

SOEN6471 - Advanced Software Architecture

Team G - Deliverable 1

## iCare System Architecture Design

Summer 2020

Student ID	Student Name
40124288	Van Tuan Tran
40151903	Jasmeet Walia
40102956	Haifeng Wu
40091993	Himen Hitesh Sidhpura
40110312	Yashwanth Vummaneni

**Professor Pankaj Kamthan**

Department of Computer Science and Software Engineering  
Gina Cody School of Engineering and Computer Science  
Concordia University

# Contents

<b>1</b>	<b>Vision</b>	<b>4</b>
1.1	Scope . . . . .	4
1.2	Goals . . . . .	4
<b>2</b>	<b>Stakeholders and Concerns</b>	<b>6</b>
2.1	User . . . . .	7
2.2	Negative User . . . . .	8
2.3	Software Provider Organisation . . . . .	8
2.4	Regulatory Authorities . . . . .	9
2.5	Investors . . . . .	10
2.6	Customer Support . . . . .	10
<b>3</b>	<b>Requirements</b>	<b>10</b>
3.1	Non-Functional Requirements . . . . .	10
3.2	Functional Requirements . . . . .	12
<b>4</b>	<b>Viewpoints and Views</b>	<b>15</b>
4.1	Use case views . . . . .	15
4.1.1	Use cases related to "User" stakeholder . . . . .	15
4.1.2	Other use cases . . . . .	17
4.2	Logical View . . . . .	18
4.3	Process View . . . . .	19
4.3.1	Other Process views . . . . .	20
4.4	Implementation View . . . . .	21
4.4.1	Overall Implementation . . . . .	21
4.4.2	Backend Implementation . . . . .	21
4.5	Deployment View . . . . .	22
<b>5</b>	<b>Architectural Decisions</b>	<b>23</b>
<b>6</b>	<b>Proof-of-Concept</b>	<b>29</b>
6.1	Van Tuan Tran . . . . .	29
6.1.1	Parallel Development . . . . .	29
6.1.2	Result Demo . . . . .	30
6.1.3	Source code . . . . .	31
6.2	Himen Hitesh Sidhpura . . . . .	31
6.2.1	Result Demo . . . . .	31
6.2.2	Source code . . . . .	33
6.3	Jasmeet Walia . . . . .	33
6.3.1	Result Demo . . . . .	33
6.3.2	Source code . . . . .	35
6.4	Yashwanth Vummaneni . . . . .	35
6.4.1	Result Demo . . . . .	35
6.4.2	Source code . . . . .	37
6.5	Haifeng Wu . . . . .	37
6.5.1	Result Demo . . . . .	37
6.5.2	Source code . . . . .	38

<b>A</b>	<b>Appendix: Team Collaboration and Communication</b>	<b>39</b>
A.1	Collaboration . . . . .	39
A.2	Communication . . . . .	39
A.3	Tools . . . . .	39
<b>B</b>	<b>Appendix: Glossary</b>	<b>40</b>

## List of Figures

1	Mind-Map of Stakeholders in iCare system. . . . .	7
2	4+1 View Model of Architecture: Version 2. . . . .	15
3	Use case diagram for Authentication and Registration. . . . .	15
4	Use case diagram for view and edit user profile. . . . .	16
5	Use case diagram for appointment management. . . . .	17
6	Use case diagram for Frequency Asked Questions. . . . .	17
7	Use case diagram for Insurance. . . . .	18
8	Use case diagram for Government. . . . .	18
9	iCare components. . . . .	18
10	Interactions between iCare components when making an appointment. . . . .	19
11	Interactions between iCare components when view or modifying user profile. . . . .	20
12	iCare implementation view. . . . .	21
13	iCare system's backend implementation view. . . . .	22
14	iCare system deployment view. . . . .	23
15	Front-end and Back-end development in parallel. . . . .	29
16	API provided by Java Spring Application. . . . .	30
17	Web Application served by static HTML, CSS and JavaScript built from Reactjs. . . . .	30
18	Registration Form for doctor in iCare Application. . . . .	31
19	Registration Form for Patient in iCare Application. . . . .	32
20	Login Page of iCare Application. . . . .	32
21	Login page showing wrong user & password. . . . .	33
22	Mobile-friendly page of iCare Application. . . . .	34
23	Edit Profile page of Doctor Profile. . . . .	34
24	Home page of Doctor. . . . .	35
25	View profile page. . . . .	36
26	Edit profile page. . . . .	36
27	Patient profile page. . . . .	37
28	Interceptor and setting session in-active invalid time. . . . .	38
29	Automatic logout after 5 seconds' inactive. . . . .	38

## List of Tables

1	Architecture Decision 1 - Overall Architecture. . . . .	24
2	Architecture Decision 2 - Separation of front-end and back-end. . . . .	24
3	Architecture Decision 3 - Back-end language. . . . .	25
4	Architecture Decision 4 - Front-end language and framework. . . . .	25
5	Architecture Decision 5 - RESTful API principle. . . . .	26
6	Architecture Decision 6 - Back-end framework. . . . .	26
7	Architecture Decision 7 - MVC design pattern. . . . .	27
8	Architecture Decision 8 - Database type. . . . .	27
9	Architecture Decision 9 - Monolithic vs Micro-service architecture. . . . .	28
10	Architecture Decision 10 - Clustering and load balancing. . . . .	28

# 1 Vision

At the beginning of 2020, a novel corona virus has caused severe damage to the health system in many countries in the world. The best precaution to avoid infected is ***social distancing***. This has made the need for a Health Information System to rise.

In this report, we designed a software architecture for a mobile-friendly web application for a Hospital in Montréal, Québec, Canada. The main goal of the application was to provide hospital services without physical contact, following social distancing rules.

## 1.1 Scope

Due to the short duration of the course and available human resource of Team G, we decided to scope the project to the following constraints.

- Location: Montréal, but the services can be accessed anywhere.
- Single hospital.
- No department.

## 1.2 Goals

We decomposed the main goal into many smaller goals for more details. To represent the goals, we use the Goal-Question-Metric model. [2].

### (a) Goal 1

Goal	Purpose	iCare must be accessible within Canada by people using a mobile device.
	Issue	Using mobile devices
	Object	Product
	Viewpoint	User
Question		Should it be an App or a Website?
Metric		Market share of mobiles platforms and web browsers.
Question		Is it support offline mode?
Metric		Size of data that can be stored offline.

### (b) Goal 2

Goal	Purpose	Registered users can retrieve doctor profiles.
	Issue	Profile
	Object	Resources
	Viewpoint	Patient
Question		What is the specialty of a doctor?
Metric		How many departments that iCare support?
Question		What information should be included in the profile?
Metric		Is data modified by the appropriate user?

(c) Goal 3

Goal	Purpose	Registered users can retrieve patient records.
	Issue	Profile
	Object	Resources
	Viewpoint	Doctor
Question		What information should be included in the profile?
Metric		How much data should be presented?

(d) Goal 4

Goal	Purpose	Allow users to book appointments with doctors.
	Issue	Schedule
	Object	Resources
	Viewpoint	Patient
Question		How flexible is the time window for booking or canceling a doctor's appointment?
Metric		How many available time-slots a patient can book?
Question		Is the appointment system accessible for 24*7?
Metric		Downtime for the system (99.99% up-time).

(e) Goal 5

Goal	Purpose	Only authorized personnel can access to specific functions.
	Issue	Authentication
	Object	Resources
	Viewpoint	User
Question		Does the system protect patient information from the unauthorized persons?
Metric		How many tried before blocking unauthorized access?
Question		Is data modified by the appropriate user?
Metric		How many tried before blocking unauthorized access?

(f) Goal 6

Goal	Purpose	Saving patients' time and transportation cost.
	Issue	Remote treatment
	Object	Product
	Viewpoint	Hospital
Question		Are we reducing the number of times of seeing a doctor in person for chronic disease?
Metric		The number of online prescription. (Each online prescription means one time convenience).

(g) Goal 7

Goal	Purpose	Gain praise from users.
	Issue	User satisfaction
	Object	Product
	Viewpoint	Manager
Question		Are the users satisfied with the system?
Metric		Get at least 80% 4 out of 5 credits.

(h) Goal 8

Goal	Purpose	User feedback regarding issues.
	Issue	Feedback
	Object	Product
	Viewpoint	Developer
Question		How many bugs in the system and the necessary enhancements to the app?
Metric		Amount of time spent in resolving bugs, enhancements, maintenance.

(i) Goal 9

Goal	Purpose	FAQ regarding how to use the app.
	Issue	Question & Answer
	Object	Resources
	Viewpoint	Doctor
Question		How the app can be used?
Metric		How many functionalities covered in the document?

(j) Goal 10

Goal	Purpose	COVID-19 PPE Inventory system.
	Issue	Storage
	Object	Resources
	Viewpoint	Doctor
Question		How many PPE suits are available?
Metric		The number of available PPE suits.
Question		How to order PPE suits?
Metric		A list of vendors.
Question		How many PPE suits are used in one day?
Metric		The number of PPE suits throw away every day.

## 2 Stakeholders and Concerns

Stakeholders are those individuals which access the systems. There are several stakeholders considered while preparing System Architectural Description of the system. These stakeholders are grouped into users, Software Provider organizations, Regular Authorities, Investors, and Customer Support.

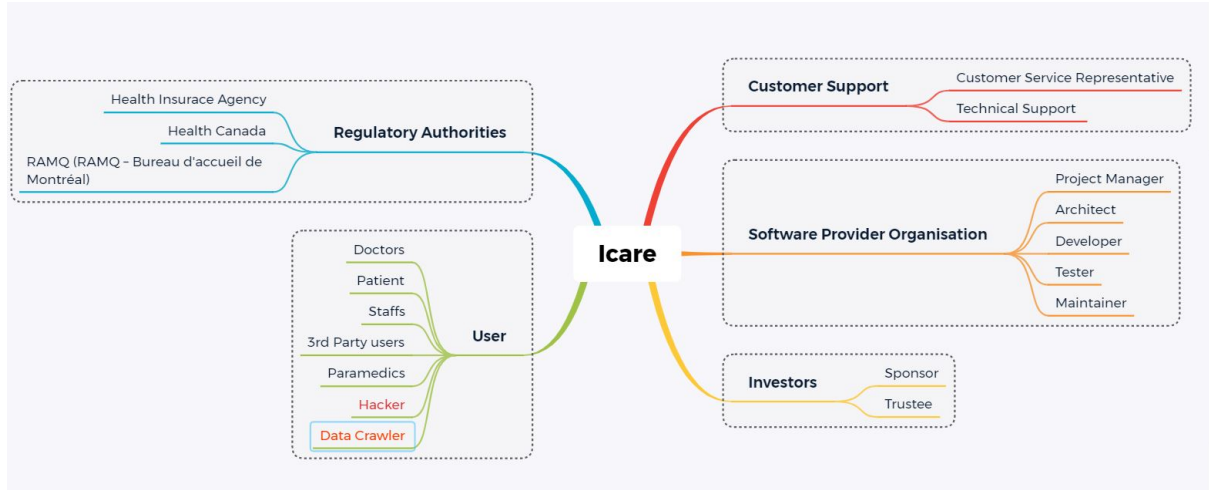


Figure 1: Mind-Map of Stakeholders in iCare system.

## 2.1 User

1. **Doctor:** A qualified practitioner of medicine; a physician.

Concerns	Quality Attributes
Is all patient information is validated?	Correctness
Does the system protect from all unauthorized access?	Security
Is it easy to operate the system?	Usability

2. **Patient:** A patient is any recipient of health care services performed by healthcare professionals.

Concerns	Quality Attributes
Is all doctor information is validated?	Correctness
Does the system protect from all unauthorized access?	Security
Is it easy to operate the system?	Usability

3. **Staff:** Medical staff is approved and credentialed to provide health care to patients.

