

REPORT

SYSTEM DESIGN	2
CONTEXT ANALYSIS	2
Pathological condition	2
Aim of the work	2
Literature review	3
Main assumptions	4
UML Modelling and System Description	5
DATABASE DESIGN	27
Modeling and describing the database (via E-R Diagram)	27
Technical description of the tables	29
SYSTEM IMPLEMENTATION	34
Login	34
Account activation	34
Admin interface	35
Doctor interface	37
Patient interface	40
CONCLUSIONS	43
REFERENCES	46

Bergamin Lidia	10862237
Bertoli Luca	10853566
Cioffo Jacopo Nathan	10894143
Lozano Vivas Maria Fernanda	11118110
Mozzi Andrea	11178398

SYSTEM DESIGN

CONTEXT ANALYSIS

Pathological condition

Atrial fibrillation (AF) is the most common cardiac arrhythmia worldwide, affecting approximately 37.6 million people worldwide, with the prevalence predicted to double over the next 35 years impacting on societal health and health economy [1, 2]. AF is associated with an increased risk of all-cause mortality and significant morbidity from stroke and heart failure (HF). Modern wearables represent a practical and effective tool to screen for AF with potential to reduce mortality rates [3]. On the other hand, with an estimate of 26 million people affected worldwide, HF is one of the main reasons for hospitalization in the United States and Europe [4]. Studies using wearable devices to improve HF management have reported mixed results, however several randomized controlled trials have assessed whether noninvasive telemonitoring interventions decrease HF readmission or all cause-mortality [3].

Aim of the work

The aim of this work is to develop a telemonitoring system that enables cardiologists to monitor patients diagnosed with atrial fibrillation or heart failure, both at home and during clinic consultations. The system allows patients to perform a continuous tracking of lifestyle changes and vital signs related to their condition, using their own wearable device to import physiological data, when available. Additionally, it allows patients to connect easily with their treating physician and manage their appointments. On the other hand, the system enables doctors to monitor all their patients remotely, having at their disposal all information reported by patients. Moreover, doctors will be able to contact patients in a fast and direct manner in case of abnormalities detected in the data, ensuring faster medical attention in case of clinical emergencies. Finally, the person with administrator access to the platform can check performance statistics and track the number of patients and doctors using this monitoring system.

Literature review

Source info	Main findings
N. A. Cruz-Ramos, G. Alor-Hernández, L. O. Colombo-Mendoza, J. L. Sánchez-Cervantes, L. Rodríguez-Mazahua, and L. R. Guarneros-Nolasco, "mHealth Apps for Self-Management of Cardiovascular Diseases: A Scoping Review," <i>Healthcare</i>, vol. 10, no. 2, p. 322, Feb. 2022, doi: 10.3390/healthcare10020322.	• The majority of the mHealth apps for cardiovascular diseases (CVD) include these functionalities: medical recommendations, reminding notifications or alerts for medical appointments. Additionally monitor such as physical activity, step count, weight control, blood pressure, heart rate, pulse wave, and body temperature. • It is essential to consider the use of wearable devices for monitoring biomedical variables automatically. One of the main challenges in these apps include the interoperability with other systems.
Z. Yang, L. Xu, Y. Gao, C. Zhang, and A. Wang, "Tailored persons for self-management in home-based cardiac rehabilitation for patients with coronary heart disease: A qualitative study," <i>Int J Nurs Stud</i>, vol. 163, p. 105000, Mar. 2025, doi: 10.1016/j.ijnurstu.2025.105000.	• Different patient's groups can be identified based on their motivation for self-management, adherence to treatment and willingness to seek external support. • Based on the patient group, their needs in a home-based monitoring system can vary. Within the needs that can be highlighted are: - o Professional feedback via digital platforms, phone or visits - o Real-time health monitoring devices to track progress - o Personalized goal-oriented symptom management plans with check-ins
A. Hughes, M. M. H. Shandhi, H. Master, J. Dunn, and E. Brittain, "Wearable Devices in Cardiovascular Medicine," Mar. 03, 2023, <i>Lippincott Williams and Wilkins</i>. doi: 10.1161/CIRCRESAHA.122.322389.	• There are several factors limiting the clinical use of wearables including concerns about data privacy, device accuracy, regulatory and reimbursement policies, and lack of clinical staff to monitor, interpret, and respond to the vast amounts of data generated by devices. • There is a need for user-friendly interfaces to engage both the providers and the patients in the routine use of wearables in clinical practice. • For wearable devices to reach its full potential, it will be essential that strategies allow clinicians to easily identify actionable data while ignoring

	the noise that the abundant wearable data will inevitably generate.
N. Alugubelli, H. Abuissa, and A. Roka, "Wearable Devices for Remote Monitoring of Heart Rate and Heart Rate Variability—What We Know and What Is Coming," <i>Sensors</i> , vol. 22, no. 22, p. 8903, Nov. 2022, doi: 10.3390/s22228903.	• Heart rate variability (HRV) may predict adverse cardiovascular events, especially after exercise in healthy individuals, as well as in patients with HF. • Recently, AF has been proposed as a good candidate to be monitored using HRV measures.
A. B. Sheikh <i>et al.</i> , "Blood Pressure Variability in Clinical Practice: Past, Present and the Future," <i>J Am Heart Assoc</i> , vol. 12, no. 9, May 2023, doi: 10.1161/JAHA.122.029297.	• Blood pressure variability (BPV) is a significant risk factor for HF, AF, myocardial infarction and other cardiac conditions. • Continuous BP monitoring and controlling have a potential diagnostic and therapeutic value considering its implication on several cardiovascular conditions.

Main assumptions

Listed below are the main assumptions considered during the development of the system.

1. **Clinical management:** Each patient has only one primary cardiologist for monitoring purposes, although real clinical scenarios may involve multiple professionals. Additionally, the system tracks only one disease determined by the physician as "Main Disease" even when comorbidities can exist.
2. **Wearable device data input:** System was designed to be compatible with health data measured by Apple Watch. Patients will use their own wearable device, if available and supported by the platform, to upload physiological parameters. If a patient does not have a compatible smartwatch, parameters measured with other medical devices can be entered manually. In the current design, only one active wearable device per patient is considered.
3. **Role-based system management:** The role "administrator" corresponds to a hospital worker who has access to the platform and is responsible for the registration of all doctors that should have access to it. The initial registration of the different medical centers with their administrators, will be carried out by system owners according to hospital requests. After that initial registrations, administrations will be in charge of creating doctor accounts, and similarly doctors are responsible for creating patients accounts.

UML Modelling and System Description

Textual descriptions

1. Users

Name:	Log-in
Description:	Log-in procedure for all types of users of the system.
Actors:	User (patient, doctor or administrator) and system
Standard process:	1. User inserts type of user, username and password 2. System verifies login info 3. User gains access to the platform
Alternative processes	2.a. Wrong type of user/username/password: Recover password 2.b First log in: Insert personal info and a new password, status of account change to active

Name:	Account activation
Description:	After an account has been created, the user should activate it, change password and update personal data.
Precondition:	User responsible for creating the account has already been created with an username and the minimal data required, and the new user received by email the username and a temporary password.
Actors:	User (patient, doctor or administrator) and system
Standard process:	1. User must log-in using the username and the temporary password 2. User must choose a new password different from the temporary one 3. User complete personal info (name, surname, age, sex, tax code, email, phone number...) 4. System verify data registered and update new password and personal information for the user
Alternative processes	2.a. The password chosen is not valid (same as the temporary one, not strong enough, empty) 3.a. The information added is not valid (e.g. incorrect data type)

Name:	Manage personal info
Description:	In the patient or doctor interface, users can access their personal information and update if needed.
Precondition:	User account has been already activated and personal information has been completed during the registration process.
Actors:	User (patient and doctor) and system
Standard process:	1. User opens the related window 2. User view actual information 3. User selects what information wants to modify and insert the new data 4. User confirms the change 5. System verify new data registered and update new personal information for the user
Alternative processes	3.a. The information added is not valid (e.g. incorrect data type)

2. Patient

Name:	Manage lifestyle data
Description:	In the patient interface, patients can see lifestyle data registered and update it if needed.
Precondition:	Patient have an active account.
Actors:	Patient and system
Standard process:	1. Patient opens the “Lifestyle data” panel 2. Patient insert info about diet, physical activity, medication, or sleep quality 3. Patient confirm changes 4. System update data and register last update date
Alternative processes	

Name:	Manage physiological parameters
Description:	In the patient interface, patients can see current physiological parameters registered and update them if needed. Update can be done manually or by importing information from the wearable device.
Precondition:	Patient have an active account and wearable device (if applicable) is compatible with the system
Actors:	Patient and system
Standard process:	1. Patient opens the “Physiological parameters” panel 2. Patient insert manually new values or import data from wearable device 3. Patient confirm changes if data was inserted manually 4. System verifies new information registered and add it in the database 5. Patient refresh the system to observe new values and updated graphics
Alternative processes	2.a. The information added is not valid (e.g. out of expected values) 2.b. Wearable device is not compatible with the system

Name:	Create a note
Description:	In the patient interface, patients can create a note to notify their physician about their current health status. Patients can add a priority if they consider that the doctor should review this data urgently.
Precondition:	Patient have an active account.
Actors:	Patient and system
Standard process:	1. Patient opens the “Note” panel 2. Patient insert note and corresponding priority (high, medium, or low) 3. Patient confirm changes 4. System update data and register last update date
Alternative processes	

Name:	Manage appointments
Description:	In the patient interface, patients can view the information of previous and future appointments with their physicians and if there is any new proposed date for the next appointment can accept it or deny it.
Precondition:	Patient have an active account and the doctor has sent at least one appointment proposal.
Actors:	Patient, doctor and system
Standard process:	1. Patient opens the “Appointments” panel 2. Patient view information of all previous appointments (if there is any) 3. Patient review if there are pending proposals and accept it 4. System update the status of the appointment and if include it in doctor’s calendar
Alternative processes	3.a. Appointment is rejected by patient

3. Doctor

Name:	Monitor patients
Description:	In the doctor interface, doctors can check an overview of the status of all active patients under treatment, also they can select one patient and watch more detailed information for that specific patient.
Precondition:	Doctor have an active account and at least one active patient registered in the system.
Actors:	Doctor, patient and system
Standard process:	1. Doctor opens the “Patients overview” panel 3. System shows color alerts (yellow, red) in case there is any abnormality in any parameter reported by patient 3. Doctor selects one specific patient to see detailed information 4. System displays a new panel with the information of the patient selected (Parameters graphics, notes, next appointments) 5. Doctor review patient information and returns to main screen
Alternative processes	1.a. Filters can be applied to check only critical patients, one in specific, one that has not entered information in a long time 5.a. Doctor can also select the option of sending a mail to patient suggesting to attend to medical services if notice any abnormal parameters or alarm in patient notes

Name:	Manage appointments
Description:	In the doctor interface, doctors can see the calendar with all their past and future appointments. They can also create a new appointment for a patient, modify or cancel one already created.
Precondition:	Doctor have an active account and at least one active patient registered in the system.
Actors:	Doctor, patient and system
Standard process:	1. Doctor opens the “Appointments” panel 2. Doctor select option to create or modify appointment 3. The doctor selects date, time and patient to whom wants to assign a new appointment 4. System create or modify the appointment with a pending status 5. System shows new appointment proposal to patient
Alternative processes	3.a. Schedule conflict, if the doctor selects a non-available or overlapping time, the system send a error message

Name:	Create new patient account
Description:	In the doctor interface, doctors have the option to create the account of a new patient.
Precondition:	Doctor have an active account.
Actors:	Doctor, patient and system
Standard process:	1. Doctor opens the “Create new patient account” panel 2. Doctor inputs username, main disease and temporary password 3. System verifies that all the forms have been filled in and that there is no subject registered with this username 4. System creates the patient account 5. System sends by email to the new patient the information for accessing the app including the temporary credentials
Alternative processes	3.a. All forms haven’t been filled in or username already registered

