Ministry of Education and Science of Ukraine

ODESA NATIONAL UNIVERSITY OF TECHNOLOGY

International Competition of
Student Scientific Works

BLACK SEA SCIENCE 2026

PROCEEDINGS

ODESA, ONUT 2026

Ministry of Education and Science of Ukraine

Odesa National University of Technology

International Competition of Student Scientific Works

BLACK SEA SCIENCE 2026

Proceedings

Odesa, ONUT
2026

MONITORING SYSTEM FOR ELECTRIC SCOOTERS BASED ON IoT AND WEB TECHNOLOGIES

Author: Ivan Shvets
Advisor: Juliy Boiko
Khmelnytskyi National University (Ukraine)

Abstract. *This paper presents the development and study of a web-oriented monitoring system for electric scooters based on Internet of Things (IoT) technologies. The proposed solution is implemented as a multi-layer client–server system that integrates an ESP8266-based IoT controller with a web platform for real-time data acquisition, processing, and visualization. The system enables continuous monitoring of scooter location, operational status, battery level, and movement history using Wi-Fi communication and modern web technologies. At the design stage, UML-based modeling techniques were applied to describe system structure, interaction logic, and operational workflows. The conceptual models were further transformed into functional and schematic representations, ensuring consistency between the system architecture and its hardware–software implementation. The server-side component was developed using a modern JavaScript framework and provides a RESTful API, while real-time interaction and visualization are supported through WebSocket communication and Google Maps integration.*

The developed system supports intelligent zoning of urban areas, automated detection of rule violations, and administrative tools for fleet monitoring and management. Comprehensive testing at the unit, integration, system, and user interface levels confirmed stable operation and compliance with functional and non-functional requirements. Practical validation demonstrated that the proposed solution is suitable for real-world deployment and has significant potential for scalability, integration with smart city services, and further development toward data analytics and intelligent micro-mobility management.

Keywords: *Internet of Things, electric scooter monitoring, IoT-based system, ESP8266, web application, real-time telemetry, smart mobility, client–server architecture, Google Maps API.*

I. INTRODUCTION

The rapid development of information and communication technologies (ICT) in the 21st century has significantly transformed modern society, affecting transportation systems, urban infrastructure, and human–machine interaction. One of the most visible manifestations of this transformation is the emergence and large-scale deployment of intelligent mobility solutions, including electric scooters as a component of micro-mobility systems in smart cities. These systems combine embedded electronics, wireless communication, cloud platforms, and web technologies, forming cyber-physical environments that require reliable monitoring and control. In recent years, electric scooters have become widely used worldwide due to their energy efficiency, low environmental impact, and convenience for short-

distance urban transportation. According to global trends, shared and private e-scooter fleets are actively expanding in large cities, creating new challenges related to safety, fleet management, infrastructure load, and data-driven decision-making. Effective operation of such systems is impossible without automated monitoring of technical parameters, location, and usage conditions in real time.

The Internet of Things (IoT) [1] paradigm provides a technological foundation for building distributed monitoring systems that integrate sensors, communication modules, and data processing platforms. IoT-based solutions enable continuous telemetry acquisition, remote diagnostics, and intelligent control of mobile devices. When combined with modern web technologies, such systems allow the development of scalable client–server architectures with intuitive user interfaces, real-time data visualization, and flexible access control.

For Ukraine, the relevance of IoT-based monitoring systems for electric transport is particularly high. Urban mobility in Ukrainian cities is undergoing active transformation, while energy efficiency, transport safety, and infrastructure optimization remain critical issues. At the same time, the deployment of modern digital platforms is often constrained by limited resources, heterogeneous hardware, and the need for lightweight, cost-effective solutions. Therefore, the development of flexible and scalable monitoring systems based on open web technologies is of both scientific and practical importance. Despite the availability of commercial fleet management platforms, many existing solutions are closed, expensive, or poorly adaptable to specific local requirements. Moreover, a significant number of systems focus on basic GPS tracking only, without comprehensive telemetry analysis, real-time interaction, or advanced visualization capabilities. This creates a demand for research-oriented implementations that combine IoT technologies with modern web frameworks and provide extensibility for future automation and intelligent analytics.

This work is devoted to the development and study of an IoT-based monitoring system for electric scooters using web technologies. The proposed system is designed as a client–server web application that enables real-time collection, transmission, processing, and visualization of telemetry data, including geographical location, speed, battery charge level, and operational status of devices. Special attention is paid to system scalability, modular architecture, and real-time interaction using WebSocket technology [2].

