TEAM FOUR: PAYLOAD/TRANSPORT AND NOTIFICATION PROCESSING

By Adam Mauch, Chip Koch, and Osvaldo Hernandez

Table of Contents

Executive Summary

The Payload/Transport and Notification Processing components are the driving force behind our innovative ridesharing platform, orchestrating the seamless movement of people and goods while keeping users informed every step of the way. At the core of these essential subsystems lies the unwavering commitment to delivering an unparalleled transportation experience that redefines convenience, efficiency, and customer satisfaction.

The Payload Processing component acts as the cerebral cortex, intelligently processing ride requests, optimizing resource allocation, and coordinating the intricate logistics of transportation services. Through advanced algorithms and real-time data analysis, this component ensures that every ride request is matched with the most suitable vehicle, considering factors such as proximity, availability, and user preferences. By maximizing resource utilization, we not only enhance operational efficiency but also contribute to a more sustainable future by reducing unnecessary vehicle emissions.

Seamlessly integrating with the Payload Processing component, the Transport Management component takes the reins, overseeing the execution of rides from start to finish. Acting as a virtual conductor, it monitors vehicle movements, ensures timely pickups and drop-offs, and proactively responds to any unforeseen circumstances that may arise during a journey. This component's real-time coordination capabilities are further augmented by its integration with other critical systems, such as Route Management, Incident Management, and Monitoring Systems, enabling us to provide uninterrupted service and adapt to dynamic conditions.

Complementing these core functionalities, the Notifications component plays a pivotal role in fostering transparency and establishing a sense of trust with our users. Through timely updates on ride statuses, vehicle locations, estimated arrival times, and any pertinent alerts or messages, this component ensures that users remain informed and empowered throughout their ridesharing experience. By leveraging cutting-edge communication technologies, we eliminate the uncertainties often associated with traditional transportation methods, providing users with a heightened sense of control and peace of mind.

Together, these three components form a cohesive ecosystem, working in harmony to revolutionize the way we think about transportation. By streamlining processes, optimizing resource allocation, and fostering open communication, we are poised to set new industry standards and redefine the boundaries of what a ridesharing platform can achieve.

As we continue to expand our reach and refine our offerings, the Payload/Transport and Notification Processing components will remain the driving force behind our success, propelling us towards a future where transportation is not merely a means to an end but an experience that embodies the values of convenience, sustainability, and customer-centricity.

Problem Statement

In the modern era, transportation remains a critical challenge that demands innovative solutions to address the limitations of traditional modes. The current landscape is characterized by traffic congestion, limited accessibility, environmental concerns, and the financial burdens associated with car ownership.

As urban populations continue to grow, the demand for efficient and reliable transportation services has surged, leaving commuters, business travelers, and individuals from all walks of life grappling with the complexities of navigating congested streets, locating parking spaces, and adhering to rigid public transit schedules. This struggle not only impacts productivity and quality of life but also contributes to rising carbon emissions and environmental degradation.

Moreover, the need for seamless communication and real-time updates has become paramount in an age where instant gratification is the norm. Consumers expect transparency, convenience, and heightened control over their transportation experiences, and failing to meet these expectations can lead to frustration, dissatisfaction, and a loss of customer loyalty.

The rideshare company recognizes these challenges and the urgent need to revolutionize the transportation industry. To remain competitive and meet the ever-evolving demands of users, the company must ensure proper communication between its services and customers, enabling swift, meaningful, and significant feedback and alerts as necessary. These notifications are crucial not only during active rides but also during the planning phases, such as route selection for immediate or future use, and during unexpected incidents or events that may occur on the road, at transportation hubs, or at charging stations, with or without a user present.

Effective communication is essential for fostering trust and building long-lasting relationships with customers. By keeping users informed about ride statuses, vehicle locations, estimated arrival times, and any potential disruptions or delays, the company can provide a sense of control and transparency that is often lacking in traditional transportation modes.

Furthermore, the ability to receive and respond to real-time feedback from users is invaluable in identifying areas for improvement, addressing concerns promptly, and adapting services to meet

the changing needs of the market. Failure to establish robust communication channels can lead to missed opportunities, customer dissatisfaction, and ultimately, a loss of market share.

To address these challenges, the rideshare company seeks to develop a comprehensive solution that seamlessly integrates advanced transportation management systems with cutting-edge notification and communication frameworks. By leveraging autonomous electric vehicles, intelligent routing algorithms, and a subscription-based pricing model, the company aims to revolutionize the transportation industry, setting new standards for efficiency, convenience, and customer satisfaction while contributing to a more sustainable future for urban mobility.

.

The Product Vision

Our vision is to develop a cutting-edge ridesharing platform that revolutionizes the way people commute, offering a seamless, eco-friendly, and convenient transportation solution. This platform will cater to urban dwellers and frequent travelers who value efficiency, sustainability, and a hassle-free mobility experience.

What is the product to be developed? We are developing a comprehensive ridesharing platform that combines advanced technology, intelligent routing algorithms, and a user-friendly interface to provide a superior transportation service. The platform will enable users to easily book rides, track vehicles in real-time, and enjoy a seamless door-to-door experience.

Who are the target customers and users? Our target customers and users are environmentally conscious individuals, commuters, and frequent travelers who seek a reliable, cost-effective, and sustainable alternative to traditional transportation methods. This includes urban professionals, students, families, and anyone looking to reduce their carbon footprint while enjoying the convenience of on-demand transportation.

Why should customers buy this product? Customers should choose our ridesharing platform for its unparalleled convenience, cost-savings, and commitment to sustainability. By leveraging our platform, users can avoid the hassles of driving, finding parking, and contributing to traffic congestion while reducing their environmental impact. Our platform offers a seamless experience, real-time tracking, and transparent pricing, ensuring a stress-free commuting experience every time.

What is the functionality of this system? The new system should support the following key functionalities:

1. User Registration and Account Management
2. Ride Booking and Reservation
3. Vehicle Allocation and Dispatch
4. Real-Time Tracking and Notifications
5. Payment Processing and Billing
6. Incident Reporting and Resolution
7. Feedback and Rating System
8. Analytics and Reporting

Constraints

Organizational

1. Limited resources and budget for initial development and deployment.
2. Integration with existing transportation infrastructure and data sources.

Implementational

1. Compliance with relevant regulations and industry standards.
2. Ensuring data privacy and security for users.
3. Scalability to accommodate increasing demand.

Target Platform

1. Cross-platform compatibility (web, iOS, and Android).
2. Optimized for various device types and screen sizes.
3. Seamless integration with third-party mapping and payment services.

Delivery

1. The platform must be launched within 12 months to capitalize on market demand.
2. Phased rollout planned, starting with major metropolitan areas.

Rideshare Target Environment

1. Everyone who has access to our platform and currency in digital form.
2. Initial launch in selected cities for user feedback and performance evaluation.
3. Gradual expansion to other markets based on demand and resource availability.

Test Market

To ensure the successful launch and widespread adoption of our innovative ridesharing platform, a phased rollout strategy has been meticulously designed. This approach will allow us to gather valuable user feedback, evaluate real-world performance, and refine our offering before embarking on a broader expansion.

In the initial phase, the platform will be launched in carefully selected cities, chosen based on a comprehensive analysis of market dynamics, transportation infrastructure, and projected demand.

These test markets will serve as living laboratories, providing us with invaluable insights into user preferences, pain points, and potential areas for improvement.

By closely monitoring user engagement, ride patterns, and feedback in these initial launch cities, we will be able to fine-tune various aspects of the platform, from user interfaces and booking processes to vehicle allocation and routing algorithms. This iterative approach will enable us to identify and address any potential issues proactively, ensuring a seamless and delightful experience for our customers as we expand into new markets.

The selection of test markets will be a meticulous process, taking into account factors such as population density, traffic patterns, existing transportation infrastructure, and the prevalence of environmentally conscious consumers. By targeting cities that align with our target demographics and possess the necessary infrastructure to support our innovative mobility solution, we increase the likelihood of a successful pilot and maximize the potential for valuable user feedback.

Once the platform has proven its mettle in the initial test markets, we will embark on a gradual expansion strategy, carefully evaluating demand and resource availability in potential new markets. This measured approach will ensure that we can scale our operations efficiently, while maintaining the highest standards of service quality and customer satisfaction.

Throughout the expansion process, we will continue to gather and analyze user feedback, market trends, and performance data, leveraging advanced analytics and machine learning techniques to identify opportunities for improvement and tailor our offering to the unique needs of each new market.

By adopting a phased rollout strategy and prioritizing user feedback, we demonstrate our commitment to continuous improvement and our dedication to delivering a truly exceptional ridesharing experience. This approach not only increases the likelihood of success but also positions us as a leader in the urban mobility space, setting the benchmark for innovation, sustainability, and customer-centricity in the transportation industry.

Product Schedule

Phase 1: Requirements Gathering and Analysis (2 months)
Phase 2: System Design and Architecture (3 months)
Phase 3: Development and Integration (6 months)
Phase 4: Testing and Deployment (3 months)
Targeted Delivery Date: Q4 2024

Client Acceptance Criteria

1. Seamless integration of all components and subsystems
2. Efficient vehicle allocation and dispatch
3. Accurate real-time tracking and notifications
4. Secure payment processing and billing
5. Robust incident reporting and resolution mechanisms
6. User-friendly interface and exceptional user experience
7. Compliance with relevant regulations and industry standards

Functional Requirements

1. User Registration and Authentication: Allow users to create new accounts, log in securely, and manage their profiles and payment information.
2. Ride Booking: Enable users to book rides by specifying pickup and drop-off locations, selecting vehicle types, and desired pickup times.
3. Vehicle Dispatch and Assignment: Efficiently assign available vehicles to user ride requests based on proximity, vehicle type, and availability.
4. Route Planning and Navigation: Provide optimal route planning and turn-by-turn navigation for car, considering traffic conditions and real-time updates.
5. Payment Processing: Integrate with secure payment gateways to process ride payments, subscription fees, and other transactions.
6. Notifications and Tracking: Send real-time notifications to users about ride status, car details, vehicle information, and enable ride tracking.
7. Feedback and Rating System: Allow users to provide feedback and ratings on their ride experiences for service improvement.
8. Subscription Management: Enable users to manage subscriptions, upgrade or downgrade subscription tiers, and access associated benefits.
9. Delivery Management: Support the delivery of packages and goods, including scheduling pickups, tracking shipments, and delivery notifications.
10. Reporting and Analytics: Provide reporting and analytics capabilities for administrators to monitor performance, analyze data, and make data-driven decisions.
11. User Registration and Authentication: Allow users to create new accounts, log in securely, and manage their profiles and payment information.
12. Ride Booking: Enable users to book rides by specifying pickup and drop-off locations, selecting vehicle types, and desired pickup times.
13. Vehicle Dispatch and Assignment: Efficiently assign available vehicles to user ride requests based on proximity, vehicle type, and availability.
14. Route Planning and Navigation: Provide optimal route planning and turn-by-turn navigation for cars, considering traffic conditions and real-time updates.
15. Payment Processing: Integrate with secure payment gateways to process ride payments, subscription fees, and other transactions.

16. Notifications and Tracking: Send real-time notifications to users about ride status, vehicle information, and enable ride tracking.
17. Feedback and Rating System: Allow users to provide feedback and ratings on their ride experiences for service improvement.
18. Subscription Management: Enable users to manage subscriptions, upgrade or downgrade subscription tiers, and access associated benefits.
19. Delivery Management: Support the delivery of packages and goods, including scheduling pickups, tracking shipments, and delivery notifications.
20. Reporting and Analytics: Provide reporting and analytics capabilities for administrators to monitor performance, analyze data, and make data-driven decisions.
21. Vehicle Maintenance Scheduling: Allow scheduling of vehicle maintenance and repairs to ensure a well-maintained fleet.
22. Lost and Found Management: Enable users to report lost items during rides and facilitate the process of recovering and returning lost items.
23. Incident Reporting and Management: Provide a system for users and administrators to report incidents (accidents, vehicle breakdowns, etc.) and track their resolution.
24. Emergency Assistance: Integrate with emergency services and allow users to request immediate assistance during rides in case of emergencies.
25. Accessibility Options: Offer options for users with special needs or disabilities to request accessible vehicles or accommodations.
26. Referral Program: Implement a referral program to incentivize existing users to refer new users and reward them for successful referrals.
27. Surge Pricing: Dynamically adjust pricing based on real-time demand levels and apply surge pricing during periods of high demand.
28. Promotional Offers and Discounts: Enable the creation and management of promotional offers, discounts, and coupon codes for users.
29. Integration with Third-Party Services: Provide integration capabilities with other services, such as mapping providers, weather APIs, or payment gateways.
30. Multi-Language Support: Offer language localization and support for multiple languages to cater to a global user base.

