Table of contents

1. INTRODUCTION

1.1 Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict individual aspects of the system. It is intended to capture and convey the significant architectural decisions that have been made on the system.

1.2 Scope

The architecture described in this document concerns the e-police project. It is compliant with the laws of Ukraine on police and data protection.

1.3 Document Content Overview

After summarizing the architectural representation, goals and constraints, this document describes the system using several architectural views (Use Case, logical, process, deployment, implementation and data) and then concludes with size, performance and quality considerations.

2. ARCHITECTURAL REPRESENTATION

The next two sections of the document describe the architectural goals and constraints. Architecturally relevant Use Cases are described by a Use Case diagram and a short explanation of their impact on the architecture. The following views will also be provided:

- A use-case view provides a high-level view of the platform presenting the structure of the system through its use-cases and their interactions.
- An implementation view describes the software layers and the main software components. A component diagram is used in this view.
- A process view provides a description of the processes happening in the system and how they are linked together.
- A data view provides information about the data persistence. A class diagram will be used to model the main system data.

UML diagrams are systematically used to represent the different views of the system.

3. ARCHITECTURAL GOALS AND CONSTRAINTS

The following non-functional requirements that affect the architectural solution have been identified:

Non-functional requirement	Description
Robustness	The system shall be use stable and tested technologies, implement data backup and recovery procedures
Scalability	The system architecture should support the increase in systems' users and the deployment of new features
Maintainability	The system should be easy to adapt to changing requirements

4. USE-CASE VIEW

This section provides a representation of the use cases relevant for the architecture.

