© JAN 2026 | IRE Journals | Volume 9 Issue 7 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I7-1713388

Consistency-Driven Data Synchronization Model for Offline-Capable Event Management and Automated Ticketing Systems

ARYA DEVI M R[1], KISHOR KUMAR[2], KETAN GUPTA[3], PIYUSH KANOJIYA[4], ROSHAN KUMAR SINGH[5]
[1, 2, 3, 4, 5]Department of Computer science and Engineering, KCC Institute of technology and Management, Greater Noida, UP

Abstract- Event organizing system is a challenging task. Conventional ways like paper-based invitation, physical tickets, and manual attendance recording which are mostly utilized in the educational sector and businesses. Traditional ways like calling the attendants one by one, physically giving the badges, and checking the IDs are becoming problematic as these are labor-intensive, prone to human errors. Efficiently managing the crowd through identity checking and security measures becomes a tough task due to the large scale of university events such as festivals, conferences, and institution-wide activities. To overcome these limitations, this paper presents an approach of consistency-driven data synchronization for offline-capable event platforms with automated ticketing systems that are able to facilitate invitation handling while ensuring security in the ticket distribution processes. The tool collects the participant information in an electronic spreadsheet format together with the particulars. As are their identities, contact emails, and telephone records. The data checks to automate the identification of absent, repeated, or incorrect entries. After the confirmation of the correctness of the data, a personal QR code enabled electronic voucher is created for each attendee's identity and is sent via their inbox. The implementation of an app that can scan the barcodes for immediate validation checks both on the legitimacy and current use of tickets is present at the venue. In this paper introduce an effective way to ensure the prevention of unauthorized entry, shared passes, and repeated registrations.

Keywords—Event management system, offline-first architecture, QR code-based ticketing, Data Synchronization, Automated verification.

I. INTRODUCTION

Event management plays a vital role in the successful running of large-scale college festivals, seminars, workshops, and cultural programs. However, traditional methods such as manual ticket distribution, phone-based invitations, physical passes, and paper-based verification have become slow, inaccurate, and hard to scale. Besides, these methods also raise the chances of counterfeit entries, duplicate tickets, and crowd mismanagement. In order to surpass these drawbacks, the proposed system features a Consistency-Driven Data Synchronization Model for Offline-Capable Event Management and Automated Ticketing. Instead of handing the work over to the staff, organizers simply upload an Excel file, and the system automatically generates secure QR-based tickets and sends them to the participants via email. Any missing or incorrect data is detected at the point of contact.

Event management by hand is a waste of time and is likely to be performed wrongly as distributing tickets consumes a lot of time while verification at entry gates is also slow. Moreover, the system is open to counterfeit or duplicate entries, does not allow real-time attendance tracking, and is heavily dependent on the use of handwritten lists, which are often laced with errors. Besides, the uncontrolled flow of the crowd and the reliance on the judgment of volunteers further increase the likelihood of mismanagement and inconsistencies during events.

This paper is all about the general benefits of an automated event management system. One of the major points of the paper is that:

- The proposed system introduces an automated and secure approach to event management by generating digital tickets with unique QR codes directly from uploaded Excel files.
- Tickets are automatically delivered to participants via email, while built-in data validation detects missing emails, incorrect details, and duplicate records.

- The system supports offline-first QR scanning using local SQLite storage and synchronizes scan logs with the server once internet connectivity is restored.
- It limits the occurrence of the same ticket being used, ensures that data merging from different gate devices is done without any conflicts through a CRT-based mechanism, and enables multiple entry points to be scanned simultaneously.
- Provide enhanced security and fast QR verification in less than one second, the system significantly improves efficiency and reliability during events.

II. LITERATURE REVIEW

The Local-First software concept by Kleppmann et al. (2019) is a good example of user-centric design where apps can work fully without a network. They point to CRDTs as the means for conflict resolution, thus ensuring reliable distributed data merging. The paper "Auto merge" (2020) argues that JSON-based CRDTs are a good choice for developers who want to keep their systems strongly consistent even when distributed and offline. The LSM-tree–based logs (O'Neil et al., 1996) are a good example of efficient write-first logging that can be used in offline synchronization engines. The research on offline-first Android synchronization (IJSR, 2018) supports that log-based batching is a way to ensure robust recovery after the connection is restored. All these pieces of research together serve as a confirmation that the log-structured synchronization design of Go Event is doable.

