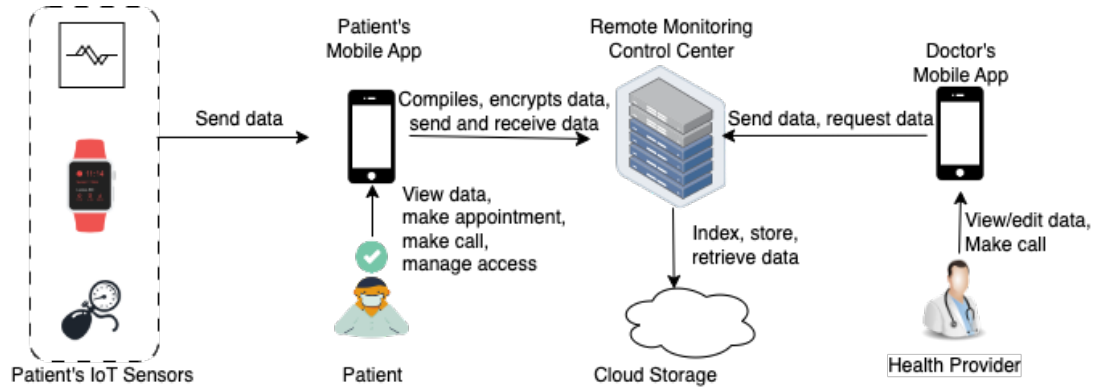
Final Project

Due Date: 10 May 2023

This project should be done individually.

In this project you will design and simulate the full working of a remote health monitoring system that can support patients with limited mobility to be able to consult remotely with their health providers, and in which the health providers can remotely collect vitals and other health information from patients through body sensors.

The following figure depicts the system, with all of the components you will implement.



The major components are described below. Please note that a lot of details are left open-ended for you to implement as you see fit. The more thorough, close to real-world implementation you provide, the more marks you will gain. At the minimum, you must simulate and demonstrate all the features of the components described below.

IoT Sensors:
Remote sensing is accomplished through wearable Internet of Things (IoT) devices. Each patient wears a number of sensing devices, for example, those that measure vitals like oxygen saturation, pulse, BMI, blood pressure etc., specific movement sensors (e.g. a wearable device that measures the hand tremors indicative of Parkinson's disease), and general movement sensors that record general movement, detect falls, or beep when a patient needs to change position (e.g. when a patient has been sitting for too long and needs to move about).

Remote Monitoring Control Center:
The wearable devices send data in the form of periodic updates to an app on the patient's mobile phone. The app compiles a daily update of a patients' vitals, encrypts it, and sends it to the Remote Monitoring Control Center (RMC2) over the Internet. The RMC2 indexes and stores the data in cloud storage. When a health provider wishes to view a patient's data (typically before a patient's consultation), they authenticate themselves to RMC2, and request a particular kind of data (e.g Patient 1234's ECG history for March 2023). RMC2 then checks if the particular health provider has authorization (granted by the patient) to view the data. If

yes, they can then view the data, and if not, they can either withdraw their request or request the patient for permission to access the data.

Electronic Health Records (EHRs):
The RMC2 maintains patients' medical history in the form of EHRs, or Electronic Health Records, which are files containing all of a patient's consultation data – appointment history, prescriptions and recommendations, as well as a general profile of the patient containing their ID and demographic data as well as dietary restrictions, allergies, regular medications, chronic health conditions and so on. These EHRs are stored encrypted in the cloud, and when a health provider wishes to add data into a patient's EHR or edit a data item, the RMC2 decrypts it, edits it, re-encrypts and and stores it back in the cloud. Thus, for each patient, the RMC2 maintains two kinds of data: the historical log of all data sent by wearable IoT devices, and the EHR, which is regularly updated.

Patients' mobile app:
The patients' mobile app contains the following features:

- **Appointments and Consultation:**
  The patients' mobile app allows patients (or their carers) to search for doctors and make appointments for remote consultation. The patient can make and receive calls to doctors through the app, at set appointment times, and the calls may be audio or video. The calls can be recorded as well and played back later. The patient can make notes during a call. Doctors can add notes for the patient during a call. The app maintains the call history (time, length, associated notes, associated prescriptions etc.) so that the patient or a carer can see this history at a later time.

- **Data Viewing and Analysis:**
  The patients' mobile app also allows them to view and search through past data, see stats and summaries of their past health trends, view prescriptions and doctors' recommendations, and view (but not edit) their EHRs.

- **Security and Access Control:**
  The mobile app is responsible for several of the security features of the monitoring system. It encrypts patients' data before sending it to the RMC2 and decrypts it before the patients view any retrieved data. For this, it manages encryption keys (passwords) set by the patient. It also allows patients to perform access control, as the key idea of the system is to allow patients to maintain control over their own data. Patients can choose which doctors can view which of their data – for example, a patient may set permissions such that ECG and pulse data can be seen by the patient's cardiologist but not their dentist. A patient can also view and approve or deny requests by health providers for providing access to patient data.

Health Provider Application:
Another app that is part of the monitoring system is for health providers. Health providers can perform the following in their app:

- Log in
- View (and approve or reject) appointment requests;
- View past consultation history of any patient;
- Search for particular data of any of their patients using a search string composed of data type (e.g. ECG/hand movement sensor) and a date or date range;
- Request access to additional data
- Add data into a patient's medical record.
- Edit their patients' EHR (this is only allowed if a patient has approved a doctor for editing their EHR AND the RMC2 verifies that the doctor is qualified to make the change).

Data and File Handling:
You must include sample data for 10 different patients, at least 5 doctors with different specialisations, and for each patient, at least 3 months' history of device data, EHR with all current and historical details and history of at least 3-4 consultations per patient. This data should be maintained in files that you submit with your program and you must implement binary file handling for reading/writing in these files**. All data that is stored and retrieved must be in files; no marks for keeping data in memory only.**

Design:
Start your project by designing a class diagram. Carry out CRC modelling to identify classes, responsibilities and collaborators. Then identify the nature of the relationships between classes and draw a class diagram summarising them. You should use all concepts studied in the course – this means that your design should include composition, aggregation, association and inheritance relationships. Your design will be marked separately. Your design must demonstrate a strong grasp on object-oriented design principles such as encapsulation and abstraction as well as modularity/maintainability.

Open-ended Problems:
A lot of the details of how you will implement the features of the various components are left to you. For example:

- the encryption and access control functionality is left to your own thinking,
- as is the design of menus and the flow of the program,
- as well as how you will simulate communication between various components (e.g. IoT device to RMC2)
- and also the analysis of past data in the patients' mobile app, calculation of statistics, rendering any graphs etc.

Add as much detail and functionality to the project on your own as needed. Your implementation should demonstrate file handling, operator overloading, polymorphism and virtual functions, as well as all other concepts taught in the course. Output should be clear and easy to follow. Write your driver program such that you can easily demonstrate all features

of your program. Bonus marks will be awarded if you implement any features extraordinarily well and thoroughly, and for original thought and design.

Marking Rubric:

Item	Description	Marks
Appropriate design	1. Class diagram is appropriate for the problem 2. Maps to the code 3. Does not violate OOP principles of encapsulation, modularity, abstraction 4. Correctly implements relationships.	Total: 20 Perfect: 20 Good: 15 Average: 10 Poor: 5
Sufficient data pre-populated for testing program	Includes all data mentioned in "data and file handling" section.	Yes: 10 marks No: 0 marks Partially: 5 marks
IoT Sensors	1. Good variety of sensors is implemented 2. IoT sensors regularly generate and send sensible, realistic data to mobile app. 3. Data is well formatted. 4. Data is indexed appropriately to be easily searchable.	Total: 20 5 marks for each item.
Patient Mobile app	1. Compiles daily updates from IoT sensor data with appropriate formatting and indexing 2. Encryption function (using patient provided password) 3. Access control functionality 4. Allows making appointments (first gets list of available appointments from RMC2) 5. Simulate consultation calls (saves recording, runs timer, saves associated notes etc.) 6. Past data analysis functionality with stats calculation 7. View prescriptions, EHRs 8. View and approve data access requests	Total: 60 marks 5 marks each for items 1, 4, 7, 8 10 marks each for items 2, 3, 5, 6
EHR	1. Includes all relevant fields 2. Is well formatted 3. Easily searchable 4. Can be updated	Total: 20 5 marks for each item
RMC2	1. Communicates with patient app, health provider app 2. Indexes IoT sensor updates received from patient app 3. Manages data search requests coming from patient and health provider 4. Stores data in cloud	Total: 60 marks 10 marks for each item.

	5. Manages access to EHRs and IoT data according to patient preference, updating access when patient requires it. 6. Stores in cloud storage (simulate a cloud storage that is actually on local disk). 7. Binary file handling appropriately implemented.	
Health provider application	1. Log in functionality 2. Appointment requests management 3. View past consultation history of any patient 4. Search for specific portions of patient data 5. Request access to additional data from patient 6. Add data into a patient's medical record. 7. Edit their patients' EHR 8. Appropriate access control	Total: 60 marks 5 marks each for items 1, 2, 5, 6 10 marks each for items 3, 4, 7, 8
Communication	Communication between all components is appropriately simulated and does not violate OOP principles (do not use global functions, public data members or too many friend functions).	Total: 20 Perfect: 20 Good: 15 Average: 10 Poor: 5
Driver program and output	1. All functions of the program are well demonstrated; no marks for implemented code that is not demonstrated in the output. 2. Good menu design 3. Easy user interaction.	Total: 30 marks 10 marks each.

Submission Instructions:

Combine all separate files into a single zip file before submitting.
Your code MUST be in separate .cpp and .h files for each class, and a .cpp file called driver.cpp that is the driver code.
Submit your code on Google Classroom with the following naming convention:

Rollnumber1_rollnumber2_finalProject.zip

Late submissions are not allowed. There will be no marks for late submissions.