Concerns	Quality Attributes
Is the information is validated?	Correctness
Does the system protect from all unauthorized access?	Security
Is it easy to operate the system?	Usability
Does the system give a quick response?	Performance
Is the system available all the time?	Availability

4. **Third-Party User:** A user who has the right to access confidential details such as Regulatory Authorities.

Concerns	Quality Attributes
Is all the information provided by the user is validated?	Correctness

5. **Paramedics:** A group of the user who responds to emergency calls outside a hospital/clinic.

Concerns	Quality Attributes
Get the work done as fast as possible	Performance
Available anytime, anywhere	Availability
Is it easy to operate the system?	Usability

## 2.2 Negative User

1. **Hacker:** A group of the user who tries to access unauthorized information of the system.

Concerns	Quality Attributes
Does the system protect from all unauthorized access?	Security

2. **Data crawler:** A user who tries to crawl the unauthorized information.

Concerns	Quality Attributes
Which information should make public by the system?	Security

## 2.3 Software Provider Organisation

1. **Project Manager:** An Individual is responsible for the planning and execution of the project.

Concerns	Quality Attributes
Does the software will deliver on time and also in the planned budget?	Feasibility
Are enough skilled resources are available to develop the system?	Feasibility
Which methodologies should use for development, agile, or waterfall development?	Efficiency/Effectiveness
Is the easy to modified implementation of the system?	Maintainability

2. **Software Architect:** An Individual is responsible for choosing the high-level design of the project and enforce coding standards.

Concerns	Quality Attributes
Is the system can adapt to the new environment?	Portability
Can we inherit a similar system design?	Effectiveness
Is the technology used to build a system is outdated? If yes, Is it easy to upgrade the system?	Maintainability
Is the architecture extendable? (to form a product line)	Scalability
Are the features of a system able to interact among themselves?	Interoperability

3. **Developer:** A group of the user who develops software.



Concerns	Quality Attributes
What is the environment of our system? (Hardware, Operating system)	Adaptability
Can we inherit similar system design?	Effectiveness
Is the technology used to build a system is outdated? If yes, Is it easy to upgrade the system?	Maintainability
Is the architecture extendable?(to form a product line)	Scalability
Are the features of a system able to interact among themselves?	Interoperability

4. **Tester:** An Individual who tests functional and non-functional requirements in the project.

Concerns	Quality Attributes
Is it easy to test the code?	Testability
Do the test cases cover all possible scenarios to validate the system?	Correctness

5. **Maintainer:** An Individual who adds additional functionalities or migrates the software from one platform to other.

Concerns	Quality Attributes
Is the system adaptable to the new environment?	Portability
Is it easy to update the system to a new version?	maintainability

## 2.4 Regulatory Authorities

1. **Health Canada:** Health Canada is responsible for helping Canadians maintain and improve their health[8].

Concerns	Quality Attributes
Easy verification of legality and authenticity of doctors and professionals.	Authenticity and non-repudiability

2. **RAMQ (RAMQ – Bureau d’accueil de Montréal):** The Régie de l’assurance maladie du Québec administers public health and drug insurance plans and pays health professionals[9].

Concerns	Quality Attributes
Information of user must be verified and validated.	Authenticity and non-repudiability

3. **Health Insurance Agency:** An organization that provides medical insurance to an individual[8].

Concerns	Quality Attributes
All the intense actions and services offered must be verified and validated.	Authenticity and non repudiability

## 2.5 Investors

1. **Trustee:** An individual person or member of a board given control or powers of administration of property in trust with a legal obligation to administer it is solely for the purposes specified.

Concerns	Quality Attributes
Is all available resources are properly utilized?	Resource Utilization
Is the project being delivered on time?	Integrity
Does the system protect from all unauthorized access?	Security

2. **Sponsors:** An individual who provides funds to the Project.

Concerns	Quality Attributes
Is all available resources are properly utilized?	Resource Utilization
Cost of the system development.	Affordability

## 2.6 Customer Support

1. **Customer Service Representative:** Interact with customers to handle complaints and provide information about system.

Concerns	Quality Attributes
Quality of the System.	Usability

2. **Technical Support:** Provide support to Individuals experiencing software problems of all kinds.

Concerns	Quality Attributes
Manage system maintenance.	Maintainability

## 3 Requirements

In this section, we present the **Architecturally Significant Requirements** only, since these requirements have impact on architectural design decisions.

Architecturally significant requirements are those requirements that have a measurable effect on a computer system's architecture.[3] Below is the list of architecturally significant requirements of iCare system. The requirement is represented as suggestion in part 5.2.4 in ISO/IEC/IEEE 29148:2018(E) standard[11].

The most important quality attributes (maintainability, privacy, security, and usability) was emphasised in **bold** text.

### 3.1 Non-Functional Requirements

**Requirement 1: The iCare system shall be a mobile phone accessible website accessible within Canada and other places.**

- Quality Attribute: **Usability**
- Sub-Quality Attribute: Accessibility
- Subject: iCare System
- Action: Being accessed

- Constraints of Action: Accessible on mobile phones and available anywhere

**Requirement 2: The back-end and front-end of iCare shall be independent.**

- Quality Attribute: **Maintainability**
- Sub-Quality Attribute: Testability, Modularity, Reusability
- Subject: iCare System
- Action: Develop back-end and front-end
- Constraints of Action: Independent with each other

**Requirement 3: Users must register and login to use iCare.**

- Quality Attribute: **Security**
- Sub-Quality Attribute: Authenticity
- Subject: User
- Action: Register and Login
- Constraints of Action: Valid user only

**Requirement 4: iCare system shall respond within 5 seconds.**

- Quality Attribute: Performability
- Sub-Quality Attribute: Time Behaviour
- Subject: iCare system
- Action: Response
- Constraints of Action: In 5 seconds

**Requirement 5: Doctors and Patients data must be encrypted, and only be decrypted for an authorized user.**

- Quality Attribute: **Privacy**
- Sub-Quality Attribute: Confidentiality
- Subject: User data
- Action: Encrypted
- Constraints of Action: Can be decrypted for an authorized user

**Requirement 6: System shall have more than 95 percent up-time.**

- Quality Attribute: Reliability
- Sub-Quality Attribute: Availability
- Subject: iCare System
- Action: Available

- Constraints of Action: Availability more than 95 percent

**Requirement 7: No more than 1 per 1000 system action should result in a failure such that the application needs a restart.**

- Quality Attribute: Reliability
- Sub-Quality Attribute: Maturity
- Subject: iCare System
- Action: Availability
- Constraints of Action: 999 success actions per 1000 actions

**Requirement 8: In the event of a breakdown, the system will allow a restart period of less than or equal to 60 seconds.**

- Quality Attribute: Reliability
- Sub-Quality Attribute: Recoverability
- Subject: iCare System
- Action: Restart/Availability
- Constraints of Action: Below 60 seconds

**Requirement 9: At least 10 users can access the application concurrently.**

- Quality Attribute: Performability
- Sub-Quality Attribute: Capacity
- Subject: iCare System
- Action: Access the system
- Constraints of Action: At least than 10 Users

### **3.2 Functional Requirements**

**Requirement 1: The user shall be able to book, cancel, and reschedule an appointment 1 week before the appointment date.**

- Quality Attribute: Usability
- Sub-Quality Attribute: Accessibility
- Subject: User
- Action: Book and Edit booking
- Constraints of Action: 1 week before the appointment date

**Requirement 2: Allow users to register to the iCare.**

- Quality Attribute: Usability

- Sub-Quality Attribute: Accessibility
- Subject: iCare System
- Action: Register
- Constraints of Action: Allow valid users

**Requirement 3: Allow registered users to login to the iCare.**

- Quality Attribute: **Security**
- Quality Attribute: Authenticity
- Subject: iCare System
- Action: Login
- Constraints of Action: Allow valid users

**Requirement 4: A user can try to login 5 times before getting blocked.**

- Quality Attribute: **Security**
- Sub-Quality Attribute: Authenticity
- Subject: User
- Action: Login
- Constraints of Action: Allow 5 times to try

**Requirement 5: Allow valid users to recover the password in a case when the user forgets the password.**

- Quality Attribute: **Usability**
- Sub-Quality Attribute: Operability
- Subject: iCare System
- Action: Recover Password
- Constraints of Action: Valid user

**Requirement 6: Allow registered patients to view Doctor's profile.**

- Quality Attribute: **Security, Usability**
- Sub-Quality Attribute: Authenticity, Accessibility
- Subject: iCare System
- Action: View Doctors' Profile
- Constraints of Action: User must be logged in

**Requirement 7: Allow registered users to view their appointments.**

- Quality Attribute: **Security, Usability**

- Sub-Quality Attribute: Authenticity, Accessibility
- Subject: iCare System
- Action: View Patient's Booking
- Constraints of Action: User must be logged in

**Requirement 8: Allow users to see the drug quantity.**

- Quality Attribute: **Usability**
- Sub-Quality Attribute: Accessibility
- Subject: iCare System
- Action: Website
- Constraints of Action: User must be logged in

**Requirement 9: Allow doctors to see the user's medical history.**

- Quality Attribute: **Usability**
- Sub-Quality Attribute: Accessibility
- Subject: iCare System
- Action: View the patient's profile
- Constraints of Action: Doctor must be logged in

**Requirement 10: The system shall allow the patient to view and edit their basic information.**

- Quality Attribute: **Usability**
- Sub-Quality Attribute: Accessibility
- Subject: iCare System
- Action: View & Edit Profile
- Constraints of Action: Must be logged in

**Requirement 11: The system shall allow the doctor to view and edit their basic information.**

- Quality Attribute: **Usability**
- Sub-Quality Attribute: Accessibility
- Subject: iCare System
- Action: View & Edit Profile
- Constraints of Action: Must be logged in

## 4 Viewpoints and Views

We followed the 4+1 View Model of Architecture [15] to describe the architecture views of iCare system. Each type of view serves specific stakeholders that have concerns described in that view. The "Scenario View" was replaced by "Use case view" as the model evolved in Rational Unified Process (RUP) [14].

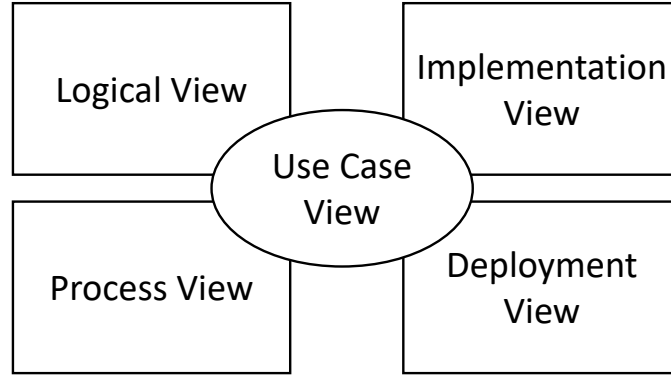


Figure 2: 4+1 View Model of Architecture: Version 2.

### 4.1 Use case views

This view is also known as Scenario View. In iCare application, there are several important use cases which are mentioned below.

#### 4.1.1 Use cases related to "User" stakeholder

The "User" stakeholder contains Doctors, Patients, Staff, 3rd Party users (other systems that use iCare data), Paramedics, and some negative stakeholders such as Hackers and Data Crawler.

- **Register:** User of iCare system must be able to register an account with username and password.
- **Login:** Registered user of iCare system must be able to login with registered username and password.

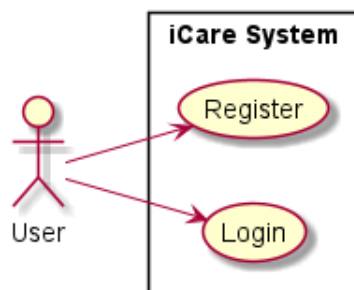


Figure 3: Use case diagram for Authentication and Registration.

- **View Doctor Profile:** Registered users of iCare system can view and edit their profile.

