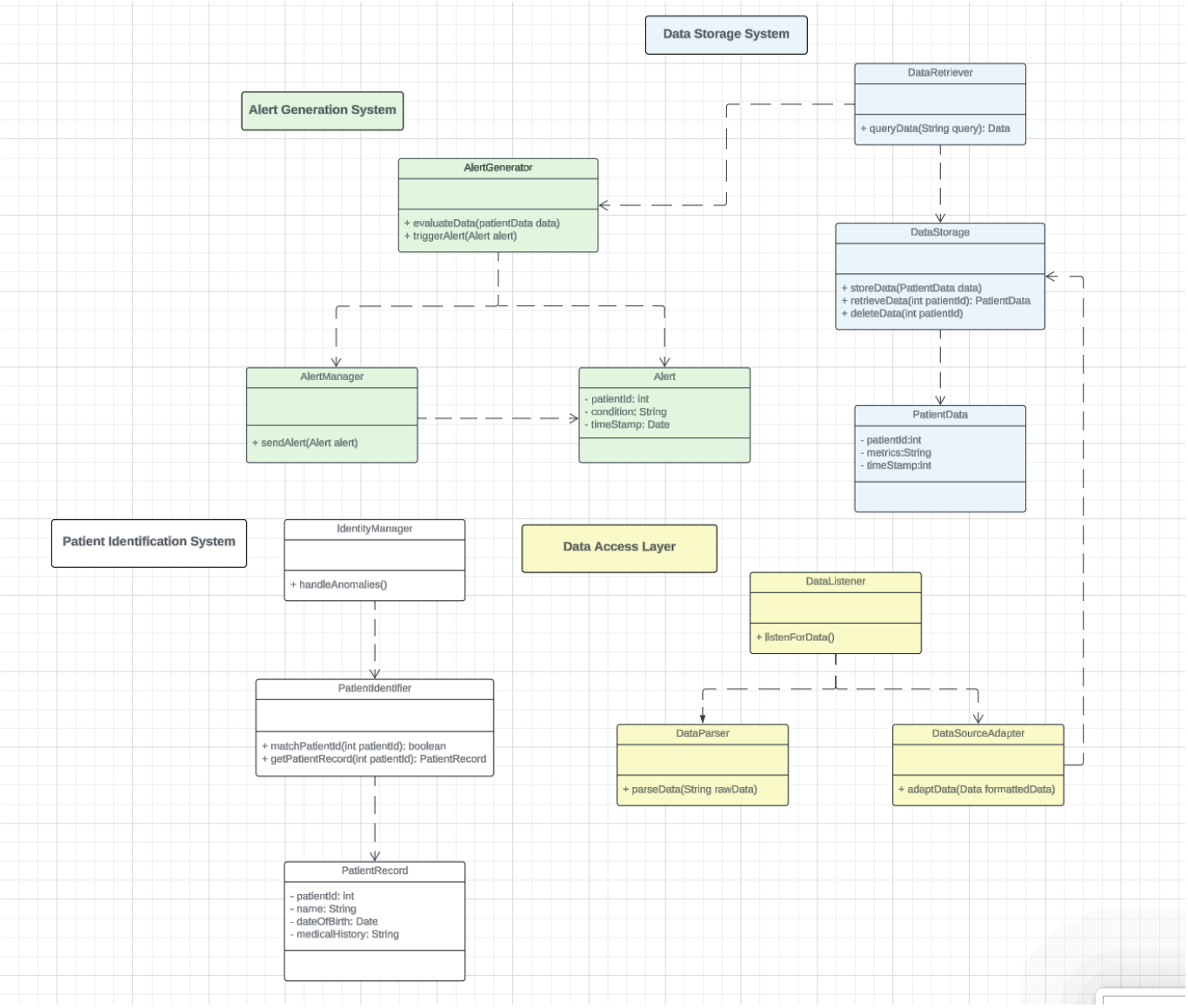
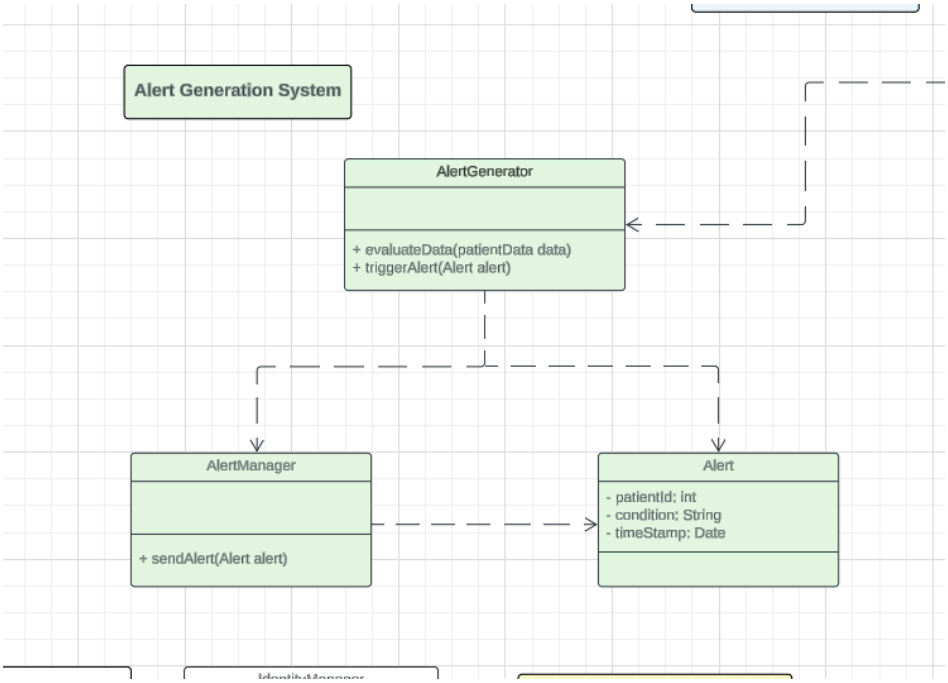
The UML diagram of the Health Monitoring System

