Software Requirements Specification (SRS)

1. Introduction  
1.1 Purpose  
The purpose of this document is to define the software requirements for [Project Name], serving as a guideline for the development team.  
1.2 Scope  
This SRS document covers the functional and non-functional requirements for [Project Name].  
1.3 Definitions, Acronyms, and Abbreviations  
⦁ SRS: Software Requirements Specification  
⦁ [Add any specific terms or acronyms relevant to your project]  
1.4 References  
[Include any documents, standards, or external references relevant to this SRS.]  
2. Overall Description  
The Software Requirements Specification (SRS) document for the taxi app outlines the fundamental aspects of the software system, including its context within the broader ecosystem, high-level features, user classes and characteristics, operating environment, design and implementation constraints, as well as assumptions and dependencies.  
2.1 Product Perspective  
The taxi app is a standalone software system designed to facilitate seamless transportation services for users. It operates within a larger transportation ecosystem, which includes users, drivers, administrators, and external interfaces such as payment gateways and map services. The app interacts with these external components to provide users with a reliable and efficient taxi booking experience.  
2.2 Product Features  
The taxi app offers a range of high-level features and functionalities, including but not limited to:  
⦁ Ride Booking: Users can book rides by specifying pickup and drop-off locations and choose from various vehicle options.  
⦁ Real-Time Tracking: Real-time tracking of drivers' locations allows users to monitor the progress of their rides.  
⦁ Driver Management: Drivers can register, accept ride requests, and manage their availability.  
⦁ Payment Processing: Secure payment processing supports various payment methods for fare settlement.  
⦁ Rating and Feedback: Users can rate drivers and provide feedback after each ride.  
⦁ User and Driver Support: Comprehensive support for users and drivers via chat, phone, and in-app help center.  
⦁ Safety Features: Emergency services and safety measures for users and drivers in case of critical situations.  
⦁ Localization: Multi-language and currency support to cater to a diverse user base.  
2.3 User Classes and Characteristics  
The taxi app caters to three primary user classes:  
⦁ Customers: Users who seek transportation services. They create ride requests, interact with drivers, and pay for services.  
⦁ Drivers: Individuals who offer transportation services. They accept ride requests, provide rides, and earn through the platform.  
⦁ Administrators: System administrators responsible for managing users, drivers, rides, and overseeing the platform's operations.  
Each user class possesses unique characteristics, requirements, and roles within the application.  
2.4 Operating Environment  
The taxi app operates in a dynamic environment that includes:  
⦁ Mobile Platforms: Android and iOS for user and driver mobile applications.  
⦁ Server Infrastructure: Backend servers to handle ride requests, manage user profiles, and process payments.  
⦁ Network Connectivity: Reliable internet connectivity for real-time communication between users, drivers, and servers.  
2.5 Design and Implementation Constraints  
The design and implementation of the taxi app are subject to specific constraints, including technology choices, budget limitations, and compliance with local transportation regulations. Additionally, the app must ensure data privacy and security in accordance with relevant laws and regulations.  
2.6 Assumptions and Dependencies  
Assumptions made during the development process include assumptions about user behavior, system performance, and third-party service availability. The taxi app depends on external services for features such as map integration, payment processing, and communication services.  
This SRS document serves as a comprehensive guide for the development, testing, and maintenance of the taxi app, providing stakeholders with a clear understanding of its scope, features, and operational requirements.  
3. System Features  
3.1 Feature 1: Ride Booking  
  
3.1.1 Description  
  
The Ride Booking feature allows users to request rides by specifying their pickup and drop-off locations and selecting from various vehicle options.  
  
3.1.2 Requirements  
  
Functional Requirements:  
  
Users must be able to enter pickup and drop-off locations using text input fields or by selecting locations on a map.  
  
Users should have the option to choose from a list of available vehicle types, such as standard, premium, or shared rides.  
  
The app must provide estimated arrival times (ETA) for each vehicle type based on real-time traffic data.  
  
Users should receive confirmation notifications with ride details, including driver information and vehicle details, after successfully booking a ride.  
  
Users should be able to cancel a ride request within a specified time frame without incurring any cancellation charges.  
  
The app should allow users to specify additional ride preferences, such as car seats for children or wheelchair accessibility.  
  
