## Alert Generation System

The AlertGenerator checks patient data that is sent to the AlertGenerator in real-time with evaluateData, comparing it against personalissed thresholds for each patient. If any data crosses the thresholds that were compromised, the triggerAlert method creates an Alert object with patient details and the issue. The alert's details include the patient's ID, the specific condition, and a timestamp. The AlertGenerator might also get historical data from DataStorage to ensurethe severity of the alert. Finally, the AlertManager sends the alert to the medical staff's devices to keep them in the loop and ensure quick response times.

## Data Storage System

The Data Storage System ensures all patient info is stored safely and is easy to access when needed, supporting both real-time monitoring and long-term analysis.It securely keeps all patient data to make sure that it is organized and easy to retrieve. It stores data using storeData, making sure everything is encrypted to protect patient privacy. Authorized staff can get the data they require with retrieveData, allowing them to view both current and historical patient data. Additionally, old or unnecessary data can be removed using deleteData to maintain efficient storage management.Patient Identification System

## Patient Identification System

The Patient Identification System matches incoming data with existing patient records to ensure and provide accuracy. The PatientIdentifier uses matchPatientId to cross-check IDs with the hospital's database, ensuring each piece of data is linked to the correct patient. The IdentityManager oversees this process, handling any discrepancies that might arise and ensuring data integrity. When needed, getPatientRecord pulls up the patient's full record, including medical history and personal details. This system helps keep track of patient data accurately, avoids mix-ups, and maintains up-to-date records that are crucial for effective patient care.

## Data Access Layer

The Data Access Layer connects to different data sources using various DataListener classes like TCPDataListener, WebSocketDataListener, and FileDataListener. These listeners getthe data from the generator and then send it to the DataParser, which in return will turn it into standardised PatientData objects. Afterwards, the DataSourceAdapter then sends this data to DataStorage for secure keeping and further processing. This design ensures that data from different sources is handled efficiently and consistently. By supporting multiple communication protocols, this layer makes it easy to adapt to various setups and ensures that the system remains flexible and robust.