

**SAAD DAHLEB UNIVERSITY OF BLIDA**  
**FACULTY OF SCIENCE**  
DEPARTMENT OF COMPUTER SCIENCE



**MASTER'S THESIS**  
**In Computer Science**

Option: Software Engineering

**TOPIC :**

**Software Product line for Healthcare  
Platforms compatible with IoMT**

Produced by  
Gherab Radouane

Supervised by  
Mme.Nesrine Lahiani

June 2023

## **ACKNOWLEDGEMENTS**

I first thank Allah who has given me the strength and courage to complete my studies and develop this modest work.

My sincere thanks go to my supervisor: Mrs.Lahiani Nesrine, for her unwavering support, expertise, and invaluable insights. Her guidance and mentorship have been instrumental in shaping this thesis and pushing me to explore new perspectives.

I would like to express my gratitude to all the people without whom these years of study would have been a pale reflection of the ones we have spent.

I would like to express my gratitude to my parents for their unconditional love, support and constant encouragement throughout this endeavor. Their belief in my abilities and their sacrifices have been the driving force behind my success.

## **Abstract**

The rapid advancement of healthcare technologies and the emergence of the Internet of Medical Things (IoMT) have introduced new challenges and opportunities in the development of healthcare platforms. Software product lines (SPLs) have gained significant attention as an effective approach for managing variability and complexity in software systems. In this thesis, we propose a Software Product Line for Healthcare Platforms compatible with IoMT.

By adopting SPL principles and techniques, we were able to achieve reuse, efficient management of variability, and customization. We present a comprehensive analysis of the healthcare domain and the IoMT landscape to identify commonalities and variabilities. Based on this analysis, we develop a feature model and a set of reusable assets that encapsulate the common and variable components of healthcare platforms.

After that, we created our ontology based on this feature model to reduce complexity of SPL approach. At the end, we derive an e-health platform from our product line to highlight the steps followed and how all these three big subjects work together.

### **Key words:**

Software Product Line, Variability, Feature Model, Ontology, IoMT, healthcare.

## Resumé

Les avancées rapides des technologies de santé et l'émergence de l'Internet des Objets Médicaux (IoMT) ont introduit de nouveaux défis et opportunités dans le développement de plates-formes de santé. Les lignes de produits logiciels (SPL) ont suscité une attention considérable en tant qu'approche efficace pour gérer la variabilité et la complexité des systèmes logiciels (SPL).

Dans cette thèse, nous proposons une ligne de produits logiciels pour les plateformes de santé compatibles avec l'IoMT. En adoptant les principes et techniques des SPL, nous avons pu réaliser la réutilisation, la gestion efficace de la variabilité et la personnalisation. Nous présentons une analyse complète du domaine de la santé et du paysage de l'IoMT afin d'identifier les similarités et les variabilités. Sur la base de cette analyse, nous développons un modèle de fonctionnalités et un ensemble d'éléments réutilisables qui encapsulent les composants communs et variables des plates-formes de santé.

Ensuite, nous avons créé notre ontologie basée sur ce modèle de fonctionnalités pour réduire la complexité de l'approche SPL.

À la fin, nous dérivons une plateforme de santé électronique à partir de notre ligne de produits pour présenter les étapes suivies et comment ces trois grands sujets fonctionnent.

### **Mot clés:**

Ligne de produits logiciels, variabilité, modèle de caractéristiques, Ontologie, IoMT, Santé.

## ملخص

تطورت تكنولوجيا الرعاية الصحية بسرعة هائلة وظهور إنترنت الأشياء الطبية قد أدى إلى تحديات وفرص جديدة في اهتمامًا كبيرًا كنهج فعال لإدارة التنوع (SPLs) تطوير منصات الرعاية الصحية. حققت سلاسل منتجات البرمجيات ( والتعقيد في أنظمة البرمجيات. في هذه الأطروحة، نقتراح سلسلة منتجات برمجية لمنصات الرعاية الصحية متوافقة مع ، تمكنا من تحقيق إعادة الاستخدام وإدارة التنوع الفعالة SPL إنترنت الأشياء الطبية. من خلال اعتماد مبادئ وتقنيات والتخصيص. نقدم تحليلاً شاملاً لمجال الرعاية الصحية والمشهد العام لإنترنت الأشياء الطبية لتحديد ما هو مشترك ومتغير. بناءً على هذا التحليل، نطور نموذج ميزة ومجموعة من الأصول القابلة لإعادة الاستخدام التي تجمع بين المكونات المشتركة . في SPL والمتغيرة لمنصات الرعاية الصحية. بعد ذلك، قمنا بإنشاء أونتولوجيا قائمة على نموذج الميزة لتقليل تعقيد نهج النهائية، استخلصنا منصة الرعاية الصحية الإلكترونية من سلسلة المنتجات لتعرض الخطوات التي تم اتباعها وكيفية تفاعل هذه المواضيع الثلاثة معًا.

### كلمات مفتاحية:

سلسلة المنتجات البرمجية، التغيير، نموذج الميزة، علم الأونتولوجيا، الرعاية الصحية.

---

# Contents

---

<b>List of Figures</b>	<b>VIII</b>
<b>List of Tables</b>	<b>X</b>
<b>Introduction</b>	<b>1</b>
Context: . . . . .	1
Problématique . . . . .	2
Objectives . . . . .	2
Thesis Organisation . . . . .	3
<b>1 State of art</b>	<b>4</b>
1.1 Software product line . . . . .	4
1.1.1 Introduction . . . . .	4
1.1.2 Definition . . . . .	4
1.1.3 Software product line engineering . . . . .	5
1.1.3.1 Domain engineering . . . . .	5
1.1.3.2 Application engineering . . . . .	6
1.1.4 Variability . . . . .	7
1.1.4.1 Variability management . . . . .	7
1.1.4.2 Representing variability . . . . .	8
1.1.5 Benefits and inconvenient . . . . .	10
1.1.5.1 Benefits . . . . .	10
1.1.5.2 Inconvenients . . . . .	10
1.2 Internet of things . . . . .	11
1.2.1 Introduction . . . . .	11
1.2.2 Definition . . . . .	11
1.2.3 Sensors . . . . .	12
1.2.3.1 Definition . . . . .	12

1.2.3.2	Types of sensors .....	12
1.2.4	IOT COMMUNICATION TECHNOLOGIES .....	13
1.2.4.1	Radio Frequency Identification (RFID) . . . . .	13
1.2.4.2	Wireless Sensor Network (WSN) . . . . .	13
1.2.4.3	WI-FI . . . . .	13
1.2.4.4	Bluetooth . . . . .	13
1.2.4.5	M2M . . . . .	13
1.2.5	Different process of : . . . . .	13
1.2.5.1	Identification: . . . . .	14
1.2.5.2	Sensing (Collection): . . . . .	14
1.2.5.3	Communication: . . . . .	14
1.2.5.4	Computation: . . . . .	14
1.2.5.5	Services (Function): . . . . .	14
1.2.5.6	Semantics: . . . . .	14
1.2.6	Internet of Things domain Applications .....	15
1.2.6.1	Smart Urban Cities.....	15
1.2.6.2	Smart Home System.....	15
1.2.6.3	Wearables.....	16
1.2.6.4	e-Health and IoMT.....	17
1.2.7	Application of IOT in Healthcare domain:.....	17
1.2.8	Benefits And inconvenient.....	20
1.2.8.1	Benefits .....	20
1.2.8.2	Inconvenient.....	20
1.3	Different applications of SPL in IOT .....	20
1.3.1	Work of: Angel Cañete , Mercedes Amor, Lidia Fuentes[12] .....	20
1.3.2	Work of: Inmaculada Ayala, Mercedes Amor, Lidia Fuentes and José.Troya[7] .....	20
1.3.3	Work of: ASAD ABBAS, ISMA FARAH SIDDIQUI, SCOTT UK-JIN LEE[3].....	21
1.3.4	Work of: Angel Cañete, Mercedes Amor, and Lidia Fuentes[13].....	21
1.4	Analysis: .....	22
1.5	Conclusion:.....	22
<b>2</b>	<b>Domain Engineering</b> .....	<b>23</b>
2.1	Introduction .....	23
2.2	Healthcare platforms in Algeria ?.....	23
2.3	SPL Development Process for the IoT .....	24
2.3.1	Domain Analysis.....	24
2.3.1.1	Use case Diagram: .....	25
2.3.2	Variability Modelling.....	26

2.3.3	Mapping Feature Model to Ontology.....	30
2.3.3.1	Transformation Rules: Converting Feature Model to Ontologie	31
2.3.3.2	Classes: .....	32
2.3.3.3	Relations: .....	32
2.3.4	Domain design: .....	33
2.3.5	Domain Implementation .....	34
2.4	Architecture IoT.....	34
2.4.1	Data collection: .....	36
2.4.2	Data evaluation: .....	37
2.5	Conclusion:.....	38
<b>3</b>	<b>Implementation</b>	<b>39</b>
3.1	Introduction: .....	39
3.2	Development tools:.....	39
3.2.1	Used Tools: .....	39
3.2.2	Back end:.....	40
3.2.3	Front end: .....	41
3.3	Application: .....	41
3.3.1	Analyses: .....	41
3.3.2	Home Page: .....	45
3.3.3	Patient Dashboard: .....	47
3.3.4	Risk Factors: .....	48
3.3.5	Patients’s Profile: .....	50
3.3.6	Doctor Dashboard: .....	51
3.3.7	Book an Appointment with a doctor: .....	51
3.3.8	Appointment Room:.....	52
3.4	Conclusion:.....	53
	<b>Conclusion and perspectives</b>	<b>54</b>
3.5	Conclusion:.....	54
	<b>Bibliography</b>	<b>55</b>



---

# List of Figures

---

1	Development processes in SPLE[11] . . . . .	5
2	Sample Car Model Feature Model . . . . .	8
3	Sample Cardinality Based Feature Model [31] . . . . .	9
4	Sample Orthogonal Variability Mode [27] . . . . .	9
5	Economics of software product line engineering [4] .....	10
6	IoT sensors .....	12
7	IOT elements [6] .....	15
8	Smart Home [32] .....	16
9	Smart Home [16] .....	17
10	SPL developing process. ....	24
11	Patient’s use case.....	25
12	Doctor’s use case.....	26
13	E-health Feature model. ....	27
14	Software Feature Model .....	27
15	API Feature Model .....	28
16	UserManagement Feature Model .....	28
17	Diagnostic Feature Model .....	28
18	Appointment Feature Model .....	28
19	Dashboard Feature Model .....	28
20	Prescription Feature Model .....	28
21	Data Feature model. ....	29
22	Hardware Feature model. ....	30
23	Classes of the proposed ontology .....	32
24	Relations of the proposed ontology .....	33
25	Class Diagram .....	34
26	Component Diagram .....	34
27	IoMT Architecture.....	35
28	Data Collection phase.....	37
29	Data Evaluation phase .....	38

30	E-Health Feature model.....	41
31	Data Feature model .....	42
32	Software Feature model.....	42
33	API Feature model.....	42
34	UserManagement Feature model.....	42
35	Diagnostic Feature model.....	42
36	Appointment Feature model.....	43
37	Dashboard Feature model.....	43
38	PrescriptionManagement Feature model.....	43
39	Hardware Feature model .....	44
40	Home Page.....	45
41	Patient Register Interface.....	46
42	Doctor Register Interface .....	47
43	Login Interface .....	47
44	Patient Dashboard .....	48
45	Count Risk Factors .....	49
46	Evaluate Patient .....	50
47	Update Patient's medical information.....	50
48	Doctor Dashboard .....	51
49	Patient book an appointment.....	52
50	e.g.: example of date in the past .....	52
51	Appointment's Room .....	53

---

# List of Tables

---

1. Medical Sensors .....	19
2. Comparison of Papers on Variability Management in IoT.....	21
3. From Feature model to Ontology .....	30
4. Cardiovascular evaluation risk factors .....	46

---

# Introduction

---

## Context

A Software Product Line (SPL) is useful for improving a family of software that shares common and differ in features and increasing the reusability of existing code. The reusability of these features makes it easier to control and update future software programs. In the healthcare industry, software programs focused on health require constant changes and enhancements due to the rapid advancement of technology. In the health sector, real-time data processing and communication between multiple devices enhance the efficiency of patient care.

Software engineering solutions in e-Healthcare architecture take advantage of the Internet of Medical Things (IoMT) to collect and integrate sensory data. Updating IoMT-based software applications to meet environmental requirements, such as installing heat sensors at various hospital branches (for patients receiving care both indoors and outdoors), should be done with minimal effort, expense, and loss of productivity. Feature modeling is the ideal paradigm for selecting feature requirements based on end-user specifications. Feature modeling involves managing SPL's core assets to ensure high reusability of existing features, which contributes to low-cost and rapid development of software applications. Feature models are tree-like structures composed of common and variable features (alternative, optional, and OR groups). Feature modeling improves the reusability of features[15] and facilitates data interchange, thereby enhancing the effectiveness of patient care and essential clinical activities, such as blood pressure monitoring.

Organizations use different strategies to develop IoT applications, one of which is the sequential strategy. This strategy requires developing each IoT application from scratch, leading to increased costs, time requirements, and quality issues. Additionally, if minor updates or modifications are needed, the entire application must be altered[14]. Therefore, this strategy is not suitable for situations where applications need to be produced quickly and cheap, such as in e-Health scenarios that require both indoor and outdoor sensors.

# Problematic

IoMT (Internet of medical things) is a promising paradigm due to the growing range of connected devices, defined as “things”. Managing and modelling these “things” continues to be taken into consideration as an assignment. Tackling this problem can be easier with the aid of software product line (SPL) paradigm and the variability Management (VM) activities. SPL engineering consists of mechanisms that offer identity, illustration, and traceability, which may be helpful to “things” management supported by using VM organizational and technical activities.

Thorough review of existing studies in the fields of SPL and IoMT In order to draw some conclusions about the limits of this area. This leads us to ask ourselves some crucial questions, which are:

1. How are software product lines (SPL) being applied in the context of the Internet of Medical Things (IoMT) applications?
2. How is the variability management (VM) of the SPL is adopted in IoMT devices?
3. Which processes, frameworks or platforms use SPL in IoMT structures?

# Objectives

The goal of our work is to design and develop a product line that allows for:

- Analyze and identify possible procedures, frameworks, or systems that apply SPL standards across exclusive IoMT structures domains.
- Identify similarities and variability between Health applications.
- Develop the domain core assets (reusable element):
  - A domain ontology
  - The domain modeling diagrams.
  - The reusable component.
- Develop e-Health applications based on core assets.
- Increase the reusability of software elements based on a component-oriented approach.
- Develop specific components for final applications if available.

# Thesis Organization

To carry out our thesis, we have organized our work into three chapters.

- **State of Art:** In this chapter, we provide an overview of the SPL approach and IoMT (Internet of Medical Things). We explore the concept, benefits, challenges, and best practices of software product lines. It also discusses the role of IoT and IoMT in domains like healthcare, smart homes, and industrial automation. The chapter highlights few studies that build IoT applications and the challenges they faced.
- **Domain Engineering:** In this chapter, we provide a comprehensive overview of the analysis and design process employed in developing our application. We break down the steps involved, from requirements gathering to system architecture design. By following this systematic approach, we ensure a well-planned and structured application that aligns with the intended objectives.
- **Implementation:** In the final chapter, we specifically describe the environment in which we programmed our application, the programming languages used, and we conclude with the description of the main interfaces of the application.

# Chapter 1

---

## State of art

---

### 1.1 Software product line

#### 1.1.1 Introduction

Healthcare has undergone significant transformations in recent years, largely due to the emergence of the Internet of Medical Things (IoMT). This technology enables the collection of vast amounts of health data from various devices and platforms, allowing healthcare providers to offer more personalized and effective treatment options. However, developing software solutions that are compatible with the diverse healthcare platforms and devices can pose significant challenges. One promising approach to address this challenge is the concept of a software product line (SPL). In this context, an SPL can be utilized to develop a suite of software products that are compatible with IoMT devices and platforms. The aim of this thesis is to explore the application of an SPL in the healthcare and IoMT context, and to investigate its potential benefits in terms of reducing development costs, improving time-to-market, and ensuring consistency across software products. This introduction provides an overview of the topic and sets the stage for the subsequent chapters, which will delve deeper into the specifics of the software product line for healthcare platforms compatible with IoMT.

#### 1.1.2 Definition

A software product line is a group of software that shares some core assets while differing in implemented features. The term was first proposed in 1976 by David L. Parnas, who defined it as sets of programs whose common properties are so extensive that it is advantageous to study the common properties of the programs before analyzing individual members [25]. This definition highlights the presence of both commonality and variability, which are two important aspects of a software product line. Identifying commonality is a crucial phase in the SPL approach. If the software are completely different, it is difficult to refer to them as a family of software, and we cannot reap the benefits of applying a software product line. Clement defined

The software product line as a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way[14]. Software product line engineering allows a company to shift its focus from the development and evolution of individual software to the entire software product family. One advantage of this approach is that it becomes easier to identify relationships between software, their commonalities, and variabilities. This is essential for meeting market needs and facilitating the reuse of development artifacts.

### 1.1.3 Software product line engineering

Software Product Line Engineering (SPLE) is a software engineering paradigm that guides companies in developing software applications from core assets, rather than starting each development process from scratch. The life cycle of SPL development typically involves two main processes: domain engineering and application engineering [22].

