NetMapper : System for Mapping and Optimization of Network

Authors: **Vijay Patil, Atharv Darp, Kunal Chikram, Sanket Dhamne**
Guide: **Dr. Ashwini Garkhedkar.**
Department of MCA, P.E.S. Modern College of Engineering, Pune

Abstract

This project presents NetMapper, a mobile- based system designed to analyze, map, and optimize mobile network performance across diverse locations. The app collects real-time data on network signal strength and internet speed, then visualizes it as a heatmap using Google Maps API. This helps users and telecom providers identify weak signal zones, monitor connectivity, and improve infrastructure planning. The system also features an Admin Dashboard that allows data review, manual ticket generation, and network performance tracking. NetMapper aims to bridge the digital divide between urban and rural areas by empowering providers to make data-driven decisions

about network improvement and tower placement.

Keywords: Network Strength, Signal Mapping, Heatmap Visualization, Real- Time Speed Tracking, Admin Dashboard, Rural Connectivity

1. Introduction

In the modern era, access to reliable mobile networks has transcended convenience to become a fundamental necessity for both urban and rural populations. This dependency has been sharply accelerated by the post-pandemic global shift towards remote work, e-learning and digital commerce.

The NetMapper project was developed to address this critical gap between user experience and provider-side analytics. It proposes a mobile-based, crowdsourced solution to gather, visualize, and manage real-time network performance data. The system empowers end-users to record signal strength (dBm) and internet speed at their specific locations. This data is then instantly visualized on a heatmap, providing an intuitive map of network quality.

Crucially, NetMapper bridges the gap from data collection to actionable intelligence by including an integrated Admin Panel. This back-end system allows telecom providers or network administrators to manage the collected data, proactively monitor weak zones, and generate and track investigation tickets. This closed-loop system aims to equip providers with the data-driven insights needed to optimize infrastructure, plan tower placements, and ultimately, enhance network reliability for all users.

Objectives

- To design and implement a mobile module for Android (using React Native and Java) to collect and transmit real-time network metrics, including signal strength (in dBm), data carrier, internet speed (upload/download), and precise GPS coordinates.

- To architect and deploy a scalable cloud backend using Firebase (Firestore) to securely store, manage, and serve all user-collected data points and administrative information in real-time.

- To develop a dynamic data visualization module by integrating the Google Maps API, capable of rendering the collected geospatial data as a color-coded heatmap (from green/excellent to red/poor) to provide an intuitive, at-a-glance understanding of network quality.

- To engineer a functional, dual- interface prototype that enforces a clear separation of roles:
- User Interface: For authenticated users to perform data collection.
- Admin Interface: A secure dashboard for administrators to review, filter, and analyze all submitted data.

- To build an integrated ticket management system within the Admin Dashboard, allowing administrators to proactively identify weak coverage zones from the heatmap data and create/track investigation tickets for network providers.
- To analyze the feasibility of integrating future AI-based predictive models to forecast network quality in un-mapped areas, establishing a baseline for future project development.

2. Literature Review.

Researchers have explored various methods for analyzing and visualizing mobile network performance. The existing literature provides a strong foundation for the core components of the NetMapper project.

A. Signal Visualization: Work by Y. Zhang and X. Li validates the use of heatmaps as a powerful tool for visualizing signal distribution. This method provides an intuitive, at-a-glance understanding of network quality, which we have adopted for our Google Maps API integration.

B. Rural Application: The challenges of network planning in underserved areas are highlighted by S. K. Singh and
R. Mehra. Their work on signal mapping for rural enhancement confirms the need for data collection tools like NetMapper to help bridge the "digital divide."

C. Data Collection Models: The concepts of "crowdsourcing" for telecom planning (T. Banerjee and P. Rao) and "cloud-based analytics" for real-time monitoring (M. Chen, L. Zhou, and K. Wang) are well-established. These papers justify our architectural choices of using a user-driven data collection model and a scalable Firebase backend.

3. Research Gap

Although several network mapping tools and academic studies exist and commercial apps (like OpenSignal) prove the crowdsourcing model, many lack real-time integration between user data collection and private administrative monitoring.