4. Administrator

Name:	Create new doctor account
Description:	In the admin interface, administrators have the option to create new accounts for doctors.
Precondition:	Administrator have an active account.
Actors:	Administrator, doctor and system
Standard process:	1. Administrator opens the “Create doctor account” panel 2. Administrator inputs username and temporary password 3. System verifies that all the forms have been filled in and that there is no subject registered with this username 4. System creates the doctor account 5. System sends by email to the new doctor the information for accessing the app including the temporary credentials
Alternative processes	3.a. All forms haven’t been filled in or username already registered

Name:	Modify or delete existing account
Description:	In the admin interface, administrators can view all accounts of patients and doctors with their current status. They can also block or delete accounts.
Precondition:	Administrator have an active account and there is at least one doctor or patient active account that can be modified.
Actors:	Administrator and system
Standard process:	1. Administrator opens the “Manage patients” or “Manage doctors” panel 2. Administrator chooses which account they want to manage 3. Administrator block, reactivate or delete the account
Alternative processes	

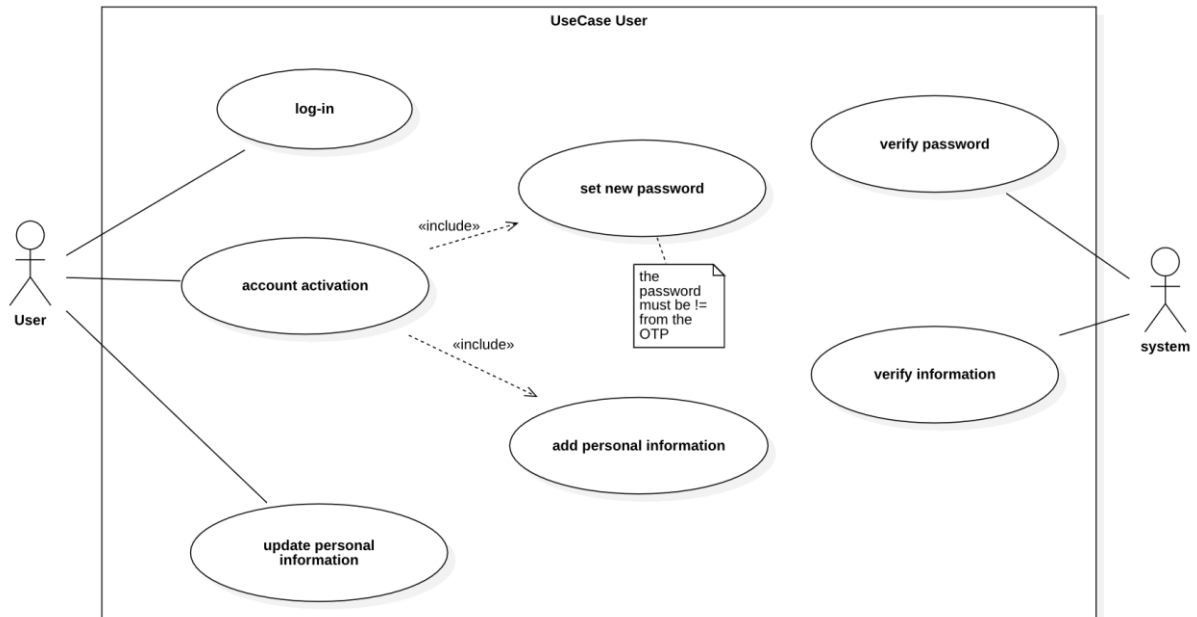
Name:	View functional statistics
Description:	In the admin interface, administrators have a panel to observe statistics of function of the system.
Precondition:	Administrator have an active account.
Actors:	Administrator and system
Standard process:	1. Administrator opens the “Dashboard” panel 2. Administrator gains access to the statistics of all parameters

Alternative processes	
------------------------------	--

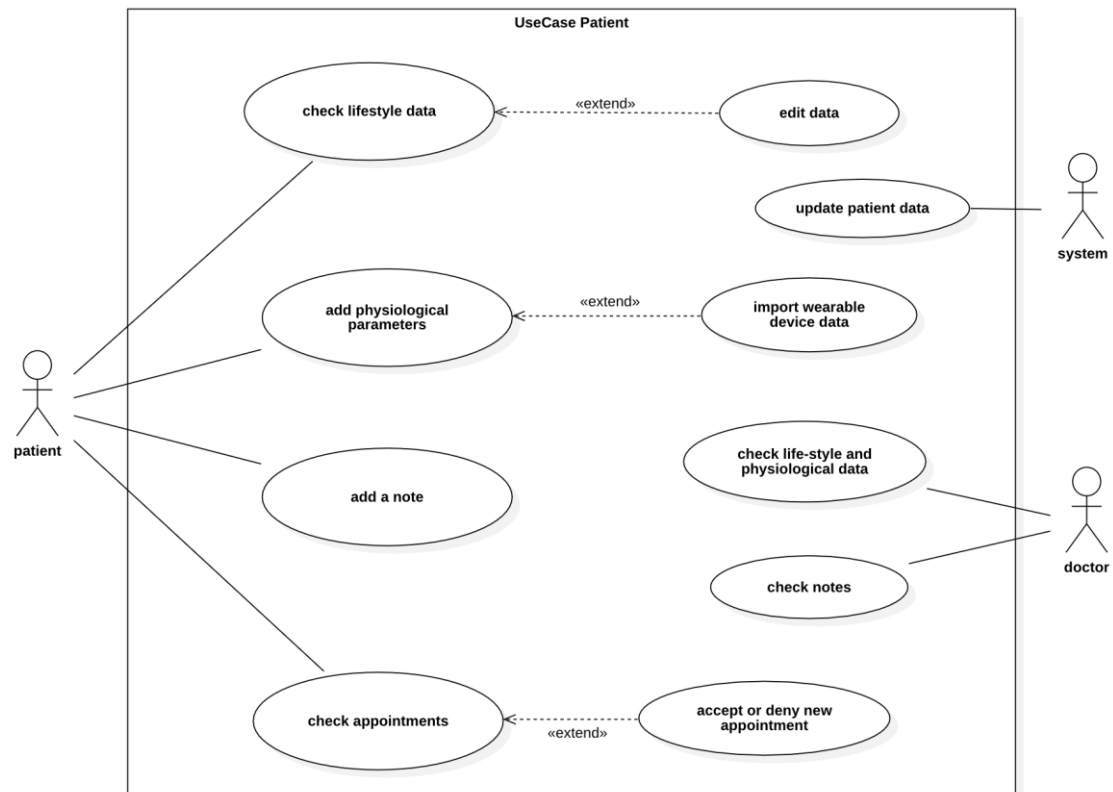
Name:	Add new disease term and parameter normal ranges
Description:	In the admin interface, administrators have the option to add new diseases terms with the normal ranges that the hospital considers to monitor the physiological parameters for that specific condition.
Precondition:	Administrator have an active account.
Actors:	Administrator and system
Standard process:	1. Administrator opens the “New disease” panel 2. Administrator check all current diseases available 3. Administrator enter new disease name and normal ranges for the physiological parameters of interest 4. System include new disease in the database
Alternative processes	2.a. Administrator can modify normal ranges from current diseases 3.a. The information added is not valid or main disease already exist