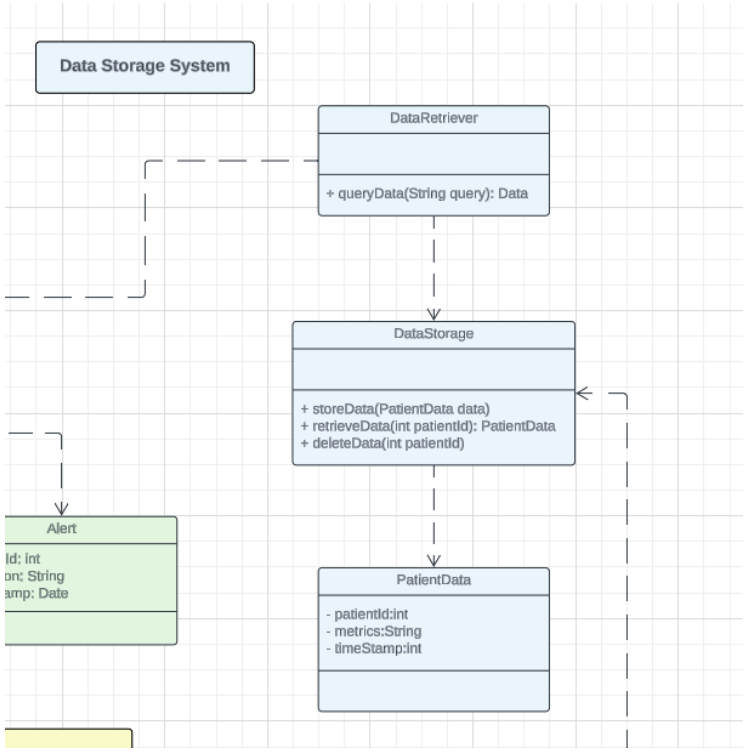


The UML diagram of the Alert Generation System



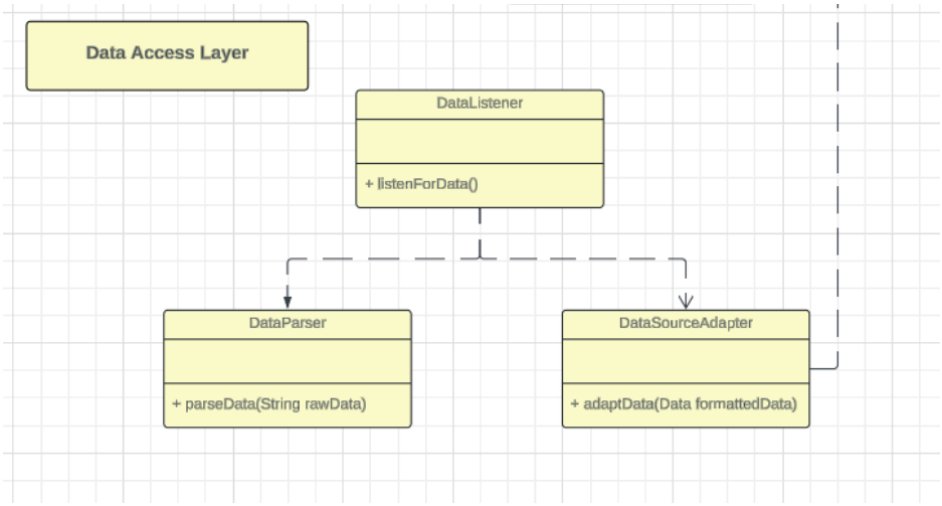
The UML diagram for the Alert Generation System sums up and visually represents the essential components and interactions that are crucial for a seamless functionality of the program. It contains three main classes: AlertGenerator, Alert and AlertManager. The AlertGenerator class is responsible for continuously monitoring and evaluating patient data streams, from heart rate monitors, blood pressure cuffs and other devices. It's main two methods are: evaluating the data and triggering an alert. It compares the patients data to the personalised alert threshold and in case of an abnormal state and alert is triggered, in which case, the Alert class captures the critical information. These Alert instances carry important details such as the patient's ID, the specific condition detected, and the time it occurred. They represent the system's way of flagging abnormalities in patient data. The alerts are managed by the AlertManager class. It is responsible for sending the alerts to relevant medical staff. We chose this design because the communication of these three classes create a flow of data evaluation, alert generation and communication with other systems, like fetching historical data of specific patients for more complex condition evaluations from the Data Storage system and interfacing with the Patient Identification system to retrieve patient details based on the patient ID.

