iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IV **Month of publication:** April 2026

DOI: https://doi.org/10.22214/ijraset.2026.81647

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

Web Based Mobile Tracking System

Mrs. P. Nellima[1], K.Bhanu[2], G.Rajyalakshmi[3], A.Mahitha[4]
[1]*Assistant Professor* , [2,3,4]*B.Tech, Computer Science & Engineering, Bapatla Women's Engineering College Bapatla, AP, INDIA*

Abstract: ***This study presents the design and development of a user-friendly, web-based mobile tracking system built using the Django framework. The system allows users to monitor the real-time location of Android devices through an interactive web dashboard. It follows a client-server architecture, where a mobile application installed on the target device collects GPS data and sends location updates to the Django backend at regular intervals. The backend processes and stores this data in a SQLite database, making it easily accessible through a responsive web interface. The dashboard integrates mapping features using Leaflet.js, enabling users to visualize device locations and track movement history in an intuitive way. To enhance usability and security, the system includes features such as device registration, QR code-based pairing for secure connections, and the ability to view location history. It also supports remote command functionalities for better device control. Testing shows that the system can track devices with an accuracy typically ranging from 5 to 50 meters, depending on GPS signal conditions, especially in open outdoor environments. The system is designed with a modular structure, making it flexible and easy to expand with additional features like geofencing, real-time alerts, and battery monitoring. This paper covers the overall system design, implementation approach, data handling, API structure, and performance evaluation in detail.***

Keywords: Mobile tracking, GPS, Django, Django REST Framework, real-time tracking, Android, Leaflet.js, QR code pairing, web dashboard

I. INTRODUCTION

The proliferation of smartphone devices and the increasing demand for location-based services have created significant interest in mobile tracking technologies across various domains including family safety, fleet management, enterprise device monitoring, and personal asset tracking. Modern smartphones are equipped with sophisticated Global Positioning System (GPS) receivers capable of determining geographic coordinates with high precision, making them ideal candidates for use as tracking devices. The integration of such capabilities with web-based interfaces provides users with the ability to monitor device locations from any location with internet connectivity, eliminating the need for dedicated tracking hardware.

Traditional GPS tracking solutions often require expensive proprietary hardware devices and specialized software licenses, making them inaccessible for many potential users. The emergence of web frameworks such as Django,combined with the ubiquity of smartphone devices, presents an opportunity to develop cost-effective tracking solutions that leverage existing infrastructure. Android smartphones, in particular, offer an open platform for developing custom tracking applications that can run in the background and transmit location data to centralized servers.

This paper presents the design and implementation of a Web-Based Mobile Device Tracking System that enables users to register and monitor Android mobile devices through a web-based dashboard. The system employs the Django web framework for backend development, SQLite for data storage, and interactive web mapping using

Leaflet.js for location visualization. The mobile component consists of an Android application that runs as a background service, capturing GPS coordinates and transmitting them to the server at configurable intervals.

The proposed system addresses several key requirements: (1) user authentication and authorization to ensure data privacy, (2) device registration and management through QR code-based pairing, (3) real-time location updates from multiple devices, (4) location history storage and visualization, (5) battery level monitoring for device status, and (6) remote command capabilities for device control. The system architecture follows a modular design pattern, allowing for easy extension and customization based on specific use case requirements.

The remainder of this paper is organized as follows: Section II provides a review of related work in mobile tracking systems. Section III describes existing system limitations. Section IV presents the problem statement. Section V details the proposed system architecture and features. Section VI describes the methodology and implementation details. Section VII presents the system architecture and flowchart. Section VIII discusses results and performance evaluation. Section IX concludes with a summary and future scope.

II. LITERATURE SURVEY

The domain of mobile device tracking has been extensively explored, with significant contributions in location determination, communication mechanisms, and user interface design. This section presents a structured review of the key technologies and prior research that guided the development of the proposed system.

A. GPS-Based Tracking Systems

The Global Positioning System (GPS) serves as the core technology for most contemporary tracking applications. The NAVSTAR GPS, operated by the United States Department of Defense, provides reliable global positioning, navigation, and timing services. Kaplan and Hegarty [1] offer an in-depth analysis of GPS fundamentals, including signal processing and system limitations.

