MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
DR. YAHIA FARES UNIVERSITY OF MEDEA
FACULTY OF SCIENCES
MATHEMATICS AND COMPUTER SCIENCE DEPARTMENT



MASTER THESIS

Presented in order to obtain the

MASTER DEGREE IN COMPUTER SCIENCE

Specialty: Systems Engineering and Web Technology

Presented by:

Kori Yahia Sid Ali

Daci Mohamed Salah Eddine

Theme

Design and implementation of a web interface for a collaborative WoT platform

Supervisor:

Dr. Tiberkak Allal

Examining board:

Mr. MAZARI Redha

Mr. TOULAIT Sofiane

Academic year: 2019/2020

Abstract

Web, referring to the World Wide Web (WWW), is a well acceptable platform that enables the global communication. Its original intention is to provide the “universe of network accessible information”. Situated on top of the Internet, Web allows users to present and exchange text, video and other multimedia information and to collaborate between each others.

New trend is Web of Things (WoT) where the open Web standards are supported for information sharing and device interoperation. By penetrating smart things into existing Web, the conventional web services are enriched with physical world services. This WoT vision enables a new way of narrowing the barrier between virtual and physical worlds.

The objective of our study is to set up a web platform respecting the ergonomic criteria of human-machine interface to ensure good management of connected objects and ensure good communication and collaboration between users, whether by audio or video chat.

Keywords: Web, Web of Things, Human Machine Interface, collaborative platform.

ملخص

تعد الويب التي تشير إلى شبكة الويب العالمية (WWW) ، نظاما أساسيا يتيح الاتصال العالمي. هدفها الأصلي هو توفير عالم المعلومات التي يمكن الوصول إليها عبر الشبكة . تقع على قمة الإنترنت ، تتيح الويب للمستخدمين تقديم وتبادل النصوص والفيديوهات ومعلومات الوسائط المتعددة الأخرى والتعاون بين بعضهم البعض.

الاتجاه الجديد هو ويب الاشياء Web of Things (WoT) حيث يتم دعم معايير الويب المفتوحة لمشاركة المعلومات والتشغيل البيني للجهاز. من خلال دمج الأشياء الذكية في الويب الحالي . يتم إثراء الخدمات بخدمات العالم المادي. تتيح رؤية ويب الاشياء WoT هذه طريقة جديدة لتقليص الحاجز بين العالمين الافتراضي والمادي.

الهدف من دراستنا هو إنشاء منصة ويب تحترم المعايير المريحة لواجهة الإنسان والآلة لضمان الإدارة الجيدة للكائنات المتصلة وضمان التواصل الجيد والتعاون بين المستخدمين، سواء عن طريق الدردشة الصوتية أو المرئية.

الكلمات المفتاحية: الويب ، ويب الأشياء ، واجهة الإنسان والآلة ، منصة تعاونية.

Résumé

Web, faisant référence au World Wide Web (WWW), est une plate-forme bien acceptable qui permet la communication globale. Son intention initiale est de fournir «l'univers de l'information accessible par réseau». Situé au sommet d'Internet, le Web permet aux utilisateurs de présenter et d'échanger du texte, des vidéos et d'autres informations multimédias et de collaborer entre eux.

La nouvelle tendance est le Web of Things (WoT), où les normes Web ouvertes sont prises en charge pour le partage d'informations et l'interopérabilité des appareils. En pénétrant les objets intelligents dans le Web existant, le Web conventionnel les services sont enrichis de services mondiaux physiques. Cette vision de WoT permet une nouvelle façon de réduire la barrière entre les mondes virtuel et physique.

L'objectif de notre étude est de mettre en place une plateforme web respectant les critères ergonomiques d'interface homme-machine pour assurer une bonne gestion des objets connectés et assurer une bonne communication et collaboration entre les utilisateurs, que ce soit par chat audio ou vidéo.

Mot-clés : Web, Web des objets, interface homme-machine, plateforme collaborative.

Acknowledgement

First and foremost, we would like to thank God Almighty for giving us the strength, knowledge, ability and opportunity to undertake this research study and to persevere and complete it satisfactorily. Without his blessings, this achievement would not have been possible. In our journey towards this degree, we have found a teacher, a friend, an inspiration, a role model and a pillar of support in our Guide, Dr. "Tiberkak Allal". He has been there providing his heartfelt support and guidance at all times and has given as invaluable guidance, inspiration and suggestions in our quest for knowledge. He has given as all the freedom to pursue our research, while silently and non-obtrusively ensuring that we stay on course and do not deviate from the core of our research. Without his able guidance, this thesis would not have been possible and we shall eternally be grateful to him for his assistance. Finally, we would like to express our sincere thanks to all teachers of mathematics and computer science and everyone who has been involved in this work from near and far.

Dedications

I dedicate this modest work to:

My dear parents for their patience, their love, their Support and encouragement.

To my dear brother and my dear sister.

To all my family, and my friends,

And to all those who have contributed from near or far to make

This project possible

I say thank you.

Kori Yahia Sid Ali

Dedications

I dedicated This work To my dear parents

For everything you gave me

For everything You did for me

To my dear brothers and sisters

No words can describe your dedication and sacrifice.

To all my friends, As a testament to the sincere friendship that has bound us and the good times we have spent together.

I dedicate this work to you

Wishing for a bright future full of good promises.

To all the people who believed in me and who make me want to go forward.

I say Thank you.

Daci Mohamed Salah Eddine

Contents

Abstract i

Acknowledgement iv

Dedications v

Dedications vi

General introduction **1**

I Prerequisites **3**

1.1 Introduction 3

1.2 Internet of Things 3

1.2.1 IoT history 3

1.2.2 Nature of Things 4

1.2.3 IoT architecture 4

1.2.4 IoT protocols and technologies 6

1.2.4.1 MQTT 6

1.2.4.2 COAP 6

1.2.4.3 Bluetooth 6

1.2.4.4 WiFi 7

1.2.4.5 ZigBee 7

1.2.5 Platforms used to link IoTs to the Internet 7

1.2.5.1 Arduino 7

1.2.5.2 Raspberry Pi 8

1.3 Web of Things 9

1.3.1 Web of Things architecture 9

1.3.1.1 Access layer 10

1.3.1.2 Find layer 11

1.3.1.3 Share layer 11

1.3.1.4 Compose layer 11

1.3.2 WoT technologies and protocols 12

1.3.2.1 WebSockets 12

1.3.2.2 SSE (Server Send Event) 12

1.3.2.3 HTTP . . . 12
1.3.2.4 RESTful API . . . 13
1.3.2.5 Web Real-Time Communication . . . 13
1.4 IoT vs WoT . . . 13
1.5 Conclusion . . . 14

II Collaborative platforms **15**
2.1 Introduction . . . 15
2.2 Categories of collaborative platforms . . . 15
2.2.1 Synchronous or real-time collaboration software . . . 15
2.2.2 Asynchronous or non-real time collaboration software . . . 15
2.3 Types of collaborative platform . . . 16
2.3.1 Communication . . . 17
2.3.2 Conferencing . . . 17
2.3.3 Coordination . . . 17
2.4 Examples of collaborative Platforms . . . 17
2.4.1 Skype . . . 17
2.4.2 Mockplus . . . 17
2.4.3 Google Docs . . . 18
2.4.4 Dropbox . . . 18
2.4.5 Trello . . . 18
2.4.6 ProofHub . . . 18
2.4.7 Collaborative platform for managing IoTs using WebRTC . . . 19
2.4.7.1 General architecture . . . 19
2.5 Impact of collaborative platforms in companies . . . 20
2.6 Conclusion . . . 20

III Human Machine Interface **21**
3.1 Introduction . . . 21

3.2 Definition . . . 21
3.3 List of Ergonomic Criteria . . . 22
3.3.1 Guidance . . . 22
3.3.1.1 Prompting . . . 22
3.3.1.2 Grouping/Distinction of Items . . . 23
3.3.1.3 Immediate Feedback . . . 23
3.3.1.4 Legibility . . . 24
3.3.2 Workload . . . 24
3.3.2.1 Brevity . . . 24
3.3.2.1.1 Concision . . . 25
3.3.2.1.2 Minimal Actions . . . 25
3.3.2.2 Information Density . . . 25
3.3.3 Explicit Control . . . 26
3.3.3.1 Explicit User Action . . . 26
3.3.3.2 User Control . . . 26
3.3.4 Adaptability . . . 27
3.3.4.1 Flexibility . . . 27
3.3.4.2 User Experience . . . 27
3.3.5 Error Management . . . 28
3.3.5.1 Error Protection . . . 28
3.3.5.2 Quality of Error Messages . . . 28
3.3.5.3 Error Correction . . . 29
3.3.6 Consistency . . . 29
3.3.7 Significance of codes . . . 29
3.3.8 Compatibility . . . 30
3.4 Conclusion . . . 31

IV Design **32**

4.1 Introduction . . . 32

4.2 Presentation of our platform . . . 32
4.3 Platform architecture . . . 32
4.4 Development methodology . . . 33
4.4.1 Definition of Unified Process . . . 33
4.4.2 Unified Process phases . . . 33
4.4.2.1 Inception Phase . . . 34
4.4.2.2 Elaboration Phase . . . 34
4.4.2.3 Construction Phase . . . 35
4.4.2.4 Transition Phase . . . 35
4.4.3 Unified Process activities . . . 36
4.4.3.1 Business modeling . . . 36
4.4.3.2 Requirements . . . 36
4.4.3.3 Analysis and design . . . 36
4.4.3.4 Implementation . . . 37
4.4.3.5 Test . . . 37
4.4.3.6 Deployment . . . 37
4.5 Modeling Language . . . 37
4.5.1 UML . . . 37
4.5.2 UML diagrams . . . 37
4.5.3 Used UML diagrams . . . 39
4.6 Use case diagram . . . 39
4.6.1 Identification of the use cases . . . 40
4.6.2 Textual description of use cases . . . 42
4.6.2.1 Use case N°1: Authenticate . . . 42
4.6.2.2 Use case N°2: Management of users . . . 43
4.6.2.3 Use case N°3: Management of COs . . . 44
4.6.2.4 Use case N°4: Discover of COs . . . 45
4.6.2.5 Use case N°5: Invoke action . . . 45
4.6.2.6 Use case N°6: Notify users . . . 46

4.6.2.7 Use case N°7: Communicate 46
4.7 Sequence diagram 46
4.7.1 Sequence diagram of the use case "Authenticate" 47
4.7.2 Sequence diagram of the use case "Management of users" 48
4.7.3 Sequence diagram of the use case "Management of COs" 52
4.7.4 Sequence diagram of the use case "Discover of COs" 55
4.7.5 Sequence diagram of the use case "Invoke action" 56
4.7.6 Sequence diagram of the use case "Notify users" 57
4.7.7 Sequence diagram of the use case "Communicate" 58
4.8 Class diagram 59
4.9 HMI 60
4.9.1 Authentication (Login) 60
4.9.2 Navigation 62
4.9.3 Customization 62
4.9.4 Communication 63
4.9.4.1 Messages 63
4.9.4.2 Video calls 63
4.9.5 Home Pages 65
4.9.6 Profiles 66
4.10 Conclusion 67

V Implementation **68**

5.1 Introduction 68
5.2 Software tools 68
5.2.1 Languages 68
5.2.1.1 HTML5 68
5.2.1.2 CSS3 68
5.2.1.3 JavaScript 68
5.2.2 Servers 69

