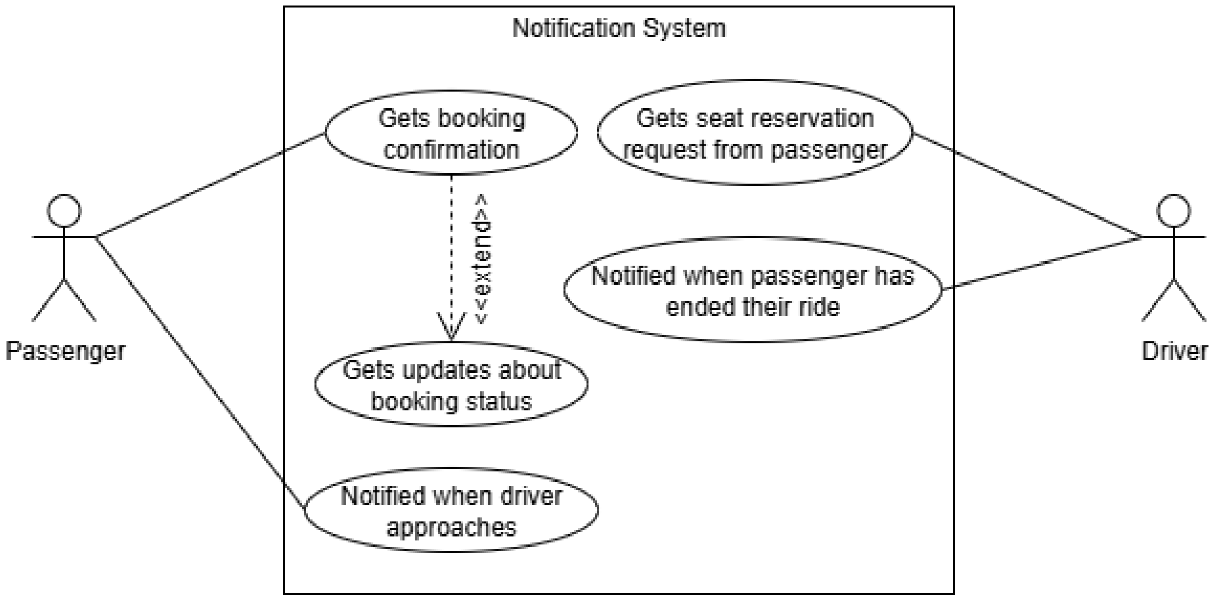
- R12.1: The system should include emergency contact features.