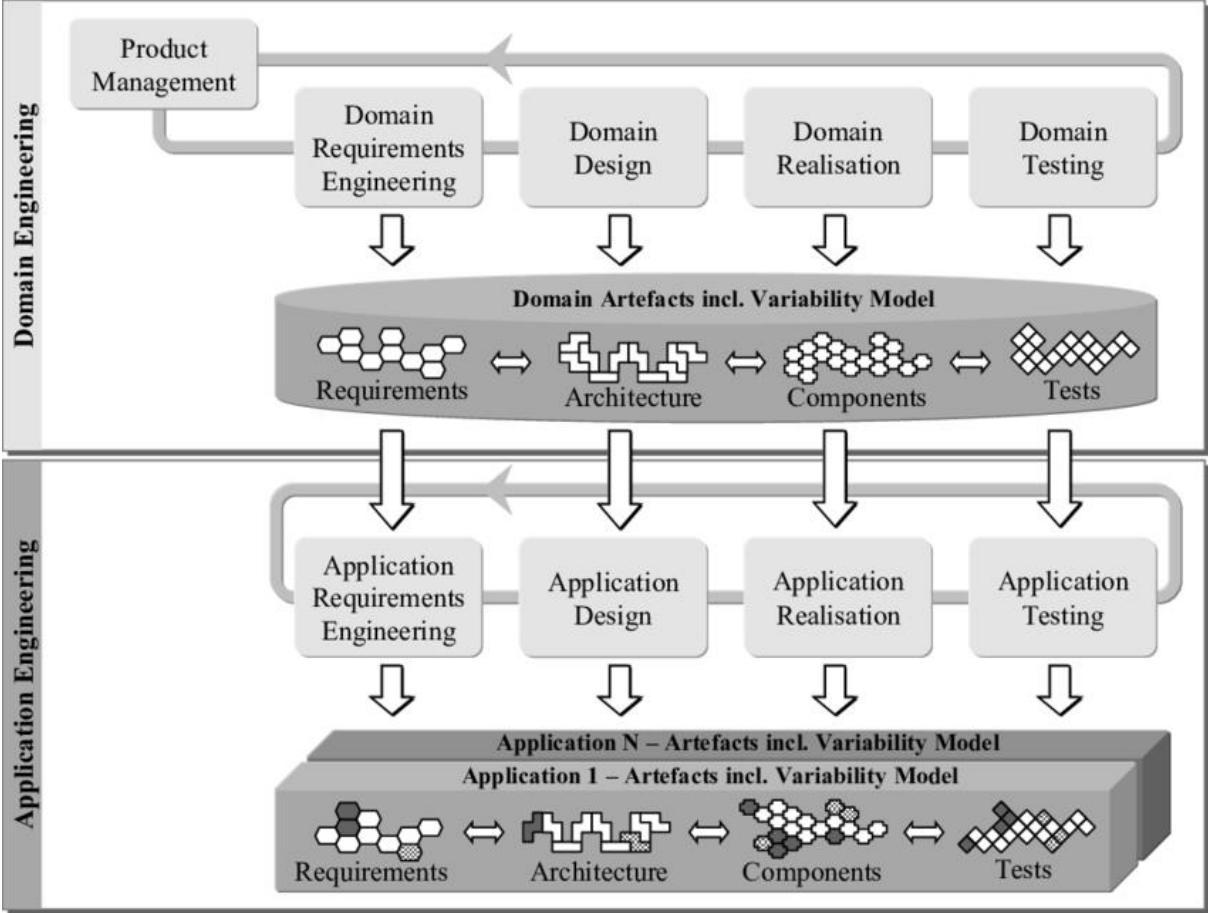


Figure 1: Development processes in SPLE[11]

#### 1.1.3.1 Domain engineering

During this phase, our primary objective is to discern the shared characteristics and differences within the system and generate the corresponding domain artifacts that manifest them.



These domain artifacts comprise requirements artifacts, such as Use case diagrams, architectural artifacts like class diagrams, implementation artifacts encompassing source code, files, and libraries, as well as test artifacts, including test cases. The domain requirements and product line architecture are pivotal elements of the product line platform, with the latter serving as the reference architecture. The objectives of this phase encompass precisely defining the shared characteristics and differences present in the software product line, identifying potential variations and creating reusable artifacts that facilitate the construction of the desired applications[22].

- **Product management:** The main purpose of product management in SPLE is product line scoping. One aspect of product line scoping is the definition of a set of applications offered by organizations for a certain market. Other aspects usually include the reusable features and artifacts. If the domain is large, the effort to realize domain artifacts may increase because they become generic. On the other hand, if the domain is narrow, the desired features by customers may not be covered. Due to the importance of this phase, it has to be taken care of by technical experts and senior developers.
- **Domain requirements engineering:** Domain requirements engineering sub process has all the activities of extracting and documenting the commonalities and variabilities of the product line. In addition, we define constraints to achieve stability and flexibility.
- **Domain design:** Domain design sub-process defines the reference architecture of the software product line. Numerous methods are available for modeling variability. It provides high-level structure for all product line applications.
- **Domain realization:** Domain realization sub process handles how to design and implement reusable software components.
- **Domain testing:** Domain testing sub process is responsible for verification and validations of reusable components. The validations are done by testing the components with their specifications. In addition, it develops reusable test artifacts for application testing.

### 1.1.3.2 Application engineering

The goal of application engineering phase is to derive objects from domain artifacts, thereby defining the specific features and requirements for the application. In a broader sense, the applications should fulfill their intended purposes by leveraging variability and capitalizing on commonalities. In practical terms, it is not always feasible to completely achieve end-user-specific requirements by reusing domain artifacts alone. In such cases, these specific application requirements must be characterized as application-specific variations and documented within the application variability model. The primary objectives of this phase encompass the development of an application that optimizes the reuse of domain assets and maximizes the utilization of

commonalities and variabilities throughout the development process[34].

- **Application requirement engineering:** Application requirement engineering sub process include all the specifications to analyze, develop and validate the requirement application.
- **Application design:** Application design selects and configures the required parts of the reference architecture and combines it with the specifications.
- **Application realization:** Application realization creates the application taking into consideration the configuration of reusable software parts and the application specific assets.
- **Application testing:** In this step, we verify and validate the application against its specification.

### 1.1.4 Variability

Variability is a critical aspect in any SPL approach, Rommes and al said: It covers the whole life cycle. It starts with the early steps of scoping, covering all the way to implementation and testing and finally going into evolution.[29]. Variability is the main reason why two products from the same product line look different, for example in terms of properties, features, and functions. Variability enables the development of customized products by reusing predefined and adjustable artifacts. It is defined by Bachmann and Clements as: variability means the ability of a core asset to adapt to usages in the different product contexts that are within the product line scope.[9] by this definition we see that the variations in a product line are predicted. The developers of core assets have thought about the consequences of the variations and obliged them in a way that the results of core assets support the requirements taking into consideration the time and budget. When dealing with variability in software product line engineering we can identify 3 main types: Commonality: common features in all the software product family. Variability: features that distinguish applications of the same family. Product-specific: elements that are unique to a particular software. These types are not constant during the SPL life cycle. They may change but no matter how do they vary they will always be one of these three.

#### 1.1.4.1 Variability management

Managing variability becomes easier by answering the following questions: What does vary? Why does it vary? And how does it vary? The first question, “what does vary” is used to identify the specific points of variability in the software product line. The second question, “why does it vary” highlight the reason and the cause. The third question “how does it vary” is used to enumerate variations of a specific variability point. By answering these questions, software product line engineers can understand and manage variability in a product line more effectively. These are the four main activities involved in variability management:

1. Variability identification: Involves identifying the product differences and their location

within the Product Line artifacts.

2. Variability delimitation: Defines the binding time and multiplicity.
3. Variability implementation: Involves the selection of implementation mechanisms.
4. Variant management: Controls the variants and variation points.

#### 1.1.4.2 Representing variability

Several modeling techniques exist to support and assist developers in representing variability within a product line. Each technique is suitable for specific situations, and while there is no single optimal technique, one popular approach is “Feature Model”. Feature model is based on the Feature-Oriented Domain Analysis (FODA) analysis method.

- **Feature-Oriented Domain Analysis:** is a popular method for domain analysis, the Software Engineering Institute (SEI) develops it. In 1990, Kang et al introduced FODA concepts and introduced feature modeling. Feature model models variabilities and commonalities using feature diagrams of a software product family.[21]
- **Feature model (FM)** is a simple method compared to other complex techniques such as unified modeling language (UML). Feature model represents all software systems in a product family in terms of features. Feature model is a tree graph (tree of features). In which, we identify features as nodes and edges representing relations between features.

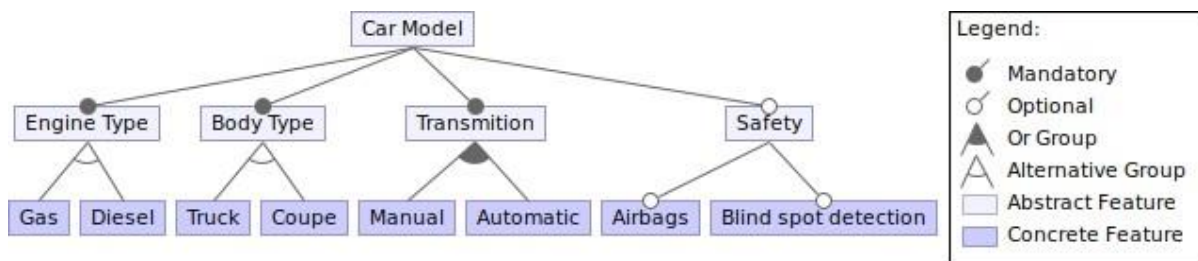


Figure 2: Sample Car Model Feature Model

- **Cardinality-Based Feature Modeling (CBFM):** Cardinality-based Feature Modeling is a hierarchical feature model. It comes as a way to optimize the reuse of features and commonalities (core assets). CBFM extends feature model by adding the following:
  1. Cardinality: shows how many variations a feature can have.
  2. Feature groups: helps to organize features and shows group members.
  3. Attribute types: allows specifying values during configuration.

4. Feature model references: allows tackling big feature models by splitting them to small models.
5. Object Constraint Language (OCL): enables the description of constraints.

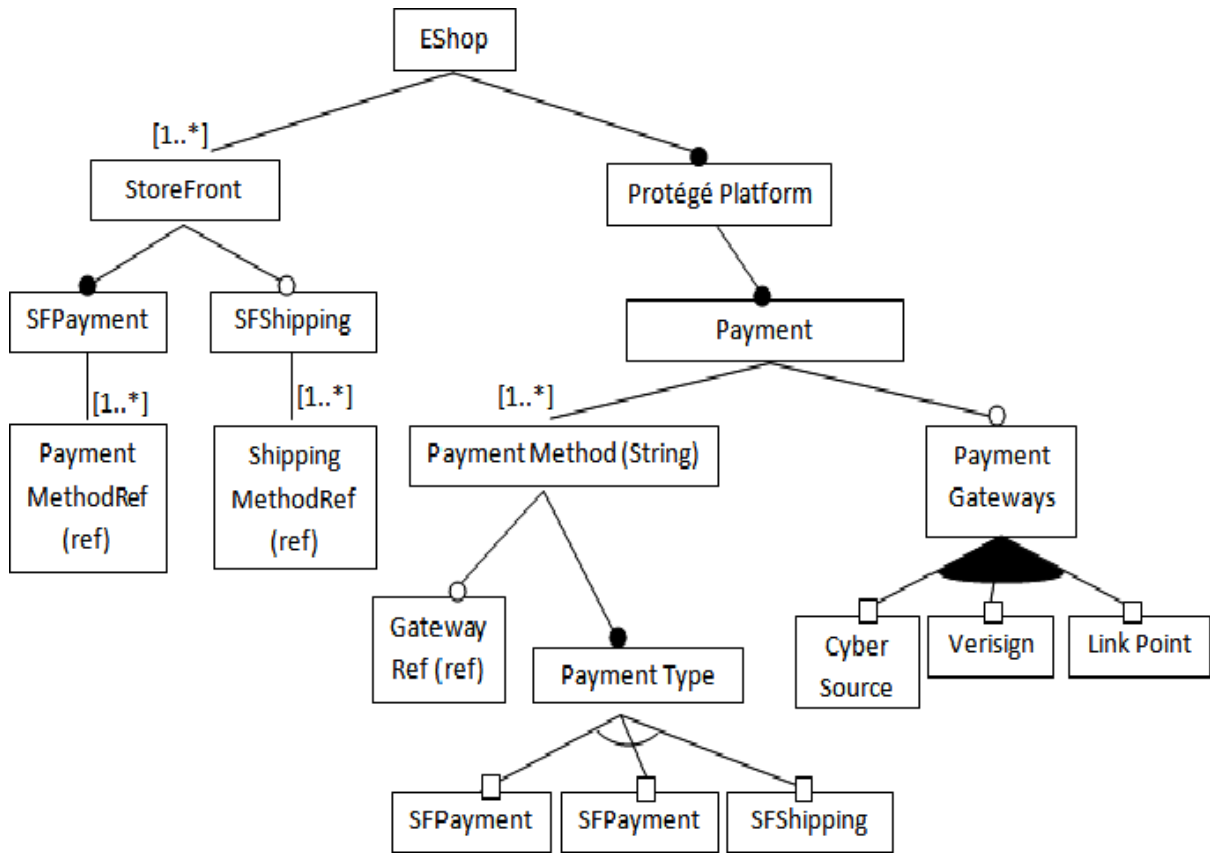


Figure 3: Sample Cardinality Based Feature Model [31]

**Orthogonal variability model (OVM):** Orthogonal variability model is a language that mainly focuses on variability and documents it using Variation Points, Variant and Dependency. Commonalities are expressed as variation points with only one variant, which means that all products in the product line share the same value for that variation point.

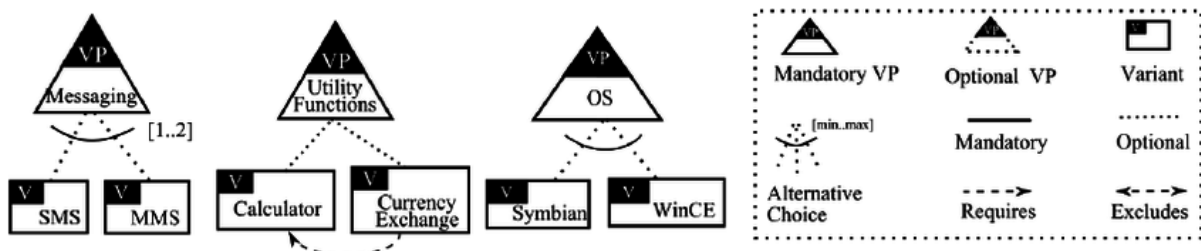


Figure 4: Sample Orthogonal Variability Mode [27]

## 1.1.5 Benefits and inconvenient

Despite all the benefits and advantages SPL promises. It is not a perfect approach, it still has its own downside and inconveniences that makes it costly if used improperly. In this section we will talk a little bit about the two faces of SPL coin.

### 1.1.5.1 Benefits

- ✓ **Higher Quality:** Because numerous projects can use the same core assets. Core assets are required to be of higher quality and with more tests, each project provides they continue to improve their quality.
- ✓ **Reduced Cost:** Developing applications using the SPL approach relies heavily on the re-usage of the shared core assets (commonalities), which decrease the costs of maintenance.
- ✓ **Improved Time to Market:** By taking advantage of the pre-built components and just adding the specific features, we can speed up the time to enter a market.
- ✓ **Flexible Staffing and Productivity:** When working on big projects it is much easier to assign different employees to the project because all the projects share the same core assets. Which leads to more flexibility inside the organizations and it helps employees to be more productive.

### 1.1.5.2 Inconvenients

- **Developing cost:** The initial budget needed to start is expensive compared to single products. This is why developers and organizations must have a big picture of the result.
- **Slow start:** The initial phase yields no benefits, but after reaching the break-even point, the benefits outweigh the investment.

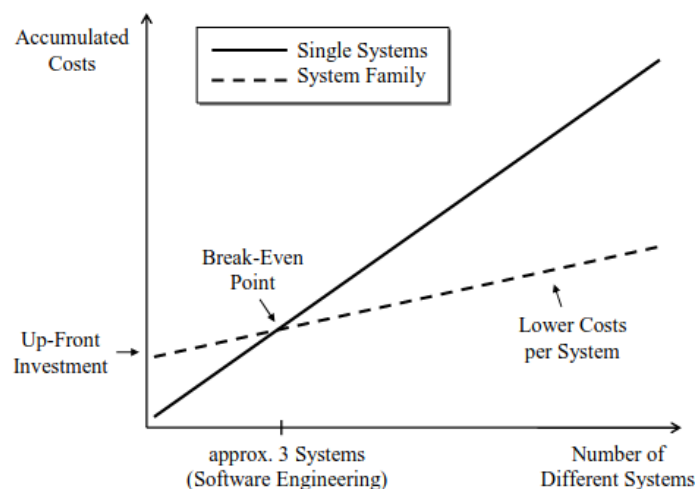


Figure 5: Economics of software product line engineering [4]

## **1.2 Internet of things**

### **1.2.1 Introduction**

Internet of things (IOT) is a growing topic and an important field due to the massive potential of it controlling how we live and how we work. With the wide availability and the low cost of connecting many devices created with built-in WIFI and sensors. Many organizations invested in this field and most of them have already started to grow thanks to it.

### **1.2.2 Definition**

Internet of things is defined as : “The concept of basically connecting any device with an on and off switch to the Internet (and/or to each other).This includes everything from cell phones, coffee makers, washing machines, headphones, lamps, wearable devices and almost anything else you can think of ”.[23] It also extends to machine components, such as a jet engine of an airplane or the drill of an oil rig.

Somayya, M et al highlight few critical aspects about IoT “An open and comprehensive network of intelligent objects that have the capacity to auto-organize, share information, data and resources, reacting and acting in face of situations and changes in the environment” [35].

The internet no longer concerns computers only, it has evolved to include mostly every object in life no matter what type is it. If we can attach a device to this object, then it is part of the internet of things. These connected devices communicate and share information with the help of certain techniques and protocols. The Internet of things simply put is any object with an on and off switch allowing it to connect and disconnect from the internet as well as share and communicate with other objects. The Internet of things mainly focuses on connecting the world using multiple devices.

## 1.2.3 Sensors

### 1.2.3.1 Definition

A sensor is a device that sends and receives signals or stimuli in the form of electrical signals. The input signals can be anything from physical, chemical or biological signals and convert them to electrical signals.

### 1.2.3.2 Types of sensors

Sensors differ from each other based on applications, input signal, conversion mechanism, material used sensor characteristics such as cost, accuracy or range. There are two types of sensors: passive and active sensors.

**Passive sensors:** A passive sensor does not need an energy source. It transforms input energy to output signal energy. Example of passive sensor: electric field sensing.

**Active sensors:** Active Sensors require an external source of power to operate. To produce the output signals, active sensors detect and respond to input signals. The active sensors also known as parametric sensors (sensors with properties). These properties can be modified in responseto environmental input then changed into electrical signals. Active sensors mostly used in places not observable by naked eye or in harsh environments.

Different types of sensors are used in IoT applications are given in Figure 6.

