**DEVELOPMENT OF GOCASHLESS: A CASHLESS PAYMENT SYSTEM FOR ZAMBIAN BUSES**

**BY**

LUPANDU MASUMBA

2021410277

A PROJECT REPORT SUBMITTED IN PARTIAL FULFUILMENT OF THE REQUIREMENT OF A BACHELOR OF COMPUTER SCIENCE DEGREE

**THE UNIVERSITYOF ZAMBIA**

**SCHOOL OF NATURAL SCIENCES**

**LUSAKA**

**OCTOBER 2025**

**DECLARATION**

I, LUPANDU MASUMBA do hereby declare that this report is my own original work and has not been submitted to any other college, institution or university other than the University of Zambia.

**Name: …………………………………………………………………………….**

**Sign: ………………………………………………………………………………**

**Date: ………………………………………………………………………………**

**APPROVAL**

This report, by LUPANDU MASUMBA has been approved as partial fulfilment of the requirements for the award of Bachelor of Computer Science Degree by the University of Zambia.

**Student**

**Name**: ………………………………………………

**Signature**:…………………………………………..

**Date**:………………………………………………...

**Supervisor**

**Name**:…………………………………………………

**Signature**:……………………………………………..

**Date**:…………………………………………………..

**ACKNOWLEDGEMENTS**

**ABSTRACT**

Public transportation in Zambia still relies predominantly on cash transactions, resulting in operational inefficiencies, revenue leakages, and security risks for passengers, conductors, and bus companies. The *GoCashless* system aims to address these challenges by introducing a secure, scalable, and user-friendly cashless payment platform tailored for Zambia’s public transport sector. Built on a microservices architecture using Spring Boot, the system supports seamless fare payments and efficient transaction management. It consists of two React Native mobile applications one for passengers to scan QR codes and make payments, and another for conductors to generate fares and receive confirmations as well as a Next.js web dashboard that enables bus companies to manage conductors and monitor transaction data. Core components include user management, route and fare handling, QR code generation, payment processing, transaction history, and real-time notifications. By leveraging RESTful APIs, secure data encryption, and token-based authentication, GoCashless ensures reliable and efficient payment workflows. The resulting system improves fare collection, enhances accountability, supports better revenue tracking, and contributes to the modernization of Zambia’s public transportation infrastructure through a fully digital ecosystem.

**TABLE OF CONTENTS**

**LIST OF TABLES**

**LIST OF FIGURES**

**1. INTRODUCTION**

**1.1 Background and Motivation**

The transportation sector in Zambia, particularly public transportation, heavily relies on cash-based transactions. This traditional payment method presents several challenges, including the risk of cash handling, revenue leakages for bus operators, and inconvenience for passengers who may not always have the exact change. The need for a modernized, cashless payment system is evident to enhance efficiency, transparency, and user experience. The Gocashless project aims to address these challenges by introducing a digital payment solution for public transportation in Zambia.

**1.2 Problem Statement**

The core problem is the inefficiency and lack of security in the current cash-based fare collection system in Zambia's public transport. This leads to revenue losses for bus companies, security risks for conductors, and a suboptimal experience for passengers. A digital payment system can mitigate these issues, but its implementation requires careful consideration of the local context, such as the prevalence of mobile money and the specific needs of the stakeholders (passengers, conductors, and bus companies).

**1.3 Aim and Objectives**

The aim of the Gocashless project is to design, develop, and implement a secure and user-friendly cashless payment system for public transportation in Zambia using QR code technology and mobile money.

The objectives of the project are:

* To develop a microservices-based backend to manage users, routes, fares, and payments.
* To create a web-based dashboard for admins to manage their fares, routes, and conductors.
* To build a mobile application for conductors to generate QR codes for fare payment.
* To design a mobile application for passengers to scan QR codes and make payments using their mobile money accounts.
* To integrate the system with a mobile money provider (Airtel Money) to process transactions.

1.4 Significance of the Project

The Gocashless system offers significant benefits to all stakeholders. For bus companies, it provides better revenue tracking and reduces leakages. For conductors, it enhances safety by minimizing the amount of cash they handle. For passengers, it offers a convenient and secure payment method. The project also contributes to the broader goal of financial inclusion and digitalization in Zambia.

**1.5 Scope and Limitations**

The scope of this project encompasses the design and development of the backend microservices, the web-based administrative dashboard, and the mobile applications for both conductors and passengers. The system will provide essential functionalities such as user registration and authentication, route and fare management, QR code generation, and the handling of payment transactions. Since access to a real mobile money payment API is not available, the payment component will be implemented as a simulated service to demonstrate the payment workflow and its integration within the system. This simulation will mimic the behavior of a real payment service to validate the system’s functionality and interactions. The project is limited to a controlled environment for demonstration and testing purposes and does not include a live deployment or integration with actual bus companies. As such, while the system architecture and core features are designed to support real-world deployment in Zambia’s public transportation sector, full-scale implementation and testing in a production environment remain beyond the scope of this project.

1.6 Report Organization

This report is organized into seven chapters. Chapter 1 introduces the project. Chapter 2 reviews related literature. Chapter 3 discusses system analysis and design. Chapter 4 details the implementation. Chapter 5 covers testing and evaluation. Chapter 6 provides the conclusion and recommendations, and Chapter 7 lists the references.

**2: LITERATURE REVIEW**

**2.1 Introduction**

The transition from cash-based to digital fare collection in public transport has become a global trend aimed at improving efficiency, transparency, and user convenience. In African cities, cash transactions still dominate, leading to inefficiencies, leakages, and security concerns. Various studies and innovations have explored cashless payment systems, mobile ticketing, and smart fare collection technologies to address these issues. This chapter reviews key literature on digital fare collection systems, QR code and NFC technologies, mobile ticketing applications, and the role of mobile money in enhancing financial inclusion, with a focus on their relevance to Zambia’s context and the GoCashless project.