Existing apps often fail to provide an end-to-end solution that includes live signal visualization *and* a structured, internal issue-tracking system. The NetMapper system bridges this gap by offering a dual-interface design (User and Admin) with secure authentication. This allows an organization to manage its own private network data and, crucially, empowers administrators to take immediate, data-driven action by managing tickets, a feature not commonly found in public-facing applications.

4. Methodology

1. We used the Prototype Model for this project. This allowed us to quickly build and test the core features (data collection and heatmap visualization) and refine them based on early feedback, which was essential for validating our system's 60-80% accuracy.
2. The project's architecture is a streamlined, real-time data pipeline:

a) Data Collection (User): using native Java modules, allows users to collect signal strength (dBm), internet speed, and GPS location data.

b) Cloud Storage (Backend): This data is immediately sent to Firebase, which acts as our secure, real-time cloud database (Firestore) and handles user authentication.

c) Visualization (Presentation): The collected data is fetched from Firebase and displayed as a
color-coded heatmap using the Google Maps API in both the user app and the admin panel.

d) Admin Management: A separate Admin Dashboard provides secure access to review all user data, analyze weak signal clusters, and manually generate issue tickets for network providers.

5. Technologies Used

The selection of technologies for the NetMapper project was based on the need for rapid development, cross-platform compatibility, real-time data handling, and native hardware access.

1. JavaScript: This framework was the primary choice for the frontend. Its ability to create a single codebase that runs on both Android and iOS was crucial

for rapid prototyping. Its component-based architecture simplified the development of the user and admin interfaces.

2. Firebase (Cloud Database): Firebase was chosen as the Backend-as-a- Service (BaaS). This was a critical decision for several reasons:

a) Real-time Database: Firestore allowed the heatmap to update instantly as new data was collected.

b) Scalability: It automatically handles scaling without any server management.

c) Authentication: It provided a built-in, secure system for user and admin login.

d) Cost-Effectiveness: Its generous free tier was ideal for a student project.

3. Java: While JavaScript handled the UI, Java was essential for the core data collection. It was used to write a native Android module to access the low-level TelephonyManager API. This hardware-level access is required to read the signal strength (in dBm), which is not possible with JavaScript alone.

4. Google Maps API: This was used for the primary visualization feature. Its robust heatmap layer allowed us to plot the collected geospatial data and apply a color gradient (green to red) based on signal strength.

5. Network APIs: This refers to the specific Android (Java) APIs, like TelephonyManager and ConnectivityManager, used to gather signal strength, carrier name, and internet speed data.

6. Visual Studio (VS) Code: This was used as the primary Integrated Development Environment (IDE) for writing and debugging the React Native, Java, and XML code.

7. Expo Go: This tool was vital for rapid testing. It allowed us to run and live- reload the React Native application on a physical Android device by scanning a QR code, which was essential for testing the real-world signal collection.

6. Implementation Details

The prototype was developed using JavaScript in VS Code and tested on physical devices using Expo Go. This was essential for testing real-world signal collection, which is not possible in an emulator.

The app's implementation is divided into two main parts:

1. User Interface: A simple interface for data collection. The key challenge was accessing hardware-level signal strength. We solved this by building a Native Java Module to bridge Android's TelephonyManager (for dBm readings) with our JavaScript code.

2. Admin Dashboard: A secure interface for data management.

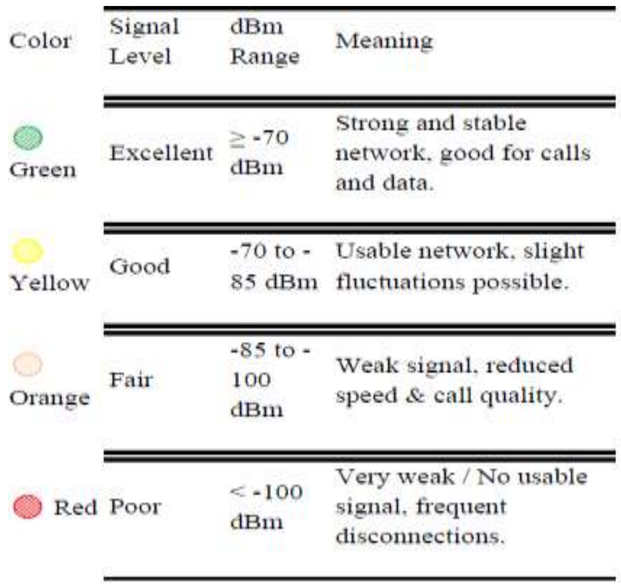
The core workflow is as follows:

- The user's app collects data (dBm, GPS, speed) and sends it to Firebase (Firestore).