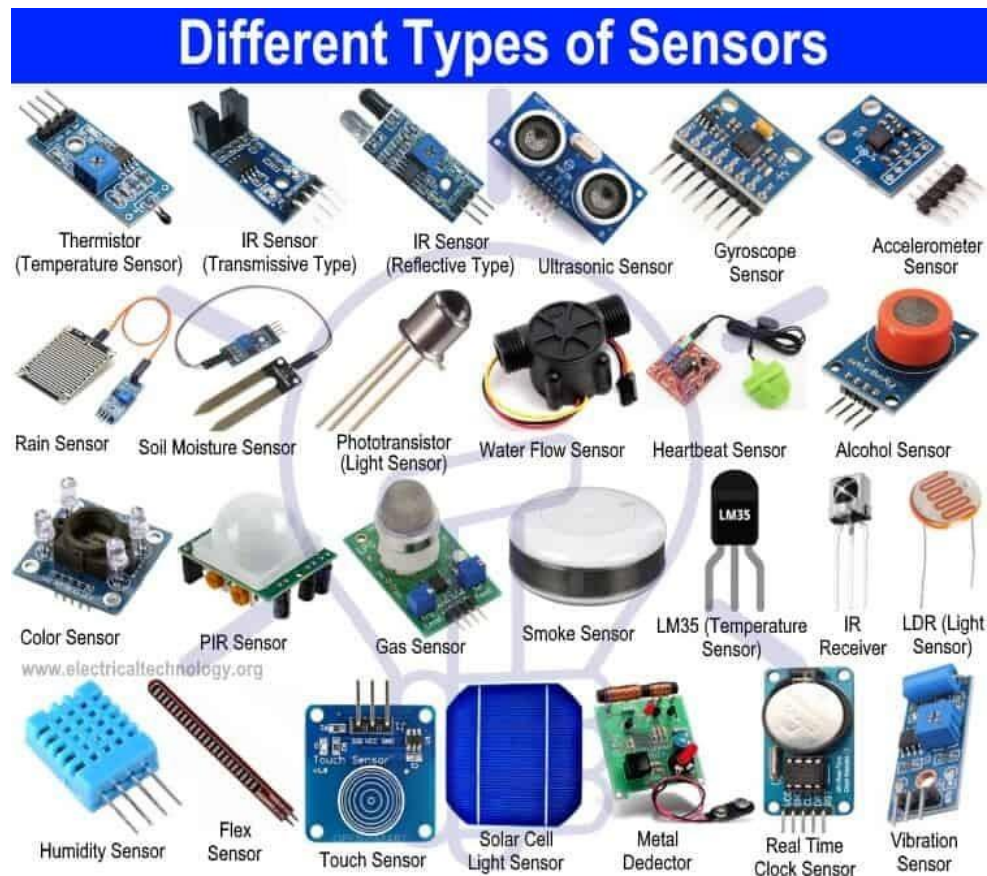


Figure 6: IoT sensors

## **1.2.4 IOT COMMUNICATION TECHNOLOGIES**

The internet of things enabled users to bring physical objects into the sphere of cyber world. This is possible thanks to various tagging technologies like RFID, NFC and 2D barcode, which permits physical objects to be identified and referred over the internet [26] in this phase we will mention a few communication technologies:

### **1.2.4.1 Radio Frequency Identification (RFID):**

RFID is a form of wireless communication to uniquely identify an object. It uses electromagnetic fields to automatically identify and track tags even if far away and not in line-of-sight of the reader.

### **1.2.4.2 Wireless Sensor Network (WSN):**

Wireless Sensor network is a network of devices transmitting information collected from a monitored field through wireless links. These monitored fields usually are vibration, temperature, pressure and vital body functions. WSN is an important aspect in the IOT paradigm. Due to the huge number of nodes (sensors), some sensors may not have a unique ID. WSN attracted many eyes in many fields such as healthcare.

### **1.2.4.3 WI-FI:**

Wi-Fi is a type of wireless network used to connect to the internet. Wi-Fi is radio waves diffused from a Wi-Fi router, a device detects and deciphers the waves then replies with data to the router. Nearby radio, TV antenna, and two-way radios can interrupt these waves.

### **1.2.4.4 Bluetooth:**

Bluetooth allows short-range communication between devices. The main purpose of Bluetooth is to get rid of cables that usually connect computers, devices . . . etc. While keeping the data transmission between them.

### **1.2.4.5 Machine to Machine (M2M):**

M2M uses point-to-point communications between machines. IoT systems use IP to send collected data to gateways or middleware platforms.

## **1.2.5 Different process of Internet of things (IoT):**

The Internet of Things (IoT) encompasses a multitude of interconnected devices and systems that collaborate seamlessly to enhance our daily lives. Various processes play crucial roles in enabling the seamless flow of data and information.



From identification and data collection to communication, computation, and semantics, each process contributes to the overall functionality of IoT as is shown in figure 7.

#### **1.2.5.1 Identification:**

The way a “thing” identifies in the world. Usage of IP (internet protocol) comes in hand to help achieve this task. There are plenty of ways to identify a device, for example IPV6.

#### **1.2.5.2 Sensing (Collection):**

Once a device connects to the Internet, sensors connected to the devices give the ability to collect data from the environment.

#### **1.2.5.3 Communication:**

Collected data is sent using different communication technologies such as RFID, Wi-Fi or Bluetooth.

#### **1.2.5.4 Computation:**

After collecting and transmitting the data, it is important to make it treatable to use it. Most devices have built-in microcontrollers for such a process.

#### **1.2.5.5 Services (Function):**

Function or operation simply put is how the device is going to react after processing data. For example, gas leaking raises an alarm

#### **1.2.5.6 Semantics:**

Having the same model of data is critical to have a common ground between devices or devices and computers. Because it makes sharing and reusing data easy and simple.

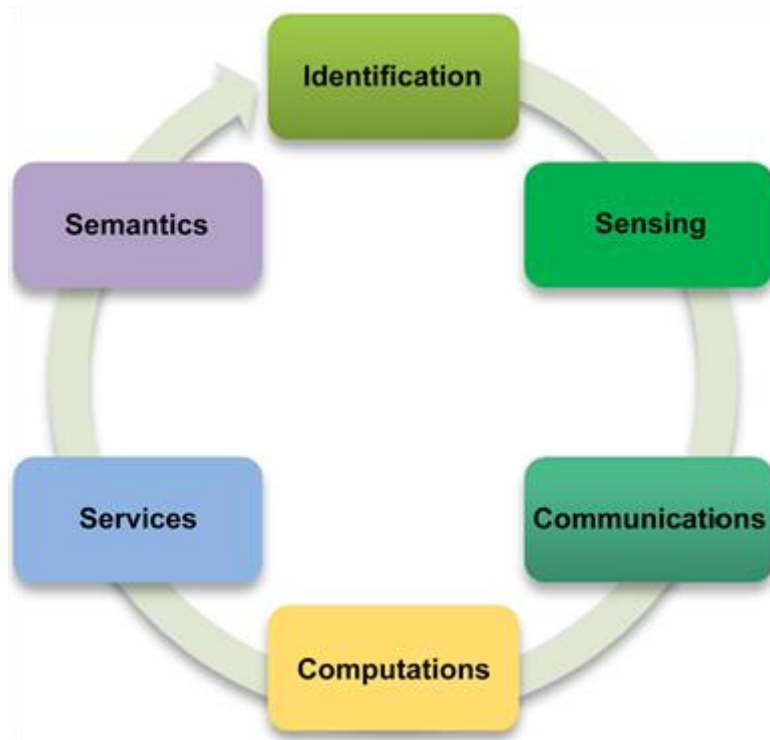


Figure 7: IOT elements [6]

## 1.2.6 Internet of Things domain Applications

The Internet of Things (IoT) represents a network where various smart objects are interconnected and can be individually identified. This intricate infrastructure opens up a plethora of fascinating applications across a broad spectrum. The domains of IoT applications cover a wide range, encompassing smart environment, smart metering, security, emergencies, retail, logistics, industrial control, smart agriculture, smart animal farming, domestic and home automation, as well as eHealth [10].

### 1.2.6.1 Smart Urban Cities

Smart cities leverage the capabilities of the Internet of Things (IoT) [24]. These cities incorporate various IoT applications such as automated transportation, urban security systems, smart energy management, surveillance, water supply management, and environmental tracking [19]. The implementation of IoT in smart urban environments aims to enhance the quality of life for residents. For instance, IoT solutions address challenges like traffic congestion, reduce noise pollution, and improve overall urban security. By harnessing the potential of IoT, smart cities offer promising solutions to enhance the living standards in urban communities today.

### 1.2.6.2 Smart Home System

Smart homes are gaining popularity due to two primary reasons. Firstly, advancements in sensor technology, along with wireless sensor systems, have become more accessible. Secondly, people

nowadays rely on technology to enhance their comfort and home security [33]. Figure 8 illustrates an example of a smart home system.

In smart homes, various sensors are deployed to provide intelligent and automated services to the residents [8]. These sensors assist in automating daily tasks and maintaining routines for individuals who may forget. They contribute to energy conservation by automatically turning off lights and electronic devices. Motion sensors are commonly used for this purpose and can also be employed for security purposes.

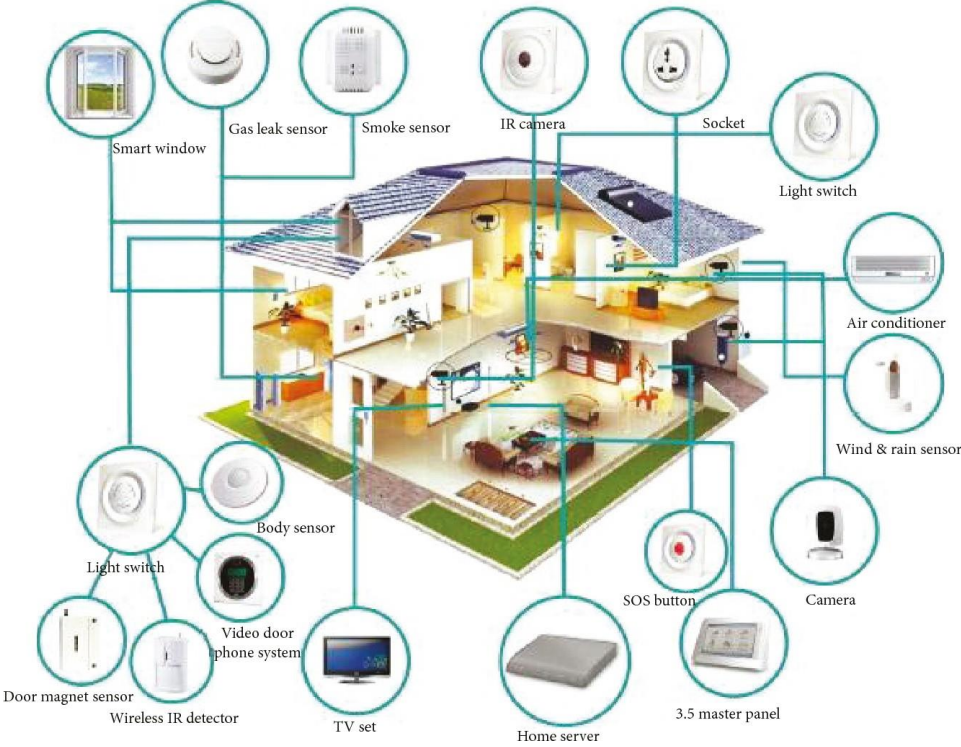


Figure 8: Smart Home [32]

### 1.2.6.3 Wearables

Wearable technology has witnessed a surge in demand in recent years [17]. This tech product has garnered a diverse user base, including teenagers, middle-aged individuals, and even older adults, owing to its user-friendly nature and associated health benefits such as sleep tracking, heartbeat sensors, oximeter, and pulse calculation, among others. Prominent technology giants like Google, Apple, and Samsung have made substantial investments in the development of such wearable devices [20]. These devices cater to the domains of health, fitness, and entertainment. Beneath their sleek exteriors, wearables are equipped with a range of sensors, complemented by a user-friendly display. Typically, they come bundled with the manufacturer’s software, offering users an exceptional user experience.

#### 1.2.6.4 E-Health and IoMT

Internet of medical things (IoMT) is an IOT-based solution to improve the relation between patients and healthcare facilities. Basically IoMT is a connected infrastructure of health system such as medical devices, software applications and services as shown in Figure 9.

E-Health refers to the integration of medical devices with the Internet to enable various remote medical services. These services include remote monitoring of patients, supervision of elderly individuals, online medical consultations, and even control of robotic arms for surgical interventions. These connected devices play a crucial role in continuously measuring medical parameters such as ECG, blood pressure, and temperature. They also facilitate activity recognition and monitoring, as well as remote medical evaluations.

In this thesis, we are mainly interested by E-Health application domain

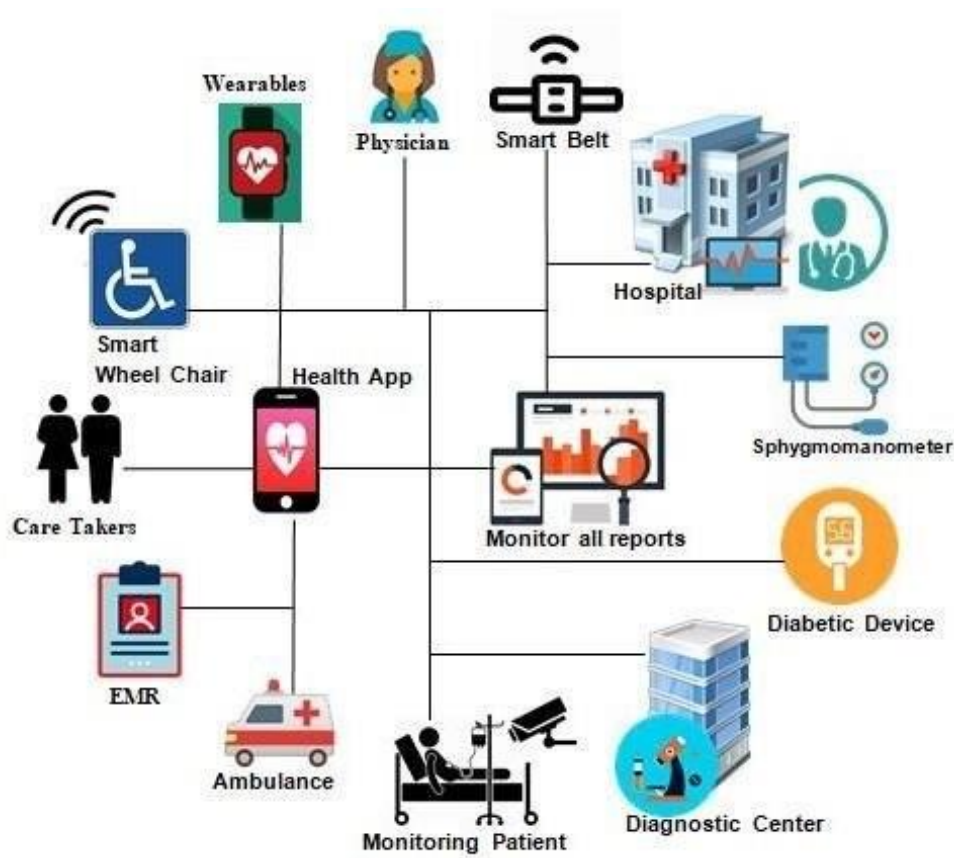


Figure 9: Smart Home [16]







#### 1.2.7 Application of IOT in Healthcare domain:

IOT technologies can bring many benefits to the healthcare domain, and the way these technologies applies is grouped into tracking, identification and authentication, automatic data collection and finally sensing. [30].

- **Tracking:** Tracking aims at the real time position tracking and identification of the device. For example to prevent left-ins during surgery.

- **Identification and authentication:** With patient identification, it is more like every patient has an id. This id can prevent incidents such as wrong drugs, dose and procedure.
- **Data collection:** Automatic data collection and transfer aims at reducing processing time, process automation (including data entry and collection and errors), automated care and procedure auditing, and medical inventory management.
- **Sensing:** Sensor devices provide real-time information on a patient's health, which plays a major role in identifying diseases as well as finding solutions [5].

**Table 1:** Medical Sensors

Feature	Sensor	Description
Temperature Sensor		DS18B20 is Temperature sensor manufactured by Dallas Semiconductor Corp, One of the easiest and inexpensive sensors to sense temperature
Pulse sensor		A pulse sensor is a device that is utilized to measure heart rate. It can be attached to a person's finger or earlobe, and then connected via a cable to an Arduino or similar device. The sensor incorporates an integrated optical amplification circuit and a circuit designed to eliminate noise that is commonly associated with heart rate monitoring.
Blood Pressure		In simple terms, blood pressure is the force exerted by the circulating blood against the arterial walls as it is pumped by the heart throughout the body. Blood pressure is expressed as two values: the systolic pressure, which is the pressure during a heartbeat, and the diastolic pressure, which is the pressure when the heart is in a resting phase between beats.
Breathing rate sensor		A nasal/mouth airflow sensor is a medical device that is utilized to monitor a patient's breathing rate. This device typically comprises a flexible thread that fits behind the ears, as well as a pair of prongs that are inserted into the patient's nostrils. The prongs detect airflow during breathing, allowing the device to accurately measure the respiratory rate.
Oxygen saturation sensor		oxygen saturation is determined through the detection of two different forms of hemoglobin: hemoglobin and deoxyhemoglobin. To make this measurement, two different wavelengths of light are utilized to detect the difference in the absorption spectra of these two forms of hemoglobin. Specifically, deoxygenated hemoglobin (Hb) absorbs more light at 660 nm, while oxygenated hemoglobin (HbO <sub>2</sub> ) absorbs more light at 940 nm. The concentration of Hb and HbO <sub>2</sub> in the bloodstream affects their absorption coefficients at these two wavelengths. The non-absorbed light from the LEDs is then detected by a photo-detector, allowing for the calculation of arterial oxygen saturation
Glucometer sensor		CCGM measure glucose levels in the interstitial fluid, which is the fluid that surrounds the cells in the body. The sensor is small, flexible and it is inserted just under the skin using a tiny needle. The sensor contains a small electrode that measures glucose levels in the interstitial fluid. As glucose levels change, the sensor generates an electrical signal that is transmitted to the CGM device. The device then converts this signal into a glucose reading.

## 1.2.8 Benefits and Inconvenient

### 1.2.8.1 Benefits

- ✓ IoT and big data: IOT and big data are relatable, since iot devices generate huge data that can be stored, processed and analyzed to make it meaningful and accurate in the eyes of experts to perform and make precise decisions.
- ✓ Intelligent IOT: Researchers start applying artificial intelligence to IOT use cases. Machine Learning can come in handy in image processing if well optimized. Even though it is still a possibility, the future of utilization of machine learning in iot systems is not far away.
- ✓ Smart devices: Smart devices refers to the ability to use real time data to support people in their everyday activities through the hidden built in intelligence in the devices.

### 1.2.8.2 Inconvenient

- × Energy efficiency: is crucial for IoT. Majority of the IoT devices rely on batteries to operate. Many IoT systems demand devices that are energy efficient to run for many years. Efficient techniques are required to achieve the accurate motes synchronization along with the RDC.
- × Security: New technologies are always threatened by new attacks, such attacks can affect availability or integrity of data sent to the clinic in case of healthcare systems. Therefore, security has, is and will always be a crucial aspect. In any new technology.
- × Rapid growth of data: IOT devices will generate huge data that can be hard to store it all with limited space. Some devices will generate as much as a gigabyte a minute for example high quality camera recording, storing all the data can be expensive. This is why some centers filter what data to store and what data to get rid of.

