Basil V Mathew, 2026, 14:3
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology
An Open Access Journal

Atezhare: A Session-Based Cloud Integrated File Sharing Mobile Application

Basil V Mathew[1], Abineeth Tm[2], Inbarasu M[3], Koushick Ganapathi Venkat M[4], Dr. Rajesh Babu[5]

[1,2,3,4]UG Students, Department of Computer Science and Engineering
[5]Head of Department, Department of Computer Science and Engineering Tamilnadu College of Engineering, Coimbatore, Tamil Nadu, India
(Approved by AICTE & Affiliated to Anna University, Chennai | NAAC Accredited Grade 'A')

Abstract—The rapid growth of mobile technology has created significant demand for seamless, secure, and efficient file sharing solutions. Traditional file sharing methods often rely on internet connectivity, cloud servers, or third-party intermediaries, raising concerns about privacy, speed, and data security. This paper presents Atezhare, a session-based Hy (H-(H-P2P)) file sharing mobile application that enables direct device-to-device file transfer without the need for internet access or external servers. The application is built natively for Android and utilizes a dual-mode pairing mechanism comprising QR code scanning and a 6-digit alphanumeric session code to establish cryptographically bounded, time-limited connections between devices. Atezhare incorporates a cloud-integrated session management backend powered by Spring Boot and hosted on Render, with Supabase providing persistent session metadata storage. Experimental evaluations demonstrate that the system achieves reliable Hybrid-peer-to-peer connectivity with connection establishment times averaging 3–5 seconds, transfer speeds comparable to cloud-based alternatives under local wireless conditions, and enhanced data privacy due to the elimination of third-party server routing. The paper presents the system's architecture, module description, security mechanisms, implementation details, and comparative evaluation against existing file sharing solutions.

Keywords—Peer-to-Peer; File Sharing; Android Application; QR Code; Session-Based Authentication; Cloud Integration; Mobile Computing; Spring Boot; Supabase.

I. INTRODUCTION

The proliferation of smartphones and portable devices has fundamentally transformed information sharing paradigms. File sharing, once constrained to physical media such as USB drives or internet-dependent cloud platforms, has evolved to meet the demands of increasingly mobile and connectivity-diverse user populations. However, prevalent solutions continue to exhibit critical limitations that compromise user privacy, operational autonomy, and transfer efficiency.

Hybrid-peer-to-peer (H-(H-P2P)) file transfer technology presents a compelling alternative by enabling direct communication between endpoint devices over local wireless networks. This approach eliminates the intermediary server dependency, thereby reducing latency, improving throughput, and preserving data privacy. Nevertheless, existing (H-P2P) tools frequently suffer from poor user experience, inadequate security mechanisms, and platform-specific restrictions.

Atezhare is a mobile application designed to address these challenges by delivering a simple, fast, and secure (H-P2P) file sharing platform for Android devices. The application employs a dual pairing mechanism — QR code scanning and 6-digit session code entry — to establish secure, session-bounded connections. A cloud-integrated backend implemented in Spring Boot manages session lifecycle events, while Supabase provides scalable data persistence for session metadata.

This paper describes the design, architecture, implementation, and evaluation of Atezhare. Section II presents the problem statement; Section III reviews the existing systems; Section IV describes the system workflow; Section V presents the proposed system

architecture; subsequent sections cover modules, security, implementation, testing, results, and conclusions.

II. PROBLEM STATEMENT

Despite the availability of numerous file sharing applications, users continue to encounter several unresolved challenges with existing solutions:

- Dependence on internet connectivity for file transfer operations, rendering solutions unavailable in offline or restricted-network environments.
- Privacy concerns arising from mandatory upload of files to third-party cloud servers prior to delivery to the intended recipient.
- Reduced transfer speeds due to upload-download routing through geographically distant servers rather than direct local transmission.
- Compatibility limitations across heterogeneous operating systems and device generations.
- Absence of robust, user-friendly pairing mechanisms to prevent unauthorized connection establishment.
- Complex or cluttered user interfaces that hinder adoption by non-technical user demographics.

These limitations collectively demonstrate the need for a lightweight, offline-capable, and security-conscious file sharing application capable of operating entirely over local wireless connections without compromising data integrity or user privacy.

