Journal Homepage: www.ijarpr.com ISSN: 3049-0103 (Online)

International Journal of Advance Research Publication and Reviews

Vol 03, Issue 10, pp 771-777, October, 2026

SafePath AI – AI-Powered Safety Navigation System

Arun Kumar S[1*], Arjun Chokkalingam K[2], Arunkumar C[3], Darshan JB[4], Dharshan K[5], E. Subramanian[6]

[1–5]Undergraduate Students, Department of Computer Science and Engineering (CSE), Sri Shakthi Institute of Engineering and Technology, Coimbatore, Tamil Nadu, India

[6]Project mentor: Assistant Professor, Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, Coimbatore, Tamil Nadu, India

[1]arunkumar16112007@gmail.com, [2]arjunchokkalingam07@gmail.com, [3]arunkumarc25cs@srishakthi.ac.in, [4]darshanlokith@gmail.com, [5]dharshan1503.03@gmail.com, [6]esubramaniancse@siet.ac.in

ABSTRACT

Conventional navigation applications optimise travel time and distance and give limited support to travellers who also want to weigh personal-safety information. This paper describes SafePath AI, a full-stack web prototype that combines news-derived crime incident data, road-network route alternatives, emergency-facility discovery and live proximity alerts in one interface. The frontend uses React and Vite; the backend uses Node.js and Express with MongoDB and an in-memory fallback. A scheduled pipeline reads RSS feeds of Indian news publishers, filters crime-related items with keyword rules, locates them with a hand-built gazetteer, merges duplicates and applies a rolling ten-day eligibility rule. Candidate routes from the OpenStreetMap-based OSRM service are ranked by a transparent weighted score that combines time-decayed incident and historical-crime proximity with distance and with proxy terms for lighting, traffic and crowding. The system contains no trained machine-learning model; "AI" here means rule-based decision support. The backend automated test suite of 99 tests passed, but no accuracy, latency or user-study results are reported and some data sources are sample data. SafePath AI supports decisions only and cannot guarantee the safety of any route.

Keywords: SafePath AI; safety-aware navigation; crime news analysis; route risk assessment; emergency services; OpenStreetMap; decision support

1. Introduction

Navigation services are designed around travel time and distance. A traveller who also wants to avoid areas with recently reported crime, or to know where the nearest hospital or police station is, must normally consult several unrelated sources. Research on urban navigation has argued that the shortest path is not always the most suitable one and that safety can be treated as a second routing objective next to length (Galbrun et al., 2016). Crime is also spatially uneven, and mapping methods differ in how well they anticipate where it will occur (Chainey et al., 2008).

SafePath AI is a college software-engineering prototype that applies this idea in an Indian setting. It combines news-derived incident reports, route alternatives from a road-network routing service, a facility finder for emergency services and a live alert page. This paper documents what the prototype actually implements, states what has and has not been verified, and identifies the limits that follow from its data sources. The contribution is an integration and implementation approach for one decision-support application; it is not a new routing algorithm or a trained prediction model.

2. Problem Statement and Objectives

Safety-related information (recent incident reports, nearby emergency facilities and emergency helplines) is scattered across separate tools and is rarely connected to route choice. The project objectives are to connect recent incident information with map-based route comparison; to present up to three road-based route alternatives ranked with the available evidence; to show nearby police, hospital, fire and other emergency facilities within a selectable radius; to warn users of reported incidents near their location; and to state the limitations caused by incomplete, delayed or geographically biased data. These are design objectives; Section 5 separates what was observed from what remains unverified.

3. Related Work

Galbrun et al. (2016) built a risk model for the street networks of Chicago and Philadelphia from public crime data and treated navigation as a biobjective shortest-path problem, returning a small set of paths that trade distance against risk. SafePath AI also offers alternatives rather than a single route, but uses a much simpler proximity-based score and has not been evaluated against such a model. CrowdSafe, a demonstration system, combined crowdsourced crime reporting on mobile devices with search of crimes near a location, hotspot analytics and safety routing (Shah et al., 2011); SafePath AI likewise accepts user reports but draws its main incident stream from news feeds and adds emergency-facility discovery.

