PROG8060
REQUIREMENTS AND SPECIFICATIONS

28-09-2023

"WHEELSEYE"

SHIVAM MAHAJAN (8856622)

GURPREET KAUR (8911147)

SULIKA SHARMA (8925953)

POOJA THAKKAR (8799270)

1. Purpose

The purpose of this car rental app is to provide a convenient and efficient platform for users to easily rent vehicles for their transportation needs. This app aims to convey the message that renting a car should be a hassle-free experience, offering users the freedom and flexibility to select the type of vehicle they need, choose rental durations, and make reservations effortlessly. Whether it's for daily commutes, weekend getaways, or special occasions, this app seeks to emphasize its commitment to simplifying the car rental process, offering a wide range of vehicle options, and ensuring a seamless and enjoyable experience for its users. Ultimately, it intends to communicate that renting a car should be as easy as a few taps on your smartphone, making transportation more accessible and convenient for everyone.

2. Overall Description

2.1 Product Perspectives:

The car rental app is designed to function as a standalone product, allowing users to rent vehicles easily and conveniently through a mobile application. However, it can also integrate with external services such as payment gateways, GPS navigation systems, and vehicle tracking services to enhance the user experience and provide additional functionalities.

2.2 User Roles and Characteristics:

- **Customers:** These are the primary users of the app who are looking to rent vehicles for personal or business purposes. They may have varying levels of familiarity with technology and car rental processes.

- **Administrators**: App administrators manage the fleet of vehicles, user accounts, and resolve disputes. They have access to backend functionalities to ensure the smooth operation of the service.

2.3 Operating Environment(s):

The car rental app operates on mobile devices (smartphones and tablets) with iOS and Android platforms. It requires an internet connection for real-time updates, location services for GPS navigation, and access to the device's camera for vehicle condition reporting.

2.4 Constraints:

1. Availability of Network Connectivity for Real-time Data Updates:

- This constraint emphasizes the reliance on a stable internet connection for the car rental app to function effectively. Users need a network connection to access the app, search for vehicles, make reservations, and receive real-time updates, such as notifications, location information, and booking confirmations.

- The app's dependency on network connectivity means that users in areas with poor or no network coverage may experience limitations or disruptions in their ability to use the app. Offline functionality and data caching may be implemented to mitigate this constraint to some extent.

2. Compatibility with a Range of Mobile Devices and Operating Systems:

- To ensure a wide user base, the app must be compatible with various mobile devices, including smartphones and tablets, and support multiple operating systems such as iOS (Apple) and Android (Google).

- Developing and maintaining compatibility across a diverse range of devices and platforms can be a challenge, as each may have unique hardware specifications and software behaviors. Careful testing and development practices are necessary to address this constraint.

3. Compliance with Local Regulations and Licensing for Car Rentals:

- Car rental services often operate under local, regional, and national regulations and licensing requirements. These regulations may involve vehicle safety standards, insurance, driver's license checks, and taxation.

- The app must adhere to these regulations to ensure legal and responsible car rental operations. Non-compliance could lead to legal issues, fines, or shutdowns, making it essential to thoroughly understand and address local legal constraints.

4. Adequate Server Infrastructure for Handling User Data and Reservations:

- The app relies on server infrastructure to manage user data, reservations, and other backend processes. This includes storing user profiles, vehicle information, transaction records, and maintaining real-time availability data.

- The constraint highlights the need for robust and scalable server infrastructure to handle increased user demand and data storage requirements. Inadequate server capacity can lead to slow response times, data loss, and system crashes.

2.5 Assumptions:

Let's provide further explanation for each of these assumptions:

1. Users Have Access to Compatible Mobile Devices:

- This assumption implies that users have smartphones or tablets with compatible hardware and operating systems to run the car rental app.

- It's crucial to confirm the app's compatibility with a wide range of devices and OS versions to ensure accessibility for a broad user base.

2. Users Are Responsible for Complying with Local Traffic Laws and Returning Vehicles in Good Condition:

- This assumption places responsibility on users to adhere to local traffic laws while using the rented vehicles. This includes obeying speed limits, parking regulations, and other traffic rules.

- Users are also expected to return the vehicles in the same condition as they received them, barring normal wear and tear. Any damage or issues resulting from misuse may incur additional charges.

