STM JOURNALS

Journal of
Microelectronics and Solid State Devices

ISSN: 2455-3336
Volume 12, Issue 3, 2025
September–December
DOI (Journal): 10.37591/JoMSD

https://journals.stmjournals.com/jomsd

Research JoMSD

Building a Smart Chatting Platform for Social Networking: "Guftgu"

Ajay Sharma[1], Nandan Kumar[1], Surender[1], Stuti Saxena[2,*]

Abstract

This research paper details the development of a location-based social networking application with a focus on offline-first functionality. Guftgu is a communication application developed using React Native for the frontend and Node.js for the backend. It facilitates seamless, real-time conversations between strangers while prioritizing user privacy through complete anonymity. The platform's intuitive interface, cross-platform compatibility, and efficient backend architecture enable smooth performance and user engagement. Designed with privacy at its core, Guftgu eliminates the need for personal identification, making it ideal for open dialogue without judgment or exposure. Whether for casual chatting or sharing thoughts freely, the app ensures secure and uninterrupted communication. Its architecture supports scalability and robust interaction, creating a trusted space for anonymous social connection. This paper explores the design and implementation of a messaging application that ensures a consistent user experience despite network disruptions. It details the architecture and technology stack, highlighting the use of local data storage, message queuing, and synchronization techniques. These methods enable uninterrupted communication in both direct chats ('Guftgu') and location-based group broadcasts ('Mehfil'). The system is engineered to support seamless transitions between offline and online modes. Additionally, the platform emphasizes user anonymity, allowing individuals to connect with others in a specific geographic area without disclosing their identity, thereby fostering private yet meaningful social interactions.

Keywords: Offline-first, React Native, Node.js, location-based social networking, WatermelonDB, real-time communication, mobile application development

INTRODUCTION

Communication has undergone a radical transformation over the past few decades. Traditional forms of communication, such as letters and phone calls, have been replaced by modern instant messaging applications that enable real-time conversations regardless of geographical location. With the rise of social media, remote work, and online collaboration, the demand for seamless and instantaneous messaging has never been higher [1]. However, modern real-time chat applications, while offering instantaneous messaging, often struggle with network dependence. Users expect these applications to work seamlessly, especially in scenarios with poor or intermittent connectivity, such as those encountered in mobile social networking. This paper addresses the challenge of maintaining a seamless user experience in a location-based social networking application by implementing an offline-first strategy [2]. This approach prioritizes local data management and synchronization, allowing users to continue

***Author for Correspondence**
Stuti Saxena
E-mail: stutisaxena@eitfaridabad.co.in

[1]Student, Department of Computer Science and Engineering, Echelon Institute of Technology, Faridabad, Haryana, India
[2]Associate Professor, Department of Computer Science and Engineering, Echelon Institute of Technology, Faridabad, Haryana, India

Received Date: July 05, 2025
Accepted Date: July 31, 2025
Published Date: December 12, 2025

Citation: Ajay Sharma, Nandan Kumar, Surender, Stuti Saxena. Building a Smart Chatting Platform for Social Networking: "Guftgu". Journal of Microelectronics and Solid State Devices. 2025; 12(3): 32–37p.

messaging and interact with location-based groups even when a network connection is unavailable. The paper presents an in-depth study on the architecture, development, and key features of 'Guftgu,' a platform designed to connect users within a specific area, focusing on its offline capabilities. The goal is to create a platform that is efficient, secure, and capable of handling unreliable networks and large user bases. The application provides a platform for users to connect with new people in a specific area without revealing their identity, send friend requests, and chat.

OBJECTIVES

The primary objectives of this research are to:

a. Design an interactive and responsive user interface using React Native, with considerations for offline usability in direct chats ('Guftgu') and location-based group broadcasts ('Mehfil').
b. Implement WebSockets for real-time communication and incorporate a mechanism for offline message queuing and subsequent synchronization for 'Guftgu' and 'Mehfil' sections.
c. Ensure data privacy and security through encryption techniques, secure local data storage, and anonymous location-based interactions.
d. Develop a local data storage mechanism using WatermelonDB in React Native to persist messages, user data, and location-based group information offline.
e. Implement offline message queuing and synchronization to ensure messages in 'Guftgu' and 'Mehfil' are delivered once connectivity is restored.
f. Design conflict resolution strategies to handle potential discrepancies between local and server data in user profiles, direct chats, and group broadcasts.
g. Implement a map feature ('Map') that displays online users while optimizing data synchronization to minimize network usage and preserve offline functionality.