III. EXISTING SYSTEMS

The contemporary file sharing landscape encompasses multiple categories of solutions, each exhibiting characteristic limitations. Cloud-based services — including Google Drive, Dropbox, and Microsoft OneDrive — mandate continuous internet connectivity and persist user data on remote servers, introducing latency, storage cost, and privacy exposure.

Messaging platforms such as WhatsApp and Telegram support file sharing but impose size restrictions and route all data through centralized infrastructure, precluding offline operation and raising regulatory data residency concerns. Dedicated Hybrid-peer-to-peer transfer utilities such as SHAREit and Xender offer partial offline functionality via WiFi Direct or mobile hotspot connections, but are widely criticized for embedding excessive advertising, bundled bloatware, and exhibiting documented privacy vulnerabilities.

Bluetooth-based sharing, while universally available, is constrained by severely limited transfer speeds (approximately 2–3 Mbps) and short operational range. Additionally, most existing solutions require manual IP entry or rely on broadcast discovery protocols that expose device presence to unintended peers.

Table I presents a structured comparison of Atezhare against representative existing systems:

Table I: Comparative Analysis of File Sharing Systems

Feature	Atezhare	SHAREit	GDrive	BT Share
Offline Mode	✓	Partial	✗	✓
No Cloud Server	✓	✓	✗	✓
Secure Pairing	✓ QR+Code	Partial	Link-based	PIN
Session Expiry	✓	✗	✗	✗
Ad-Free	✓	✗	✓	✓
High Speed (LAN)	✓	✓	Limited	✗

IV. SYSTEM WORKFLOW

The operational workflow of Atezhare follows a clearly defined sequence of steps governing session establishment, file transmission, and session termination. The complete workflow proceeds as follows:

1. User launches the application and authenticates using a registered User ID and Password credential pair.

2. The Sender navigates to the Send module and selects one or multiple files from device storage for transmission.
3. The Sender initiates a transfer session via the Spring Boot backend, generating a unique 6-digit session code and corresponding QR code with a time-bounded expiry.
4. The Receiver navigates to the Receive module and scans the displayed QR code or manually enters the 6-digit session code.
5. The backend validates the session code, transitions the session status from PENDING to ACTIVE, and facilitates session handshake metadata exchange.
6. A secure Hybrid-peer-to-peer socket connection is established directly between the two devices over the local wireless network.
7. Files are transmitted in chunked streams directly between devices, with real-time progress monitoring displayed on both endpoints.
8. Upon completion, the session status transitions to DELIVERED, and completed transfers are recorded in the device's sent and received file history. The session code is invalidated immediately, preventing replay use.
Fig. 1 illustrates the application screenshots captured from the final review demonstration, showing the Home Screen, QR Code pairing screen, file selection interface, and transfer mode selection dialog.

Fig. 1: Atezhare Application Screenshots — (a) Home Screen, (B) QR Code Pairing, (c) File Selection, (d) Transfer Mode

V. PROPOSED SYSTEM

Atezhare is a Hybrid-peer-to-peer file sharing mobile application establishing direct device-to-device connections over local wireless networks. The system eliminates internet dependency and external server routing for file data, transmitting content directly between paired devices.

A. System Architecture

Atezhare follows a hybrid decentralized architecture. File payload data travels exclusively via direct (H-P2P) socket connections between devices, while session lifecycle metadata (session creation, pairing validation, expiry) is managed by a centralized Spring Boot backend hosted on Render and persisted in Supabase. This design ensures that sensitive file content is never exposed to external servers while maintaining robust session management and auditability.

B. The Architecture Comprises Four Primary Layers:

• Application Layer (Android): Handles UI rendering, user input, navigation, and event management using Android SDK components and Material Design 3 guidelines.
• Session Management Layer (Spring Boot / Render): Exposes RESTful API endpoints for session creation, code validation, status updates, and session expiry enforcement.
• Data Persistence Layer (Supabase): Stores session metadata records including session code (partition key), session status (PENDING / ACTIVE / DELIVERED / EXPIRED), receiver and sender user identifiers, file metadata, creation timestamp, and expiry timestamp.
• Transfer Layer: Manages direct (H-P2P) socket-based chunked file streaming between paired devices following session authentication.