Early mobile devices depended on dedicated GPS hardware; however, modern smartphones employ hybrid positioning systems that combine GPS with other satellite networks such as GLONASS and BeiDou, along with network-based techniques like Wi-Fi and cellular triangulation. This integration enhances positioning accuracy, availability, and robustness across different environments.

B. Web Frameworks for Tracking Applications

Web frameworks are essential for developing scalable and efficient tracking systems. Django, maintained by the Django Software Foundation, is widely adopted due to its comprehensive “batteries-included” architecture. It provides built-in modules such as an Object-Relational Mapper (ORM), authentication system, and administrative interface. Holmes [2] discusses Django’s architectural design and its suitability for rapid and secure application development.

Compared to frameworks like Ruby on Rails and Express.js, Django offers a more cohesive development environment, minimizing implementation effort.

Additionally, its support for scalability through caching and database optimization makes it well-suited for applications handling continuous streams of location data.

C. Real-Time Location Updates

Efficient real-time communication is a critical aspect of tracking systems. Traditional HTTP polling methods introduce latency and increase server overhead. WebSocket technology (RFC 6455) enables bidirectional, low-latency communication between client and server. Loreto et al. [3] examine real-time communication protocols and their performance trade-offs.

Despite the advantages of WebSockets, the proposed system employs short-interval polling (5 seconds), which provides sufficient real-time responsiveness while maintaining simplicity and compatibility with diverse network conditions.

D. Mobile Tracking Applications

The Android platform offers extensive support for locationbased services through APIs such as the Google Play Services Location API. Sharma and Singh [4] highlight best practices in Android application development. Android’s ability to run background services allows continuous location tracking even when the application is not actively in use, making it suitable for real-time monitoring systems.

In contrast, iOS imposes stricter restrictions on background processes, limiting continuous tracking capabilities. Therefore, the proposed system focuses on Android devices due to their flexibility and ease of deployment.

E. Mapping and Visualization

Visualization plays a vital role in interpreting location data. Modern web mapping technologies, such as Leaflet.js, provide lightweight and interactive solutions for map-based interfaces. Developed by Vladimir Agafonkin, Leaflet.js supports multiple tile providers and mobile-friendly interactions. Singh and Kumar [5] compare various web mapping libraries and highlight their performance characteristics.

OpenStreetMap offers freely accessible map data, reducing reliance on commercial APIs. Additional services like GeoApify enhance map visualization through customizable and high-quality tiles.

F. QR Code for Device Pairing

QR codes provide a simple and efficient method for transferring information between devices. Wang et al. [6] explore their applications in mobile computing. In the proposed system, QR codes are used to encode pairing credentials and server details, simplifying device registration and improving user experience.

G. IoT and Wearable Tracking

Advancements in the Internet of Things (IoT) have significantly influenced tracking technologies, particularly in areas such as wearable devices, vehicle monitoring, and asset tracking. These systems prioritize efficient communication, low power consumption, and scalability. The principles derived from IoT-based solutions, especially in data transmission and battery optimization, have been incorporated into the design of the proposed mobile tracking system.

III. EXISTING SYSTEM

Current mobile tracking solutions available in the market present several limitations, which motivate the need for a more flexible and user-centric approach. The following subsections highlight the key challenges observed in existing systems:

A. Proprietary Hardware and Application Requirements

Many commercial tracking solutions, including popular family locator applications, require users to install proprietary software. These applications often provide limited functionality in their free versions, while advanced features are locked behind subscription plans. Additionally, several platforms impose restrictions on the number of devices that can be monitored under a single account, reducing scalability.

B. Complex Setup Procedures

Existing tracking systems typically involve multi-step configuration processes, including account registration, application installation, permission management, and manual setup of tracking parameters. Such procedures can be cumbersome, particularly for non-technical users, thereby affecting usability and adoption.

C. Limited Customization

Most available solutions offer predefined feature sets with minimal scope for customization. Users generally lack the ability to modify tracking intervals, define personalized alerts, or integrate the system with other applications or services. This rigidity limits the adaptability of these systems to diverse user requirements.