31. Real-Time Traffic Monitoring: Integrate with traffic data providers to monitor real-time traffic conditions and adjust routes and estimated arrival times accordingly.
32. Carpool/Rideshare Matching: Implement a feature to match users with similar routes for carpooling or ridesharing opportunities.
33. Vehicle Management: Provide a system for onboarding, managing, and monitoring vehicles, including inspection checks, license plate verification, and performance tracking.
34. Vehicle Telematics: Integrate with vehicle telematics systems to monitor vehicle diagnostics, fuel efficiency, and transportation behavior.
35. Loyalty Program: Implement a loyalty program to reward frequent users with points, discounts, or other incentives based on their usage.
36. In-App Communication: Enable communication between users and vehicle through an in-app messaging or calling feature for coordination purposes.
37. Multi-Payment Support: Support multiple payment methods, including credit/debit cards, mobile wallets, and other local payment options.
38. Predictive Demand Forecasting: Utilize machine learning algorithms to forecast demand patterns and optimize resource allocation accordingly.
39. Geofencing and Location-Based Services: Implement geofencing and location-based services to enhance user experiences and offer location-specific features or promotions.
40. Ride History and Trip Summaries: Maintain a detailed history of user rides and provide trip summaries, including routes, distances, costs, and other relevant details.

Nonfunctional Requirements

Performance:

- Ride options should be presented to users within 10 seconds of requesting a ride.
- Booking confirmations should be displayed within 5 seconds after payment processing.
- Subscription upgrades or cancellations should be processed within 10 seconds.
- Feedback submissions should be processed within 3 seconds.
- User information updates should be completed within 5 seconds.

Reliability:

- The system should have robust data backup and recovery strategies to ensure data integrity and facilitate recovery in case of failures or disasters.
- The system should facilitate seamless collaboration and communication across different regions and time zones for global development teams.
- Rigorous testing and quality assurance processes should be implemented to ensure the reliability, functionality, and performance of system features.

Security:

- The system should implement fraud detection and prevention measures to ensure secure transactions and protect user data.
- Access to system resources should be regulated through user roles, permissions, and access control mechanisms to ensure data security.
- The system should detect and reject user feedback containing suspicious or sensitive information to maintain data integrity and privacy.

Availability:

- The system should have minimal downtime and ensure high availability for users to access the service.
- Scheduled maintenance windows should be communicated to users in advance and minimized to reduce service disruptions.

Usability:

- User interfaces and interactions, such as ride booking, subscription management, and feedback submission, should be intuitive and easy to navigate.
- Information presented to users, including ride options, booking details, subscription tiers, and notifications, should be clear and understandable.

Product Backlog

Our Team's Backlog

User Story ID	Source Scenario	User Role	User Story Title	User Story/Story Short Description	Priority	Priority Rank	Status	User Story Type	Story Effort est. (hrs)	Dependencies	Product Owner
410	Delivery Processing	System/Admin	Ride Alerts	Ensure that notifications about ride changes, confirmations, and alerts are sent promptly	Must Have	1	Ready for Implementation	Delivery	2	Route Reservation	Group 4
411	Manage Subscription	User	Ease of Registration	Ensure easy registration for user to allow for discounts and the like to be prompted	Would Have	4	Ready for Consideration	User Interface	0.5	UI	Group 4
412	Delivery Processing	User	Ride Changes Notification	Ride updates are sent promptly and appropriately	Must Have	1	Ready for Implementation	Delivery	0.5	Route Reservation	Group 4
413	Delivery Processing	System/Admin	Transport Processing	Optimize vehicle locations by processing transport details	Could Have	3	Ready for Consideration	Route Managment	1.5	Route Reservation/Delivery	Group 4
414	Book Ride	User	Accessibility Information	Informs about vehicle accessibility availability	Could Have	3	Ready for Consideration	Reservations	0.5	Route Reservation	Group 4
415	Book Ride	User	Speedy Booking	Prompts various routes to customer quickly	Should Have	2	Ready for Consideration	Map Management	1	Route Reservation	Group 4
416	Delivery Processing	User	Ride Status	Timely notifications about ride status are given to user	Must Have	1	Ready for Implementation	Delivery	1	Route Reservation	Group 4
417	Reserve Ride	User	Vehicle Availability Notification	Vehicle availability is given to the user as soon as possible	Could Have	3	Ready for Refinement	Reservations	1	Route Reservation	Group 4
418	Book Ride	User	Quick Payment	Complete payment process is done quickly and the user is let know of their successful transaction	Should Have	2	Ready for Consideration	User Interface	1	UI	Group 4
419	Manage Subscription	User	Discount Offers	Discounts that could be applied to the current ride are prompted	Should Have	2	Ready for Implementation	User Interface	1.5	UI; Membership/Pricing/Billing	Group 4
420	Manage Account	User	Payment Options	Various payment methods are let known to customer in case of doubt	Would Have	4	Ready for Consideration	User Interface	0.5	UI	Group 4
421	Reserve Ride	User	Booking Confirmation	Booking confirmation is sent to customer as soon as possible	Must Have	1	Ready for Implementation	Reservations	1.5	Membership/Pricing/Billing	Group 4
422	Book Ride	System/Admin	Secure Payment	Ease of payment is informed to the user	Could Have	3	Ready for Consideration	User Interface	0.5	Membership/Pricing/Billing	Group 4
423	Delivery Processing	User	Ride Arrival Notification	Ride delivery status is informed to the user to allow for ride preparation	Should Have	2	Ready for Implementation	Route Managment	1.5	Delivery	Group 4
424	Delivery Processing	System/Admin	Customer Flux	Customer use spike is informed to user as requested and allowed	Should Have	2	Ready for Refinement	Reservations	1	Route Reservation	Group 4
425	Manage Account	User/Admin	Executive Priority	Executive users are informed of swiftly of vehicle availability and are given an itinerary	Should Have	2	Ready for Consideration	Subscriber Interface	1.5	Membership/Pricing/Billing	Group 4
426	Manage Subscription	User	Personalized Offer Alerts	Special offers are prompted to the user	Could Have	3	Ready for Implementation	User Interface	1.5	Membership/Pricing/Billing	Group 4
427	Delivery Processing	User	Traffic Updates	Real time traffic notifications are given to user as need be	Should Have	2	Ready for Implementation	Route Managment	1	Route Reservation/Delivery	Group 4
428	Delivery Processing	User	Busyness Status	High influx of users is informed to allowed users	Could Have	3	Ready for Consideration	Reservations	1	Route Reservation	Group 4
429	Book Ride	User	Vehicle Information	Vehicle information is given to allow user to learn about their booked vehicle's safety	Should Have	2	Ready for Implementation	Reservations	1.5	Route Reservation; Transport	Group 4

All Teams' Backlog

User Story ID	Source Scenario	User Role	User Story Title	User Story/Story Short Description	Priority	Priority Rank	Status	User Story Type	Story Effort est. (hrs)	Dependencies	Product Owner
101	Delivery Processing	Admin	Car Availability	This will be based on the number of self driving cars that are available during a specific time and within a specific area	Must Have						
102	Book Ride	User	Ride Matching	This will be an algorithm which matches riders with available cars based on destination, common patterns, etc,	Must Have						
103	Delivery Processing	Admin	Close/Open Depot	Feature that allows Depot managers to close/open depots in case of emergency or supply/demand dynamics.	Must Have						
104	Reserve Ride		Carpooling	Allows users to join reserved rides from another user, and adds their stop to the ride (append reservation)	Must Have						
105	Delivery Processing	Admin	Killswitch	As a depot manager, allows the manager to deactivate all cars in case of emergency.	Must Have						
106	Delivery Processing	Admin	Predictive Maintenance Alerts	The system provides predictive maintenance alerts for vehicles nearing maintenance thresholds. This proactive approach minimizes downtime by scheduling maintenance before issues arise.	Must Have						
107	Delivery Processing	Admin	Emergency Response Protocol	An emergency response protocol is in place for quick deployment of vehicles during emergencies like accidents or medical situations. This ensures timely assistance to users in critical situations.	Must Have						
108	Delivery Processing	Admin	Real-Time Traffic Integration	he system integrates real-time traffic data to optimize route planning and vehicle dispatching. This leads to faster and more efficient rides by avoiding congested routes.	Must Have						
109	Delivery Processing	Admin	Automated Vehicle Assignment	The system automatically assigns available vehicles to ride requests based on factors like proximity, user preferences, and real-time demand. This optimizes response times and ensures efficient use of resources.	Must Have						
110	Delivery Processing	Admin	User Notification of Vehicle Availability	Users are notified of available vehicles nearby to encourage ride requests and improve user satisfaction. This feature enhances user experience by providing timely information on vehicle availability.	Should Have						
111	Delivery Processing	Admin	read notifications	As a depot manager, allows the manager to read inbox of notifications	Should Have						
112	Delivery Processing	Admin	Reallocate Cars	This will be based on the demands reallocating unreserved cars .	Should Have						
113	Delivery Processing	Admin	Check Status	Function that allows depot managers to check the main status of cars	Should Have						
114	Delivery Processing	Admin	Check Maintenance	Function that allows depot managers to check the maintenance status of cars	Should Have						
115	Delivery Processing	Admin	Check Availability by depot	Feature that allows users to sort available cars by depot	Could Have						
116	Book Ride	User	Pricing	This will be an algorithm that determines the prices of rides based on how many cars are available	Could Have						
301	Membership	User	Member Registration	As a new user, I want to register for a membership through an online platform so that I can access autonomous vehicle rideshare services.	Must Have	1	Ready for Implementation	Subscriber Interface	2	UI	Group 3
302	Membership	User	Membership Tiers	As a user, I want to choose from various membership tiers (e.g., standard, premium) based on frequency of use and adc	Must Have	1	Ready for Consideration	Subscriber Interface	2.5	UI	Group 3
303	Membership	User	Manage Membership Details	As a member, I want to manage my membership details online so that I can update payment methods and contact information, or upgrade my membership tier as needed.	Must Have	1	Ready for Implementation	Subscriber Interface	1.5	UI	Group 3
304	Membership	User	Membership Cancellation	As a user, I want to cancel my membership easily without any penalties to ensure flexibility in my choices.	Should Have	2	Ready for Refinement	Subscriber Interface	0.5	UI	Group 3
305	Membership	User	Membership Temporary Suspension	As a member, I want the option to temporarily suspend my membership during times when I will not be using the service, without losing my benefits or rewards.	Could Have	3	Ready for Consideration	Subscriber Interface	0.5	UI	Group 3
306	Membership	User	Membership Swapping	As a member, I want the ability to fluidly switch between membership tiers to have the membership that best suits my needs.	Should Have	2	Ready for Implementation	Subscriber Interface	1	UI	Group 3
307	Pricing	User	Transparent Pricing	As a user, I want to see clear and transparent pricing before booking a ride so that I can make informed financial decisions.	Must Have	1	Ready for Implementation	Route Managment	1	UI, Route Reservation	Group 3
308	Pricing	User	Surge-Pricing Notifications	As a user, I want to receive notifications about surge pricing during high-demand periods so that I can decide the best tir	Should Have	2	Ready for Consideration	Route Managment	1.5	UI, Notifications	Group 3
309	Pricing	User	Ride Prepay	As a user, I would like to be able to pre-pay for a certain number of rides at a discounted rate to save money on transpoı	Should Have	2	Ready for Refinement	Reservations	1	UI, Route Reservation	Group 3
310	Pricing	User	Split Fare	As a user, I want the ability to split my fare between multiple people using the same ride allowing us to share the total cost of the service.	Could Have	3	Ready for Consideration	Reservations	2	UI, User Integration	Group 3
311	Pricing	User	Discount Codes	As a user, I want the ability to access various discount codes or promotional offers to further save money using this service.	Could Have	3	Ready for Consideration	User Interface	1	UI	Group 3
312	Billing and Payment	User	Payment Storage	As a user, I want to securely store multiple payment methods in my profile for ease of billing.	Must Have	1	Ready for Implementation	Subscriber Interface	2	UI	Group 3
313	Billing and Payment	User	Automated Billing	As a user, I want automated billing after each ride with a detailed breakdown sent to my email so that I can keep track of expenses without manual intervention.	Must Have	1	Ready for Implementation	User Interface	2	UI	Group 3
314	Billing and Payment	User	Billing History	As a user, I want to view my billing history and download past invoices for personal record-keeping and budgeting.	Should Have	2	Ready for Implementation	User Interface	1.5	UI	Group 3
315	Billing and Payment	User	Payment Expiration Notifications	As a user, I want to receive real-time notifications when my payment method is going to expire so that I can update it before there is any disruption in payment processing.	Should Have	2	Ready for Consideration	User Interface	2	UI, Notifications	Group 3
316	Billing and Payment	User	Automatic Payments	As a user, I want the ability to set up an automatic, default payment process for more spontaneous rides.	Could Have	3	Ready for Consideration	User Interface	1	UI, User Integration	Group 3
317	Loyalty and Rewards	User	Loyalty Program	As a frequent rider, I want to participate in a loyalty program where I can earn points for every ride and redeem them for discounts or free rides.	Could Have	3	Ready for Implementation	Subscriber Interface	2	UI	Group 3
318	Loyalty and Rewards	User	Gift Rewards	As a loyalty/rewards member, I want the option to gift rides to family members or friends with the rewards I have accumulated, allowing me to share my benefits with others.	Could Have	3	Ready for Consideration	Subscriber Interface	1.5	UI	Group 3
319	Loyalty and Rewards	User	Partnered Rewards	As a loyalty/rewards member, I want the ability to receive rewards or discounts for partnered businesses based on my use of the rideshare service.	Could Have	3	Ready for Consideration	Subscriber Interface	2	UI	Group 3
320	Loyalty and Rewards	User	Pool Rewards	As a loyalty/rewards member, I want the ability to pool my rewards with other members and use the total rewards towards discounts on a shared ride.	Could Have	3	Ready for Consideration	Subscriber Interface	2	UI	Group 3
410	Delivery Processing	System/Admin	Ride Alerts	Ensure that notifications about ride changes, confirmations, and alerts are sent promptly	Must Have	1	Ready for Implementation	Delivery	2	Route Reservation	Group 4
411	Manage Subscription	User	Ease of Registration	Ensure easy registration for user to allow for discounts and the like to be prompted	Would Have	4	Ready for Consideration	User Interface	0.5	UI	Group 4
412	Delivery Processing	User	Ride Changes Notification	Ride updates are sent promptly and appropriately	Must Have	1	Ready for Implementation	Delivery	0.5	Route Reservation	Group 4
413	Delivery Processing	System/Admin	Transport Processing	Optimize vehicle locations by processing transport details	Could Have	3	Ready for Consideration	Route Managment	1.5	Route Reservation/Delivery	Group 4
414	Book Ride	User	Accessibility Information	Informs about vehicle accessibility availability	Could Have	3	Ready for Consideration	Reservations	0.5	Route Reservation	Group 4
415	Book Ride	User	Speedy Booking	Prompts various routes to customer quickly	Should Have	2	Ready for Consideration	Map Management	1	Route Reservation	Group 4
416	Delivery Processing	User	Ride Status	Timely notifications about ride status are given to user	Must Have	1	Ready for Implementation	Delivery	1	Route Reservation	Group 4
417	Reserve Ride	User	Vehicle Availability Notification	Vehicle availability is given to the user as soon as possible	Could Have	3	Ready for Refinement	Reservations	1	Route Reservation	Group 4
418	Book Ride	User	Quick Payment	Complete payment process is done quickly and the user is let know of their successful transaction	Should Have	2	Ready for Consideration	User Interface	1	UI	Group 4
419	Manage Subscription	User	Discount Offers	Discounts that could be applied to the current ride are prompted	Should Have	2	Ready for Implementation	User Interface	1.5	UI; Membership/Pricing/Billing	Group 4

