Patient Monitoring Service (PMS)

Part 3: Architecture extension

1 Part 3 – Extension of the initial architecture

In this final part of the assignment, you will extend the initial PMS software architecture that you have studied in part 2 of the assignment. This extension is driven by a number of quality attribute scenarios (QASs). For each requirement, the relative importance (combination of business value and architectural impact) for PMS system is given in Table 4. Quality attribute scenarios marked as *Bonus* are not strictly required for completing the assignment.

Importance	QAS	Bonus
High	Sec1, Av2	
Medium	M1, P1	Av1
Low	None	

(a) For three-student project teams.

Importance	QAS	Bonus
High	Sec1, Av2	
Medium	P1	M1, Av1
Low	None	

(b) For two-student project teams.

Importance	QAS	Bonus
High	Sec1, Av2	
Medium		P1, M1, Av1
Low	None	

(c) For single-student project teams.

Table 1: Overview of the importance of the requirements.

1.1 Sec1: Patient identification (H)

The PMS needs to reliably identify the patient in several information flows, to correctly associate the different events (incoming sensor data, emergencies) to the corresponding patient. When receiving new sensor data readings, the system should correctly identify the wearable unit (at the basis of device identifier or the identifiers of individual sensors) and its association to the patient. In addition, the patient gateway should correctly present the patient's identity to the back-end services such that malicious actors cannot feed the system with false sensor data. Similarly, the PMS should be able to correctly identify the relevant patient when sending out notifications to other systems.

- **Source:**
  - Unregistered user or patient gateway
  - Registered patient gateway
- **Stimulus:**
  - A malicious actor presents crafted/falsified sensor data readings (cf. *UC4*: send sensor data) to the patient gateway
  - A malicious actor presents crafted/falsified sensor data readings (cf. *UC4*: send sensor data) to the PMS backend services.
  - A malicious actor falsely sends an emergency notification (cf. *UC3*: send emergency notification) via a patient gateway device that is unregistered or tampered with.
  - New sensor data readings (cf. *UC4*: send sensor data) or an emergency notification (cf. *UC3*: send emergency notification) for a patient is received via a patient gateway registered to another patient.

- **Artifact:** The subsystems that processes incoming requests.
- **Environment:** Normal operation
- **Responses:**
  1. The patient registration procedure ensures that the PMS backend has the necessary information to identify the patient gateway and the different sensors associated with the newly-registered patient's account: the device identifiers of wearable units, sensors, etc. are correctly linked to the patient's identity.
  2. Access to the PMS app is subject to authentication, i.e. a patient has to provide a username and password before further app functionality can be used.
  3. Interaction between the patient gateways and the backend system is subject to authentication, i.e. each patient gateway is authenticated before any further operations such as providing new sensor data readings or retrieving patient information are allowed.
  4. Patient information is subject to access control, i.e. patients can only consult their own information, only the treating physician can access patient data.
  5. The confidentiality and integrity for all information transmitted between the a patient's app and the PMS backend is actively preserved.
  6. Each attempt to interact with the backend system is logged, where each log entry contains
     – the time and date of the request,
     – identification of the requesting patient gateway and/or patient (before actual authentication, these are the IP addresses),
     – identification of the requested operation(s), and
     – whether the request was granted.
- **Response measures:**
  1. It is computationally infeasible to guess correct authentication information. For example, by relying on proper cryptographic keys and sufficiently long and random identifiers.
  2. It is computationally infeasible to bypass the authentication mechanism in the PMS app.
     – User passwords are enforced to be minimally 8 characters and maximally 64 characters long, and should *not* be commonly used passwords.[1]
     – The identity of the patient is verified during the registration process.
  3. It is computationally infeasible to spoof a registered app, e.g., by relying on proper public-key cryptography or secure API tokens.
  4. It is computationally infeasible to bypass the access control mechanism. Each app only receives information that concerns the patient that is associated with the app in question.
  5. It is computationally infeasible to breach the confidentiality of transmitted data or modify transmitted information without such modifications being detected.
  6. Each request is logged before it is processed by the backend. Logs are kept for at least 2 years.

