International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 14, Issue 1, 2026

Project Sahayak

G Dayakar Reddy[1],P Rishitha Reddy[2],Ch Roshini[3],K Shresta[4]

[1]Associate Professor & Vice Principal; Department Of Computer Science And Engineering Bhoj Reddy Engineering College For Women Hyderabad India

[2,3,4]B.Tech Students; Department Of Computer Science And Engineering Bhoj Reddy Engineering College For Women Hyderabad India

Mail Id; dayakar_reddy7@yahoo.co.in[1],rishithaporeddy1404@gmail.com[2], chava.roshini01@gmail.com[3], shresta.kulkarni8@gmail.com[4]

Abstract

Ensuring the safety of travelers in unfamiliar environments has become an important challenge in modern tourism. This paper presents a ***Tourist Security Application****, a mobile-based system designed to enhance traveler safety through integrated trip management, real-time monitoring, emergency response, and location-aware recommendations. The proposed system allows users to plan structured day-wise itineraries where destinations are automatically mapped using open-source geolocation services. During travel, the application continuously monitors user location and visually represents movement through an interactive mapping interface.An administrative dashboard is implemented to support centralized supervision, including user account management, trip activity monitoring, and emergency alert handling. The system architecture utilizes* ***Flutter*** *for cross-platform mobile development, while the backend is developed using* ***Flask*** *integrated with* ***MongoDB*** *for scalable data management. Open-source geospatial technologies such as* ***OpenStreetMap*** *and* ***Nominatim*** *are employed for location mapping and geocoding services.*

Keywords*: Tourist safety, mobile travel application, emergency response system, real-time location tracking, trip management, location-based recommendations, geospatial mapping, OpenStreetMap, Nominatim geocoding, Flutter mobile development, Flask backend framework, MongoDB database, secure authentication, travel assistance technology, scalable tourism security system.*

Introduction

Tourism has become a significant part of modern society, enabling individuals to explore new locations for leisure, cultural exchange, and business purposes. However, traveling to unfamiliar environments often exposes individuals to various risks, including accidents, health emergencies, navigation difficulties, and communication barriers. Ensuring the safety of travelers has therefore become an important concern in the digital tourism ecosystem. With the rapid growth of mobile technologies and location-based services, there is an increasing opportunity to develop intelligent systems that can assist travelers while also enhancing their safety.To address these challenges, this research proposes a **Tourist Security Application**, a mobile-based platform designed to enhance traveler safety through integrated technological solutions. The application supports itinerary planning, real-time location tracking, emergency alert mechanisms, and location-aware recommendations within a unified system. By leveraging modern mobile frameworks and open-source geospatial technologies, the system aims to provide travelers with reliable assistance while maintaining system scalability and data security.The **Tourist Security Application** is designed to address these limitations by providing a comprehensive mobile platform that integrates travel planning with safety support mechanisms. The system enables users to create structured travel itineraries and manage their trips through a centralized interface. During the journey, the application continuously monitors the user's location and displays travel progress using an interactive map interface.A key feature of the system is the **SOS emergency alert mechanism**, which allows users to instantly send emergency notifications along with their real-time location to predefined contacts and essential emergency services. This functionality helps ensure quick assistance during critical situations. Additionally, the application offers **location-based recommendations** for nearby food services, accommodation facilities, and transportation options to improve travel convenience.The system is implemented using **Flutter** for cross-platform mobile development, while the backend is built using **Flask** integrated with **MongoDB** for scalable data management. Open-source mapping tools such as **OpenStreetMap** and **Nominatim** are used to support geolocation services. An administrative dashboard enables system monitoring and management, while secure authentication mechanisms help maintain data privacy and system reliability.

Existing System

In the current travel technology landscape, users typically rely on multiple independent applications to manage different aspects of their journeys, such as navigation, accommodation booking, trip scheduling, and communication. These systems operate in isolation and do not provide a unified

platform that integrates travel management with safety features.Most available applications primarily focus on convenience-related services and lack mechanisms for real-time safety monitoring or automated emergency response. For example, many trip planning platforms allow users to create travel schedules but do not offer live updates or geolocation-based tracking during the trip. Similarly, navigation applications assist users in finding routes but do not provide integrated emergency alert functionalities.In emergency situations, travelers usually have to manually search for local emergency services or communicate with contacts, which may delay assistance. Additionally, the absence of a centralized monitoring system prevents authorities or administrators from effectively tracking user activity or responding to emergency alerts.

