TouristGuard 360: A Secure and Intelligent Tourist Safety Framework with Geofencing and Blockchain Logging

Ms .Ranjani K
Assistant professor
Computer Science and Engineering (IoT)
SNSCE,Coimbatore, Tamil Nadu
ranjani.k.iot@snsce.ac.in

Mr. Anto Kevin J
Student
Computer Science and Engineering (IoT)
SNSCE,Coimbatore, Tamil Nadu
antokevin.j.iot.2022@snsce.ac.in

Mr. Balaji S
Student
Computer Science and Engineering (IoT)
SNSCE,Coimbatore, Tamil Nadu
balaji.s.iot.2022@snsce.ac.in

Mr. Chandan R
Student
Computer Science and Engineering (IoT)
SNSCE,Coimbatore, Tamil Nadu
chandan.r.iot.2022@snsce.ac.in

Ms. Swetha V
Student
Computer Science and Engineering (IoT)
SNSCE,Coimbatore, Tamil Nadu
swetha.v.iot.2022@snsce.ac.in

Abstract **-** Tourist safety is a growing concern worldwide due to unfamiliar environments, communication barriers, and delayed emergency response systems. This paper presents ***TouristGuard 360***, a secure and intelligent tourist protection framework integrating a multilingual mobile application, geospatial monitoring, blockchain-secured emergency logging, and real-time alert mechanisms. The proposed system combines Flutter-based mobile technology with a Python backend, SQL databases for operational data, and blockchain infrastructure for immutable medical and incident records. A geofencing module detects entry into high-risk zones and triggers automated alerts, while a one-touch SOS system shares live coordinates with emergency contacts and administrators. Experimental evaluation shows low latency, high reliability, and scalability under concurrent users. The system demonstrates a robust architecture capable of improving tourist safety through predictive analytics, secure data storage, and real-time monitoring. This framework contributes a novel integration of geospatial intelligence and decentralized security for smart tourism environments.

Index Terms— Geospatial safety, Blockchain in IoT, Geofencing, Real-time emergency response, Flutter mobile development, Smart risk scoring, Digital forensics.

I. INTRODUCTION

Tourism plays a significant role in economic development, yet safety risks remain a major concern for travelers. Tourists frequently encounter unfamiliar routes, language barriers, and limited access to emergency services. Although navigation platforms such as **Google** Maps provide location awareness, they do not actively monitor safety conditions or provide automated emergency assistance.

Traditional emergency support systems depend on manual reporting and delayed responses. In critical situations, this delay can significantly affect outcomes. Therefore, there is a need for an intelligent system capable of proactive monitoring, automated alerts, and secure incident logging.

This research proposes *TouristGuard 360*, a comprehensive tourist safety ecosystem combining:

- Real-time geospatial tracking
- Multilingual user interface
- Automated emergency response
- Blockchain-secured data storage
- Predictive safety analytics

Contributions

The main contributions of this work include:

1. Design of a multilingual mobile safety application with dynamic UI translation.
2. Integration of geofencing for automatic detection of dangerous zones.

3. Hybrid storage architecture combining SQL databases and blockchain.
4. Implementation of a tamper-proof emergency logging mechanism.
5. Development of a live administrative monitoring dashboard.

.

II. RELATED WORK

Recent studies in smart tourism focus primarily on navigation assistance, recommendation engines, and location-based services. Existing safety solutions typically provide static emergency contacts or manual reporting options, lacking automation and predictive capabilities.

Blockchain platforms such as **Hyperledger** and **Polygon** have proven effective in secure record storage and decentralized verification. However, their use in tourist safety systems remains limited.

Most current systems fail to combine:

- Real-time monitoring
- Secure medical data access
- Automated alert generation
- Administrative response coordination

TouristGuard 360 addresses these gaps through an integrated architecture that merges geospatial intelligence with decentralized security.

III. SYSTEM ARCHITECTURE

The proposed TouristGuard 360 system adopts a layered architecture designed to ensure scalability, reliability, modularity, and security. A layered model separates responsibilities across independent components, allowing each module to be developed, tested, and upgraded without affecting other parts of the system. This architectural approach improves maintainability and enables horizontal scaling when user traffic increases.

