International Journal of Scientific Research & Engineering Trends
Volume 12, Issue 2, Mar-Apr-2026, ISSN (Online): 2395-566X

Happenings and Happiness: A Comprehensive Android-Based Wedding and Event Planning Platform with Real-Time Communication, Payment Integration, and AI-Ready Architecture

Darshankumar Patel
Department of Computer Science & Engineering
Parul Institute of Engineering & Technology, Parul University Vadodara, Gujarat, India — 391760

Abstract- The Indian wedding and event planning industry, valued at over INR 3 lakh crore annually, remains largely fragmented and dependent on manual processes. This paper presents Happenings and Happiness, a comprehensive dual-role Android application that digitizes the complete wedding planning lifecycle for both event organizers and service vendors. The system integrates Firebase Firestore for real-time data synchronization, Firebase Cloud Messaging (FCM) V1 API with JWT-based OAuth2 authentication for push notifications, RazorPay payment gateway for secure transaction processing, and Cloudinary for cloud-based media management. The application implements real-time bidirectional chat, vendor discovery and booking management, guest list management, budget tracking, and an earnings analytics dashboard. The dual-role architecture supports seamless interaction between users and vendors within a single application while maintaining strict role-based data access through Firestore security rules. Performance evaluation demonstrates sub-second message delivery in real-time chat, reliable push notification delivery across both user roles, and consistent UI performance across Android 10–14 devices. The system addresses critical gaps in existing wedding planning applications by providing an end-to-end digital solution tailored for the Indian market.

Keywords – Android Development, Java, Firebase Firestore, Firebase Cloud Messaging, RazorPay, Cloudinary, Real-Time Chat, Push Notifications, Wedding Planning Application, Vendor Management, Event Planning, Mobile Computing.

I. INTRODUCTION

The rapid proliferation of smartphones and mobile internet connectivity has fundamentally transformed how individuals plan, organize, and manage significant life events. India's wedding industry stands as one of the largest celebration markets globally, with an estimated annual valuation exceeding INR 3–4 lakh crore and involving millions of transactions between event organizers and service vendors spanning photography, catering, decoration, venue management, makeup artistry, and entertainment services [1].

Despite this scale, the majority of vendor discovery, booking, negotiation, and payment workflows remain fragmented across phone calls, WhatsApp messages, and in-person meetings. This fragmentation leads to inefficiencies including missed booking opportunities, payment disputes, lack of accountability, and significant time investment in coordination tasks that could be digitized. Event organizers frequently struggle to maintain oversight of guest lists, track wedding budgets across multiple vendors, and manage logistics across different planning phases.

Existing digital solutions in this space present significant limitations. International platforms such as WeddingWire and Zola are designed primarily for Western markets and lack the vendor ecosystem, payment infrastructure, and cultural context relevant to Indian weddings. Generic event management applications do not provide the specialized vendor- booking workflow, real-time communication, or planning tool integration required for comprehensive wedding management.

This paper presents Happenings and Happiness, a production-ready Android application that addresses these gaps through an integrated platform supporting both event organizers and service vendors. The system's key contributions include: (1) a dual-role architecture within a single APK supporting both User and Vendor roles with distinct dashboards and feature sets; (2) real-time bidirectional chat using Firebase Firestore snapshot listeners; (3) a comprehensive push notification system using FCM V1 API with JWT-based service account authentication; (4) RazorPay payment gateway integration with complete transaction lifecycle management; (5) Cloudinary-based media management; and (6) a suite of wedding planning

© 2026 IJSRET

tools including guest list management, budget tracking, and task checklists.

II. BACKGROUND AND LITERATURE REVIEW

A. Mobile Application Development Landscape

Android, developed by Google, commands a global mobile operating system market share exceeding 72% as of 2024, making it the dominant platform for mobile application deployment [2]. The Android development ecosystem provides comprehensive tooling through Android Studio, a rich set of Jetpack libraries, and tight integration with Google Cloud services including Firebase. Java and Kotlin serve as the primary programming languages, with Java maintaining widespread adoption in enterprise and academic contexts due to its mature ecosystem and extensive documentation.

Firebase, Google's Backend-as-a-Service (BaaS) platform, has emerged as the preferred backend solution for mobile application development due to its real-time synchronization capabilities, built-in authentication, and scalable NoSQL database architecture. Firebase Firestore's document-collection model provides flexible schema design and native support for offline data persistence, making it particularly well-suited for mobile applications requiring robust connectivity management [3].