- The data is then fetched from Firebase and rendered on the Google Maps API as a heatmap for both users and admins (Figure 1, Figure 2).

- The Admin Dashboard also has views to see the raw data records (Figure 3) and manage the ticket system by creating/updating tickets in a separate Firebase collection (Figure 4).

7. Network Signal Strength Classification

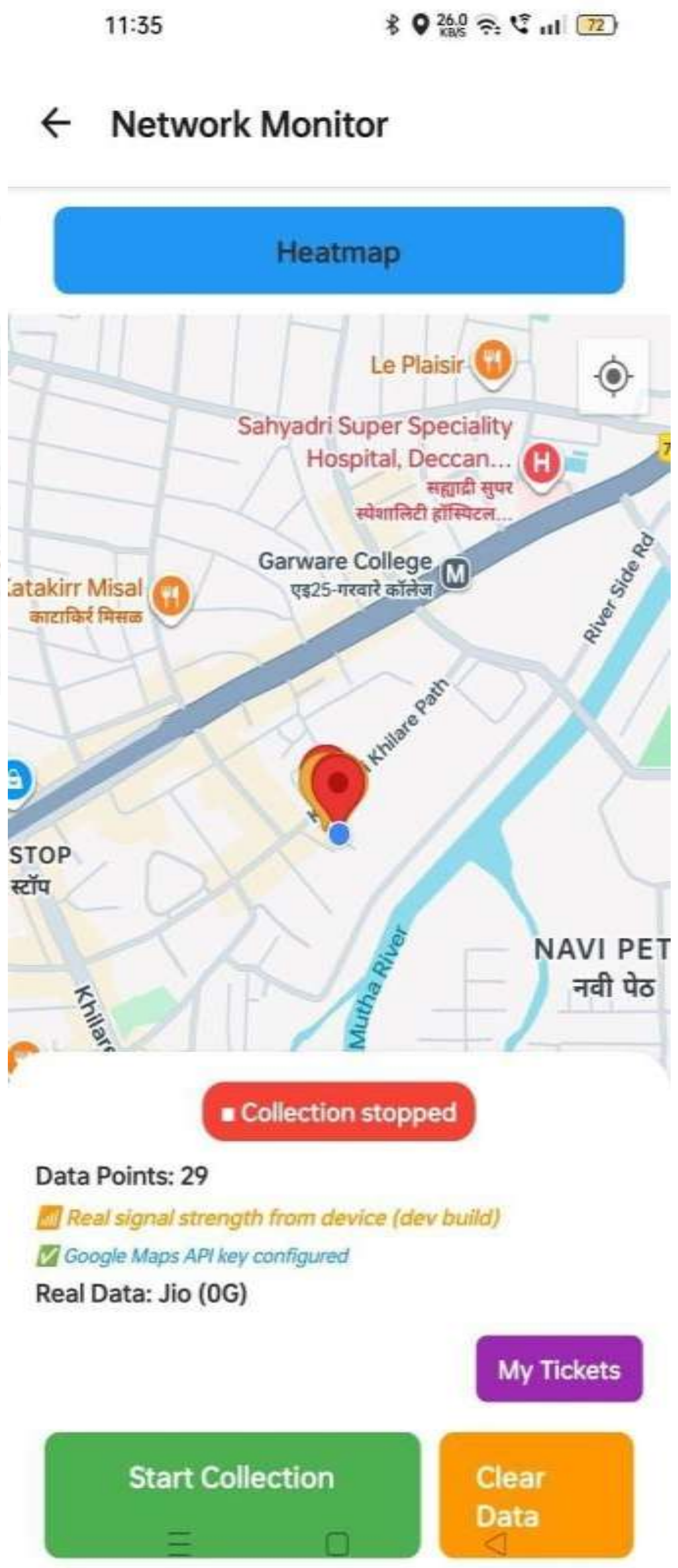


Color	Signal Level	dBm Range	Meaning
🟢 Green	Excellent	≥ -70 dBm	Strong and stable network, good for calls and data.
🟡 Yellow	Good	-70 to -85 dBm	Usable network, slight fluctuations possible.
🟠 Orange	Fair	-85 to -100 dBm	Weak signal, reduced speed & call quality.
🔴 Red	Poor	< -100 dBm	Very weak / No usable signal, frequent disconnections.

