Urban Shield: Crime Risk Mapping and Alert System

Sharvi Kotpalliwar[1], Manasvi Ainchwar[2], Shruti Bucche[3], Chaitanya Jogekar[4], Soham Sirpurwar[5], Prof. Madhavi Sadu[6]
[1,2,3,4,5]*Student, Department of CSE, Rajiv Gandhi College of Engineering Research and Technology, Chandrapur, Maharashtra, India*
[6]*Assistant Professor, Department of CSE, Rajiv Gandhi College of Engineering Research and Technology, Chandrapur, Maharashtra, India*

***Abstract*—Urban crime management presents a persistent challenge for law enforcement agencies in rapidly growing Indian cities where reactive policing methods often fall short. UrbanShield is a full-stack web application that integrates crime prediction, real-time emergency alerts, live location sharing, and community-based incident reporting into a single deployed platform. The backend is built on Node.js with Express.js and MongoDB Atlas for persistent storage. The frontend delivers an interactive crime heatmap using Leaflet.js over OpenStreetMap tiles. Emergency notifications are dispatched via the Brevo HTTP API. The system includes a one-tap SOS emergency alert with live GPS tracking, a voluntary location-sharing module, a crime rate predictor that classifies neighbourhoods into four risk tiers using a weighted scoring algorithm, and a community incident reporting system with Claude AI pre-screening followed by admin review and approval. The application is deployed on Railway.app and is accessible on any device without installation. Testing confirms that all major features function correctly across registration, OTP verification, SOS activation, location sharing, crime reporting, and admin moderation workflows.**

***Index Terms*—Crime Prediction, SOS Alert, Live Location Sharing, Crime Heatmap, Urban Safety, Node.js, Express.js, MongoDB Atlas, Leaflet.js, Brevo HTTP API, Claude AI, Railway.app, OpenStreetMap**

I. INTRODUCTION

Rapid urbanisation across Indian cities has placed significant strain on traditional policing infrastructure. Chandrapur, Maharashtra, an industrial city with a growing population, has seen rising incidents of theft, assault, and harassment in recent years. Despite this, citizens are offered little more than emergency helpline numbers, with no tools to anticipate risk or communicate distress in real time.

Existing safety systems operate reactively. Alerts are issued only after incidents have been reported and logged, meaning citizens receive no warning before entering a high-risk area. Furthermore, when emergencies do occur, there is no simple mechanism for a person to share their live location with trusted contacts or to notify them automatically.

UrbanShield is a full-stack web application designed to address these gaps. It brings together crime prediction, real-time emergency response, live location sharing, and community incident reporting into a single accessible platform deployable on any modern browser without installation.

The main objectives of the system are:

- To develop a crime risk prediction tool that classifies neighbourhoods into safety tiers based on historical crime data.
- To provide a one-tap SOS emergency alert that shares the user's live GPS location with pre-registered emergency contacts.
- To enable voluntary live location sharing with a stated destination and safe-arrival confirmation.
- To allow citizens to submit incident reports that are screened by Claude AI and reviewed by an administrator before appearing on the public crime map.
- To provide an interactive, filterable crime heatmap for situational awareness.

The system is deployed on Railway.app with MongoDB Atlas as the cloud database and Brevo HTTP API for transactional email delivery. This paper presents the system design, technology stack, feature implementation, and testing outcomes of UrbanShield.

II. LITERATURE REVIEW

A. Crime Mapping Using GIS

Geographic Information Systems have been central to criminological research since the 1990s. Kernel Density Estimation remains a widely used technique for visualising crime concentration, converting point incident data into continuous risk surfaces that reveal spatial patterns invisible in raw records. Studies consistently show that crime clusters at predictable environmental features such as commercial areas, transit nodes, and poorly lit corridors [1],[2]. UrbanShield implements a browser-native equivalent using Leaflet.js and the leaflet.heat plugin, streaming incident data directly from MongoDB Atlas and rendering weighted heatmap layers without requiring any external GIS software.