3. Payment Information Provided by Users Is Accurate and Valid:

- This assumption implies that users provide accurate and valid payment information when making reservations and payments within the app.

- Ensuring the accuracy of payment information is essential to prevent payment disputes, failed transactions, or fraudulent activities. The app should also have mechanisms in place to handle failed payments and disputes.

4. Availability of Sufficient Vehicle Inventory for Rental:

- This assumption suggests that the app operates with a sufficient inventory of vehicles to meet user demand.

- Maintaining an adequate fleet of vehicles is essential to prevent disappointment among users who may be unable to find an available vehicle when needed. The app should implement effective inventory management and notifications to users when vehicles are in short supply.

These assumptions guide the development and operation of the car rental app, but it's important to remember that real-world conditions may vary. As such, it's essential to continuously monitor and adapt the app's features and policies to address any deviations from these assumptions and ensure a positive user experience.

2.6 Risks:

Certainly, let's further explain these risks associated with developing and operating a car rental app:

1. Data Security and Privacy Breaches:

- Car rental apps collect and store sensitive user data, including personal information, payment details, and location data. There's a significant risk of data breaches if security measures are not robust.

- Breaches can result in unauthorized access to user data, financial losses, identity theft, and damage to the app's reputation. To mitigate this risk, strong data encryption, access controls, regular security audits, and compliance with data privacy regulations (e.g., GDPR) are essential.

2. Technical Issues Leading to App Downtime or Errors:

- Technical issues such as software bugs, server outages, or network failures can disrupt the app's functionality and lead to downtime.

- App downtime can frustrate users, result in lost revenue, and harm the app's reputation. Thorough testing, monitoring, and a well-defined incident response plan are crucial to minimize these technical risks.

3. Negative User Experiences Due to Vehicle Availability and Condition:

- Users rely on car rental apps to provide accurate information about vehicle availability and condition. If the app fails in this regard, users may experience dissatisfaction and frustration.

- Misrepresenting vehicle availability or condition can lead to poor user reviews, lost trust, and potential legal disputes. Ensuring accurate and up-to-date information through regular vehicle inspections and timely updates is vital.

4. Legal and Regulatory Challenges in Various Regions:

- Car rental services are subject to a range of legal and regulatory requirements that can vary significantly by region and jurisdiction. These regulations may include licensing, insurance, safety standards, and tax compliance.

- Expanding the app's services to different regions may require adapting to local laws and regulations, which can be complex and time-consuming. Non-compliance can result in fines, legal actions, and the suspension of operations in certain areas.

2.7 Dependencies:

- Integration with payment gateways for financial transactions.

- Access to third-party services for GPS navigation and location data.

- Maintenance and servicing partnerships to ensure vehicle quality.

- Compliance with local regulations and permits for car rental operations.

3. System Features

3.1 language preference setting:

- offer multilingual support to accommodate a diverse user base.

- Automatically detect or allow users to select their preferred language.

3.2 Vehicle Search and Selection:

- Users can search for available vehicles using specific criteria such as the type of vehicle (e.g., sedan, SUV), vehicle model, and location.

- The chatbot provides a catalog or list of vehicles that match the user's search criteria, allowing them to browse and select the one that suits their needs.

3.3 Pricing:

- The chatbot should display rental rates, additional fees, and promotions for each car model, allowing users to compare prices across different models and durations.

- calculates the total cost of a car rental by selecting a model and duration, including base rate, additional services, and discounts, providing a detailed breakdown and allowing the user to proceed or adjust their choices.

3.4 Real-time availability checker:

- Ask the chatbot to check the availability of rental cars for a specific location and date range so that user can plan and book a car for their upcoming trip.

- User want to be notified in real time when a car becomes available at a specified area, so they can quickly book it for trip.

3.5 Rental History and Billing Records:

- Users can access their rental history within the app. This feature allows users to view details of past rentals, including dates, vehicle information, and pricing.

- Billing records are stored, and users can review past transactions, download invoices, and keep track of their spending.

3.6 Refund and cancellation handling:

- Explain the refund and cancellation policies.

- Facilitate the cancellation process and refund requests.

