Bachelor's Thesis

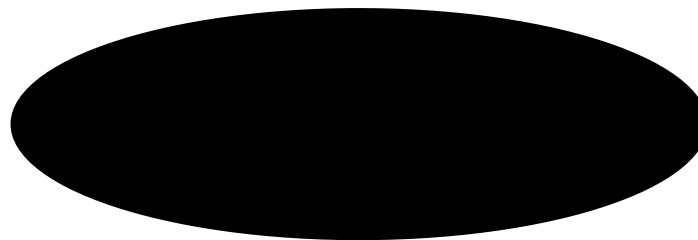# Integration of Rescuetrack into the International Standard Accident Number System

Omar Zitouni

Informatik (B.Sc.)

Peter L. Reichertz Institute for Medical Informatics (PLRI)

Supervisor

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Abschlussarbeit selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Die wörtliche oder sinngemäße Übernahme von Abschnitten aus Texten Dritter sowie aus eigenen vorangegangenen Veröffentlichungen habe ich kenntlich gemacht.

Ferner versichere ich, dass es sich hier um eine Originalarbeit handelt, die noch nicht in einer anderen Prüfung vorgelegen hat.

(Ort, Datum, Unterschrift)

Erklärung zur Abgabe der gedruckten Abschlussarbeit

Hiermit versichere ich, dass die vorliegende gedruckte Abschlussarbeit mit der elektronisch abgegebenen (hochgeladenen) Abschlussarbeit exakt übereinstimmt und dass ich keine unerlaubten Änderungen vorgenommen habe.

(Ort, Datum, Unterschrift)

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Abschlussarbeit selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Die wörtliche oder sinngemäße Übernahme von Abschnitten aus Texten Dritter sowie aus eigenen vorangegangenen Veröffentlichungen habe ich kenntlich gemacht.

Ferner versichere ich, dass es sich hier um eine Originalarbeit handelt, die noch nicht in einer anderen Prüfung vorgelegen hat.

(Ort, Datum, Unterschrift)

Erklärung zur Abgabe der gedruckten Abschlussarbeit

Hiermit versichere ich, dass die vorliegende gedruckte Abschlussarbeit mit der elektronisch abgegebenen (hochgeladenen) Abschlussarbeit exakt übereinstimmt und dass ich keine unerlaubten Änderungen vorgenommen habe.

(Ort, Datum, Unterschrift)

Peter L. Reichertz Institut für Medizinische Informatik
der TU Braunschweig und der Medizinischen Hochschule Hannover
Postfach 3329 · 38023 Braunschweig

PETER L. REICHERTZ INSTITUT FÜR MEDIZINISCHE INFORMATIK

der TU Braunschweig und
der Medizinischen Hochschule
Hannover

Mühlenpfordtstr. 23
38106 Braunschweig

http://www.plri.de

Tel.: +49 531 391-2130 (Sekr.)
Fax: +49 531 391-9502

Ihr Ansprechpartner:
Paulo Haas
Tel.:+49 531 391 2126
Fax: +49 531 391 9502
Mobil:: +49 160 1166234
paulo.haas@plri.de

Omar Zitouni
Matrikelnr.
Studiengang Informatik (B. Sc.)

Braunschweig, 12. Februar 2025

Aufgabenstellung zur Bachelorarbeit „Integration von Rescuetrack in das International Standard Accident Number-System"

In dem Projekt „International Standard Accident Number" (ISAN) wurde eine Plattform entwickelt, um Unfalldaten zwischen dem Unfallort, dem Rettungsdienst und den Gesundheitseinrichtungen zu auszutauschen.

Im Rahmen der Bachelorarbeit soll das ISAN-Projekt erweitert werden und das Rescuetrack-Gerät in das System für den Rettungsdienst integriert werden. Das Gerät ist im Forschungsfahrzeug eingebaut und erfasst Standortdaten, die über eine API abgerufen werden können. Das System für die Gesundheitseinrichtungen soll den Standort über das System für Rettungsdienst live abgerufen können und mit dem voraussichtlichen Fahrtweg und Ankunftszeit ergänzen.

Weitere Krankenwägen sollen mit einer Simulation ergänzt werden, deren Daten auch live angezeigt werden sollen. Die Implementierung wird mit zwei Szenarien getestet.

Im ersten Szenario wird ein Unfall mit drei Fahrzeugen simuliert. Jeder Unfall erhält eine eigene ISAN und einen zuständigen Krankenwägen. Die Krankenwägen fahren zum Unfallort und holen die Patienten ab.

Im zweiten Szenario wird ein Unfall mit einem Fahrzeug simuliert. Der zuständige Krankenwagen fällt aus und das ISAN-System weist dem Unfall einen neuen Krankenwagen zu.

Die Implementierung erfolgt entsprechend der Spezifikationen der

Geschäftsführender Direktor:
Prof. Dr. Thomas M. Deserno

Leiter Standort Braunschweig:
Prof. Dr. Thomas M. Deserno

Leiter Standort Hannover:
Prof. Dr. Dr. Michael Marschollek

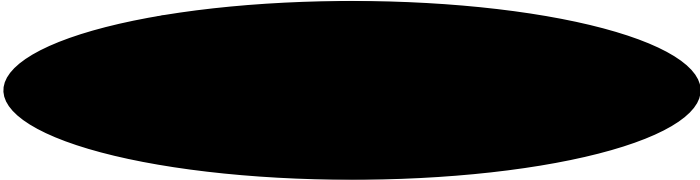**Technische Universität Braunschweig**

MHH

ISAN-Plattform. Die Aufgabenstellung umfasst sowohl Planung als auch Validierung der Implementierung. Die Arbeit ist nach aktuellen Leitlinien zur Sicherung guter wissenschaftlicher Praxis zu erstellen. Die Vorlagen und Gestaltungsrichtlinien des PLRI sind zu beachten.

Ausgabedatum: 2024-12-02
Bearbeitungszeit: 12 Wochen
Ausgabedatum: 2024-03-02

Omar Zitouni (Studierender)

Abstract

Emergency response coordination is often hindered by fragmented information and communication technology systems, leading to inefficiencies in data exchange between emergency responders and healthcare facilities. The International Standard Accident Number (ISAN) system aims to address this challenge by enabling seamless data integration across emergency services. However, ISAN lacks real-time tracking capabilities for responding systems, such as ambulances, which are critical for effective emergency management. To address this limitation, this thesis investigates the integration of the Rescuetrack into the ISAN system to enable real-time tracking of emergency vehicles. This integration seeks to enhance rescue coordination, minimize response times, and improve hospital preparedness by displaying real-time tracking data, including vehicle locations, routes, and estimated arrival times. To achieve this, we developed a functionality to transmit GPS locations, retrieved by the Rescuetrack, from the Responding System to the Curing System through the ISAN system. Additionally, we implemented a tracking interface in the Curing System to display real-time tracking data and developed a rerouting functionality to assign a new ambulance in case of a vehicle breakdown. To evaluate the developed functionalities, we used a research vehicle to test real-time tracking in a simulated emergency response. During the test, GPS data was correctly retrieved and transmitted to the Curing System, where real-time tracking data was visualized. However, the average interval between consecutive position updates from Rescuetrack was 16.10 ± 2.8 seconds, leading to longer update delays on the map. Additionally, simulation scenarios were conducted, confirming that the ISAN platform supports multi-ambulance tracking and automated rerouting in case of vehicle breakdowns. These findings confirm the successful integration. However, further improvements are needed to refine and enhance the assignment and tracking of emergency responders within the ISAN platform.

Zusammenfassung

Die Koordination von Notfalleinsätzen wird oft durch fragmentierte Informations- und Kommunikationstechnologiesysteme erschwert, was zu Ineffizienzen beim Datenaustausch zwischen dem System für Rettungsdienste und dem System für Gesundheitseinrichtungen führt. Das International Standard Accident Number (ISAN)-System zielt darauf ab, diese Herausforderung zu bewältigen, indem es eine nahtlose Datenintegration zwischen Notfalldiensten ermöglicht. Allerdings fehlt dem ISAN-System eine Echtzeitverfolgung für Rettungssysteme wie Krankenwagen, die für ein effektives Notfallmanagement entscheidend sind. Um diese Einschränkung zu überwinden, untersucht diese Arbeit die Integration des Rescuetrack in das ISAN-System, um eine Echtzeitverfolgung von Einsatzfahrzeugen zu ermöglichen. Diese Integration soll die Koordination von Rettungseinsätzen verbessern, Reaktionszeiten minimieren und die Krankenhausvorbereitung optimieren, indem Echtzeit-Tracking-Daten wie Fahrzeugstandorte, Routen und geschätzte Ankunftszeiten angezeigt werden. Dazu wurde eine Funktion entwickelt, um GPS-Positionen, die vom Rescuetrack erfasst wurden, aus dem System für Rettungsdienste in das System für Gesundheitseinrichtungen über das ISAN-System zu übertragen. Zusätzlich wurde eine Tracking-Schnittstelle im System für Gesundheitseinrichtungen implementiert, um Echtzeit-Tracking-Daten darzustellen, sowie eine Umleitungsfunktion entwickelt, um im Falle einer Fahrzeugpanne ein neues Einsatzfahrzeug zuzuweisen. Zur Evaluierung der entwickelten Funktionen wurde ein Forschungsfahrzeug genutzt, um die Echtzeitverfolgung in einem simulierten Notfalleinsatz zu testen. Während des Tests wurden GPS-Daten erfolgreich erfasst und an das System für Gesundheitseinrichtungen übertragen, wo die Echtzeit-Tracking-Daten visualisiert wurden. Allerdings betrug das durchschnittliche Intervall zwischen aufeinanderfolgenden Positionsaktualisierungen durch Rescuetrack 16,10 ± 2.8 Sekunden, was zu längeren Aktualisierungsverzögerungen auf der Karte führte. Zusätzlich wurden Simulationen durchgeführt, die bestätigten, dass das ISAN-System die Verfolgung mehrerer Einsatzfahrzeuge sowie eine automatisierte Umleitung im Falle einer Fahrzeugpanne unterstützt. Diese Ergebnisse bestätigen die erfolgreiche Integration. Dennoch sind weitere Verbesserungen erforderlich, um die Zuweisung und Verfolgung von Einsatzfahrzeugen innerhalb des ISAN-Systems weiter zu optimieren.

Contents

List of Figures

List of Tables

1 Introduction

Despite today's technological advancements, rescue chains still rely heavily on manual processes, with humans playing the central role in rescue operations [1]. These can lead to undesirable results, as the victims in accidents could be unconscious and actually in need of a second person to recognize the event and notify emergency services [2]. Once the emergency is reported, responders face the challenge of gathering relevant details ("What?", "Who", "Where?", etc.) and dispatching the appropriate rescue team [3], which leads to delays and could also be vulnerable to miscommunication or language impediment of the reporter [4].

With the growth of the Internet of Things (IoT), rescue chains have no longer to rely only on human intervention. Smart wearables, cars, or homes can detect accidents in real-time and generate autonomous alerts [2]. These devices can communicate essential details such as location and vital data with the emergency responders and healthcare providers at the hospital, giving a better understanding of the situation, and reducing delays and the risk of human error. However, there is currently no unified approach to link the separate information and communication technology (ICT) systems utilized by the various parties in this communication [3].

For this reason, the International Standard Accident Number (ISAN) communication platform was introduced to connect isolated systems, avoid misinterpretation of data, and integrate additional data sources [1]. This platform (Fig. 1) consists of an Alerting System (AS), which represents devices that can generate an emergency alarm such as smart homes, cars, and wearables. The Responding System (RS), represents rescue teams such as ambulances, air rescue, and sea rescue that will respond to the generated alert. The Curing System (CS) encircles hospitals and emergency rooms, where victims receive essential health care.

At the core of the ISAN platform is the ISAN System, which ensures effective coordination and data exchange among the different ICT systems involved. This system consists of the Workflow Manager (WFM), which oversees the sequence of actions ensuring that all rescue operations are carried out in the correct order, the Communication Manager (ComM), which handles the secure and reliable transmission of data such as location and vital data of the victim, and specific System Managers for the alerting, responding, and curing systems to ensure the correct integration of each system within the platform.
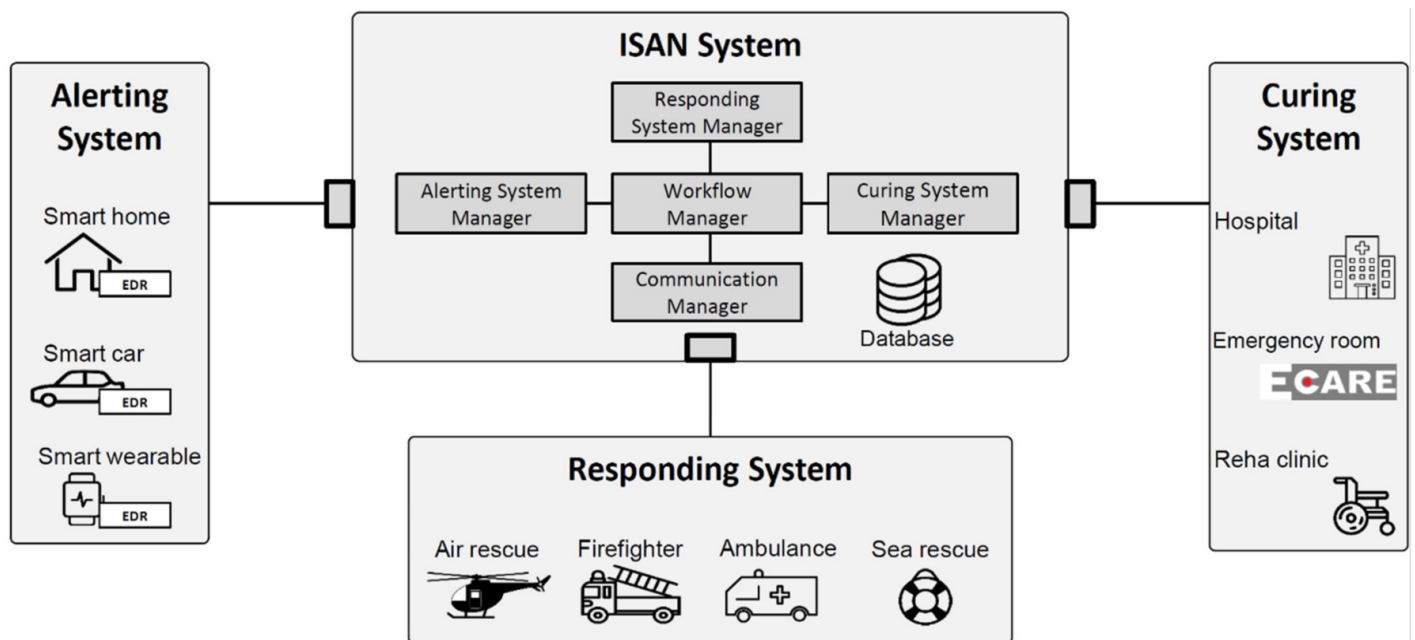
Figure 1: System architecture of the ISAN emergency communication platform [1]

The ISAN project is already capable of performing advanced tasks such as the transmission of Electrocardiogram (ECG) data between the AS and CS or visualizing emergency situations by displaying both the location of the incident and the hospital to which the patient will be transferred on a map (Fig. 2). To further enhance the capabilities of this platform, this thesis focuses on the integration of Rescuetrack, specifically aiming to include the real-time locations of emergency vehicles into the map, enabling more dynamic and accurate coordination of rescue efforts.
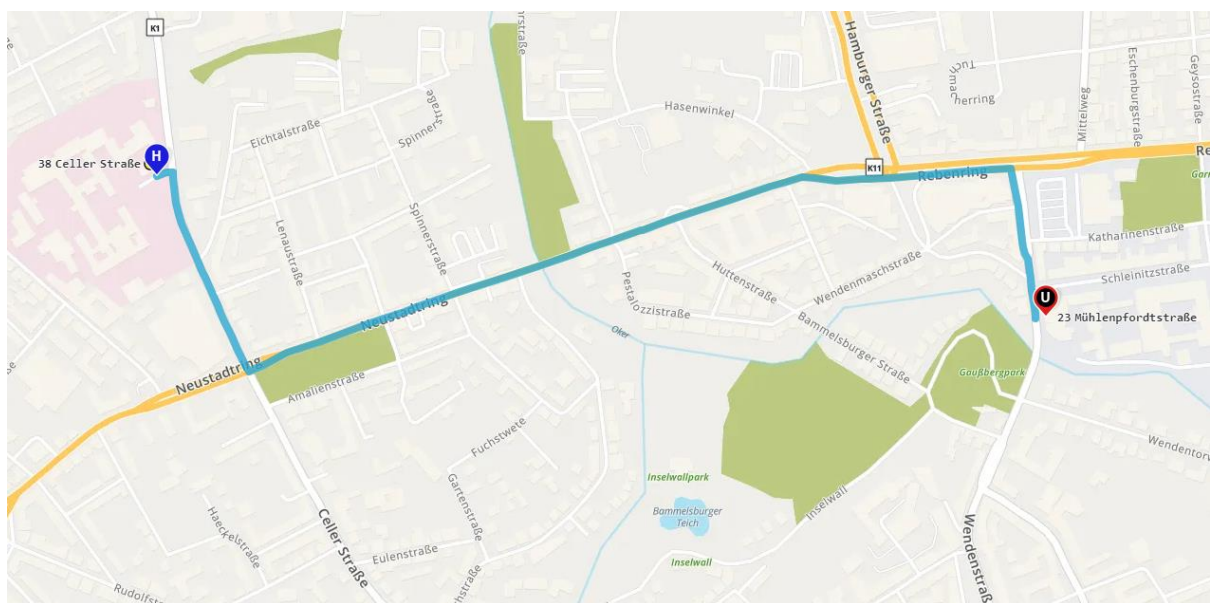
Figure 2: The displayed map in the CS after alarm generation (U) and hospital (H) assignment [1]

1.1 Motivation and Problem Statement

One of the motivations for this integration is the ability to provide real-time visibility of all emergency vehicles, which is currently missing in the ISAN platform. Without this functionality, it is difficult to coordinate rescue efforts, especially in complex scenarios, such as multiple accidents occurring in the same or nearby areas. In these situations, having a visual depiction of the incidents' locations, along with the positions of rescue teams and their estimated arrival times to the accident sites, is essential, as it enhances better coordination between rescue teams, avoids overlapping, and ensures that all incidents are managed efficiently and without delays.

Additionally, with the help of IoT and Car-to-Car (C2C) Communication, emergency vehicles can receive the best routes to incident or hospital locations that take into consideration real-time traffic conditions, such as the presence of traffic jams or flooded roads [5]. Similarly, in case of a vehicle breakdown or a situation where the initial ambulance is no longer available, the platform currently lacks the capability to reroute a new ambulance to the incident. This integration will address such issues by automatically assigning the next available emergency vehicle.

Another motivation is that the integration of Rescuetrack will ensure that hospitals are better prepared for incoming patients. Currently, hospitals do not receive estimated arrival times of ambulances, which delays preparations. With the display of estimated arrival times at the hospital, the medical team will be ready to prepare the necessary equipment and allocate staff required for the patient's condition, ensuring a faster and more effective response upon arrival [6].

In an emergency, every second counts and we can see from above that the integration of this Rescuetrack plays a role in reducing the delays of the rescue team transporting the patient and hospitals preparing for their arrival, which means, that we will have more chances of saving people's lives, especially in serious conditions [7].

1.2 Task Description

Based on the motivations, the following tasks have been identified to achieve the objectives of this thesis.

1. Retrieval Of Coordinates From Rescuetrack
- Access the Rescuetrack Application Programming Interface (API) and retrieve real-time location data (e.g. latitude and longitude) of an ambulance.

2. Integration And Data Display On Map
- Implement functionality to monitor the movement of the emergency vehicle on the map (Fig. 2) with respect to the communication protocols of the ISAN platform.

3. Additional Dynamic Adjustments:
- Display and update the routes and the estimated arrival time of the emergency vehicles on the map.
- Implement functionality that reroutes a new emergency vehicle to an incident if the initial vehicle experiences a breakdown or becomes otherwise unavailable.

4. Simulation Scenario:
- The ability to create simulation scenarios involving multiple emergency vehicles and incidents. This will include the functionality of monitoring various vehicles on the map simultaneously, rather than just a single vehicle.
- Two example scenarios:
  - The first scenario is a major car accident involving three vehicles, where an ambulance is assigned to each vehicle's driver, first responding to the accident site and then transporting them to the corresponding hospital.
  - The second scenario is a single-car accident where the driver is initially assigned an ambulance. The ambulance becomes unavailable due to a breakdown, and the system automatically reroutes a second ambulance to the driver.
- Ensure that the map dynamically updates the locations, shows the routes of all involved vehicles, and estimated arrival time providing an accurate representation of the emergency response.

1.3 Research Questions

Based on the above, the research questions of this thesis are:

- **RQ1:** How feasible is the integration of Rescuetrack into the ISAN platform?

- **RQ2:** How well can the ISAN platform dynamically update emergency vehicle positions on the map?

- **RQ3:** Can the ISAN platform feasibly display multiple emergency vehicles on the map as required by the simulation?

- **RQ4:** Is it feasible for the ISAN platform to automatically reroute a new emergency vehicle in the event of a vehicle breakdown?

- **RQ5:** How well can the ISAN platform update the estimated arrival times of emergency vehicles?

- **RQ6:** How well can the ISAN platform update the routes of emergency vehicles?

2 Fundamentals

2.1 Literature Survey

2.1.1 Literature Collection

To better understand the theme of this bachelor thesis, we performed a literature search based on the Scopus[1] and PubMed[2] databases. We applied advanced search techniques using logical conjunction "AND" and "OR" to combine keywords.

- Scopus Search Query: TITLE-ABS-KEY(("Ambulance tracking" OR "Ambulance location monitoring" OR "Vehicle tracking") AND ("GPS" OR "Global Positioning System" OR "GIS" OR "Geographic Information System" OR "Geolocation") AND ("Real-time" OR "Live" OR "Continuous")) AND ("API" OR "Application Programming Interface")

- PubMed Search Query: ((Ambulance tracking[Title/Abstract]) OR (Ambulance location monitoring[Title/Abstract]) OR (Vehicle tracking[Title/Abstract])) AND ((GPS[Title/Abstract]) OR (Global Positioning System[Title/Abstract]) OR (GIS[Title/Abstract]) OR (Geographic Information System[Title/Abstract]) OR (Geolocation[Title/Abstract])) AND ((Real-time[Title/Abstract]) OR (Live[Title/Abstract]) OR (Continuous[Title/Abstract])) AND ((API[All Fields]) OR (Application Programming Interface[All Fields]))

From the query on Scopus, we obtained 27 documents from 2008 to 2024, but from Pubmed, we found only one document. On Scopus, we applied document type limitations to constraint results in "Conference paper" and "Articles", which left us with 25 results. Further, we filtered the results by subject area focusing on "Computer Science". After this step, we had 20 results left.

After basic filtering, we had 21 results. The single document found in Pubmed was also found in Scopus so we ended up again with 20 unique results. We then reviewed the abstracts and titles to identify relevant papers, narrowing the selection down to 11 results. In the last step, we read through the remaining results and kept 5 documents, which we will then summarize in the following section.

---

[1] Scopus. Access to research. Available from: https://www.scopus.com/. Accessed 2024 Oct 10.
[2] Pubmed. National Library of Medicine. Available from: https://pubmed.ncbi.nlm.nih.gov/.
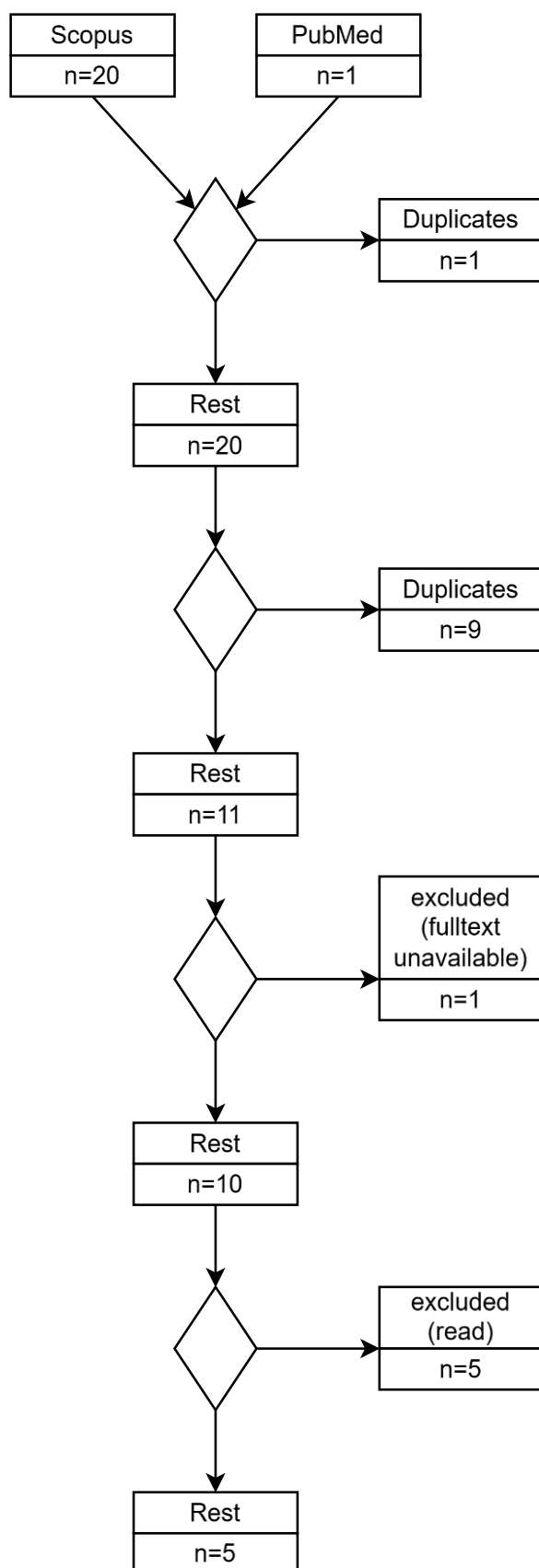
Figure 3: Flowchart of the literature search process

2.1.2 Literature Summary

The selected papers presented a variety of Global Positioning System (GPS) based vehicle tracking systems, each tailored to a different use case but sharing the common goal of real-time tracking.

