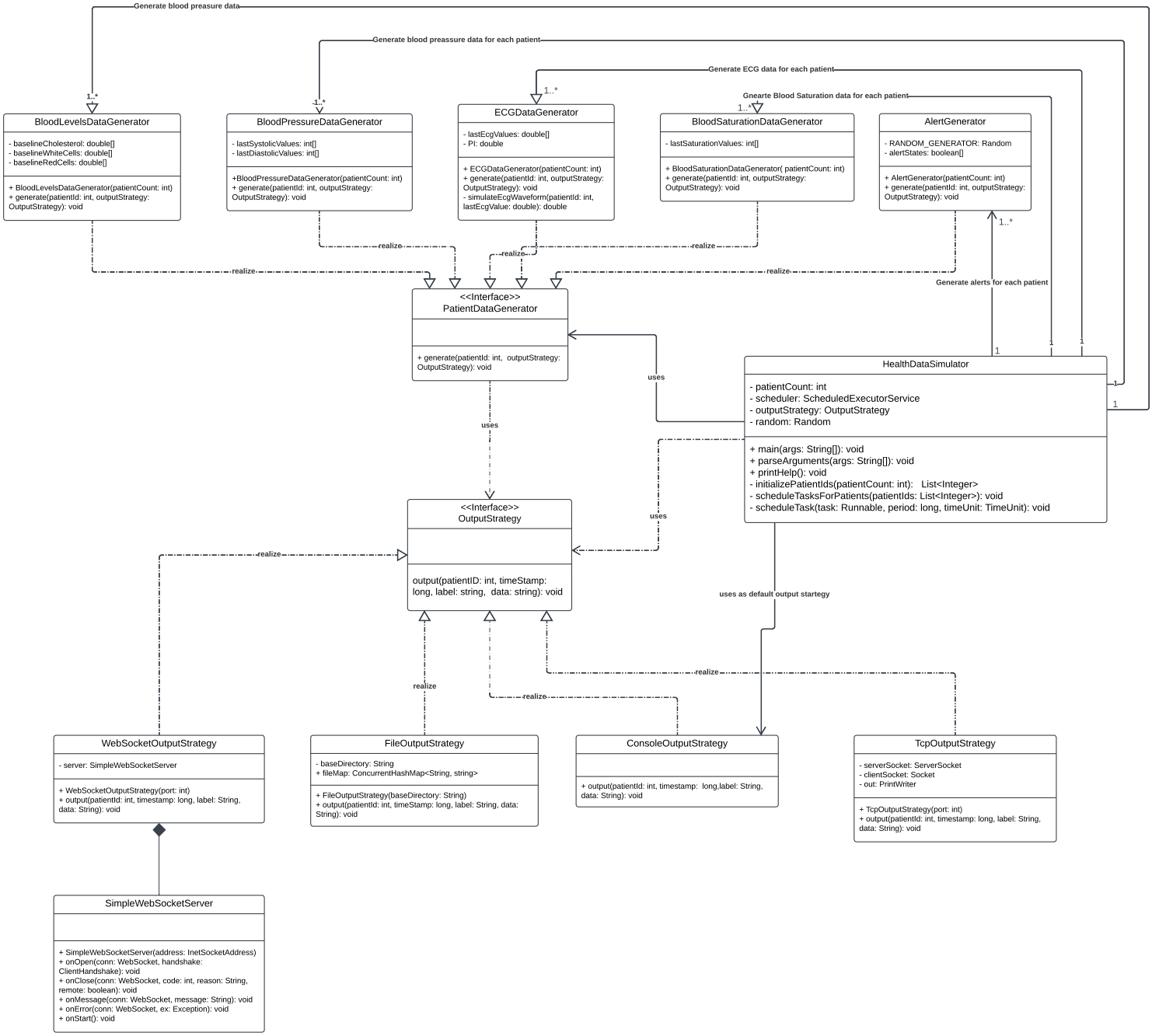
General Class Diagram for the Signaling Project