Chainey et al. (2008) compared four hotspot-mapping techniques for predicting burglary, street crime and vehicle theft and found kernel density estimation the most accurate. The usefulness of past incident locations therefore depends on the method and the crime type; SafePath AI uses a distance-and-age weighted count and no density estimation. Routing on open data depends on map completeness: OSRM is a routing engine for OpenStreetMap data (Luxen & Vetter, 2011), and a global assessment found OpenStreetMap roughly 83% complete by road length, with completeness varying by country and density (Barrington-Leigh & Millard-Ball, 2017).

Among the sources reviewed we did not find a system that joins news-derived incident ingestion, route alternatives, emergency-facility discovery and proximity alerts for Indian users; this statement is limited to those five sources, which do not amount to a systematic review, and is not a claim that no such system exists.

4. System Design and Methodology

4.1 Architecture and data layer

SafePath AI is a client–server web application. A React and Vite frontend (Vite 6 development server) calls a REST API served by Node.js and Express 4 under /api. The backend applies Helmet security headers, request rate limiting, JSON parsing and a global error handler, exposes a health endpoint, serves the built frontend, and starts a scheduled ingestion job. Data are stored through Mongoose models in MongoDB. On start-up the server tries a configured MongoDB URI, then a local MongoDB instance, and otherwise switches to an in-memory store that is saved to a JSON file on disk so that records survive restarts. Table 1 lists the components.

Table 1 - Components confirmed from the inspected source

Component	Confirmed detail
Frontend	React, Vite 6.4.3
Backend	Node.js, Express 4, Helmet, express-rate-limit, CORS, node-cron, axios
Persistence	Mongoose models (incidents, historical crime, lighting, crowd, route history, users); in-memory fallback saved to a JSON file
Routing	OSRM public servers (OpenStreetMap data) with a backup mirror, 10-minute route cache
Emergency data	OpenStreetMap (Overpass, Nominatim, Photon) plus a hard-coded directory of 47 entries
Alerts	Server-Sent Events hub with in-memory buffer of 100 incidents
Authentication	bcryptjs, JSON Web Tokens, Google ID-token verification
Testing	Node.js built-in test runner (node --test) over the backend test files

Five collections were inspected. Each record stores latitude and longitude with a GeoJSON point and a 2dsphere index. Recent incidents hold crime type, dates, location name, source name and URL, a confidence level, a verification status, a severity from 1 to 5 and an expiry date that removes the record 30 days after creation. Route history stores a signed-in user's selected route with origin and destination coordinates, which is location-sensitive data.

4.2 Incident acquisition pipeline

The pipeline runs once at start-up and then hourly. Its registry lists 40 sources: 34 enabled RSS feeds of national and regional publishers (by our count), five entries that are disabled (two official police portals and three news feeds blocked or inactive) and one optional news-API connector that needs a key. Feeds are read in batches of five with a 4.5-second timeout and at most 60 items per feed. Each item passes the following steps: (1) its URL must belong to one of 30 trusted domains and look like an article page, not a section page; (2) a keyword filter rejects political, sports, entertainment, business and

weather items; (3) keyword rules in English and Tamil assign one of seven categories, the first matching category in a fixed order winning; (4) a hand-built gazetteer of about one hundred Indian places gives coordinates, using the headline dateline, an "in/at/near" phrase or a place keyword in that order, and items with no resolved place are discarded; (5) a severity from 1 to 5 is assigned by rules; (6) a rolling ten-day eligibility test is applied, with the article's publication time used as the incident time; and (7) duplicates are merged.

Duplicates are detected first by identical URL and then when two items share a compatible crime type, fall within 24 hours, have matching location names and share at least 35% of their headline keywords; merged items keep all source URLs. A set of ten curated sample incidents stored in a data file is added to the feed. The "verified" status shown by the system means only that the publisher or domain is on the trusted list; the content itself is not independently checked.