The UML diagram of the Data Storage System



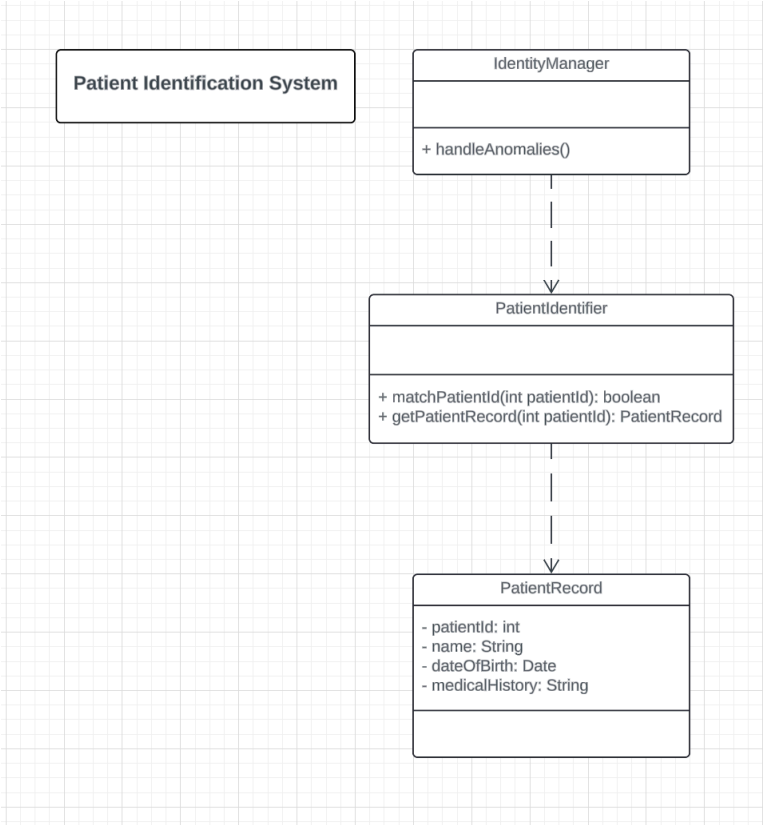
The UML Class Diagram for the Data Storage System illustrates the main components and operations that are needed for managing and storing patients data securely and efficiently. It contains three main classes: DataRetriever, DataStorage and PatientData. The core of the system is the DataStorage class. Its task is to store and handle patients data that are both historical and actual. It provides two other methods for deleting obsolete records and retrieving existing data. It extends the PatientData class, which represents individual patient records, including the patients ID, vital metrics such as heart rate and blood pressure, and timestamps. The DataRetriever class is responsible for securely retrieving the data of patients, that is handled in the DataStorage class. Since the DataRetriever class extends the DataStorage class, it is fetching data from it. The retrieval of the data is not only secured, but also followed by the privacy regulations and access control policies. The Data Storage system serves as an interface and ensures stabile communication with the Alert Generation system and Data Access layer. The Data Access layer system gives the DataStorage class, processed and unified into one standardised format data of the patients, which is then passed on to the AlertGenerator class. The Alert Generation system, then evaluates the data. We chose this design since it ensures real-time monitoring and a seamless flow of communication of the systems.