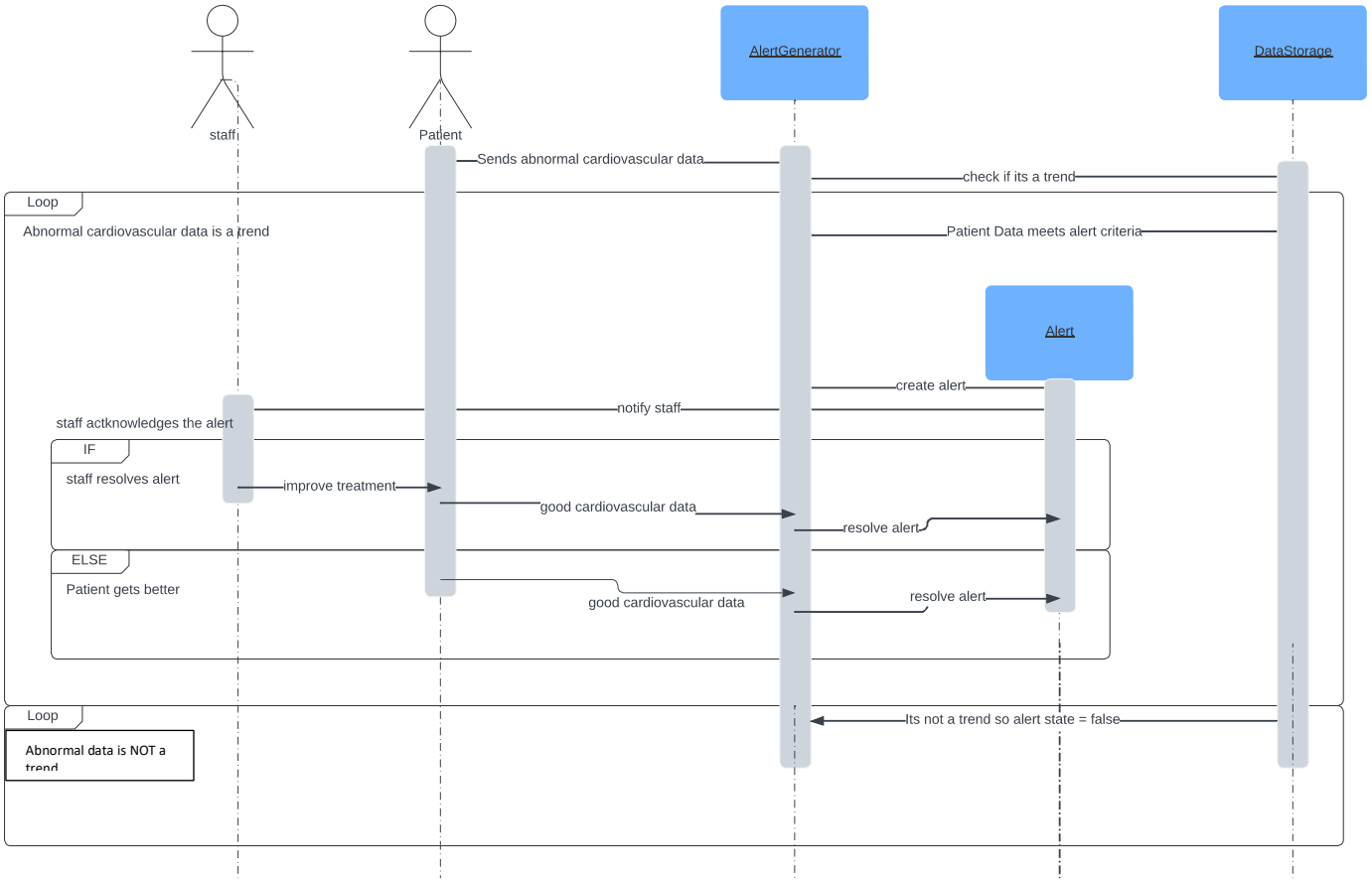
The class diagram depicts the structure and relationships of classes within the cardio_generator system.
At the core is the HealthDataSimulator class, responsible for orchestrating data generation and output strategies.
It maintains an outputStrategy attribute, allowing flexible output configurations.

Four output strategies are supported: ConsoleOutputStrategy, FileOutputStrategy, WebSocketOutputStrategy, and TcpOutputStrategy. Each strategy implements the OutputStrategy interface, defining a output() method for transmitting data. ConsoleOutputStrategy is the default option. Also, WebSocketOutputStrategy owns a private class SimpleWebSocketServer responsible for server activities specific for this project.

