embrace

pan-European Management of Biological toxin incidents through standaRdisAtion initiatives for Crisis response Enhancement

D5.1
Intoxicated person registration & tracking system

Funded by the European Union

EMBRACE is funded by the European Union's Horizon Europe Research and Innovation funding programme, Grant Agreement N° 101168322.

Lead author(s)	Sebastian Simonsen, Fabio Dijkshoorn, Mees Egberts and Melvin Olsthoorn
Lead beneficiary	PRO
Status	Complete
Version	1.0
Due Date	31-05-2026
Delivery Date	29/05/2026
Dissemination Level	PU
Work Package	WP5
Task	T5.1 Intoxicated persons' care chain tracking
Contributors	PRO, MUG, SMU, ARC, PUI & RAN
Reviewers	BIOX, AIR
Language	English
Format	DEM
Keywords	EMBRACE, intoxicated person tracking, casualty registration, care-chain traceability, biotoxin incidents, CBRN response, digital health records, synthetic population generation, speech-to-text
Abstract	This deliverable presents the EMBRACE Casualty Tracking System for digitally registering monitoring and tracing intoxicated persons during biotoxin-related CBRN incidents. It describes the casualty tracking workflow and backend implementation, together with supporting work on synthetic population generation and speech-to-text assisted data capture.
Referencing this document (filename)	EMBRACE_ D5.1_PRO_Version_1.0
Disclaimer	The information and views set out in this publication are those of the author(s) and do not necessarily reflect the official opinion of the European Communities. Neither the European Union institutions and bodies, nor any person acting on their behalf, may be held responsible for the use which may be made of the information contained herein

Revision history

Version	Date	Partner	Description / Modification
0.1	20.03.2026	PRO	Table of Contents
0.2	13.04.2026	PRO	Sections 2.1
0.3	20.04.2026	PRO	Section 3 and 4 first draft
0.4	13.05.2026	PRO	Completion for review
0.5	27.05.2026	PRO	Incorporation of reviewers comments
1.0	27.05.2026	PRO	Submission ready version

Abbreviations

API	Application Programming Interface
CBRN	Chemical, Biological, Radiological and Nuclear
CQRS	Command Query Responsibility Segregation
CRUD	Create, Read, Update and Delete
DEM	Demonstrator
DTO	Data Transfer Object
EHR	Electronic Health Record
EMS	Emergency Medical Service
FHIR	Fast Healthcare Interoperability Resources
FTX	Field Training Exercise
GI	Gastrointestinal
HTTP	Hypertext Transfer Protocol
IT	Information Technology
IV	Intravenous
JSON	JavaScript Object Notation
LLM	Large Language Model
MCP	Model Context Protocol
NFC	Near Field Communication
PPE	Personal Protective Equipment
PU	Public
QR code	Quick Response Code
REST	Representational State Transfer

SNR	Signal-to-Noise Ratio
UUID	Universally Unique Identifier

Table of Contents

Figures

Tables

EXECUTIVE SUMMARY

The EMBRACE project develops digital capabilities to support the registration, monitoring and traceability of intoxicated persons during incidents involving biological toxins. In such events, responders must be able to identify casualties, record relevant clinical and operational information, and maintain continuity of information across the care chain under time-critical and potentially hazardous conditions. D5.1 presents the development of the Casualty Tracking System, which provides the technical and operational basis for registering casualties, documenting practitioner encounters, and recording observations, conditions and procedures throughout an incident response.

The work builds on earlier research and demonstrator systems developed in TOXI-triage and RESPOND-A, particularly the Tag & Trace approach for casualty tracing and the use of digital records in emergency care-chain environments. These earlier efforts demonstrated the importance of maintaining electronic casualty records, preserving information after decontamination, and supporting traceability in CBRN-related operational contexts. Within EMBRACE, this foundation is extended towards a casualty-centred tracking application focused specifically on intoxicated persons and biotoxin-related scenarios.

The Casualty Tracking Application described in this deliverable is designed to support field practitioners in creating and maintaining structured casualty records. It enables casualties to be associated with an incident, linked to external identifiers such as wristbands, QR codes or NFC tags, and followed through repeated encounters. During these encounters, practitioners can record clinical findings, suspected conditions, measurements and procedures in a structured manner. The system therefore supports both immediate operational use and later reconstruction of the casualty's care-chain history.

A central design principle of the system is traceability. The casualty record acts as the stable digital anchor for all related information, while encounters provide the temporal and operational context in which observations, conditions and procedures are recorded. The backend implementation applies the same event-sourced and domain-driven architectural principles used in earlier EMBRACE tracking work, allowing state changes to be preserved as auditable events and projected into operational read models for retrieval and use.

This deliverable also includes research and supporting components relevant to the broader EMBRACE digital toolchain. A synthetic population generation approach based on Synthea was developed to create privacy-preserving casualty datasets for biotoxin scenarios, with a ricin-specific module used to simulate symptom progression and care-chain events. In addition, speech-to-text and large language model based methods were investigated as assistive mechanisms for capturing casualty information in field conditions where manual data entry may be impractical. These components support development, testing and future validation of the casualty tracking workflow.

Together, the work documented in D5.1 establishes a technical and methodological foundation for intoxicated person registration and tracking in EMBRACE. It provides the core backend modules, operational workflow and supporting research needed to track casualties across an incident response, while identifying current limitations and future development steps for broader operational validation.

1 INTRODUCTION

This deliverable describes the development of the EMBRACE Casualty Tracking System for intoxicated person registration and care-chain tracking. The system is intended to support responders and practitioners during incidents involving biological toxins by providing a structured digital workflow for registering casualties, associating them with an incident, documenting practitioner encounters, and recording casualty-related clinical or operational information. The deliverable focuses on the technical and methodological work carried out under T5.1, while also including supporting research components that contribute to testing, validation and future development of the casualty tracking workflow.

The deliverable is organised around three main areas of work.

a) The first area concerns the generation of synthetic casualty data for biotoxin-related scenarios. Because real-world datasets from biological toxin incidents are rare, sensitive and difficult to standardise, the work investigates the use of Synthea[1] to create synthetic casualty cohorts. A ricin-specific module is included to simulate intoxication-related symptom progression, observations and care-chain events. This synthetic data generation capability provides a controlled basis for system testing and for future research on alerting and decision-support functions.

b) The second area concerns the capture and structuring of casualty information in operational field conditions. In emergency response settings, manual data entry can be difficult because practitioners may be treating casualties, wearing personal protective equipment, or operating in noisy and time-critical environments. The deliverable therefore examines speech-to-text as a possible input method for casualty-related information. It also describes the use of large language models to transform unstructured speech transcripts into structured casualty data that can be mapped to the Casualty Tracking Application Protocol Interface (API). This work is treated as an assistive data-entry approach and not as a replacement for validation, review or controlled backend processing.

c) The third and central area of the deliverable is the Casualty Tracking Application and its backend implementation. The application provides the operational workflow for registering casualties, linking them to external physical and digital identifiers such as wristbands with QR codes and/or NFC tags, starting and ending encounters, and recording conditions, observations and procedures. The backend extends the existing Sample Tracking architecture with casualty-specific modules based on FHIR[2]-aligned concepts. These modules are implemented using the same domain-driven, CQRS[3] and event-sourced principles applied elsewhere in the digital platform, allowing casualty-related state changes to be recorded, audited and reconstructed over time.

The deliverable also defines the operational workflow supported by the system. This includes casualty registration within an incident context, encounter tracking for practitioner-casualty interactions, and the recording of structured clinical or operational records during those encounters. The workflow is

[1] Synthea Github page https://github.com/syntheticahealth/synthea

[2] Fast Healthcare Interoperability Resources (FHIR) is a technical standard for the healthcare domain https://fhir.org/

[3] Command Query Responsibility Segregation https://learn.microsoft.com/en-us/azure/architecture/patterns/cqrs

designed so that the casualty record remains the stable digital anchor, while encounters provide the temporal and operational context for observations, conditions and procedures. This structure supports continuity of information across repeated contact moments and enables later reconstruction of the casualty's care-chain history.

Finally, the deliverable documents the current technical implementation, known limitations and next development steps. The implementation section describes the main system architecture, core entities, API surface, access control considerations, and event logging approach. The limitations section identifies the current boundaries of the work, including the restricted scope of the ricin-focused synthetic population generator, the selected subset of FHIR-based casualty modules, and the need for further operational validation. The next steps focus on improving commander-level situational awareness and evaluating the workflow during field trial activities.

In this way, D5.1 breaks down the Casualty Tracking System from research basis to operational workflow and technical implementation. It explains how the system is structured, how casualty information is captured and represented, and how the developed components contribute to intoxicated person registration and tracking within the broader EMBRACE response framework.

2 RESEARCH & METHODOLOGY

2.1 Synthetic Population Generation via Synthea

The EMBRACE Project needs to develop and validate digital tools which are capable of registering, tracking and providing downstream decision support for intoxicated persons. However, access to sufficient numbers of structured real world casualty data sets from biotoxin incidents is severely limited due to the rarity of such events, heterogeneity in both the presentation and operational context of each incident, and the sensitivity of the medical and incident response data associated with each event. As a result, real datasets may be unavailable, incomplete, difficult to harmonize across organizations, or unsuitable for use in system development and machine learning applications.

As such, a synthetic population generation approach is used within EMBRACE utilizing Synthea (Walonoski et al., 2018) to create a baseline synthetic casualty population. The main advantage of using Synthea is its ability to generate privacy preserving data suitable for technical development, testing and validation without dependence on real casualty records. Additionally, the ricin specific module utilized within EMBRACE allows the project to define and control incident-specific progression logic, producing structured, reproducible trajectories of symptom development, severity changes and operational care-chain events. From an analytical perspective, the resulting synthetic cohort provides a foundation for future model development by generating longitudinal training data which captures temporal patterns of clinical deterioration and escalation.

2.1.1 Rationale for Synthetic Cohorts in Biotoxin Research

The generation of synthetic cohorts in biotoxin research is driven by three types of limitations: ethical, practical, and scientific. Incidents of toxic exposure to biotoxins are rare, and the associated data may exist in various forms (clinical, emergency response, public health) within several organizations. Data related to toxicity is highly sensitive with respect to confidentiality and privacy issues. Therefore, it is very challenging to develop sufficient, standardized, and sharable datasets for research, systems validation and/or training machine learning algorithms.

This issue is significant for the EMBRACE project. The project's purpose is to provide tools to track digitally intoxicated individuals and to assist in decision-making processes regarding those individuals. To accomplish this goal, the project needs to represent casualties' longitudinal progressions over time, including both static registration data and evolving clinical statuses. It is noted that there is an extreme scarcity of longitudinal data regarding biotoxin exposures. Additionally, each incident of biotoxin poisoning is unique due to variations in type of toxin exposed to, method of exposure, amount of toxin ingested/exposed to, the environment the individual is in at the time of exposure, and how long after exposure treatment occurs. Due to these variables, using only empirical case studies as a basis for development and testing does not result in a robust enough dataset for either development or testing purposes.

Synthetic cohorts can serve as a viable solution. Synthetic cohorts allow researchers to generate hypothetical populations of casualties that are similar to actual casualty populations. Therefore, synthetic cohorts allow researchers to generate incident-relevant datasets without the need to utilize directly identifiable personal health care information. This aspect of synthetic cohorts is of particular importance in multi-organization collaborative projects like EMBRACE where all parties involved

require access to consistent and reproducible data sets for their respective roles in developing, integrating and evaluating the systems the consortium develops. Synthetic cohorts can also be recreated/extended/made more complex as the needs of the system being evaluated grow. Additionally, synthetic cohorts enable researchers to test systems under varying assumptions/scenarios.

In the case of the EMBRACE project, Synthea is utilized to generate a baseline population from which biotoxin-exposed casualties are simulated. The use of Synthea enables the consortium to simulate casualty level heterogeneity (e.g., age, sex, pre-existing conditions, etc.) that affects the presentation and progression of intoxication.

2.1.2 Modelling Biotoxin-specific Symptom Progression

To go beyond simple, generic synthetic casualty creation, EMBRACE creates a customized Synthea module to create synthetic casualties with ricin (from castor bean) poisoning. The goal of this project is to demonstrate how the clinical status of synthetic casualties evolves over time after biotoxin exposure by transforming the general population of synthetic casualties into relevant operational trajectories.

The model is based upon the fact that each synthetic casualty starts as a baseline person with common demographic and health information; when a synthetic casualty experiences a ricin exposure, they enter a trajectory representing the development, exacerbation and eventual resolution of the symptoms associated with ricin poisoning. This module is based upon a comprehensive clinical review and case series authored by Challoner and McCarron (1990), which evaluates 424 reported incidents of castor bean poisoning (and includes three new incident reports: a child who consumes ≥ 2 seeds from the plant and two adults each consume four seed pods from the plant); the authors document in detail, using evidence-based data, the onset and progression of symptoms due to acute gastroenteritis, dehydration, Gastrointestinal bleeding, hemolysis, and hypoglycemia; the diagnostic findings obtained during medical evaluations; the treatments provided (IV fluids and supportive care); the clinical outcomes observed; and the mortality rate experienced among those exposed (8.1% if left untreated vs. 0.4% if treated). As there is no evidence of delayed cytotoxic effects; these data points are converted into Synthea state transition(s), conditional logic, and temporal rules to produce realistic longitudinal records. Due to the limited availability of publicly accessible research on inhaled Ricin poisoning the Synthea Ricin module is for ingested Ricin symptoms and effects.

The EMBRACE ricin module generates longitudinally-coherent and privacy-protective casualty trajectories that represent the range of possible variations in real-time in terms of: onset timing and combinations of symptoms; In addition to enabling thorough testing of tracking workflows, these databases also provide structured training data sets for developing "smart" alerting systems designed to identify clinically-relevant changes in an individual's condition at an early stage.

In conclusion, the EMBRACE ricin-specific Synthea module represents a significant methodological advancement for the EMBRACE program. The ricin module transforms a generic synthetic population generator into a biotoxin-focused simulation tool that can generate casualty trajectories that are temporally-rich and operationally-relevant for the purpose of developing, validating and further researching intelligent alerting systems.