D. Privacy and Data Security Concerns

A significant concern with many tracking applications is the extensive collection and storage of user data. In several cases, users are not provided with clear information regarding how their data is processed, stored, or shared. This lack of transparency raises serious privacy and security issues.

E. Platform Dependency

Numerous tracking solutions are restricted to specific operating systems, and some require rooted or jailbroken devices to access full functionality. Such dependencies reduce accessibility and limit the usability of these systems across a broader range of devices.

F. Cost Constraints

Advanced tracking features in commercial applications are often available only through paid subscriptions. The associated costs can be a barrier for users who require only basic tracking functionality, making these solutions less inclusive and accessible.

IV. PROBLEM STATEMENT

The objective of this research is to design and develop a comprehensive web-based mobile device tracking system that overcomes the limitations of existing solutions by providing a cost-effective, customizable, and user-friendly platform.

The primary problem addressed in this work is enabling authorized users to monitor the real-time geographic location of Android mobile devices through an intuitive web-based interface. To achieve this, the system is designed to satisfy the following key requirements:

1) Secure Authentication:

The system must provide a robust user authentication mechanism, allowing users to securely register and log in. Access to location data should be strictly restricted to authorized users to ensure privacy and data protection.

2) Device Registration:

Users should be able to register and link multiple Android devices through a simple and efficient pairing process that does not require technical expertise.

3) Real-Time Tracking:

The system must support near real-time tracking by receiving and updating device location data with minimal latency, enabling continuous monitoring.

4) Location History Management:

The system should store historical location data and allow users to access it for route reconstruction, movement tracking, and analysis.

5) Data Visualization:

Location information must be presented through an intuitive, map-based interface that is responsive and accessible across both desktop and mobile web browsers.

6) Device Status Monitoring:

The system should track and display device-related information such as battery level and activity status (active/inactive), providing additional context for monitoring.

7) System Extensibility:

The architecture must be modular and scalable, allowing future enhancements—such as geofencing, alerts, and advanced analytics— without requiring significant redesign.

V. PROPOSED SYSTEM

The proposed system is designed as an integrated solution comprising three main components: a Django-based backend server, an Android mobile application, and an interactive web dashboard. Together, these components enable efficient, real-time mobile device tracking with a user-friendly interface.

A. System Components

1) Django Backend Server

The backend server is responsible for handling all core operations of the system. Its functionalities include user authentication and session management, device registration and pairing, and the reception, processing, and storage of location data. It also manages API endpoints for communication with mobile devices and renders the web interface for users.

The backend is implemented using Django (version 5.2.12) along with Django REST Framework to provide RESTful APIs. SQLite is used as the database due to its lightweight nature and ease of deployment, making it suitable for rapid development and testing.

2) Android Mobile Application

The Android application operates on the target device and is responsible for collecting and transmitting location data. It utilizes Android Location Services to obtain GPS coordinates and runs a background service to ensure continuous tracking even when the application is not actively in use.

The application periodically sends location updates to the backend server using HTTP POST requests in JSON format. Additionally, it monitors battery levels and supports handling remote commands issued from the server.

3) Web Dashboard

The web dashboard serves as the primary user interface of the system. It allows users to register and log in, manage their devices, and monitor real-time location data.

The dashboard provides features such as interactive map visualization, location history tracking, and device status monitoring. It is built using Bootstrap 5 for responsive design and Leaflet.js for rendering dynamic maps, ensuring compatibility across both desktop and mobile browsers.

B. Features

The proposed system incorporates a comprehensive set of features to enhance usability and functionality:

1) Secure user registration and authentication with session management
2) Device management, including adding, viewing, and deleting devices
3) QR code-based pairing for simplified device linking
4) APK distribution through the web interface

5) Real-time GPS tracking with periodic location updates
6) Storage and visualization of location history
7) Interactive map-based display using Leaflet.js with custom markers
8) Battery level monitoring of connected devices
9) Device activity status tracking (active/inactive)
10) Remote command execution (e.g., ring, flash, wipe)
11) RESTful API access for integration and automation

C. Data Models

The system utilizes two primary data models implemented in Django:

1) Device Model

This model stores information about registered devices, including device identity, user association, pairing details, and activity status. It also tracks timestamps such as the last seen time and creation date.