The scientific novelty of this work lies in the integration of IoT telemetry [3], real-time communication mechanisms, and web-based visualization within a unified information system tailored for electric scooter monitoring. Unlike conventional tracking solutions, the proposed approach emphasizes extensibility, support for real-time data streams, and the use of modern frontend and backend technologies suitable for rapid deployment and further automation.

The practical significance of the study is determined by the possibility of applying the developed system for managing electric scooter fleets, analyzing usage patterns, improving operational efficiency, and supporting decision-making processes in urban transport systems. The results may also be used as a basis for further research in the fields of smart mobility, automation, and cyber-physical systems [4].

The main problem addressed in this work is the development of a functional and

scalable monitoring system that ensures real-time control and visualization of electric scooter parameters using IoT and web technologies.

To achieve this goal, the following objectives are defined:

- to analyze the requirements for monitoring electric scooters in IoT-based transportation systems;
- to design a client–server architecture for real-time data exchange and processing;
- to implement telemetry collection and visualization using modern web technologies;
- to evaluate the functionality and applicability of the proposed system for urban mobility scenarios.

From a practical implementation perspective, the proposed monitoring system is based on a real hardware–software prototype of an electric scooter IoT unit. The power supply subsystem operates with an input voltage of 48 V, which is converted to 5 V and 3.3 V levels to ensure stable operation of the ESP8266 microcontroller and peripheral modules. Telemetry data transmission is performed via a Wi-Fi communication channel [5] using the ESP8266 module, which provides reliable connectivity between the scooter and the server-side infrastructure. The system supports continuous monitoring of electrical parameters, including battery voltage and power consumption, as well as the acquisition of sensor data required for operational control. Data processing and control logic are implemented on the microcontroller level using embedded software, while higher-level data aggregation, analysis, and visualization are performed on the server side. Communication between the device and the server is organized through real-time data exchange mechanisms, enabling prompt response to changes in device status.

Furthermore, the system design includes the development of structured interaction between the client application, server modules, and cloud services. Special attention is paid to the collection and processing of telemetry data, the design of data transmission algorithms, and the implementation of an administrator web interface with map-based visualization, filtering tools, security mechanisms, and access control. This approach ensures practical applicability of the proposed solution and creates a foundation for further automation and intelligent data analysis in IoT-based micro-mobility systems.

II. LITERATURE ANALYSIS

2.1. Web Technologies and Architectural Approaches for IoT Monitoring Systems

The rapid development of web technologies and Internet of Things (IoT) solutions has significantly influenced the design of modern monitoring and control systems in the field of intelligent transportation and micro-mobility. Contemporary web applications act not only as information portals but also as full-featured platforms for real-time data acquisition, processing, and visualization. This evolution has enabled the creation of cross-platform systems that are independent of specific operating systems and hardware configurations, allowing users to access services through standard web browsers [6, 7].

A web application is generally defined as a software system that performs its functionality using a web browser as a client interface. Such applications may range from simple information services to complex, distributed, multi-user systems with real-time interaction capabilities. One of the key advantages of web-based solutions is their platform independence, which significantly reduces development and maintenance costs and ensures broad accessibility [8, 6]. This characteristic is especially important for IoT-based monitoring systems, where heterogeneous devices and user terminals are involved.

Modern web applications rely heavily on Application Programming Interfaces (APIs) that provide structured access to server-side services and external platforms. APIs enable seamless communication between frontend interfaces, backend logic, and external services such as mapping platforms and cloud infrastructures. In IoT-oriented systems, APIs play a critical role in integrating telemetry data streams with data storage, analytics, and visualization layers [9, 10].

A significant component of web-based monitoring systems is geospatial visualization. Mapping services such as Google Maps and OpenStreetMap provide powerful tools for representing spatial data, enabling real-time tracking, route analysis, and location-based filtering [10, 11]. These services are widely used in transportation and smart city applications due to their scalability, accuracy, and extensive ecosystem support. Their integration into IoT platforms enhances situational awareness and supports decision-making in urban mobility management [12, 13].

The literature identifies several approaches to web application development, including manual HTML-based development, the use of visual design tools, content management systems (CMS), application frameworks, and Software-as-a-Service (SaaS) platforms. Early web development relied primarily on manual HTML coding, which proved to be labor-intensive and insufficient for building dynamic and scalable systems. As a result, higher-level development tools and frameworks have become dominant in modern practice [6].

Content management systems provide rapid deployment capabilities and separate content from presentation logic, making them suitable for informational websites and simple services. However, CMS-based solutions often lack flexibility and are poorly suited for complex IoT systems requiring real-time data processing, device communication, and custom business logic [7]. Consequently, application frameworks have become the preferred choice for developing sophisticated web-based monitoring platforms.