Starting from similar contexts, Kim et al. developed an ambulance tracking system consisting of three main components (Fig. 4): The Ambulance System, The Control Maps Server, and The Center Monitoring System [8]. The Control Maps Server acts as a bridge between the Ambulance System or the Ambulance Location Monitoring System and the Google Maps Open API so that it will retrieve map data once from the Google Maps API and send it to the other components, instead of each component requesting data separately from the Google Maps Open API, which will increase the costs and not guarantee synchronization. The Ambulance System is equipped with GPS receivers to get the coordinates (Latitude and Longitude) of the ambulance from the satellite and a wireless modem to transmit them to the Center Monitoring System. It also had a portable Ultra Mobile PC (UMPC) for ambulance drivers to access navigation data. Lastly, the Center Monitoring System provides medical staff with real-time tracking of ambulance locations, which provides the possibility to give pre-hospital services to emergency patients if needed.
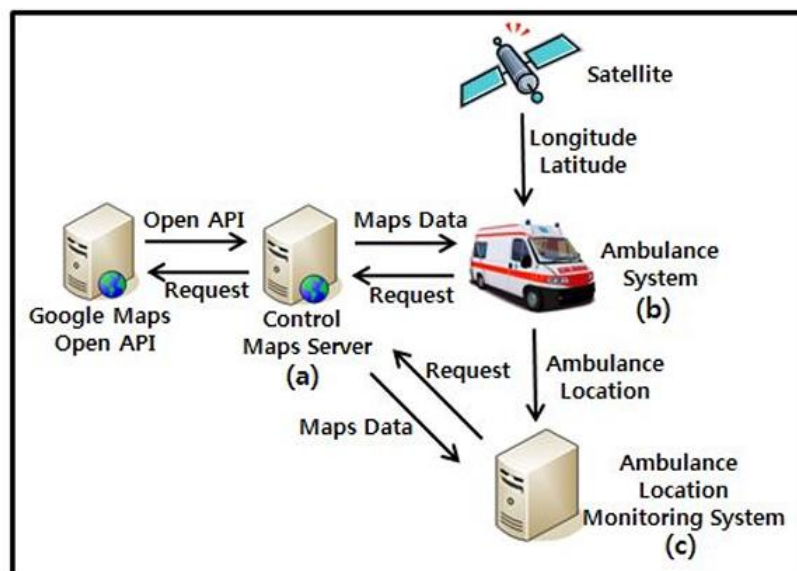
Figure 4: The proposed three-part system architecture [8]

Polamreddy et al. designed an Android application to help traffic cops manage ambulance routes using GPS tracking [9]. On the hardware, the system consists of GPS modules, an emergency button, IoT models, and Arduino microcontrollers which will be installed in the ambulance.

Once, in an emergency, the emergency button is pressed, real-time location data will be transmitted to a central server and it will generate a force alarm, which adjacent traffic cops will receive even if their smartphones are locked, displaying, with the use of Google Maps API, the ambulance location and the routes that the ambulance will take. This way, traffic enforcers will be able to clear the path or remove the traffic jams on these routes in advance. The system also includes login authentication to ensure that only authorized personnel can monitor the ambulance's movements.

Rudramurthy et al. present a vehicle tracking system designed for smart cities [10]. This system is primarily used for tracking garbage collection vehicles and providing users updates on their location through the Google Maps API. On the hardware, a tracking device is installed in vehicles, equipped with a GPS module to get location data and a Global System For Mobile Communications (GSM) for transmitting the data to a centralized server. The server uses the Google Maps API to visualize the vehicle's position on a mobile application. The application includes features, such as triggering notifications when the garbage collection vehicles enter or exit a predefined area. A database stores historical data, and so can the system generate reports for analytics. Additionally, the system offers helpful features, such as reporting events and getting maintenance reminders or alerts in the case of excessive speed.

To address the issue of vehicle theft, Shah presents a vehicle tracking system to transmit location data to a web portal, where the vehicle's location is displayed on Google Maps API [11]. It is built using Arduino and the SIM908 module, which combines GPS and General Packet Radio Service (GPRS) for both location tracking and data transmission to a database for storage. The data can be displayed on a web portal using the Google Maps API, enabling users to track their vehicles. A notable feature is that the users can get on-demand the current location of the vehicle through Short Message Service (SMS) without having to log in to the website in case the internet is unavailable. The website is built using PHP, MySQL, JavaScript, Asynchronous JavaScript, and XML (AJAX) to ensure real-time updates without page reloads.

Shibghatullah et al. developed a vehicle tracking application (Fig. 5) that aims to minimize the waiting times for public transportation by calculating the estimated time of arrival using Google's Distance Matrix API, considering the current traffic conditions [12]. The architecture includes two primary applications: one for drivers and another for users [12]. The driver-side application is deployed on an Android device positioned in the vehicle, leveraging GPS to record real-time location data at intervals of one second. The coordinates will then be uploaded to a Firebase real-time database that ensures synchronized updates across the users. The user-side one is also an Android application that displays the real-time locations of the vehicles using the Google Maps API.

The estimated arrival times for each stop are also displayed. Users can also access schedules, that are stored in MySQL database, which is integrated with the application using PHP.
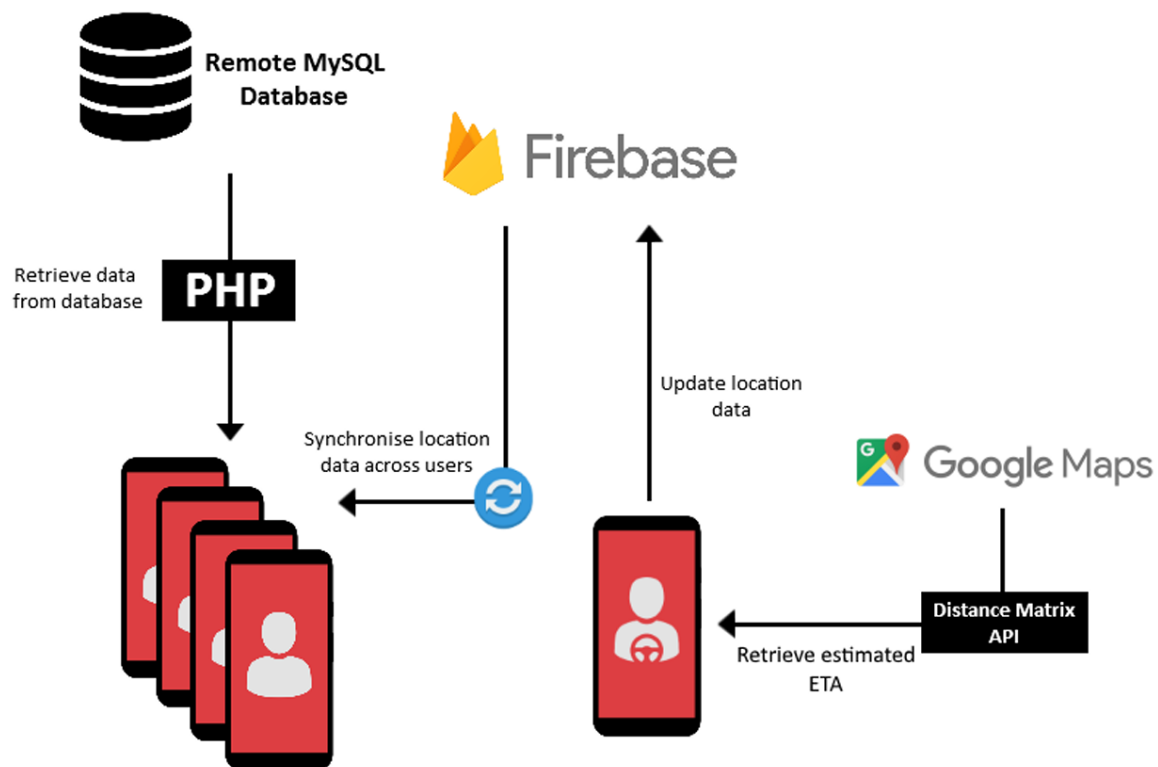
Figure 5: System architecture of Shibghatullah's vehicle tracking application [12]

2.2 International Standard Accident Number System

As introduced earlier, the ISAN platform [1] is designed to enhance emergency response by integrating multiple isolated ICT systems. It supports communication and coordination among the components of the rescue chain, including alerting, responding, and curing systems. An AS can be, for example, a smart wearable or a smart vehicle, each capable of identifying critical situations—such as a severe fall, a car crash, or a sudden health emergency—and automatically generating an alarm [2]. This alarm contains essential details, including the incident's location, time, and, if possible, medical data, which are transmitted to the ISAN platform to trigger a coordinated emergency response.

Once an alarm is sent, the RSM determines which RS unit is best suited to handle the incident. The selected RS unit then navigates to the scene, provides immediate medical assistance, and ensures patient transportation to the CS. The RS will be particularly relevant to this thesis, as in this component, real-time ambulance locations will be retrieved from Rescuetrack and later used for tracking.

At the core of the ISAN platform, the ISAN System manages communication and workflow execution. The WFM ensures that all rescue operations follow a structured and sequential order, preventing miscommunication. The ComM will play a crucial role in this thesis, as it will be responsible for retrieving real-time locations from all RSs that are responding to an emergency and forwarding them to all active CSs that are willing to track emergency operations.

These CSs consist of hospitals and emergency departments, where patients are transported and receive medical treatment after an emergency. The CS will be another important component in this thesis because it will play a crucial role in tracking emergency response data. It will serve as the visualization interface, displaying ambulance locations, real-time routes to incidents and hospitals, estimated arrival times, and the locations of both the incident and the assigned hospital.

2.3 Haversine Formula

When working with location-based systems, accurately calculating the distance between two points on the Earth's surface is fundamental. This is achieved using the Haversine Formula [13], a widely used mathematical approach for determining great-circle distances. This formula is useful in determining distances between two points over curved surfaces based on their latitude and longitude coordinates, ensuring accuracy for location-based calculations.

Step	Formula	Explanation
The Haversine Formula	$\mathrm{hav}(\theta) = \sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1) \cdot \cos(\phi_2) \cdot \sin^2\left(\frac{\Delta\lambda}{2}\right)$	Calculates the haversine of the central angle θ based on the differences in latitude ($\Delta\phi$) and longitude ($\Delta\lambda$).
Central Angle	$\theta = 2 \cdot \arcsin\left(\sqrt{\mathrm{hav}(\theta)}\right)$	Uses the result of the haversine function to compute the central angle between two points on a sphere.
Distance	$d = R \cdot \theta$, where R is the Earth's radius	Converts the central angle θ into a distance d using the radius R of the Earth (commonly approximated as 6371 km or 6371000 meters).

Table 1: Calculating Process Using the Haversine Formula

In this thesis, this formula will be utilized to calculate the distance between the ambulance's current location and its predefined route. If the ambulance deviates from this route, a new route will be dynamically calculated and drawn to ensure it follows the most efficient path to its destination.

Based on the insights from related works, the next sections focus on the foundational technologies, whose understanding is essential to accomplish the described tasks in this bachelor's thesis.

2.4 Application Programming Interface (API)

An API is a set of rules or protocols that enables different software applications to communicate with each other to exchange information [14]. It acts as a bridge between applications. It will allow developers to access specific functionalities or data without the need to understand how everything works inside the system. This interaction occurs through API endpoints, which are specific Uniform Resource Locators (URLs) that represent a functionality within the system. Developers will send requests to these endpoints and receive responses, typically in structured formats. The most common types of APIs are RESTful APIs and SOAP APIs, which we will cover in the next sections.

In this thesis, the Rescuetrack API will be utilized to retrieve real-time location data from the Rescuetrack device, while the MapQuest API provides the mapping and routing services to visualize ambulance movements.

2.5 Representational State Transfer (REST)

REST is an architectural style for providing standards between computer systems on the web, making it easier for systems to communicate with each other [15]. It was first introduced by Fielding in his dissertation [16]. One of the key features of REST is statelessness, which means that the server and client don't need to know about each other's status. This design allows the server and client to operate and be implemented independently, which improves flexibility and scalability.

REST is commonly used to design APIs, known as RESTful APIs. These APIs follow REST principles to enable communication between clients and servers using standard Hypertext Transfer Protocol (HTTP) methods: GET to get data, POST to send data, PUT to modify data, and DELETE to delete data. Each resource will be identified by a unique URL, which acts as a unique address for specific functionalities or data. For example, an endpoint like http://example/route?from={start}&to{end} might represent a service for calculating routes, and sending a GET request to this endpoint would retrieve the route coordinates from the start to the endpoint.

The data exchanged between clients and servers in RESTful systems is typically structured using lightweight formats such as Javascript Object Notation (JSON). JSON is a simple format to organize data. It uses key-value pairs (e.g. "address": "Braunschweig") to make information easy to read by both humans and machines.

2.6 MapQuest Developer Platform

The MapQuest Developer platform [17] offers a suite of RESTful APIs and Software Development Kits (SDKs) that provide advanced geospatial solutions for developers. Key offerings include the Geocoding API, which enables both the conversion of addresses to geographic coordinates and geographic coordinates back to addresses and the Directions API, which calculates routes between locations. To utilize these services, developers must obtain an API key by signing up on the MapQuest Developer portal. This API key allows the initialization of MapQuest map-object with specified parameters such as center coordinates. These tools enable developers to integrate mapping, routing, and location-based functionalities into their applications.

2.7 Simple Object Access Protocol (SOAP)

SOAP is a lightweight protocol for the exchange of information in a decentralized, distributed environment. [18]. SOAP follows a more rigid set of standards and relies only on Extensible Markup Language (XML) for formatting its messages. Each message is wrapped through an envelope as a container for the whole content, including optional headers for metadata and a mandatory body that has the actual data.

Unlike RESTful APIs which are limited only to HTTP as their communication protocol SOAP can operate over various transport protocols such as HTTP, and Transmission Control Protocol (TCP), which provides flexibility for systems that require secure communications. In addition, SOAP includes built-in error-handling mechanisms that make it more suitable for use cases where reliability is critical.

In the context of this thesis, SOAP is utilized by the Rescuetrack API to provide real-time location data for the ambulances, which ensures high reliability.

2.8 Rescuetrack

Rescuetrack [19] is a telematics and fleet management system designed to help emergency services route and manage their vehicles more efficiently. It allows real-time monitoring of vehicle locations and provides tools to improve communication and coordination during emergencies. Its primary goal is to optimize resource allocation and improve response times during critical situations. The system uses devices like

the RDG1110 communication unit, which connects vehicles to the BOS-Private-Cloud using dual SIM cards for reliable connectivity. For optimal signal reception, Rescuetrack provides antennas such as the combined LTE and GNSS roof-mounted antenna for vehicle roof installation and the adhesive antenna for windshield mounting. The system supports integration with third-party software through its SOAP API interface. Among the provided API functionalities, the "GetPositions" service enables the retrieval of location data for emergency vehicles, which is vital for monitoring and routing.

2.9 Flask

Flask is a web framework for Python to simplify the development of web applications [20]. It provides essential tools such as routing, and request handling. One of Flask's key features is its use of Jinja2, a tool that helps create dynamic HTML pages.
It allows developers to combine Python code with HTML templates, which makes it easier to display data from the backend on the website.

In the ISAN platform, Flask is utilized as the backend framework to manage communication between various components. It handles HTTP requests from the client, processes data retrieved from APIs like Rescuetrack, interacts with the database to store and retrieve information, and sends responses back to the client.

2.10 Docker

Docker is a software platform that makes it possible to package applications and their dependencies into containers [21]. These containers are lightweight, portable units that include everything needed to run an application across different environments.

In the ISAN platform, Docker is used to containerize various components. These components are divided into separate containers, such as the AS, RS, and CS, along with their corresponding managers. This setup allows us to simulate a fully integrated system by running each component in its isolated environment, while still enabling communication between them. Each container can work independently, making it easier to test, and deploy individual components without affecting the others.

2.11 HyperText Markup Language (HTML)

HTML is the standard markup language for creating Web pages [22]. It defines elements such as paragraphs, links, and images which form the building blocks of a webpage.

In the ISAN platform, HTML is used to design the interface for components like the CS, allowing users to interact with buttons and other elements displayed. These interactions send actions to the backend, which enables the system to process them and provide dynamic responses

2.12 Cascading Style Sheets (CSS)

CSS is a style sheet language used in web development to control the presentation and formatting of HTML documents [23]. It allows control of the layout, colors, and overall design of elements defined in HTML.

In the ISAN platform, CSS is used to enhance the user interface of components such as the CS.

2.13 JavaScript

JavaScript is a lightweight programming language used to add dynamic interactions to web pages [24]. Unlike HTML and CSS, which define structure and style, Javascript enables functionality such as real-time updates and animations.

In this thesis, Javascript will be utilized to dynamically update the ambulance markers and routes on the map as the ambulances move in real-time.

2.14 Document Object Model (DOM)

The DOM is an API for web documents that represents the structure of a web page as a tree of objects [25]. Each element, attribute, and text in the document corresponds to a node in the DOM tree, to enable dynamic editing and customization of the document's content, arrangement, and visual appearance. It acts as a bridge between HTML and JavaScript. While HTML defines the static content and structure of a webpage, the DOM provides the dynamic capabilities to update, delete, or add elements at run time through JavaScript functions.

In this thesis, the DOM will be utilized to dynamically manage the Remaining Time Table, including generating new rows for ambulances assigned to emergencies and updating the remaining time for tracked ambulances in operation.

2.15 WebSockets and Socket.IO

Websockets is a communication protocol that provides full-duplex communications between the client and server [26]. It maintains a persistent connection, enabling real-time data exchange with less latency. This makes WebSockets ideal for applications that require continuous updates, such as live tracking.

Socket.io is a library built on top of WebSockets and offers additional features such as event-based communication and automatic reconnection [27].

In this thesis, Flask-SocketIO will be used to allow continuous transmission of ambulance tracking data from the backend to the frontend, allowing the map to dynamically update in real-time.
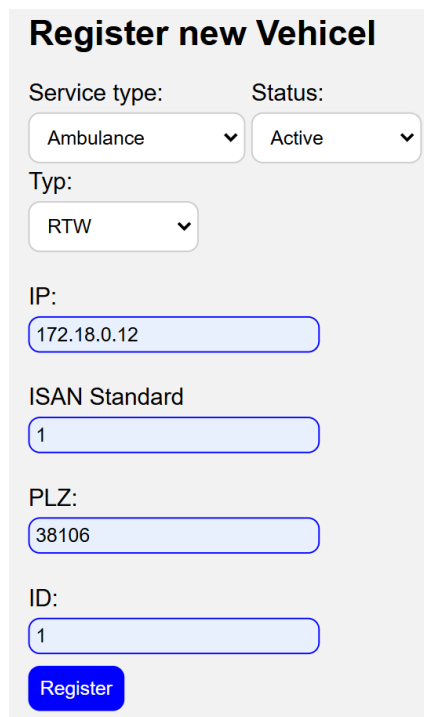
3 Materials and Methods

This chapter focuses on the implementation, algorithms, and data used to fulfill the tasks of this thesis. The following sections describe the process of creating tracking simulation scenarios, delve into the technical implementation of the integration workflows for both real-time ambulance tracking and simulated tracking scenarios, emergency vehicle rerouting implementation, and explore the visualization implementation used to display tracking data.

3.1 Creation of Simulated Tracking Scenarios

Simulated tracking scenarios provide a controlled environment for testing the ISAN platform's functionality, allowing different emergency situations to be created, visualized, and evaluated. This includes simulating ambulance movements in real-time, displaying their routes, estimating arrival times, and even testing scenarios such as ambulance breakdowns. Before diving into the technical implementation, this section outlines the process of creating such scenarios.

First, we need to create as many RS containers as required for the simulation. Then, for each RS localhost (Fig. 6), an active ambulance must be registered and assigned the internal IP address of its respective container. These active ambulances from different RS localhosts collectively form the pool of available ambulances, ready to respond to simulated alarms.

Figure 6: RS Localhost [1]: Register new Vehicle

Next, simulated alarms are generated through the AS localhost (Fig. 7), accessible at http://localhost:5555/. Each alarm represents a specific incident and includes details such as the incident location. Before sending an alarm, we have to make sure, that we have already created an active ambulance with the same responsibility area (PLZ) as the postcode of the generated alarm. Once an alarm is sent, the ISAN platform dynamically assigns an available ambulance to the generated ISAN.

System settings
Provided information:
☐ ECG ☐ Floorplan ☐ Doorcode ☐ Rescue card ☑ DAR ☐ Video ☐ Audio ☐ ICD-11

Invoke alarm
Generate ISAN:
Postcode: 38106
City: Braunschweig
Street: Mühlenpfordtstraße
Street number: 23
Generate
Send Alarm:
1|2025-02-22T16:39:26+00:00|1|00:00:52|2|38106^Germany^^Braunschweig^^Mühlenpfordtstraße^^23^^^442^DE|0|50|1|+1|0|0|1|37:12:C2:BE:F1:39|
Alarm

Figure 7: AS Localhost [1]: Send Alarm

After an ambulance is assigned to an alarm, a hospital must be manually assigned to the ISAN (Fig. 8). It will be the destination where the victim will be transported after the ambulance reaches the incident location.

Curing Systems

Klinikum Celler Straße ⌄

2025-02-22 | 16:31:52 at 38106 Germany Braunschweig Mühlenpfordtstraße 23 | 442 DE ⌄

Send ISAN to CS

Figure 8: RS Localhost [1]: Send ISAN to CS

With these steps complete, the simulation is now fully configured. By opening the CS tracking page, accessible at http://localhost:5557/, the simulation can be visualized. The ambulances are shown driving from a random start position in Braunschweig's center to the incident location (Fig. 9) and then transporting the victim to the assigned hospital (Fig. 10). The map dynamically updates the ambulances' movements, routes, and estimated arrival times, providing a realistic representation of an emergency response.

Figure 9: Ambulance to incident
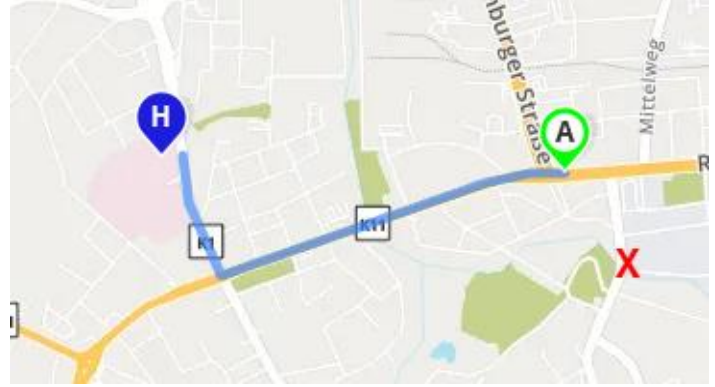
Figure 10: Ambulance to hospital

If we wish to simulate breaking down an ambulance. The "Breakdown !" button (Fig. 11) corresponding to the ambulance can be used. The reassignment of a new ambulance will only occur if another ambulance is available.

Figure 11: Breakdown button

Lastly, to start a new simulation with new incident locations, the "Delete Simulation Data" button (Fig. 12) can be used. It will clear all data associated with the previous simulation, which will provide a clean state for creating and running new scenarios.

Figure 12: Delete simulation data button

3.2 Integration of Tracking Workflows into the ISAN Platform

The workflows for integrating real-time ambulance tracking and simulated tracking scenarios into the ISAN Platform share a significant portion of their logic, with specific differences to fulfill their respective use cases. This section introduces the technical implementation of both variants. For clarity, "practical" will refer to real-time ambulance tracking, while "simulation" will refer to the simulated tracking.