8. Heatmap Visualization Features

- The Heatmap screen plots signal points on Google Maps.
- Each collected location point is colored based on signal strength.
- The heatmap allows identifying:
  - Network dead zones
  - Areas with variable coverage
  - Locations with consistent strong network
- The heatmap updates in real-time whenever new signal readings are recorded.
- Admin and User both see the heatmap, but only Admin can manage data & tickets.

Fig.1.0



Data Points: 29
Real signal strength from device (dev build)
✓ Google Maps API key configured
Real Data: Jio (0G)

9. User Login Interface

Description:

- This screen is designed to authenticate the user before allowing access to the app features.

- The interface provides two login modes:

1. User Mode: For general users who will collect real-time network signal data.
2. Admin Mode: For authorized administrators

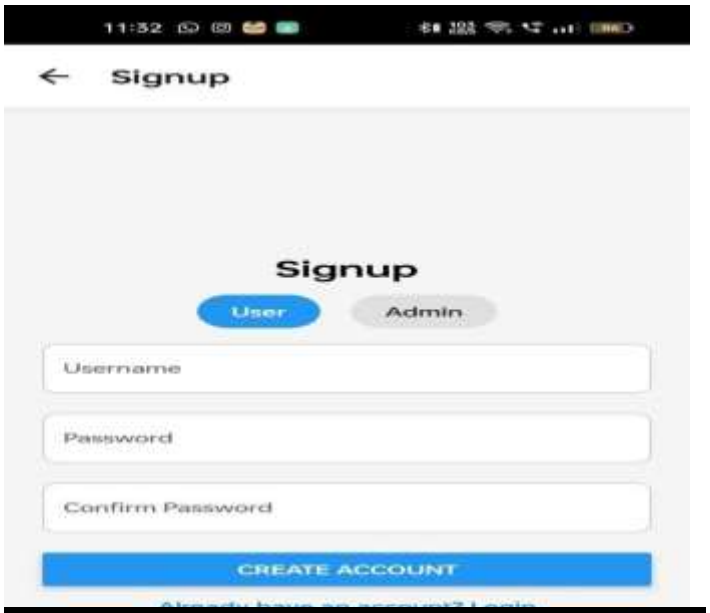
3. responsible for monitoring collected data and managing issue tickets.

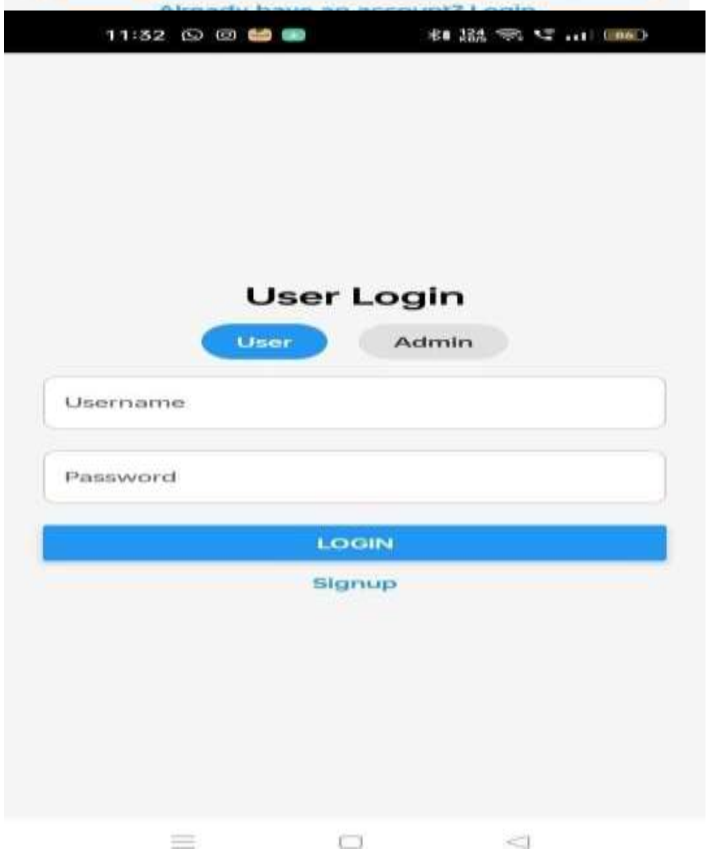
- The login form includes:
  - ○ Username Field: Accepts user's registered username.
  - ○ Password Field: Accepts secure password input (masked for privacy).

