Namma Saarathi

Ishwar Prakash Joshi

Executive Summary

Problem Statement

Consider a case, where the customer starts a trip with a totally unknown driver i.e., an auto wala, independent or an off-duty cab driver who is offering his services without an application-based booking platform, the customer can unknowingly be in a totally jeopardized situation if the cab driver is sketchy or has intentions to commit a crime.

also, if we consider another case, where the cab driver before starting your trip asks you to cancel the booking made through the mobile application and offers you lesser fare and you too being in a thrifty state of mind accepts his offer, there will be no assistance from the side of Ola or Uber etc in case of any mishap.

Proposed Solution

To counter this issue, we have come up with the Idea of our own application which will be known as ‘Namma Saarathi’.

Namma Saarathi Verification will be a ride-hailing application that provides a secure and trustworthy experience for customers in Bangalore, India. It will be designed to address the critical issue of safety and security for customers, particularly women and the elderly, who are often vulnerable to harassment and unsafe situations while using ride-hailing services.

The application will offer a comprehensive verification process for cab drivers, ensuring that they meet strict eligibility criteria before being allowed to register on the platform. This verification process will include submitting e-Aadhaar and Identity confirmation through facial recognition, along with the validation of their driver's license and vehicle documents.

Customers will also be required to register on the platform with their details along with a trusted email and can scan a unique QR code provided to the drivers to access their details, ensuring that they are matched with a verified driver. Once the customers are matched with the verified driver the details of the verified driver the application will also provide them with the functionality of forwarding the driver details to a trusted e-mail that they have added. In case the app cannot verify the driver, the application will advise the customer to not proceed with the ride.

Namma Saarathi Verification aims to provide customers with a sense of security and safety while using ride-hailing services. By using technology to create a safe and secure ride-hailing experience for all, it will have the potential to transform the ride-hailing industry for the better.

Introduction

The recent years have been very flourishing for the mobility providers such as Ola, Uber, Rapido etc. Especially in the metro cities such as Bangalore, which is being the IT hub of India, one of the busiest cities by road in India. One click on your smartphone app, and you get a cab or some other mobility facility to wherever you want to go; but during your trip, these applications are the only one to add a layer of security to your rides.

Consider a case, where the customer starts a trip with a totally unknown driver i.e., an auto wala, independent or an off-duty cab driver who is offering his services without an application-based booking platform, the customer can unknowingly be in a totally jeopardized situation if the cab driver is sketchy or has intentions to commit a crime.

also, if we consider another case, where the cab driver before starting your trip asks you to cancel the booking made through the mobile application and offers you lesser fare and you too being in a thrifty state of mind accepts his offer, there will be no assistance from the side of Ola or Uber etc in case of any mishap.

Not just in the hypothetical cases mentioned above, such incidents have occurred in Bangalore as well as other parts of the country recently such as 'Rapido Rape Case', 'Woman jumps off moving Rapido bike taxi in Bangalore to escape harassment', 'Cabbie gets 1 year in jail for sexually harassing a woman passenger' etc. These were the cases where the alleged criminals were found and were brought to justice. But after talking to some people, we came to know that there has been so many similar cases of harassment and eve teasing by the cabbies which could not be reported because either the driver was completely unknown, or he fled away.

Namma Sarthi Application will provide a layer of security to the rides which are unsecure and too will add an extra layer of security to the rides which have their own layer of security provided by the booking applications.

When a customer needs to take a ride, they will scan the QR code of the driver through the same app. If the QR code of the cab driver is verified by the app, the app will show the driver details to the customer and advise the customer to move forward with the choice. The app will also forward the driver details to the trusted email given by the customer. However, if the application cannot verify the driver, the application will advise the customer not to proceed with the ride.

Outline of the application design

Here's an outline of how the application can be designed:

I. Home screen: The home screen of the app will have two buttons - one for cab drivers and the other for customers.

II. Cab Driver Sign-up: The cab drivers can sign up on the app by providing their details, such as name, contact number, address, and car details. Once the driver provides all the details, the app will generate a unique QR code, which will be displayed on the driver's screen. The driver can save the QR code or take a screenshot for future use.
III. Customer Sign-up: The customers can sign up on the app by providing their details, such as name, contact number, and a trusted email address. Once the customer provides all the details, they can start using the app.
IV. QR Code Scanning: When a customer wants to hire a cab, they can scan the QR code of the driver through the app. If the QR code is verified, the app will show the driver's details to the customer.
V. Driver Verification: The app will verify the driver's credentials based on the data stored in the cloud database. If the app cannot verify the driver, it will advise the customer not to proceed with the ride.
VI. Customer Feedback: After completing the ride, the customer can provide feedback on the driver's service. The feedback system will help other customers choose the best driver.
VII. Driver Dashboard: The driver can access the app's dashboard to view their ride history, ratings, and earnings.
VIII. Customer Dashboard: The customer can access the app's dashboard to view their ride history, ratings, and feedback history.

