IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, May 2026
Impact Factor: 8.2

Taxi Manager: A Cross-Platform Mobile Application for Role-Based Taxi Fleet and Booking Management

Piyush Prajapati[1] & Rajendra Singh[2]
[1] Department of Computer Science and Engineering
[2] Dean, School of Engineering and Technology, Department of Computer Science and Engineering
Raffles University, Neemrana, Rajasthan, India
piyushprajapati0102@gmail.com[1], rajendra.singh@rafflesuniversity.edu.in[2]

Abstract: *Small and medium taxi fleet operators in India continue to rely on manual, paper-based methods for booking management, driver coordination, and payment tracking despite the rapid proliferation of smartphones across the country. This paper presents "Taxi Manager," a cross-platform mobile application built using Expo and React Native with TypeScript, providing role-based fleet management for two user types: Owners and Drivers. The system integrates Firebase as its complete backend, utilizing Firestore for real-time data synchronization, phone-number OTP-based authentication, Firebase Storage for document management, and Cloud Functions for push notifications and automated report aggregation. A dual-layer Role-Based Access Control (RBAC) architecture enforces data security at both the client routing layer and the Firestore server-side rules layer. A bilingual English and Hindi interface ensures accessibility for users across different literacy levels. The application is actively deployed at GM Taxi Service, Behror, Rajasthan, validating its real-world applicability. Experimental evaluation across 15 test cases demonstrates a 100% pass rate. The system reduces booking creation time from 5–10 minutes to under one second and delivers push notifications to drivers within 2–3 seconds of booking assignment, demonstrating the transformative potential of modern cross-platform mobile development and serverless cloud infrastructure in the transportation management domain.*

Keywords: Taxi Management, React Native, Expo, Firebase, Firestore, Role-Based Access Control, Mobile Application, Fleet Management, Real-Time Database, NativeWind, TypeScript, Bilingual Interface, Cloud Functions

I. INTRODUCTION

The transportation and cab-hire industry in India has witnessed exponential growth over the past decade, driven by urban expansion and a growing need for reliable, on-demand mobility. According to the Ministry of Road Transport and Highways, India had over 5 million registered commercial vehicles as of 2023, a significant portion of which operate under individual owners or small fleet operators [1]. Despite this scale, fleet management in this segment remains largely informal, relying on phone calls, physical ledgers, and spreadsheets for booking records, driver coordination, and payment tracking.

Traditional taxi management workflows suffer from critical inefficiencies. A taxi owner managing even a small fleet of five to ten vehicles must simultaneously juggle driver availability, customer bookings, vehicle assignments, payment collection, and document compliance. Without a centralized system, critical information is scattered across notebooks and phone conversations, leading to double bookings, missed assignments, delayed payments, and poor customer experience.

DOI: 10.48175/IJARSCT-35764

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 10, May 2026 Impact Factor: 8.2

The rapid proliferation of smartphones in India, with over 750 million smartphone users as of 2024, has created an opportunity to deliver sophisticated fleet management capabilities through mobile applications [2]. Cross-platform frameworks such as React Native and Expo allow developers to build applications targeting both Android and iOS from a single codebase, significantly reducing development time and cost. Paired with cloud backend platforms like Firebase, which provide real-time data synchronization, secure authentication, scalable storage, and serverless computing, it is now possible to build production-grade fleet management applications without maintaining dedicated servers.

This paper makes the following contributions:

- A complete cross-platform mobile fleet management application using Expo, React Native, and TypeScript deployed on Firebase
- A dual-layer RBAC architecture combining Expo Router grouped layouts with Firestore security rules
- A bilingual translation service with AsyncStorage caching for English and Hindi support
- A serverless Cloud Functions architecture for real-time push notifications and automated nightly report aggregation
- Real-world deployment validation at GM Taxi Service, Behror, Rajasthan

II. LITERATURE REVIEW

A. Cross-Platform Mobile Development

React Native, introduced by Meta in 2015, allows developers to write mobile applications in JavaScript targeting native platform components [3]. Unlike hybrid frameworks that render content in a web view, React Native bridges JavaScript code to actual native UI components, providing near-native user experience. Bidve et al. (2022) conducted a comparative analysis of cross-platform frameworks including React Native, Flutter, and Xamarin, finding that React Native offered the best balance of developer productivity, community support, and native performance for data-driven applications [4]. Nawrocki et al. (2021) further found that React Native applications achieved 85–95% of native performance on standard UI operations while requiring approximately 60% less development time [5].