B. Machine Learning for Crime Prediction

Several studies have applied machine learning to crime prediction with varying approaches. Decision Trees, Naive Bayes, K-Nearest Neighbors, Random Forest, and Support Vector Machines have all been evaluated on structured crime datasets. Random Forest has consistently shown strong results on tabular crime data due to its resistance to overfitting and its ability to handle mixed feature types [3],[4]. The concept of feature weighting from Random Forest inspired the scoring algorithm used in UrbanShield, where crime severity, incident volume, and recency are combined as weighted inputs to produce a risk score.

C. Emergency Alert and Location Sharing Systems

Mobile safety applications such as SafetiPin and bSafe provide emergency contact notification and location sharing features. However, most rely on simple push-button alerts without GPS route history or automatic stop timers. Few provide a combined platform that includes both emergency SOS and voluntary journey tracking. UrbanShield extends this category by implementing a full SOS lifecycle including activation, continuous GPS telemetry, live route display for contacts, and safe-arrival confirmation [5].

D. Community-Based Incident Reporting

Crowdsourced crime reporting platforms allow citizens to contribute incident data that supplements official records. However, open reporting systems are prone to spam, vague, or false submissions. UrbanShield addresses this through an automated Claude AI screening layer followed by an admin moderation layer where every report passing AI screening is reviewed before it influences the public crime map, ensuring data quality [6].

E. Research Gaps Addressed

Existing systems typically address one function in isolation, either prediction, emergency alerting, or reporting. UrbanShield uniquely integrates all three within a single deployed application, with real-time database connectivity, OTP-verified user accounts, and cross-feature data flow where approved community reports feed directly into the same crime map used by the predictor and heatmap modules.

III. SYSTEM ARCHITECTURE

A. Technology Stack

Table I presents the complete technology stack used in UrbanShield. Every component listed was used in the actual implementation.

Table I. UrbanShield Technology Stack

Component	Technology Used
Frontend	HTML5, CSS3, Vanilla JavaScript
Map Rendering	Leaflet.js + leaflet.heat plugin
Map Tiles	OpenStreetMap (free, no API key)
Backend	Node.js with Express.js
Database	MongoDB Atlas via Mongoose
Email Service	Brevo HTTP API via axios
AI Verification	Anthropic Claude API
HTTP Client	axios
Environment	dotenv
Deployment	Railway.app

B. Database Collections

UrbanShield uses five MongoDB collections managed through Mongoose schemas. The User collection stores name, email, contact number, password, and an array of emergency contacts each carrying an email

address and a verified boolean flag. The SOS collection records active emergency sessions with the user email, a GPS location array capped at 100 entries, tracking duration, and an active flag.

The ShareSession collection records voluntary sharing sessions with a destination string, a location array capped at 200 entries, and an active flag. The Report collection stores community incident submissions with category, description, location text and coordinates, severity level, incident time, AI verdict, admin note, and a status field of pending, approved, or rejected. The Crime collection holds the historical crime dataset with type, area name, latitude, longitude, date, and time, which is used by both the heatmap and the predictor.

C. API Design

The Express server exposes REST endpoints organised into five groups. Authentication endpoints handle /register, /verify-otp, /resend-otp, and /login. Profile endpoints handle /get-profile, /update-profile, /update-emails, /update-emergency-email, /delete-emergency-email, /send-emergency-otp, and /verify-emergency-otp. Emergency endpoints handle /send-sos, /update-location, /stop-sos, /reached-safe, and /get-sos/:id. Location sharing endpoints handle /start-share, /update-share-location, /get-share/:id, /reached-destination, and /stop-share. Reporting endpoints handle /submit-report, /my-reports, /admin-reports, /update-report-status, /approved-reports, /get-users, and /crimes. All endpoints return a JSON object containing a success boolean and a message string. CORS is enabled globally to allow browser access.