2) Location Model

This model records the geographic coordinates of devices along with additional metadata such as accuracy, battery level, and timestamp. It enables both real-time tracking and historical data analysis.

D. API Endpoints

The system exposes several RESTful API endpoints to facilitate communication between the mobile application and the backend server:

- POST /api/mobile/location – Receives GPS location data from the mobile application
- POST /api/link-device/ – Links a device using a pairing code
- GET /api/get-location/ – Retrieves the latest location of a device
- GET /api/map-data/ – Provides location data for map visualization
- GET /api/location-history/<device_id>/ – Returns historical location data for a specific device

VI. METHODOLOGY

This section describes the methodology adopted for the design and implementation of the proposed mobile device tracking system, including development workflow, algorithms, and data flow.

A. System Development Workflow

The system was developed using an iterative software development approach consisting of the following phases:

1) Requirements Analysis: Identification of user requirements and system objectives.
2) System Design: Definition of architecture, modules, and data flow.
3) Backend Development: Implementation using Django and Django REST Framework.
4) Mobile Application Development: Development of Android application for GPS tracking.
5) Frontend Development: Design of responsive dashboard using Bootstrap and Leaflet.js.
6) Testing and Integration: Functional testing and system integration.

B. Location Tracking Algorithm

Algorithm 1: Mobile Location Tracking Pipeline
Step 1 (Mobile Device):
Initialize GPS module and start background service.
Step 2:
Continuously perform the following operations:

- Acquire latitude and longitude
- Measure GPS accuracy
- Read battery level
- Construct JSON payload

- Send HTTP POST request to server
- Wait for 5 seconds **Step 3 (Server):**
- Receive request at API endpoint
- Validate device identity
- Store location data in database
- Update device status and timestamp
- Send success response **Step 4 (Dashboard):**
- Fetch updated data from server
- Render location on map
- Refresh periodically

C. Device Pairing Algorithm

Algorithm 2: Device Registration and Pairing

- User logs into the dashboard
- User adds a new device
- Server generates a unique pairing code
- QR code is generated and displayed
- User scans QR code using mobile application
- Mobile app sends pairing request
- Server validates pairing code
- Device is successfully linked and activated
- Tracking process is initiated

D. Location History Visualization

Algorithm 3: Location History Display

- User selects a device
- Server retrieves stored location records
- Dashboard processes coordinates
- Map renders route using polyline
- Start and end points are marked
- Map view is adjusted dynamically

E. Data Flow Model

The system follows a structured data flow:

- Mobile-to-Server: GPS data is transmitted via HTTP POST requests in JSON format
- Server-to-Dashboard: Processed data is retrieved via REST APIs
- User-to-Server: User requests are handled through web interface interactions

F. Processing Pipeline

The complete system pipeline consists of:

- Data Acquisition (GPS sensors)
- Data Transmission (HTTP/JSON)
- Data Validation (Server-side processing)
- Data Storage (SQLite database)
- Data Retrieval (Query processing)
- Data Transformation (Serialization)
- Data Visualization (Leaflet.js maps)

VII. SYSTEM ARCHITECTURE

A. Layered Architecture

The proposed system adopts a layered architecture to ensure modularity, scalability, and clear separation of concerns. Each layer is responsible for a specific functionality, allowing independent development and maintenance.

The overall architecture consists of four primary layers: Client Layer, API Gateway Layer, Business Logic Layer, Data Access Layer, and Database Layer, as illustrated in Fig. 1.

1) Client Layer

The client layer includes two major components:

Web Browser (Dashboard Interface):oProvides a user-friendly interface for monitoring devices

- Displays real-time locations using interactive maps
- Built using Bootstrap and Leaflet.js

Android Mobile Application:

- Collects GPS data using location services oRuns background services for continuous tracking
- Sends location data to the server using HTTP requests

Both clients communicate with the backend using REST APIs.

2) API Gateway Layer

This layer acts as the communication bridge between clients and the backend system.