A Login button validates the credentials and redirects to the respective dashboard.

A Signup link is provided for new users to create an account.

Fig.2.0. & Fig.2.1.



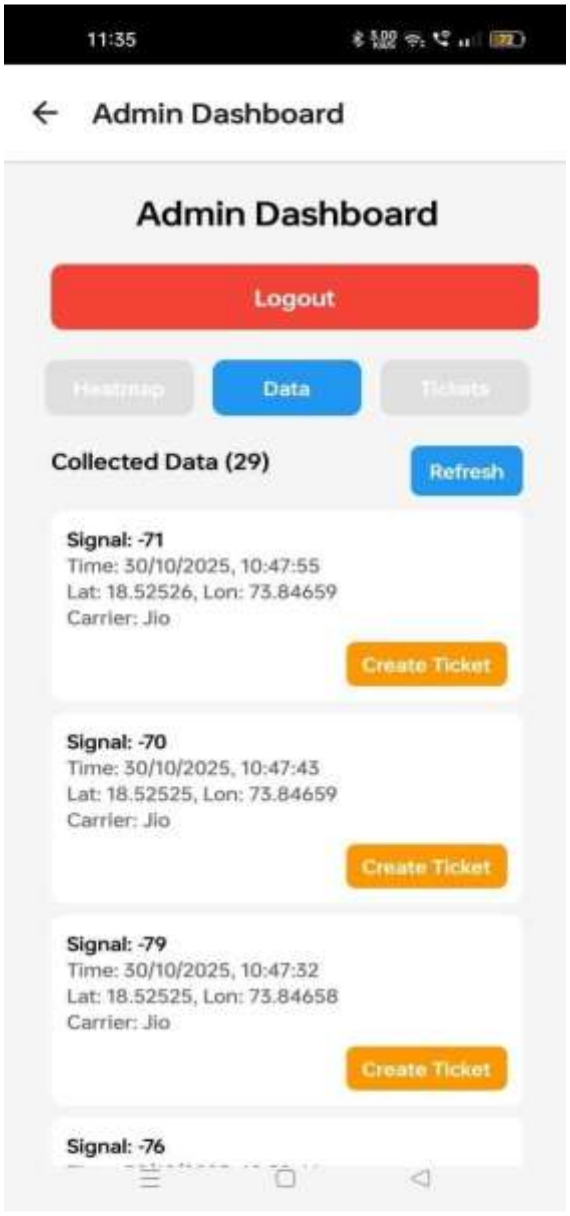


10. Collected Signal Data View

Key Points:

- Each entry logs:
  - Signal Strength (dBm)
  - Latitude & Longitude
  - Timestamp
  - Carrier name
- Enables detailed offline analysis and performance auditing.

Fig.3.0. & Fig.3.1.