2.2 Speech-to-text Input for Capturing EHR Data

2.2.1 Speech-to-text

Speech-to-text is investigated as a possible method for capturing casualty-related information at the incident scene. A central motivation for this research is the practical challenge of documentation in time-critical emergency care. First responders are often required to use both hands to assess, stabilize, and treat casualties, making manual data entry impractical or disruptive during active intervention. Requiring practitioners to pause lifesaving actions to document observations, treatments, or incident details may delay care and increase cognitive and operational burden. In the context of biotoxin incidents, practitioners may also be wearing PPE, making touchscreen interaction even less suitable. This research therefore evaluates spoken input as a less intrusive method for capturing responder-provided information, with the specific goal of converting speech into an accurate and usable transcript that can support subsequent documentation and analysis.

2.2.1.1 Scope

The research compares multiple locally run speech-to-text models to determine their suitability for responder-facing documentation in emergency care contexts. The practical evaluation focuses primarily on CohereLabs/cohere-transcribe-03-2026 and SenseVoiceSmall, as these models are considered relevant candidates for converting spoken responder input into usable transcripts. Testing is conducted locally, reflecting the broader objective of assessing whether speech-to-text functionality can be deployed in a local or controlled environment rather than depending entirely on external cloud-based services. The systems are evaluated against criteria relevant to field use, including multilingual capability, robustness to background noise, feasibility of local or controlled deployment, inference speed, and failure behaviour when audio quality degrades beyond reliable transcription.

The noise test is designed to compare how the speech-to-text models behave when clean spoken audio is made more difficult by added background noise. Clean speech samples are mixed with babble noise and police siren noise to approximate elements of an emergency environment. The same noisy samples are tested on both Cohere Transcribe and SenseVoiceSmall, allowing their outputs to be compared under equivalent conditions. Multiple signal-to-noise ratio levels are used, including -5 dB, -12.5 dB, and -20 dB SNR, to progressively increase the difficulty of the audio. The model outputs are then compared against the original reference transcript to assess how closely each model preserves the intended speech, how much information is lost or altered, and how each model behaves when the audio becomes difficult to transcribe.

The language test is designed to compare how the speech-to-text models perform on non-English input. In addition to the noise tests, the models are evaluated using German and French speech samples to assess whether they can produce usable transcripts across multiple languages. The objective is not to fully validate multilingual field performance, but to obtain an initial indication of how well the models handle different languages, spelling, wording, punctuation, and named entities. The outputs are compared against reference transcripts to assess how closely each model preserves the original speech and whether the resulting transcript remains usable for later processing.

2.2.1.2 Outcomes

Test Condition	Cohere Transcribe	SenseVoice Small	Main Takeaway
English, -5 dB noise	Partly usable output, but with errors	Partly usable output, but with different error types	Both models remain somewhat useful, but error profiles differ
English, -12.5 dB noise	Remains closer to the reference transcript	Shows stronger degradation	Cohere is more robust at moderate-to-severe noise levels
English, -20 dB noise	Produces a longer but incorrect sentence	Often collapses to minimal output	Both fail, but failure behaviour differs; concise failure may be safer than fluent hallucination
German tests	Generally close to the reference, with mostly punctuation, casing, or minor wording differences	Less reliable; sometimes wrong language detection or unusable output	Cohere performs better in German
French, first test	Less conclusive due to unsuitable clips/reference transcripts	Less conclusive for the same reason	Dataset quality limited direct comparison
French, cleaner second test	Closer to the reference text, though named entities and place names remain difficult	Weaker relative performance	Cleaner samples confirm Cohere's stronger performance, but entity recognition remains challenging

Table 1. Comparison between Cohere and Sensevoice models in test conditions

Overall, the results show that speech-to-text can produce useful transcripts, but reliability depends strongly on language, noise level, model behaviour, and processing speed. The negative findings are important: lower latency does not necessarily imply higher reliability, multilingual support is inconsistent, and extreme noise can lead to unsafe outputs. For a medical workflow, the model should therefore not only transcribe accurately, but also fail safely when the audio is too unclear.

2.2.2 Parse Text for Casualty Tracking API

Once the text from the speech-to-text module is returned, it needs to be structured to become the casualty's record in the Casualty Tracking system. This structuring of the raw speech is a suitable job for Large Language Models (LLM)s[4]. LLMs are able to handle and parse large amounts of human language while returning structured data. They can be run locally on devices at the incident scene or in the cloud.

[4] https://aws.amazon.com/what-is/large-language-model/

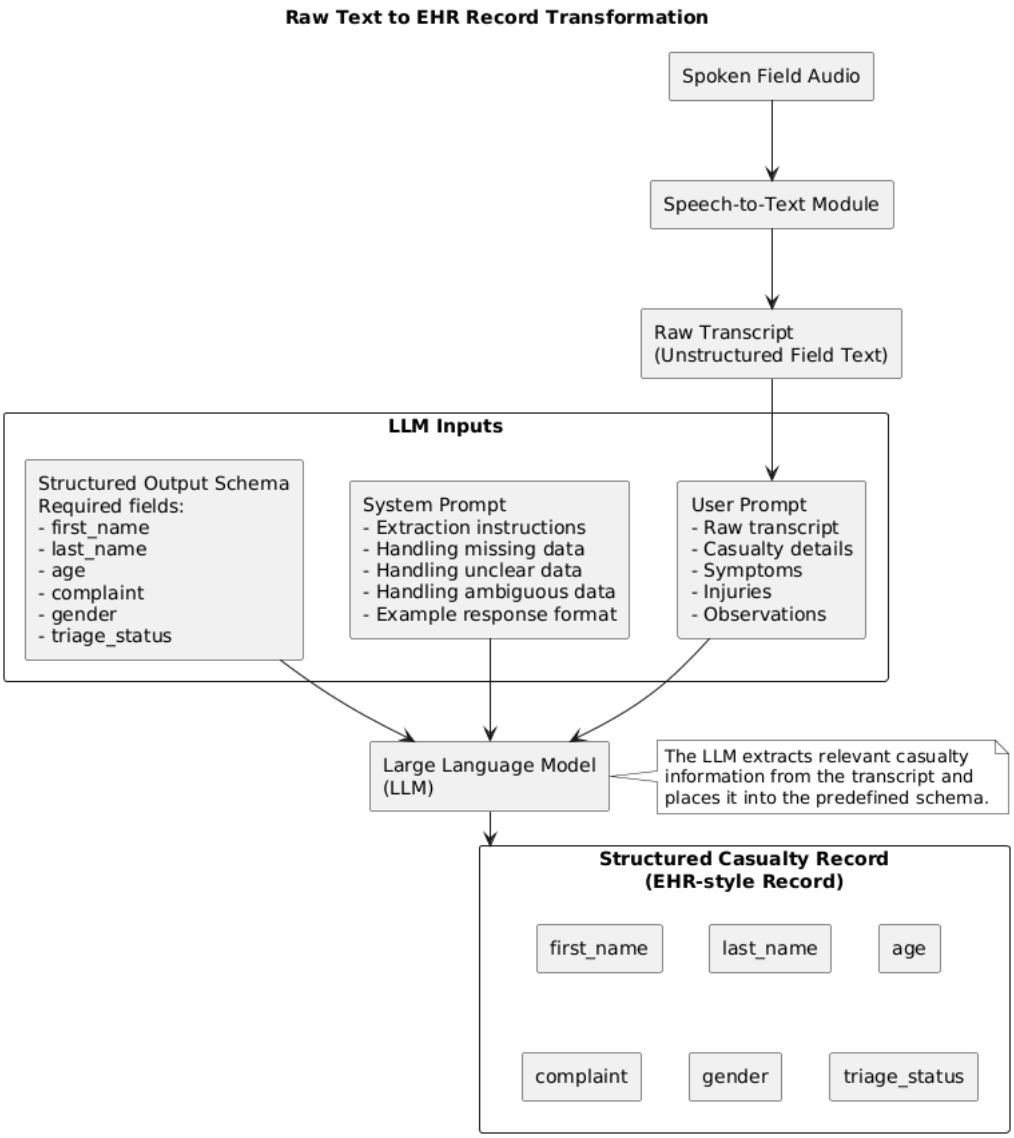


Figure 1. Component diagram of structuring raw text to EHR

2.2.2.1 Raw Text to EHR Record Transformation

In the Casualty Tracking Application LLM is utilised to convert unstructured conversation text field information into a structured casualty record. The information is first captured as spoken audio in the field and then converted into a raw text transcript by the speech-to-text system described in 2.2.1 section. This transcript may contain casualty details such as first name, last name, age, symptoms, injuries, complaints, observations, or other information reported by the responder.

The transcript is provided to the LLM through the user prompt. The user prompt contains the actual information that needs to be processed. In this case, it represents the raw field transcript that the model must analyse and extract information from.

The LLM also receives a system prompt. The system prompt defines how the model should handle the input. It describes the extraction task, explains which information should be identified, and gives instructions for handling unclear, missing, or ambiguous details. The system prompt can also include an example of the expected response, so the model has a clear reference for the type of structured output that should be produced.

In addition to the prompts, the application provides a structured output schema with each LLM request. This schema defines the exact fields that the model is expected to return, such as first name, last name, age, and complaint. It also defines the required structure of the response, making the output easier for the application to process and validate.

The schema below:

```
 {
  "type": "object",
  "properties": {
    "first_name": {
      "type": "string"
    },
    "last_name": {
      "type": "string"
    },
    "age": {
      "type": "number"
    },
    "complaint": {
      "type": "string"
    },
    "gender": {
      "type": "string"
    },
    "triage_status": {
      "type": "string"
    }
  },
  "required": [
    "first_name",
    "last_name",
    "age",
    "complaint",
    "gender",
    "triage_status"
  ]
}
```

The system prompt and the schema serve different purposes. The system prompt explains the task and gives behavioural instructions to the model. The schema defines the shape of the returned data. Together, they guide the model to extract relevant information from the transcript and return it in a predictable, machine-readable format.

During processing, the model applies the instructions from the system prompt to the information contained in the user prompt. It then generates a structured output where relevant casualty information is placed into the correct predefined fields. For example, if the transcript states that the casualty's name is John Smith, 42 years old, and is complaining of chest pain, the model should return those details in the corresponding fields for first name, last name, age, and complaint.

This structure is important because raw speech-to-text transcripts are not always clean or complete. Spoken field audio may include background noise, incomplete sentences, corrections, unclear

phrasing, or recognition errors from the transcription system. The LLM therefore needs clear instructions and a defined output structure to reduce inconsistency and improve extraction accuracy.

The model's output is still not automatically guaranteed to be correct. Because LLMs generate responses probabilistically, they may omit information, place details in the wrong field, or infer information that was not explicitly stated. For this reason, the prompt, schema, and evaluation process are important parts of improving the reliability of the system.

To support this, a prompt engineering tool described in section 2.2.2.2 was created to test how accurately the LLM extracts casualty information from transcripts. The tool allows different prompt versions and model configurations to be tested repeatedly. The extracted outputs can then be evaluated to identify mistakes, compare performance, and refine the system prompt. This makes the development process more systematic, because prompt changes can be measured against structured test cases rather than judged only through informal manual testing.

2.2.2.2 LLM System Prompt Engineering Tool

To improve the reliability of the voice-to-text in the Casualty Tracking Application, this research develops a dedicated benchmarking tool, casualty-llm-benchmark. The tool is designed to evaluate how consistently a large language model can transform raw speech-to-text transcriptions from casualty and paramedic interactions into a predefined structured data format. Its purpose is not to operate as a production casualty-record system, but to provide a repeatable evaluation framework for measuring extraction quality, comparing model behaviour, and refining the system prompt used by the application.

The benchmark reflects a realistic field reporting workflow. In the intended use case, casualty information is first spoken by a responder and converted into an unstructured transcription. This transcription may contain incomplete phrasing, conversational noise, corrections, implicit context, and medically relevant information expressed in natural language rather than fixed form fields. The language model is then responsible for interpreting the transcription and returning structured casualty information in a consistent format suitable for downstream use. The benchmark is therefore designed to evaluate this specific transformation step and to identify where the model performs reliably, where it loses information, and where it introduces errors.

The input data for the benchmark is derived from the publicly available **EMSDialog** (Ge, X., Murtaza, S., Cortez, A., and Alemzadeh, H.) research paper and its associated GitHub repository. EMSDialog provides synthetic multi-speaker emergency medical service dialogues generated from real-world electronic casualty care reports, with speaker roles and turn-level topics intended to reflect operational Emergency Medical Service (EMS) interactions. The paper describes the dataset as containing 4,414 realistic multi-party EMS conversations annotated with 43 diagnosis classes, and the authors state that the datasets and code are publicly available through the project repository. In this research, the practitioner and casualty text used for benchmark cases is taken from those recorded and transcribed scenario materials, providing domain-relevant input examples for testing whether the LLM can convert responder-style speech transcripts into structured casualty information.

The evaluation process separates the system into two model roles. The first is the primary model, which performs the extraction task. The second is the benchmark model, typically a stronger or more reliable model, which reviews and scores the primary model's output. This separation allows different

models, prompt versions, and provider configurations to be compared under controlled conditions. It also provides a repeatable method for determining whether changes to the system prompt improve extraction accuracy, reduce error rates, or introduce new failure modes.

The tool supports multiple model providers, including OpenAI, Anthropic, and Google, depending on the configured environment. During each benchmark run, the system executes a batch of casualty-related information extraction tasks, records the structured output, evaluates the result, and captures scoring data, qualitative assessment notes, model configuration, and token usage. This makes the evaluation process reproducible and allows prompt versions to be compared over time using a consistent testing procedure.

The primary research objective is to use benchmark feedback to improve the LLM system prompt. Rather than relying on informal manual testing, the benchmark provides structured evidence about recurring extraction errors. These may include missed casualty details, incorrectly assigned fields, hallucinated information, inconsistent formatting, or failure to preserve uncertainty from the original transcription. By analysing these results across multiple test cases, the prompt can be refined to better instruct the model on how to handle ambiguous, incomplete, or noisy voice-to-text input.

This methodology supports iterative prompt development. A prompt version can be tested against the benchmark set, reviewed using the evaluation output, modified, and then tested again under the same conditions. Over time, this process enables the development team to improve extraction reliability, compare model and provider choices, and document the quality of the structured casualty extraction process. The benchmark therefore provides a lightweight but systematic research method for improving the LLM component of the voice-to-text in the Casualty Tracking Application.