Use case diagrams

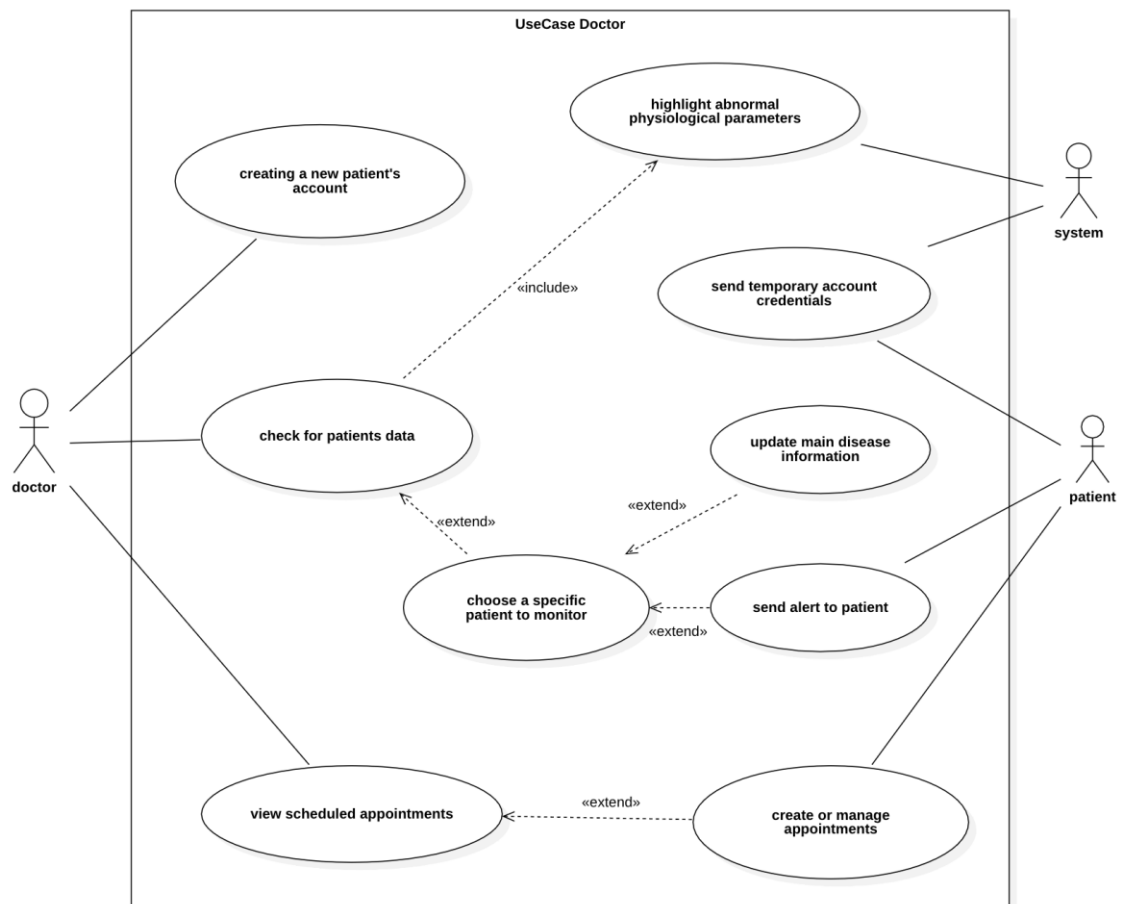
1. User Use case diagram



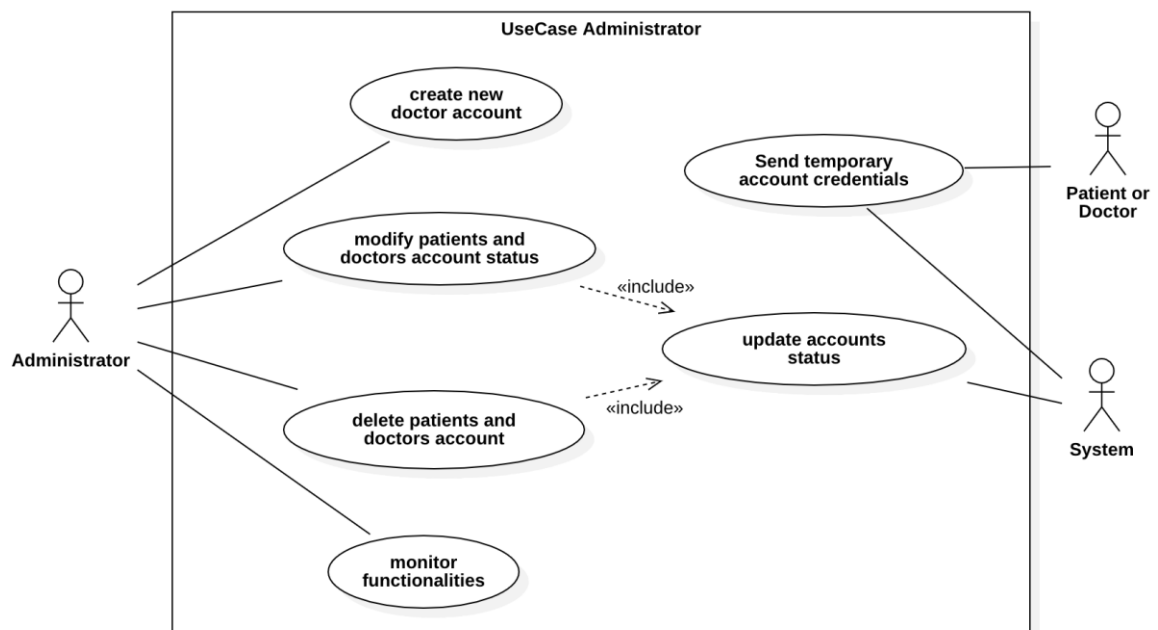
2. Patient Use case diagram



3. Doctor Use case diagram



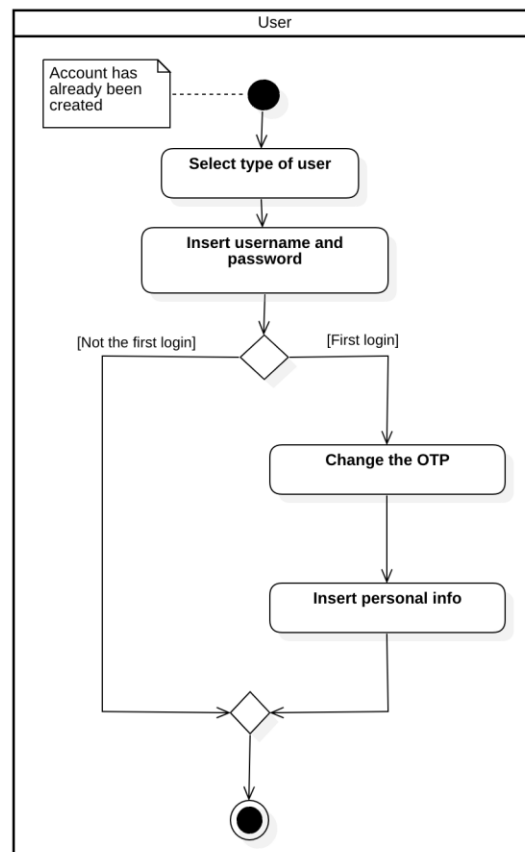
4. Administrator Use case diagram



Activity diagrams

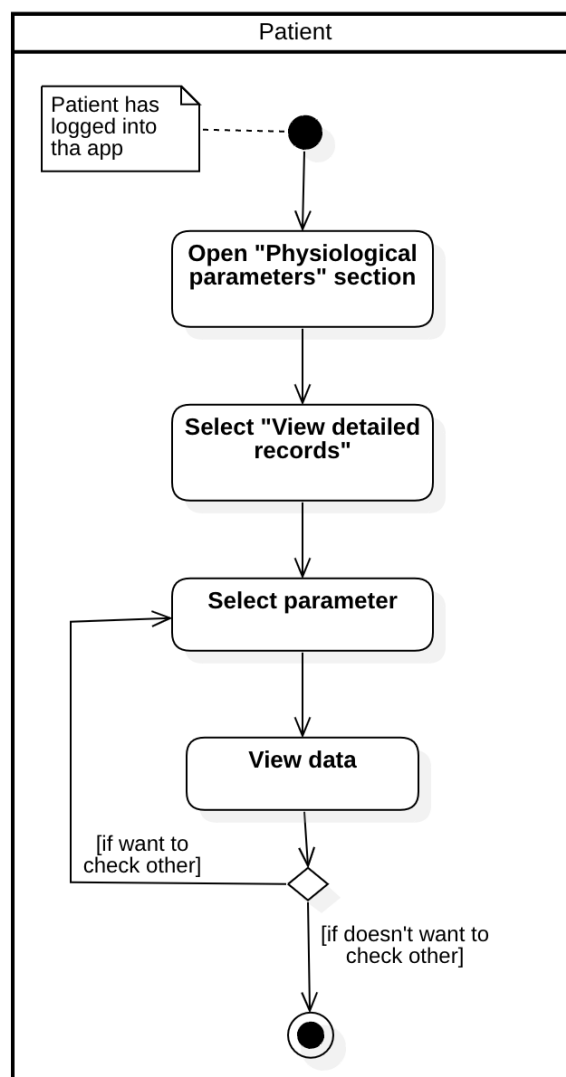
1. Log-in

Name:	Log-in
Description:	Log-in procedure for all types of users of the system.
Precondition:	Account already created by the corresponding actor.
Actors:	User (patient, doctor or administrator) and system
Standard process:	1. User inserts type of user, username and password 2. System verifies login info 3. User gains access to the platform
Alternative processes	2.a. Wrong type of user/username/password 2.b First log in: Insert personal info and a new password, status of account change to active - User must choose a new password different from the temporary one - User complete personal info (name, surname, age, sex, tax code, email, phone number...)



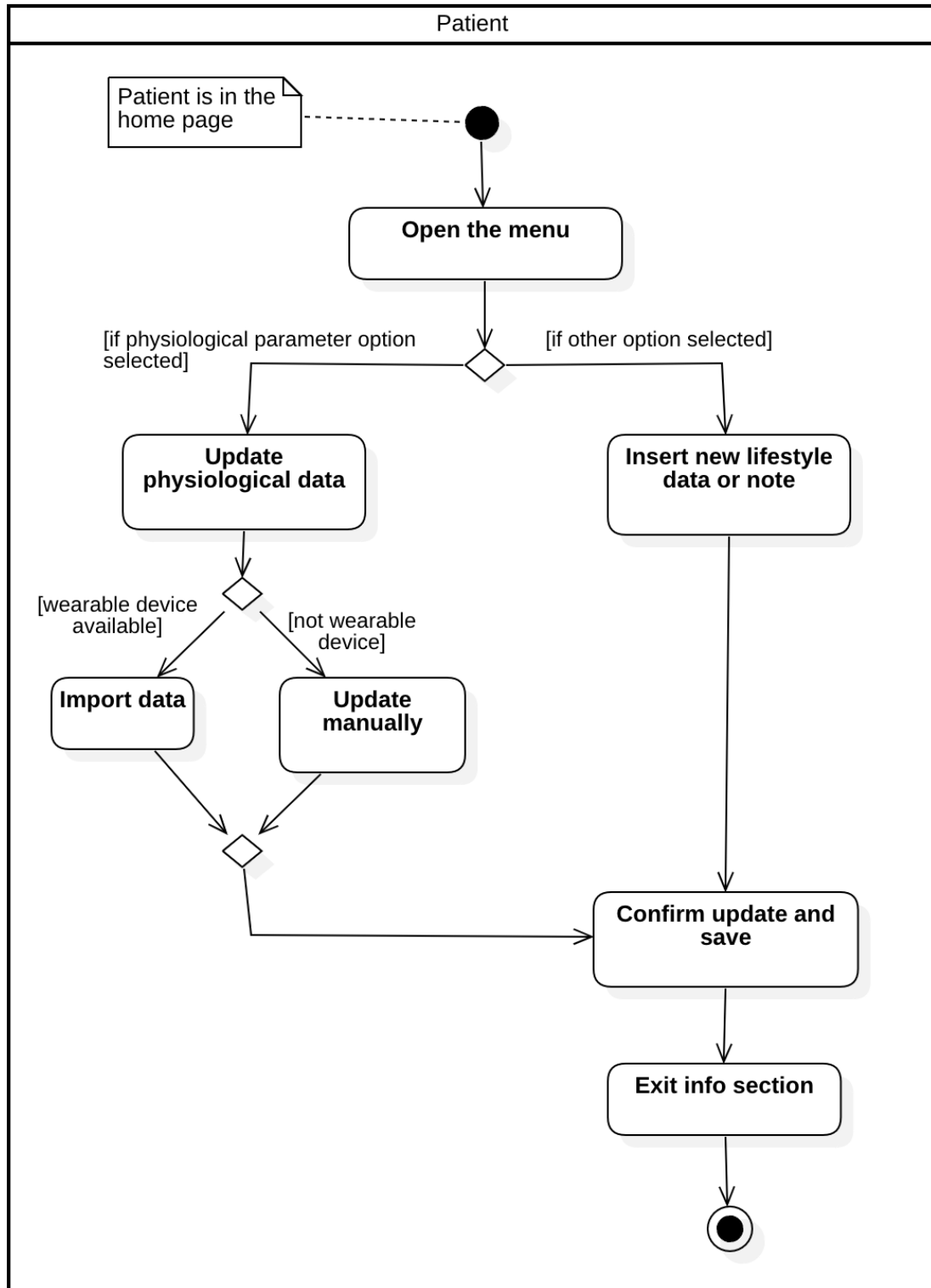
2. Check physiological data trends

Name:	Check physiological data trends
Description:	In the patient interface, patients can see all the physiological parameters data registered over time.
Precondition:	Patient have an active account and at least one record in the physiological data section.
Actors:	Patient and system
Standard process:	1. Patient opens the “Physiological parameters” panel 2. Patient select “View detailed records” option 3. Patient select an specific parameter that want to review 4. System shows graphics and average values for that parameter 5. Patient close the panel
Alternative processes	5.a. Patient can return and check the detailed data of a different physiological parameter



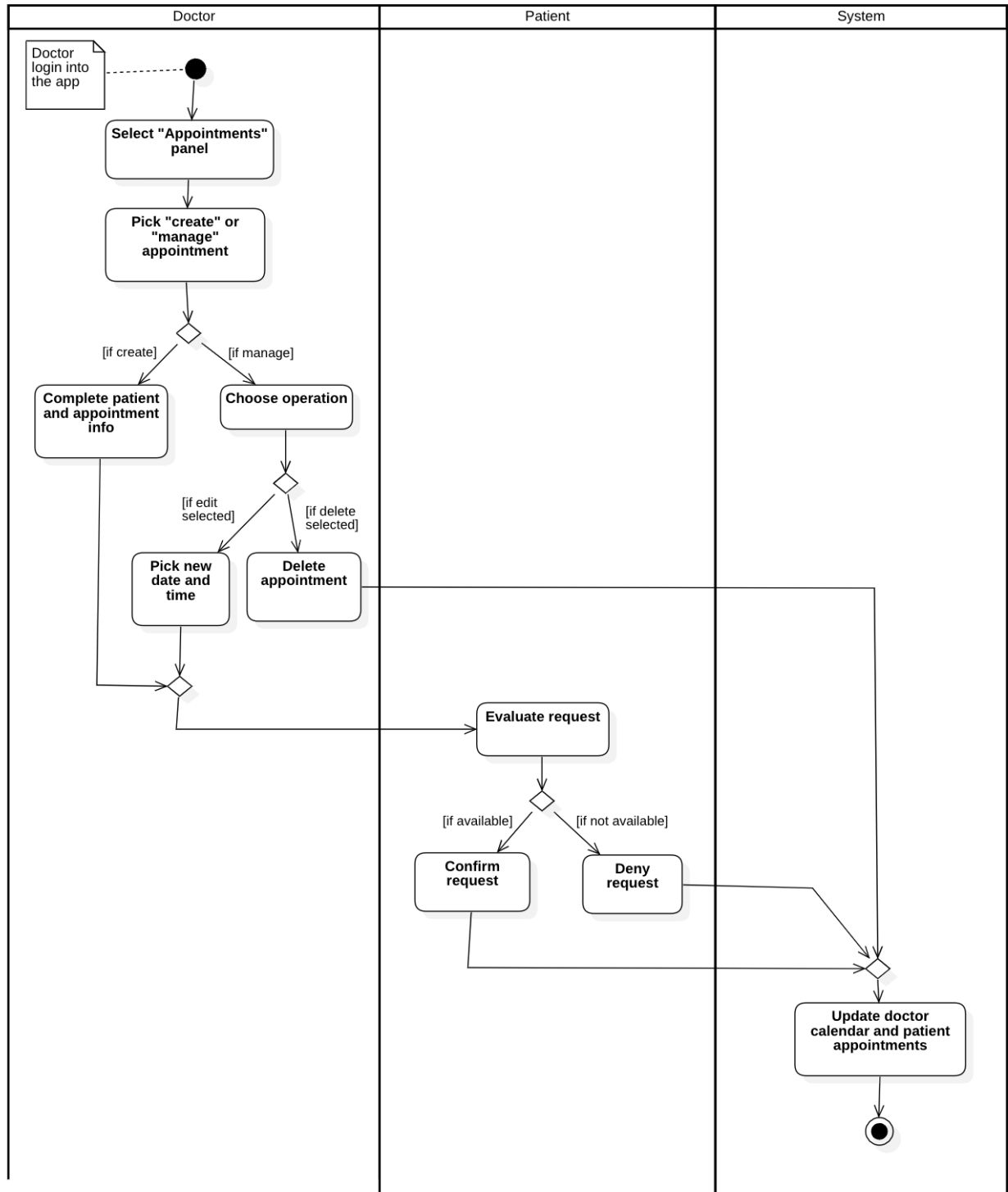
3. Insert and update patient data

Name:	Insert and update patient data
Description:	In the patient interface, patients can update physiological parameters registered manually or import information of their wearable device. Additionally information related with lifestyle and notes to doctors can also be updated.
Precondition:	Patient have an active account and wearable devices (if applicable) are compatible with the system.
Actors:	Patient and system
Standard process:	1. Patient opens the “Physiological parameters” panel 2. Patient insert manually new values or import data from wearable device 3. Patient confirm changes if data was inserted manually 4. System verifies new information registered and add it in the database 5. Patient refresh the system to observe new values and updated graphics
Alternative processes	1.a. Patient opens “Lifestyle data” panel - Patient insert info about diet, physical activity, medication, sleep quality - Patient confirm changes - System update data and register last update date 1.b. Patient opens “Notes” panel - Patient insert note and corresponding priority (high, medium, or low) - Patient confirm changes - System update data and register last update date 2.a. The information added is not valid (e.g. values out of expected ranges) 2.b. Wearable device is not compatible with the system