## 1.3 Different applications of SPL in IOT

### 1.3.1 Work of: Angel Cañete , Mercedes Amor, Lidia Fuentes[12]

The goal of this study is to propose a framework to support the deployment of Internet of Things (IoT) applications on edge-based infrastructures using multi-layer feature model. The feature model captures the different aspects of an IoT application, including the functional requirements, non-functional requirements, and deployment requirements.

### 1.3.2 Work of: Inmaculada Ayala, Mercedes Amor, Lidia Fuentes and José.Troya[7]

The authors proposed a software product line process for developing software agents for IOT applications and demonstrated its effectiveness. They tackled the challenges of heterogeneity

and scalability for IoT systems.

### 1.3.3 Work of: ASAD ABBAS, ISMA FARAH SIDDIQUI, SCOTT UK-JIN LEE[3]

The authors aim to address the challenges of managing the variability by developing a feature model that captures the different features and their dependencies in an IoT application. They proposed an approach involves defining a set of features and their relationships, and then using XML to represent the feature model.

### 1.3.4 Work of: Angel Cañete, Mercedes Amor, and Lidia Fuentes[13]

The authors proposed an SPL approach for deploying energy-efficient IoT (Internet of Things) applications in edge-based infrastructures. The method involves using energy-aware deployment strategies that take into account the specifications of the edge-based infrastructure, such as energy consumption of the devices and available resources

**Table 2:** Comparison of Papers on Variability Management in IoT

Approach	Key Concepts	Variability Management	Case Study	Advantages	Limitations
Angel Cañete et al (2022)[12]	Multi Layer Feature Model	4 Layer FM: Application Function Service Infrastructure	Methods and procedures to help deploy an IoT app	Ensure efficiency	Does not support/consider medical devices Limited scalability and flexibility
Inmaculada Ayala Et al (2015) [7]	Software product line. IoT Agents. Variability Modeling.	Feature model. Architecture model. Common variability language (CVL).	Automated Vehicule systems.	Improved reliability of the developed agents. Increased productivity and quality of agents.	No flexibility to satisfy all client requirement. Limited scalability.
Asad Abbas et al (2017) [3]	XML based feature model. IoT application.	Feature model based on XML.	Manage Temperature and lighting based on user's preferences and patterns.	Managing and controlling variability becomes easier	Not suitable for all IoT applications. Limited Scalability and reusability.
Angel Canete et al (2021) [13]	Edge based infrastructure	Feature model. Software Product line.	Set of rules to better deploy IoT applications	Energy efficiency. Scalability. Reusability.	Complex to Implement. Limited computational.



## **1.4 Analysis:**

Developing IoT applications with the software product line approach is popular, but yet each study lack the guidance and the right plan to reduce the complexity of this approach, as shown in the previous studies, each study used the spl approach in its own way which makes it even more complex than it already is. However, with the right guidance this complexity can be reduced to a certain point, this is where ontologies come in handy to provide just that.

## **1.5 Conclusion:**

In this chapter, we explored two key technologies: SPL (Software Product Lines) and IoT (Internet of Things) and what benefits they provide. We saw how SPL can enable the development of highly configurable software systems, while IoT brings these highly configurable systems into the world. We also introduced the concept of ontology, which plays the role of a guidance to represent knowledge and relationships within a domain, and how it can be used to enhance the interoperability and adaptability of SPL and IoT systems.

## Chapter 2

---

# Domain Engineering

---

## 2.1 Introduction

After having seen the main domains of our system starting with product lines, Internet of Things and ontology, it is now time to move on to the design of our SPL. In other words, it is about determining how to use these domains to achieve our objective. In this chapter, we will present both the reasons that prompted us to undertake this work and our proposed solution. We will also provide an overview of the design process, illustrated by a schema detailing the different stages involved.

## 2.2 Healthcare platforms in Algeria ?

In Algeria, the utilization of IoT devices and sensors to gather real-time patient data in healthcare platforms is not widespread, and the e-health platforms existing are too specific and unique to one clinic. These type of platforms suffer from scalability and maintenance problems, which is why it is a good idea to use the software product line approach in the healthcare scene:

1. **Telmedic:** Telemedic allows patients to consult with healthcare professionals online. Patients can book appointments with healthcare professionals, and receive medical advice and prescriptions through the platform[36].
2. **Tasshilat:** Tasshilat is an e-health platform that provides online medical consultations, appointment booking, and medical records management. The platform also allows users to purchase health insurance policies and make online payments for healthcare services.

Few healthcare platforms integrate IoT devices and sensors for real-time patient data collection, which can offer valuable insights to doctors for informed decision-making.

## 2.3 SPL Development Process for the IoT

Our goal is to design and develop a Software Product Line (SPL) in the context of the Internet of Things (IoT). The process manages an SPL as a whole rather than considering each product as a separate entity. Thus, the SPL must consider the needs of all targeted user categories. Defining the members of the planned product line helps to identify and plan for the implementation of reusable common elements and the differences between them. In this phase we will adopt the SPL development process and enhance it to satisfy our end goal.

The different parts of our process are:

- **Domain Analysis.**
- **Variability Modeling.**
- **Mapping Feature to Ontology.**
- **Domain Design.**
- **Domain Implementation.**

The details of each phase will be shown in the following subsections, a diagram summarizing these various points is illustrated in the figure below:

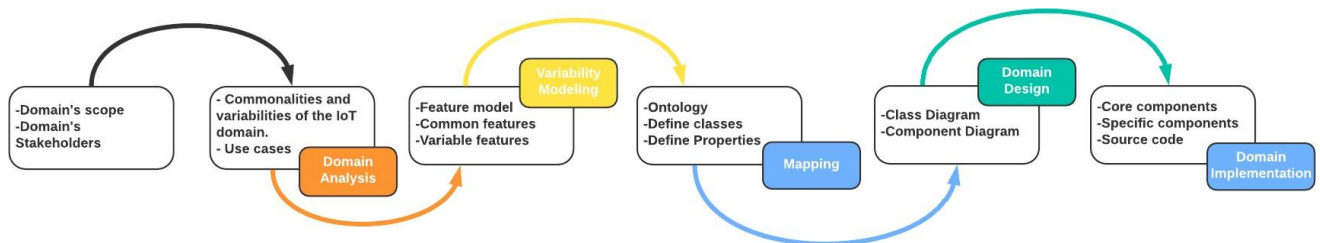


Figure 10: SPL developing process.

### 2.3.1 Domain Analysis

The first step in the development process is to conduct a domain analysis to understand the problem domain and identify the commonalities and variabilities of the IoT domain. The domain analysis should focus on the requirements, stakeholders, and technologies involved in the IoT domain.

To further facilitate this understanding, we will utilize use case diagram. By using a use case diagram, we can effectively capture and communicate the fundamental functionalities and relationships present within the IoT domain.

### 2.3.1.1 Use case Diagram:

Identifying the actors and the functionalities of our product is crucial before building our feature model, we split the use case diagram into two parts

**Patient's use case:**

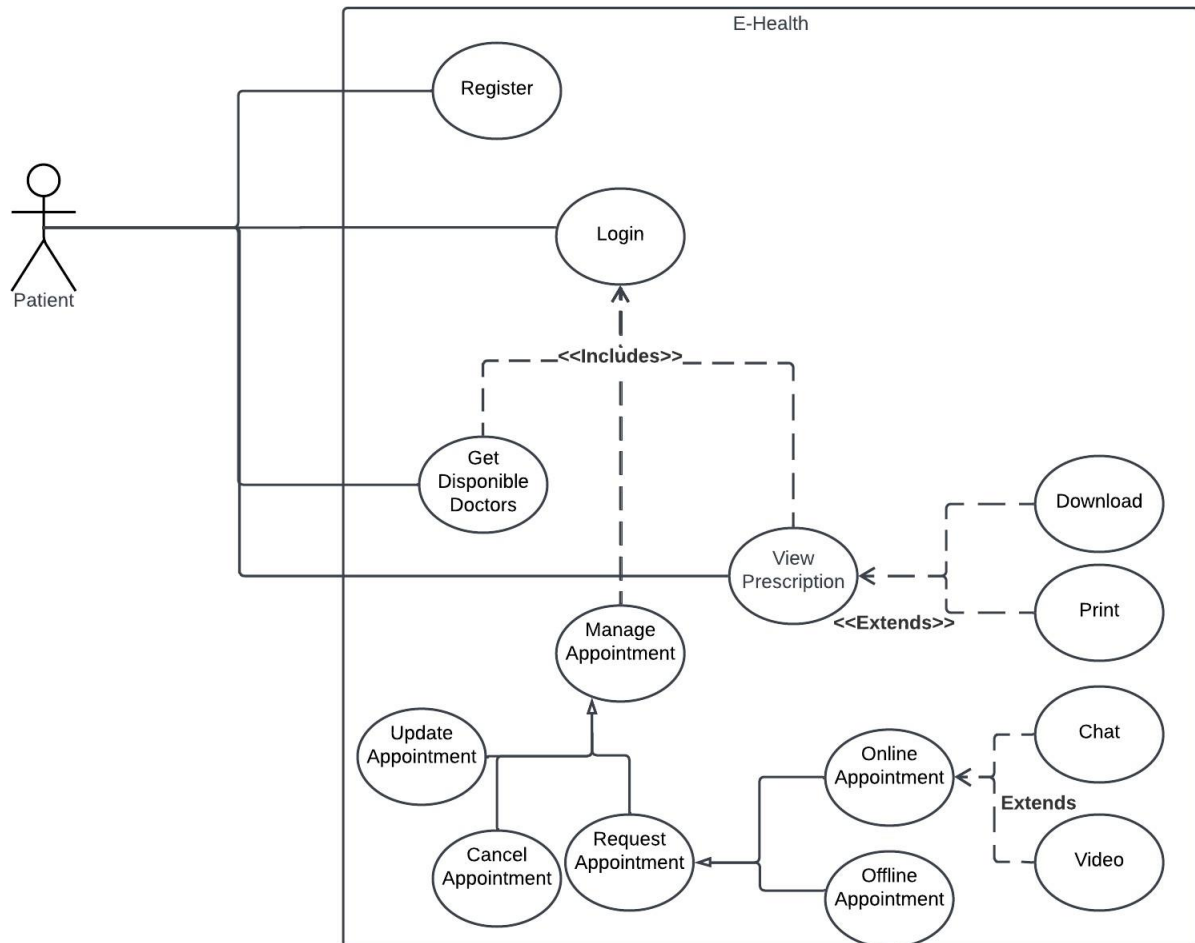


Figure 11: Patient's use case.

## Doctor's use case:

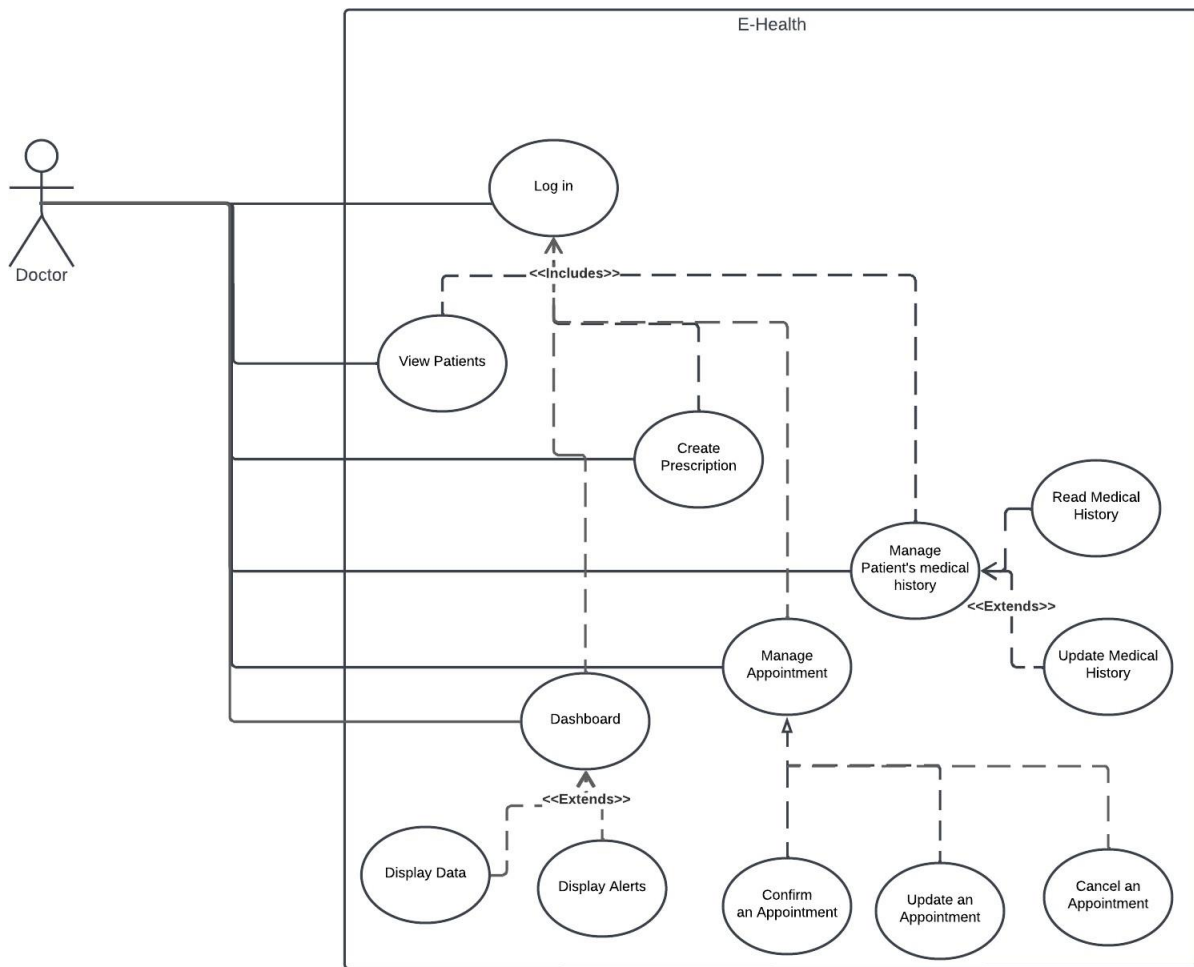


Figure 12: Doctor's use case.

### 2.3.2 Variability Modelling

Variability modeling is a crucial aspect of software product line engineering. Linden et al defines it variability: “It covers the whole life-cycle. It starts with the early steps of scoping, covering all the way to implementation and testing and finally going into evolution.”[22] It involves identifying and representing the common and varying features of a product line. The purpose of variability modeling is to facilitate the efficient and effective management of product line variability. In SPL, variability modeling involves defining a set of features that can be included or excluded from individual products within the product line. These features can be described in terms of their characteristics, dependencies, and relationships with other features. The variability model captures the commonalities and variabilities among the products in the product line, which can be used to generate product configurations.

There are several approaches to variability modeling, the most commonly used approach is feature modeling, and it involves representing the features of the product line as a hierarchical structure of feature models. In our project, we will be using feature modeling as our approach to variability modeling.

With feature modeling, we will represent the features of the product line as a hierarchical structure of feature models. Each feature model captures the commonalities and variabilities among the products in the product line, allowing us to efficiently manage product line variability.

In our feature model, we will define a set of features that can be included or excluded from individual products within the product line. We will describe the characteristics, dependencies, and relationships of each feature, and specify any constraints on valid product configurations. The established Feature model consists of three models, which are distinguished according to the types of features it contains.

- a. **E-Health Feature Model:** The feature model for our E-Health application is divided into three main sections: data, software, and Hardware. The data feature model contains features related to the data itself (collect, storage. . .) . The software feature model includes features related to user interface and patient management. Lastly the hardware feature model includes features that are related to the sensors used, the physical devices.

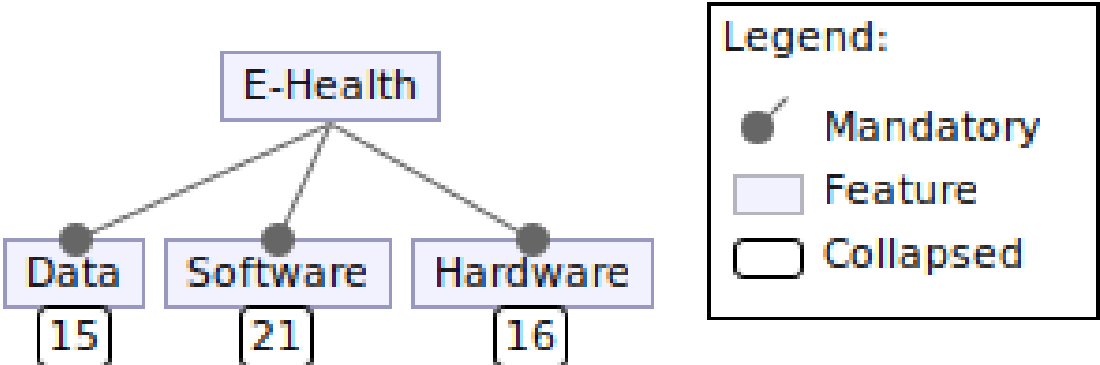


Figure 13: E-health Feature model.