3 CASUALTY TRACKING APPLICATION

The Casualty Tracking Application is developed to support the digital registration and monitoring of intoxicated persons during a biological toxin incident. The system provides a structured workflow for identifying casualties, documenting practitioner contact moments, recording clinical observations and registering procedures performed during the operational care chain.

The application is implemented as a standalone mobile application. It is separate from the Sample Tracking mobile application described in EMBRACE's *D3.1 Sampling devices and sample tracking – 1st Iteration*, and does not share the sample collection, collection sealing or custody transfer workflows. However, the casualty tracking capability uses the same backend foundation as the Sample Tracking API. This decision was made because both systems use the same underlying architectural principles, including Domain-Driven Design, Command Query Responsibility Segregation and Event Sourcing.

The Casualty Tracking API is implemented by adding casualty-specific modules to the existing backend API. These modules are based on the FHIR standard and provide the core structures required to register intoxicated persons and document their clinical status during the response process.

3.1 System Architecture Overview

The Casualty Tracking Application consists of two main components: a standalone mobile application and the backend API. The mobile application provides the operational interface used by practitioners to identify casualties, start and end encounters, and record observations or procedures. The backend API serves as the authoritative system layer for authentication, authorisation, data validation, event recording and query access.

The mobile application is the primary operational interface for casualty tracking. Practitioners use it in the field or care-chain environment to scan a casualty identifier, retrieve the casualty record and document the encounter.

The backend API contains the casualty tracking modules and ensures that all operations are processed through a controlled and auditable system layer. This ensures that casualty records, encounters, observations, conditions and procedures remain structured and traceable.

3.1.1 Mobile Application

The Casualty Tracking mobile application is designed for practitioners involved in the registration, assessment and monitoring of intoxicated persons. The application enables authorised users to identify a casualty, start and end an encounter, and record observations or procedures while the encounter is active.

The application does not contain sample tracking functionality. It does not support sample collection, NFC-based sample container registration, collection sealing or custody transfer. Its workflow is focused exclusively on person-centred registration and care-chain documentation.

The mobile application uses the same general technology approach as the other EMBRACE mobile applications, but the workflow and user interface are specific to casualty tracking.

3.1.2 Backend API

The backend API serves as the authoritative core of the Casualty Tracking Application. The casualty tracking functionality is implemented through additional modules that are incorporated into the existing Sample Tracking API as separate bounded contexts.

The casualty-specific modules are based on FHIR concepts (see below). They provide a structured representation of the casualty, the contact moments between practitioners and casualties, and the clinical information recorded during those contact moments.

FHIR Concepts Summary: *FHIR (Fast Healthcare Interoperability Resources) is an HL7 standard designed to enable seamless, modern exchange of health data across systems. It uses modular "resources"—such as Patient, Observation, and Medication—combined with widely adopted web technologies like REST APIs, JSON, and XML to simplify integration. FHIR improves on earlier HL7 standards by being easier to implement, highly flexible through profiles and extensions, and suitable for mobile apps, EHRs, cloud systems, and large-scale providers. Its goal is to deliver consistent, granular, and interoperable health information exchange that supports clinical workflows, patient access, and innovation across diverse healthcare environments.*

Module	Description	Operational Use
Casualties	Represents the intoxicated person being registered and tracked within the care chain.	Serves as the central record to which encounters, observations, conditions and procedures are linked.
Encounters	Represents time-bounded contact moments between a practitioner and a casualty.	Starts when a practitioner scans the casualty's external identifier and ends when the identifier is scanned again.
Observations	Represents measured or reported clinical findings recorded during an active encounter.	Used to document symptoms, measurements or assessment findings while the practitioner is with the casualty.
Conditions	Represents relevant health conditions or suspected intoxication-related states.	Used to document the interpreted clinical status or suspected condition of the casualty.
Procedures	Represents actions or interventions performed by a practitioner during an active encounter.	Used to record care-chain actions or interventions performed for the casualty.

Table 2. Casualty-specific modules in backend API based on FHIR

These modules provide the core data structure for the casualty tracking workflow. The purpose is not to implement a complete electronic health record system, but to provide the relevant subset of structured information required for intoxicated person registration and tracking during an incident.

3.1.3 Relationship with Sample Tracking

The Casualty Tracking Application and the Sample Tracking Application described in D3.1 Sampling devices and sample tracking - 1st Iteration are separate operational tools. They support different workflows and are used for different purposes. The Sample Tracking Application supports sample

registration, physical-digital binding, collection sealing, custody transfer and analysis linkage. The Casualty Tracking Application supports casualty identification, practitioner encounters, observations, conditions and procedures.

The relationship between the systems is mainly architectural. The casualty tracking modules are added to the same backend API because the same event-based and domain-driven approach is suitable for both use cases. This avoids unnecessary duplication of backend infrastructure while keeping the application workflows separate.

At application level, the systems remain independent. The Casualty Tracking mobile application does not depend on the Sample Tracking mobile application and does not require the user to perform any sample tracking actions.

3.2 Role Model and Access Control

Access to the casualty tracking modules is governed through role-based access control. Only users assigned the role "practitioner" are permitted to perform create, read, update and delete operations within the casualty tracking API modules.

The practitioner role represents personnel authorised to register and maintain casualty-related information. Users without this role are not permitted to perform CRUD operations on casualty, encounter, observation, condition or procedure records.

The "commander" user is able to see an overview of the incident scene in terms of casualties registered in the system.

The following table summarises the access model for the casualty tracking modules:

Module	Principal Function	Permitted Role
Casualties	Register and maintain casualty records.	Practitioner, Commander (monitor)
Encounters	Start, maintain and end practitioner contact moments.	Practitioner
Observations	Register and maintain clinical findings or measurements during an active encounter.	Practitioner
Conditions	Register and maintain relevant health conditions.	Practitioner
Procedures	Register and maintain performed interventions or actions during an active encounter.	Practitioner

Table 3. Role-Based Access Control Matrix for Casualty Tracking API Modules

This role model ensures that casualty-related records are created and modified only by authorised personnel. It also ensures that all state-changing actions can be linked to an authenticated practitioner.

3.3 Identification and Traceability

The Casualty Tracking Application uses external identifiers to support reliable casualty lookup in field conditions. These identifiers may be implemented through NFC tags, QR codes or other supported identifier types. The identifier is physically attached to the casualty and used to retrieve the corresponding digital casualty record.

The casualty record is the stable entity within the system. External identifiers are used to locate this record, but they do not act as the permanent source of identity for the casualty's care-chain history. If an identifier is damaged, removed or incorrectly assigned, a new identifier can be linked to the same casualty record.

The same identifier-based lookup mechanism is also used to control the encounter lifecycle. A scan by a practitioner can start an encounter when no active encounter exists for the casualty-practitioner contact. A subsequent scan can close the active encounter. In both cases, the identifier is used to retrieve the casualty first; the encounter is then linked to the casualty record.

Traceability is supported by recording the relationship between the casualty record, the active encounter and the practitioner actions performed during that encounter. For each casualty, the system can determine when encounters occurred, which practitioner was involved and what observations or procedures were recorded during each contact moment.

This structure ensures that replacing an external identifier does not break the documented care-chain history. The casualty record remains the anchor for all related records.

4 APPLICATION OPERATIONAL WORKFLOW

This section describes the operational workflow supported by the Casualty Tracking API extension. The workflow mirrors the structure of the operational workflow section in D3.1, but focuses on casualty and casualty tracking rather than sample collection, sample analysis or custody transfer. Those sample tracking workflows remain documented in D3.1 and are not repeated here.

The Casualty Tracking API supports three primary operational processes:

- casualty registration within an incident context;
- encounter tracking for casualty interactions;
- recording of conditions, observations and procedures within an encounter.

The workflow is described from the perspective of an authorised consuming client (technical application getting data from the API), such as the Casualty Tracking mobile application or the Supervisor Dashboard.

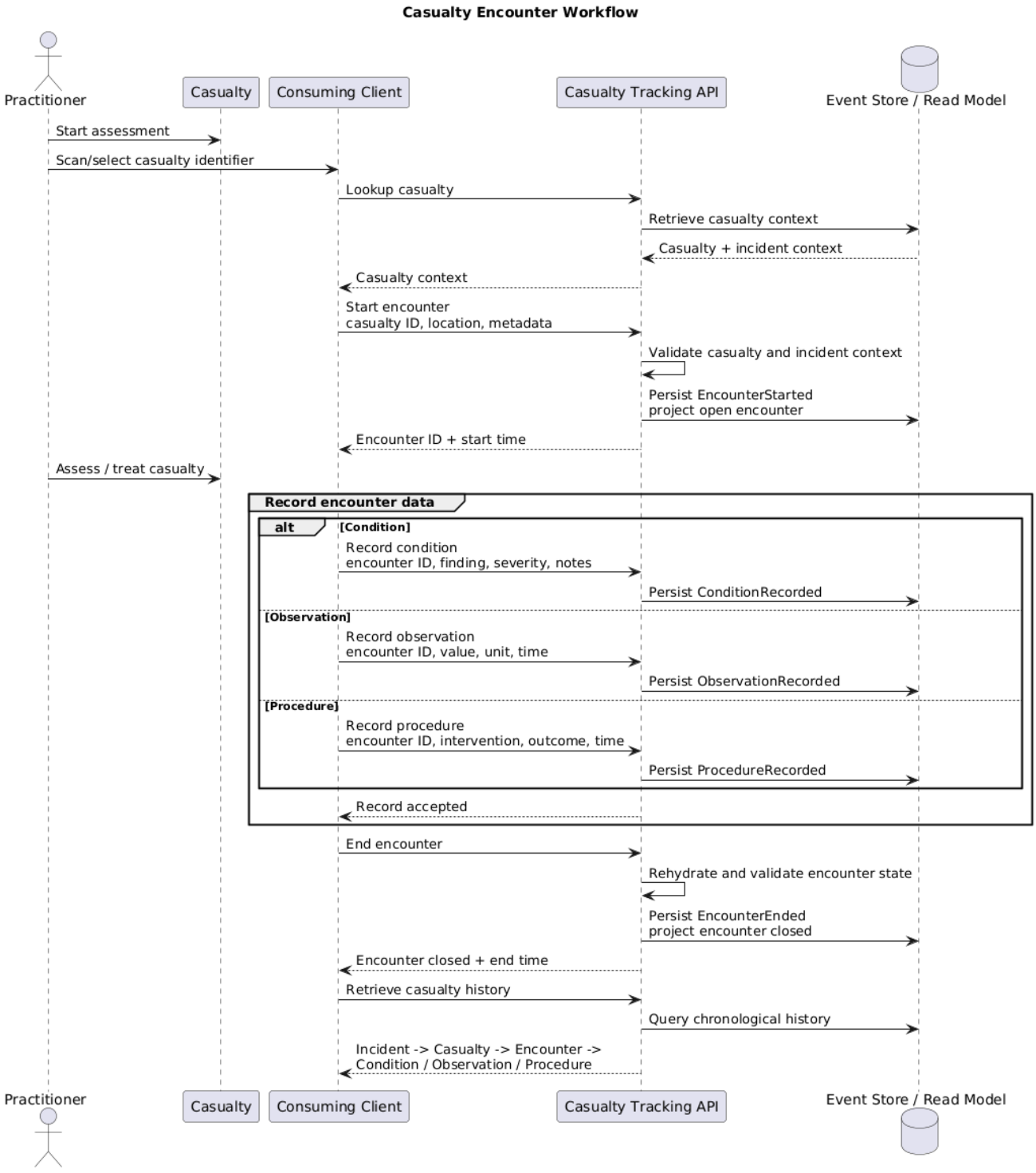


Figure 2. Casualty Encounter Diagram from the perspective of a practitioner using the Casualty Tracking System

4.1 Casualty Registration Process

The Casualty Registration Process is executed when a responder or authorised client creates a digital record for a person involved in an incident. The purpose of the workflow is to ensure that each casualty is registered within a defined operational context, assigned a stable digital identity and linked to an external field identifier that can be used throughout the incident response.

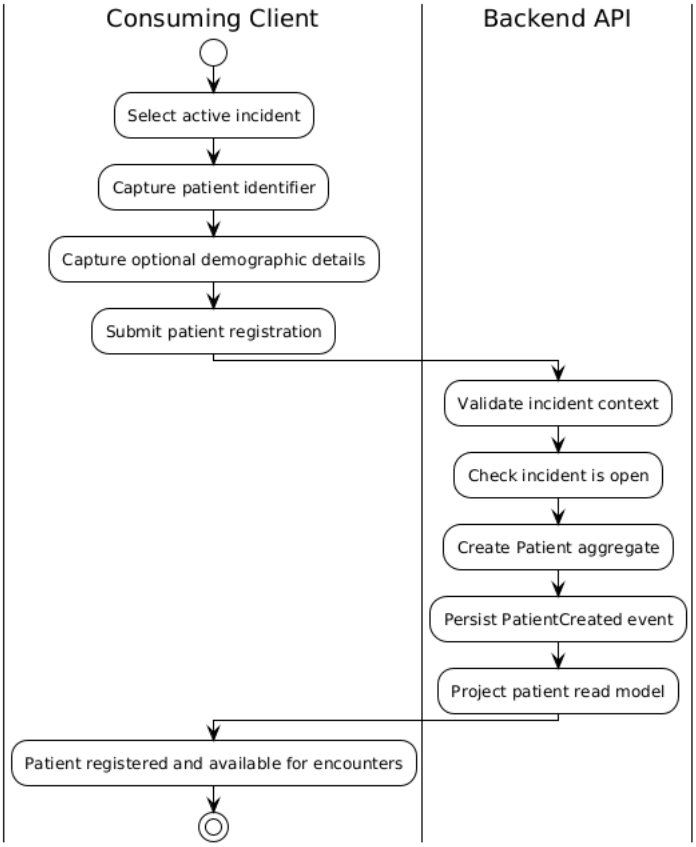


Figure 3. Activity diagram of casualty registration

4.1.1 Incident Association

Before a casualty can be registered, the consuming client must establish the incident context under which the casualty record will be created. This can be achieved by selecting an existing incident, scanning an incident reference, or receiving the incident identifier from another operational system. The mechanism used by a specific client is outside the scope of this section; the API requires that casualty records are linked to a valid incident identifier.