The UML diagram of the Data Access Layer



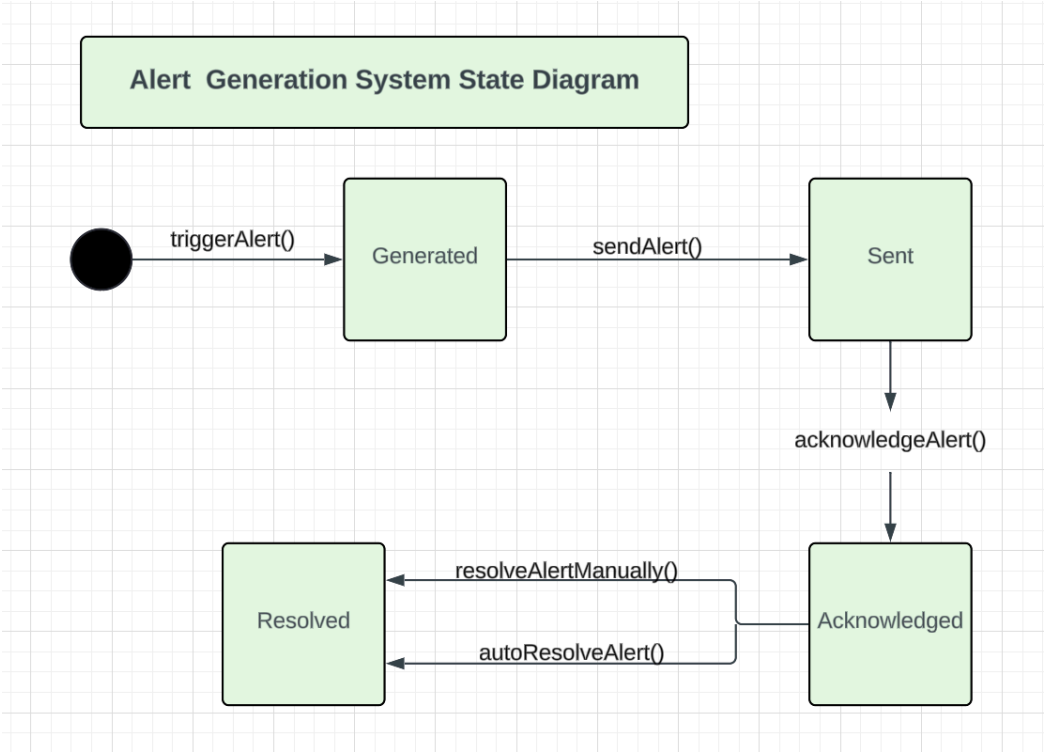
The UML Class Diagram of the Data Access Layer visually represents the operations and integration of classes that are necessary for effectively handling the format of patient data, inflowing from diverse sources. It encapsulates three classes: DataListener, DataParser and DataSourceAdapter. Each play a different role in the data inflow, processing and storing. The DataListener class serves as an interface and is responsible for ingesting prompt and raw data from specific data sources such as, TCP/IP, Websocket and file uploads, and transferring it into the system. The data is passed on to the DataParser class, which standardizes the raw data into a unified format. This standardization process ensures consistency and compatibility across different data sources, enabling smooth processing and analysis downstream.The DataSourceAdapter class's task is to manage the final processing and storing the unified and organised data. It passes it into the Data Storage System. We chose this mechanism because it is flexible, robust and ensures that the patients data can be measured from various devices and sources. This flexibility not only ensures the safety and accuracy of data measures, but also enables the system to adapt to evolving data sources and technologies. It makes the healthcare monitoring system more effective and reliable.