- **View Patient Profile:** Registered users of iCare system can view and edit their profile.

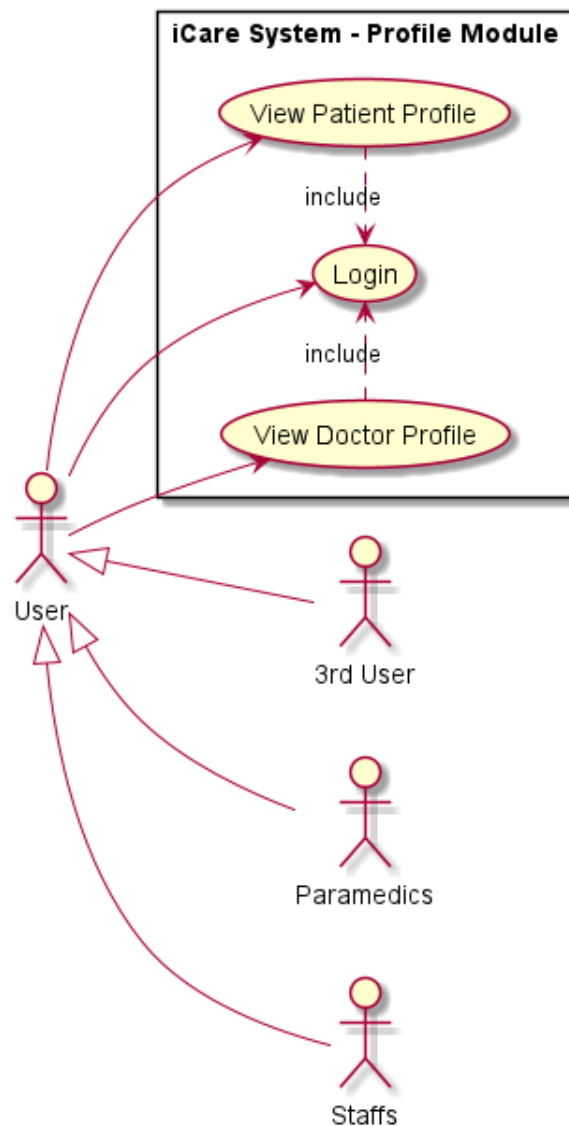


Figure 4: Use case diagram for view and edit user profile.

- **Book Appointment:** Registered patients can book an appointment.
- **View Appointments:** Registered user can view their upcoming appointments.
- **Cancel Appointments:** Registered patient can cancel their upcoming appointments with doctors.



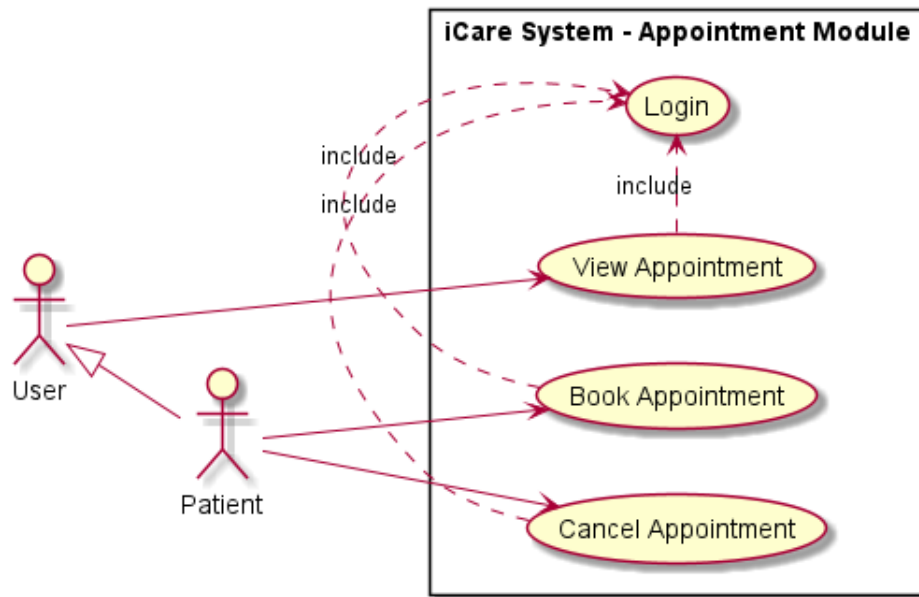


Figure 5: Use case diagram for appointment management.

- **Frequency Asked Questions:** Every user of iCare (registered or not) can view the frequency asked questions.

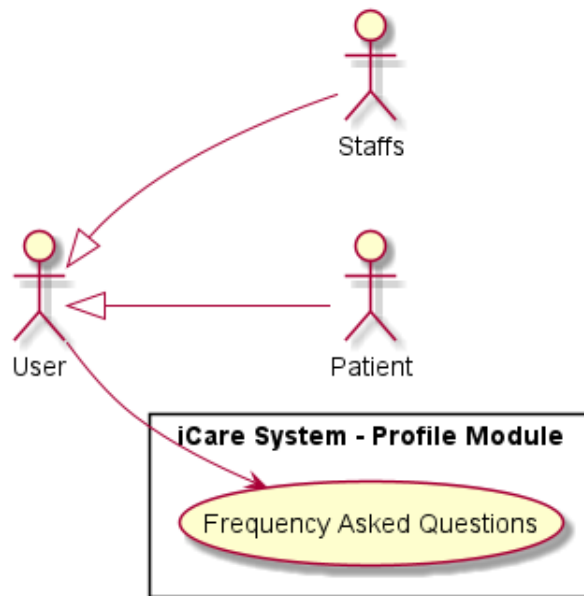


Figure 6: Use case diagram for Frequency Asked Questions.

#### 4.1.2 Other use cases

There are other use cases that are concerned with other stakeholders than "User", such as Health Insurance Agency, Government.

- **Claim Benefit:** Patient can edit their unprocessed claims. Also Health Insurance Agency and Patient can view processed claims.

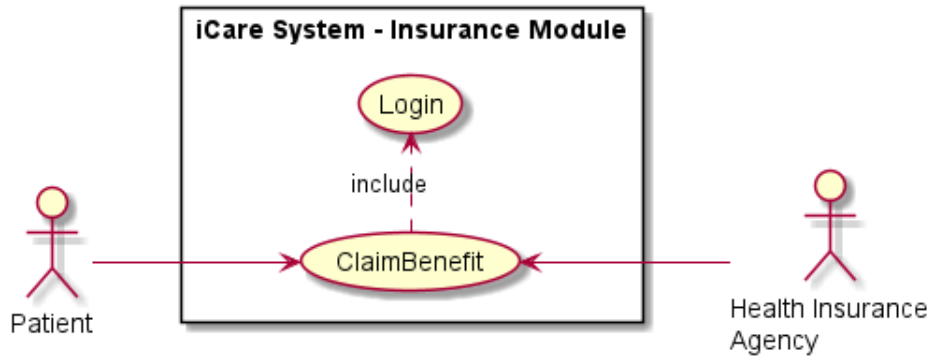


Figure 7: Use case diagram for Insurance.

- **Verify Authenticity:** Health Canada, RAMQ can verify doctor details.

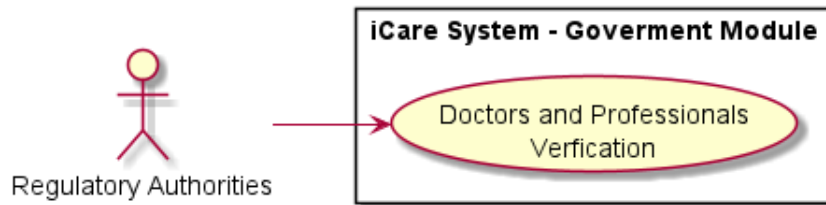


Figure 8: Use case diagram for Government.

## 4.2 Logical View

**Purpose:** This view focuses on functionality inspired by abstractions in the problem domain. It is responsible for the conceptual organization of layers and high-level functionality of components in each layer[12].

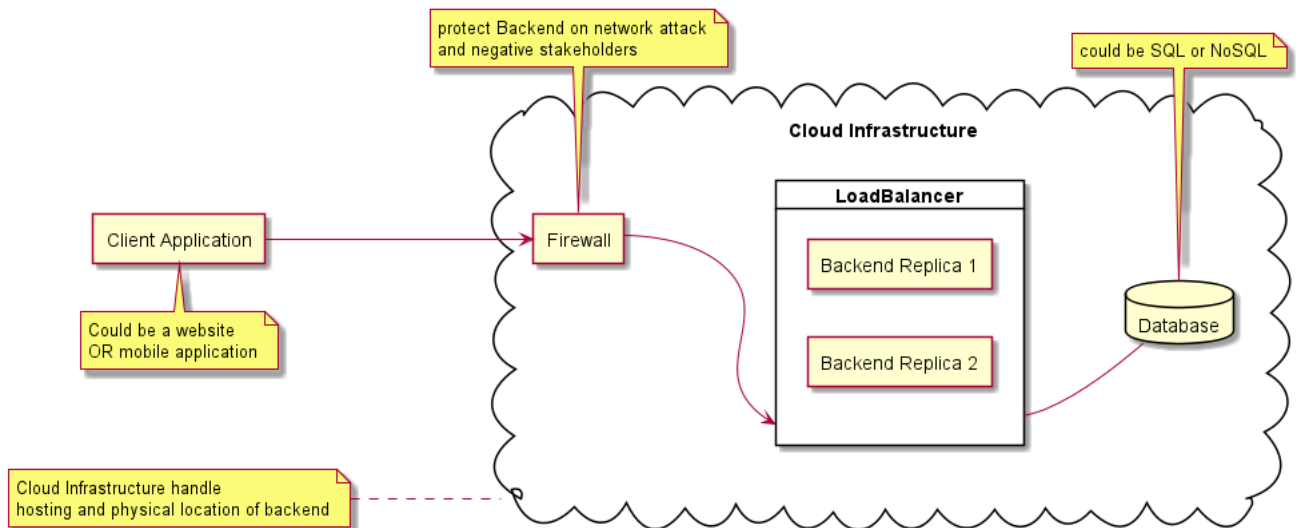


Figure 9: iCare components.

- **Client Application:** The application that consumes iCare system API. This application could be a website, a mobile application, or other systems using iCare system API.

- **Firewall:** Protect iCare backend against network attack such as Denial-of-service Attack (DDOS), spam, spoofing by negative stakeholders.
- **Load Balancer:** A service implemented by Cloud infrastructure to replicate the backend of iCare system and navigate request to the least busy replicated backend using a specific algorithm.
- **Cloud Infrastructure:** A service that provides hosting to host iCare systems, provide bandwidth and services (like Load Balancer service).
- **Database:** Database of iCare system. The database could be SQL or No-SQL, also hosted on Cloud Infrastructure, and have its own mechanism to backup, load balancing, and replicating.

### 4.3 Process View

**Purpose:** This view focuses on (concurrent) tasks performed by software at run-time. It is responsible for the process, threads, and control[12].

For each module of the iCare backend system, there is a corresponding process and can be represented by a sequence diagram. In this document, we present a sample process view of the Appointment Module.