4.3 Route generation

The browser sends origin, destination, travel mode and a well-lit-road preference to the backend. The backend asks OSRM for up to three alternatives with full road geometry. Two routes are treated as duplicates when at least 85% of 30 sampled points of each lie within 45 m of the other. When OSRM returns fewer than three distinct routes, extra candidates are generated by routing through waypoints placed to either side of the straight line at 35%, 50% and 65% of its length, and accepted if their length is between 0.4 and 2.2 times that of the first route; these are detours, not native OSRM alternatives. Walking, cycling and motorcycle times are estimated from distance or by a factor. A request for public transit is not served by live transit data: the controller falls back to road corridors and returns a notice stating that transit needs official agency feeds. If every routing server fails, an error is returned and no synthetic route is created. If the backend itself is unreachable, the browser queries public OSRM servers directly and applies a simpler count-based score, $S = \max(25, 92 - 18n - 1.5d)$, where n is the number of locally stored user-reported incidents within 350 m of the route and d is its length in km.

4.4 Route risk score

The route controller reads the stored historical crime records (cached for 15 minutes) and the recent incidents inside the rolling ten-day window, adds live user reports buffered by the alert hub, and passes to the risk engine only incidents that are marked verified or carry a source URL; pending user reports without a source are excluded from route penalisation, although they still trigger alerts. For each candidate route the engine finds incidents within a buffer B of the route geometry and computes two proximity terms. The backend configuration sets the buffer to 250 m, and the engine's own default is 300 m when no value is supplied.

$$s_{\text{rec}} = \min(100, \Sigma_i\, v_i \cdot 15 \cdot c_i \cdot \exp(-a_i/21) \cdot \max(0, 1 - d_i/B)) \quad (1)$$

$$s_{\text{hist}} = \min(100, \Sigma_j\, v_j \cdot 5 \cdot \exp(-a_j/365) \cdot \max(0, 1 - d_j/B)) \quad (2)$$

where v is severity (1–5), a is the age of the record in days, d is the distance from the route in metres and c is a confidence weight (1.0, 0.8 or 0.6 for high, medium or low confidence, and 0.45 for unverified user reports where the engine receives them). The decay factors $\exp(-a/21)$ and $\exp(-a/365)$ have half-lives of about 14.6 and 253 days respectively. The overall risk and safety scores are

$$R = \min(95, \max(5, \text{round}(\Sigma_k\, w_k\, s_k))), \quad S = \max(5, 100 - R) \quad (3)$$

where k ranges over recent incidents, historical crime, lighting, traffic, crowding and distance. The weights come from the backend configuration: 0.20 for recent incidents, 0.30 for historical crime, 0.20 for lighting, 0.15 for traffic, 0.10 for distance and 0.05 for crowding, which sum to 1.00. If no configured weights are available the engine falls back to 0.35, 0.25, 0.15, 0.10, 0.05 and 0.10 for recent, historical, lighting, traffic, crowd and distance, and it uses 0.30, 0.15, 0.35, 0.07, 0.05 and 0.08 when the user selects the well-lit preference. Table 2 shows how each term is obtained and, importantly, which terms rest on real data.

Table 2 - Terms of the route risk score and their data basis

Term	Computation in code	Data basis
Recent incidents	Eq. (1)	News-derived and user-reported incidents; most locations are city-level gazetteer points
Historical crime	Eq. (2)	Stored historical records, labelled NCRB Open Government Crime Data in the API response; record-level provenance not documented here

Term	Computation in code	Data basis
Lighting	12, 28 or 45 by route rank when a lighting provider is passed (15 otherwise); minus 10 if more than five mapped streetlamps lie within 80 m; scaled by 1.4 or 0.7 with the well-lit option	Rank-based proxy, not measured lighting; mapped lamps, when supplied, give only a small reduction
Traffic	min(90, round(2.2 × minutes))	Proxy from travel time; the traffic provider is a mock, so no live traffic data
Crowding	Constant 20	Placeholder
Distance	min(100, round(6 × km))	Route length from OSRM

Routes are sorted by *R*. The lowest is labelled Safest (green), the second Moderate Risk (amber) and the third Higher Risk (red), the last being called Danger only if $R \geq 50$ or an incident lies within the buffer. Labels therefore express rank among the returned alternatives, not a threshold on absolute risk, and a single returned route is still labelled Safest. Each route carries an explanation listing the number and types of nearby incidents, the distance to the closest one, and a data-limitation notice. Mapped streetlamps and CCTV points, when supplied, are reported as supplementary context only and the notice states that mapped cameras do not prove lower risk.