B. Related Work

Kaur et al. [4] proposed a mobile-based event management system using cloud computing principles but limited their scope to event discovery without addressing the vendor-booking workflow or payment integration. Singh and Sharma [5] developed a Firebase-based social platform demonstrating real-time data synchronization capabilities but did not address the dual-role architecture or multi-party transaction management required in a commercial service marketplace.

Patel and Mehta [6] examined push notification systems for Android applications, highlighting the security limitations of the legacy FCM server key approach and recommending migration to OAuth2-based FCM V1 API. Their findings directly informed the notification architecture adopted in this work. Research by Kumar et al. [7] on digital payment integration in Indian mobile applications underscored the importance of RazorPay as the preferred gateway for Indian market deployment due to its UPI support and regulatory compliance.

C. Research Gap

Despite the advances documented in existing literature, a comprehensive review reveals the following unaddressed gaps: (1) absence of integrated dual-role applications supporting both service consumers and providers within a single platform; (2) limited adoption of FCM V1 API in published mobile application research, with most implementations still using the deprecated legacy server key; (3) lack of integration between real-time communication, payment processing, and planning tools within a unified wedding management context; and (4) insufficient attention to Firestore security rule design for multi-tenant applications with role-based access requirements.

III. SYSTEM ARCHITECTURE AND DESIGN

A. Overall Architecture

The Happenings and Happiness application follows a client-server architecture with Firebase serving as the cloud backend. The system comprises three primary layers: (1) the Android client layer implementing the presentation and business logic; (2) the Firebase backend layer managing authentication, real-time database, and cloud messaging; and (3) the third-party service layer integrating Cloudinary for media management and RazorPay for payment processing.

The architecture adopts a dual-role design where a single APK supports both User (event organizer) and Vendor (service provider) roles. Role determination occurs post-authentication through Firestore collection membership: authenticated users with documents in the users/{uid} collection are routed to the User dashboard, while those with documents in vendors/{uid} are routed to the Vendor dashboard.

Table I: System Architecture Components

Layer	Component	Technology
Presentation	Android UI (Activities/Fragments)	Java, XML Layouts, SDP/SSP
Business Logic	Activity/Fragment Controllers	Java, Android SDK
Authentication	User & Vendor Auth	Firebase Authentication
Database	Real-time NoSQL Database	Firebase Firestore
Messaging	Push Notifications	FCM V1 API + JWT OAuth2
Media	Image Upload & Management	Cloudinary Android SDK
Payments	Transaction Processing	RazorPay Android SDK
Version Control	Source Code Management	Git / GitHub

Database Schema Design

The Firestore database is organized into seven top-level collections with appropriate subcollections to support the application's data requirements while maintaining query efficiency and security isolation:

- **users/{uid}:** User profiles including bride/groom names, wedding date, budget, FCM token, and username

© 2026 IJSRET

- **users/{uid}/guests/{guestId}:** Guest list entries with RSVP status and member count
- **users/{uid}/expenses/{expenseId}:** Budget expense records with amount and estimated values
- **users/{uid}/activities/{activityId}:** Activity feed entries for recent actions
- **vendors/{uid}:** Vendor profiles, verification status, services, and FCM token
- **vendors/{uid}/services_list/{serviceId}:** Individual service offerings with pricing
- **bookings/{bookingId}:** Booking records with status lifecycle (pending → accepted → paid)
- **chats/{chatId}/messages/{msgId}:** Real-time messages with sender ID and timestamp
- **payments/{paymentId}:** Transaction records with RazorPay payment IDs
- **reviews/{reviewId}:** User reviews with star ratings for vendor reputation management
- **usernames/{username}:** Username reservation for uniqueness enforcement

Security Architecture
Firestore security rules implement role-based access control ensuring data isolation between users and vendors. Critical security rules include: user documents are readable by any authenticated user (required for vendor chat name resolution) but writable only by the owning user; vendor documents are publicly readable (enabling discovery) but writable only by the owning vendor; chat documents and messages are accessible by any authenticated user but chatId construction (userId_vendorId) ensures implicit access control; and username documents are publicly readable for uniqueness validation but write-protected to prevent hijacking.

IV. KEY TECHNICAL IMPLEMENTATIONS

A. FCM V1 Push Notification System

The application implements push notifications using the Firebase Cloud Messaging HTTP V1 API, which supersedes the deprecated Legacy FCM API that relied on static server keys. The V1 API requires short-lived OAuth2 Bearer tokens obtained through a Service Account credential exchange, providing significantly improved security through time-limited access tokens and eliminating the risk of permanent server key exposure.