D. System Flow

The client communicates with the Express backend over HTTPS using the browser fetch API. The backend interacts with MongoDB Atlas for all persistent data operations. For email delivery, the backend makes direct HTTP POST requests to the Brevo API using axios, passing the API key from an environment variable stored securely on Railway.app. For report verification, the backend calls the Anthropic Claude API before saving any report. GPS coordinates are collected client-side using the browser Geolocation API and posted to the backend at defined intervals. The Leaflet.js library renders all maps entirely on the client side using free OpenStreetMap tiles.

IV. FEATURE IMPLEMENTATION

A. User Registration and OTP Verification

The registration form collects full name, email address, contact number, password of at least six characters, and exactly three emergency email addresses. On submission, the server generates a random six-digit OTP, stores the pending user data in server memory, and dispatches the OTP to the user's email via the Brevo HTTP API. The user enters the OTP on the verification page. If it matches, the user record is saved to MongoDB with verified: true and the pending data is cleared from memory. A resend endpoint regenerates and dispatches a new OTP. Emergency contacts also go through a separate OTP verification flow. Only contacts marked as verified receive SOS and location sharing email alerts.

B. Crime Heatmap

The crime map page fetches all documents from the Crime collection via GET /crimes and passes the coordinate array to the Leaflet.heat plugin. The heatmap gradient is set with green at low density, yellow at medium, and red at high density, with a radius of 40 and blur of 25. A dropdown menu allows filtering by crime type including Theft, Assault, Robbery, and Burglary without reloading the page. The dashboard homepage also displays a lighter version of the same heatmap as a preview. Approved community reports with valid coordinates are fetched from /approved-reports and overlaid on the same map, meaning citizen-submitted data feeds directly into the heatmap over time.

C. Crime Rate Predictor

The predictor page allows a user to enter an area name and optionally filter by time of day. The system filters the locally cached Crime array for entries whose area field matches the input. A weighted risk score between 0 and 100 is then computed from three features. Crime severity contributes 50 percent of the score. Each crime type is mapped to a severity weight, for example murder is 10, robbery is 8, assault is 7, theft is 5, and vandalism is 3. The average severity across matched crimes is normalised to the range 0 to 1. Crime volume contributes 30 percent and is the total number of

matching incidents divided by a normalisation maximum of 50. Crime recency contributes 20 percent and is the count of incidents in the last 30 days divided by 20. The composite score maps to four tiers: Safe for 30 or below, Moderate for 31 to 60, High for 61 to 75, and Critical above 75. The result page shows an animated donut chart, a crime type breakdown with percentage bars, a time slot distribution, and safety recommendations tailored to the risk tier.

D. SOS Emergency System

The SOS button is accessible from both the homepage and the dashboard. The user selects a tracking duration of 5, 10, 30, or 60 minutes. The browser requests GPS coordinates via the Geolocation API and posts them with the user email and duration to /send-sos. The server creates a SOS document, stores the initial coordinates, and sends an alert email to every verified emergency contact via Brevo HTTP API.

The email contains a live tracking link. The user is redirected to sos.html showing a countdown timer, a live Leaflet map, and emergency helpline numbers: Police 112, Ambulance 108, Women Helpline 1091, Fire 101, and Disaster Management 1078. The browser posts updated GPS coordinates to /update-location every 60 seconds. The array is capped at 100 entries. Emergency contacts who open the tracking link see a read-only page that polls /get-sos/:id every 4 seconds and renders the route as a polyline. SOS stops manually or on timer expiry, both sending a stop notification email.

E. Live Location Sharing

The location sharing feature is separate from SOS and is for voluntary journey tracking. The user enters an optional destination, grants geolocation permission, and the browser posts initial coordinates to /start-share. The server creates a ShareSession and sends a notification email with a viewer link to all verified contacts. In sharer mode the GPS is sent every 60 seconds with an auto-stop timer and an I Reached Safely button. In viewer mode the page polls /get-share/:id every 15 seconds and renders the accumulated route. Pressing I Reached Safely sends a safe-arrival email to all contacts. Sessions support up to 200 waypoints.