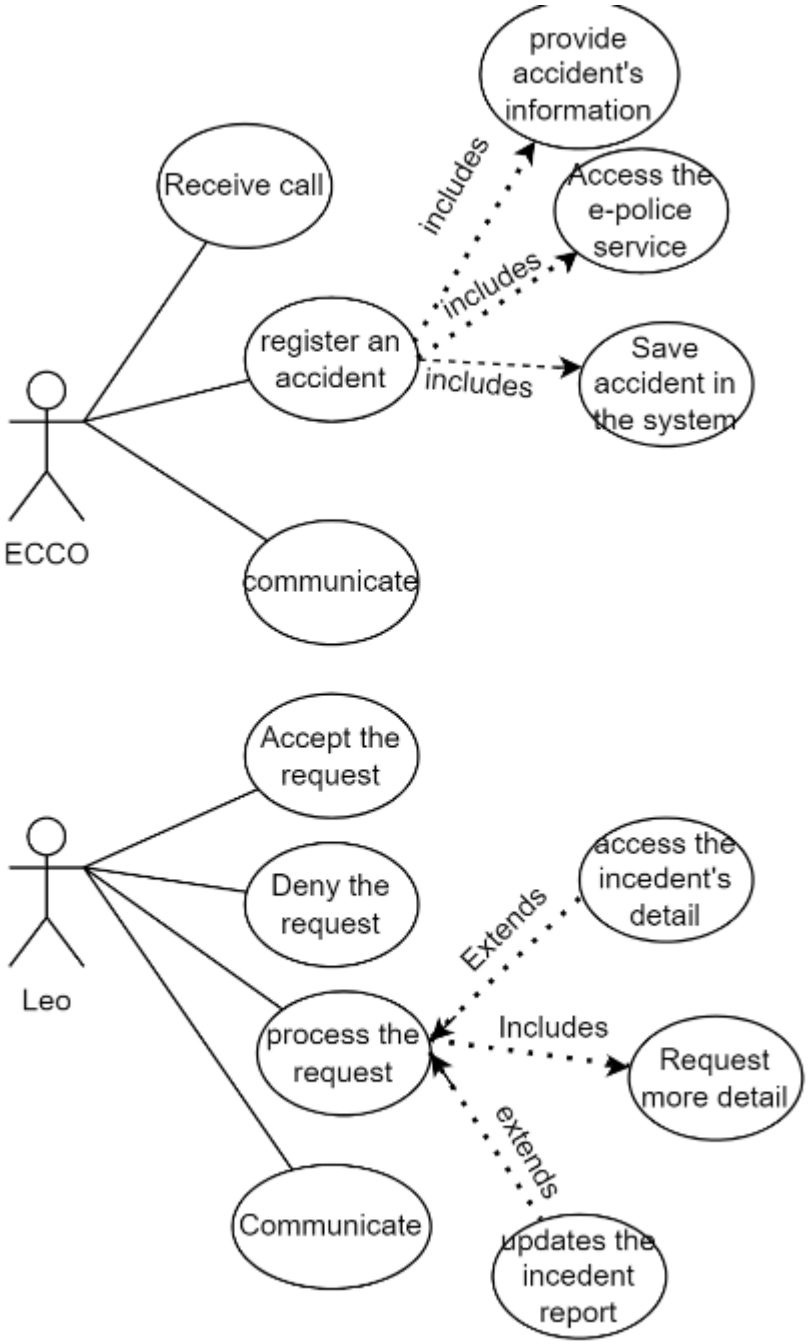
4.1. Selection Rationale

The use cases relevant for the architecture have been selected based on the following criteria:

- Use cases affecting the exchange between the call center system and the patrol squad's tablet client.
- Use cases representing critical parts of the architecture, thereby addressing the technical risks of the project at an earlier stage.

The following use cases have been selected:

- Call registration and handling
- Call acceptance and LEO status reporting
- Storing information discovered on the ground


provide accident's information
Receive call
includes
Access the e-police service
includes
register an accident
Save accident in the system
includes
ECCO
communicate
Accept the request
Deny the request
access the incedent's detail
Extends
Leo
process the request
Includes
Request more detail
extends
Communicate
updates the incedent report

5. IMPLEMENTATION VIEW

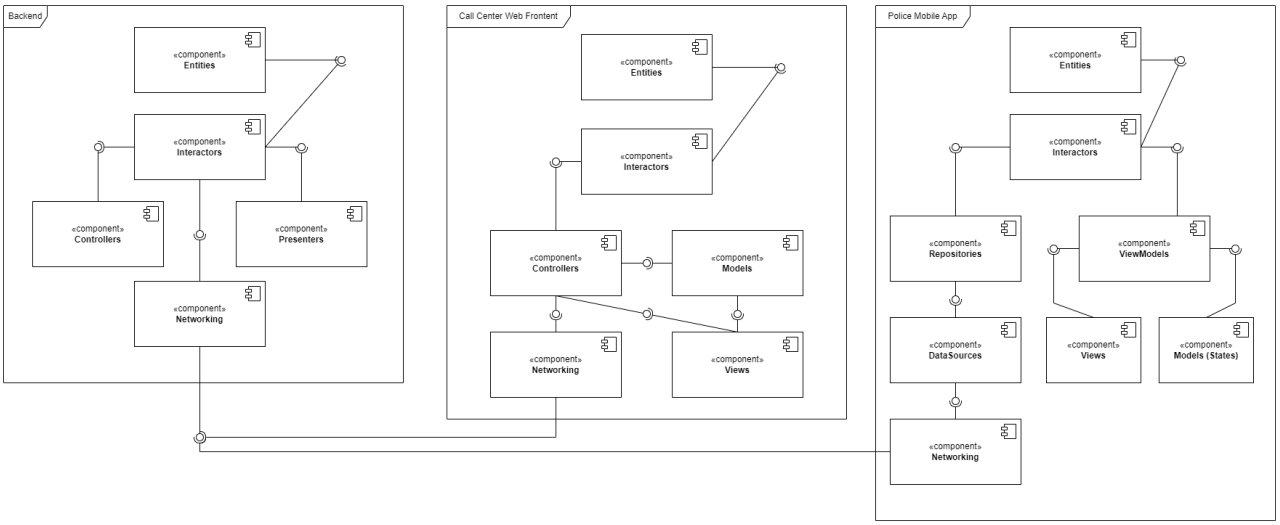
5.1 Overview

The following diagram describes the software layers of the system and their components.

Backend stores all the data and is responsible for the communication between the call center operators and police patrol squads.

