Online Toll Notification System

Software Requirements Specification

Version 1.0

Submitted in Partial Fulfillment for the Award of Degree of Bachelor of Technology in Information Technology from Rajasthan Technical University, Kota

MENTOR:

Dr. Sanwta Ram Dogiwal

(Associate Professor, Dept. of Information Technology)

COORDINATOR:

Dr. Priyanka Yadav

(Assistant Professor, Dept. of Information Technology)

SUBMITTED BY:

Harshit Kumar Jain (21ESKIT053)

Himansh Singh (21ESKIT055)

Chirag (21ESKIT036)

DEPARTMENT OF INFORMATION TECHNOLOGY

SWAMI KESHWANAND INSTITUTE OF TECHNOLOGY, MANAGEMENT & GRAMOTHAN

Ramnagaria (Jagatpura), Jaipur (Dept. of Information Technology)**– 302017**

SESSION 2024-25

Revision History

Date	Version	Description	Author
05/Dec/24	1.0	Initial version of the SRS created	Harshit Kumar Jain

Table of Contents

1. Introduction

The Online Toll Notification System is designed to modernize and streamline the toll collection process, leveraging advanced technologies like geofencing, secure payment gateways, and real-time notifications. This document provides a comprehensive overview of the project, detailing its purpose, scope, terminology, references, and the technologies involved. The system aims to address inefficiencies in current toll payment methods by automating processes and enhancing user experience.

1.1 Purpose

The purpose of the Online Toll Notification System is to:

- Automate toll payments, reducing the need for manual interventions at toll booths.
- Provide real-time notifications to users as they approach toll plazas, enabling them to prepare for payments.
- Facilitate secure and quick transactions using digital payment methods such as mobile wallets and credit cards.
- Issue electronic receipts embedded with QR/barcodes for easy verification, ensuring transparency and record-keeping for both users and toll operators.

By addressing these aspects, the system aims to enhance efficiency at toll plazas, reduce traffic congestion, and improve the overall user experience.

1.2 Scope

The system is envisioned to offer a comprehensive solution for toll management with the following key features:

1. **Proximity Notifications**:
    - Users receive alerts on their mobile devices when they are within 1–2 km of a toll plaza.

    - Notifications include the toll plaza name, applicable fee, and payment options.
2. **Digital Payments**:
    - Integration with mobile wallets (e.g., Google Pay, Paytm) and credit cards to ensure seamless transactions.
    - Secure processing of payments with real-time success/failure feedback.
3. **Electronic Receipts**:
    - Automatic generation of receipts containing transaction details.
    - QR/barcode integration for efficient verification at toll booths.
4. **Toll Plaza Directory**:
    - A searchable list of toll plazas, providing information such as location, fees, and available services.

Outcomes of the System:

- **Streamlined Toll Management**: Automating payment processes reduces manual overhead for operators.
- **Reduced Waiting Times**: Efficient digital payments alleviate congestion at toll booths.
- **Improved Record-Keeping**: Electronic receipts ensure accurate and transparent record management.

1.3 Definitions, Acronyms, and Abbreviations

To ensure clarity and avoid ambiguity, the following terms and abbreviations are used throughout this document:

- **GPS**: *Global Positioning System* - A satellite-based navigation system used for determining user location.
- **QR Code**: *Quick Response Code* - A type of barcode that stores information, scannable by toll operators for receipt verification.
- **REST API**: *Representational State Transfer Application Programming Interface* - A standard protocol for backend communication.

- **Geofencing**: A technology that defines virtual boundaries around real-world geographic areas, triggering actions when users enter or exit these boundaries.

1.4 References

The following resources were consulted in designing and planning the Online Toll Notification System:

1. **Ministry of Road Transport**: Toll Policy Documentation providing standards and guidelines for toll management.
2. **Firebase Cloud Messaging**: Guides and documentation for implementing push notifications.
3. **Stripe and PayPal APIs**: Documentation for integrating secure and reliable payment gateways.
4. **QR Code Libraries**: Resources for embedding QR/barcodes in receipts for easy scanning and verification.

1.5 Technologies to be Used

The Online Toll Notification System utilizes a combination of modern technologies to achieve its goals:

- **Frontend Development**:
  - *React.js*: Framework for building a responsive and dynamic web interface.
  - *React Native*: Framework for developing cross-platform mobile applications.
- **Backend Development**:
  - *Node.js with Express*: Provides RESTful API services for managing user interactions and transactions.
- **Database Management**:
  - *MongoDB*: A NoSQL database for storing user, toll plaza, and transaction information dynamically.
- **Additional Tools**:

- *Firebase*: Used for delivering real-time notifications to users' devices.
- *Stripe*: A secure payment gateway for handling transactions.
- *QR Code Libraries*: Tools for generating and managing QR/barcodes within electronic receipts.

1.6 Overview

This document outlines the Online Toll Notification System in detail, presenting a roadmap for its development and deployment. Key areas covered include:

1. **Requirements**: Functional and non-functional specifications defining system capabilities.
2. **Architecture**: An overview of the system's client-server architecture, communication interfaces, and database design.
3. **Implementation**: Insights into the tools, frameworks, and technologies used in development.
4. **Design Models**: Data flow diagrams, E-R diagrams, and use-case models for visualizing system interactions.
5. **Future Enhancements**: Possibilities for integrating AI, multilingual support, and traffic management capabilities.

By ensuring clear and concise documentation, this SRS provides a solid foundation for all stakeholders to understand, develop, and evaluate the project effectively.

2. Literature Survey

This section provides an overview of the current landscape of toll payment systems, highlighting existing methodologies, identifying gaps, and presenting a comparative analysis to justify the need for the proposed system.

2.1 Review of Related Work

The current toll collection systems predominantly rely on two approaches:

1. **Manual Methods**:
    - Drivers pay toll fees at the plaza by handing cash or cards to toll operators.
    - This method leads to delays, especially during peak hours, due to the time required for processing transactions and issuing receipts.
2. **RFID-Based Systems**:
    - Radio-Frequency Identification (RFID) tags are used for automatic toll deduction from linked accounts.
    - These systems require users to install RFID tags on their vehicles and pre-load accounts, which can be inconvenient for occasional toll users.

While RFID systems reduce manual intervention, they lack flexibility and widespread adoption due to the need for specific hardware. Furthermore, very few existing solutions provide:

- **Location-Based Notifications**: Alerting users about upcoming tolls dynamically based on their GPS location.
- **Digital Receipts**: Instant generation of electronic receipts with QR or barcodes for easy scanning and future reference.