Proximity is measured by the proximity service. A bounding box around the route, widened by the buffer, first discards distant items; the remaining items are compared with the route polyline using the Haversine formula (Earth radius 6371 km). The comparison is made against the vertices of the polyline, not against the segments between them, so an incident beside a long straight segment with few vertices is assigned a larger distance than its true distance to the road.

4.5 Emergency-service discovery

Nearby facilities are gathered in tiers. OpenStreetMap Overpass (two mirrors, at most 40 elements per query) is tried first, then Nominatim and then Photon if nothing is returned. Entries from a hard-coded directory of 47 facilities (19 of them in Coimbatore, the rest in other large cities) are always merged in, duplicates are removed, and results are sorted by straight-line Haversine distance. The search radius is clamped to 1–35 km (5, 10, 15 and 25 km in the interface, 10 km by default) and results are cached for 15 minutes. A “road” distance is displayed as 1.28 times the straight-line distance, and travel time assumes 30 km/h; neither is a routed value. A corridor endpoint searches within 8 km of the origin, mid-route points and destination and keeps facilities within 4 km of the route or near its ends. Contact numbers, descriptions and source labels of the directory entries were entered by hand and have not been independently verified, and hospitals, pharmacies, police and fire facilities are marked as open 24 hours by category unless the OpenStreetMap record says otherwise.

4.6 Alerts

An alert hub keeps open Server-Sent Event connections from browsers. Each client registers a location, a radius and a category; the client default is 10 km, while the backend configuration defines a default alert radius of 15 km. When the ingestion job or a user report supplies an eligible incident, the hub removes reporter identity, ignores it if the same identifier was already sent, and pushes it to every connected client whose Haversine distance and category match. The alerts page scans up to 50 km. Delivery happens only while the page is open; the code contains no mobile push, SMS or e-mail mechanism, and no SOS module was among the files inspected.

4.7 Authentication and security measures

Passwords are hashed with bcrypt and sessions use JSON Web Tokens. Google sign-in verifies the ID token on the server and refuses to link a Google identity to an existing password account with the same e-mail address. Password reset issues a random 32-byte token, stores only its SHA-256 hash with an expiry, accepts it once, and returns the same response for unknown addresses to prevent account enumeration. Rate limiting and Helmet are enabled. Three weaknesses remain: the server accepts requests from any origin while credentials are enabled, the start-up seed creates default demo accounts, and the configuration falls back to a built-in default JWT signing secret when none is set in the environment. All three must be removed or replaced before any deployment.

5. Implementation Status and Observations

This section reports what was observed in the code, in an automated test run of the backend and in screenshots taken on 28 October 2026 from a local development instance. The backend repository contains ten test files covering registration, login, Google sign-in with a mocked verifier, password recovery, location extraction, distance and radius filtering, the ten-day policy, the news pipeline, routing, route risk and emergency-service discovery. A single run of *npm test* (Node.js built-in test runner) reported 99 tests, 99 passed, 0 failed, 0 cancelled and 0 skipped, in about 74.8 s (Table 3). Several

tests call live routing, map-data and news servers; the emergency-service and routing groups took between about 18 s and 32 s each, and results may vary between runs and networks. In one test the live source for The Hindu (Coimbatore) returned a successful response with 60 items scanned, and a failure-isolation test confirmed that one failing news source did not stop the others. No repeated trials, timing measurements, accuracy tests or user studies were carried out.

Table 3 - Automated backend test run (npm test, 99 tests, all passed)

Test group	What it checks	Outcome
Authentication and storage	Registration, duplicate rejection, login, JWT session, persistence across a simulated restart, exclusion of the password hash	Passed
Google sign-in	Valid, expired, wrong-issuer and unverified-email tokens; refusal to link to a password account	Passed
Password recovery and security	Generic reply for unknown e-mail, hashed single-use reset token, protected-route rejection	Passed
Location and distance	Place extraction (Ooty never resolved to Coimbatore), Haversine distance, radius filtering, separation of GPS and alert locations	Passed
Ten-day policy and alerts	Rolling ten-day eligibility, report-to-alert dispatch, report-to-route-risk integration	Passed
News pipeline	Canonical URL validation, source isolation, deduplication, official versus media attribution, one live endpoint	Passed
Routing and risk engine	Three ranked tiers, de-duplicated geometry, honest error on provider failure, route cache, four travel modes	Passed
Emergency services	Radius and category filtering; discovery near Peelamedu (11.0408, 77.0738) for facilities, police and pharmacies	Passed
Total	99 tests, 0 failed, 0 skipped; duration about 74.8 s	99 / 99

Table 4 summarises the status of the main features.