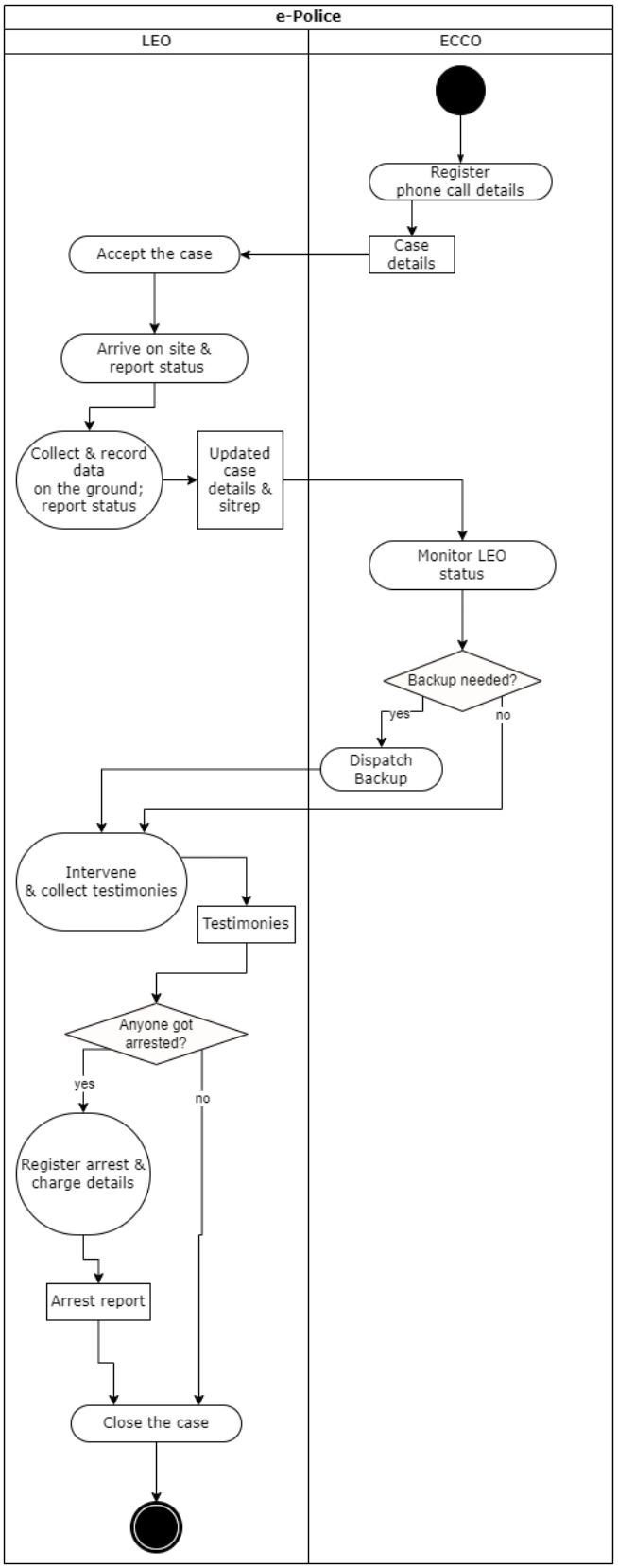
Call Center Web Frontend provides the interface for call center operators to register received calls, monitor LEOs status and dispatch reinforcements when needed.

Police Mobile App provides the interface for patrol squads to accept calls, report their status, register data gathered on the ground, and request reinforcements if needed.