This step establishes the operational frame for all subsequent casualty tracking actions. Once the incident context has been established, each casualty and encounter created through the API can be traced back to the same incident. Conditions, observations and procedures are then traced to the incident through the encounter and casualty hierarchy. This prevents un-associated casualty records and ensures that casualty data can be interpreted in relation to a defined event, location or response operation.

The backend validates the incident reference before accepting the casualty registration. If the referenced incident has been concluded, new casualty registration is rejected. This preserves the integrity of the incident lifecycle and prevents records from being added to a closed operational context.

4.1.2 Casualty Identifier Capture

After the incident has been selected, the user or consuming client captures the casualty's external identifier. This identifier represents the field-facing reference used to recognise the casualty outside

the API. Depending on the operational setup, it may correspond to an NFC tag, QR code, wristband, triage card or identifier received from another information system.

The external identifier is distinct from the internal Universally Unique Identifier (UUID) generated by the API. The internal UUID provides a stable system reference, while the external identifier provides continuity between the digital record and the physical person or field marker. This distinction allows responders and external systems to work with practical identifiers while the backend maintains globally unique internal references.

The casualty identifier becomes the principal operational link used by consuming clients to retrieve or update casualty-related information. Once the casualty is registered, encounters and clinical-event records can be associated with the casualty record.

4.1.3 Demographic and Contextual Data Capture

During registration, the client may provide optional demographic information such as given name, family name, sex at birth and birth date. These fields are not mandatory because, in incident response settings, casualties may be unidentified, unconscious, partially identified or registered under field identifiers before formal identity is known.

The workflow therefore supports both minimal and enriched registration. In the minimal case, the casualty can be created with an incident reference and external identifier only. In the enriched case, demographic attributes are added at registration time or updated later when more reliable information becomes available.

This approach allows casualty tracking to begin immediately while preserving the ability to improve record completeness as the operational situation develops.

4.1.4 Creation of the Casualty Record

When the registration request is submitted, the backend validates the payload[5], checks the incident context and creates the Casualty aggregate[6]. The creation action is recorded as a domain event and projected into the read database so that the casualty becomes available for subsequent encounter and clinical-event workflows.

The casualty record is then returned to the consuming client with its generated internal identifier. From this point onward, the casualty can be referenced by API clients when starting encounters. Conditions, observations and procedures are recorded within an encounter, and therefore inherit their casualty association through that encounter.

4.2 Encounter Tracking Process

The Encounter Tracking Process describes how interactions with a casualty are recorded over time. An encounter represents a time-bounded interaction, assessment, handover or care activity involving a casualty. A single casualty may have multiple encounters during an incident response.

[5] https://www.docsie.io/blog/glossary/payload/

[6] https://www.eventsourcing.dev/best-practices/designing-aggregates

[7]

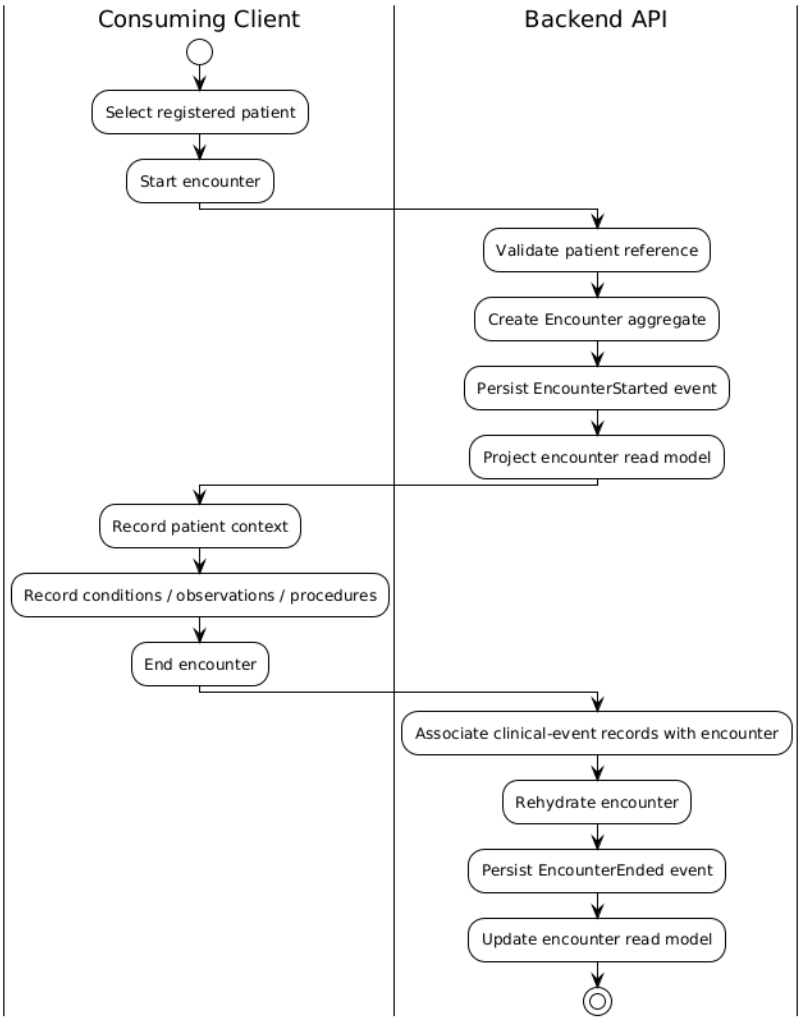


Figure 4. Activity diagram of encounter tracking

4.2.1 Initiation of an Encounter

An encounter is initiated when a user or consuming client begins a casualty interaction. The client provides the casualty identifier and location information for the point at which the encounter starts. The backend records the start timestamp and creates the encounter record.

This step establishes a defined temporal and spatial context for the interaction. Any condition, observation or procedure recorded during the workflow is linked to the encounter, and the casualty association is derived from the encounter's casualty reference. This makes the encounter the container for casualty-related clinical and field records captured during a specific interaction.

The encounter creation event does not replace or modify the casualty record. Instead, it extends the casualty's operational history with a new interaction. This allows repeated assessments to be captured without losing earlier casualty state.

[7] Rehydrate in event sourcing architecture context https://learn.microsoft.com/en-us/azure/architecture/patterns/event-sourcing#:~:text=is%20known%20as-,rehydration,-.%20It%20can%20occur

4.2.2 Casualty Context During an Encounter

During an active encounter, the consuming client can retrieve the casualty record and display relevant incident and identifier information. The encounter acts as the context in which new clinical or operational data is captured.

Typical information associated with an encounter includes the casualty reference, location, start time, encounter type, priority or reason code, depending on the client workflow. These attributes allow a later reviewer or integration system to understand why the encounter occurred and how it fits into the wider incident response.

The encounter model also supports repeated interactions. A casualty can be registered once and then linked to multiple encounters, for example initial assessment, transport handover, decontamination review and clinical reassessment.

4.2.3 Closure of an Encounter

When the interaction has ended, the consuming client sends a request to close the encounter. The backend rehydrates the encounter state, records the end timestamp and persists an encounter-ended event. The read model is updated so that the encounter is no longer treated as open.

Closing the encounter is an important lifecycle step because it defines the endpoint of a specific interaction. Clinical-event records captured before closure remain linked to the encounter, while later records can either be associated directly with the casualty or linked to a subsequent encounter.

This separation supports chronological reconstruction of casualty interactions across the response operation.

4.3 Recording Conditions, Observations and Procedures

The third operational process concerns the recording of casualty-related clinical or field data. Conditions, Observations and Procedures provide the main structures for this purpose. In this model, they are associated with an encounter. Because each encounter is associated with exactly one casualty, the casualty relationship is inherited through the encounter rather than stored as a separate primary workflow association.

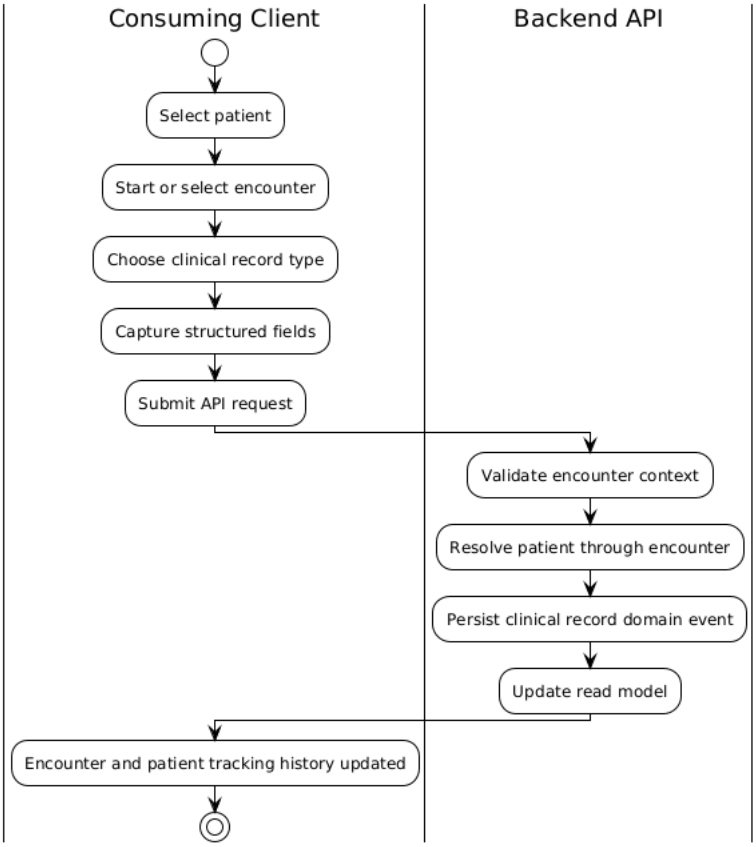


Figure 5. Activity diagram of condition, observation and procedure recording

4.3.1 Recording Conditions

A condition represents a casualty state or clinical finding identified during an encounter. The condition record can include coded condition information, category, clinical status, verification status, severity, body site, timing information and notes.

This allows the workflow to capture immediate field findings in the interaction where they were identified. For example, a responder may record a suspected exposure-related condition during an initial encounter, while later updates can refine its status or severity as more information becomes available within that encounter context.

By recording conditions as structured encounter records rather than free text only, the API supports consistent retrieval, filtering and later integration with other systems.

4.3.2 Recording Observations

An observation represents a measurement, finding or assessed value. Observations may include vital signs, triage findings, exposure indicators, field measurements or other recorded values. Each observation can include a code, category, status, effective time, issued time, value, unit, interpretation, body site, method and notes.

The observation workflow is designed to support both numeric and textual values. This allows the same API structure to represent quantitative measurements, such as a numerical value with a unit, as well as qualitative findings recorded during field assessment.

Because each observation is linked to an encounter, the system can reconstruct the context in which it was captured. This is important for understanding whether a value was recorded during initial assessment, reassessment, transport, treatment or another operational phase.

4.3.3 Recording Procedures

A procedure represents an intervention, treatment step or operational action performed for the casualty. Procedure records can include coded procedure information, category, status, status reason, performed start and end times, body site, method, reason code, outcome, performer and notes.

This workflow allows the API to document what was done during a casualty encounter, when it was done and in which interaction context it occurred. Examples include first aid actions, decontamination-related procedures, triage interventions or other field care actions.

As with conditions and observations, procedures are recorded as discrete events within the encounter lifecycle. Because the encounter belongs to the casualty, these interventions still become part of the traceable casualty history rather than being overwritten by later updates.

4.4 Traceability Across the Casualty Tracking Workflow

The Casualty Tracking API applies the same traceability principles used in D3.1, but to casualty and casualty data rather than sample custody data. Each operational action creates or updates a domain record and is preserved through the event-based backend architecture described in Section 5.

For any given casualty, the system can reconstruct:

- incident under which the casualty was registered;
- external identifier used to connect the casualty to the digital record;
- encounters created for the casualty;
- locations and timestamps of those encounters;
- conditions recorded within each encounter;
- observations or measurements captured during each encounter;
- procedures or interventions performed during each encounter;
- casualty association for each clinical-event record through its encounter.

This creates a chronological casualty tracking history from registration through repeated encounters and recorded interventions. The operational workflow therefore supports accountability and continuity of information without relying on a Supervisor Dashboard or on the sample tracking workflows documented in D3.1.

4.5 Voice-to-text Casualty Information

The Casualty Tracking API can support a voice-to-text workflow for casualty information capture. The purpose of this capability is to allow responders to capture casualty information verbally during field operations and transform that speech into the structured casualty tracking data model used by the API.

This capability is not intended to replace the existing structured API model. Instead, it is designed as an assisted data-entry layer on top of the current model. Spoken input is first transcribed and then

interpreted by a LLM. The LLM maps the transcript to the casualty tracking data model and invokes the appropriate API operations through a controlled tool interface, such as a Model Context Protocol (MCP)[8]-style adapter that exposes selected backend endpoints as callable tools. Confirmed outputs are then committed through the same command, validation and event-sourcing mechanisms described in Section 5.

4.5.1 Operational Concept

In time-critical environments, responders may not always be able to enter structured data manually. A voice-driven workflow would allow a responder to speak observations such as casualty identifiers, symptoms, measurements or interventions while continuing to work hands-free. The system would convert the spoken information into structured draft records.

The workflow consists of six stages:

- The responder records or streams spoken casualty information.
- A speech-to-text component converts the audio to a text transcript.
- An LLM interprets the transcript and identifies the relevant operational intent and data elements.
- The LLM maps the extracted information to the casualty tracking model.
- The LLM calls the appropriate backend operations through an MCP-style tool layer or equivalent controlled API adapter.
- A user or authorised client reviews and confirms the structured records before final submission where confirmation is required, the LLM does not have access to update the record independently.

Human confirmation remains important. Voice input can be ambiguous, noisy or incomplete in field conditions, and LLM interpretation can produce uncertain or incomplete mappings. The workflow therefore treats extracted records as proposed data until they are reviewed and accepted by a user or trusted workflow component.