11:35

Admin Dashboard

Admin Dashboard

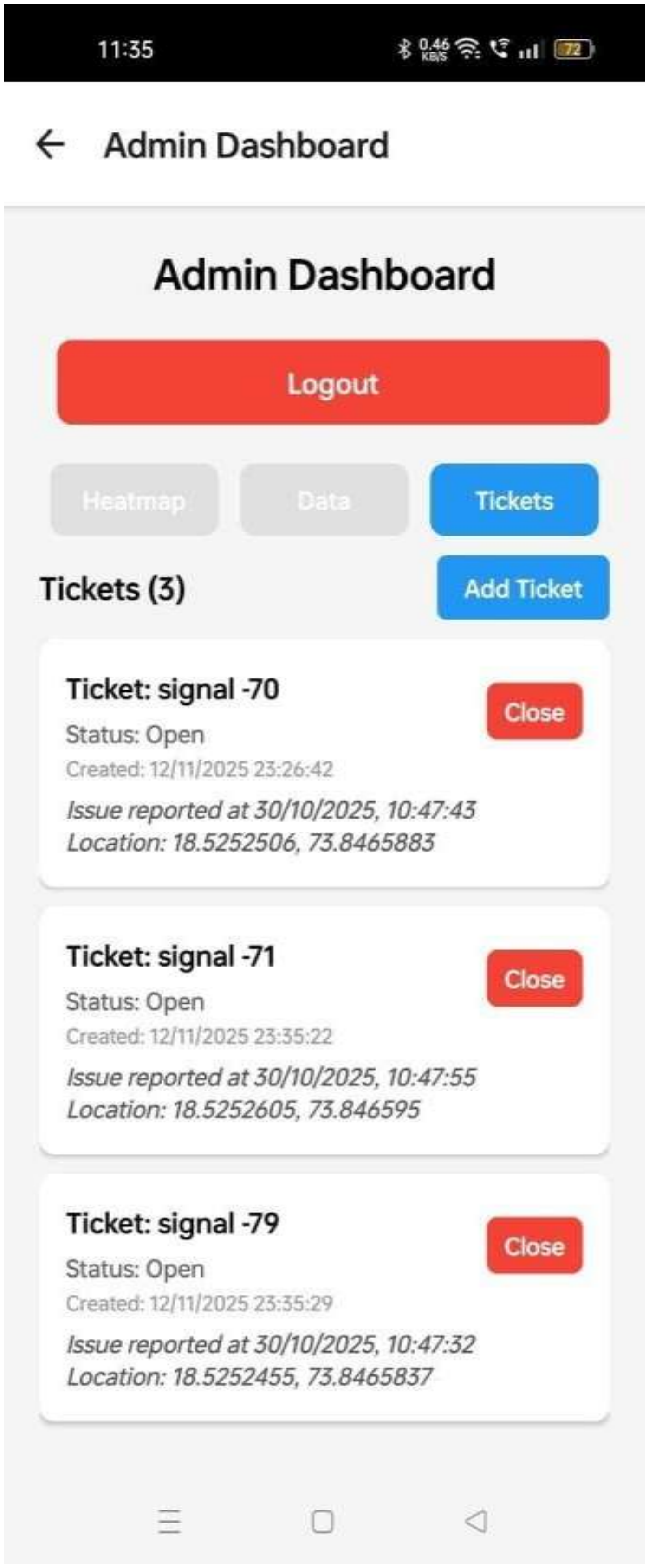
Collected Data (29)

Signal: -71
Time: 30/10/2025, 10:47:55
Lat: 18.52526, Lon: 73.84659
Carrier: Jio

Signal: -70
Time: 30/10/2025, 10:47:43
Lat: 18.52525, Lon: 73.84659
Carrier: Jio

Signal: -79
Time: 30/10/2025, 10:47:32
Lat: 18.52525, Lon: 73.84658
Carrier: Jio

Signal: -76



11:35

Admin Dashboard

Admin Dashboard

Tickets (3)