1.2 M1: Include third party lifestyle tracker data in clinical model computations (M)

Lifestyle trackers or 'fitness monitoring devices' (e.g., wristband sensors) are becoming increasingly popular, especially among those users that deliberately want to adopt an active and healthy lifestyle. In its initial development, the PMS directly collects the necessary data through the wearable unit dedicated for the concrete medical purpose of CVD monitoring. However, the data collected by these lifestyle tracker devices represent a wealth of complementary information that can be used for confirmation or contextualization (exercise regime, dietary habits, etc.). In the future, the PMS would like to be able to incorporate such additional data sources. Therefore, it should be easy to extend the PMS with support to interact with the backend services of different relevant lifestyle tracking apps, e.g., the data sharing services offered by Strava or Google Fit, to retrieve additional data related to a patient. This extension will only apply to patients that are also users of such lifestyle tracking devices and are willing to share such information, and such data intake from external sources can only be accomplished with explicit permission and configuration by the patient.

- **Source:** The research department

[1]Candidate passwords should be verified using libraries such as zxcvbn or havibeenpwned.

- **Stimulus:** Wishes to improve the accuracy of the clinical model computation and the resulting predictions by incorporating additional types of data.
- **Artifact:** This modification affects the clinical model computation, the patient dashboard, and communication subsystem(s).
- **Environment:** At design time or at runtime
- **Responses:**
  1. The patient app should be easily extensible with functionality to allow patients to give permission to the PMS to systematically retrieve their data from other systems, e.g., by allowing patients to set an appropriate authentication token.
  2. The PMS should be easily extensible with dedicated communication subsystem(s) to interact with different third-party backends, including support to regularly synchronize the relevant data.
  3. This modification should have no impact on the clinical model computation subsystem beyond potentially using a new model that accepts the new type(s) of data as additional input.
  4. This modification should not affect the existing interface(s) through which data is received, i.e. raw sensor data and information from the HIS.
  5. Storing the new types of data and the new data sets should not require changes to existing storage subsystems, e.g., the storage of (raw) sensor data readings or patient profiles.
  6. Additional attention must be paid to ensure that lack of or limited performance or availability of the third-party services will not affect the performance or availability of the risk estimations performed by our system.
- **Response measures:**
  1. The extension of the app should not take longer than 1 person month to develop, test, and deploy.
  2. Extending PMS with additional communication subsystem(s) to interact with third parties does not take longer than 3 person months to develop, test, and deploy.
  3. Once deployed, including another, additional third party service should not take longer than 2 person weeks to develop, test, and deploy.

1.3 P1: Data exchange with physicians (M)

The PMS exchanges data with several physicians, e.g., a cardiologist performs an on-demand consultation and the physicians registered for a patient need to be notified when their patient's risk level changes. Such exchanges of information should not hinder any medical services provided to the patient and should be conducted in a timely fashion.

- **Source:** Physician
- **Stimulus:**
  - A physician consults a patient's status (cf. *UC6*: consult patient status), e.g., during a consultation with the patient.
  - A cardiologist requests a patient's current data (cf. *UC9*: perform on-demand consultation), e.g., during a consultation with the patient.
  - A cardiologist (re)configures the risk assessment for a patient (cf. *UC7*: configure patient risk assessment).
  - A cardiologist updates the risk level of a patient (cf. *UC8*: update risk level), for example, after reviewing a received notification.
  - The PMS looks up the appropriate recipients and sends a notification about a patient's status (cf. *UC5*: notify).
- **Artifact:** The subsystem(s) responsible to handle requests from physicians and send out notifications.
- **Environment:** Normal mode
- **Responses:**
  1. The system is able to process all requests for patient information in a timely fashion.
  2. Any risk assessment triggered by an on-demand consultation or risk assessment configuration is initiated within a short time frame of the request. In case of remote on-demand consultations, the results of the risk assessment are provided to the physician in a timely fashion.
  3. The system is able to process all changes to a patient's risk level in a timely fashion.
  4. The system is able to send notifications concerning (changes to) a patient's status to all appropriate recipients in a timely fashion, as long as the number of notifications to send stays within a threshold.