5.2.2.1 Node.js . . . 69
5.2.2.2 MySQL . . . 69
5.2.3 Editors . . . 69
5.2.3.1 Visual Studio Code . . . 69
5.2.3.2 StarUML . . . 69
5.2.3.3 Adobe XD . . . 70
5.3 Hardware tools . . . 70
5.3.1 Laptop . . . 70
5.3.2 Arduino Mega 2560 . . . 70
5.3.3 Router . . . 70
5.4 Platform interface . . . 71
5.4.1 Login pages . . . 71
5.4.2 Home/Dashboard pages . . . 72
5.4.3 Chat pages . . . 74
5.4.4 CO pages . . . 75
5.5 Conclusion . . . 76

General conclusion **77**

References **82**

List of Figures

1.1 Nature of Things in IoT. 4
1.2 IoT architecture. 5
1.3 Arduino board. 8
1.4 Raspberry Pi board. 9
1.5 Web of Things architecture. 10

2.1 Types of collaborative platforms. 16
2.2 Global architecture of collaorative platforms 19

3.1 Human-machine system shown as a closed control loop. 22

4.1 Platform architecure. 32
4.2 A software development process. 33
4.3 Unified Process phases 34
4.4 Unified process lifecycle 36
4.5 The hierarchy of UML 2.5 diagrams 38
4.6 Use case Diagram 41
4.7 Sequence diagram of the use case "Authenticate". 47
4.8 Sequence diagram of the use case "Management of users". 48
4.9 Sequence diagram of the operation add in the use case "Management of users". . 49
4.10 Sequence diagram of the operation delete in the use case "Management of users". 50
4.11 Sequence diagram of the operation modify in the use case "Management of users". 51
4.12 Sequence diagram of the use case "Management of COs". 52
4.13 Sequence diagram of the operation add in the use case "Management of COs". . . 53

4.14 Sequence diagram of the operation delete in the use case "Management of COs". 54
4.15 Sequence diagram of the operation modify in the use case "Management of COS". 55
4.16 Sequence diagram of the use case "Discover of COs". 56
4.17 Sequence diagram of the use case "Invoke action". 57
4.18 Sequence diagram of the use case "Notify users". 58
4.19 Sequence diagram of the use case "Communicate". 59
4.20 Class diagram. 60
4.21 Authentication 61
4.22 Mobile Navbar 62
4.23 Customization 63
4.24 Chat section 64
4.25 Video Call Section 64
4.26 Home/Admin pages 66
4.27 profiles pages 67

5.1 Arduino Mega 2560 70
5.2 Login page mobile Web-application 71
5.3 Login page disktop Web-application 72
5.4 Dashboard page mobile Web-application 72
5.5 Home page desktop 73
5.6 Dashboard page desktop Web-application 73
5.7 Chat pages mobile Web-application 74
5.8 Camera as a CO 75
5.9 Edit CO page Desktop 76

List of Tables

1.1 Comparison between IoT and WoT. 14

4.1 Used UML diagrams 39
4.2 Identification of use cases. 40
4.3 Description of the use case "Authenticate". 42
4.4 Description of the use case "Management of users". 43
4.5 Description of the use case "Management of COs". 44
4.6 Description of the use case "Discover of COs". 45
4.7 Description of the use case "Invoke actions". 45
4.8 Description of the use case "Notify users". 46
4.9 Description of the use case "Communicate". 46

Abreviations List

API . . . . . Application Programming Interface.

CD . . . . . Compact Disc.

CO . . . . . Connected Object.

COAP . . . . . Constrained Application Protocol.

CoRE . . . . . Constrained RESTful Environments.

CSS . . . . . Cascading Style Sheet.

DVD . . . . . Digital Versatile Disc or Digital Video Disc.

FIFO . . . . . First In First Out.

HDMI . . . . . High Definition Multimedia Interface.

HMI . . . . . Human-Machine Interface.

HTML . . . . . HyperText Markup Language.

HTML5 . . . . . HyperText Markup Language 5.

IETF . . . . . Internet Engineering Task Force.

IoT . . . . . Internet of Things.

IP . . . . . Internet Protocol.

JSON-LD . . . . . JavaScript Object Notation for Linked Data.

LED . . . . . Light Emitting Diode.

LTR . . . . . Left To Right.

MDA . . . . . Model Driven Architecture.

MMI . . . . . Man-Machine Interface.

MQTT . . . . . Message Queue Telemetry Transport.

M2M . . . . . Machine to Machine.

NFC . . . . . Near Field Communication.

OMG . . . . . Object Management Group

OSI . . . . . Open System Interconnection.

PIM . . . . . Personal Information Managers

QR . . . . . Quick Response.

REST . . . . . Representation State Transfer.

RFID . . . . . Radio Frequency Identification.

RTL . . . . . Right To Left.

SDK . . . . . Software Development Kit.

SQL . . . . . Structured Query Language.

SSE . . . . . Server Send Event.

TCP . . . . . Transmission Control Protocol.

TLS . . . . . Transport Layer Security.

UML . . . . . Unified Modeling Language.

UP . . . . . Unified Process.

USB . . . . . Universal Serial Bus.

WiFi . . . . . Wireless Fidelity.

WoT . . . . . Web of Things.

WSN . . . . . Wireless Sensor Networks.

WWW . . . . . World Wide Web.

W3C . . . . . World Wide Web Consortium.

ZigBee . . . . . Zonal intercommunication Global-Standard.

General introduction

For a few years now smart devices that are connected to the internet and communicate with each other are no longer a fantasy only existing in science function movies. Nowadays there are various devices with sensors and actuators available from different manufacturers that can measure and interact with their surroundings and can be used in home automation, smart cities and many other fields. These gadgets include many items such as smart light bulbs, temperature or air quality sensors, wireless buttons or smart power switches [1].
Currently, the web has already become the major medium of communication and collaboration to share data, make audio or video calls and work from different locations in today's Internet. On the other hand, tiny web server technology has been researched for decades and now various embedded tiny web servers are available. More specifically, web services have been proven to be indispensable in creating interoperable applications on today's Internet. Smart things with embedded web servers can be abstracted as web services and seamlessly integrated into the existing web.
It is natural to reuse existing web technologies and standards to unify the cyber-world and the physical-world. As a result, one research trend treats IoT as Web of Things (WoT) [2].

Problematic

- Connected Object often use different, proprietary communication protocols which are not interoperable.
- No communication between users and no collaboration when managing a connected object, current collaboration tools have problems and must be installed.
- Normal users have a lot of difficulty when interacting with websites and platforms

Objective

The objective of this work is to provide a simplified collaborative WoT platform that respect Ergonomic Criteria of Human Machine Interface, that with it we can manage several Connected Object in collaboration between users across the Web with a simple design.

Workplan

this work is made up of five chapters:

- The first chapter is prerequisites, We will talk about the different concepts in IoT and WoT, history, definition, architecture, technologies and protocols.
- The second chapter is collaborative platform, It will be devoted to the study of collaboration platform, Categories, types, examples and Impact of collaborative platforms in companies.
- The third chapter is Human Machine Interface, it will talk about definition of Human Machine Interface and list of ergonomic criteria.
- Chapter 4 provides a detailed overview of the application we created by introducing our application, giving structure to it, choosing a development methodology and suggesting a UML design using various diagrams (use case diagram, sequence diagram, class diagram), and state diagram a diagram.
- The last chapter is the implementation of our solution developed in the previous chapter.

Chapter I

Prerequisites

1.1 Introduction

The disruption of the digital world by the massive arrival of Connected Objects (CO), made that the Internet is undergoing a major revolution, full of opportunities for all users. There are our daily perception of the real world and our social world, in which we live, we share and we consume, because of that two technologies appeared Internet of Things (IoT) and Web of Things (WoT).

1.2 Internet of Things

The IoT is a system of physical objects that can be discovered, monitored, controlled, or interacted with by electronic devices that communicate over various networking interfaces and eventually can be connected to the wider Internet [3]. IoT used in many fields like logistics, health, alimentary, Automation, industry, etc.
Like every technology, IoT has a lot of things to discuss in:

1.2.1 IoT history

The term IoTs is 16 years old. But the actual idea of connected devices had been around longer, at least since the 70s. Back then, the idea was often called "embedded Internet" or "pervasive computing". But the actual term "Internet of Things" was coined by Kevin Ashton[1] in 1999 during his work at Procter&Gamble[2]. Ashton who was working in supply chain optimization, wanted to attract senior management's attention to a new exciting technology called RFID (Radio Frequency Identification). Because the Internet was the hottest new trend in 1999 and because it somehow made sense, he called his presentation "Internet of Things". Even though Kevin grabbed the interest of some P&G [2] executives, the term IoT did not get widespread attention for the next 10 years [4].

[1] is a British technology pioneer who cofounded the Auto-ID Center at the Massachusetts Institute of Technology.

[2] The Procter & Gamble Company (P&G) is an American multinational consumer goods corporation headquartered in Cincinnati,Ohio,Founded in 1837 by William Procter and James Gamble.

1.2.2 Nature of Things

Research in the Internet of Things usually considers Things in the broad sense of physical objects. Things can include (but is not limited to) tagged objects (RFID, NFC(Near Field Communication), QR(Quick Response) codes, Barcodes, Image Recognition) to WSN (Wireless Sensor Networks), machines, vehicles and consumer electronics [5] as we illustrate in "Figure 1.1" [6].

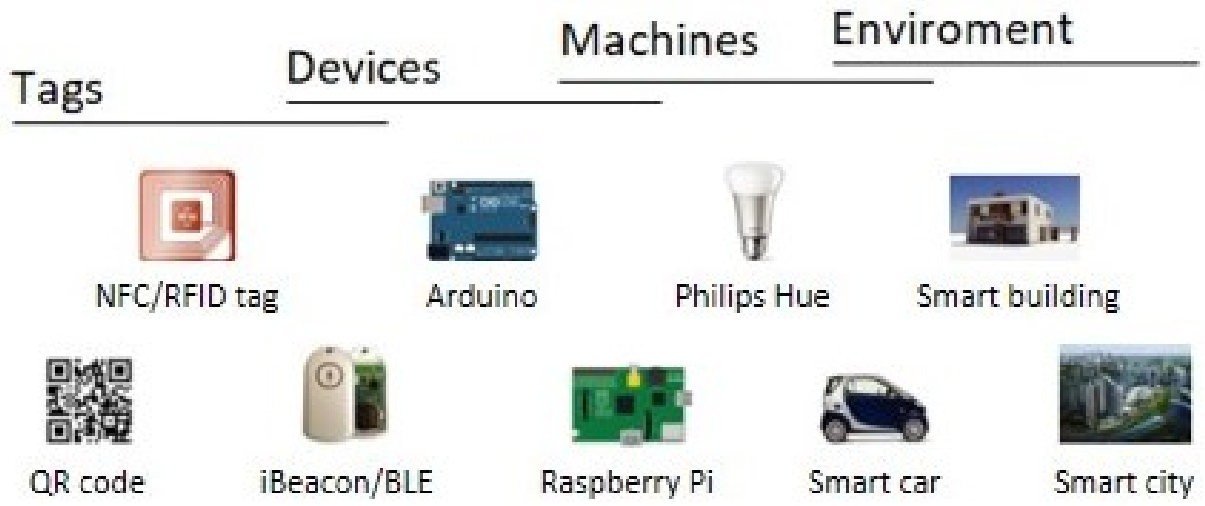


Figure 1.1: Nature of Things in IoT.