2.2 Knowledge Gaps

The limitations of current toll management systems reveal several gaps that the proposed solution aims to address:

1. **Geofencing-Based Alerts**:
    - Existing systems do not utilize geofencing technology to detect and notify users about approaching toll plazas.
    - A real-time alert system could improve user preparedness, reduce delays, and enhance overall user satisfaction.
2. **User-Friendly Payment Options**:
    - Mobile payments, which are widely adopted for other transactions, are rarely supported in existing toll systems.
    - Many systems rely on specialized RFID accounts or manual payment methods, limiting convenience.
3. **Dynamic and Transparent Information**:
    - Users often lack access to dynamic toll data, such as fees, location details, and available services at toll plazas.

2.3 Comparative Analysis

The following table compares features of existing toll systems with the proposed solution:

Feature	**Current Systems**	**Proposed System**
Manual Payment	Predominantly used	Completely eliminated
RFID Support	Widely adopted	Optional

Feature	Current Systems	Proposed System
Real-Time Notifications	Not available	Enabled via geofencing
Mobile Payments	Rarely supported	Fully supported (e.g., Google Pay, Paytm)
Digital Receipts	Minimal implementation	Fully integrated with QR/barcodes

Analysis:

- Current systems are operational but lack flexibility and real-time adaptability.
- The proposed system modernizes toll payments, integrating advanced technologies like geofencing and digital wallets to address user pain points effectively.

2.4 Summary

The **Online Toll Notification System** bridges critical gaps in existing toll management practices by introducing features such as:

- Real-time proximity alerts using GPS and geofencing.
- Seamless integration with widely used payment methods like mobile wallets and credit cards.
- Automated generation of electronic receipts with QR/barcodes, ensuring transparency and efficiency.

3. Specific Requirements

This section outlines the detailed functional and non-functional requirements of the Online Toll Notification System, along with its hardware, software, and supplementary needs. It ensures the system is designed to meet both user and operational expectations.

3.1 Functional Requirements

The system is designed to provide seamless functionality across notifications, payments, receipt generation, and toll data management:

Notifications

- The system will detect a user's proximity to a toll plaza (within 2 km) using geofencing technology and GPS tracking.
- Notifications will provide essential information, including:
    - Toll plaza name.
    - Applicable fee.
    - Payment options.
- Notifications must dynamically update based on the user's real-time location.

Payments

- Support multiple payment methods, including:
    - **Credit Cards**: Integration with secure payment gateways like Stripe or PayPal.
    - **Mobile Wallets**: Compatibility with Google Pay, Apple Pay, and other popular wallets.
- Ensure real-time transaction processing, with immediate feedback on payment success or failure.
- Automatically handle payment errors or retries.

Receipts

- Generate electronic receipts upon successful payment.
- Each receipt will include:
    - Transaction ID.
    - Payment amount.
    - Toll plaza name.
    - QR/Barcode for scanning and verification at toll booths.
- Store receipts securely in a user-accessible database for future reference.

Toll Data

- Provide real-time updates on toll plaza information, such as:
    - Toll fee structure.
    - Available facilities/services at the plaza.
    - Traffic conditions, if applicable.

3.2 Non-Functional Requirements

To ensure system reliability and efficiency, the following non-functional requirements must be met:

Performance

- Notifications must be delivered within 3 seconds of proximity detection.
- Payment processing should not exceed 5 seconds.

Security

- Encrypt all sensitive user data, including payment details and personal information, using **SSL/TLS protocols**.
- Implement secure authentication methods (e.g., OAuth) for user accounts.

Scalability

- The system must support up to **1 million concurrent users** during peak usage times.

- Maintain performance with minimal latency under high traffic loads.

3.3 Hardware Requirements

The system requires the following hardware capabilities for effective operation:

- **Smartphones**:
    - Devices equipped with GPS functionality for geofencing and real-time tracking.
    - Internet connectivity to enable notifications, payments, and data retrieval.

3.4 Software Requirements

The system must operate on the following software platforms:

- **Mobile Platforms**:
    - Android (version 9.0 or later).
    - iOS (version 12.0 or later).
- **Server Environments**:
    - **Windows Server** or **Linux distributions** (e.g., Ubuntu) for hosting backend services.
- **Tools and Frameworks**:
    - Node.js and Express for server-side logic.
    - MongoDB for database management.
    - Firebase for real-time notifications.

3.5 Agile Methodology

The development process will follow an Agile approach with the following key practices:

- **Weekly Sprints**:
    - Development tasks will be planned and executed in weekly cycles, with regular progress reviews.
- **Stakeholder Involvement**:

- Stakeholders will participate in sprint reviews to provide feedback and validate progress.

- **Continuous Integration/Continuous Deployment (CI/CD)**:
  - Automated tools will be used for building, testing, and deploying code.

3.6 Business Process Model

The Online Toll Notification System follows a straightforward business process:

1. **Registration**:
   - Users create an account with basic details, such as name, vehicle information, and preferred payment methods.
2. **Notification**:
   - Users receive real-time alerts as they approach toll plazas.
3. **Payment**:
   - Users select a payment method and complete the transaction through the app.
4. **Receipt**:
   - An electronic receipt is generated and stored.
5. **Verification**:
   - The QR/barcode from the receipt is scanned at the toll booth for final verification.

3.7 Supplementary Requirements

The system includes additional features to enhance usability and user experience:

- **Regional Language Support**:
  - Provide a multilingual interface to cater to users from different regions.
  - Include text-to-speech features for accessibility.
- **Alternate Routes**:
  - Offer route suggestions to users based on toll costs, traffic conditions, distance.
  - Integrate with navigation apps for seamless redirection.

4. System Architecture

The **System Architecture** describes the overall design of the Online Toll Notification System. It includes the interaction between various components like the **Client**, **Server**, **Database**, and **Communication Interfaces**. The architecture follows a **Client-Server Model** and uses **REST APIs** and **Firebase Cloud Messaging** for communication.

4.1 Client-Server Architecture

The system is designed with a **client-server architecture** that ensures modularity, scalability, and easy interaction between different system components.

Client Side:

- **Mobile Apps**: Developed using **React Native**, providing cross-platform support for **Android** and **iOS**.
- **Web Application**: Developed using **React.js**, optimized for both desktop and mobile web browsers.