METHODOLOGY

System Architecture and Technology Stack

The 'Guftgu' application is designed with a two-tiered architecture, comprising a React Native frontend for the user interface and a Node.js backend for server-side logic and data management.

- *Frontend (React Native):* React Native enables cross-platform development for broad user accessibility. This choice allows for a single codebase to be used for both iOS and Android platforms, reducing development time and cost. For offline data persistence, 'Guftgu' utilizes WatermelonDB, a performant and scalable offline-first database built on top of SQLite. WatermelonDB's architecture is optimized for React Native, allowing for efficient storage and retrieval of messages, user data, and location-based group information [3–9]. The application logic is designed to interact primarily with this local database, ensuring that core functionalities like direct messaging ('Guftgu') and viewing location-based broadcasts ('Mehfil') remain accessible even without a network connection. Synchronization with the remote database occurs in the background when connectivity is available. The 'Map' section displays online users, and its data is also cached locally in WatermelonDB to maintain partial functionality offline. However, the 'Map' section's real-time updates are limited during offline periods to conserve data and battery life. Tailwind CSS is used for designing a responsive and modern user interface.
- *Backend (Node.js with Express.js):* Node.js, with its non-blocking, event-driven architecture, is well-suited for handling real-time communication using Socket.IO. Express.js facilitates the creation of RESTful APIs for user authentication, data synchronization, and managing location-based data. The backend is designed to handle message synchronization requests from clients, ensuring that messages sent during offline periods are correctly processed and delivered upon reconnection. MongoDB is used to store user data, message data, and location-based group information. MongoDB's flexible schema allows for efficient storage and retrieval of diverse data types [10]. The backend also implements conflict resolution logic to manage data inconsistencies that may arise from offline modifications. For the 'Mehfil' section, the backend uses a combination of real-time updates via Socket.IO (when online) and stored procedures to efficiently manage and distribute location-based broadcasts to users within a specific

geographical area. Load balancing and a microservices-based architecture are employed to ensure scalability and handle high user loads.

Workflow of Chat Application

- The 'Guftgu' application is divided into three main sections, each with distinct functionalities and workflows: 'Guftgu' (direct chats), 'Mehfil' (location-based broadcasts), and 'Map' (online user display) as shown in Figure 1.
- *Offline Workflow:*
- *Guftgu (Direct Chats):* This section enables one-to-one private conversations between users who have established a connection.
- *Offline Message Storage:* When a user sends a direct message in 'Guftgu' while offline, the message is immediately stored in the local WatermelonDB database on the device. The message is also added to an outbound message queue [11].
- *Offline Message Composition:* Users can continue to compose and send direct messages in 'Guftgu' even when offline. These messages are stored locally and added to the outbound queue, ensuring that no message is lost due to network unavailability.
- *Connectivity Monitoring:* The React Native application continuously monitors network connectivity. This is crucial for determining when to initiate the synchronization process..
- *Automatic Synchronization:* When the application detects that the device has regained network connectivity, it initiates a synchronization process. The application sends the messages in the outbound queue to the Node.js backend via a dedicated API endpoint [12].
- *Server-Side Processing:* The Node.js backend receives the offline messages, stores them in MongoDB, and uses Socket.IO to deliver them to the intended recipients (if they are online). This ensures that offline messages are integrated into the real-time communication flow.
- *Data Consistency:* After sending the queued messages, the client requests any new 'Guftgu' messages from the server and updates the local WatermelonDB database. This bi-directional synchronization ensures that the local data is consistent with the server data.
- *Conflict Resolution:* In the event of a conflict (e.g., a message being edited offline and also being updated by another user online), the system employs a last-write-wins strategy, with the server's version taking precedence, and the client is notified of the change. This simplifies conflict resolution in most scenarios [13].
- *Mehfil (Location-Based Broadcasts):* This section allows users to broadcast messages to other users within a specific geographical area. It is designed to facilitate community interaction and information sharing among people in close proximity.
- *Offline Broadcast Storage:* When a user sends a location-based broadcast in 'Mehfil' while offline, the broadcast message and the user's location are stored in the local WatermelonDB database. This ensures that users can compose broadcasts even without a network connection.
- *Offline Viewing:* Users can view previously received 'Mehfil' broadcasts and locations of other users within their vicinity from the local WatermelonDB database. This allows users to stay informed about local updates even when offline.
- *Synchronization:* When the device regains connectivity, the offline 'Mehfil' broadcasts are synchronized with the server. The server processes these broadcasts and distributes them to relevant online users based on location.
- *Location Updates:* User location updates are stored locally and synchronized with the server upon reconnection. The server updates the location data for online users and distributes the updated locations to maintain accurate proximity-based information.
- *Conflict Resolution:* Conflicts in 'Mehfil' primarily involve location data. A timestamp-based conflict resolution strategy is used, where the most recent location update is considered the valid one. This ensures that location information is as current as possible [14].
- *Map (Online User Display):* This section provides a visual representation of online users' locations on a map. It helps users discover other people in their vicinity.
- *Offline Map Viewing:* The 'Map' section can display a cached version of recently online users'