429	Book Ride	User	Vehicle Information	Vehicle information is given to allow user to learn about their booked vehicle's safety	Should Have	2	Ready for Implementation	Reservations	1.5	Route Reservation; Transport	Group 4
RM-001	Reserve Ride	User adjusts	Add/Change Reservation Drop-Off Location	As a corpoprate executive, I want to update my return delivery with multiple drop-offs so that I can stop by after work for dinner and a movie.	Should Have	2	Ready for implementation	Route Managment	1	User Integration, Route Reseravation, Route Managment	Group 2
UI-001	Register Account	User registers	Receive Discount Code for Registering	As a prospective user, I want to be tempted with a discount for registering so that I can sign up for services at a discounted price.	Would Have	1	Ready for consideration	User Interface	0.5	UI	Group 2
UI-002	Reserve Ride	User enters	Display Transportation Options for Ride Reservations	As a user, I want to enter my location, date, and time on the reservation form so that I can see available transportation options for my needs.	Must Have	1	Ready for implementation	User Interface	1	User Integration, UI, Reservation	Group 2
UI-003	Book Ride	User enters	Display Transportation Options for Ride Bookings	As a user, I want the system to use my location, date, and time when booking a ride and I want to have the option to manually enter a location so that I can see available transportation options for my needs.	Must Have	2	Ready for implementation	User Interface	2	User Integration, UI, Reservation	Group 2
RES-001	Book (or Reserve) Ride	User selects	Select Transportation Type	As a user, I want to select a transportation type from a sorted list based on type and proximity so that I can choose the most suitable transport option for my journey.	Must Have	3	Ready for implementation	Reservations	2.5	User Integration, UI, Reservation, ATP	Group 2
RES-003	Book (or Reserve) Ride	User confirms	Confirm Reservation/Booking	As a user, I want to confirm my reservation/booking so that the system can finalize my itinerary, calculate the initial price, and prompt me for payment.	Must Have	1	Ready for implementation	Reservations	1	User Integration, Pricing/Billing, Notifications	Group 2
RES-004	Book (or Reserve) Ride	User selects	Select Vehicle Type	As a user, I want to be able to select my desired type of vehicle so that I can ride in a car I like and/or ride in a car that accommodates my needs.	Should Have	1	Ready for refinement	Reservations	2	User Integration, UI, ATP	Group 2
RES-005	Book (or Reserve) Ride	User completes	Complete Ride Payment	As a user, I want to complete payment for my ride so that my ride can be locked and a delivery created.	Must Have	1	Ready for implementation	Reservations	0.5	User Integration, Payment/Billing	Group 2
RES-006	Book (or Reserve) Ride	N/A	Receive Ride Confirmation	As a user, I want to receive confirmation that my ride has been scheduled so that I can have the confidence that I will be picked up	Must Have	3	Ready for implementation	Reservations	0.5	User Integration, UI, Notifications	Group 2
UI-004	User Settings	User sets up	Set Up Reservation Notifications	As a user, I want to set up notifications for my reservation, so that I can be updated about my changes or important information regarding my trip.	Must Have	3	Ready for implementation	Reservations	1	User Integration, Notifications, Reservations	Group 2
UI-005	Log-in/Log-out	User is able to	Log-out of Account	As a user, I want to be able to log-out of my account so that I can exit the app when I am not using it and/or allow other users to log-in on my device	Must Have	2	Ready for implementation	User Interface	1	User Integration, UI	Group 2
UI-006	Register Account	N/A	Display First-Time Visitor Discount	When a new user visits the website, the system should detect the new user and offer a discounted subscription if they register then to entice the user to register.	Would Have	1	Ready for consideration	User Interface	1	UI	Group 2
UI-007	Register Account	N/A	Link New Users to Reservation Form	When a user attempts clicks to register, the system should navigate the user to the registration form.	Should Have	3	Ready for refinement	User Interface	0.5	UI	Group 2
RES-007	Book (or Reserve) Ride	User selects	Optimizing Vehicle Pick-Up Time	When a user selects a transport type from a list based on type and proximity, the system should give the user the most suitable transport type.	Must Have	3	Ready for implementation	User Interface	2.5	User Integration, Reservations, Transit Processing	Group 2
RES-008	Book (or Reserve) Ride	User selects	Refining Vehicle Options	When a user selects a transport type, the system should display only the transports that match the user's selection.	Must Have	2	Ready for implementation	User Interface	1.5	User Integration, UI, Reservations, Transit Processing	Group 2
RES-009	Book (or Reserve) Ride	N/A	Propose Alternative Routes	When the requested transport is not immediately available, the system should propose an alternative timing or an upgraded transport option.	Should Have	2	Ready for refinement	User Interface	2.5	Transit Processing, ATP	Group 2
RES-010	Book (or Reserve) Ride	N/A	Generate Ride Confirmation Details	When the user confirms the reservation (and payment), the system should finalize the booking, create a delivery record, and generate the necessary confirmation details.	Should Have	1	Ready for implementation	User Interface	1.5	Reservations, Payment/Billing, Notifications	Group 2
UI-008	Book (or Reserve) Ride	N/A	Payment Confirmation Notification	When payment is confirmed, the system should send a confirmation email to the user and set up follow-up notifications.	Should Have	1	Ready for implementation	User Interface	0.5	User Integration, Notifications	Group 2
UI-009	Log-in/Log-out	NA	Secure Personal User Data	When a user logs out of the system, the system returns the user to the main screen and keeps their data securely saved.	Must Have	1	Ready for implementation	User Interface	0.5	UI	Group 2
UI-010	Manage Account	User creates	User Profile Personalization	As a user, I want to create a profile with my preferences and ride history so that booking future rides is quicker and personalized.	Should Have	1	Ready for refinement	User Interface	0.5	User Integration, UI	Group 2
RES-011	Book (or Reserve) Ride	N/A	Vehicle Upgrade	As a subscriber, when my preferred ride isn't available, I want to be offered alternative or upgraded vehicles without delay or additional charges so that I can maintain my schedule.	Could Have	2.5	Ready for consideration	Subscriber Interface	2	User Integration, UI, Subscriber Interface (SI), ATP	Group 2
UI-011	Book (or Reserve) Ride	N/A	Various Payment Options	As a user, I want various payment methods and secure storage of my payment information to streamline the checkout process.	Could Have	2	Ready for consideration	User Interface	1	User Integration, UI, Payment/Billing	Group 2
DEL-001	Delivery Process	User reports	Report Issue	As a user, I want to be able to report issues or receive notifications if there's a problem with my ride so that I can still reach my destination on time.	Should Have	1	Ready for refinement	Delivery	1	User Integration, UI, Delivery, Incident Management	Group 2
DEL-002	Delivery Process	N/A	Collect Rider Feedback	As Marketing/Sales, I want to gather user feedback on their experience so that we can improve our service and increase user satisfaction.	Should Have	1	Ready for refinement	User Interface	3	User Integratioin, UI	Group 2
MAP-001	Book (or Reserve) Ride	User adjusts	Adjust Map Location	As a user, I want to be able to adjust my location on the map so that the vehicle picks me up at my desired location	Should Have	3	Ready for consideration	Map Management	2	UI, Map Managment	Group 2
RM-002	Delivery Process	User adds	Multiple Drop-Off Spots	As a user, I want to be able to add drop-off stops throughout my ride so that I can drop people off at multiple places	Should Have	2	Ready for refinement	Route Managment	2	UI, Route Managment	Group 2
RM-003	Delivery Process	User adds	Multiple Pick-Up Spots	As a user, I want to be able to add pick-up spots throughout my ride so that I can pick people up at multiple places	Should Have	2	Ready for refinement	Route Managment	2	UI, Route Managment	Group 2
SI-001	Manage Subscription	User adjusts	Adjust Subscription Plan	As a user, I want to be able to upgrade or downgrade my existing subscription so that I can adjust how much I pay as needed	Must Have	2	Ready for implementation	Subscriber Interface	1	User Integration, SI, Paying/Billing	Group 2
MAP-002	Delivery Process	N/A	Real-Time Route/ETA Updates	As a user, I want to be able to see route updates in real-time so that I know whether to expect delays or changes in ETA	Should Have	3	Ready for refinement	Map Management		UI, Map Managment	Group 2
MAP-003	Delivery Process	N/A	Explainations for Route Changes	As a user, I want to be provided with an explaination when the route changes so that I know why the vehicle is going off-route	Would Have	1	Ready for consideration	Map Management		UI, Map Managment	Group 2

Use Cases

Use Case: Pre-Ride Information Communication

- Participating Actors: User, System
- Flow of Events:
    1. User opens the app and navigates to the "Book Ride" option.
    2. User enters their pickup location and desired destination.
    3. User selects preferred pickup time and vehicle type, if applicable.
    4. System processes the request and checks vehicle availability.
    5. System confirms booking details and assigns a vehicle.
    6. System displays car model, plate number, and exact pickup location/time on the user interface.
    7. System sends a confirmation notification/email to the user with all ride details.
- Entry Conditions: User must be logged into the app.
- Exit Conditions: User receives all necessary pre-ride details.
- Exceptions: If no vehicles are available, the system informs the user and suggests alternative times or vehicle types.
- Quality Requirements: System response time must not exceed 5 seconds for each interaction step.