Web frameworks offer a structural foundation for building dynamic applications by organizing system components and reducing repetitive coding tasks. Unlike CMS platforms, frameworks allow developers to implement custom logic, database models, and administrative interfaces tailored to specific problem domains. Frameworks are particularly effective in IoT systems where fine-grained control over data flows, security mechanisms, and scalability is required [8, 14].

Most modern web frameworks are based on the Model–View–Controller (MVC) architectural pattern, which separates data management, user interface, and control logic into independent components. This separation enhances system maintainability, modularity, and scalability. MVC architecture is widely adopted in IoT platforms and cloud-based services due to its ability to support concurrent users and distributed data processing [7, 15, 16].

2.2. IoT Communication Technologies, Hardware Platforms, and Application Domains

In the context of real-time monitoring systems, communication technologies play a crucial role. Protocols such as WebSocket and MQTT are commonly used for bidirectional, low-latency data exchange between devices and servers. WebSocket technology enables continuous real-time communication within web applications, making it suitable for telemetry visualization and live status updates [9]. MQTT, on the other hand, is optimized for lightweight IoT messaging and is frequently applied in resource-constrained environments [17,18].

From the hardware perspective, microcontrollers with integrated wireless connectivity, such as the ESP8266, are widely used in IoT monitoring systems. These devices provide sufficient computational power, low energy consumption, and native support for Wi-Fi communication, making them suitable for mobile and embedded applications [19, 20]. Numerous studies highlight the effectiveness of ESP-based platforms in smart transportation, environmental monitoring, and urban infrastructure systems [14, 21].

Recent research emphasizes the importance of integrating IoT monitoring systems with cloud-based infrastructures and smart city platforms. Such integration enables large-scale data aggregation, advanced analytics, and long-term storage, supporting intelligent decision-making and predictive maintenance [16, 22]. In the field of micro-mobility, IoT-based monitoring systems contribute to improved fleet management, enhanced safety, and optimized energy consumption [23].

Despite the availability of various commercial and research solutions, existing platforms often exhibit limitations related to closed architectures, limited extensibility, or insufficient support for real-time interaction and advanced visualization. This creates a research gap in the development of open, scalable, and web-oriented IoT monitoring systems tailored specifically to electric scooter applications. Addressing this gap requires combining modern web frameworks, real-time communication technologies, and embedded IoT devices within a unified architectural approach.

III. OBJECT, SUBJECT, AND METHODS OF RESEARCH OF THE ELECTRIC SCOOTER MONITORING SYSTEM

3.1. Object and Subject of Research

The rapid growth of urban micro-mobility services, particularly electric scooter rental systems, has created a strong demand for efficient tools for monitoring, management, and analytical assessment of vehicle fleets within urban environments. This demand is especially relevant for large cities, where the number of electric scooters is continuously increasing, making operational control, technical supervision, and compliance with municipal regulations critically important.

The object of research is the process of monitoring and managing electric scooter fleets in urban environments using modern information technologies.

The subject of research includes methods, software tools, and web-based technologies for the development of an IoT-oriented monitoring system that provides real-time visualization, data analysis, and operational control of electric scooters.

The proposed system is intended to support real-time tracking of scooter locations, monitoring of technical parameters (battery level, operational status), and analysis of movement history. The system interacts with a local API that supplies telemetry data, including geographic coordinates, device identifiers, battery charge level, scooter model, and event logs related to system usage, technical faults, or violations such as exiting permitted operating zones.

3.2. Purpose and Tasks of the Research

The purpose of this research is to design and implement a web-based monitoring system for electric scooters that enables real-time visualization, efficient fleet management, and analytical support for operational decision-making.

To achieve this purpose, the following research and development tasks were defined:

- development of a web-based administrative and analytical interface for monitoring electric scooters;
- implementation of real-time search and filtering mechanisms based on scooter model, status, battery level, and geographic location;
- integration of the Google Maps API for interactive visualization of scooter locations, including real-time updates, map scaling, and marker clustering;
- implementation of coverage zones, including permitted, restricted, and parking zones, with automatic detection of violations;
- provision of tools for viewing historical movement data to analyze routes and detect abnormal usage patterns;
- design of a scalable system architecture based on the JavaScript framework Adonis.js, enabling future expansion and integration with external services.

The developed system is intended for scooter rental operators, municipal authorities, and technical maintenance personnel, providing them with a unified platform for monitoring, control, and analysis of micro-mobility services.

3.3. Functional and Non-Functional Requirements

The system requirements were defined to ensure reliable operation, usability, and scalability of the monitoring platform.

Functional requirements of the system are summarized in Table 1.