3.2.1 Integration During Ambulance Assignment

The Sequence Diagram (Fig. 13) outlines the original workflow for handling alarms within the ISAN platform. It illustrates the communication between the components, triggered when an ISAN is sent by the AS to the WFM. This workflow verifies the alarm, assigns it to an available ambulance, and records the relevant data. For the detailed algorithm, refer to Algorithm 1 in the Appendix.

The modifications implemented are highlighted in red: The ambulance status will be changed from "active" to "occupied upon assignment, which ensures that one ambulance will not be assigned multiple emergencies at the same time. The ambulance ID will also be included in the communication. Finally, the RS initiates a separate thread to execute the appropriate function based on the tracking case.
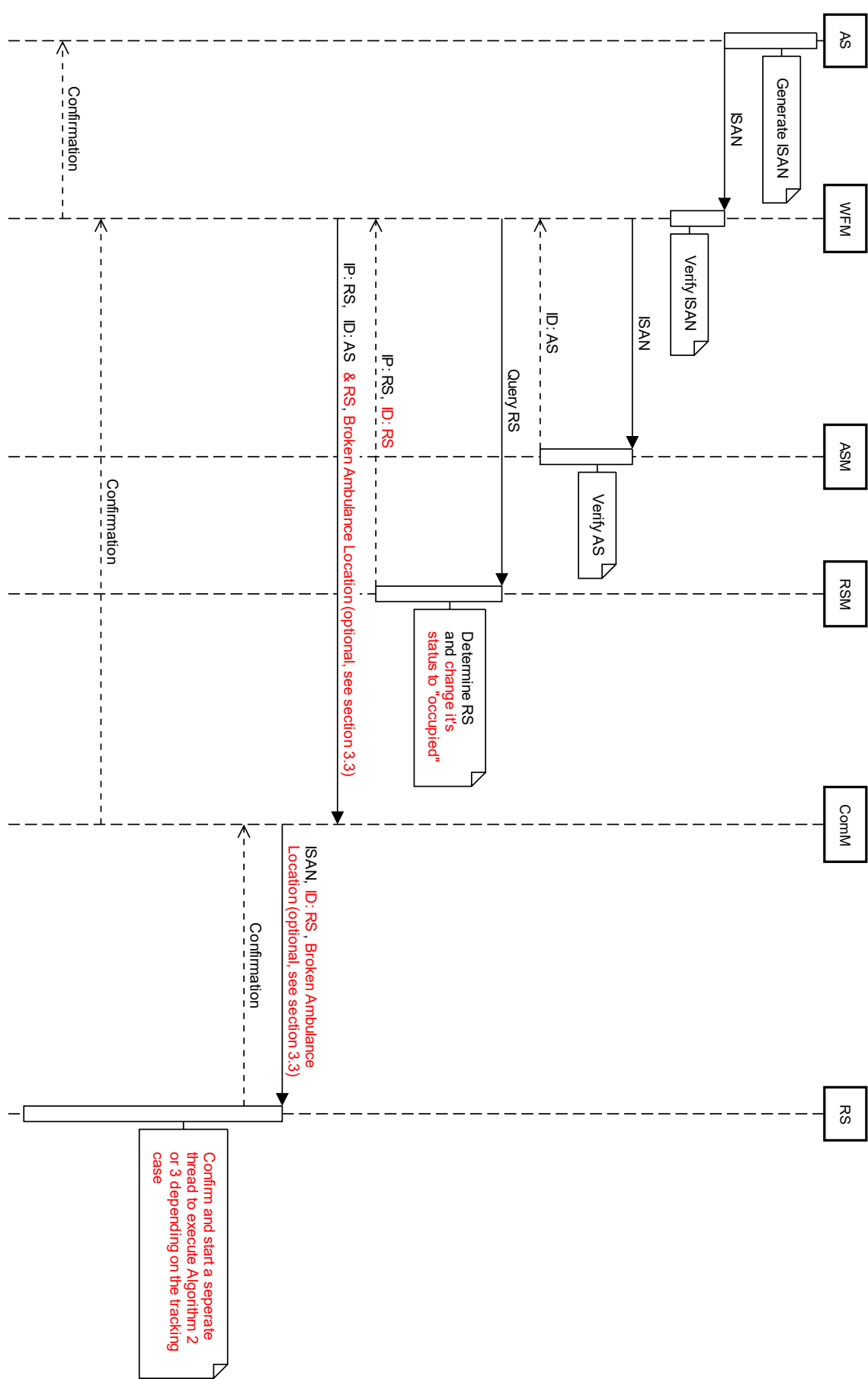
Figure 13: Sequence diagram to algorithm 1: Handle Alarm

In the practical case, the "Request Coordinates from the Rescuetrack" function (illustrated in Figure 14) is executed, which retrieves live GPS coordinates of the ambulance by sending SOAP request (Code Snippet 1) to the Rescuetrack API, inserts them into the database, and then prepares the tracking data in the appropriate format to be retrieved. This process continues automatically, with each request retrieving the latest ambulance's coordinates, until the patient arrives at the hospital.
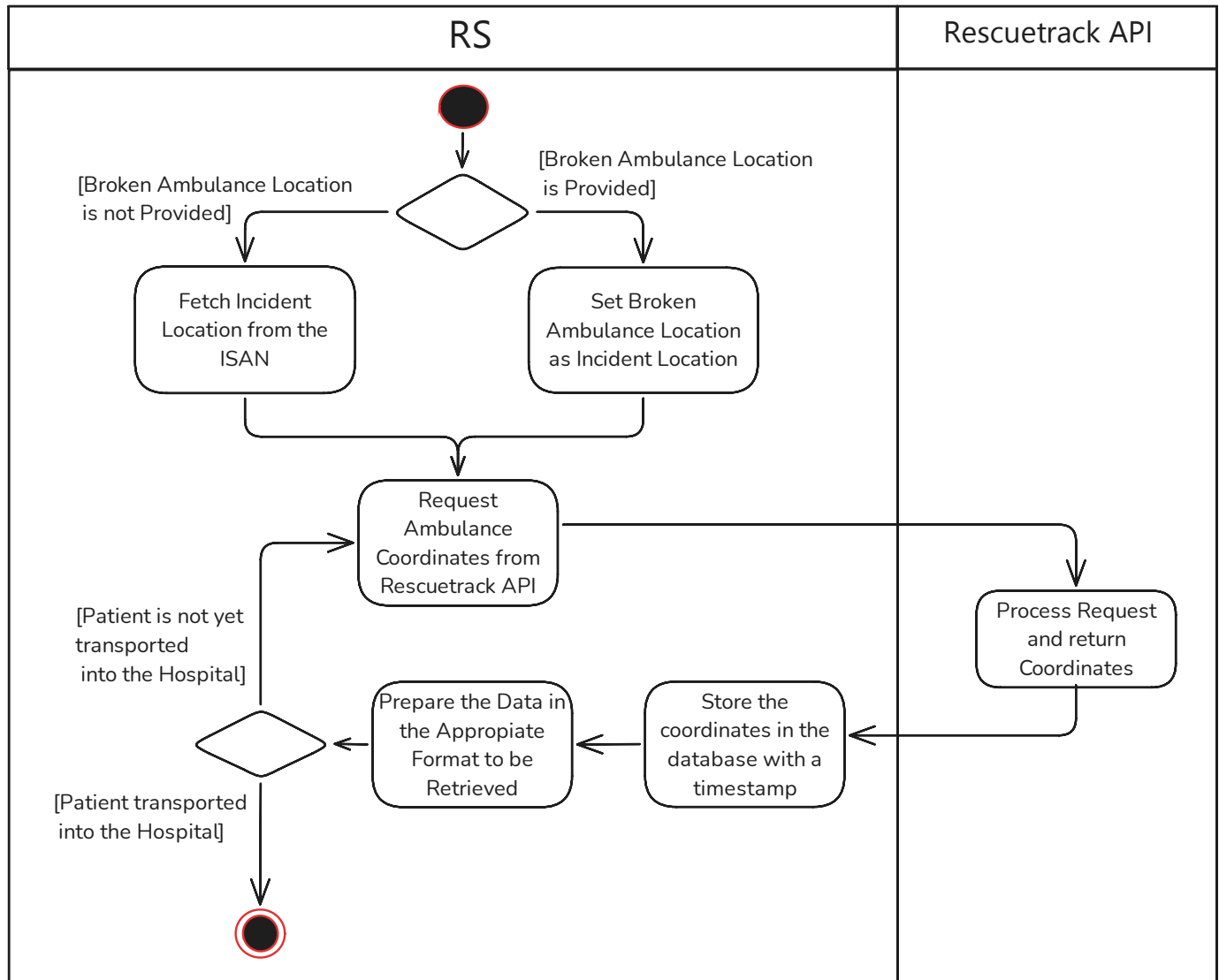
Figure 14: Activity diagram to algorithm 2: Request   Coordinates   from   the
Rescuetrack

Code Snippet 1: SOAP Request sent to the Rescuetrack API

```
url = "https://api.rescuetrack.de/ws/v4/rescuetrack.asmx"
headers = {
    "Content-Type": "text/xml; charset=utf-8",
    "SOAPAction": "http://www.rescuetrack.de/GetPositions"
}
data = f"""<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <GetPositions xmlns="http://www.rescuetrack.de/">
            <username>username</username>
            <password>password</password>
            <lastPositionId>0</lastPositionId>
            <options>NoOptions</options>
        </GetPositions>
    </soap:Body>
</soap:Envelope>"""
```

The tracking data includes the ambulance's latitude and longitude, along with the incident's location if the patient has not yet been loaded into the ambulance. Once the patient is loaded, the tracking data replaces the incident location with the hospital location. Finally, when the patient is transported to the hospital, the tracking data contains only the ambulance's location and a confirmation message indicating that the transport is complete. For more details on the implementation, see Algorithm 2 in the Appendix.

The conditions for determining the status of the patient —whether the patient is not yet loaded into the ambulance, in the ambulance, or transported to the hospital—are evaluated manually. At the start of the rescue operation, the patient's status is set to "Not Yet Loaded Into Ambulance". When the patient is loaded into the ambulance, the responder must click the first button (Fig. 15), updating the status accordingly. Similarly, once the patient is transferred to the hospital, the second button (Fig. 15) is pressed to confirm the transportation of the patient into the hospital, so that the ambulance can be used for other emergencies. To minimize errors during such critical tasks, the system prompts a confirmation message each time a button is clicked.

For instance, if the "Patient Loaded into Ambulance" button is clicked, a message appears asking, "Are you sure that the patient is loaded into the ambulance ?". This additional step ensures the accuracy of status updates and prevents accidental changes.
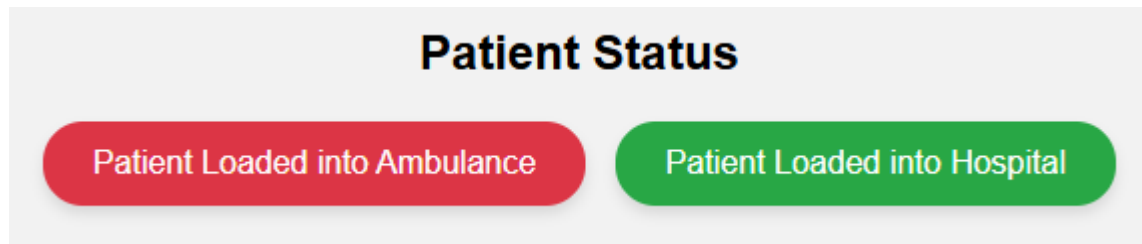
Figure 15: Buttons for updating patient status during rescue operations

In the simulation case, the "Simulation Write Route To Incident" function (illustrated in Figure 16) is executed, which selects first a random starting position from a predefined set of locations within Braunschweig. The incident location is then extracted from the ISAN, and a GET request is sent to the MapQuest API to retrieve a set of coordinates forming the route from the start position to the incident location. These coordinates, consisting of latitude and longitude pairs, are then written into a CSV file named simulation_coordinates.csv. Once the route data is saved, the process terminates. For further details on the implementation, refer to Algorithm 3 in the Appendix.
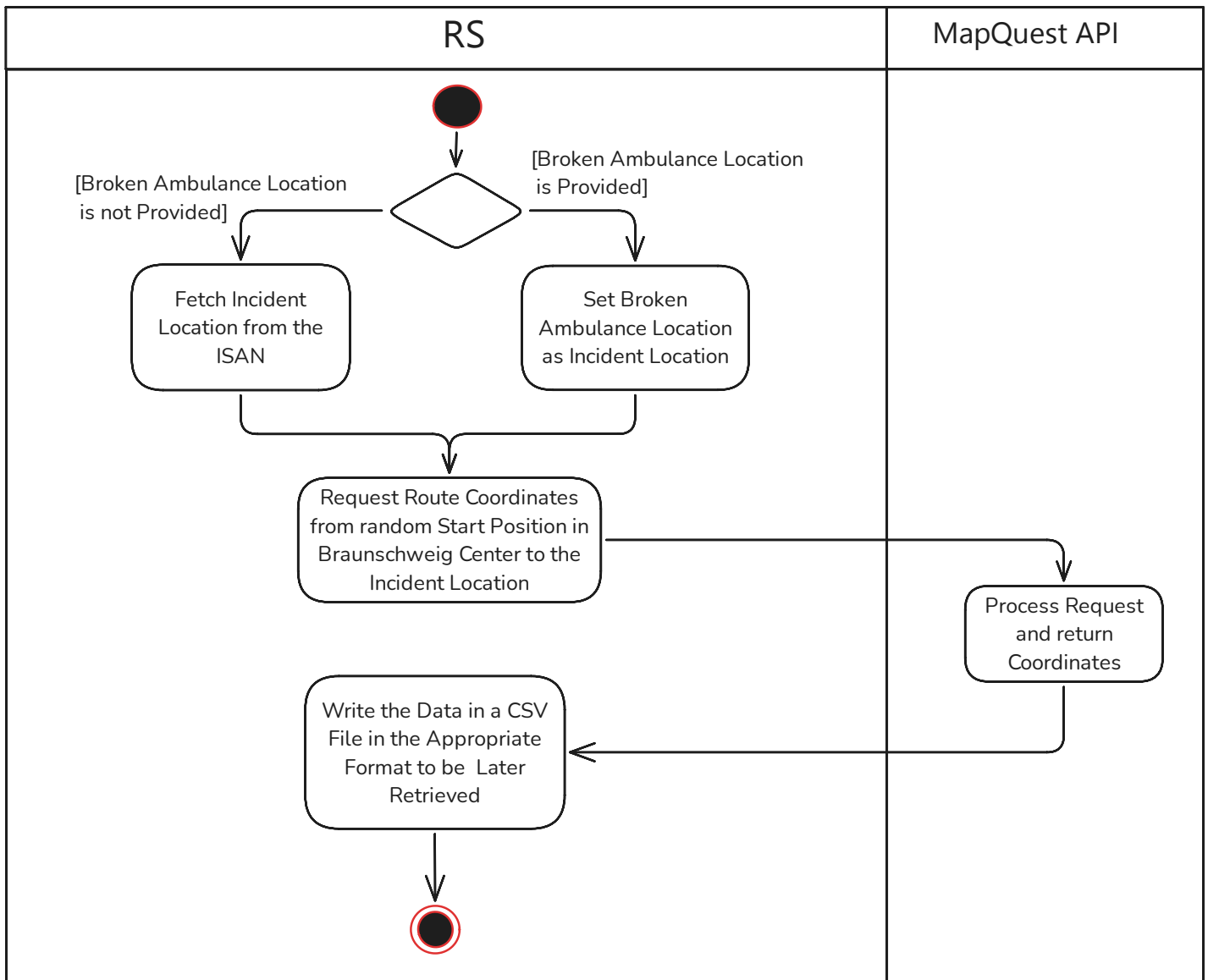
Figure 16: Activity diagram to algorithm 3: Simulation Write Route to Incident

3.2.2 Integration During Hospital Assignment

When an ISAN gets an ambulance assigned, the next step is to assign a hospital. The Sequence Diagram (Fig. 17) outlines the original workflow of assigning a hospital to an ISAN, with the modification implemented highlighted in red. Before sending the assigning request to the WFM, the RS initiates a separate thread to execute the appropriate function based on the tracking case.
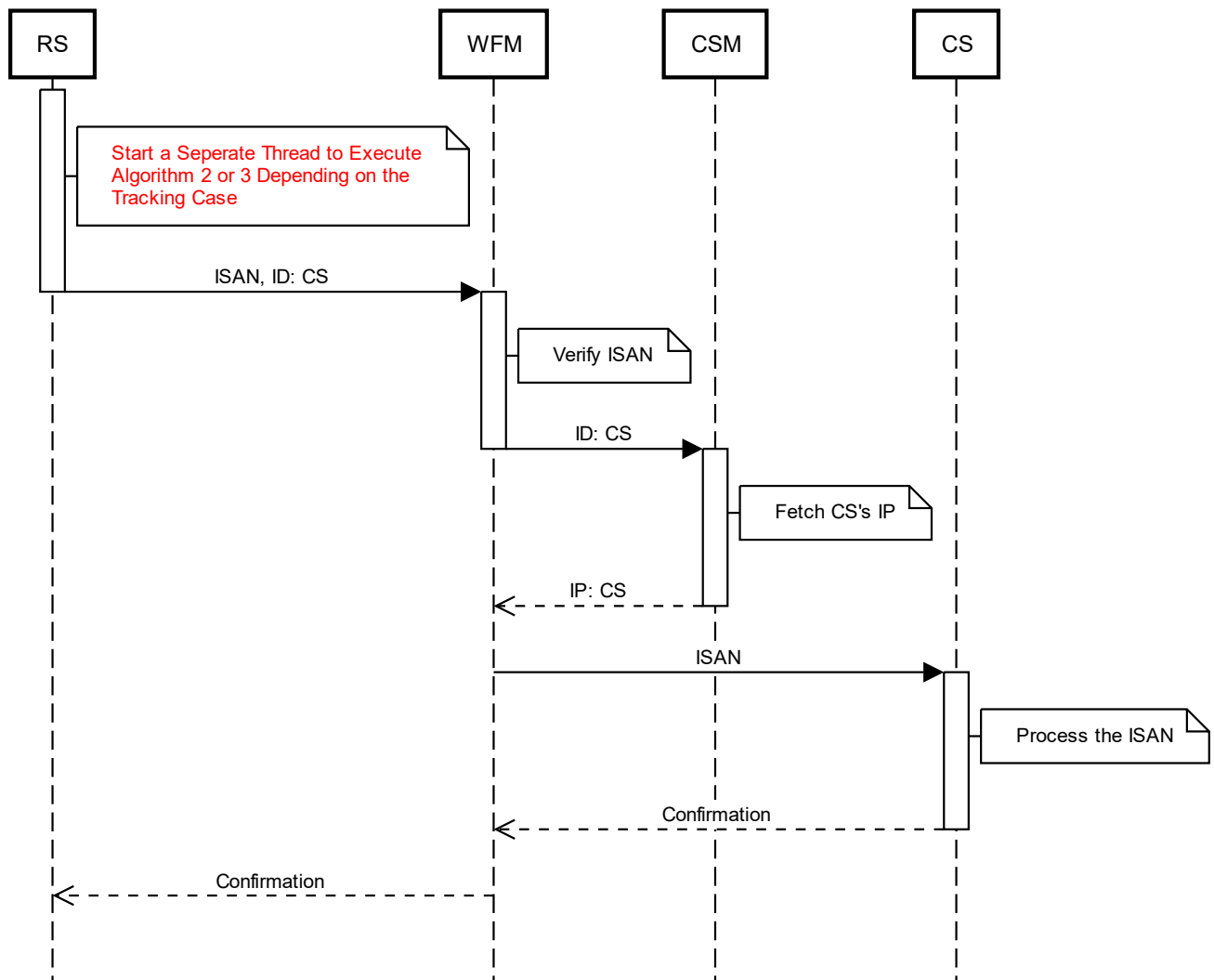
Figure 17: Sequence diagram to algorithm 4: Send ISAN to CS

In the practical case, the "Set Hospital Location" function (Algorithm 5) is executed, which simply sets the value of the hospital location for the responding system.

In the simulation case, the "Simulation Write Route To Hospital" function (illustrated in Figure 18) is executed, which appends route data to the existing CSV file created when the alarm was generated. The incident location is first extracted from the file, and a GET request is sent to the MapQuest API to retrieve a set of coordinates forming the route from the incident location to the hospital location. The retrieved GPS coordinates, consisting of latitude and longitude pairs, are then appended to the CSV file, maintaining continuity in the simulated route. Once the new coordinates are appended to the file, the process terminates, making the overall route coordinates ready for simulation. For more details on the implementation, refer to Algorithm 6 in the Appendix.
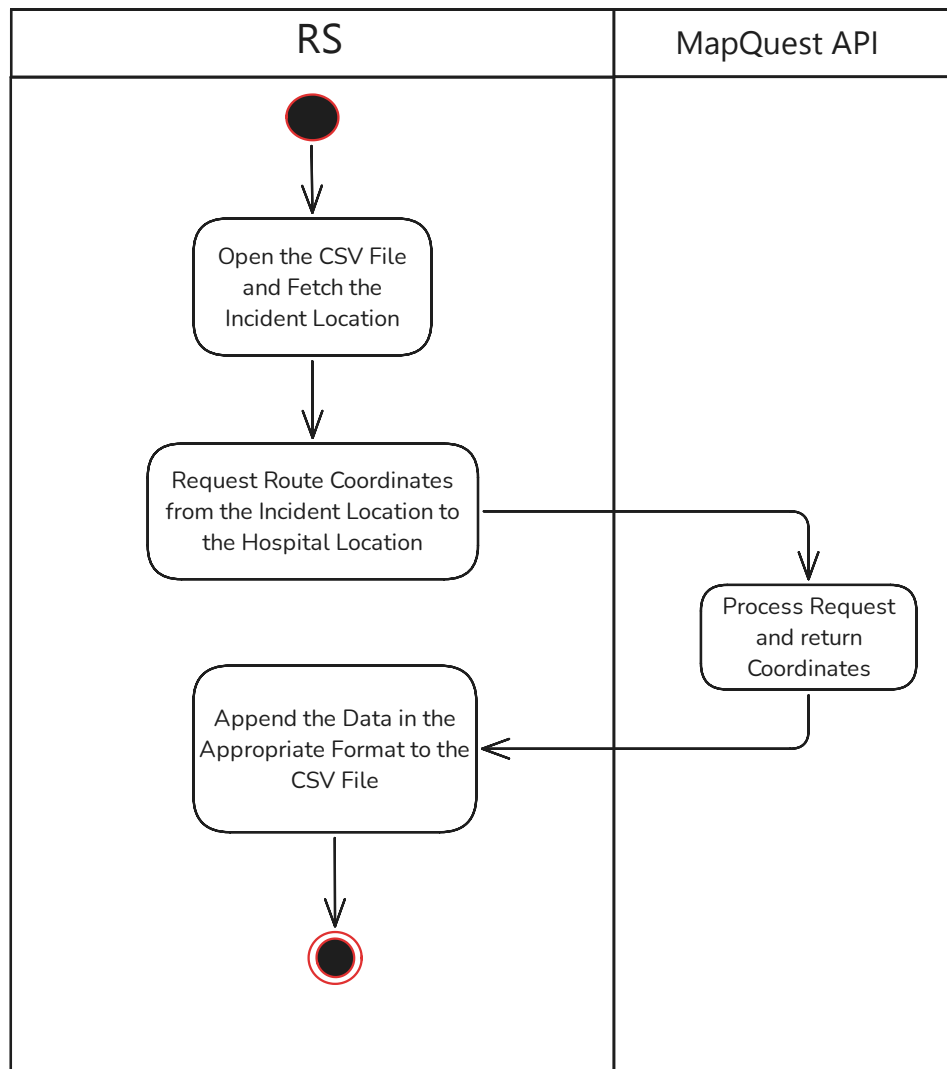
Figure 18: Activity diagram to algorithm 6: Simulation Write Route To Hospital

3.2.3 Integration of Tracking Data Transmission

Finally, we come to the last integration step, which involves the workflow for transmitting tracking data from the RS to the CS, where the data will be displayed. It will start when the tracking page on the CS host opens, which will trigger two events on the client side via Socket.IO:

1) Requesting the Main Ambulance ID: The first event requests the main ambulance ID. This corresponds to the ID of the ambulance to the specific ISAN selected for tracking when the user opens the CS page. All the tracking data related to this ISAN will be highlighted with a distinct color on the map to help operators easily identify the tracked ambulance among other ambulances.

2) Requesting Tracking Data: The second event initiates the process of getting tracking data from the RSs currently in operation. This ensures continuous updates about the ambulance's necessary visualization data.