- b. **Software Feature Model:** The software feature model is a crucial part in our application, due to its direct connection with the end user(user interface) and it plays the role of a middleware between the hardware and data

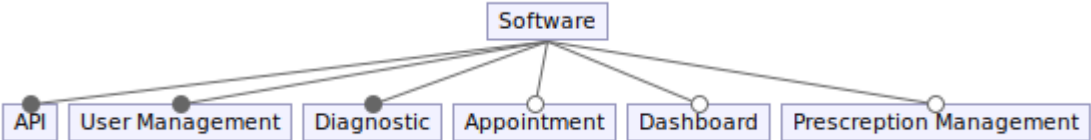


Figure 14: Software Feature Model

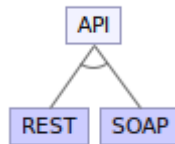


Figure 15: API Feature Model

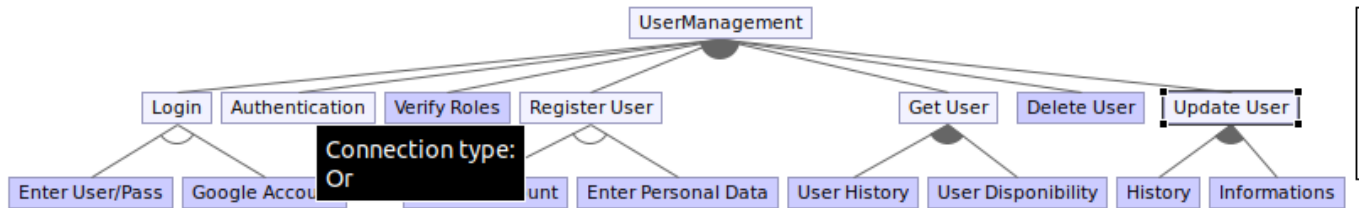


Figure 16: UserManagement Feature Model

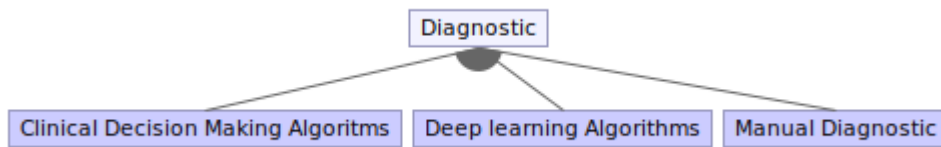


Figure 17: Diagnostic Feature Model

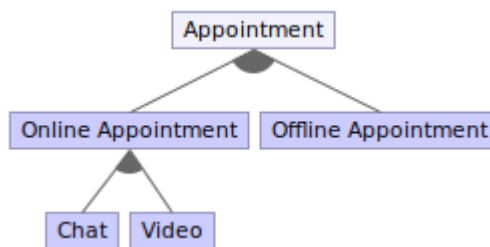


Figure 18: Appointment Feature Model

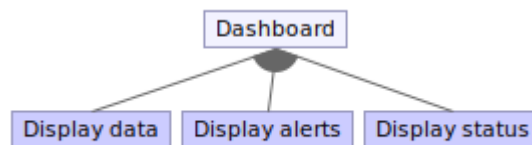


Figure 19: Dashboard Feature Model

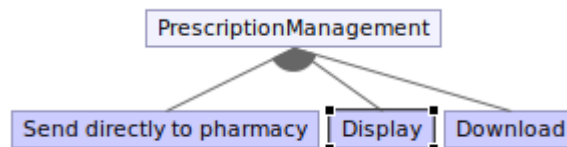


Figure 20: PrescriptionManagement Feature Model

**Constraints:**

REST and SOAP requires WIFI or Cellular Data

Login requires Register User

Login requires Authentication

Verify roles requires Login

Register User requires WIFI or Cellular Data

Get User requires Login

Delete User requires Login and Verify roles

Update User requires Login and Authentication

Appointment requires User Disponibility

Display Data hasSensor Monitor.

- c. **Data Feature Model:** Data feature model includes features related to how we collect data, how we send the data from the device to the final endpoint, how we receive it and where we store it.

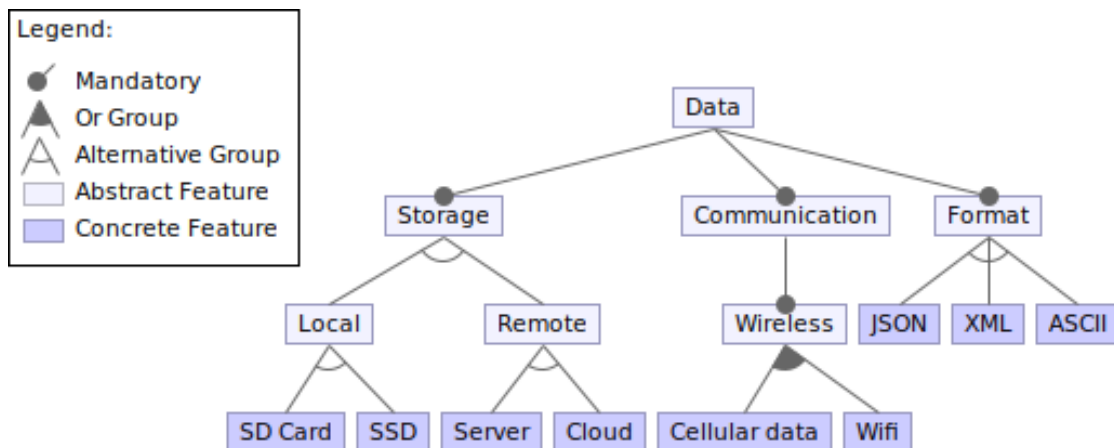


Figure 21: Data Feature model.

**Constraints:**

Remote requires WIFI or Cellular Data



- d. **Hardware Feature Model:** Data feature model contains features related to the physical devices and sensors used to capture data and how they differ from each other plus what they use to operate internally.

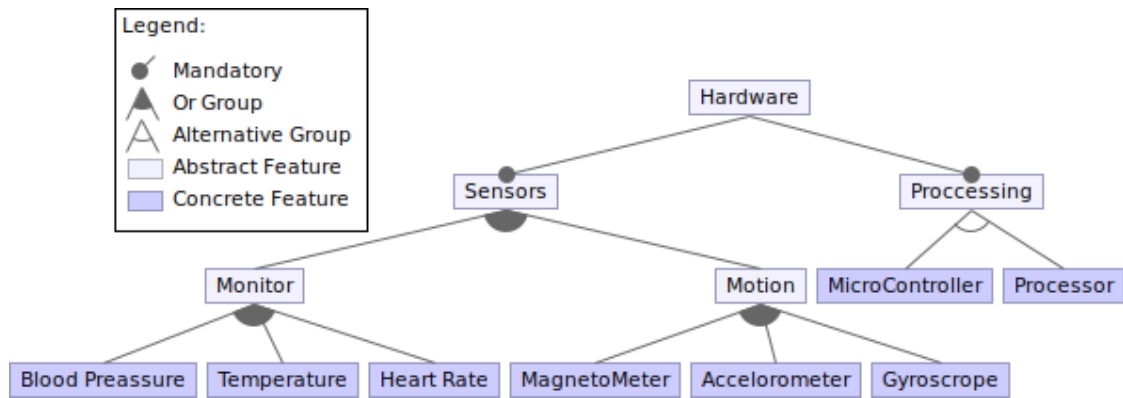


Figure 22: Hardware Feature model.

**Constraints:**

Sensors requires Processing.

**2.3.3 Mapping Feature Model to Ontology**

The third step involves the development of an ontology, which encompasses a collection of classes, properties, and relationships that accurately represent the concepts and their interconnections within a specific domain. This ontology should also possess the necessary flexibility to accommodate new concepts and relationships as the domain evolves over time.

In our work, we build upon and expand the ontology proposed by Tenório et al. [28] to illustrate the feature model. Additionally, we enhance the work of Filho [18] by incorporating ontologies to depict the feature model. To construct our ontology, we utilize Protégé [Sta]. We define the concepts and relationships within the ontology to formalize the functionalities and their potential associations.

Both the feature model and ontology operate at the same level of abstraction, providing meta-information about the knowledge within a given domain [15]. The ontology serves as a representation of the knowledge within the domain in a network-like model. It captures the relationships between concepts through triple statements consisting of a subject, predicate, and object. However, the ontology can be formally represented using the 2, which is an ontology language based on description logic. In OWL, the subject of a statement corresponds to a class, while the predicate can be categorized as an object property if it relates to another class membership, or a datatype property if it pertains to data values. Additionally, the OWL ontology can be extended with rules (e.g., Semantic Web Rule Language<sup>3</sup>) that can be executed to derive further knowledge.

### 2.3.3.1 Transformation Rules: Converting Feature Model to Ontology

In order to convert the Feature Model into an ontology, a series of key transformations need to be applied. The main transformations are defined and illustrated in Table 3. The first column indicates the symbol in the Feature Model, the second contains the meaning of the symbol, and the third is the corresponding axiom in the logical description.

**Table 3:** From Feature model to Ontology.

Feature model	Signification	Ontology Notation
	Root Feature F	F:Feature
	Mandatory Feature	F : MandatoryFeature
	Optional Feature	F : OptionalFeature
	F1 is the parent of F2 and F3	isParentOf(F1, F2) isParentOf(F1, F3)
	Or decomposition	F2 :OrFeature F3 :OrFeature hasOrFeature(F1,F2) hasOrFeature(F1, F3)
	Xor decomposition	F2 :AlternativeFeature F3 :AlternativeFeature hasAlternativeFeature(F1, F2) hasAlternativeFeature(F1, F3)
//	Feature has sensor	hasSensor(F1, F2)
//	F1 requires F2	Requires(F1, F2)
//	F1 excludes F2	Excludes(F1, F2)

### 2.3.3.2 Classes:

Mapping features to classes is a crucial step in connecting our feature model with our ontology. Features, representing the application functionalities, correspond to classes in the ontology. This alignment creates a structured representation of the system's capabilities within the ontology, facilitating a comprehensive understanding of the system's components and relationships. Mapping features to classes bridges the gap between feature modeling and ontology.

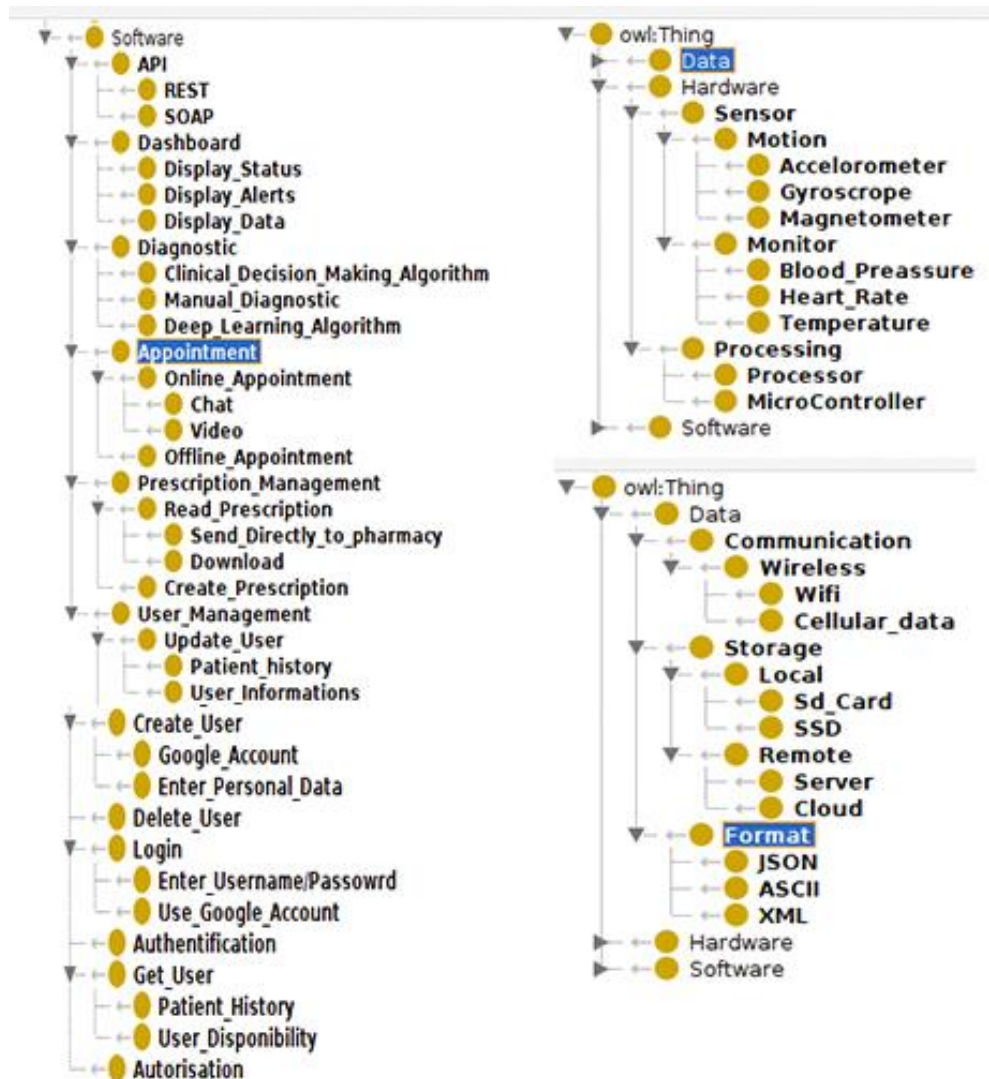


Figure 23: Classes of the proposed ontology

### 2.3.3.3 Relations:

Object properties play a fundamental role in ontology modeling, representing relationships between classes and instances within a domain. These properties capture the connections and interactions among entities, enabling rich and expressive knowledge representation. In our ontology we added **hasSensor** relation, which enables connecting the software part responsible for processing, treating or displaying the health sensor in our ontology.



Figure 24: Relations of the proposed ontology.

### 2.3.4 Domain design:

Once the Feature model is complete, the next step is to develop a product line architecture based on common and variable features of the IoT products. The architecture should be designed to accommodate the variabilities in the IoT domain, such as the differences in hardware platforms, communication protocols, and data formats.

In our case we use UML (Unified Modeling Language) to create clear and precise visual representations of the SPL. UML provides a standard notation for modeling software systems, and supports various types of diagrams that can be used to capture different aspects of the system. In the context of domain design for software product lines, UML can be used to create models that capture the commonalities and variabilities across a family of related software systems.

- a. **Class Diagram:** The class diagram shows the internal structure of the system. It provides an abstract representation of system objects that will interact to make use cases. The purpose of a class diagram is to model the entities of an information system, which can represent all the final informations managed by the domain. These informations is structured (grouped into classes). After the previous analysis, we obtained the class diagram, as the shows the following Figure

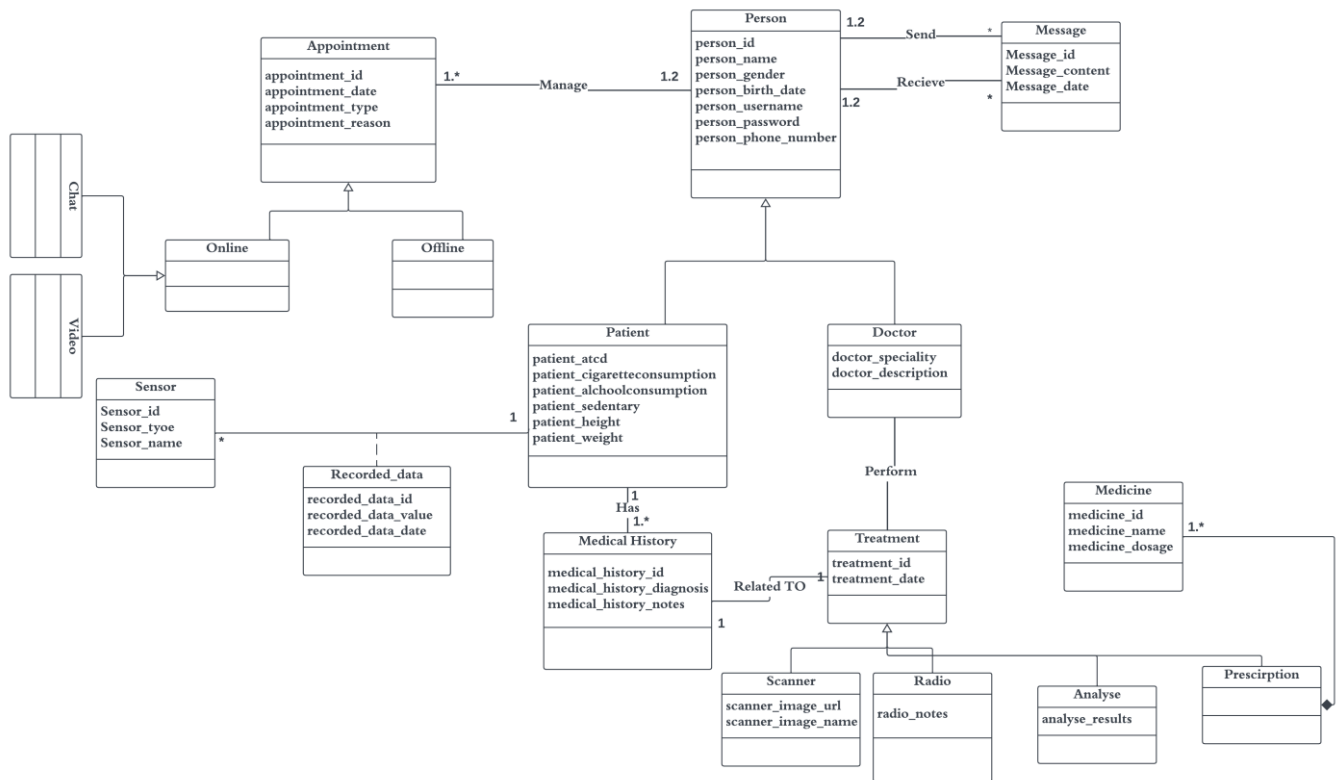


Figure 25: Class Diagram

### 2.3.5 Domain Implementation

This step involves developing the software artifacts that will constitute the IoT SPL, including the core components, the domain-specific components, and the configuration and customization tools. The input for this step may include different UML Diagrams and the output may include source code, binary packages, and documentation.

## 2.4 Architecture IoT

The architecture consists of three layers: data collection, data storage and data analysis and processing (Figure 27).

- **Data collection:** The Data collection phase encompasses the acquisition of diverse health-related parameters, including but not limited to heart rate, blood pressure, and temperature. The collection of this data can occur either continuously or intermittently, depending on the specific requirements outlined in the product’s specifications.

- **Data normalization:** The data obtained from these devices cannot be directly accessed and utilized; it necessitates initial cleansing and formatting procedures.
- **Data storage:** The phase of data storage assumes a pivotal role in the overall process, as it bears the responsibility of securely and reliably housing all the amassed health data. One of the paramount considerations is to ascertain the scalability of the data storage mechanism, which can be achieved through the meticulous design of an apt database schema. Typically, the collected data is stored in a structured format to ensure efficient organization and retrieval.
- **Data Treatments:** Data applies the sophisticated algorithms, data mining techniques, and machine learning methodologies to effectively process and analyze real-time data. Such endeavors aim to enhance decision-making processes and deliver personalized care to patients.

