

Shadownet: Offline Mesh Communication and Emergency Coordination Platform

Asmi Pekhale¹, Riya Deshmukh², Divya Vedpathak³, Mayuri Waghire⁴,
Mrs. V. M. Khanapure⁵, Dr. D.N. Rewadkar⁶

Students, Department of Information Technology¹⁻⁴

Lecturer, Department of Information Technology⁵

Head of Department, Department of Information Technology⁶
Government Polytechnic, Pune, India

Abstract: Communication becomes critical during emergencies, natural disasters, large gatherings, and situations where Internet connectivity is unavailable or unreliable. Conventional communication applications depend on centralized servers and network connectivity, limiting their availability during network failures. ShadowNet is an offline mesh communication and emergency coordination platform that combines Internet-based communication with local mesh communication to support essential operations during connectivity loss. It provides messaging, emergency coordination, secure authentication, role-based access, and AI Network Intelligence. The system uses React.js for the frontend, Node.js and Express.js for backend services, and Supabase with PostgreSQL for database and authentication services. Supported offline information is stored temporarily and synchronized when connectivity is restored. ShadowNet aims to provide resilient communication by combining centralized and decentralized approaches.

Keywords: ShadowNet, Offline Communication, Mesh Network, Peer-to-Peer Communication, Emergency Coordination, WebRTC, Offline Synchronization, AI Network Intelligence

I. INTRODUCTION

Communication systems have become an essential part of modern society for exchanging information, coordinating activities, and responding to emergency situations. Most modern communication applications depend on Internet connectivity, centralized servers, cloud services, and continuously available network infrastructure. Although these systems provide fast and convenient communication under normal conditions, their dependence on infrastructure becomes a major limitation when connectivity is interrupted.

Network failures can occur because of natural disasters, power failures, infrastructure damage, network congestion, remote geographical conditions, or unexpected technical problems. During such situations, users may be unable to access conventional messaging and coordination platforms. Emergency situations also require rapid distribution of alerts and coordination between multiple teams and network members.

The Capstone project identifies several limitations in existing communication approaches, including heavy dependence on centralized servers, limited functionality during Internet outages, network congestion during emergencies, difficulty in distributing emergency information, lack of unified team communication, limited offline synchronization, and insufficient local peer-to-peer communication.

ShadowNet is proposed to address these limitations through a dual-mode communication architecture. When Internet connectivity is available, the system operates using conventional centralized web services. When connectivity becomes unavailable, selected communication and coordination functions can operate using local communication mechanisms. Once connectivity is restored, pending information can be synchronized with the central system.

The system combines communication and emergency coordination features into a single platform. These include general messaging, team communication, emergency signals, broadcast alerts, network management, notifications,



activity monitoring, secure authentication, role-based controls, file sharing, offline synchronization, and AI Network Intelligence.

The main objective is therefore to provide a more resilient communication platform that can maintain essential communication capabilities even when conventional network infrastructure becomes unavailable.

II. LITERATURE REVIEW

A. Delay-Tolerant and Disruption-Tolerant Networking

Delay-Tolerant Networking (DTN) is designed for environments where continuous end-to-end connectivity cannot be guaranteed. Unlike conventional Internet communication, DTN-based systems can store information temporarily and forward it when communication opportunities become available. This concept is highly relevant to ShadowNet because communication may need to continue even when a continuous Internet connection is unavailable.

DTN approaches demonstrate that temporary storage, forwarding, and synchronization can improve communication reliability in intermittently connected environments.

B. Mobile Ad Hoc and Mesh Networks

Mobile Ad Hoc Networks allow participating devices to communicate without depending completely on fixed infrastructure. Devices can potentially act as communication nodes and forward information to other devices.

Mesh networking extends this concept by allowing multiple devices to participate in communication. If a direct connection between two devices is unavailable, intermediate devices can potentially forward information. This principle is relevant to ShadowNet's proposed peer and relay-based communication model.

C. Emergency Communication Systems

Emergency communication systems focus on providing rapid information exchange during disasters and network failures. Such systems generally require:

- Rapid alert distribution
- Emergency signals
- Team coordination
- Reliable local communication
- Network monitoring
- Alternative communication paths

ShadowNet applies these concepts to a unified communication platform by combining emergency signals, broadcasts, team communication, messaging, and network management.

D. Offline-First and Synchronization Systems

Offline-first systems allow selected operations to continue locally when connectivity is unavailable. Data generated during the offline period can be stored temporarily and synchronized with a central database after connectivity is restored.

ShadowNet uses this concept for its offline communication and synchronization functionality. The Capstone project specifically describes local storage/caching, pending offline actions, connectivity restoration detection, and synchronization with the backend.

E. Peer-to-Peer Communication

Peer-to-peer communication allows participating devices to exchange information directly rather than always depending on a central server. WebRTC-related technologies provide mechanisms for browser-based peer communication and are relevant to the offline communication layer of ShadowNet.



The project uses peer discovery and WebRTC-based peer-to-peer communication concepts in its communication workflow.

F. Research Gap

Existing communication, emergency-alert, mesh-networking, cloud, and messaging systems generally address individual requirements. However, there is a need for an integrated platform combining:

- Online communication
- Offline communication
- Emergency signals
- Team coordination
- Broadcast alerts
- Local storage
- Synchronization
- Network management
- Secure authentication
- Role-based access
- Network intelligence

ShadowNet attempts to address this gap by integrating centralized and decentralized communication concepts into a single platform.

III. PROPOSED METHODOLOGY

The proposed methodology of ShadowNet is based on a hybrid communication architecture that combines online centralized services with local peer-to-peer communication. The system is designed to maintain selected communication and emergency coordination functions when conventional Internet connectivity becomes unreliable or unavailable. The methodology is implemented using a modular architecture consisting of the frontend, backend, database, communication layer, offline data handling, synchronization, and AI Network Intelligence components. The system design and implementation follow the architecture, data-flow, use-case, and activity models defined for ShadowNet.

A. Proposed System

ShadowNet is proposed as a dual-mode communication and emergency coordination platform. In normal connectivity conditions, users access communication services through the web application, backend services, and centralized database. When conventional connectivity is unavailable, supported communication functions can operate through the local communication layer and temporarily maintain information for later synchronization.

The proposed system supports the following implemented functions:

- User registration and login
- Two-factor authentication where enabled.
- Network creation and joining.
- Network and member management.
- General messaging.
- File sharing.
- Team creation and team communication.
- Emergency signals.
- Broadcast alerts.
- Notifications.
- Offline communication.



- Offline action storage and synchronization.
- AI Network Intelligence.
- AI-assisted automatic routing.
- User profile management.

These functions are divided into independent modules so that the communication, emergency, administrative, and synchronization components can be developed and integrated separately. The Capstone implementation specifically demonstrates the network gateway, admin dashboard, broadcast module, network management, member management, team creation, member dashboard, emergency signal, messaging, AI Network Intelligence, automatic routing, and profile settings.

B. System Architecture

The ShadowNet architecture separates the system into the user interface, backend services, database services, communication mechanisms, and synchronization logic.

The frontend is developed using React.js and provides the user dashboard, messaging, team, emergency, notification, network, and profile interfaces. The backend uses Node.js and Express.js for API handling, application logic, communication processing, network operations, administrative operations, and synchronization requests.

Supabase and PostgreSQL are used for authentication and persistent storage. The database maintains information related to users, networks, teams, messages, broadcasts, emergency signals, notifications, and other application records. The communication layer provides the local peer-to-peer communication functionality. The offline layer handles local information and pending actions, while the synchronization mechanism transfers supported pending information to the backend after connectivity is restored.

The architecture also incorporates modular design principles, allowing individual components to perform specific functions while interacting through defined interfaces. The frontend communicates with backend services through REST APIs, which process requests, validate inputs, and manage communication with the database. Supabase provides PostgreSQL-based persistent storage and authentication services, while Supabase Storage can be used to manage supported file-sharing operations. This separation improves maintainability and allows individual modules to be enhanced without requiring major changes to the entire application.

The communication mechanism is designed to support both centralized and local communication workflows. In online mode, user requests are handled by the backend and relevant information is stored in the central database. During network interruptions, the local communication layer enables supported peer-to-peer operations between connected devices. Local caching helps retain selected information temporarily, allowing users to access supported data without continuously contacting the backend. The availability of offline operations depends on the communication method and the capabilities of the connected devices.

The synchronization component manages the transfer of pending offline information when Internet connectivity becomes available again. It coordinates queued operations with backend services and updates the application according to the synchronization results. Error handling and validation help manage unsuccessful transfers and reduce the possibility of inconsistent records. This mechanism provides continuity between online and offline modes while maintaining the central database as the persistent source of application data.

Furthermore, the architecture supports role-based functionality for administrators and regular users. Access restrictions are applied to operations according to user permissions, while backend validation provides an additional layer of control. The modular structure also allows future integration of enhanced network monitoring, intelligent routing mechanisms, additional communication channels, and improved security features. Overall, the proposed architecture combines web-based services, local peer-to-peer communication, temporary offline storage, and synchronization to improve the reliability and flexibility of ShadowNet.