Proposed System

The proposed **Tourist Security Application** introduces a safety-focused travel management platform that integrates trip planning, real-time tracking, emergency response mechanisms, location-based recommendations, and administrative supervision within a unified system.The platform allows users to register and obtain a **unique blockchain-based identifier**, which serves as a secure and globally unique digital identity within the system. This identifier enhances system reliability by ensuring tamper resistance and secure association of user data, including travel plans, emergency alerts, and administrative records.An **administrative dashboard** provides centralized control for system administrators, enabling them to monitor registered users, track travel activities, and review SOS alert history. This centralized monitoring capability improves transparency and allows faster intervention during critical situations.

Requirement Analysis

Requirement analysis is an important phase in software development that identifies the expected functionality, system constraints, and operational requirements of the proposed application. It provides a clear understanding of what the system should accomplish and defines the resources needed for implementation. In the case of the Tourist Security Application, requirement analysis focuses on defining functional capabilities, performance expectations, computational resources, and the development methodology required to build a reliable and secure travel assistance platform.

Functional Requirements

Functional requirements describe the services and operations that the system must perform to satisfy user needs. The proposed Tourist Security Application is structured into three major components: the User Module, the System Module, and the Admin Module. Each module contributes to the overall functionality of the application and ensures efficient coordination between users, system processes, and administrative control.The User Module provides all functionalities that are directly accessible to the traveler using the application. This module allows users to register and create personal accounts, authenticate securely, and manage their profile information. During the registration process, each user is assigned a blockchain-based unique identifier to ensure secure identity management and reliable association of data within the system. Users can maintain a list of emergency contacts and manage authentication processes such as login and logout. The module also supports trip management features, enabling users to create, modify, view, or delete travel itineraries. Travelers can track their ongoing trips, maintain a history of completed journeys, and mark trips as finished when necessary. Additionally, the application integrates map-based services to display real-time location information and visually track travel progress. Based on the user's current geographic location, the system provides contextual recommendations for nearby food services, accommodation options, and transportation facilities. In emergency situations, users can activate the SOS alert feature, which immediately transmits location-based notifications to predefined contacts and emergency services.The Admin Module provides centralized monitoring and management capabilities for system administrators. This module includes secure administrative authentication and access to an administrative dashboard that displays system activity and user information. Administrators can review details of registered users, monitor travel records, and examine the history of SOS alerts generated within the application. The dashboard also allows administrators to update the status of emergency alerts and ensure that appropriate actions are taken when incidents occur. Through this module, administrators can maintain system transparency, monitor operational performance, and support effective emergency response.

Non-Functional Requirements

Non-functional requirements define the quality attributes of the system, including performance, security, reliability, usability, and maintainability. These characteristics ensure that the application operates efficiently and provides a reliable user experience.The system must deliver high performance and scalability by maintaining quick response times and smooth rendering of map-based interfaces, even when handling a growing number of users, trips, and emergency alerts. Efficient database management and optimized communication between the mobile application and backend services are essential to maintain system performance.Usability and compatibility are equally important to guarantee accessibility for a wide range of users. The application should provide an intuitive and user-friendly interface with clear instructions

and meaningful error messages. It should also be compatible with multiple Android devices and support cross-platform integration through modern mobile development frameworks.Maintainability and extensibility ensure that the system can evolve over time. The application architecture should follow a modular design and include proper documentation so that developers can easily maintain, update, and extend the system. This design approach will support future enhancements such as artificial intelligence–based risk prediction, multilingual support, and integration with external tourism services.

Computational Resource Requirements

Computational resource requirements specify the software and hardware environment needed for developing and running the proposed system. These requirements help establish the technical infrastructure necessary for implementing the application efficiently.