The UML diagram of the Patient Identification System



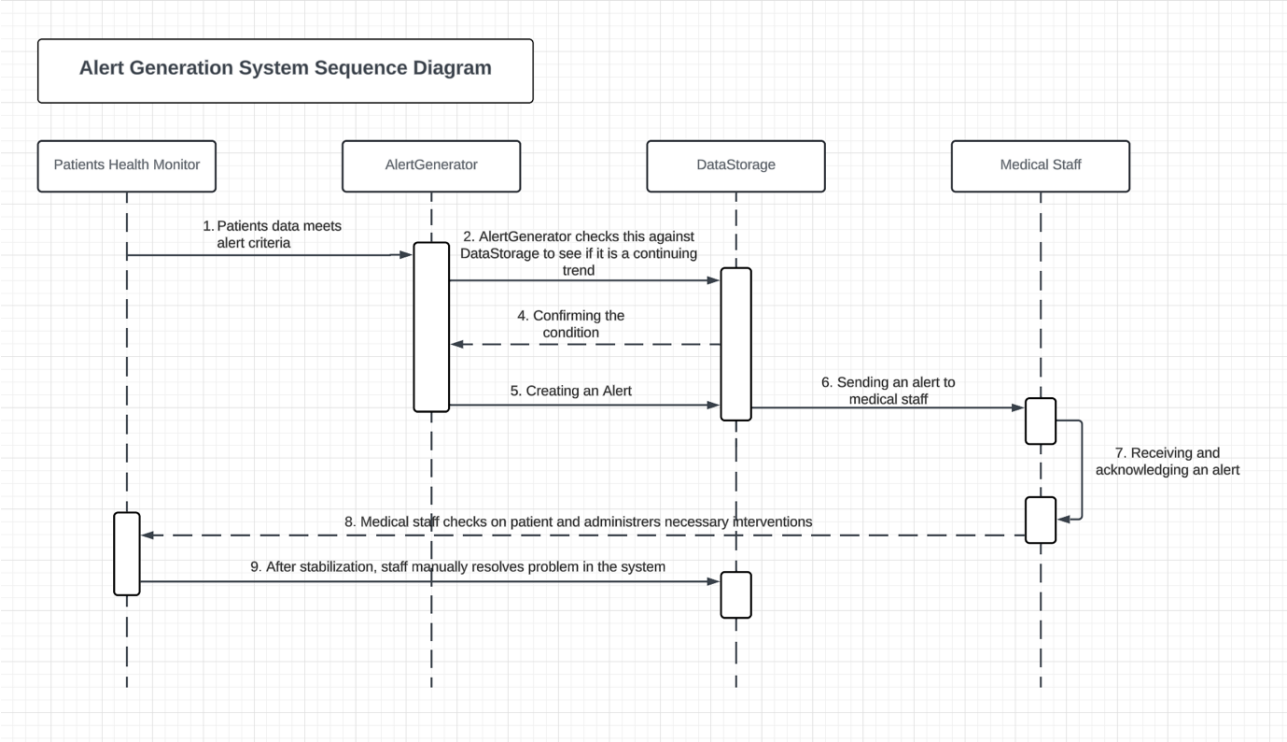
The UML diagram of the Patient Identification System represents the three essential classes and operations which are involved in accurately linking incoming patient data to the correct patient profiles. The three classes that are involved in the system are: IdentityManager, PatientIdentifier and PatientRecord. The Patient Identifier class cross-references incoming data identifiers with the hospital's main patient database to ensure each piece of data is correctly attributed. The class also encapsulates two methods in case there is a scenario where patient records are sources from external hospital databases. These two methods handle matching patient ID's and retrieving patient records. The PatientIdentifier class extends the PatientRecord class which represents comprehensive patient records, including personal information, medical history, the name of the patient and the date of birth. The IdentityManager class oversees the process of cross-referencing and handles discrepancies or anomalies in data linkage, thus maintaining the integrity of patient records. It extends the PatientIdentifier class. We chose this design because the integration of the classes ensures accuracy of the patient identification which prevents future medical mistakes and decisions. This system's role plays in the patients safety which is extremely important in healthcare systems.