1.2.3 IoT architecture

IoT architecture consists of different layers of technologies supporting IoT. It serves to illustrate how various technologies relate to each other and to communicate the scalability, modularity and configuration of IoT deployments in different scenarios.

The lowest layer is made up of smart objects integrated with sensors. The sensors enable the interconnection of the physical and digital worlds allowing real-time information to be collected and processed. The next layer consists of IoT data acquisition systems and gateways that collect the great mass of unprocessed data, convert it into digital streams, filter and pre-process it so that it is ready for analysis. The third layer is represented by The management service renders the processing of information possible through analytics, security controls, process modeling and

management of devices. The IoT application (fourth layer) covers “smart” environments/spaces in domains such as:

Transportation, Building, City, Lifestyle, Retail, Agriculture, Factory, Supply chain, Emergency, Healthcare, User interaction, Culture and tourism, Environment and Energy [7].

We illustrate in “Figure 1.2” [7] the four layers of IoT architecture from the first layer to the last layer:

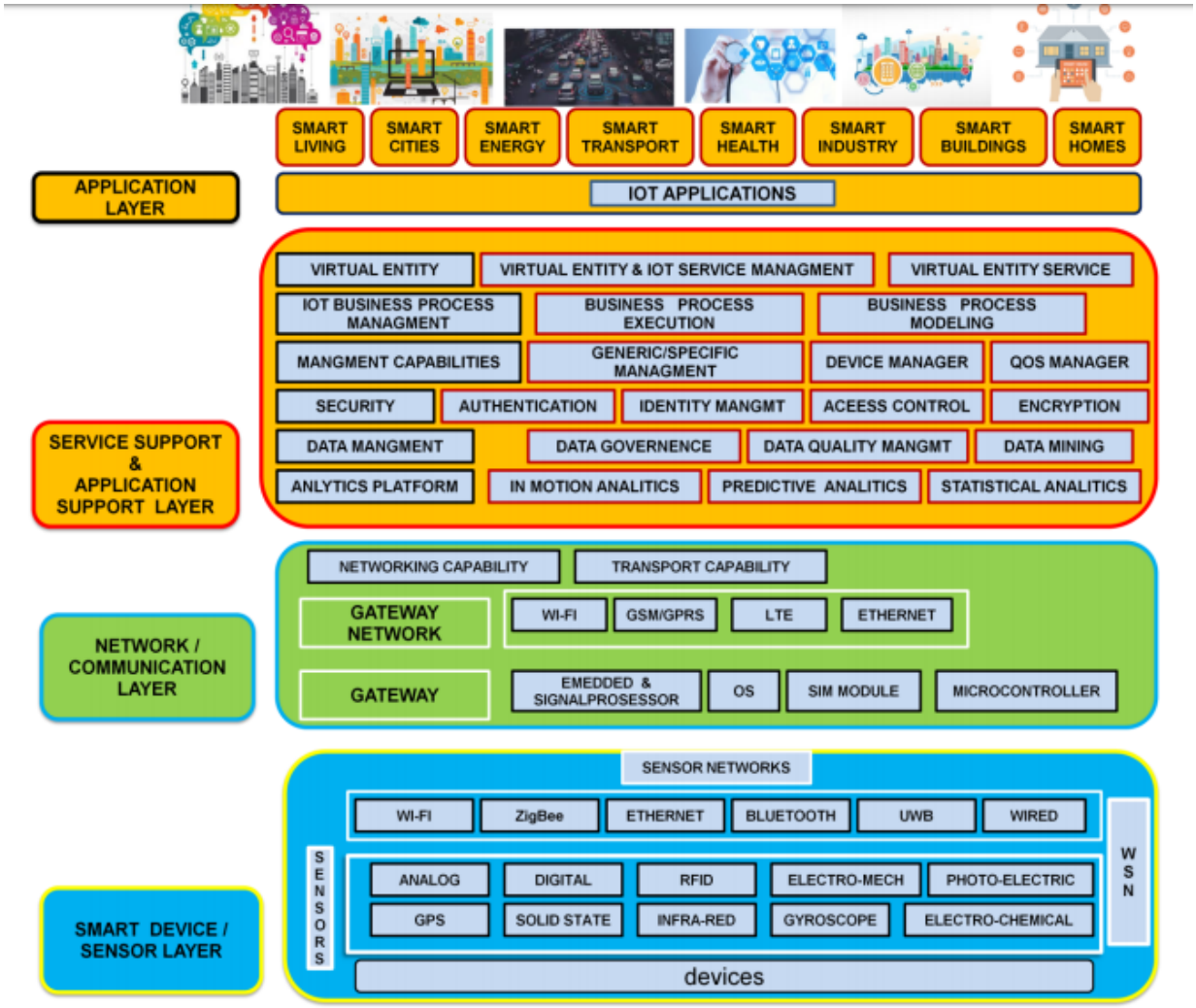


Figure 1.2: IoT architecture.

1.2.4 IoT protocols and technologies

IoT protocols are a crucial part of the IoT technology stack without them, hardware would be rendered useless as the IoT protocols enable it to exchange data in a structured and meaningful way.

but we mention just the most protocols and technologies commonly used:

1.2.4.1 MQTT

MQTT (Message Queuing Telemetry Transport Protocol) is one of the oldest Machine to Machine (M2M) communication protocols, which was introduced in 1999. It was developed by Andy Stanford-Clark[3] of IBM[4] and Arlen Nipper[5] of Arcom Control Systems Ltd (Eurotech). It is a publish/subscribe messaging protocol designed for lightweight M2M communications in constrained networks [8].

1.2.4.2 COAP

CoAP (Constrained Application Protocol) is a lightweight M2M protocol from the Internet Engineering Task Force[6] (IETF) Constrained RESTful Environments[7] (CoRE) Working Group. CoAP supports both request/response and resource/observe (a variant of publish/subscribe) architecture [8].

1.2.4.3 Bluetooth

Bluetooth is a universal radio interface in the 2.45 GHz frequency band that enables portable electronic devices to connect and communicate wirelessly via short-range, ad hoc networks. Each unit can simultaneously communicate with up to seven other units per piconet. Moreover, each unit can simultaneously belong to several piconets [9].

[3]is the Chief Technology Officer for IBM in UK and Ireland. He is an IBM Distinguished Engineer and Master Inventor with more than 40 patents.

[4]International Business Machines (IBM), is a global technology company that provides hardware, software, cloud-based services and cognitive computing.

[5]is one of the early architects of pervasive computing and the Internet of Things and co-invented MQTT.

[6]is the leading Internet standards body. It develops open standards through open processes with one goal in mind is to make the Internet work better.

[7]is an IETF working that provides a framework for REST applications intended to run on constrained IP networks.

1.2.4.4 WiFi

WiFi (Wireless Fidelity) is a wireless networking technology that allows devices such as computers (laptops and desktops), mobile devices (smart phones and wearables), and other equipment (printers and video cameras) to interface with the Internet. It allows these devices to exchange information with one another, creating a network [10].

1.2.4.5 ZigBee

ZigBee (Zonal intercommunication Global-Standard) is a network- and application-level wireless standard. It makes use of physical and medium access control layers. It provides data rates of up to 250 kbits/s [11].

1.2.5 Platforms used to link IoTs to the Internet

There is a several platforms used to link IoT with the Internet, we mention just two, Arduino and Raspberry Pi.

1.2.5.1 Arduino

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs (Light on a sensor, a finger on a button, etc.) and turn it into an output (activating a motor, turning on an LED (Light Emitting Diode), publishing something online) as “Figure 1.3” [12] illustrates. You can tell your board what to do by sending a set of instructions to the micro-controller on the board [13].

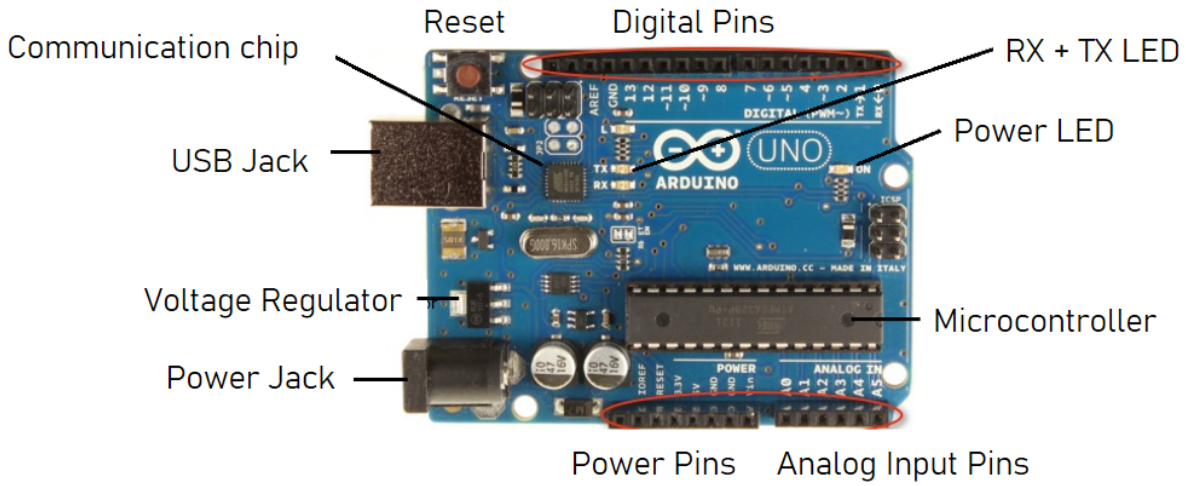


Figure 1.3: Arduino board.

1.2.5.2 Raspberry Pi

The Raspberry Pi is a small, inexpensive personal computer, Raspberry Pi zero W and Nano Pie Neo cost 10$ [14]. Although it lacks the capacity for memory expansion and can't accommodate on-board devices such as CD, DVD, and hard drives, it has everything a simple personal computer requires. As "Figure 1.4" shows, Rasberry Pi 3 has two USB ports, an Ethernet port, HDMI (and composite) video, and even an audio connector for sound [15], there are different types like Raspberry Pi 2, Raspberry Pi 3, Raspberry Pi 3 b+, Raspberry Pi 4, Raspberry Pi zero, etc.