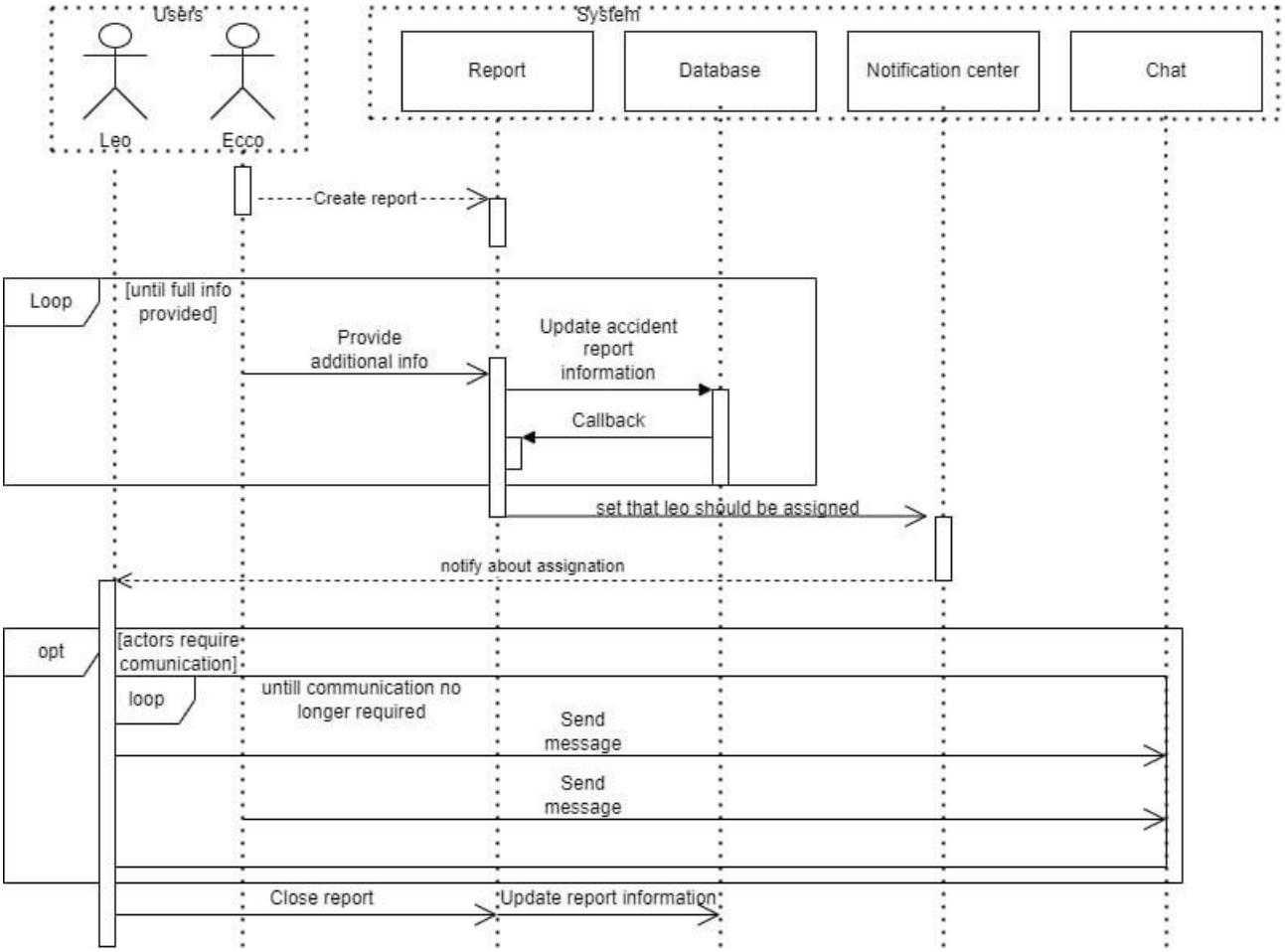


6. PROCESS VIEW

The process model describes the handling of an individual call in the system.