Non-Functional Requirements:  
  
Performance:  
  
The system should respond to ride booking requests within 5 seconds on average, even during peak usage times.  
The app should load maps and real-time data efficiently, minimizing loading times.  
Security:  
  
User location data and personal information must be encrypted during transmission and storage.  
Payment information should be securely processed and comply with industry security standards (e.g., PCI DSS).  
Usability:  
  
The user interface for ride booking should be intuitive and easy to use, with clear instructions for entering pickup and drop-off locations.  
The app should support multi-language and multi-currency options to accommodate diverse user needs.  
Availability:  
  
The ride booking service should be available 24/7, with a target uptime of 99.9%.  
Redundancy and failover mechanisms should ensure service availability in the event of server failures.  
Scalability:  
  
The system should be designed to handle a minimum of 10,000 concurrent ride booking requests without significant performance degradation.  
Data Privacy and Compliance:  
  
The app must comply with data privacy regulations (e.g., GDPR) and ensure the protection of user data.  
Ride booking records should be retained in accordance with legal requirements and data retention policies.  
3.2 Feature 2: Real-Time Tracking  
  
3.2.1 Description  
  
The Real-Time Tracking feature enables users to track the real-time location of their assigned driver during a ride.  
  
3.2.2 Requirements  
  
Functional Requirements:  
  
Users should have access to a map displaying the real-time location of their assigned driver.  
  
The app should provide an estimated time of arrival (ETA) based on the driver's current location and route.  
  
Users should be able to contact the driver through in-app messaging or by making a phone call.  
  
The app should update the driver's location on the map at least every 15 seconds during the ride.  
  
Non-Functional Requirements:  
  
Performance:  
  
The real-time tracking feature should have a minimal delay, with location updates displayed within 5 seconds of receiving new data.  
Security:  
  
User and driver location data must be protected and not accessible to unauthorized users.  
Availability:  
  
Real-time tracking should be available throughout the duration of the ride, with minimal interruptions.  
3.3 Feature 3: Driver Management  
  
3.3.1 Description  
  
The Driver Management feature allows drivers to register, accept ride requests, and manage their availability.  
  
3.3.2 Requirements  
  
Functional Requirements:  
  
Drivers should be able to create driver profiles with personal information, vehicle details, and required documentation.  
  
Drivers should receive ride requests and have the option to accept or decline them.  
  
The app should allow drivers to set their availability status, such as "Available," "Busy," or "Offline."  
  
Drivers should have access to a dashboard displaying ride details, earnings, and performance metrics.  
  
Non-Functional Requirements:  
  
Performance:  
  
The driver registration process should be completed within 5 minutes on average.  
Security:  
  
Driver registration and profile data should be securely stored and encrypted.  
Availability:  
  
The driver management functions should be available 24/7, allowing drivers to accept ride requests at any time.  
3.4 Feature 4: Payment Processing  
  
3.4.1 Description  
  
The Payment Processing feature supports secure payment processing for fare settlement using various payment methods.  
  
3.4.2 Requirements  
  
Functional Requirements:  
  
Users should be able to add and manage payment methods, including credit cards and digital wallets.  
  
The app must securely handle payment transactions, including authorization, capture, and refunds.  
  
Users should receive digital receipts via email or within the app after completing rides.  
  
The app should support multiple currencies for international users.  
  
Non-Functional Requirements:  
  
Security:  
  
Payment transactions should comply with Payment Card Industry Data Security Standard (PCI DSS) requirements.  
Performance:  
  
Payment processing should not exceed 10 seconds, including authorization and confirmation.  
Availability:  
  
Payment processing services should have high availability to prevent disruptions during fare settlement.  
  
3.5 Feature 5: Rating and Feedback  
  
3.5.1 Description  
  
The Rating and Feedback feature allows users to rate drivers and provide feedback after each ride.  
  
3.5.2 Requirements  
  
Functional Requirements:  
  
Users should have the option to rate their ride experience, typically on a scale of 1 to 5 stars.  
  
Users should be able to leave comments or feedback about their ride experience.  
  
The app should provide drivers with access to their ratings and feedback.  
  
Aggregate driver ratings should be displayed to users when considering a driver for a ride.  
  
Non-Functional Requirements:  
  
Performance:  
  
Ratings and feedback submission should be processed in real-time, with confirmation provided to users.  
Usability:  
  
The rating and feedback interface should be user-friendly and easily accessible after each ride.  
Data Privacy:  
  
User ratings and feedback should be anonymized to protect user privacy.  
3.6 Feature 6: User and Driver Support  
  
3.6.1 Description  
  
The User and Driver Support feature provides comprehensive support to users and drivers through chat, phone, and an in-app help center.  
  
3.6.2 Requirements  
  
Functional Requirements:  
  
Users and drivers should have access to customer support representatives through in-app chat or phone support.  
  
An in-app help center should provide answers to frequently asked questions, troubleshooting guides, and support articles.  
  