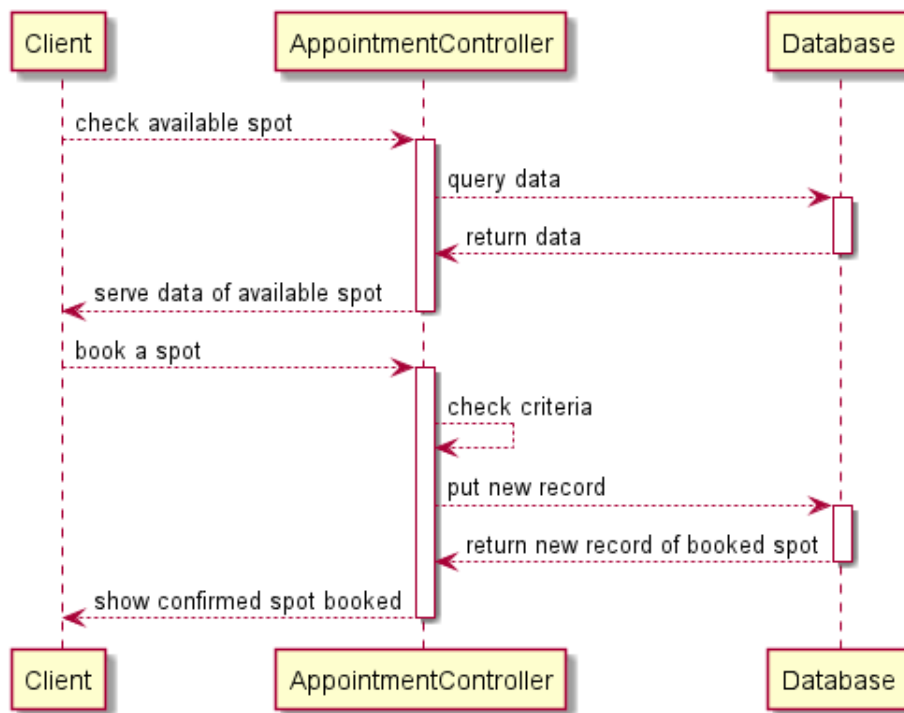


Figure 10: Interactions between iCare components when making an appointment.

At first, the client needs to login to be able to view or book an appointment. When logged in, the process as follow:

- **Step 1:** The client sends a check request to AppointmentController to check for an available spot.
- **Step 2:** The AppointmentController query the database and serve the data of available spot to the client as JSON.
- **Step 3:** Client sends a booking request to AppointmentController with selected spot.

- **Step 4:** AppointmentController check if the request satisfies all the criteria (the spot is not taken in the database, the spot is in the future, etc).
- **Step 5:** AppointmentController put a new record on the database for the booked spot.
- **Step 6:** AppointmentController waits for database return and sends confirmation to the client.

#### 4.3.1 Other Process views

This is another example of process view for Viewing and Updating user profile with Profile module.

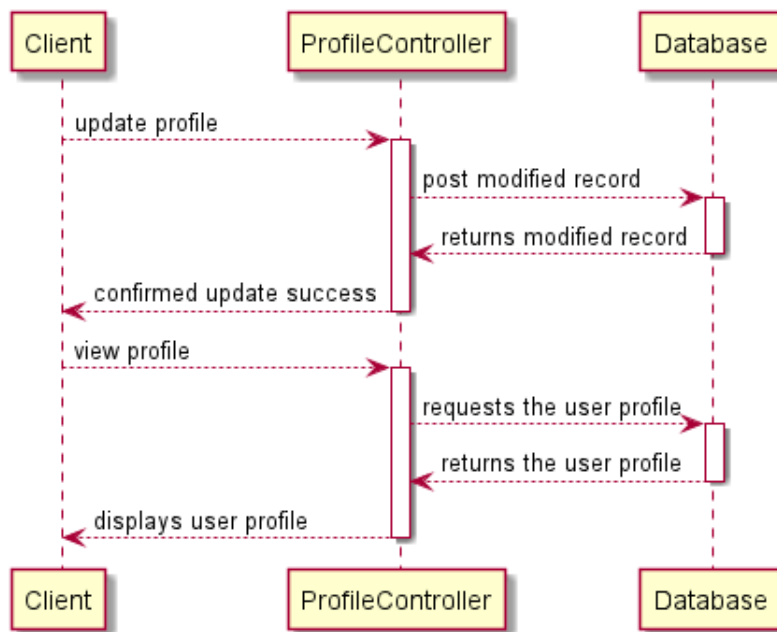


Figure 11: Interactions between iCare components when view or modifying user profile.

At first, the client needs to login to be able to view or book an appointment. When logged in, the process as follow:

##### Update User profile

- **Step 1:** The client sends a request to ProfileController to modify the profile.
- **Step 2:** ProfileController updates user details in the database. If successful, the database returns the modified data to the ProfileController
- **Step 3:** ProfileController send confirmation of update success to client.

##### View User profile

- **Step 1:** The client sends a request to ProfileController to view profile.
- **Step 2:** ProfileController sends a request to database.
- **Step 3:** ProfileController receive the queried data from Database and serve as JSON to client

## 4.4 Implementation View

**Purpose:** This view focuses on the actual source code, data files, and executable[12].

### 4.4.1 Overall Implementation

On the highest level of abstraction in implementation, iCare system is composed of some components:

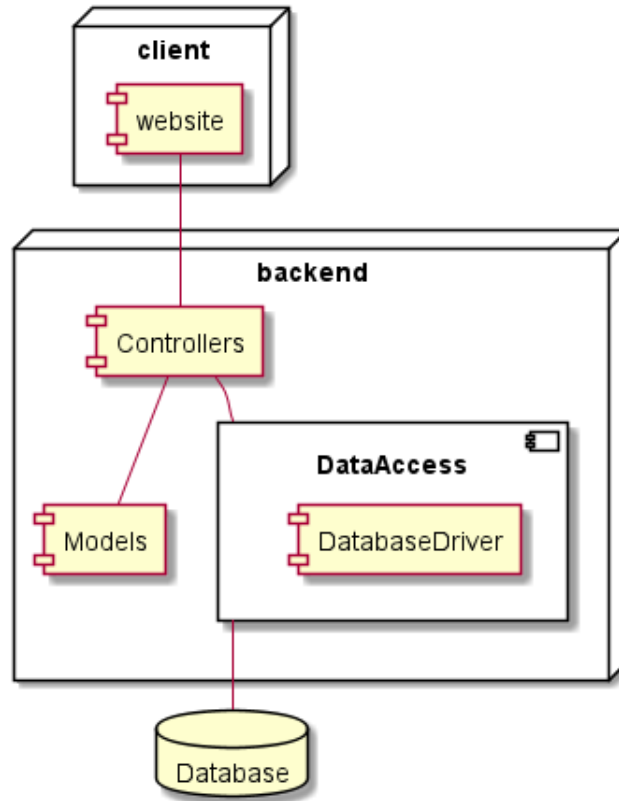


Figure 12: iCare implementation view.

The backend uses Models-Views-Controllers pattern with some modification. Because the backend of the iCare system is an API application, so there are no View components. Instead, each action in the controller will serve the response as a JSON string.

The Controller access database through a DataAccessLayer package. Inside this package, the implementer may include multiple DatabaseDriver components. This design allows changing database easily without re-writing the controllers.

### 4.4.2 Backend Implementation

In this section, we present a more detail implementation of iCare system backend

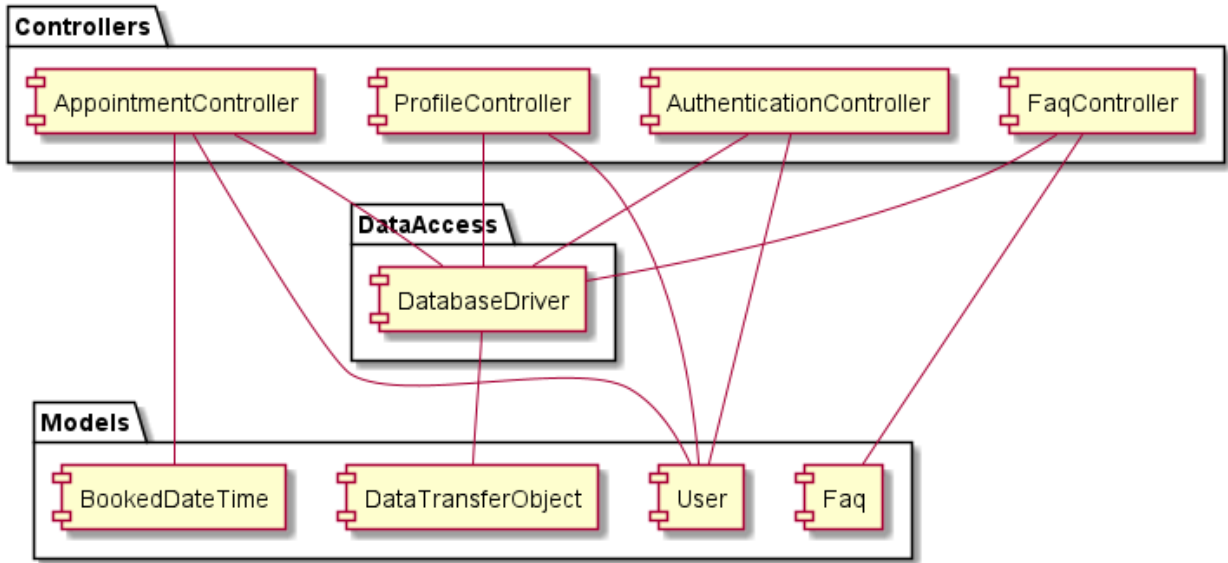


Figure 13: iCare system's backend implementation view.

The backend of iCare system composed of 3 primary packages: Controllers, DataAccess, and Models.

- **Controllers:** The request handlers for 4 main groups of the functionality of iCare, which are Authentication, Profile, Appointment, and serving static data (for now it is Frequency asked questions). These controllers handle client requests, connect to the database through DataAccess package, and serve the result as JSON.
- **DataAccess:** Handle all connections to and from the database. The implementer can organize classes in this package as each table in the database could have a corresponding service class in this package to separate concerns.
- **Models:** Contains the data transfer object and the metaphor objects of iCare system. Each model carries its own logic, represents the business logic of iCare system.

## 4.5 Deployment View

**Purpose:** This view focuses on the physical deployment of processes and components to processing nodes. It is responsible for the physical network configuration (topology) between nodes. This is also known as Physical View[12].

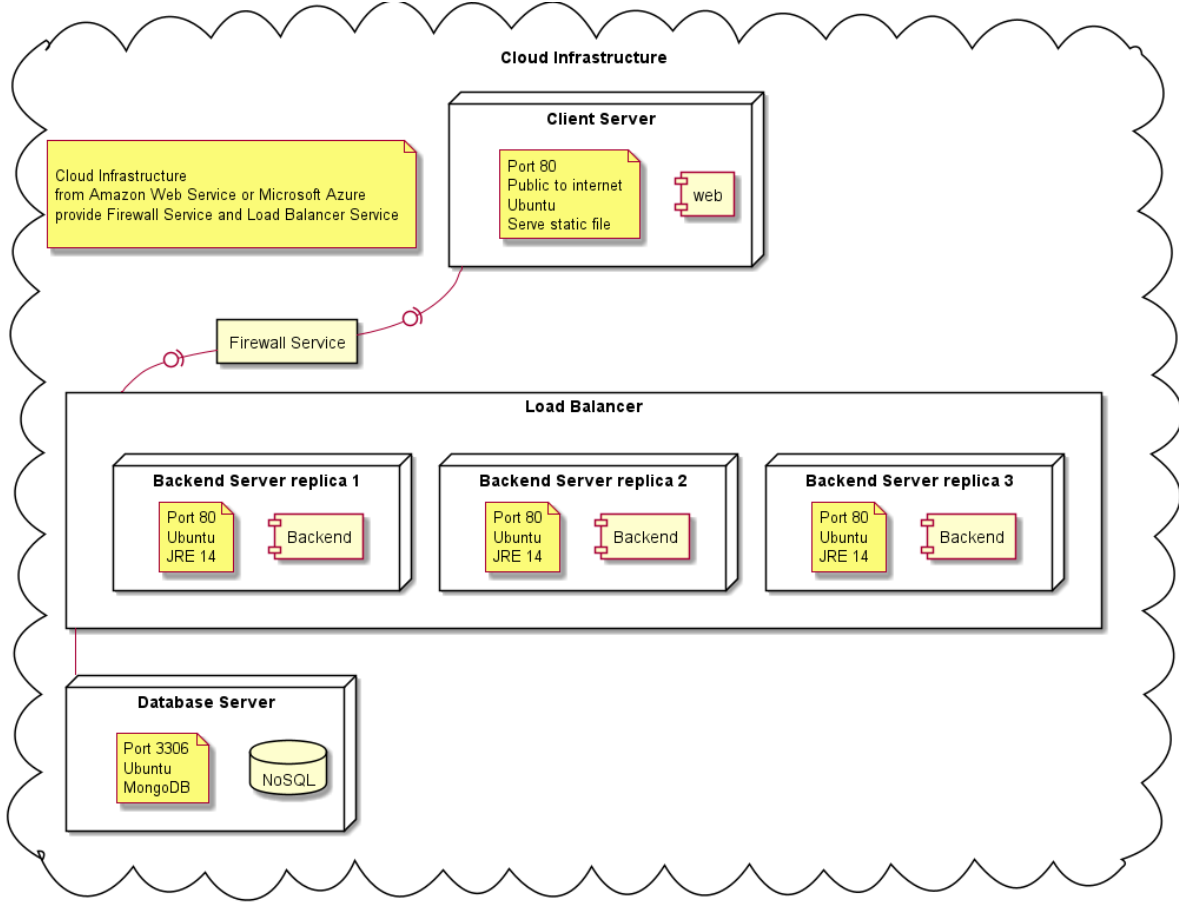


Figure 14: iCare system deployment view.