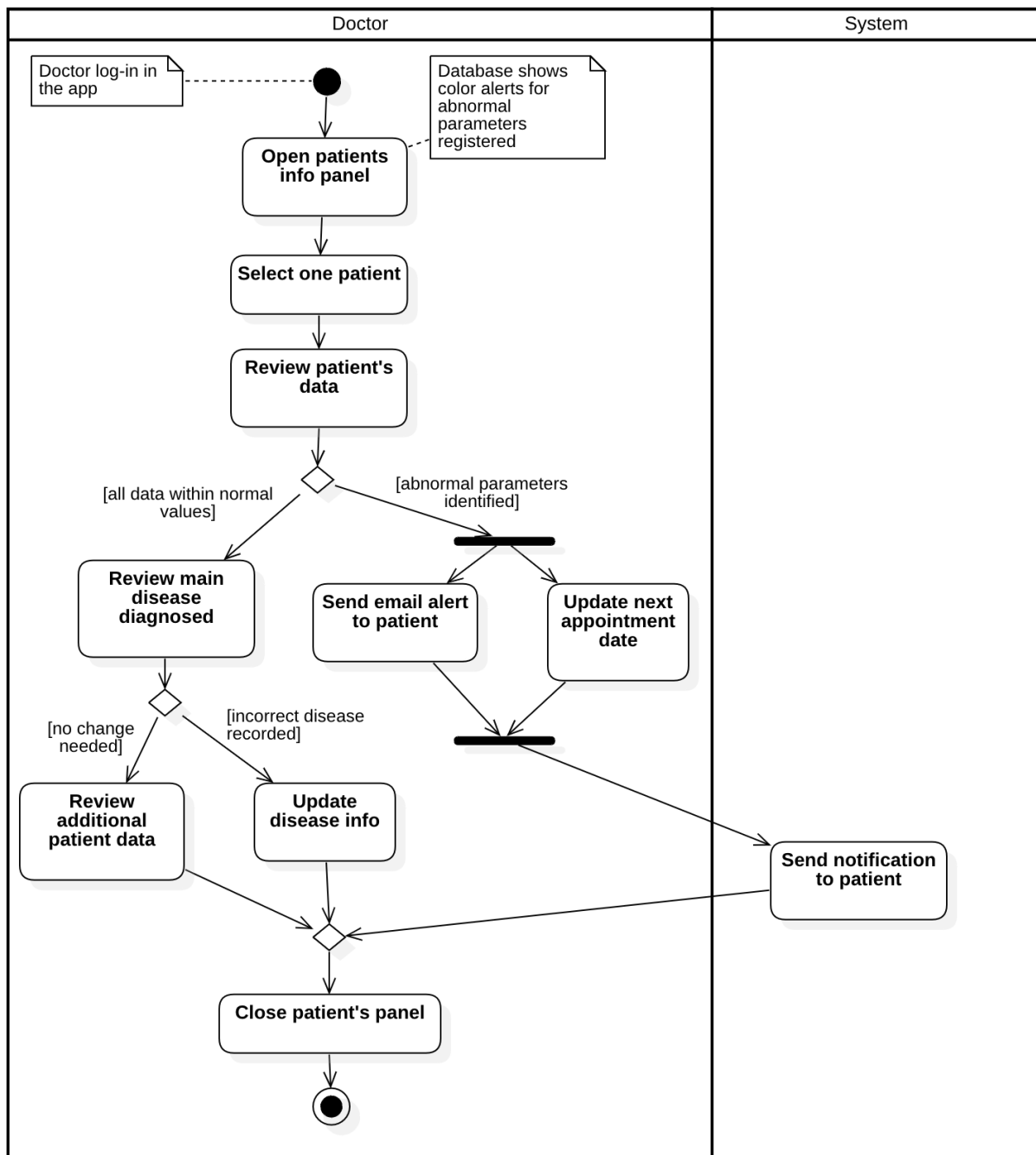
4. Manage appointments

Name:	Manage appointments
Description:	In the doctor interface, doctors can see the calendar with all their past and future appointments. They can also create a new appointment for a patient, modify or cancel one already created. Once an appointment is created or modified, patients have the option to accept or deny the proposed date.
Precondition:	Doctor have an active account and at least one active patient registered in the system
Actors:	Doctor, patient and system
Standard process:	1. Doctor opens the “Appointments” panel 2. Doctor select option to create or modify appointment 3. The doctor selects date, time and patient to whom wants to assign a new appointment 4. System create or modify the appointment with a pending status 5. System shows new appointment proposal to patient 6. Patient opens the “Appointments” panel 7. Patient review if there are pending proposals and accept it 8. System update the status of the appointment and if include it in doctor’s calendar
Alternative processes	3.a. Schedule conflict, if the doctor selects a non-available or overlapping time, the system send a error message 7.a. Appointment is rejected by patient



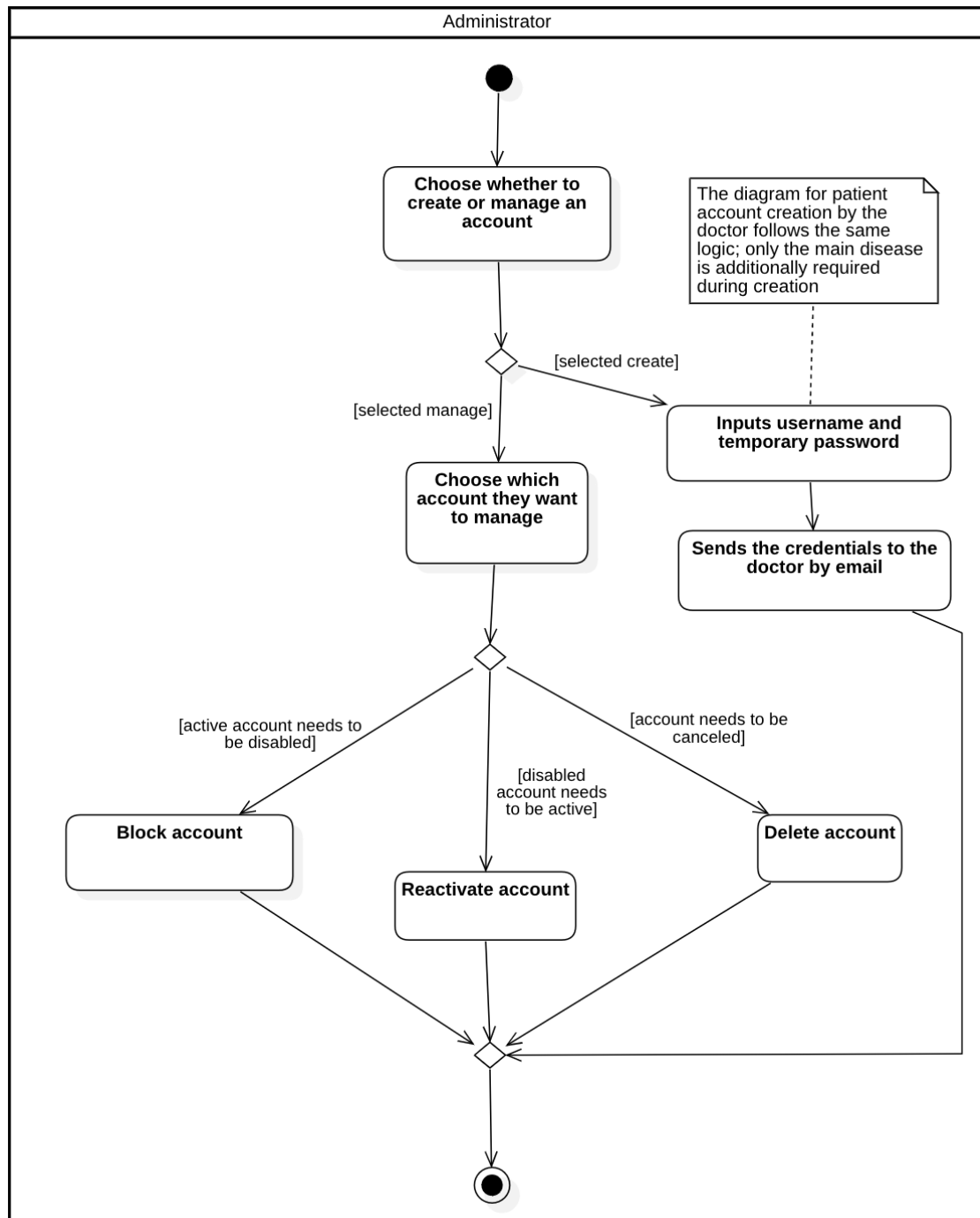
5. Monitor patients

Name:	Monitor patients
Description:	In the doctor interface, doctors can check an overview of the status of all active patients under treatment, also they can select one patient and review more detailed information for that specific patient.
Precondition:	Doctor have an active account and at least one active patient registered in the system
Actors:	Doctor, patient and system
Standard process:	1. Doctor opens the “Patients overview” panel 3. System shows color alerts (yellow, red) in case there is any abnormality in any parameter reported by patient 3. Doctor selects one specific patient to see detailed information 4. System displays a new panel with the information of the patient selected (Parameters graphics, notes, next appointments) 5. Doctor review patient information and returns to main screen
Alternative processes	1.a. Filters can be applied to check only critical patients, one in specific, one that has not entered information in a long time 5.a. Doctor can also select the option of sending a mail to patient suggesting to attend to medical services if notice any abnormal parameters or alarm in patient notes 5.b. Doctor can also prioritize next appointment of patient following the appointment management pathway