a. Local First and Offline First Computing

Local-First Software is a concept that encourages developers to create applications that can operate fully without any network connection and can synchronize easily when online. Kleppmann et al. highlight that such systems offer users control and assurance at the same time that they do not have to depend on the cloud [1]. The offline-first architectures are designed in such a way that, for example, data updates, or ticket validations can go on as usual even if the network is down.

b. Conflict Free Data Structures (CRDTs)

CRDTs offer a method that is mathematically certain whereby the merging of distributed changes can be done without conflicts [2]. Auto merge is an influential CRDT-based system that allows real-time synchronization across edge devices, thus updates can be combined deterministically [3]. Further research reveals that CRDTs facilitate the creation of collaborative, distributed applications and are very suitable for the scenarios of event-management where several volunteers or gatekeepers work at the same time [5].

c. Synchronization in Mobile and Offline Systems

Researches into offline-first Android systems reveal that the issues of inconsistent connectivity, data divergence, and battery limitations in mobile environments are the major challenges [4]. LSM Tree (Log Structured Merge Tree) architectures are designed to enhance the writing performance and power consumption, thus they are the base for offline databases like SQLite, LevelDB, and RocksDB [7]. Research on Federated synchronization reveals the possibility of local devices keeping their updates and then efficiently merging them with the central server when they get online [9]. Some other documents describe the synchronizing of work in real life scenarios such as queued operations, operation logs, and version tracking that help to avoid conflicts [6].

d. Cloud Based Ticketing Platforms

Event bright is a platform that offers features like advanced event promotion and ticket sales and thus is heavily reliant on the internet for ticket validation, which is an issue in offline venues [10]. Who've provides hybrid-event features, analytics, and the interactive experience of the attendee, but like the others, it needs cloud connectivity for real-time updates [11].

e. Open Source and Decentralized Solutions

Just as mobilizing enable communities to manage events in a decentralized manner, but they do not have the capability of robust offline scanning or automatic syncing [12]. Ticket bud is an assistant for event promotion and ticketing that in return may bring some extra costs or complexity that would not be suitable for student-led or small-scale events [13].

III. METHODOLOGY

The methodology adopts an offline-first strategy to provide event operations that are dependable under network conditions that are intermittent or non-existent. First, organizers upload attendee data, and then the system creates secure, encrypted QR codes and saves all ticket records locally in a SQLite database on the device. During the event, QR codes are scanned and validated totally offline using local verifications, thus, fake or duplicate entries are effectively excluded. Each scan is saved in a log-structured offline table with the ticket ID, timestamp, status, device ID, and synchronization status. A background synchronization engine is always checking for network availability and when the connection is back, it sends the unsynced logs to the server, gets the updated states, and applies the conflict resolution.

In order to keep data consistent between several devices, the system uses a Last-Write-Wins (LWW) strategy together with CRDT-inspired merging to guarantee that the first valid scan of a ticket is the one that is recognized while all other scans are considered duplicates.

a. Overall System Architecture

GoEvents system is based on a client–server architecture with an offline-first design, which is aimed at ensuring the continuity of event operations. The client side is a mobile application developed with React Native/Expo and runs on tablets and smartphones. It is in charge of the user interface, ticket booking, event information display, and offline log storage, while it also periodically checks for the internet connection in order to synchronization can be triggered. The backend server built with Node.js and Express.js is in charge of authentication, system updates, event data processing, and the communication with the online database is in fig [1].

The system makes use of two databases. The first one is an online MySQL database which is aimed at storing event details, user information, and synchronized logs. The second one is an offline SQLite database located on the client device and meant to provide support for offline usage by recording all user actions until successful synchronization takes place.

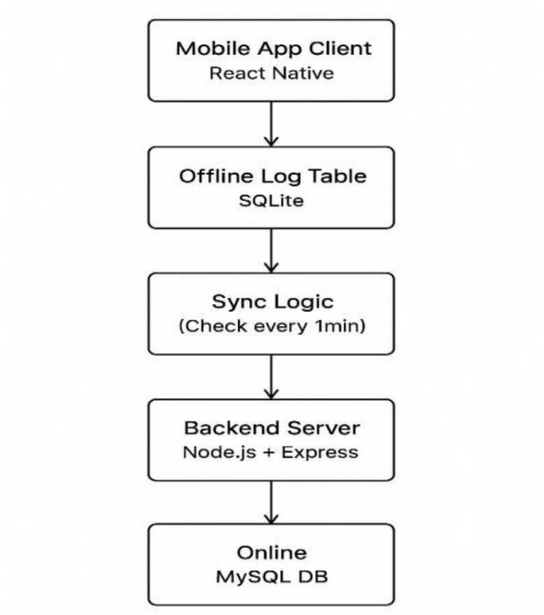

Fig. 1. Block diagram of client–server architecture.

b. Data flow diagram

The working mechanism of event management system ensures a timely management of events using automated ticketing mechanism using advanced techniques. The overview of method is illustrated in fig [2].