Table 4 - Implementation status of the main features

Feature	Evidence	Status
Authentication	Code, tests, screenshot	Implemented; automated tests passed
Incident ingestion	Code, tests	Implemented with keyword rules; output accuracy not independently measured
Route alternatives	Code, tests, screenshot	Implemented; extra routes partly from waypoint detours
Risk score	Code, tests	Implemented; weights and constants not calibrated or validated
Emergency services	Code, tests, screenshots	Implemented; directory data and contact numbers unverified
Live alerts	Code, tests, screenshot	Implemented via Server-Sent Events; no push or SMS
Lighting, CCTV, traffic, crowd	Code, demo data	Proxies or demo data only; no live source
SOS	None	Not seen in the code inspected

On the emergency page, with the browser location set in Coimbatore, 38 facilities were listed within 10 km (Fig. 1). For the six cards visible, the displayed road distance was between 1.278 and 1.282 times the straight-line distance (Table 5), which agrees with the fixed factor in the code. The alerts page

showed nine nearby incidents and nineteen items in the news feed at the time of capture. The feed includes the curated sample incidents, so these counts do not measure live ingestion alone. The screenshots show that the interfaces exist; they do not show correct classification, correct locations or safer routes.

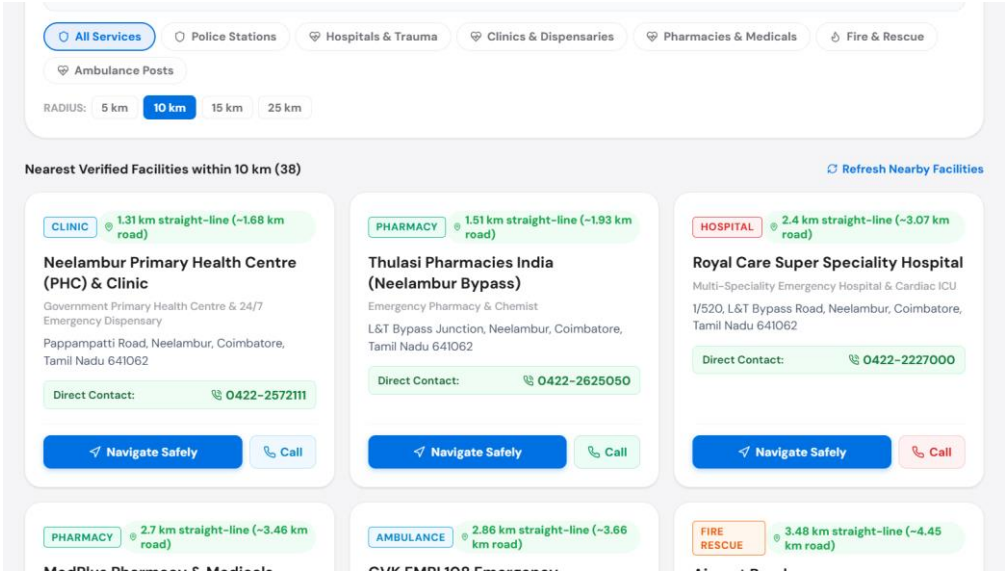


Fig. 1 - Emergency services page listing nearby facilities (prototype screenshot, 28 October 2026)

Table 5 - Displayed distances for the six visible facility cards

Facility type	Straight-line (km)	Road shown (km)	Ratio
Clinic	1.31	1.68	1.282
Pharmacy	1.51	1.93	1.278
Hospital	2.40	3.07	1.279
Pharmacy	2.70	3.46	1.281
Ambulance	2.86	3.66	1.280
Fire and rescue	3.48	4.45	1.279

6. Discussion

The prototype shows that incident ingestion, route alternatives, facility discovery and live alerts can be assembled in one web application from open components, and the passing automated tests show that the implemented behaviour is consistent with its own specification. They do not show that the output is correct in the real world. The scoring is transparent but uncalibrated, and the structure of Eq. (3) means that terms built from proxies (lighting by route rank, traffic by travel time, a constant crowd term) can influence the ranking without reflecting measured conditions. The interface uses words such as verified and real-time; because the verified flag follows the publisher and the pipeline reads news feeds hourly, this paper uses the terms news-derived and near-real-time. A route labelled safest is only the best-ranked among those returned, and the absence of recorded incidents does not show that an area is safe.