6. Manage user accounts

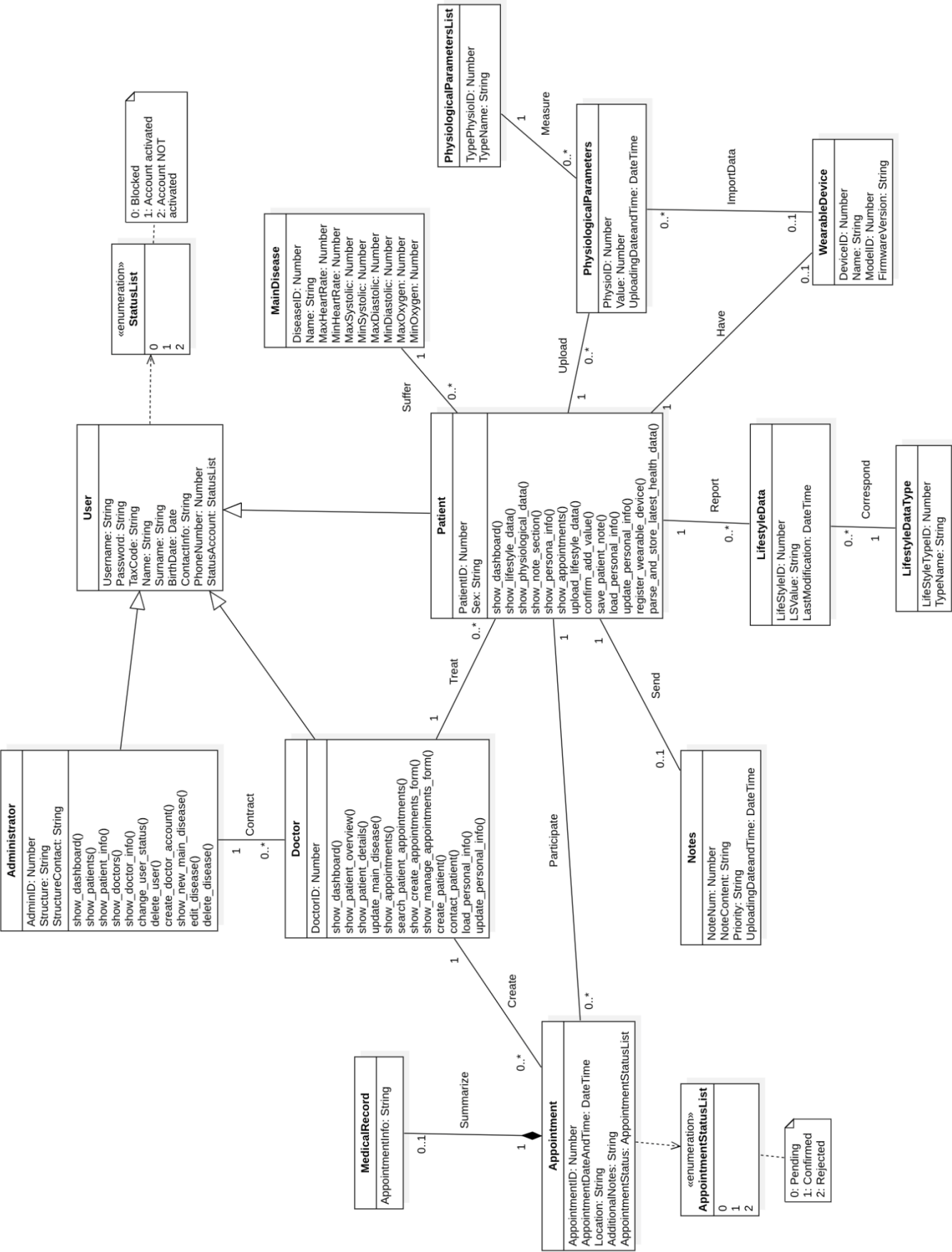
Name:	Manage users account
Description:	In the admin interface, administrators have the option to create new accounts for doctors, modify the status or delete doctors and patients accounts. Similarly, doctors have a panel to create a new patient account.
Precondition:	Administrator or doctor who will create the new account have an active account
Actors:	Administrator, doctor, patient and system
Standard process:	1. Administrator/Doctor opens the “Create account” panel 2. Administrator/Doctor inputs username and temporary password (in the case of new patient also inputs main disease information) 3. System verifies that all the forms have been filled in and that there is no subject registered with this username 4. System creates the new account 5. System sends by email the information for accessing the app including the temporary credentials to the new user
Alternative processes	1.a. Administrator opens the “Manage patients” or “Manage doctors” panel - Administrator chooses which account they want to manage - Administrator block, reactivate or delete the account 3.a. All forms haven’t been filled in or username already registered



Class diagrams

Below is the class diagram of the Beat the Disease system, a platform designed to monitor patients with atrial fibrillation and heart failure. The system integrates functions for three main types of users (patients, doctors and administrators), which share all the information contained inside the **User** class. At the center of the design is the **Patient** class, which is linked with all the classes that represent the different types of data that will be monitored in the system; including **Note**, **LifestyleData** and **PhysiologicalParameters** classes. Notes class allows patients to write one current status note with an associated priority level, which is overwritten every time it is updated. Lifestyle data captures habits such as diet, physical activity, medicine intake and sleep quality; and similarly, the latest update overwrites the previous values. In contrast, physiological parameters include variables such as heart rate, systolic and diastolic blood pressure, oxygen saturation and weight. These parameters can be updated manually or can be imported directly using a wearable device compatible with the system, which is represented by the **WearableDevice** class.

The relationship between **Patient** and **Doctor** models a simplified relation in which each patient is monitored only by one physician. Doctors can create and manage the appointments, detailed in the **Appointment** class, create a **MedicalRecord** entity to summarize the outcome of the appointment and check patients' variables and notes remotely. Finally, the **Administrator** class has a primary relationship with the **Doctor** class. In an initial instance hospital administrators are responsible for creating doctor accounts, managing users accounts status and updating the catalogue of main diseases normal ranges, that are used to monitor physiological variables in each patient.

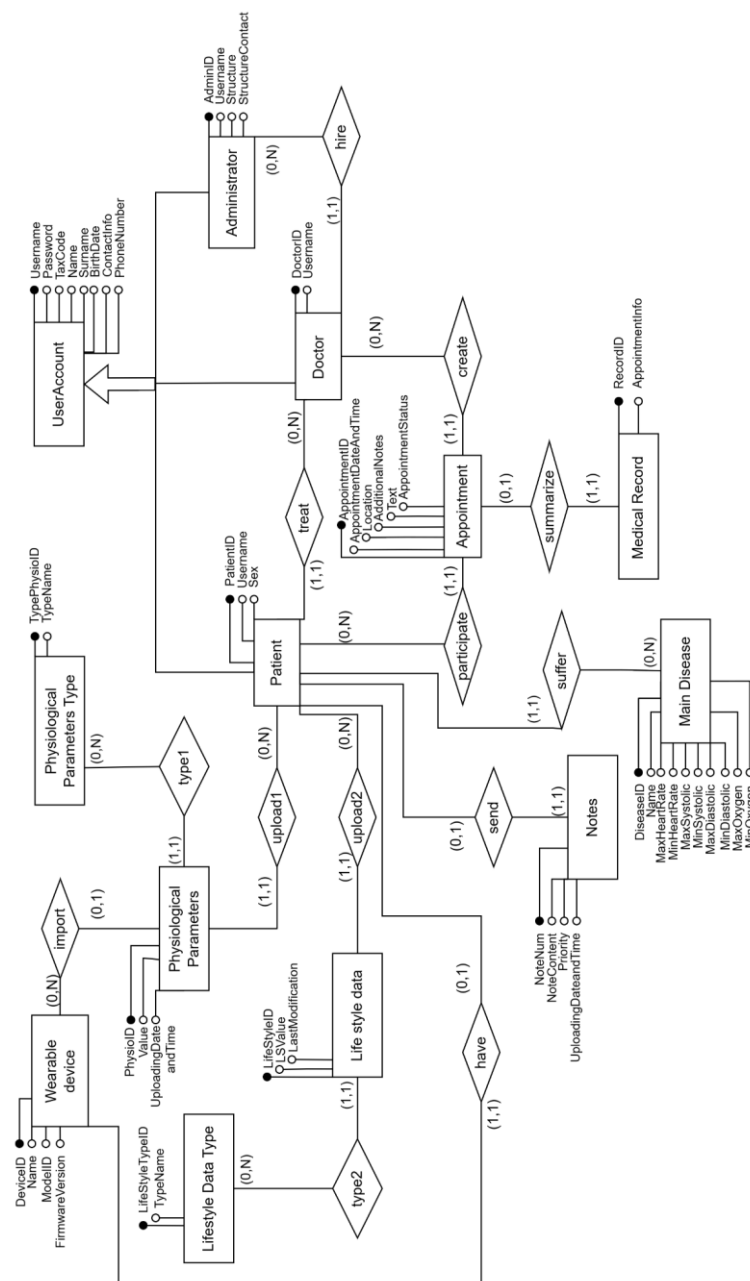


DATABASE DESIGN

The design of our database was based on the previously developed diagrams, especially the class diagram. Initially, we started with the creation of the ER diagram. It must be said that for simplicity we preferred not to use composite attributes as primary keys, but to always rely on an ID, also to make the queries in the application simpler. Natural keys such as Username or TaxCode may change over time or be subject to external constraints, while surrogate keys provide a stable and efficient way to uniquely identify records. Additionally, avoiding composite keys reduces complexity in joins and foreign key references.

Modeling and describing the database (via E-R Diagram)

Below is the ER diagram:



The first thing that immediately catches the eye when looking at the ER diagram is the user management with a generalization hierarchy where **UserAccount** is the superclass. We modeled users using a generalization hierarchy to separate common attributes from role-specific ones. This approach improves modularity and avoids redundancy. We adopted a table-per-subclass strategy, which allows each subtype (*Administrator*, *Doctor*, *Patient*) to store only its specific attributes while maintaining a clear link to the shared UserAccount entity.

In *UserAccount*, the information common to all 3 types of users has been saved: *Username*, *Password*, *TaxCode*, *Name*, *Surname*, *BirthDate*, *ContactInfo* (which is the mail address), *PhoneNumber*. The subclasses were therefore the entities related to the users:

- **Administrator** with attributes *AdminID*, *Username*, *Structure* and *StructureContact*
- **Doctor** with attributes *DoctorID* and *Username*
- **Patient** with attributes *PatientID*, *Username* and *Sex*

Each of these presents unique attributes. These 3 entities are linked to each other through the **treat** and **hire** relationships, both are one-to-many relationships:

- We modeled the relationship between Administrator and Doctor as one-to-many because each doctor is assumed to belong to a single healthcare structure managed by one administrator. This reflects a simplified organizational model where administrative responsibilities are centralized.
- We chose a one-to-many relationship between Doctor and Patient to represent a simplified scenario in which each patient is assigned to a primary doctor responsible for their care. While in real-world systems patients may interact with multiple doctors, this design focuses on a primary-care model to reduce complexity.

The doctor and the patient share an intermediate entity which is the **Appointment**, which refers to the visits carried out by doctors and patients. As can be seen from the graph, it is linked to them through two relationships, **participate** and **create**, with a one-to-many cardinality. This means that for each appointment there is necessarily only one patient and one doctor, and obviously each patient and each doctor can have multiple visits. Its attributes are *AppointmentID* (primary key), *AppointmentDateAndTime*, *Location*, *AdditionalNotes*, *Text* and *AppointmentStatus*.

Appointment is linked to the **Medical Record** (primary key is *RecordID*) entity, which is the one that records the information related to the visit (in the *AppointmentInfo* attribute). This relationship **summarize** was modeled as an optional one-to-one association because a medical record is created only if the appointment actually takes place. This avoids storing unnecessary or incomplete records for cancelled or missed appointments.

At this point, the graph branches out from the *Patient* side. We assumed that each patient is associated with a single main disease to simplify the monitoring logic of physiological parameters. This design focuses on tracking the primary condition of the patient, which is the most relevant for the application's purposes. So, *Patient* is associated with one and only one main disease through the one-to-many **suffer** relationship, which is decoded in the **Main Disease** entity (primary key is *DiseaseID*). The name of the disease (*Name*) is recorded as an attribute, along with all the thresholds related to the various physiological parameters that define the ranges within which the values are considered normal given the pathology (*MaxHeartRate*, *MinHeartRate*, *MaxSystolic*, *MinSystolic*, *MaxDiastolic*, *MinDiastolic*, *MaxOxygen*, *MinOxygen*).

We modeled the relationship **send** between *Patient* and *Notes* as one-to-one to represent a simplified communication mechanism where only the most recent note is relevant. This choice avoids storing

historical messages and reduces data complexity, as the system is designed to highlight the latest patient text rather than maintain a full communication log. Its attributes are *Notenum* (primary key), *NoteContent*, *Priority* and *UploadingDateandTime*. The *Priority* attribute is used to show the doctor the classification of the notes based on the priority set at the patient's discretion.