B. Backend-as-a-Service Platforms

Firebase has emerged as the dominant BaaS platform for mobile applications. Garg and Verma (2021) evaluated Firebase against traditional server-based backends for real-time mobile applications, finding that Firestore's real-time listener model significantly reduced implementation complexity by eliminating the need for polling or manual WebSocket management [6]. Firebase's JavaScript SDK is optimized for React Native and provides a comprehensive feature set including Firestore, Authentication, Storage, and Cloud Functions on a free Spark tier suitable for small-scale deployments.

C. Role-Based Access Control

Johnson and Williams (2022) categorized RBAC implementations in cloud-native mobile applications into three tiers: presentation-layer RBAC (client-side routing only), data-layer RBAC (server-side rules only), and dual-layer RBAC [7]. Their study found that dual-layer RBAC provides the strongest security posture for applications handling sensitive business data. Firestore security rules provide a server-side enforcement mechanism that prevents unauthorized data access even if a user circumvents the client-side UI.

D. Fleet Management and Technology Adoption

Research by Jain and Soni (2020) on technology adoption among small taxi fleet operators in Tier 2 and Tier 3 Indian cities identified cost, complexity, and language as the primary barriers [8]. Their study recommended lightweight, mobile-first, vernacular-language applications as the most effective approach for this underserved market segment.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 10, May 2026 Impact Factor: 8.2

Commercial solutions such as Samsara and Verizon Connect, priced at $30–150 per vehicle per month, are inaccessible to small fleet operators with one to twenty vehicles.

E. Real-Time Data Synchronization

Gupta et al. (2023) compared polling-based and event-driven data synchronization architectures in mobile fleet management applications, finding that event-driven architectures reduced average data staleness from 30 seconds to under 500 milliseconds, with 40% lower bandwidth consumption [9]. This is particularly valuable in fleet management where booking status changes and payment records must be reflected across devices instantly.

III. PROPOSED SYSTEM

A. System Architecture

Taxi Manager follows a three-tier architecture: the Presentation Layer (Expo/React Native frontend), the Application Layer (Firebase services), and the Data Layer (Firestore and Storage). The mobile frontend communicates with Firebase services exclusively through the Firebase JavaScript SDK, eliminating the need for a custom API server.

The system provides two role-based interfaces: the Owner module for fleet management, booking creation, and financial reporting; and the Driver module for booking execution, payment recording, and expense logging. Both interfaces share the same Firebase backend with strict data isolation enforced by Firestore security rules.

B. Authentication and Onboarding

Phone-number OTP authentication was chosen over email/password for two reasons: it requires no prior account creation and is familiar to all smartphone users in India regardless of literacy level; and it eliminates the risk of forgotten passwords. After OTP verification, the system queries the user's Firestore profile to determine their role (Owner or Driver) and routes them to the appropriate grouped layout in Expo Router.

C. Dual-Layer RBAC Architecture

Security is enforced at two independent layers. At the presentation layer, Expo Router grouped layouts ((owner)/ and (driver)/) ensure that screens intended for owners are inaccessible to drivers within the application. The AuthContext provider checks the user's role from Firestore on every app launch and redirects accordingly. At the data layer, Firestore security rules enforce that owners can only access bookings, drivers, and taxis where the ownerId field matches their authenticated UID, while drivers can only read bookings assigned to them and write payment records for their own bookings.

D. Owner Module

The Owner module comprises five primary screens. The Dashboard displays daily net profit, income, expenses, and active fleet count with quick-action tiles for the six most frequent operations. The Bookings screen provides a horizontal date carousel, booking cards with payment status badges (Paid, Partial, Unpaid), and inline Edit and Payment actions. The booking creation form supports multi-stop trips with pickup, intermediate stops, and drop-off fields, fare amount, advance payment, and driver/taxi assignment. The Drivers screen manages driver profiles with monthly salary and payment day configuration. The Reports screen shows categorized expense breakdowns (fuel, food, toll, maintenance) as horizontal progress bars with percentage labels, and a chronological transaction history.