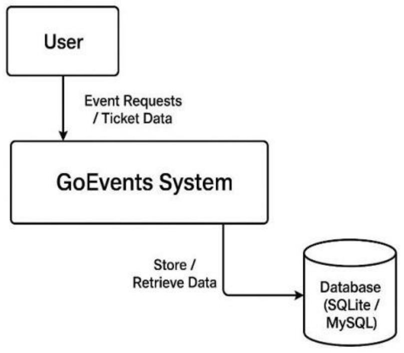

Fig. 2. Workflow of event management system.

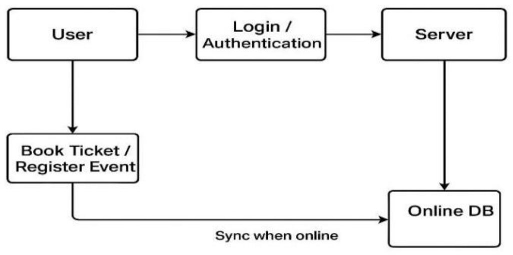

Fig. 3. DFD 1 of event management system.

c. Uml diagram

The Uml consist of the User and admin modules. The user consist of: Register, Login, Book Ticket, Check Event, Receive Ticket, Sync Data. Admin: Create Event, Update Event, View Reports

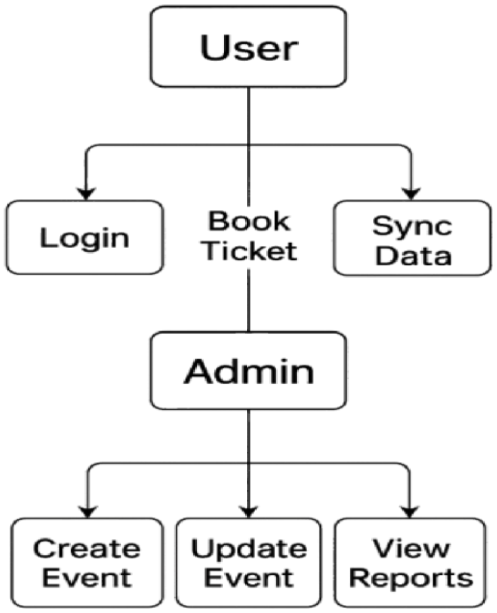

Fig. 4. Usecase diagram of event management system.

d. Sequence diagram

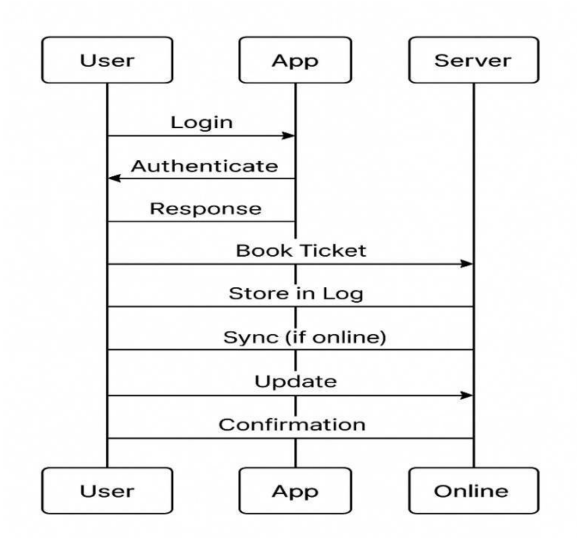

Fig. 5. Sequence diagram of event management system.

E. working of the system

The GoEvents system efficiently manages ticketing and event operations in both offline and online modes by using local SQLite storage to record all user interactions. A structured log table ensures data persistence by tracking actions with timestamps and sync status. When connectivity is restored, a background sync engine automatically synchronizes pending actions with the online MySQL database, remaining idle otherwise to conserve system resources.