The first event in the tracking process involves the CS backend requesting the main ambulance ID associated with a given ISAN. The accompanying sequence diagram (Fig. 19) illustrates the communication flow between the components. The CS backend initiates the request by sending the ISAN to the WFM. After performing the necessary verification, the RSM broadcasts the ISAN to all occupied RSs, instructing them to check their alarm list tables. The ISAN is then matched by the correct RS, which confirms that it is handling the emergency. The RSM retrieves the ID of the confirming RS and sends it to the ComM, which forwards the ID to the CS backend that originally requested it. The CS backend then emits the ID to the frontend tracking page. For further details on the implementation, refer to Algorithm 7 in the Appendix.
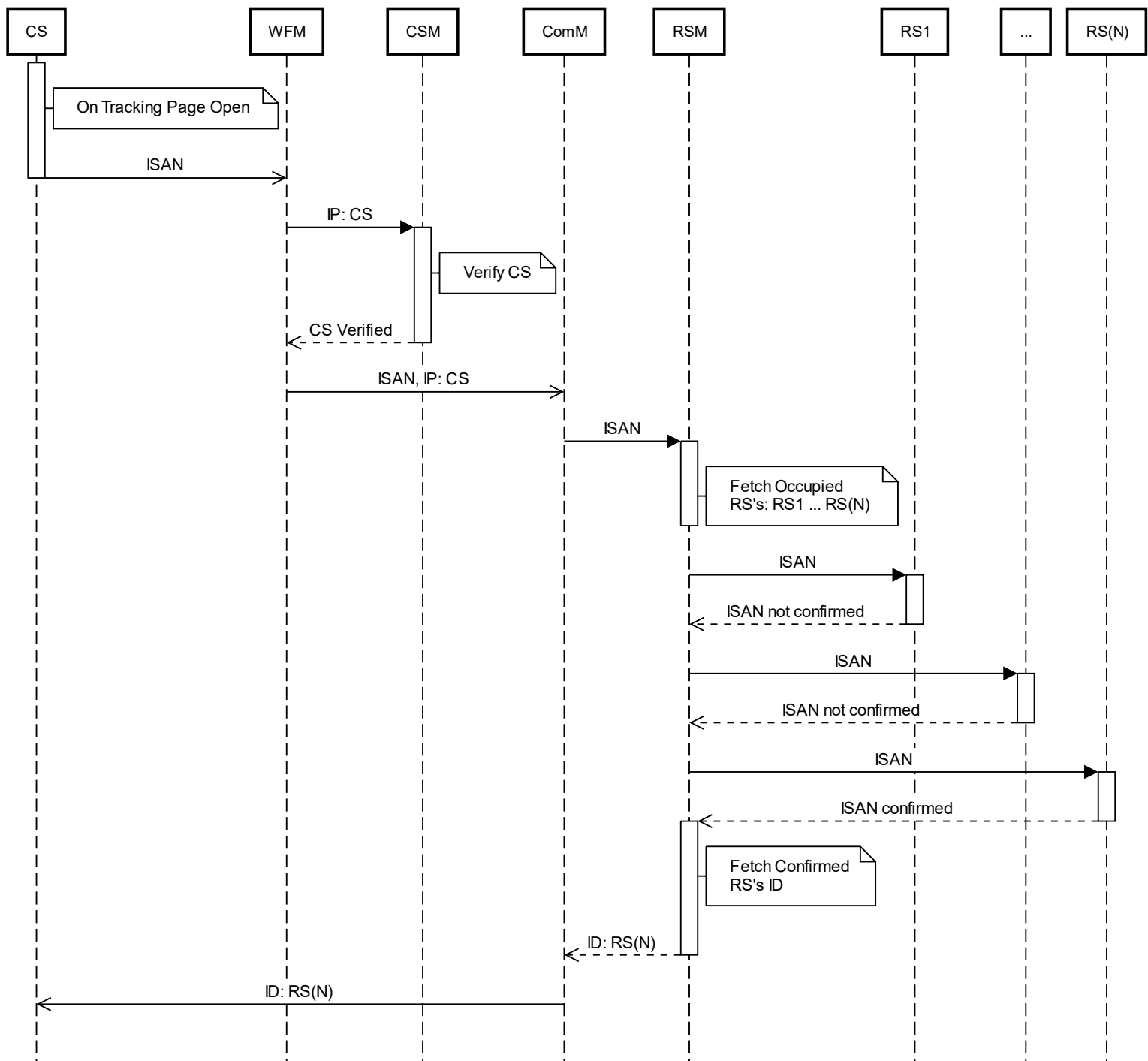
Figure 19: Sequence diagram to algorithm 7: Curing System Requests the Main Ambulance ID

Building upon the successful assignment of the main ambulance ID, the sequence diagram (Fig. 20) outlines the process by which the CS backend requests the ambulance coordinates in the practical case. For more details on the implementation, refer to Algorithm 8 in the Appendix.

In the practical case, once a CS is verified, its IP is added to the active CS list managed by the ComM. If this is the first CS to request tracking data, the ComM initiates two separate threads. For any additional CS instances, the threads remain active and are not re-initiated. The first thread periodically retrieves the IPs of the active RS instances every 5 seconds (illustrated in Figure 21; refer to Algorithm 9 in the Appendix for more details). The second thread, running in parallel, also executes every 5 seconds, requesting the latest coordinates from these RS instances using their IPs retrieved by the first thread. The second thread then aggregates the data into a unified set and forwards the tracking data to all active CS instances (illustrated in Figure 22; refer to Algorithm 10 in the Appendix for more details). When a CS leaves the tracking page, its IP is removed from the active CS list (see Algorithm 12 in the Appendix for more details). If all CS instances leave the tracking page, both threads are automatically terminated.

In the simulation case, the process follows a structure similar to the practical case, but with a key difference in how the second thread operates. The second thread (illustrated in Figure 23; refer to Algorithm 11 in the Appendix for more details) first sends POST requests to all RS instances retrieved by the first thread. Upon receiving a request, each RS starts a separate thread for the corresponding ambulance ID, which reads data line by line from its associated CSV file and stores the current row's data in a global variable. This initialization step is performed only once. Afterward, the second thread continuously retrieves the latest stored data from each RS instance. Finally, the retrieved tracking data is forwarded to the CS, with this process repeating every 5 seconds.
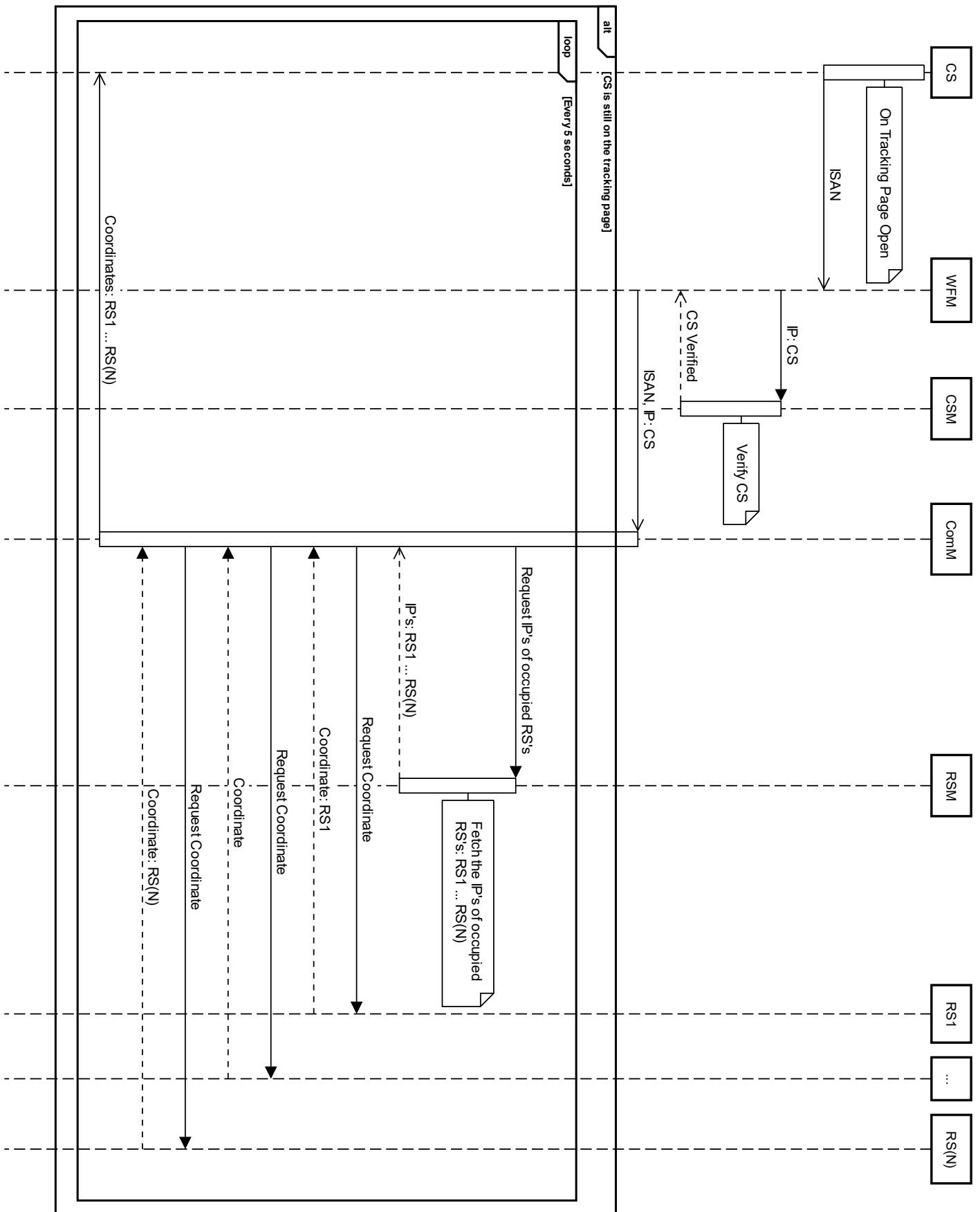
Figure 20: Sequence diagram to algorithm 8: Curing System Requests Ambulances Coordinates in the **practical** case

ComM	RSM

Request Occupied RS IPs from the RSM

Process Request and Return IPs

[One or more CS instances are still on the tracking page]

Update the List Contraining RS IPs

Sleep for 5 Seconds

[No CS instances are on the tracking page]
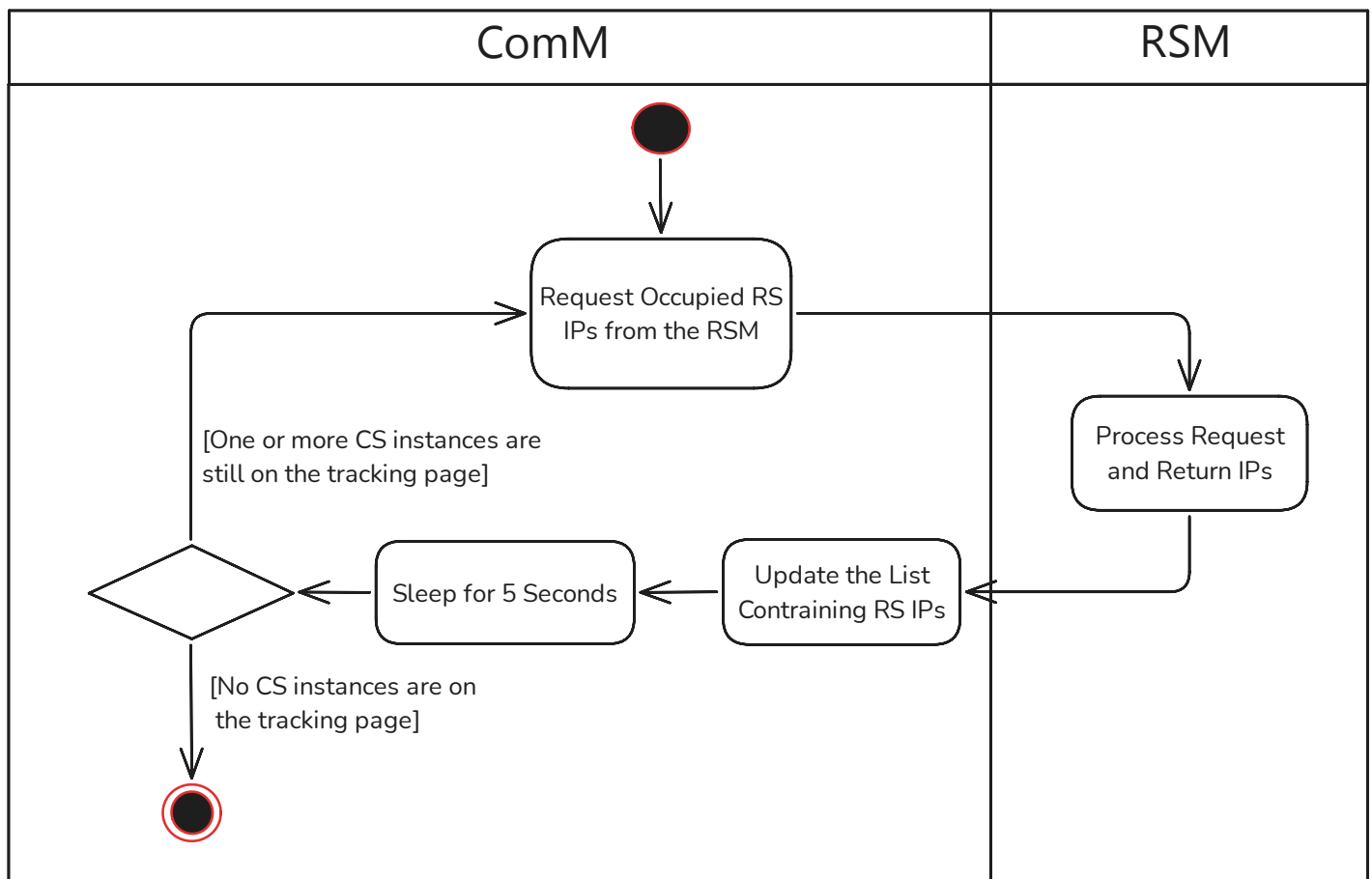
Figure 21: Activity diagram to algorithm 9: Periodic Request Occupied Ambulances Ips
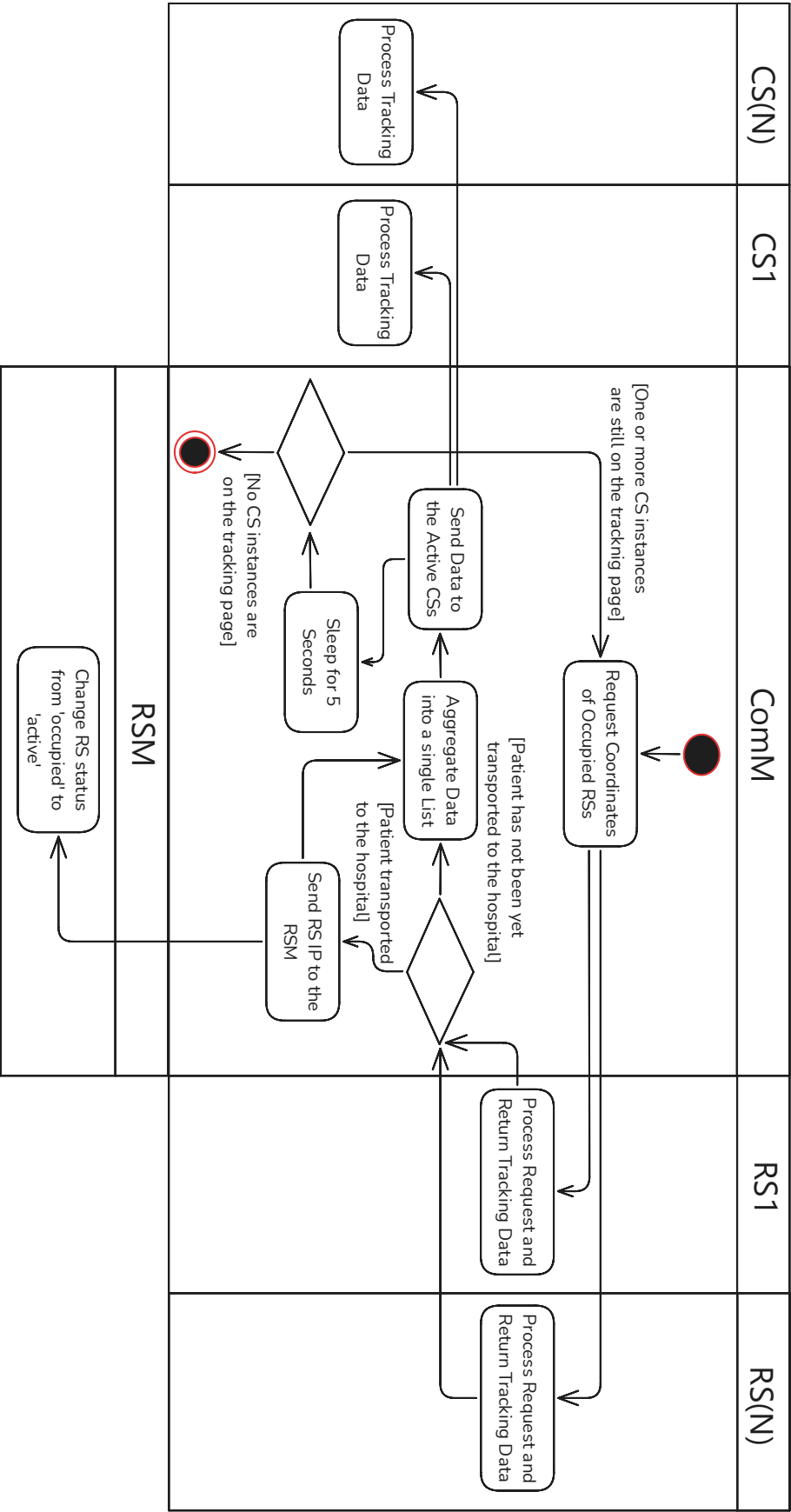
Figure 22: Activity diagram to algorithm 10: Practical Periodic Request Occupied
Ambulances Coordinates
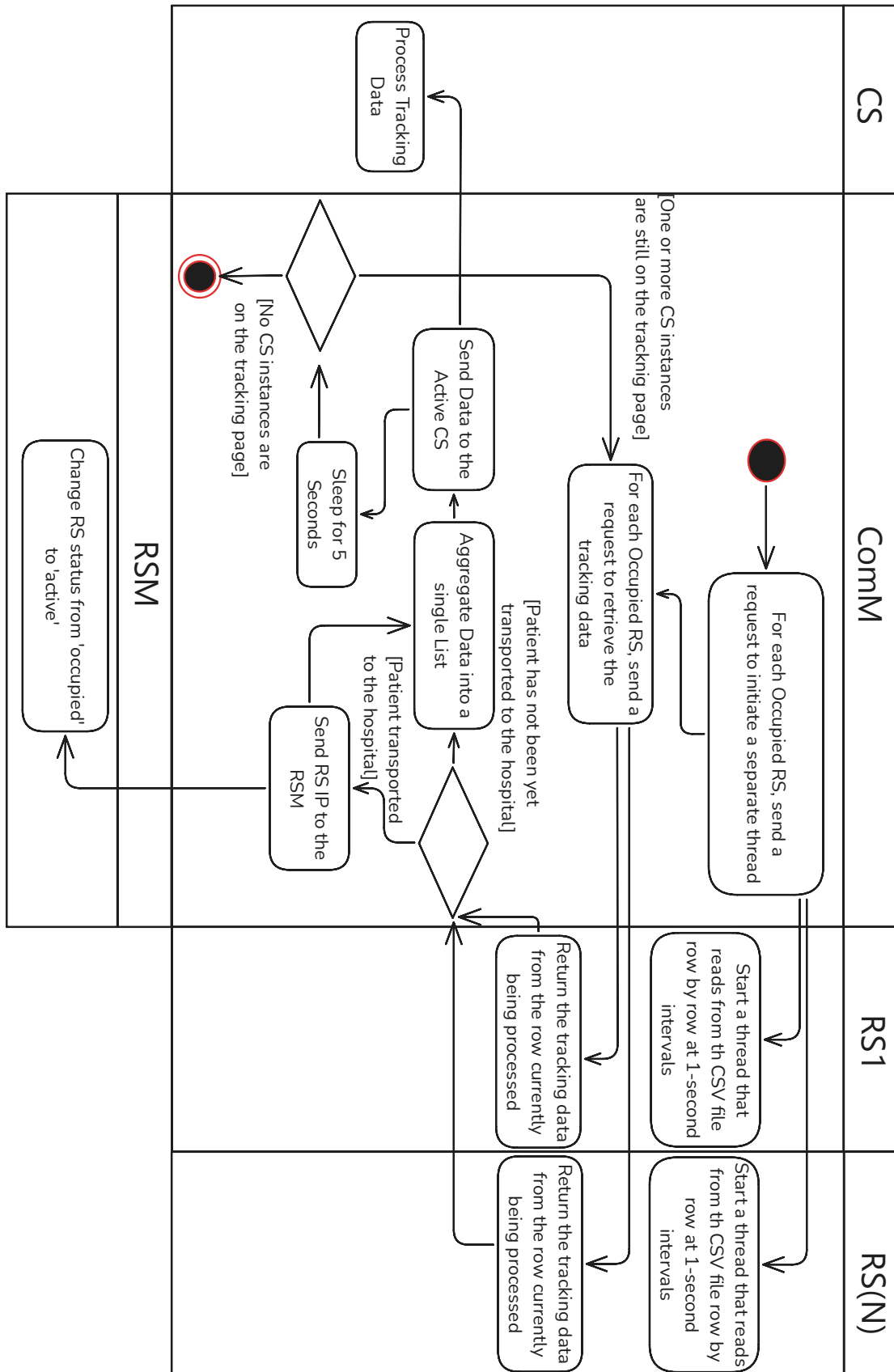
Figure 23: Activity diagram to algorithm 11: Simulation Periodic Request Occupied
Ambulances Coordinates

3.3 Emergency Vehicle Rerouting

This section details the implementation of the emergency vehicle rerouting functionality, to ensure continuous response capability and minimize delays in incident resolution in the event of an emergency vehicle breakdown.

When an ambulance breaks down while responding to an emergency, the Sequence Diagram (Fig. 24) outlines the rerouting process for assigning a new ambulance to the incident. The broken ambulance's status is first updated from "occupied" to "broken" in the RSM's responding systems table. If the breakdown occurs when the ambulance is still driving to the ISAN location, the broken ambulance resends the same ISAN to the WFM, reapplying the process described in Algorithm 1. The ComM updates the active communication table to assign the newly assigned RS to the ISAN, redirecting all further communication to the newly assigned RS. In this case, no modifications to existing algorithms (13, 1, 2, 3) are required, as the process remains identical. However, if the breakdown occurs while the ambulance is transporting the victim to the hospital, Algorithms 13, 1, 2, and 3 are modified to include an optional parameter representing the broken ambulance's location. The new RS is assigned the original ISAN, but instead of proceeding to the ISAN location, it is directed to the broken ambulance's location in the presence of this optional parameter. For more details on the implementation, refer to Algorithm 13 in the Appendix.
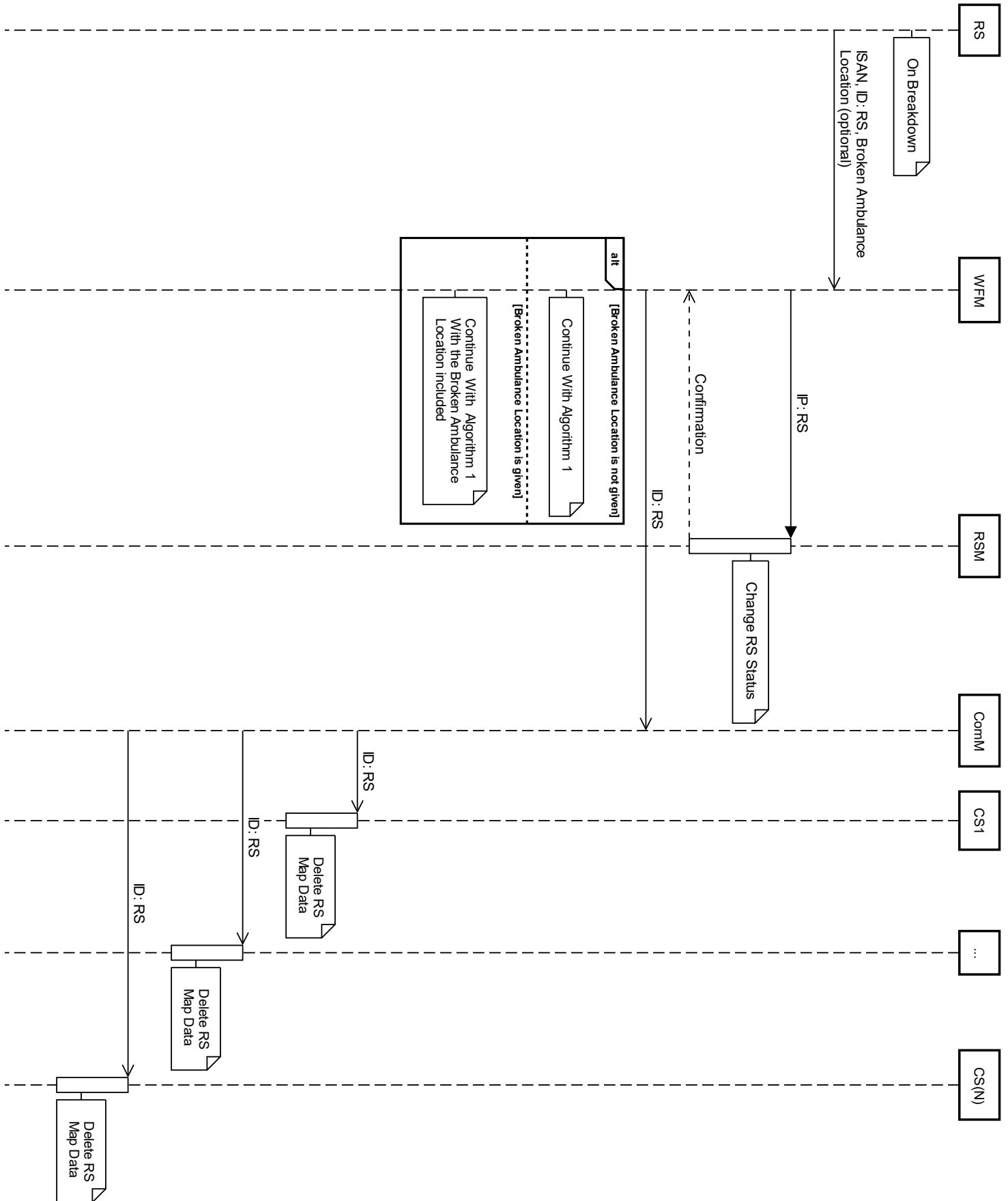
Figure 24: Sequence diagram to algorithm 13 in the **practical** case: Handle Breakdown

3.4 Map-Based Data Visualization

Having detailed the integration for managing and processing ambulance tracking data and rerouting, this section focuses on the frontend visualization of monitoring data.