That's a basic outline of how the application can be designed. We can add more features based on your requirements.

In-Scope Parameters

- Cab driver’s registration and verification process.
- Customers registration and access to verified drivers through QR codes.
- Cloud database to store driver and customer details.
- Trusted email feature for customers to receive driver details.
- Admin panel to manage driver verification process and customer support.
- Verification process to ensure driver eligibility criteria.
- Validation of driver's license and vehicle documents.
- Application security and data privacy.
- User-friendly interface for both drivers and customers.
- A mobile application for Android devices.

Out-of-Scope Parameters

- Payment processing or integration with any payment gateway.
- Ride booking functionality.

Solution Design Approach

The solution design approach for Namma Saarathi Verification would involve the following steps:

I. **Requirements gathering**: The first step would involve gathering detailed requirements from stakeholders and users, identifying their needs and expectations, and documenting them.
II. **Technology selection**: The next step would be selecting the appropriate technologies and tools required for developing the application, including Java-based technologies, cloud-based database, and security frameworks.
III. **System architecture**: Based on the requirements and technology selection, the system architecture would be designed, including the mobile application, database schema, and server-side APIs.
IV. **Design patterns**: Appropriate design patterns would be identified and implemented to ensure scalability, maintainability, and extensibility of the application.
V. **User interface design**: User interface design would be created for both cab drivers and customers, ensuring ease of use and intuitive navigation.
VI. **Testing**: The application would be tested at different stages of development, including unit testing, integration testing, and system testing, to ensure the quality of the application.
VII. **Deployment**: The application would be deployed to a cloud server, and appropriate mechanisms would be put in place to ensure the security and privacy of user data.
VIII. **Maintenance and support**: The final step would involve providing maintenance and support for the application, ensuring that it continues to meet the evolving needs of its users.

Overall, the solution design approach for Namma Saarathi would involve a systematic and comprehensive approach, ensuring that the application meets the needs and expectations of its users, while providing a secure and reliable platform for cab drivers and customers.

Benefits of the Application

1. Safety and Security: The app ensures the safety and security of customers by verifying the identity of the cab driver through the unique QR code.
2. Trustworthy Service: By verifying the identity of the cab driver, the app builds trust between the customer and the driver, creating a more reliable and trustworthy service.
3. Ease of Use: The app is user-friendly and simple to use, making it accessible to a wide range of users.

4. Reduced Harassment: By verifying the identity of the cab driver, the app can potentially reduce incidents of harassment or unsafe behaviour towards customers, particularly women and elderly passengers.
5. Data Analytics: The app can collect data on the usage patterns and preferences of customers, allowing for analysis and potential improvements to the service in the future.
6. Increased Business Opportunities: The app can provide an opportunity for cab drivers to increase their business by building trust with customers and potentially increasing their customer base.

Technicalities

Functional Requirements

1. **Cab Driver Registration:**

→ Cab drivers should be able to register on the application by providing their name, contact number, address, car details, and a profile picture.

→ The application should store this information in the cloud database and generate a unique QR code for each driver.

2. **Customer Registration:**

→ Customers should be able to register on the application by providing their name, contact number, and a trusted email address.

→ The application should store this information in the cloud database.

3. **QR Code Scanning:**

→ Customers should be able to scan the QR code of a cab driver through the application.

→ The application should decode the QR code and verify if the CabDriver data matches the scanned QR code.

→ If the verification is successful, the driver's details should be displayed to the customer.

4. **Driver Details Display:**

→ The application should display the cab driver's name, contact number, car details, and a profile picture to the customer after a successful QR code verification.

5. **Email Notification:**

→ The application should forward the cab driver's details to the customer's trusted email address after a successful QR code verification.

→ The email should contain the driver's name, contact number, car details, and a profile picture.