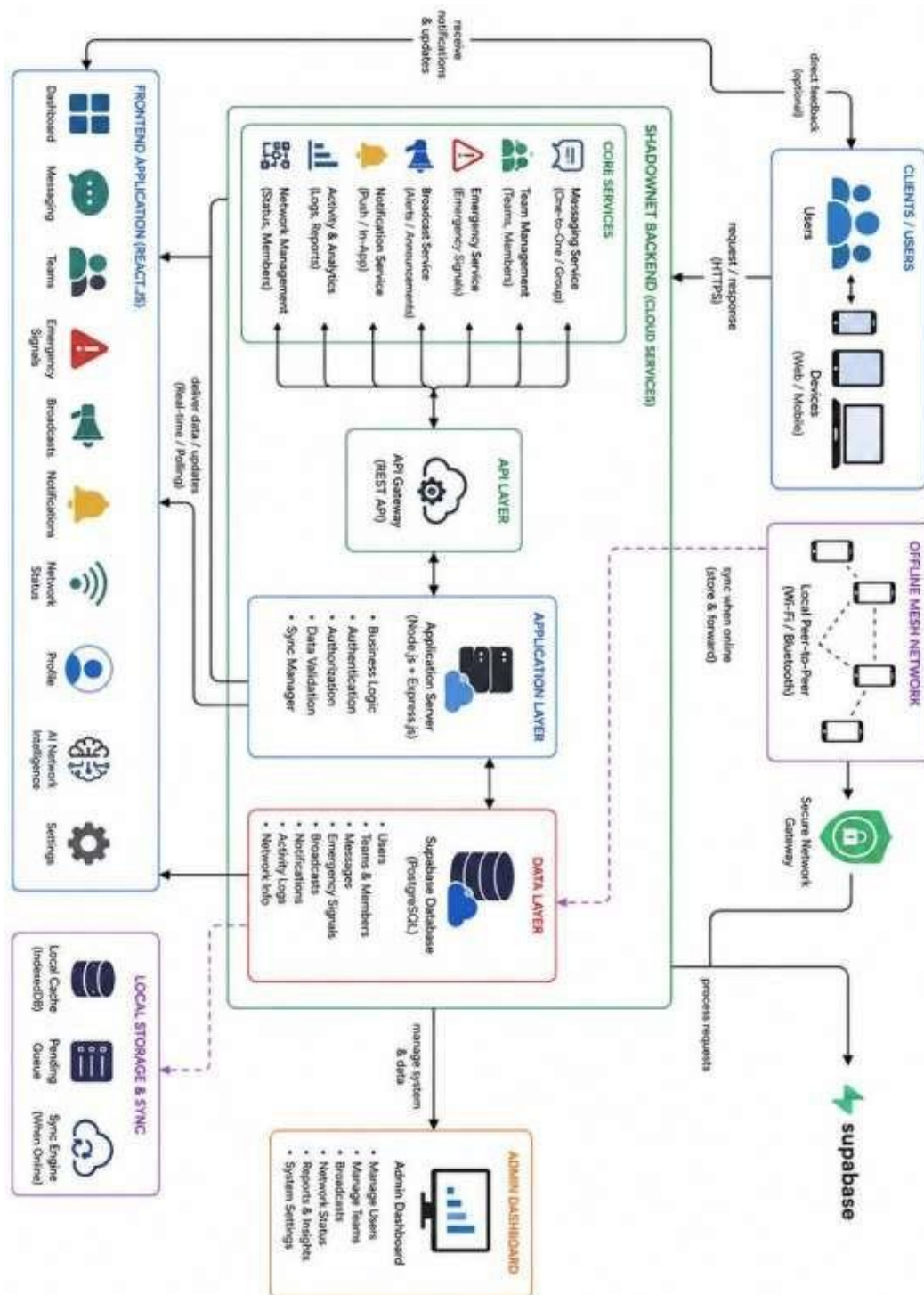


Fig. 1. ShadowNet System Architecture



C. Overall System Flow

The overall system flow follows the activities performed by a user from authentication to network participation and communication. Initially, the user registers or logs into ShadowNet and accesses the dashboard. The user can then create or join a network and use the available communication features according to assigned permissions. These features include messaging, emergency signals, broadcast alerts, notifications, and network monitoring. This process establishes the foundation for secure communication and effective coordination among participating users. After the user joins a network, the system determines the available connectivity conditions and selects the appropriate communication approach. ShadowNet follows a dual-mode architecture that combines centralized online services with supported local mesh communication mechanisms. When Internet connectivity is available, the React.js frontend communicates with backend services through API requests. Node.js and Express.js process these requests, while Supabase and PostgreSQL support authentication and persistent data management.

During online operation, users can exchange messages, send emergency signals, and issue broadcast alerts through the available application modules. Relevant requests are processed by the backend, and information is stored in the central database according to the application's workflow. Notifications and activity records help users remain informed about communication events and network changes. Role-based access controls also restrict administrative operations to authorized users, supporting organized communication management.

When Internet connectivity becomes unavailable, ShadowNet uses its supported local mesh communication layer to maintain selected communication and coordination operations. Connected devices can exchange supported information through available local communication mechanisms without depending entirely on centralized Internet services. Local storage temporarily maintains relevant offline information and pending operations. The availability of these functions depends on device connectivity and the capabilities supported by the implemented offline communication mechanism.

When connectivity is restored, the synchronization module identifies supported pending operations and transfers relevant information to the backend. The backend processes incoming requests and updates the corresponding database records according to application rules. Validation and error-handling mechanisms help manage unsuccessful transfers and reduce inconsistencies between locally maintained information and central records. The application can then update relevant interface elements and display synchronization status, helping users understand the progress of pending operations.

Throughout the workflow, network monitoring, notifications, activity tracking, and access controls support communication management under changing network conditions. Emergency signals and broadcast alerts help distribute critical information, while status updates provide visibility into connectivity and synchronization activities. Overall, ShadowNet integrates authentication, team coordination, online communication, offline operations, and data synchronization into a unified system. This approach aims to improve communication continuity during network disruptions and provides a foundation for emergency coordination in college campuses, technical events, remote areas, and large public gatherings.



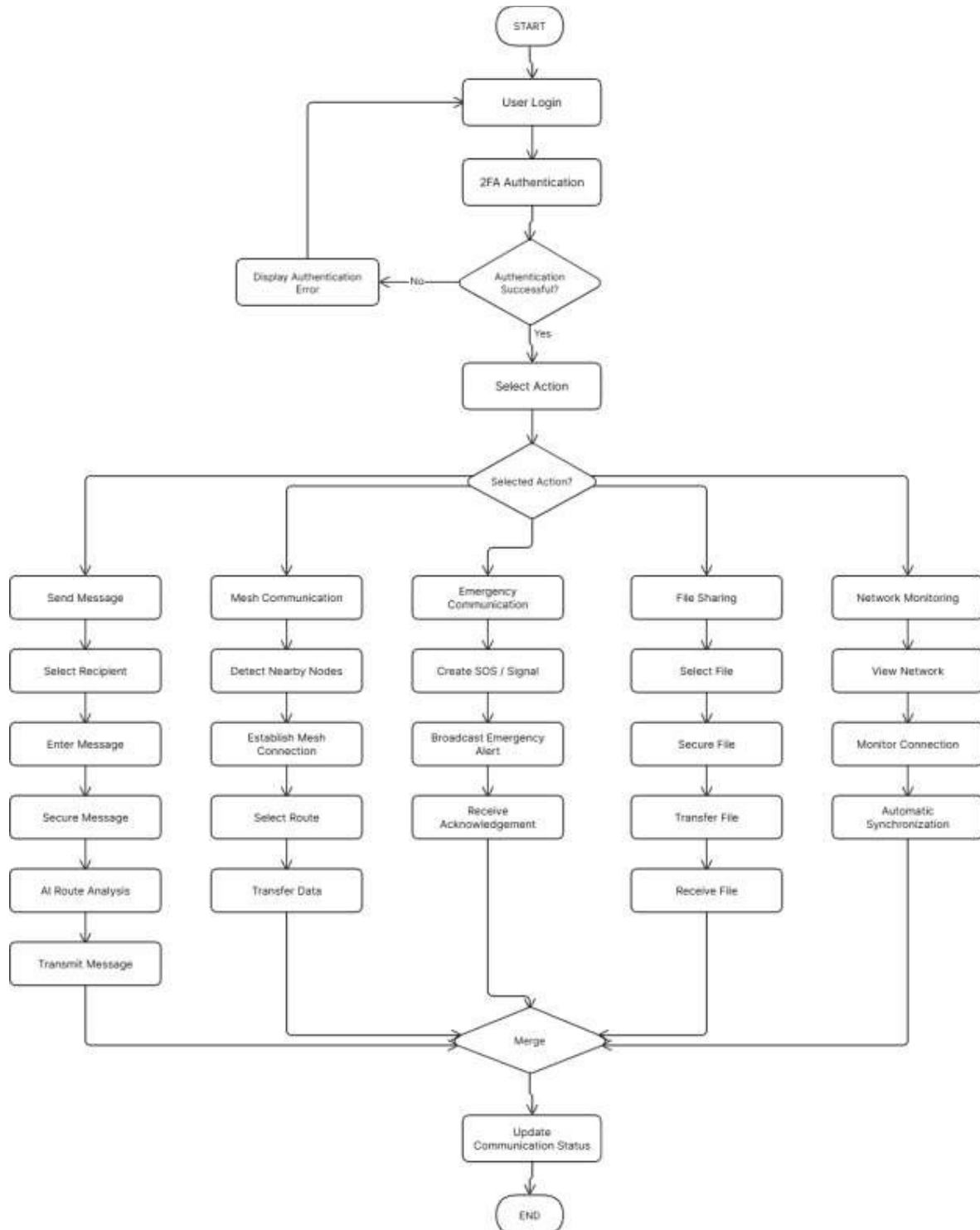


Fig. 2. Overall ShadowNet System Flow



D. Communication and Routing Method

The communication methodology is based on peer-to-peer communication concepts. When users are connected to the ShadowNet communication network, available peers can be identified and communication can be established between participating devices.

The implemented communication workflow supports messaging, file sharing, and emergency alerts. According to the Capstone workflow, a message can be represented using packet information such as source, destination, next hop, TTL, and hop count. When the destination is not directly reachable, an intermediate peer can act as a relay and forward the packet.