These strategies facilitate modular and extensible data output. Additionally, the HealthDataSimulator class orchestrates the simulation process through PatientDataGenerator Interface, which is realized by BloodSaturationDataGenerator, BloodPressureDataGenerator, BloodLevelsDataGenerator, and ECGDataGenerator classes.

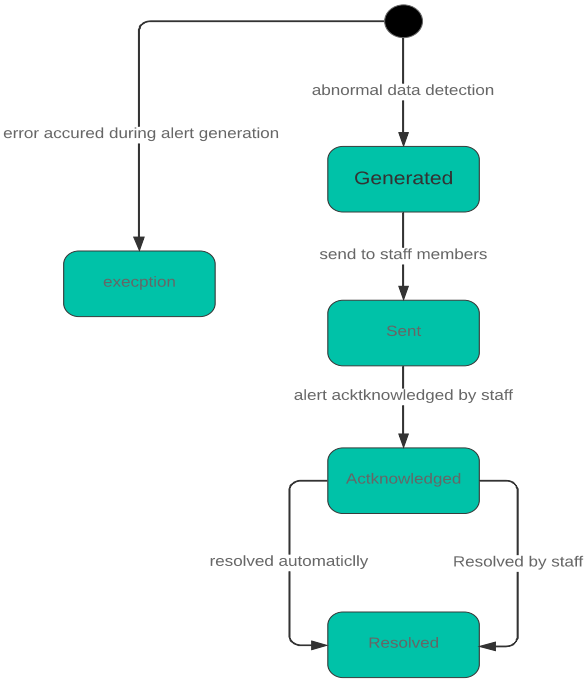
Each of these classes inherits a generate() method responsible for generating data for each patient and then outputting it with a user-specified OutputStrategy.

Sequence diagram for alerts



The sequence diagram depicts the alert generation process, starting from patient data detection.
The AlertGenerator interacts with DataStorage to confirm trends or retrieve past incidents.
Once confirmed, the AlertGenerator creates and sends alerts. Medical staff receive and Acknowledge alerts, triggering necessary interventions to resolve the issue OR the issue resolves itself, a.k.a. patient gets better.

If however, the data storage disconfirms the trends based on past data, then the alert is not created and the problem is resolved for the time being.