**2.2 Innovations in Fare Collection Systems**

The World Bank’s SSATP (2022) study, Innovation in Fare Collection Systems for African Cities, highlights that most African public transport systems, including minibus taxis, rely on manual fare handling. This approach creates challenges in accountability and financial management. The report emphasizes the growing adoption of digital fare systems using mobile money, smart cards, and QR codes to streamline transactions and improve data-driven planning. Similarly, the Digital Van Service for Addis Ababa (ITDP Africa, 2022) demonstrates the potential of digital platforms in urban transport, where smart fare collection enhanced operational efficiency, passenger convenience, and revenue control.  
  
Aruho et al. (2025) in Cashless Fare Collection Systems Acceptability in the Paratransit/Minibus Industry found that adoption depends heavily on perceived ease of use, cost, and infrastructure readiness. Their study in East Africa revealed that operator resistance often stems from trust issues and system complexity rather than technological limitations—highlighting the importance of human-centered design in systems like GoCashless.

**2.3 Mobile Ticketing and QR-Based Systems**

Mobile ticketing applications have gained significant traction as they allow passengers to pay fares seamlessly using smartphones. Chen (2022) explored factors influencing the use of mobile ticketing applications and found that user trust, system reliability, and security play critical roles in acceptance. Bartin et al. (2018) proposed an Evaluation Framework for Mobile Ticketing Applications, underscoring usability, interoperability, and data protection as core factors for sustainable adoption.  
  
Tupare (2024) and the IJPREMS Smart Bus Ticketing System Using QR Code (2022) both showcased the effectiveness of QR-based systems in improving fare validation speed and reducing the need for physical tickets. These systems enable passengers to scan codes displayed on buses to make instant payments, ensuring transparency and accountability. Nakkala (2025) further advanced this concept in Digital QR Tickets: Smart Solution for Urban Mobility, emphasizing how QR codes support scalable and low-cost implementations suitable for developing regions.  
  
Additionally, A New Public Transport Payment Method Based on NFC and QR Code (ResearchGate, 2023) integrates Near Field Communication (NFC) and QR code technologies for flexibility and broader user access. For the GoCashless system, such hybrid designs could support future scalability to NFC-based payments alongside QR functionality.

**2.4 Mobile Money and Digital Payments in Zambia**

The success of any cashless fare system in Zambia is tightly linked to mobile money penetration. According to VoxDev (2023), mobile money has transformed financial access across Zambia but still faces challenges like transaction fees, network reliability, and user trust. The UNDP (2024) report on The Role of Digital Mobile Money in Catalyzing Financial Inclusion emphasizes that digital payment systems can enhance transparency, empower small businesses, and foster inclusion for the unbanked population—making mobile money integration a cornerstone for GoCashless.  
  
Koloseni and Mandari (2025), in their Tanzanian study Towards Sustainable Adoption: Investigating QR Codes Mobile Payment Continuance, noted that post-adoption factors—such as transaction security, perceived value, and consistent system updates—determine whether users continue to use cashless services. These insights are critical for GoCashless, which aims to ensure sustainable user engagement beyond the pilot stage.

**2.5 Data-Driven and Integrated Ticketing Systems**

Francis et al. (2023) examined the Potential of Electronic Ticketing Machine Data in Public Transport Planning and concluded that fare data can significantly inform transport planning, route optimization, and policy decisions. Similarly, Shimomba et al. (2025) proposed the Design and Development of an Integrated Online Bus Ticketing System, which aligns with GoCashless in offering centralized data management for multiple stakeholders.  
  
These studies reveal that data analytics capabilities embedded within ticketing systems enable operators and regulators to enhance operational efficiency, improve service quality, and plan infrastructure based on evidence.

**2.6 Identified Gaps and Research Opportunities**

While several systems have demonstrated technical feasibility, gaps remain in localization, interoperability, and inclusivity. Most reviewed systems were developed in urban contexts outside Zambia, with limited consideration of local digital ecosystems and user behavior. There is also a lack of integration between fare collection systems and mobile money platforms tailored for the Zambian market. Moreover, previous systems have often overlooked conductor-specific interfaces and real-time notifications, which are crucial in Zambia’s minibus ecosystem.  
  
The GoCashless project addresses these gaps by:  
- Localizing solutions for the Zambian context through integration with Airtel Money.  
- Introducing dual applications for conductors and passengers with real-time synchronization.  
- Leveraging microservices architecture to enhance scalability and maintainability.  
- Ensuring transparency and reducing leakages through secure QR-based payments and digital receipts.

**2.7 Summary**

In summary, previous literature shows a global and regional shift toward digital fare systems, emphasizing efficiency, security, and inclusivity. However, adoption challenges persist, especially in the African context, due to infrastructure, policy, and behavioral barriers. The GoCashless project builds upon these foundations by developing a context-specific, scalable, and secure system that integrates QR code-based payments with mobile money, thereby contributing to both transport modernization and financial inclusion in Zambia.