E. Driver Module

The Driver module is designed for simplicity and speed, with three screens: Dashboard, Bookings, and Payments. The Dashboard displays monthly trip count, customer cash collected, and upcoming assigned trips. The Bookings screen

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, May 2026
Impact Factor: 8.2

shows full route details for each assigned trip. The Payments screen provides a per-driver financial summary showing salary, received cash, received online, customer cash collected, food expenses, total paid, and remaining balance.

F. Cloud Functions Architecture

Two primary Cloud Functions are implemented. A Firestore onCreate trigger fires when a new booking document is created, reads the assigned driver's FCM token, and delivers a push notification via Firebase Cloud Messaging within 2–3 seconds. A scheduled function runs nightly at 23:59 IST using a Pub/Sub trigger, aggregates all payment and expense data by owner, and writes pre-computed report documents to the reports collection, ensuring the Reports screen loads in under 800 milliseconds without on-demand aggregation queries.

G. Bilingual Translation Service

A custom translation service queries the MyMemory API for English-to-Hindi translations and caches results in AsyncStorage with a key prefix scheme. After the first load, all Hindi UI labels are served from local cache with sub-10 millisecond latency, providing a seamless bilingual experience without network dependency on subsequent sessions.

H. Technology Stack

The complete technology stack is summarized in Table I.

Table I: Technology Stack

Technology	Version	Purpose
Expo (React Native)	SDK 52	Cross-platform app shell
TypeScript	5.x	Type-safe development
NativeWind	4.x	Tailwind CSS for React Native
Expo Router	3.x	File-system based navigation
Firebase Auth	Latest	Phone OTP authentication
Firestore	Latest	Real-time NoSQL database
Firebase Storage	Latest	Document and receipt storage
Cloud Functions	Node.js 18	Serverless backend logic
AsyncStorage	Latest	Local caching layer
EAS Build	Latest	CI/CD build pipeline

IV. EXPERIMENTAL RESULTS

A. Test Cases and Results

A comprehensive test suite of 15 test cases was executed on a physical Android device (Redmi Note 11, Android 13) and the iOS Simulator (iOS 17). Key test cases and results are summarized in Table II. All 15 test cases passed, yielding a 100% pass rate. Critical validations included end-to-end booking creation with push notification delivery (TC-03), multi-stop booking rendering (TC-04), real-time payment status updates (TC-05, TC-06), and bilingual translation with local caching (TC-12).

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, May 2026
Impact Factor: 8.2

Table II: Selected Test Case Results

TC	Scenario	Expected Result	Result
TC-01	Owner OTP Login	Routed to owner dashboard	PASS
TC-02	Driver OTP Login	Routed to driver dashboard	PASS
TC-03	Create Booking	Booking created; driver notified in 2s	PASS
TC-04	Multi-Stop Booking	All stops shown in booking card	PASS
TC-05	Record Partial Payment	Balance updated; status → Partial	PASS
TC-06	Record Full Payment	Balance = 0; status → Paid	PASS
TC-07	Financial Reports	Correct P&L aggregation shown	PASS
TC-12	Hindi Language Switch	UI translated and cached locally	PASS

B. Performance Metrics

Application performance metrics collected during testing are presented in Table III.

Table III: Application Performance Metrics

Metric	Measured Value
OTP delivery time	< 5 seconds
Booking creation time	< 1 second
Push notification delivery	2–3 seconds
Real-time status update	< 500 ms
App cold start (mid-range Android)	2–3 seconds
Reports screen load time	< 800 ms
Translation (cached)	< 10 ms
Translation (uncached)	0.8–1.2 seconds

C. Comparative Analysis

Table IV compares the traditional manual approach against Taxi Manager across key operational dimensions.

Table IV: Traditional Approach vs. Taxi Manager

Operational Aspect	Traditional	Taxi Manager
Booking creation	5–10 minutes (phone + register)	< 1 second
Driver notification	Phone call (may be missed)	Instant push notification
Payment tracking	Paper receipts; easily lost	Digital; real-time balance
Financial reporting	Manual monthly tallying	Automated daily P&L
Language support	Verbal Hindi only	Full English and Hindi in-app
Operational cost	High time and error cost	Zero (Firebase free tier)

DOI: 10.48175/IJARSCT-35764

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 10, May 2026 Impact Factor: 8.2

D. Firebase Free Tier Usage

The system operates entirely within Firebase's Spark (free) tier. Estimated daily usage at GM Taxi Service represents approximately 4% of the 50,000 free daily document reads and 1% of the 20,000 free daily document writes, confirming zero-cost operability for small fleet operators.