The implementation follows a four-step process executed on a background thread: (1) the Service Account private key is read from the application's assets directory at runtime; (2) a JWT is constructed with the service account email as the issuer, FCM messaging scope, and 3600-second expiration, then signed using RS256 (SHA-256 with RSA); (3) the JWT is exchanged for a short-lived OAuth2 access token via the Google token endpoint; and (4) the access token is used as a Bearer token in the FCM V1 HTTP POST request. Access tokens are cached for 55 minutes to minimize authentication overhead across multiple notification dispatches.

The notification system differentiates between three notification types — booking/payment notifications (dispatched via sendToUser() with Firestore FCM token lookup) and chat notifications (dispatched via sendChatNotification() with direct token passing) — with corresponding Android notification channels: channel_bookings, channel_payments, and channel_chat. This channel separation allows users to configure notification preferences independently for each event type.

B. Real-Time Chat Architecture

The real-time chat system employs a deterministic chat document ID formula: chatId = userId + "_" + vendorId. This approach ensures that regardless of which party initiates a conversation, exactly one conversation thread exists per user-vendor pair, eliminating duplicate chat creation and simplifying conversation lookup.

Firestore's addSnapshotListener() is applied to the messages subcollection ordered by timestamp in ascending order. The listener responds to DocumentChange.Type.ADDED events, processing only newly inserted messages rather than re-processing the complete message history on each update. This incremental update approach provides sub-second message delivery without the overhead of polling or full collection reloads.

The MessageAdapter implements a two-view-type RecyclerView pattern distinguishing sent (TYPE_SENT) and received (TYPE_RECEIVED) messages based on a comparison between the message's senderId and the currently authenticated user's UID. Both the ChatActivity (user perspective) and VendorChatActivity (vendor perspective) reuse the same activity_chat.xml layout and MessageAdapter, with the currentUserId parameter determining message alignment from each participant's viewpoint.

C. Username Uniqueness System

The application implements a username uniqueness enforcement system that persists usernames in a dedicated Firestore collection (usernames/{username} → {uid}). During registration, a two-step validation is performed: (1) client- side format validation ensuring the username contains only alphanumeric characters and underscores within acceptable length bounds; and (2) a Firestore document existence check against the usernames collection to verify availability. Upon successful registration, both the user document (users/{uid}.username) and the username reservation document (usernames/{username}) are written atomically, preventing race conditions in concurrent registration scenarios.

© 2026 IJSRET

D. Vendor Earnings Analytics

The vendor earnings dashboard implements a programmatic bar chart using Android's View class hierarchy without external charting libraries. Monthly earnings data is aggregated from the bookings collection by filtering paid bookings by month-year extracted from the paidAt timestamp field. The chart dynamically scales bar heights proportional to the maximum monthly earnings value, with bars rendered as GradientDrawable objects with rounded top corners. This approach eliminates the dependency on third-party chart libraries while providing a lightweight, customizable visualization component.

V. APPLICATION MODULES AND FEATURES

The application comprises thirteen distinct functional modules serving both user and vendor roles. Table II provides a comprehensive overview of all implemented modules with their corresponding features and underlying technologies.

Table II: Application Modules and Feature Overview

Module	Key Features	Technology
Authentication	Register with unique username, Login, Forgot password, Google Sign-In	Firebase Auth + Firestore
User Dashboard	Wedding countdown, Spend progress, Booking stats, Activity feed bell	Firestore, CountDownTimer
Vendor Discovery	Category listing, Real-time search, Detail view with image slider, Reviews	Firestore queries, ViewPager2
Booking System	Service selection chips, Date picker, Status tracking (pending/accepted/paid)	Firestore, FCM notifications
Real-Time Chat	Bidirectional messaging, Chat history, Message edit/delete, Chat lists	Firestore snapshot listeners
Push Notifications	Booking, payment, and chat notifications with deep-link navigation	FCM V1 API, JWT OAuth2
Payment Gateway	RazorPay checkout, Success/failure handling, Transaction records	RazorPay Android SDK
Vendor Dashboard	Earnings card, Stats row, Monthly bar chart, Transaction history	Firestore aggregation
Guest List	CRUD operations, RSVP status, Member count, Search/filter	Firestore subcollections
Budget Tracker	Expense recording, Progress visualization, Within/Over budget status	Firestore, ProgressBar
Checklist	Task creation, Completion tracking, Category management	Firestore subcollections
Review System	Star rating dialog, Review text, Vendor avg rating recalculation	Firestore transactions
Activity Feed	Recent activity bottom sheet, Mark as read, Emoji-categorized entries	Firestore activities

VI. RESULTS AND PERFORMANCE ANALYSIS

A. Feature Implementation Outcomes

All thirteen planned application modules were successfully implemented and integrated within the development timeline. The application was tested across five Android devices spanning API levels 29–34 (Android 10–14), demonstrating consistent functionality and visual fidelity across the tested device range. Dark mode support using Android's DayNight theme was validated across both light and dark system preferences.