The communication process can therefore be represented as:

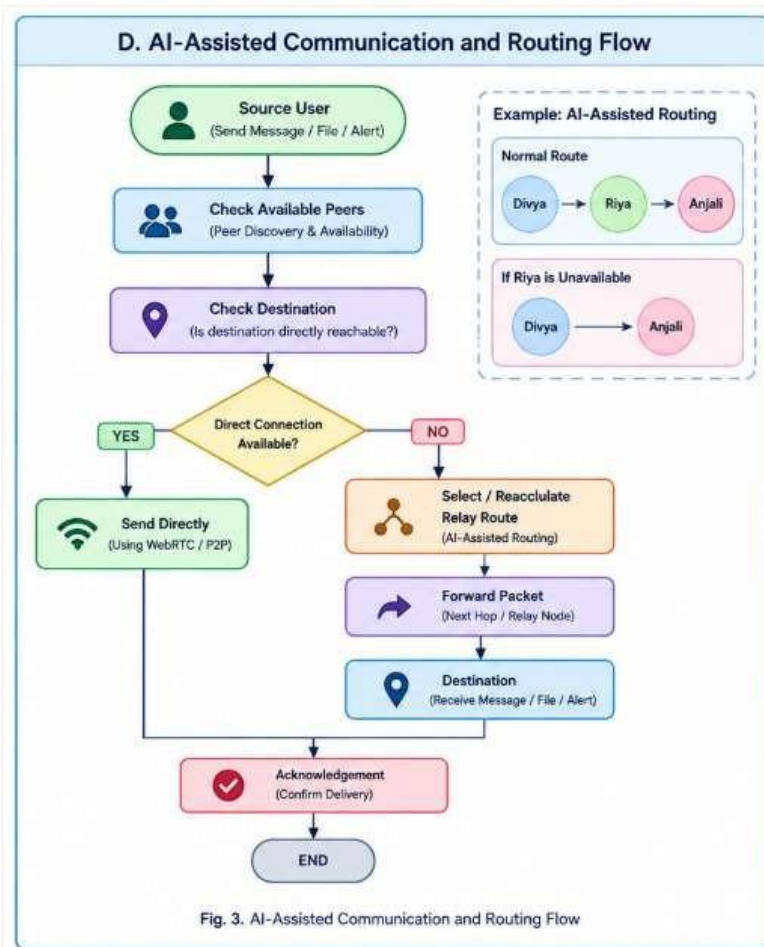


Fig. 3. ShadowNet Communication and Routing Flow

The Project demonstrates the routing concept using the following example: Normal Route:

Divya → Riya → Anjali When Riya is unavailable:

Divya → Anjali

In this case, the system can recalculate the communication path when an intermediate node becomes unavailable. The Capstone identifies this as AI-based automatic routing, where AI Network Intelligence assists the routing decision.

The methodology therefore focuses on route selection and route recalculation based on available peer/network information, rather than claiming a specific machine-learning algorithm that is not documented in the Capstone.



E. Offline Communication and Synchronization

Offline communication is an important part of the ShadowNet methodology. The system supports selected communication and coordination operations when normal connectivity is unavailable.

The Capstone implementation specifically describes offline messaging, file sharing, emergency signals, and broadcast-related functionality. Offline actions can be temporarily stored and processed after connectivity is restored.

The synchronization process follows:

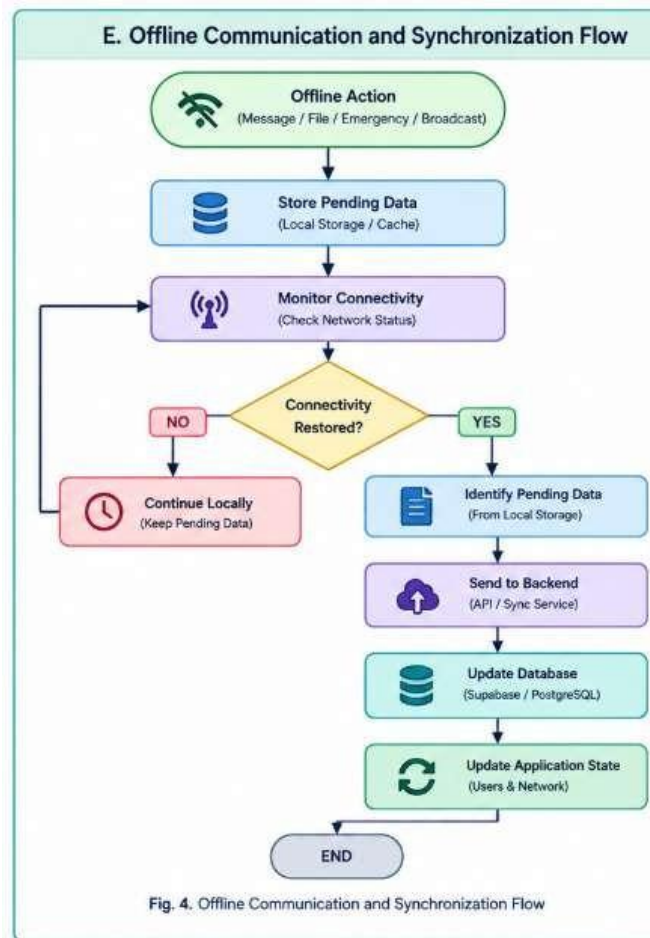


Fig. 4. Offline Communication and Synchronization Flow

The synchronization module therefore connects the local/offline operation with the centralized application. Its main tasks are to identify pending information, send supported data to the backend, update the persistent database, and update the application state.

F. Major Functional Modules

The proposed methodology is implemented through the following major modules:

1. Authentication Module:

Handles user registration, login, authentication, and two-factor authentication where enabled.

2. Network Module:

Provides network creation, joining, network information, and network management.



3. Messaging Module:

Provides general messaging and file sharing. The Capstone identifies the Messages page as the main communication module and demonstrates offline messaging and file sharing in mesh mode.

4. Team Communication Module:

Supports team creation, member management, and team-based communication.

5. Emergency Module:

Provides emergency signals and priority communication functionality. The member dashboard includes the Send Signal functionality for emergency communication.

6. Broadcast Module:

Allows authorized users or administrators to send broadcast alerts to network members. The Capstone demonstrates the broadcast module in the admin dashboard.

7. Notification Module:

Provides notifications related to important communication and network activities.

8. Offline Communication Module:

Handles selected communication functions under limited or unavailable connectivity and supports local data handling and pending-action storage.

9. Synchronization Module:

Processes pending offline information after connectivity is restored and updates the backend and database.

10. AI Network Intelligence Module:

Provides network-related information and supports network-related analysis and routing decisions. The Capstone includes a dedicated AI Network Intelligence interface and AI-based automatic routing demonstration.

11. Profile Module:

Allows users to manage their profile information.

12. Administration Module:

Provides administrative functions such as broadcast alerts, network management, member management, and team creation.

G. Development and Testing Methodology

ShadowNet follows an Agile, modular, component-based, and API-based development approach. The development process follows:

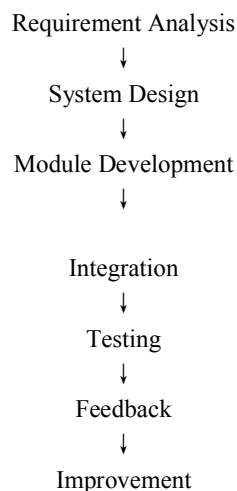


Fig. 5. ShadowNet Development Methodology



During requirement analysis, requirements related to authentication, communication, teams, emergency signals, broadcasts, network management, offline operation, and synchronization were identified.

During system design, the architecture, DFD, use-case model, activity flow, database structure, and module interactions were defined. The Capstone specifically includes the Architecture Diagram, Level 0 DFD, Use-Case Diagram, Gantt Chart, and Activity Diagram as part of the system design.

During implementation, React.js was used for the frontend, Node.js and Express.js for backend services, and Supabase/PostgreSQL for authentication and persistent data storage. The system was divided into independent modules to simplify development and maintenance.

During integration, the frontend, backend, database, authentication, communication, emergency, notification, offline, and synchronization modules were connected to form the complete application.

Testing was then performed to verify the implemented functionality. The Capstone includes functional, integration, performance, usability, security, and regression testing. The documented test cases cover registration, login, two-factor authentication, messaging, team communication, offline communication, synchronization, emergency signals, broadcasts, notifications, network status, role-based access, profile management, AI Network Intelligence, and responsive interface behavior.

Therefore, the proposed methodology directly connects the design diagrams → system modules → implementation → communication workflow → offline synchronization → testing, providing a practical methodology for the implemented ShadowNet prototype.