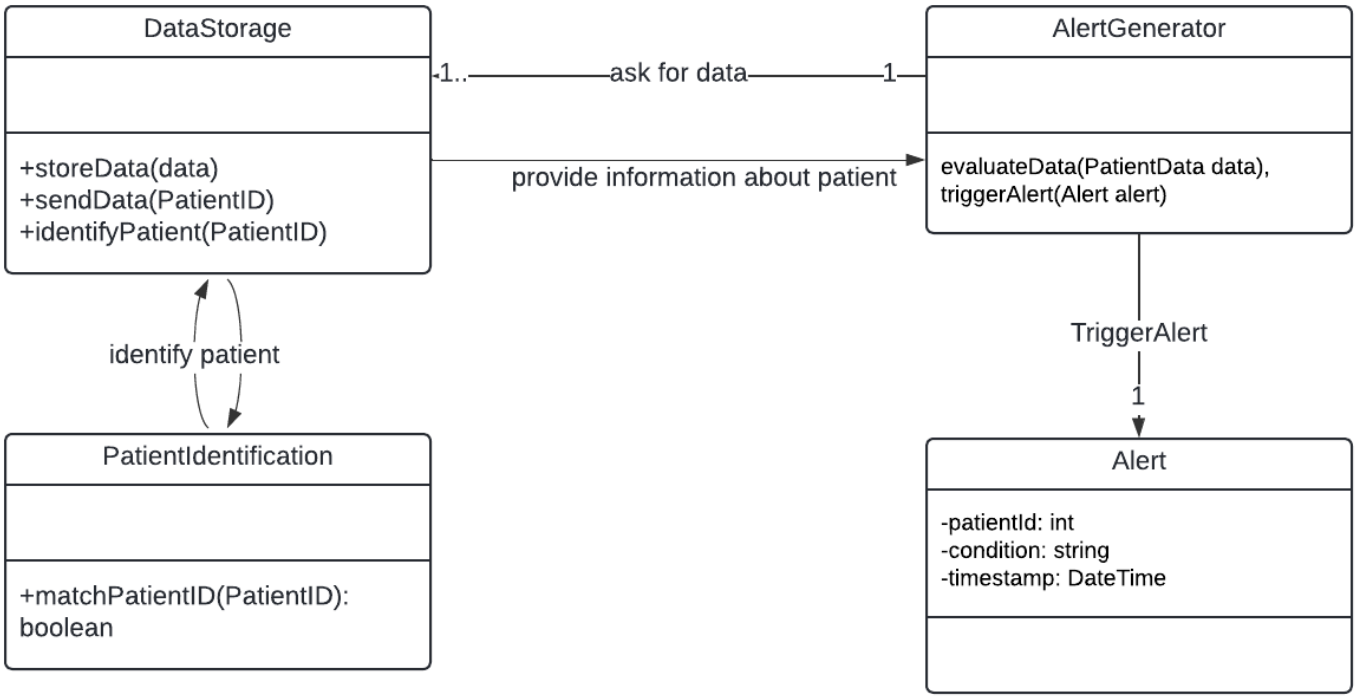


State diagram for alerts

The state Diagram depicts the creation of alerts and possible errors along the way. Once created the alert
is sent to the medical staff and ready to be acknowledged. After being acknowledged, the staff can either
resolve the issue themselves by improving the patient's treatment

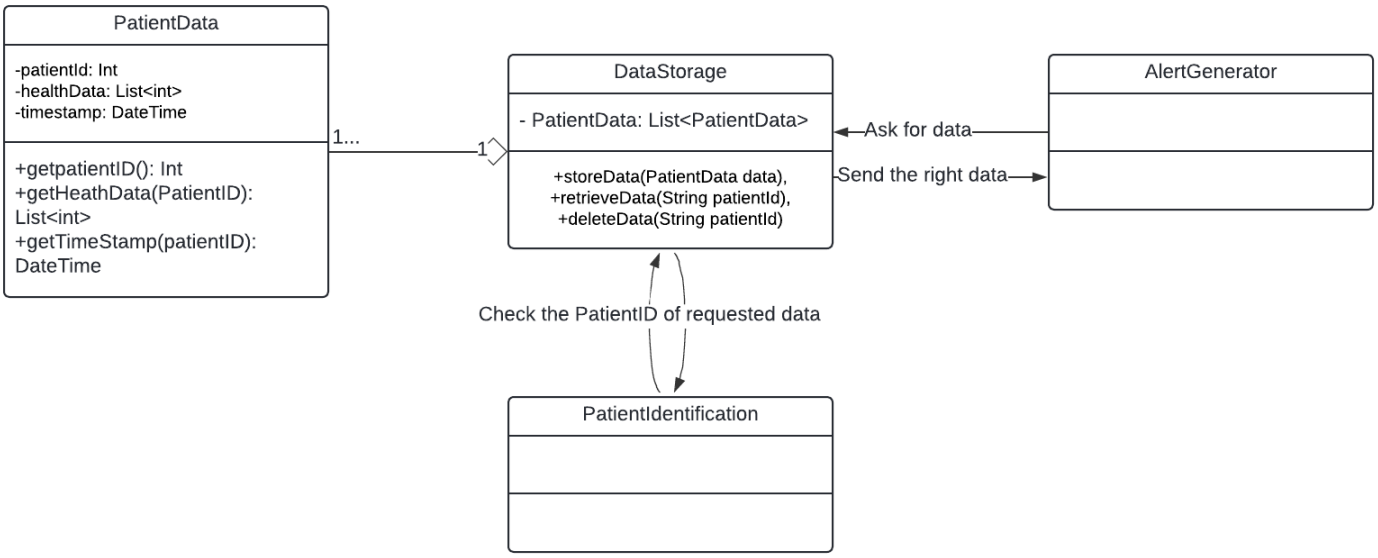
OR wait for the patient to automatically
get better and the issue resolves itself.