B. Performance Evaluation

Table III: Performance Evaluation Metrics

Performance Metric	Observed Result	Benchmark
Real-time message delivery latency	< 800ms (WiFi), < 1.5s (4G)	< 2 seconds
FCM notification delivery time	< 3 seconds (foreground/background)	< 5 seconds
Vendor list load time (50 vendors)	< 1.2 seconds	< 2 seconds
Image load time (Cloudinary CDN)	< 900ms (cached)	< 1.5 seconds
Payment checkout initiation	< 1.5 seconds	< 2 seconds
Firestore read operations/day (free tier)	< 20,000 (testing phase)	50,000 limit
App cold start time	< 2.1 seconds	< 3 seconds

C. Comparative Analysis with Existing Systems

Table IV: Comparative Analysis of Wedding Planning Applications

Parameter	Wedding Wire	Shaadi.com	Local Platforms	Happenings & Happiness
Real-Time Chat	No	No	Limited	Yes (FCM + Firestore)
Vendor Booking	Yes	Partial	Yes	Yes (Full lifecycle)
Payment Gateway	Yes (US)	No	Partial	Yes (RazorPay/India)
Push Notifications	Email only	SMS	Limited	FCM V1 (all events)
Guest Management	Basic	No	No	Full CRUD + RSVP
Budget Tracking	Yes	No	No	Yes + Visualization

© 2026 IJSRET

Vendor Analytics	No	No	No	Earnings + Charts
India Market Focus	No	Partial	Yes	Yes (INR + Indian vendors)
Dual Role (Single App)	No	No	No	Yes

D. Security Evaluation

The application's security architecture was evaluated against common mobile application security risks. FCM V1 API adoption eliminates the static server key vulnerability present in legacy FCM implementations, as OAuth2 access tokens expire after 3600 seconds and are never embedded in the application binary. Firestore security rules prevent unauthorized cross-user data access, validated through manual penetration testing scenarios including attempting to read another user's guest list and modifying another vendor's services. The service_account.json file is included in .gitignore to prevent accidental exposure through version control systems.

VII. CHALLENGES AND SOLUTIONS

Table V: Technical Challenges and Solutions

Challenge	Root Cause	Solution Applied
FCM Legacy API deprecation	Google deprecated server key auth in June 2024	Implemented JWT + OAuth2 Service Account for FCM V1
Chat list race condition	Async Firestore fetches completing out of order	AtomicInteger counter pattern waiting for all fetches before UI update
Vendor subcollection permission denied	Firestore rules missing nested collection match	Added wildcard match /{subcollection}/{docId} within vendor rules
Username fetch showing 'User'	Firestore read rule blocking cross-user reads	Changed users read rule to allow any authenticated read
Card shadow clipping in XML	Parent layout lacked padding for elevation shadow	Added paddingTop/Bottom to all card row LinearLayouts
Cloudinary init crash	MyApp class not registered in AndroidManifest	Added android:name='.activities.MyApp' to application tag
Service account key in GitHub	Accidental git add before .gitignore configuration	Used git rm --cached and added correct .gitignore entry

VIII. ADVANTAGES AND LIMITATIONS

A. Key Advantages

- End-to-end digitization of the wedding planning workflow within a single application
- Dual-role architecture eliminates the need for separate vendor and user applications
- FCM V1 API implementation provides secure, scalable push notification delivery
- Real-time chat with deterministic chatId eliminates duplicate conversation threads
- Firebase free tier supports up to 50,000 reads/day, enabling cost-effective scaling for small to medium deployments
- RazorPay integration provides UPI, card, and net banking support suitable for the Indian market
- Cloudinary CDN ensures fast image delivery through geographically distributed edge servers

B. Current Limitations

- Application is limited to the Android platform; iOS support requires separate development
- Real-time features require active internet connectivity; offline support is limited to cached data
- FCM V1 implementation includes the service account private key in the application assets, requiring APK obfuscation for production deployment
- The bar chart implementation, while functional, lacks interactive features available in dedicated charting libraries

Future Scope

Several enhancements are identified for future development phases. Artificial intelligence integration represents the most significant opportunity: a recommendation engine trained on user preferences, budget constraints, and booking history could surface relevant vendors with higher precision than category-based browsing. A conversational AI wedding planner chatbot could provide personalized planning guidance, vendor suggestions, and timeline management through natural language interaction.