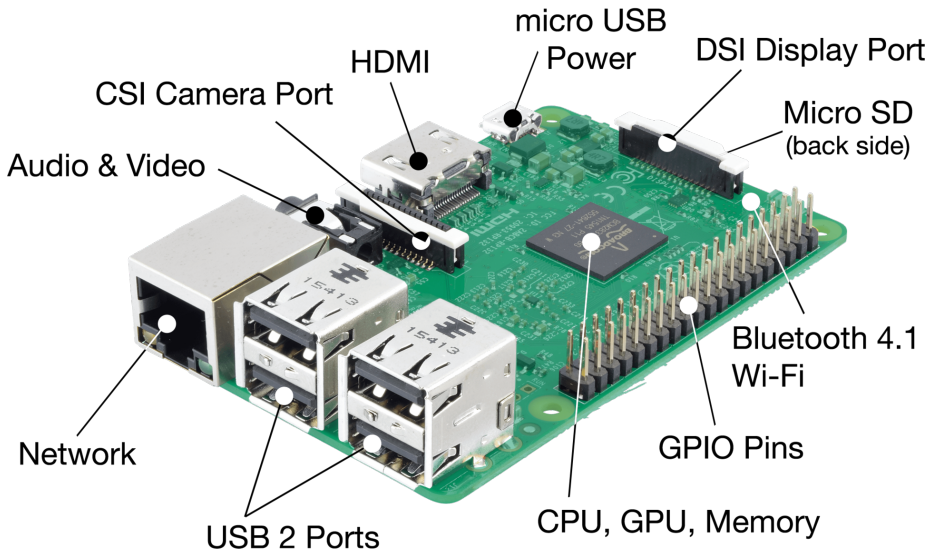


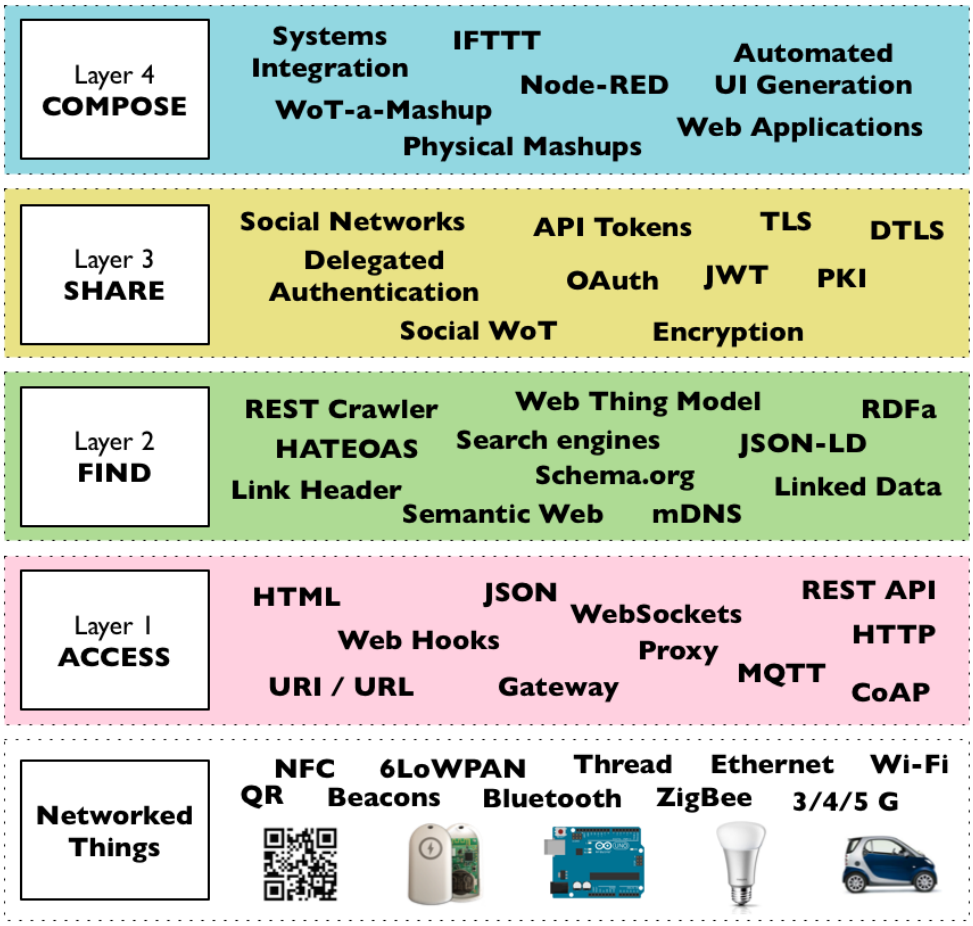
Figure 1.4: Raspberry Pi board.

1.3 Web of Things

WoT is a software architectural styles and programming patterns that allow realworld objects to be part of the World Wide Web. Web is one of the general applications of internet. Similarly an application layer is provided by WoT which simplifies the way we deal with IoT devices. It uses the existing Web in a better and faster way for the betterment of humanity [16].
In this section, we talked about WoT definition, WoT architecture and WoT technologies and protocols.

1.3.1 Web of Things architecture

Just like the OSI (Open Systems Interconnection) layered architecture organizes protocols and standards of the Internet, the WoT architecture is an attempt to structure the galaxy of Web protocols and tools into a useful framework for connecting any device or object to the Web [17], as we illustrate in “Figure 1.5” .



Source: Building the Web of Things: book.webofthings.io
Creative Commons Attribution 4.0

Figure 1.5: Web of Things architecture.

1.3.1.1 Access layer

This layer is responsible for turning any Thing into a WoT that can be interacted with using HTTP (Hypertext Transfer Protocol) requests just like any other resource on the Web. In other words, a WoT is a REST API (Representation State Transfer Application Programming Interface) that allows to interact with something in the real world, like opening a door or reading a temperature sensor located across the planet [5].

1.3.1.2 Find layer

Marking Things accessible via an HTTP and WebSocket API is a great but it doesn't mean applications can really "understand" what the CO is what data or services it offers, and so on. This is where the second layer – Find – becomes interesting. This layer ensures that your CO Web of Things architecture: can not only be easily used by other HTTP clients but can also be findable and automatically usable by other WoT applications. The approach here is to reuse Web semantic standards to describe CO and their services. This enables searching for CO through search engines and other Web indexes as well as the automatic generation of user interfaces or tools to interact with CO. At this level, technologies such as JSON-LD [8] (JavaScript Object Notation for Linked Data) are in use. This is also where standards such as the Web Things Model and the work of the W3C[9] (World Wide Web Consortium) WoT group help (They define an abstract set of REST resources that CO should offer) [5].

1.3.1.3 Share layer

The IoT will only blossom if CO have a way to securely share data across services. This is the responsibility of the Share layer, which specifies how the data generated by CO can be shared in an efficient and secure manner over the Web. At this level, another batch of Web protocols help. First, TLS, the protocol that makes transactions on the Web secure. Then, techniques such as delegated Web authentication mechanisms like OAuth[10] which can be integrated to the CO APIs [5].

1.3.1.4 Compose layer

Finally, once COs are on the Web (layer 1) where they can be found by humans and machines (layer 2) and their resources can be shared securely with others (layer 3), it's time to look at how to build large-scale, meaningful applications for the WoT. In other words, the integration of data and services from heterogeneous CO into an immense ecosystem of Web tools such as analytics

[8] JSON-LD is a language for semantically annotating JSON.

[9] W3C is the main international standards organization for the World Wide Web.

[10] Open standard for access delegation, commonly used as a way for Internet users to grant websites or applications access to their information on other websites but without giving them the passwords.

software and mashup[11] platforms is very important to understand. Web tools at the Compose layer range from Web toolkits for example, JavaScript SDKs [12] offering higherlevel abstractions to dashboards, with programmable widgets, and finally to physical mashup tools such as Node-RED6[13]. Inspired by Web 2.0 participatory services and in particular Web mashups, physical mashups offer a unified view of the classical Web and WoT and empower people to build applications using data and services from WoT without requiring programming skills [5].

1.3.2 WoT technologies and protocols

There are many technologies and protocols used in WoT, but we mention just the most commonly used ones:

1.3.2.1 WebSockets

WebSocket is a naturally full-duplex, bidirectional, single-socket connection. With WebSocket, your HTTP request becomes a single request to open a WebSocket connection,), and reuses the same connection from the client to the server, and the server to the client [18].

1.3.2.2 SSE (Server Send Event)

SSE (Server Sent Events) is a technology that enables a browser (Client) to receive automatic updates like text-based event data from a server via HTTP connection. The logic behind SSE is to create a seamless avenue for the browser to automatically receive data from the server without explicitly asking for it. This construct makes working with real-time data very efficient, as it uses just one long-lived HTTP connection [19].

1.3.2.3 HTTP

HTTP is predominantly a web messaging protocol, which was originally developed by Tim Berners-Lee. Later, it was developed by IETF[14] and W3C jointly and first published as a standard protocol in 1997. HTTP supports request/response RESTful (Representational State Transfer)

[11]Mashup is a mixture or fusion of disparate elements.

[12]The JavaScript SDK is a collection of software tools for the creation of applications and libraries that use Amazon Web Services (AWS) resources.

[13]Node-RED is a flow-based development tool for visual programming developed originally by IBM for wiring together hardware devices, APIs and online services as part of the IoTs.

[14]IETF is an open standards organization, which develops and promotes voluntary Internet standards

Web architecture [8].

1.3.2.4 RESTful API

Representational state transfer RESTful API or RESTful web services is a way of providing interoperability between computer systems on the Internet. REST-compliant Web services allow requesting systems to access and manipulate textual representations of Web resources using a uniform and predefined set of stateless operations [20].

1.3.2.5 Web Real-Time Communication

Web Real-Time communication (WebRTC) is a new standard that lets browsers communicate in real time using a peer-tp-peer architecture. it is about secure, consent-based, audio/video (and data) peer-to-peer communication between HTML5 browsers. This is a disruptive evolution in the Web applications world, since it enables, for the very first time, Web developers to build real-time multimedia applications with no need for prietary plug-ins [21].

1.4 IoT vs WoT

The ultimate goal of the WoT is maximizing existing and emerging tools and techniques used on the Web and applying them to the development of IoT scenarios. The Web of Things is concerned with only the highest OSI layer 7 the named Application layer, which handles applications, services, and data. Working with such a high level of abstraction makes it possible to connect data and services from many devices regardless of the actual transport protocols they use. In contrast, the IoT doesn't advocate a single Application-level protocol and usually focuses on the lower layers of the OSI stack which means Physical layer, Data Link layer, Network layer, Transport layer, Session layer and Presentation layer. The WoT emphasizes the use of Web standards to exchange data between devices, the WoT build a new generation of IoT solutions that are more flexible, scalable, and interoperable by building upon and using the infrastructure, tools, and experience accumulated since the Web was invented in the early 90s [6]. In short, "Table 1.1" shows a comparison between IoT and WoT:

IoT	WoT
- layer 1 to 6 in OSI model. - More complex. - Depend to devices. - Various transport protocols used.	- layer 7 in OSI model. - Easier to program. - Agnostic to the physical and transport - Use HTTP and WebRTC to interact with embedded devices. - Openly connected to the Web and publicly accessible. - layer protocols used by devices.

Table 1.1: Comparison between IoT and WoT.

1.5 Conclusion

In this first chapter, we talked about IoT and WoT, concerning IoT we adduced IoT history, nature of things, architecture, protocols and technologies used to link IoT to the Internet. concerning WoT we adduced WoT architecture, technologies and protocols, in the end we add IoT vs WoT.

Chapter II

Collaborative platforms

2.1 Introduction

Several Web-based platforms have emerged to ease the development of interactive or near real-time IoT applications by providing a way to connect things and services together and process the data they emit using a data flow paradigm. While these platforms have been found to be useful on their own.

Communication and collaboration tools play an essential role in the daily lives of people and businesses, since they make chat, video broadcasting and sharing possible.

- Collaborative platforms are independent applications, that works in conjunction with other applications (e.g. web server, email server, etc.).[22].
- Collaborative software or groupware is a new wave and is designed to boost teams productivity as it allows to share information, files and documents and helps them communicate faster in an enterprise environment [23].