6. **Feedback System:**

→ Customers should be able to rate the driver's service after the ride.

→ The application should store the feedback in the cloud database and update the driver's rating accordingly.
→ The feedback system should help other customers choose the best driver.

7. **Dashboards:**

→ The application should have dashboards for Cab Driver and Customer that will show their ride history, ratings, and earnings.
→ The dashboards should be updated in real-time based on the data stored in the cloud database.

Non-Functional Requirements

1. **Performance:**

→ The application should be responsive and fast.
→ The application should be able to handle a large number of users and transactions simultaneously.
→ The application should be able to load the cab driver's details quickly after a successful QR code verification.

2. **Security:**

→ The application should ensure the security of user data and transactions.
→ The application should use encryption to secure the data stored in the cloud database.
→ The application should use two-factor authentication for logins.

3. **User Interface:**

→ The application should have an intuitive and user-friendly interface.
→ The application should have clear and concise instructions for using the QR code scanning feature.
→ The application should have a consistent design across all views.

4. **Compatibility:**

→ The application should be compatible with different Android devices and versions.
→ The application should work on both smartphones and tablets.

5. **Reliability:**

→ The application should be reliable and available at all times.
→ The application should handle errors gracefully and provide informative error messages to users.
→ The application should have a backup system in case of server failure.

6. **Maintainability:**

→ The application should be easy to maintain and update.
→ The application should follow standard coding practices and have clear documentation.
→ The application should have a testing environment to test updates before deploying them to production.

Java based technologies should we use to make this product more scalable and lightweight

1. **Spring Framework:** Spring Framework is a popular Java-based technology that is widely used for building scalable and lightweight web applications. It provides a comprehensive framework for building web applications that are easy to maintain, test, and deploy. Spring provides features like dependency injection, AOP, and MVC, which make it easier to write clean, modular, and extensible code. Spring also provides support for data access, security, and transaction management, which makes it a good choice for building enterprise-grade applications.

2. **Hibernate:** Hibernate is an Object-Relational Mapping (ORM) framework that provides a high-level abstraction over relational databases. Hibernate allows developers to work with persistent objects, rather than raw SQL queries, which makes it easier to build scalable and maintainable applications. Hibernate provides features like lazy loading, caching, and transaction management, which can help improve performance and scalability. Hibernate can be integrated with Spring to provide a complete solution for building scalable and lightweight web applications.

3. **Apache Kafka:** Apache Kafka is a distributed messaging system that can be used to build highly scalable and fault-tolerant applications. Kafka provides a distributed and fault-tolerant publish-subscribe messaging system that is designed to handle large volumes of data. Kafka can be used to build real-time data pipelines, event-driven architectures, and microservices-based applications. Kafka can be integrated with Spring to provide a scalable and lightweight messaging system for our application.

4. **Apache Spark:** Apache Spark is a distributed computing engine that can be used to process large volumes of data in parallel. Spark provides a unified API for batch processing, streaming, machine learning, and graph processing. Spark can be used to build scalable and real-time data processing pipelines. Spark can be integrated with Spring to provide a scalable and lightweight data processing engine for our application.

5. **Docker:** Docker is a containerization platform that can be used to build and deploy applications in a lightweight and portable way. Docker provides a way to package applications and their dependencies into containers, which can be run on any platform that supports Docker. Docker can be used to build scalable and portable microservices-based applications. Docker can be integrated with Spring to provide a scalable and lightweight containerization platform for our application.

In summary, to make this product more scalable and lightweight, we can use a combination of Spring Framework, Hibernate, Apache Kafka, Apache Spark, and Docker. These technologies provide a comprehensive solution for building scalable and maintainable web applications that can handle large volumes of data and traffic.

Meeting the Functional Requirements

To meet the functional requirements for this project, we need to ensure that the application functions as expected and provides the required features to both cab drivers and customers. Here are some steps we can take to meet the functional requirements:

Cab Driver Registration

- Create a registration form for cab drivers to input their details such as name, contact number, and license number.
- Validate the input data and store the driver details in a cloud database.
- Generate a unique QR code for each cab driver and associate it with their details in the database.
- Provide a confirmation message to the driver on successful registration.

Customer Registration

- Create a registration form for customers to input their details such as name, contact number, and email address.
- Validate the input data and store the customer details in a cloud database.
- Provide a confirmation message to the customer on successful registration.