State diagram of Alert Generation System



The Alert Generator System State Diagram represents the states and transitions that happen in the system. The main goal of this system is to ensure timely detection of distress or critical medical conditions in patients by continuously evaluating incoming data and triggering alerts when it is necessary.
The first state in which the alert finds itself, is waiting for incoming patient data to evaluate. Once the data is evaluated and meets the criteria for generating an alert, which may include elevated heart rate or irregular blood pressure levels, the system transitions to the Generated state. In this state the Alert Generator class sends an alert containing critical information, including the patientsID, the condition that triggered the alert and a timestamp. The Alert Generator transitions to the Acknowledged state. When the medical staff acknowledge the receipt of an alert, either manually or automatically, the Alert Generator transitions to the Acknowledged state. Finally, alerts are resolved through either automatic detection or manual confirmation by medical staff after assessing and potentially adjusting treatment. The Alert Generator transitions into the Resolved state.
This visual representation offers a comprehensive understanding of how alerts are managed within the system, facilitating timely responses to critical medical conditions and ultimately enhancing patient care and safety, by ensuring effective communication and intervention.

Sequence diagram of Alert Generation System



Sequence diagram of an Alert Generation System represents flow of interactions between the key components (Patient Health Monitor, Alert Generator, Data Storage and Medical Staff) in response to the detecting and handling of an alert triggered by Patient Health Monitor (for example too fast heart rate).

First, the Patient Health Monitor detects the stimulus triggering an alert and informs Alert Generator about it. Then the Alert Generator checks the trend of the detected data against the Data Storage to determine if it signifies a continuing condition. Upon confirmation the Alert Generator generates an alert and sends it to the Data Storage and next to Medical Staff. When they get it, they take necessary actions to solve the problem and when the condition is stabilized, they manually resolve the alert in the system.

The sequence diagram gives us an opportunity to show the chronological order of the actions and message exchanges. Each separate event contributes to the patient safety and medical intervention, reflecting the system's ability to support healthcare professionals. We chose a simple design in which every arrow represents the start/ finish of an occurrence and every block is an execution. Above very arrow there is a message indicating an action.