With the recent development of Cloud services such as Amazon Web Service or Microsoft Azure, iCare system can be hosted and deployed with minimal cost for physical machine and maintenance. Another thing is these cloud infrastructures provide security services such as Firewall and Load Balancer, save development cost.

The client-server only serve static file and does not contain any sensitive information. Therefore, the client-server may not need advanced protection from the firewall.

For testing purposes, DevOps can combine all of the servers in the deployment view in figure 14 into a single server running Ubuntu and necessary services.

## 5 Architectural Decisions

To present the architectural design decisions, we use a template introduced by Rik Farenhorst and Hans van Vliet in their research.[5]. Because the scale of this project, we do not use the field "Status", which is telling the approval status of the decisions by Project Manager.

<b>Related requirements</b>	The iCare system shall be a mobile phone accessible website accessible within Canada and other places.
<b>Alternative</b>	Mobile Application (Android and iOS).
<b>Constraints</b>	1. Group member techniques. 2. Given the time period.
<b>Assumption</b>	All computers, tablets, smartphones have the browser installed.

#### Architecture Decision

<b>Identifier</b>	#1
<b>Description</b>	iCare will be a web application, with responsive web pages.
<b>Rationale</b>	1. The client does not need to install, as long as there is a Web browser. 2. Team members are familiar with web application development. 3. Responsive web pages are accessible for all kinds of devices. 4. All PCs, tablets, mobile phones have web browsers installed.

Table 1: Architecture Decision 1 - Overall Architecture.

<b>Related requirements</b>	1. The back-end and front-end of iCare shall be independent. 2. iCare system shall respond within 5 seconds. 3. iCare should be maintainable.
<b>Alternative</b>	1. Back-end and front-end separated. 2. Back-end and front-end integrated.
<b>Constraints</b>	1. team member techniques. 2. Compatibility of front end technique with web browsers.
<b>Assumption</b>	All browsers support ECMAScript 5 or higher.

#### Architecture Decision

<b>Identifier</b>	#2
<b>Description</b>	iCare will be separated from front-end with back-end, using Asynchronous JavaScript and XML (Ajax) and JSON format to exchange data.
<b>Rationale</b>	1. Separation of front-end and back-end reduces the pressure of the server to render the pages, improve average response time. 2. Decoupling of front-end and back-end improves the maintainability of iCare. Separation of responsibility help maintainer debugging. 3. Improve Scalability of iCare. They don't bother each other when they are scaling. 4. Faster development. The development of front-end and back-end is also decoupled, they just follow defined API, don't have to wait for each other.

Table 2: Architecture Decision 2 - Separation of front-end and back-end.



<b>Related requirements</b>	iCare should be maintainable.
<b>Alternative</b>	<ol style="list-style-type: none"> <li>1. Java</li> <li>2. C#</li> <li>3. PHP</li> <li>4. node.js</li> <li>5. Python</li> </ol>
<b>Constraints</b>	<ol style="list-style-type: none"> <li>1. Maturity of techniques.</li> <li>2. Performance.</li> <li>3. Platform compatibility.</li> </ol>
<b>Assumption</b>	<ol style="list-style-type: none"> <li>1. The chosen framework should be free to use for both commercial and non-commercial purposes.</li> <li>2. The chosen framework should be familiar with the developer.</li> </ol>
<b>Architecture Decision</b>	
<b>Identifier</b>	#3
<b>Description</b>	iCare will choose Java as the back-end language.
<b>Rationale</b>	<ol style="list-style-type: none"> <li>1. Java is widely used in enterprise application, it has been proved to be a mature back-end language.</li> <li>2. Java has rich framework support for the server application.</li> <li>3. Java is known as platform independence.</li> <li>4. Java has a good performance in the server application.</li> </ol>

Table 3: Architecture Decision 3 - Back-end language.

<b>Related requirements</b>	iCare should be maintainable.
<b>Alternative</b>	<ol style="list-style-type: none"> <li>1. Native HTML, CSS, JavaScript.</li> <li>2. Reactjs</li> <li>3. Angular</li> <li>4. Vuejs</li> </ol>
<b>Constraints</b>	<ol style="list-style-type: none"> <li>1. Maturity of techniques.</li> <li>2. Performance.</li> <li>3. Platform compatibility.</li> </ol>
<b>Assumption</b>	<ol style="list-style-type: none"> <li>1. The chosen framework should be free to use for both commercial and non-commercial purposes.</li> <li>2. The chosen framework should be familiar with the developer.</li> </ol>
<b>Architecture Decision</b>	
<b>Identifier</b>	#4
<b>Description</b>	iCare will use Reactjs as front-end language.
<b>Rationale</b>	<ol style="list-style-type: none"> <li>1. Reactjs is developed and maintained by Facebook, therefore it is mature and well written.</li> <li>2. Reactjs has a rich extension and support.</li> <li>3. Reactjs (which will be compiled to native HTML, CSS and JavaScript) natively support all browsers, screen sizes.</li> <li>4. Reactjs has good performances and a minor learning curve.</li> </ol>

Table 4: Architecture Decision 4 - Front-end language and framework.

<b>Related requirements</b>	<ol style="list-style-type: none"> <li>1. The back-end and front-end of iCare shall be independent.</li> <li>2. iCare system shall respond within 5 seconds.</li> <li>3. The system shall have equal to or more than 95% uptime.</li> <li>4. iCare should be maintainable.</li> </ol>
<b>Alternative</b>	<ol style="list-style-type: none"> <li>1. Java Server Pages.</li> <li>2. Microsoft ASP.NET Website.</li> <li>3. RESTful API (with front-end application).</li> </ol>
<b>Constraints</b>	<ol style="list-style-type: none"> <li>1. The developer must be familiar with the chosen technology.</li> <li>2. The developer can be changed/replaced without affecting the development.</li> </ol>
<b>Assumption</b>	RESTful application is the first choice for almost any new website development with focus on data and services[18].

#### Architecture Decision

<b>Identifier</b>	#5
<b>Description</b>	iCare Back-end will implement RESTful API
<b>Rationale</b>	<ol style="list-style-type: none"> <li>1. RESTful is standard for API application.</li> <li>2. RESTful is natively supported by chosen back-end and front-end framework.</li> </ol>

Table 5: Architecture Decision 5 - RESTful API principle.

<b>Related requirements</b>	<ol style="list-style-type: none"> <li>1. The back-end and front-end of iCare shall be independent.</li> <li>2. iCare should be maintainable.</li> </ol>
<b>Alternative</b>	<ol style="list-style-type: none"> <li>1. Java Spring framework.</li> <li>2. JSP, servlet techniques.</li> <li>3. ASP.NET core.</li> </ol>
<b>Constraints</b>	<ol style="list-style-type: none"> <li>1. License and cost of the framework.</li> <li>2. Maturity of the framework.</li> <li>3. Extensibility of framework.</li> <li>4. Developer technique stack.</li> </ol>
<b>Assumption</b>	<ol style="list-style-type: none"> <li>1. The chosen framework should be free to use for both commercial and non-commercial purposes.</li> <li>2. The chosen framework should be familiar with developer.</li> </ol>

#### Architecture Decision

<b>Identifier</b>	#6
<b>Description</b>	iCare chooses Spring framework as the back-end technique.
<b>Rationale</b>	<ol style="list-style-type: none"> <li>1. Spring framework is the most popular Java framework, which means it is a mature technology.</li> <li>2. Spring framework has wonderful documentation and community support.</li> <li>3. Spring framework has a strong ability of integration. It can integrate many other technologies, e.g. SpringMVC, Mybatis, Hibernate, etc.</li> <li>4. Spring Boot provides a quick start of a Spring framework configuration, speed up the development.</li> </ol>

Table 6: Architecture Decision 6 - Back-end framework.

<b>Related requirements</b>	iCare should be maintainable.
<b>Alternative</b>	1. Models-Views-Controllers (MVC). 2. Non-MVC(e.g. JSP model 1 architecture)
<b>Constraints</b>	1. The developer must be familiar with the chosen technology. 2. The developer can be changed/replaced without affecting the development.
<b>Assumption</b>	1. Every web developer knows how to implement MVC pattern. 2. MVC pattern is familiar to both technical and non-technical stakeholders.

#### Architecture Decision

<b>Identifier</b>	#7
<b>Description</b>	iCare will choose MVC as the implementation principle, Spring MVC framework will be selected.
<b>Rationale</b>	1. MVC is the trend of developing web applications. 2. We may need multiple views for different users, MVC gives the flexibility of presenting views. 3. MVC enables logical grouping of related actions on a controller together. The views for a specific model are also grouped together[wiki]. 4. We have decided on the separation of front-end and back-end, which means views are independent. In this case, we would go further by dividing Models and Controllers, this is helpful for separation of responsibilities between operational logic and consistency of data.

Table 7: Architecture Decision 7 - MVC design pattern.

<b>Related requirements</b>	1. Doctors and Patients data must be encrypted, and only be decrypted for an authorized user. 2. At least 10 users can access the application concurrently.
<b>Alternative</b>	1. Relational database. 2. Document-based NoSQL. 3. Native XML database NoSQL.
<b>Constraints</b>	1. Big volume of patient data, medicine data, history treatment records, treatment history. 2. Performance of database should be good.
<b>Assumption</b>	1. The database technology is difficult to change. 2. There will be a lot of data (Gigabytes).

#### Architecture Decision

<b>Identifier</b>	#8
<b>Description</b>	iCare will use a Document-based NoSQL database.
<b>Rationale</b>	1. NoSQL is easy to do horizontal scaling across commodity servers, good for medical data scalability. 2. Queries in NoSQL databases can be faster than SQL databases. 3. Flexible data schema makes it easier to add or remove data relationships. 4. In [Madariaga, Ricardo & Muñoz, el], it concludes that NoSQL databases seem to be more appropriate than standard relational SQL databases when comes medical data[19].

Table 8: Architecture Decision 8 - Database type.