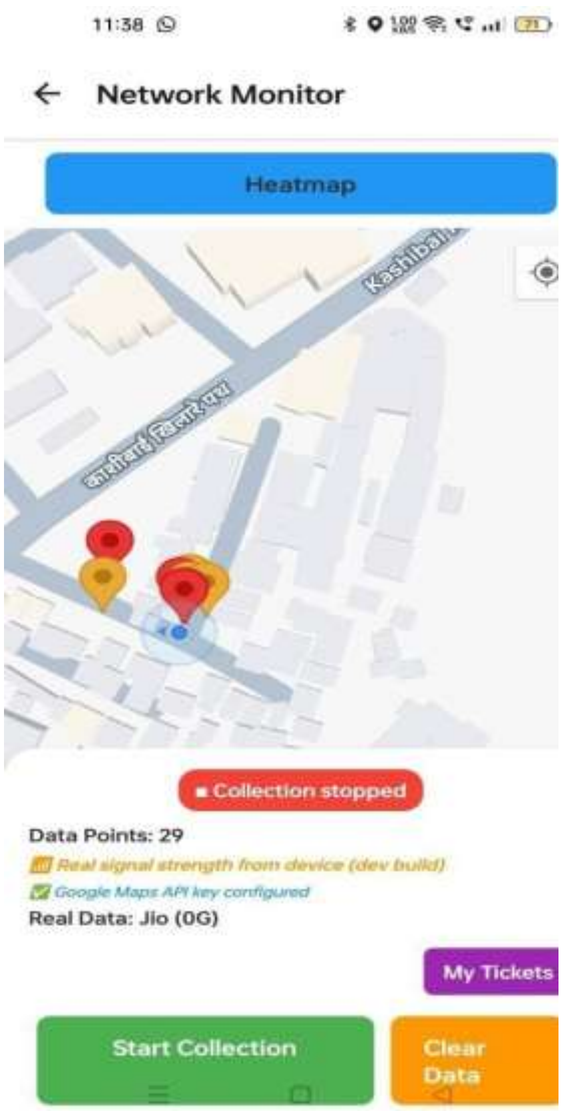
Ticket: signal -70
Status: Open
Created: 12/11/2025 23:26:42
Issue reported at 30/10/2025, 10:47:43
Location: 18.5252506, 73.8465883

Ticket: signal -71
Status: Open
Created: 12/11/2025 23:35:22
Issue reported at 30/10/2025, 10:47:55
Location: 18.5252605, 73.846595

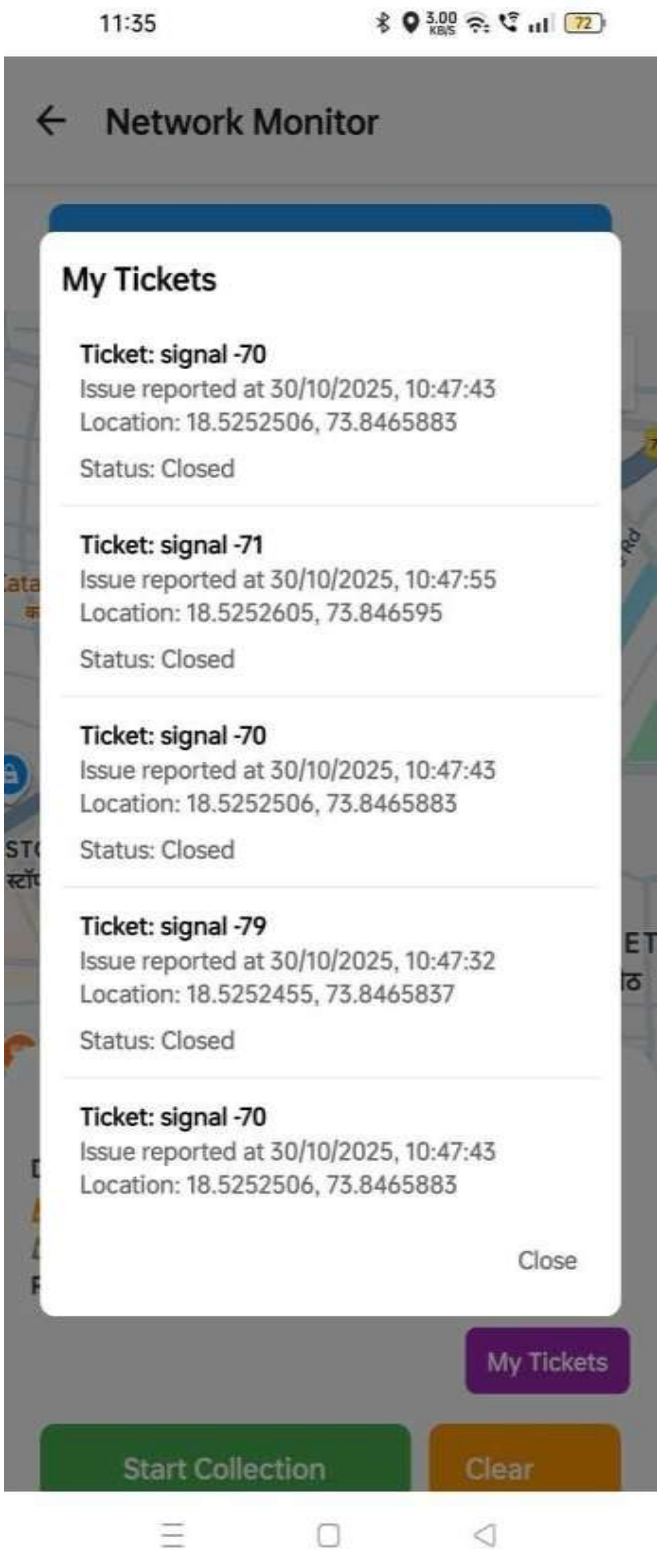
Ticket: signal -79
Status: Open
Created: 12/11/2025 23:35:29
Issue reported at 30/10/2025, 10:47:32
Location: 18.5252455, 73.8465837