Use Case: User Delay Notification

- Participating Actors: User, System
- Flow of Events:
    1. User accesses their current bookings through the app.
    2. User selects the "Modify Booking" option for a specific ride.
    3. User enters a new desired pickup time.
    4. System checks availability for the new time.
    5. System confirms the change and updates the booking details in the database.
    6. System sends a detailed confirmation of the updated pickup time to the user via the app and email.
- Entry Conditions: User has an existing booking.
- Exit Conditions: User receives confirmation of the new pickup time.
- Exceptions: If the new time is not available, the system notifies the user and requests another time or offers to cancel the booking with a possible voucher or discount for the inconvenience.
- Quality Requirements: Notification should be sent within 2 minutes of user request.

Use Case: Incident Reporting

- Participating Actors: User, System, Incident Manager
- Flow of Events:
    1. System continuously monitors all ongoing rides for incidents.
    2. An incident occurs (e.g., traffic accident, vehicle breakdown).
    3. System detects the incident through vehicle sensors reports.
    4. System automatically logs the incident and categorizes it based on severity.
    5. System notifies the Incident Manager with detailed incident reports.
    6. Incident Manager evaluates the incident and initiates appropriate response measures.
    7. System informs the user about the incident, its impact on the trip, and expected resolution steps.
    8. System offers alternative transportation options to the user if necessary.
- Entry Conditions: An incident occurs during an active ride.
- Exit Conditions: User is informed about the incident and resolution steps.
- Exceptions: If the incident cannot be resolved quickly, the system offers the user alternative transportation options or refunds.
- Quality Requirements: Incident notifications must be sent within 1 minute of detection.

Use Case: Charging Station Occupancy

- Participating Actors: User, System
- Flow of Events:
    1. User opens the app and navigates to the charging station locator feature.
    2. User inputs their current location or desired area for charging.
    3. System accesses real-time data from various charging stations in the requested area.
    4. System analyzes the availability and operational status of each station.
    5. System displays a list of available charging stations, including details about occupancy levels, charging speeds, and costs.
    6. User selects a charging station and optionally reserves a charging slot if the feature is available.
    7. System confirms the reservation and provides navigation instructions to the station.
- Entry Conditions: User needs to charge their electric vehicle.
- Exit Conditions: User receives information about charging station availability.
- Exceptions: If no charging stations are available, the system suggests nearby alternatives or future available times.
- Quality Requirements: Information must be up-to-date and reflect real-time availability with updates every 5 minutes.

Use Case: Customer Feedback Prompt

- Participating Actors: User, System
- Flow of Events:
    1. Ride ends, and the system triggers the feedback sequence.
    2. System prompts the user with a customizable feedback form on the app.
    3. User rates the ride on various aspects such as cleanliness, and overall experience.
    4. User submits additional comments in a text field.
    5. System collects and stores the feedback in the database for quality control and analytics.
    6. System sends a thank-you message to the user for providing feedback.
    7. Data is later used for car performance reviews and service improvement measures.
- Entry Conditions: Ride has ended.
- Exit Conditions: Feedback is submitted and recorded.
- Exceptions: If the user skips the feedback, the system logs the non-response and sends a reminder after a certain period.
- Quality Requirements: The feedback prompt should load immediately after the ride ends, and the system should store feedback securely and confidentially.

Use Case: Weather Alert

- Participating Actors: User, System
- Flow of Events:
    1. System continuously monitors weather conditions for all active and scheduled ride locations.
    2. Adverse weather conditions are forecasted for a user's location.
    3. System assesses the potential impact on the user's ride.
    4. System sends a weather alert to the user's app and optionally via other communication channels like SMS or email.
    5. User receives the notification and views detailed weather conditions and safety advice.
    6. System offers to adjust ride timing or routes if feasible.
    7. User makes a decision to proceed with, reschedule, or cancel the ride based on the alert.
- Entry Conditions: User has an upcoming or active ride.
- Exit Conditions: User is informed about adverse weather conditions.
- Exceptions: If the system fails to fetch weather data due to a network issue, an error is logged and the system attempts to reconnect.
- Quality Requirements: Weather alerts should be timely, based on the latest meteorological data, and provided at least 30 minutes before the predicted weather event.

Use Case: Lost Items

- Participating Actors: User, System
- Flow of Events:
    1. User realizes they have left an item in the vehicle after a ride.
    2. User opens the app and navigates to the 'Lost Items' section.
    3. User fills out a lost item report, detailing the item and the ride.
    4. System logs the report and initiates a search by checking the vehicle.
    5. If the item is found, the system notifies the user and arranges for item return.
    6. If the item is not found, the system registers the item as lost and continues to search, notifying the user of any updates.
    7. System offers compensation or a service credit if the item cannot be found after a specified period.
- Entry Conditions: User has completed a ride.
- Exit Conditions: User is informed about the status of their lost item.
- Exceptions: If the item is not found and the user is dissatisfied, customer service steps in to handle the situation.
- Quality Requirements: The system should update the user about the search status within 24 hours and handle all user data with privacy and security.

Use Case: Real-Time Tracking

- Participating Actors: User, System
- Flow of Events:
    1. User requests real-time tracking of their approaching vehicle via the app.
    2. System accesses the vehicle's GPS data and begins transmitting its location to the user.
    3. User receives a map view on their app showing the vehicle's movement toward the pickup location.
    4. System sends incremental notifications when the vehicle is 10 minutes away, 5 minutes away, and 1 minute away.
    5. Upon vehicle arrival, the system sends an "Arrived" notification.
    6. User meets the car at the pickup point.
- Entry Conditions: User has booked a ride and the vehicle is dispatched.
- Exit Conditions: Vehicle arrives at the pickup location.
- Exceptions: If the vehicle is delayed or takes a wrong turn, the system updates the estimated time of arrival and notifies the user.
- Quality Requirements: Location updates must be accurate within a 30-second timeframe and provided in real-time.

Use Case: High-Demand Alert

- Participating Actors: User, System
- Flow of Events:
    1. System monitors vehicle availability and booking rates across different regions.
    2. System identifies high-demand conditions based on data analytics.
    3. System calculates the impact on wait times and service availability.
    4. System proactively sends high-demand alerts to users who are likely to be affected, suggesting optimal booking times or offering premium options.
    5. Users receive the alert and can decide to book immediately, wait, or choose another service option.
    6. System continues to monitor and adjust its predictions and user notifications as demand changes.
- Entry Conditions: Demand for rides increases significantly.
- Exit Conditions: User is informed about high-demand conditions and makes an informed decision.
- Exceptions: If system predictions are incorrect, leading to user inconvenience, the system may offer discounts or credits.
- Quality Requirements: Alerts must be sent within 5 minutes of demand spike detection and be 90% accurate in predicting high-demand periods.

Use Case: Service Delay Notification

- Participating Actors: User, System
- Flow of Events:
    1. An unexpected delay (traffic, vehicle issue, etc.) affects a scheduled service.
    2. System detects the delay through real-time vehicle monitoring reports.
    3. System calculates the new estimated time of arrival based on the delay.
    4. System updates the booking details and notifies the user of the delay through the app and optional SMS.
    5. User receives the updated time and can choose to wait or cancel the ride with a full refund.
    6. System offers alternative solutions if the delay exceeds a certain threshold, such as a different vehicle or a discount on future rides.
- Entry Conditions: User has an active ride booking.
- Exit Conditions: User is informed of the delay and updated estimated arrival time.
- Exceptions: If the delay extends significantly beyond the updated estimate, the system automatically offers compensation.
- Quality Requirements: Delay notifications should be communicated within 2 minutes of detection.

Use Case: Surge Pricing Notification

- Participating Actors: User, System
- Flow of Events:
    1. System continuously monitors ride demand, traffic conditions, and other relevant data to calculate fare rates.
    2. A spike in demand triggers surge pricing algorithms.
    3. System calculates the new fare rates and prepares notifications for users.
    4. Before user books a ride, the system displays a clear surge pricing notification in the app.
    5. User receives detailed information about the reason for the surge, the new rates, and options to wait until the surge drops or proceed with the booking.
    6. System provides an option to alert the user when surge pricing ends if they choose to wait.
    7. User makes an informed decision based on the provided information.
- Entry Conditions: High demand is detected in the system.
- Exit Conditions: User is notified about surge pricing and makes a booking decision.
- Exceptions: If the system fails to detect surge conditions accurately, incorrect pricing information is communicated, leading to potential refunds or customer complaints.
- Quality Requirements: Surge pricing notifications must be accurate and delivered within 1 minute of detection.

Use Case: Cancellation of Pre-Set Transportation

- Participating Actors: User, System
- Flow of Events:
    1. System identifies operational issues or unforeseen circumstances that require cancellation of pre-set routes.
    2. System assesses the impact and identifies affected users.
    3. System automatically notifies affected users about the cancellation, detailing the reasons and immediate impacts.
    4. System offers alternative transportation options and assistance in rebooking.
    5. User receives the notification and selects an alternative option or accepts a full refund.
    6. System processes the user's choice and confirms the new arrangements.
- Entry Conditions: User has a pre-set transportation schedule.
- Exit Conditions: User is notified of the cancellation and either rebooks or receives a refund.
- Exceptions: If no alternatives are available, the system offers additional compensation, such as discounts on future rides.
- Quality Requirements: Notification of cancellation should be sent as soon as the decision is made, ideally within 5 minutes.

Use Case: Recommendations for User

- Participating Actors: User, System
- Flow of Events:
    1. User completes a ride.
    2. System gathers data from the completed ride and analyzes the user's travel patterns and preferences.
    3. System uses machine learning algorithms to generate personalized recommendations for future services.
    4. System presents these recommendations to the user in a special section of the app.
    5. User reviews the recommendations and can directly book a suggested service.
    6. System monitors user interaction with the recommendations to refine future suggestions.
- Entry Conditions: Ride is completed.
- Exit Conditions: Recommendations are displayed to the user.
- Exceptions: If there is insufficient data to generate recommendations, the user is prompted to provide more preferences.
- Quality Requirements: Recommendations should be relevant and based on up-to-date user preferences and data accuracy.

Use Case: Traffic Alert

- Participating Actors: User, System
- Flow of Events:
    1. System monitors real-time traffic conditions on major routes and around user's current or future locations.
    2. Significant traffic disruption is detected on the user's planned route.
    3. System calculates alternative routes and the impact of the disruption on travel time.
    4. System sends an alert to the user's device with updated travel time and alternative route suggestions.
    5. User receives the alert and chooses whether to follow the suggested alternative routes.
    6. System updates the ride details according to the user's choice and navigates accordingly.
- Entry Conditions: User is en route in a ride.
- Exit Conditions: User is informed about traffic conditions and decides on the course of action.
- Exceptions: If traffic data is unavailable due to a system outage, the system uses historical data to make predictions.
- Quality Requirements: Traffic data should be refreshed every 5 minutes to ensure accuracy and timeliness of alerts.

Use Case: Service Incident Response

- Participating Actors: User, System, Incident Manager
- Flow of Events:
    1. A service incident occurs (e.g., accident, severe traffic, system malfunction).
    2. System detects the incident and categorizes it based on severity and urgency.
    3. Incident Manager is notified with all relevant details.
    4. System communicates the incident and expected impact to the affected user.
    5. Incident Manager coordinates with relevant teams to manage and resolve the incident.
    6. System updates the user on resolution progress and final outcome.
    7. After resolution, a follow-up is conducted with the user to ensure satisfaction and gather feedback.
- Entry Conditions: An incident occurs affecting the user's service.
- Exit Conditions: User is informed about resolution steps and the incident is resolved.
- Exceptions: If the incident cannot be resolved in a timely manner, alternative arrangements or compensations are offered.
- Quality Requirements: Incident notifications must be precise and delivered within 3 minutes of incident detection.

Use Case: System Service Monitoring

- Participating Actors: System Administrators, System
- Flow of Events:
    1. System continuously monitors the operational status and performance metrics of all active services.
    2. Anomalies, such as delays, high cancellation rates, or negative feedback, are automatically detected.
    3. System alerts System Administrators with detailed anomaly reports and possible causes.
    4. System Administrators analyze the data, diagnose issues, and initiate corrective measures.
    5. System tracks the implementation and effectiveness of these measures.
    6. Periodic reports are generated for review and future planning.
- Entry Conditions: System is operational.
- Exit Conditions: Administrators receive notifications about service issues and take necessary actions.
- Exceptions: If the monitoring system fails, manual checks are initiated and system reboot procedures are followed.
- Quality Requirements: System must detect and report anomalies within 10 minutes of occurrence.