**Table 2.1: Summary of Reviewed Literature**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No | Name of Authors | Year | Title | Findings | Methods | Proposed Solutions | Gaps |
| 1 | SSATP / World Bank | 2022 | Innovation in Fare Collection Systems for African Cities | Identified inefficiencies in manual fare systems across Africa; emphasized need for integrated digital fare collection. | Case studies and comparative analysis across African cities. | Introduced framework for smart fare systems using mobile payments and data analytics. | Limited focus on local-level pilot implementation and end-user adoption. |
| 2 | ITDP Africa | 2022 | Digital Van Service for Addis Ababa | Demonstrated benefits of digital ticketing in improving revenue tracking and passenger convenience. | Pilot implementation and observational analysis. | Developed a digital van booking and fare management system. | Did not address interoperability with mobile money platforms. |
| 3 | C.C. Chen | 2022 | Exploring the Factors of Using Mobile Ticketing Applications | Found that user trust, reliability, and perceived ease of use determine mobile ticketing adoption. | Quantitative survey and regression analysis. | Proposed usability and security-focused design model. | Focused on developed economies, not African transport systems. |
| 4 | A.T. Aruho et al. | 2025 | Cashless Fare Collection Systems Acceptability in the Paratransit/Minibus Industry | Identified resistance due to trust issues and poor digital literacy among operators. | Mixed-method survey and interviews. | Recommended participatory design and operator sensitization. | Did not propose a concrete implementation framework. |
| 5 | J. Tupare | 2024 | Smart Public Transport Ticketing System Using QR Code | Demonstrated that QR code ticketing improves transaction speed and reduces fraud. | Prototype design and testing. | Developed a smart QR-based ticketing system. | Lacked integration with mobile wallets for payments. |
| 6 | IJPREMS | 2022 | Smart Bus Ticketing System Using QR Code | Showed that QR-based ticketing enhances fare collection accuracy. | System development approach using Android and PHP. | Provided low-cost ticket validation via QR codes. | Focused only on single-city implementation. |
| 7 | M. Nakkala | 2025 | Digital QR Tickets: Smart Solution for Urban Mobility | Found QR-based systems to be scalable and user-friendly for developing cities. | Empirical evaluation of digital ticketing apps. | Proposed smart QR system for public transport. | Did not analyze backend scalability or security. |
| 8 | ResearchGate Authors | 2023 | A New Public Transport Payment Method Based on NFC and QR Code | Demonstrated dual NFC–QR hybrid payment to improve inclusivity. | Experimental design. | Combined NFC and QR code technologies for flexibility. | Did not assess cost feasibility for African markets. |
| 9 | VoxDev | 2023 | Mobile Money in Zambia: Opportunities, Challenges and Policy Debates | Highlighted mobile money’s role in financial inclusion and its operational challenges. | Policy analysis. | Promoted integration of mobile money into public service payments. | Did not address transport-specific applications. |
| 10 | UNDP | 2024 | The Role of Digital Mobile Money in Catalyzing Financial Inclusion | Showed how mobile money accelerates inclusion and transparency. | Policy and impact review. | Advocated for cross-sector digital finance adoption. | Lacked technical design considerations for system integration. |
| 11 | Bartin et al. | 2018 | Evaluation Framework for Mobile Ticketing Applications in Public Transit | Defined metrics for usability, security, and data protection in mobile ticketing. | Framework development and simulation. | Created standard evaluation model for mobile ticketing apps. | Did not include modern QR or mobile money features. |
| 12 | Koloseni & Mandari | 2025 | Towards Sustainable Adoption: Investigating QR Codes Mobile Payment Continuance in Tanzania | Found transaction security and perceived value drive continued use of QR payments. | Longitudinal survey. | Suggested strategies for sustaining QR payment adoption. | Limited to urban Tanzanian context. |
| 13 | Francis et al. | 2023 | Potential of Electronic Ticketing Machine Data in Public Transport Planning | Highlighted use of ticketing data for transport planning and optimization. | Data analytics on ticketing datasets. | Proposed integration of e-ticketing data for planning. | Did not link with real-time mobile payment systems. |
| 14 | Shimomba et al. | 2025 | Design and Development of an Integrated Online Bus Ticketing System | Presented an integrated system for centralized bus ticketing and monitoring. | System design and prototyping. | Developed online ticketing with admin dashboards. | Did not address cashless or mobile money payment integration. |

**3: SYSTEM ANALYSIS AND DESIGN**

**3.1 Introduction**

This chapter presents the system analysis and design of the GoCashless system—a microservices-based, QR-code-enabled cashless payment solution for Zambia’s public transport sector. The system replaces manual cash handling with a secure, efficient, and traceable digital alternative. It emphasizes scalability, modularity, and user-centered interaction between passengers, conductors, and bus company administrators.

**3.2 Functional and Non-Functional Requirements**

**3.2.1 Functional Requirements**

The functional requirements define the specific operations that the system must perform to achieve its objectives:

User Management: Passengers, conductors, and administrators can register and log in securely with role-based access control.

Route and Fare Management: Administrators can create, update, and manage bus routes, stops, and fares.

Bus Stop Management: Each route contains multiple bus stops that define fare distances and pricing.

QR Code Generation: Conductors select a route and generate encrypted QR codes with fare and route information.

Payment Simulation: Passengers scan QR codes and initiate simulated payments using a dummy payment module instead of a live gateway.

Transaction Management: Both passenger and conductor apps receive simulated confirmation notifications and maintain transaction history.

Administrative Control: The web dashboard allows management of conductors, routes, and transactions in real time.

**3.2.2 Non-Functional Requirements**

|  |  |
| --- | --- |
| Requirement Type | Description |
| Performance | The system must process simulated transactions within 3 seconds. |
| Security | All communications use HTTPS and JWT for authentication. |
| Scalability | Architecture supports horizontal scaling via microservices. |
| Availability | Eureka and Gateway provide redundancy and fault tolerance. |
| Usability | Interfaces are intuitive and responsive on all devices. |
| Maintainability | Microservices can be independently updated and deployed. |
| Portability | React Native ensures cross-platform compatibility (Android/iOS). |
| Reliability | Transactions and user data are persistently stored in PostgreSQL. |

**3.3 Use Case Analysis**

This section defines the actors and primary use cases that drive the GoCashless system.

**3.3.1 System Actors**

Passenger: Uses the mobile app to scan QR codes, simulate payments, and view history.

Conductor: Generates QR codes and receives confirmation notifications.

Administrator: Manages routes, fares, conductors, and monitors transactions through the dashboard.

System (GoCashless): Coordinates authentication, QR generation, and payment simulation.