Class diagram for Alert Generation System



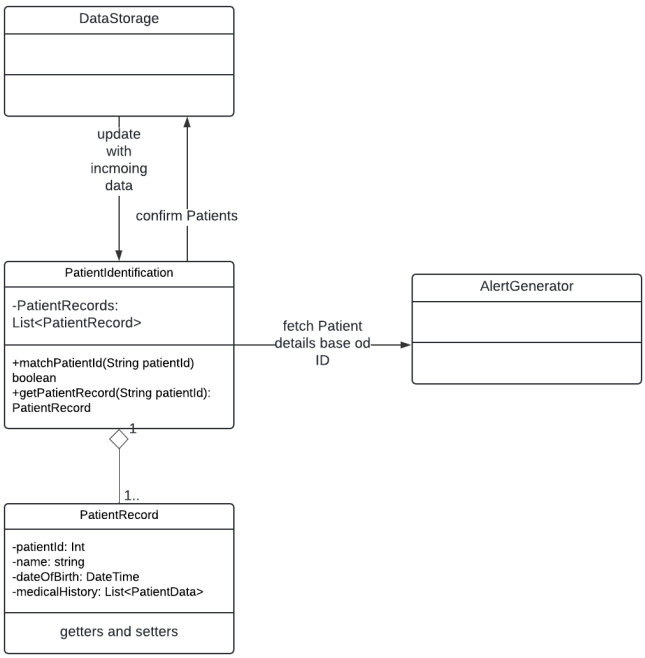
When AlertGenerator checks whether the current abnormal data is a trend or not, it has to ask Data Storage for that information about a patient. In order for the DataStorage to give the right data it has to first check whether it's the correct patient in the PatientIdentification. Once the DataStorage provides back the data to AlertGenerator, it evaluates the data and possibly triggers an alert which is an object of class alert.

Class diagram for Data Storage System



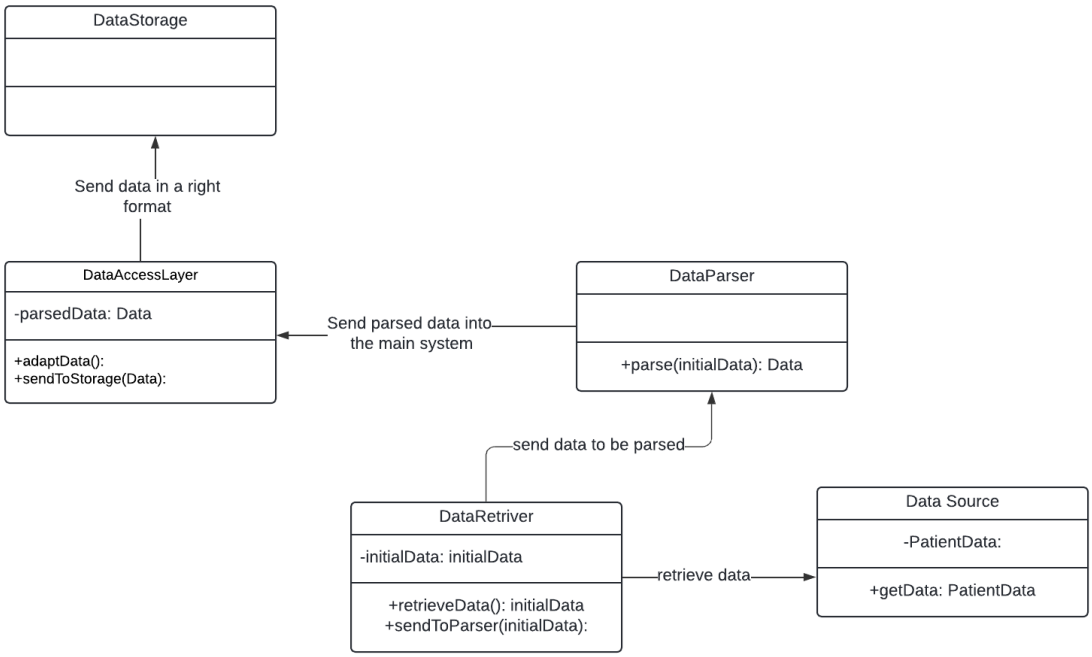
The DataStorage is itself a container class for PatientData with abilities to communicate and abstruct the data. It is used for storage of new PatientData as well as retrieval of old ones, for example, by AlertGenerator. It also has the ability to confirm Patients with its interactions with PatientIdentification class.

Class diagram for Patient Identification System



The main idea behind the PatientIdentification class is to be able to store personal, identifiable information about each patient in the form of PatientRecord class. It is also able to fetch that data to AlertGenerator, as well as confirm patients for DataStorage and update its records based on data flowing from DataStorage.

Class diagram for Data Access Layer



Here the Data is first retrieved from the force in the source-specific format. Then that data is sent to the parser, which conforms the data into right format and sends it into the data access layer, which can then adapt it further to the specific storage's need and send it to be stored there, in the correct format.