Table 1. Functional requirements of the electric scooter monitoring system

No.	Functionality	Description
1	Search and filtering	Search scooters by ID, model, status, battery level, location, and time
2	Map visualization	Real-time display of scooters on an interactive map
3	Device information	Display detailed scooter parameters and last activity
4	Zone control	Visualization of allowed, restricted, and parking zones
5	Event monitoring	Detection and notification of zone violations and low battery levels
6	Route history	Visualization of movement history for selected scooters

Non-functional requirements include high system responsiveness (response time not exceeding 1–2 seconds for standard operations), reliability under unstable network

conditions, scalability to support an increasing number of devices, and security mechanisms such as user authentication, API protection using JWT tokens, input validation, and protection against common web attacks. The system interface must be adaptive and compatible with modern web browsers across various devices.

3.4. Input and Output Data Description

The system operates by processing input data received from users and external data sources and transforming it into meaningful output information presented through the web interface. Input data include user-entered search parameters and filtering criteria, such as scooter identifiers, operational status, battery charge level, and geographic constraints. Additionally, the system receives telemetry data from the backend API, including real-time coordinates, technical status updates, and event logs.

Output data are presented in visual, textual, and analytical forms. These include interactive map markers representing scooter locations, informational panels with detailed device parameters, notification messages related to system events, and analytical elements such as movement routes and statistical summaries of fleet status. The output information is dynamically updated based on API responses and user interactions, ensuring adaptability and real-time system behavior.

3.5. Research Methods and Development Stages

The development of the electric scooter monitoring system followed a classical software development life cycle model. This approach ensured systematic planning, structured implementation, and controlled validation of system functionality.

The main development stages are presented in Table 2.

Table 2. Stages of software development

Stage	Description
1	Justification of system development
2	Analysis of existing solutions and methods
3	Selection and justification of technologies
4	System design and architecture development
5	Software implementation
6	Testing and validation
7	Deployment
8	Documentation preparation

To evaluate development effort, the duration of each stage was estimated and recorded, as shown in Table 3.

During the implementation phase, the backend was developed using the Adonis.js framework with a RESTful API architecture, while the frontend utilized modern JavaScript components for data visualization. Integration with Google Maps API enabled real-time geospatial visualization and interaction with scooter markers. The system was tested using functional, load, and interface testing methods before deployment.

Table 3. Duration of development stages

Stage	Minimum (days)	Maximum (days)	Actual (days)
1	1	3	1
2	3	6	4
3	3	7	6
4	11	21	13
5	11	21	18
6	2	4	3
7	2	4	2
8	2	4	3
Total	35	70	50

Overall, the applied research methods and development approach ensured the creation of a scalable, reliable, and functionally complete monitoring system suitable for urban electric scooter services.

IV. RESULTS

4.1. System Architecture and Implementation Results

As a result of the conducted research and development process, a fully functional web-based system for monitoring and managing electric scooters was designed, implemented, and validated. The developed system is based on a client–server architecture, which proved to be an effective solution for ensuring scalability, modularity, and maintainability under real operational conditions. The server-side component was implemented using the AdonisJS framework (Node.js, TypeScript) and provides a RESTful API for communication with client applications. This choice enabled a clear separation between business logic, data access, and presentation layers, which significantly improved system extensibility and testability. The server application can be deployed both locally and in cloud environments (e.g., AWS, DigitalOcean, or Vercel), demonstrating platform independence and deployment flexibility.

The architectural structure of the system includes the following logical layers:

- user interaction layer (web interface),
- API controllers for handling HTTP requests,
- service layer implementing business logic,
- repository layer encapsulating data access,
- relational database for persistent data storage.

This multi-layer architecture ensured compliance with the Separation of Concerns principle and enabled independent development and testing of system components (Fig.1).

The implemented architecture demonstrated stable performance when processing concurrent client requests and allowed seamless integration with external services, including mapping services and notification modules.

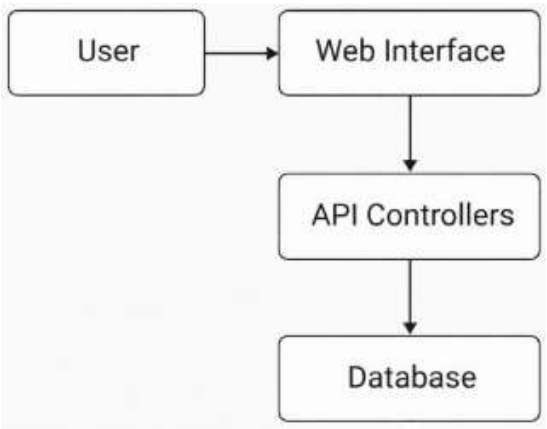

Fig. 1 - Block diagram of the client–server system architecture

4.2. UML-Based Modeling Results

At the system design stage, UML-based modeling techniques were applied to conceptually analyze the structure and behavior of the proposed solution and to support architectural decision-making. UML elements were used as analytical tools to describe system functionality, interaction logic, and operational workflowsprior to full-scale implementation.