Software Requirements

The development of the Tourist Security Application requires a stable software environment that supports both mobile application development and backend processing. The system can be developed on a Windows operating system, preferably Windows 7 or later versions. The backend services are implemented using Python 3.x along with the Flask framework, which provides lightweight and flexible server-side functionality. The mobile application is developed using the Flutter framework, enabling cross-platform development and efficient user interface design. MongoDB is used as the database system for storing user information, trip data, and emergency alerts due to its scalability and flexible document-based structure.

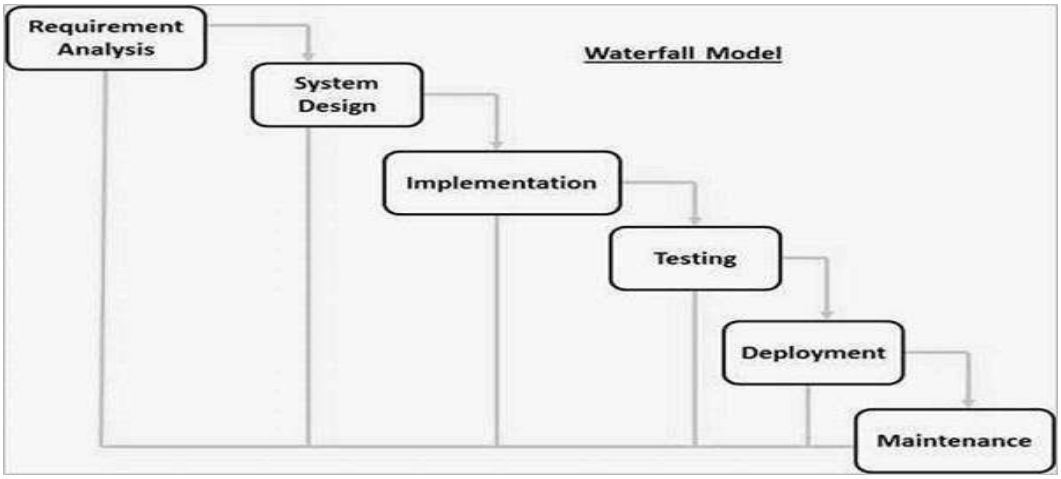


Fig 1 Waterfall Model

Hardware Requirements

Hardware requirements define the physical computing resources necessary for development, testing, and deployment of the application. A system with an Intel Core i3 processor or higher is recommended to ensure adequate processing capability. At least 8 GB of RAM is required to support application development tools, database operations, and testing environments. Additionally, a storage capacity of 256 GB or more is recommended to accommodate development files, system libraries, and database storage.

Software Development Life Cycle Model

The development of the Tourist Security Application follows the Waterfall Model, which is a sequential approach to software development. In this model, each stage of the project is completed before moving on to the next phase, allowing clear documentation and structured progress throughout the development process.In the implementation phase, the application is developed using Flutter for the frontend interface and Flask with MongoDB for backend services. Individual modules such as user management, trip planning, map integration, and emergency alert processing are implemented as separate components to maintain modularity.Following implementation, integration and testing are conducted to ensure that all modules work correctly together. Unit testing is performed to verify the functionality of individual components, while integration testing ensures proper communication between system modules. System testing evaluates the performance of real-time tracking and emergency alert features.

System Architecture

Architecture

System architecture defines the overall structure of the proposed application and explains how different components interact to process user requests. It provides a structured representation of the system by identifying the main modules, their responsibilities, and the sequence in which operations occur. A well-defined architecture enables developers to understand how different technologies and subsystems collaborate to deliver the required

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 1, 2026

functionality. In the Tourist Security Application, the architecture organizes the interaction between the mobile interface, backend services, database storage, and external mapping services. The architectural design supports efficient request handling, data processing, and communication between modules.The project architecture can be broadly categorized into two perspectives: **software architecture** and **technical architecture**. Software architecture focuses on the logical organization of the system, including application modules, user interactions, and functional workflows. Technical architecture describes the technological infrastructure used to implement the system, including programming frameworks, communication protocols, and database management systems.