2.2 Categories of collaborative platforms

Collaboration software technologies are generally categorized along two main dimensions:

2.2.1 Synchronous or real-time collaboration software

Real-time or synchronous collaboration software programs allow two or more people to view and edit documents or information even if they are from different areas. Shared whiteboards, for instance, allow people to work collaboratively on a problem on a Webbased platform. Many shared whiteboards are used for informal discussions, but these tools may also serve communications that are structured or more sophisticated, including collaborative graphic design, engineering, or publishing applications [24].

2.2.2 Asynchronous or non-real time collaboration software

Users are doing work together in one place (also known as “collocated” or “face-to-face”) or in various places (known as “non-collocated” or “distance”), like emails, mailing lists and news groups, workflow systems and group calendars [24].

2.3 Types of collaborative platform

There are many types of collaborative platforms, we mention three like "Figure 2.1" [24] shows:



Figure 2.1: Types of collaborative platforms.

2.3.1 Communication

Early collaboration software applications like Lotus Notes[1] are more focused on communications. Other platforms include instant messaging or chat [24].

2.3.2 Conferencing

Groupware[2] software tools consist of a real-time collaboration of project members who are presented with a unified view screen. In most cases, one presenter controls the view screen, while others provide their inputs by talking or chatting [24].

2.3.3 Coordination

The most recent collaborative software type and the one with the highest growth potential today, coordination software solutions help you manage complex or big interdependent tasks with a common goal [24].

2.4 Examples of collaborative Platforms

Below there is a list of tools for remote teams to collaborate and be more efficient.

2.4.1 Skype

One of the biggest reasons that Skype is the most popular communication tool is that it's free. Skype allows audio and video calls between multiple devices. In other words, one person could be on a laptop and another can be on their phone [25].

2.4.2 Mockplus

Mockplus is the most robust all-in-one product platform for prototyping, collaboration, and design systems. It not only help to create interactive prototypes for websites, mobile apps, desktop, and Web apps faster and easier, but also give a more connected product design workflow that brings designers, product managers, and front-end developers to work better and together [25].

[1]Lotus Notes is a brand of groupware that is now owned by IBM. Lotus Notes is used with a variety of local and collaborative server applications, including email, calendars, personal information managers (PIM) and the Web.

[2]Groupware refers to programs that help people work together collectively while located remotely from each other.

2.4.3 Google Docs

Google Docs is a Web-based document management application for creating and editing private and public, word processing and spreadsheet documents. These documents can be stored both online on the Google cloud and/or on the user's computer. Access to these files is available from any computer with an Internet connection and a fully-featured Web browser. The documents may be viewed by other google groups and members with the document owner's permission [25].

2.4.4 Dropbox

Dropbox is a personal cloud storage service (sometimes referred to as an online backup service) that is frequently used for file sharing and collaboration. The Dropbox application is available for Windows, Macintosh and Linux desktop operating systems. There are also apps for iPhone, iPad, Android, and BlackBerry devices [26].

2.4.5 Trello

Trello is project management software offered as a cloud service. The Trello app can be used on iOS and Android devices, and on the Web using any major modern browser. It can also be used integrated inside of a Slack[3] client. The Trello service was launched in 2011. The same year, Wired magazine called it "One of the 7 coolest startups you haven't heard of yet." By 2017, Trello had grown to over 1 million average monthly users, and was purchased by Atlassian[4] for $425 million [27].

2.4.6 ProofHub

ProofHub is a cloud – hosted project management solution that helps everyone to stay on top of deliverables and deadlines. It has scalable features and pay terms that match the requirements of any business size from small startups to large enterprise. The Walnut, CAbased company has attracted big ticket clients, such as, TripAdvisor, Harvard University, and Wipro, since the

[3]Slack is basically a messaging app on steroids. It's meant for teams and workplaces can be used across multiple devices and platforms.

[4]Atlassian Corporation is an Australian enterprise software company that develops products for software developers, project managers, and content management

software was launched in 2011. Likewise, ProofHub was among the Cloudswave Awards 2014 top ten project management software for excellent features and performance [28].

2.4.7 Collaborative platform for managing IoTs using WebRTC

That work is an application allowing several operators to control an IoT platform in collaboration using WebRTC. with this application users can cooperate with each other to carry out different tasks, collective communication in real time and control an IoT platform (announcement of the need for the object, retrieve object state, retrieve item descriptions, receive events generated by an object, perform actions and do tasks, announcement of the need for the object, retrieve object state, retrieve item descriptions, receive events generated by an object and rerform actions and do tasks.

2.4.7.1 General architecture

The general architecture of the last example (listed above) shows in “Figure 2.2” [29].

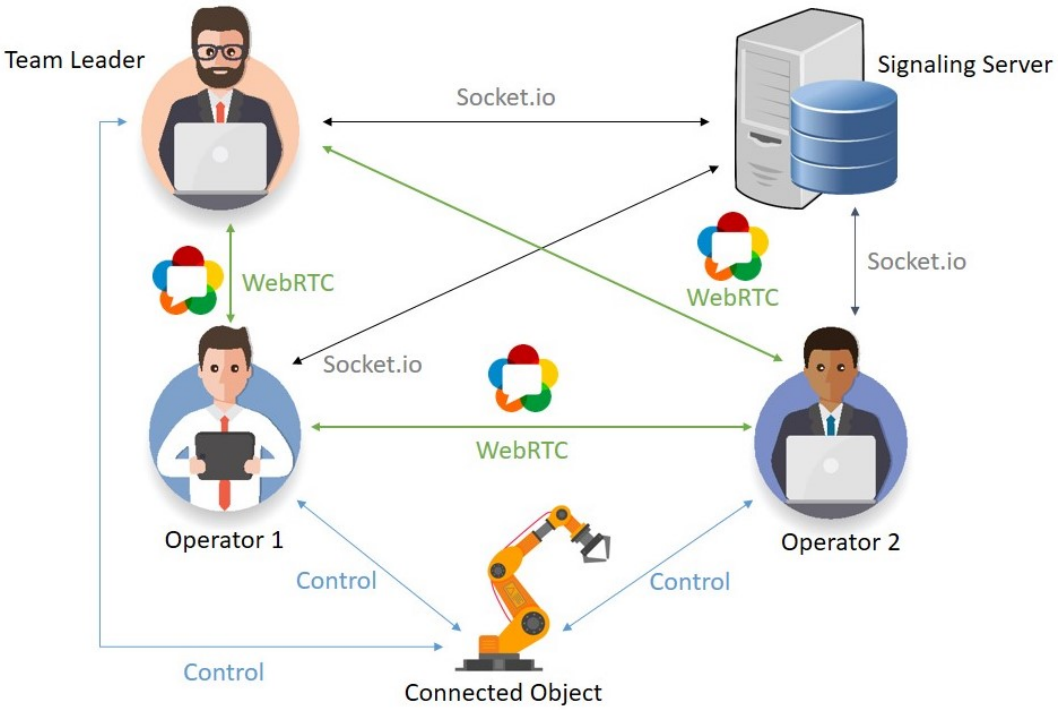


Figure 2.2: Global architecture of collaorative platforms

There is different operator, there is a leader, CO and signaling server, each operator can register or authenticate so that he can access the application, find a team and join it, list and control COs, cooperate with other team using WebRTC.

Team leader manages the functionalities of teams in the platform where he allows to create teams, update operators (add, delete or modify), update COs.

2.5 Impact of collaborative platforms in companies

Companies are under considerable strain. Several events or developments such as the global economic crises or the ongoing globalization process contribute to this situation. Meanwhile, however, companies have realized that the concept of virtual organizations represents an efficient measure to counter the current challenges.
More and more companies make use of collaboration platforms because a lot of advantages:

- Enable communication in different ways such as email, instant messaging or wiki systems between the alliance partners.
- Collaboration platforms change the way companies organize themselves internally and also impose the need to open their processes to external partners.
- People acting in collaboration platforms have the chance and are obliged to interact with many different co-workers with different backgrounds and across varying companies [30].

2.6 Conclusion

In this chapter we talked about collaborative platforms, we adduced definition of collaborative platform, categories (Synchronous or real-time and Asynchronous or non-real time), types of collaborative platforms (communication, conferencing, coordination), examples, and impact of collaborative platforms in companies.

Chapter III

Human Machine Interface

3.1 Introduction

Accidents involving machines are often attributed to human error. However, when these accidents are investigated in detail, it can often be concluded that they were in fact caused by poor ergonomic design of the interface or by problems of interaction between the machine and the user rather than by negligent or careless use. It is therefore essential that the design of interaction and of the interfaces be based on principles governing the dialogue and human physical and mental characteristics. The following chapter will illustrate these basic human characteristics and show how they may best be considered by the design of interfaces.

3.2 Definition

Human-machine interface (HMI) is a component of certain devices that are capable of handling human-machine interactions as "Figure 3.1" [31] shows below. The interface consists of hardware and software that allow user inputs to be translated as signals for machines that, in turn, provide the required result to the user. HMI technology has been used in different industries like electronics, entertainment, military, medical, etc. HMIs help in integrating humans into complex technological systems. HMI is also known as man-machine interface (MMI), computer-human interface or human-computer interface. In HMI, the interactions are basically of two types, i.e., human to machine and machine to human. Since HMI technology is ubiquitous, the interfaces involved can include motion sensors, keyboards and similar peripheral devices, speech recognition interfaces and any other interaction in which information is exchanged using sight, sound, heat and other cognitive and physical modes are considered to be part of HMIs [32].

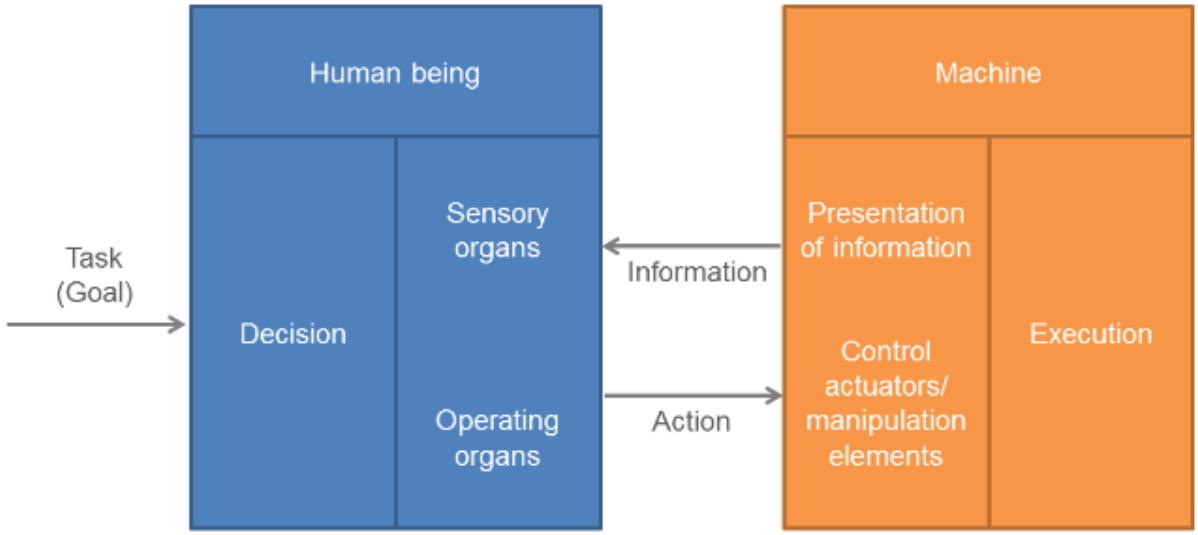


Figure 3.1: Human-machine system shown as a closed control loop.