The results of this modeling stage are reflected in the graphical. In particular, block and schematic diagrams (Fig. 2 and 3) illustrate the logical sequence of operations related to zone creation and the functional relationships between system components. These diagrams represent the system logic and process flow derived from UML activity and sequence modeling concepts.

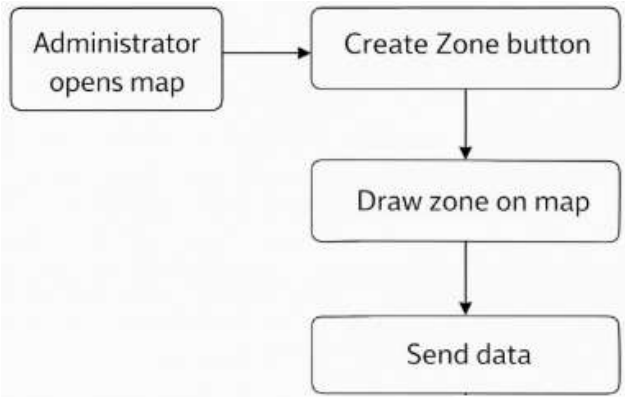

Fig. 2 - Workflow diagram of zone creation by the administrator

The practical realization of the designed workflows is demonstrated through a series of user interface views (Fig. 3–5), which visualize user interaction scenarios, form-based data input, device information monitoring, command execution, movement history analysis, and zone representation on an interactive map. Together, these figures confirm the consistency between the conceptual UML-based design and its software implementation.

In addition, a functional schematic diagram (Fig. 3) was developed to reflect the hardware-level realization of the UML-based system design, emphasizing the ESP8266 microcontroller as the core embedded control unit of the IoT system and illustrating its interaction with peripheral modules via the I²C communication bus.

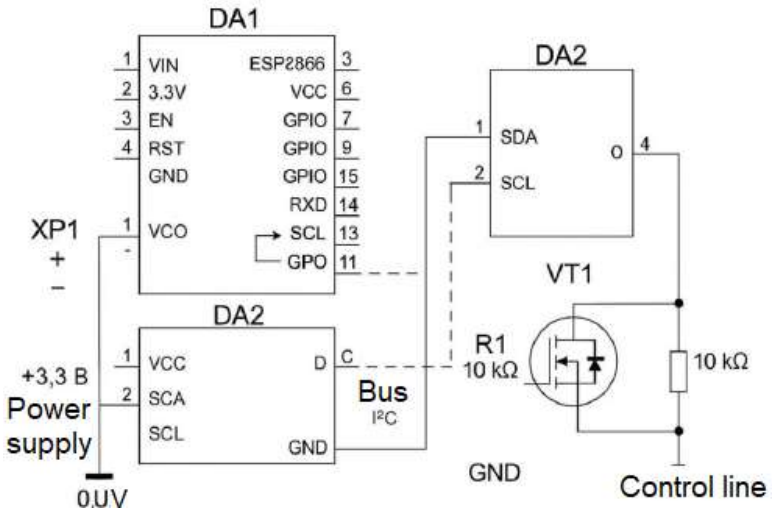

Fig. 3. Functional schematic diagram of the system derived from UML-based design

This diagram represent the system logic and process flow derived from UML activity and sequence modeling concepts.

4.3. User Interface Design and Functional Results

The developed user interface (UI) serves as the primary interaction point between administrators and the monitoring system. The UI design focused on real-time visualization, operational simplicity, and rapid response to critical events such as low battery level, zone violations, or device malfunction (Fig. 4).

The interface follows the principles of:

- cognitive simplicity (single main map-based screen),
- minimal interaction cost (key actions performed in no more than three clicks),
- real-time feedback (WebSocket-based push notifications),
- adaptive layout (responsive design for desktop and mobile devices).

The UI includes the following functional elements:

- global toolbar with connection status and search,
- interactive Google Maps view with real-time markers,
- side panel for filtering and analytics,
- administrative section for managing users, roles, zones, and maintenance schedules.