The client is responsible for:

- Sending user location and payment requests to the server.
- Displaying real-time notifications and updates (such as toll fees, receipts).
- Handling user interactions, including payment and receipt scanning.

Server Side:

- **Backend Server**: Built using **Node.js** with **Express.js** to handle requests and process data.
- **APIs**: The server exposes RESTful APIs to interact with the client and other services.
    - **REST APIs** handle user authentication, toll data retrieval, payment processing, and receipt generation.

The server is responsible for:

- Processing requests sent from the client.
- Handling logic for geofencing, payment transactions, and toll plaza information.
- Communicating with the database and other services.

Database:

- **MongoDB**: A NoSQL database used for dynamic and scalable storage.
    - **User Data**: Information about users, their preferences, and payment history.
    - **Toll Data**: Information about toll plazas, fees, and services.
    - **Transaction Data**: Records of user payments and receipts.

Communication flow:

- The client sends a request to the backend server (via REST APIs) for user actions, such as initiating a payment, fetching toll data, or receiving notifications.
- The server processes the data, fetches required information from MongoDB, and returns the response to the client.

4.2 Communication Interfaces

Effective communication interfaces are essential for seamless interaction between the client and server. The system uses **REST APIs** for client-server communication and **Firebase Cloud Messaging (FCM)** for real-time notifications.

REST APIs:

- Used for communication between the client (mobile/web) and the server.
- Key functionalities:
    - **User Authentication**: Handling user logins and registrations.
    - **Toll Plaza Data**: Retrieving data like toll fees, locations, and available services.

- **Payment Processing**: Sending payment data to external services (e.g., Stripe, PayPal) for processing.
- **Receipt Generation**: Providing the generated receipt after payment confirmation.

Firebase Cloud Messaging:

- Used for real-time push notifications.
- Sends **proximity-based notifications** to users when they are near a toll plaza (detected via geofencing).
- Real-time updates (payment success, error messages) are communicated to the client using FCM.

System Architecture Diagram

The following diagram illustrates the **Client-Server Architecture** and how the components interact.

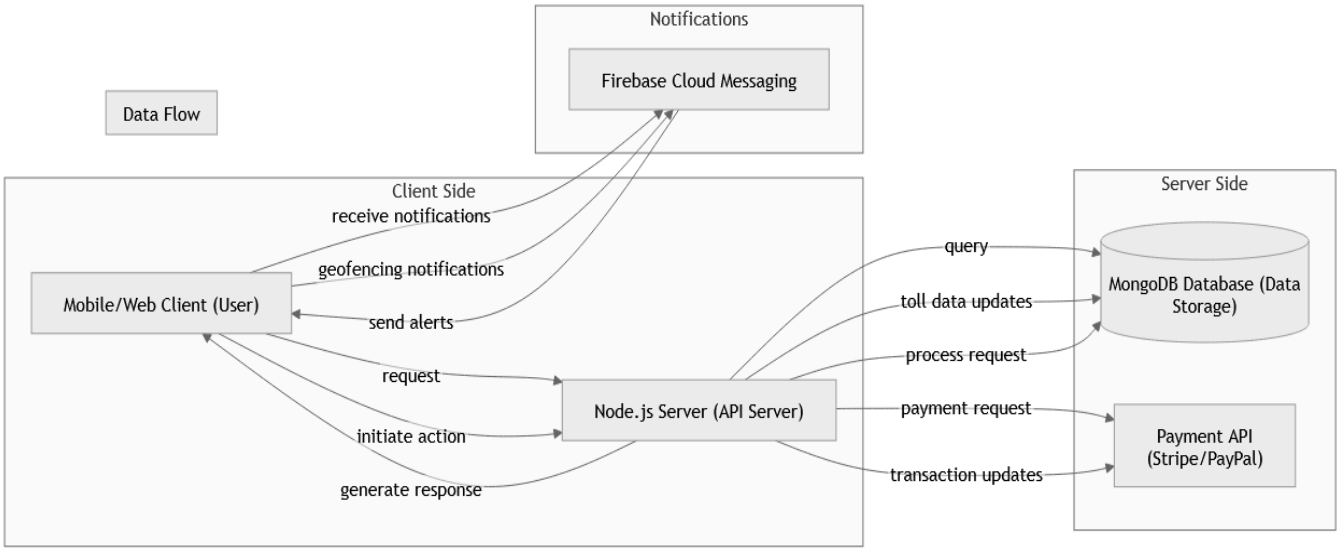

Fig. System Architecture Diagram for Online Toll Notification System

Key Components in the Diagram:

1. **Mobile/Web Client**: The user interacts with the system through a mobile app or web interface. It sends requests to the backend and displays notifications and data.
2. **Node.js Server**: The backend logic is handled by the server, which interacts with the database and payment APIs.
3. **MongoDB Database**: Stores user data, toll plaza details, transaction history, and other essential information.
4. **Firebase Cloud Messaging**: Handles real-time notifications, sending alerts to users based on proximity to toll plazas.
5. **Payment API**: Payment gateways like Stripe or PayPal are used to process user payments.

System Communication Flow

The communication flow between the client and server is as follows:

1. **User Initiates Action**:
   The user opens the mobile app or website and initiates an action (e.g., checking toll fees, making a payment).
2. **Client Sends Request**:
   The client sends a request to the server via **REST API** (e.g., POST /payment, GET /toll-plazas).
3. **Server Processes Request**:
   The server processes the request by querying the **MongoDB database** for toll plaza information or processing payment details.
4. **Payment Processing**:
   If the request involves a payment, the server sends payment details to an external payment provider (e.g., Stripe or PayPal) via an API.
5. **Generate Response**:
   The server generates a response (e.g., confirmation of payment, toll fee details) and sends it back to the client.

6. **Real-Time Notifications**:

   If the user is approaching a toll plaza, **Firebase Cloud Messaging** sends a push notification to the user based on their location.

Advantages of the Architecture:

1. **Scalability**:

   The client-server model allows scaling of each component independently (e.g., adding more servers or database replicas as user demand increases).
2. **Flexibility**:

   The use of REST APIs and Firebase allows for easy integration of new features or third-party services without major changes to the architecture.
3. **Reliability**:

   Node.js's event-driven model and MongoDB's flexible storage ensure high availability and minimal latency in processing requests.
4. **Real-Time Interaction**:

   Firebase Cloud Messaging enables the system to send instant notifications, improving user experience with real-time alerts and updates.