C. Main Users

The system defines two primary user roles. The Sender is the device initiating file transfer operations; it generates session codes, selects files, and manages outbound transfers. The Receiver is the device accepting incoming file transfers; it presents a QR code or code entry interface to establish the session and manages inbound file storage.

D. Core Functionalities

• Secure user authentication with User ID and Password with BCrypt-hashed credential storage.
• Dual-mode device pairing via QR code scanning (ZXing library) and manual 6-digit code entry.

File	**selection,**	**queuing, and batch**	**transfer supporting all file types.**

• Session lifecycle management with automatic expiry and single-use code invalidation.
• Comprehensive sent and received file history management.
• Application settings, preferences, and account management.

VI. MODULE DESCRIPTION

Login Module

The Login Module provides a secure authentication interface requiring entry of a valid User ID and Password. Credential validation is performed against stored BCrypt hashes. Invalid credential attempts trigger error feedback; repeated failures enforce a temporary lockout policy to mitigate brute-force attempts.

Send Module

The Send Module enables users to select one or multiple files from device storage for transmission. Selected files are queued and displayed with metadata including file name and size. Upon session establishment with a receiver, files are transmitted sequentially via the (H-P2P) socket layer with real-time progress updates.

Receive Module

The Receive Module presents a dynamically generated QR code and 6-digit session code for display to the sender. Upon successful pairing, the module manages incoming socket streams, writes received files to the designated local directory, and updates the received file history log.

Home Module

The Home Module serves as the central navigation hub, providing access to all primary application features via quick-action buttons and a bottom navigation bar. It displays the user's recent transfer activity summary and application status indicators.

Directory Module

The Directory Module enables file system navigation for file selection prior to transfer. It renders directory hierarchies with file metadata — name, type, and size — supporting multi-file selection for batch transfers.

Settings Module

The Settings Module provides configurable preferences including default save directory, notification preferences, account information display, and application logout functionality. It also exposes version information and support contact details.

VII. SECURITY MECHANISMS

Atezhare employs a multi-layered security architecture designed to protect user data and constrain file transfers exclusively to explicitly authorized device pairs.

• Session Code Integrity: The 6-digit session code is generated using a cryptographically seeded random number generator. Each code is single-use and expires immediately upon successful connection establishment or after a configurable timeout period, preventing replay attacks.
• Physical Proximity Enforcement: QR code scanning requires the receiver to possess physical line-of-sight access to the sender's screen, providing an implicit physical access control layer that cannot be circumvented remotely.
• Direct Transfer Isolation: All file payload data is transmitted exclusively via direct (H-P2P) socket connections between paired devices. No file data traverses any external server, eliminating third-party interception vectors.

User	**Authentication:**	**Application-level authentication prevents unauthorized use of stored**

• Session Status State Machine: The backend enforces strict session state transitions (PENDING → ACTIVE → DELIVERED → EXPIRED), preventing session reuse or unauthorized state manipulation.
• Transport Security: All communications with the Spring Boot backend are enforced over HTTPS with TLS 1.3, and HTTP Strict Transport Security (HSTS) headers prevent protocol downgrade attacks.

VIII. IMPLEMENTATION

A. Technologies Used

Atezhare version 1.2 is implemented as a native Android application using Kotlin with the Android SDK targeting API level 33 and above. The user interface adheres to Material Design 3 (M3) guidelines, ensuring a modern, accessible, and intuitive experience across device form factors.

QR code generation and scanning is implemented using the ZXing (Zebra Crossing) open-source barcode library, providing fast and reliable camera-based code decoding. Hybrid-peer-to-peer file transfer is conducted via Java NIO socket connections established over the local wireless network interface following session handshake completion.

The Spring Boot 3.x backend exposes RESTful session management endpoints and is deployed on Render's cloud platform as a managed container service. Supabase provides the PostgreSQL-backed relational data store for session metadata, accessed via the Supabase Java client library. Table II summarizes the complete technology stack.