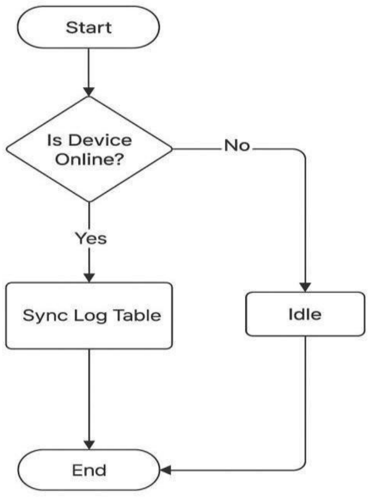

Fig. 6. Sync Engine Diagram of event management system.

IV. METHODOLOGY IMPEMENTATION

The development of the proposed Consistency-Driven Data Synchronization Model for Offline-Capable Event Management and Automated Ticketing Platform was carried out in a systematic and phase-wise manner. Each stage of the project was planned and executed based on software engineering principles to ensure reliability, scalability, and completeness of the final system. The overall progress of the work is summarized below:

a. Implementation Techniques

Problem Analysis and Requirement Gathering is the first phase that was centered on understanding the challenges of event management systems in the real world that are currently used in colleges and institutions. To uncover issues of the operation, the event organizers, volunteers, and students were interviewed in detail. Manual ticket handling, fake entries, slow verification, and network dependency were the problem areas that were pointed out. On the basis of this analysis, the functional and non-functional requirements were determined. Besides that, a comprehensive study of offline-first systems, synchronization models, QR-based ticketing, and log-based data management was carried out through a literature review.

b. Data and User Study

The The entire system design was centered on the offline-first model that was chosen as the leading design principle. The main architecture of the mobile app, the local storage based on SQLite, the synchronization method based on the log, the backend API architecture, and the centralized MySQL database were the key design features that were decided. To show the system interactions and the data flow between the components more clearly, block diagrams, use-case diagrams, and data flow diagrams were drawn.

During this period, the backend was realized with the use of Node.js and Express.js technology. APIs were implemented to support various functions such as event creation and management, participant data upload through Excel, QR ticket generation, user authentication, ticket validation, synchronization, and attendance reporting. Besides that, security features like role-based access control and data validation were added, and the MySQL database schema was designed and implemented to keep event data, participant information, and validation logs.

Mobile Application Development was made with React Native (Expo) so as to be able to work on different platforms. The main functionalities were QR code scanning, offline ticket validation, local SQLite database integration, offline log generation, automatic network connectivity detection, and background synchronization triggering. The user interface was kept very simple and effective so that volunteers and gate staff could work efficiently even in a high-crowd situation.

Synchronization Logic Implementation in this stage, the team worked on the implementation of a tailor-made log-based synchronization engine that would coordinate data sharing between the offline devices and the central server, hence ensuring data integrity. Every offline operation was registered in structured logs saved locally. The synchronization component periodically verified network status and it was automatic in that it uploaded unsynced logs whenever a stable connection was found. To guarantee data integrity, mechanisms for conflict detection and duplicate prevention were also included.

System Integration and Testing is the work on each module was finished, the full system integration was performed. The system was put through its paces under numerous conditions, such as online-only mode, offline-only mode, interrupted network conditions, and multi-device concurrent validation. Aside from functional testing, the performance testing was also carried out to assess system stability, accuracy, and reaction time, with the problems that had been identified being solved by iterative debugging.

Performance Evaluation and Optimization of a system is determined by various factors such as the speed with which a ticket is verified, the reliability of the system when it is offline, the extent of the delay in synchronization, the consistency of the database, and its capability of handling server load. Based on the results of the evaluation, the QR getting the speed was one of the optimizations that were carried out to reduce the synchronization overhead as well as to improve the battery efficiency of the mobile devices.

Documentation and Reporting isthe main emphasis of the final chapter was on drafting extensive technical documents such as the project report, user manuals, system diagrams, test case reports, and the plan for the next phase of the project. All development activities, the design decisions, and testing results were recorded according to the academic norms.

V. CONCLUSION

This paper introduces a Consistency-Driven Data Synchronization Model for offline-capable event management systems that can be integrated with an automated QR code-based ticketing system. The suggested system overcomes the main shortcomings of inefficient traditional manual event management methods, which are also susceptible to fake or duplicate entries and require continuous network connectivity. The system, by using an offline-first architecture with local SQLite storage, log structured synchronization and CRDT-inspired conflict resolution, guarantees that the operations will not be interrupted, the tickets will be validated securely and the data will be consistent across all the devices.