6 Use Case Diagrams

Overall System

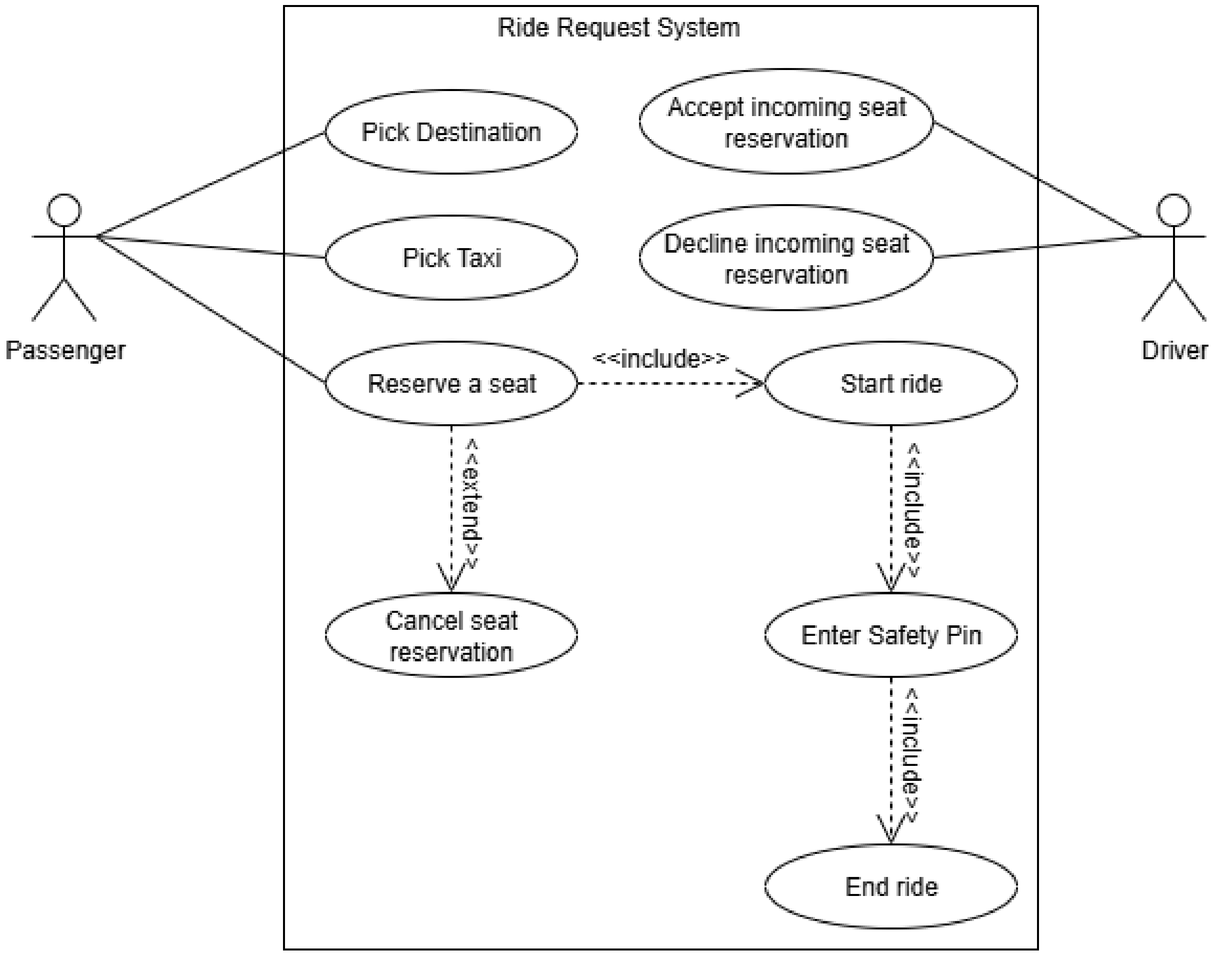


Notification System

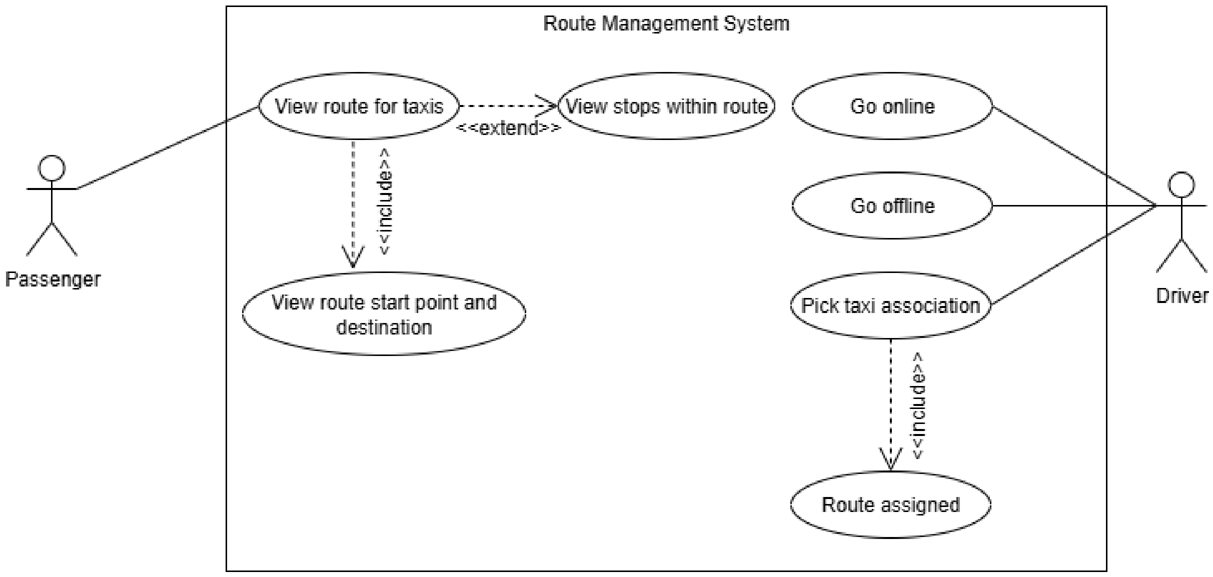


Notification System
Gets booking confirmation
Gets seat reservation request from passenger
<<extend>>
Notified when passenger has ended their ride
Passenger
Driver
Gets updates about booking status
Notified when driver approaches

