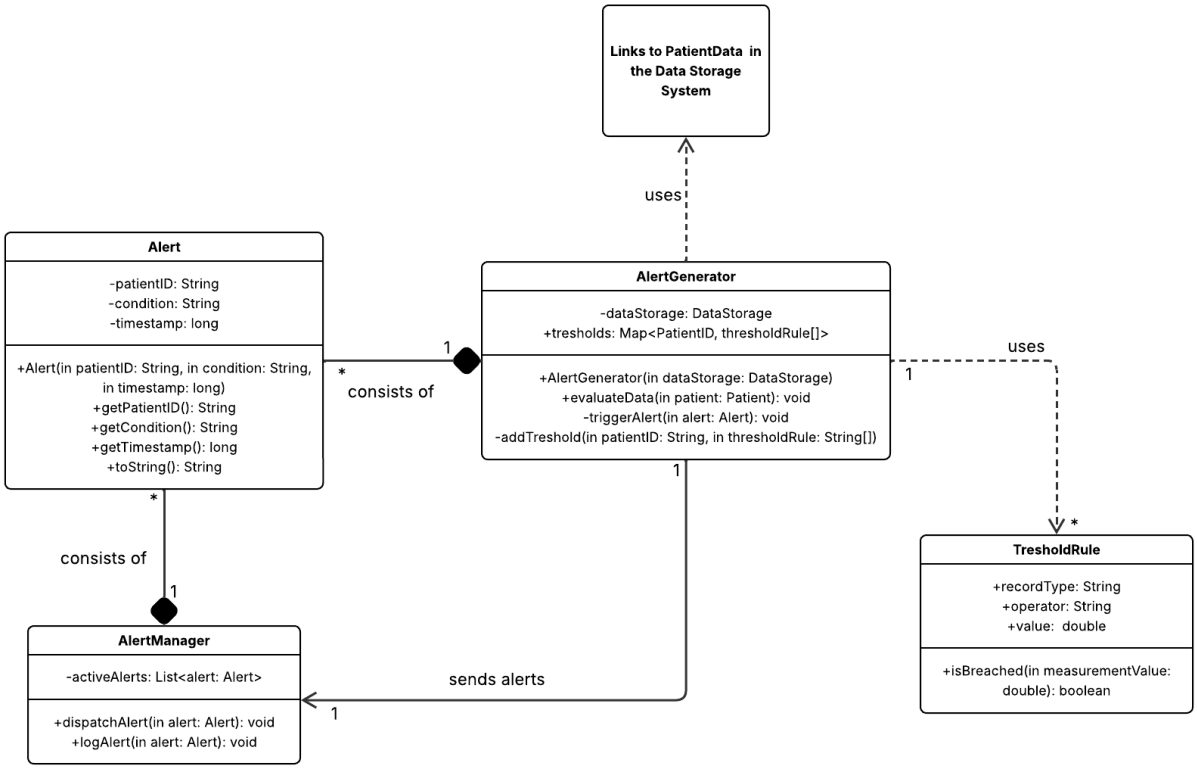
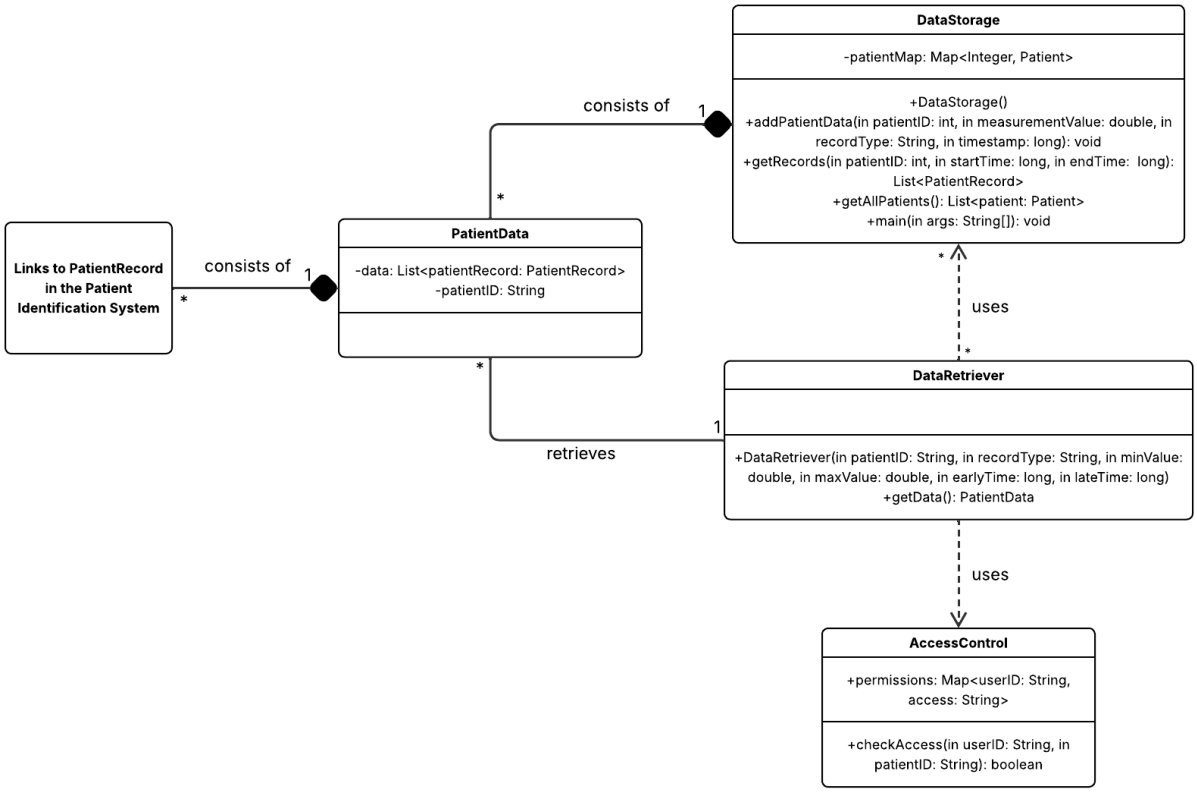
WEEK 2 ASSIGNMENT