Ride Verification

- When a customer needs a ride, they can open the app and scan the QR code of the cab driver.
- The app will validate the QR code and check if the cab driver is registered and verified in the database.
- If the cab driver is verified, the app will display the driver details to the customer, including their name, contact number, and license number.
- The app will send an email to the customer's trusted email address with the driver details.
- If the cab driver is not verified, the app will display a warning message to the customer and advise them not to proceed with the ride.

User Management

- Provide a login mechanism for both cab drivers and customers to access their accounts.
- Allow users to update their profile details such as name, contact number, and email address.
- Implement a mechanism for resetting forgotten passwords.

Database Management

- Ensure the database is properly designed and normalized to prevent data redundancy and inconsistencies.
- Use a cloud-based database to ensure data availability and scalability.
- Implement database backups and disaster recovery mechanisms to prevent data loss.

Application Management

- Ensure the application is properly designed and follows best practices in software engineering.
- Use a modern framework such as Spring or Hibernate to ensure code quality and maintainability.
- Implement unit tests and integration tests to ensure the application is functional and reliable.
- Use a continuous integration and continuous delivery (CI/CD) pipeline to automate the deployment process and ensure a consistent and reliable release process.

By implementing these measures, we can ensure that the application meets the functional requirements and provides a reliable and user-friendly experience to both cab drivers and customers.

Meeting the Non-Functional Requirements

To meet the non-functional requirements for this project, we need to consider several factors such as performance, scalability, security, and reliability.

Performance

To ensure the application runs smoothly and performs well, we can take the following steps:

- Use efficient algorithms and data structures for operations such as QR code scanning and database queries.
- Implement caching mechanisms to reduce database reads and improve response time.
- Optimize code to reduce the number of database requests and minimize processing time.
- Use connection pooling to minimize the overhead of opening and closing database connections.
- Perform load testing to identify bottlenecks and optimize system performance.

Scalability

To ensure the application can handle increasing traffic and users, we can take the following steps:

- Use a cloud-based infrastructure such as Amazon Web Services or Microsoft Azure that can scale horizontally to handle increasing demand.
- Implement a distributed database to ensure data availability and scalability.
- Use containerization and microservices architecture to break down the application into smaller, independent components that can be easily scaled up or down as needed.
- Implement a load balancer to distribute traffic across multiple instances of the application.
- Monitor performance metrics and scale resources proactively to handle traffic spikes.

Security

To ensure the application is secure and protects user data, we can take the following steps:

- Implement encryption for sensitive data such as passwords and email addresses.
- Use secure communication protocols such as HTTPS to protect data in transit.
- Implement authentication and authorization mechanisms to ensure only authorized users can access the application and data.
- Use input validation and data sanitization to prevent SQL injection attacks and other forms of injection attacks.
- Perform regular security audits and vulnerability assessments to identify and address security vulnerabilities.

Reliability

To ensure the application is reliable and minimizes downtime, we can take the following steps:

- Implement a backup and disaster recovery plan to ensure data availability in case of hardware or software failures.
- Use automatic failover mechanisms to minimize downtime in case of server failures.
- Implement a monitoring system to detect and alert on performance issues, errors, and failures.
- Implement logging and error reporting mechanisms to track errors and quickly identify and resolve issues.
- Perform regular maintenance and upgrades to ensure the application is up to date and stable.

By implementing these measures, we can ensure that the application is performant, scalable, secure, and reliable, meeting the non-functional requirements for the project.

Architecture Overview

Here's an example High-Level Design (HLD) template for this app:

1. Architecture Overview:

The app will follow a client-server architecture. The client-side will be an Android application and the server-side will be a cloud-based server. The app will use RESTful API to communicate with the server.

2. Client-Side:

The client-side of the app will be developed using Java-based technologies. The app will have two main user interfaces, one for cab drivers and one for customers.

2.1 Cab Driver Interface:

The cab driver interface will allow cab drivers to sign up, fill in their details, and receive a unique QR code. The interface will also allow cab drivers to view their ride history and update their details as necessary.

2.2 Customer Interface:

The customer interface will allow customers to sign up, fill in their details, and search for cab drivers. The interface will allow customers to scan a cab driver's QR code to verify their identity and view their details. The interface will also allow customers to book a ride and provide feedback on the ride after it is completed.

3. Server-Side:

The server-side of the app will be developed using Java-based technologies. The server will be a cloud-based server and will use a database to store cab driver and customer details.