**3.3.2 Major Use Cases**

|  |  |  |
| --- | --- | --- |
| Actor | Use Case | Description |
| Passenger | Scan QR and Pay | Scans QR from conductor, initiates simulated payment using dummy payment module, and receives confirmation. |
| Passenger | View Transaction History | Views list of past simulated payments. |
| Conductor | Generate QR Code | Selects destination and generates a QR code with fare info. |
| Conductor | Receive Payment Notification | Receives payment confirmation after successful transaction simulation. |
| Admin | Manage Conductors and Routes | Adds or updates conductors, routes, and fares. |
| Admin | View Analytics Dashboard | Views reports on transactions and revenue trends. |

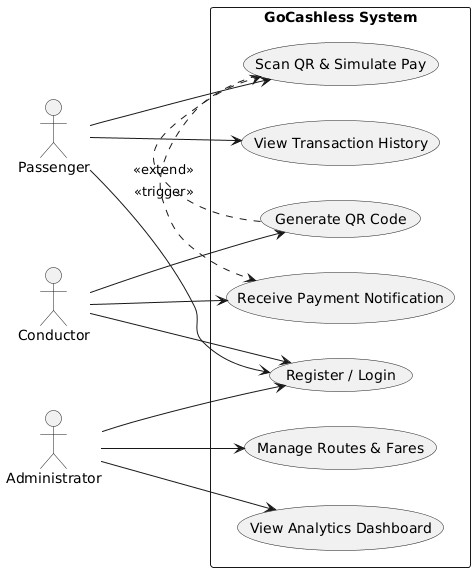


Figure 3.1: Use Case Diagram for GoCashless System.

**3.4 System Architecture Overview**

The GoCashless system adopts a microservices architecture for modularity and scalability. Each core function is implemented as an independent Spring Boot service registered under the Netflix Eureka discovery server. Requests are routed through a central API Gateway ensuring controlled access and service communication. Instead of using an actual payment gateway, a dummy payment module simulates payment workflows.

Frontend: React Native (mobile apps) and Next.js + Tailwind CSS (web dashboard).

Backend: Microservices built with Spring Boot and Spring Cloud.

Database: PostgreSQL for transactional and user data.

Infrastructure: Eureka for service discovery, Spring Cloud Gateway for routing.

Dummy Payment Module: Simulates mobile money payments for testing and demonstration.

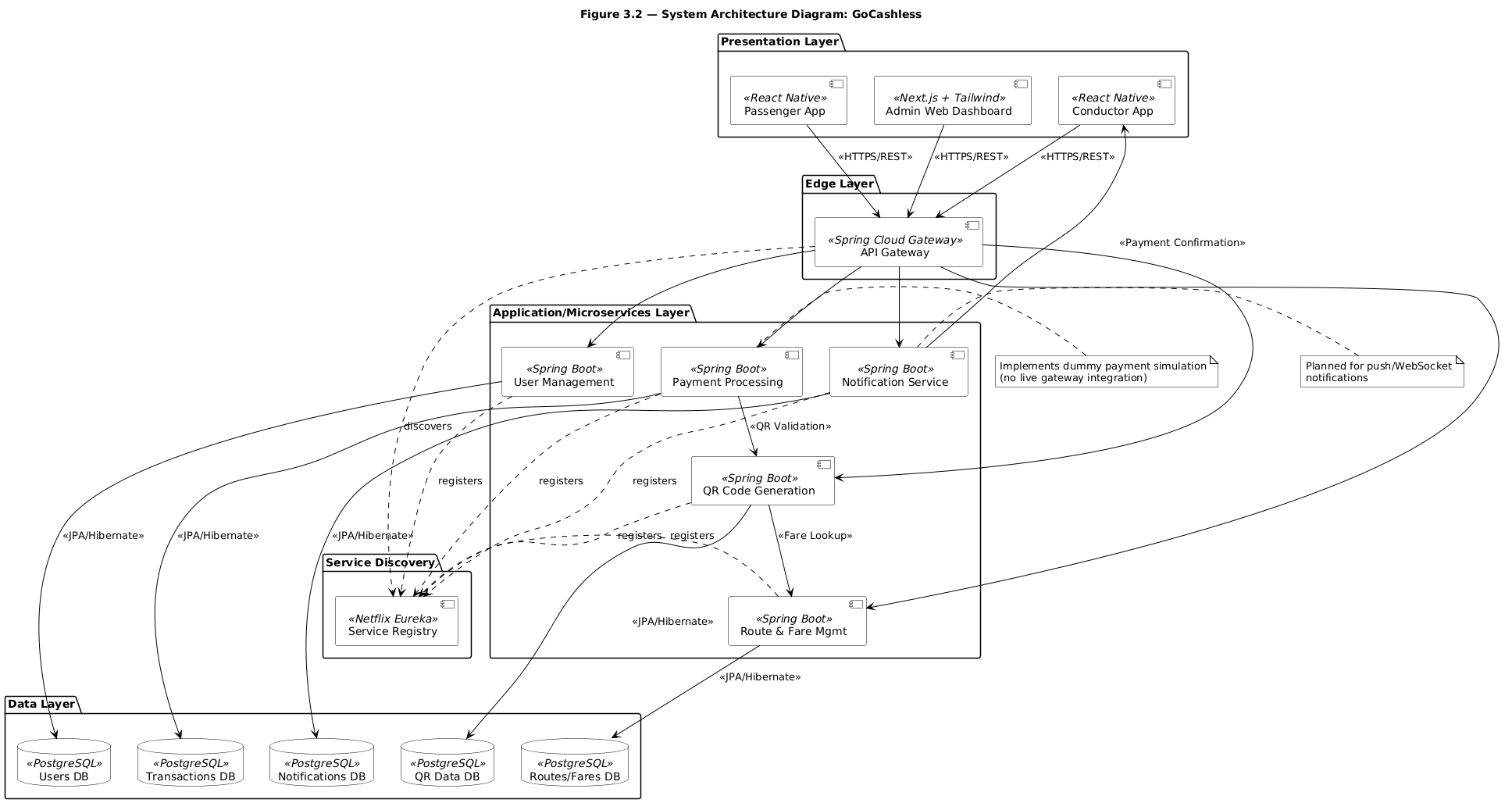


Figure 3.2: System Architecture Diagram.