Cross-platform development using Flutter would extend the application's reach to iOS users while maintaining a shared codebase. Video calling integration using WebRTC would enable virtual vendor consultations, reducing the barrier for remote vendor discovery. Google Maps integration would provide location-based vendor discovery with distance filtering and navigation support.

Advanced analytics capabilities including revenue forecasting for vendors, spending trend analysis for users, and guest attendance prediction based on RSVP patterns would add significant value for power users. Integration with digital invitation platforms and automated guest communication workflows would further reduce manual coordination overhead for event organizers.

IX. CONCLUSION

This paper presented Happenings and Happiness, a comprehensive Android-based wedding and event planning platform that successfully addresses the fragmentation and inefficiency characterizing the current state of India's wedding planning ecosystem. The application delivers a production-

© 2026 IJSRET

ready solution integrating real-time communication, push notifications, payment processing, media management, and comprehensive planning tools within a unified dual-role architecture.

The technical contributions of this work include a secure FCM V1 push notification implementation using JWT-based OAuth2 authentication, a real-time bidirectional chat system with deterministic conversation management, a vendor earnings analytics dashboard with programmatic chart rendering, and a robust Firestore security architecture supporting role-based access control in a multi-tenant environment.

Performance evaluation demonstrates that the system meets established benchmarks for real-time message delivery, notification dispatch, and UI responsiveness across Android 10–14 devices. Comparative analysis confirms that Happenings and Happiness provides a more comprehensive feature set than existing solutions for the Indian wedding planning market, particularly in the areas of real-time communication, payment integration, and dual-role vendor management.

Future work will focus on AI-powered recommendation integration, cross-platform development, and advanced analytics capabilities to further enhance the platform's value proposition for both event organizers and service vendors.

Acknowledgment

The author expresses sincere gratitude to Ms. Shubhangi Dhaygude, Faculty Mentor at Parul Institute of Engineering & Technology, Parul University, and to Brainybeam Infotech Pvt. Ltd., Surat, for providing the technical guidance, development environment, and industry mentorship that made this work possible. The Firebase, Cloudinary, and RazorPay developer communities provided invaluable documentation and technical support resources.

REFERENCES

1. India Brand Equity Foundation, "Wedding Industry in India," IBEF Market Report, 2024.
2. StatCounter Global Stats, "Mobile Operating System Market Share India," StatCounter, 2024. [Online]. Available: https://gs.statcounter.com
3. Google Firebase, "Cloud Firestore Documentation," Google Developers, 2024. [Online]. Available: https://firebase.google.com/docs/firestore
4. R. Kaur, A. Singh, and P. Kumar, "Cloud-Based Mobile Event Management System," International Journal of Computer Applications, vol. 185, no. 12, pp. 1–6, 2023.
5. A. Singh and R. Sharma, "Real-Time Social Platform Development Using Firebase," International Journal of Engineering Research, vol. 9, no. 4, pp. 234–240, 2023.
6. D. Patel and S. Mehta, "Secure Push Notification Architecture for Android Applications Using FCM V1 API," Journal of Mobile Computing and Applications, vol. 11, no. 2, pp. 45–52, 2024.
7. V. Kumar, P. Sharma, and A. Joshi, "Digital Payment Integration in Indian Mobile Applications: A Comparative Study," IEEE Transactions on Mobile Computing, vol. 23, no. 1, pp. 112–124, 2024.
8. Google Firebase, "Firebase Cloud Messaging HTTP V1 API Reference," Google Developers, 2024. [Online]. Available: https://firebase.google.com/docs/cloud-messaging/migrate-v1
9. RazorPay, "Android Payment Integration Documentation," RazorPay Developer Docs, 2024. [Online]. Available: https://razorpay.com/docs/payments/payment-gateway/android-integration
10. Cloudinary, "Android SDK Integration Guide," Cloudinary Documentation, 2024. [Online]. Available: https://cloudinary.com/documentation/android_integration
11. B. Phillips, C. Stewart, and K. Marsicano, Android Programming: The Big Nerd Ranch Guide, 4th ed. Atlanta: Big Nerd Ranch, 2022.
12. OWASP Mobile Security Project, "Mobile Application Security Verification Standard," OWASP Foundation, 2023. [Online]. Available: https://owasp.org/www-project-mobile-app-security

© 2026 IJSRET