Support requests should be tracked and managed efficiently, with timely responses to user inquiries.  
  
Non-Functional Requirements:  
  
Availability:  
  
Support services should be available during operating hours and responsive to user inquiries in a timely manner.  
Usability:  
  
The help center should be organized, searchable, and easy to navigate.  
Security:  
  
User and driver support interactions should be secure and protected.  
3.7 Feature 7: Safety Features  
  
3.7.1 Description  
  
The Safety Features feature includes emergency services and safety measures for users and drivers in case of critical situations.  
  
3.7.2 Requirements  
  
Functional Requirements:  
  
Users should have access to an emergency button or feature for immediate assistance in case of emergencies.  
  
Drivers should have a panic button in the app to alert authorities or request help.  
  
The app should provide safety tips and guidelines for both users and drivers.  
  
Non-Functional Requirements:  
  
Performance:  
  
Emergency assistance requests should be processed with the highest priority and minimal delay.  
Security:  
  
Panic button activations should be secure and not susceptible to false alarms.  
Availability:  
  
Safety features should be accessible at all times, even in low-network or offline conditions.  
3.8 Feature 8: Localization  
  
3.8.1 Description  
  
The Localization feature offers multi-language and multi-currency support to cater to a diverse user base.  
  
3.8.2 Requirements  
  
Functional Requirements:  
  
The app should support multiple languages for user interfaces, allowing users to choose their preferred language.  
  
Users should be able to select their preferred currency for fare calculations and payment processing.  
  
The app should provide accurate regional information, such as time zones and local taxes.  
  
Non-Functional Requirements:  
  
Usability:  
  
The localization should provide a seamless experience for users, with accurate translations and currency conversions.  
Performance:  
  
Language and currency selection should not significantly impact app performance.  
  
3.9 Feature 9: Ride History  
  
3.9.1 Description  
  
The Ride History feature allows users to view their ride history, including past rides, payment details, and driver information.  
  
3.9.2 Requirements  
  
Functional Requirements:  
  
Users should have access to a list of their previous rides, including dates, pickup, and drop-off locations.  
  
Users should be able to review detailed ride information, including fare charges, payment methods, and driver details.  
  
The app should allow users to filter and search for specific rides within their ride history.  
  
Non-Functional Requirements:  
  
Performance:  
  
The ride history should load within 3 seconds and support quick filtering and sorting options.  
Usability:  
  
The user interface for ride history should be intuitive and easy to navigate.  
Data Privacy:  
  
User ride history data should be securely stored and only accessible to authorized users.  
3.10 Feature 10: Promo Codes and Discounts  
  
3.10.1 Description  
  
The Promo Codes and Discounts feature allows users to apply promo codes and discounts to their rides for cost savings.  
  
3.10.2 Requirements  
  
Functional Requirements:  
  
Users should have the option to enter promo codes during the booking process.  
  
The app must validate and apply valid promo codes to fare calculations.  
  
Discounts should be clearly displayed to users during the booking process, including the applied discount amount and the final fare.  
  
Non-Functional Requirements:  
  
Performance:  
  
The promo code validation process should be quick and not introduce delays in the booking process.  
Usability:  
  
The app should provide clear instructions for entering and applying promo codes.  
3.11 Feature 11: Driver Earnings and Reports  
  
3.11.1 Description  
  
The Driver Earnings and Reports feature enables drivers to track their earnings and generate reports related to their rides and income.  
  
3.11.2 Requirements  
  
Functional Requirements:  
  
Drivers should have access to a dashboard displaying their earnings over time, including daily, weekly, and monthly summaries.  
  
Drivers should be able to generate income reports, including details of individual rides, earnings breakdowns, and expenses.  
  
The app should provide year-end income reports for tax purposes.  
  
Non-Functional Requirements:  
  
Performance:  
  
Earnings and reports should load quickly, even for drivers with a large number of completed rides.  
Usability:  
  
The earnings dashboard and report generation should be user-friendly and intuitive.  
3.12 Feature 12: Push Notifications  
  
3.12.1 Description  
  
The Push Notifications feature provides real-time push notifications to users and drivers for important updates, such as ride confirmations, driver arrivals, and promotions.  
  
3.12.2 Requirements  
  
Functional Requirements:  
  
Users and drivers should receive push notifications on their mobile devices for ride-related events and updates.  
  
Notifications should be customizable, allowing users to choose their notification preferences (e.g., sound, vibration).  
  
The app should support promotional notifications to inform users of special offers and discounts.  
  
Non-Functional Requirements:  
  
Performance:  
  
Push notifications should be delivered promptly, ensuring users and drivers are informed in real-time.  
Usability:  
  
Users should have the option to manage their notification preferences within the app.  
  
3.13 Feature 13: Multi-Platform Support  
  
3.13.1 Description  
  
The Multi-Platform Support feature ensures that the taxi app is available and functional on both Android and iOS platforms.  
  
3.13.2 Requirements  
  
Functional Requirements:  
  
Develop and maintain separate mobile applications for Android and iOS platforms.  
  
Ensure feature parity and a consistent user experience on both Android and iOS versions of the app.  
  
Keep both versions of the app up to date with the latest features and bug fixes.  
  
Non-Functional Requirements:  
  
Usability:  
  
Users on both Android and iOS should have a consistent and intuitive experience when using the app.  
Performance:  
  
The app should perform smoothly and responsively on both Android and iOS devices.  
Security:  
  
Ensure that security measures are consistent across both platforms to protect user data.  
3.14 Feature 14: Feedback and Dispute Resolution  
  
3.14.1 Description  
  
The Feedback and Dispute Resolution feature allows users and drivers to raise disputes or provide feedback in case of issues or disputes with a ride.  
  
3.14.2 Requirements  
  
Functional Requirements:  
  
Users should have the ability to report issues, accidents, or disputes related to their rides.  
  
Drivers should be able to report issues or disputes with users.  
  
The app should facilitate dispute resolution and communication between users and drivers, including a record of dispute outcomes.  
  
Non-Functional Requirements:  
  
Performance:  
  
The dispute resolution process should be efficient and timely, with clear communication to all parties involved.  
Security:  
  
User and driver dispute information should be securely stored and confidential.  
3.15 Feature 15: Driver Background Checks  
  
3.15.1 Description  
  
The Driver Background Checks feature ensures that drivers undergo background checks and verification processes before offering services on the platform.  
  
3.15.2 Requirements  
  
Functional Requirements:  
  
Implement a comprehensive driver background check process that includes criminal record checks and checks of driving history.  
  
Ensure that only drivers who pass the background checks are allowed to register and provide services on the platform.  
  
Non-Functional Requirements:  
  
Security:  
  
The background check process should be secure and compliant with legal and regulatory requirements.  
Data Privacy:  
  
Driver background check data should be securely stored and only accessible to authorized personnel.  
3.16 Feature 16: Surge Pricing  
  
3.16.1 Description  
  
The Surge Pricing feature enables dynamic pricing during high-demand periods to balance supply and demand.  
  
3.16.2 Requirements  
  
Functional Requirements:  
  
Implement surge pricing algorithms that adjust fares based on demand and availability of drivers.  
  
Clearly communicate surge pricing to users during peak times, including the multiplier applied to fares.  
  
Offer fare estimates that take surge pricing into account when users book rides.  
  
Non-Functional Requirements:  
  
Performance:  
  
Surge pricing calculations should be fast and responsive to changing demand conditions.  
Usability:  
  
Users should understand the concept of surge pricing and how it affects fares.  
3.17 Feature 17: In-App Chat  
  
3.17.1 Description  
  
The In-App Chat feature provides an in-app messaging system for communication between users and drivers.  
  
3.17.2 Requirements  
  
Functional Requirements:  
  
Users and drivers should have the ability to send text messages within the app.  
  
Implement message delivery and read receipts to track message status.  
  
Provide a chat history for users and drivers to review previous conversations.  
  
Non-Functional Requirements:  
  
Performance:  
  
Messages should be delivered promptly, and the chat system should handle concurrent conversations efficiently.  
Usability:  
  
The chat interface should be user-friendly and provide real-time notifications for new messages.  
  
4. External Interface Requirements  
4.1 User Interfaces  
  
User Mobile App Interface:  
Requirements:  
The user mobile app interface should be available on both Android and iOS platforms.  
It must support multiple languages and allow users to select their preferred language.  
The interface should include the following screens:  
Login/Registration: Users can create accounts or log in securely.  
Ride Booking: Users can enter pickup and drop-off locations, select vehicle types, and book rides.  
Ride Tracking: Real-time tracking of the driver's location during a ride.  
Payment: Users can manage payment methods and view transaction history.  
User Profile: Users can update their profiles and view ride history.  
Help Center: Access to FAQs, contact support, and emergency services.  
The interface should be intuitive, with a clear layout and navigation.  
Driver Mobile App Interface:  
Requirements:  
The driver mobile app interface should be available on both Android and iOS platforms.  
It must support multiple languages and allow drivers to select their preferred language.  
The interface should include the following screens:  
Driver Registration: Drivers can create accounts and submit required documentation.  
Ride Requests: Drivers can accept or decline ride requests.  
Navigation: Real-time navigation to pickup and drop-off locations.  
Earnings Dashboard: Drivers can view earnings, ride history, and ratings.  
Help Center: Access to FAQs, contact support, and emergency services.  
The interface should be user-friendly, with clear instructions and real-time updates.  
Administrator Dashboard Interface:  
Requirements:  
The administrator dashboard should be web-based and accessible from modern web browsers.  
It must support user roles with varying levels of access (e.g., superadmin, support staff).  
Features include user and driver management, ride tracking, and reporting tools.  
The interface should provide detailed insights and reporting capabilities for system administrators.  
It should be secure and require appropriate authentication for access.  
4.2 Hardware Interfaces  
  
Mobile Device Hardware:  
  
Requirements:  
The user and driver mobile apps should be compatible with common mobile devices, including smartphones and tablets.  
The apps should leverage device features such as GPS, cameras, and network connectivity for real-time tracking and communication.  
Server Infrastructure:  
  
Requirements:  
The server infrastructure should include servers for handling ride requests, user profiles, and payment processing.  
Hardware resources should be scalable to accommodate increased user demand.  
Redundancy and failover mechanisms should be in place to ensure high availability.  
4.3 Software Interfaces  
  
Payment Gateway Integration:  
  
Requirements:  
Integration with one or more payment gateways to securely process payment transactions.  
Compliance with payment gateway APIs and protocols for authorization, capture, and refund operations.  
Map and Navigation Services:  
  
Requirements:  
Integration with map and navigation services (e.g., Google Maps, Mapbox) for real-time location tracking, route planning, and directions.  
Use of relevant APIs to display maps and calculate routes efficiently.  
Database System:  
  
Requirements:  
Integration with a database system (e.g., MySQL, PostgreSQL) for storing user data, ride history, and driver profiles.  
Use of standard database connectors and queries for data retrieval and storage.  
5. Non-functional Requirements  
5.1 Performance Requirements  
  
Response Time:  
Requirement: The app should load and respond to user interactions within 2 seconds on average.  
Throughput:  
Requirement: The system should support a minimum of 1,000 concurrent users without significant performance degradation.  
Data Retrieval:  
Requirement: The app should retrieve and display ride information, including driver details and map data, efficiently.  
5.2 Security Requirements  
  
Data Encryption:  
Requirement: All communication between the app and servers should use industry-standard encryption protocols to protect user and payment data.  
User Authentication:  
Requirement: User accounts should be secured with strong password policies and offer multi-factor authentication options.  
Driver Background Checks:  
Requirement: The app should ensure that only verified and background-checked drivers are allowed to register and provide services on the platform.  
5.3 Availability Requirements  
  
Uptime:  
Requirement: The app should be available and operational 24/7 with a target uptime of 99.9%.  
Redundancy:  
Requirement: Redundant server infrastructure should be in place to ensure service availability even in the case of server failures.  
5.4 Scalability and Capacity  
  
Database Scalability:  
Requirement: The database should be designed to handle a growing volume of user data and ride records.  
Load Balancing:  
Requirement: Load balancing mechanisms should distribute incoming traffic evenly across server instances to ensure high availability during peak usage times.  
5.5 Usability and User Experience  
  
User Interface Consistency:  
Requirement: The user interface (UI) should provide a consistent and intuitive experience across different mobile platforms (Android and iOS).  
Accessibility:  
Requirement: The app should be accessible to users with disabilities, complying with relevant accessibility standards.  
5.6 Data Privacy and Compliance  
  
Data Privacy:  
Requirement: The app must comply with data privacy regulations (e.g., GDPR) and ensure the protection of user data.  
Compliance with Local Regulations:  
Requirement: The app should adhere to local transportation regulations and legal requirements in all regions of operation.  
5.7 Error Handling and Logging  
  
Error Handling:  
Requirement: The app should gracefully handle errors and exceptions, providing clear error messages to users when necessary.  
Logging:  
Requirement: Comprehensive logging mechanisms should be in place to track and monitor system activity for troubleshooting and auditing purposes.  
5.8 Network and Connectivity  
  
  
Offline Mode:  
Requirement: The app should provide limited functionality and offline access to essential features (e.g., viewing ride history) in the absence of a network connection.  
Low Bandwidth Mode:  
Requirement: The app should optimize data usage for users with limited bandwidth or in areas with poor network connectivity.  
  
6. Other Requirements  
[Include any other requirements not covered in previous sections.]  
7. Appendix A: Use Cases  
   
  
8. Appendix s: Glossary  
[List and define any domain-specific terms or acronyms.]