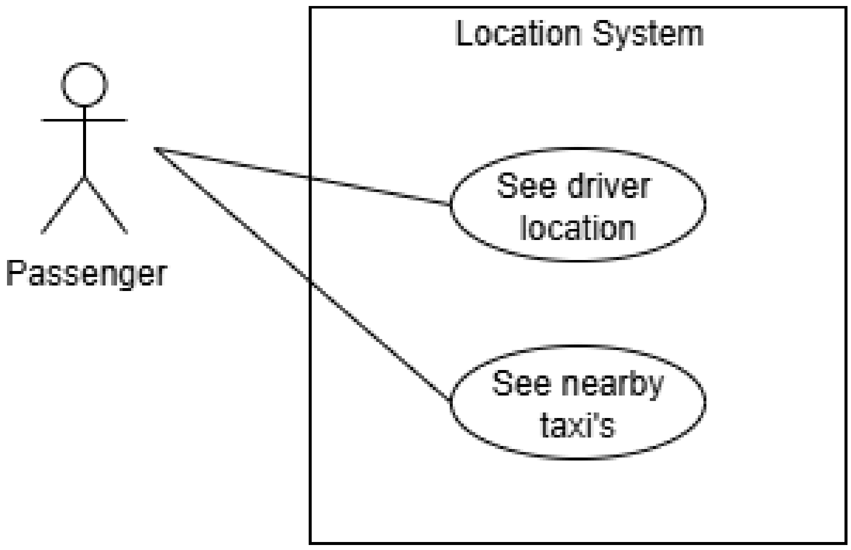
Ride Request



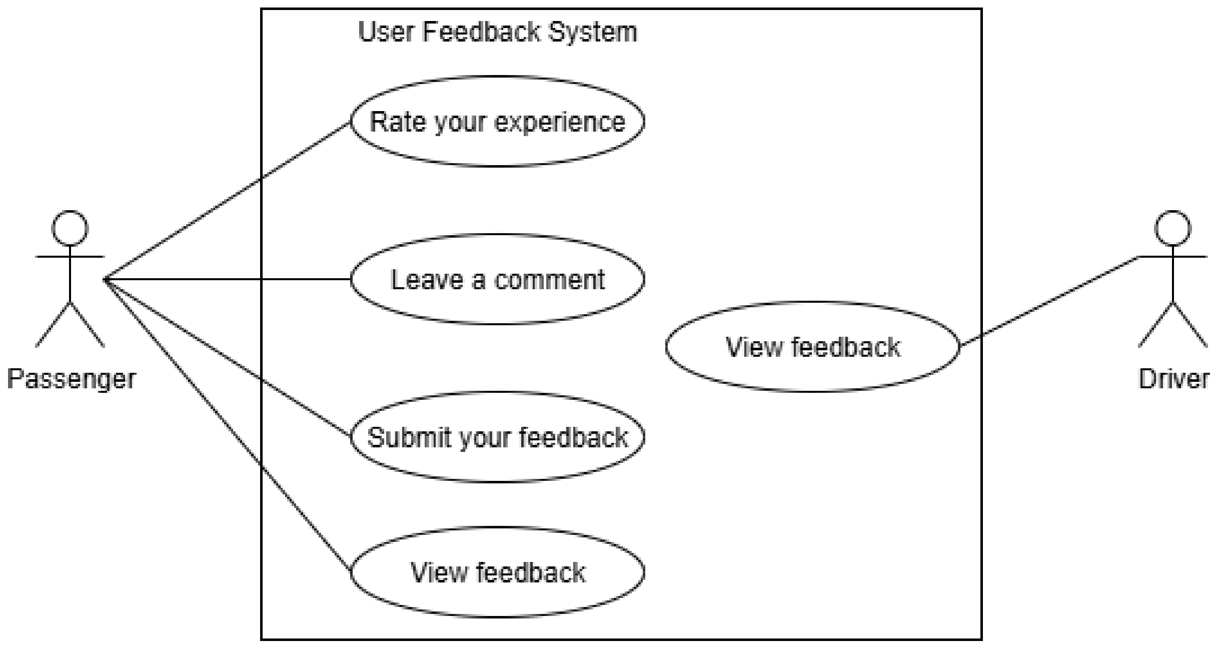
Route Management System



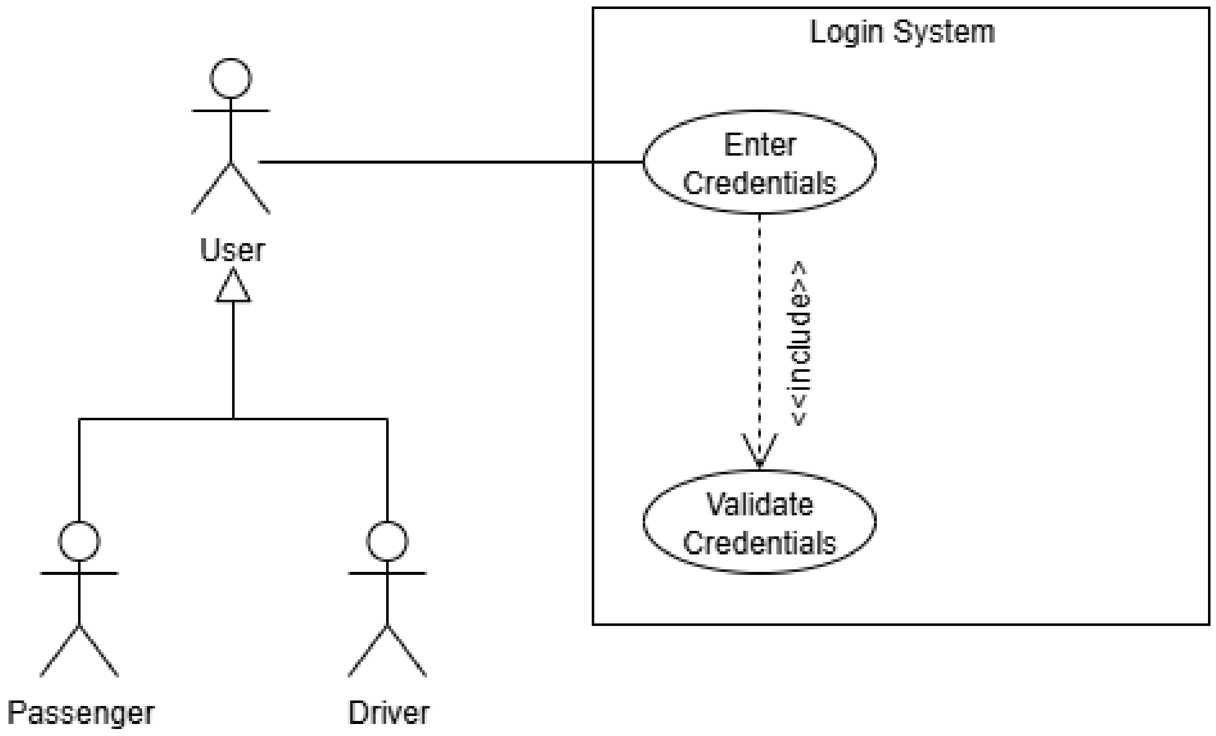