IV. RESULT AND DISCUSSION

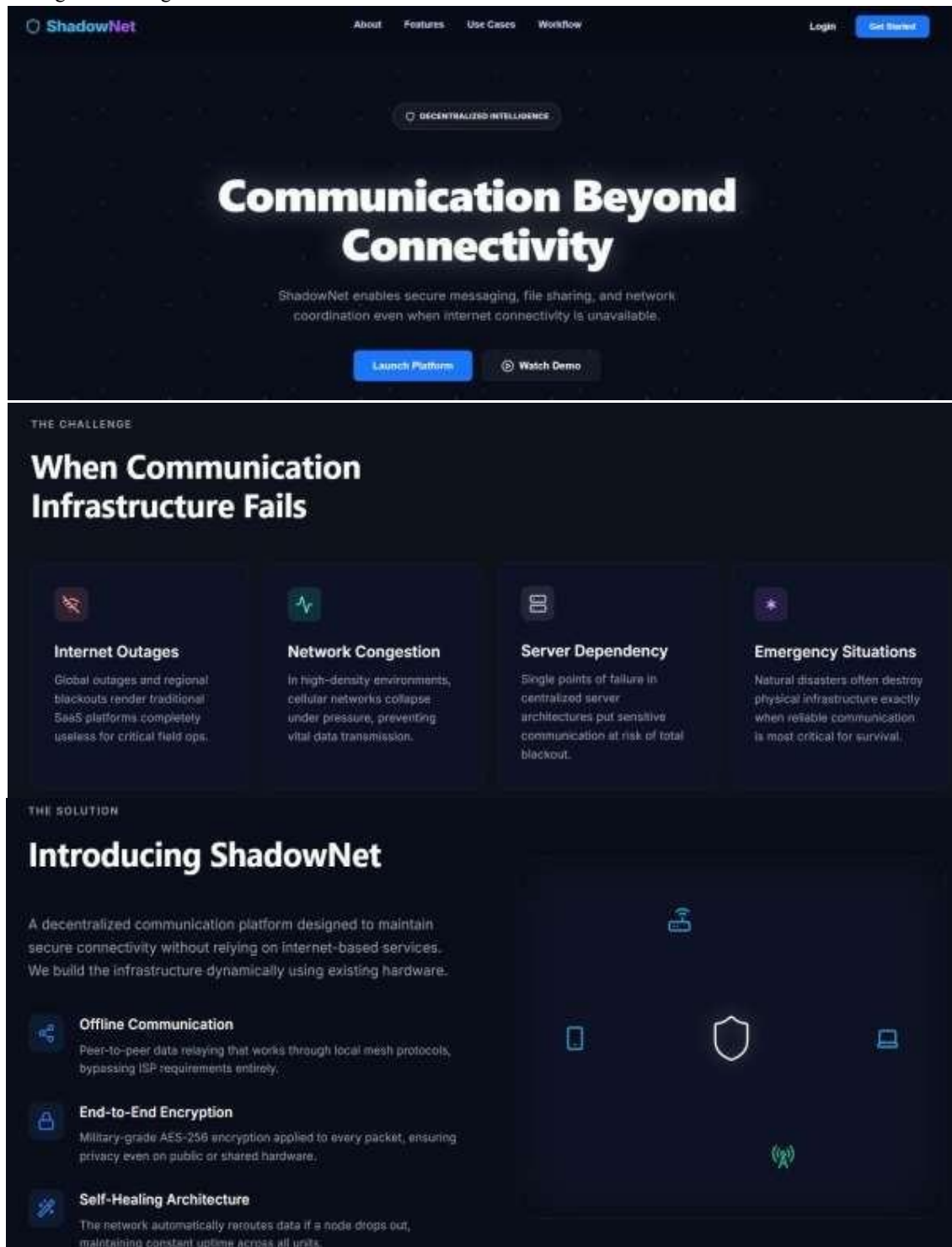
ShadowNet Project Workflow

ShadowNet is an AI-Assisted Secure Mesh Communication and Emergency Coordination Platform designed to maintain communication when conventional network infrastructure becomes unreliable or unavailable. The workflow begins when users securely log in and create or join a ShadowNet network, after which each device becomes a peer in the network. The system performs peer discovery and heartbeat monitoring to identify available devices and establish connections between them using WebRTC peer-to-peer communication. Once connected, users can exchange chat messages, files, and emergency alerts through the mesh. Each message is converted into a packet envelope containing information such as the source, destination, next hop, TTL, and hop count. The routing system uses the current network topology, neighbor information, and AI-assisted network intelligence to determine an appropriate communication path. If the destination is not directly connected, intermediate devices act as relay nodes and forward the packet toward the destination while maintaining packet tracking and preventing duplicate delivery.

When the destination receives the packet, it delivers the message to the user and sends an acknowledgement back through the network. At the same time, ShadowNet continuously monitors peer availability, network topology, connection status, and routing conditions, allowing the system to adapt when nodes join, leave, or become unavailable. Thus, the overall workflow combines secure communication, WebRTC-based P2P networking, multi-hop packet forwarding, dynamic routing, AI-assisted network optimization, and emergency coordination into a single mesh-based communication system.



1. Landing / Home Page:



The screenshot shows the ShadowNet website landing page. The header includes the ShadowNet logo, navigation links (About, Features, Use Cases, Workflow), and a 'Get Started' button. The main heading is 'Communication Beyond Connectivity', with a subtext: 'ShadowNet enables secure messaging, file sharing, and network coordination even when internet connectivity is unavailable.' Below this are 'Launch Platform' and 'Watch Demo' buttons. The 'THE CHALLENGE' section is titled 'When Communication Infrastructure Fails' and lists four challenges: Internet Outages, Network Congestion, Server Dependency, and Emergency Situations. The 'THE SOLUTION' section is titled 'Introducing ShadowNet' and describes it as a decentralized communication platform. It lists three features: Offline Communication, End-to-End Encryption, and Self-Healing Architecture.

ShadowNet About Features Use Cases Workflow Login Get Started

DECENTRALIZED INTELLIGENCE

Communication Beyond Connectivity

ShadowNet enables secure messaging, file sharing, and network coordination even when internet connectivity is unavailable.

Launch Platform Watch Demo

THE CHALLENGE

When Communication Infrastructure Fails



Internet Outages

Global outages and regional blackouts render traditional SaaS platforms completely useless for critical field ops.



Network Congestion

In high-density environments, cellular networks collapse under pressure, preventing vital data transmission.



Server Dependency

Single points of failure in centralized server architectures put sensitive communication at risk of total blackout.



Emergency Situations

Natural disasters often destroy physical infrastructure exactly when reliable communication is most critical for survival.

THE SOLUTION

Introducing ShadowNet

A decentralized communication platform designed to maintain secure connectivity without relying on internet-based services. We build the infrastructure dynamically using existing hardware.



Offline Communication

Peer-to-peer data relaying that works through local mesh protocols, bypassing ISP requirements entirely.



End-to-End Encryption

Military-grade AES-256 encryption applied to every packet, ensuring privacy even on public or shared hardware.



Self-Healing Architecture

The network automatically reroutes data if a node drops out, maintaining constant uptime across all units.



Core System Capabilities



Secure Messaging

Real-time instant messaging with delivery receipts, all operating on local protocols.



File Transfer

Chunk-based binary file distribution that survives network interruptions and node swaps.



Broadcasts

Push emergency announcements to all connected devices in the network instantly.



Topology Monitoring

Live visual map of all connected nodes, signal strengths, and traffic flow paths across the mesh network.



AI Route Optimization

Intelligent packet routing that predicts network congestion and finds the fastest physical path.

Mission-Critical Deployment

ShadowNet is built for the most demanding environments where failure is not an option.



Disaster Response

Coordinating rescue efforts when local towers are down.



Military Field Ops

Tactical communication in radio-silent or jammed zones.



Festivals & Events

Maintaining staff coordination in ultra-high density crowds.





Build Communication Networks That Never Stop

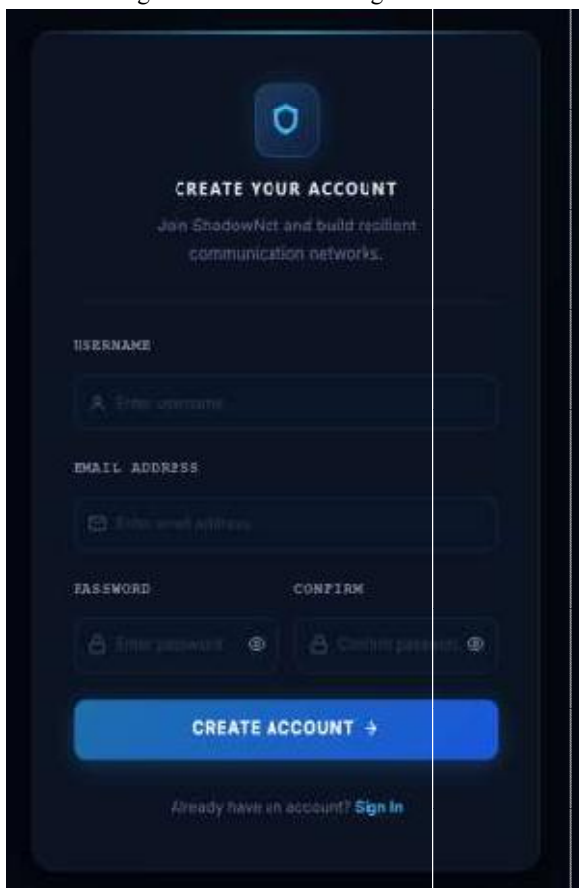
Join the decentralized revolution. Deploy your own secure communication infrastructure in minutes, no internet required.

[Launch Platform](#)
[Watch Demo](#)

© 2026 SHADOWNET SYSTEMS. SECURE. DISCONNECTED. ABSOLUTE.



2. On Clicking launch Platform – Registration screen:



CREATE YOUR ACCOUNT
Join ShadowNet and build resilient communication networks.