7. Limitations and Threats to Validity

News data. News reports are incomplete, delayed, duplicated and unevenly distributed across places; classification and location errors are possible and unmeasured. Only places in the gazetteer can produce incidents, which biases coverage towards listed cities.

Location and time. Most incident coordinates are city-centre points, so a route buffer of a few hundred metres is not meaningful for them; the incident time is the article's publication time, not the time of the crime. Distances to a route are measured to polyline vertices rather than segments.

Score design. The score constants, buffer and weights are hand-set and unvalidated, and the buffer differs between the configuration (250 m) and the engine default (300 m). In the code inspected, the historical-crime term reads a date field that the data model does not use, so all records receive the default age.

Proxy and demo inputs. Lighting, traffic and crowd terms are proxies, the lighting and crowd data sets are demo data, the traffic provider is a mock, and the provenance of the historical records needs to be documented.

Emergency data. Contact numbers and 24-hour labels come from hand-entered or category-based rules and need manual verification; "road" distances are estimates.

Routing. Routing depends on public OSRM servers and open map data of varying completeness; live transit is not supported, and some alternatives are detours by construction.

Evidence. Evidence is limited to code inspection, one automated test run, and one session on a local instance; there is no accuracy test, independent evaluation or user study, and the literature review covers five sources.

Security and privacy. Wildcard cross-origin access with credentials, default demo accounts, a built-in fallback JWT secret and stored origin–destination histories require correction and a data-protection policy before any real use.

SafePath AI provides decision support only and cannot guarantee personal safety.

8. Future Scope

Planned work, not yet implemented, includes street-level incident geocoding, density-based or biobjective risk modelling (Chainey et al., 2008; Galbrun et al., 2016), calibration of the score against held-out incident data, use of real lighting and traffic sources, routed facility distances, verified emergency contact data, working push notification and SOS features, privacy-conscious handling of location data, multilingual support, and evaluation in several cities with real users.

9. Conclusion

SafePath AI is a working prototype that brings news-derived incident data, road-network route alternatives, emergency-facility discovery and live alerts into one web interface. On the code, test run and screenshots inspected, its route score is a transparent rule-based heuristic, several of its inputs are proxies or sample data, and some components still need verification or completion. No performance or effectiveness claim is made. Calibrated risk modelling, documented data provenance and a systematic evaluation are needed before the system is relied on for real travel decisions.

Acknowledgements

The authors express their sincere gratitude to E. Subramanian, Assistant Professor, Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, Coimbatore, Tamil Nadu, India, for his valuable guidance, encouragement, and support throughout the development of the project, "SafePath AI – AI-Powered Safety Navigation System."

References

Barrington-Leigh, C., & Millard-Ball, A. (2017). The world's user-generated road map is more than 80% complete. *PLoS ONE, 12*(8), e0180698. https://doi.org/10.1371/journal.pone.0180698

Chainey, S., Tompson, L., & Uhlig, S. (2008). The utility of hotspot mapping for predicting spatial patterns of crime. *Security Journal, 21*(1–2), 4–28. https://doi.org/10.1057/palgrave.sj.8350066

Galbrun, E., Pelechrinis, K., & Terzi, E. (2016). Urban navigation beyond shortest route: The case of safe paths. *Information Systems, 57*, 160–171. https://doi.org/10.1016/j.is.2015.10.005

Luxen, D., & Vetter, C. (2011). Real-time routing with OpenStreetMap data. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (pp. 513–516). ACM. https://doi.org/10.1145/2093973.2094062

Shah, S., Bao, F., Lu, C.-T., & Chen, I.-R. (2011). CrowdSafe: Crowd sourcing of crime incidents and safe routing on mobile devices. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (pp. 521–524). ACM. https://doi.org/10.1145/2093973.2094064