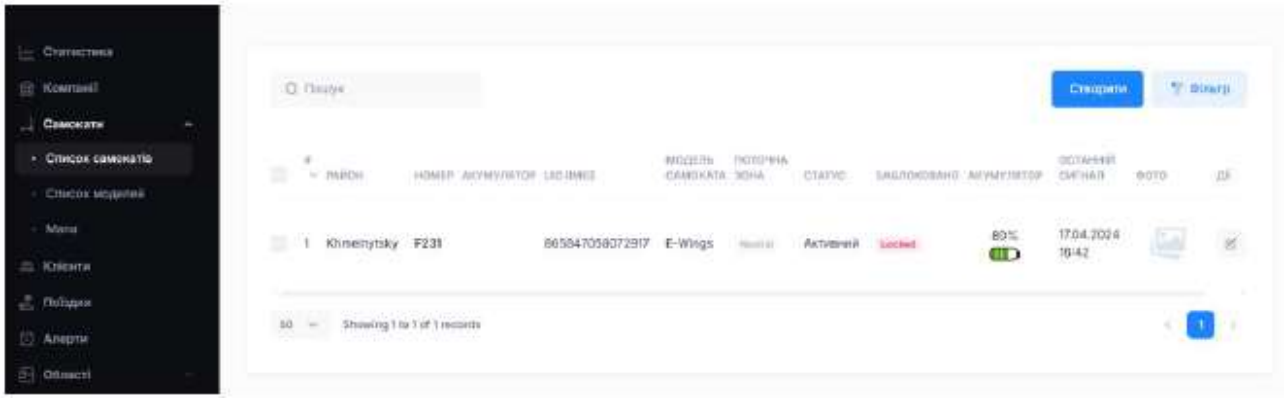

Fig. 4 - General layout of the web interface

A key functional result is the implemented zone creation workflow, which provides administrators with the ability to define polygonal zones directly on the map and configure their operational parameters. The logical sequence of this process was previously formalized at the modeling stage and is illustrated in Fig. 2. The practical execution of the zone creation process is carried out through an intuitive graphical interface and synchronized with the backend via API calls, confirming the consistency between the conceptual process model and its UI-level implementation.

4.4. Software Implementation Results

The system was implemented using TypeScript as the primary programming language, which enabled strong typing, improved code reliability, and enhanced development productivity. The backend logic adheres to SOLID principles, ensuring modularity and long-term maintainability of the software architecture.

The project structure includes:

- configuration modules,
- domain-driven business logic modules,
- HTTP controllers and middleware,
- validation components,
- service and use-case layers,
- database models and migration scripts.

A PostgreSQL relational database was used for data persistence. The Lucid ORM provided full support for relational mappings and complex query construction, enabling efficient data manipulation in accordance with business requirements.

The practical results of the software implementation are demonstrated through the user interface components that provide detailed device monitoring and control. In particular, the system allows administrators to access comprehensive information about each scooter, including operational status, technical parameters, and contextual metadata (Fig.5).

Additionally, extended data related to scooter operation, diagnostics, and historical parameters are available through a dedicated interface, enabling deeper analysis and informed decision-making (Fig. 6).

As a result, the implemented software demonstrates:

- stable operation under load,
- clear architectural organization,

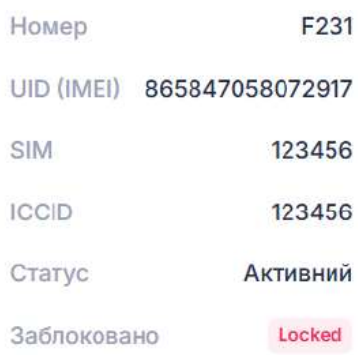

Fig. 5 - Scooter information view

Fig. 6. Extended scooter information view

- readiness for functional extension,
- compliance with modern web development standards.

4.5. Testing and Validation Results

Comprehensive testing was conducted to evaluate system correctness, stability, and usability. The testing process covered multiple levels:

- Unit testing, focusing on individual services and modules,
- Integration testing, verifying interactions between system components,
- System testing, validating compliance with functional requirements,
- Manual UI testing, ensuring correctness of user interactions.

Test scenarios were derived from system documentation and real-world usage cases. The results confirmed correct system behavior under both normal and boundary conditions (Fig. 7).

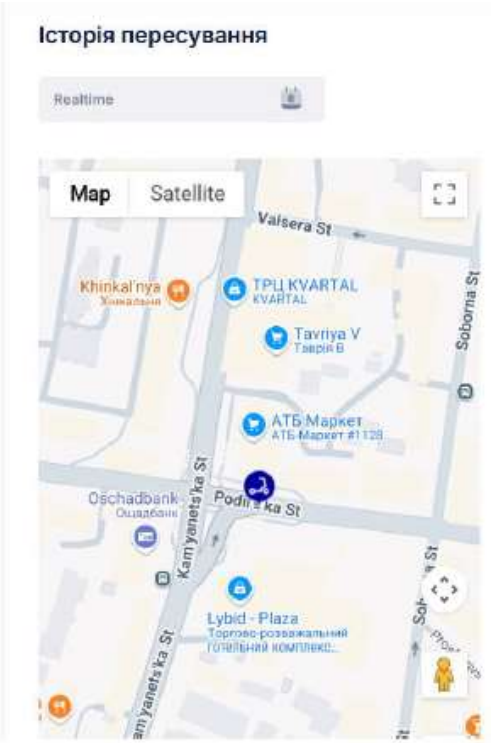

Fig. 7. Example of system testing results: user interface validation for movement history visualization

The conducted testing demonstrated that the system meets functional requirements and is ready for real-world deployment.