locations from the local WatermelonDB database. This allows users to view a general overview of the area even when offline. Real-time updates are not available offline.

- *Online User Updates:* When the device is online, the 'Map' section receives real-time updates of online user locations from the Node.js backend via Socket.IO, and this data is also stored in Watermelon DB. This ensures that the map reflects the current online user distribution.
- *Synchronization Frequency:* To minimize data usage and battery consumption, location updates for the 'Map' section are synchronized at a lower frequency than chat messages. This balances the need for up-to-date information with efficiency.

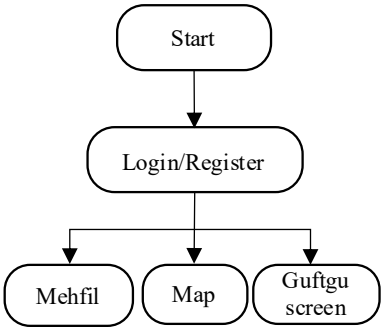

Figure 1. Distinct functionalities and workflows of software.

CHALLENGES AND FUTURE IMPROVEMENTS

Challenges

The development of 'Guftgu' presented several challenges, particularly in implementing the offline-first functionality and the unique features of the application:

- *Offline Data Consistency:* Maintaining data consistency between the local WatermelonDB database and the server is a significant challenge, especially with location-based data in 'Mehfil' and real-time user locations on the 'Map.' Ensuring that data is synchronized correctly and that conflicts are resolved appropriately is crucial for reliable user experience.
- *Local Storage Management:* Efficiently managing the size of the local database, especially with cached map data and location history, is crucial. Mobile devices have limited storage, and strategies for purging or archiving data are needed to prevent the application from consuming excessive storage space.
- *Synchronization Complexity:* Implementing robust and efficient synchronization mechanisms for 'Guftgu,' 'Mehfil,' and 'Map' data, considering different data update frequencies and priorities, is complex. Different data types (chat messages, location broadcasts, user locations) require tailored synchronization strategies [15].
- *Real-time Updates and Offline Availability:* Balancing the need for real-time updates in 'Mehfil' and 'Map' with the goal of providing useful offline functionality is a key challenge. Some features, like the 'Map,' inherently rely on real-time data, and providing a meaningful offline experience requires careful design.
- *User Privacy:* Ensuring user privacy, especially with location data in 'Mehfil' and 'Map,' both online and offline, requires careful consideration of data storage, transmission, and access control. Protecting user location information is paramount.
- *Scalability of Location-Based Broadcasts:* Efficiently handling many concurrent location-based broadcasts in 'Mehfil' presents a scalability challenge for both the frontend and backend. Distributing broadcasts to users in relevant geographical areas requires efficient algorithms and data structures.

Future Improvements

To address these challenges and further enhance the application, the following future improvements are proposed:

- *Optimized Synchronization:* Investigating more efficient synchronization strategies, such as differential synchronization and delta updates, to minimize data transfer for 'Guftgu,' 'Mehfil,' and 'Map' data. This can reduce data usage and improve synchronization speed.
- *Improved Conflict Resolution:* Implementing more sophisticated conflict resolution algorithms, such as CRDTs, to handle concurrent updates in 'Guftgu' and 'Mehfil' more gracefully, especially for message edits and location updates. This can lead to more intuitive conflict resolution [16].
- *Enhanced Map Functionality:* Exploring ways to provide more useful offline functionality for the 'Map' section, such as pre-caching map tiles or allowing users to view previously visited locations offline. This can improve the utility of the 'Map' when connectivity is limited.
- *Privacy Enhancements:* Implementing enhanced privacy features for location data in 'Mehfil' and 'Map,' such as differential privacy or location obfuscation techniques. This can provide users with more control over their location information [17].
- *Scalability Improvements:* Optimizing the backend architecture to handle a larger volume of location-based broadcasts and concurrent online users, including techniques like distributed messaging queues and database sharding. This can ensure that the application remains responsive as the user base grows.
- *Proximity-Based Features:* Exploring additional proximity-based features for 'Mehfil,' such as offline discovery of nearby users using Bluetooth or Wi-Fi Direct. This can enable interaction even without an internet connection.