Furthermore, two data upload relationships derive from *Patient*, namely **upload1** and **upload2**. These refer to the possibility for the patient to upload their own physiological parameters (**Physiological Parameters** entity with attributes *PhysioID*, *Value* and *UploadingDateandTime*) and the data related to their lifestyle (**Life style data** entity with attributes *LifeStyleID*, *LSValue* and *LastModification*). The cardinalities of the relationships are one-to-many: each loaded parameter refers to one and only one patient; the minimum cardinality of 0 is linked to the fact that a patient might not have uploaded anything yet or, on the contrary, might have uploaded several parameters over time.

Both the *Physiological Parameters* and *Life style data* entities are respectively linked through the one-to-many relationships **type1** and **type2** to parameter decoding entities, namely **Physiological Parameters Type** and **Lifestyle Data Type**. These entities have as primary key respectively *TypePhysioID* and *LifeStyleTypeID*; both have *TypeName* as an additional attribute.

We gave the patient the possibility to upload their physiological parameters by importing them directly from a wearable device, specifically an Apple Watch. This information is present in the **Wearable device** entity. This entity is linked to the physiological parameter imported from it through the one-to-many relationship **import**: each device may have imported zero or more parameters; each parameter could have been imported by an external device or manually, hence the minimum cardinality at zero. The primary key of *Wearable device* is *DeviceID*, while the other attributes are *Name*, *ModelID* and *FirmwareVersion*.

The *Wearable Device* entity is also linked to the *Patient* with the one-to-one relationship **have**: each patient can own one and only one wearable, which in turn is associated with a single patient.

Technical description of the tables

After the creation of the ER diagram, we moved on to the phase where, starting from the diagram, we generated the database tables. The SQL script for the creation of each individual table is presented below.

The transition from the ER diagram to the relational logical schema was carried out by applying standard mapping rules. Specifically, one-to-many (1:N) associations were translated by inserting the primary key of the entity on the 'one' side as a foreign key within the table of the entity on the 'many' side. One-to-one (1:1) associations were instead resolved by inserting a foreign key into one of the two involved tables, to which the UNIQUE constraint was strictly applied to preserve the uniqueness restriction imposed by the cardinality. A clear example of this implementation is represented by the *AppointmentID* attribute, inserted as a *foreign key* with a UNIQUE constraint within the *MedicalRecord* table.

The generalization present in the ER diagram regarding accounts was treated by creating a table for the superclass and a table for each subclass, linking them through the *foreign key* of the *Username*, which is the primary key in *UserAccount*.

The first table we examine is **UserAccount**. The *primary key* is *Username* and *StatusAccount* is the attribute that encodes the state of a user's account. *Check* allows the attribute to assume only the following values:

- 0: blocked
- 1: active
- 2: to be activated, which is the state the account is in after being created by the administrator or the doctor, before personal information is entered.

```
CREATE TABLE UserAccount (
    Username VARCHAR(50) PRIMARY KEY,
    Password VARCHAR(255) NOT NULL,
    TaxCode VARCHAR(20) UNIQUE,
    Name VARCHAR(50),
    Surname VARCHAR(50),
    BirthDate DATE,
    ContactInfo VARCHAR(255),
    PhoneNumber VARCHAR(50),
    StatusAccount INT CHECK (StatusAccount IN (0,1,2)) DEFAULT 2
);
```

The *Administrator* table has *AdminID* as its primary key, an attribute that is set as *autoincrement*. It is connected to *UserAccount* with the *Username* foreign key.

```
CREATE TABLE Administrator (
    AdminID INTEGER PRIMARY KEY AUTOINCREMENT,
    Username VARCHAR(50) UNIQUE NOT NULL,
    Structure VARCHAR(50),
    StructureContact VARCHAR(100),
    FOREIGN KEY (Username) REFERENCES UserAccount (Username)
);
```

The same logic was maintained for the *Doctor* table: the primary key is *DoctorID* and *Username* (foreign key) serves as the link to *UserAccount*. In this case, the table also inherits the *AdminID* attribute (foreign key) given the union with the *hire* relationship.

```
CREATE TABLE Doctor (
    DoctorID INTEGER PRIMARY KEY AUTOINCREMENT,
    Username VARCHAR(50) UNIQUE NOT NULL,
    AdminID INT NOT NULL,
    FOREIGN KEY (Username) REFERENCES UserAccount (Username),
    FOREIGN KEY (AdminID) REFERENCES Administrator (AdminID)
);
```

The *Patient* table also provides *PatientID* as its primary key and *Username* (foreign key) is the foreign key for the *UserAccount* table. It inherits *DoctorID* and *DiseaseID* as foreign keys from the union with the *treat* and *suffer* relationships.

```

CREATE TABLE Patient (
    PatientID INTEGER PRIMARY KEY AUTOINCREMENT,
    Username VARCHAR(50) UNIQUE NOT NULL,
    Sex VARCHAR(1) CHECK (Sex IN ('M', 'F')),
    DoctorID INT NOT NULL,
    DiseaseID INT,
    FOREIGN KEY (Username) REFERENCES UserAccount (Username) ,
    FOREIGN KEY (DoctorID) REFERENCES Doctor (DoctorID)
);

```

The **MainDisease** table presents, as usual, a primary key which is an *ID* (*DiseaseID*) with an *autoincrement* setting.

```

CREATE TABLE MainDisease (
    DiseaseID INTEGER PRIMARY KEY AUTOINCREMENT,
    Name VARCHAR(50) UNIQUE NOT NULL,
    MaxHeartRate INT,
    MinHeartRate INT,
    MaxSystolic INT,
    MinSystolic INT,
    MaxDiastolic INT,
    MinDiastolic INT,
    MaxOxygen INT,
    MinOxygen INT
);

```

The **WearableDevice** table has *DeviceID* set as *autoincrement* as its primary key and inherits *PatientID* as a foreign key from the union with the *have* relationship.

```

CREATE TABLE WearableDevice (
    DeviceID INTEGER PRIMARY KEY AUTOINCREMENT,
    Name VARCHAR(50) NOT NULL,
    ModelID INT NOT NULL,
    FirmwareVersion VARCHAR(50) ,
    PatientID INT NOT NULL,
    FOREIGN KEY (PatientID) REFERENCES Patient (PatientID)
);

```

The **PhysiologicalParametersType** table has *TypePhysioID* as its primary key and the *TypeName* attribute has a *check* on the values it can assume: Heart Rate, Systolic Blood Pressure, Diastolic Blood Pressure, Oxygen Saturation, Weight.

```

CREATE TABLE PhysiologicalParametersType (
    TypePhysioID INTEGER PRIMARY KEY,
    TypeName VARCHAR(50) NOT NULL
    CHECK (TypeName IN ('Heart Rate', 'Systolic Blood Pressure',
        'Diastolic Blood Pressure', 'Oxygen Saturation', 'Weight'))
);

```

The **LifestyleDataType** table has *LifeStyleTypeID* as its primary key and in this case as well, *TypeName* has a *check* on the values: Diet, PhysicalActivity, Medications, SleepQuality.

```
CREATE TABLE LifestyleDataType (
    LifestyleTypeID INTEGER PRIMARY KEY,
    TypeName VARCHAR(50) NOT NULL
    CHECK (TypeName IN ('Diet', 'PhysicalActivity',
        'Medications', 'SleepQuality'))
);
```

The *Appointment* table has *AppointmentID* set as *autoincrement* as its primary key. The *AppointmentStatus* attribute was stored as an INT and its encoding is:

- 0: pending request
- 1: accepted request
- 2: refused request

The table also inherits *DoctorID* and *PatientID* as foreign keys from the merger with the *participate* and *create* relationships.

```
CREATE TABLE Appointment (
    AppointmentID INTEGER PRIMARY KEY AUTOINCREMENT,
    AppointmentDateAndTime DATETIME NOT NULL,
    Location VARCHAR(100),
    AdditionalNotes TEXT,
    AppointmentStatus INT, -- 0: Pending 1: Accepted 2:Refused
    DoctorID INT NOT NULL,
    PatientID INT NOT NULL,
    FOREIGN KEY (DoctorID) REFERENCES Doctor(DoctorID),
    FOREIGN KEY (PatientID) REFERENCES Patient(PatientID)
);
```

The *MedicalRecord* table has *RecordID* set as *autoincrement* as its primary key. It inherits *AppointmentID* as a foreign key from the union with the *summarize* relationship.

```
CREATE TABLE MedicalRecord (
    RecordID INTEGER PRIMARY KEY AUTOINCREMENT,
    AppointmentInfo TEXT,
    AppointmentID INT UNIQUE NOT NULL,
    FOREIGN KEY (AppointmentID) REFERENCES Appointment(AppointmentID)
);
```

The *Notes* table has *NoteNum* set as *autoincrement* as its primary key. It inherits *PatientID* as a foreign key from the union with the *send* relationship.

```
CREATE TABLE Notes (
    NoteNum INTEGER PRIMARY KEY AUTOINCREMENT,
    NoteContent TEXT NOT NULL,
    Priority VARCHAR(20),
    UploadingDateandTime DATETIME NOT NULL,
    PatientID INT UNIQUE NOT NULL,
    FOREIGN KEY (PatientID) REFERENCES Patient(PatientID)
);
```

The *PhysiologicalParameters* table has *PhysioID* set as *autoincrement* as its primary key. It inherits *PatientID*, *DeviceID* and *TypePhysioID* as foreign keys from the union with the *upload1*, *import* and *type1* relationships.

```

CREATE TABLE PhysiologicalParameters (
  PhysioID INTEGER PRIMARY KEY AUTOINCREMENT,
  Value DECIMAL(10, 2) NOT NULL,
  UploadingDateandTime DATETIME NOT NULL,
  PatientID INT NOT NULL,
  DeviceID INT NULL,
  TypePhysioID INT NOT NULL,
  FOREIGN KEY (PatientID) REFERENCES Patient(PatientID),
  FOREIGN KEY (DeviceID) REFERENCES WearableDevice(DeviceID) ,
  FOREIGN KEY (TypePhysioID) REFERENCES PhysiologicalParametersType(TypePhysioID)
);

```

The ***LifestyleData*** table has *LifeStyleID* set as *autoincrement* as its primary key. It inherits *PatientID* and *LifeStyleTypeID* as foreign keys from the union with the *upload2* and *type2* relationships.

```

CREATE TABLE LifestyleData (
  LifeStyleID INTEGER PRIMARY KEY AUTOINCREMENT,
  LSValue VARCHAR(255) NOT NULL,
  LastModification DATETIME NOT NULL,
  PatientID INT NOT NULL,
  LifeStyleTypeID INT NOT NULL,
  FOREIGN KEY (PatientID) REFERENCES Patient(PatientID) ,
  FOREIGN KEY (LifeStyleTypeID) REFERENCES LifestyleDataType(LifeStyleTypeID)
);

```

SYSTEM IMPLEMENTATION

The system handles the monitoring of patients admitted to three different hospitals located in the city of Milan: Niguarda Hospital, San Raffaele Hospital, and Policlinico. Therefore, it offers different services to administrators, who can manage the profiles of the users utilizing the app; doctors, who can monitor the health status of the patients under their care; and patients, who can upload their personal information to receive more in-depth monitoring.

To implement the system we created three different users' interfaces, for each possible actor involved in the usage of the software.

Login

The splash screen shows the login interface which the program uses to assess the user's identity.

The system requires the user's role, username, and password, ensuring that the entered data matches the data in the database. If the entered username or password does not match any data in the database, this will be displayed on the screen, also indicating whether the error relates to the first or the second parameter. If the user selected the wrong role, it will be displayed on the screen. Furthermore, the system provides the option to 'hide' or 'reveal' the password during entry.

Account activation

If the user is a patient and is logging in for the first time, the system requires them to choose a new password of at least four characters, which must be different from the OTP. Subsequently, a screen opens for entering personal data, requesting: First Name, Last Name, Email, Phone Number, Tax Code, Birthdate, and Sex. The user can then save their profile using the dedicated button, or clear the entered data. Once the data is saved, they can proceed to log in by entering the new password.