As part of the map-based visualization process, three distinct types of tracking data (Fig. 25) will be handled, as described in Algorithm 2 for the practical case or in Algorithms 3 and 6 for the simulation case. Each type of tracking represents a specific stage in the emergency response. The first type, ambulance-to-incident data (a), visualizes the ambulance's journey from its current location to the incident location. The second type, ambulance-to-hospital data (b), monitors the ambulance's journey from the incident location to the hospital. Finally, confirmation data (c) indicates whether the ambulance has transported the victim to the hospital.

- ambulance ID - ambulance latitude - ambulance longitude - incident latitude - incident longitude - "incident location"	- ambulance ID - ambulance latitude - ambulance longitude - hospital latitude - hospital longitude - "hospital location"	- ambulance ID - ambulance latitude - ambulance longitude - "at hospital"
(a)	(b)	(c)
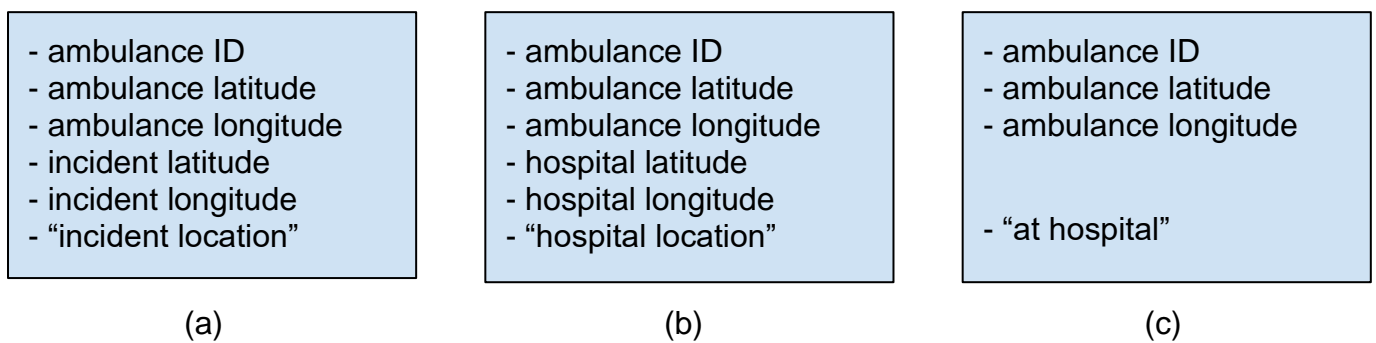
Figure 25: Types of tracking data

The activity diagram in Figure 26 illustrates the process of managing ambulance-to-incident tracking data. It begins by determining if an ambulance marker with the given ID already exists on the map. If it does not exist, new markers for both the ambulance and the incident at the specified coordinates will be created. Following this, a route is requested from the MapQuest API, from the ambulance location to the incident location. The route coordinates will be received in return and a polyline will be drawn for these coordinates. If the marker already exists, the ambulance marker's location will be updated with the new coordinates. Next, the distance to the nearest point on the existing route polyline will be calculated using the Haversine Formula. If the distance is less than or equal to 50 meters, the route polyline is sliced to begin at the nearest point. But if the distance is more than 50 meters, the current polyline is deleted, and a new route is calculated and drawn.
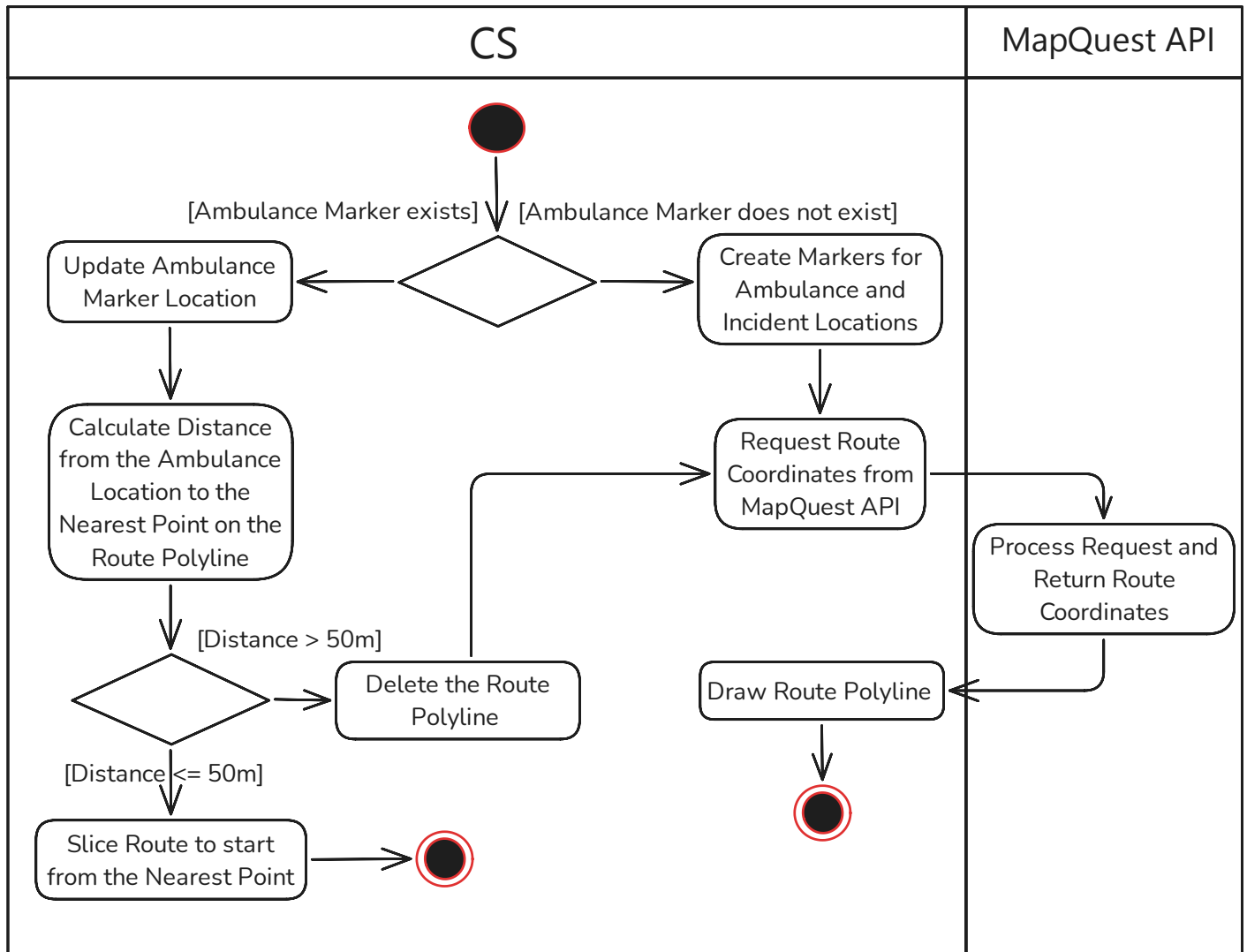
Figure 26: Activity diagram for ambulance-to-incident tracking data processing

Once the patient has been loaded into the ambulance, the activity diagram in Figure 27 illustrates the process of managing ambulance-to-hospital tracking data. While the overall logic is similar to the ambulance-to-incident tracking process, this diagram introduces new elements specific to this stage. Notably, upon receiving tracking data, the existence of a hospital marker associated with the ambulance will be checked. If the hospital marker does not exist, the incident marker will be deleted and replaced with an "X" marker at its location, and the hospital marker will be created at the specific coordinates. Similarly to the first process, a route request will be sent to the MapQuest API from the ambulance location to the hospital location to get the route coordinates and draw the route polyline. For cases where the hospital marker already exists, the ambulance marker updates its position, and the system recalculates or adjusts the route polyline based on the ambulance's current location relative to the route.
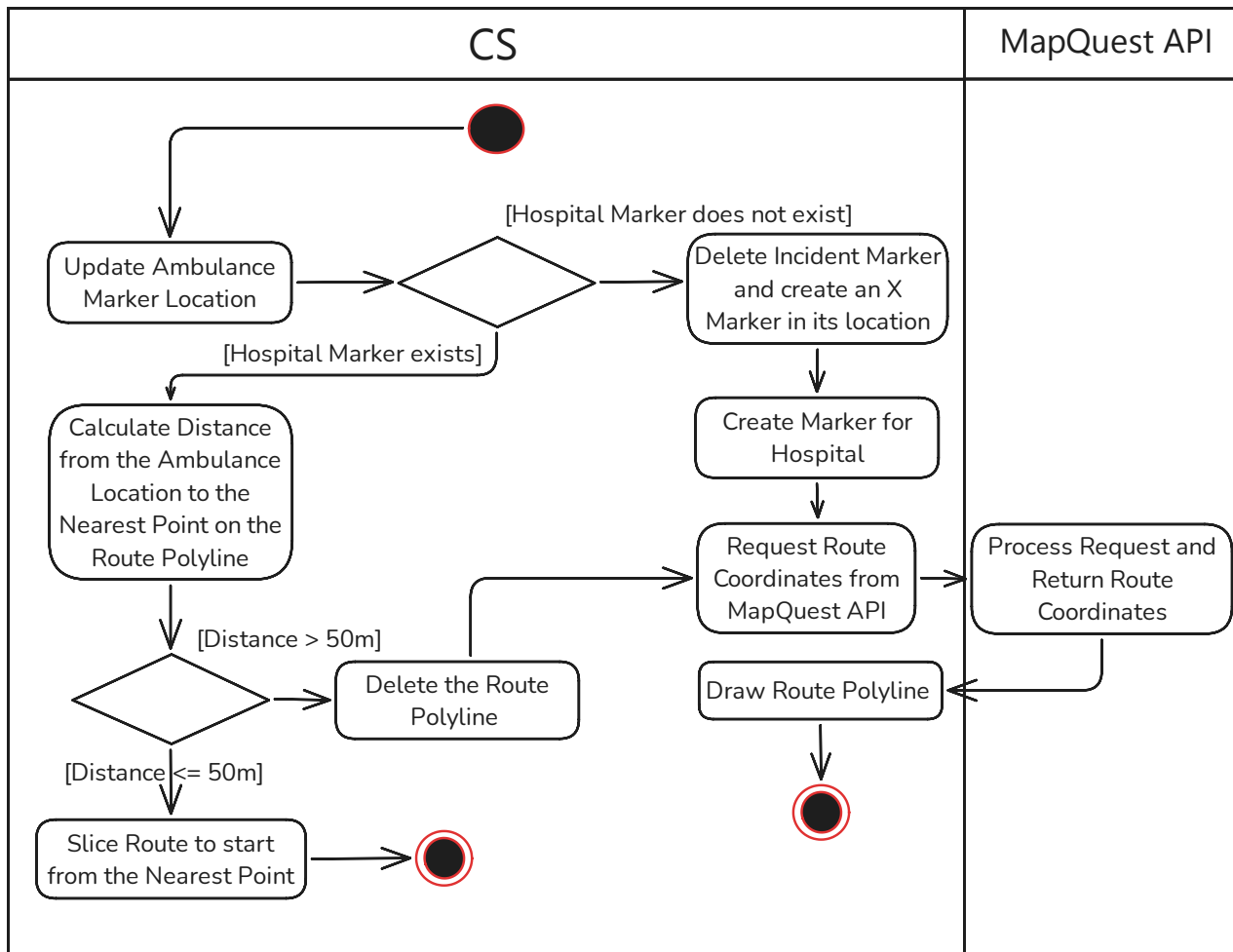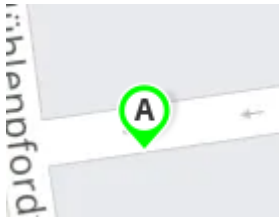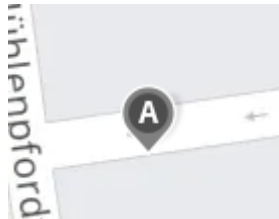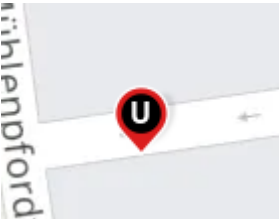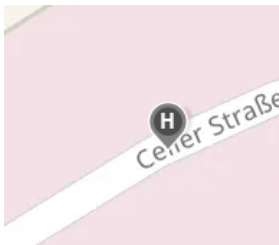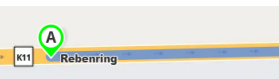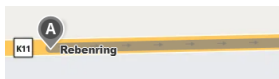
Figure 27: Activity diagram for ambulance-to-hospital tracking data processing

Lastly, when the patient is delivered to the hospital, the activity diagram in Figure 28 reflects the final type of tracking data processing, which serves as confirmation that the ambulance transported the victim to the hospital. It is a one-step process involving the deletion of all markers associated with the ambulance: The ambulance, hospital, and X markers.
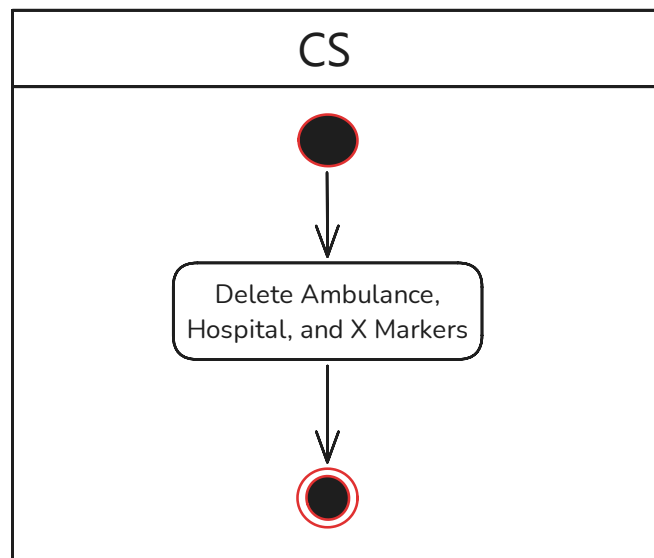
Figure 28: Activity diagram for confirmation data processing

To enhance the clarity of the map-based visualization, special colors are used to distinguish the main ambulance (the one selected for tracking in the Curing System) from other ambulances. The main ambulance is displayed in unique colors, while all other ambulances are represented by a neutral grey color scheme. This differentiation allows operators to identify and monitor the primary ambulance during tracking easily. Table 2 outlines the visual representation assigned to each map element.

	Main Ambulance	Other Ambulances
Ambulance Marker		
Incident Marker		
Hospital Marker		
Route Polyline		

Table 2: Visual representation of map elements

In addition, a user-friendly toggle button is implemented, as shown in Figure 29, which allows users to hide or show map data for other ambulances. When set to "Hide Other Ambulances Map Data" only the main ambulance's data is displayed, making it easier to focus on. Switching to "Show Other Ambulances Map" the map data for all ambulances are displayed.

(a)   (b)

Figure 29: Toggle button

Finally, the activity diagram in Figure 30 illustrates the process of managing and updating the remaining timetable, ensuring accurate real-time tracking of ambulance journeys. The first time an ambulance gets a route from the MapQuest API, the total remaining time, the time between the segments, and their corresponding transition locations are stored. A new row will be then inserted in the table with the ambulance ID, total remaining time, and if the destination is the incident or hospital (Fig. 31 (a)). When a polyline exists, the distance between the ambulance's current location and the first transition point in the array will be calculated by receiving ambulance tracking data (Fig. 25) using the Haversine Formula. Similarly, the distance between the fifth polyline coordinate and the first transition point will be calculated. If the second distance is bigger than the first distance, which means that the ambulance is around or has passed the transition location, then the total remaining time is updated by subtracting the passed segment time. If the remaining time drops below 3 minutes for hospital destinations, the corresponding remaining time text is highlighted in red (Fig. 31 (b)) to alert operators, that the ambulance is nearing the hospital, ensuring attention to the incoming victim. The row is deleted if the remaining time reaches zero for hospital destinations, indicating journey completion.
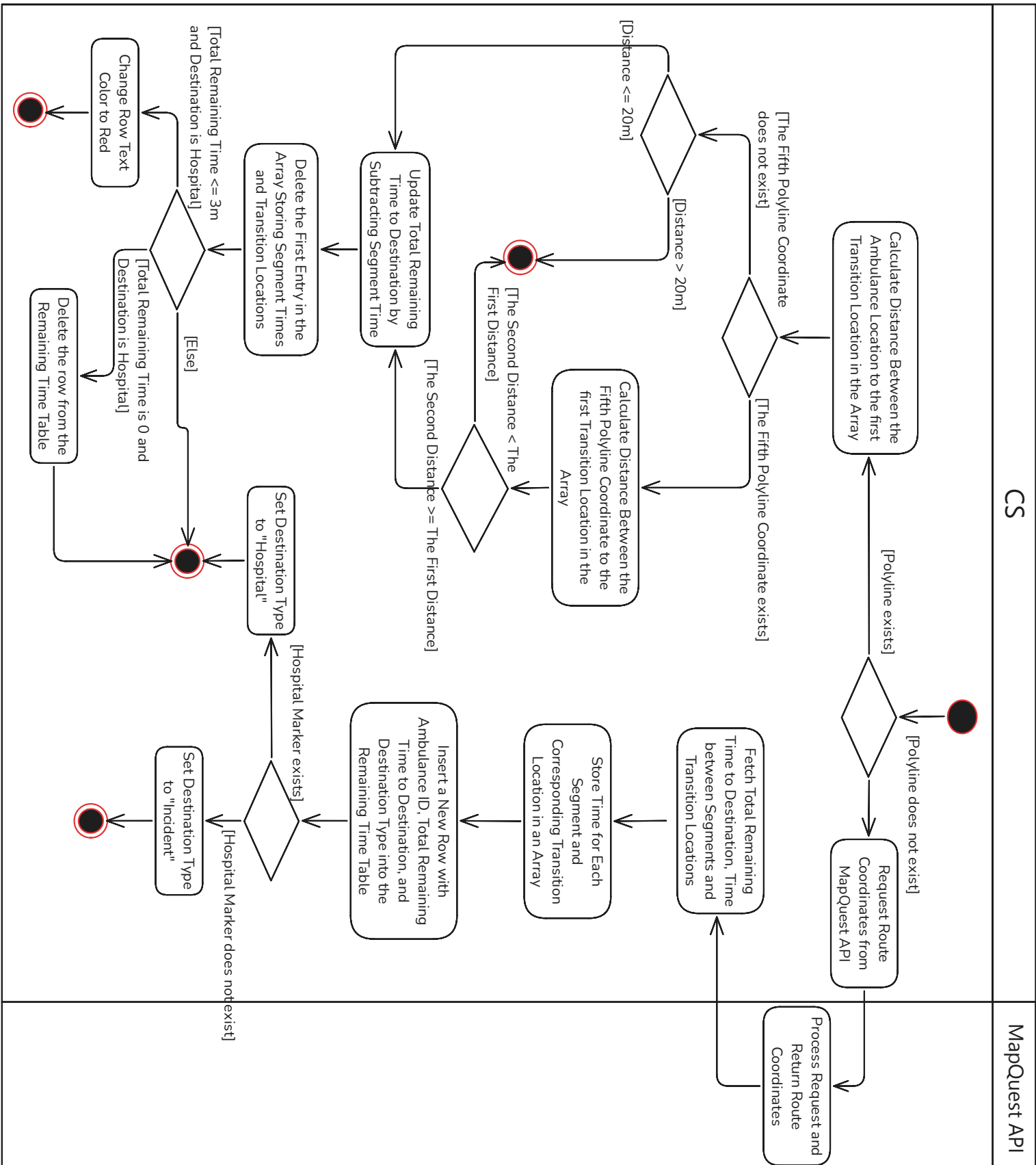
Figure 30: Activity diagram for updating remaining timetable

Ambulance ID	Remaining Time	To	
1	00:02:57	Incident	Breakdown!

(a)

Ambulance ID	Remaining Time	To	
1	00:01:53	Hospital	Breakdown!

(b)

Figure 31: Remaining time table

4 Results

This section presents the results of the implementation, beginning with real-time tracking of emergency response using the Rescuetrack within the ISAN platform, followed by the simulation of emergency scenarios.

4.1 Real-Time Tracking of Emergency Response using the Rescuetrack

The driving test demonstrated that the ISAN platform has been successfully integrated with Rescuetrack, enabling real-time tracking of an emergency response.

After generating a simulated alarm and assigning an ambulance, the RS started retrieving the live coordinates of the research vehicle from the Rescuetrack using the SOAP request (Code Snippet 1). In this setup, the alarm address was set to Mühlenpfordstraße 23, Braunschweig, serving as the incident location while the assigned ambulance was simulated by the research vehicle, which was physically driven during the test. The start position of the vehicle was Celler Straße 81.

The SOAP request returned the position data stored in Rescuetrack over the past 30 minutes. The latest recorded position was extracted and saved in the ISAN database, including a timestamp, latitude, and longitude, and was subsequently used for real-time tracking within the ISAN platform.

To visualize the tracking, we assigned Celler Straße Hospital to the generated ISAN and accessed the CS tracking page, where the vehicle's real-time position was correctly displayed on the map (A), along with the incident location (U), hospital location (H), the route from the ambulance to the incident (Fig. 32). The remaining timetable showed the estimated arrival time. A simulated emergency response was performed, and we drove to the incident location.

Upon reaching the incident location, the "Patient Loaded into Ambulance" button (Fig. 15) was used to simulate loading the victim into the ambulance. The incident location (U) marker was then replaced with an (X) marker, and the system generated a new route to the hospital, updating the remaining timetable accordingly (Fig. 33).

Upon arrival at Celler Straße Hospital, the "Patient Loaded into Hospital" button (Fig. 15) was clicked. At this point, tracking requests to Rescuetrack stopped automatically, the map was cleared of all data (Fig. 34), and the ambulance status was updated from 'Occupied' to 'Active'.

A total of 68 coordinates were recorded in this simulated emergency response. The complete list of recorded coordinates, including timestamps and GPS locations, is provided in the Appendix (Table 6). The average interval between two consecutive locations retrieved from the Rescuetrack was 16.1 ± 2.8 seconds. As a result, the ambulance's position on the map was refreshed on average approximately at the same interval.

Table 3 presents the estimated remaining time compared to the actual time taken to reach both the incident and hospital locations, as well as the difference between them.

Destination	Estimated Remaining Time (mm:ss)	Actual Time Taken (mm:ss)	Difference (s)
Incident	05:30	08:30	+ 180
Hospital	04:00	04:07	+ 7

Table 3: Comparison of estimated and actual time taken to reach destinations

During tracking, the ambulance's position was updated along with its drawn route and the remaining timetable as the ambulance moved. However, it was observed that in some cases, the route polyline did not start precisely at the ambulance's marker position as intended (Fig. 35).

Ambulance ID	Remaining Time	To	
1	00:05:30	Incident	Breakdown!

Figure 32: The ambulance en route to the incident during the driving test.