Scenarios

Pre-Ride Information Communication:

Participating Actors: User, System

- User opens app and navigates to "Book Ride" option
- User enters pickup location and destination
- User selects preferred pickup time and vehicle type
- System processes request and checks vehicle availability
- System confirms booking details and assigns vehicle
- System displays car model, plate number, pickup location/time
- System sends confirmation notification to user

User Delay Notification:

Participating Actors: User, System

- User accesses current bookings in app
- User selects "Modify Booking" for a ride
- User enters new desired pickup time
- System checks availability for new time
- System confirms change and updates booking
- System sends updated pickup time to user

Incident Reporting:

Participating Actors: User, System, Incident Manager

- System monitors ongoing rides for incidents
- Incident occurs (accident, breakdown, etc.)
- System detects incident via sensors report
- System logs incident and categorizes severity
- System notifies Incident Manager
- Incident Manager evaluates and initiates response
- System informs user about incident, impact, resolution steps
- System offers alternative transportation if needed

Charging Station Occupancy:

Participating Actors: User, System

- User opens app and navigates to charging station locator
- User inputs location for charging
- System accesses real-time data from charging stations
- System analyzes availability and status of each station
- System displays list of available stations with occupancy levels
- User selects station and optionally reserves charging slot
- System confirms reservation and provides navigation

Customer Feedback Prompt:

Participating Actors: User, System

- Ride ends, system triggers feedback sequence
- System prompts user with feedback form on app
- User rates ride aspects and submits comments
- System collects and stores feedback
- System sends thank you message to user
- Feedback used for quality control and analytics

Weather Alert:

Participating Actors: User, System

- System monitors weather for active/scheduled rides
- Adverse conditions forecasted for user's location
- System assesses potential impact on user's ride
- System sends weather alert to user's app/communication channels
- User receives alert with conditions and safety advice
- System offers to adjust ride timing/routes if feasible
- User decides to proceed, reschedule or cancel based on alert

Lost Items:

Participating Actors: User, System

- User realizes item left in vehicle after ride
- User navigates to 'Lost Items' section in app
- User fills out lost item report with details
- System logs report and initiates search
- If item found, system notifies user and arranges return
- If not found, system registers as lost and continues search
- System offers compensation if item not recovered

Real-Time Tracking:
Participating Actors: User, System
- User requests real-time vehicle tracking via app
- System accesses vehicle's GPS data
- User receives map view showing vehicle movement
- System sends notifications at 10, 5, 1 minute(s) away
- System sends "Arrived" notification on vehicle arrival
- User meets vehicle/transportation at pickup

High-Demand Alert:
Participating Actors: User, System
- System monitors vehicle availability and booking rates
- System identifies high-demand conditions based on analytics
- System calculates impact on wait times and availability
- System sends high-demand alerts to affected users
- Users receive alert suggesting optimal booking times/premium options
- Users decide to book immediately, wait, or choose another option
- System continues monitoring and adjusting predictions/notifications

Service Delay Notification:
Participating Actors: User, System
- Unexpected delay affects scheduled service
- System detects delay via monitoring report

- System calculates new estimated arrival time
- System updates booking and notifies user of delay
- User receives update and can wait or cancel for refund
- System offers alternatives if delay exceeds threshold

Surge Pricing Notification:
Participating Actors: User, System
- System monitors demand, traffic, data to calculate fares
- Spike in demand triggers surge pricing algorithms
- System calculates new rates and prepares notifications
- Before booking, system displays surge pricing notification
- User receives surge details, reasons, options to wait
- User makes informed decision based on provided info

Cancellation of Pre-Set Transportation:
Participating Actors: User, System
- System identifies issues requiring cancellation
- System assesses impact and identifies affected users
- System notifies affected users about cancellation and reasons
- System offers alternative options and rebooking assistance
- User receives notification and selects alternative or refund
- System processes user's choice and confirms arrangements

Recommendations for User:
Participating Actors: User, System
- User completes a ride
- System analyzes user's travel patterns and preferences
- System uses algorithms to generate personalized recommendations
- System presents recommendations to user in app
- User reviews and can directly book suggested service
- System monitors user interaction to refine future suggestions

Traffic Alert:

Participating Actors: User, System

- System monitors real-time traffic on user's routes
- Significant disruption detected on user's planned route
- System calculates alternative routes and impact
- System sends alert to user with updated times, alternatives
- User receives alert and chooses suggested alternatives
- System updates ride details per user's choice

Service Incident Response:

Participating Actors: User, System, Incident Manager

- Service incident occurs (accident, system malfunction, etc.)
- System detects and categorizes incident by severity/urgency
- Incident Manager notified with incident details
- System communicates incident and expected impact to user
- Incident Manager coordinates resolution efforts
- System updates user on resolution progress and outcome
- Follow-up conducted with user for satisfaction/feedback

Exception Scenarios

User Registration Exception:

- The user enters invalid or incomplete information during the registration process.
- The system detects a duplicate registration attempt with an existing user's credentials.

Ride Reservation Exception:

- The user attempts to book a ride outside of the service's operational hours or service area.
- The system encounters an error while retrieving the list of available vehicles from the ATP.

Transport Selection and Trip Confirmation Exception:

- The user's preferred transport type is not available for the requested pickup time and location.
- The system fails to propose an alternative pickup time due to a lack of available vehicles.

Payment Processing and Confirmation Exception:

- The user's payment method is declined or encounters an error during the payment processing.
- The system fails to generate a confirmation number, PIN code, or invoice receipt due to a technical issue.

Ride Completion and Feedback Exception:

- The user attempts to provide feedback for a ride that was not completed or does not exist in the system's records.
- The feedback submission fails due to a network or system error.

Subscription Upgrade Exception:

- The user's subscription is not eligible for an upgrade due to account restrictions or outstanding payments.
- The system encounters an error while updating the user's subscription tier and associated benefits.

Subscription Cancellation Exception:

- The user attempts to cancel a subscription that has already been terminated or does not exist.
- The system fails to stop recurring charges or refund any outstanding payments after subscription cancellation.

User Information Update Exception:

- The user attempts to update information with invalid or restricted data.
- The system encounters a data validation or integrity issue while updating the user's profile information.

Data Backup and Recovery Exception:

- The backup process fails due to a hardware, software, or network issue.
- The system is unable to restore data from the backup due to corruption or compatibility issues.

User Access Control Exception:

- The system encounters an error while granting or revoking user permissions and access levels.
- An unauthorized user attempts to access restricted system resources or data.

Vehicle Dispatch Exception:

- The ATP is unable to assign a vehicle due to a shortage of available vehicles or technical issues.
- The assigned vehicle encounters an issue and cannot reach the user's pickup location.

Real-time Tracking Exception:

- The system fails to retrieve the vehicle's location data due to a GPS or network issue.
- The user's device or app encounters an error while displaying the real-time tracking information.

Route Adjustment Exception:

- The system is unable to recalculate the route or adjust the fare due to a technical issue.
- The user attempts to change the drop-off location outside of the service's operational area.

Incident Reporting Exception:

- The user's incident report contains incomplete or invalid information.
- The system fails to notify the appropriate authorities due to a communication or integration issue.

Lost Item Recovery Exception:

- The user provides inaccurate or incomplete information about the lost item.
- The system is unable to locate the lost item due to a lack of information or tracking capabilities.

Surge Pricing Notification Exception:

- The system fails to detect high demand conditions or accurately calculate the surge pricing rates.
- The user's device or app encounters an error while displaying the surge pricing notification.

Accessibility Request Exception:

- The system does not have any available vehicles that meet the user's accessibility requirements.
- The user's accessibility request contains invalid or incomplete information.

Emergency Assistance Exception:

- The system fails to initiate the emergency protocol or notify the appropriate authorities.
- The user's location or communication channel is unavailable or unreliable during the emergency situation.

Referral Program Exception:

- The system encounters an error while processing referral rewards or tracking referral activities.
- The user attempts to abuse or exploit the referral program in an unauthorized manner.

Promotional Offer Redemption Exception:

- The user provides an invalid or expired promotional code or offer.
- The system encounters an error while applying the discount or validating the promotional offer.

Payment Method Update Exception:

- The user attempts to update their payment method with an invalid or unsupported payment type.
- The system encounters an error while processing the payment method update request.

Ride History Exception:

- The system is unable to retrieve or display the user's ride history due to a data or system issue.
- The user's ride history contains incomplete or inaccurate information.

Vehicle Maintenance Exception:

- The system fails to detect or schedule necessary vehicle maintenance or repairs.
- The assigned vehicle is unavailable due to an unresolved maintenance issue.

Traffic Alert Exception:

- The system is unable to receive or process real-time traffic data from external sources.
- The traffic alert provided to the user is inaccurate or outdated.

Charging Station Occupancy Exception:

- The system fails to retrieve or update the occupancy status of charging stations for electric vehicles.
- The charging station occupancy information displayed to the user is incorrect.

Ride Cancellation Exception:

- The user attempts to cancel a ride that has already been completed or is in progress.
- The system encounters an error while processing the ride cancellation request.

User Feedback Review Exception:

- The system fails to aggregate or analyze user feedback data for service improvements.
- The user feedback review process encounters data integrity or security issues.

Service Outage Exception:

- The system experiences a partial or complete service outage due to technical failures or maintenance.
- Users are unable to access the ride-sharing service or receive notifications during the outage.

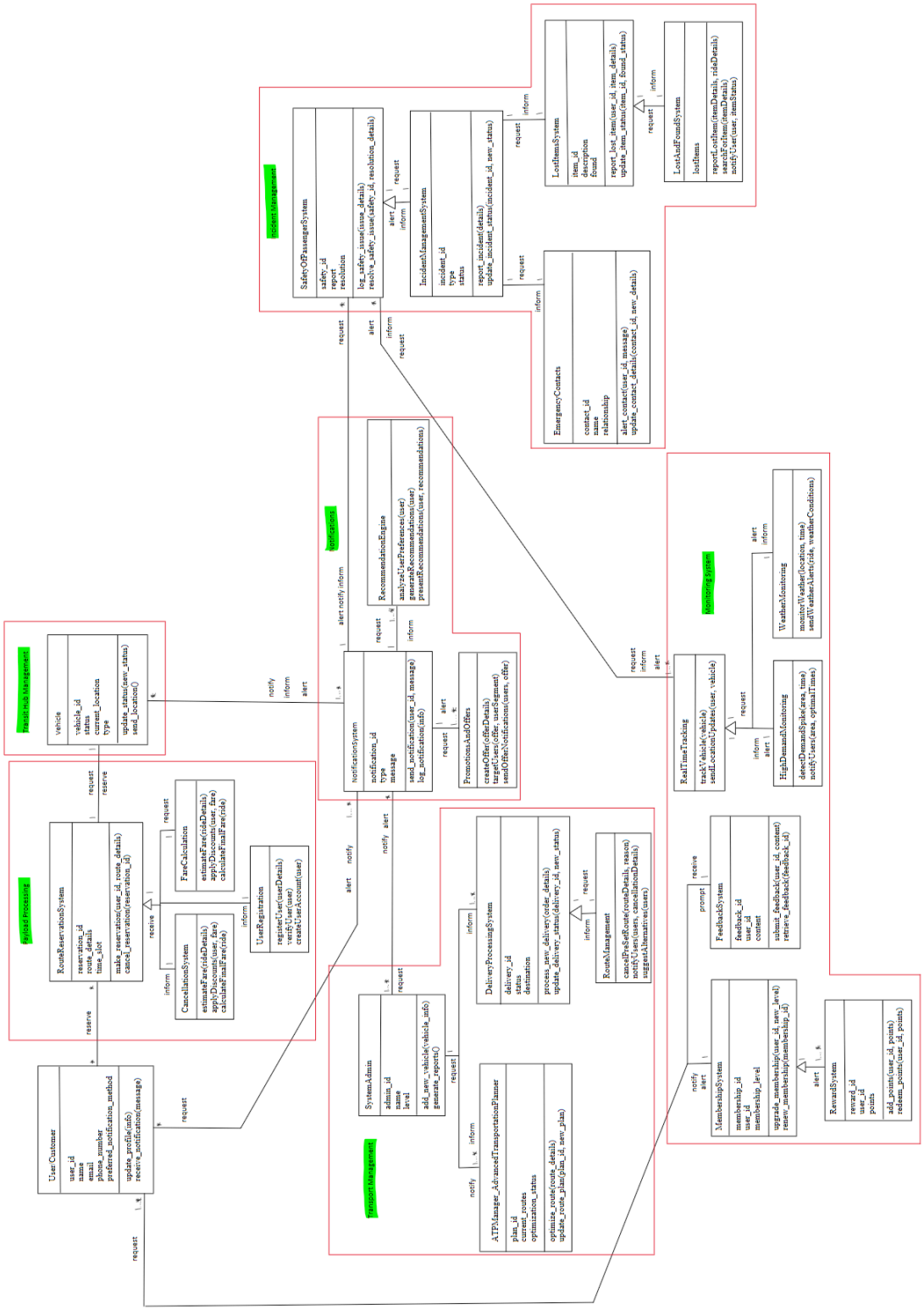
Vehicle Breakdown Exception:

- A vehicle experiences a breakdown or mechanical issue during a ride.
- The system fails to reassign another available vehicle or notify the user about the situation.