[8] https://modelcontextprotocol.io/docs/getting-started/intro

Spoken information	Extracted meaning	Target data model element
"Casualty wristband P-104, exposed to vapour, coughing heavily."	Casualty identifier and condition.	Casualty external identifier; Condition linked to the active Encounter.
"Respiratory rate twenty-eight, oxygen saturation ninety-two percent."	Numeric observations with units.	Observation records linked to the Encounter.
"Started decontamination at fourteen twenty."	Procedure and performed time.	Procedure record linked to the Encounter.
"Casualty reassessed, symptoms improving."	Updated condition or follow-up observation.	Condition update or Observation linked to the Encounter.
"Encounter complete."	Encounter lifecycle transition.	Encounter end command.

Table 4. Voice-to-data transformation examples

4.5.2 LLM-Based Mapping to the Casualty Tracking Model

The LLM-based mapping approach transforms speech-derived information into the existing hierarchy:

Incident -> Casualty -> Encounter -> Condition / Observation / Procedure

The incident context is expected to be selected before the voice workflow starts. The casualty is identified through an external identifier such as a wristband, QR code, NFC tag or spoken casualty code. The encounter provides the operational context for the spoken casualty information. Conditions, Observations and Procedures extracted from speech are associated with that encounter, and their casualty association is derived through the encounter.

The LLM does not write directly to the database. Instead, it selects from a controlled set of available actions, such as looking up a casualty by external identifier, starting an encounter, recording an observation, recording a condition, recording a procedure or ending an encounter. These actions correspond to backend API endpoints and are exposed to the LLM through a tool-calling layer. An MCP is a suitable pattern for this because it can present backend capabilities as typed tools with defined inputs, outputs and validation rules.

Extracted concept	Example	API target
Casualty identifier	"P-104" or "wristband 104"	findPatientByExternalIdentifier tool or casualty lookup endpoint.
Encounter action	"start assessment" or "encounter complete"	startEncounter or endEncounter tool.
Symptom or condition	"coughing", "burns", "possible exposure"	recordCondition tool for the active Encounter.

Measurement	"pulse 110", "temperature 38.5"	recordObservation tool for the active Encounter.
Intervention	"oxygen given", "decontamination started"	recordProcedure tool for the active Encounter.
Timing phrase	"at fourteen twenty", "five minutes ago"	Timestamp field on Encounter, Observation or Procedure.
Uncertainty phrase	"suspected", "possible", "unclear"	Verification status, confidence marker or note.

Table 5. Mapping from extracted speech concepts to API records

4.5.3 Technical Processing Pipeline

The processing pipeline is designed as an assistive layer rather than a separate source of truth. The source of truth remains the event-sourced API. The LLM interprets the transcript and uses a controlled MCP-style adapter, or equivalent tool-calling interface, to invoke the same casualty tracking operations that a manual client would call.

The target pipeline is:

- Audio capture by the consuming client.
- Speech-to-text transcription.
- Transcript normalisation, including timestamps, units and identifiers.
- LLM interpretation of intent, entities and clinical or operational concepts.
- Tool selection by the LLM, such as casualty lookup, encounter start, observation recording, condition recording, procedure recording or encounter closure.
- Tool execution through an MCP-style adapter that translates tool calls into validated API requests.
- Confidence scoring, validation and user review where required.
- Submission to the Casualty Tracking API through normal endpoints.
- Event persistence and read-model projection.

4.5.4 Validation and Safety Considerations

Voice-derived information and LLM-generated tool calls should not bypass normal validation. The generated records must satisfy the same Data Transfer Object (DTO)[9] validation, encounter context validation and domain rules as manually created records. If the system cannot confidently identify the casualty, encounter, record type or coded value, it should return the extracted text as an unresolved note or require manual correction before submission.

Important design considerations include:

- Noisy field environments may reduce transcription accuracy;
- LLM tool access must be limited to approved casualty tracking operations;
- Tool schemas must be explicit so that generated calls can be validated before execution;
- Identifiers such as wristband codes must be confirmed carefully;

[9] https://learn.microsoft.com/en-us/aspnet/web-api/overview/data/using-web-api-with-entity-framework/part-5

- Numeric measurements and units are reviewed before submission;
- Ambiguous symptoms should be marked as uncertain rather than over-classified;
- Generated records should remain editable before confirmation;
- The original transcript may be retained as supporting context where appropriate;
- Confirmed records should be committed through the normal event-sourced API path.

This capability improves speed and ergonomics during casualty handling while preserving the structured, traceable data model established by the Casualty Tracking API.

5 TECHNICAL IMPLEMENTATION

5.1 Synthetic Casualty Generation System Architecture and Technical Stack

A capability for generating a synthetic population with the aid of a Synthea-based synthetic population generator has been developed in terms of a modular system (component-oriented) which consists of an extension of the existing Synthea Core Runtime by means of a new custom-made EMBRACE Biotoxin Module. The standard Synthea-modules represent the base-line functionality for the generation of demographics and medical history and illness of casualties, whereas the custom-module represents the additional exposure to biotoxins and symptoms progression logic that was required for this particular use-case. At run-time, both the module-loader and the runtime-engine interpret these definitions to produce longitudinal synthetic casualty-sequences representing the evolution over time of intoxicated individuals. This produced output will be transferred via the export-layer to transform them into structural synthetic records and to aggregate them into a synthetic-cohort-dataset. With this modular structure it is possible for EMBRACE to develop upon a well-established synthetic health-data-framework and introduce modelling specificities adapted to biotoxin related incident-research.

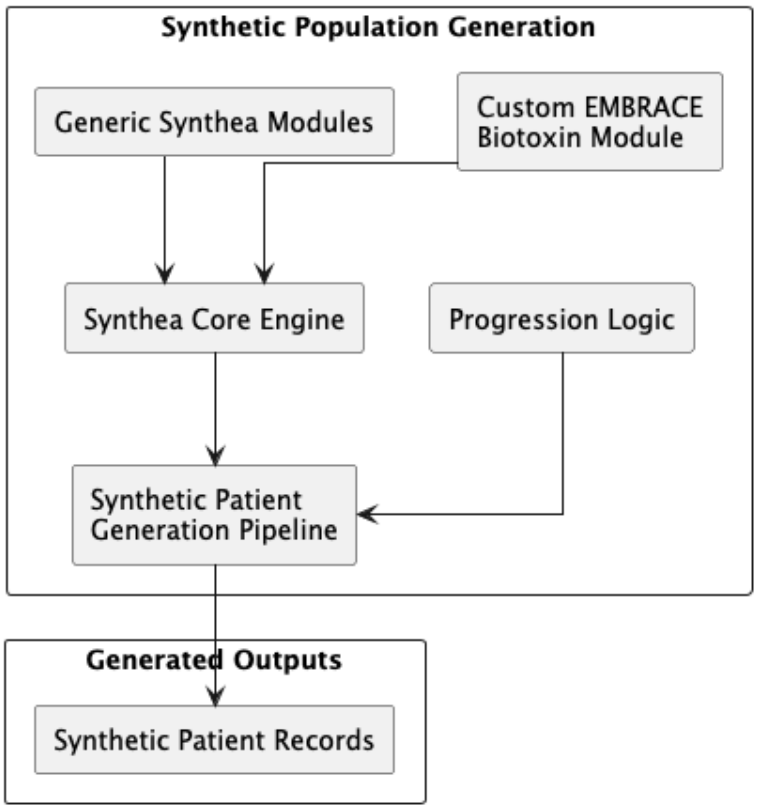


Figure 6. Synthetic population generation flow diagram

5.2 Casualty Tracking System Technical Stack

This section describes the technical implementation of the Casualty Tracking API extension. The API extends the backend architecture delivered in EMBRACE's D3.1 Sampling devices and sample tracking

– 1st Iteration, by adding casualty-oriented registration, encounter tracking and clinical event recording capabilities to the same modular NestJS[10], CQRS and event-sourced foundation. The sample tracking, collection, custody transfer and analysis workflows described in D3.1 are outside the scope of this section and are therefore not repeated here.

Where the implementation reuses architectural mechanisms already documented in D3.1, this section refers back to the D3.1 technical implementation rather than restating the same material. This applies in particular to the NestJS backend structure, Swagger/OpenAPI[11] generation, TypeORM[12] persistence, the split between event store and read database, and the general CQRS/event sourcing approach.

5.2.1 System Architecture and Technology Stack

The Casualty Tracking API extension is implemented as part of the same backend service described in D3.1. It is written in TypeScript and built with NestJS version 11, running on Node.js[13]. The API uses the NestJS module system to isolate domain areas into separate bounded contexts and uses @nestjs/cqrs to separate command handling, query handling and domain event publication.

No separate Supervisor Dashboard is introduced for this deliverable. The implemented scope is the backend API extension and the domain modules required to support casualty tracking. Client applications or operational user interfaces can consume the API through its OpenAPI contract[14], but the design and implementation of such clients are not part of this section.

The extension adds the following backend modules to the existing platform: Incidents, Casualties, Encounters, Conditions, Observations and Procedures. Incidents provide the top-level operational context. Casualties represent persons registered under an incident. Encounters record time-bounded interactions with a casualty. Conditions, Observations and Procedures record clinical or field medical information associated with a casualty and, where applicable, a specific encounter.

The system continues to use two PostgreSQL[15] databases, as described in D3.1. The event store persists immutable domain events and remains the authoritative source of state changes. The read database stores projected relational entities that support operational API responses and lookup use cases. Both database connections are configured through TypeORM and environment-based configuration.

[10] https://nextjs.org/docs
[11] https://swagger.io/resources/open-api/
[12] https://typeorm.io/docs/getting-started
[13] https://nodejs.org/en/about
[14] An OpenAPI contract: a machine-readable specification that defines an API's endpoints, inputs, outputs, data schemas, authentication rules, and error responses
[15] https://www.postgresql.org/

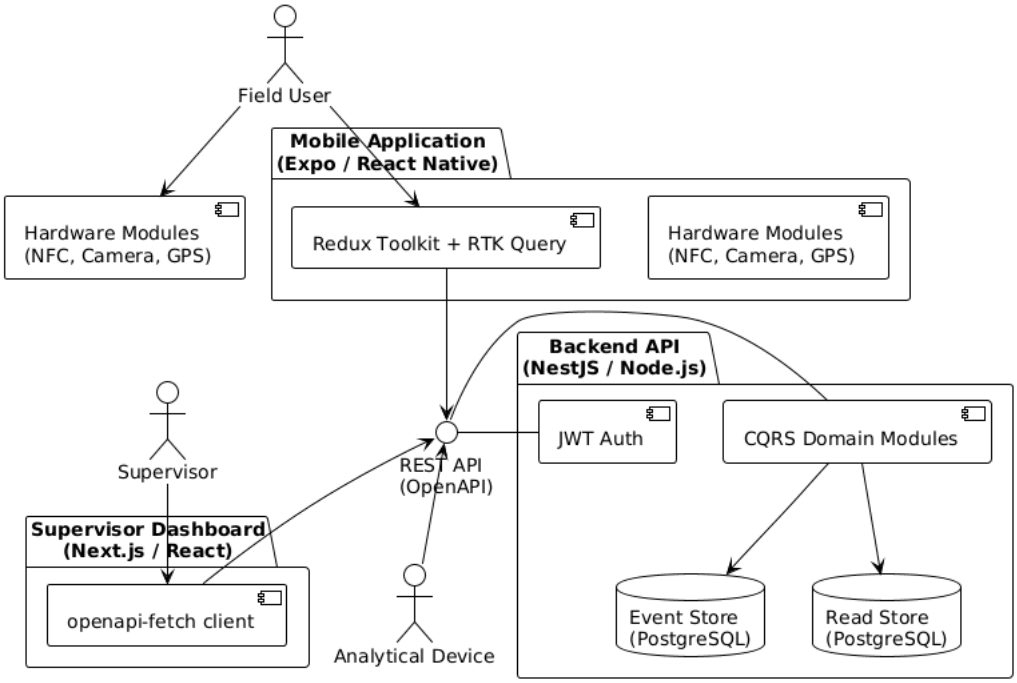


Figure 7. Component diagram of Casualty Tracking App

5.2.2 Data Architecture and Core Entities

The Casualty Tracking API extension represents its operational domain through casualty-centred entities. These entities complement the sample tracking entities described in D3.1 but do not replace or restate them. The extension reuses the incident as the root operational context and adds casualty, encounter, condition, observation and procedure records beneath that context.

The Incident entity is reused as the top-level organisational unit. In the casualty tracking extension, an incident represents the event, location or operational situation under which casualties are registered. Each incident records a name and geographic coordinates. Casualty creation is only permitted for incidents that remain open, using the same policy-oriented approach that prevents operations against concluded incident contexts.

The Casualty entity represents an individual person being tracked within an incident. A casualty may include a given name, family name, sex at birth and birth date, although these demographic fields are optional to support field conditions in which a full identity may not be available. Each casualty must include an external identifier. This identifier can represent an NFC tag, QR code, wristband identifier or other operational identifier used to bind the physical person to the digital record.

The Encounter entity represents an interaction with a casualty. An encounter is linked to a casualty and records the geographic coordinates and timestamp at which the encounter started. It can subsequently be ended, at which point an end timestamp is recorded. This allows the API to reconstruct when and where a casualty was seen or assessed, and how long the interaction remained open.

The Condition entity represents a clinical or field-assessed condition associated with a casualty. Each condition is linked to a casualty and may also be linked to the encounter during which it was identified or recorded. The entity supports structured clinical metadata including identifier, code, category,

clinical status, verification status, severity, body site, onset time, abatement time, diagnosis time, recorded time and free-text notes.

The Observation entity represents a measurement, finding or assessed value relating to a casualty. Observations are linked to a casualty and may optionally be linked to an encounter. Each observation records a code and category, lifecycle status, effective and issued timestamps, value type, textual or numeric value, unit, interpretation, body site, method and notes. This supports vital signs, exposure assessments, triage observations and other field measurements without hard-coding each measurement type into the API model.

The Procedure entity represents an intervention, procedure or action performed for a casualty. A procedure is linked to a casualty and may also be linked to a specific encounter. The entity records code, category, status, status reason, performed start and end times, body site, method, reason code, outcome, performer and notes.

The relationships between the entities follow an incident-to-casualty-to-encounter structure. An incident may have many casualties. A casualty may have many encounters. Conditions, Observations and Procedures refer to the casualty as their subject and may also refer to an encounter as their operational context. This allows the system to distinguish long-lived casualty-level information from data captured during a specific field interaction.