Location System



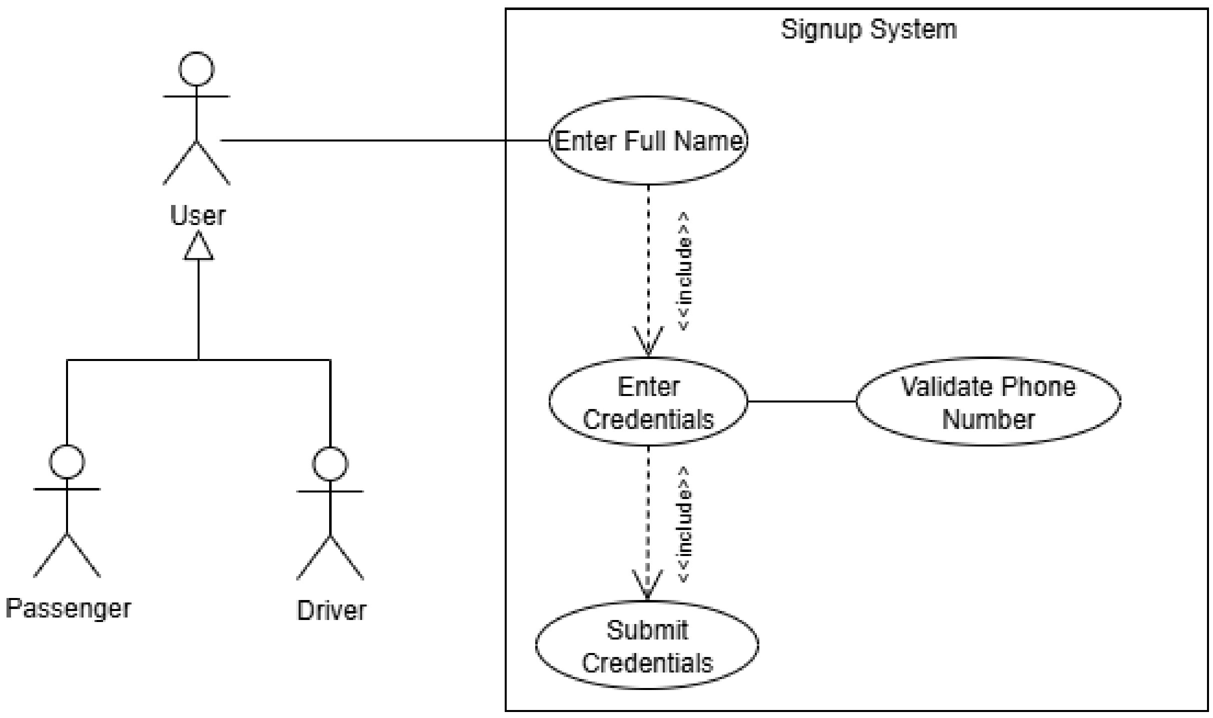
Feedback System



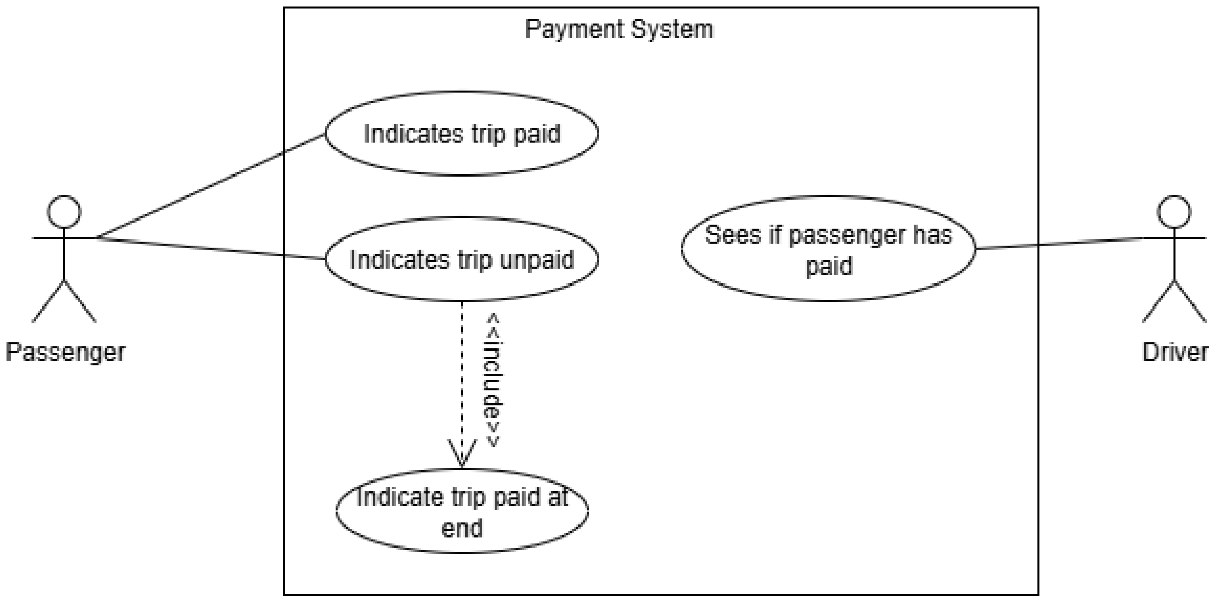
Login System



Signup System



Payment System



7 Technology Requirements

7.1 Frontend

Expo (React Native with TypeScript)
Why we chose to use Expo?