5. When the number of notifications to send exceeds the threshold the system prioritizes more urgent notifications.
6. A large number of notifications to be sent or requests arriving to the system should have no impact on the (a) the processing of incoming raw sensor reading data; or (b) ongoing risk estimations.

- **Response measures:**
  1. The system is able to process at minimum 25 requests for patient information per minute (cf. *UC6*: consult patient status)
     - Requests for high-risk patients are processed in 2 seconds.
     - Requests for medium-risk level patients are processed in 5 seconds.
     - Requests for low-risk patients are processed in 10 seconds.
  2. The system is able to process at least 20 requests for an on-demand consultation (cf. *UC9*: perform on-demand consultation) and/or risk assessment configurations per minute (cf. *UC7*: configure patient risk assessment).
     - The risk assessments triggered in an on-demand consultation are prioritized over regular risk estimations triggered by scheduled arrival of new sensor data.
     - Each risk assessment (cf. *UC12*: estimate risk level) triggered in an on-demand consultation is initiated within 3 minutes of receiving the request from the physician, independent of the patient's risk level.
     - In case of an on-demand consultation, the results of the risk assessment are provided to the physician within 1 minute after its completion.
  3. The system is able to process 20 risk-level updates per minute (cf. *UC8*: update risk level).
     - Updates to a patient's EHR are forwarded within 1 minute.
  4. The system should be able to process and send at least 100 notifications per minute (cf. *UC5*: notify).
     - A red notification should be sent to all recipients within 10 seconds.
     - A yellow notification should be sent to all recipients within 30 seconds.
     - A green notification should be sent to all recipients within 1 minute.
  5. When the system must send more than 100 notifications (cf. *UC5*: notify) in a minute, it prioritizes red notifications over yellow and green ones, and yellow notifications over green ones. No red or yellow notifications may be lost.

1.4 Av1: Internal pms database failure (M)

The internal subsystem of the PMS that is responsible for storing the (raw) sensor data readings fails or crashes. Significant and/or frequent downtime of this data store is problematic, as it will impede the execution of risk estimation processes that rely upon historical sensor data and limit our capacity to persist new and incoming sensor data.

- **Source:** Internal
- **Stimulus:** The internal subsystem responsible for storing the sensor data readings fails or crashes.
- **Artifact:** Internal subsystem
- **Environment:** Normal execution
- **Responses:**
  1. This does not affect the availability of other types of persistent data, such as (a) the registered parties that should receive notifications, (b) the pairing of sensors, patient gateway and patient (and their enrollment), (c) the authentication credential data, (d) the risk estimation results, notifications, etc.
  2. A crash does not lead to any loss of already stored (raw) sensor data readings, to data inconsistency or lack of integrity.
  3. Prevention:
     - The storage subsystem should have a minimal guaranteed up-time.
  4. Detection:
     - In case a crash occurs, a PMS system administrator is notified of the problem.
     - The PMS is able to detect this problem and goes into degraded mode.

    * Sensor data readings that arrive during downtime are temporarily stored elsewhere and are added to the storage subsystem when it returns operational.
    * Requests for sensor data for clinical model computations fail gracefully, such that the computations can be rescheduled, or users can appropriately be informed.
  5. Resolution:
     – A PMS system administrator addresses the problem, e.g., by replacing failed hardware or by restarting software, and, if necessary, reverts the subsystem to a previous consistent state, using a backup.
     – The PMS automatically brings the repaired storage subsystem up to date by adding the (raw) sensor data readings that arrived while it was unavailable.