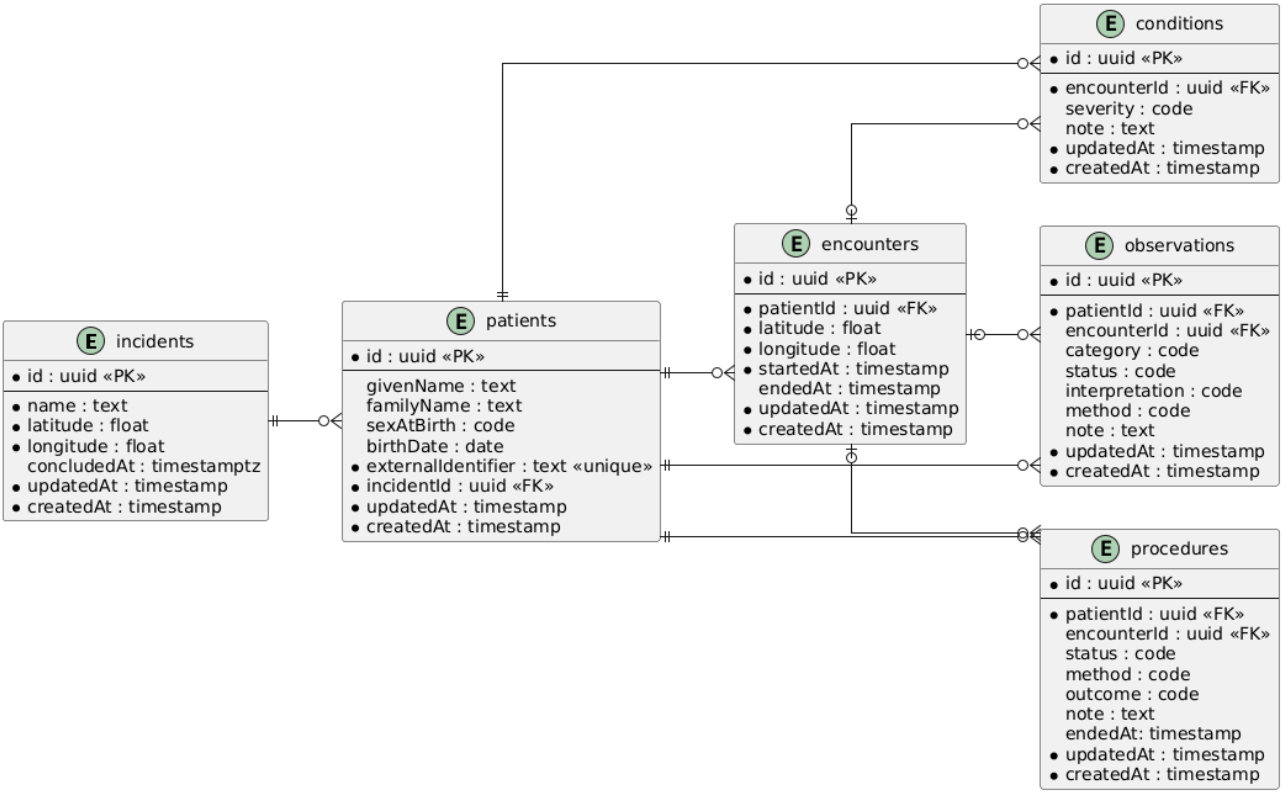


Figure 8. Casualty Tracking API entity relationship diagram

5.2.3 API Design and Integration Surface

The API follows the same REST and OpenAPI conventions described in D3.1. The extension exposes casualty-centred functionality through resource-oriented HTTP endpoints, with JSON request and

response bodies validated by class-validator and class-transformer through the global NestJS validation pipe.

The casualty tracking API surface includes POST /casualties for creating casualty records, POST /encounters for starting encounters, and POST /encounters/{encounterId}/end for ending encounters. It also includes POST, GET and PATCH operations for Conditions, Observations and Procedures, allowing clinical-event records to be created, retrieved and updated in relation to the relevant casualty and encounter context. Incident endpoints provide the operational context reused by the extension.

The API is versioned using URI-based versioning with version 1 as the default API version. The Swagger/OpenAPI documentation is generated at runtime using @nestjs/swagger and is exposed through the same documentation mechanism described in D3.1. The API title is configured as "Sample & Casualty Tracking API", reflecting that the service contains both the original D3.1 capabilities and the casualty tracking extension, even though sample tracking workflows are not in scope for this deliverable section.

Input validation is applied at the API boundary. Incident and encounter locations are validated as latitude and longitude values. Casualty records require an external identifier and incident identifier. Encounter creation requires a valid casualty identifier. Clinical-event records require the casualty reference and structured coded fields appropriate to their type. Unknown request body properties are rejected because the global validation pipe is configured with whitelisting and non-whitelisted property rejection.

No separate external analytical device workflow is introduced by the casualty tracking extension. Integration is achieved through the generic REST/OpenAPI interface. Any mobile application, clinical application, command-and-control system or integration adapter can consume the published OpenAPI specification and communicate with the API using standard HTTP and JSON conventions.

5.2.4 Security and Access Control

The casualty tracking extension does not introduce a new dashboard-specific permission model or a new role taxonomy. Security controls that are unchanged from the D3.1 platform should be treated as inherited platform concerns and referenced directly from D3.1.

For the extension-specific functionality, the primary domain-level control is incident-state enforcement. Casualty creation is guarded by an incident-open policy, which rehydrates the incident aggregate from the event store and prevents creation of a casualty under an incident that has already been concluded. This ensures that casualty registration remains bound to an active operational context. The same principle applies to encounter, condition, observation and procedure workflows that depend on the incident context inherited through the casualty record.

The extension also preserves integrity through the same event-sourced write model described in D3.1. State-changing operations are expressed as commands, validated by command handlers and domain aggregates, and then committed as immutable events. Read models are projections of the event stream. This means that the operational read database can be rebuilt from the event history and that the event store remains the authoritative record of casualty, encounter, condition, observation and procedure lifecycle changes.

When this API is deployed in an environment that includes the authentication and role-based authorisation components described in D3.1, the same principle applies to the new casualty tracking endpoints: authorisation decisions are enforced at the backend boundary rather than delegated to client applications. The extension does not require a Supervisor Dashboard to make those controls effective.

5.2.5 Event Logging, Traceability and Lifecycle Reconstruction

The casualty tracking extension uses the same event sourcing and CQRS implementation as the D3.1 system. The details of the event store, event serialisation, event deserialisation, event class registry, aggregate rehydration and projection flow are unchanged and should be referenced from D3.1.

The extension adds casualty, encounter, condition, observation and procedure lifecycle events to that shared mechanism. When a casualty is created, the API creates a Casualty aggregate, emits a casualty-created event, persists the event to the event store and projects the casualty into the read database. When an encounter is started, the API creates an Encounter aggregate, emits an encounter-started event, persists the event and creates the encounter read model. When an encounter is ended, the aggregate is rehydrated from its event stream, an encounter-ended event is emitted, and the read model is updated with the encounter end timestamp.

Condition, Observation and Procedure aggregates follow the same flow. Commands create or update the relevant aggregate, the aggregate validates its state transition, the event store persists the resulting event and event handlers project the change into the operational read database. This gives all casualty-related records the same traceability and replay characteristics as the original D3.1 evidence-related records.

The encounter lifecycle follows a clear event-driven sequence: a field user or consuming application starts an encounter for an existing casualty; the backend receives a start-encounter command; the Encounter aggregate emits an encounter-started event; the event is persisted; and the encounter read model is created. When the interaction ends, the backend receives an end-encounter command, rehydrates the Encounter aggregate, emits an encounter-ended event, persists it and updates the read model.

Because each casualty-related lifecycle transition is represented as an event, the system can reconstruct historical state from the event stream. For a casualty, the system can determine the incident under which the casualty was registered, the operational identifier used to bind the casualty to the digital record and the encounters created for that casualty. For conditions, observations and procedures, the system can reconstruct what was recorded, when it was recorded, the casualty to whom it applied and the encounter context in which it was captured.

This preserves the core traceability principle established in D3.1 while applying it to casualty and casualty tracking rather than sample custody tracking. The result is a backend extension that shares the same evidentiary architecture but applies it to operational casualty records and time-bounded casualty interactions.

6 KNOWN LIMITATIONS

6.1 Biotoxin Intoxicated Synthetic Population Generator

The current Biotoxin Intoxicated Synthetic Population Generator is limited to a ricin intoxication scenario. This is due to the limited availability of structured and reliable data on biotoxin dose, exposure route, symptom onset, symptom progression, severity and clinical timeline.

As a result, the generator can support ricin-focused alerting concepts, but it should not be interpreted as a general model for all biotoxin incidents. It does not support comparison between different biotoxins, alternative exposure routes or toxin-specific progression patterns.

The current ricin module also simplifies real-world intoxication. It includes selected symptoms, observations, locations and care-chain events, but does not fully model complications, treatment effects, comorbidities, diagnostic uncertainty or operational constraints such as triage capacity, transport delays and decontamination queues.

6.2 Casualty Tracking Application

The Casualty Tracking Application represents an implementation focused on the core operational requirements for intoxicated person registration and tracking. Several limitations remain at this stage.

The current implementation focuses on a selected subset of FHIR-based modules. The included modules provide the core structures required for casualty tracking, but they do not represent a complete FHIR implementation or a full electronic health record.

The mobile application is standalone and separate from the Sample Tracking mobile application. This separation supports operational clarity, but there is currently no direct application-level workflow linking a casualty record to sample collection activities.

Access to the casualty tracking modules is restricted to the practitioner role. This provides a clear and controlled permission model, but future iterations may require more granular role distinctions if additional care-chain actors are introduced.

The present implementation reuses the existing backend foundation. This provides consistency and avoids duplication, but careful module separation and API governance remain important as the system evolves.

7 NEXT STEPS

7.1 Casualty Tracking Application

Further development of the Casualty Tracking Application will focus on extending the system from practitioner-centred casualty registration and encounter documentation towards broader operational coordination and field validation. The current implementation provides the core workflow for registering casualties, linking them to incidents, starting and ending encounters, and recording conditions, observations and procedures through the backend API. These capabilities establish the technical foundation for intoxicated person tracking across the care chain.

A key next development step is the implementation of an improved commander overview. While the practitioner role is focused on creating and maintaining casualty-related records, the commander role should support operational situational awareness during an incident. This overview should allow authorised commander users to monitor the incident scene by showing the number of registered casualties, their latest known locations, active and closed encounters, recorded observations or conditions, and relevant status changes over time. The purpose of this functionality is not to replace practitioner documentation, but to provide a consolidated view of casualty distribution, casualty flow and operational progression during a biotoxin incident.

The FTX provides an opportunity to evaluate whether the implemented workflow is operationally usable under realistic incident-response conditions. In particular, the exercise should assess whether practitioners can register casualties quickly, retrieve casualty records using external identifiers, manage encounters correctly, and record observations, conditions and procedures without disrupting field operations.

8 CONCLUSION

D5.1 has resulted in a working foundation for the registration and tracking of intoxicated persons within the EMBRACE digital response framework. The main outcome of this deliverable is the development of a Casualty Tracking System that enables responders to create casualty records, associate them with an incident, link them to external identifiers such as wristbands, QR codes or NFC tags, and document their progression through repeated practitioner encounters. The system therefore moves intoxicated person tracking from an informal or paper-based process towards a structured, digital and traceable workflow.

The deliverable shows that the casualty record can function as the central anchor for all information related to an intoxicated person. Encounters provide the time- and context-specific record of practitioner contact, while observations, conditions and procedures capture the clinical and operational information recorded during those encounters. As a result, the system can support both immediate field use and later reconstruction of what happened to a casualty during the response operation, including when they were seen, where they were located, which findings were recorded and which interventions were performed.

A second important outcome is the reuse and extension of the existing EMBRACE backend architecture. By adding casualty-specific modules to the same event-sourced, CQRS-based and domain-driven platform used for tracking workflows elsewhere in the project, the implementation avoids creating a separate isolated system. Instead, casualty tracking becomes part of a broader digital architecture in which state changes are validated, stored as auditable events and projected into operational read models. This gives the system a traceable technical basis and makes it suitable for further extension in later project phases.

D5.1 has also produced supporting components that contribute to the longer-term development of the EMBRACE casualty tracking capability. The ricin-focused Synthea module demonstrates how synthetic casualty populations can be generated for biotoxin scenarios where real-world data is scarce, sensitive or unavailable. This provides a controlled basis for testing, validation and future alerting concepts. The speech-to-text and LLM-based parsing work demonstrates how spoken responder input could be transformed into structured casualty data, reducing the burden of manual data entry in field conditions. These components are not yet final operational capabilities, but they establish useful research directions and technical building blocks for future development.

The outcome of the work is therefore not only a set of technical modules, but a coherent casualty tracking workflow. The system supports registration, identification, encounter management, structured recording and traceability across the casualty care chain. It also identifies the boundaries of the current implementation, including the ricin-specific scope of the synthetic population generator, the selected subset of FHIR-based casualty modules, the standalone nature of the mobile application and the need for further operational validation.

Overall, D5.1 delivers the first complete iteration of the EMBRACE intoxicated person registration and tracking capability. It establishes how casualties can be digitally registered, followed through encounters and linked to structured clinical and operational records during a biotoxin incident. The next stage is to evaluate this workflow in realistic field-trial conditions, strengthen commander-level situational awareness and expand the system where needed based on practitioner and operational

feedback. This will determine how the developed capability can best support responders and commanders in maintaining reliable, timely and traceable casualty information during complex CBRN response operations.

9 REFERENCES

Challoner, K. R., & McCarron, M. M. (1990). Castor bean intoxication. *Annals of Emergency Medicine*, *19*(10), 1177-1183.

EMBRACE Consortium. 2026. D3.1 Sampling Devices and Sample Tracking – 1st Iteration. Horizon Europe Project EMBRACE.

Ge, X., Murtaza, S., Cortez, A., and Alemzadeh, H. EMSDialog: Synthetic Multi-person Emergency Medical Service Dialogue Generation from Electronic Casualty Care Reports via Multi-LLM Agents. University of Virginia.

RESPOND-A Consortium. 2022. D3.4 Triage System and IT Platform for Emergency Care Chain. Horizon 2020 Project RESPOND-A.

TOXI-Triage Consortium. 2019. D5.3 Casualty Tracing Tag & Trace Demonstrator. Horizon 2020 Project TOXI-Triage.