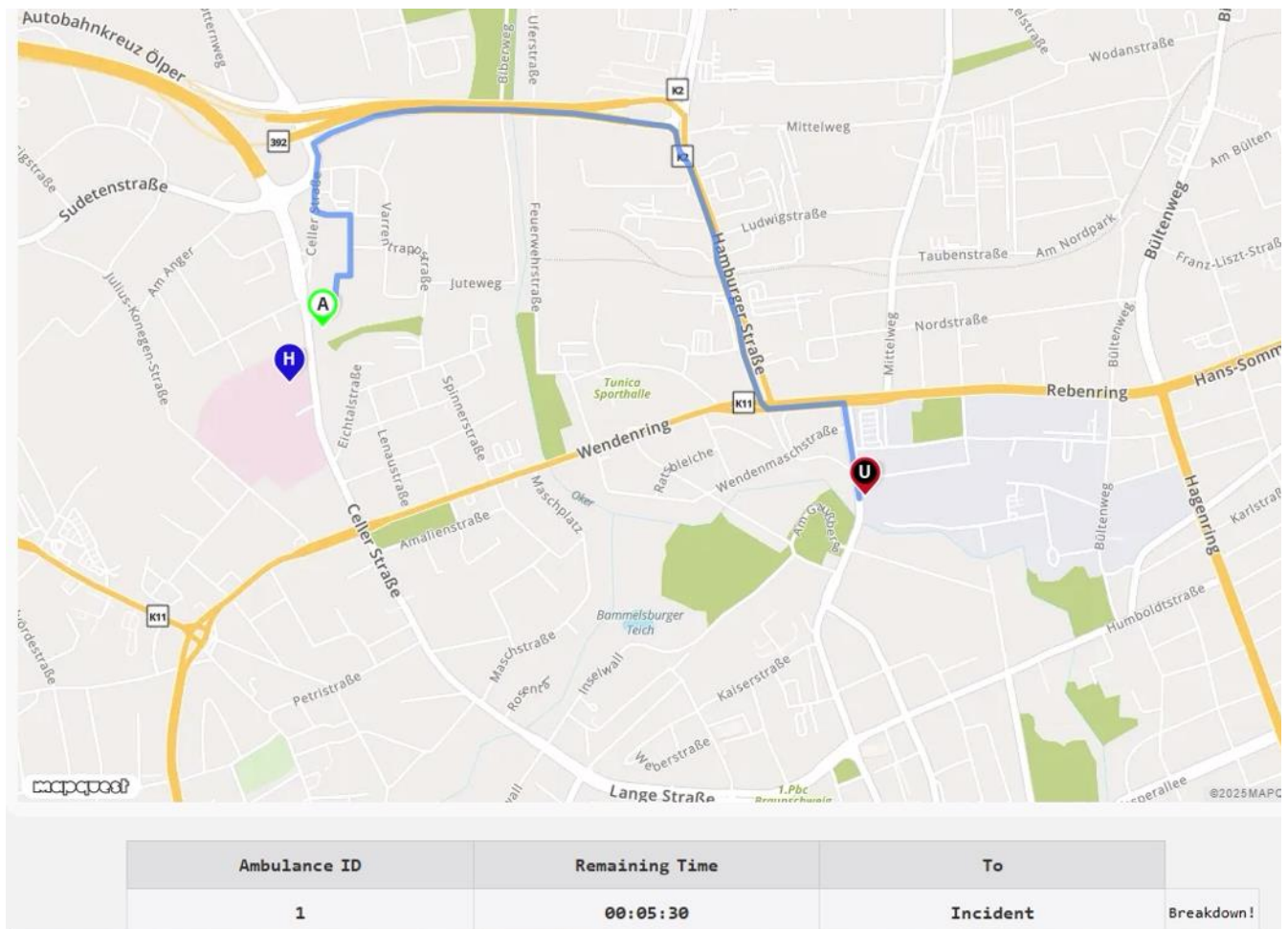| Ambulance ID	Remaining Time	To	
1	00:04:00	Hospital	Breakdown!
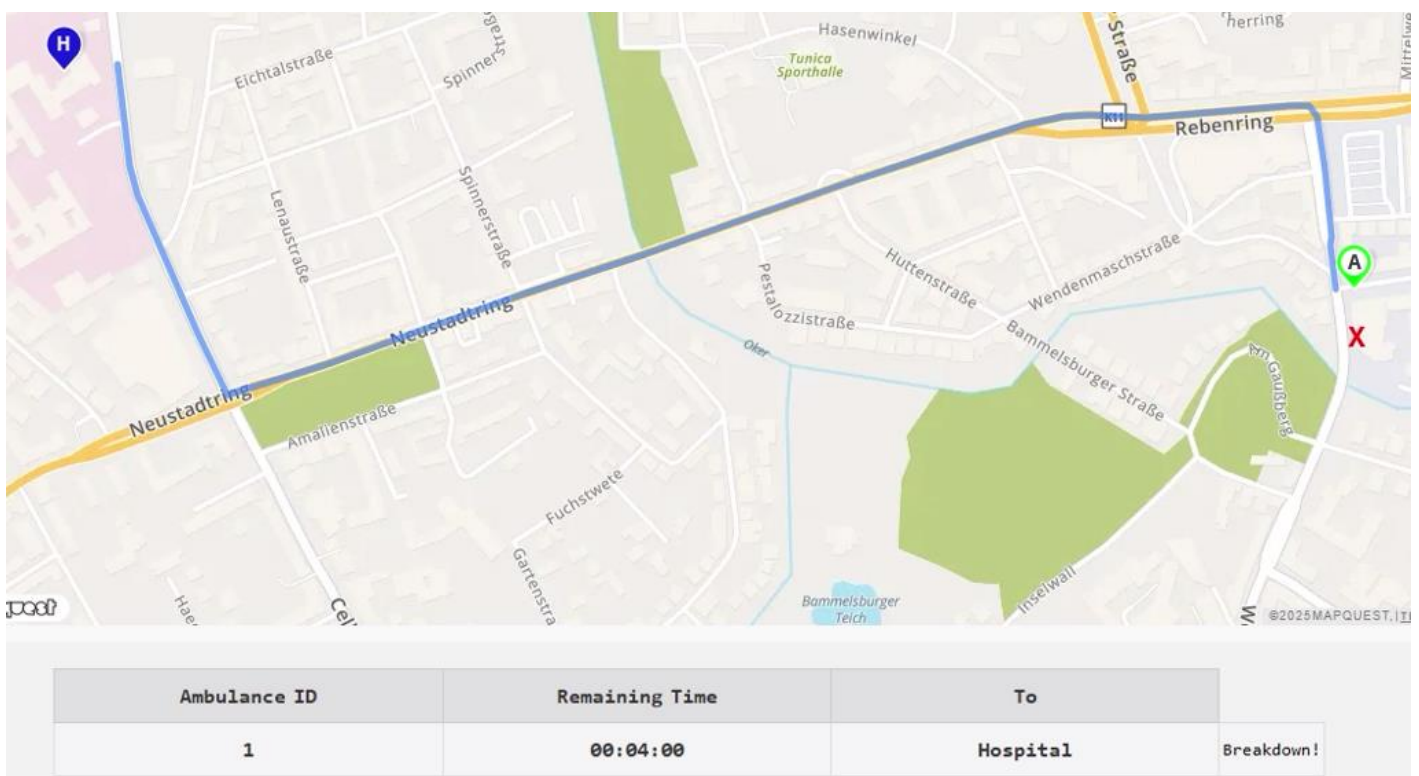
Figure 33: The ambulance en route to the hospital during the driving test.
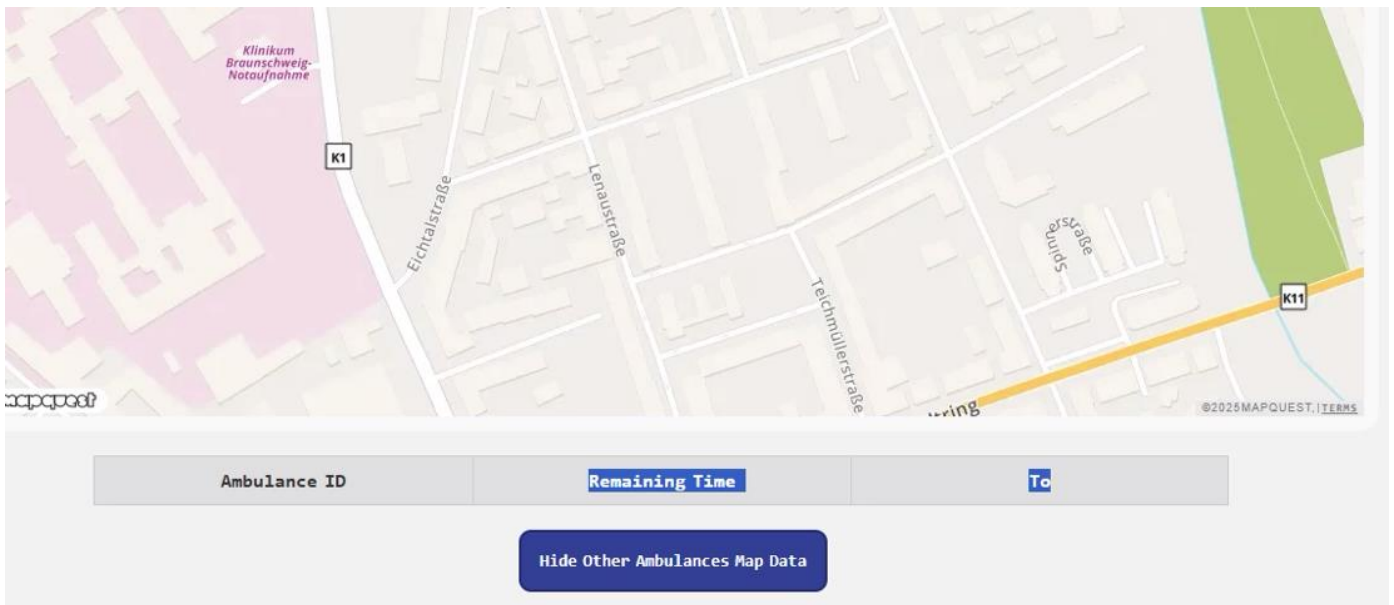
Figure 34: Deletion of the ambulance's map data upon arrival at the hospital.
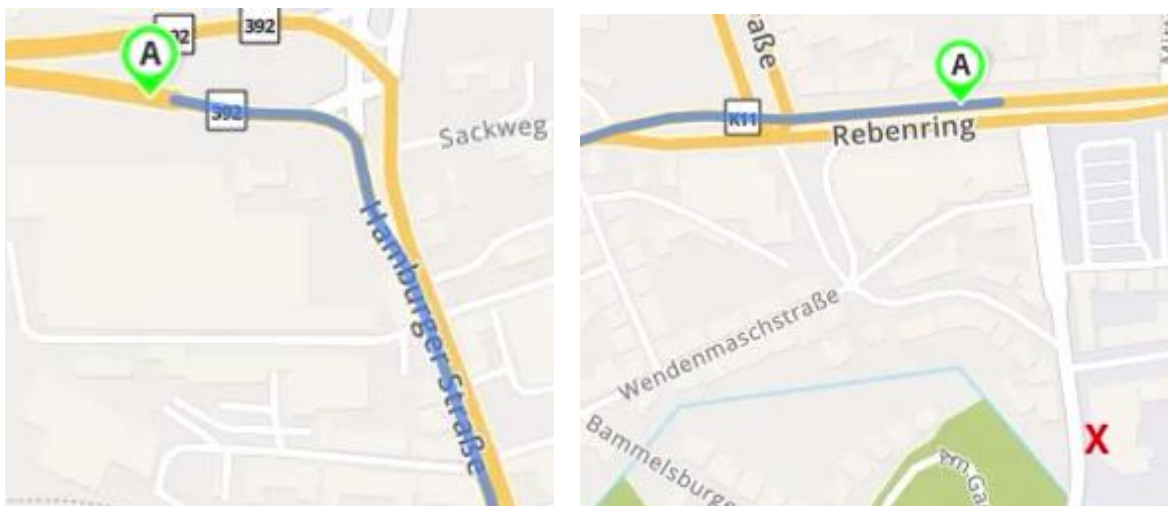
Figure 35: The ambulance marker is not aligned exactly with the starting point of the route polyline.

4.2 Simulation-Based Tracking of Emergency Response

The results demonstrate that the ISAN platform now supports the tracking of multiple ambulances simultaneously and provides rerouting capability in the case of breakdown. To illustrate this, we present results for the two simulation scenarios outlined in the task description (Section 1.2): multi-ambulance tracking and an ambulance breakdown requiring reassignment.

4.2.1 Multi Ambulance Tracking

In this scenario, a three-vehicle accident was simulated. Upon starting the simulation as described in Section 3.1, each ambulance was routed to its assigned incident.

The incident location (U) of the selected ISAN, along with its corresponding ambulance (A), route, and hospital (H), was highlighted in a distinct color, while all other ISANs' data remained in grey (Fig. 36). When the "Hide Other Ambulances" button was clicked, only the selected ISAN's data remained visible on the map, while all other ISAN-related information was hidden (Fig. 37). Clicking the 'Show Other Ambulances' button again restored all previously hidden information. The remaining timetable displayed the ID, estimated arrival time, and current destination "Incident" for each ambulance, with the selected ISAN's values highlighted in bold (Fig. 38).

The ambulances moved along their assigned routes. Clicking on an ambulance marker displayed its unique ID in a pop-up window (Fig. 37), which could be dismissed by clicking again. As the ambulances progressed along their routes, the remaining timetable was dynamically updated whenever an ambulance passed a transition location (Fig. 39).

Upon reaching the incident location, the incident marker was replaced with an X marker, indicating that the incident had been handled. A new route was then generated from the ambulance's current position to the assigned hospital, and the destination in the remaining time-table was updated from "Incident" to "Hospital" (Fig. 40). The ambulances moved then along their assigned routes to Hospitals (Fig. 41).

When an ambulance's remaining time dropped below three minutes, its time value was displayed in red, indicating that the ambulance was approaching the hospital (Fig. 42).

Once an ambulance arrived at the hospital, its data was automatically removed from the map (Fig. 43) and the remaining timetable (Fig. 42), signaling the end of its emergency response.

When all ambulances completed their assigned tasks and reached their respective hospitals, the entire dataset, including map data and timetable entries, was cleared.
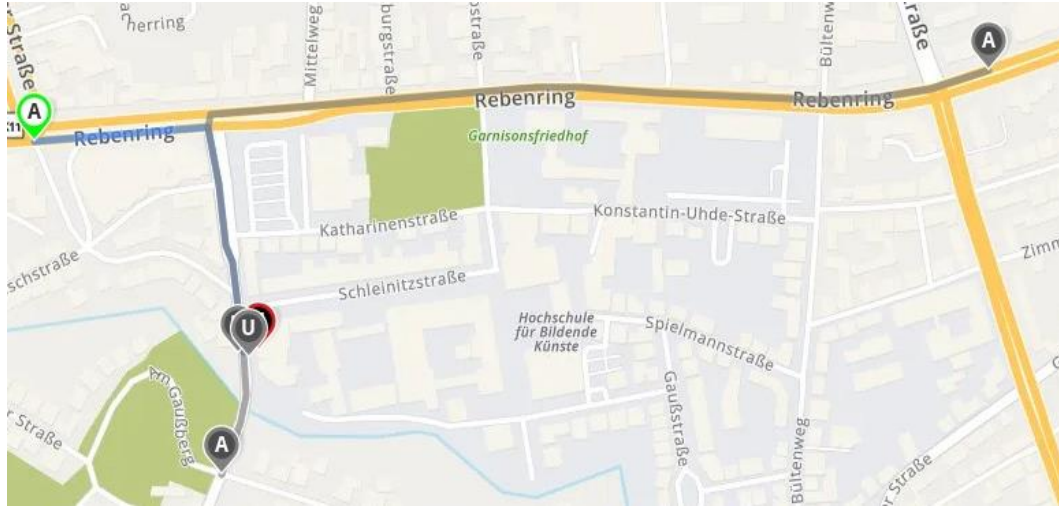
Figure 36: Ambulances in transit to incident locations
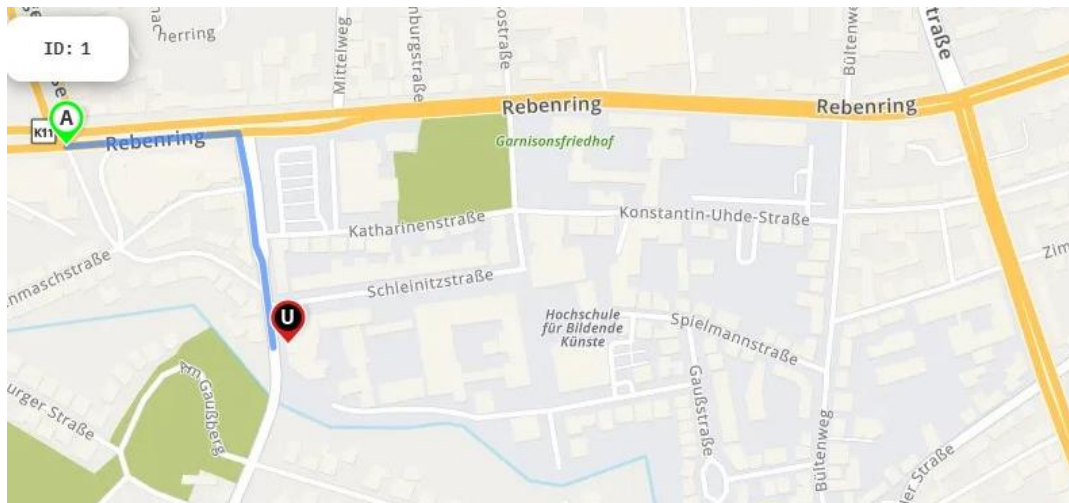
Figure 37: The interface displaying only the data for the selected ISAN, focusing on the ambulance with the ID 1

Ambulance ID	Remaining Time	To	
1	**00:01:50**	**Incident**	Breakdown!
2	00:03:07	Incident	Breakdown!
3	00:00:43	Incident	Breakdown!

Hide Other Ambulances Map Data

Delete Simulation Data

Figure 38: The display of estimated remaining time for ambulances to reach incidents.

Ambulance ID	Remaining Time	To	
1	**00:02:06**	**Incident**	Breakdown!
3	00:00:40	Incident	Breakdown!
2	00:03:14	Incident	Breakdown!

Ambulance ID	Remaining Time	To	
1	**00:01:15**	**Incident**	Breakdown!
3	00:00:28	Incident	Breakdown!
2	00:02:35	Incident	Breakdown!

Figure 39: The update of remaining time estimates as ambulances pass transition locations along their route.
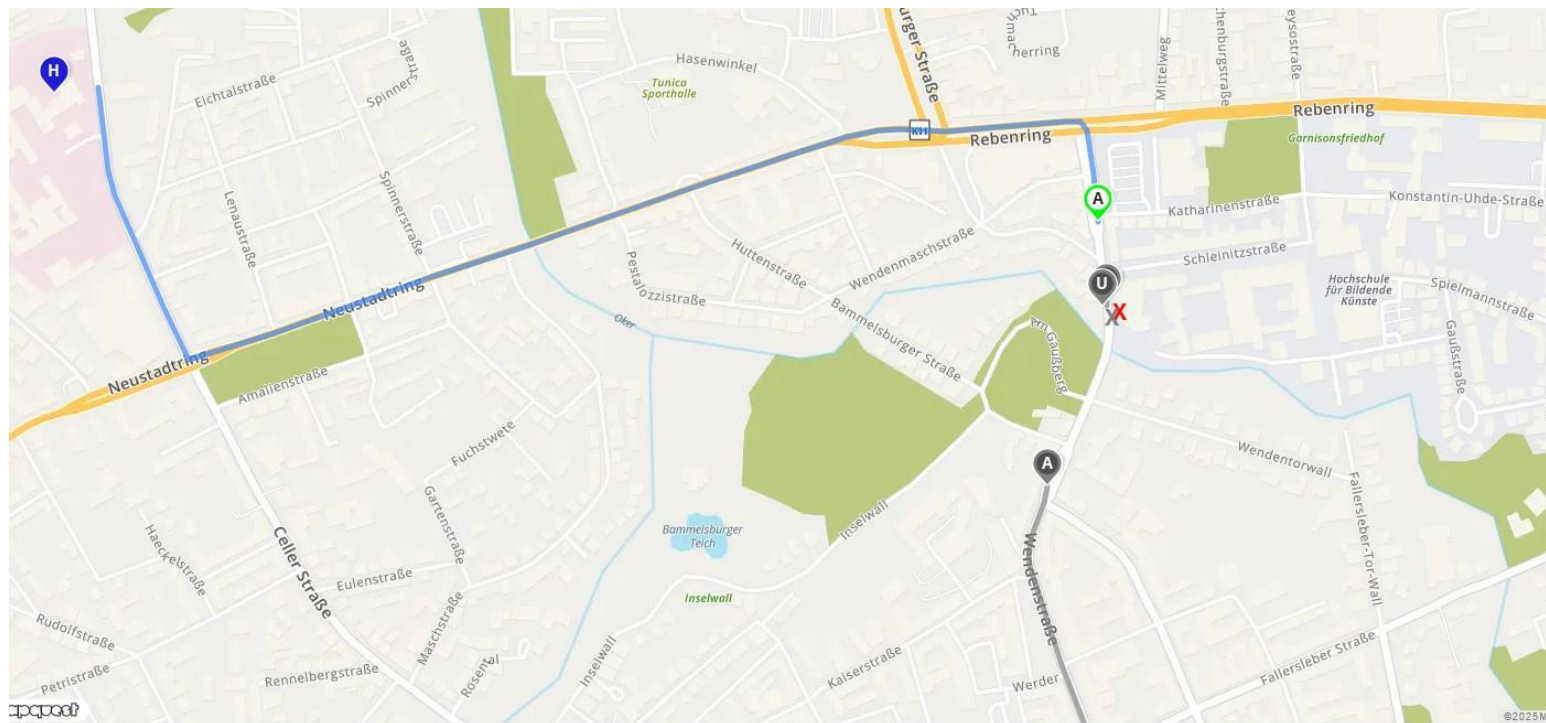
Ambulance ID	Remaining Time	To	
1	**00:05:49**	**Hospital**	Breakdown!
2	00:00:00	Incident	Breakdown!
3	00:13:03	Hospital	Breakdown!

Figure 40: When the patient is loaded into the ambulance, the incident marker is replaced with "X." A new route to the hospital is generated, and the destination is updated to "Hospital."
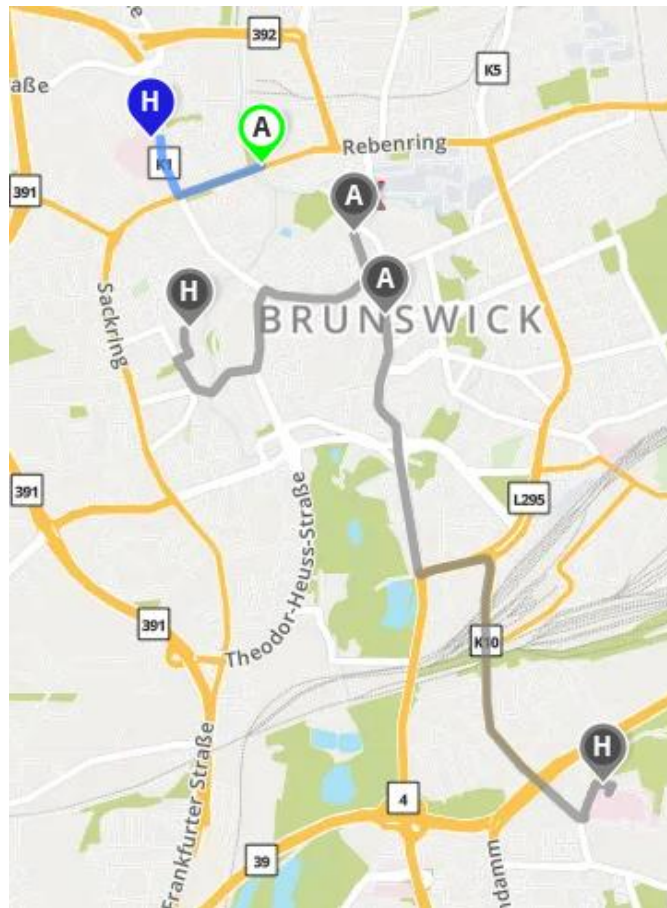
Figure 41: Ambulances in transit to hospitals

Ambulance ID	Remaining Time	To	
2	00:01:50	Hospital	Breakdown!
3	00:05:39	Hospital	Breakdown!

Figure 42: Remaining time value turning red for ambulance ID 2 and removal of ambulance ID 1 row's data

Figure 43: Deletion of map data for the ambulance with ID 1.

At the end of the simulation, the only data retained in the ISAN platform consists of route coordinates stored in a CSV file for each ambulance. The table below summarizes the number of coordinates recorded for each ambulance on its journey from the start position to the incident location and then to the assigned hospital for the first simulation scenario.