F. Community Incident Reporting

The report form presents eight category chips: Theft, Assault, Harassment, Vandalism, Suspicious Person, Suspicious Vehicle, Drug Activity, and Other. The user writes a description, enters a location, optionally detects GPS coordinates, selects date and time, and chooses a severity level. On submission the server calls the Claude AI verification function. Claude evaluates whether the report is genuine or vague and returns a JSON response with an approved boolean and a reason string. If Claude rejects the report, the user receives the reason immediately and the report is not saved. If Claude approves, the report is saved to MongoDB with status pending and the AI verdict is stored alongside it for admin reference.

G. Admin Dashboard

The admin dashboard is accessible after login with the administrator account. It displays summary counts for total, pending, approved, and rejected reports. A filterable and searchable table lists all submitted reports. The admin can open a detail view for any report and either approve or reject it with an optional note. Approval triggers a confirmation email to the submitting user. Rejection triggers a rejection email with the admin note. Approved reports with coordinates appear on the public crime map. The dashboard also has a User's section listing all registered accounts with their verification status and emergency contacts.

H. Profile Management

The profile page allows users to update their display name and contact number. Emergency contacts are displayed as cards showing verification status. Each contact can be updated, deleted, or verified via OTP. A new contact can be added at any time. Only verified contacts receive SOS and location sharing alerts.

V. SYSTEM TESTING

UrbanShield was tested manually through end-to-end workflow execution covering all major user journeys. Testing was conducted in the live deployed environment on Railway.app with MongoDB Atlas as the active database and Brevo handling real email delivery.

A. Registration and OTP Flow

Test accounts were created using valid email addresses. OTP emails were received within seconds via Brevo. Entering a correct OTP successfully created the user record in MongoDB with verified: true. Entering an incorrect OTP returned the appropriate error message. The resend OTP endpoint generated a new code. Emergency contact OTP verification confirmed that only verified contacts were notified during SOS and location sharing tests.

B. SOS and Location Sharing

SOS was activated from the dashboard using a test account with three verified emergency contacts. Alert emails containing the live tracking link were received by all three contacts.

The countdown timer started correctly on sos.html. GPS location posted to the backend every 60 seconds and the map marker updated. The emergency contact tracking page correctly displayed the route polyline refreshing every 4 seconds. Stop SOS marked the session inactive and a stop notification email was confirmed. The same workflow was repeated for location sharing and the I Reached Safely confirmation email was received.

C. Crime Heatmap and Predictor

The crime map loaded historical crime data and rendered the heatmap correctly. The filter dropdown updated the heatmap for each crime type without page reload. The predictor was tested with area names present in the database and returned correct risk tier results with all visual components rendering. Searching for an unknown area returned the No Data Found state.

D. Reporting and Admin Moderation

Reports were submitted with both detailed and vague descriptions to test the Claude AI screening layer. Vague submissions were correctly rejected by Claude AI with a reason shown to the user. Detailed reports were approved by Claude AI and entered the admin review queue. The admin dashboard displayed all pending reports correctly. Approve and reject actions updated the report status and triggered the appropriate email. Approved reports appeared on the crime map.

E. Functional Test Summary

Table II. Functional Test Results

Feature	Status	Notes
User Registration + OTP	Pass	Email delivered
Login / Logout	Pass	Session stored
Emergency Contact OTP	Pass	Verified flag set
SOS Activation	Pass	Emails sent
Live GPS Tracking	Pass	60s updates
SOS Stop + Email	Pass	Contacts notified
Location Sharing	Pass	Viewer mode works
Safe Arrival Email	Pass	All contacts notified
Crime Heatmap	Pass	Filter works
Crime Rate Predictor	Pass	4 tiers shown
Claude AI Screening	Pass	Vague rejected
Admin Approve/Reject	Pass	Email on decision
Crime Map Update	Pass	Approved shown
Profile Management	Pass	Updates saved

VI. DISCUSSION

A. Integration of Features

A key strength of UrbanShield is that its features are not isolated from each other. Community reports that pass Claude AI screening and are approved by the admin automatically feed into the crime heatmap and become part of the dataset used by the predictor. This means the system improves over time as more citizens contribute verified incident data. The SOS and location sharing modules share the same emergency

contact infrastructure, meaning a single OTP verification step covers both features for each contact.

B. Use of Free and Freemium Services

The entire system is built on free or freemium services. MongoDB Atlas provides a free tier sufficient for this scale of deployment. Railway.app provides managed hosting with environment variable support. OpenStreetMap and Leaflet.js require no API key. Brevo provides a free email tier. This makes the system reproducible and deployable at minimal cost, which is relevant for civic technology in developing regions.

C. Limitations

The crime predictor uses a simple weighted scoring formula rather than a trained machine learning model. Its accuracy is therefore limited by the quality and completeness of the seed crime dataset. The predictor does not learn from new data over time. GPS tracking relies on the browser Geolocation API, which may be less accurate indoors or in areas with poor signal. Email delivery via Brevo is not instantaneous and does not guarantee the contact will see the alert immediately. Push notifications would improve response time but are not currently implemented.

VII. ETHICAL CONSIDERATIONS

A. User Consent and Data Control

GPS location data is only collected when the user explicitly activates SOS or location sharing. No background tracking occurs. Users can stop sharing at any time. Emergency contacts must actively verify their email addresses via OTP before receiving any alerts, ensuring they have consented to receive notifications.

B. Fairness in Prediction

The crime rate predictor uses only crime type, location, and time as inputs. No demographic data including religion, caste, gender, or economic status is used at any stage. A minimum evidence threshold means that areas with very few historical records do not receive inflated risk scores.

C. Human Oversight in Reporting

All community-submitted incident reports pass through two layers of review. The Claude AI layer filters out vague, spam, or test submissions. The admin layer then makes the final decision on whether a report is credible enough to appear on the public crime map. No automated decision directly influences the map without human approval.

D. Credential Security

All API keys including the Brevo API key, the MongoDB connection string, and the Claude API key are stored as environment variables on Railway.app. No credentials are hardcoded in source code or exposed to the client side.

VIII. CONCLUSION

UrbanShield is a full-stack urban safety web application that successfully integrates crime risk prediction, one-tap SOS emergency alerting, voluntary live location sharing, interactive crime heatmap visualisation, community incident reporting with Claude AI pre-screening, and admin moderation into a single deployed platform. The system uses Node.js, Express.js, MongoDB Atlas, Leaflet.js, OpenStreetMap, the Brevo HTTP API, the Anthropic Claude API, and Railway.app, all of which are free or freemium services that make the system reproducible at minimal cost.

Manual end-to-end testing confirmed that all major features function correctly in the live deployment. The system demonstrates that civic safety tools can be built and shipped by a small development team without expensive infrastructure, making this approach viable for other cities facing similar urban safety challenges.

Future improvements will focus on replacing the weighted scoring formula with a trained machine learning model updated on new data, adding browser push notifications alongside email for faster SOS delivery, introducing time-of-day risk forecasting, and conducting a formal user study with citizens and law enforcement in Chandrapur to measure real-world impact on safety perception and incident response times.

ACKNOWLEDGMENT

The authors gratefully acknowledge the guidance of Prof. Madhavi Sadu throughout the design, development, and testing phases of this project. Thanks, are also due to the faculty of the Department

of Computer Science and Engineering at RCERT, Chandrapur, for their valuable feedback during internal reviews. The project made use of several open-source and freemium tools including Leaflet.js, Express.js, Mongoose, MongoDB Atlas, Railway.app, Brevo, OpenStreetMap, and the Anthropic Claude API.

REFERENCES

[1] Gupta, R. Sharma, and S. Mehta, "Crime prediction using machine learning techniques," *International Journal of Computer Applications*, vol. 185, no. 12, pp. 25–30, 2023.

[2] P. Sharma and K. Verma, "A study on crime data analysis using data mining," International Journal of Data Science and Analytics, vol. 8, no. 2, pp. 45–52, 2022.

[3] N. Singh, "Analysis of crime data using Python," International Journal of Computer Science Trends and Technology, vol. 11, no. 3, pp. 90–96, 2023.

[4] A. Kumar, V. Mishra, and P. Joshi, "Smart crime detection and prediction system using AI," International Journal of Advanced Research in Computer Science, vol. 16, no. 1, pp. 10–18, 2025.

[5] S. K. Dubey and R. Tiwari, "Crime forecasting using machine learning and deep learning techniques," International Journal of Emerging Technologies, vol. 12, no. 4, pp. 112–120, 2023.

[6] R. Verma, S. Patel, and A. Singh, "Crime rate prediction using machine learning algorithms," Journal of Artificial Intelligence Research, vol. 15, no. 1, pp. 60–70, 2024.

[7] T. Roy and A. Chakraborty, "Crime hotspot detection using K-means clustering," International Journal of Computer Applications, vol. 184, no. 9, pp. 15–21, 2022.

[8] K. Reddy and S. Kumar, "Crime prediction using supervised learning techniques," Journal of Machine Learning Applications, vol. 9, no. 1, pp. 55–63, 2023.

[9] J. Brown and M. Davis, "Urban crime analysis using big data analytics," IEEE Access, vol. 11, pp. 10234–10245, 2023.

[10] L. Wang, H. Li, and Y. Chen, "Deep learning for crime prediction in smart cities," IEEE Transactions on Industrial Informatics, vol. 19, no. 2, pp. 1456–1465, 2024.

[11] P. Singh and R. Kaur, "Comparative study of machine learning algorithms for crime prediction," Journal of Computer Science and Engineering, vol. 13, no. 2, pp. 120–130, 2023.

[12] D. Lee and K. Park, "Spatio-temporal crime prediction using neural networks," Expert Systems with Applications, vol. 198, pp. 116–125, 2022.

[13] B. Kumar and S. Srivastava, "Use of AI in crime detection and prevention," International Journal of Innovative Research in Technology, vol. 10, no. 5, pp. 200–207, 2023.

[14] E. Johnson and R. Miller, "Data-driven approaches for crime analysis in urban areas," Journal of Big Data, vol. 10, no. 1, pp. 1–12, 2023.

[15] M. Patel and D. Shah, "Analysis of crime patterns using clustering algorithms," Journal of Data Mining and Knowledge Discovery, vol. 10, no. 3, pp. 75–84, 2022.

[16] S. Das and P. Banerjee, "Crime trend analysis using data mining techniques," International Journal of Advanced Computer Science, vol. 15, no. 1, pp. 67–75, 2023.

[17] R. Thomas and J. Wilson, "Predictive policing and machine learning: A review," ACM Computing Surveys, vol. 55, no. 4, pp. 1–28, 2023.

[18] V. Gupta and N. Jain, "Predictive analytics in crime data using Python," International Journal of Scientific Research in Computer Science, vol. 11, no. 2, pp. 33–40, 2023.

[19] W. Chen and Y. Liu, "Smart city crime prevention using IoT and ML," IEEE Internet of Things Journal, vol. 10, no. 5, pp. 4321–4330, 2023.

[20] H. Zhang and X. Liu, "Crime prediction with random forest and SVM," Journal of Artificial Intelligence and Data Mining, vol. 11, no. 2, pp. 101–110, 2024.