CONCLUSION

Real-time communication and location-based social networking have become integral to modern interaction. This paper has explored the technological stack, implementation strategies, and challenges in building 'Guftgu,' an offline-first location-based social networking application. The ability to function seamlessly in offline or unreliable network conditions is crucial for such applications, especially in scenarios where users may experience intermittent connectivity. Future research should focus on further optimizing data synchronization, enhancing user privacy in location-based services, and exploring new proximity-based interaction models to create more robust and user-friendly social networking platforms.

REFERENCES

1. Fielding RT. Architectural styles and the design of network-based software architectures. University of California, Irvine; 2000.
2. Podgorelec B, Czerny R, Zefferer T, Prünster B, Corici AA, Wich T, Hühnlein D. et al. Design and Architecture of Mobile Cross-Border Services Building Blocks. InFrom Electronic to Mobile Government. Cham: Springer Nature Switzerland; 2024 Jul 13. pp. 45–62.
3. Talukder AK. The Next Generation Web: Technologies and Services. InInternational Conference on Big Data Analytics. Cham: Springer International Publishing; 2020 Dec 15. pp. 209–229.
4. Tadi SR. Interactive Document Editing and Distributed Synchronization Using Azure Cosmos and WebSockets. Environments. 2024 Aug;4(1).
5. Srivastava T, Pandey A, Khan R. A Study of Node. js Using Injection Vulnerabilities. Int J Adv Res Comput Sci Softw Eng. 2018;8(5):64.
6. Burhan M, Alam H, Arsalan A, Rehman RA, Anwar M, Faheem M, Ashraf MW et al. A comprehensive survey on the cooperation of fog computing paradigm-based IoT applications: layered architecture, real-time security issues, and solutions. IEEE Access. 2023 Jul 12;11:73303–29.
7. Bishop M, Dark M, Futcher L, Van Niekerk J, Ngambeki I, Bose S, Zhu M. et al. Learning principles and the secure programming clinic. In IFIP World Conference on Information Security Education. Cham: Springer International Publishing; 2019 Jun 19. pp. 16–29.
8. Heidari S, Simmhan Y, Calheiros RN, Buyya R. Scalable graph processing frameworks: A taxonomy and open challenges. ACM Computing Surveys (CSUR). 2018 Jun 12;51(3):1–53.
9. Jang-Jaccard J, Nepal S, Celler B, Yan B. WebRTC-based video conferencing service for

telehealth. Computing. 2016 Jan;98(1):169–93.
10. Lin CC, Huang AY, Yang SJ. A review of ai-driven conversational chatbots implementation methodologies and challenges (1999–2022). Sustainability. 2023 Feb 22;15(5):4012.
11. Nosina KC, Swarna Latha T. Introduction to Cybersecurity with AI, ML, and Blockchain. InNext-Generation Cybersecurity: AI, ML, and Blockchain. Singapore: Springer Nature Singapore; 2024 May 19. pp. 1–21.
12. Laricheva M, Liu Y, Shi E, Wu A. Scoping review on natural language processing applications in counselling and psychotherapy. Brit J Psychol. 2024 Aug 2.
13. Abraham A, Dutta P, Mandal JK, Bhattacharya A, Dutta S. Emerging technologies in data mining and information security. Proceedings of IEMIS-2018. 2018.
14. Cherukuri BR. Containerization in cloud computing: Comparing Docker and Kubernetes for scalable web applications. Int J Sci Res Appl. 2024;13(01):3302–15.
15. Freeman MJ. FPGA-Based Embedded Systems for Smart Grid Integration. Am J Embedded Sys VLSI Desig. 2024 Apr 30;5(2):5–8.
16. Schulzrinne H. Voice communication across the Internet: A network voice terminal. Amherst, MA, USA: University of Massachusetts at Amherst, Department of Computer and Information Science; 1992 Jul 29.
17. Keshav S, Kesahv S. An engineering approach to computer networking: ATM networks, the Internet, and the telephone network. Reading: Addison-Wesley; 1997 May 5.