1. Alert Generation System



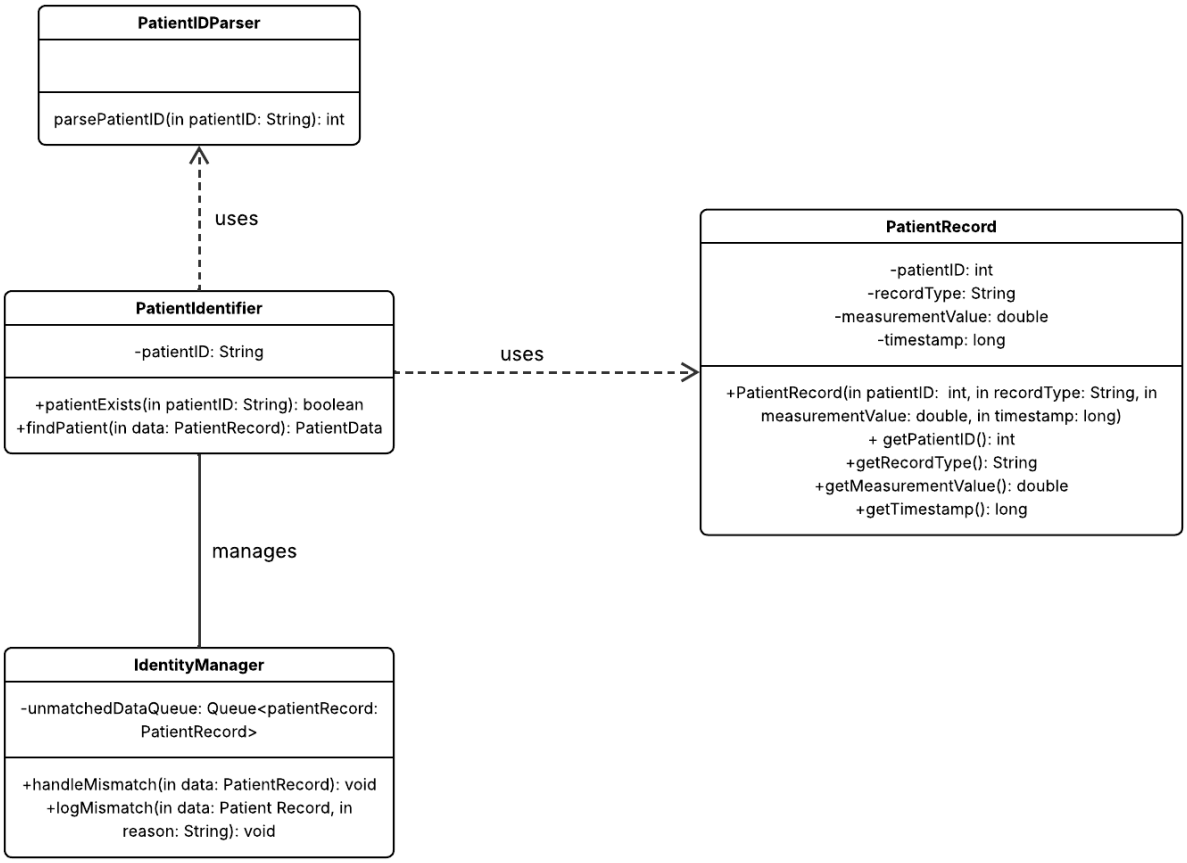
This subsystem generates alerts if the data in some patient's records are beyond certain boundaries. These boundaries might be different for each patient, this is why I chose to make the TresholdRule class responsible for the thresholds objects. These thresholds are stored in an instance field in AlertGenerator that maps each patientID to an array of their personal thresholds. These thresholds are then accessed by AlertGenerator in order to discern whether the data is beyond the thresholds, if it is, an alert is triggered and sent to the AlertManager. Alert Manager then logs the alerts it receives and notifies the necessary medical staff.