USERNAME
[Enter username]

EMAIL ADDRESS
[Enter email address]

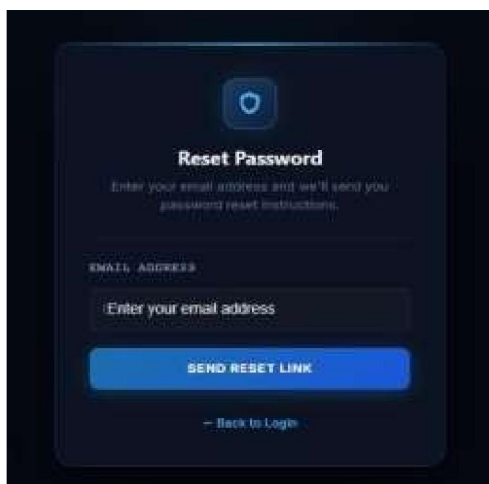
PASSWORD [Enter password] CONFIRM [Confirm password]

CREATE ACCOUNT →

Already have an account? [Sign In](#)

3. Already registered- Login page:

4. Forgot Password feature

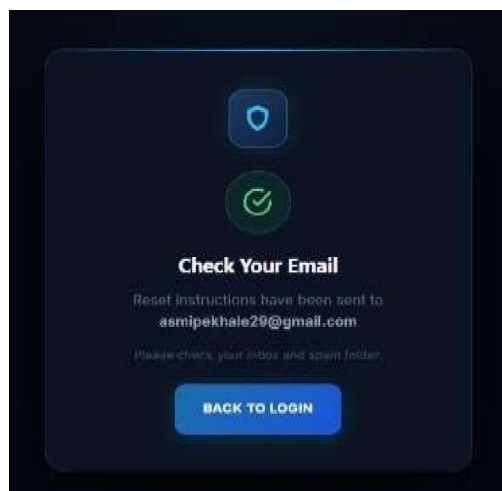


Reset Password
Enter your email address and we'll send you password reset instructions.

EMAIL ADDRESS
[Enter your email address]

SEND RESET LINK

[← Back to Login](#)



Check Your Email
Reset instructions have been sent to asmipekhale20@gmail.com.
Please check your inbox and spam folder.

BACK TO LOGIN



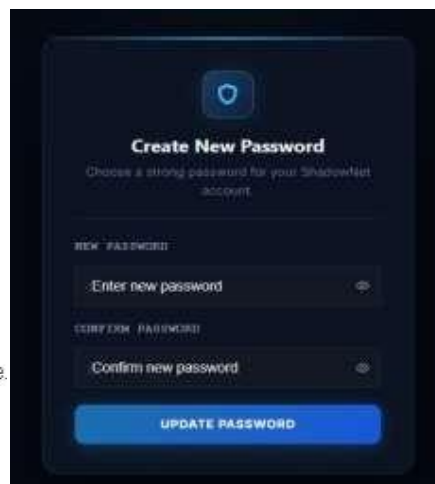


Reset your password

We received a request to reset your password. Follow the link below to choose a new one.

[Reset password](#)

If you didn't request this, you can safely ignore this email.



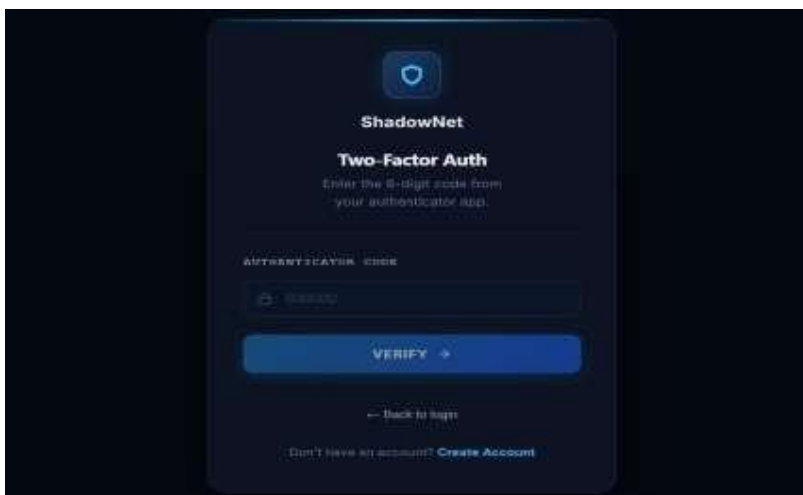
Create New Password
Choose a strong password for your ShadowNet account.

NEW PASSWORD
Enter new password

CONFIRM PASSWORD
Confirm new password

UPDATE PASSWORD

5. Two Factor Authentication – If enabled:



ShadowNet
Two-Factor Auth
Enter the 6-digit code from your authenticator app.

AUTHENTICATOR CODE
XXXXXX

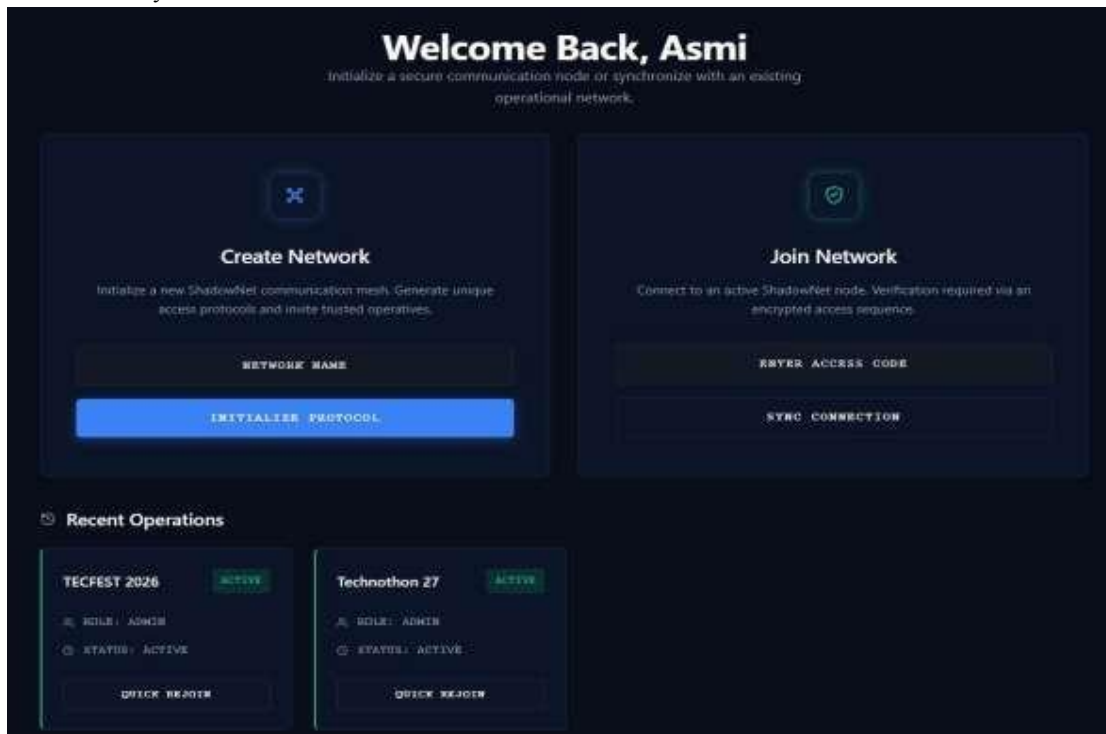
VERIFY

[Back to login](#)

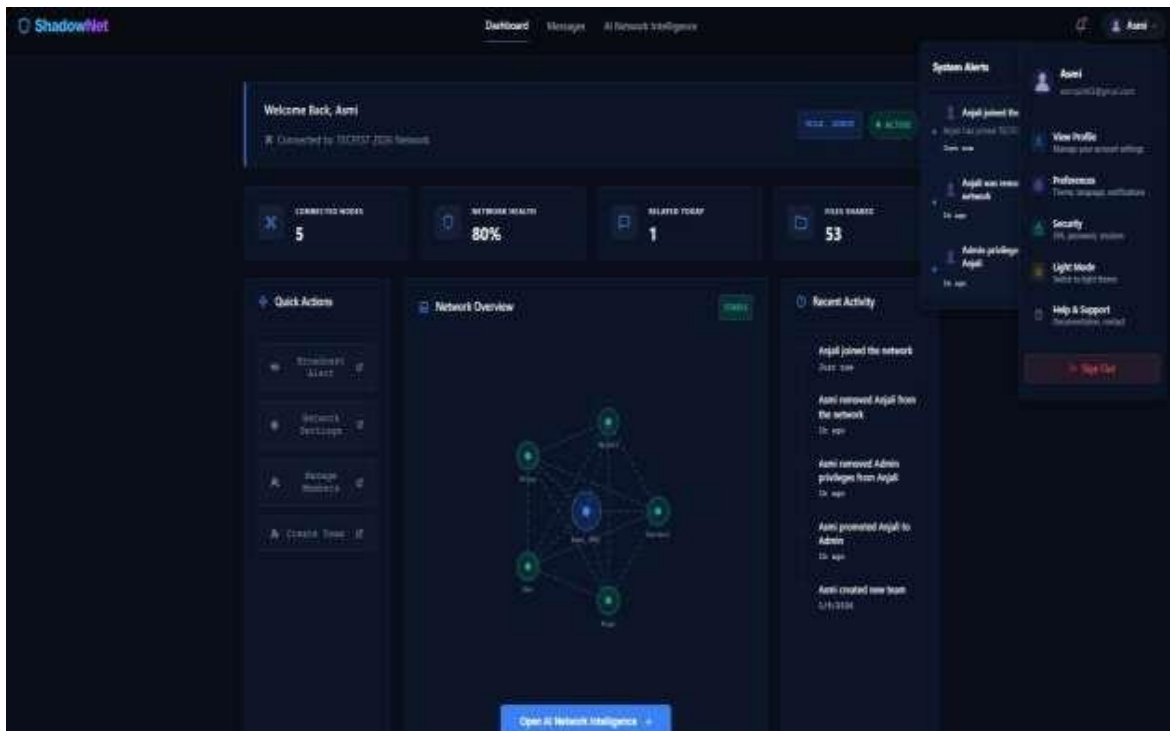
Don't have an account? [Create Account](#)



