UML Diagrams Descriptions

Alert Generation

Through its connection with “DataRetriever”, the interface “DataStorage” is able to access patient data. The interface “DataStorage” is bidirectionally connected with the interface “PatientData”. The two interfaces collaborate to ensure that there is up-to-date patient information within the Cardiac Health Monitoring System’s database. “DataStorage” uses its connection with “PatientIdentifier” to match the data with the patient. Matching the patient with their data is important so that the on-duty medical personnel know which patient requires urgent attention.

The data stored in “DataStorage” is used in “AlertGenerator”. “AlertGenerator” evaluates this in real-time and assesses the state of its patients. It compares the patient’s vitals (data) against a certain set of predetermined thresholds which indicate that a patient may be in medical distress. Once a patient’s vitals reach the particular thresholds “AlertGenerator” creates an “Alert” object.

The “Alert” is then connected to a “AlertManager” class. “AlertManager” is then connected to “DirectMessage”. “DirectMessage” is how the medical personnel on shift are informed of the medical emergency occurring. The “DirectMessage” uses the “OutputStrategy” interface to build the structure of the alert message. Each alert message contains; the patient experiencing an emergency (patientID), when the alert was triggered (timestamp), the cardiac event (label), and the patient’s vitals (data).

Data Storage System

Within the Data Storage System of the Cardiac Health Monitoring System
Patient Identification System, the two interfaces “DataStorage” and “PatientData” are bidirectionally connected. Their connection is how patient data is stored and edited within the CHMS.

“DataStorage” uses “DataEncryption”. Data that has gone through “DataEncyption” is encrypted, therefore further maintaining security and anonymity for patients.

In order to access data, a security protocol is performed. Users trying to access medical or personal patient information and data must first go through the “UserAuthenticator” class which confirms their username and password. This event is handled by the “AuthorisationManager”. Once a user is authorised by the system, their information goes through the “SecurityCheck” (an arrow from the authentication to the check). Once the security check is performed the data is decrypted through “DecipherData”. Once the data is decrypted, it is then given to “DataRetriever” which feeds all of the data to the “DataStorage” interface.

Building the Data Storage System in this manner is highly important to the functioning of the hospital as a whole – not only limited to the Cardiac Health Monitoring System. Hospital staff have to hold onto extremely sensitive data about each patient, that may even extend beyond the patient themselves (family health history).

Patient Identification System

In the Patient Identification System, the “PatientIdentifier” class has a bidirectional association with the “PatientRecord” class. Firstly, this ensures patient anonymity and privacy. By assigning each patient a “patient ID”, their medical information is not directly tied to their identity. The manner in which they are identified within the hospital (the IDs) are solely maintained within the hospital and cannot be traced otherwise. Secondly, this relationship also ensures all information is connected to the correct patient.

“PatientRecord” also has a bidirectional association with “DataStorage”. The mutual relationship between these two classes allows the employees to use the system to store and edit patient records throughout their shifts. “DataStorage” is able to edit the patient records as it uses its relationship with “PatientIdentifier” in order to locate with record to edit with the new information.

“PatientIdentifier” is also related to “IdentityManager” (an arrow from the former to the latter). This relationship is another way in which the hospital and the Cardiac Health Monitoring System maintain patient privacy. As aforementioned, in the relationship between “PatientRecord” and “PatientIdentifier”, the patient ID’s ensure that no patient is directly identifiable. Therefore, Only authorised personnel are able to identify patients – by using through their unique patient IDs.

Data Access Layer

The interface “PatientData” is connected to the “LegacySystems” class (arrow from former to the latter) because “LegacySystems” refers to an older database that the hospital might still have, so patients’ data is being uploaded to it. “LegacySystems” is connected to “FileDataListener” to allow it to read the data that was uploaded in the “LegacySystems”. “FileDataListener” class is then connected to “DataSourceAdapter” so that it can process this data and do some desired operation with it. There is an arrow coming from “DataParser” to “DataSourceAdapter” ensuring that the raw data can be further processed.

“DataListener” has an arrow connected to “DataParser” since it listens to the parsed data for it to then be received by “TCPDataListener” and generate a signal from “SignalGenerator”. This class is connected to “PatientIdentifier” so that the signal can be made and used to identify the patient by matching its ID and getting the record.

“PatientRecord” class is connected to “SecureWebPortal” so that it can parse the data and be received by “WebSocketListener”. This class is connected back with “DataParser” to parse new data which is expected to be updated and more recent. Finally, the “DataParser” is connected back to “DataSourceAdapter” which continues the diagram. Note that all the arrows are one-way.