2. Data Storage System



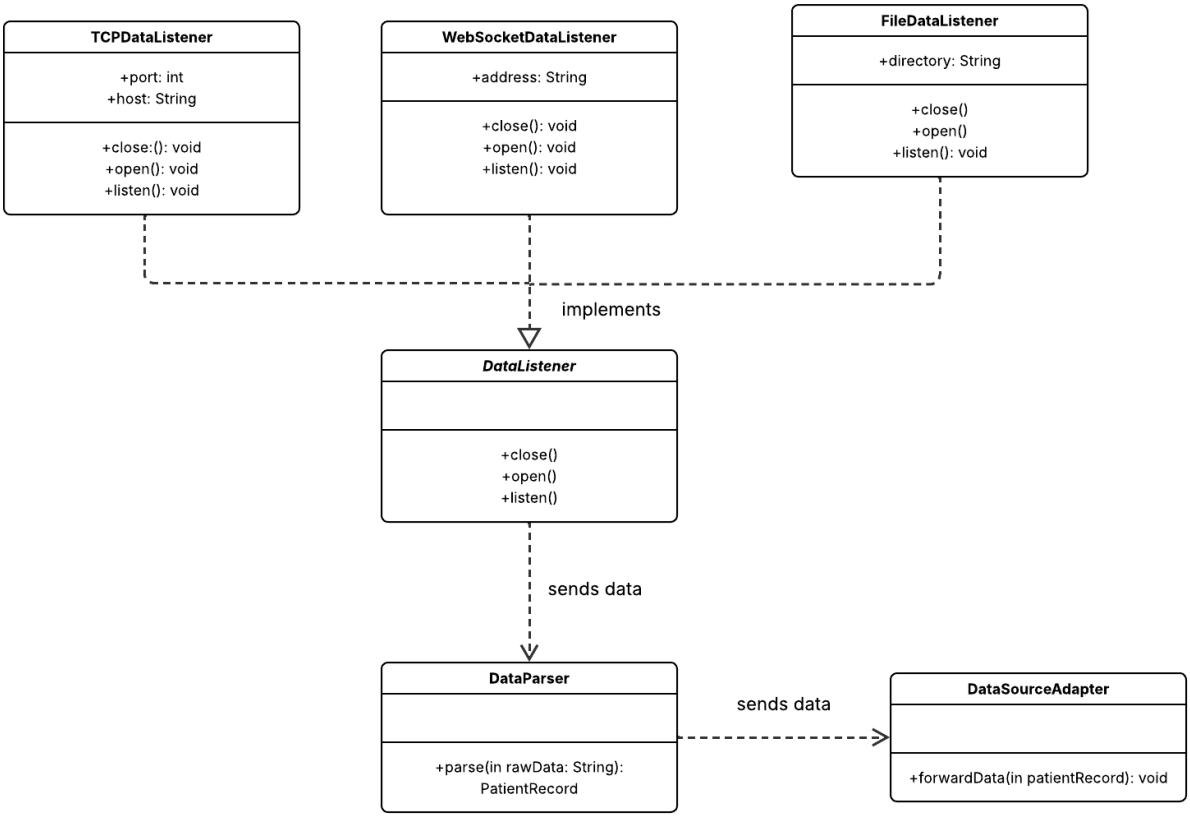
The data storage system acts as a database for the patient's data, that adds, stores and retrieves patient's data. The DataStorage class is in charge of storing such data, whereas the DataRetriever class is responsible for retrieving data with the use of queries from the medical staff. Due to the sensitive nature of the data, I decided to add an AccessControl class responsible for validating if the user is authorized to access the patient's records. This adds an extra layer of security to stop the stored data from being compromised.

3. Patient Identification System



The Patient Identification System's goal is to receive incoming data and match them to patient's records. The PatientIdentifier class does this, with the help of the PatientRecord class. Because the incoming data received by PatientIdentifier has the patientID in a String data type, but PatientRecord stores patientIDs as integers, I decided to add a PatientIDParser class responsible for parsing patientIDs from Strings to integers to be used. This way PatientIdentifier can do its job properly, and IdentityManager handles any possible mismatches between the incoming data and the data in the patient's records.

4. Data Access Layer



Lastly, the data access layer receives incoming data as inputs to various machines. In order to standardize the inputs from a wide variety of kinds for later processing, the DataListener interface outlines the basic functionalities that all input formats have in common. This interface is then implemented by the three different data sources, each with its own class. The data is then transferred to the DataParser as raw data for processing, this class turns the data into PatientRecord objects to be properly stored in the rest of the system. Lastly the data is transferred to the DataSourceAdapter class that is responsible for forwarding the processed data to the rest of the system for appropriate storing.