- **Response measures:**
  1. Prevention:
     – For high-risk patients, the raw sensor database should be available 99% of the time (measured per month).
     – For medium-risk patients, the raw sensor database should be available at least 95% of the time (measured per month).
     – For low-risk patients, the raw sensor database should be available at least 90% of the time (measured per month).
  2. Detection:
     – A PMS administrator is notified within 5 minutes of the detection of the failure.
     – Detection of failed hardware or crashed software happens within 1 minute of the failure or crash.
     – In case of high or medium patient risk status,
       * there is a seamless transition to the temporary backup storage (no disruption);
       * no data updates are omitted, regardless of the size or frequency of the updates (patient data readings typically represent detailed and thus high-volume data)
     – In case of a low risk level, at most two of the regular updates may be omitted.
  3. Resolution:
     – It should be possible to restart the storage subsystem within a minute, and
     – roll-back to a previous or backup state happens within 5 minutes.

1.5 Av2: Communication between patient gateways and pms backend (H)

The PMS depends on the interoperation between the patient gateway and the backend services. Key to this interoperation is the communication channel between a patient gateway and the PMS backend, used to relay raw sensor data and emergency notifications. Vice versa, functionality such as the support for on-demand consultation relies upon a reliable communication channel between the PMS backend services and the patient gateway.

Key functionality of PMS can be compromised if (1) a part of the intermediate telecommunication infrastructure fails; (2) a patient gateway of one or more patients fails or crashes; or (3) internal communication subsystem(s) of PMS involved in the aforementioned interactions fail(s) or crash(es). For example, emergency notifications from a patient's gateway can no longer be received and sensor data readings can not be sent from the patient gateway(s) to the PMS for further analysis.

- **Source:** external or internal
- **Stimulus:**
  – the (external) communication channel between a patient gateway and the PMS becomes unavailable, e.g., due to failure or (scheduled) maintenance;
  – a patient gateway becomes unavailable, e.g., its battery may have run empty or it has no network connection; or
  – an internal communication subsystem of the PMS fails or crashes.
- **Artifact:** external communication channel(s), external device(s), internal subsystem(s)
- **Environment:** Normal executions
- **Responses:**
  1. Prevention:

  - PMS has negotiated a Service-Level Agreement (SLA) with the intermediate telecommunication operator that stipulates
    - an availability of at least 99.5% (measured per year) of the communication channel is guaranteed;
    - at least 80% mobile network coverage in the broad region in which the PMS will operate; and
    - an average bandwidth of at least 64Kb/s.
  - The app on the patient's gateway warns the patient in time when the battery of their patient gateway unit is running low.
  - The app on the patient's gateway warns the patient when they have no or limited network connectivity.

2. Detection:
   - The PMS back-end services should be able to autonomously detect any failures based on the lack of sensor data updates from one or more patient gateways.
   - The PMS back-end services should be able to autonomously detect any failures of the relevant internal communication subsystems.
   - The patient gateway app should be able autonomously detect any communication failures and goes into degraded mode, which involves temporarily storing sensor data and notifications for later synchronization, systematic retrying to complete the communications, and use of possible back-up communication channels.
   - The PMS keeps track of how long there was a lack of communication.
3. Resolution:
   - The PMS proactively notifies the relevant stakeholders of the detected problem.
     - In case of a failing communication subsystem, the system administrator must redeploy the failing component(s), or revert it to a previously working state.
     - In case of a failing communication channel, the system administrator must contact the telecommunication operator to resolve this.
     - In case of a failing patient gateway, the patient should be contacted to resolve the issues.
   - In degraded mode, the patient gateway application
     - uses a back-up communication channel for emergency notifications, e.g., SMS; and
     - keeps retrying to deliver sensor data readings in a proper fashion, e.g., using exponential back-off; and
     - the patient gateway is able to store the sensor data readings (and unsent notifications) for the past 24 hours.