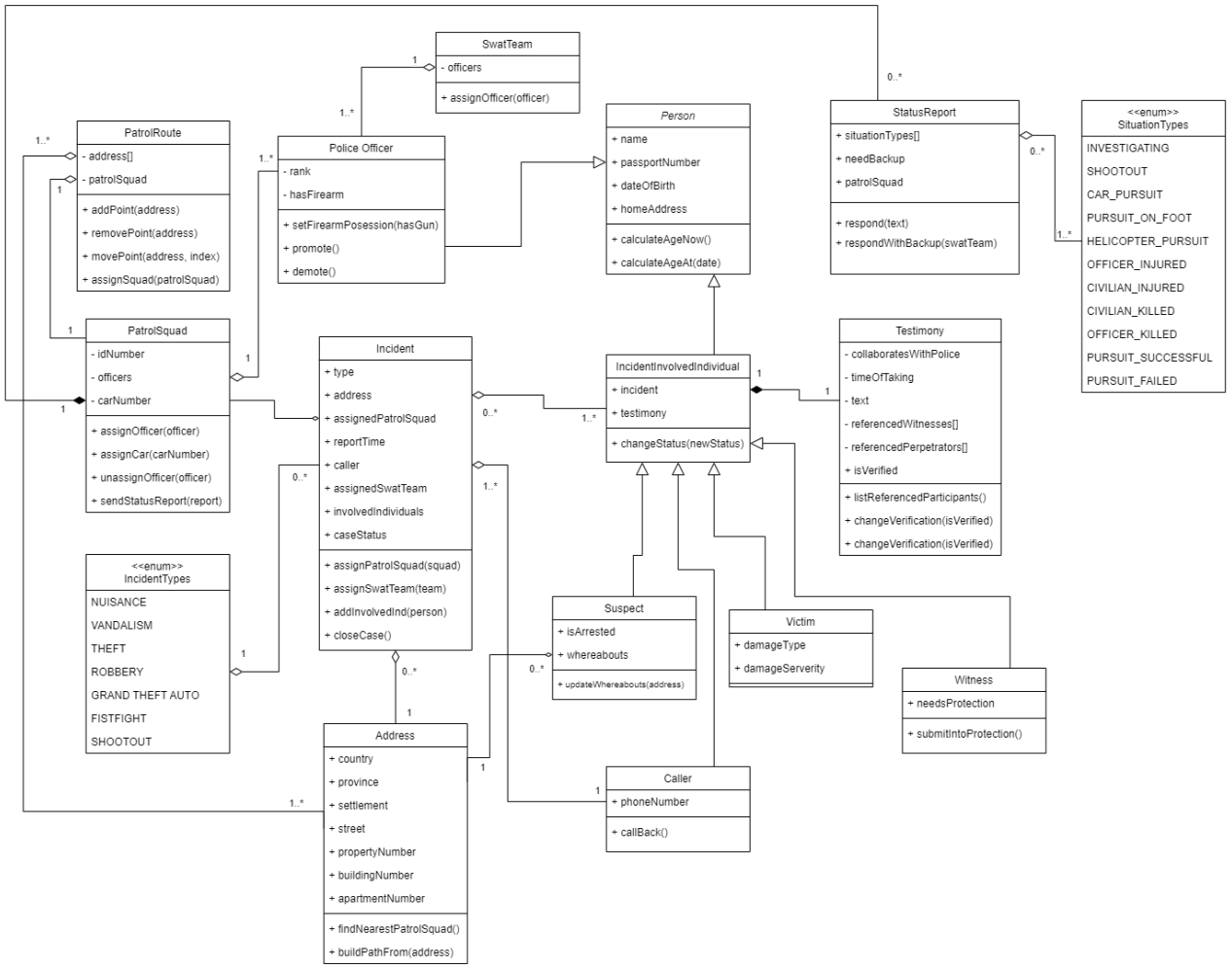
The following diagram describes the sequence of steps in the system when working on a case.