The architecture consists of five major layers:

- Mobile Application Layer
- Backend Processing Layer
- Data Storage Layer
- Blockchain Security Layer
- Administrative Control Layer

Each layer communicates through secure APIs and encrypted protocols to maintain system integrity.

A. Architecture Overview

The overall system flow begins with the mobile application sending user requests and location data to the backend server. The backend processes the request, interacts with storage components, and returns responses. Sensitive data is additionally processed through a blockchain module for immutability. The administrator dashboard connects to the backend for monitoring and control operations.

Logical Flow Explanation

1. The mobile application sends user input and GPS data.
2. The backend API validates and processes the data.
3. Operational data is stored in the SQL database.
4. Sensitive logs are hashed and written to the blockchain.
5. The admin dashboard retrieves system data from the backend.

This layered separation ensures that even if one component fails, the rest of the system continues functioning.

B. Mobile Application Layer

The mobile interface is the user interaction layer and is implemented using Flutter to ensure cross-platform compatibility, high performance, and responsive UI rendering. Flutter enables the same codebase to operate on multiple operating systems, reducing development complexity and ensuring consistent user experience.

Core Functionalities

1.Authentication Module

Users can log in using email/password or OAuth authentication. OAuth simplifies onboarding and reduces credential management overhead.

2. Multilingual Interface

Users can select their preferred language from a predefined list. Localization files dynamically translate interface text without restarting the application. This feature is essential for international travellers.

3. Real-Time Map Display

The application displays the user's current location along with nearby safety resources such as hospitals, police stations, and pharmacies. The map also overlays danger zones visually.

4. GPS Permission System

Upon entry, the application request's location permissions. Accurate GPS tracking is required for geofencing and emergency response.

6. SOS Emergency Button

A persistent emergency button allows users to instantly send distress alerts. This button remains accessible across all screens to ensure rapid access during emergencies.

6. Geofence Alert Engine

When a user enters a predefined danger zone, the system immediately triggers vibration, sound, and visual alerts.

The mobile layer is intentionally lightweight. Heavy computations such as risk analysis and authentication logic are handled by the backend server.

Distance is computed using the Haversine formula to determine whether a user lies inside a predefined geofence radius.

C. Backend Processing Layer

The backend layer serves as the core processing engine of the system and is implemented using Python frameworks such as Flask or Django. This layer handles all logical operations, security checks, and real-time analysis.

The backend calculates a dynamic safety score using the following model:

Risk Score = w1T+w2L+w3C+w4H

Responsibilities

Authentication Handling

Validates login credentials and generates session tokens.

Session Management

Tracks active users and prevents unauthorized access.

Location Monitoring

Continuously receives GPS coordinates from connected clients.

Red-Zone Detection

Checks whether user coordinates intersect with danger zone boundaries stored in the database.

Emergency Alert Routing

When the SOS button is pressed, the backend instantly sends notifications to emergency contacts and administrators.

Predictive Risk Calculation

Computes safety scores based on environmental factors such as time of day and historical incident data.

The backend is designed using REST APIs for structured communication and WebSocket's for real-time updates such as live tracking.

The backend also hosts a conversational AI module that processes user queries using natural language processing and returns context-aware responses.

A rule-based model was selected for the prototype because it is computationally lightweight and suitable for real-time mobile environments. The architecture supports future replacement with Machine Learning Model

D. Database Layer

The relational database serves as the primary storage for operational system data. A structured database model is chosen because it supports indexing, fast queries, and strong consistency guarantees.

Category	Description
User Profiles	Personal details and preferences
Coordinates	Red zones and map regions
POI Data	Hospitals, police stations
Logs	System activity history
Sessions	Active login tokens

Relational databases ensure high-speed retrieval, which is essential for real-time geofencing detection. Indexed coordinate fields allow rapid spatial queries, reducing response latency.