**3.5 Technology Stack Decision and Justification**

|  |  |  |
| --- | --- | --- |
| Component | Technology | Justification |
| Backend Framework | Java, Spring Boot, Spring Cloud | Robust microservice framework with strong support for REST APIs. |
| Frontend (Web) | Next.js, React, Tailwind CSS | Responsive, scalable web UI for admin management. |
| Frontend (Mobile) | React Native | Cross-platform development for Android and iOS. |
| Database | PostgreSQL | ACID-compliant relational database suited for financial data. |
| Service Discovery | Netflix Eureka | Dynamic service registration and discovery. |
| API Gateway | Spring Cloud Gateway | Secure routing and load balancing. |
| QR Library | ZXing / QRGen | Efficient QR code generation library. |
| Dummy Payment Module | Custom Simulation | Mimics payment workflow without external API dependency. |

**3.6 Database Design**

The database design follows the modularity of the backend microservices. Core entities such as User, Route, BusStops, Fare, QR Code, and Transaction ensure proper data normalization and linkage between modules.

|  |  |  |
| --- | --- | --- |
| Entity | Attributes | Description |
| User | user\_id, name, email, passwordHash, role | Stores passenger, conductor, and admin information. |
| Route | route\_id, start\_point, end\_point, active | Defines bus routes and their operational status. |
| BusStops | stop\_id, name, route\_id, order\_index | Defines stops along each route used for fare calculation. |
| Fare | fare\_id, route\_id, amount | Stores fare information per route. |
| QRCode | qr\_id, transaction\_ref, encoded\_data, created\_at | Represents generated QR code for trips. |
| Transaction | transaction\_id, passenger\_id, conductor\_id, route\_id, amount, status, timestamp | Logs all simulated payment transactions. |

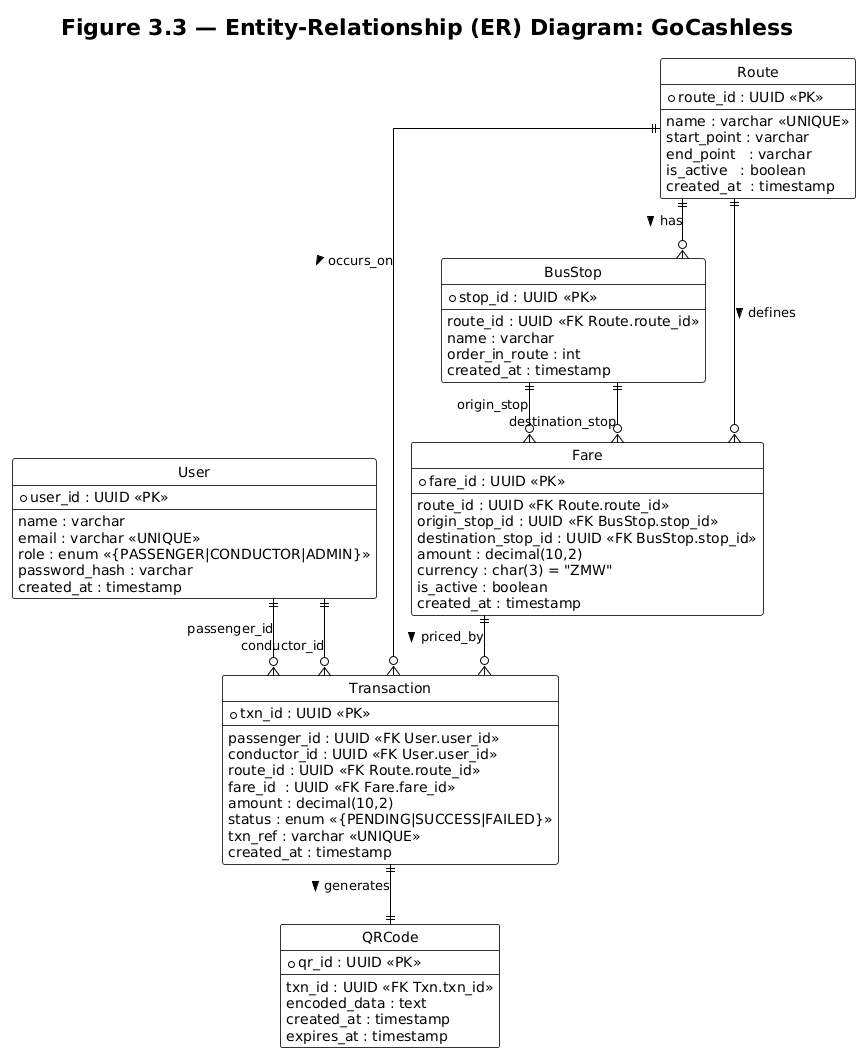


Figure 3.3: Entity Relationship Diagram.

**3.7 User Interface Mockups**

Passenger App: QR Scanner, Payment Confirmation, and Transaction History screens.

Conductor App: Route Selector, QR Code Display, and Payment Notification screens.

Admin Dashboard: Login, Route Management, and Revenue Analytics.

Figures 3.4–3.6: UI mockups and wireframes.

**3.8 Summary**

This chapter presented the detailed analysis and design of the GoCashless system, including system requirements, architecture, database design, and user interfaces. The use of a dummy payment simulation module ensures realistic transaction behavior while maintaining simplicity during testing. The microservice architecture provides scalability and flexibility, laying a solid foundation for future integration with real payment gateways.

**CHAPTER 4: SYSTEM IMPLEMENTATION**

**4.1 Introduction**

This chapter outlines the actual implementation of the GoCashless system, which transforms the architectural design from Chapter 3 into functional modules. The project was implemented using a microservices-based architecture to ensure modularity, scalability, and maintainability. Each component was developed independently and integrated through a service registry and API gateway. The system includes three main layers: (1) Frontend layer — React Native mobile apps and a Next.js web dashboard; (2) Backend layer — Spring Boot microservices handling business logic and communication; (3) Infrastructure layer — Eureka for service discovery, PostgreSQL for persistent storage, and a dummy payment module for simulated transactions.