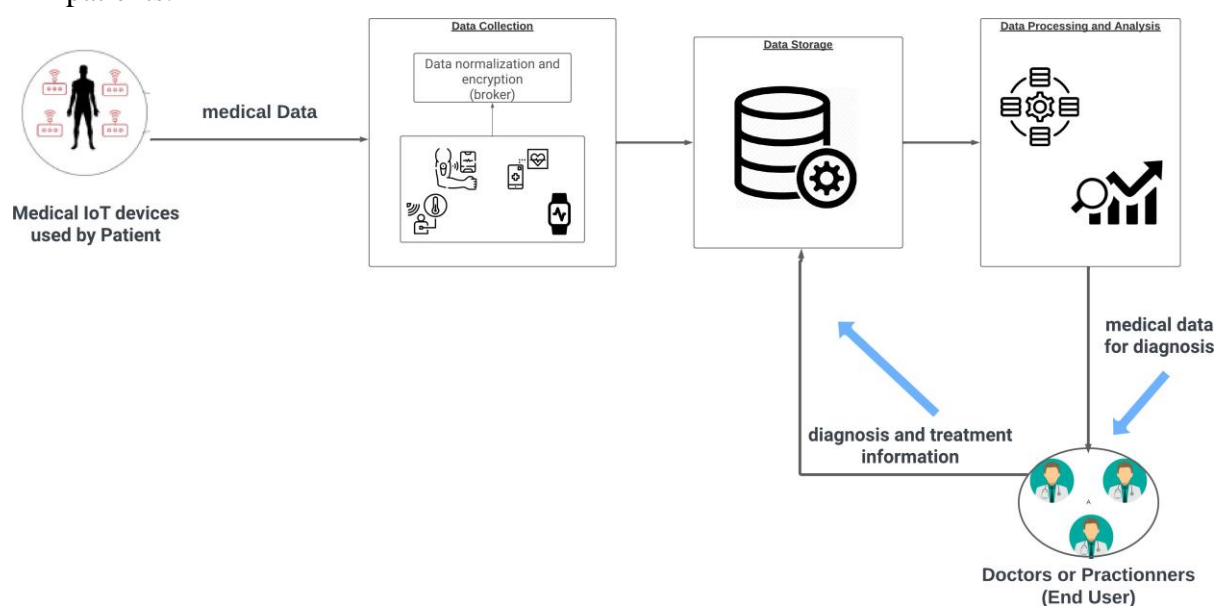


Figure 27: IOMT Architecture

## 2.4.1 Data collection:

Before diving into the sequence diagram illustrating the data collection process from the different sensors, it is crucial to understand the underlying objective and context. Data collection from sensors plays a vital role in gathering real-time information and providing valuable insights for our patient's health details. The sequence diagram (Figure 28) will showcase the step-by-step interactions and flow of events between each sensor and our application. By examining this diagram, we can gain a comprehensive understanding of how the data is captured and transmitted, enabling us to make informed decisions based on the collected information.

The following algorithm outlines the steps taken by the sender side (sensor) in order to transmit data efficiently:

---

**Algorithm 1** Sender Side (Sensor)

---

- 1: Post request with patientID
  - 2: Subscribe to topic
  - 3: **while** NEW DATA AVAILABLE **do**
  - 4:     Collect the data from the sensor
  - 5:     Send data to broker
  - 6: Terminate the connection
- 

Then, the sender side (broker) employs the following algorithm to receive and process data from the sensor before forwarding it to the appropriate API endpoint:

---

**Algorithm 2** Sender Side (Broker)

---

- 1: **while** RECEIVE DATA FROM SENSOR **do**
  - 2:     **if** SENSOR IS SUBSCRIBED **then**
  - 3:         Format the received data
  - 4:         encode the formatted data
  - 5:         Send to API endpoint
  - 6:     **else**
  - 7:         Subscribe sensor to certain topic
- 

Finally, the receiver side (server) utilizes the following algorithm to securely receive, decode, and store data while providing acknowledgment of successful reception:

---

**Algorithm 3** Receiver Side (Server)

---

- 1: **while** DATA IS RECEIVED **do**
  - 2:     Decrypt the received data
  - 3:     Decode the decrypted data
  - 4:     Store the decoded data
  - 5:     Send an acknowledgment
-

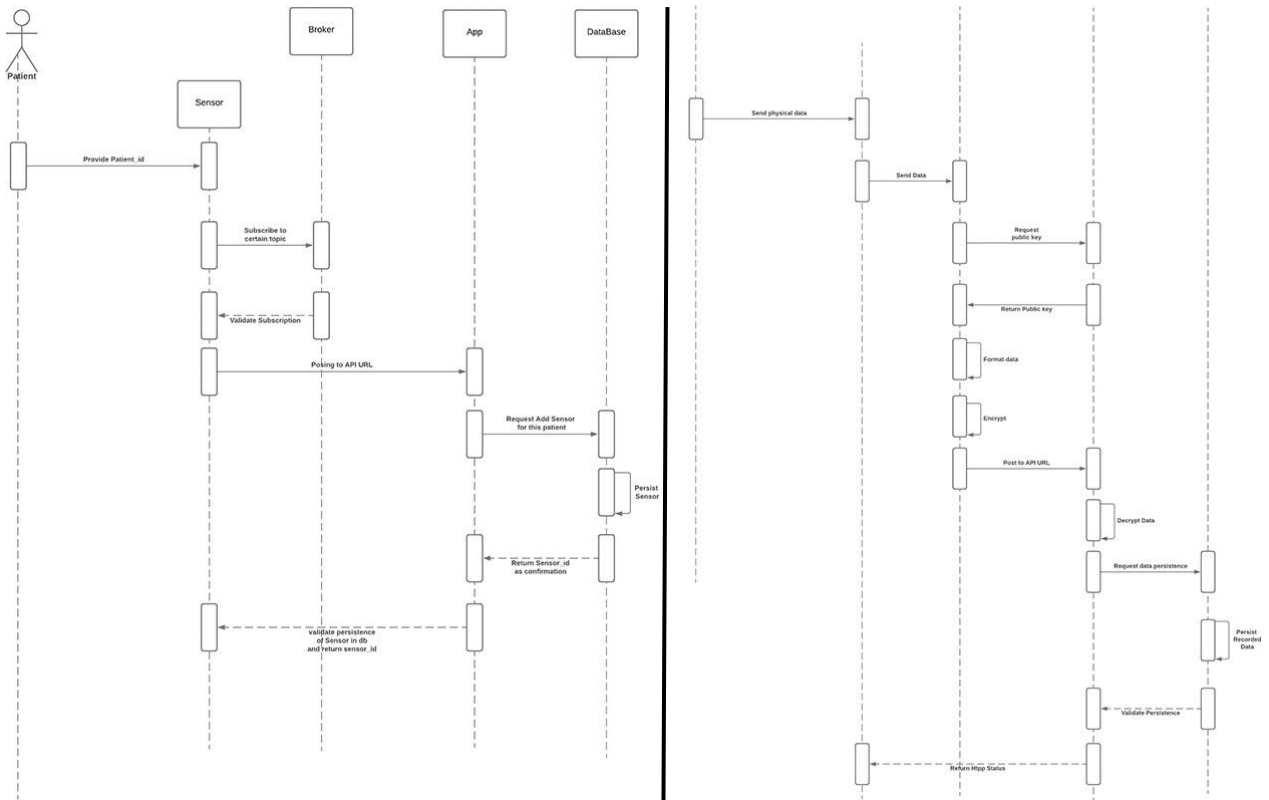


Figure 28: Data Collection phase

### 2.4.2 Data evaluation:

Before diving into the intricate details of the sequence diagram for data evaluation, it is essential to grasp the overall context of this process. The data evaluation stage plays a pivotal role in extracting meaningful insights from the data gathered, aiding in informed decision-making. This sequence diagram (Figure 29) display the interactions and flow of information between various components involved in the evaluation process. By carefully analyzing the incoming sensor data, the system can identify the cardiovascular risk.



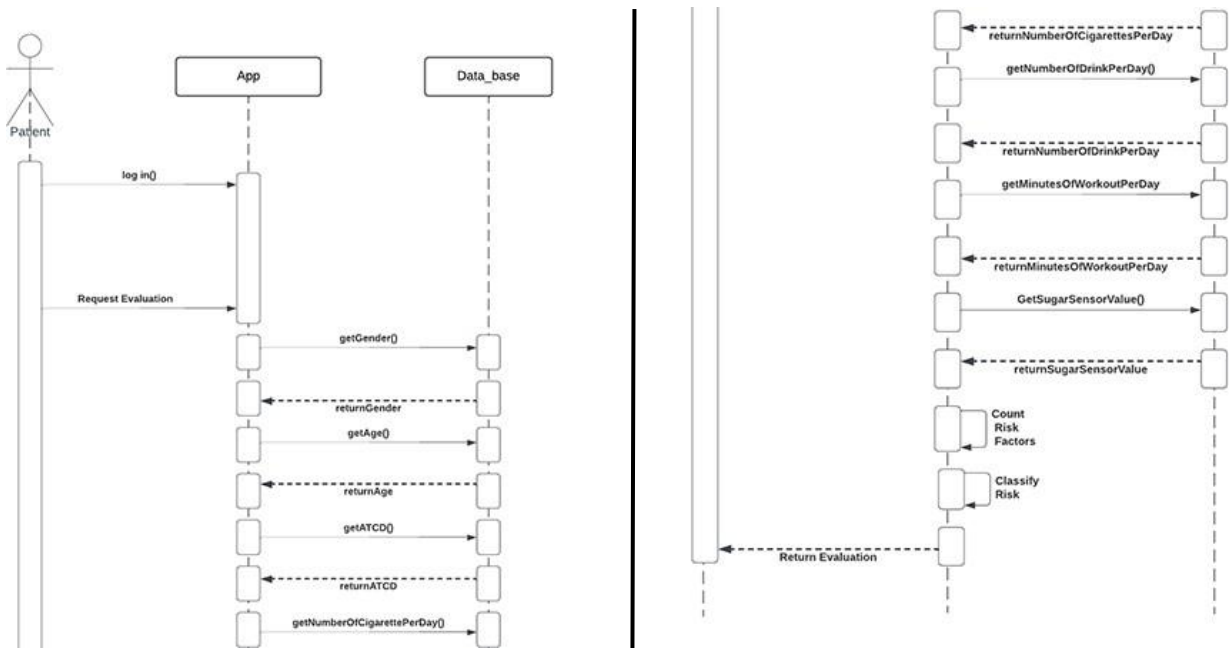


Figure 29: Data Evaluation phase

## 2.5 Conclusion:

In this chapter, we have introduced the design of our software product line and the role of ontology in guiding and representing our feature model. Next, we discuss how to derive an application example based on specific requirements.

## Chapter 3

---

# Implementation

---

### 3.1 Introduction:

In this chapter, our objective is to implement a practical application that demonstrates the concept of a software product line within an e-Health platform that supports the Internet of Medical Things (IoMT). The purpose of this application is to collect and utilize data for the betterment of healthcare. The data we aim to capture is primarily associated with a patient's vital signs, such as heart rate and blood sugar levels. However, due to resource constraints, we have simulated blood pressure and sugar levels sensors. We have chosen to mimic these sensors based on widely adopted protocols in the market, specifically the MQTT protocol. By utilizing these simulated values, we can showcase how our application processes and handles real patient data in a broader sense.

### 3.2 Development tools:

#### 3.2.1 Used Tools:

- **VS Code:** Visual Studio Code is a streamlined code editor with support for development operations, it also know as VS Code made by Microsoft and runs on Mac-Os, Linux, and Windows. [1]
- **Node JS:** Node.js is an open-source, cross-platform, back-end JavaScript run time environment that runs on the V8 engine and executes JavaScript code outside a web browser. A Node.js app runs in a single process, without creating a new thread for every request.
- **PostgreSQL:** PostgreSQL, also known as Postgres, is a free and open-source relational database management system emphasizing extensibility and SQL compliance.

- **Protégé:** Protégé is a free, open source ontology editor and a knowledge management system. The Protégé meta-tool was first built by Mark Musen in 1987 and has since been developed by a team at Stanford University. The software is the most popular and widely used ontology editor in the world.
- **Eclipse:** Eclipse is an integrated development environment used in computer programming. It contains a base workspace and an extensible plug-in system for customizing the environment.
- **Eclipse Mosquitto:** Eclipse Mosquitto is an open source (EPL/EDL licensed) message broker that implements the MQTT protocol versions 5.0, 3.1.1 and 3.1.

### 3.2.2 Back end:

- **PG:** PG is a node package that stands for "PostgreSQL." It is a small and lightweight package that provides a simple and efficient way to interact with PostgreSQL databases using JavaScript or Node.js.
- **Express:** back end web application framework for building RESTful APIs with Node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs.
- **Express session:** An Express session refers to the use of session management in web applications built with the Express.js framework. It allows the server to store and maintain user-specific data across multiple HTTP requests, such as user authentication details or custom preferences.
- **Method-Override:** Method-Override provides a way to simulate HTTP methods such as PUT or DELETE, which are not supported by HTML forms.
- **WebSocket (WS):** WebSocket is a JavaScript-based node package that enables real-time, bidirectional communication between a web browser and a server.
- **Multer:** Multer is a node.js middleware for handling multipart/form-data, which is primarily used for uploading files. It is written on top of busboy for maximum efficiency.
- **Passport:** Passport is a small Node.js package that simplifies the process of authenticating users and managing sessions, making it a popular choice for adding authentication functionality to Node.js applications.

### 3.2.3 Front end:

- **Java Script (JS):** Java script is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS. As of 2022, most of websites use JavaScript on the client side for webpage behavior, often incorporating third-party libraries.
- **EJS:** EJS is a simple templating language that lets you generate HTML markup with plain JavaScript. No religiousness about how to organize things. No reinvention of iteration and control-flow. It's just plain JavaScript.
- **Full Calendar:** FullCalendar is a flexible and customizable JavaScript library that allows you to create interactive calendars on web pages. It includes displaying events, managing time slots and handling drag-and-drop interactions.
- **bootstrap:** Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development.
- **CSS:** Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language such as HTML or XML.

## 3.3 Application:

In this section, we will take a general look of the web application as well as a deep look to highlight the important stages in order to explain the complicated parts.

### 3.3.1 Analyses:

The application derived from our product line handles the majority of the feature model functionalities of e-health feature model from the previous chapter. It authenticates users using a username and password and utilizes a relational database.

#### E-Health Feature model:

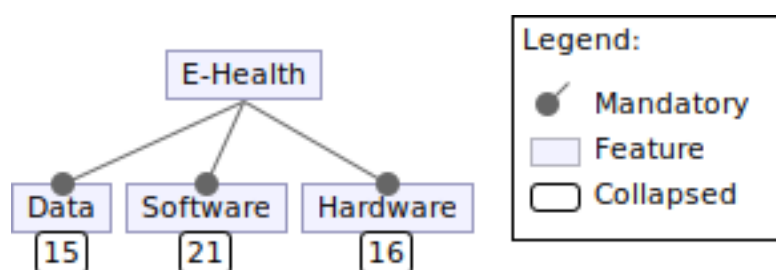


Figure 30: E-Health Feature model

### Data Feature model:

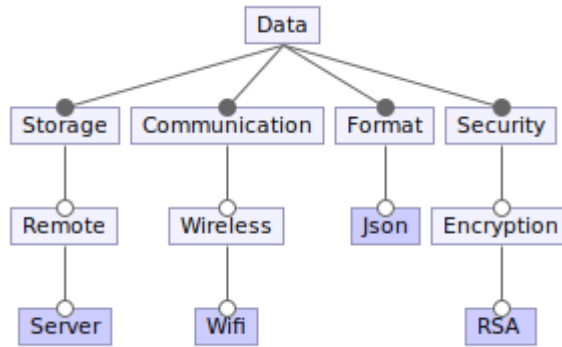


Figure 31: Data Feature model

Constraints: Remote requires Wireless.

### Software Feature model:

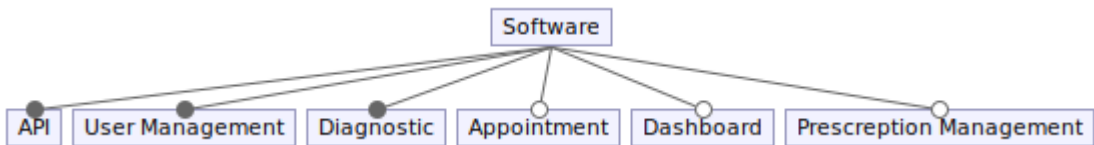


Figure 32: Software Feature model

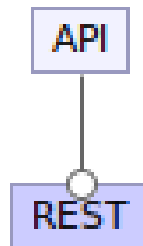


Figure 33: API Feature model

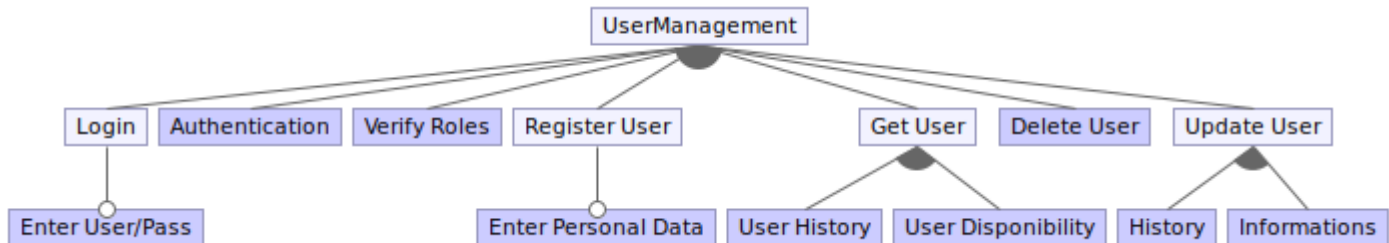


Figure 34: UserManagement Feature model

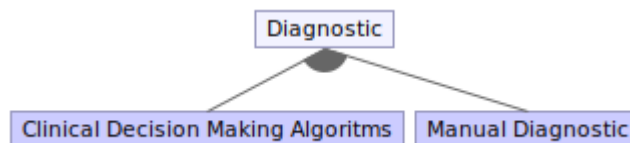


Figure 35: Diagnostic Feature model

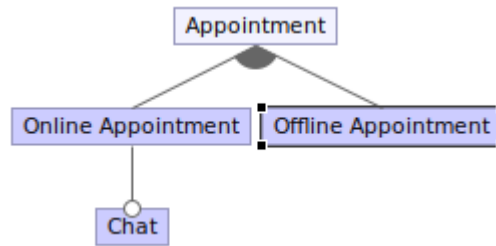


Figure 36: Appointment Feature model

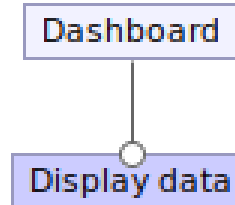


Figure 37: Dashboard Feature model

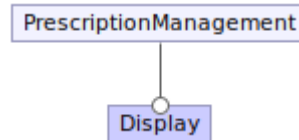


Figure 38: Prescription Feature model

**Constraints:**

- API requires Wireless.
- Login requires Authentication.
- Get User requires Login and Verify Roles.
- Update User requires Login and Verify Roles.
- Delete User requires Login.
- Diagnostic requires Medical Informations.
- Display Data hasSensor Blood Pressure.
- Display Data hasSensor Sugar Level.
- Prescription Management requires Login.