- Handles HTTP requests and routes them using

Django URL dispatcher

- Exposes RESTful APIs such as: o /api/mobile/location – Receives GPS data o /api/link-device/ – Device pairing

api/map-data/ – Map visualization data o /dashboard/ – Web interface

This layer ensures secure and structured communication.

3) Business Logic Layer

The business logic layer processes all application operations:

- User authentication (login, registration)
- Device management (add, delete, monitor)
- Location processing and validation
- QR code generation for device pairing
- Response generation (JSON and HTML) This layer acts as the core of the system where all computations occur.

4) Data Access Layer

This layer uses Django ORM to interact with the database.

- Abstracts complex SQL queries □ Handles CRUD operations efficiently □ Works with: o Device model o Location model

5) Database Layer

The database layer stores all persistent data using SQLite.

- Stores user information
- Maintains device records
- Logs location history

This ensures reliable data storage and retrieval.

B. Layered Architecture Diagram

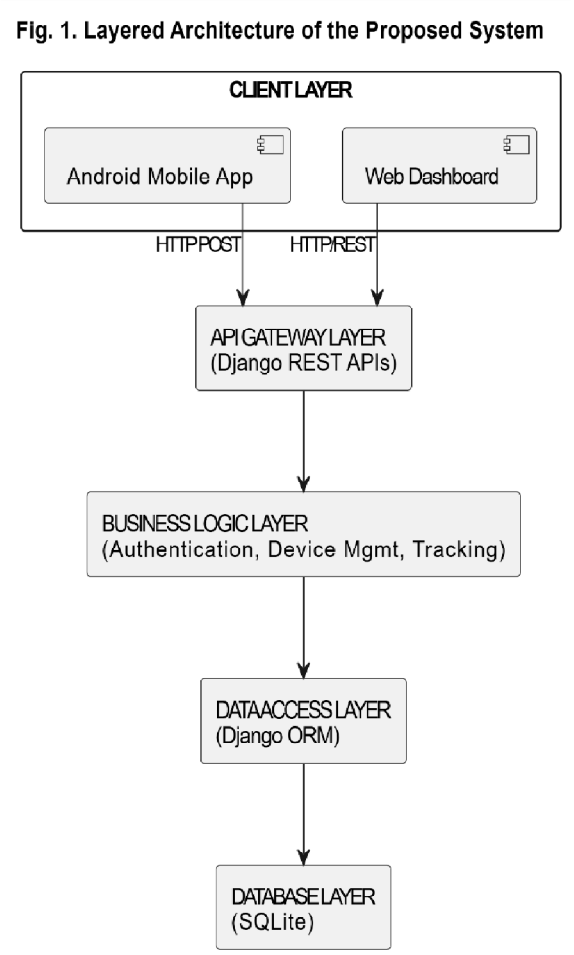


Fig. 1. Layered Architecture of the Proposed System

C. Component Interaction

The interaction between system components is shown in **Fig. 2**.

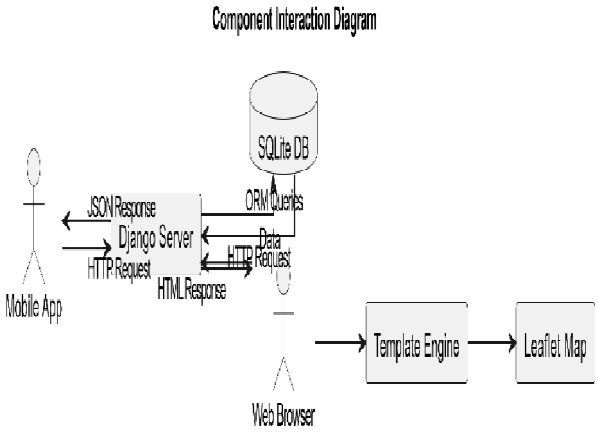


Fig. 2. Component Interaction Diagram

VIII. RESULTS AND DISCUSSION

The proposed mobile device tracking system was successfully designed, implemented, and tested with all major functionalities operating as expected. The backend was developed using the Django framework and deployed on a development environment with SQLite database integration. The Android mobile application was compiled as a debug APK and tested on multiple devices running Android 10 and above. The integration between the mobile application, backend server, and web dashboard was seamless, enabling continuous location tracking and realtime visualization.