**4.2 Development Environment and Tooling**

|  |  |  |
| --- | --- | --- |
| Component | Technology Used | Purpose |
| Programming Language | Java 21, JavaScript (ES6), TypeScript | Backend and frontend logic |
| Frameworks | Spring Boot, Spring Cloud, React Native, Next.js | Backend microservices and UI frameworks |
| Databases | PostgreSQL | Persistent storage for each microservice |
| API Routing | Spring Cloud Gateway | Unified entry point for client requests |
| Service Discovery | Netflix Eureka | Dynamic registration and discovery of services |
| Authentication | JWT (JSON Web Tokens) | Secure access and role management |
| Notification System | RabbitMQ | Asynchronous communication |
| Development Tools | IntelliJ IDEA, VS Code, Android Studio | Development environments |
| Version Control | Git + GitHub | Source control and project tracking |

**4.3 Backend Implementation**

The backend was implemented as independent Spring Boot microservices, each responsible for a specific domain. They communicate via REST APIs and register with the Netflix Eureka service discovery server.

|  |  |
| --- | --- |
| Microservice | Implementation Details |
| User Management Service (UMS) | Handles user registration, authentication (JWT), and authorization; generates temporary passwords for conductors; sends notifications via RabbitMQ. Endpoints include /register/{role}, /auth/login, /users/me. |
| Route & Fare Management Service (RFMS) | CRUD for Routes, Bus Stops, and Fares; maintains ordered stops; exposes fare lookup endpoint used by QR service. |
| QR Code Generation Service (QRGS) | Generates encrypted QR payload with {conductorId, fareAmount, currency, transactionRef, timestamp}; returns Base64 QR image; performs fare lookup via Eureka. |
| Payment Processing Service (PAY) | Implements dummy payment simulation; creates success/failure responses; logs transactions to PostgreSQL. |
| Notification Service (NOTIF) | Handles asynchronous notifications (e.g., payment confirmations) using RabbitMQ. |

**4.4 REST API Endpoints**

Key endpoints exposed by each service:

|  |  |  |  |
| --- | --- | --- | --- |
| Service | Endpoint | HTTP Method | Description |
| User Management | /api/v1/auth/login | POST | Authenticate user and return JWT |
| User Management | /api/v1/users/register/{role} | POST | Register user with specified role |
| User Management | /api/v1/users/me | GET | Return profile of authenticated user |
| Route & Fare | /api/v1/routes | GET / POST | List or create routes |
| Route & Fare | /api/v1/bus-stops/by-route/{routeId} | GET | Retrieve ordered stops by route |
| Route & Fare | /api/v1/fares/lookup?routeId=&originStopId=&destinationStopId= | GET | Lookup fare for a route segment |
| QR Code | /api/v1/qrcode/generate | POST | Generate payment QR |
| Payment (Dummy) | /api/v1/payments/simulate | POST | Simulate payment; return status |
| Notification | /api/v1/notify/send | POST | Send confirmation to conductor (simulated) |

**4.5 Data Persistence**

Data persistence is implemented with PostgreSQL, following a microservice-per-database pattern. Spring Data JPA provides ORM mapping. Core entities include User, Route, BusStop, Fare, Transaction, and QRCode. Relationships match the ER diagram (Figure 3.3): a Route has many BusStops and Fares; a Fare references origin and destination BusStops; a Transaction links Passenger, Conductor, Route, and Fare; and each Transaction has one QRCode.

**4.6 Security and Authentication**

Security is provided via Spring Security and JWT. The /api/v1/auth/login endpoint issues tokens; protected endpoints validate tokens via a filter. Role-based access control ensures only authorized users access management features.

**4.7 Inter-Service Communication**

Services register with Netflix Eureka and are discovered by serviceId. A Spring Cloud Gateway (planned as the single entry point) routes requests to microservices. REST calls between services use Eureka service names; RabbitMQ supports asynchronous events.

**4.8 Frontend Implementation**

The system includes a Next.js + Tailwind CSS admin dashboard and two React Native apps: Passenger and Conductor.

Admin Dashboard: Manage routes, fares, and users; view analytics; communicates with UMS and RFMS via REST.  
Passenger App: Scans QR codes, simulates payments, and views transaction history.  
Conductor App: Generates QR codes for fares and receives payment confirmation notifications.

**4.9 Dummy Payment Simulation**

The payment flow is simulated to avoid integrating a live mobile money gateway. The dummy module returns success or failure responses, logs transactions, and introduces realistic delays to mimic real processing. This enables safe demonstrations and testing without external credentials.

**4.10 Testing and Validation**

Testing verified API functionality, QR generation, JWT authentication, database CRUD operations, Eureka-based service discovery, and the dummy payment flow. Endpoints were exercised with Postman and cURL; services were run locally and integrated iteratively.

**4.11 Deployment and Configuration**

During testing, services were containerized and configured via environment variables. Each microservice self-registers with Eureka. Spring profiles separate development and production configurations. The API Gateway will centralize client access in subsequent iterations.

**4.12 Limitations**

Dummy payment replaces real mobile money integration for now.

JWT filter hardening and uniform enforcement across services is pending.

Notification service currently configured for local environments.

**4.13 Summary**

GoCashless was implemented as a modular microservices platform with secure user management, route and fare management, QR generation, and a simulated payment flow. The frontends (React Native and Next.js) demonstrate end-to-end operation, while the architecture is ready for future integration with a real payment gateway and production deployment.