3.1 Database Schema:

The database schema will include tables for cab drivers and customers. The cab driver table will include fields for name, phone number, email, cab number, and a unique QR code. The customer table will include fields for name, phone number, email, and a trusted email.

3.2 RESTful API:

The server will expose a RESTful API that will be used by the client-side app. The API will include endpoints for cab drivers to sign up, update their details, and view their ride history. The API will also include endpoints for customers to sign up, search for cab drivers, scan QR codes, book rides, and provide feedback.

4. Non-Functional Requirements:

The app should be scalable and able to handle a large number of users. The app should also be secure and implement industry-standard security measures to ensure the safety and

security of both cab drivers and customers. The app should have a fast response time and be easy to use and navigate.

5. Future Enhancements:

Future enhancements for the app could include adding payment processing, allowing customers to schedule rides in advance, and integrating the app with third-party mapping services for real-time tracking of rides. Additionally, the app could expand to other regions and countries.

Frequently Asked Questions

1. What problem does this app solve?

This app solves the problem of verifying the authenticity and identity of cab drivers for customers who require a ride. By providing a unique QR code to each verified driver, the app ensures that only verified and authorized drivers are associated with each code. Customers can scan the QR code to receive driver details and have an added sense of security while taking a cab ride.

2. How does this app generate revenue?

This app can generate revenue through several ways, such as:

- Charging a commission fee to cab drivers for registering and becoming verified drivers in the app.
- Charging a subscription fee to cab drivers for access to the app and its features.
- Offering premium features to customers, such as access to premium verified drivers, for a fee.

3. What is the market opportunity for this app?

The market opportunity for this app is significant, as there is a growing demand for safe and reliable cab services in many regions around the world. Additionally, the rise of ride-sharing apps has shown that there is a large market for on-demand transportation services. The app can cater to customers who are looking for an extra layer of safety and security when taking a cab ride.

4. How does this app ensure the safety and security of users?

The app ensures the safety and security of users by verifying the identity and authenticity of cab drivers through a unique QR code. Additionally, the app stores the driver and customer details in a secure cloud-based database. The app also sends driver details to the customer's trusted email address for added safety and security. The app can further ensure safety and security by implementing regular security updates and following best practices in software engineering.

5. What is the competitive landscape for this app?

The competitive landscape for this app includes ride-sharing apps and traditional taxi services. However, this app differentiates itself by providing an added layer of safety and security for customers by verifying the identity and authenticity of cab drivers. Additionally, the app can target customers who are concerned about safety and reliability and are willing to pay extra for these features.

6. What is the target market for this app?

The target market for this app includes customers who are concerned about safety and reliability when taking a cab ride. This could include women, late-night commuters, and tourists, among others. Additionally, the app can target customers who are looking for an extra layer of safety and security when taking a cab ride, even if they have to pay extra for it.

7. How does the app handle disputes or complaints from customers?

The app can handle disputes or complaints from customers by providing a customer support system within the app. This system can include a dedicated customer support team to handle complaints and issues from customers. Additionally, the app can implement a rating and review system for drivers, which can provide valuable feedback to both drivers and customers. If a driver receives multiple complaints or negative reviews, the app can take appropriate action, including revoking their verification status.

8. What are the technical requirements for running this app?

The technical requirements for running this app include a cloud-based server, a database to store driver and customer details, and an internet connection. The app can be developed using Java-based technologies and can be deployed on Android devices. The app should be scalable and able to handle a large number of users, as well as regular security updates to ensure the safety and security of the app and its users.

Conclusion

In conclusion, Namma Saarathi Verification is an innovative and necessary addition to the ride-hailing industry. It addresses the critical issue of safety and security for customers, particularly women and the elderly. Its unique features and comprehensive verification process make it stand out from other ride-hailing apps.

By using technology to create a safe and secure ride-hailing experience for all, Namma Saarathi Verification is leading the way towards a safer future. It is a shining example of how technology can be harnessed for the greater good, and it has the potential to transform the ride-hailing industry for the better.

Overall, Namma Saarathi Verification is a much-needed solution to a significant problem. It provides customers with a sense of security and ensures that they have a worry-free ride every time. With its user-friendly interface and focus on safety, it has the potential to become the go-to ride-hailing app for customers in Bangalore and beyond and upon implementing, this application can ensure the safety of the rides booked by a customer without an application.