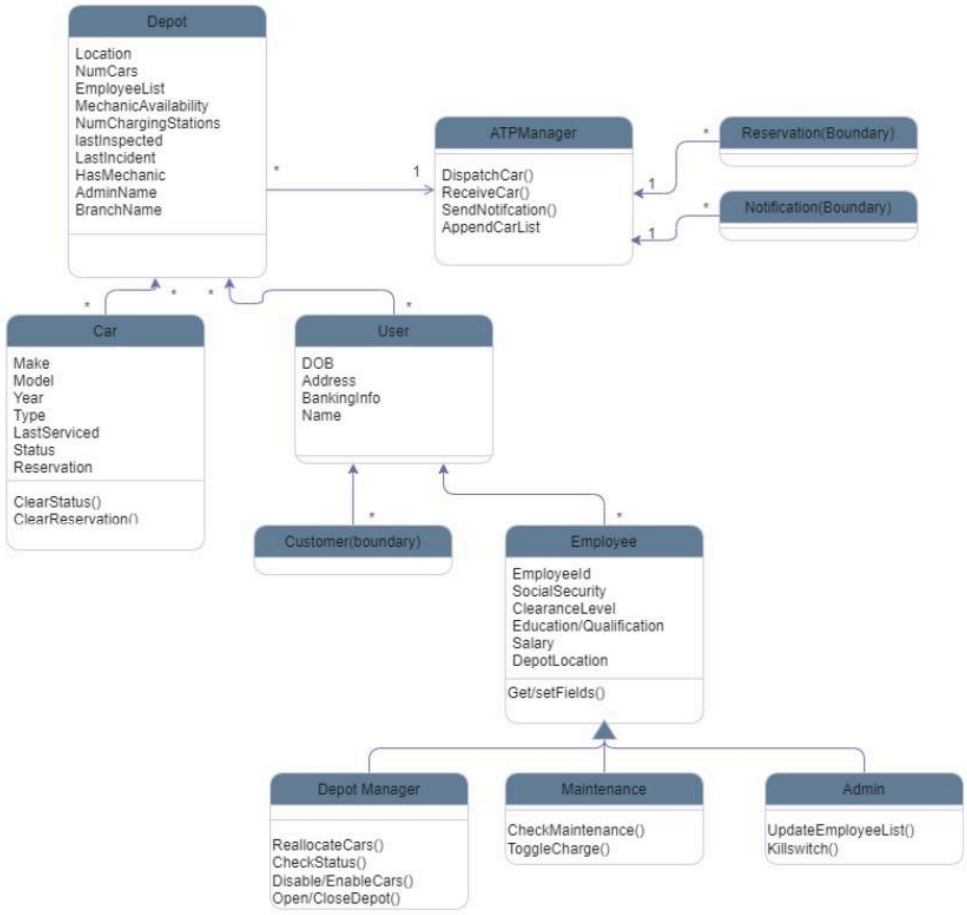
Delivery Exception:

- The system encounters an error while processing a delivery request or tracking a delivery item.
- The delivery process encounters delays or issues due to external factors, such as weather or traffic conditions.

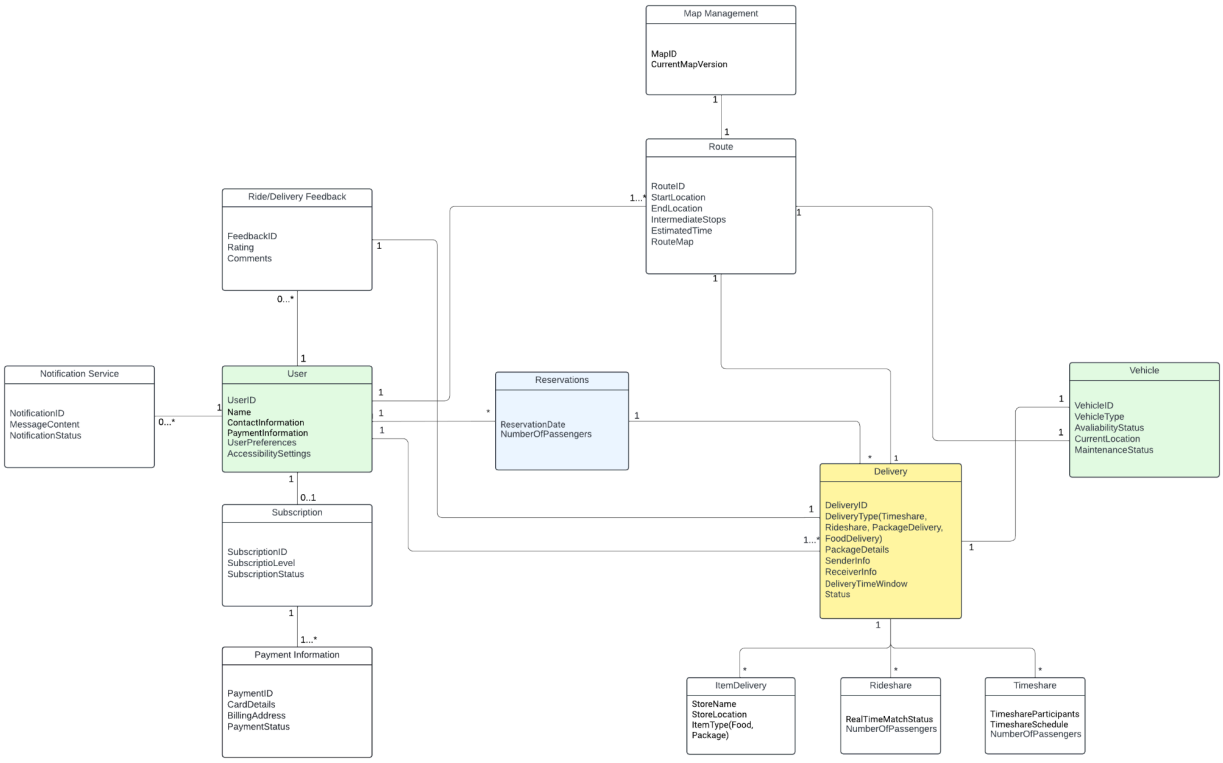
Class Model/Sub-System Decomposition
Team Four (Our Team)



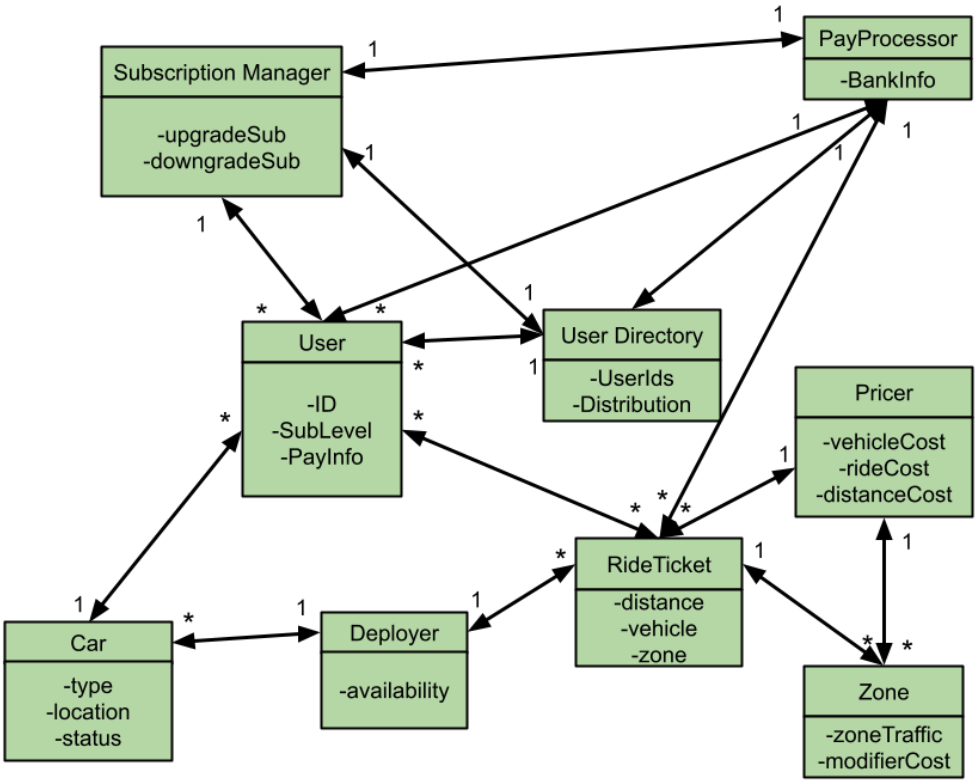
Team One


Depot
Location
NumCars
EmployeeList
MechanicAvailability
NumChargingStations
lastInspected
LastIncident
HasMechanic
AdminName
BranchName
ATPManager
DispatchCar()
ReceiveCar()
SendNotifcation()
AppendCarList
Reservation(Boundary)
Notification(Boundary)
Car
Make
Model
Year
Type
LastServiced
Status
Reservation
ClearStatus()
ClearReservation()
User
DOB
Address
BankingInfo
Name
Customer(boundary)
Employee
EmployeeId
SocialSecurity
ClearanceLevel
Education/Qualification
Salary
DepotLocation
Get/setFields()
Depot Manager
ReallocateCars()
CheckStatus()
Disable/EnableCars()
Open/CloseDepot()
Maintenance
CheckMaintenance()
ToggleCharge()
Admin
UpdateEmployeeList()
Killswitch()

Team Two

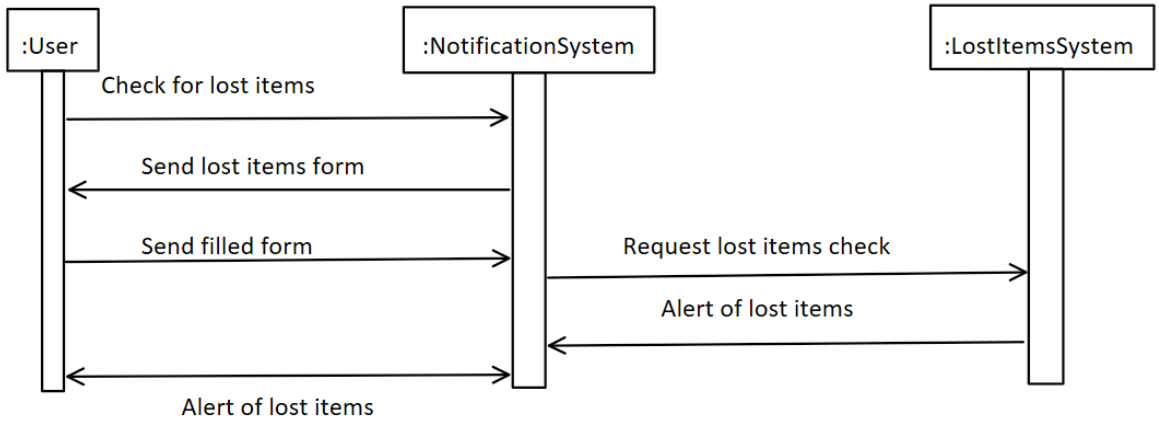

Map Management
MapID
CurrentMapVersion
Route
RouteID
StartLocation
EndLocation
IntermediateStops
EstimatedTime
RouteMap
Ride/Delivery Feedback
FeedbackID
Rating
Comments
Notification Service
NotificationID
MessageContent
NotificationStatus
User
UserID
Name
ContactInformation
PaymentInformation
UserPreferences
AccessibilitySettings
Reservations
ReservationDate
NumberOfPassengers
Vehicle
VehicleID
VehicleType
AvaliabilityStatus
CurrentLocation
MaintenanceStatus
Subscription
SubscriptionID
SubscriptioLevel
SubscriptionStatus
Delivery
DeliveryID
DeliveryType(Timeshare, Rideshare, PackageDelivery, FoodDelivery)
PackageDetails
SenderInfo
ReceiverInfo
DeliveryTimeWindow
Status
Payment Information
PaymentID
CardDetails
BillingAddress
PaymentStatus
ItemDelivery
StoreName
StoreLocation
ItemType(Food, Package)
Rideshare
RealTimeMatchStatus
NumberOfPassengers
Timeshare
TimeshareParticipants
TimeshareSchedule
NumberOfPassengers

Team Three

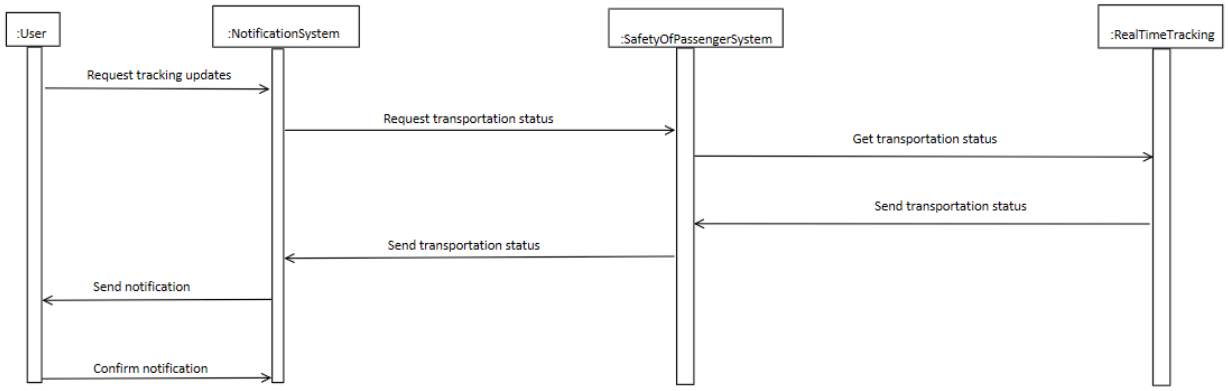

Subscription Manager
-upgradeSub
-downgradeSub
PayProcessor
-BankInfo
User
-ID
-SubLevel
-PayInfo
User Directory
-UserIds
-Distribution
Pricer
-vehicleCost
-rideCost
-distanceCost
RideTicket
-distance
-vehicle
-zone
Car
-type
-location
-status
Deployer
-availability
Zone
-zoneTraffic
-modifierCost

Sequence Diagrams

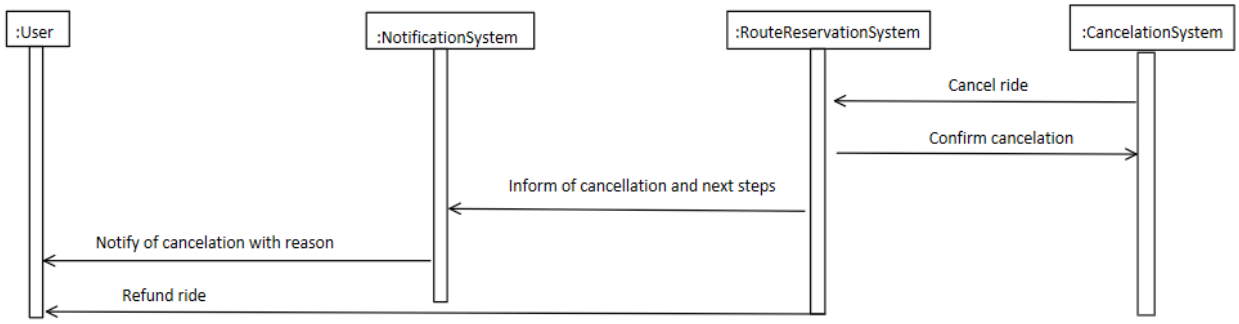
LostItems



RealTimeTracking

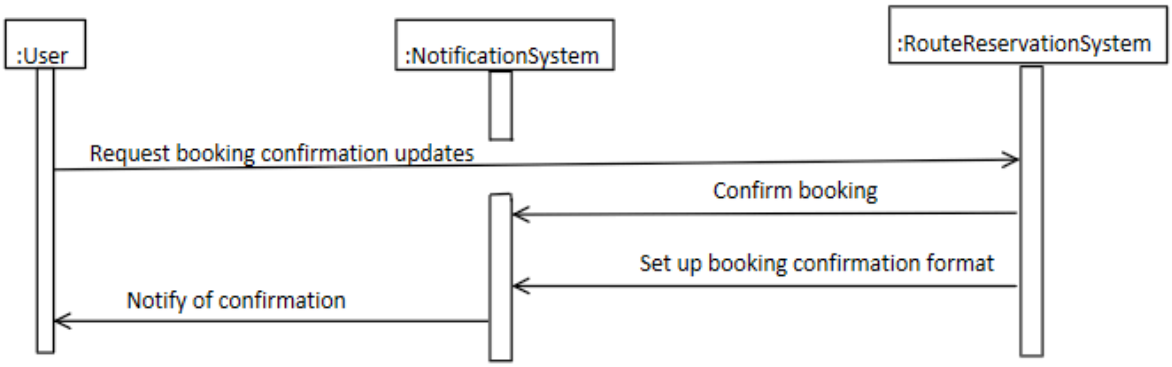


CanceledTransportation

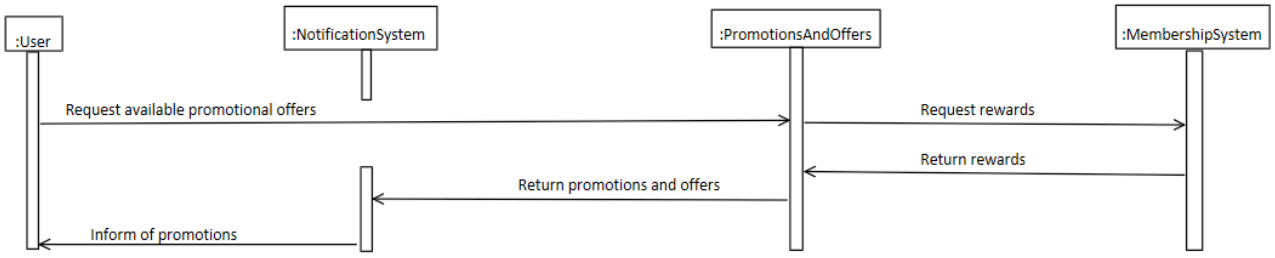


VehicleArrival

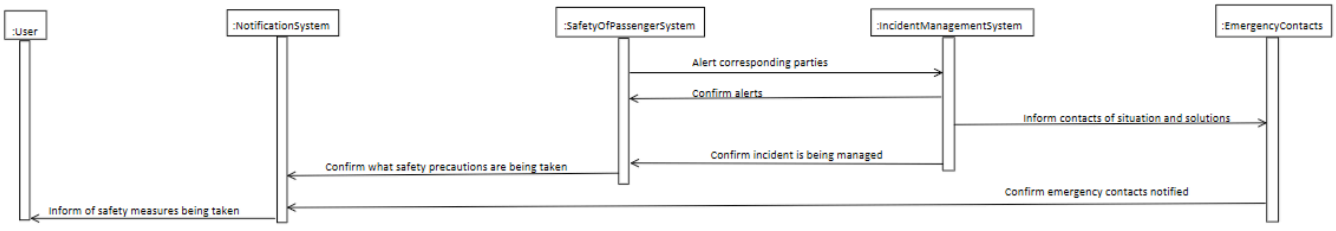
BookingConfirmation



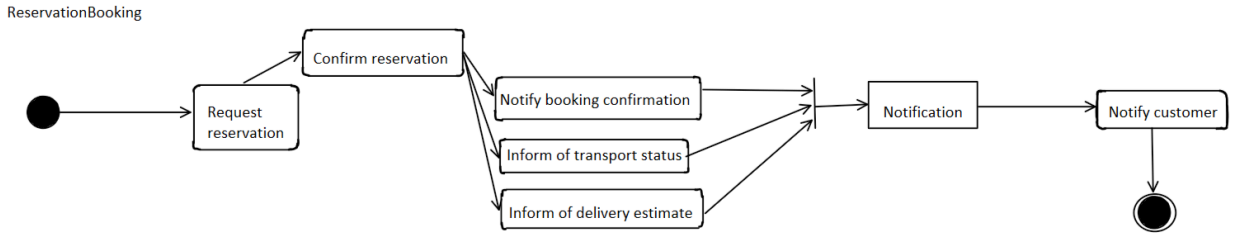
PromotionalOffers



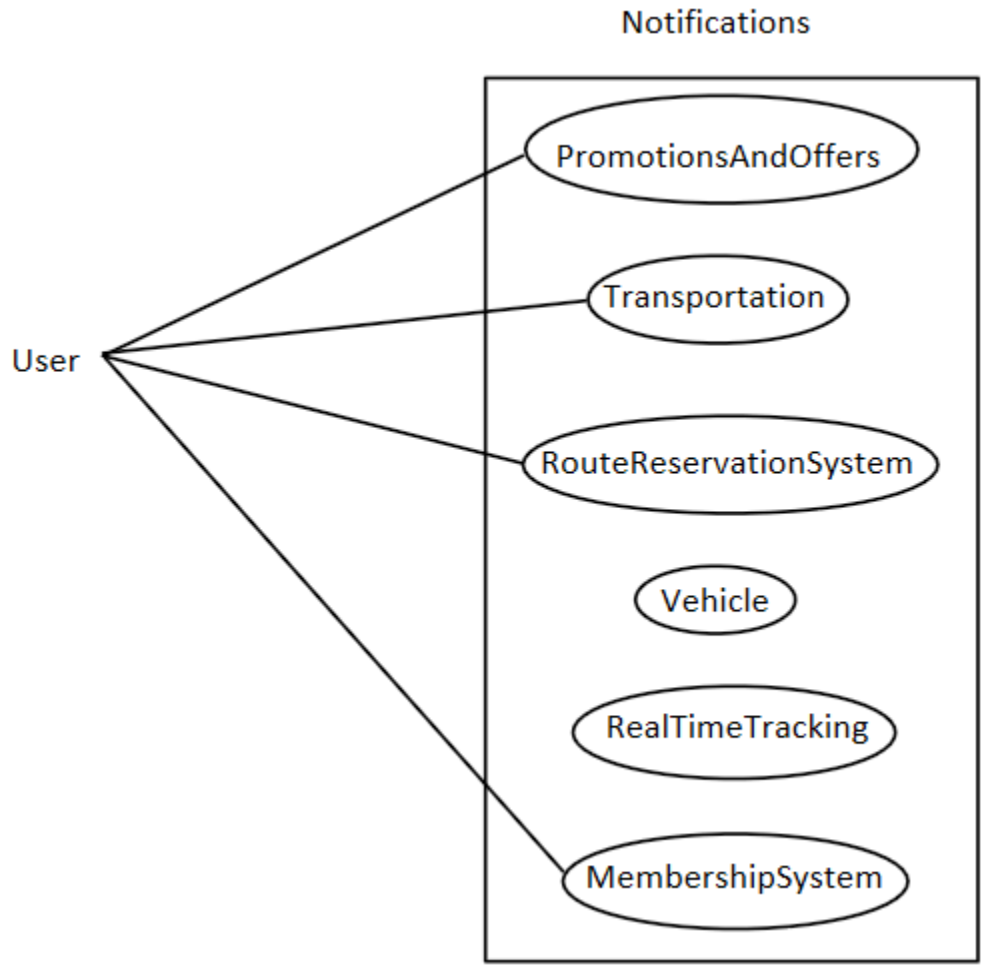
EmergencyAssistance



Activity Diagram



User Interface (Diagram)



Analysis Objects Identification

Entity Objects:

1. **User**: Stores details about users, such as name, email, phone, and payment information.
2. **Vehicle**: Represents details about the vehicles such as type, location, and availability.
3. **Reservation**: Holds details about ride reservations, including pickup location, time, vehicle, and user.
4. **Payment**: Manages payment information, status of transactions, and payment history.
5. **Feedback**: Contains user feedback data, ratings, and comments post-ride.
6. **RideSummary**: Stores information about completed rides, including times, distances, and costs.

Boundary Objects:

1. **UserInterface**: Handles user interactions such as registration, login, feedback submission, and display of available vehicles.
2. **PaymentGateway**: Manages the processing of payments and interacts with external payment systems.
3. **EmailSystem**: Sends out confirmation emails, promotional offers, and feedback requests.
4. **NotificationSystem**: Sends real-time updates during the ride, and other notifications related to the ride status.

Control Objects:

1. **UserManager**: Controls the process of user registration and the gathering of user details.
2. **ReservationManager**: Handles ride reservations, including vehicle availability checks, reservation details storage, and scheduling.
3. **PaymentController**: Manages the logic of payment approvals, cancellations due to non-payment, and transaction finalization.
4. **VehicleDispatcher**: Manages assigning vehicles to user reservations, updating vehicle statuses, and logging route details.
5. **FeedbackManager**: Controls the feedback collection process, including prompting for feedback and handling responses.
6. **PromotionManager**: Decides which promotional offers to send to users based on eligibility criteria.

Deployment//Pseudocode

```
// User Registration
IF new_user:
        DISPLAY registration_discount_offer
        GET user_details (name, email, phone, payment_info)
        VERIFY user_details
        REGISTER user

// Ride Reservation
GET pickup_location, desired_date, desired_time
VALIDATE pickup_location
CALL create_available_vehicles_list(desired_date, desired_time)
SORT available_vehicles_list BY vehicle_type, proximity_to_pickup
GOTO reservation_form

// Transport Selection and Trip Confirmation
DISPLAY available_vehicle_types
GET selected_vehicle_type
FILTER available_vehicles_list BY selected_vehicle_type
IF closest_vehicle.eta > desired_time + 30mins:
        proposed_time = closest_vehicle.eta
        DISPLAY proposed_time
        GET user_response
        IF user_response == accepted:
        STORE reservation_details
        CREATE delivery_order
        COMPUTE initial_price
        GOTO payment_processing
        ELSE:
        FILTER available_vehicles_list BY desired_time
        IF no_vehicles_available:
        DISPLAY upgrade_offer
        GET user_response
        IF user_response == accepted:
                COMPUTE upgraded_price
                GOTO payment_processing
        ELSE:
                CANCEL reservation
        ELSE:
```

```
        STORE reservation_details
        CREATE delivery_order
        COMPUTE initial_price
        GOTO payment_processing

// Payment Processing and Confirmation
GET payment_approval
VALIDATE payment_details
IF payment_approved:
        LOCK assigned_vehicle
        SET route_details IN dispatcher_logs
        GENERATE confirmation_number, pin_code, invoice
        SEND confirmation_email
        SETUP ride_notifications
ELSE:
        CANCEL reservation

// Real-Time Tracking
WHILE ride_in_progress:
        GET vehicle_location
        COMPUTE eta
        SEND vehicle_location_update
        IF incident_occurred:
        TRACK new_vehicle_location
        SEND new_vehicle_location_updates

// Customer Feedback Prompt
AFTER ride_completed:
        PROMPT user_for_feedback
        GET feedback_ratings, comments
        IF positive_feedback:
        SEND discount_offer
        ELSE:
        INVESTIGATE negative_feedback

// Promotional Offers
IF user_eligible:
        SEND promotional_offer

// Ride Completion
```

```
SEND ride_summary
DISPLAY goodbye_message

// Incident Handling
IF incident_detected:
        INITIATE incident_response_protocol
        NOTIFY emergency_services IF necessary
        ENSURE passenger_safety
        DOCUMENT incident_details
        COMMUNICATE with_car
        ASSESS potential_injuries_or_damages
        INITIATE insurance_claim_process IF necessary

// Incident Reporting Notification
IF incident_detected:
        LOG incident_details
        NOTIFY user WITH incident_details, expected_delay
        INITIATE incident_resolution_protocol

// Incident Resolution
IF incident_occurred:
        ASSIGN new_car
        SEND incident_resolution_details
        SEND new_vehicle_details

// Incident Analysis and Prevention
AFTER incident_resolved:
        ANALYZE root_cause
        IMPLEMENT preventive_measures
        REPORT incident_to_authorities IF necessary


// Vehicle Maintenance
SCHEDULE regular_vehicle_inspections
IF vehicle_requires_maintenance:
        NOTIFY vehicle_maintenance
        SCHEDULE vehicle_maintenance
        REMOVE vehicle_from_active_fleet UNTIL maintenance_completed

// Customer Support
```

```
PROVIDE customer_support_channels (phone, email, chat)
MONITOR customer_inquiries
RESPOND to_customer_inquiries
ESCALATE complex_issues_to_specialized_team

// Data Analytics
COLLECT ride_data, user_data, incident_data
ANALYZE data_for_insights
IDENTIFY areas_for_improvement
OPTIMIZE operations_based_on_insights
```

Implementation Example

```
class IncidentManagement:
    def handle_incident(self):
        # Incident Handling
        if incident_detected:
            self.initiate_incident_response_protocol()
            self.notify_emergency_services_if_necessary()
            self.ensure_passenger_safety()
            self.document_incident_details()
            self.communicate_with_car()
            self.assess_potential_injuries_or_damages()
            self.initiate_insurance_claim_process_if_necessary()

    def report_incident(self):
        # Incident Reporting Notification
        if incident_detected:
            self.log_incident_details()
            self.notify_user_with_incident_details_and_expected_delay()
            self.initiate_incident_resolution_protocol()

    def real_time_tracking(self):
        # Real-Time Tracking
        while ride_in_progress:
            vehicle_location = self.get_vehicle_location()
            eta = self.compute_eta(vehicle_location)
            self.send_vehicle_location_update(eta)
            if incident_occurred:
                new_vehicle_location = self.track_new_vehicle_location()
                self.send_new_vehicle_location_updates(new_vehicle_location)

    def resolve_incident(self):
        # Incident Resolution
        if incident_occurred:
            new_car = self.assign_new_car()
            incident_resolution_details = self.get_incident_resolution_details()
            new_vehicle_details = self.get_new_vehicle_details(new_car)
            self.send_incident_resolution_details(incident_resolution_details)
            self.send_new_vehicle_details(new_vehicle_details)
```

```
 def incident_analysis_and_prevention(self):
        # Incident Analysis and Prevention
        if incident_resolved:
            root_cause = self.analyze_root_cause()
            preventive_measures = self.implement_preventive_measures(root_cause)
            self.report_incident_to_authorities_if_necessary(root_cause)

class UserManagement:
    def register_user(self):
        # User Registration
        if new_user:
            registration_discount_offer = self.display_registration_discount_offer()
            user_details = self.get_user_details()
            self.register_user(user_details)
            self.goto_reservation_form()
        else:
            self.goto_reservation_form()

class RideReservation:
    def reserve_ride(self):
        # Ride Reservation
        pickup_location, desired_date, desired_time = self.get_pickup_location_and_desired_time()
        available_vehicles_list = self.create_available_vehicles_list(desired_date, desired_time)
        sorted_available_vehicles_list = self.sort_available_vehicles_list(available_vehicles_list)
```

```
 # Transport Selection and Trip Confirmation
        available_vehicle_types =
self.display_available_vehicle_types(sorted_available_vehicles_list)
        selected_vehicle_type = self.get_selected_vehicle_type(available_vehicle_types)
        filtered_vehicles_list = self.filter_available_vehicles_list(sorted_available_vehicles_list,
selected_vehicle_type)

        if self.closest_vehicle_eta_too_late(filtered_vehicles_list, desired_time):
            proposed_time = self.get_proposed_time(filtered_vehicles_list)
            user_response = self.display_proposed_time_and_get_user_response(proposed_time)
            if user_response == 'accepted':
                reservation_details = self.store_reservation_details(proposed_time,
selected_vehicle_type)
                delivery_order = self.create_delivery_order(reservation_details)
                initial_price = self.compute_initial_price(reservation_details)
                self.goto_payment_processing(initial_price)
            else:
                self.cancel_reservation()
        else:
            if not filtered_vehicles_list:
                upgrade_offer = self.display_upgrade_offer()
                user_response = self.get_user_response(upgrade_offer)
                if user_response == 'accepted':
                    upgraded_price = self.compute_upgraded_price(selected_vehicle_type)
                    self.goto_payment_processing(upgraded_price)
                else:
                    self.cancel_reservation()
            else:
                reservation_details = self.store_reservation_details(desired_time,
selected_vehicle_type)
                delivery_order = self.create_delivery_order(reservation_details)
                initial_price = self.compute_initial_price(reservation_details)
                self.goto_payment_processing(initial_price)
```

```
def goto_payment_processing(self, price):
        # Payment Processing and Confirmation
        payment_approval = self.get_payment_approval(price)
        if payment_approval:
            assigned_vehicle = self.lock_assigned_vehicle(delivery_order)
            route_details = self.set_route_details_in_dispatcher_logs(delivery_order,
assigned_vehicle)
            confirmation_details = self.generate_confirmation_details(delivery_order,
assigned_vehicle)
            self.send_confirmation_email(confirmation_details)
            self.setup_ride_notifications(confirmation_details)
        else:
            self.cancel_reservation()

class RideTracking:
    def real_time_tracking(self):
        # Real-Time Tracking
        while ride_in_progress:
            vehicle_location = self.get_vehicle_location()
            eta = self.compute_eta(vehicle_location)
            self.send_vehicle_location_update(eta)

class CustomerFeedback:
    def prompt_for_feedback(self):
        # Customer Feedback Prompt
        if ride_completed:
            feedback_ratings, comments = self.get_feedback_ratings_and_comments()
            if self.is_positive_feedback(feedback_ratings):
                discount_offer = self.send_discount_offer()

class PromotionalOffers:
    def send_promotional_offer(self):
        # Promotional Offers
        if user_eligible:
            promotional_offer = self.get_promotional_offer()
            self.send_promotional_offer(promotional_offer)
```

```
class RideCompletion:
    def complete_ride(self):
        # Ride Completion
        ride_summary = self.get_ride_summary()
        self.send_ride_summary(ride_summary)
        self.display_goodbye_message()
```

Data Dictionary

1. User: The main actor who uses the ride-sharing service to request rides, make reservations, schedule deliveries, or utilize the timeshare option.
2. Additional Rider(s): Secondary actors who may accompany the primary user during a ride.
3. Vehicle: The transportation mode assigned to fulfill ride requests.
4. Law Enforcement: Actors representing law enforcement agencies that may be involved in certain scenarios, such as emergencies or traffic incidents.
5. Hospitals: Actors representing healthcare facilities that may be involved in emergency scenarios.
6. System Admins: Administrators responsible for managing and maintaining the ride-sharing service system.
7. ATPManager (Availability to Promise Manager): The system component responsible for managing vehicle availability and dispatching ride requests.
8. UI (User Interface): The interface through which users interact with the ride-sharing service.
9. Route Reservation System: The system component responsible for handling ride reservations and route planning.
10. Delivery Processing System: The system component responsible for managing the delivery of items through the ride-sharing service.
11. Membership/Pricing/Billing System: The system component responsible for handling user subscriptions, pricing, billing, and payment processing.
12. Payload/Transport and Notification Processing System: The system component responsible for managing notifications related to ride requests, vehicle dispatching, and other transportation-related events.
13. Sender: The actor who initiates a delivery request through the ride-sharing service.
14. Receiver: The actor who receives the delivery through the ride-sharing service.
15. Emergency Contacts: Actors representing individuals or entities that may be notified in case of emergencies or deviations from the expected ride route or timing.
16. Feedback System: The system component responsible for gathering and processing user feedback and ratings.

17. Service Monitoring System: The system component responsible for monitoring the overall performance and service quality of the ride-sharing service.
18. Vehicle Maintenance System: The system component responsible for managing vehicle maintenance and repairs.
19. Surge Pricing System: The system component responsible for adjusting pricing based on demand levels.
20. Accessibility Services: The system component responsible for handling requests for accessible vehicles or accommodating special needs.
21. Emergency Assistance System: The system component responsible for providing emergency assistance to users during rides.
22. Traffic Monitoring System: The system component responsible for monitoring and providing updates on traffic conditions.
23. Charging Station System: The system component responsible for managing charging station availability and occupancy for electric vehicles.
24. Lost and Found System: The system component responsible for handling lost and found items left in vehicles.
25. Recommendation System: The system component responsible for providing personalized recommendations to users based on their usage patterns and preferences.