6. Network Gateway / Create & Join Network:

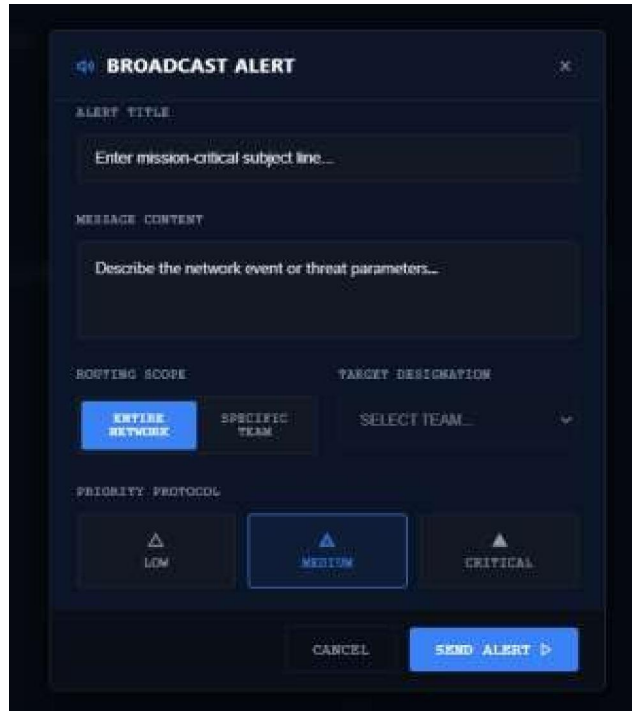


7. Admin Dashboard:



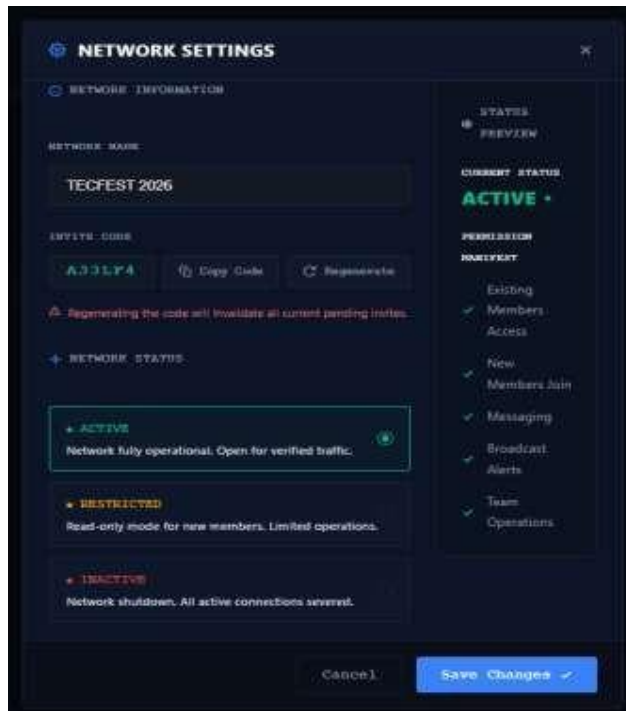
Admin Dashboard features:

a) Broadcast alert module – works in offline mode:



The screenshot shows the 'BROADCAST ALERT' interface. It includes a title bar with a close button. Below the title, there are two text input fields: 'ALERT TITLE' with a placeholder 'Enter mission-critical subject line...' and 'MESSAGE CONTENT' with a placeholder 'Describe the network event or threat parameters...'. Below these fields, there are two sections: 'ROUTING SCOPE' and 'TARGET DESIGNATION'. The 'ROUTING SCOPE' section has two buttons: 'ENTIRE NETWORK' (highlighted in blue) and 'SPECIFIC TEAM'. The 'TARGET DESIGNATION' section has a 'SELECT TEAM...' button with a dropdown arrow. Below these sections, there is a 'PRIORITY PROTOCOL' section with three buttons: 'LOW', 'MEDIUM' (highlighted in blue), and 'CRITICAL'. At the bottom, there are two buttons: 'CANCEL' and 'SEND ALERT' (highlighted in blue).

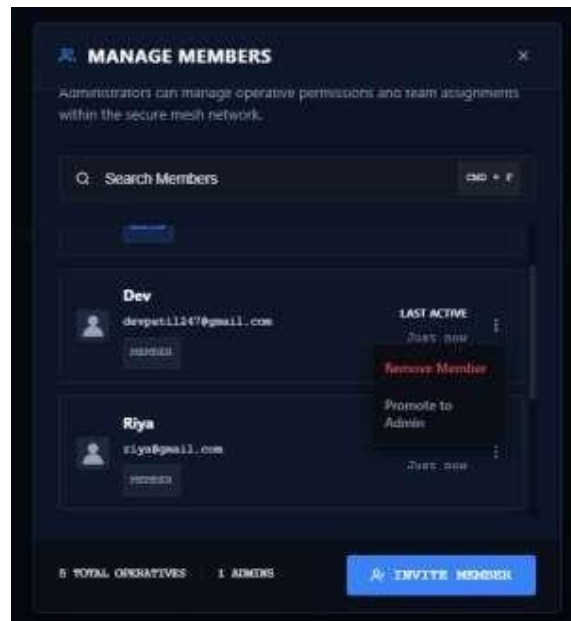
b) Network management module:



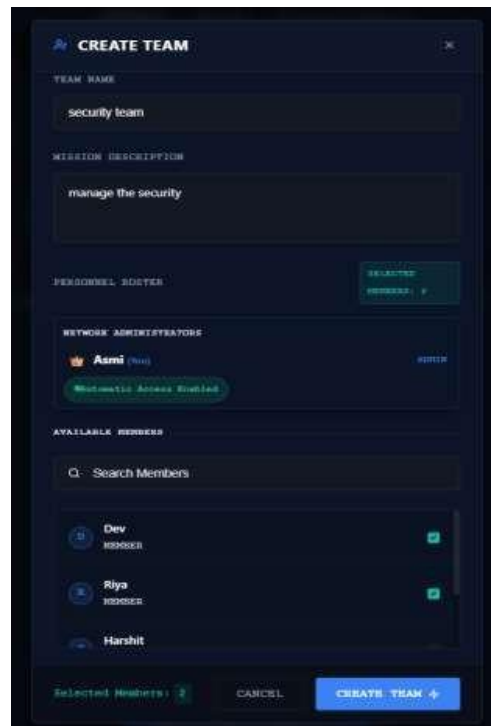
The screenshot shows the 'NETWORK SETTINGS' interface. It has a title bar with a close button. Below the title, there is a 'NETWORK INFORMATION' section with a 'NETWORK NAME' field containing 'TECFEST 2026' and an 'INVITE CODE' field containing 'A33LY4'. There are buttons for 'Copy Code' and 'Regenerate'. Below this, there is a warning message: 'Regenerating the code will invalidate all current pending invites.' Below the warning, there is a 'NETWORK STATUS' section with three status cards: 'ACTIVE' (highlighted in green), 'RESTRICTED' (highlighted in yellow), and 'INACTIVE' (highlighted in red). Each card has a description of the status. On the right side, there is a 'STATUS' section with a 'PREVIEW' button and a 'CURRENT STATUS' section showing 'ACTIVE'. Below this, there is a 'PERMISSION' section with a 'PERMIT' button and a list of permissions: 'Existing', 'New', 'Members Join', 'Messaging', 'Broadcast', 'Alerts', 'Team', and 'Operations'. At the bottom, there are two buttons: 'Cancel' and 'Save Changes' (highlighted in blue).



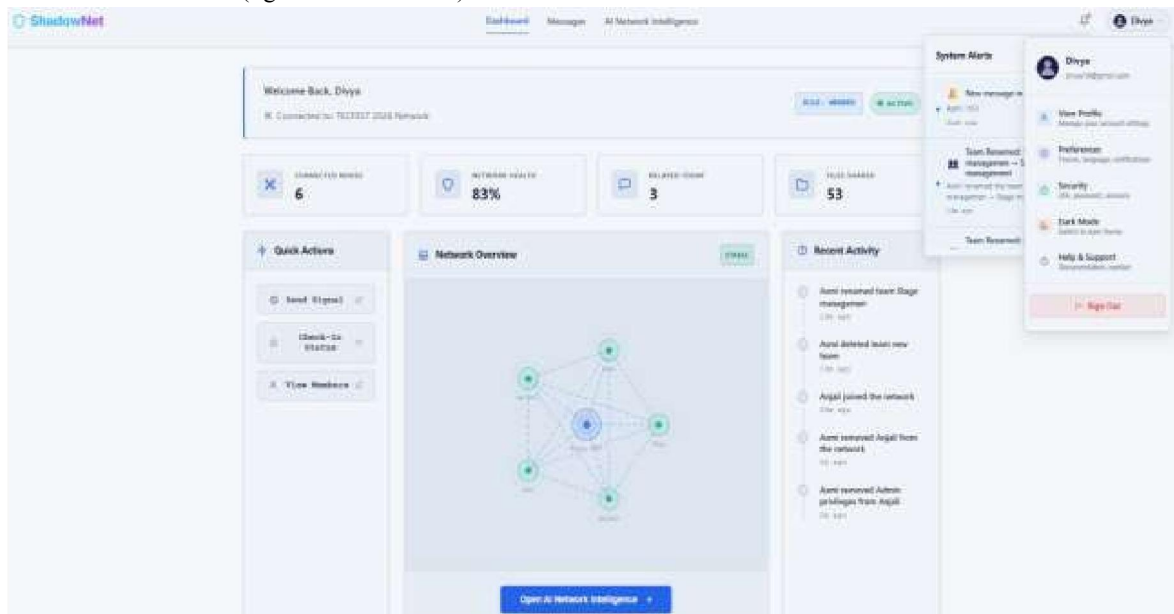
c) Manage members module:



d) Create Team Module:



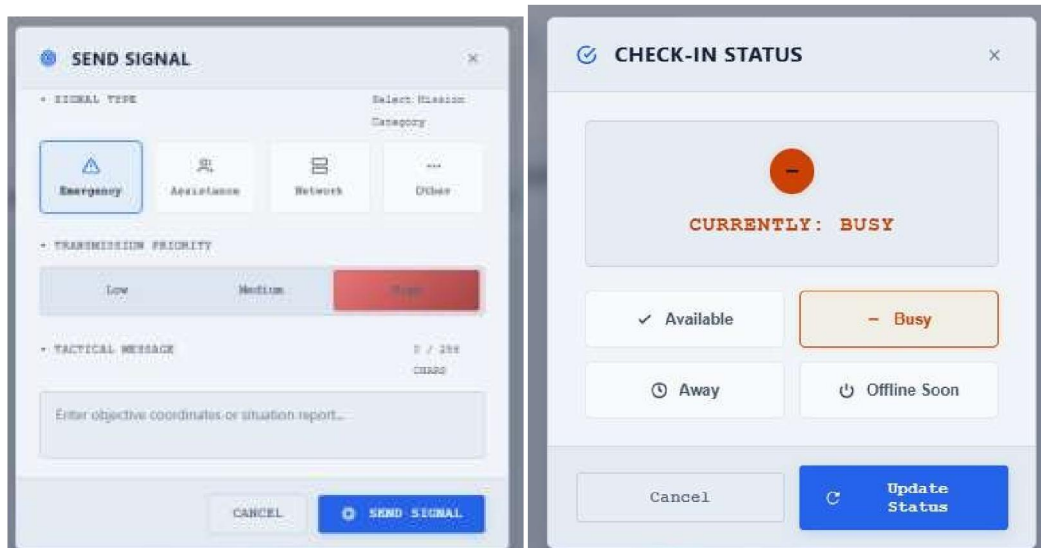
8. Member Dashboard – (light mode reference):



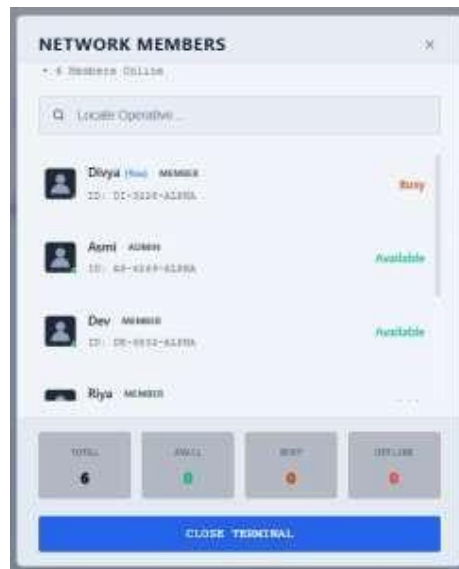
Member dashboard features:

a) Send Signal Module – works in b) Check-in Status

Module: offline mode:

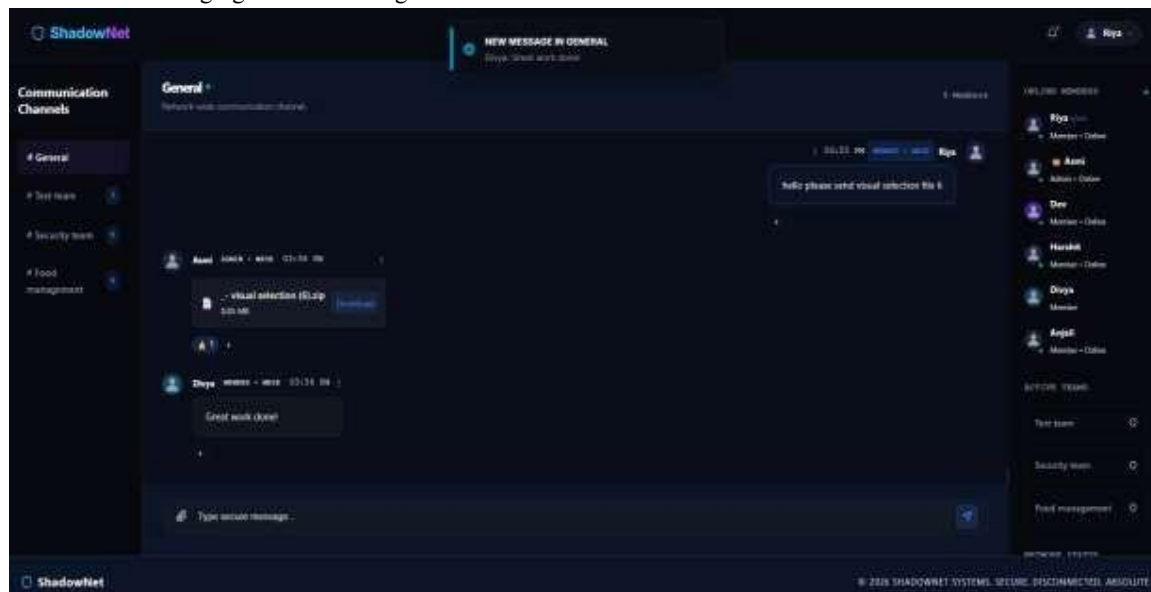


c) View Members Module:



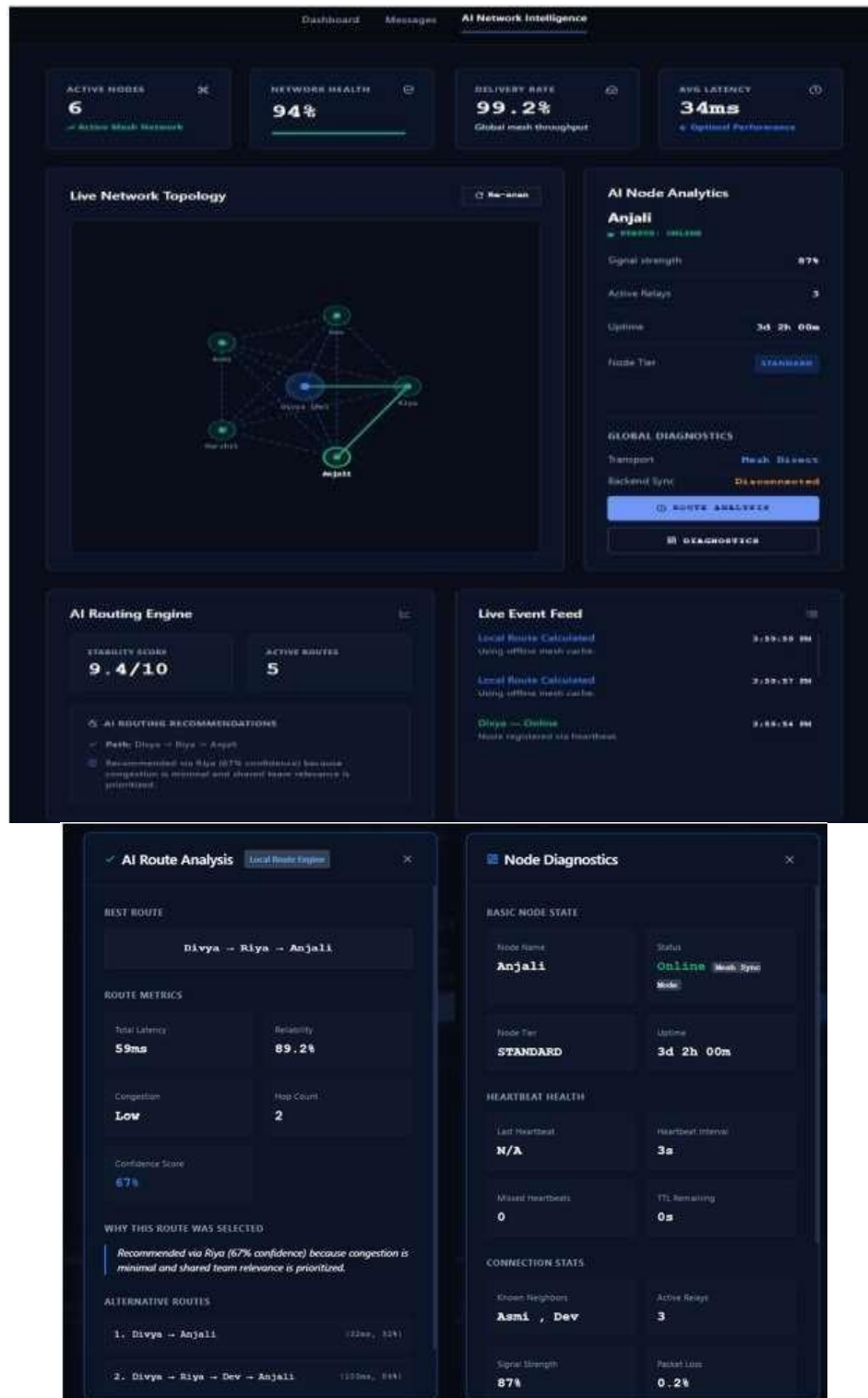
9. Messages Page – main communication module

Provides offline messaging and file sharing in mesh mode:



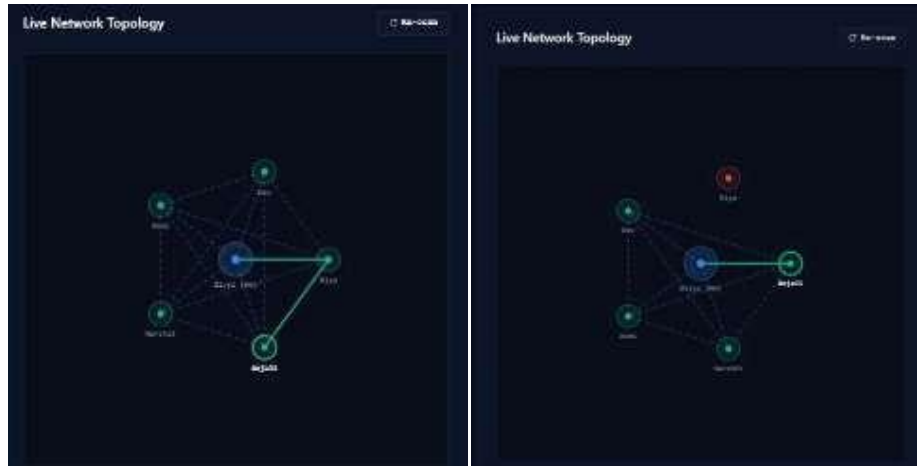
10 . AI Network Intelligence:



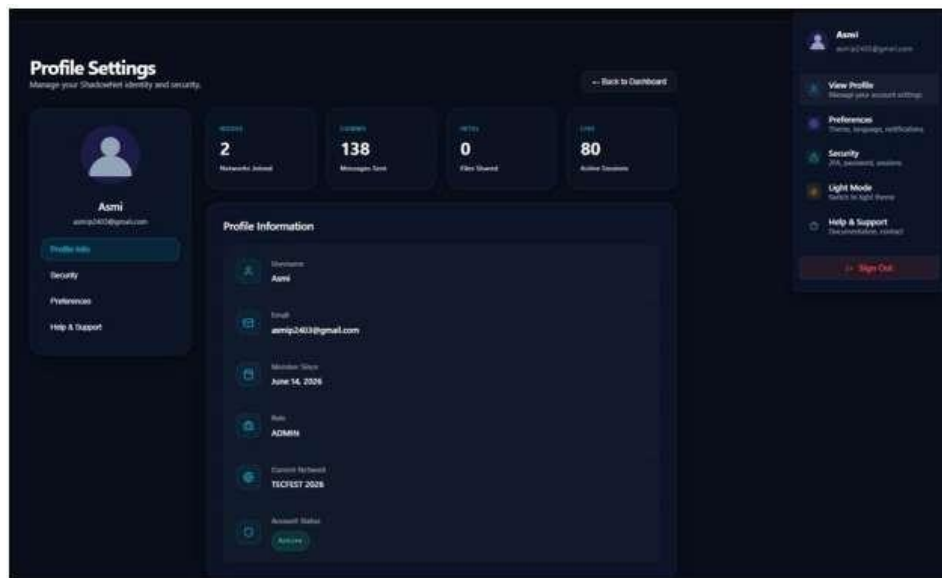


11. AI-Based Automatic Routing:

Divya → Riya → Anjali Riya unavailable → AI recalculates route → Divya → Anjali



10. Profile settings page:



V. CONCLUSION

ShadowNet addresses the limitations of communication systems that depend on continuous Internet connectivity by combining centralized online services with offline mesh communication. The platform uses React.js, Node.js, Express.js, and Supabase to support online communication and selected offline operations during network failures. Its key features include messaging, team coordination, emergency signals, broadcast alerts, network monitoring, notifications, activity tracking, synchronization, secure authentication, and AI Network Intelligence. Through its Dual-Mode Architecture, ShadowNet adapts to available connectivity and synchronizes supported offline information when the connection is restored. The platform can be used in emergencies, natural disasters, college campuses, technical events, festivals, remote areas, and large public gatherings. Future enhancements may include multi-hop mesh



networking, native mobile applications, AI-powered routing, IoT integration, and advanced security, further improving its reliability and emergency coordination capabilities.

VI. FUTURE SCOPE

In the future, ShadowNet can be enhanced to support larger mesh networks, automatic peer discovery, stronger authentication, and efficient multi-hop routing. AI-assisted routing can be improved by considering connection quality, latency, traffic load, node availability, and battery level. Additional features such as voice and video communication, location sharing, priority messaging, and advanced emergency alerts can also be introduced. The platform can be extended with dedicated mobile applications and integration of GPS, Bluetooth, and Wi-Fi Direct for improved local communication. Further enhancements may include automatic SOS alerts, disaster-specific communication channels, improved offline synchronization, and stronger encryption. Testing with larger groups of devices in disaster scenarios, remote areas, and campus emergencies can help evaluate its reliability and effectiveness. These improvements can help ShadowNet evolve into a more reliable, infrastructure-independent emergency communication platform.

REFERENCES

1. React. React Documentation. Available at: <https://react.dev/>. Used for developing the component-based user interface.
2. Node.js. Node.js Documentation. Available at: <https://nodejs.org/en/docs>. Used for backend development and server-side execution.
3. Express.js. Express Web Framework Documentation. Available at: <https://expressjs.com/>. Used for developing REST APIs and backend services.
4. Supabase. Supabase Documentation. Available at: <https://supabase.com/docs>. Used for PostgreSQL database management, authentication, real-time services, and cloud synchronization.
5. WebRTC. WebRTC: Real-Time Communication for the Web. Available at: <https://webrtc.org/>. Used as a reference for peer-to-peer real-time communication.
6. W3C. WebRTC 1.0: Real-Time Communication Between Browsers. Available at: <https://www.w3.org/TR/webrtc/>. Reference for browser-based peer-to-peer communication technologies.
7. Mozilla Developer Network. WebRTC API Documentation. Available at: https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API. Used for understanding browser-based real-time communication.
8. Google Developers. Nearby Connections API Documentation. Available at: <https://developers.google.com/nearby/connections/overview>. Reference for nearby peer-to-peer communication concepts.
9. IEEE Standards Association. IEEE 802.11 Wireless LAN Standards. Available at: <https://standards.ieee.org/ieee/802.11/7028/>. Reference for wireless communication technologies relevant to local mesh connectivity.
10. OpenAI. Artificial Intelligence Documentation and Resources. Available at: <https://openai.com/>. Used as a reference for AI-assisted network intelligence and analysis.
11. C. E. Perkins, E. M. Belding-Royer, and S. Das, Ad hoc On-Demand Distance Vector (AODV) Routing, RFC 3561, Internet Engineering Task Force, 2003. Available at: <https://www.rfc-editor.org/rfc/rfc3561>. Reference for dynamic routing in mobile ad hoc networks.
12. D. B. Johnson, Y.-C. Hu, and D. A. Maltz, The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4, RFC 4728, Internet Engineering Task Force, 2007. Available at: <https://www.rfc-editor.org/rfc/rfc4728>. Reference for source routing and multi-hop communication.
13. T. Clausen and P. Jacquet, Optimized Link State Routing Protocol (OLSR), RFC 3626, Internet Engineering Task Force, 2003. Available at: <https://www.rfc-editor.org/rfc/rfc3626>. Reference for proactive routing in mobile ad hoc networks.



14. K. Fall, "A Delay-Tolerant Network Architecture for Challenged Internets," in Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, ACM, 2003, pp. 27–34. Reference for communication in intermittently connected networks.
15. V. Cerf et al., Delay-Tolerant Networking Architecture, RFC 4838, Internet Engineering Task Force, 2007. Available at: <https://www.rfc-editor.org/rfc/rfc4838>. Reference for store-and-forward communication and disrupted networks.
16. A. Lindgren, A. Doria, and O. Schelén, "Probabilistic Routing in Intermittently Connected Networks," ACM SIGMOBILE Mobile Computing and Communications Review, vol. 7, no. 3, pp. 19–20, 2003. Reference for routing in intermittently connected networks.
17. T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and Wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks," in Proceedings of the 2005 ACM SIGCOMM Workshop on Delay-Tolerant Networking, 2005, pp. 252–259. Reference for efficient message forwarding in disconnected environments.
18. A. Vahdat and D. Becker, "Epidemic Routing for Partially Connected Ad Hoc Networks," Duke University Technical Report CS-200006, 2000. Reference for message dissemination in intermittently connected networks.
19. C. Mascolo et al., "Middleware for Mobile Ad Hoc Networks," in Proceedings of the 1st International Conference on Middleware, 2000. Reference for communication support in mobile and infrastructure-less networks.
20. M. Conti and S. Giordano, "Mobile Ad Hoc Networking: Milestones, Challenges, and New Research Directions," IEEE Communications Magazine, vol. 52, no. 1, pp. 85–96, 2014. Reference for mobile ad hoc and infrastructure-less networking.
21. I. F. Akyildiz, X. Wang, and W. Wang, "Wireless Mesh Networks: A Survey," Computer Networks, vol. 47, no. 4, pp. 445–487, 2005. Reference for wireless mesh network architecture, routing, and communication.
22. J. Jun and M. L. Sichitiu, "The Nomadic Community Wireless Mesh Network," in Proceedings of the IEEE Wireless Communications and Networking Conference, 2003. Reference for community-based wireless mesh communication.
23. M. Abolhasan, T. Wysocki, and E. Dutkiewicz, "A Review of Routing Protocols for Mobile Ad Hoc Networks," Ad Hoc Networks, vol. 2, no. 1, pp. 1–22, 2004. Reference for comparison of routing protocols in mobile ad hoc networks.
24. A. Boukerche, "Performance Evaluation of Routing Protocols for Ad Hoc Wireless Networks," Mobile Networks and Applications, vol. 8, pp. 497–512, 2003. Reference for evaluating routing approaches in ad hoc communication environments.
25. P. Bellavista, A. Corradi, C. Giannelli, and E. Foschini, "Mobile Computing and Wireless Communication in Emergency and Disaster Scenarios," reference material for resilient communication and emergency networking. Used to support the emergency communication context of ShadowNet