V. CONCLUSION

This paper presented Taxi Manager, a cross-platform mobile application that addresses the fundamental problems of small taxi fleet management in India: manual booking coordination, inefficient driver communication, poor payment tracking, unclear financial visibility, and language accessibility barriers. The system is built on Expo SDK 52 with React Native and TypeScript, Firebase for all backend services, NativeWind for design consistency, and Expo Application Services for the build and deployment pipeline.

The key technical contributions include a dual-layer RBAC system combining Expo Router grouped layouts with Firestore security rules, a real-time coordination architecture using Firestore snapshot listeners and Cloud Functions push notifications, a comprehensive financial reporting module enabling data-informed business decisions, a zero-cost cloud deployment on Firebase's free tier, and a bilingual English-Hindi interface with AsyncStorage caching. The system is operationally deployed and actively used at GM Taxi Service, Behror, Rajasthan, providing real-world validation of its design and feature set.

Future work includes real-time GPS tracking integration, in-app payment gateway integration (Razorpay/Paytm), a customer-facing booking portal, automatic route optimization, offline-first architecture, multi-language expansion beyond Hindi, and a web-based admin dashboard for enterprise fleet operators.

ACKNOWLEDGMENT

I would like to sincerely thank Rajendra Singh, Dean, School of Engineering and Technology, Department of Computer Science and Engineering, Raffles University, for his valuable guidance, continuous support, and encouragement throughout this project.

I am also grateful to the Department of Computer Science and Engineering, Raffles University, for providing the academic support and necessary environment to complete this research work.

REFERENCES

1. Ministry of Road Transport and Highways, "Road Transport Yearbook 2022–23," Government of India, 2023.
2. IAMAI, "India Internet Report 2024," Internet and Mobile Association of India, 2024.
3. Meta Platforms Inc., "React Native: Build Mobile Apps with JavaScript," Official Documentation, 2023. [Online]. Available: https://reactnative.dev
4. V. Bidve, P. Sarkar, and R. K. Tripathy, "A Comparative Study of Cross-Platform Mobile Development Frameworks: React Native, Flutter, and Xamarin," International Journal of Computer Applications, vol. 184, no. 12, pp. 1–7, 2022.
5. P. Nawrocki, K. Wrona, M. Marczak, and M. Jarosz, "A Comparison of Native and Cross-Platform Frameworks for Mobile Applications," Computer, vol. 54, no. 3, pp. 18–27, 2021.
6. R. Garg and S. Verma, "Firebase as Backend-as-a-Service for Real-Time Mobile Applications: A Performance Analysis," International Journal of Engineering and Technology, vol. 9, no. 4, pp. 112–118, 2021.
7. A. Johnson and T. Williams, "Role-Based Access Control Patterns in Cloud-Native Mobile Applications," IEEE Access, vol. 10, pp. 45612–45624, 2022.
8. Jain and B. Soni, "Technology Adoption Barriers among Small Taxi Fleet Operators in Tier 2 Indian Cities," Journal of Transport and Technology Policy, vol. 12, no. 3, pp. 45–58, 2020.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, May 2026
Impact Factor: 8.2

9. R. Gupta, A. Sharma, and N. Patel, "Event-Driven vs. Polling Architectures in Mobile Fleet Management Systems," Journal of Mobile Computing and Applications, vol. 15, no. 2, pp. 88–101, 2023.
10. Google Firebase, "Firebase Documentation: Firestore, Authentication, Storage, Cloud Functions," 2024. [Online]. Available: https://firebase.google.com/docs
11. Expo Inc., "Expo SDK 52 Documentation," 2024. [Online]. Available: https://docs.expo.dev
12. NativeWind, "NativeWind: Tailwind CSS for React Native," 2024. [Online]. Available: https://www.nativewind.dev
13. Shneiderman, C. Plaisant, M. Cohen, S. Jacobs, and N. Elmqvist, Designing the User Interface: Strategies for Effective Human-Computer Interaction, 6th ed. Pearson, 2016.
14. M. Alotaibi and D. Roussinov, "Usability of Mobile Applications for Transport Workers: A Systematic Review," International Journal of Human-Computer Studies, vol. 95, pp. 32–47, 2016.