4.6. Operational Results and Practical Applicability

The developed system was successfully deployed in a local environment and validated through real user interaction scenarios (see Fig. 8). Administrators can:

- register and manage electric scooters,
- monitor real-time location and movement history,
- issue control commands (lock, unlock, speed limitation),
- configure operational zones,
- analyze usage data through visual dashboards.

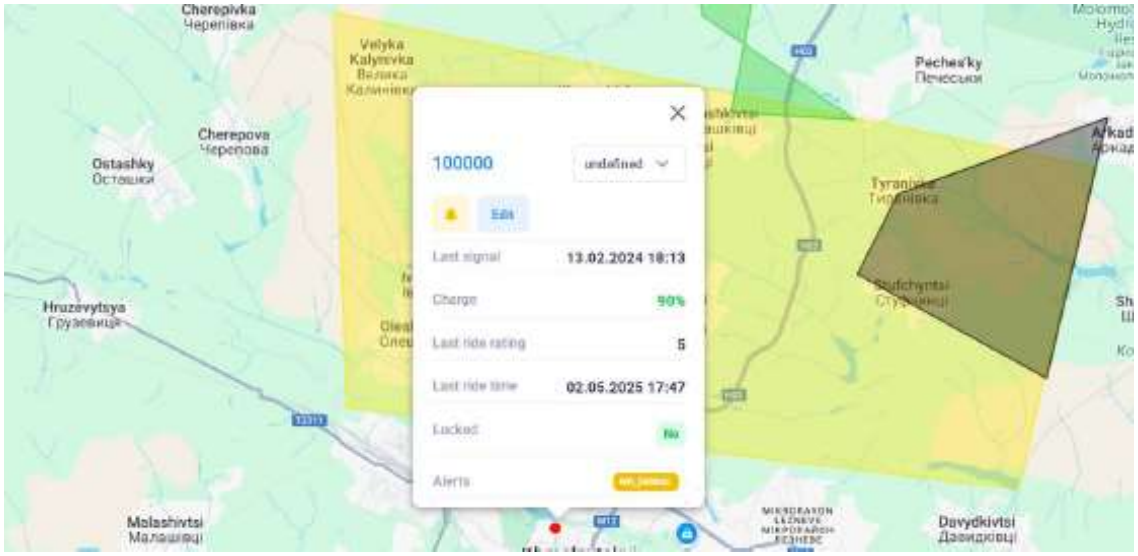

Fig. 8. Map view with active scooters and zones

The obtained results confirm that the proposed solution is practical, scalable, and suitable for further commercialization or integration into smart city infrastructure.

V. CONCLUSIONS

In this work, a web-oriented electric scooter monitoring system was developed and investigated. The system is based on a multi-layer client–server architecture and utilizes Internet of Things (IoT) technologies. The proposed solution integrates the hardware component of an electric scooter equipped with an ESP8266-based IoT controller with a software platform for real-time acquisition, transmission, processing, and visualization of telemetry data.

During the research, a complete system development cycle was implemented, ranging from requirement formalization and UML-based modeling to software implementation, testing, and practical validation of system functionality. UML diagrams were used to structurally describe the system operation logic, user interaction scenarios, and sequences of operations. Their subsequent transformation into functional and schematic diagrams confirmed the correctness and consistency of the adopted architectural decisions.

The developed system provides:

- accurate tracking of electric scooter locations in an urban environment using Google Maps cartographic services;

- intelligent territorial zoning with support for permitted, restricted, paid, bonus, and speed-limited zones;
- automated control of scooter operation rules with event generation and notification mechanisms;
- collection and analysis of telemetry data, including movement routes, speed, stops, and power consumption parameters;
- a user-friendly web-based administrative interface for monitoring, control, and analytical assessment of the scooter fleet status.

The use of the ESP8266 microcontroller as the core IoT control unit ensured reliable data exchange via a Wi-Fi communication channel [5], low hardware implementation cost, and system scalability. The implemented power supply scheme and interaction with peripheral modules confirm the suitability of the proposed approach for mobile micro-mobility devices. Testing results obtained at the module, integration, system, and user interface levels confirmed stable system operation, correct handling of normal and boundary scenarios, and compliance with both functional and non-functional requirements. Practical operation of the system in a local environment demonstrated its readiness for real-world deployment.