<b>Related requirements</b>	<ol style="list-style-type: none"> <li>1. The back-end and front-end of iCare shall be independent.</li> <li>2. iCare system shall respond within 5 seconds.</li> <li>3. The system shall have equal to or more than 95% up-time.</li> <li>4. iCare should be maintainable.</li> </ol>
<b>Alternative</b>	<ol style="list-style-type: none"> <li>1. Monolith application.</li> <li>2. Polyolithic software architectures (Micro-services).</li> </ol>
<b>Constraints</b>	<ol style="list-style-type: none"> <li>1. The cost to deploy and host iCare system.</li> <li>2. The cost of development and maintain Micro-service system.</li> </ol>
<b>Assumption</b>	<ol style="list-style-type: none"> <li>1. The service coverage of the hospital is limited to residents of a city (Montreal).</li> <li>2. The current requirements are not required to establish micro-service system.</li> </ol>

#### Architecture Decision

<b>Identifier</b>	#9
<b>Description</b>	iCare will be a monolith application.
<b>Rationale</b>	<ol style="list-style-type: none"> <li>1. For a hospital, it doesn't have a wide range of customers, multiple independent databases. The logic of use case is simple, straight forward, a monolithic application meets the needs.</li> <li>2. Monolith application is easy to be developed, deployed, debugged, tested.</li> </ol>

Table 9: Architecture Decision 9 - Monolithic vs Micro-service architecture.

<b>Related requirements</b>	<ol style="list-style-type: none"> <li>1. At least 10 users can access the application concurrently.</li> <li>2. System shall have equal to or more than 95% up-time.</li> </ol>
<b>Alternative</b>	<ol style="list-style-type: none"> <li>1. Cluster with load balancing.</li> <li>2. Single web server.</li> </ol>
<b>Constraints</b>	<ol style="list-style-type: none"> <li>1. The number of total users.</li> <li>2. Users' usage frequency of iCare.</li> <li>3. Scalability of iCare.</li> </ol>
<b>Assumption</b>	The cloud infrastructure will provide services for clustering, fire-wall, hosting and bandwidth.

#### Architecture Decision

<b>Identifier</b>	#10
<b>Description</b>	iCare will choose an extensible cluster with a load balancing solution.
<b>Rationale</b>	<ol style="list-style-type: none"> <li>1. A hospital in a city may don't have thousands of concurrency, but we do hope iCare to be able to extend its capacity, therefore, we may have a cluster in the future.</li> <li>2. The problem of the cluster includes load balancing, session synchronization, multiple server IP addresses, etc. Therefore, we need a reverse proxy solution for a cluster to solve multiple IP address problem, which we can introduce Nginx.</li> <li>3. It's easy to make use of Nginx technique to implement a simple load balancing schema.</li> </ol>

Table 10: Architecture Decision 10 - Clustering and load balancing.

## 6 Proof-of-Concept

### 6.1 Van Tuan Tran

Student ID: 40124288

Chosen architecturally significant requirements:

Requirement: The back-end and front-end of iCare shall be independent.

In part 5, Architectural Decisions, we already choose to implement iCare system with Browser/Server architecture (decision #1, #2). The back-end will be implemented using MVC design patterns with the help of Java Spring framework. The front-end will be implemented using Reactjs with basic components.

In this section, I will present the proof-of-concept that iCare back-end and front-end are independent, and can be developed separately and/or in parallel.

#### 6.1.1 Parallel Development

The proof-of-concept project was divided into two folder corresponding for front-end and back-end development. The front-end can be develop first with pseudo user interface and integrate with back-end API when the API is available.

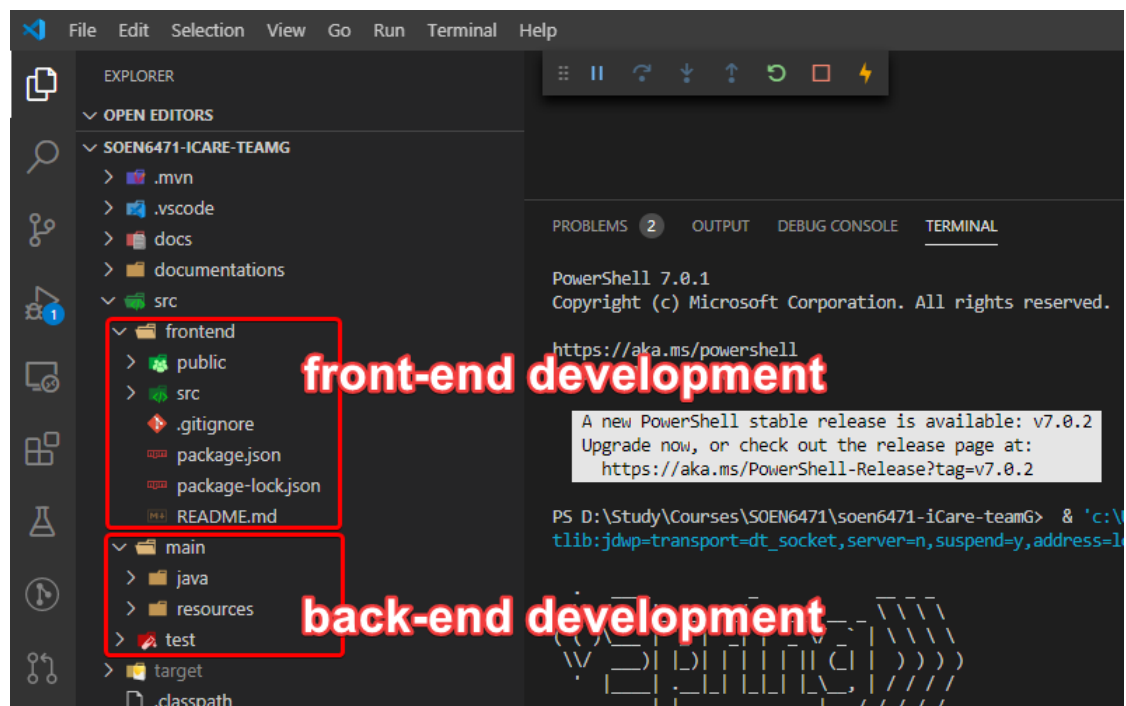


Figure 15: Front-end and Back-end development in parallel.

The Java Spring framework host the API application at default address <http://localhost:8080>. The Reactjs front-end framework host the web application at default address <http://localhost:3000>.

### 6.1.2 Result Demo

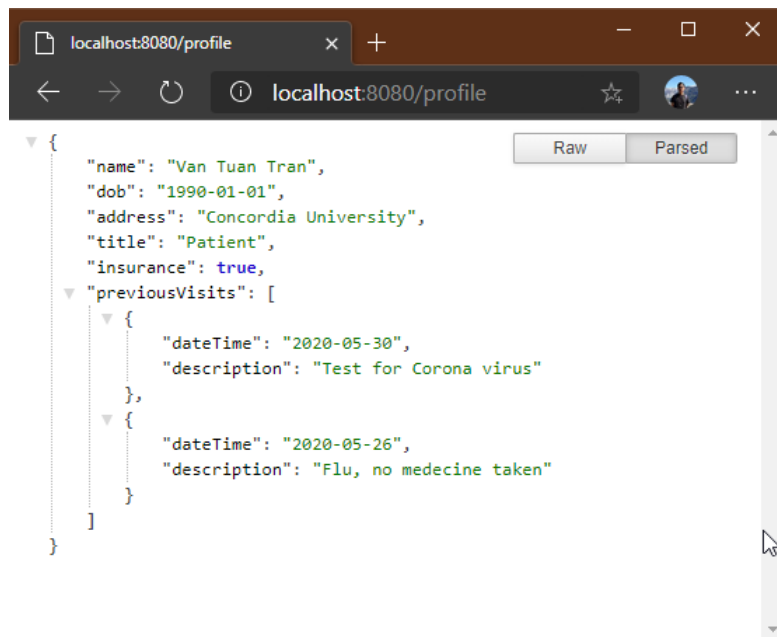


Figure 16: API provided by Java Spring Application.

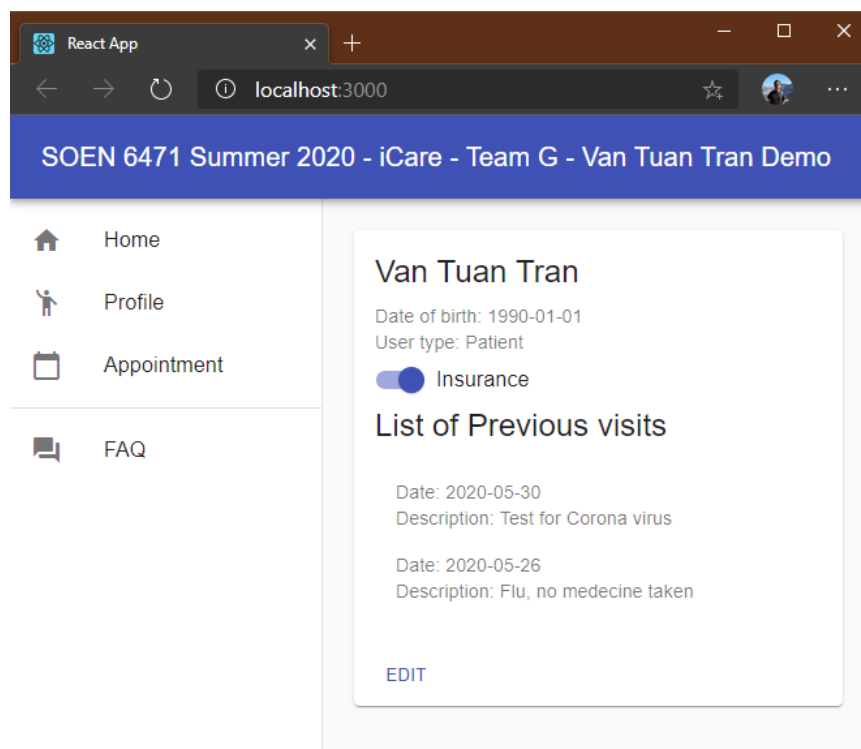


Figure 17: Web Application served by static HTML, CSS and JavaScript built from Reactjs.

The result of this implementation are presented in figure 16 and 17. The information presented in the JSON format was consumed by the web application and display in a user interface.

### 6.1.3 Source code

Source code for this implementation can be found at <https://github.com/huntertran/soen6471-iCare-teamG/tree/independent>.

This repository is in private mode. A copy of the source code can be downloaded at <https://drive.google.com/file/d/1VE5BLv6kPdjslnufV-GJtMPivJbyGU15/view?usp=sharing>.

## 6.2 Himen Hitesh Sidhpura

Student ID: 40091993

Chosen architecturally significant requirements:

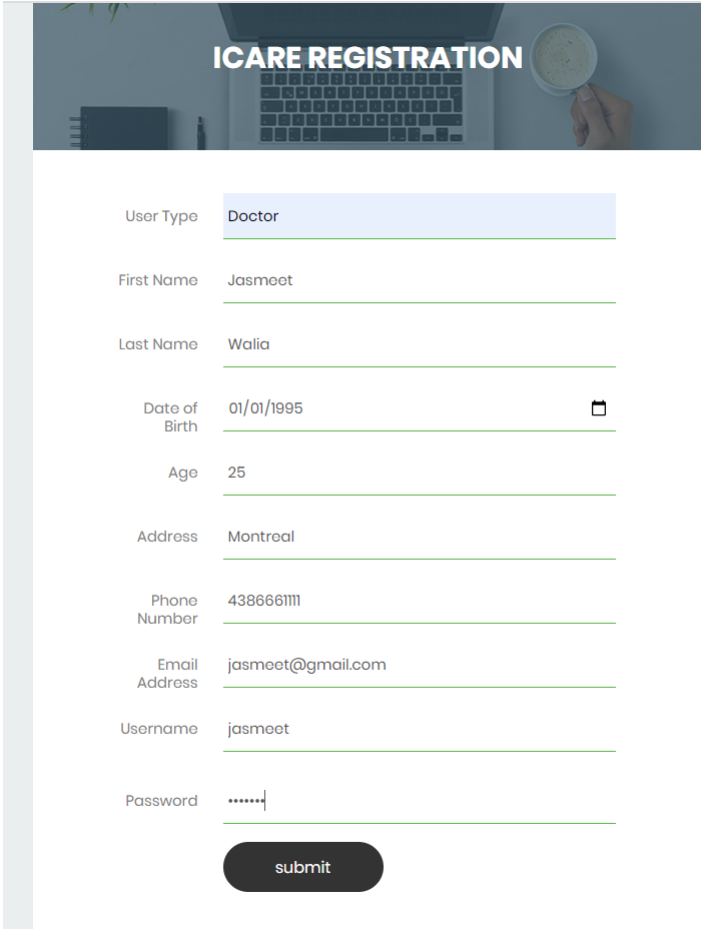
Requirement: Users must register and login to use iCare.