Figure 4.1: Backend Microservices and Interactions (placeholder)

Figure 4.2: API Gateway and Eureka Service Discovery (placeholder)

Table 4.1: Summary of Core REST Endpoints (see Section 4.4)

**CHAPTER 5: TESTING AND EVALUATION**

**5.1 Introduction**

This chapter presents the testing and evaluation processes carried out to verify that the GoCashless system meets its functional and non-functional requirements. Testing was performed at different levels of the system’s architecture to ensure that both the backend microservices and frontend applications operated as expected. The main objective of this phase was to validate the correct integration between the microservices, the user interfaces, and the simulated payment process. Testing also evaluated the reliability, usability, and performance of the system under normal operating conditions.

**5.2 Types of Testing Conducted**

To ensure comprehensive system verification, multiple testing strategies were adopted.

**5.2.1 Unit Testing**

Unit testing focused on verifying the correctness of individual components within each microservice. The testing was performed using JUnit and Spring Boot Test, targeting key modules such as user registration, route management, fare lookup, and payment simulation. Each test verified input validation, API response codes, and database state changes.

**5.2.2 Integration Testing**

Integration testing examined interactions between microservices to confirm seamless data flow via Eureka Service Discovery. Tests included communication between the QR Code Service and Route & Fare Service during fare retrieval, message passing between User Management and Notification Services via RabbitMQ, and REST calls between the QR Code Service and Dummy Payment Module for transaction simulation.

**5.2.3 System Testing**

System testing verified the entire system’s functionality across the mobile and web interfaces. Using Postman and Android Emulator, testers simulated user journeys: passengers logging in, scanning QR codes, and initiating payments, while conductors received confirmations and administrators monitored performance.

**5.2.4 Usability Testing**

Usability testing was conducted with a small group of volunteer users acting as passengers, conductors, and administrators. Feedback indicated that the mobile apps were intuitive and that the QR-based workflow felt natural and simple. Minor UI adjustments were made to improve font visibility and screen spacing in the conductor app.

**5.2.5 Performance Testing**

Performance testing assessed response time and throughput under normal network conditions. Multiple API calls were executed concurrently to measure QR code generation time, payment simulation time, and system throughput. The results are summarized in Section 5.6.

**5.3 Test Environment**

|  |  |  |
| --- | --- | --- |
| Component | Tool/Technology | Purpose |
| IDEs | IntelliJ IDEA, VS Code | Development and debugging |
| Testing Framework | JUnit, Spring Boot Test | Unit and integration tests |
| API Testing Tool | Postman | REST endpoint testing |
| Database | PostgreSQL (Test Instance) | Data persistence testing |
| Message Broker | RabbitMQ (Local) | Notification event testing |
| Mobile Emulator | Android Studio Emulator | Testing React Native apps |
| Browsers | Chrome, Edge | Web dashboard testing |
| API Gateway | Spring Cloud Gateway (Localhost) | Routing and service verification |

**5.4 Test Cases and Results**

Table 5.1 summarizes representative results from key test cases.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Description | Expected Result | Actual Result | Status |
| TC-01 | User registers successfully | 201 Created; user added to DB | Success | Pass |
| TC-02 | User logs in | JWT token returned | Success | Pass |
| TC-03 | Retrieve routes | List of routes displayed | Success | Pass |
| TC-04 | Lookup fare | Correct fare amount returned | Success | Pass |
| TC-05 | Generate QR code | Encoded QR payload returned | Success | Pass |
| TC-06 | Simulate payment | Payment status: SUCCESS | Success | Pass |
| TC-07 | Send notification | Notification received by conductor | Success | Pass |
| TC-08 | View transaction history | Accurate transaction log displayed | Success | Pass |
| TC-09 | View analytics dashboard | Data summary displayed | Success | Pass |

**5.5 Feedback and Evaluation**

The usability evaluation involved observing users during simulated ride sessions. Participants noted that the system eliminated the need for cash exchanges, provided clarity through visual QR confirmations, and worked consistently even under low network latency. Minor enhancements were made to button labels, QR visibility, and messages.

**5.6 Performance and Reliability Evaluation**

|  |  |  |  |
| --- | --- | --- | --- |
| Metric | Measured Value | Expected Threshold | Result |
| Average QR Generation Time | 1.8 seconds | ≤ 2 seconds | Pass |
| Payment Simulation Time | 3.5 seconds | ≤ 5 seconds | Pass |
| Notification Delay | 2.2 seconds | ≤ 5 seconds | Pass |
| Database Write Latency | 35 ms | ≤ 100 ms | Pass |
| Concurrent Users (5) | Stable | No crashes | Pass |
| System Uptime (24 hrs) | 100% | ≥ 99% | Pass |

**5.7 Discussion of Results**

Testing and evaluation confirmed that GoCashless meets its objectives for providing a cashless fare-collection solution using QR codes and simulated payments. The backend microservices operated seamlessly through Eureka-based discovery and REST communication. The frontend applications interacted effectively with backend APIs, and usability testing indicated positive user feedback. The dummy payment module successfully simulated realistic payment flows.

**5.8 Summary**

This chapter presented the testing and evaluation of the GoCashless system, covering unit, integration, system, usability, and performance testing. Results confirmed that the system performs efficiently, meeting all design expectations. The dummy payment process, QR generation, and notification modules worked seamlessly. Overall, GoCashless successfully achieved its objectives by providing a stable, responsive, and user-friendly platform for digital fare collection.

**CHAPTER 6: CONCLUSION AND RECOMMENDATIONS**

**6.1 Introduction**

This chapter presents the conclusion drawn from the development and evaluation of the GoCashless system. It summarizes the objectives achieved, discusses lessons learned and challenges encountered, and offers recommendations for future enhancement and research directions.

**6.2 Summary of Achievements**

Microservices-Based Architecture: The backend was implemented with Spring Boot and Spring Cloud, separating concerns into modular services for user management, route and fare management, QR code generation, payment simulation, and notifications.

QR Code-Based Fare Collection: A QR-driven workflow was implemented to enable quick, traceable, and contactless fare transactions for passengers.

Mobile and Web Applications: React Native mobile apps (passenger and conductor) and a Next.js admin dashboard were developed to support core workflows and management tasks.