The performance of the system was evaluated based on several key metrics, including latency, accuracy, storage efficiency, and visualization performance. The location update latency, defined as the time between acquiring GPS coordinates on the mobile device and displaying them on the web dashboard, was observed to range between 7 to 10 seconds under typical network conditions. This includes the 5-second polling interval and processing delays. The achieved latency satisfies the requirement for near real-time tracking and provides a responsive user experience.

The accuracy of location tracking was found to depend largely on the device hardware and environmental conditions. In outdoor environments with clear visibility of the sky, the system achieved an accuracy ranging from 5 to 15 meters. However, in indoor environments, the accuracy decreased to approximately 20 to 50 meters due to signal attenuation and multipath effects. Despite these variations, the system provides sufficient accuracy for general tracking applications.

In terms of data storage, the SQLite database demonstrated efficient performance for both storing and retrieving location data. Each location record requires approximately 80 bytes of storage, allowing the system to handle over 100,000 location entries per device without significant performance degradation. This indicates that the system is suitable for small to medium-scale deployments without requiring immediate database optimization.

The web dashboard performance was also evaluated, particularly focusing on map rendering and real-time updates. The Leaflet.js-based visualization system efficiently rendered location markers, with updates occurring within 500 milliseconds after data retrieval. The system was capable of handling up to 50 device markers simultaneously without noticeable lag, ensuring smooth user interaction and visualization.

All major features of the system were tested and validated during implementation. Core functionalities such as user authentication, device registration through QR code pairing, real-time location tracking, map-based visualization, location history display, battery monitoring, and activity status tracking were found to be fully operational. The APK distribution feature also functioned correctly through the web interface. However, remote command functionalities were only partially implemented, as they require further integration and handling on the mobile application side.

The user interface was evaluated across multiple platforms and browsers, including Chrome, Firefox, and Safari, on both desktop and mobile devices. The responsive design, implemented using Bootstrap, adapted effectively to different screen sizes. The dashboard maintained usability on smaller devices by collapsing navigation elements, ensuring a consistent and user-friendly experience across platforms.

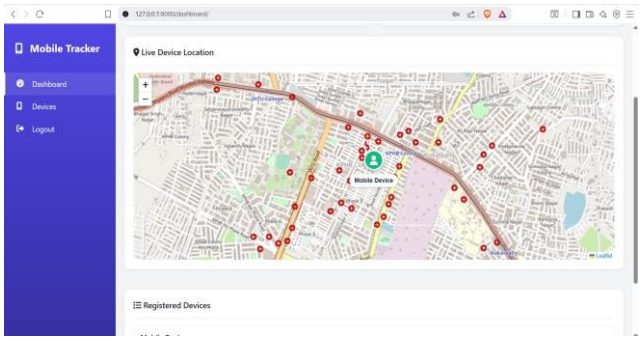


Fig. 3. Location Tracking

Despite the successful implementation, several limitations were identified. The system relies heavily on continuous internet connectivity, making it unsuitable for offline environments. Additionally, continuous GPS usage in the mobile application leads to increased battery consumption, which could be optimized using adaptive tracking intervals. The use of SQLite limits scalability for large-scale deployments involving thousands of devices, where more robust database solutions would be required. Furthermore, while Django provides built-in authentication mechanisms, production deployment would require additional security measures such as HTTPS, token-based authentication, and secure API handling.

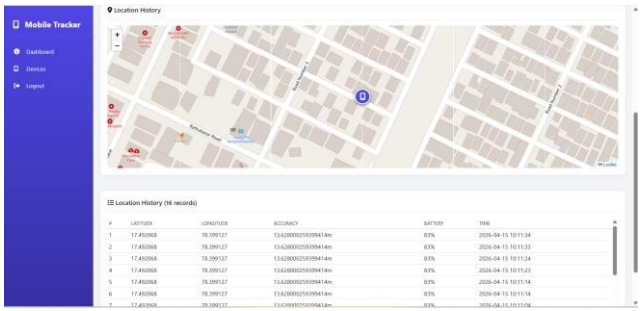


Fig. 4. Location History Tracking Overall, the developed system successfully meets the objectives outlined in the problem statement. It provides secure user authentication, efficient device registration through QR code pairing, near real-time tracking capabilities, location history management, interactive map visualization, and device status monitoring. The modular architecture ensures flexibility and supports future enhancements, making the system a reliable and extensible solution for mobile device tracking applications.