In this section, we will present the proof-of-concept that users can register and login in iCare Application to use iCare functionalities.

One of the main functionality is to allow doctors and patients to register to the iCare Application in order to allow accessibility. Moreover, only registered users (doctors and patients) are allowed for login otherwise an error message should prompt as invalid Username and password.

### 6.2.1 Result Demo

The result of the implementation are presented in figure 18, figure 19 and figure 20.

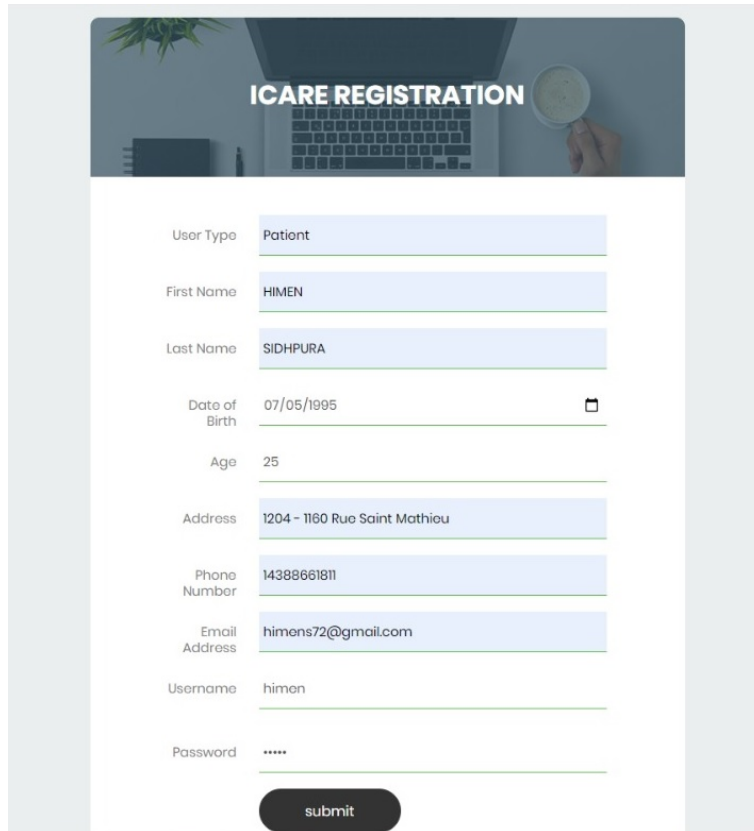


The screenshot displays the 'iCARE REGISTRATION' form for a doctor. The form is set against a background image of a desk with a laptop and a bowl of food. The registration details are as follows:

Field	Value
User Type	Doctor
First Name	Jasmeet
Last Name	Walia
Date of Birth	01/01/1995
Age	25
Address	Montreal
Phone Number	4386661111
Email Address	jasmeet@gmail.com
Username	jasmeet
Password	.....

A 'submit' button is located at the bottom of the form.

Figure 18: Registration Form for doctor in iCare Application.

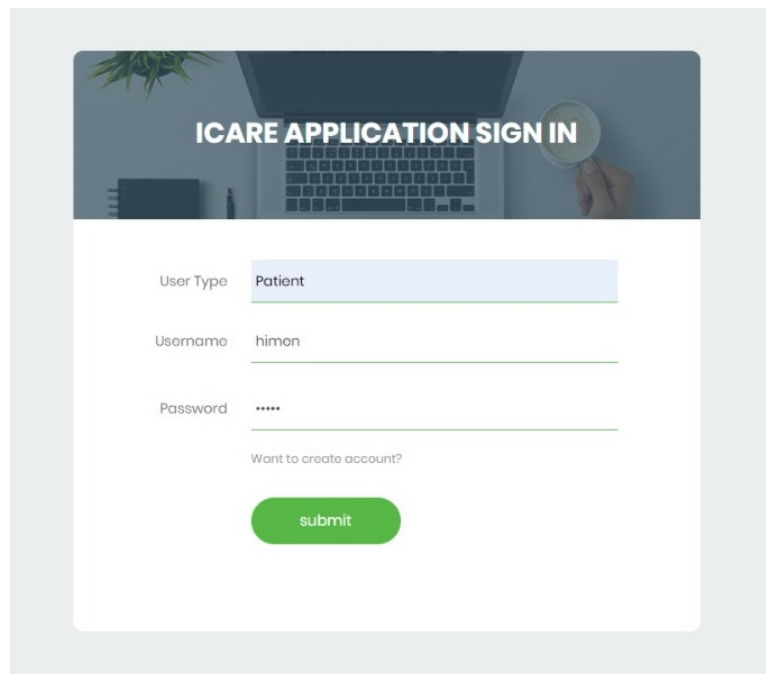


The image shows a registration form titled "ICARE REGISTRATION" with a header image of a laptop and a hand holding a magnifying glass. The form contains the following fields:

Field	Value
User Type	Patient
First Name	HIMEN
Last Name	SIDHPURA
Date of Birth	07/05/1995
Age	25
Address	1204 - 1160 Rue Saint Mathieu
Phone Number	14388661811
Email Address	himens72@gmail.com
Username	himon
Password	*****

submit

Figure 19: Registration Form for Patient in iCare Application.



The image shows a login page titled "ICARE APPLICATION SIGN IN" with the same header image as Figure 19. The form contains the following fields:

Field	Value
User Type	Patient
Username	himon
Password	*****

Want to create account?

submit

Figure 20: Login Page of iCare Application.



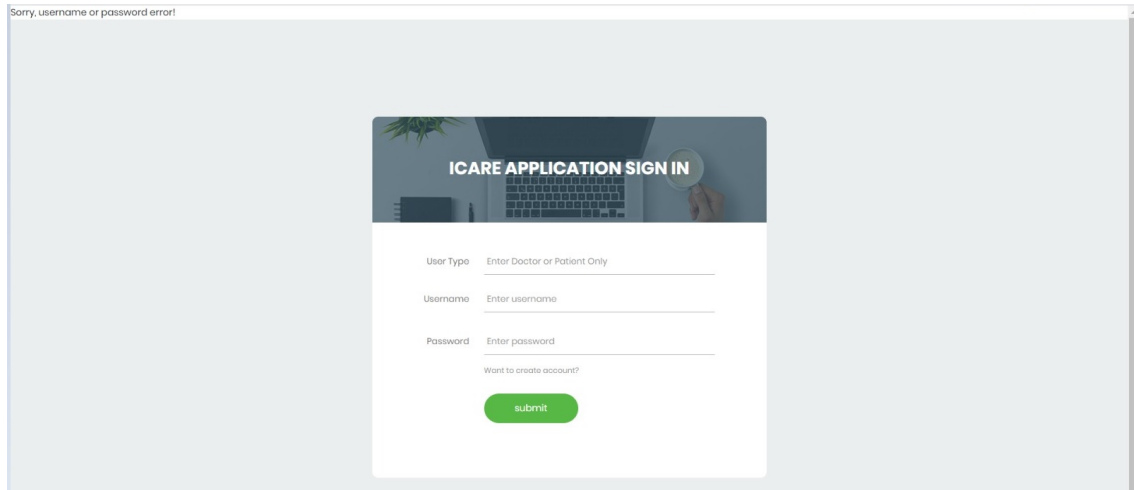


Figure 21: Login page showing wrong user & password.

### 6.2.2 Source code

Source code for this implementation can be found at

<https://drive.google.com/drive/folders/1A5sQDkRuCqUacbtgpeMg77MMbv583QEB?usp=sharing>

## 6.3 Jasmeet Walia

Student ID: 40151903

Chosen architecturally significant requirements:

Requirement: Mobile-friendly page of iCare Application.

Requirement: The system shall allow the doctor to view and edit their basic information.

In part 5, Architectural Decisions, we already choose to implement iCare system with Browser/Server architecture (decision #1) and iCare will be a web application, with responsive web pages as responsive web pages are friendly for all kinds of devices.

In this section, we will present the proof-of-concept that iCare Application is mobile-friendly allows flexibility to access the website from a mobile device, laptop, or tablet and it also allows the doctors to view and edit their basic information.

### 6.3.1 Result Demo

The result of the implementation are presented in figure 22, figure 23 and figure 24.

iPhone X 375 x 812 82% Online

## ICARE APPLICATION SIGN IN

User Type  
Enter Doctor or Patient Only

Username  
Enter username

Password  
Enter password

Want to create account?

submit

Figure 22: Mobile-friendly page of iCare Application.

## DR. JASMEET WALIA PROFILE

Edit Profile

User Type Doctor

First Name Jasmoot

Last Name Walia

Date of Birth 01/01/1995

Age 25

Address Montreal

Phone Number 4386661111

Email Address jasmoot@gmail.com

Username jasmoot

submit

Figure 23: Edit Profile page of Doctor Profile.

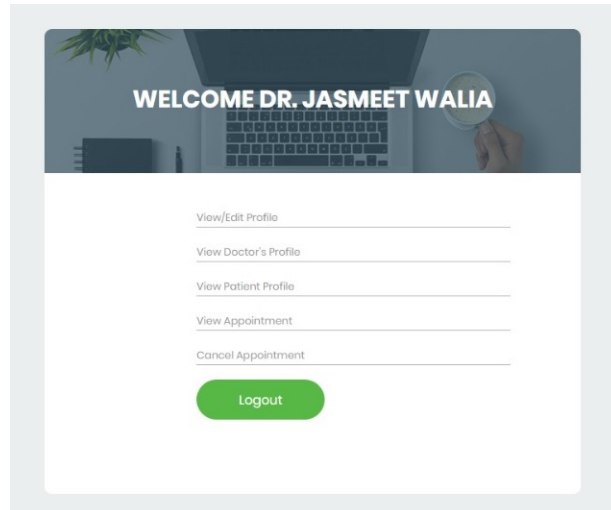


Figure 24: Home page of Doctor.

### 6.3.2 Source code

Source code for this implementation can be found at <https://drive.google.com/drive/folders/1A5sQDkRuCqUacbtgpeMg77MMbv583QEB?usp=sharing>

## 6.4 Yashwanth Vummaneni

Student ID: 40110312

Chosen architecturally significant requirements:

Requirement: iCare system shall respond within 5 seconds.

Requirement: The system shall allow the patient to view and edit their basic information.

In this section, we will present the proof-of-concept that iCare Application shall respond within 5 seconds, and also it allows the patient to view and edit their basic information. In the future, a Load Balancer can be implemented to accommodate more users to avoid delay during application usage.

### 6.4.1 Result Demo

The result of the implementation are presented in figure 25, figure 26 and figure 27. Once the Patient is successfully logged in, they can view and edit their profile.

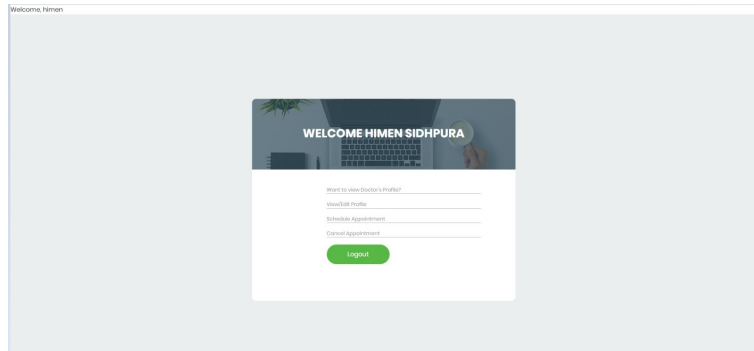


Figure 25: View profile page.