3.3 List of Ergonomic Criteria

The set of criteria consists of eight main criteria of "Dominique L. Scapin and J. M. Christian Bastien", some of which are divided into sub-criteria ,each sub-criteria contains some examples.

3.3.1 Guidance

User guidance refers to the means available to advise, orient, inform, instruct, and guide the users throughout their interactions with a computer (messages, alarms, labels, etc.), including lexical issues. The criterion guidance is subdivided into four criteria: prompting, grouping/distinction of items, immediate feedback and legibility.

3.3.1.1 Prompting

As it is used here, Prompting has a broader definition than usual. Here it refers to the means available in order to lead the users to making specific actions whether it be data entry or other tasks. This criterion also refers to all the means that help users to know the alternatives when several actions are possible depending on the contexts. Prompting also concerns status information, that is information about the actual stateor context of the system, as well as information concerning help facilities and their accessibility.

Examples:

- For each data field, display an associated label.
- Provide a title for each window.

3.3.1.2 Grouping/Distinction of Items

The criterion Grouping/Distinction of Items concerns the visual organization of information items in relation to one another. This criterion takes into account the topology (Location) and some graphical characteristics (Format) in order to indicate the relationships between the various items displayed, to indicate whether or not they belong to a given class, or else to indicate differences between classes. This criterion also concerns the organization of items within a class. The criterion grouping/distinction of Items is subdivided into two criteria: grouping/distinction by location and grouping/distinction by format.

Examples:

- Grouping/Distinction by Location:
  - Organize items in hierarchical lists.
  - Organize the options of a menu dialogue as a function of the objects to which they apply.
- Grouping/Distinction by Format:
  - Provide clear visual distinction of areas having different functions (command zone, message zone, etc.).
  - Provide clear visual distinction of data fields and their labels.

3.3.1.3 Immediate Feedback

Immediate Feedback concerns system responses to users' actions. These actions may be simple keyed entries or more complex transactions such as stacked commands. In all cases computer responses must be provided, they should be fast, with appropriate and consistent timing for different types of transactions. In all cases, a fast response from the computer should be provided with information on the requested transaction and its result.

Examples:

- All users' entries should be displayed except for secure entries. Even in this case, every keyed entry should produce a perceptible feedback (e.g., symbols such as stars ' *** ').
- Following user interruption of data processing, display an advisory message assuring the user that the system has returned to its previous status.

3.3.1.4 Legibility

Legibility concerns the lexical characteristics of the information presented on the screen that may hamper or facilitate the reading of this information (character brightness, contrast between the letter and the background, font size, interword spacing, line spacing, paragraphs spacing, line length, etc.). By definition, the criterion Legibility does not concern feedback or error messages.

Examples:

- Titles should be centered.
- Labels should be displayed in upper case letters.
- Cursors should be distinguished from other displayed items.

3.3.2 Workload

The criterion workload concerns all interface elements that play a role in reducing the users' perceptual or cognitive load, and in increasing the dialogue efficiency. The criterion workload is subdivided into two criteria: Brevity (which includes conciseness and minimal actions), and information density.

3.3.2.1 Brevity

The criterion Brevity concerns the perceptual and cognitive workload both for individual inputs and outputs, and for sets of inputs (i.e., sets of actions needed to accomplish a goal or a task). Brevity corresponds to the goal of limiting the reading and input workload and the number of action steps. The criterion Brevity is subdivided into two criteria: Concision and Minimal Actions.

3.3.2.1.1 Concision

The criterion Concision concerns perceptual and cognitive workload for individual inputs or outputs. By definition, Concision does not concern feedback or error messages.

Examples:

- For numerical data, entry of leading zeros should not be necessary.
- Allow users short data entries.

3.3.2.1.2 Minimal Actions

The criterion Minimal Actions concern workload with respect to the number of actions necessary to accomplish a goal or a task. It is here a matter of limiting as much as possible the steps users must go through.

Examples:

- Minimize the number of steps required to make a selection in a menu.
- Do not require data entry by the user when the computer can handle it.

3.3.2.2 Information Density

The criterion Information Density concerns the users' workload from a perceptual and cognitive point of view with regard to the whole set of information presented to the users rather than each individual element or item.

Examples:

- Provide only necessary and immediately usable data for any transaction; do not overload displays with extraneous data.
- Data should not require unit translations.

3.3.3 Explicit Control

The criterion explicit control concerns both the system processing of explicit user actions, and the control users have on the processing of their actions by the system. The criterion explicit control is subdivided into two criteria: explicit user action, and user control.

3.3.3.1 Explicit User Action

The criterion Explicit User Action refers to the relationship between the computer processing and the actions of the users. This relationship must be explicit, i.e., the computer must process only those actions requested by the users and only when requested to do so.

Examples:

- Always require a user to take an explicit ENTER action to initiate processing of entered data; do not initiate processing as a side effect (e.g., updating a file) of some other action (e.g., printing a file).
- If menu selection is accomplished by pointing, provide for dual activation, in which the first action (positions a cursor) designates the selected option, followed by a separate second action that makes an explicit control entry.
- Users' command entries should be completed with an ENTER action following editing facilities.

3.3.3.2 User Control

The criterion Explicit User Action refers to the relationship between the computer processing and the actions of the users. This relationship must be explicit, i.e., the computer must process only those actions requested by the users and only when requested to do so.

Examples:

- Always require a user to take an explicit ENTER action to initiate processing of entered data; do not initiate processing as a side effect (e.g., updating a file) of some other action (e.g., printing a file).

- If menu selection is accomplished by pointing, provide for dual activation, in which the first action (positions a cursor) designates the selected option, followed by a separate second action that makes an explicit control entry.
- Users' command entries should be completed with an ENTER action following editing facilities.

3.3.4 Adaptability

The adaptability of a system refers to its capacity to behave contextually and according to the users' needs and preferences. The criterion adaptability is subdivided into two criteria: Flexibility and user experience.

3.3.4.1 Flexibility

The criterion flexibility refers to the means available to the users to customize the interface in order to take into account their working strategies and/or their habits, and the task requirements. Flexibility is reflected in the number of possible ways of achieving a given goal.

Examples:

- When user requirements are uncertain, provide users with some means to control display configuration.
- When interface designers cannot predict which default values will be helpful, permit users to define, change or remove default values for data entry.
- When some displays are unnecessary, the users should be able to remove them temporarily.

3.3.4.2 User Experience

The criterion user experience refers to the means available to take into account the level of user experience.

Examples:

- Allow experienced users to by-pass a series of menu selections and make an equivalent command entry or keyboard shortcuts directly.

- Allow experienced users to key in a series of commands at one time, and inexperienced users only step by step keying.
- Dialogue types must be designed to match the needs of different users.

3.3.5 Error Management

The criterion error management refers to the means available to prevent or reduce errors and to recover from them when they occur. Errors in this context include invalid data entry, invalid format for data entry, incorrect command syntax. The criterion error management is subdivided into three criteria: error protection, quality of error messages, and error correction.

3.3.5.1 Error Protection

The criterion Error Protection refers to the means available to detect and prevent data entry errors, command errors, or actions with destructive consequences.

Examples:

- When a user requests LOG-OFF and if any pending transaction will not be completed, or if data will be lost, display an advisory message requesting user confirmation.
- Protect field labels from accidental change by users.
- Fields designed for information display should be protected: users should not be allowed to change the information contained in these fields.

3.3.5.2 Quality of Error Messages

The criterion Quality of Error Messages refers to the phrasing and the content of error messages that is their relevance, readability, and specificity about the nature of the errors (syntax, format, etc.) and the actions needed to correct them.The criterion Error Protection refers to the means available to detect and prevent data entry errors, command errors, or actions with destructive consequences.

Examples:

- If the user selects an invalid function key, no system action should result except a message indicating the functions appropriate for that transaction step.

- Make the error messages as specific as possible.
- Make error messages brief but informative.

3.3.5.3 Error Correction

The criterion Error Correction refers to the means available to the users to correct their errors.

Examples:

- Users should be allowed to edit an extended command during its composition before taking an explicit action to ENTER the command.
- Following error detection, require the user to re-enter only that portion of a data/command entry which is not correct.
- If a data entry transaction has been completed and errors detected, permit users to make corrections directly and immediately.

3.3.6 Consistency

The criterion consistency refers to the way interface design choices (codes, naming, formats, procedures, etc.) are maintained in similar contexts, and are different when applied to different contexts.

Examples:

- Window titles should always be located in the same place.
- Use similar screen formats.
- Use similar procedures to access menu options.
- Consistent phrasing and punctuation should be used in all prompts.
- Data entry fields should always be the same.

3.3.7 Significance of codes

The criterion significance of codes qualifies the relationship between a term and/or a sign and its reference. Codes and names are significant to the users when there is a strong semantic relationship between such codes and the items or actions they refer to.

Examples:

- Titles should be distinct and meaningful.
- Make abbreviation rules explicit.
- Codes should be meaningful and familiar rather than arbitrary (e.g., “M” for Male, and “F” for Female rather than “1” and “2”).

3.3.8 Compatibility

The criterion compatibility refers to the match between users’ characteristics (memory, perceptions, customs, skills, age, expectations, etc.) and task characteristics on the one hand, and the organization of the output, input, and dialogue for a given application, on the other hand.

Examples:

- When data entry involves transcription from source documents, ensure that form filling displays match those documents.
- Dialogues should reflect data structures or organizations which are perceived by users as being natural.
- Calendar formats should follow users’ customs (American vs. European calendar).
- Labels, prompts, and user guidance messages should be familiar to the users and task oriented.
- Units of measurement should be familiar to the user.
- Displays of textual data, messages, or instructions, should follow design conventions for printed text [33].

3.4 Conclusion

Interfacing with a machine is not only mastering the physical interface between the human user and the system but it is also mastering the mental model implemented in the machine's architecture and logic.

In this chapter we talked about HMI, we adduced definition of HMI, and a list of ergonomic criteria.

Chapter IV

Design

4.1 Introduction

An outline of the areas in which work must be undertaken during the design stage of a project is presented. It is vital that the project manager should ensure sound decisions are made at the design stage, as these are likely to determine the success or failure of the project. Design selection, building up a cohesive project team, project strategy and use of past experience are analysed as necessary ingredients in the favourable outcome of any project.

4.2 Presentation of our platform

Our platform is an application which enables different users to control different COs collaboratively on the Web. Our platform contains two types of consumers:
Admin and users, Admin can create and distribute users accounts, manage users list (add, delete, update) and COs list (add, delete, update), and communicate with users. Users can invoke actions, read description of COs, view status and communicate with users and admin.

4.3 Platform architecture

"Figure 4.1" shows the general architecture of our platform:

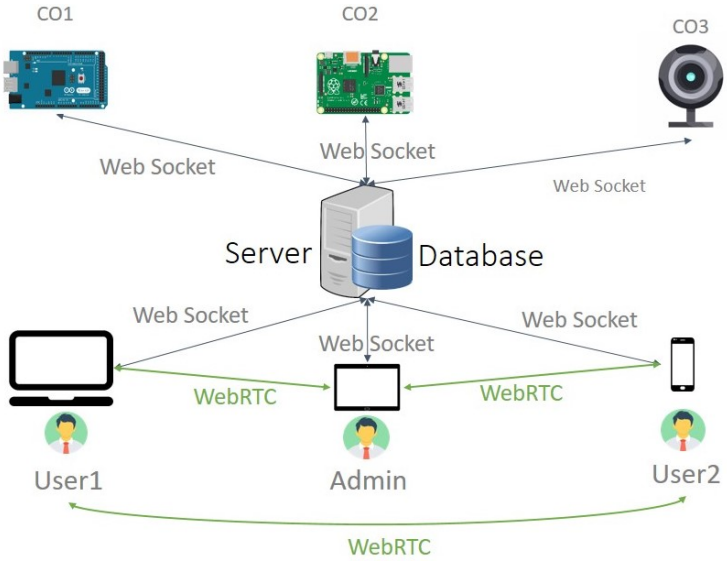


Figure 4.1: Platform architecure.