Thus, the proposed system represents an effective tool for monitoring and managing electric scooters in urban conditions. It demonstrates a high potential for scalability, integration with smart city services, and further development, particularly in the directions of data analytics, load forecasting, and the application of artificial intelligence methods to optimize micro-mobility transportation systems.

VI. REFERENCES

1. Liu, B., Feng, Q., Zhong, L., & Chen, Y. (2025). Physical-information-functional architecture for Internet of Things. *IEEE Internet of Things Journal*, 12(20), 42456–42483. https://doi.org/10.1109/JIOT.2025.3593339

2. Li, B.-C., & Yu, S.-Z. (2016). Keyword mining for private protocols tunneled over WebSocket. *IEEE Communications Letters*, 20(7), 1337–1340. https://doi.org/10.1109/LCOMM.2016.2565465

3. Farooq, M. S., Mir, R. P., Alvi, A., Tutusaus, K., Garcia Villena, E., Alrowais, F., Karamti, H., & Ashraf, I. (2025). Harnessing AI forward and backward chaining with telemetry data for enhanced diagnostics and prognostics of smart devices. *Scientific Reports*, 15, 7577. https://doi.org/10.1038/s41598-025-89266-9

4. Cao, M., Ye, M., & Zino, L. (2026). Control of networked cyber–physical–human systems. *Nature Reviews Electrical Engineering*, 3, 32–45. https://doi.org/10.1038/s44287-025-00224-z

5. Boiko, J., Pyatin, I., & Druzhynin, V. (2023). Possibilities of the MUSIC algorithm for Wi-Fi positioning according to the IEEE 802.11az standard. *2023 IEEE International Conference on Information and Telecommunication Technologies and Radio Electronics (UkrMiCo)*, 1–6. https://doi.org/10.1109/UkrMiCo61577.2023.10380354

6. Node.js. (n.d.). Node.js Documentation. https://nodejs.org/en/docs/

7. World Wide Web Consortium (W3C). (2020). Web of Things (WoT) Architecture. https://www.w3.org/TR/wot-architecture/

8. AdonisJS. (n.d.). AdonisJS Framework Documentation. https://docs.adonisjs.com

9. Mozilla Contributors. (n.d.). WebSocket – MDN Web Docs. https://developer.mozilla.org/en-US/docs/Web/API/WebSocket

10. Google. (n.d.). Google Maps JavaScript API Documentation.

https://developers.google.com/maps/documentation/javascript/overview

11. OpenStreetMap contributors. (n.d.). OpenStreetMap Documentation. https://wiki.openstreetmap.org/wiki/Main_Page

12. Taneja, M., & Patel, D. (2021). Big data and IoT for smart cities. Springer. https://link.springer.com/book/10.1007/978-981-33-4739-2

13. SmartCitiesWorld. (n.d.). IoT in urban mobility. https://www.smartcitiesworld.net/

14. IEEE Internet of Things Journal. (n.d.). IEEE Internet of Things Journal. https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=6488907

15. Minerva, R., Biru, A., & Rotondi, D. (2015). Towards a definition of the Internet of Things (IoT). IEEE IoT. https://iot.ieee.org/images/files/pdf/IEEE_IoT_Towards_Definition_Internet_of_Things_Revision1_27MAY15.pdf

16. Lee, I., & Lee, K. (2015). The Internet of Things (IoT): Applications, investments, and challenges for enterprises. Business Horizons. https://doi.org/10.1016/j.bushor.2015.03.008

17. MQTT.org. (n.d.). Lightweight IoT Messaging Protocol. https://mqtt.org

18. ISO. (2018). ISO/IEC 30141:2018 – Internet of Things (IoT) Reference Architecture. https://www.iso.org/standard/65695.html

19. Espressif Systems. (n.d.). ESP8266 Technical Documentation. https://www.espressif.com/en/products/socs/esp8266/resources

20. Arduino. (n.d.). Arduino Documentation (IoT with ESP8266). https://docs.arduino.cc/hardware/nodemcu-esp8266

21. Intel Corporation. (n.d.). Smart mobility with IoT. https://www.intel.com/content/www/us/en/internet-of-things/overview.html

22. McKinsey & Company. (2022). The Internet of Things: Mapping the value beyond the hype. https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/the-internet-of-things

23. United Nations Environment Programme. (n.d.). Electric two- and three-wheelers. https://www.unep.org/resources/report/electric-two-and-three-wheelers