The screenshot shows a web application interface for editing a profile. At the top, there is a dark blue header with the text 'HIMEN SIDHPURA PROFILE'. Below this header, there is a green button labeled 'Edit Profile'. The main content area contains a form with the following fields and values: 'User Type' (Patient), 'First Name' (HIMEN), 'Last Name' (SIDHPURA), 'Date of Birth' (07/05/1995), 'Age' (25), 'Address' (1204 - 1160 Rue Saint Mathieu), 'Phone Number' (14388661811), 'Email Address' (himons72@gmail.com), and 'Username' (himen). At the bottom of the form, there is a green 'submit' button.

Figure 26: Edit profile page.

Figure 27: Patient profile page.

#### 6.4.2 Source code

Source code for this implementation can be found at <https://drive.google.com/drive/folders/1A5sQDkRuCqUacbtpeMg77MMbv583QEB?usp=sharing>

### 6.5 Haifeng Wu

Student ID: 40102956

Chosen architecturally significant requirements:

Requirement: The application should support automatic log-out, after a specific time of inactivity.

This requirement is regarding iCare's integrity and confidentiality, protecting user's information from being modified or stolen.

#### 6.5.1 Result Demo

In Spring framework, the interceptor is a component that does pre-handle, post-handle, after-completion for defined request. Figure 28 is showing how does it work.

In our case, in order to check whether the token(username) that requests is taking have a valid session, we require all request to go through this interceptor:

- If the request has a valid session, reset the session and session's inactive invalid time(In this demo, here is 5 seconds), then pass the request to corresponding controller.
- If the request has no valid session, redirect the request to login page.

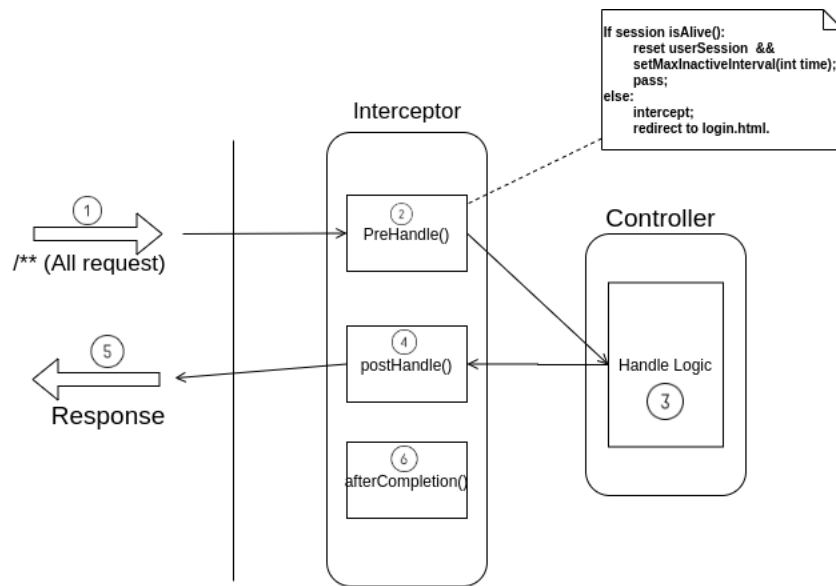


Figure 28: Interceptor and setting session in-active invalid time.

Figure 29 is showing a simple test:

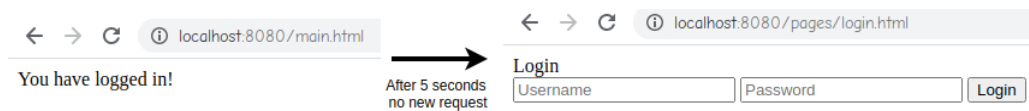


Figure 29: Automatic logout after 5 seconds' inactive.

### 6.5.2 Source code

Source code for this implementation can be found at

<https://drive.google.com/file/d/1ID1X8cptjyq1hKZAozFafDHgJhjp0dDp/view?usp=sharing>.

## A Appendix: Team Collaboration and Communication

Team G use a set of tools to collaborate on the project as well as keep up the communication.

### A.1 Collaboration

- **Google Spreadsheet:** Quickly note ideas and discussion on each part of the project. The google spreadsheet is one of the application in Google Drive, allow collaborator easily add content, comment or edit on other collaborator's content while keeping a history of all changes.
- **Overleaf:** Collaborate on writing the report with Latex. Normally, Overleaf free user only allow 2 collaborators on a single document. In the event of novel Corona virus in Summer 2020, Overleaf allow it's free users to upgrade to premium account to encourage work from home.
- **Github:** Github is a service profile hosting for source code using Git version control. Team G use Github for version control the Latex report (connected with Overleaf) and manage the project (store and share meeting minutes, assign tasks, proof-of-concept implementation). These Github project is in private mode, only collaborators can view and make changes.

The report project:

<https://github.com/huntertran/soen6471-iCareReport-teamG>

The management and source code:

<https://github.com/huntertran/soen6471-iCare-teamG>

### A.2 Communication

- **Zoom:** Team G use Zoom platform to schedule and meeting online. One limitation of Zoom is that free user only have a meeting in 40 minutes. We use this limitation to keep to meeting short and efficient. If the meeting require more than 40 minutes, we simply reuse the previous scheduled meeting URL.
- **Whatsapp:** Team G use Whatsapp chat service to share documents, demo and discuss when not in a Zoom meeting. Whatsapp provide end-to-end encryption for group chat, secured the chat content.

### A.3 Tools

- **Mind-mapping tools:** X-Mind.

<https://sourceforge.net/projects/xmind3/>

X-Mind originated as an open source Mind mapping tool to version 3.7. The later versions of X-Mind is commercialized.

- **UML tool:** Plantuml.

<https://plantuml.com/>

Plantuml is a powerful tool to draw UML diagrams, using a simple and human readable text description. The text description can be treated as any source code and be version controlled.

In this project, team G agreed to use UML version 2.5.1, adopted on December 2017.

## B Appendix: Glossary

- **Microservice:** A microservice is a cohesive, independent process interacting via messages.[4]
- **Microservice Architecture:** A microservice architecture is a distributed application where all its modules are microservices.[4]
- **Data Access Object (DAO):** In computer software, a data access object is a pattern that provides an abstract interface to some type of database or other persistence mechanism. Use a Data Access Object (DAO) to abstract and encapsulate all access to the data source. The DAO manages the connection with the data source to obtain and store data.[1]
- **Application Programming Interface (API):** An application programming interface (API) is a computing interface which defines interactions between multiple software intermediaries. It defines the kinds of calls or requests that can be made, how to make them, the data formats that should be used, the conventions to follow, etc. It can also provide extension mechanisms so that users can extend existing functionality in various ways and to varying degrees.[6]
- **Model-View-Controller (MVC):** MVC programming is the application of a three-way factoring, whereby objects of different classes take over the operations related to the application domain, the display of the application's state, and the user interaction with the model and the view.[13]
- **JSON format:** JSON (JavaScript Object Notation) is a data format based on the data types of the JavaScript programming language.[17]
- **NoSQL:** A NoSQL (originally referring to "non-SQL" or "non-relational") database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases. Such databases have existed since the late 1960s, but the name "NoSQL" was only coined in the early 21st century.[16]
- **Interoperability:** The ability of two or more systems or components to exchange information and to use the information that has been exchanged.[7]
- **Security:** degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization.[10]
- **Privacy:** The ability of an individual, group, or an organization to seclude themselves, or information about themselves, and thereby express themselves selectively.[20]
- **Maintainability:** degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers.[10]
- **Usability:** The degree to which a software product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction, in a specified context of use.[10]
- **Portability:** The degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another.[10]
- **Reliability:** The ability of a system or component to perform its required functions under stated conditions for a specified period of time.[10]



- **Performability:** The degree to which a software product can provide services, relative to the amount of resources used, under stated conditions.[10]

## References

- [1] ALUR, D., MALKS, D., CRUPI, J., BOOCH, G., AND FOWLER, M. *Core J2EE Patterns (Core Design Series): Best Practices and Design Strategies*. Sun Microsystems, Inc., 2003.
- [2] CALDIERA, V. R. B. G., AND ROMBACH, H. D. The goal question metric approach. *Encyclopedia of software engineering* (1994), 528–532.
- [3] CHEN, L., BABAR, M. A., AND NUSEIBEH, B. Characterizing architecturally significant requirements. *IEEE software* 30, 2 (2012), 38–45.
- [4] DRAGONI, N., GIALLORENZO, S., LAFUENTE, A. L., MAZZARA, M., MONTESI, F., MUSTAFIN, R., AND SAFINA, L. Microservices: yesterday, today, and tomorrow. In *Present and ulterior software engineering*. Springer, 2017, pp. 195–216.
- [5] FARENHORST, R., AND VAN VLIET, H. Experiences with a wiki to support architectural knowledge sharing. In *3rd Workshop on Wikis for Software Engineering (Wikis4SE)* (2008), Citeseer.
- [6] FISHER, S. Os/2 ee to get 3270 interface early. *Info World* 10, 34 (1988), 5.
- [7] GERACI, A., KATKI, F., MCMONEGAL, L., MEYER, B., LANE, J., WILSON, P., RADATZ, J., YEE, M., PORTEOUS, H., AND SPRINGSTEEL, F. *IEEE standard computer dictionary: Compilation of IEEE standard computer glossaries*. IEEE Press, 1991.
- [8] GOVERNMENT, C. Health canada. <https://www.canada.ca/en/health-canada.html>.
- [9] GOVERNMENT, Q. Ramq quebec. <https://www.ramq.gouv.qc.ca/>.
- [10] ISO. Iso/iec/ieee 25010:2011(e) - systems and software engineering — systems and software quality requirements and evaluation (square) — system and software quality models. Standard, International Organization for Standardization, Geneva, CH, 2011.
- [11] ISO. Iso/iec/ieee 29148:2018(e) - systems and software engineering — life cycle processes — requirements engineering. Standard, International Organization for Standardization, Geneva, CH, Mar. 2018.
- [12] KAMTHAN, P. P. Soen 6471.
- [13] KRASNER, G. E., POPE, S. T., ET AL. A description of the model-view-controller user interface paradigm in the smalltalk-80 system. *Journal of object oriented programming* 1, 3 (1988), 26–49.
- [14] KROLL, P., AND KRUCHTEN, P. *The rational unified process made easy: a practitioner's guide to the RUP*. Addison-Wesley Professional, 2003.
- [15] KRUCHTEN, P. B. The 4+ 1 view model of architecture. *IEEE software* 12, 6 (1995), 42–50.
- [16] LEAVITT, N. Will nosql databases live up to their promise? *Computer* 43, 2 (2010), 12–14.
- [17] PEZOA, F., REUTTER, J. L., SUAREZ, F., UGARTE, M., AND VRGOČ, D. Foundations of json schema. In *Proceedings of the 25th International Conference on World Wide Web* (2016), pp. 263–273.
- [18] RICHARDSON, L., AND RUBY, S. *RESTful web services*. ” O’Reilly Media, Inc.”, 2008.

- [19] SÁNCHEZ-DE MADARIAGA, R., MUÑOZ, A., LOZANO-RUBÍ, R., SERRANO-BALAZOTE, P., CASTRO, A. L., MORENO, O., AND PASCUAL, M. Examining database persistence of iso/en 13606 standardized electronic health record extracts: relational vs. nosql approaches. *BMC medical informatics and decision making* 17, 1 (2017), 123.
- [20] WIKIPEDIA. Privacy. <https://en.wikipedia.org/wiki/Privacy>.