4.4 Development methodology

Successful projects should be managed well. To manage a project efficiently, the manager or development team must choose the software development methodology that will work best for the project at hand, because of that we choose Unified Process(UP).

4.4.1 Definition of Unified Process

the UP is a software development process. A software development process is the set of activities needed to transform a user's requirements into a software system as "Figure 4.2".

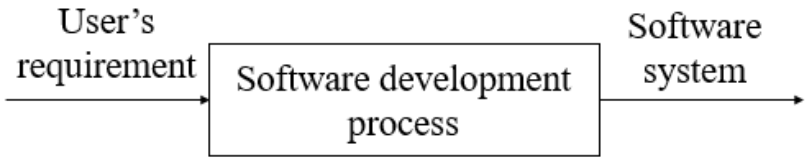


Figure 4.2: A software development process.

However, the UP is more than a single process; it is a generic process framework that can be specialized for a very large class of software systems, for different application areas, different types of organizations, different competence levels, and different project sizes.
The UP is an architecture-centric, use-case driven, iterative and incremental development process. The UP is component-based, which means that the software system being built is made up of software components interconnected via well-defined interfaces [34].

4.4.2 Unified Process phases

The UP divides the project into four phases, as "Figure 4.3" [35] illustrates:

- Inception Phase.
- Elaboration Phase.
- Construction Phase.
- Transition Phase.

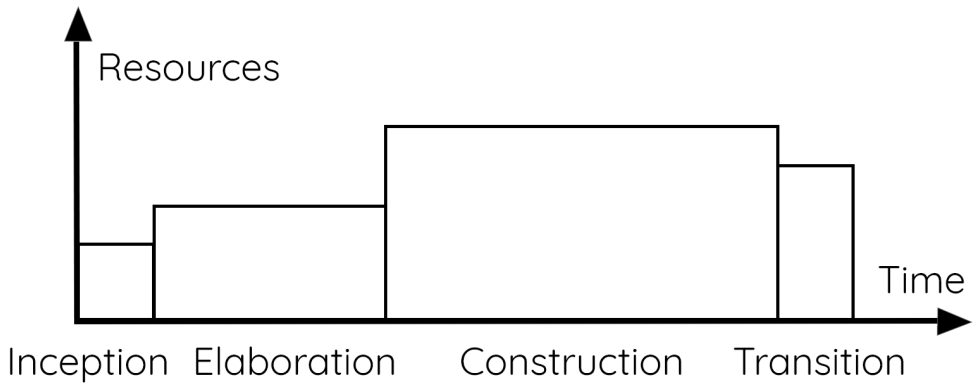


Figure 4.3: Unified Process phases

4.4.2.1 Inception Phase

Inception is the smallest phase in the project, and ideally it should be quite short. If the Inception Phase is long then it may be an indication of excessive up-front specification, which is contrary to the spirit of the UP. The following are typical goals for the Inception phase:

- Establish a justification or business case for the project.
- Establish the project scope and boundary conditions.
- Outline the use cases and key requirements that will drive the design tradeoffs.
- Outline one or more candidate architectures.
- Identify risks.

4.4.2.2 Elaboration Phase

During the Elaboration phase the project team is expected to capture a healthy majority of the system requirements. However, the primary goals of Elaboration phase are to address known risk factors and to establish and validate the system architecture. Common processes undertaken in this phase include the creation of use case diagrams, conceptual diagrams (class diagrams with only basic notation) and package diagrams (architectural diagrams). The architecture is validated primarily through the implementation of an Executable Architecture Baseline. This is a partial implementation of the system which includes the core, most architecturally significant, components. It is built in a series of small, timeboxed iterations. By the end of the Elaboration

phase the system architecture must have stabilized and the executable architecture. baseline must demonstrate that the architecture will support the key system functionality and exhibit the right behavior in terms of performance, scalability and cost. The final Elaboration phase deliverable is a plan (including cost and schedule estimates) for the Construction phase. At this point the plan should be accurate and credible, since it should be based on the Elaboration phase experience and since significant risk factors should have been addressed during the Elaboration phase. The Lifecycle Architecture Milestone[1] marks the end of the Elaboration phase.

4.4.2.3 Construction Phase

Construction is the largest phase in the project. In this phase the remainder of the system is built on the foundation laid in Elaboration. System features are implemented in a series of short, timeboxed iterations. Each iteration results in an executable release of the software. It is customary to write full text use cases during the construction phase and each one becomes the start of a new iteration. Common Unified Modeling Language (UML) diagrams used during this phase include Activity, Sequence, Collaboration State (Transition) and Interaction Overview diagrams. The Initial Operational Capability Milestone marks the end of the Construction phase.

4.4.2.4 Transition Phase

The final project phase is Transition phase. In this phase the system is deployed to the target users. Feedback received from an initial release (or initial releases) may result in further refinements to be incorporated over the course of several Transition phase iterations. The Transition phase also includes system conversions and user training. The Product Release Milestone[2] marks the end of the Transition phase [35].

[1] At this point, developer examine the detailed system objectives and scope, the choice of architecture, and the resolution of the major risks.

[2] At this point, the developer decide if the objectives were met, and if he should start another development cycle. In some cases this milestone may coincide with the end of the inception phase for the next cycle.

4.4.3 Unified Process activities

Each phase (lisetd above) consists of a set of predefined activities, as "Figure 4.4"[36] shows:

- Business modeling.

- Requirements.

- Analysis and design.

- Implementation.

- Test.

- Deployment.

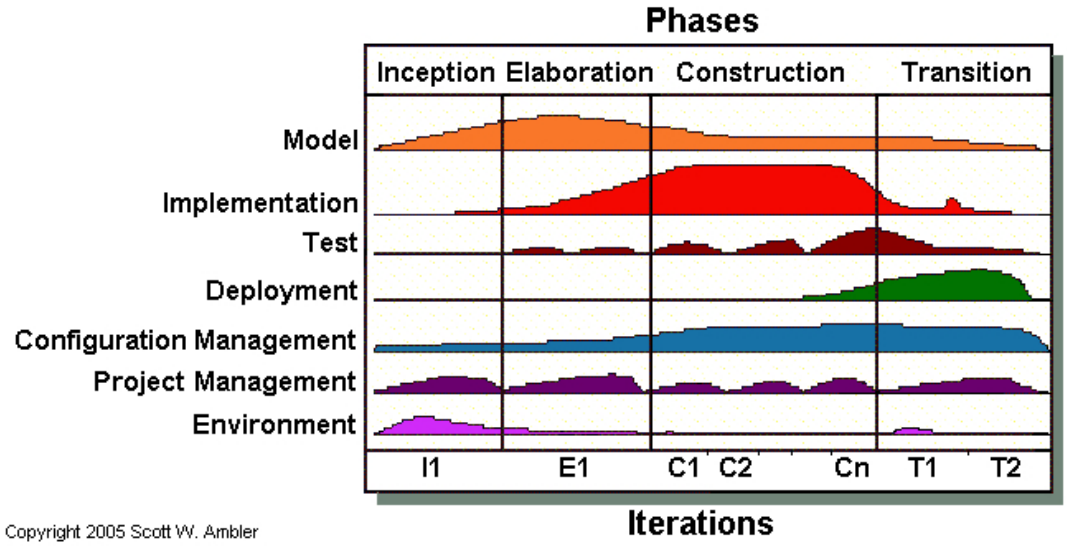


Figure 4.4: Unified process lifecycle

4.4.3.1 Business modeling

Business modeling is the domain object modeling and dynamic modeling of the business processes.

4.4.3.2 Requirements

Requirements are the requirements analysis of system under consideration. Includes activities like writing use cases and identifying nonfunctional requirements.

4.4.3.3 Analysis and design

Analysis and design covers aspects of design, including the overall architecture.

4.4.3.4 Implementation

Implementation is the programming and building of the system (except the deployment).

4.4.3.5 Test

Test involves testing activities such as test planning, development of test scenarios.

4.4.3.6 Deployment

Deployment contain the deployment activities of developed system [37].

4.5 Modeling Language

Knowing that the UP requires the use of UML, our modeling will be done using UML diagrams.

4.5.1 UML

UML is a visual modeling language for modeling system requirements, describing designs, and depicting implementation details.

UML emerged as the software industry's dominant modeling language. UML is not only the de facto modeling language standard for specifying, visualizing, constructing, and documenting the components of software systems, but it has also been accepted by the Object Management Group[3] (OMG) as a standard language for object-oriented analysis and design [38].

4.5.2 UML diagrams

A UML diagram is a diagram with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system [39].

UML specification defines two major kinds of UML diagram, structure diagrams and behavior diagrams.

Structure diagrams show the static structure of the system and its parts on different abstraction and implementation levels and how they are related to each other. The elements in a structure diagram represent the meaningful concepts of a system, and may include abstract, real world and

[3]The Object Management Group (OMG) is a consortium geared toward creating a standard architecture for distributed network objects

implementation concepts.

Behavior diagrams show the dynamic behavior of the objects in a system, which can be described as a series of changes to the system over time [40]. UML vesrion 2.5 diagrams could be categorized hierarchically as shown in "Figure 4.1" [40].