Table II: Technology Stack

Technology	Version	Role
Android SDK (Kotlin)	API 33+	Mobile client application
Android Studio	Hedgehog	IDE for Android dev
ZXing Library	3.5.x	QR code gen/scan
Spring Boot	3.2.x	Session mgmt backend
Render	Cloud PaaS	Backend hosting
Supabase	PostgreSQL	Session data persistence
Material Design 3	M3	UI/UX design system

B. System Data Model

The Supabase session table maintains the following schema per transfer session: id (UUID primary key), sessionCode (6-digit unique string used in QR), senderUserId, receiverUserId, status (WAITING / FILE_UPLOADED / DELIVERED / EXPIRED), fileUri or fileId, message (optional), createdAt, and expiresAt.

This schema directly reflects the TransferSession entity shown in the system flowchart from the final review documentation.

C. Application Ui Overview

The application presents four primary screens navigable via a bottom navigation bar: Home, Manage (file browser), Shared (history), and Me (account/settings). The Send flow presents a file selection list with checkboxes, a session initiation button, and a transfer mode dialog offering 'Live' (immediate transfer while sender is active) and 'Countdown' (file accessible until session expiry) modes. The Receive flow displays the pairing QR code and a manual code entry form with a 'Submit Code' action.

IX. TESTING

Unit Testing

Unit testing validated individual components in isolation, including the session code generator, QR code encoder/decoder pipeline, file queue manager, and authentication logic. Each module was verified against boundary conditions, empty inputs, maximum file size inputs, and concurrent access scenarios.

Integration Testing

Integration testing verified cross-layer interactions between the Android client, Spring Boot session backend, Supabase data store, and the (H-P2P) socket transfer layer. Critical scenarios included session creation and code validation roundtrips, transfer state machine transitions, session expiry enforcement, and file integrity verification via SHA-256 checksum comparison of source and received files.

User Acceptance Testing (UAT)

UAT was conducted with a cohort of 20 end users performing representative real-world scenarios including account registration, device pairing via both QR and manual code entry, single and batch file transfers across multiple file types and sizes (10 KB to 500 MB), and history review. User feedback informed interface refinements in the final release version.

Performance Testing

Performance testing measured connection establishment latency, file transfer throughput, and application responsiveness under varying network conditions and file sizes. Testing was conducted over a standard IEEE 802.11n Wi-Fi network at 2.4 GHz and 5 GHz bands. Results confirmed consistent performance within accepted operational parameters.

Security Testing

Security testing verified that session codes cannot be reused following successful pairing, that sessions expire correctly at the configured timeout, that unauthenticated API calls to the Spring Boot backend are rejected with HTTP 401 responses, and that no file payload data is logged or persisted on any external server infrastructure.

X. RESULTS AND DISCUSSION

The system was evaluated across comprehensive real-world scenarios encompassing user authentication, dual-mode device pairing, single and batch file transfers, history management, and session expiry enforcement. Experimental results validate the system's performance claims and confirm suitability for production deployment.

Table III presents quantitative performance metrics measured during evaluation:

Table III: Experimental Performance Results

Metric	Result	Notes
Session establishment time	3–5 sec	QR scan path
Code entry pairing time	5–8 sec	Manual entry path
Transfer speed (LAN, 5GHz)	~22 MB/s	100MB file, avg.
Transfer speed (LAN, 2.4GHz)	~9 MB/s	100MB file, avg.
Session code replay attack	0% success	Single-use enforced
File integrity check pass rate	100%	SHA-256 verified
UAT satisfaction score	4.3 / 5.0	n=20 users

Transfer speeds measured over a standard local wireless network were substantially superior to Bluetooth-based alternatives (2–3 Mbps) and comparable to cloud upload-download cycles for local network scenarios, with the additional benefit of eliminating internet dependency. The user interface received consistently positive feedback for its clarity and ease of use, with particular commendation for the dual pairing mechanism and real-time progress display.

The simulation and implementation screenshots presented in Fig. 1 (extracted from the project final review documentation, dated 09.04.2026, Tamilnadu

College of Engineering) demonstrate the functional application across the complete user interaction flow. The Atezhare application successfully handles the end-to-end transfer lifecycle from home screen navigation through QR-based pairing to file selection and transfer mode configuration.