IX. CONCLUSION

This paper presented the design and implementation of a web-based mobile device tracking system developed using the Django framework and an Android mobile application. The proposed system enables real-time GPS-based location tracking through an interactive web dashboard, providing users with an efficient and accessible monitoring solution. The integration of core features such as secure user authentication, device registration through QR code pairing, real-time location updates, location history visualization, and battery status monitoring demonstrates the effectiveness of the system in addressing the limitations of existing tracking solutions. The overall architecture ensures seamless communication between the mobile application, backend server, and web interface, resulting in a reliable and user-friendly system.

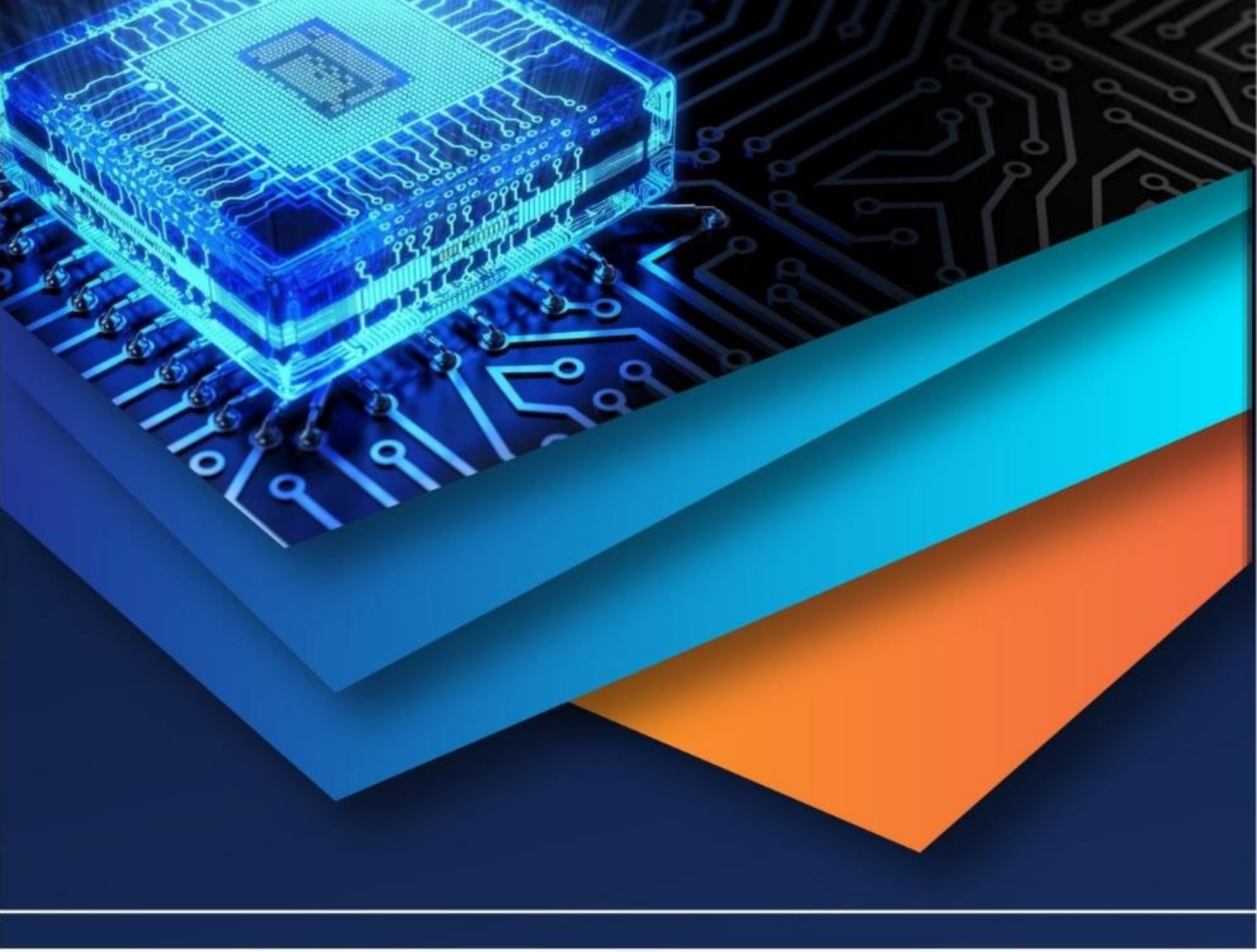
The developed system offers several advantages that make it practical and efficient for real-world applications. It is cost-effective, as it utilizes existing smartphone hardware without requiring additional specialized devices. The QR code-based pairing mechanism simplifies the device registration process, making it accessible even for nontechnical users. The web-based dashboard ensures crossplatform compatibility, allowing users to access the system from any modern browser. Furthermore, the system provides near real-time updates, with location data typically available within a few seconds of acquisition. The modular architecture of the Django framework enhances extensibility, enabling easy integration of new features. Additionally, the use of open-source technologies eliminates licensing costs and allows for customization based on specific user requirements.