The image displays three overlapping screenshots of the 'Beat the Disease - Medical Information System' interface. The topmost window is titled 'Personal Information' and 'Complete your profile', asking the user to enter personal information to finish activation. It includes input fields for Name (e.g., Mario), Surname (e.g., Rossi), Email (e.g., mario.rossi@example.com), Phone Number (e.g., +39 123 456 7890), Tax Code (e.g., RSSMRA90A01H501Z), Birthdate (with dropdowns for year, month, and day), and Sex (a dropdown menu). At the bottom are 'Cancel' and 'Save profile' buttons. The middle window is titled 'Account Activation' and 'Set a new password', asking the user to enter and confirm a new password. It features two password input fields with toggle icons for visibility and 'Cancel' and 'Save' buttons. The bottom-most window is the splash screen, showing the system title and a 'Login' button.

Admin interface

If the user holds an admin role, the Dashboard presents hospital-specific statistics, showing the number of doctors, patients, and users reporting to them. Respectively, a 'pending account' category is displayed, referring to users undergoing registration who are awaiting approval from the corresponding administrator. Furthermore, graphs illustrate both the disease distribution and the account status of the users under the administrator's management. Meanwhile, the sidebar includes the following screens:

- **Manage Patients:** renders the username, status, and a button to view patient details. These details encompass personal data (full name, date of birth, tax code, email address, phone number, main disease, account status) and the doctor assigned to the patient. The system provides functionality to delete the patient's account if required, or suspend it in case of necessity, and to redirect the user to the "Manage Doctors" management screen.

Query used to retrieve the data:

""""

```
SELECT
u.Name, u.Surname, u.BirthDate,u.TaxCode,
u.ContactInfo,u.PhoneNumber,
u.StatusAccount,
m.Name,
du.Name, du.Surname,
du.Username
FROM UserAccount u
LEFT JOIN Patient p ON u.Username = p.Username
LEFT JOIN Doctor d ON p.DoctorID = d.DoctorID
LEFT JOIN UserAccount du ON
d.Username = du.Username
LEFT JOIN MainDisease m ON
m.DiseaseID = p.DiseaseID
WHERE u.Username = ?
""", (username,)
```

Query used to delete account:

```
"DELETE FROM Appointment WHERE PatientID = ?", (p_id,)
"DELETE FROM Notes WHERE PatientID = ?", (p_id,)
"DELETE FROM PhysiologicalParameters WHERE PatientID = ?", (p_id,)
"DELETE FROM LifestyleData WHERE PatientID = ?", (p_id,)
"DELETE FROM WearableDevice WHERE PatientID = ?", (p_id,)
```

- **Manage Doctors:** renders the username, status, and a button to view doctor details. Users can view personal information (full name, date of birth, tax code, email address, phone number, workplace, workplace contact, account status), as well as the patients assigned to them, with a shortcut to navigate back to the patient management screen. Additionally, options are available to delete the account or suspend it in case of policy violations, by changing its status.
- **Create Doctor Account:** Allows the user to input a username and temporary password to provision a new doctor account.

- **New Custom Disease:** Allows the user to view main diseases along with their respective parameters (Name, Max Heart Rate, Min Heart Rate, Max Systolic Pressure, Min Systolic Pressure, Max Diastolic Pressure, Min Diastolic Pressure, Max O2 Saturation, Min O2 Saturation), and to provision a new 'Custom Disease' by manually inputting the disease name and all parameter values. A validation error is displayed on screen if any required fields are left blank.

Query used to add a new disease:

```

""""
INSERT INTO MainDisease (
    Name, MaxHeartRate, MinHeartRate,MaxSystolic,
    MinSystolic,MaxDiastolic,MinDiastolic,
    MaxOxygen,MinOxygen
)
VALUES (?,?,?,?,?,?,?,?,?)
""", (disease_data)

```

Menu

Dashboard
Manage Patients
Manage Doctors
Create Doctor Account
Custom Disease

Logout

Add a new Custom Disease into the system

Name of the disease:

Insert the thresholds related to the disease below.

Max Heart Rate:

Min Heart Rate:

Max Systolic Blood Pressure:

Min Systolic Blood Pressure:

Max Diastolic Blood Pressure:

Min Diastolic Blood Pressure:

Max Oxygen Saturation:

Min Oxygen Saturation:

Add disease

Current Diseases

Name	Max HR	Min HR	Max SBP	Min SBP	Max DBP	Min DBP	Max O2	Min O2		
Heart Failure	110	50	140	80	90	50	100	90	Edit	Delete
Atrial Fibrillation	130	60	150	85	95	55	100	92	Edit	Delete
HF Custom1	120	55	140	85	90	50	100	92	Edit	Delete

- **Logout:** Allows the user to log out and return to the login page.

Doctor interface

If the user holds a doctor role, the main screen will open to a home page featuring three different widgets. The first one allows the user to open their work email, the second provides information about the next appointment (date, time, and patient), while the third leads directly to the patient file for the upcoming visit.

In addition to the home page, the sidebar contains the following screens:

- **Patients Overview:** Allows the user to use a search bar to find a subject, and provides a patients overview. It displays their personal data (First Name, Last Name, Main Disease), physiological parameters (HR, SBP, DBP, spO2, Weight), and the notes written by the patients themselves, along with the self-assigned priority level (low, medium, or high). Additionally, it offers the ability to view patient alerts, if a note has a medium or high priority, and/or if at least one of the patient's latest recorded physiological parameters falls outside their safe range. It also features a 'View' button, which redirects to the patient's personal page.

The 'View Patient' section allows the user to display a range of information about the patient and perform several functions. Specifically, it enables them to execute 'Update Disease Data' to modify the patient's main disease, or to send an email alert to the patient if the doctor deems that their condition is worsening significantly. Additionally, it displays the patient's latest note (each note is overwritten with every modification) along with its upload date, time, and priority level; as well as the subject's next scheduled appointment. There are also two widgets for lifestyle data and physiological parameters. The first shows information on diet, medications, physical activity, and sleep quality, indicating when this data was last modified (each update overwrites the previous one).

Query used to view latest lifestyle data:

```
SELECT ldt.TypeName, ld.LSValue, ld.LastModification
FROM LifestyleData ld
JOIN LifestyleDataType ldt
ON ld.LifeStyleTypeID = ldt.LifeStyleTypeID
WHERE ld.PatientID = ?
AND ld.LastModification = (
    SELECT MAX(ld2.LastModification)
    FROM LifestyleData ld2
    WHERE ld2.PatientID = ld.PatientID
    AND ld2.LifeStyleTypeID = ld.LifeStyleTypeID
)
ORDER BY ldt.TypeName
,,,,, (patient_id,))
```

The second displays the following parameters: Heart Rate, Systolic Blood Pressure, Diastolic Blood Pressure, Oxygen Saturation, and Weight (updated to the most recent value).

Menu

Dashboard

Patients Overview

Appointments

Create New Patient

Personal Info

Logout

← Back to Patients

Lorenzo Verdi

Main Disease: Heart Failure

Sex: M

Contact email: lorenzo.verdi@email.com

Contact phone: +39 333 1234567

Update Disease Data

Send Alert

Last Note

"Ho notato un lieve gonfiore alle caviglie la sera negli ultimi due giorni, ma respiro bene."

MEDIUM Updated: May 28, 2026 at 06:45 PM

Next Scheduled Appointment

No pending appointments

Current Lifestyle Data

Now data will overwrite existing data.

Diet

Low-sodium and low-fat diet

Last modified: May 01, 2026 at 10:00 AM

Medications

Ramipril 5mg/day

Last modified: May 15, 2026 at 09:00 AM

Physical Activity

30 minutes walking 4x/week

Last modified: May 01, 2026 at 10:05 AM

Sleep Quality

Fair (6-7h, occasional awakenings)

Last modified: May 28, 2026 at 06:45 PM

Latest Physiological Parameters

Heart Rate

72 bpm

Systolic Blood Pressure

125 mmHg

Diastolic Blood Pressure

82 mmHg

Oxygen Saturation

97.9%

Weight

81.5 kg

Below these widgets, charts illustrate the chronological trends for Heart Rate, Blood Pressure, Blood Oxygen, and Weight (the time window updates with each new data entry).

- **Appointments:** Allows the user to view and manage appointments through three different widgets.

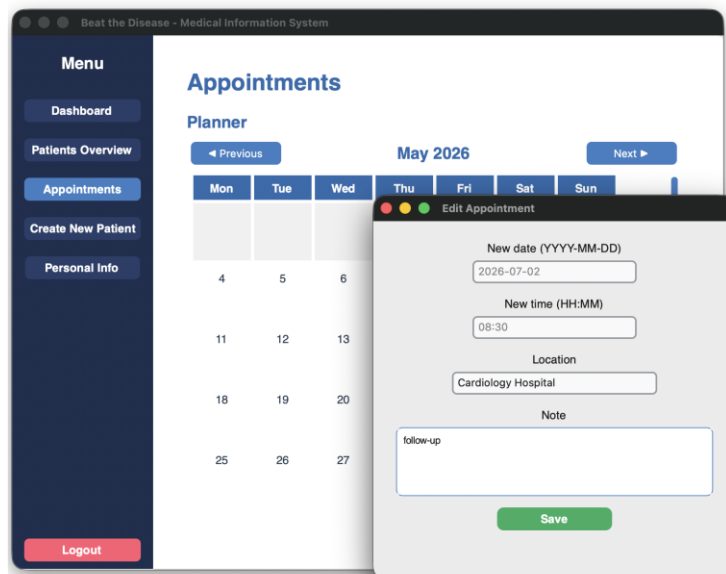
The first widget consists of an electronic agenda, displaying a calendar (covering 5 years in the past to 2 years in the future) that highlights the current date and marks the days with scheduled appointments, while also specifying the patient's first and last name, their appointment time, and whether the appointment has been accepted or not. Specifically, it highlights in green the days when all appointments have been accepted or completed, and in blue the days with pending appointments.

The second widget features a search bar where, by entering the patient's first and last name, a history of their past appointments as well as upcoming, uncompleted appointments is displayed. The third widget allows the user to either create or modify appointments. For creation, under the 'Create' section, the system requires entering: Patient First Name, Patient Last Name, Appointment Time and Date, Appointment Location (the location will be filled automatically based on the doctor's assigned hospital, but can be changed) and Additional Notes. For modifying appointments, under the 'Manage' section, the system requires entering the First Name and Last Name, and allows the user to either delete the appointment or modify it.

Query used to retrieve appointments information:

""""

```
SELECT a.AppointmentID, a.AppointmentDateAndTime,
a.Location, a.AdditionalNotes
FROM Appointment a
INNER JOIN Patient p ON a.PatientID = p.PatientID
INNER JOIN UserAccount ua ON p.Username = ua.Username
WHERE ua.Name LIKE ?
AND ua.Surname LIKE ?
AND a.DoctorID = ?
AND datetime(a.AppointmentDateAndTime) >=
      datetime('now', 'localtime')
ORDER BY a.AppointmentDateAndTime
""", (f"%{name}%", f"%{surname}%", doctor_id))
```



- **Personal Info:** Allows the user to edit their personal information, namely: First Name, Last Name, Username, Birthdate, Contact Info, and to change their password.
- **Create New Patient:** Allow the user to create a new patient account, by adding their: Patient Username, Temporary Password, Patient Tax Code and Main Disease (one of the possible options). The patient will then be allowed to add their other information and to modify the password.
- **Logout:** Allows the user to log out and return to the login page.

Menu

Dashboard

Lifestyle Data

Physiological Parameters

Patient Notes

Appointments

Personal Info

Logout

Patient Dashboard

Main Disease: Heart Failure

Welcome back, patient19!

Your Main Doctor

Dr. Andrea Barbieri

Contact: andrea.barbieri@hospital.it

Hospital: Ospedale Niguarda Milano



Next Appointment

No upcoming appointment scheduled.

Location: —

Notes: —



Latest Physiological Readings

Diastolic Blood Pressure	82 mmHg	Normal
Heart Rate	72 bpm	Normal
Oxygen Saturation	97.9%	Normal
Systolic Blood Pressure	125 mmHg	Normal
Weight	81.5 kg	N/A

Patient interface

If the user holds a patient role, the main screen opens to a dashboard featuring three distinct widgets. The first widget displays information regarding the patient's assigned physician, such as their name, workplace contact details, and the hospital where they are located. The second widget provides specific details about the patient's next appointment, if it has been scheduled, and if so the location, the time and any additional notes about the appointment appended by the doctor. The third widget displays the patient's most recent physiological parameters: Diastolic Blood Pressure, Heart Rate, Oxygen Saturation, and Weight. Furthermore, it highlights values in green and categorizes them as 'normal' if they fall within the healthy range; otherwise, it highlights them in red and classifies them as 'abnormal', based on the disease of the patient.