Software Architecture

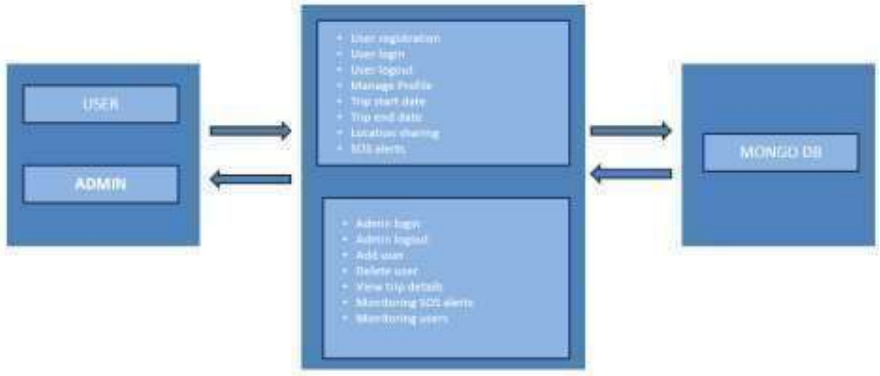


Fig 2 Software Architecture

The software architecture represents the logical arrangement of system modules and their interactions. It illustrates how the mobile application communicates with backend services to process requests such as user authentication, trip creation, location tracking, and SOS alert handling. The architecture follows a layered approach where the mobile interface handles user interactions, the backend processes application logic, and the database stores user information, trip data, and emergency alerts. External geolocation services such as OpenStreetMap and Nominatim support map visualization and location-based services. This layered architecture ensures modular development and simplifies maintenance and future system upgrades.

Technical Architecture

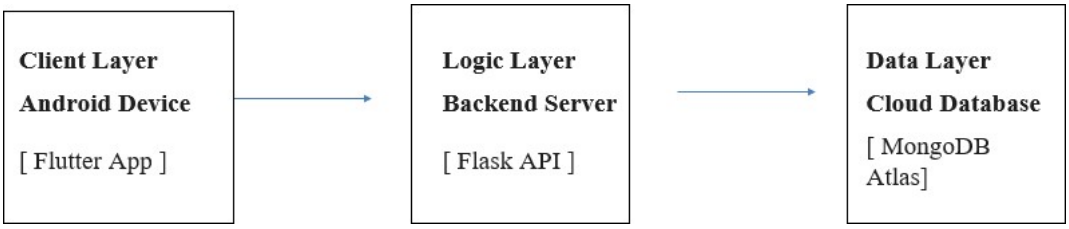


Fig 3 Technical Architecture

The technical architecture describes the technological components used to implement the system and the communication mechanisms between them. The mobile application is developed using Flutter, which enables cross-platform interface development and smooth user interactions. The backend server is implemented using the Flask framework in Python, which handles API requests, authentication processes, and business logic execution. MongoDB is used as the primary database due to its flexible document-based data model and scalability for handling dynamic datasets. Communication between the mobile application and backend server is achieved through RESTful APIs, enabling secure and efficient data exchange. Additionally, external geospatial services are integrated to support real-time location tracking and map visualization.

Implementation

The Tourist Security Application is implemented using modern web and mobile development technologies to ensure efficiency, scalability, and cross-platform compatibility. The frontend mobile application is developed using Flutter, which provides a responsive user interface and seamless performance across devices. The backend server is

built using the Flask framework in Python, which manages application logic, API communication, and authentication processes. MongoDB is used as the database system to store user data, trip information, and emergency alerts due to its flexibility and scalability. Real-time location tracking and map visualization are supported through the integration of OpenStreetMap and Nominatim geolocation services. Communication between the frontend and backend components is implemented through RESTful APIs, ensuring structured data exchange and secure connectivity between system modules.

Implementation Process

The implementation of the system follows a modular approach in which different functional layers operate together to deliver the complete application. Initially, the user authentication module allows travelers to securely register and access the system. User profile data and emergency contacts are stored within the database to ensure quick retrieval and updates.The trip management module allows users to create and manage travel itineraries, while the location tracking module continuously monitors the user's geographic position using GPS services. Map integration enables visual representation of the user's location and nearby points of interest, improving navigation and travel assistance.During emergency situations, the SOS activation feature enables users to generate instant alerts. When triggered, the system collects the user's current location, generates an alert message containing relevant details, and forwards notifications to emergency contacts and services. The alert information is stored in the database for monitoring purposes, while administrators can track such events through the administrative dashboard to ensure timely response and system oversight.