7. DATA VIEW

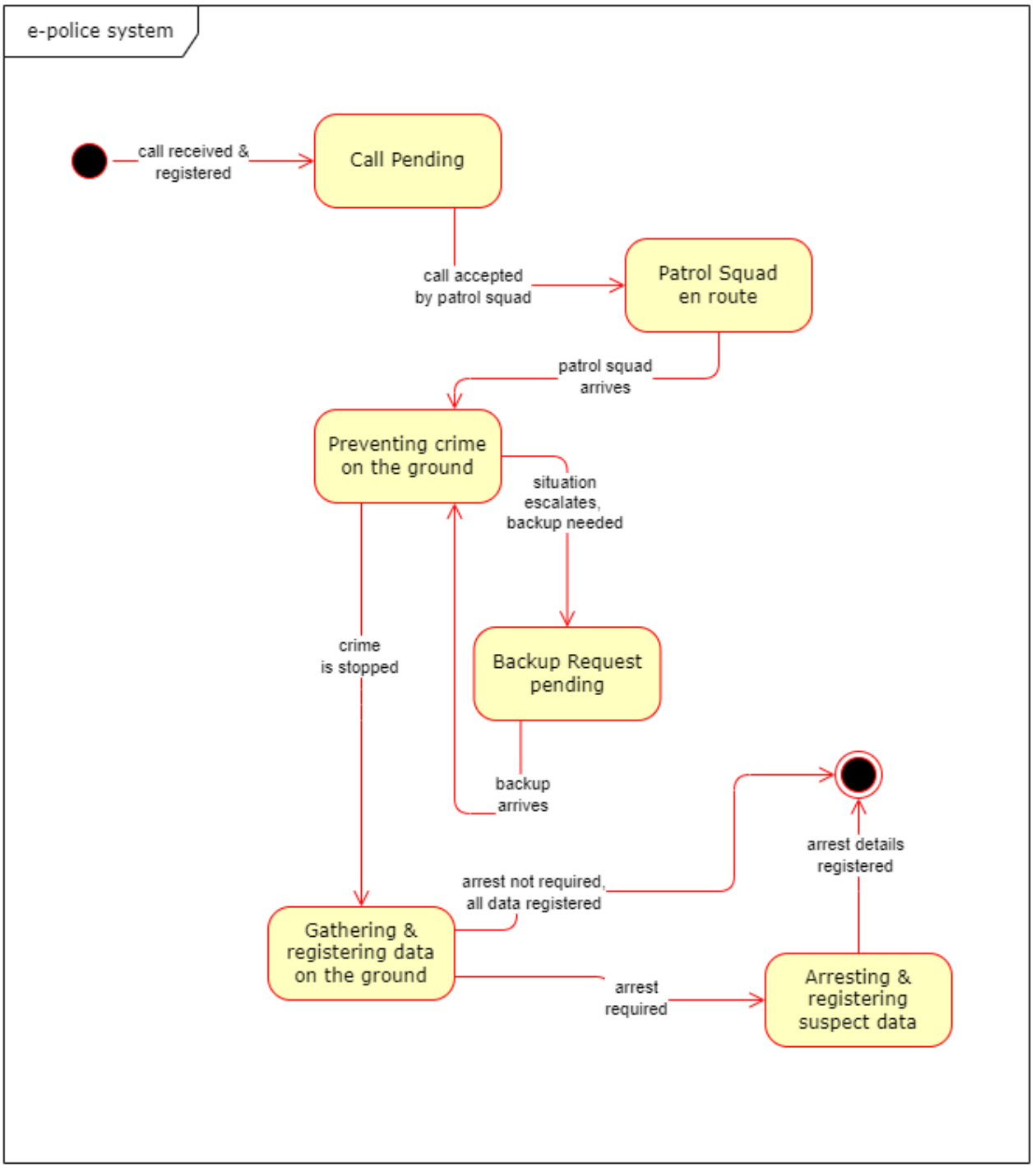
7.1 Data Model

The following diagrams show a high-level abstraction of the data entities, which must be implemented by the system:



7.2 State Machines

The system has the following state machine:


e-police system
call received & registered
Call Pending
call accepted by patrol squad
Patrol Squad en route
patrol squad arrives
Preventing crime on the ground
situation escalates, backup needed
Backup Request pending
backup arrives
crime is stopped
arrest not required, all data registered
arrest details registered
Gathering & registering data on the ground
arrest required
Arresting & registering suspect data

8. PERFORMANCE

An important decision that benefits the performance of the system includes the caching of database results for data related to cases currently in work. The front-end parts are implemented with the best performance practices relevant for web and mobile apps.

9. QUALITY

9.1 Maintainability

The usage of clean architecture and clear architectural boundaries makes it easy to implement new features as requirements evolve, as well as fix any bugs found during usage.

9.2 Portability

The architectural decision to decouple the business rules from the details of implementation, including the operating system, the database, and UI frameworks, creates the opportunity to easily swap the database, port mobile app to desktop with Java Swing, port backend software to a different operating system.

9.3 Reliability

An automatic retry policy ensures that parts of the system can continue functioning without losing data when an issue occurs in a specific component. The system has data backup & restore capabilities, as well as disaster recovery capabilities.

10. CACHING

In order to enhance the performance the system caches data from the database that is relevant to the cases currently being handled.