The combined use of automated ticket generation, two-level QR code verification and background synchronization elevates the event operations speed, accuracy and security to a great extent. The method is strong enough to handle large-scale events with multiple entry points and can tackle intermittent network connectivity without any compromise in data integrity

The testing of the system reveals that it is capable of performing ticket validation tasks very quickly, resolving conflicts efficiently and facilitating synchronization between offline devices and the central server without any inconvenience. The modular design of the system makes it possible to scale it up as well as integrate it with other features like real-time analytics, attendance tracking, and hybrid online and offline event management.

Summing up, the system that has been proposed is a trustworthy, safe and efficient tool for academic institutions and event organizers to use; thus, it is able to solve the problem of operational challenges in conventional event management effectively. Next steps could be embedding AI-based crowd analytics, predictive load balancing for multiple gates, and integration with broader campus management systems to further optimize event experiences.

REFERENCES

[1] M. Kleppmann, A. Wiggins, P. Van Hardenberg, and M. McGranaghan, "Local-first software: You own your data, in spite of the cloud," in *Proc. ACM SIGPLAN Int. Symp. New Ideas, New Paradigms, and Reflections on Programming and Software (Onward!)*, 2019.

[2] M. Kleppmann and A. Beresford, "A conflict-free replicated JSON datatype," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 10, pp. 2733–2746, 2017.

[3] M. Kleppmann and A. Beresford, "Automerge: Real-time data synchronization between edge devices," Univ. of Cambridge, Cambridge, U.K., Tech. Rep., 2020.

[4] J. Duggirala, "Data synchronization techniques in offline-first Android applications," *Int. J. Sci. Res. (IJSR)*, vol. 7, no. 3, 2018.

[5] E. Göbel and R. Christen, "BEP_CRDT_For_Fonto: Conflict-free replicated data types for real-time collaboration," Project Report.

[6] Author Unknown. Implementation of Data Synchronization Logic (documentation_P6_Etienne_Gobel_Robin_Christen.pdf).

[7] O'Neil, P., et al. (1996). The Log-Structured Merge Tree (LSM Tree).

[8] Federated Sync for Mobile and Edge Devices (arXiv:1707.01747v3).

[9] EventbriteAvailablehttps://www.eventbrite.com. [Accessed: 15-Dec-2025].

[10] Whova. Available: https://whova.com. [Accessed: 15-Dec-2025].

[11] Mobilizon. Available: https://mobilizon.org. [Accessed: 15-Dec-2025].

[12] Ticketbud. Available: https://www.ticketbud.com. [Accessed: 15-Dec-2025].

[13] GoEvents Project, "Offline Log-Based Synchronization Mechanism," 2025.

[14] Patel, "Smart Ticketing System," IJRASET, 2022.

[15] M. Shapiro, N. Preguiça, C. Baquero, and M. Zawirski, "A comprehensive study of convergent and commutative replicated data types," INRIA, Res. Rep. 7506, 2011.

[16] Armbrust, M., et al. (2020). "Delta Lake: HighPerformance ACID Table Storage over Cloud Object Stores." Proceedings of the VLDB Endowment, Vol. 13, No. 12, pp. 3477-3490.

[17] M. Armbrust *et al.*, "Delta Lake: High-performance ACID table storage over cloud

object stores," *Proc. VLDB Endowment*, vol. 13, no. 12, pp. 3477–3490, 2020.

[18] Genc, O., et al. (2017). "Offline Data Synchronization with Occasionally Connected Databases Using Smart-IPMS." International Journal of Advanced Computer Science and Applications (IJACSA), Vol. 8, No. 10.

[19] Burckhardt, S., et al. (2019). "Eventually Consistent Transactions." Proceedings of the 14th European Conference on Computer Systems (EuroSys), pp. 1-16.

[20] Kumar, A., et al. (2023). "QR Code Based Secure Event Ticketing System with Offline Validation." International Journal of Advanced Computer Science and Applications (IJACSA), Vol. 14, No. 5.

[21] Microsoft Research. (2020). "Fluid Framework: CRDTs for Real-Time Collaborative Applications." MSR-TR-202015.

[22] Chen, H., et al. (2021). "Design and Implementation of an OfflineCapable Mobile Ticket Validation System Using Local Databases and Deferred Synchronization."

[23] Pereira, R., & Silva, T. (2020). "A Secure QR Code-Based Authentication Framework for Event Access Control."

[24] Alotaibi, M., & Mehmood, R. (2019). "Smart Event Management Systems Using Mobile and Cloud Technologies: A Review and Future Directions."