- **Response measures:**
  1. Prevention:
     - The app warns the patient when the battery of their patient gateway has dropped to 15% or less charge.
  2. Detection:
     - Detection time depends on the transmission rate configured on the patient gateway (and indirectly on the risk level), but does not exceed this with more than 5 minutes. So, if an expected update does not arrive at expected time T, detection must have taken place before $T + 5$ minutes.
     - The PMS detects any failures of internal communication subsystems within 5 seconds after the failure.
     - A patient gateway detects that the wearable unit has failed within 3 minutes of the failure.
  3. Resolution:
     - Once detected, in 90% of the cases, the notifications sent to the system administrator arrive within 2 minutes if the patient has a high risk level, within 5 minutes if the patient has a medium risk level, and within 10 minutes in case of a low risk level.
     - Redeployment or roll-back of the communication subsystem does not take longer than 5 minutes.
     - The SLA with the telecommunication operator stipulates availability of technical support within 10 minutes and resolution within the hour.

2 Instructions

The goal of this part of the assignment is to create a software architecture for PMS that addresses these requirements. The focus is on accomplishing the quality requirements, but while doing so you will have to complete functional aspects of the system.

In addressing these requirements, you can choose between a more systematic or structured approach (such as the Attribute-Driven Design), or a more freestyle/ad hoc approach. Regardless of the adopted approach, it is advised to plan your efforts well before starting to avoid repeatedly having to backtrack or revise decisions. This can be accomplished by grouping (parts of) requirements that you expect to interact strongly and addressing these together.

You are asked to compile a report that is structured as follows:

1. (optional) Introduction
2. **Architectural decisions for each QAS**: Discuss each QAS in a separate subsection containing the following elements:
   (a) A brief overview of your *principal architectural decisions*, highlighting the applied tactics and patterns (if any).
   (b) A detailed discussion of the *rationale* for your architectural decisions. Explain your solutions in terms of the relevant design elements, e.g. components or nodes, and the appropriate views, e.g. client-server view or deployment view (use the `nameref` command in LaTeX to create references in the report). This discussion must be a self-contained and complete explanation of the solution to the quality at hand.
   (c) A short discussion of any *alternative decisions or approaches* that you considered, e.g. a different tactic or an alternative deployment. For each considered alternative also summarize why you choose not to apply it.

   Note well: The main purpose of this section is to *explain your decisions and rationale* ("we chose to use strategy *X* rather than *Y* to solve Av1 because . . . '), and *not* to transcribe the diagrams (not: "*we added a component X that provides interface I as a child of component Y*").
3. **Team approach and time spent:** Discuss shortly the overall approach you have adopted as a team. Discuss positive and negative lessons learned with regards to team and work organization. Explain how you organized solving the requirements.
   In addition, keep detailed track of the time spent on this part of the assignment, maintaining distinction between time spent on (i) architectural decision-making, (ii) the modeling of the outcome of architectural decisions in VP, and (iii) writing up and completing the report. Aggregate/summarized numbers (in man hours) are sufficient, but if so desired, you can back these numbers up with a more detailed timesheet.
4. (optional) **Other decisions:** Document any other important architectural decisions you made that do not easily fit under the main non-functional requirements discussed in the previous section.
5. **Discussion:** Use this section to (1) discuss the resulting architecture in retrospect. For example, discuss the strong points and the weak points of your solutions (pro/contra). Discuss the *architectural sensitivity and trade-off points* that played a significant role in your design and a rationale for these decisions.
6. **Final architecture** Include the final architecture descriptions (diagrams and element catalog), which can be generated from the final Visual Paradigm project using the SAPlugin.

Formatting rules Use the LaTeX template used in the exported report (also separately available on Toledo). The plugin for Visual Paradigm exports the diagrams and element catalog to LaTeX and you can use the `nameref` command to create references to elements of the catalog (e.g., components, interfaces).

Delivery The deadline to turn in your report for part 3 is **Friday, May 13**. You are expected to hand in both (1) the report (PDF) (including the generated diagrams and element catalog), as well as (2) the Visual Paradigm Project (VPP) source file.

Good luck!
The Software Architecture team.