### Hardware Feature model:

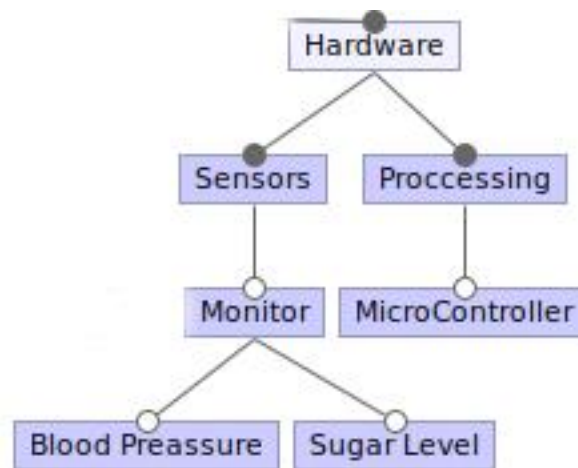


Figure 39: Hardware feature model

### Constraints:

- Sensors requires Processing.
- MicroController requires Wireless.

### 3.3.2 Home Page:

In beginning, when a visitor tries to access our application through their web browser, they land on the website's homepage.

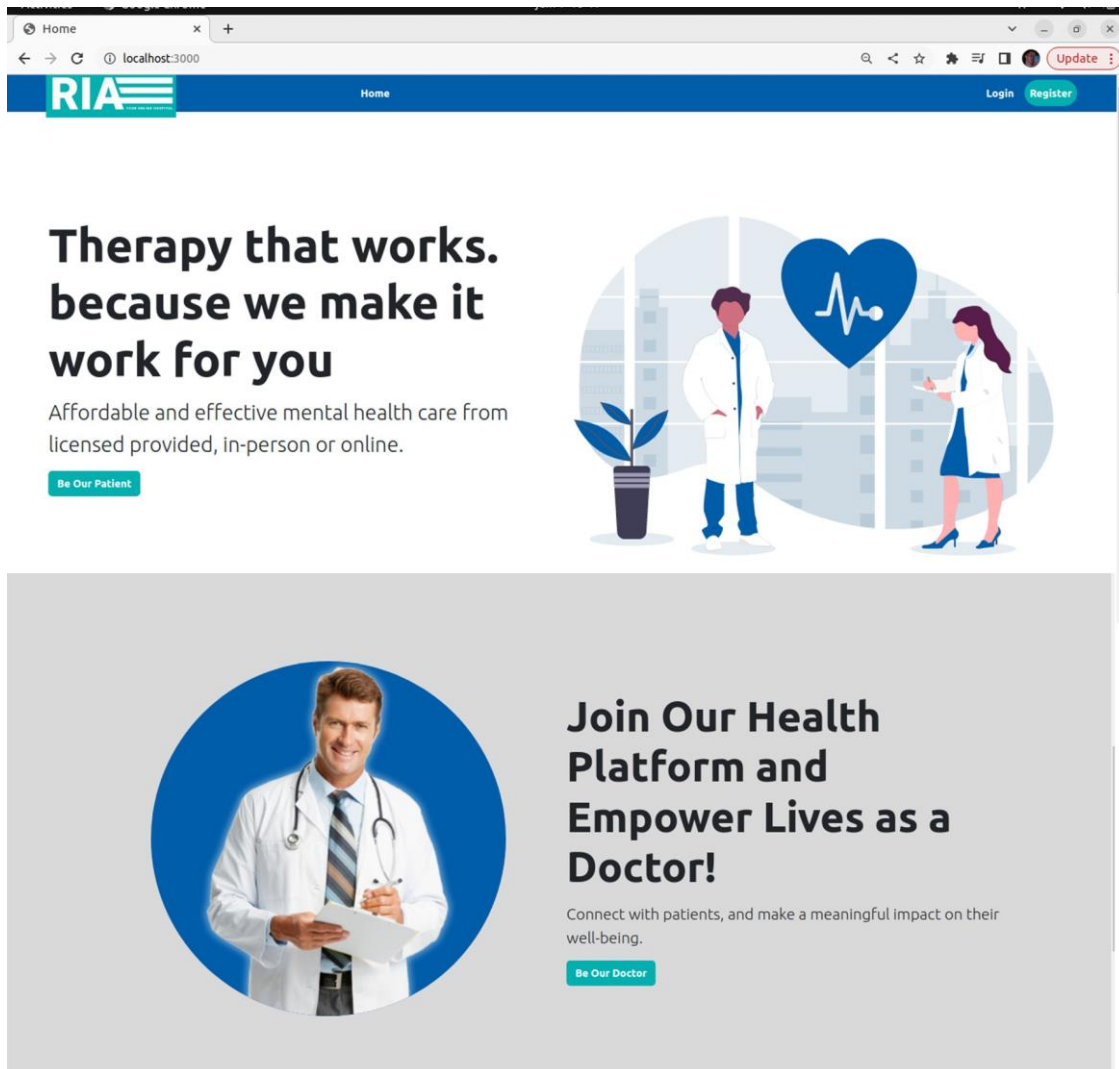


Figure 40: Home Page

If the visitor is not already registered, they can create an account by clicking on the 'Register' button on top or the 'Be Our Patient' and 'Be Our Doctor' buttons found on the homepage. These buttons will redirect them to the registration interface where they need to enter their information as shown in next Figures. They must also specify whether they want to join as a Patient or a Doctor. Otherwise, he can be redirected to login page via the login button.



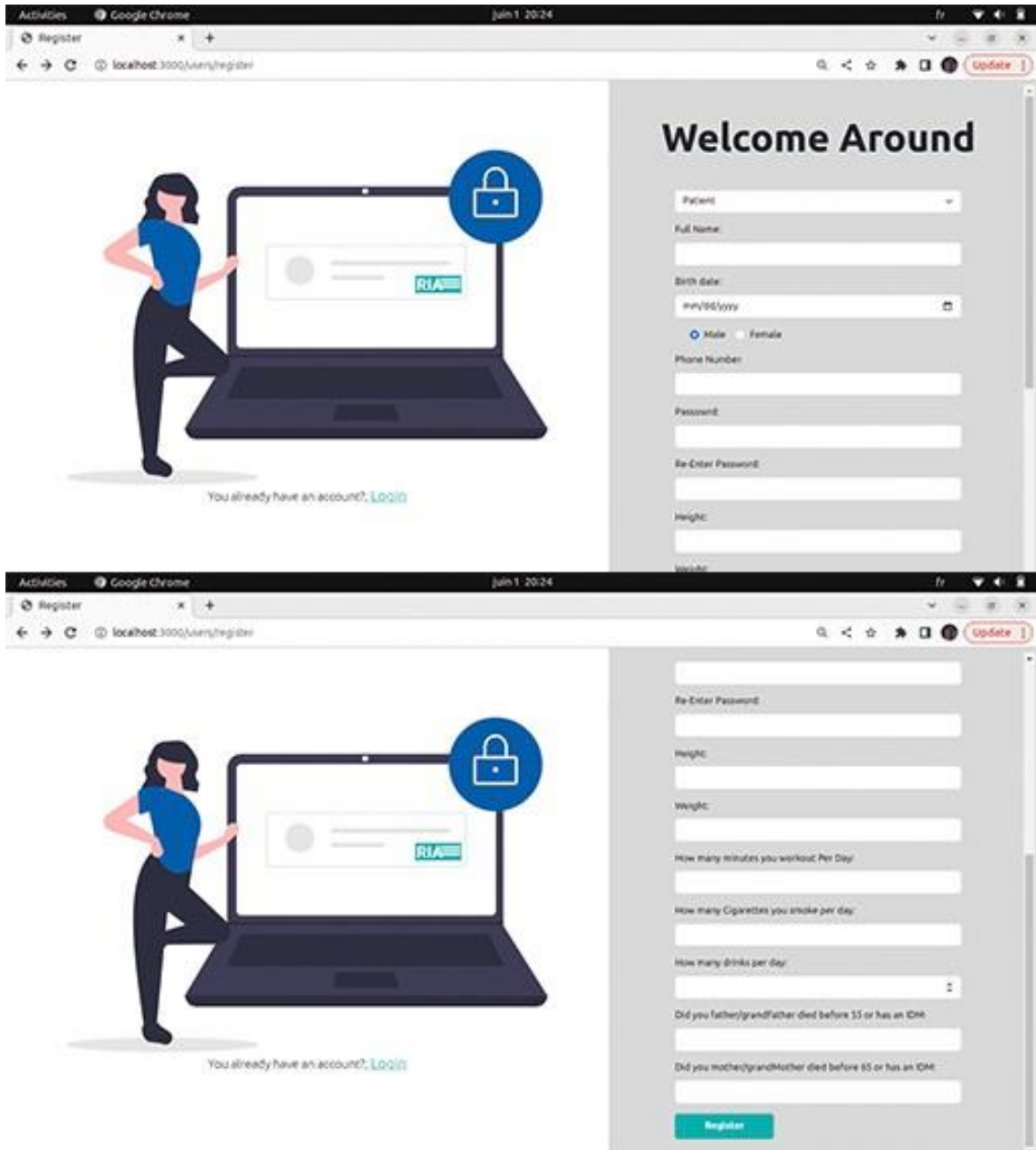


Figure 41: Patient Register Interface

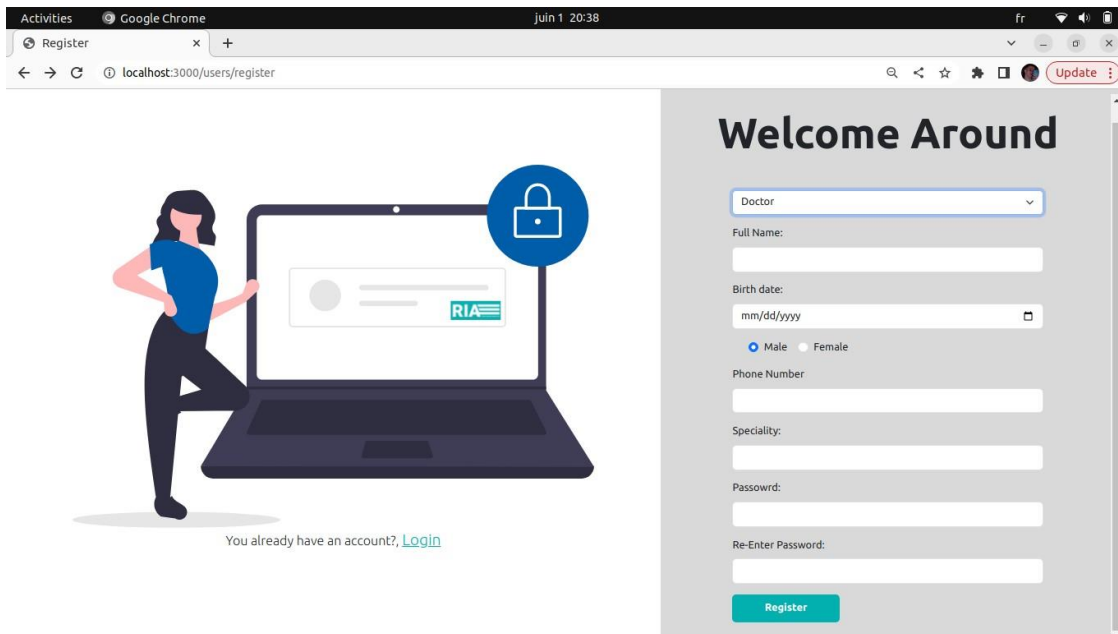


Figure 42: Doctor Register Interface

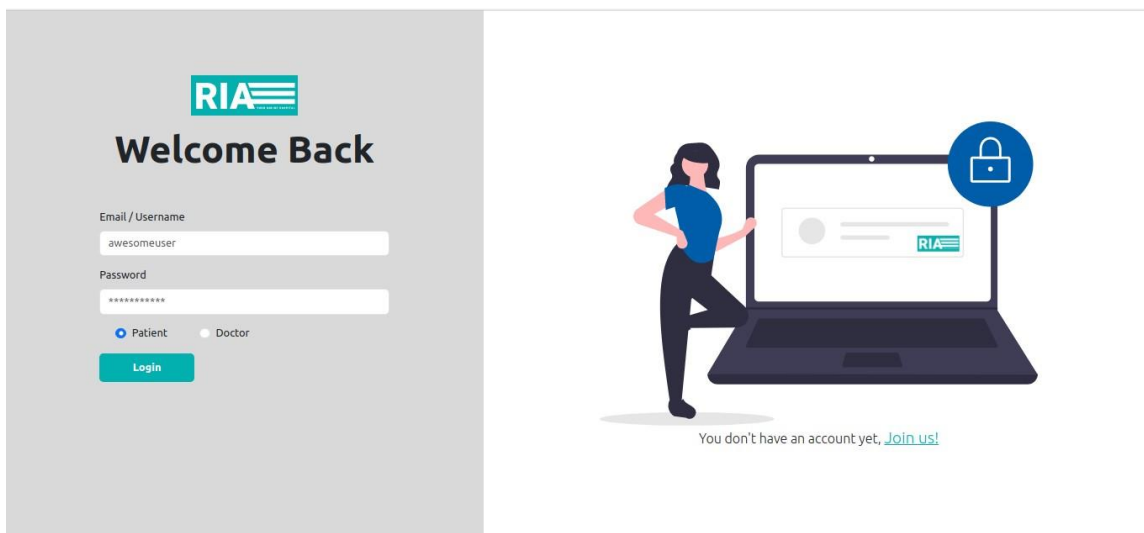


Figure 43: Login Interface

### 3.3.3 Patient Dashboard:

The patient dashboard is comprised of three distinct sections that are designed to enhance the patient's experience. Positioned on the far left, patients are provided with the capability to engage in real-time conversations with their doctors, enabling them to seek assistance and guidance as needed. The middle section is further divided into three parts, where patients can access their health data, which is transmitted by various sensors. They can also stay informed about upcoming appointments and gain access to their comprehensive medical history, empowering them to actively participate in their own care. The right portion of the dashboard consists of two sections: the first section provides an overview of the patient's cardiovascular risk factors, thereby aiding in the understanding of potential cardiovascular health concerns, while the second section

comprises a chat box. With these distinct divisions, the patient dashboard offers a comprehensive and user-friendly platform for personalized healthcare management.

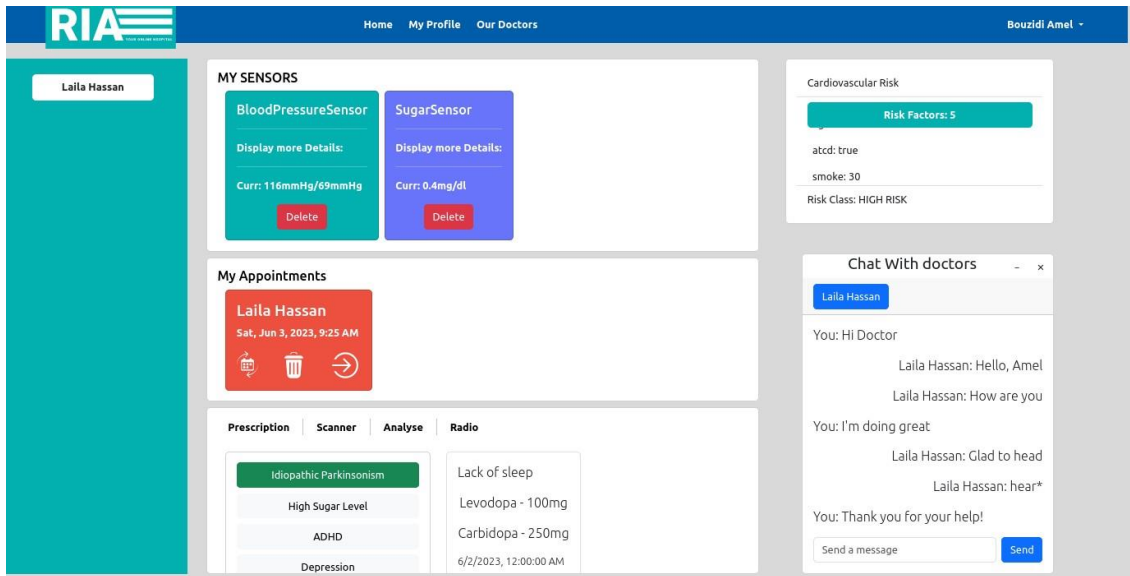


Figure 44: Patient Dashboard

### 3.3.4 Risk Factors:

This section pertains to the assessment of cardiovascular risk and serves as a significant informational asset. Within this section, both patients and medical professionals can acquire valuable knowledge regarding the diverse elements that influence the emergence of cardiovascular risk. A comprehensive presentation of these factors is provided in the table below.

**Table 3:** Cardiovascular evaluation risk factors

Parameters	Points
Gender == male	+1
Male and age > 50	+1
Female and age > 60	+1
History of Disease	+1
Cigarettes > 10/day	+1
Alcohol > 3cups/day	+1
Workout < 30min/day	+1
Sugar level > 1.2g/dl	+1

By visualizing these risk factors, patients can develop a comprehensive understanding of their cardiovascular health and it helps doctors to make informed decisions to mitigate potential risks. With this valuable information readily available, patients can take steps towards adopting healthier habits, seeking appropriate medical interventions, and ultimately reducing their risk of cardiovascular diseases.

**Evaluate patient:** The evaluate patient function calculates the cardiovascular risk of a patient based on their risk factors, categorizes them into risk classes, and returns an object with the relevant information. From the previous explication we can identify two phases:

**Count risk factors:** The count risk factors function calculates the cardiovascular risk of a patient based on their risk factors, categorizes them into risk classes, and returns an object with the relevant information.

```
async function countCardioVascularRiskFactors(patient_id) {
  let counter = 0;
  const factors = {};
  const gender = await getGender(patient_id);
  const age = await getAge(patient_id);
  const ATCD = await getATCD(patient_id);
  const cigarettePerDay = await getSmokeState(patient_id);
  const cupPerDay = await getDrinkState(patient_id);
  const minutesPerDay = await getSidentarite(patient_id);
  const sensors = await Patient.findAllSensorsMostRecentDataForOnePatient(patient_id);
  sensors.forEach(element => {
    if(element.sensor_type == 'SugarSensor') {
      const value = parseFloat(element.recorded_data_value.slice(0, 3));
      if (value > 1.2) {
        counter++;
        factors['SugarLevel'] = value;
      }
    }
  });
  if (gender == 'MALE') {
    factors['gender'] = gender;
    counter++;
    if (age > 50) {
      factors['age'] = age;
      counter++;
    }
  }
  if (gender == 'FEMALE')
    if (age > 60) counter++;
  if (ATCD.atcd) {
    counter++;
    factors['atcd'] = ATCD.atcd;
  }

  if (cigarettePerDay.cigaretteperday > 10) {
    counter++;
    factors['smoke'] = cigarettePerDay.cigaretteperday;
  }
  if (cupPerDay > 3) {
    counter++;
    factors['drink'] = cupPerDay;
  }
  if (parseInt(minutesPerDay) < 30) {
    counter++;
    factors['workout'] = minutesPerDay;
  }
  const result = {counter, factors};
  return result;
}
```

Figure 45: Count Risk Factors

**Evaluate patient:** Evaluate patient function classify the factors into three classes. LOW RISK, MODERATE RISK and HIGH RISK. The figure below highlights how this classification work.

```

async function evaluatePatient(patient_id) {
  const riskFactors = await countCardioVascularRiskFactors(patient_id);
  const { counter, factors } = riskFactors;
  let riskClass = null;
  if (counter < 1) {
    riskClass = 'NO RISK';
  } else if (counter == 1) {
    riskClass = 'LOW RISK'
  } else if (counter == 2) {
    riskClass = 'MODERATE RISK'
  } else {
    riskClass = 'HIGH RISK';
  }
  const answer = {
    counter,
    riskClass,
    factors
  }
  return answer;
}

```

Figure 46: Evaluate Patient

### 3.3.5 Patient's Profile:

The “My Profile” page helps patients to update their crucial medical information. This user-friendly interface includes a comprehensive form where patients can input essential data relevant to their health. The form covers a range of factors, such as the number of cigarettes smoked per day, alcohol consumption habits, weight, and more. These details play a crucial role in counting the individual’s cardiovascular risk. By allowing patients to update their medical information, they contribute to the evaluation of their overall heart health and enable healthcare professionals to provide tailored recommendations and interventions to mitigate potential risks.

