***UML Class Diagram***

The class diagram we’ve created includes all four of the systems and explains the monitoring, storage, management, and identification process of the CHMS. The Alert Generation System features the “AlertGenerator” class, which is responsible for continuously monitoring incoming patient data and evaluating it to determine if an alert should be triggered. This class interacts with the “DataStorage” class to fetch historical data, allowing a more thorough analysis of patient trends, and with the “PatientIdentifier” class to retrieve patient details, ensures accurate alert generation. The “AlertManager” class manages the alerts and sends notifications to the medical staff, encapsulating the alert handling process.

In the Data Storage System, the “DataStorage” class is designed to handle the storage, retrieval, and deletion of patient data. It aggregates “PatientData” instances, which represent the data for a patient at a specific point in time, and interacts with the “AlertGenerator” to provide necessary data for alert evaluation. This design controls smooth data management and access control.

The Patient Identification System includes the “PatientIdentifier” class, which matches patient IDs from incoming data with patient records, which are presented by the “PatientRecord” class. This ensures accurate data attribution, which is critical for making informed medical decisions. The “IdentityManager” oversees this process and handles any discrepancies and anomalies, maintaining the integrity of patient records. The composition relationship between “PatientIdentifier” and “PatientRecord” signifies that patient records are integral to the identification process.

The Data Access Layer is designed to handle diverse data input from multiple sources such as TCP/IP, WebSocket, and file uploads. This is achieved through the “DataListener” interface and its implementations (TCPDataListener, WebSocketDataListener, and FileDataListener), which listen for incoming data and pass it to the “DataParser” class for standardization. The “DataSourceAdapter” then processes and stores this standardized data in the “DataStorage” system. The aggregation of “PatientData” within “DataStorage” allows for independent existence of patient data, ensuring flexibility in data handling.

***UML State Diagram***

The state diagram we have created has 4 main states; “Generated”, “Sent”, “Acknowledged”, “Resolved”. The “generated” state represents the initial creation of an alert when the system detects patient data that meets predefined threshold criteria. It is also the entry point. The second state, “sent”, is nested within the “Generated” state to signify that sending is part of the alert generation process. It describes once the alert is generated, it is sent to the designated medical staff's monitoring screens/mobile devices. “Acknowledged” state is nested within the “Sent” state, because acknowledgment is part of the alert handling process after it is sent. It indicates that the medical staff has acknowledged the receipt of the alert, either manually or automatically. “Resolved” state signifies that the alert condition has been addressed and the alert process is concluded. It has two substates; “AutomaticallyResolved”- which represents automatic resolution and “ManuallyResolved” –which represents manual resolution by medical staff after assessing the patient.

The transitions in our state diagram is explained as follows; When an alert is generated, it is immediately sent to the designated medical staff, triggering the transition from the "Generated" state to the "Sent" state. The transition from "Sent" to "Acknowledged" occurs when the medical staff acknowledges the alert -either manually or automatically. Once acknowledged, the alert can transition to the "Resolved" state, which happens when the condition that triggered the alert is either normalized automatically or manually resolved. Within the "Resolved" state, the alert can reach its final resolution through one of two transitions: "AutomaticallyResolved," which is triggered by patient data showing normal readings for a predefined period, or "ManuallyResolved," which is triggered when the medical staff manually confirms that the alert condition no longer exists.

***UML Sequence Diagram***

In the sequence diagram we have created, the process starts with the medical staff (or the system) initiating the monitoring of incoming patient data by interacting with the “AlertGenerator”. The “AlertGenerator” retrieves data from the “DataStorage” to confirm trends or retrieve past records. Once the “DataStorage” returns the requested historical data, the “AlertGenerator” compares the incoming data against predefined criteria. If the criteria are met, the “AlertGenerator” generates a new alert. The alert details are then stored in the “DataStorage”, and the Alert acknowledges that it has been created. Then, the “AlertGenerator” sends a notification to the medical staff containing the details.

After receiving the notification, the medical staff acknowledges the receipt of the alert, and the Alert notifies the “AlertGenerator” that it has been acknowledged. The medical staff then resolves the alert, which can occur in two ways; one being automatic resolution, in which the alert continuously evaluates patient data to determine if the condition has normalized. If that’s the case then the alert is resolved automatically, and the “AlertGenerator” is notified. Second way is manual resolution, in which the medical staff confirms the resolution after intervention, and the alert is resolved manually, and again, the “AlertGenerator” is informed of the resolution.