SOS Alert Algorithm

The emergency alert process follows a simple algorithm to ensure rapid response during critical situations. The process begins when the user activates the SOS button within the application. The system immediately captures the user's current GPS coordinates, including latitude and longitude values. After retrieving the emergency contact details stored in the database, the system generates an alert message containing the user's identification information and location details. This message is then transmitted to the registered emergency contacts or services. Simultaneously, the alert record is stored in the database and displayed in the administrative dashboard for monitoring purposes. The process concludes once notifications have been successfully delivered.

Testing

Software testing is a critical stage in the software development lifecycle that evaluates whether the developed system meets the specified requirements and functions as expected. The objective of testing is to identify potential defects and ensure that the final product operates reliably and efficiently. In modern digital environments, many everyday services depend heavily on software systems. Therefore, even minor software errors can lead to significant operational or financial consequences. For this reason, thorough testing is necessary to deliver a stable and high-quality application.

Testing contributes to several important aspects of software development, including cost reduction, improved product quality, enhanced security, and greater user satisfaction. By detecting and resolving issues early in the development process, organizations can minimize development costs and ensure reliable system performance.

Dimensions of Testing

Software testing can be examined from multiple perspectives, including the layer of the application being tested, the scale of testing performed, the type of testing applied, and the methodology used during testing. Testing may focus on different layers of the system such as the database layer, application programming interfaces (APIs), and user interface components. Additionally, testing can be conducted at various levels including unit testing, module testing, integration testing, and full system evaluation. Different testing techniques such as functional testing, performance testing, and security testing are used depending on the objectives of the testing process. Testing may also be performed through exploratory approaches, scripted manual procedures, or automated tools.

Stages of Testing

Unit Testing

Unit testing focuses on verifying the functionality of individual components or modules within the system. Each module is tested independently to ensure that it performs its intended function correctly. This form of testing is typically performed by developers during the early stages of development and often uses white-box testing techniques to examine internal logic and program structure.

Integration Testing

Integration testing evaluates how different modules interact when combined together. Even if individual modules function correctly in isolation, issues may arise when they communicate with each other. Integration testing helps identify interface defects and ensures that data flows correctly between components.

System Testing

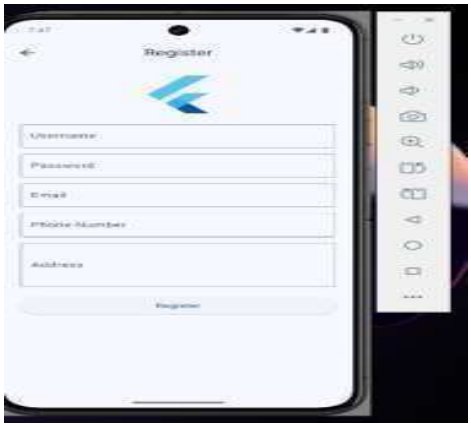
System testing evaluates the complete application as a unified system. This phase verifies that the entire system meets the functional and technical requirements specified during the design stage. Testing is usually conducted in an environment that closely resembles the actual production environment to ensure realistic evaluation.

Acceptance Testing

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 1, 2026

Acceptance testing is the final stage of testing where the system is evaluated from the user's perspective. The goal is to determine whether the application meets user expectations and business requirements. Once the system successfully passes this stage, it can be approved for deployment and real-world usage.

