**Alert Generation System UML Diagram**

The Alert Generation System UML diagram illustrates the architecture and interaction of components within the system responsible for generating alerts based on patient data. The primary components include the AlertGenerator, Alert, and AlertManager. The AlertGenerator is responsible for evaluating patient data and triggering alerts when specific conditions are met. It contains methods such as evaluateData() and triggerAlert(). The Alert component holds information about the alert, including patientId, condition, and timestamp. This class serves as the data structure for storing alert details. The AlertManager handles the distribution and acknowledgment of alerts, with methods like sendAlert() and acknowledgeAlert(). The diagram shows the relationships between these components: the AlertGenerator triggers the creation of an Alert, which is then managed by the AlertManager. This setup ensures a clear separation of responsibilities, allowing for efficient alert handling and management within the healthcare monitoring system.

**Data Storage System UML Diagram**

The Data Storage System UML diagram outlines the structure and interactions of the components involved in storing and retrieving patient data. The main entities are DataStorage, PatientData, and DataRetriever. The DataStorage component provides methods for storing, retrieving, and deleting data (storeData(), retrieveData(), deleteData()). The PatientData class contains attributes such as patientId, metrics, and timestamp, encapsulating the information related to patient health metrics. The DataRetriever is responsible for querying the stored data using the queryData() method. The diagram highlights the relationships among these components: DataStorage contains PatientData and interacts with DataRetriever to facilitate data access. This organization ensures a systematic approach to managing patient data, enabling efficient storage and retrieval operations essential for healthcare applications.

**Patient Identification System UML Diagram**

The Patient Identification System UML diagram presents the components involved in identifying and managing patient records. The key entities are PatientIdentifier, PatientRecord, and IdentityManager. The PatientIdentifier is tasked with matching patient IDs through its matchPatientId() method. The PatientRecord class holds patient details such as patientId and patientDetails, serving as the data repository for patient information. The IdentityManager handles discrepancies and manages patient identities using the handleDiscrepancies() method. The relationships shown in the diagram indicate that PatientIdentifier identifies records within PatientRecord, and the IdentityManager oversees the entire identification process. This structure supports a robust system for accurate patient identification, crucial for maintaining reliable and consistent patient records.

**Data Access Layer UML Diagram**

The Data Access Layer UML diagram details the components facilitating the access and processing of data within the system. The primary components include DataListener, DataParser, and DataSourceAdapter. The DataListener is responsible for listening to incoming data streams through the listen() method. The DataParser processes the received data using the parseData() method, ensuring the data is correctly interpreted and formatted. The DataSourceAdapter handles the final processing and adaptation of the data into a usable form with its processData() method. The diagram shows that the DataListener listens to data streams, which are then parsed by the DataParser, and ultimately processed by the DataSourceAdapter. This layered approach ensures that data access is handled systematically, with each component responsible for a specific part of the process, promoting efficiency and reliability in data management.

**State Diagram for Alert Generation System**

The State Diagram for the Alert Generation System depicts the various states an alert can transition through within the system. The states include [Start], [Generated], [Sent], [Acknowledged], and [Resolved]. The diagram begins with the [Start] state, indicating the initial point before an alert is generated. Once an alert is created, it moves to the [Generated] state. The alert is then sent out, transitioning to the [Sent] state. Upon receipt, the alert is acknowledged, moving it to the [Acknowledged] state. Finally, after necessary actions are taken in response to the alert, it reaches the [Resolved] state. Arrows between the states represent the transitions, showing the progression from one state to the next. This diagram provides a clear visualization of the lifecycle of an alert within the system, ensuring that each stage is clearly defined and managed effectively.