Walonoski, J., Kramer, M., Nichols, J., Quina, A., Moesel, C., Hall, D., ... & McLachlan, S. (2018). Synthea: An approach, method, and software mechanism for generating synthetic casualties and the synthetic electronic health care record. *Journal of the American Medical Informatics Association*, *25*(3), 230-238.

ANNEXES

Annex A. Intoxicated Population Generation Module Output

Embrace Main Module


```
{
 "name": "Embrace Complete",
 "gmfVersion": "1.0",
 "specialty": "general",
 "remarks": [],
 "states": {
  "Initial": {
   "type": "Initial",
   "direct_transition": "Call_Train_Location",
   "name": "Initial"
  },
  "Call_Train_Location": {
   "type": "CallSubmodule",
   "submodule": "embrace/location_train",
   "direct_transition": "Casualty_Ratio"
  },
  "Casualty_Ratio": {
   "type": "Simple",
   "distributed_transition": [
    {
     "distribution": 0.5,
     "transition": "Call_Hospital_Location"
    },
    {
     "distribution": 0.5,
     "transition": "Terminal"
    }
   ]
  },
  "Call_Hospital_Location": {
   "type": "CallSubmodule",
   "submodule": "embrace/location_hospital",
   "direct_transition": "Symptoms"
  },
  "Symptoms": {
   "type": "Simple",
   "distributed_transition": [
    {
     "distribution": 0.5,
     "transition": "Call_Ricin_Submodule_Default"
    },
    {
     "distribution": 0.5,
     "transition": "Call_Ricin_Submodule_One_Symptom"
    }
   ]
  },
  "Call_Ricin_Submodule_Default": {
   "type": "CallSubmodule",
   "submodule": "embrace/ricin/default",
```

```
    "direct_transition": "Terminal"
   },
   "Call_Ricin_Submodule_One_Symptom": {
    "type": "CallSubmodule",
    "submodule": "embrace/ricin/one_symptom",
    "direct_transition": "Terminal"
   },
   "Terminal": {
    "type": "Terminal"
   }
  }
}
```

Embrace Ricin Module

```
{
 "name": "Default Ricin Case",
 "remarks": [],
 "states": {
  "Initial": {
   "type": "Initial",
   "direct_transition": "Sample_Symptoms"
  },
  "Sample_Symptoms": {
   "type": "MultiSymptom",
   "symptoms": [
    {
     "name": "Abdominal pain",
     "probability": 0.75
    },
    {
     "name": "Vomiting",
     "probability": 0.8
    },
    {
     "name": "Fatigue",
     "probability": 0.7
    },
    {
     "name": "Diarrhea",
     "probability": 0.8
    }
   ],
   "direct_transition": "Check Abdominal Pain"
  },
  "Check Abdominal Pain": {
   "type": "Simple",
   "conditional_transition": [
    {
     "condition": {
      "condition_type": "Symptom",
      "symptom": "Abdominal pain",
      "operator": ">",
      "value": 0
     },
     "transition": "Abdominal_Pain_Observation"
```

```
    },
    {
     "transition": "Check Vomiting"
    }
   ]
  },
  "Abdominal_Pain_Observation": {
   "type": "Observation",
   "codes": [
    {
     "system": "LOINC",
     "code": "44833-2",
     "display": "Abdominal pain severity",
     "value_set": ""
    }
   ],
   "value": 7,
   "unit": "scale",
   "direct_transition": "Check Vomiting"
  },
  "Check Vomiting": {
   "type": "Simple",
   "conditional_transition": [
    {
     "condition": {
      "condition_type": "Symptom",
      "symptom": "Vomiting",
      "operator": ">",
      "value": 0
     },
     "transition": "Vomiting_Observation"
    },
    {
     "transition": "Check Fatigue"
    }
   ]
  },
  "Vomiting_Observation": {
   "type": "Observation",
   "codes": [
    {
     "system": "LOINC",
     "code": "28540-3",
     "display": "Vomiting [Presence]",
     "value_set": ""
    }
   ],
   "value": "Present",
   "unit": "scale",
   "direct_transition": "Check Fatigue"
  },
  "Check Fatigue": {
   "type": "Simple",
   "conditional_transition": [
    {
     "condition": {
      "condition_type": "Symptom",
```

```
        "symptom": "Fatigue",
        "operator": ">",
        "value": 0
       },
      "transition": "Fatigue_Observation"
    },
    {
     "transition": "Check Diarrhea"
    }
  ]
},
"Fatigue_Observation": {
  "type": "Observation",
  "codes": [
    {
     "system": "LOINC",
     "code": "41977-7",
     "display": "Fatigue severity [Reported]",
     "value_set": ""
    }
  ],
  "value": 7,
  "unit": "scale",
  "direct_transition": "Check Diarrhea"
},
"Check Diarrhea": {
  "type": "Simple",
  "conditional_transition": [
    {
     "condition": {
       "condition_type": "Symptom",
       "symptom": "Diarrhea",
       "operator": ">",
       "value": 0
      },
     "transition": "Diarrhea_Delay"
    },
    {
     "transition": "Presentation_Delay"
    }
  ]
},
"Diarrhea_Delay": {
  "type": "Delay",
  "range": {
    "low": 0.5,
    "high": 8,
    "unit": "hours"
  },
  "direct_transition": "Diarrhea_Observation"
},
"Diarrhea_Observation": {
  "type": "Observation",
  "codes": [
    {
     "system": "LOINC",
     "code": "29746-5",
```

```
        "display": "Diarrhea [Presence]",
        "value_set": ""
       }
      ],
      "value": "Present",
      "unit": "score",
      "direct_transition": "Presentation_Delay"
     },
     "Presentation_Delay": {
      "type": "Delay",
      "range": {
       "low": 4,
       "high": 20,
       "unit": "hours"
      },
      "direct_transition": "ED_Encounter"
     },
     "ED_Encounter": {
      "type": "Encounter",
      "encounter_class": "emergency",
      "codes": [
       {
        "system": "SNOMED-CT",
        "code": "50849002",
        "display": "Emergency room admission"
       }
      ],
      "direct_transition": "End_Encounter"
     },
     "End_Encounter": {
      "type": "EncounterEnd",
      "direct_transition": "Terminal"
     },
     "Terminal": {
      "type": "Terminal"
     }
    }
}
```


Embrace Single Symptom


```
{
 "name": "Embrace Ricin Poisoning (One Symptom)",
 "remarks": [""],
 "states": {
  "Initial": {
   "type": "Initial",
   "distributed_transition": [
    { "transition": "Abdominal_Pain_Symptom", "distribution": 0.25 },
    { "transition": "Vomiting_Symptom", "distribution": 0.25 },
    { "transition": "Fatigue_Symptom", "distribution": 0.25 },
    { "transition": "Diarrhea_Symptom", "distribution": 0.25 }
   ]
  },
  "Abdominal_Pain_Symptom": {
   "type": "Symptom",
   "symptom": "abdominal pain",
   "severity": 70,
```

```
  "direct_transition": "Abdominal_Pain_Observation"
 },
 "Abdominal_Pain_Observation": {
  "type": "Observation",
  "codes": [
   {
    "system": "LOINC",
    "code": "44833-2",
    "display": "Abdominal pain severity",
    "value_set": ""
   }
  ],
  "value": 7,
  "unit": "scale",
  "direct_transition": "Terminal"
 },
 "Vomiting_Symptom": {
  "type": "Symptom",
  "symptom": "vomiting",
  "severity": 70,
  "direct_transition": "Vomiting_Observation"
 },
 "Vomiting_Observation": {
  "type": "Observation",
  "codes": [
   {
    "system": "LOINC",
    "code": "28540-3",
    "display": "Vomiting [Presence]",
    "value_set": ""
   }
  ],
  "value": "Present",
  "unit": "scale",
  "direct_transition": "Terminal"
 },
 "Fatigue_Symptom": {
  "type": "Symptom",
  "symptom": "fatigue",
  "severity": 70,
  "direct_transition": "Fatigue_Observation"
 },
 "Fatigue_Observation": {
  "type": "Observation",
  "codes": [
   {
    "system": "LOINC",
    "code": "41977-7",
    "display": "Fatigue severity [Reported]",
    "value_set": ""
   }
  ],
  "value": 7,
  "unit": "scale",
  "direct_transition": "Terminal"
 },
 "Diarrhea_Symptom": {
```

```
    "type": "Symptom",
    "symptom": "diarrhea",
    "severity": 70,
    "direct_transition": "Diarrhea_Observation"
   },
   "Diarrhea_Observation": {
    "type": "Observation",
    "codes": [
     {
      "system": "LOINC",
      "code": "29746-5",
      "display": "Diarrhea [Presence]",
      "value_set": ""
     }
    ],
    "value": "Present",
    "unit": "score",
    "direct_transition": "Terminal"
   },
   "Terminal": {
    "type": "Terminal"
   }
  }
}
```


Location Hospital


```
{
 "name": "Hospital Embrace",
 "gmfVersion": "1.0",
 "specialty": "general",
 "remarks": [],
 "states": {
  "Initial": {
   "type": "Initial",
   "distributed_transition": [
    { "transition": "Visit_location_AKH_Wien",    "distribution": 0.14285714285714285 },
    { "transition": "Visit_location_Donaustadt",  "distribution": 0.14285714285714285 },
    { "transition": "Visit_location_Favoriten",   "distribution": 0.14285714285714285 },
    { "transition": "Visit_location_Floridsdorf", "distribution": 0.14285714285714285 },
    { "transition": "Visit_location_Hietzing",    "distribution": 0.14285714285714285 },
    { "transition": "Visit_location_Landstrabe",  "distribution": 0.14285714285714285 },
    { "transition": "Visit_location_Ottakring",   "distribution": 0.14285714285714285 }
   ]
  },
  "Visit_location_AKH_Wien": {
   "type": "PointOfInterest",
   "contaminated": "true",
   "config": {
    "type": "input",
    "name": "Universitätsklinikum AKH Wien",
    "label": "Universitätsklinikum AKH Wien",
    "physicalType": "hospital",
    "start_time": { "value": 9, "unit": "hours" },
    "duration":  { "min": 1, "max": 8, "unit": "minutes" }
   },
   "direct_transition": "Terminal"
  },
```

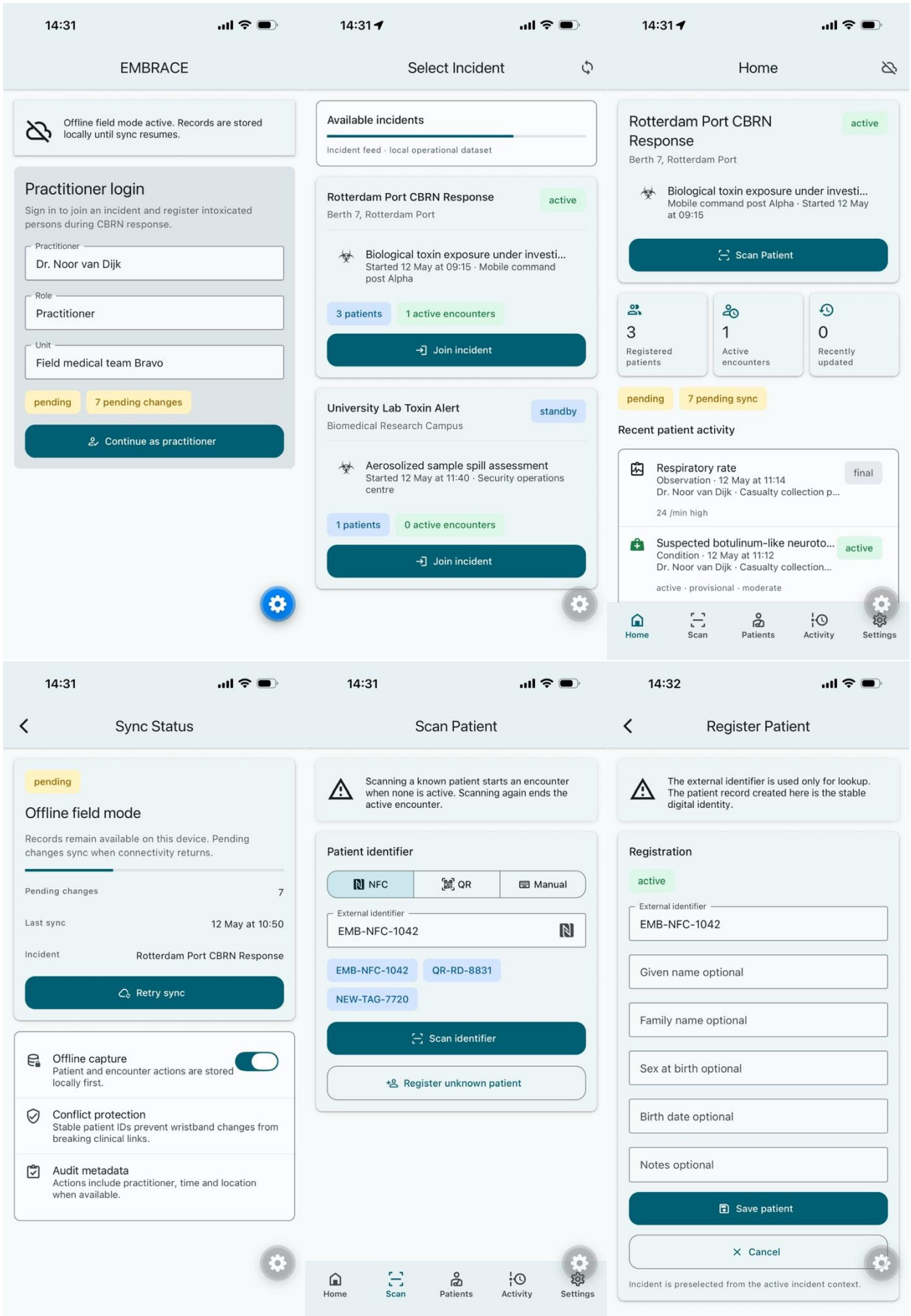
```
"Visit_location_Donaustadt": {
 "type": "PointOfInterest",
 "contaminated": "true",
 "config": {
  "type": "input",
  "name": "Klinik Donaustadt",
  "label": "Klinik Donaustadt",
  "physicalType": "hospital",
  "start_time": { "value": 9, "unit": "hours" },
  "duration":  { "min": 1, "max": 8, "unit": "minutes" }
 },
 "direct_transition": "Terminal"
},
"Visit_location_Favoriten": {
 "type": "PointOfInterest",
 "contaminated": "true",
 "config": {
  "type": "input",
  "name": "Klinik Favoriten",
  "label": "Klinik Favoriten",
  "physicalType": "hospital",
  "start_time": { "value": 9, "unit": "hours" },
  "duration":  { "min": 1, "max": 8, "unit": "minutes" }
 },
 "direct_transition": "Terminal"
},
"Visit_location_Floridsdorf": {
 "type": "PointOfInterest",
 "contaminated": "true",
 "config": {
  "type": "input",
  "name": "Klinik Floridsdorf",
  "label": "Klinik Floridsdorf",
  "physicalType": "hospital",
  "start_time": { "value": 9, "unit": "hours" },
  "duration":  { "min": 1, "max": 8, "unit": "minutes" }
 },
 "direct_transition": "Terminal"
},
"Visit_location_Hietzing": {
 "type": "PointOfInterest",
 "contaminated": "true",
 "config": {
  "type": "input",
  "name": "Klinik Hietzing",
  "label": "Klinik Hietzing",
  "physicalType": "hospital",
  "start_time": { "value": 9, "unit": "hours" },
  "duration":  { "min": 1, "max": 8, "unit": "minutes" }
 },
 "direct_transition": "Terminal"
},
"Visit_location_Landstrabe": {
 "type": "PointOfInterest",
 "contaminated": "true",
 "config": {
  "type": "input",
```