The screenshot shows a web application interface for updating medical information. At the top, there is a blue navigation bar with the RIA logo on the left and links for 'Home', 'My Profile', and 'Our Doctors' in the center. On the right of the navigation bar, the user's name 'Bouzidi Amel' is displayed. Below the navigation bar, the main content area is titled 'My Medical Informations'. This section contains several input fields with their current values: 'Cigarette Consumption' (30), 'Alcohol Consumption' (7), 'Workout (By mins)' (60), 'Height' (165.00), and 'Weight' (84.00). At the bottom of this form, there is a green 'Update' button.

Figure 47: Update Patient’s medical information

### 3.3.6 Doctor Dashboard:

The doctor's dashboard is divided into three distinct sections. The leftmost section serves as a communication hub where doctors can chat with their patients. This chat interface allows for real-time interactions, enabling doctors to provide prompt advice, address concerns, and discuss treatment options. The doctor's appointments are in the middle part. Here, doctors can manage appointments and make necessary adjustments. It also provides an organized overview of upcoming consultations, ensuring doctors can effectively manage their time and resources. Finally, the rightmost section contains the chat box, providing a convenient space for doctors to communicate with their patients.

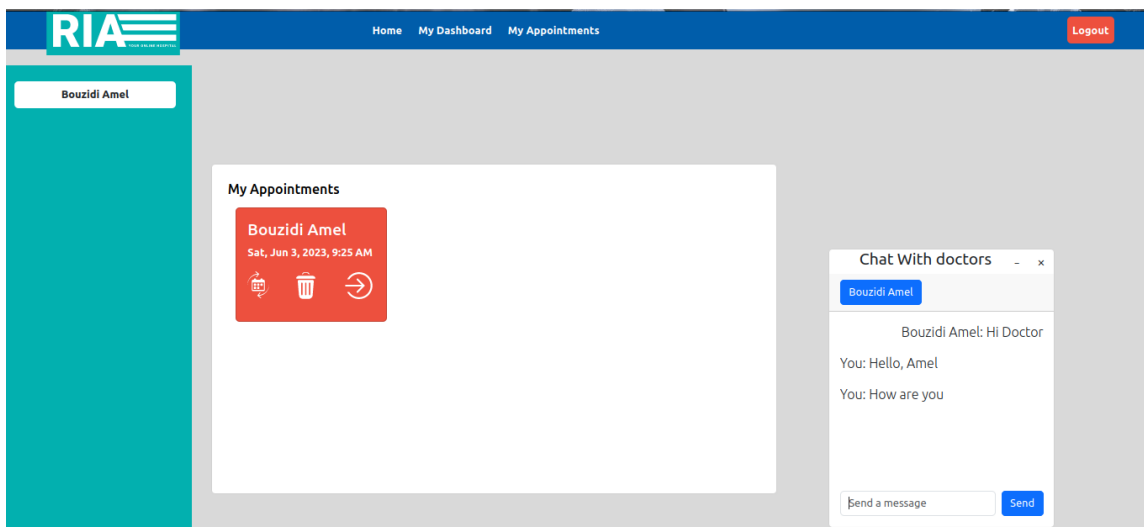


Figure 48: Doctor Dashboard

### 3.3.7 Book an Appointment with a doctor:

Patients have access to a comprehensive calendar displaying the availability of a certain doctor if they desire to book an appointment with him. This calendar allows patients to conveniently view the open slots for their preferred doctor. On the left side of the page. Patients can navigate through different dates and view the availability for each day. The right side of the page has two sections. The first section consists of essential details about the selected doctor, such as their name, specialty, and phone number. This card serves as a quick reference for patients seeking information about the doctor. The second part on the right side of the page is where patients can book their appointment. A user-friendly form prompts patients to provide the reason for their visit and select a desired date. As patients enter the date, our app checks if the selected date is in the past, not available, or available for booking. Small messages appear beneath the date field, providing helpful feedback to the patient. This streamlined process ensures that patients can easily schedule appointments with their preferred doctors, while minimizing any potential confusion or errors.

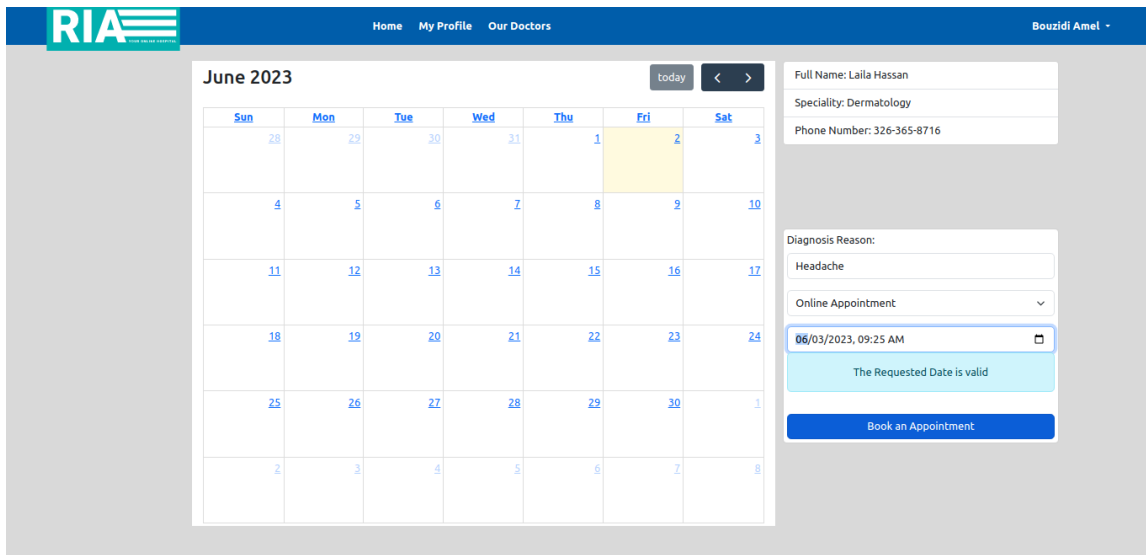


Figure 49: Patient book an appointment

In case where a patient picks a wrong date (a date in the past or an occupied date)

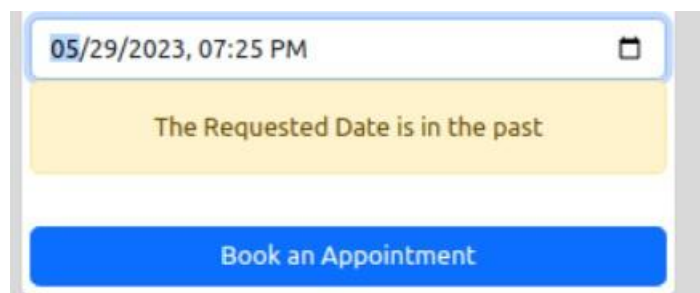


Figure 50: e.g.: example of date in the past

### 3.3.8 Appointment Room:

The “Visit Room” button, located within each appointment in the dashboard, serves as a gateway to a dedicated page where doctors and patients can conduct their appointments seamlessly. Upon clicking the button, a new page opens with two distinct sections. The first section presents the patient’s medical history, providing valuable insights for the doctor to reference during the appointment. Additionally, patients can conveniently add any treatments or medications they have received since their last visit, doctor can add treatments as well. The second section features a user-friendly chat box, facilitating real-time communication between the doctor and patient. This chat function fosters a dynamic and interactive environment, allowing both parties to discuss concerns, ask questions, and exchange information effectively.

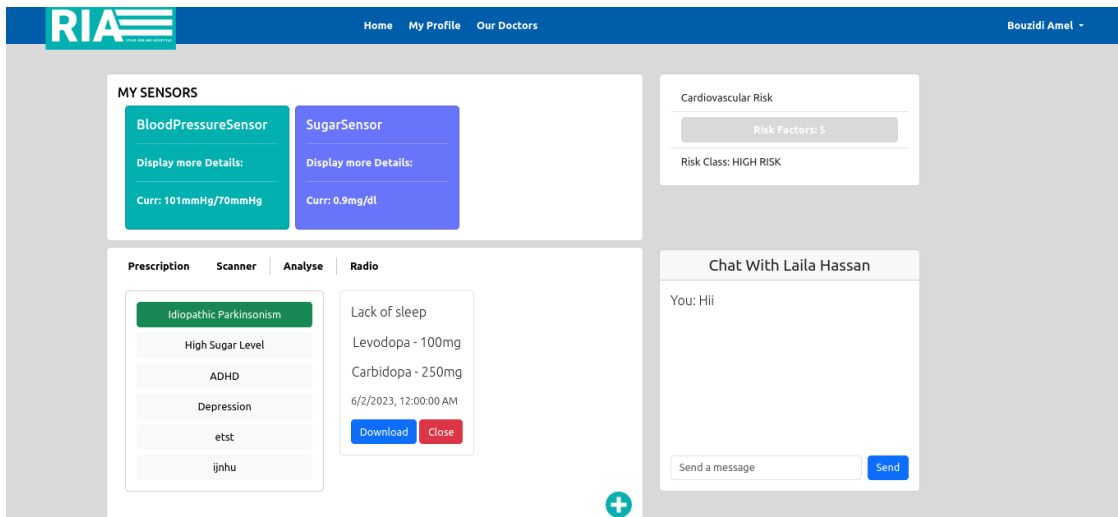


Figure 51: Appointment's Room

### 3.4 Conclusion:

In this chapter, we tried to demonstrate how the software product line approach is applied with medical sensors, which can be really time consuming at the start but so beneficial in terms of scalability and reusability when many users derive their application. We have chosen the case of cardiovascular for our final year project and it does not stop here it can be extended to include more diseases.



---

# Conclusion and perspectives

---

## 3.5 Conclusion:

In this work we conducted a literature review on software product line approaches and ontologies to acquire the basic concepts to build IoMT applications.

The principle of product lines itself is a challenge because you first think about components each member in the software product line, classify them into core assets and variables and then focus on how to abstract your code base to a point where it can be reused for the entire product line. Due to the vast medical sensors in the market the SPL approach becomes even more complex. The use of an ontology helps guide the derivation of applications which and implementing it is a challenge itself.

Our contributions in the domain engineering process included separating the software, hardware, and data components from each other to enhance their independence and reusability within the feature model representation of the software product line. Additionally, we introduced the "hasSensor" property in our feature model to establish connections between physical sensors and our application logic.

Then, we mapped the feature model to an ontology, using it as a guide to exclude complexities and ensure logical and contradiction-free derivations. The implementation process is a showcase of how to derive an application.

Finally, we can continue this work by adopting other perspectives such as:

- Automate the process of mapping from Feature Model to Ontology.
- Aim to cover a variety of evaluation with the help of deep learning algorithms. Process scanner, radios and analyses automatically.
- Automate the process of generating the features code.
- Introduce the concept of dynamic software product line to support adaptation and reconfiguration at runtime.
- Test our product line in a real-life scenario in order to test the performance.

---

# Bibliography

---

- [1] Microsoft Visual Studio Code Official Website. <https://code.visualstudio.com/>. Accessed: June 1, 2023.
- [2] Protégé project. <http://protege.stanford.edu>.
- [3] Abbas, A., Siddiqui, I., and Lee, S. U.-J. (2017). Contextual variability management of iot application with xml-based feature modelling. *Journal of Theoretical and Applied Information Technology*, 95:1300.
- [4] Alam, M. M., Khan, A., and Zafar, A. (2016). A comprehensive study of software product line frameworks. *International Journal of Computer Applications*, 115:11–17.
- [5] Atzori, L., Iera, A., and Morabito, G. (2010). The internet of things: A survey. *Computer Networks*, 54(15):2787–2805.
- [6] Author, A. and Author, B. (2021). Title of the article. *Advances in Intelligent Systems and Computing*, 114011.
- [7] Ayala, I., Pinilla, M., Fuentes, L., and Troya, J. (2015). A software product line process to develop agents for the iot. *Sensors (Basel, Switzerland)*, 15:15640–15660.
- [8] Babangida, L., Perumal, T., Mustapha, N., and Yaakob, R. (2022). Internet of things (iot) based activity recognition strategies in smart homes: A review. *IEEE Sensors Journal*.
- [9] Bachmann, F. and Clements, P. C. (2005). Variability in software product lines. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST.
- [10] Buyya, A. V. D. R. and Green, T. (2016). *Internet of Things Principles and Paradigms*. Morgan Kaufmann.
- [11] Böckle, G., Pohl, K., and Linden, F. (2005). *A Framework for Software Product Line Engineering*, pages 19–38.

- [12] Caete, A., Amor, M., and Fuentes, L. (2022). Supporting iot applications deployment on edge-based infrastructures using multi-layer feature models. *Journal of Systems and Software*, 183:111086.
- [13] Cañete, A., Amor, M., and Fuentes, L. (2021). Energy-efficient deployment of iot applications in edge-based infrastructures: A software product line approach. *IEEE Internet of Things Journal*, 8(22):16427–16439.
- [14] Clements, P. and Northrop, L. (2002). *Software product lines*. Addison-Wesley Boston.
- [15] Czarnecki, K., Hwan, C., Kim, P., and Kalleberg, K. (2006). Feature models are views on ontologies. In *10th International Software Product Line Conference (SPLC'06)*, pages 41–51. IEEE.
- [16] Dilawar, N., Rizwan, M., Ahmad, F., and Akram, S. (2019). Blockchain: securing internet of medical things (iomt). *International Journal of Advanced Computer Science and Applications*, 10(1).
- [17] Fernández-Caramés, T. and Fraga-Lamas, P. (2018). Towards the internet of smart clothing: a review on iot wearables and garments for creating intelligent connected e-textiles. *Electronics*, 7(12):405.
- [18] Filho, J. B. F., Barais, O., Baudry, B., Viana, W., and Andrade, R. M. (2012). An approach for semantic enrichment of software product lines. In *Proceedings of the 16th International Software Product Line Conference-Volume 2*, pages 188–195.
- [19] González, G., Meana-Llorián, D., Pelayo, G-Bustelo, B., Lovelle, J., and Garcia-Fernandez, N. (2017). Midgar: detection of people through computer vision in the internet of things scenarios to improve the security in smart cities, smart towns, and smart homes. *Future Generation Computer Systems*, 76:301–313.
- [20] Indrakumari, R., Poongodi, T., Suresh, P., and Balamurugan, B. (2020). *The growing role of internet of things in healthcare wearables*, pages 163–194. Academic Press.
- [21] Kang, K., Cohen, S., Hess, J., Novak, W., and Peterson, A. (1990). Feature-oriented domain analysis (foda) feasibility study.
- [22] Linden, F., Schmid, K., and Rommes, E. (2007). *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering*.
- [23] Morgan, J. (2014). A simple explanation of 'the internet of things'.
- [24] Nina, C., Trilar, J., Kos, A., Volk, M., and Duh, E. S. (2020). The use of iot technology in smart cities and smart villages: similarities, differences, and future prospects. *Sensors*, 20(14):3897.

- [25] Parnas, D. (1976). On the design and development of program families. *IEEE Transactions on Software Engineering*, SE-2(1):1–9.
- [26] Razzak, F. (2012). Spamming the internet of things: A possibility and its probable solution. *Procedia Computer Science*, 10:658–665. ANT 2012 and MobiWIS 2012.
- [27] Simmonds, J. and Bastarrica, M. (2023). Modeling variability in software process lines.
- [28] Tenório, T., Dermeval, D., and Bittencourt, I. (2014). On the use of ontology for dynamic reconfiguring software product line products. In *Proceedings of the ninth international conference on software engineering advances*, pages 545–550.
- [29] Van der Linden, F. J., Schmid, K., and Rommes, E. (2007). *Software product lines in action: the best industrial practice in product line engineering*. Springer Science & Business Media.
- [30] Vilamovska, A.-M., Hatzianreou, E., Schindler, H. R., van Oranje-Nassau, C., de Vries, H., and Krapels, J. (2009). *Study on the requirements and options for RFID application in healthcare: Identifying areas for Radio Frequency Identification deployment in health care delivery: A review of relevant literature*. RAND Corporation, Santa Monica, CA.
- [31] Wahyudianto, Budiardjo, E., and Zamzami, E. (2014). Feature modeling and variability modeling syntactic notation comparison and mapping. *Journal of Computer and Communications*, 02:101–108.
- [32] Yang, H., Lee, W., and Lee, H. (2018). Iot smart home adoption: The importance of proper level automation. *J. Sensors*, 2018:6464036:1–6464036:11.
- [33] Zaidan, A. and Zaidan, B. (2020). A review on intelligent process for smart home applications based on iot: coherent taxonomy, motivation, open challenges, and recommendations. *Artificial Intelligence Review*, 53(1):141–165.
- [34] Pohl, Klaus & Böckle, Günter & Linden, Frank. (2005). *Software Product Line Engineering: Foundations, Principles, and Techniques*. 10.1007/3-540-28901-1.
- [35] Madakam, S. , Ramaswamy, R. and Tripathi, S. (2015) Internet of Things (IoT): A Literature Review. *Journal of Computer and Communications*, **3**, 164-173. doi: [10.4236/jcc.2015.35021](https://doi.org/10.4236/jcc.2015.35021).
- [36] [https://play.google.com/store/apps/details?id=com.mytelemed.android&hl=en\\_US&pli=1](https://play.google.com/store/apps/details?id=com.mytelemed.android&hl=en_US&pli=1)