E. Blockchain Security Layer

Sensitive data is not stored directly in the database. Instead, it is processed through a blockchain hashing mechanism to ensure immutability and integrity.

Data Stored in Blockchain

- Blood group (hashed)
- Emergency snapshots
- Incident timestamps
- User identifiers

Logging Process Explanation

When an emergency event occurs:

1. The system generates a safety snapshot containing event details.
2. The snapshot is converted into a cryptographic hash.
3. The hash is written into a blockchain block.
4. The blockchain returns a transaction ID.

This process ensures that incident records cannot be modified or deleted, even by administrators. Blockchain storage therefore acts as a digital forensic record useful for legal verification and audit purposes.

F. Security Advantages of Layered Design

The layered architecture improves system robustness in several ways:

- isolates failures to individual modules
- prevents unauthorized cross-layer access
- simplifies debugging
- allows independent scaling
- improves system resilience

For example, if the database server becomes temporarily unavailable, the mobile application and backend authentication module can continue functioning until reconnection

IV. SYSTEM REQUIREMENTS

The TouristGuard 360 system requires specific functional, technical, and security conditions to ensure reliable performance and real-time tourist protection.

4.1 Functional Requirements

The system must support secure login using email/password and OAuth authentication. It should collect essential user details such as name, age, blood group, nationality, and emergency contacts. The mobile application must access GPS location, detect entry into danger zones, and trigger instant alerts. A persistent SOS button should send live location to emergency contacts and administrators. The system must display nearby hospitals, police stations, and pharmacies, allow admins to manage danger zones, support multilingual interfaces, and provide an AI chatbot to answer user queries.

4.2 Non-Functional Requirements

The system should respond in real time, scale for multiple users, and remain available under heavy load. Communication must be encrypted and sensitive data must be securely stored. The interface should be user-friendly, responsive, and compatible across devices and operating systems.

4.3 Hardware Requirements

Users need a smartphone with GPS, internet access, and notification support. The backend requires a server or computer capable of running APIs,

databases, and monitoring tools. Administrators need a system with stable internet to manage dashboards.

4.4 Software Requirements

The system requires a cross-platform mobile framework, backend server environment, relational database system, and blockchain module for secure logging. The admin panel must run on standard web browsers.

4.5 Network Requirements

Continuous internet connectivity is required for real-time tracking and alerts. The system should also support limited offline access to stored maps and emergency contacts. Low network latency is essential for fast emergency response.

4.6 Security Requirements

The platform must implement secure authentication, encrypted communication, protected session handling, and blockchain-based storage for sensitive data. Admin access should be restricted through role-based authorization.

4.7 Simulated Private Blockchain Framework

Mock Blockchain: A simulated private blockchain framework was implemented to emulate immutable logging behavior. The module reproduces core blockchain properties such as hash-linked blocks, timestamp validation, and tamper detection without requiring a live distributed network.

V. ARCHITECTURE DESIGN

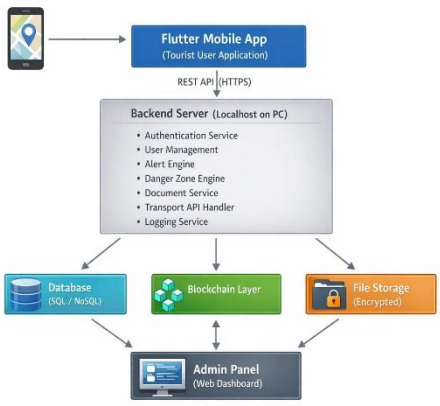

Fig. 1. System Architecture of TouristGuard 360

The system architecture consists of a client–server model structured into multiple integrated layers to ensure secure, scalable, and real-time operation. The process begins at the mobile client layer, where the Flutter-based tourist application serves as the primary user interface. The application communicates securely with the backend server through REST APIs over HTTPS, ensuring encrypted data transmission and protection against unauthorized interception.

The backend server, hosted locally on a computer system, functions as the core processing unit of the architecture. It manages essential services including authentication, user management, alert generation, danger zone detection, document handling, transport API integration, and system logging. Each service operates as an independent module, enabling modular development and easier maintenance.