- **Cross-platform Compatibility:** Code once, deploy to both Android and iOS.
- **Native Features:** Access to GPS, accelerometer, push notifications, offline storage, camera, QR scanning, etc.
- **Web Support:** Leverages Expo Web for rendering web-based dashboards and admin panels.
- **Live Reloading & Fast Iteration:** Expo Go provides hot reloading and rapid prototyping with a unified development experience.
- **Battery & Data Optimization:** React Native ecosystem provides fine-grained control over performance, reducing overhead.

7.2 Backend

Convex (TypeScript)
Why we chose to use Convex?

- **Truly Serverless:** No provisioning, no scaling headaches. Functions, database, and auth all run in one integrated environment.
- **Built-in Database:** Convex provides a powerful document-oriented database that supports relations, IDs, indexes, and real-time reactivity.
- **Type Safety:** Schema definition is in TypeScript, ensuring end-to-end type safety from backend to frontend.
- **Zero DevOps:** No need to manage infrastructure or containers. Deploy directly from your project.
- **Realtime Sync:** Built-in support for reactive queries allows passengers to see live taxi updates, seat availability, and ETA.

Convex Database Architecture

- **Document Store:** Convex uses collections of JSON-like documents, like MongoDB, but with built-in schema validation.
- **Indexes:** Automatic indexing on IDs and custom indexing for optimized query performance.
- **Relationships:** You can use Convex `v.id()` to reference documents between tables, ensuring referential integrity.
- **Realtime Subscriptions:** Query results update automatically when the underlying data changes.

Convex Free Tier (as of 2025)

- Compute: Up to 1 million function calls/month.
- Storage: 1 GB document data storage.
- Bandwidth: 5 GB of egress.
- Authentication: Integrated with third-party auth providers (Firebase Auth, Clerk, etc.).
- Deployment: 1 Production Deployment and 1 Dev Deployment per project.

Perfect for COS 301: Within budget, no surprise bills, and production-grade scalability.

7.3 Key Functional Modules & Implementation Plan

User Management Subsystem

- Authentication: Convex Auth with Clerk or Firebase integration.
- Registration/Login: Role-based registration (passenger or driver) with schema enforcement.
- Profile Updates: Mutation to update user document with profile fields.
- Security: JWT-based session validation, encryption at rest and in transit.

Location Services Subsystem

- Driver Location: Periodic GPS updates using Expo Location API.
- Passenger Location: One-time or continuous tracking during trip.
- Proximity Alerts: Triggered from Convex using background function.
- ETA Calculation: Naive approach using Haversine distance and average speed (no Google Maps API due to cost).

Taxi Request Subsystem

- Request Workflow:
  - Passenger sends request with pickup and destination location.
  - Nearby drivers notified (push notification via Expo).
  - Driver accepts or rejects request.
  - Status changes handled in real time.

Route Management Subsystem

- Driver Route Declaration: Input form for common route and destination.
- Passenger View: Map view of taxis on route and destinations.
- Optimized Routing (Optional): Historical route optimization using stored patterns (stretch goal).

Notification System

- Technology: Expo Notifications API.
- Use Cases:
  - Taxi is approaching.
  - Ride accepted or declined.
  - Route changes or delays.
- Offline Support: Caching notifications locally using AsyncStorage.

Safety and Fare Management Subsystem

- Fare Estimate: Static fare matrix per route (e.g., km-based fare slabs).

7.4 Testing Frameworks

- Backend: Jest (unit and integration tests for Convex functions).
- Frontend: React Native Testing Library.
- Manual Testing: Device tests using Expo Go and emulators.