5. Design and Implementation

This section outlines the key components involved in the **design and implementation** of the Online Toll Notification System. It explains product features, diagrams to visualize system interactions, and outlines assumptions and dependencies that impact the system's functionality.

5.1 Product Features

Geofencing-based Alerts

- The system uses **geofencing technology** to monitor the user's location in real-time.

- When the user is within a defined range (typically 1-2 km) of a toll plaza, the system triggers a notification alerting the user about the approaching toll, its fee, and payment options.
- This proactive approach ensures the user is always prepared for the toll fee and can make payments without delay.

QR-based Receipt Scanning

- After a successful payment, the system generates a **QR code** embedded in the receipt.
- This QR code contains essential transaction details, which can be scanned at the toll booth for verification.
- The user can access their receipt via the mobile app or web interface at any time for future reference or disputes.

5.2 Data Flow Diagram (DFD)

The **Data Flow Diagram (DFD)** illustrates how data moves through the system, from the user receiving a notification to completing the payment.

- **User Device (App/Web)**: The user interacts with the mobile app or web interface to initiate actions like receiving notifications or making payments.
- **Node.js Server**: Handles API requests from the client, processes payments, interacts with the database, and sends push notifications via Firebase.
- **Payment Gateway**: Integrates with third-party services like Stripe or PayPal for secure payment processing.
- **Firebase Cloud Messaging**: Handles real-time notifications, alerting the user as they approach the toll plaza.
- **MongoDB Database**: Stores user data, transaction records, toll plaza information, and receipts.

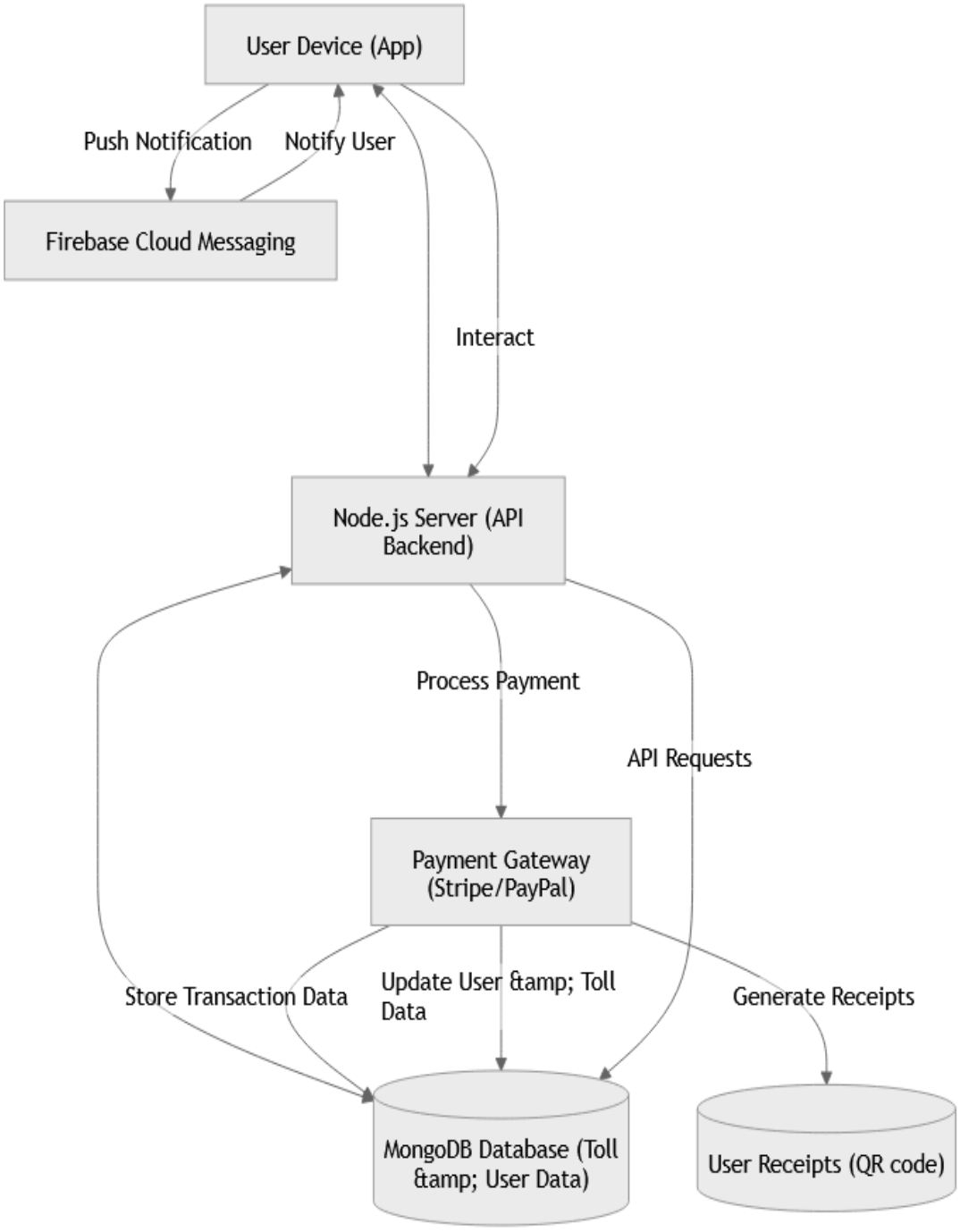

Fig. Data Flow Diagram for Online Toll Notification System

5.3 Entity-Relationship Diagram (E-R)

The **Entity-Relationship Diagram (E-R)** outlines the primary entities within the system and how they relate to one another. Here's a simplified diagram of the key entities:

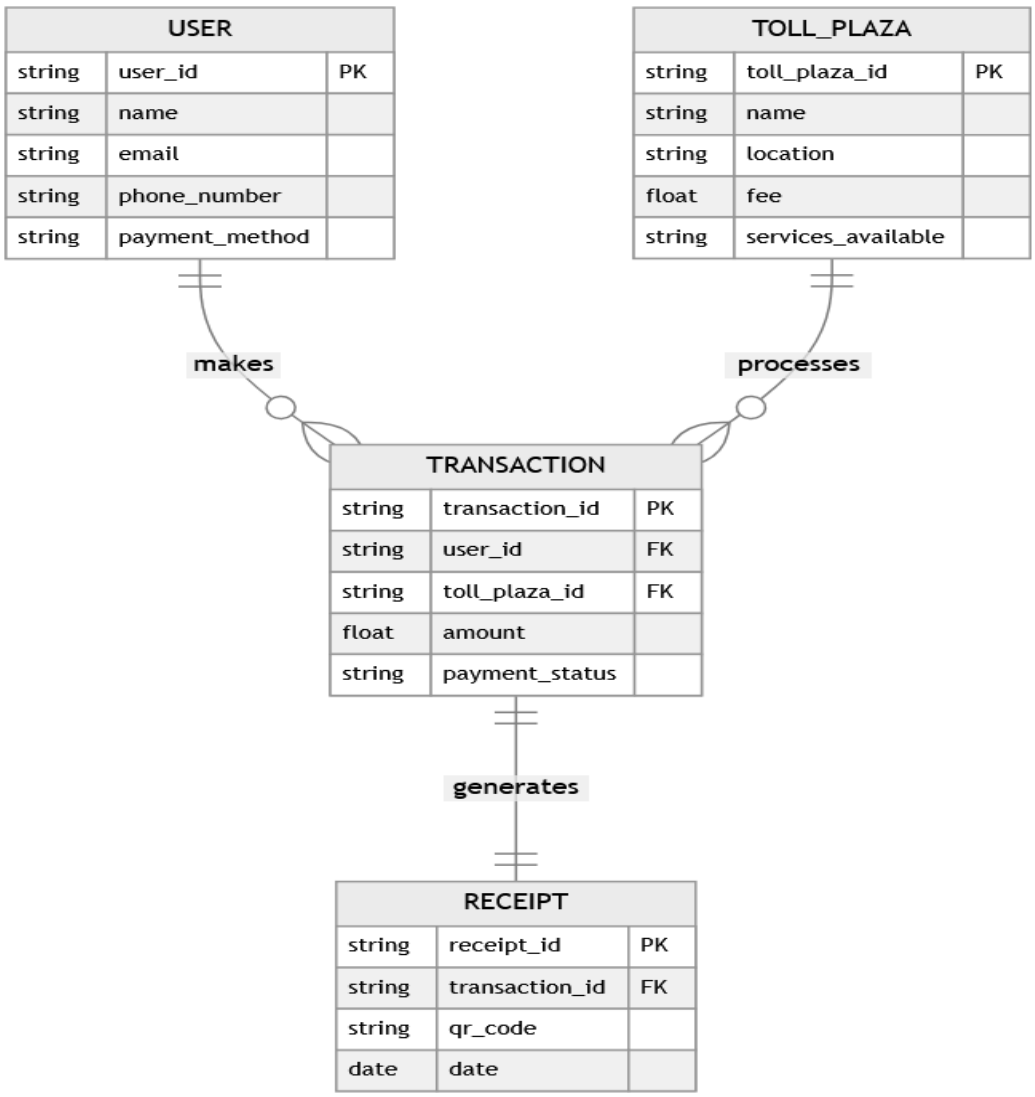

Fig. Entity Relationship Diagram for Online Toll Notification System

- **User**: Represents the user interacting with the system. Includes attributes like user ID, email, and payment method.
- **Toll Plaza**: Contains details about each toll plaza, including its location, fee, and available services.
- **Transaction**: Represents a payment made by the user, linking the user to the toll plaza with transaction details like amount and payment status.
- **Receipt**: Stores the electronic receipt, including the QR code for verification at the toll booth.

5.4 Class Diagram

The **Class Diagram** shows the main classes involved in the system and how they interact. Below is a high-level class diagram:

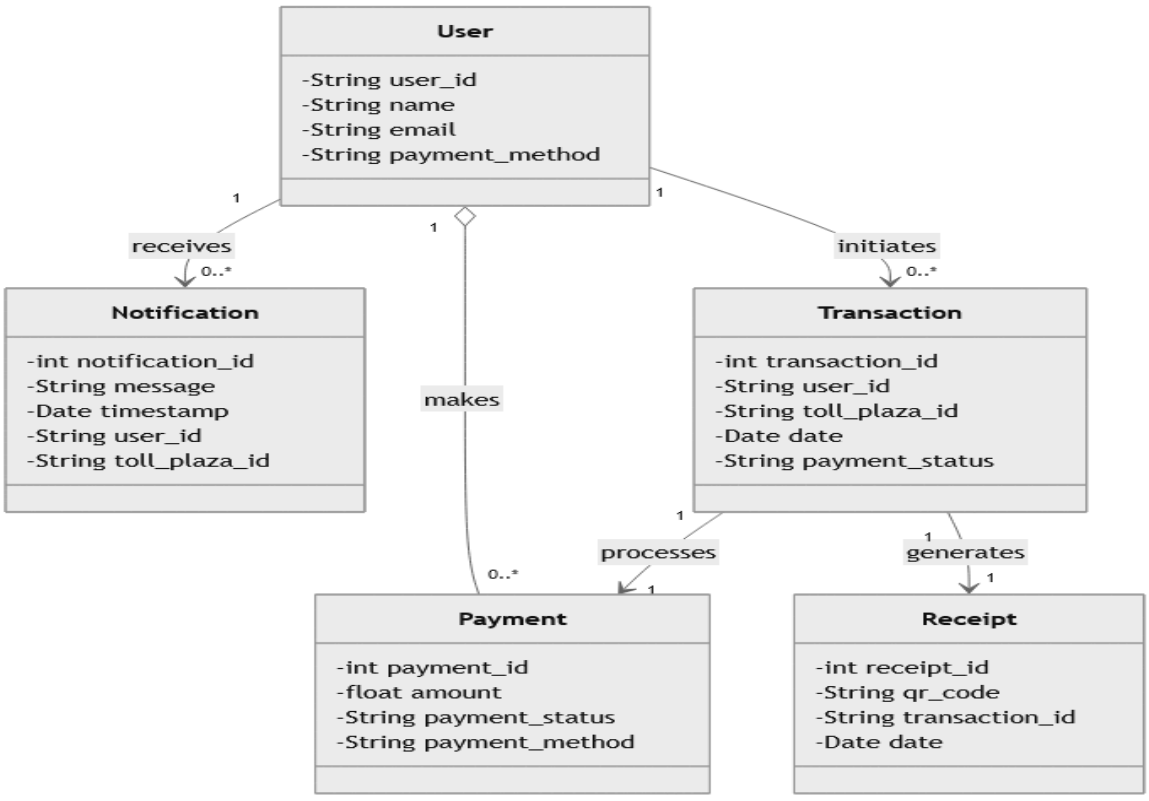

Fig. Class Diagram for Online Toll Notification System

- **User**: Contains details about the user, such as their ID, name, and payment method.
- **Payment**: Represents the details of a payment made, including amount, status, and method.
- **Transaction**: Stores information about each toll transaction, linked to a user and a toll plaza.
- **Receipt**: Stores the generated receipt with the associated QR code for payment verification.