In addition to the dashboard, the sidebar contains the following screens:

- **Lifestyle Data:** Allows patients to upload and view their current lifestyle data. The first widget enables the user to select the specific data type they wish to upload chosen from: Diet, Physical Activity, Medications, or Sleep Quality; and to enter the corresponding values as a text string. Each time new information is submitted, the respective category is overwritten. Upon successful completion, a confirmation message is displayed on the screen. The second widget allows the user to view their previously entered lifestyle data (Diet, Physical Activity, Medications, and Sleep Quality) along with the precise date and time of the upload.

Query used to update physiological parameters from wearable device:

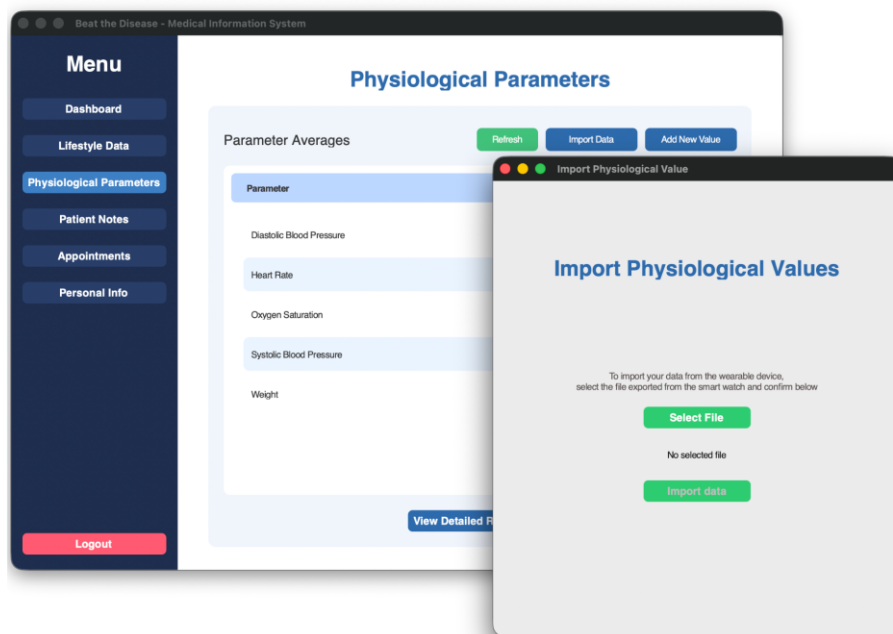
```
DELETE FROM LifestyleData
WHERE PatientID=? AND LifeStyleTypeID=?
""", (patient_id, lifestyle_type_id))

INSERT INTO LifestyleData (LSValue,
LastModification, PatientID, LifeStyleTypeID)
VALUES (?, ?, ?, ?)
""", (value, current_time, patient_id, lifestyle_type_id))
```

- **Physiological Parameters:** Allows the patient to view the average value for each Physiological Parameter, along with the total number of measurements utilized to calculate that average. Furthermore, it enables patients to either import data from a specific file generated by their wearable device or manually add new entries. When adding data manually, the user selects the parameter type and enters the corresponding numerical value, as well as the date and time (if the date and time fields are left blank, the system automatically assigns the current timestamp to the measurement). When data is instead imported from a wearable device, the system prompts the user to select a file from their device. The information exported from the wearable device as an XML file must then be selected. Once "Upload" is clicked, the data undergoes a filtering process so that only the latest measurement taken by the wearable device is entered into the system. The XML file containing the recordings of the Apple Watch can be downloaded from the Health App of iOS devices (iPhone, iPad). If the patient has no wearable device associated with their account, pressing the "Import data" button will prompt them to register one by providing the following details: Model Name, Model ID, and Firmware version. Once the information has been submitted via the Register button, the system will be updated accordingly.

Query used to insert physiological parameters:

```
""
INSERT INTO PhysiologicalParameters (Value,
UploadingDateandTime, PatientID, DeviceID, TypePhysioID)
SELECT ?, ?, ?, ?, ?
WHERE NOT EXISTS (
SELECT 1 FROM PhysiologicalParameters
WHERE PatientID = ? AND TypePhysioID = ? AND
UploadingDateandTime = ?
)
"", records_to_insert)
```



- **Patient Notes:** Allows the patient to create new notes for their healthcare provider, featuring a lower priority bar to categorize the note (low, medium, high) according to patient perception. Upon user confirmation, the note is saved. The display also features the patient's last submitted note, complete with its assigned priority, publication timestamp, and content.
- **Appointments:** Allows the patient to review pending appointment proposals from their healthcare provider, with the option to accept or decline them. Each proposal details the scheduled date, location, and physician's notes on the appointment. Furthermore, the system displays upcoming appointments (including date, time, doctor, location, and notes) and maintains a historical record of all past appointments with the same details.

Query used to view pending appointment request:

```
""""
SELECT a.AppointmentID, a.AppointmentDateAndTime,
" AS Location, " AS AdditionalNotes,ua.Name,
ua.Surname
```

```
FROM Appointment a
JOIN Doctor d ON a.DoctorID = d.DoctorID
JOIN UserAccount ua ON d.Username = ua.Username
WHERE a.PatientID=? AND a.AppointmentStatus=0
ORDER BY a.AppointmentDateAndTime

""", (patient_id,
```

- **Personal Info:** Allows the user to edit their personal information: First Name, Last Name, Username, Birthdate, Contact Info, Telephone Number, sex and to change their password.
- **Logout:** Allows the user to log out and return to the login page.

CONCLUSIONS

The system was designed for widespread implementation across public healthcare facilities throughout Italy, with the goal of supporting both healthcare professionals and the patients under their care.

The application's core **strengths** can be summarized across four dimensions.

- **Extensibility:** the platform is not limited to a single hospital and is designed to scale across a large number of facilities and users. At the same time, administrative control remains internal to each institution, ensuring that individual facilities retain operational autonomy while benefiting from a shared infrastructure.
- **Completeness:** the system consolidates patient health management into a single platform, covering physiological parameter monitoring (HR, blood pressure, SpO₂, weight), lifestyle data tracking, doctor–patient communication, and appointment scheduling. Physiological trends are visualized over time, building a detailed longitudinal record for each patient. Communication is structured around tiered priority notes (low, medium, high), while appointment management is presented through a digital agenda interface designed for intuitive use by clinicians.
- **Adaptability:** disease thresholds are configurable on a per-patient basis, accounting for individual factors such as age, sex, comorbidities, and additional pathologies. This allows for more targeted and clinically meaningful monitoring rather than relying on generic population-level reference values.
- **Clinical decision support:** doctors are aided by trend visualizations, automated alerts for out-of-range parameters, and priority-flagged notes. The system also supports email notifications, enabling prompt intervention even outside routine check-ins.
- **Profile management:** the administrator has the ability to delete patient and physician accounts when necessary. Furthermore, the administrator can suspend the profile, by temporarily revoking access to the platform, in response to physician misconduct or other operational requirements.

Some **limitations** are inherent to the nature of the system and cannot be fully eliminated:

- **Security vulnerabilities:** given the volume and sensitivity of the data collected (including personal identifiers, medical history, medications, and lifestyle information) the system presents an attractive target for cyberattacks. Robust user protection mechanisms are therefore essential and must be treated as a core requirement rather than an afterthought.
- **Human unreliability:** since measurements are self-reported by patients, the system is inherently susceptible to input errors, whether from incorrect measurement technique or simple data entry mistakes. While threshold-based alerts can partially mitigate this risk, they cannot eliminate it entirely

Other limitations are more technical in nature and can be concretely addressed in future development cycles:

- **Wearable import function:** the current implementation only retrieves the most recent measurement from the wearable's XML export, and the file must be selected and uploaded manually. There is no direct communication between the wearable device and the platform.
- **Blood pressure reliability:** the wearable device provides an estimated rather than a clinically validated blood pressure reading, which limits its diagnostic reliability compared to a conventional sphygmomanometer.
- **Interface complexity:** The current interface may present usability challenges for elderly or very young patients, potentially leading to data entry errors, accidental deletion of records, or general misuse of the application.
- **Unencrypted data storage:** passwords and personal data are currently stored in plain text, meaning that anyone with direct database access can read all user information. This constitutes a significant security flaw.
- **Oversimplification of disease management:** the system supports a limited set of configurable thresholds and does not allow for multiple concurrent disease profiles, which may result in erroneous abnormality flags for patients with complex clinical pictures.
- **Lack of integration with national health systems:** the platform currently operates in isolation from regional and national Italian healthcare infrastructure, limiting interoperability and data continuity.

Future Developments

The roadmap for future improvements directly addresses all of the limitations outlined above.

Addressing inherent limitations:

- **Stronger security posture:** implement robust authentication mechanisms (e.g., multi-factor authentication) and schedule regular security audits, so that vulnerabilities are identified and resolved proactively before they can be exploited.
- **Unrealistic measurement detection:** develop algorithms that cross-reference new entries against a patient's historical data and personalized disease profile, reducing the risk of both typing errors and undetected measurement mistakes. Patients would be notified when a submitted value appears physiologically implausible.

Addressing technical limitations:

- **Automated wearable data import:** remove the need for manual file handling by establishing a direct communication channel between the wearable device and the platform. This would allow more frequent and reliable synchronization, giving physicians access to a richer and more up-to-date parameter history.
- **Blood pressure recording modality tagging:** tag each blood pressure entry with its recording method, distinguishing sphygmomanometer readings from wearable estimates, so that clinicians can appropriately weight the reliability of each measurement in their assessments.
- **Caregiver interface:** introduce a dedicated interface for caregivers supporting elderly or very young patients, enabling them to manage data entry, appointments, and medication information on the patient's behalf. This would reduce incorrect data entry and ease the burden on both patients and physicians.
- **Full data encryption:** implement end-to-end encryption for all stored data, including passwords and personal health records, ensuring that even in the event of unauthorized database access, sensitive information remains unreadable.

- **Advanced alerting system:** upgrade alerting logic beyond simple threshold violations to incorporate trend-based alerts, multi-signal prioritization rules, and individualized thresholds that better reflect each patient's clinical context and comorbidity profile.
- **Multi-disease profile support:** extend the disease management module to allow the assignment of multiple concurrent disease profiles per patient, each with its own set of configurable thresholds. This would prevent false abnormality flags in patients with complex clinical pictures and enable more nuanced monitoring.
- **Integration with national and regional health systems:** establish interoperability with Italian regional and national healthcare infrastructure (e.g., Fascicolo Sanitario Elettronico), enabling continuity of care across institutions and reducing redundant data entry.
- **AI-based predictive models:** introduce machine learning models for deterioration risk stratification, treatment adherence scoring, and readmission prediction. These should be developed in close collaboration with clinicians and undergo rigorous validation before deployment in a real-world setting.

REFERENCES

- [1] D. A. Lane *et al.*, “Atrial fibrillation,” *The Lancet*, vol. 407, no. 10532, pp. 1000–1013, Mar. 2026, doi: 10.1016/S0140-6736(25)02166-X.
- [2] L. Bergau, P. Bengel, V. Sciacca, T. Fink, C. Sohns, and P. Sommer, “Atrial Fibrillation and Heart Failure,” *J Clin Med*, vol. 11, no. 9, p. 2510, Apr. 2022, doi: 10.3390/jcm11092510.
- [3] A. Hughes, M. M. H. Shandhi, H. Master, J. Dunn, and E. Brittain, “Wearable Devices in Cardiovascular Medicine,” *Circ Res*, vol. 132, no. 5, pp. 652–670, Mar. 2023, doi: 10.1161/CIRCRESAHA.122.322389.
- [4] S. Fabien, S. Waechter, G. A. Kayode, B. Müller, M. Roth, and F. Koehler, “Smartwatches in the Assessment of Heart Failure Patients in Epidemiology and Pathophysiology Studies: A Scoping Review,” *ESC Heart Fail*, vol. 12, no. 3, pp. 1727–1738, Jun. 2025, doi: 10.1002/ehf2.15226.