7.5 CI/CD

- Convex Deployment: Triggered via GitHub Action or manual `npx convex dev` / `convex deploy`.
- Expo Deployment: Use `eas build` and `eas submit` for App Store/Play Store releases (app store releases might note take place).
- Linting & Tests: Pre-commit lint checks with ESLint and Jest unit tests.

7.6 Version Control

- GitHub repo with main and dev branches.
- Feature branches for each core module.

7.7 Quality Requirements

Quality requirements determine the overall quality of Taxi Tap by specifying criteria that define how well the system performs and behaves.

The quality requirements mentioned below are overall, but our top 5 is specified in the architectural requirements document.

1. **Security**
   - **Encryption:** All data must be encrypted in transit and at rest using the best security practices.

- **Compliance:**
  - Data capturing and storing must adhere to the POPI act.
  - Ensure data privacy and consent handling.
- **Secure authentication:** Users must authenticate securely, and sessions must be protected.

2. **Usability**

   - **Simplicity:** The interface should be easy to use for people with varying levels of tech literacy.
   - **Accessibility:** The use of clear labels, large tap targets and minimal steps to complete key tasks.
   - **Feedback and error handling:** Provide real-time feedback for user actions, loading states and clear error messages when issues occur.

3. **Scalability**

   - The backend must scale to handle fluctuations in user or data load without performance degradation. This is automatically done by our chosen backend.

4. **Performance**

   - **Low bandwidth optimization:** The system must perform reliably under low-bandwidth or intermittent connectivity.
   - **Battery efficiency:** The app must minimize CPU, GPS and network usage to extend battery life.

5. **Reliability and Availability**

   - **Offline Support:** The app must function even without a constant internet connection, using local caching or data queuing mechanisms.
   - **High uptime:** The system should be available with minimal downtime to support driver operations throughout the day.
   - **Data integrity:** Ensure that data is not lost or duplicated during sync offline and online states.

6. **Maintainability and Extensibility**

   - **Clean architecture:** Backend and frontend systems should be modular and loosely coupled to allow easier updates, fixes, or feature additions in the future.
   - **Logging and monitoring:** Implement centralized logging and monitoring to quickly identify and resolve issues.
   - **Configurability:** Support code configurations without needing code changes.

7. **Affordability**

   - **Low data consumption:** The app must use data sparingly to remain cost-effective for users in regions with expensive or limited mobile data.
   - **Resource efficiency:** The system should minimize server and client-side consumption to reduce infrastructure and battery costs.

7.8 Design Patterns

Observer Pattern

- Pattern Type: Behavioural
- Participants:
  - Subject: Notification System
  - Observer: User
  - Concrete Observer: Passenger, Driver
- Explanation: The Observer pattern allows an object (User) to be notified automatically of state changes in another object (Notification System). This is ideal for handling events like route updates or ride status.
- Example:
  - User receives alerts from the Notification System.
  - Notification System initiates a notification when a route is announced.

Mediator Pattern

- Pattern Type: Behavioural
- Participants:
  - Mediator: Taxi Request System
  - Colleague: Passenger, Driver
- Explanation: The Mediator pattern centralizes complex communication between objects. Instead of Passenger directly interacting with Driver, requests are handled through the Taxi Request System.
- Example:
  - Taxi Request System acts as an intermediary between Passenger and Driver.
  - Passenger makes a request for pickup to a driver, but the Taxi request system acts as the middleman for this request.

7.9 Constraints

The client laid out the following constraints, by which Taxi Tap must abide, in their specification.

1. **All data must be encrypted at transit and at rest**
   All data exchanged between the mobile application and backend services will be encrypted using HTTPS with TLS (Transport Layer Security). Role-based access policies and authentication mechanisms (e.g., JWTs) ensure only authorised users can access specific system resources.

2. **POPI act**
   To ensure we abide by this, we will not collect any user data that is not necessary for the functionality of the app. With that, we will have permission set up to ensure that users are comfortable with collecting info, such as the user's location. Furthermore, we will consider providing a Terms and Conditions for the app that lays out how user data will be used.

3. **The app must function with low bandwidth, low data usage and be battery-efficient**
   We will accomplish this by having a UI that does not use too many resources and lightweight calls to the API.

4. **Budget**
   We must use AWS Free Tier platforms or any platforms that are open source or within free tier allowance.