Despite its effectiveness, the system also provides opportunities for further enhancement. Future work can focus on implementing geofencing capabilities to generate alerts when devices enter or exit predefined regions. Battery optimization techniques, such as adaptive tracking intervals based on movement patterns and battery levels, can significantly improve energy efficiency. Security can be strengthened by incorporating end-to-end encryption and deploying the system over secure HTTPS protocols. For large-scale deployments, migration to more robust database systems such as PostgreSQL or MySQL would improve scalability and performance. Additional features such as push notifications using cloud messaging services, support for multiple device types including iOS and dedicated GPS trackers, and offline data synchronization can further enhance usability. Moreover, advanced reporting tools, multi-user access control, and expanded API support for third-party integration can extend the system's functionality and applicability.

REFERENCES

[1] E. D. Kaplan and C. J. Hegarty, Understanding GPS: Principles and Applications, 2nd ed. Norwood, MA: Artech House, 2006.
[2] A. H. Holmes, The Definitive Guide to Django. Berkeley, CA: Apress, 2007.
[3] S. Loreto, M. S. H. I. Mie, and J. C. E. R. M. P. T. S. S. A. R. M., "Realtime web communication protocols," in IEEE Communications Magazine, vol. 52, no. 4, pp. 52-58, 2014
[4] D. Sharma and R. Singh, "Android application development: A comprehensive guide," in Proceedings of International Conference on Computing, Communication & Automation, 2015, pp. 772-777.
[5] R. Singh and V. Kumar, "Comparative analysis of web mapping libraries," in International Journal of Computer Science and Information Technologies, vol. 6, no. 3, pp. 2954-2958, 2015.
[6] Y. Wang, J. Zhang, and Q. Liu, "QR code based secure pairing protocol for mobile device connection," in Proceedings of IEEE International Conference on Communications, 2016, pp. 1-6.
[7] Django Software Foundation, "Django Documentation," 2025. [Online]. Available: https://docs.djangoproject.com/

[8] Google Inc., "Android Location Services," 2025. [Online]. Available: https://developer.android.com/training/location/
[9] V. Agafonkin, "Leaflet.js: The open-source JavaScript library for mobile-friendly interactive maps," 2025. [Online]. Available:
[10] https://leafletjs.com/
[11] T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1," RFC 7230, Internet Engineering Task Force, 2014.
[12] Bootstrap Team, "Bootstrap Documentation," 2025. [Online]. Available: https://getbootstrap.com/
[13] OpenStreetMap Foundation, "OpenStreetMap," 2025. [Online]. Available: https://www.openstreetmap.org/
[14] International Telecommunication Union, "Global Navigation Satellite System (GNSS) Manual," ITU-R M.1645-0, 2018.
[15] M. K. Sharma and S. K. Singh, "Real-time location tracking system using GPS and web technologies," in International Journal of Engineering and Technology, vol. 5, no. 2, pp. 892-896, 2013.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY