

**Software Requirements Specification**

**for**

Uber Application

**Version 1.0 approved**

**Prepared by Group 1**

**Group 1**

**12/11/2023**

**Table of Contents**

**Table of Contents ii**

**Revision History ii**

**1. Introduction 1**

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Project Scope 1

1.5 References 1

**2. Overall Description 2**

2.1 Product Perspective 2

2.2 Product Features 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

2.7 Assumptions and Dependencies 3

**3. System Features 3**

3.1 System Feature 1 3

3.2 System Feature 2 (and so on) 4

**4. External Interface Requirements 4**

4.1 User Interfaces 4

4.2 Hardware Interfaces 4

4.3 Software Interfaces 4

4.4 Communications Interfaces 4

**5. Other Nonfunctional Requirements 5**

5.1 Performance Requirements 5

5.2 Safety Requirements 5

5.3 Security Requirements 5

5.4 Software Quality Attributes 5

**6. Other Requirements 5**

**Appendix A: Glossary 5**

**Appendix B: Analysis Models 6**

**Appendix C: Issues List 6**

**Revision History**

| **Name** | **Date** | **Reason For Changes** | **Version** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

# **Introduction**

## **Purpose**

The Uber Application SRS covers the software requirements for the entire Uber ridesharing platform. This includes but is not limited to rider and driver functionalities, ride management, location services, and payment processing.

## **Document Conventions**

Every statement is their own priority in explaining the software. We used bold lettering to title sections and specific words for the definition. The bold statements are mainly used so the SRS is easier to read. A lot of the sections are bulleted for easy understanding.

## **Intended Audience and Reading Suggestions**

The intended audience is administration, users, testers, and management. They all need to read this to fully understand the application and how it works. To read the SRS, start with the overview and the purpose, then end with the software and how it works.

## **Project Scope**

The primary objectives of the Uber Application are as follows:

Efficient Transportation: Facilitate quick and convenient transportation for riders by connecting them with available drivers in real-time.

User-Friendly Interface: Offer a user-friendly and intuitive interface for both riders and drivers to enhance the overall experience.

Driver Management: Streamline driver operations, including ride acceptance, navigation, and fare collection, to optimize their efficiency.

Safety and Security: Prioritize user safety by implementing features such as ride tracking, driver identification, and secure payment processing.

**Benefits**

The Uber Application aims to deliver several benefits to its users:

Convenience: Riders can easily request a ride with a few taps on their smartphones, eliminating the need for traditional taxi hailing.

Income Opportunities: The platform provides drivers with flexible earning opportunities by allowing them to leverage their vehicles for ridesharing.

Real-Time Tracking: Both riders and drivers benefit from real-time tracking, ensuring transparency and safety during the entire ride.

## **References**

[*https://github.com/topics/ridesharing*](https://github.com/topics/ridesharing)

[*https://www.transportation.gov/regulations/guidance*](https://www.transportation.gov/regulations/guidance)

[*https://www.fmcsa.dot.gov/regulations*](https://www.fmcsa.dot.gov/regulations)

[*https://davisdesigninteractive.medium.com/pool-a-ux-case-study-37a61cfb39d7*](https://davisdesigninteractive.medium.com/pool-a-ux-case-study-37a61cfb39d7)

[*https://pathwayscommission.bsg.ox.ac.uk/sites/default/files/2019-09/ride-sharing\_platforms\_in\_developing\_countries.pdf*](https://pathwayscommission.bsg.ox.ac.uk/sites/default/files/2019-09/ride-sharing_platforms_in_developing_countries.pdf)

[*https://businessmodelanalyst.com/uber-mission-and-vision-statement/*](https://businessmodelanalyst.com/uber-mission-and-vision-statement/)

[*https://github.com/AMZ-Driverless/fsd-resources*](https://github.com/AMZ-Driverless/fsd-resources)

[*https://www.pubnub.com/blog/build-android-on-demand-rideshare-app-uber-lyft/*](https://www.pubnub.com/blog/build-android-on-demand-rideshare-app-uber-lyft/)

<https://krazytech.com/projects/sample-software-requirements-specificationsrs-report-airline-database>

# **Overall Description**

## **Product Perspective**

The Uber Application is a revolutionary addition to the ridesharing industry, aiming to transform traditional transportation methods. This application is a standalone product designed to connect riders with nearby drivers through a digital platform, providing a modern and convenient alternative to traditional taxi services.

**Relationship with Existing Systems**

The Uber Application is not a replacement for existing systems but introduces an innovative approach to transportation services. It stands as a self-contained product, leveraging modern technologies to enhance the overall ridesharing experience for both riders and drivers.

**Component of a Larger System**

While the Uber Application is designed as a self-contained product, it does interact with external systems and services to provide a comprehensive user experience. Major components and interactions include:

User Authentication System: Interacts with external authentication services to validate and secure user identities.

Payment Processing Gateway: Interfaces with external payment processors to facilitate secure and efficient transactions.

Geolocation Services: Utilizes external geolocation services to provide real-time tracking and navigation features.

**Subsystem Interconnections**

The major subsystems within the Uber Application include:

User Management: Responsible for user authentication, account creation, and profile management.

Ride Management: Handles the core functionality of ride requests, acceptance, tracking, and completion.

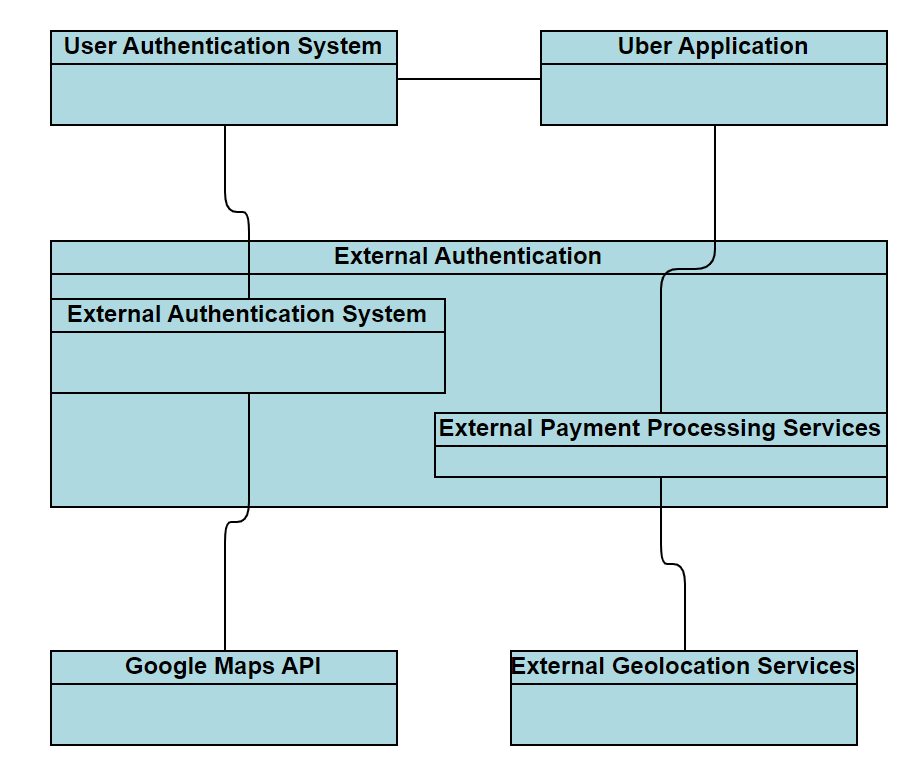
Payment Processing: Manages secure payment transactions between riders and drivers.

**External Interfaces**

The Uber Application interfaces with external components and services:

Google Maps API: Utilized for geolocation services, mapping, and navigation features.

Payment Gateways: Interfaces with external payment processors to securely handle financial transactions.



## **Product Features***>*

The Uber Application encompasses a range of features and functions that collectively enhance the ridesharing experience for both riders and drivers. These can be categorized into key functional areas:

**User Authentication and Management**

User Registration: Allow users to create accounts by providing necessary details.

User Login: Enable secure authentication for users to access the application.

Profile Management: Provide options to edit, view, and update user profiles.

**Ride Management**

Request Ride: Allow riders to request rides, specifying pickup and drop-off locations.

Accept Ride: Enable drivers to accept or decline ride requests.

Real-time Tracking: Provide live tracking of ride progress for both riders and drivers.

Complete Ride: Facilitate the completion of rides, including fare calculation.

**Location Services**

Geolocation Integration: Utilize GPS and mapping services for accurate location tracking.

Address Retrieval: Allow users to obtain addresses based on geolocation data.

**Payment Processing**

Secure Transactions: Facilitate secure and seamless financial transactions between riders and drivers.

Fare Calculation: Implement a transparent fare calculation mechanism based on ride distance and time.

**Rating and Feedback**

Driver Rating: Enable riders to provide ratings and feedback for completed rides.

User Reviews: Allow drivers to review and rate riders.

**Administrative Features**

Admin Dashboard: Provide administrators with tools to monitor and manage the platform.

User and Ride Analytics: Offer insights into user behavior and ride statistics.

**Accessibility and Usability**

Cross-Platform Compatibility: Ensure the application is accessible on various devices and platforms.

Intuitive User Interface: Design an easy-to-use interface for a seamless user experience.

**Security**

Data Encryption: Implement robust encryption mechanisms to secure user and transaction data.

Authentication Protocols: Employ secure authentication protocols to protect user accounts.

**External Integrations**

Google Maps Integration: Utilize external mapping services for navigation and location-related features.

Payment Gateway Integration: Interface with external payment processors for financial transactions.

**Notification System**

Real-time Updates: Provide timely notifications to users and drivers regarding ride status and updates.

**Support and Help**

User Support: Offer in-app support and help options for users facing issues.

## **User Classes and Characteristics**

The user classes consist of the rider, driver, ride, vehicle, location, and payment. The rider, driver, and ride are used for the app itself. They only interact with the app on a personal device. The vehicle is characterized for only the driver and taking the rider to different places. Locatio relates to all drivers, riders, and rides. The payment is given to the driver from the rider. To rate them of most importance, rider is first because without them the application wouldn’t even need to be made. Then it goes to the driver, vehicle, payment, location. They all relate to one another in how to make the system flow smoothly.

## **Operating Environment**

The Uber Application is designed to operate in a dynamic and diverse software environment. The following details outline the key components and characteristics of the environment in which the software will run:

**Hardware Platform**

Smartphones: The primary user interface for the Uber Application will be smartphones, running on various hardware configurations.

Server Infrastructure: Backend services will run on dedicated servers or cloud-based platforms to manage the application's data and business logic.

**Operating System**

iOS: The application is compatible with devices running the iOS operating system, including iPhones and iPads.

Android: The application is optimized for devices running the Android operating system.

**Software Dependencies**

Mobile Operating System Services: The application relies on core services provided by iOS and Android, including location services, notification systems, and secure storage.

Mapping and Navigation APIs: Integration with external services, such as Google Maps API, for accurate geolocation, mapping, and navigation features.

Payment Gateway Services: Interaction with third-party payment processors to handle secure financial transactions.

**Web Browser Compatibility (if applicable)**

Web-Based Interfaces: For administrative purposes, the application may include web-based interfaces accessible through standard web browsers.

Cross-Browser Compatibility: The web interfaces will be designed to be compatible with major web browsers such as Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge.

**Networking**

Internet Connectivity: The application requires a stable internet connection for real-time data exchange between the mobile devices and the backend servers.

API Communication: The application interacts with backend services through APIs, necessitating reliable network connectivity.

**Security Measures**

Secure Sockets Layer (SSL): Utilized to establish secure communication channels between the mobile application and backend servers.

Data Encryption: User data and transaction information are encrypted to ensure confidentiality.

**External Integrations**

Google Services: Integration with Google Maps for mapping and location-related features.

Payment Gateways: Integration with third-party payment gateways to facilitate secure financial transactions.

Authentication Services: Interaction with external authentication services for secure user logins.

**Database Management System**

Relational Database: Backend services manage data using a relational database management system (e.g., MySQL, PostgreSQL).

**Development Tools and Frameworks**

Mobile App Frameworks: The application is developed using frameworks such as React Native for cross-platform compatibility.

Backend Frameworks: Backend services may use frameworks like Django or Flask for efficient development.

**External Hardware (if applicable)**

GPS Sensors: The application relies on GPS sensors embedded in smartphones to provide accurate location data.

## **Design and Implementation Constraints**

The development of the Uber Application is subject to various limitations and constraints that influence the options available to the developers. Understanding these factors is crucial for planning and executing the development process effectively. The key limitations and constraints include:

**Regulatory Policies**

Data Privacy Regulations: Compliance with data protection laws and regulations, such as GDPR, influences how user data is collected, stored, and processed.

Transportation Regulations: Adherence to local and international transportation regulations may impose constraints on certain features and functionalities.

**Corporate Policies**

Security Policies: Corporate security policies dictate measures for securing user data, communications, and transactions within the application.

User Authentication Standards: Adherence to specific authentication standards and protocols defined by the corporate policies.

**Hardware Limitations**

Mobile Device Constraints: The application's performance is influenced by the varying hardware specifications of smartphones used by riders and drivers.

Server Capacity: Scalability considerations for backend servers to accommodate a growing user base.

**Interface Dependencies**

External Mapping and Navigation APIs: Reliance on third-party services, such as Google Maps API, for mapping and navigation features.

Payment Gateway Integration: Dependency on external payment gateways for secure financial transactions.

**Technology and Framework Choices**

Mobile App Frameworks: Selection of specific frameworks (e.g., React Native) for cross-platform development, impacting development options.

Backend Frameworks: Choice of backend frameworks (e.g., Django, Flask) influencing development approaches.

**Communication Protocols**

API Communication: Adherence to established communication protocols for seamless interaction between the mobile application and backend services.

**Security Considerations**

SSL Implementation: The use of SSL for secure communication, with potential impacts on latency and data transmission.

Data Encryption: Encryption of user data and transactions introduces considerations for performance and computational overhead.

**Design Conventions and Standards**

User Interface Standards: Adherence to specific design conventions for ensuring a consistent and user-friendly interface.

Coding Standards: Compliance with coding standards and conventions for maintainability and future updates.

**Database Management System**

Database Compatibility: Integration with a specific relational database management system (e.g., MySQL, PostgreSQL) based on compatibility and performance considerations.

**Parallel Operations**

Concurrency: Managing parallel operations, especially during high demand periods, to ensure a smooth user experience.

**Language Requirements**

Localization: Considerations for language localization to cater to a diverse user base.

**Maintenance Responsibilities**

Customer Maintenance: Determining whether the customer's organization will be responsible for maintaining the delivered software, impacting development choices.

## **User Documentation**

User documentation is a critical aspect of software development, providing users with the information they need to understand, operate, and troubleshoot the software. Components of user documentation for an Uber-like application may include:

**User Manuals:**

Detailed manuals providing step-by-step instructions on how to use the application. This may include sections on account creation, requesting rides, making payments, and other essential functionalities.

**Online Help:**

Interactive help resources integrated into the application. These may include tooltips, context-sensitive help buttons, or a searchable knowledge base accessible from within the app.

**Tutorials:**

Guided tutorials or walkthroughs to assist users in learning the application. These could cover the initial setup, requesting a ride, navigating the app, and any other complex features.

**FAQs (Frequently Asked Questions):**

A list of common questions and answers that users might have. This can help users quickly find solutions to common issues without having to contact support.

**In-App Messages and Notifications:**

Short, contextual messages within the application guiding users on new features, updates, or changes. These can be used for brief tips or important announcements.

**Video Demonstrations:**

Short video clips demonstrating specific features or common tasks. Visual aids can be powerful for users who prefer a more dynamic learning experience.

**Support Center:**

A dedicated support center accessible through the application or website. This could include contact information for customer support, live chat options, or a ticketing system.

**Terms of Service and Privacy Policy:**

Clear documentation outlining the terms of service and privacy policies. This is crucial for informing users about their rights, responsibilities, and how their data will be handled.

**Accessibility Documentation:**

Information on how the application complies with accessibility standards, making it usable for individuals with disabilities.

**Release Notes:**

Documentation highlighting the changes, improvements, and new features introduced in each software release. This can help users stay informed about updates.

## **Assumptions and Dependencies**

The successful development and implementation of the Uber Application rely on several assumptions and dependencies. These factors, while not guaranteed, form the basis for planning and decision-making throughout the project. Understanding these assumptions and dependencies is crucial for managing potential risks and ensuring project success.

**Assumptions**

**Third-Party API Reliability:**

Assumption: External services and APIs, such as mapping and payment gateways, will operate reliably and maintain compatibility throughout the project.

Impact if Incorrect: Unforeseen changes or disruptions in third-party services could impact the application's functionality and user experience.

**Regulatory Stability:**

Assumption: Regulatory environments governing ride-sharing services will remain stable during the development period.

Impact if Incorrect: Changes in regulations may necessitate adjustments to the application to ensure compliance.

**User Device Compatibility:**

Assumption: Users will have smartphones with compatible operating systems capable of running the Uber Application.

Impact if Incorrect: Incompatibility with a significant portion of user devices may limit the application's reach.

**Network Connectivity:**

Assumption: Users and drivers will have access to reliable internet connectivity for real-time communication and data exchange.

Impact if Incorrect: Poor network conditions may result in service disruptions and impact the real-time nature of the application.

**Security Measures Effectiveness:**

Assumption: Security measures implemented, such as SSL and data encryption, will effectively safeguard user data.

Impact if Incorrect: Security vulnerabilities could compromise user information and trust in the application.

**Dependencies**

**Google Maps API:**

Dependency: The application relies on the availability and functionality of the Google Maps API for mapping and navigation.

Impact if Unavailable: Mapping features may be compromised, affecting the user's ability to request and navigate rides.

**Payment Gateway Services:**

Dependency: Integration with third-party payment gateways for secure financial transactions.

Impact if Unavailable: The application's ability to process payments and complete transactions would be affected.

**Mobile App Development Frameworks:**

Dependency: Utilization of specific mobile app development frameworks, such as React Native.

Impact if Unavailable: A shift to alternative frameworks may be necessary, potentially impacting development timelines.

# **System Features**

The functional requirements for the Uber Application are organized based on system features, representing the major services provided by the product. Each feature encapsulates a set of functionalities aimed at delivering a comprehensive and seamless user experience.

1. **User Authentication and Authorization**

**User Registration:**

The system shall allow users to create a new account by providing necessary information such as name, email, phone, and password.

The system shall validate and store user registration data securely.

**User Login:**

Authenticated users shall be able to log in using their registered email and password.

The system shall verify user credentials and grant access to the user's account.

**User Roles:**

The system shall distinguish between rider and driver roles.

Riders and drivers shall have access to role-specific functionalities.

1. **Ride Management**

**Requesting a Ride:**

Riders shall be able to request a ride by specifying the pickup and drop-off locations.

The system shall allocate the nearest available driver for the ride.

**Accepting a Ride:**

Drivers shall receive ride requests and have the option to accept or decline.

The system shall notify riders when a driver accepts their ride request.

**Starting and Ending a Ride:**

Drivers shall have the ability to start and end a ride upon reaching the destination.

The system shall calculate the fare based on the distance traveled and time taken.

**Ride History:**

Both riders and drivers shall have access to a history of completed rides.

The system shall display details such as date, time, locations, and fare for each ride.

1. **Location and Mapping**

**Real-time Location Tracking:**

The system shall continuously track and update the real-time locations of both riders and drivers.

Users shall be able to view the live location of their assigned driver.

**Mapping and Navigation:**

The application shall integrate with mapping services to provide optimal routes for rides.

Drivers shall receive navigation assistance for reaching the rider's pickup location and the destination.

1. **Profile Management**

**Viewing and Editing Profiles:**

Users shall have the ability to view and edit their profile information.

Changes to profile data shall be reflected across the application.

**Rating and Reviews:**

After completing a ride, both riders and drivers shall have the option to rate and review each other.

The system shall calculate and display an average rating for each user.

1. **Notifications**

**Push Notifications:**

The system shall send push notifications to users for ride updates, driver acceptance, and other relevant events.

Users shall have the option to enable or disable push notifications.

## **System Feature**s

**3.1.1 Description and Priority**

1. **Real-Time Location Tracking:**

Description: Implement real-time GPS tracking for both drivers and riders to enhance the accuracy of ride requests and facilitate better coordination.

Priority: High

Component Ratings:

Benefit: 9 (Critical for ride accuracy and user safety)

Penalty: 3 (Minor inconvenience if not implemented)

Cost: 7 (Moderate due to GPS integration)

Risk: 8 (Complexity in maintaining real-time tracking)

1. **In-App Messaging System:**

Description: Integrate an in-app messaging system for communication between riders and drivers without revealing personal contact details.

Priority: Medium

Component Ratings:

Benefit: 7 (Enhances communication and safety)

Penalty: 5 (Users can still communicate through alternative means)

Cost: 6 (Moderate development effort)

Risk: 5 (Potential challenges in maintaining secure messaging)

1. **Multi-Language Support:**

Description: Enable the application to support multiple languages to cater to a diverse user base.

Priority: High

Component Ratings:

Benefit: 8 (Expands user accessibility)

Penalty: 4 (Limited impact on users comfortable with default language)

Cost: 7 (Moderate effort for localization)

Risk: 6 (Potential challenges in translation accuracy)

1. **Emergency SOS Button:**

Description: Include an emergency button that users can press to quickly connect with emergency services and share their location.

Priority: High

Component Ratings:

Benefit: 9 (Critical for user safety)

Penalty: 2 (No significant drawback)

Cost: 8 (Development and integration effort)

Risk: 7 (Ensuring reliable emergency services integration)

1. **Ride History and Ratings:**

Description: Provide users with a detailed history of past rides, along with the ability to rate and review drivers.

Priority: Medium

Component Ratings:

Benefit: 6 (Enhances user experience and accountability)

Penalty: 4 (Users can still use the app without this feature)

Cost: 6 (Database and UI development effort)

Risk: 5 (Ensuring accuracy in rating calculations)

**3.1.2 Stimulus/Response Sequences**

**User Action: Rider Requests a Ride**

User opens the app, enters the destination, and requests a ride.

System Response: The system identifies the rider's current location using real-time GPS tracking.

**User Action: Driver Accepts the Ride**

Driver receives the ride request and accepts it.

System Response: The system calculates the estimated time of arrival (ETA) based on the real-time location of the driver.

**User Action: Rider Monitors Driver's Arrival**

Rider monitors the app to track the real-time movement of the driver towards the pickup location.

System Response: The system continuously updates the driver's location on the rider's map.

**User Action: Driver Navigates to Pickup Point**

Driver uses in-app navigation to reach the rider's location.

System Response: The system provides turn-by-turn directions based on real-time GPS data.

**User Action: Driver Arrives, Rider Boards**

Driver arrives at the pickup point, and the rider boards the vehicle.

System Response: The system updates the ride status and initiates real-time tracking of the ride in progress.

**User Action: Rider Monitors Ride Progress**

Rider monitors the app to track the real-time movement of the vehicle during the ride.

System Response: The system continuously updates the vehicle's location on the rider's map.

**User Action: Driver Navigates to Destination**

Driver uses in-app navigation to reach the destination.

System Response: The system provides turn-by-turn directions based on real-time GPS data.

**User Action: Ride Completed**

Driver reaches the destination, and the ride is completed.

System Response: The system updates the ride status, calculates the fare based on the actual distance traveled, and concludes real-time tracking.

**User Action: Rider Rates the Driver**

Rider provides a rating and, optionally, feedback for the completed ride.

System Response: The system records the rating and feedback for the driver.

**3.1.3 Functional Requirements**

REQ-1: User Registration and Authentication : users can create accounts with valid email addresses and phone numbers. Users can log in securely using their credentials.

REQ-2: Ride Booking and Management:

- riders can request rides by specifying their pickup and drop-off locations

-drivers can accept or decline ride requests

-riders and drivers can canel rides with appropriate notifications

-riders can view real-time ride tracking and estimated arrival times

-riders can specify ride preferences (e.g., vehicle type, payment method).

REQ-3: Driver Management:

-drivers can register their vehicles and provide necessary documents.

-drivers can set their availability status (e.g., online, offline).

-the system assigns rides to available drivers based on location and suitability

REQ-4: Fare Calculation and Payment:

-the system calculates ride fares based on distance, time, and other factors.

-riders can view fare estimates before confirming a ride.

-payment processing includes options like credit cards, digital wallets, and cash.

-invoices and receipts are generated for completed rides.

REQ-5: Ride Rating and Review:

-riders and drivers can rate each other and provide optional reviews after each ride

-the system records and displays these ratings and reviews.

REQ-6: Ride History and Records:

-users can view their ride history, including details of past rides and receipts

-users can access past invoices and payment records

REQ-7: Notifications:

-users receive real-time notifications for ride updates, driver arrivals, and other relevant information

-notifications are sent through push notifications and/or SMS.

REQ-8: Help and Support:

-users can access customer support and report issues within the app.

-customer support agents can respond to user inquiries and issues.

REQ-9: Admin Panel:

-admins can access an administrative panel for user management and system monitoring

-admins can suspend or ban users and drivers for policy violations.

# **External Interface Requirements**

## **User Interfaces**

**Rider Interface**

**Login Screen:**

Description: The login screen allows riders to enter their credentials (email and password) to access their accounts.

**Logical Characteristics:**

Standard email and password input fields.

"Forgot Password" link for password recovery.

**Registration Screen:**

Description: This screen facilitates new users in creating an account.

**Logical Characteristics:**

Input fields for name, email, phone, and password.

"Register" button to submit registration details.

**Ride Request Screen:**

Description: Riders use this screen to request a ride by specifying pickup and drop-off locations.

**Logical Characteristics:**

Map interface for selecting locations.

"Request Ride" button to initiate the ride request.

**Ride Details Screen:**

Description: Displays details of the assigned ride, including driver information and estimated time of arrival.

**Logical Characteristics:**

Driver details (name, vehicle, rating).

"Cancel Ride" button to cancel the requested ride.

**Ride History Screen:**

Description: Lists the rider's ride history with details of past rides.

**Logical Characteristics:**

Date, time, locations, and fare details for each ride.

"View Details" option for individual rides.

**Profile Screen:**

Description: Allows riders to view and edit their profile information.

**Logical Characteristics:**

Editable fields for name, email, phone, and password.

"Save Changes" button to update profile.

**Rating and Review Screen:**

Description: Appears after the ride is completed, allowing riders to rate and review the driver.

**Logical Characteristics:**

Star rating system and text box for reviews.

"Submit Rating" button.

**Driver Interface**

Driver Login Screen:

Description: Enables drivers to log in with their credentials.

**Logical Characteristics:**

Standard login fields for email and password.

Accept/Reject Ride Screen:

Description: Appears when a ride request is received, allowing the driver to accept or reject the ride.

**Logical Characteristics:**

Ride details (pickup, drop-off, rider information).

"Accept" and "Reject" buttons.

Start/End Ride Screen:

Description: Allows the driver to start and end a ride.

**Logical Characteristics:**

"Start Ride" and "End Ride" buttons.

Fare calculation and navigation details.

**Driver History Screen:**

Description: Displays the driver's history of completed rides.

**Logical Characteristics:**

Date, time, locations, and fare details for each ride.

**Driver Profile Screen:**

Description: Allows drivers to view and edit their profile information.

**Logical Characteristics:**

Editable fields for name, vehicle details, and password.

**Common Characteristics**

**Map Interface:**

Description: Integrated map for real-time location tracking and navigation.

**Logical Characteristics:**

Continuous updates of rider and driver locations.

Route visualization for ongoing rides.

**Notification Interface:**

Description: Push notifications for ride updates, acceptance, and other relevant events.

**Logical Characteristics:**

Notifications appear as banners or alerts.

**Menu Interface:**

Description: Standardized menu across screens for navigation.

**Logical Characteristics:**

Icons or buttons for common actions (e.g., view profile, log out).

## **Hardware Interfaces**

**User Device Interface:**

**Logical Characteristics:**

Supported Devices: Android and iOS smartphones.

Responsive Design: Ensures optimal user experience on various screen sizes.

Touch-Based Interaction: Intuitive touch gestures for navigation.

**Map Interface:**

**Logical Characteristics:**

Real-Time Location: Continuously updates user and driver locations.

Interactive Map: Allows users and drivers to interact with the map for location selection and navigation.

Route Visualization: Displays the route during an ongoing ride.

**Notification Interface:**

**Logical Characteristics:**

Push Notifications: Sends timely alerts for ride updates, ride acceptance, and other relevant events.

In-App Notifications: Displays notifications within the application interface.

**Menu Interface:**

**Logical Characteristics:**

Standardized Menu: Consistent menu across screens for navigation.

Icon-Based Navigation: Uses icons or buttons for common actions (e.g., view profile, log out).

**Physical Characteristics**

**Device Compatibility:**

**Physical Characteristics:**

Rider Devices: Compatible with Android smartphones (version 5.0 and above) and iOS devices (version 10 and above).

Driver Devices: Compatible with Android smartphones (version 5.0 and above) and iOS devices (version 10 and above).

**Communication Protocols:**

**Physical Characteristics:**

HTTPS: Secure communication between the application and the server.

GPS: Utilizes device GPS capabilities for real-time location tracking.

**Data Interaction:**

**Physical Characteristics:**

Cellular Data: Requires active internet connectivity for data interaction.

Wi-Fi: Supports data interaction over Wi-Fi connections.

**Hardware Sensors:**

**Physical Characteristics:**

GPS Sensor: Utilizes the device's GPS sensor for accurate location tracking.

Touchscreen: Supports touch-based interactions for user input.

**Device Controls:**

**Physical Characteristics:**

Buttons: Utilizes device buttons for standard actions (e.g., home, back).

Touchscreen Gestures: Supports common touchscreen gestures (e.g., swipe, tap).

## **Software Interfaces**

Our application, built on Django 5.0, interacts with various software components, databases, and external services to deliver a robust and feature-rich rideshare platform.

#### Software Components

**Django Framework:**

**Connections:**

Connects to the application's backend and serves as the primary web framework.

**Data Flow:**

Handles HTTP requests and responses, manages the application's URL routing, and interfaces with the database.

**Database Management System (DBMS):**

**Connections:**

Utilizes Django's ORM (Object-Relational Mapping) to interact with the database.

**Data Flow:**

Stores and retrieves data related to users, rides, locations, payments, and other application entities.

#### External Services

**Google Maps API:**

**Connections:**

Integrated for map rendering, location services, and route planning.

**Data Flow:**

Sends requests for geocoding, reverse geocoding, and navigation information.

**Payment Gateway:**

**Connections:**

Interfaces with a third-party payment gateway for secure payment processing.

**Data Flow:**

Sends payment requests and receives payment confirmations.

#### Communication Protocols

**HTTPS:**

**Connections:**

Ensures secure communication between the application and external services.

**Data Flow:**

Encrypts data during transmission, safeguarding sensitive information.

#### Data Sharing Mechanism

**Django ORM (Object-Relational Mapping):**

**Implementation Constraint:**

Utilizes Django's built-in ORM to map objects to database entities.

**Data Flow:**

Facilitates seamless data interaction between the application's backend and the database.

#### Application Programming Interfaces (APIs)

**Google Maps API:**

**Protocol:**

Utilizes RESTful API calls for map-related functionalities.

**Data Flow:**

Sends HTTP requests to the Google Maps API and processes responses.

**Payment Gateway API:**

**Protocol:**

Implements the specified API for payment transactions.

**Data Flow:**

Establishes secure communication for initiating and confirming payment transactions.

## **Communications Interfaces**

The rideshare application involves various communication functions to ensure seamless interaction between users, the application server, and external services. Below are the requirements associated with these communication functions:

**Communication Protocols**

**HTTPS (Hypertext Transfer Protocol Secure):**

**Requirement:**

All communication between the client (user's device) and the application server must be secured using HTTPS.

**Rationale:**

Ensures the confidentiality and integrity of data during transmission, safeguarding user information.

**External Service Communication**

**Google Maps API Communication:**

**Requirement:**

The application must communicate with the Google Maps API for map rendering, location services, and route planning.

**Message Formatting:**

Requests and responses must adhere to the specifications outlined in the Google Maps API documentation.

Data formats may include JSON or XML based on API requirements.

**Communication Security:**

API requests to Google Maps must be made over HTTPS to ensure secure data exchange.

**Payment Gateway Communication:**

**Requirement:**

The application must communicate with a third-party payment gateway for processing financial transactions.

**Message Formatting:**

Adheres to the specifications provided by the chosen payment gateway.

Commonly involves exchanging data in JSON or XML formats.

**Communication Security:**

Utilizes secure communication protocols (e.g., HTTPS) to protect sensitive financial information during transactions.

**User Communication**

**Email Notifications:**

**Requirement:**

The system should be capable of sending email notifications to users for various events (e.g., ride confirmation, password reset).

**Message Formatting:**

Emails should include relevant details and conform to standard email formatting.

**Communication Security:**

Emails may contain sensitive information; therefore, secure email protocols (e.g., SMTP with TLS) should be used.

**Data Transfer Rates**

**Optimized Data Transfer:**

**Requirement:**

Data transfer rates between the client application and the server should be optimized for responsiveness.

**Rationale:**

Enhances user experience by minimizing latency in data retrieval and updates.

**Synchronization Mechanisms**

**Real-time Updates:**

**Requirement:**

The application must support real-time synchronization for dynamic data, such as live location updates during a ride.

**Rationale:**

Ensures users receive timely and accurate information, especially during critical interactions like ride tracking.

# **Other Nonfunctional Requirements**

## **Performance Requirements**

The performance requirements are that the application should provide quick response times for ride requests and updates. The response times for real-time location tracking should be minimal. The site should serve a large number of customers, be secure and protect personal information, and provide updates to drivers and riders.

## **Safety Requirements**

The safety requirements are that the user data should be protected and handled in accordance with privacy laws. The regular security assessments and vulnerability scans should be performed. The safety of our riders is of importance. In order to provide safety we provide the name, picture, and vehicle of the driver, so the rider knows what vehicle to walk up too.

## **Security Requirements**

The user data, including personal and payment information, should be securely stored and transmitted. The authentication and authorization mechanisms should prevent unauthorized access. The data encryption and secure API endpoints should be in place.

## **Software Quality Attributes**

The reliability of the software is the system should be highly available with minimal downtime, data backup and disaste recovery plans should be in place. The user interface should be intuitive and accessible on various devices and screen, the usability for the software is that the user support and help options should be readily available within the app. Our application should be compatible wit major mobile platforms and web browsers.

# **Other Requirements**

The system should provide real time location tracking from riders and drivers. Notifications should be sent to both riders and drivers to keep them informed about the ride request status. A database should be implemented in order for users to have the ability to login and retain past information. This will also benefit admins and drivers, as they are able to see users history as well as past ratings.

In terms of legal requirements, we will need to be able to conduct thorough background checks on drivers and users. If users or drivers use our app with malicious intent, our software could face legal backlash and will result in less use than competitors. There will need to be heavy censoring of information to drivers and riders to avoid these repercussions, such as location data after a ride has been finished, payment data such as credit card information, and other personal information used for authentication. The database will need to be secure in order to avoid legal backlash.

**Appendix A: Glossary**

1. **User:**

**Description:** Represents users of the application, including riders and drivers.

**Attributes:**

**UserID:** Unique identifier for the user. This will be a number that is automatically assigned to a user when they sign up.

**Username:** User's login username. The user can provide this and it will be unique to each user ID.

**FullName:** Full name of the user. Does not have to be unique, and it will mainly be used in the UI for the program to address the user.

**Email:** User's email address. This is unique to each user, that way they are not juggling two accounts.

**Phone:** User's contact phone number. This will also be unique to the user, so as to not miscommunicate with the wrong user/multiple users.

**Password:** Encrypted user password. This will allow the user to login and does not have to be unique.

**Role:** User's role. The main three roles are Rider, Driver, or Admin.

**AccountStatus:** User's account status. This should only be Active or Suspended. Active accounts should have full access whereas Suspended accounts will have restricted access.

**2. Vehicle:**

**Description:** Represents vehicles registered by drivers.

**Attributes:**

**VehicleID:** Unique identifier for the vehicle. This will be a number that is automatically assigned to a vehicle when they sign up as a Driver.

**DriverID:** Identifies the driver associated with the vehicle. Similar to the VehicleID it should also be unique.

**LicensePlate:** Vehicle's license plate number. This is unique to the Vehicle and serves as a way for Riders to confirm that they are being picked up in the correct vehicle even if it is the same Make, Model, Color, etc.

**Make:** Vehicle's make (e.g., Toyota, Honda).

**Model:** Vehicle's model (e.g., Camry, Civic).

**Color:** Vehicle's color.

**Year:** Vehicle's manufacturing year.

**Seats:** Number of available passenger seats in the vehicle.

**Status:** Vehicle's availability status (e.g., Available, Offline).

**3. Ride:**

**Description:** Represents ride transactions between riders and drivers.

**Attributes:**

**RideID:** Unique identifier for the ride.

**RiderID:** Identifies the rider requesting the ride.

**DriverID:** Identifies the driver providing the ride.

**StartLocation:** Pickup location coordinates.

**EndLocation:** Drop-off location coordinates.

**StartTime:** Timestamp indicating ride start time.

**EndTime:** Timestamp indicating ride end time.

**Distance:** Distance traveled during the ride (in kilometers or miles).

**Duration:** Duration of the ride (in minutes).

**Fare:** Total fare amount for the ride.

**Status:** Ride status (e.g., Completed, Canceled).

**4. Payment:**

**Description:** Represents payment transactions related to rides.

**Attributes:**

**PaymentID:** Unique identifier for the payment.

**RideID:** Identifies the ride associated with the payment.

**UserID:** Identifies the user making the payment (rider or driver).

**PaymentMethod:** Payment method used (e.g., Credit Card, Digital Wallet, Cash).

**Amount:** Payment amount.

**Timestamp:** Timestamp of the payment transaction.

**5. Rating and Review:**

**Description:** Represents ratings and optional reviews provided by riders and drivers for each other after a ride.

**Attributes:**

**RatingID:** Unique identifier for the rating and review.

**RideID:** Identifies the ride associated with the rating.

**RatedUserID:** Identifies the user being rated (rider or driver).

**Rating:** Numeric rating (e.g., 1 to 5 stars).

**Review:** Optional text-based review or feedback.

**Timestamp:** Timestamp when the rating and review were submitted.

**Appendix B: Analysis Models**

*<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams*.>

**Appendix C: Issues List**

*< This is a dynamic list of the open requirements issues that remain to be resolved, including TBDs, pending decisions, information that is needed, conflicts awaiting resolution, and the like.>*