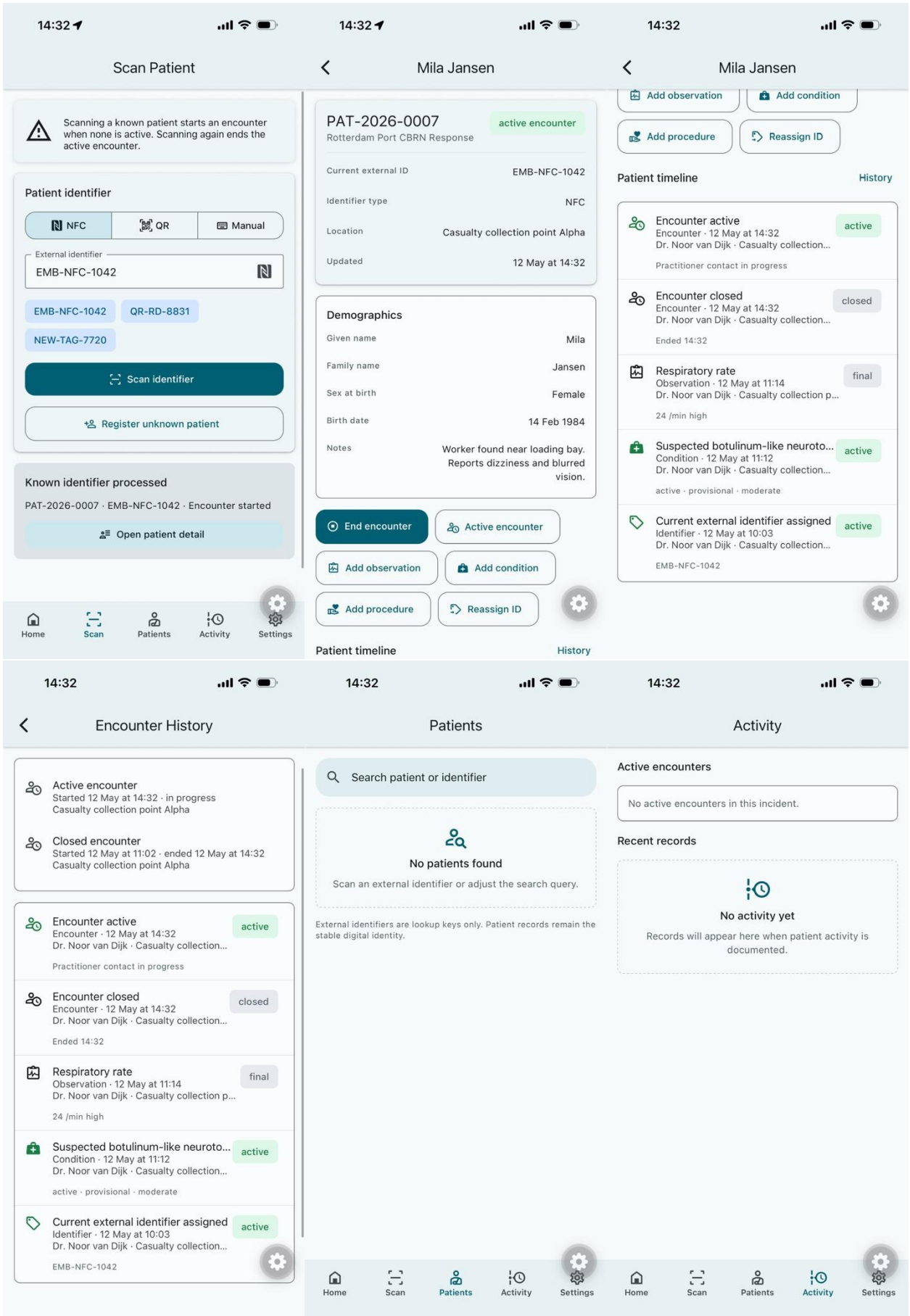
```
      "name": "Klinik Landstraße",
      "label": "Klinik Landstraße",
      "physicalType": "hospital",
      "start_time": { "value": 9, "unit": "hours" },
      "duration":   { "min": 1, "max": 8, "unit": "minutes" }
     },
     "direct_transition": "Terminal"
    },
    "Visit_location_Ottakring": {
     "type": "PointOfInterest",
     "contaminated": "true",
     "config": {
      "type": "input",
      "name": "Klinik Ottakring",
      "label": "Klinik Ottakring",
      "physicalType": "hospital",
      "start_time": { "value": 9, "unit": "hours" },
      "duration":   { "min": 1, "max": 8, "unit": "minutes" }
     },
     "direct_transition": "Terminal"
    },
    "Terminal": {
     "type": "Terminal"
    }
   }
}
```

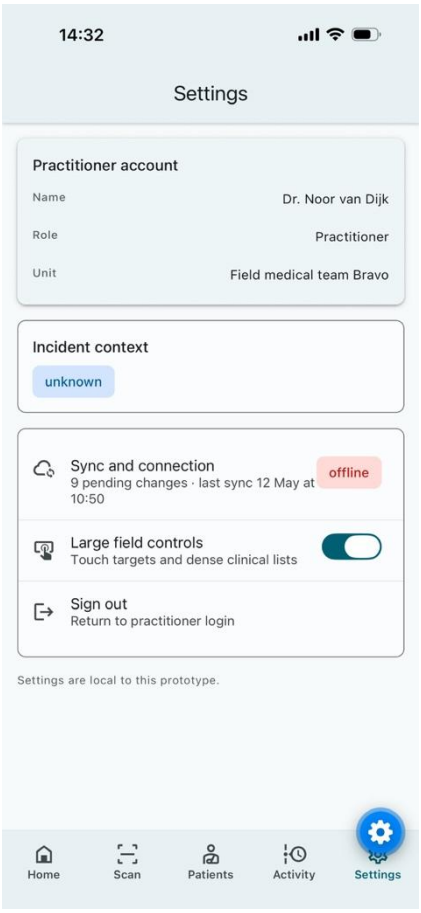

Location Train


```
{
  "name": "Hospital Embrace",
  "gmfVersion": "1.0",
  "specialty": "general",
  "remarks": [],
  "states": {
    "Initial": {
      "type": "Initial",
      "direct_transition": "Visit_train_station"
    },
    "Visit_train_station": {
      "type": "PointOfInterest",
      "contaminated": "true",
      "config": {
        "type": "input",
        "name": "Vienna hbf, Train",
        "label": "Vienna hbf, Train",
        "physicalType": "train",
        "start_time": { "value": 9, "unit": "hours" },
        "duration":   { "min": 1, "max": 8, "unit": "minutes" }
      },
      "direct_transition": "Terminal"
    },
    "Terminal": {
      "type": "Terminal"
    }
  }
}
```

Annex B. Casualty Tracking Application User Interface Screenshots


14:31
EMBRACE
Offline field mode active. Records are stored locally until sync resumes.
Practitioner login
Sign in to join an incident and register intoxicated persons during CBRN response.
Practitioner
Dr. Noor van Dijk
Role
Practitioner
Unit
Field medical team Bravo
pending
7 pending changes
Continue as practitioner
14:31
Select Incident
Available incidents
Incident feed · local operational dataset
Rotterdam Port CBRN Response
active
Berth 7, Rotterdam Port
Biological toxin exposure under investi...
Started 12 May at 09:15 · Mobile command post Alpha
3 patients
1 active encounters
Join incident
University Lab Toxin Alert
standby
Biomedical Research Campus
Aerosolized sample spill assessment
Started 12 May at 11:40 · Security operations centre
1 patients
0 active encounters
Join incident
14:31
Home
Rotterdam Port CBRN Response
active
Berth 7, Rotterdam Port
Biological toxin exposure under investi...
Mobile command post Alpha · Started 12 May at 09:15
Scan Patient
3
Registered patients
1
Active encounters
0
Recently updated
pending
7 pending sync
Recent patient activity
Respiratory rate
final
Observation · 12 May at 11:14
Dr. Noor van Dijk · Casualty collection p...
24 /min high
Suspected botulinum-like neuroto...
active
Condition · 12 May at 11:12
Dr. Noor van Dijk · Casualty collection...
active · provisional · moderate
Home
Scan
Patients
Activity
Settings
14:31
Sync Status
pending
Offline field mode
Records remain available on this device. Pending changes sync when connectivity returns.
Pending changes
7
Last sync
12 May at 10:50
Incident
Rotterdam Port CBRN Response
Retry sync
Offline capture
Patient and encounter actions are stored locally first.
Conflict protection
Stable patient IDs prevent wristband changes from breaking clinical links.
Audit metadata
Actions include practitioner, time and location when available.
14:31
Scan Patient
Scanning a known patient starts an encounter when none is active. Scanning again ends the active encounter.
Patient identifier
NFC
QR
Manual
External identifier
EMB-NFC-1042
EMB-NFC-1042
QR-RD-8831
NEW-TAG-7720
Scan identifier
Register unknown patient
Home
Scan
Patients
Activity
Settings
14:32
Register Patient
The external identifier is used only for lookup. The patient record created here is the stable digital identity.
Registration
active
External identifier
EMB-NFC-1042
Given name optional
Family name optional
Sex at birth optional
Birth date optional
Notes optional
Save patient
Cancel
Incident is preselected from the active incident context.


14:32
Scan Patient
Scanning a known patient starts an encounter when none is active. Scanning again ends the active encounter.
Patient identifier
NFC
QR
Manual
External identifier
EMB-NFC-1042
EMB-NFC-1042
QR-RD-8831
NEW-TAG-7720
Scan identifier
Register unknown patient
Known identifier processed
PAT-2026-0007 · EMB-NFC-1042 · Encounter started
Open patient detail
Home
Scan
Patients
Activity
Settings
14:32
Mila Jansen
PAT-2026-0007
Rotterdam Port CBRN Response
active encounter
Current external ID
EMB-NFC-1042
Identifier type
NFC
Location
Casualty collection point Alpha
Updated
12 May at 14:32
Demographics
Given name
Mila
Family name
Jansen
Sex at birth
Female
Birth date
14 Feb 1984
Notes
Worker found near loading bay. Reports dizziness and blurred vision.
End encounter
Active encounter
Add observation
Add condition
Add procedure
Reassign ID
Patient timeline
History
14:32
Mila Jansen
Add observation
Add condition
Add procedure
Reassign ID
Patient timeline
History
Encounter active
Encounter · 12 May at 14:32
Dr. Noor van Dijk · Casualty collection...
Practitioner contact in progress
active
Encounter closed
Encounter · 12 May at 14:32
Dr. Noor van Dijk · Casualty collection...
Ended 14:32
closed
Respiratory rate
Observation · 12 May at 11:14
Dr. Noor van Dijk · Casualty collection p...
24 /min high
final
Suspected botulinum-like neuroto...
Condition · 12 May at 11:12
Dr. Noor van Dijk · Casualty collection...
active · provisional · moderate
active
Current external identifier assigned
Identifier · 12 May at 10:03
Dr. Noor van Dijk · Casualty collection...
EMB-NFC-1042
active
14:32
Encounter History
Active encounter
Started 12 May at 14:32 · in progress
Casualty collection point Alpha
Closed encounter
Started 12 May at 11:02 · ended 12 May at 14:32
Casualty collection point Alpha
Encounter active
Encounter · 12 May at 14:32
Dr. Noor van Dijk · Casualty collection...
Practitioner contact in progress
active
Encounter closed
Encounter · 12 May at 14:32
Dr. Noor van Dijk · Casualty collection...
Ended 14:32
closed
Respiratory rate
Observation · 12 May at 11:14
Dr. Noor van Dijk · Casualty collection p...
24 /min high
final
Suspected botulinum-like neuroto...
Condition · 12 May at 11:12
Dr. Noor van Dijk · Casualty collection...
active · provisional · moderate
active
Current external identifier assigned
Identifier · 12 May at 10:03
Dr. Noor van Dijk · Casualty collection...
EMB-NFC-1042
active
14:32
Patients
Search patient or identifier
No patients found
Scan an external identifier or adjust the search query.
External identifiers are lookup keys only. Patient records remain the stable digital identity.
Home
Scan
Patients
Activity
Settings
14:32
Activity
Active encounters
No active encounters in this incident.
Recent records
No activity yet
Records will appear here when patient activity is documented.
Home
Scan
Patients
Activity
Settings


14:32
Settings
Practitioner account
Name
Dr. Noor van Dijk
Role
Practitioner
Unit
Field medical team Bravo
Incident context
unknown
Sync and connection
9 pending changes · last sync 12 May at 10:50
offline
Large field controls
Touch targets and dense clinical lists
Sign out
Return to practitioner login
Settings are local to this prototype.
Home
Scan
Patients
Activity
Settings