5.5 Use-Case Model Survey

The **Use-Case Model** outlines the different interactions between the user and the system. Below is a summary of key **actors** and **scenarios**.

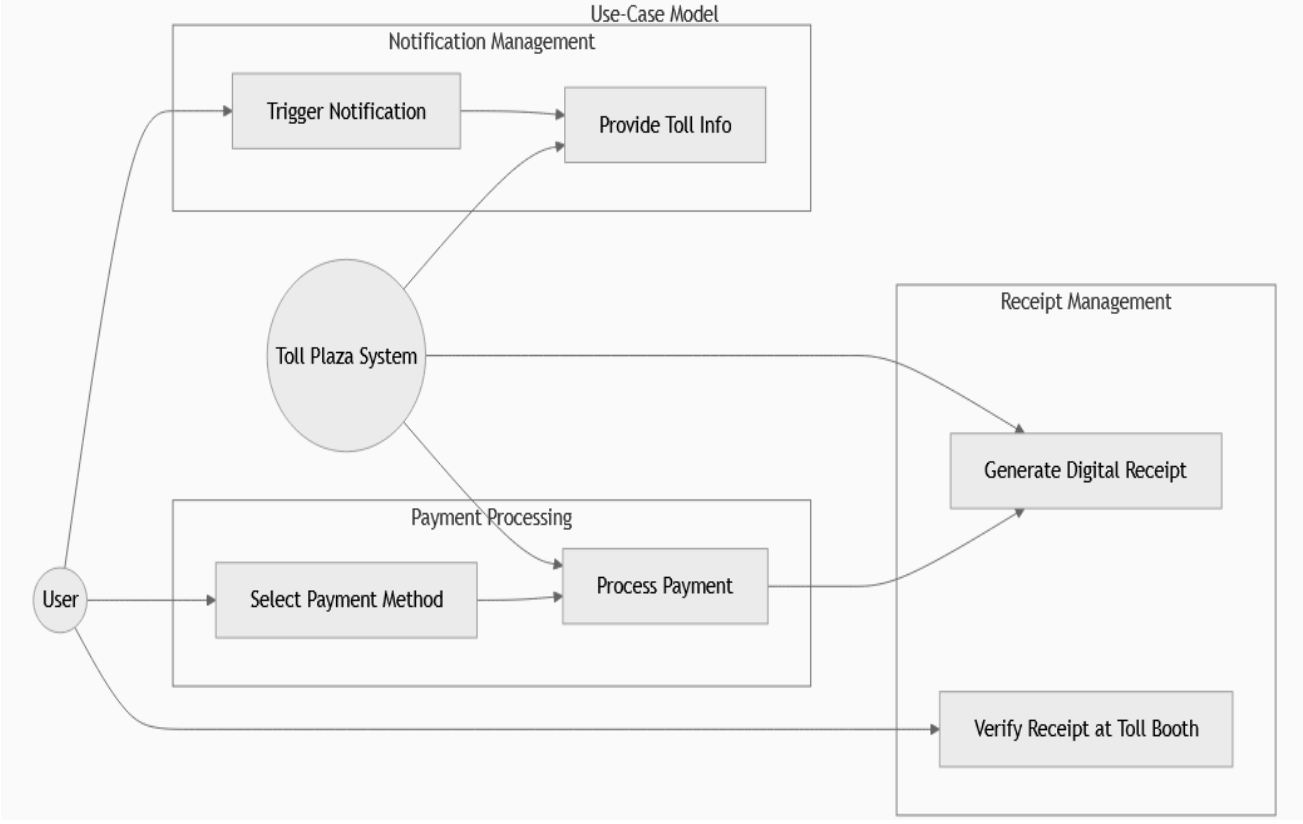

Fig. Use-Case Diagram for Online Toll Notification System

Actors:

1. **User**: The primary actor interacting with the system to make payments, receive notifications, and view receipts.
2. **Toll Plaza System**: The backend system that handles toll data, user transactions, and notifications.

Scenarios:

1. **User Receives Notification**:
    - The system triggers a proximity-based notification (e.g., 2 km before reaching the toll).
    - The notification provides the toll plaza name, fee, and payment options.
2. **User Pays Toll**:
    - The user selects a payment method (credit card or mobile wallet).
    - The system processes the payment through an external gateway (e.g., Stripe, PayPal).
3. **User Verifies Receipt**:
    - After payment, the user receives a digital receipt with a QR code.
    - The receipt can be scanned at the toll booth for verification.

5.6 Behavior Diagrams

The **Behavior Diagrams** illustrate the system's flow of actions, such as notifications and payment processes.

Sequence Diagram for Notification:

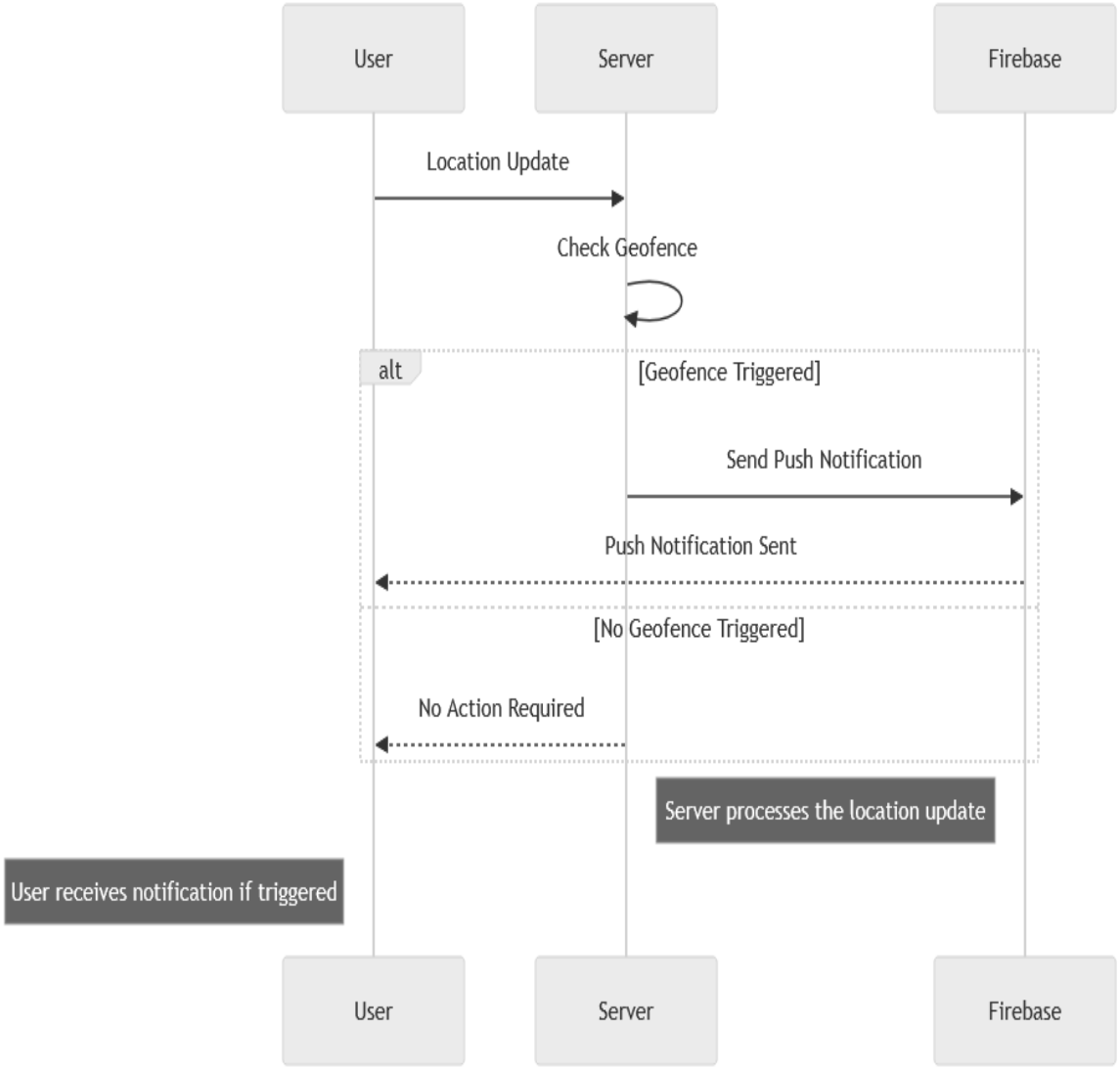

Fig. Sequence Diagram for Notification

Sequence Diagram for Payment:

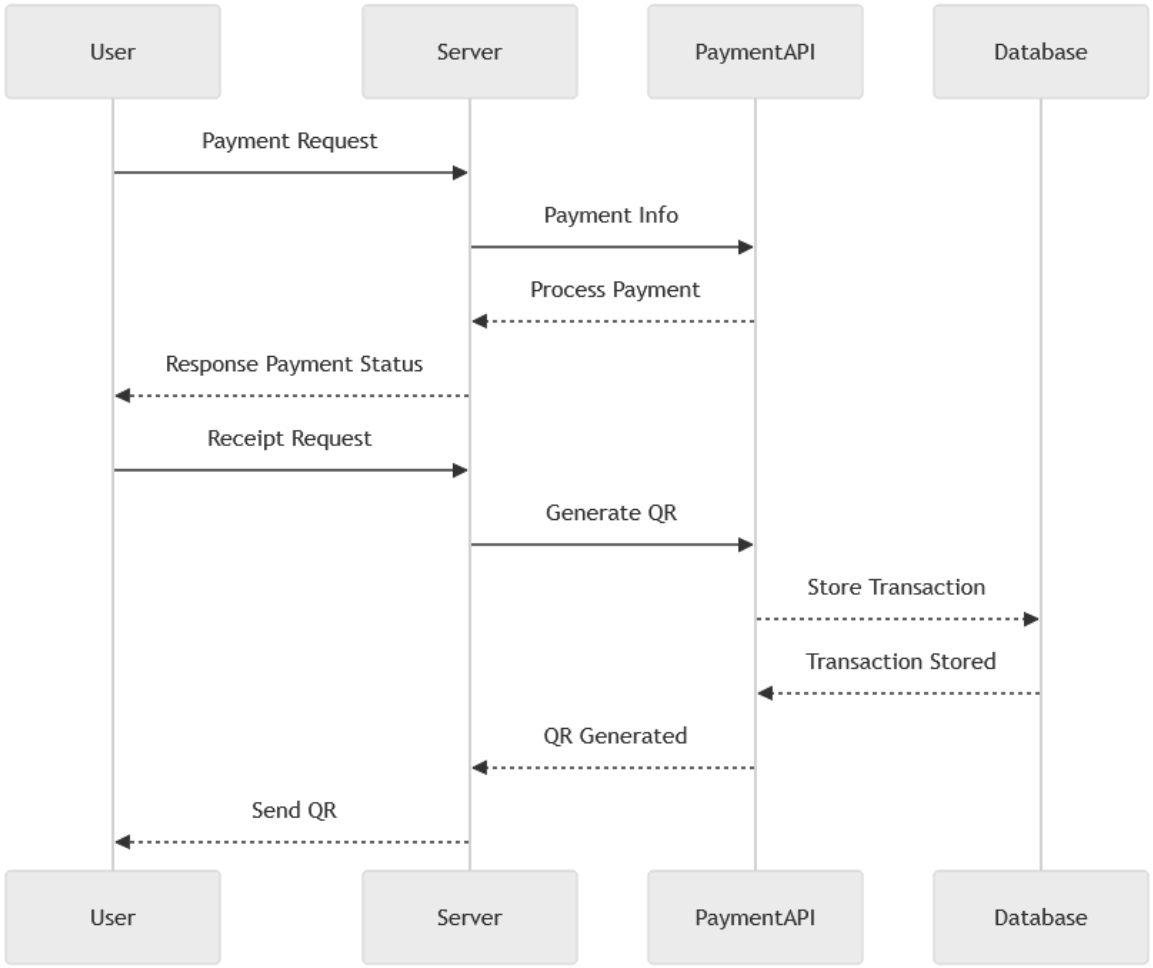

Fig. Sequence Diagram for Payment

These diagrams represent how the system behaves in response to user actions (notifications, payments).