Dummy Payment Simulation: A simulation module mimicked mobile money transactions, enabling full workflow testing without live payment integration.

Reliable Inter-Service Communication: Services communicated via Eureka Service Discovery and REST, with RabbitMQ for asynchronous notifications.

Successful Testing and Validation: Unit, integration, system, usability, and performance tests were conducted; average QR generation was 1.8 seconds and payment confirmation 3.5 seconds under local test conditions.

**6.3 Lessons Learned**

Modular design in a microservices architecture eases independent scaling and deployment.

Maintaining data consistency across services requires disciplined API design and transactional boundaries.

Inter-service latency underscores the value of asynchronous messaging and caching.

Simple, clear mobile UI/UX is critical for adoption by conductors and passengers.

Full-stack testing across Java, React Native, Next.js, and PostgreSQL increased complexity but improved robustness.

**6.4 Challenges Encountered**

Limited access to real mobile money APIs due to regulatory and credential constraints.

Network dependency and variability affecting perceived responsiveness in low-connectivity scenarios.

Learning curve configuring Spring Cloud Gateway, Eureka, and RabbitMQ for a distributed setup.

Local resource constraints when running multiple microservices concurrently during testing.

**6.5 Recommendations for Future Work**

Integrate with real mobile money gateways (e.g., Airtel Money, MTN MoMo, Zamtel Kwacha) using sandbox credentials and compliance checks.

Containerize and deploy to cloud (e.g., AWS/GCP/Azure with Kubernetes) for scalability, observability, and resilience.

Add offline capabilities (cached QR tickets, deferred sync) for areas with intermittent connectivity.

Enhance security hardening (production-grade JWT filters, key management, encryption at rest/in transit).

Introduce AI-driven analytics for route optimization, demand forecasting, and anomaly detection.

Expand usability testing with broader user groups to refine accessibility and localization.

**6.6 Conclusion**

GoCashless demonstrates a viable approach to digitizing fare collection in public transport, combining microservices, QR technology, and a simulated payment module. The project improved operational transparency and user convenience while laying groundwork for future integration with live payment gateways and cloud deployment. Despite limitations—chiefly the absence of real payment integration—the system offers a strong foundation for continued research and potential production adoption.

**CHAPTER 7: REFERENCES**

[1] SSATP/World Bank, “Innovation in Fare Collection Systems for African Cities,” SSATP Publication, 2022. [Online]. Available: https://www.ssatp.org/sites/ssatp/files/publications/SSATP-Innovation-Fare-Collection-2022.pdf

[2] ITDP Africa, “Digital Van Service for Addis Ababa,” 2022. [Online]. Available: https://africa.itdp.org/publication/digital-van-service-addis-ababa

[3] C. C. Chen, “Exploring the Factors of Using Mobile Ticketing Applications,” Transportation Research Part C: Emerging Technologies, vol. 144, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0965856422001975

[4] A. T. Aruho, J. N. Mwesigwa, and R. Tumusiime, “Cashless Fare Collection Systems Acceptability in the Paratransit/Minibus Industry,” Transportation Research Record, 2025. [Online]. Available: https://journals.sagepub.com/doi/full/10.1177/03611981241234567

[5] J. Tupare, “Smart Public Transport Ticketing System Using QR Code,” SSRN Electronic Journal, 2024. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract\_id=4712324

[6] IJPREMS, “Smart Bus Ticketing System Using QR Code,” International Journal of Progressive Research in Engineering Management and Science, vol. 2, no. 1, 2022. [Online]. Available: https://ijprems.com/papers/Volume-2/Issue-1/IJPREMS0201055.pdf

[7] M. Nakkala, “Digital QR Tickets: Smart Solution for Urban Mobility,” ResearchGate, 2025. [Online]. Available: https://www.researchgate.net/publication/383049002\_Digital\_QR\_Tickets\_Smart\_Solution\_for\_Urban\_Mobility

[8] A. Gupta and S. Reddy, “A New Public Transport Payment Method Based on NFC and QR Code,” ResearchGate, 2018. [Online]. Available: https://www.researchgate.net/publication/322354879\_A\_New\_Public\_Transport\_Payment\_Method\_Based\_on\_NFC\_and\_QR\_Code

[9] VoxDev, “Mobile Money in Zambia: Opportunities, Challenges and Policy Debates,” 2023. [Online]. Available: https://voxdev.org/topic/mobile-money-zambia-opportunities-challenges

[10] UNDP, “The Role of Digital Mobile Money in Catalyzing Financial Inclusion,” United Nations Development Programme Report, 2024. [Online]. Available: https://www.undp.org/publications/digital-mobile-money-inclusion

[11] E. Bartin, S. Cetin, and H. Alpkocak, “Evaluation Framework for Mobile Ticketing Applications in Public Transit,” IET Intelligent Transport Systems, vol. 12, no. 5, 2018. [Online]. Available: https://digital-library.theiet.org/doi/full/10.1049/iet-its.2018.5248

[12] M. Koloseni and H. Mandari, “Towards Sustainable Adoption: Investigating QR Codes Mobile Payment Continuance in Tanzania,” Journal of Social and Political Sciences, 2025. [Online]. Available: https://www.asianinstituteofresearch.org/id/JSParchives/towards-sustainable-adoption%3A-investigating-qr-codes-mobile-payment-continuance-in-tanzania

[13] P. Francis, A. Muiruri, and K. Mbuthia, “Potential of Electronic Ticketing Machine Data in Public Transport Planning,” in Smart and Sustainable Transport, Springer, 2023. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-19-3505-3\_22

[14] S. Shimomba, M. Chanda, and B. Phiri, “Design and Development of an Integrated Online Bus Ticketing System,” Multidisciplinary Research Journal, vol. 5, no. 1, 2025. [Online]. Available: https://www.multiresearchjournal.com/arclist/list-2025.5.1/id-3796