Screenshots

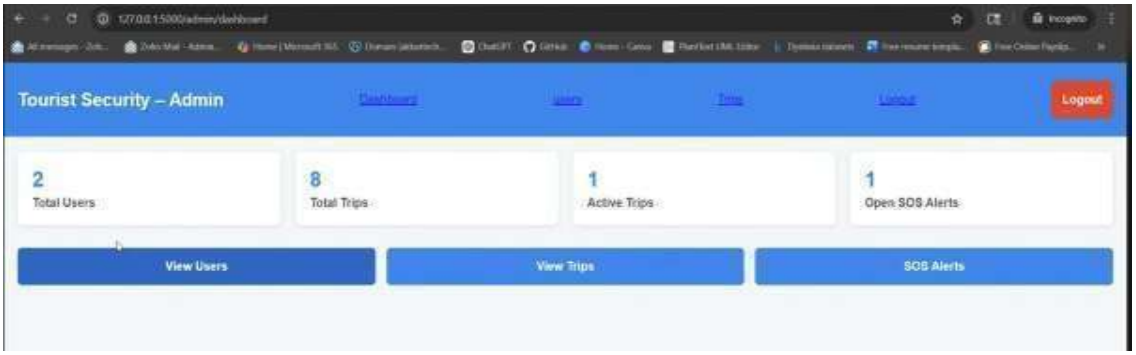


Screenshot 1: User Registration

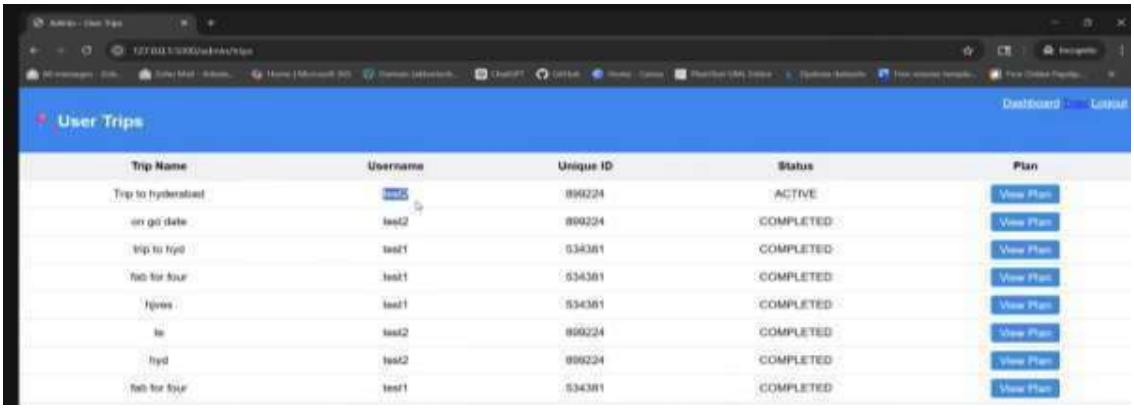
Screenshot 2 : Home Screen

Screenshot 3 : Selecting the dates of the trip

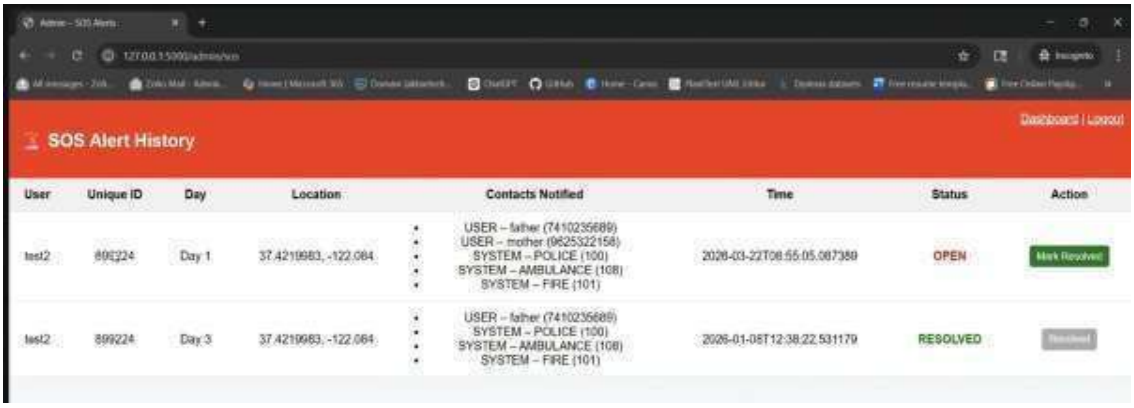
International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 1, 2026



Screenshot 5 : Admin dashboard



Screenshot 6 : User trips



Screenshot 7: SOS alert history

Types of Testing

Black Box Testing

Black box testing evaluates the functionality of the application without examining the internal code structure. Testers focus on inputs and outputs to determine whether the system behaves according to specifications. This approach can be applied to multiple testing levels, including unit, integration, system, and acceptance testing.