5.8 Assumptions and Dependencies

- **GPS Signal**: The system depends on the availability of a stable GPS signal to accurately detect the user's proximity to toll plazas for geofencing.
- **Payment Gateways**: The system relies on third-party payment services (e.g., Stripe, PayPal) for processing payments. These services must be available and functioning correctly for payment transactions to succeed.
- **External APIs**: Integration with external services (e.g., Firebase for push notifications, Stripe for payments) is crucial for smooth operation.

6. Supporting Information

This section provides additional materials that help stakeholders understand the user interface design and the system's receipt generation feature. It includes mockups of the **user interface (UI)**, as well as sample **QR codes** for receipt verification.

Mockups of User Interface

Mockups provide a visual representation of the system’s **user interface (UI)**, helping stakeholders visualize the user experience at different stages of interaction with the system. Below are some key mockups:

1. Mobile App – Notification Screen

- **Notification Alert**: This screen is displayed when a user approaches a toll plaza, showing the toll fee, plaza name, and payment options.

2. Mobile App – Payment Screen

- **Payment Screen**: After receiving the notification, users can proceed to make a payment using either a mobile wallet or credit card.

3. Mobile App – Receipt Screen

- **Receipt Screen**: After the payment is processed, the user is shown a digital receipt with a QR code that can be scanned at the toll booth.

4. Web Application – Toll Plaza List

- **Toll Plaza Directory**: Displays a list of nearby toll plazas, including their name, location, and fee.

How QR Code Works:

- **QR Code** encodes essential payment details like transaction ID, user info, toll plaza ID, and fee paid.
- At the toll booth, the **toll booth operator** scans the QR code using a scanner or mobile app to verify the payment.

Additional Supporting Information

- **User Interface Prototypes**: High-fidelity UI prototypes (e.g., using tools like **Figma** or **Adobe XD**) can be created to visually represent the flow of user interaction across mobile and web platforms.
- **Testing Results**: Screenshots or reports from usability testing can be included to show how the system was evaluated and refined for a better user experience.

7. Conclusion and Future Scope

Conclusion

The **Online Toll Notification System** successfully addresses several critical issues in the current toll management landscape. By automating the payment process, notifying users in real-time, and providing digital receipts, the system enhances **efficiency** and **user convenience** while reducing congestion at toll booths. The system's design ensures **transparency**, allowing users to track their payments and verify receipts digitally. Furthermore, the system offers scalability and flexibility, positioning it as a future-proof solution to evolving toll management needs.

Key takeaways include:

- Streamlined **toll management** that reduces operational overhead.
- Improved **user experience** through easy-to-use mobile apps and notifications.
- **Transparent** transaction records and receipts that can be stored and verified.
- **Scalability** to handle a large number of users and data.

Future Scope

The Online Toll Notification System has strong potential for growth and future enhancements, particularly in the areas of **artificial intelligence (AI)** and **multilingual support**. Below are some proposed improvements and extensions to the system:

1. **AI for Traffic Predictions**:
    - **AI-driven traffic management** can predict congestion at toll plazas based on historical data, weather conditions, and user traffic patterns.
    - **Real-time updates** on traffic flow can allow users to be redirected to less congested routes or warn them about delays before they approach a toll plaza.
2. **Multilingual Support**:
    - To cater to a wider audience, the system could support **multiple languages**, making it more accessible for users from different regions and backgrounds.
    - **Language preferences** can be set during the registration process, and the system will adjust to the user's preferred language for all interactions, notifications, and receipts.
3. **Integration with Vehicle Navigation Systems**:
    - **Seamless integration** with GPS navigation apps (e.g., Google Maps, Waze) could enable users to receive toll notifications directly within their navigation system.
    - This would allow users to make timely decisions about route choices without leaving their navigation interface.
4. **Enhanced Security Features**:

- **Biometric authentication** (e.g., facial recognition or fingerprint scanning) for payments to increase security and ease of use.
- Enhanced **encryption protocols** for financial transactions to ensure user privacy and data protection.

5. **Sustainability Features**:
   - Incorporating **eco-friendly toll payment options** (e.g., incentives for using electric vehicles or offering discounted rates for environmentally-friendly vehicles).
   - Data-driven insights could be used to optimize toll fees based on **traffic volume** and **peak hours**, encouraging users to choose non-peak times.

8. Concerns / Queries / Doubts

While the system is designed to improve toll management significantly, there are a few **concerns** and **queries** that need to be addressed to ensure its robustness:

1. What are the fallback options for failed geofencing detections?

- **Geofencing Failures**: The system relies heavily on **GPS** and **geofencing** to detect user proximity to toll plazas. In cases where **GPS signals** are weak (e.g., in tunnels or remote areas), geofencing might fail to trigger notifications accurately.

Fallback Strategies:

- **Alternative Proximity Detection**: In case geofencing fails, the system can use **cell tower triangulation** or **Wi-Fi signals** (if available) as a backup to approximate the user's location.
- **User Input**: Prompt users to manually confirm their location if the system cannot determine proximity due to geofencing failure.
- **Scheduled Alerts**: If geofencing is unavailable, the system can send **pre-scheduled notifications** (e.g., a reminder that a toll plaza is ahead within a certain range).

- **Backup GPS Method**: When a signal failure occurs, the system could fall back on **offline maps** and location tracking based on device speed and direction.

2. How to ensure payment reliability during network outages?

Payment processing, especially when integrating third-party payment gateways (e.g., Stripe, PayPal), depends on a stable **internet connection**. In the event of network outages, there is a risk that users might face failed or delayed payments.

Strategies for Ensuring Payment Reliability:

- **Offline Payment Support**: Implement an **offline payment mode** where users can store payment details and complete the transaction once the network is restored. This could be useful in areas with intermittent connectivity.
- **Retry Mechanism**: Introduce an automatic **retry mechanism** that will attempt to process the payment several times during network recovery. The user will be notified of a successful transaction once the payment is processed.
- **Payment Confirmation**: Provide a **two-step verification** for critical payments. After processing the payment, a confirmation notification is sent via **SMS or email** to ensure users are aware of the transaction status, even if the app interface experiences delays.
- **Transaction History**: Allow users to view their **transaction history** offline and synchronize data once they reconnect to the network. This would help users keep track of payments and resolve any discrepancies.