XI. FEATURE ENHANCEMENTS

While the current version of Atezhare provides a robust foundation for secure Hybrid-peer-to-peer file sharing, several enhancements are planned for future releases:

- Cross-platform support for iOS devices using React Native or Flutter to enable heterogeneous device pairing.
- End-to-end encryption of file payload data during (H-P2P) transfer using AES-256 with session-derived symmetric keys.
- Transfer resume functionality utilizing byte-range checkpointing for interrupted sessions.
- Multi-device simultaneous transfer support enabling one-to-many broadcast file distribution.
- Integration with device contact lists for rapid pairing with previously connected devices.
- Dark mode support and additional Material You dynamic color themes.
- Transfer speed analytics, usage statistics dashboard, and data consumption reporting.
- Adaptive bitrate chunking to optimize transfer performance across heterogeneous network conditions.

XII. CONCLUSION

This paper has presented Atezhare, a session-based Hybrid-peer-to-peer mobile file sharing application designed to enable fast, secure, and offline device-to-device file transfer on Android. By leveraging a dual pairing mechanism comprising QR code scanning and 6-digit session code entry, the application provides a simple yet robust connection establishment process that eliminates internet connectivity requirements and external server routing for file payloads.

The cloud-integrated session management backend, implemented in Spring Boot and deployed on Render with Supabase data persistence, provides scalable session lifecycle management while preserving the privacy-first architecture of the direct (H-P2P) transfer model. Atezhare addresses critical limitations of existing cloud-based and Bluetooth file sharing solutions, demonstrating improved speed, privacy, and user experience.

Experimental evaluation confirms reliable performance across diverse file types and sizes, 100% file integrity preservation, robust replay attack prevention, and strong user acceptance. Future work will focus on cross-platform extension, payload encryption, and multi-device transfer capabilities.

REFERENCES

1. A. Singh and P. Gupta, "Hybrid-peer-to-peer File Sharing Systems: A Survey," International Journal of Computer Applications, vol. 150, no. 3, pp. 1–6, 2016.
2. M. Reza and S. Khan, "Design and Implementation of a Wireless File Transfer Application for Android Devices," in Proc. IEEE International Conference on Mobile Computing, 2019, pp. 78–84.
3. Y. Zhang and W. Lee, "Secure Hybrid-peer-to-peer File Sharing Using End-to-End Encryption," in Proc. IEEE International Conference on Information Security and Privacy Applications (ISPA), 2019, pp. 112–119.
4. S. Kumar and R. Patel, "Session-Based Authentication for Secure Data Transmission," IEEE Transactions on Network and Service Management, vol. 17, no. 3, pp. 1842–1855, Sep. 2020.
5. L. Chen, M. Wang, and H. Zhou, "Secure Mobile File Sharing Using WiFi Direct," in Proc. IEEE International Conference on Mobile Computing and Networking (MobiCom), 2021, pp. 45–52.
6. S. Androutsellis-Theotokis and D. Spinellis, "A Survey of Hybrid-peer-to-peer Content Distribution Technologies," ACM Computing Surveys, vol. 36, no. 4, pp. 335–371, 2004.

7. G. Tzenakis et al., "Efficient File Transfer Protocols for Mobile Devices in Local Networks," Mobile Networks and Applications, vol. 22, pp. 1–12, 2017.
8. W. Stallings, Cryptography and Network Security: Principles and Practice, 8th ed. Upper Saddle River, NJ, USA: Pearson Education, 2020.
9. A. S. Tanenbaum and D. J. Wetherall, Computer Networks, 6th ed. Upper Saddle River, NJ, USA: Pearson Education, 2021.
10. ZXing (Zebra Crossing) Barcode Library, GitHub Repository. Available: https://github.com/zxing/zxing [Accessed: Apr. 2026].
11. Android Developer Documentation. Available: https://developer.android.com/docs [Accessed: Apr. 2026].
12. OWASP Foundation, "OWASP Mobile Security Testing Guide (MSTG)," OWASP, 2023. Available: https://owasp.org/www-project-mobile-security-testing-guide/