Ambulance ID	Hospital Destination	# Coordinates to Incident	# Coordinates to Hospital
1	Celler Straße	24	67
2	Holwedestraße	36	132
3	Salzdaluhmerstraße	17	246

Table 4: Recorded route coordinates for the first simulation scenario

4.2.2 Ambulance Breakdown and Reassignment

In this scenario, a single-car accident occurred. An ambulance was initially assigned to the incident and began its route toward the accident location. However, midway through the journey, the assigned ambulance was manually set to "broken" to simulate a real-world breakdown event. Two cases were considered in this scenario:

Case 1: Breakdown Before Transporting the Victim

If the initially assigned ambulance broke down before reaching the incident location and transporting the victim to the hospital (Fig. 44), its data was removed from the map and remaining timetable, and a new ambulance was assigned to the ISAN if one was available. The newly assigned ambulance proceeded to the original incident location (Fig. 45), where it took over the emergency response and then transported the victim to the designated hospital (Fig. 46).

Case 2: Breakdown While Transporting the Victim

If the initially assigned ambulance broke down while transporting the victim to the hospital (Fig. 47), a new ambulance was assigned to the ISAN if one was available. In this case, the newly assigned ambulance was routed to the breakdown location (Fig. 48) before continuing the transport of the victim to the hospital (Fig. 49).

Ambulance ID	Remaining Time	To	
1	00:02:33	Incident	Breakdown!
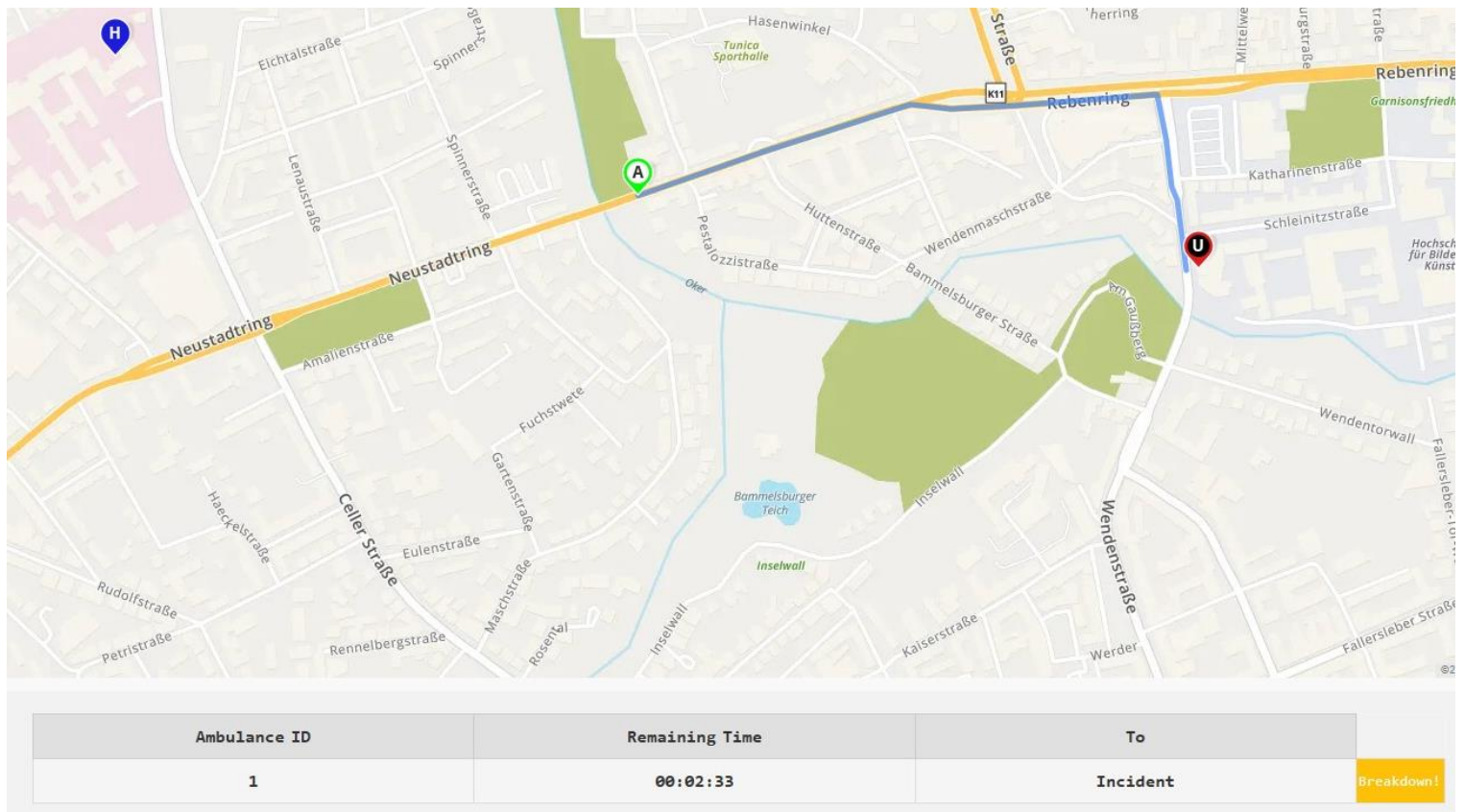
Figure 44: Breakdown event triggered before transporting the victim

Ambulance ID	Remaining Time	To	
2	00:01:53	Incident	Breakdown!
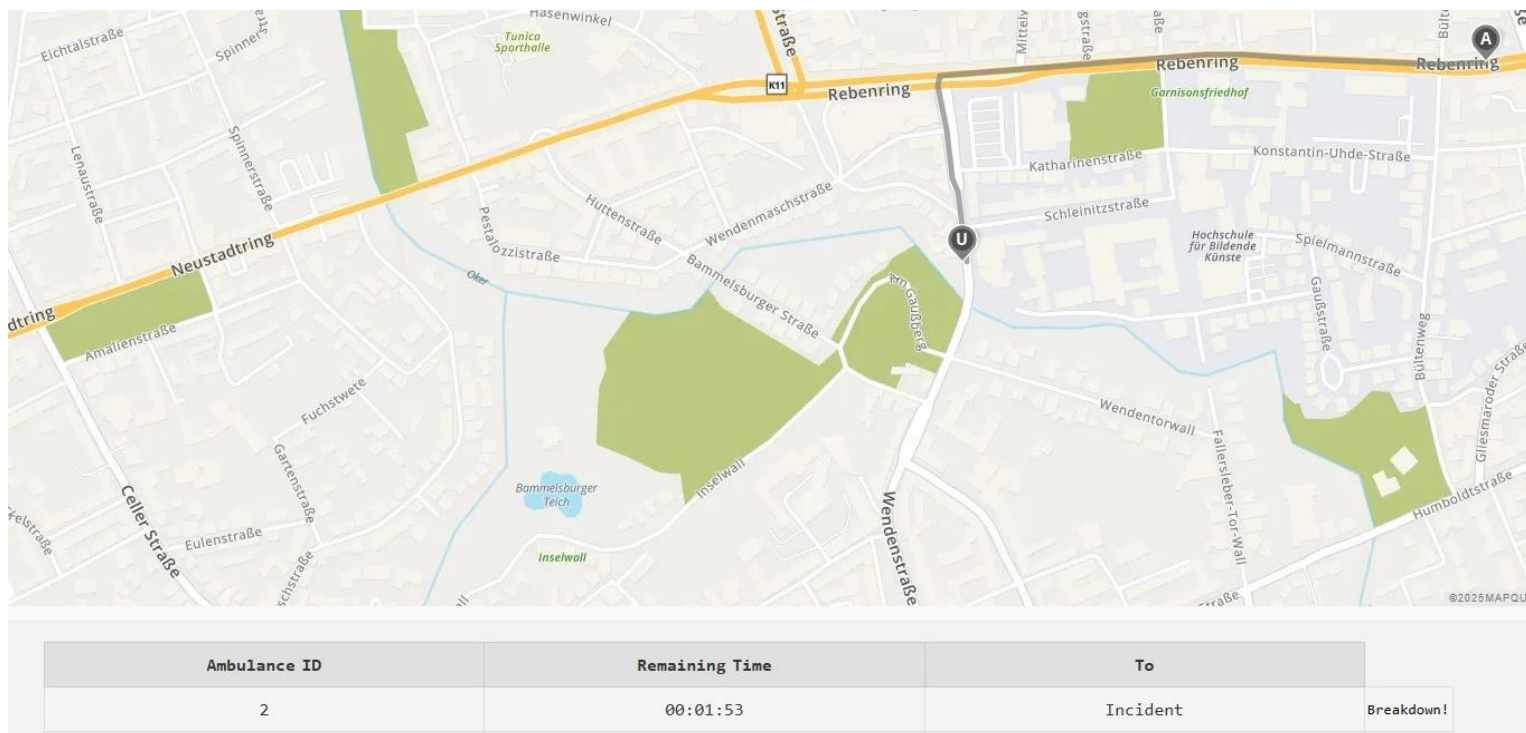
Figure 45: Reassignment of new ambulance to the ISAN location
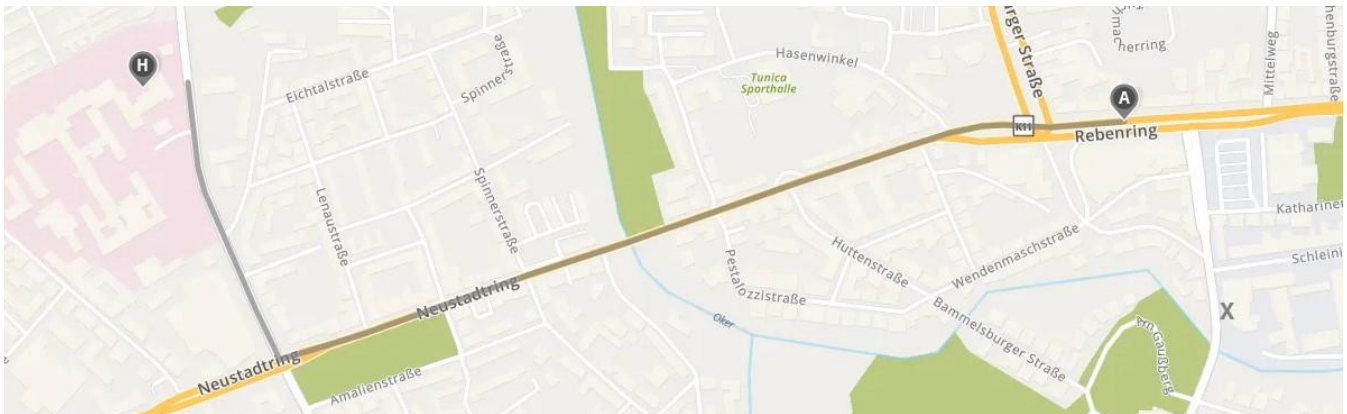
Figure 46: Newly assigned ambulance transporting the victim to the hospital

Ambulance ID	Remaining Time	To
1	00:02:02	Hospital
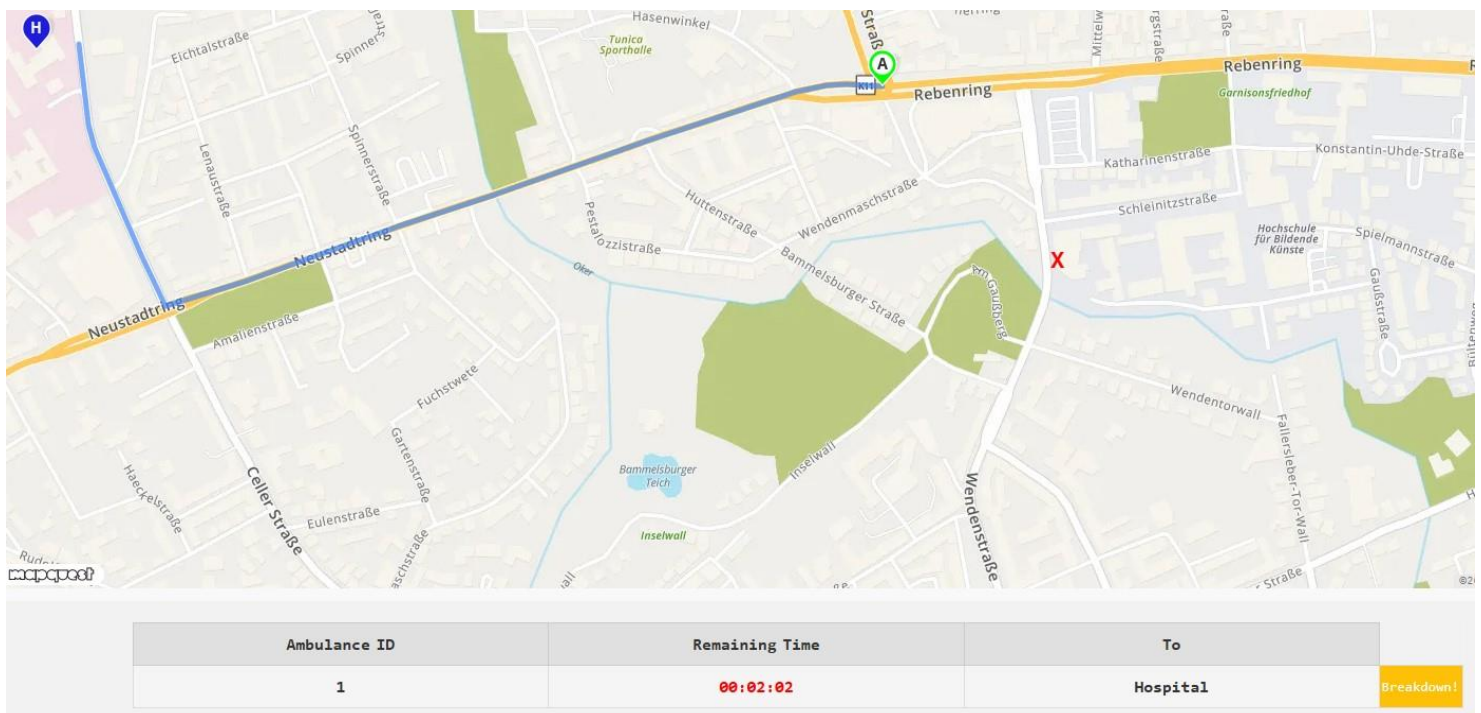
Figure 47: Breakdown event triggered while transporting the victim

Ambulance ID	Remaining Time	To	
2	00:01:24	Incident	Breakdown!
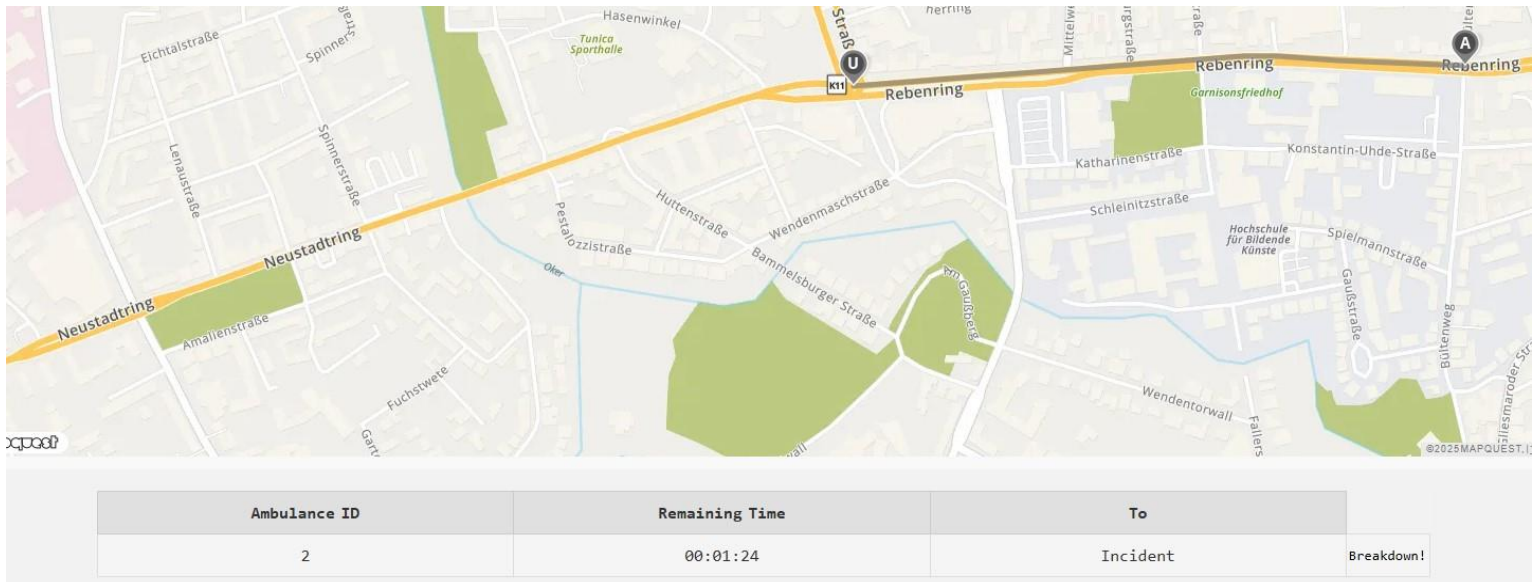
Figure 48: Reassignment of new ambulance to the breakdown location
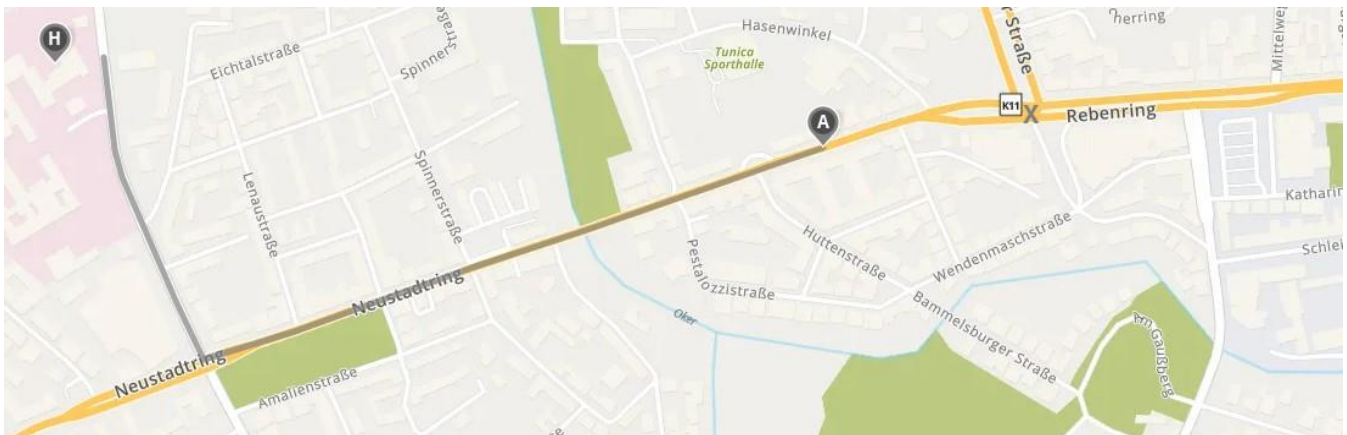
Figure 49: Newly assigned ambulance transporting the victim to the hospital

Similarly, at the end of the simulation, the only data retained in the ISAN platform consists of route coordinates stored in a CSV file for each ambulance. The table below summarizes the number of coordinates recorded for the newly assigned ambulance on its journey from the start position to the incident location and then to the hospital for both cases.

Case	Hospital Destination	# Coordinates to Incident	# Coordinates to Hospital
Case 1	Celler Straße	24	67
Case 2	Celler Straße	17	50

Table 5: Recorded route coordinates for the second simulation scenario

4.3 Traffic and Construction Data Retrieval

It is configured to retrieve and display real-time traffic congestion and construction data using the MapQuest API. However, despite enabling these layers in the implementation, they were only displayed in the USA (Fig. 50), while no such data appeared in Germany or other regions.
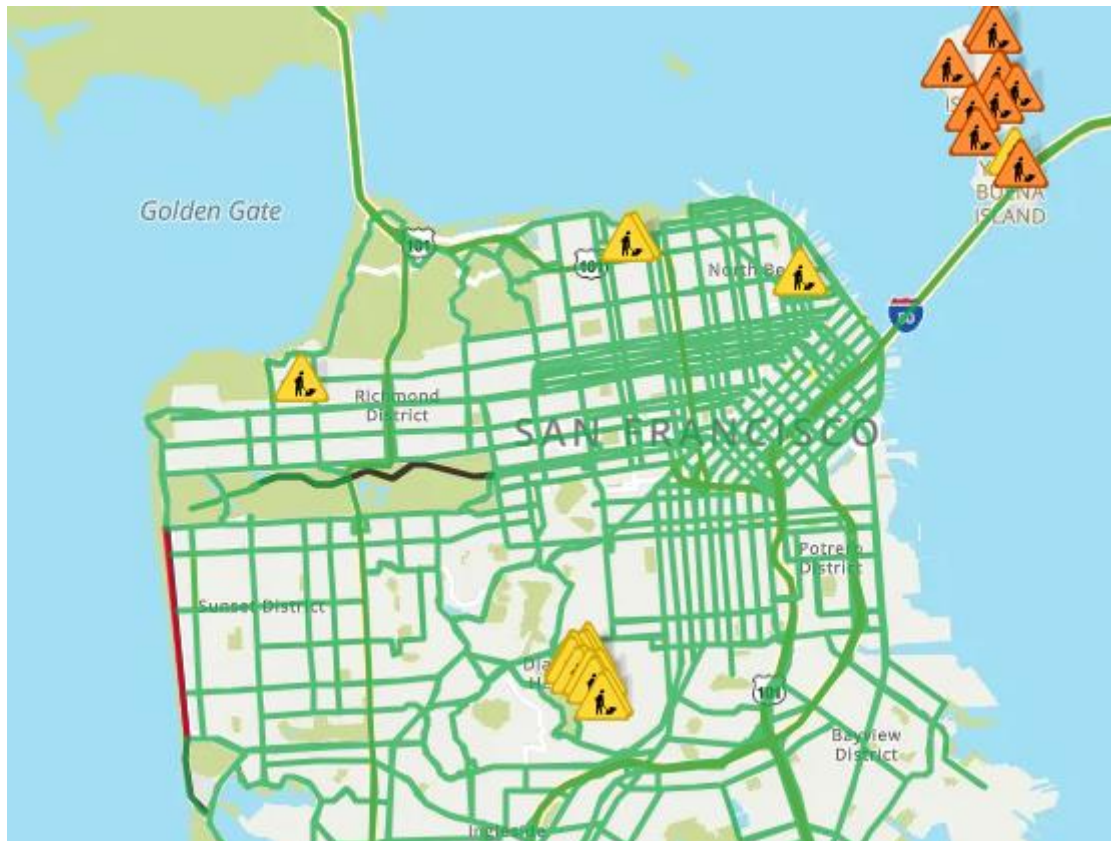
Figure 50: Traffic congestion and construction data displayed in San Francisco (USA)

5 Discussion

5.1 Interpretation of Results

The results confirm that Rescuetrack was successfully integrated into the ISAN Platform, adhering to ISAN communication protocols, thereby answering the first research question (RQ1) regarding the feasibility of this integration. This allowed for the retrieval of GPS data from the Rescuetrack and its transmission to the Curing System, where it was displayed in real-time. During tracking, the ambulance marker was both correctly updated and accurately positioned on the route, and it did not appear in incorrect locations, such as on buildings. Additionally, the drawn route polylines and remaining time estimates were dynamically updated based on the ambulance's progress. The ability to track multiple ambulances simultaneously was successfully demonstrated in the simulation environment, and the platform was able to automatically reroute a new ambulance in the event of a breakdown, positively confirming the third (RQ3) and fourth research (RQ4) questions.

One key finding was that the average interval between consecutive position updates from the Rescuetrack was 16.1 ± 2.8 seconds, meaning that the ambulance's position on the map was refreshed approximately at the same interval. This longer interval resulted in a gap between updates, where the ambulance's position on the map appeared to "teleport" from one point to another unlike in Google Maps navigation, where movement appears continuous. However, this update rate still ensured correct tracking, even though it was longer than personally expected, as a 5-second interval would have been more suitable. The likely cause of this delay is that all SOAP requests (Code Snippet 1) were sent with position ID = 0, which retrieves all position data from the past 30 minutes, leading to unnecessary data retrieval. Based on that, we can answer the second research question (RQ2) by stating that the ISAN platform performs at a moderate level in updating emergency vehicle positions on the map.

A possible optimization to improve the position update frequency from Rescuetrack and reduce the "teleportation" effect could be to modify the SOAP request structure. The first request will be sent using position ID = 0, retrieving all historical positions, while subsequent requests will use the position ID of the last received coordinate. This approach will ensure that only new position data is retrieved, reducing the amount of data sent back in each response and potentially minimizing delays in the response. Additionally, enabling the Long Polling option in the request will prevent the API from immediately returning an empty response if no new positions are available at the time of the request. Instead, the request will wait until new data becomes available before returning a response.

This mechanism eliminates the need for unnecessary repeated requests, reducing network overhead, and potentially decreasing delays.

In the driving test, the estimated remaining times were updated periodically during the journey, but a discrepancy was observed between the initial estimated and actual travel times. The journey to the simulated incident took 3 minutes longer than estimated, which is a significant deviation. In contrast, the journey to the hospital was only 7 seconds longer than estimated, which falls within an acceptable range. This allows us to conclude that the update quality is moderate (RQ5). However, no definitive conclusions can be drawn from this result, as it is based on a single test.