White Box Testing

White box testing involves analyzing the internal structure and logic of the program. Test cases are designed based on the internal implementation of the software, requiring knowledge of programming and system architecture. Common white-box testing techniques include statement coverage, branch coverage, and path coverage.

Conclusion

The proposed **Tourist Security Application** provides an integrated digital platform designed to enhance the safety and convenience of travelers. The system combines several essential travel-support features, including itinerary planning, real-time location tracking, emergency alert mechanisms, and administrative monitoring within a single mobile application. By allowing users to securely register and manage their travel plans, the platform supports

structured trip management while enabling travelers to monitor their journey through interactive map-based visualization.One of the key components of the application is the **SOS alert mechanism**, which enables users to quickly request assistance during emergency situations. When activated, the system automatically captures the user's current geographic location and sends notifications to predefined emergency contacts and relevant services. This feature significantly improves response time and ensures that accurate location information is available during critical events.In addition to emergency support, the application enhances the overall travel experience by providing **location-based recommendations** for nearby services such as food, accommodation, and transportation. These recommendations assist travelers in making informed decisions while navigating unfamiliar environments. The system also includes an **administrative dashboard** that allows authorized personnel to monitor user activities, track ongoing trips, and review emergency alerts, thereby enabling effective system supervision and faster intervention when necessary.

Future Scope

Although the current implementation provides a strong foundation for improving tourist safety, several enhancements can further extend the capabilities of the system. Future developments may include the integration of **blockchain technology** for securely storing user identities, trip records, and emergency alerts. Such an approach would improve data transparency, prevent unauthorized modifications, and strengthen trust in the system.Another potential enhancement involves the incorporation of **advanced risk detection mechanisms**. By integrating external data sources such as crime statistics, weather alerts, disaster warnings, and government safety advisories, the system could identify potentially unsafe areas and notify travelers in advance. This feature would provide proactive safety guidance and reduce exposure to hazardous environments.Artificial intelligence techniques can also be introduced to develop **predictive safety models**. Machine learning algorithms could analyze historical travel patterns, geographic risk indicators, and user behavior to generate safety scores or predict possible threats. This predictive capability would help travelers make safer decisions while planning and executing their trips.Through these future enhancements, the Tourist Security Application has the potential to evolve into a comprehensive intelligent travel safety system capable of supporting travelers worldwide.

References

[1] Mansfeld, Y., and Pizam, A., *Tourism, Security and Safety: From Theory to Practice*, Elsevier Butterworth-Heinemann, Oxford, 2019.

[2] Scott, N., Laws, E., and Prideaux, B., *Safety and Security in Tourism: Recovery Marketing after Crises*, Routledge, 2020.

[3] Hall, C. M., Timothy, D. J., and Duval, D. T., *Safety and Security in Tourism: Relationships, Management and Marketing*, Routledge, 2020.

[4] Page, S. J., *Tourism Management*, 6th Edition, Routledge, 2019.

[5] Wilks, J., Pendergast, D., and Leggat, P., *Tourism in Turbulent Times: Towards Safe Experiences for Visitors*, Elsevier, Oxford, 2022.

[6] Popescu, L., "Safety and Security in Tourism: Case Study – Romania," *Forum Geografic*, Vol. 10, No. 2, pp. 322–328, 2021.

[7] Toker, A., and Emir, O., "Safety and Security Research in Tourism: A Bibliometric Mapping," *European Journal of Tourism Research*, Vol. 34, 2023.

[8] World Tourism Organization (UNWTO) and World Health Organization (WHO), *Tourism Safety, Travel Advice, and Risk Management Guidelines*, Available: https://www.unwto.org and https://www.who.int.