11. Results and Analysis

Testing demonstrated that the heatmap module functioned accurately, with signal strength visualization between 60–80% reliability. The system correctly displayed live signal zones using color- coded points. Admins could manually generate tickets based on flagged thresholds (e.g., low-signal clusters).
Data storage in Firebase worked efficiently, allowing real-time retrieval. Fig.4.0 & Fig.4.1.



Data Points: 29
Real signal strength from device (dev build)
Google Maps API key configured
Real Data: Jio (0G)



Ticket: signal -70
Issue reported at 30/10/2025, 10:47:43
Location: 18.5252506, 73.8465883
Status: Closed

Ticket: signal -71
Issue reported at 30/10/2025, 10:47:55
Location: 18.5252605, 73.846595
Status: Closed

Ticket: signal -70
Issue reported at 30/10/2025, 10:47:43
Location: 18.5252506, 73.8465883
Status: Closed

Ticket: signal -79
Issue reported at 30/10/2025, 10:47:32
Location: 18.5252455, 73.8465837
Status: Closed

Ticket: signal -70
Issue reported at 30/10/2025, 10:47:43
Location: 18.5252506, 73.8465883

12. Testing and Evaluation

Scenario 1: Urban test (Deccan, Pune) – Data collection from multiple SIMs showed varying signal levels.
Scenario 2: Admin validation
– Admin could visualize data clusters and identify poor connectivity spots.
Scenario 3: Speed variation
– Speed thresholds triggered ticket generation.
Overall, the system achieved expected outcomes for prototype validation.

13. Future Scope

- Integration of AI-based prediction models for signal strength forecasting.
- Cloud-based analytics for long-term performance tracking.
- Offline mode for rural data collection.
- Enhanced heatmap with colored dots for granular visualization.
- Integration with telecom dashboards for automated report generation.

14. Conclusion

NetMapper successfully demonstrates the potential of combining real-time network data collection, heatmap visualization, and admin management into a unified mobile solution. Although in prototype phase, it provides an innovative foundation for telecom network optimization, helping providers identify and resolve coverage issues efficiently. With future AI and analytics integration, NetMapper can evolve into a powerful nationwide connectivity assessment platform.

References

[1] S. K. Singh and R. Mehra, "Signal Mapping for Rural Network Enhancement," Int. J. Mobile Comput., vol. 9, no. 2, pp. 45–50, 2021.
[2] M. Chen, L. Zhou, and K. Wang, "Cloud-based Mobile Analytics for Real-Time Network Monitoring," IEEE Access, vol. 10, pp. 12543–12555, 2022.
[3] T. Banerjee and P. Rao, "Crowdsourced Network Optimization for Telecom Planning," Proc. IEEE ICAC, pp. 314–320, 2020.
[4] Y. Zhang and X. Li, "Visualization of Signal Distribution using Heatmaps," IEEE Commun. Mag., vol. 57, no. 11,
pp. 62–68, 2019.

Acknowledgments

We would like to express our sincere gratitude to our project guide, Dr.
Ashwini Garkhedkar, for her invaluable guidance and constant support throughout this research.

We are also thankful to the Progressive Education Society's Modern College of Engineering, Pune, for providing the necessary facilities. We extend our appreciation to the faculty members and our fellow students who provided valuable feedback.

Finally, special thanks to the open- source community for providing the powerful tools and libraries, such as JavaScript and Firebase, that made this work possible.