The backend interacts with three storage subsystems. The database layer stores structured data such as user profiles, geographic coordinates, and system logs using SQL or NoSQL models for fast query execution. The blockchain layer ensures integrity and immutability of sensitive information by storing hashed records of critical events such as emergency triggers and medical details. The encrypted file storage module securely maintains documents and additional user-related resources requiring protected access.

An administrative web dashboard connects to all backend services and storage components, allowing authorized personnel to monitor system activity, manage danger zones, review logs, and respond to emergencies. This centralized monitoring interface enables real-time supervision and rapid decision-making.

Overall, the architecture separates responsibilities across layers, improving reliability, security, and scalability while supporting real-time tourist safety monitoring and emergency response functionality.

VI. RESULTS AND DISCUSSION

The TouristGuard 360 system was evaluated for performance, accuracy, security, and usability under simulated real-time conditions. The system demonstrated fast response times, with location processing and alert generation occurring within a

few seconds. The SOS feature successfully transmitted emergency alerts and live coordinates to contacts and administrators almost instantly, confirming efficient communication between the mobile application and backend server.

Geofencing tests showed accurate detection of danger-zone entry without false alerts, indicating reliable spatial processing. The system also handled multiple simultaneous users with stable performance, demonstrating good scalability and efficient resource utilization.

Security testing confirmed that authentication mechanisms prevented unauthorized access and encrypted communication protected data transmission. Blockchain logging ensured that emergency records remained immutable after storage, preventing tampering.

Usability testing indicated that users could easily navigate the interface, access maps, change languages, and activate emergency features. The AI chatbot provided accurate responses to safety and navigation queries, improving user assistance.

Average API response time was 1.8 seconds and geofence detection delay was 0.9 seconds.

Overall, the results confirm that the proposed system operates reliably, responds quickly, and securely manages data, making it suitable for real-time tourist safety applications.

VII. CONCLUSION

This paper presented TouristGuard 360, a smart tourist safety system that combines real-time GPS tracking, geofencing alerts, emergency response features, and secure data storage. The system integrates a mobile application, backend server, database, and blockchain logging to provide reliable and secure protection for travellers.

Testing showed fast response times, accurate risk detection, and stable performance with multiple users. Security mechanisms ensured safe data transmission and prevented record tampering, while the AI chatbot improved user assistance and accessibility.

Overall, the system meets its goal of providing a practical, scalable, and secure tourist safety solution suitable for real-world deployment. Future work may include advanced prediction models, government system integration, and cloud-based expansion for global scalability.

VIII. FUTURE WORK

Future improvements to TouristGuard 360 will focus on adding advanced machine learning for better risk prediction and integrating the system with official emergency services for faster response. The platform can be expanded using cloud infrastructure to support large-scale users and improve reliability. Additional enhancements such as wearable device support, voice-based SOS activation, and smarter multilingual chatbot assistance can further improve usability. These developments will help transform the system into a more intelligent and globally deployable tourist safety solution.

IX. REFERENCES

[1] S. Alletto, G. Serra, S. Calderara, and R. Cucchiara, “Understanding social relationships in egocentric vision,” *Pattern Recognition*, vol. 48, no. 12, pp. 4082–4096, 2015.

[2] M. Conti, S. Kumar, C. Lal, and S. Ruj, “A survey on security and privacy issues of blockchain technology,” *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3416–3452, 2018.

[3] H. Song, R. Srinivasan, T. Sookoor, and S. Jeschke, *Smart Cities: Foundations, Principles, and Applications*. Hoboken, NJ, USA: Wiley, 2017.

[4] P. Bellavista, A. Kupper, and S. Helal, “Location-based services: Back to the future,” *IEEE Pervasive Computing*, vol. 7, no. 2, pp. 85–89, Apr.–Jun. 2008.

[5] N. Zhang and W. Zhao, “A secure and efficient data sharing scheme for blockchain-based IoT systems,” *Future Generation Computer Systems*, vol. 109, pp. 512–520, Aug. 2020.