The results indicated also that the traffic layer, which visually represents roads with low, medium, high, or blocked traffic, and the construction layer, which categorizes construction zones by size and level of impact, were not available in Germany. These layers were only displayed in the USA, suggesting a regional limitation in the MapQuest API's data coverage. A solution would be to use another API that provides real-time traffic flow conditions and road accessibility visualization layers for Germany and other regions.

In real-time tracking, there were a few instances where the route polyline did not align exactly with the ambulance's position as intended (Fig. 35), which was meant to resemble Google Maps navigation tracking. This occurred because the drawn route is constructed from a predefined set of coordinates, and in these cases, the ambulance's real-time coordinate did not perfectly match the nearest coordinate from this set. However, this is not a functional requirement but rather an aesthetic improvement to enhance visual representation. Considering these findings, the quality of route updates is good but not perfect (RQ6).

In contrast, simulation-based tracking did not encounter this issue, as the coordinates used to simulate the ambulance's movement were identical to those defining the route drawn on the map. This ensured precise alignment between the ambulance's position and the start position of the displayed route. Additionally, simulation-based tracking provided consistently high tracking quality, as a large number of coordinates were processed, allowing for smooth and frequent updates. For example, in the first simulation scenario, the ambulance with ID 3 received 263 coordinates, meaning its position was refreshed 263 times, covering almost the entire route.

5.2 Limitations and Future Enhancements

One current limitation is that when an ISAN is generated and an ambulance is assigned, the system selects an active ambulance based solely on whether it shares the same responsibility area as the postal code in the generated ISAN.

However, if multiple ambulances are available within the same responsibility area, the RSM randomly selects one without considering their actual GPS locations. This means that a farther ambulance might be assigned instead of the closest available one, potentially increasing response time.

To enhance this process, ambulance tracking should be active at all times, even when ambulances are not responding to emergencies. This would allow the RSM to check the real-time locations of all available ambulances when an ISAN is sent. Instead of assigning an ambulance arbitrarily, the RSM could request route calculations from the MapQuest API for each available ambulance and determine the one with the shortest estimated arrival time to the incident location.

Additionally, the only geographic information that the RSM currently stores about an ambulance is its responsibility area (postal code), without any data on the specific city it belongs to. As a result, the tracking logic includes ambulances from all cities, which is not practicable since hospitals only need to track emergencies within their city, this lack of distinction leads to unnecessary data transmission. One possible solution is to modify the RSM to also store city information along with the responsibility area. This would allow the tracking logic to be adjusted so that hospitals only receive tracking data for emergencies occurring within their respective cities.

Another limitation is that the RS currently does not have access to the tracking page, as it is implemented within the CS. This restricts ambulances from viewing real-time tracking data, which could be valuable for their operations. However, this limitation could be resolved by extending the tracking implementation to the RS, allowing ambulances to access tracking functionality directly.

Lastly, if an ambulance breaks down and no backup ambulance is available for rerouting, or if an ISAN is generated when no ambulances are available, the system takes no action, resulting in the sent ISAN being lost. To address this, the ISAN platform should incorporate a handling mechanism for cases where an ISAN is generated but no ambulance is available. One possible solution is to save the ISAN request in a queue and notify an administrator via an alert indicating that an emergency request was made but could not be fulfilled. Once an ambulance becomes available again, the system could also automatically assign the pending ISAN to the first active ambulance. This approach would ensure that no sent ISAN is lost, even when immediate resources are unavailable.

6 Conclusion

This work successfully integrated the Rescuetrack into the ISAN platform, enabling real-time ambulance tracking, multi-ambulance tracking, and providing rerouting capability in case of an ambulance breakdown. However, further improvements are needed to refine and enhance the assignment and tracking of emergency responders within the ISAN platform.

References

[1] Haghi M, Barakat R, Spicher N, Heinrich C, Jageniak J, Söylev Öktem G, et al. **Automatic information exchange in the early rescue chain using the International Standard Accident Number (ISAN)**. *Healthcare*. 2021;9(8):996.

[2] Barakat R, Deserno TM. **Automatic alerting of accidents and emergencies: the international standard accident number and vital sign data embedded in future PACS**. Proc. SPIE MI. 2020;11318.

[3] Spicher N, Barakat R, Wang J, Haghi M, Jagieniak J, Öktem GS, Hackel S, Deserno TM. **Proposing an International Standard Accident Number for interconnecting information and communication technology systems of the rescue chain**. Methods Inf Med. 2021;60:e20 --.e31.

[4] Meischke H, Chavez D, Bradley S, Rea T, Eisenberg M. **Emergency communications with limited-english-proficiency populations**. Prehosp Emerg Care. 2010;14(2):265-71.

[5] Sangers B, Praveen P. **IoT-Enabled intelligent traffic navigation with accident management for critical emergency response in heavy congestion using machine learning**. Sage Advance. *2024.*

[6] Maris M, Berben SAA, Verhoef W, van Grunsven P, Tan ECTH. **The quality of pre-announcement communication and the accuracy of estimated arrival time in critically ill patients, a prospective observational study**. BMC Emerg Med. 2022;22:44.

[7] Pell JP, Sirel JM, Marsden AK, Ford I, Cobbe SM. **Effect of reducing ambulance response times on deaths from out of hospital cardiac arrest: cohort study**. BMJ. 2001;322(7299):1385-8.

[8] Kim DY, Kim JC, Kim DK, Choa MH, Yoo SK. **Real-time ambulance location monitoring using GPS and Maps Open API**. Proc IEEE EMBS. 2008:1561-2.

[9] Polamreddy M, Pradeepa R, Immaculate Joy S. **Prior emergency ambulance tracking android application for traffic cops using GPS technology**. Proc IEEE ICONSTEM. 2024:1-5.

[10] Rudramurthy MS, Hegde N, Tanmay K, Sufian Jawaid M, Santosh M, Sushma SA, Keerthan Kumar TG. **Real-time vehicle tracking system for smart cities**. Proc ICDSNS. 2023:1-6.

[11] Shah K. **Implementation and integration of cellular/GPS-based vehicle tracking system with google maps using a web portal**. Proc ADISC. 2016:513-20.

[12] Shibghatullah AS, Jalil A, Wahab MHA, Ng Poh Soon J, Subaramaniam K, Eldabi T. **Vehicle tracking application based on real-time traffic**. Int J Elec Electron Eng Telecommun. 2022;11(1):67-73.

[13] **Haversine formula** [Internet]. Wikipedia. Available from: https://en.wikipedia.org/wiki/Haversine_formula. [Accessed 2025 Jan 14].

[14] Goodwin M. **API: Application Programming Interface** [Internet]. IBM. Available from: https://www.ibm.com/think/topics/api. [Accessed 2025 Jan 2].

[15] **What is REST?** [Internet]. Codecademy. Available from: https://www.codecademy.com/article/what-is-rest. [Accessed 2025 Jan 2].

[16] Fielding RT. **Architectural styles and the design of network-based software architectures**. Doctoral Dissertation, University of California, Irvine. 2000.

[17] **Mapquest developer documentation** [Internet]. MapQuest. Available from: https://developer.mapquest.com/documentation/. [Accessed 2025 Jan 26].

[18] Box D, Ehnebuske D, Kakivaya G. **Simple Object Access Protocol (SOAP) 1.1**. W3. 2007. Available from: https://www.w3.org/TR/SOAP/. [Accessed 2025 Jan 4].

[19] **Rescuetrack** [Internet]. Rescuetrack. Available from: https://www.rescuetrack.de/de-de/. [Accessed 2025 Jan 26].

[20] **What is flask python?** [Internet]. Python Basics. Available from: https://pythonbasics.org/what-is-flask-python/. [Accessed 2025 Jan 4].

[21] **Docker overview** [Internet]. Docker Docs. Available from: https://docs.docker.com/get-started/overview/. [Accessed 2025 Jan 4].

[22] **HTML introduction** [Internet]. W3Schools. Available from: https://www.w3schools.com/html/html_intro.asp. [Accessed 2025 Jan 4].

[23] **Cascading style sheets (CSS)** [Internet]. MockFlow. Available from: https://mockflow.com/glossary/CSS#:~:text=Cascading%20Style%20Sheets%20(CSS)%20is,colors%2C%20fonts%2C%20and%20spacing. [Accessed 2025 Jan 4].

[24] **What is javascript?** [Internet]. Hostinger Tutorials. Available from: https://www.hostinger.com/tutorials/what-is-javascript#:~:text=JavaScript%20is%20a%20lightweight%20programming,capability%20that%20CSS%20alone%20lacks. [Accessed 2025 Jan 27].

[25] **Document object model** [Internet]. GeeksforGeeks. Available from: https://www.geeksforgeeks.org/dom-document-object-model/. [Accessed 2025 Jan 27].

[26] Fette I, Melnikov A. **The webSocket protocol** [Internet]. Internet Engineering Task Force; 2011. Available from: https://datatracker.ietf.org/doc/rfc6455/. [Accessed 2025 Jan 4].

[27] **Socket.IO introduction** [Internet]. Socket.IO. Available from: https://socket.io/docs/v4/. [Accessed 2025 Jan 4].

Appendix

Algorithm 1: Handle Alarm
Modifications in the workflow are highlighted in red
Input: ISAN, broken_ambulance_location (optional, see Algorithm 13)
Output: ISAN assigned to an ambulance if an active ambulance is available

---

1. The WFM sends the ISAN to the ASM for verification:
   - **1.1.** If verified, the ASM returns the ID of the AS (AS_ID).
   - **1.2.** if not verified, terminate the process.
2. The WFM sends the ISAN to the RSM to check for an available ambulance:
   - **2.1.** If an active ambulance is found in its responding systems table:
     - **2.1.1.** Change its status from "active" to "occupied".
     - **2.1.2.** The RSM returns its IP (RS_IP) and its ID (RS_ID).
   - **2.2.** if no active ambulance is available, terminate the process.
3. The WFM forwards the ISAN, AS_ID, RS_IP, RS_ID, and broken_ambulance_location (if provided) to the ComM.
4. ComM records a new entry in the active communications table, including:
   - AS_ID, RS_IP, and current timestamp.
5. The ComM forwards the ISAN, RS_ID, and broken_ambulance_location (if provided) to the RS using the RS_IP.
6. The RS records the ISAN in its alarm_list table along with the current timestamp.
7. The RS starts a separate thread to execute a function based on the tracking case:

   - Practical Case:
     - **7.1.** Executes "startGettingCoordinatesFromRescuetrack" (Algorithm 2).
   - Simulation Case:
     - **7.1.** Executes "simulation_write_route_to_incident" (Algorithm 3).
8. Terminate the process.

Algorithm 2: Request Coordinates from the Rescuetrack
Implementation Location: RS/practical_tracking.py
Input: ISAN, RS_ID, broken_ambulance_location (optional, see Algorithm 13)
Output: Tracking data, stored in the global variable "current_location"

1. If broken_ambulance_location is provided:
   1.1. Set the incident location (incident_latitude and incident_longitude) to the broken_ambulance_location.
2. If broken_ambulance_location is not provided:
   2.1. Extract the incident location (incident_latitude and incident_longitude) from the provided ISAN.
3. Start a while loop to fetch live coordinates until tracking is stopped:
   3.1. Send a POST request (Code Snippet 1) to the Rescuetrack API.
   3.2. Extract the last coordinate from the API response, that corresponds to the current ambulance location (ambulance_latitude and ambulance_longitude).
   3.3. Insert the data into the 'gps_logs' table in the database.
   3.4. if the patient is not yet loaded into the ambulance:
      3.4.1. Set the global current_location variable to the following structure:
      - "id": RS_ID
      - "lat": {ambulance_latitude}
      - "lng": {ambulance_longitude}
      - "incident_location":   {   "lat":   incident_latitude, "lng": incident_longitude }
   3.5. if the patient has been loaded into the ambulance:
      3.5.1. Set the global current_location variable to the following structure:
      - "id": RS_ID
      - "lat": {ambulance_latitude}
      - "lng": {ambulance_longitude}
      - "hospital_location":   {   "lat":   hospital_location.latitude, "lng": hospital_location.longitude } (Hospital location values are set by Algorithm 5)
   3.6. if the patient is transported to the hospital:
      3.6.1. Set the global current_location variable to the following structure:
      - "id": RS_ID
      - "lat": {ambulance_latitude}
      - "lng": {ambulance_longitude}
      - "isAtHospital": {}
      3.6.2. Break the while loop.
4. Terminate the process.

Algorithm 3: Simulation Write Route To Incident
Implementation Location: RS/simulation_tracking.py
Input: ISAN, RS_ID, broken_ambulance_location (optional, see Algorithm 13)
Output: A CSV file containing the route coordinates and a JSON file mapping ISANs to RS_IDs

1. Select a random starting position from a predefined pool of positions within the Braunschweig city center as the ambulance starting position.
2. If broken_ambulance_location is provided:
   2.1. Set the incident location (incident_latitude and incident_longitude) to the broken_ambulance_location.
3. If broken_ambulance_location is not provided:
   3.1. Extract the incident location (incident_latitude and incident_longitude) from the provided ISAN.
4. Send a GET request to the MapQuest API from the selected starting position to the incident location to retrieve the route coordinates (list of coordinate_latitude and coordinate_longitude pairs).
5. Write the retrieved route coordinates to the CSV file in the following format:
   - RS_ID, coordinate_latitude, coordinate_longitude, incident_latitude, incident_longitude, "incident_location".
6. Terminate the process.

Algorithm 4: Send ISAN To CS
Modifications in the workflow are highlighted in red

1. The RS starts a separate thread to execute a function based on the tracking case:
   - Practical Case:
     1.1. Executes "setHospitalLocation" (Algorithm 5).
   - Simulation Case:
     1.1. Executes "simulation_write_route_to_hospital" (Algorithm 6).
2. The RS sends the ISAN and the ID of the CS (CS_ID) to the WFM.
3. The WFM forwards the CS_ID to the CSM and gets in return the corresponding IP address of the CS (CS_IP).
4. The WFM forwards the ISAN to the CS using the CS_IP.
5. Terminate the process.

Algorithm 5: Set Hospital Location
Implementation Location: RS/practical_tracking.py
Input: Latitude of the Hospital (latitude), Longitude of the hospital (longitude)
Output: Update the global variable "hospital_location" with the specified coordinates

1. Assign the input latitude to the latitude key of the global variable hospital_location.
2. Assign the input longitude to the longitude key of the global variable hospital_location.
3. Terminate the process.

Algorithm 6: Simulation Write Route To Hospital
Implementation Location: RS/simulation_tracking.py
Input: ISAN, Latitude of the Hospital (hospital_latitude), Longitude of the hospital (hospital_longitude)
Output: Updated CSV file containing the appended route coordinates

1. Open the CSV file named simulation_tracking.csv
2. Send a GET request to the MapQuest API from the incident location to the hospital location to retrieve the route coordinates (list of coordinate_latitude and coordinate_longitude pairs).
3. Append the retrieved route coordinates to the CSV file in the following format:
   - RS_ID, coordinate_latitude, coordinate_longitude, hospital_latitude, hospital_longitude, "hospital_location".
   4.1. if the coordinate is the last point on the route, use the following format:
      - RS_ID, coordinate_latitude, coordinate_longitude, "isAtHospital".
4. Terminate the process.

Algorithm 7: Transmit The Ambulance ID of the Selected ISAN for Tracking
Input: ISAN
Output: Main Ambulance ID emitted to the CS frontend tracking page.

1. The CS backend sends the ISAN to the WFM.
2. The WFM forwards the ISAN to the CSM for verification:
    2.1. If verified, the process continues.
    2.2. If not verified, the process terminates.
3. The WFM forwards the ISAN and the CS's IP address (CS_IP) to the ComM.
4. The ComM forwards the ISAN to the RSM.
5. The RSM broadcasts the ISAN to all RSs in operation. Each RS checks its alarm list table for the ISAN. If found, the RS returns a confirmation.
6. the RSM retrieves the ID of the RS that confirms the ISAN.
7. The ComM forwards the retrieved ID to the CS using the CS_IP.
8. The CS backend emits the ID to the frontend tracking page.
9. Terminate the process.

Algorithm 8: Transmit Tracking Data
Input: ISAN
Output: Tracking Data emitted to the CS frontend tracking page.

1. The CS backend sends the ISAN to the WFM.
2. The WFM forwards the ISAN to the CSM for verification:
   2.1. If verified, the process continues.
   2.2. If not verified, the process terminates.
3. The WFM forwards the ISAN and the CS's IP address (CS_IP) to the ComM.
4. The ComM adds the CS_IP to the global active CS IPs list (ACTIVE_CS_IPS) to which tracking data will be sent.
5. Check if len(ACTIVE_CS_IPS) == 1:
   - If True:
   - Clear the global variable that contains the IPs of ambulances currently in operation (PRACTICAL_OCCUPIED_RS_IPS).
   - Start a separate thread to execute the function "periodic_request_occupied_ambulances_ips" (Algorithm 9).
   - Sleep for 1 second.
     ● Practical Case:
       1. Start another thread to execute the function "practical_request_occupied_ambulances_coordinates" (Algorithm 10).
     ● Simulation Case:
       1. Start another thread to execute the function "simulation_request_occupied_ambulances_coordinate s" (Algorithm 11).
6. Terminate the process.

Algorithm 9: Periodic Request Occupied Ambulances IPs
Implementation Location: CM/commMng.py
Output: Updates the PRACTICAL_OCCUPIED_RS_IPS
Variable Definitions:
   - PRACTICAL_OCCUPIED_RS_IPS: a global list, containing the IPs of RSs in operation.

1. While ACTIVE_CS_IPS is not empty:
2. The ComM sends a request to the RSM to retrieve the IPs of ambulances with the "occupied" status.
3. Update the PRACTICAL_OCCUPIED_RS_IPS list with the retrieved IPs.
4. Sleep for 5 seconds.
5. Terminate the process.

Algorithm 10: Practical Periodic Request Occupied Ambulances Coordinates
Implementation Location: CM/commMng.py
Output: Coordinates of ambulances in operation transmitted to active CSs

1. While ACTIVE_CS_IPS is not empty:
   1.1. Create a local copy of the global PRACTICAL_OCCUPIED_RS_IPS: local_active_ambulances_ips
   1.2. For each RS_IP in local_active_ambulances_ips:
      1.2.1. The ComM sends a request to the RS using the RS_IP to retrieve its global "current_location" (see Algorithm 2)
      1.2.2. If the current_location contains the dictionary key "isAtHospital":
         1.2.2.1. The ComM removes the entry corresponding to this RS_IP from the active communications table.
         1.2.2.2. The ComM sends the RS_IP to the RSM so that the RSM updates this RS status from "occupied" back to "active" in its responding systems table.
      1.2.3. The ComM appends the retrieved current_location to the local "positionsSet" list.
   1.3. If the positionsSet is not empty:
      1.3.1. Create a local copy of the global ACTIVE_CS_IPS: local_active_cs_ips
      1.3.2. The ComM sends the positionsSet to all CS instances whose IPs are in local_active_cs_ips.
      1.3.3. The CS backend emits the positionsSet to the frontend.
   1.4. Sleep for 5 seconds.
2. Terminate the process.

Algorithm 11: Simulation Periodic Request Occupied Ambulances Coordinates
Implementation Location: CM/commMng.py
Output: Coordinates of simulated ambulances in operation transmitted to the CS

1. Create a local copy of the global PRACTICAL_OCCUPIED_RS_IPS: local_active_ambulances_ips.
2. For each RS_IP in local_active_ambulances_ips:
    2.1. For each RS, the ComM sends a post request through it's RS_IP
    2.2. Each RS starts in a separate thread the "simulation_start_tracking_single_ambulance" function
        2.2.1. This function begins reading from the CSV file (simulation_coordinates.csv).
        2.2.2. It processes one line at a time with a 5-second interval between lines.
        2.2.3. After reading a line, it updates the global list current_location with the data from the CSV row.
    2.3. While ACTIVE_CS_IPS is not empty:
        2.3.1. if SIMULATION_OCCUPIED_RS_IPS contains an IP that is not present in local_active_ambulances_ips
            2.3.1.1. Do the steps 2.1 and 2.2 for the new RS_IP.
        2.3.2. For each RS_IP in local_active_ambulances_ips:
            2.3.2.1. The ComM sends a request to retrieve the data stored in the globalcurrent_location.
            2.3.2.2. If current_location contains "isAtHospital":
                2.3.2.2.1. The ComM sends a post request to the RS, instructing it to stop the thread reading the corresponding CSV file.
            2.3.2.3. The ComM appends the retrieved current_location to the local "positionsSet" list.
        2.3.3. If positionsSet is not empty:
            2.3.3.1. The ComM sends the positionsSet to the CSs in ACTIVE_CS_IPS.
            2.3.3.2. The CS backend emits the positionsSet to the CSs in ACTIVE_CS_IPS.
        2.3.4. Sleep for 5 seconds.
    2.4. Terminate the process.

Algorithm 12: Handle CS exiting tracking page
Output: Remove CS_IP from ACTIVE_CS_IPS list

1. The CS backend sends the ISAN to the WFM.
2. The WFM forwards the ISAN to the CSM for verification:
    2.1. If verified, the process continues.
    2.2. If not verified, the process terminates.
3. The WFM forwards the ISAN and the CS's IP address (CS_IP) to the ComM.
4. The ComM removes the CS_IP from the global active CS IPs list (ACTIVE_CS_IPS) to which tracking data will be sent.

Algorithm 13: Handle Breakdown
Input: ISAN, RS ID, broken_ambulance_location (optional)
Output: ISAN assigned to new ambulance, if another active ambulance is available

1. The WFM sends the RS IP address (RS_IP) to the RSM:
    1.1. the RSM updates the status of the corresponding RS from "occupied" to "broken" in its responding systems table.
2. The WFM forwards the broken RS_ID to the ComM.
    2.1. Simulation Case:
        2.1.1. The ComM sends a post request to the RS, instructing it to stop the thread reading from the CSV file.
    2.2. The ComM starts a separate thread that sends the broken RS_ID to all active CS instances tracking the RS's:
        2.2.1. Each CS backend emits the broken RS_ID to the frontend, ensuring that all frontend data related to this ambulance is removed from the map.
3. If broken_ambulance_location is not provided:
    3.1. Reapply Algorithm 1 to assign a new RS to the ISAN.
4. If broken_ambulance_location is provided:
    4.1. Reapply Algorithm 1 to assign a new RS to the ISAN, but with the broken_ambulance_location included.
5. Terminate the process.

Timestamp (mm:ss)	Latitude	Longitude
22:11	52.27682	10.50547
22:28	52.27683	10.50547
22:29	52.27683	10.50547
22:45	52.27683	10.50547
23:06	52.27684	10.50547
23:23	52.27737	10.50594
23:40	52.27737	10.50594
23:57	52.27802	10.50643
24:14	52.27892	10.50643
24:30	52.27937	10.50620
24:47	52.27948	10.50519
25:03	52.27979	10.50518
25:19	52.27979	10.50516
25:35	52.27979	10.50515
25:51	52.27980	10.50516
26:08	52.28085	10.50526
26:25	52.28165	10.50824
26:41	52.28162	10.51404
26:58	52.28145	10.51613
27:14	52.28146	10.51619
27:30	52.28144	10.51617
27:46	52.28141	10.51645
28:03	52.28138	10.51652
28:19	52.28139	10.51660
28:35	52.28138	10.51674
28:52	52.28138	10.51672
29:09	52.28133	10.51775
29:26	52.28133	10.51778
29:42	52.28134	10.51775
29:59	52.28128	10.51800
30:15	52.27970	10.51905
30:31	52.27884	10.51950
30:48	52.27857	10.51956
31:05	52.27802	10.51978

31:21	52.27655	10.52058
31:37	52.27552	10.52120
31:55	52.27522	10.52263
32:11	52.27517	10.52387
32:27	52.27517	10.52385
32:43	52.27531	10.52587
33:00	52.27531	10.52857
33:16	52.27458	10.52857
33:32	52.27419	10.52865
33:49	52.27394	10.52844
34:05	52.27380	10.52742
34:22	52.27367	10.52594
34:39	52.27358	10.52532
34:56	52.27361	10.52535
35:12	52.27362	10.52535
35:29	52.27362	10.52536
35:45	52.27361	10.52525
36:02	52.27352	10.52498
36:19	52.27352	10.52489
36:36	52.27357	10.52472
36:52	52.27382	10.52454
37:08	52.27463	10.52452
37:25	52.27491	10.52447
37:42	52.27491	10.52442
37:58	52.27531	10.52350
38:14	52.27528	10.52096
38:15	52.27524	10.52006
38:32	52.27491	10.51822
38:48	52.27432	10.51520
39:05	52.27373	10.51258
39:21	52.27327	10.51038
39:37	52.27281	10.50821
39:54	52.27275	10.50795
43:31	52.27510	10.50510

Table 6: Drive Test: Recorded GPS Coordinates from the Rescuetrack
(rounded to five decimal places)