3.7 Notifications and Reminders:

- Send booking confirmations, reminders for upcoming reservations, and return instructions.

- Notify users of any changes to their reservations.

3.8 Support and Assistance:

- Offer a help center or FAQ section to address common questions and issues.

- Provide live chat support for real-time assistance.

- Enable users to contact customer service through the chatbot.

4. External Interface Requirements

External Interface Requirements for a car rental app in C# may include the following:

1. User Interfaces:

- **Mobile User Interface:** The primary interface for users will be the mobile app on iOS and Android platforms, developed using C# for Xamarin or similar technologies.

- **Web Interface (Optional):** A web-based interface for users to access the car rental service from desktop or laptop computers.

2. Hardware Interfaces:

- **Mobile Device Hardware:** The app should be compatible with various mobile devices, including smartphones and tablets, running iOS or Android.

- **Camera:** Access to the device's camera is required for users to capture and upload images of the vehicle's condition.

3. Software Interfaces:

- **Database System:** Connection to a database system (e.g., SQL Server, MySQL) to store user profiles, vehicle information, reservations, and transaction records.

- **External APIs:** Integration with external services and APIs for real-time vehicle availability, weather updates, and traffic data.

- **Communication Interfaces:** Communication interfaces for in-app messaging, notifications, and customer support.

4. Communication Interfaces:

- **Push Notifications:** Integration with push notification services (e.g., Firebase Cloud Messaging for Android, Apple Push Notification Service for iOS) for sending timely updates and alerts to users.

- **Email Communication:** Use of email services for sending booking confirmations, invoices, and notifications to users.

- **Messaging Services:** Integration with messaging services (e.g., Twilio, WhatsApp API) for customer support and communication.

5. Non-Functional Requirements

Certainly, here are some non-functional requirements related to performance, security, and software quality for your car rental app:

Performance Requirements:

1. Response Time: The app should provide quick response times for actions such as searching for vehicles, making reservations, and loading user profiles. Response times should typically be within 2-3 seconds.

2. Scalability: The system should be designed to handle a growing number of users and vehicles without a significant drop in performance. It should support scalability both in terms of the user base and the database.

3. Reliability: The app should have a high level of reliability, ensuring that it is available and operational 24/7, with minimal downtime for maintenance or updates.

4. Resource Utilization: The app should be resource-efficient, utilizing device resources (CPU, memory, battery) efficiently to prevent excessive drain or slowdown of the user's device.

Security Requirements:

1. Data Encryption: User data, payment information, and communication within the app should be encrypted using secure protocols to protect against unauthorized access.

2. Authentication and Authorization: The app should implement robust authentication mechanisms to ensure that only authorized users can access the system and perform specific actions. Different user roles should have appropriate access permissions.

3. Data Privacy: The app should comply with data privacy regulations (e.g., GDPR) and ensure that user data is handled and stored securely, with clear user consent for data processing.

4. Secure Payment Processing: Payment transactions should be conducted securely, complying with Payment Card Industry Data Security Standard (PCI DSS) requirements.

5. Vulnerability Scanning: Regular vulnerability scanning and security assessments should be conducted to identify and mitigate potential security risks.

Software Quality Requirements:

1. Usability: The app should be user-friendly, with an intuitive interface that is easy to navigate. It should undergo usability testing to ensure a positive user experience.

2. Maintainability: The codebase should be well-structured and maintainable to facilitate future updates and enhancements. Code comments and documentation should be provided for reference.

3. Performance Testing: Comprehensive performance testing should be conducted to ensure that the app performs well under load and can handle peak usage periods.

4. Code Quality: Adherence to coding standards, best practices, and code review processes should be maintained to ensure code quality and minimize bugs and defects.

5. Compatibility: The app should be tested and optimized for compatibility with various mobile devices, operating systems, and screen sizes.

6. Error Handling: Effective error handling and logging should be implemented to capture and report errors, ensuring quick resolution of issues.

7. Backup and Recovery: Regular automated backups of user data and system configurations should be performed to facilitate data recovery in case of unexpected failures.

6. References Used

https://scaledagileframework.com/nonfunctional-requirements/

https://www.sciencedirect.com/topics/engineering/functional-requirement#:~

https://enkonix.com/blog/functional-requirements-vs-non-functional/