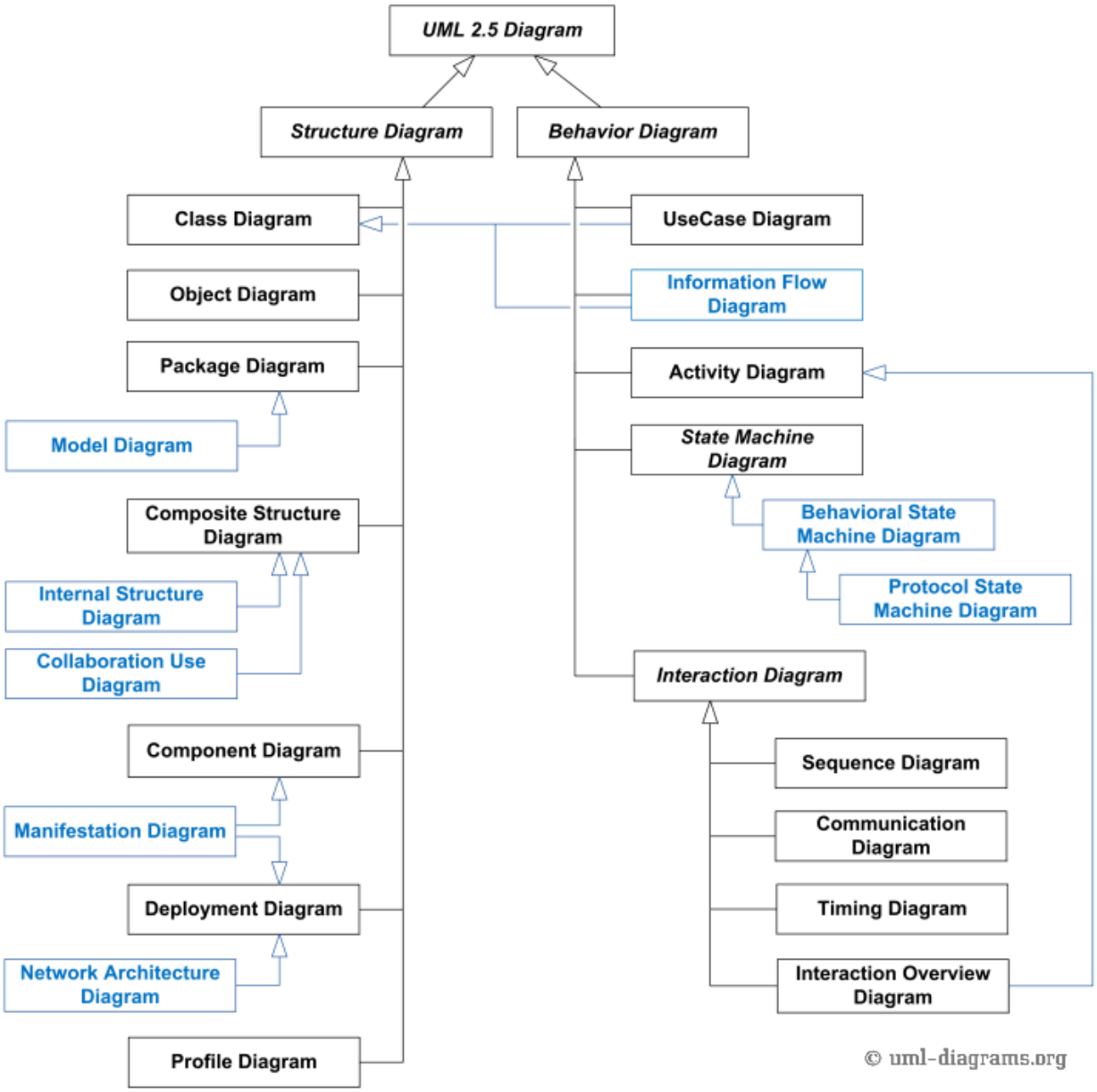


Figure 4.5: The hierarchy of UML 2.5 diagrams

4.5.3 Used UML diagrams

In our work we use three diagrams, which is detailed in "Table4.1" [40]:

- Use case diagram.

- Sequence diagram.

- Class diagram.

Diagram	Purpose	Type
Use case diagram	Describes a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors) to provide some observable and valuable results to the actors or other stakeholders of the system(s).	Behavior diagram
Sequence diagram	Most common kind of interaction diagrams which focuses on the message interchange between lifelines (objects).	Behavior diagram
Class diagram	Shows structure of the designed system, subsystem or component as related classes and interfaces, with their features, constraints and relationships associations, generalizations, dependencies, etc.	Structure diagram

Table 4.1: Used UML diagrams

4.6 Use case diagram

A use case diagram is generally straightforward. It only summarizes some of the relationships between use cases, actors, and systems. In our systems the actors are:

Admin: His role is to manage of users (add, delete, update, give responsibility to users), manage of COs (add, delete, update) and to communicate with users in collaboration.

User: His role is to control his COs and to collaborate with other users.

CO: Performs actions invoked by users, shows his status and notify users.

4.6.1 Identification of the use cases

Actor	Use case
Admin/Users	1. Authenticate
Admin	1. Discovers of COs. 2. Invokes action to COs. 3. Communicates with users. 4. Manages Users. 5. Manages COs.
Users	1. Discover of COs. 2. Invoke actions 3. Communicate with users.
COs	1. Performs of actions invoke by users. 2. Show status. 3. Notify users.

Table 4.2: Identification of use cases.

"Figure 4.6" illustrates the use case diagram that groups the functionalities of our platform.

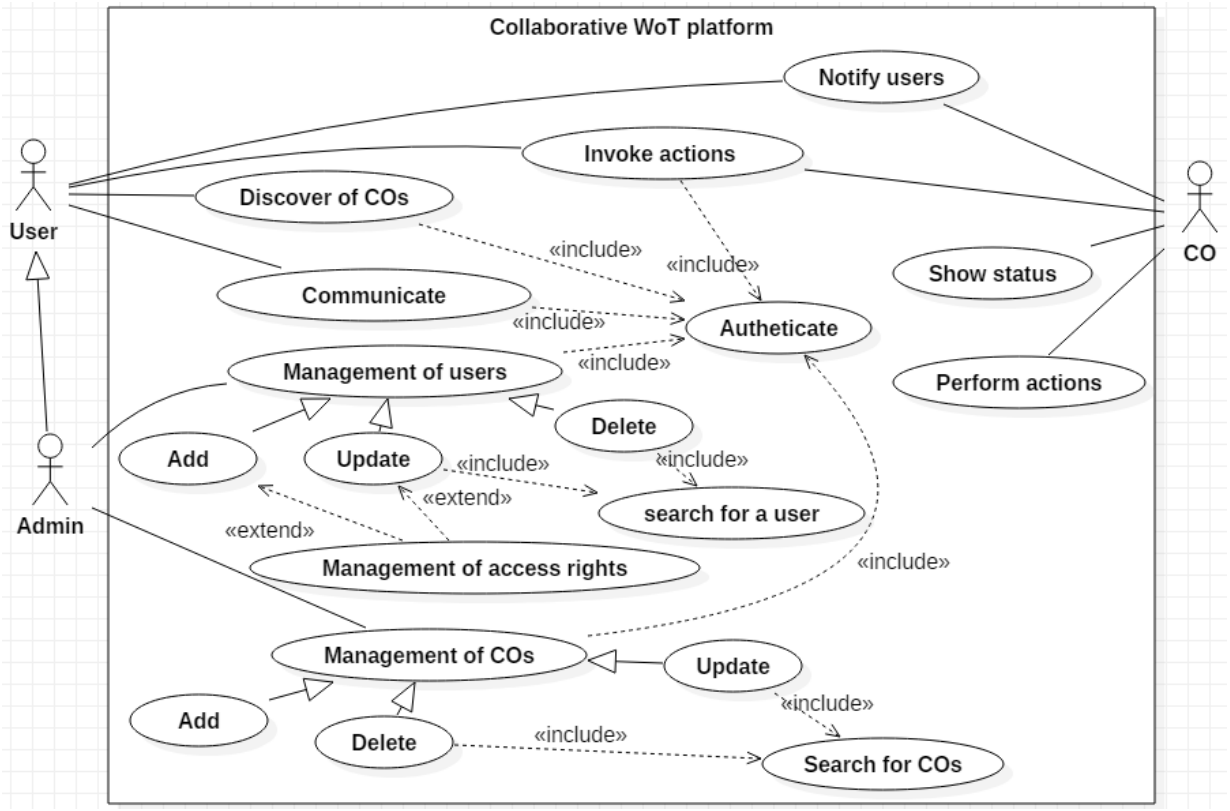


Figure 4.6: Use case Diagram

4.6.2 Textual description of use cases

In this section, we will mention for each use case a table that contains a textual description of it:

"Table4.3" for "Authenticate" use case.

"Table4.4" for "Management of users" use case.

"Table4.5" for "Management of COs" use case.

"Table4.6" for "Discover of COs" use case.

"Table4.7" for "Invoke actions" use case.

"Table4.8" for "Notify users" use case.

"Table4.9" for "Communicate" use case.

4.6.2.1 Use case N°1: Authenticate

Actor	Admin/user
Objective	Admin/user authenticate in our platform
Precondition	Admin/user must be registered in our platform.
Scenario	1. Admin/user enter the link 2. As soon as Admin/user launches the application the authentication window is displayed. 3. He fills in the authentication form (user name, password). 4. He clicks on the button (connect) to send an authentication request to the application server. 5. The application server check if the Admin/user exist in the Database. 6. Display of the main window.
Alternative	If Admin/user contact details are not valid in Database, in this case they cannot access the session and an error message will be displayed (return to 3)

Table 4.3: Description of the use case "Authenticate".

4.6.2.2 Use case N°2: Management of users

Actor	Admin
Objective	Manages of users
Precondition	Admin authenticated in the platform.
Scenario	1. Access the users management window. 2. Admin performs one of the update operations (add, delete, modify) of users information. 3. Clicks on one of the buttons (add, delete, modify) to send a request to the server. 4. The request is verified by the application server and the Database which sends a success message.
Alternative	If the response to the request sent by the server is unfavorable in this case an error message is displayed.

Table 4.4: Description of the use case "Management of users".

4.6.2.3 Use case N°3: Management of COs

Actor	Admin
Objective	Manages of COs
Precondition	Admin authenticate in the application.
Scenario	1. Access the COs management window. 2. Admin performs one of the update operations (add, delete, modify). 3. Click on one of the buttons (add, delete, modify) to send a request to the server for processing. 4. The request is verified by the application server and the Database which sends a success message.
Alternative	If the response to the request sent by the server is unfavorable in this case an error message is displayed.

Table 4.5: Description of the use case "Management of COs".

4.6.2.4 Use case N°4: Discover of COs

Actor	Admin/user
Objective	Discover of COs
Precondition	Admin/User authenticated in the platform
Scenario	1. Access the COs management window. 2. Search COs. 3. The request is verified by the Database which sends results to the application server which sends the COs list to the Web application which sends it to the users. 4. Read COs description
Alternative	COs not found

Table 4.6: Description of the use case "Discover of COs".

4.6.2.5 Use case N°5: Invoke action

Actor	Admin/User
Objective	Invokes action
Precondition	1. Admin/User authenticate in the application 2. Admin/User discover of the COs
Scenario	1. Choose CO. 2. Invoke action. 3. Sends action parameter(action name, arguments of action) . 4. Treats the request by the CO. 5. Sends notification to the users.
Alternative	COs is busy, wait a while

Table 4.7: Description of the use case "Invoke actions".

4.6.2.6 Use case N°6: Notify users

Actor	Admin/User
Objective	Notifies users
Precondition	Admin/User authenticate in the application
Scenario	1. CO changes his state. 2. The new state value sends to the application server. 3. Updates information in the Database. 4. Sends a notification to the Web application. 5. Displays a notification to the users.
Alternative	COs is busy, wait a while

Table 4.8: Description of the use case "Notify users".

4.6.2.7 Use case N°7: Communicate

Actor	Admin/User
Objective	Communicate and collaborate between colleagues
Precondition	Admin/User authenticate in the application
Scenario	1. Admin/user selects chat window. 2. Admin/user selects colleague. 3. Admin/user choose to send message or to start audio/video call. 4. Sends messages or audio/video call to colleague. 5. Displays messages or audio/video call to colleague..

Table 4.9: Description of the use case "Communicate".

4.7 Sequence diagram

Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they shows the order of the interaction visually by using the vertical axis of the

diagram to represent time what messages are sent and when [41].

There are different sequence diagrams:

4.7.1 Sequence diagram of the use case "Authenticate"

When Admin/user wants to access our application, he will be required to authenticate. As soon as Admin/user launches the application the authentication window is displayed, Admin/user fills in the authentication form (user name, password) and validates the data that will be sent to the application server. The application server checks if the user exist in the Database, the Web application will be informed, and the main window will be displayed otherwise an error message will be displayed as "Figure 4.7" shows.

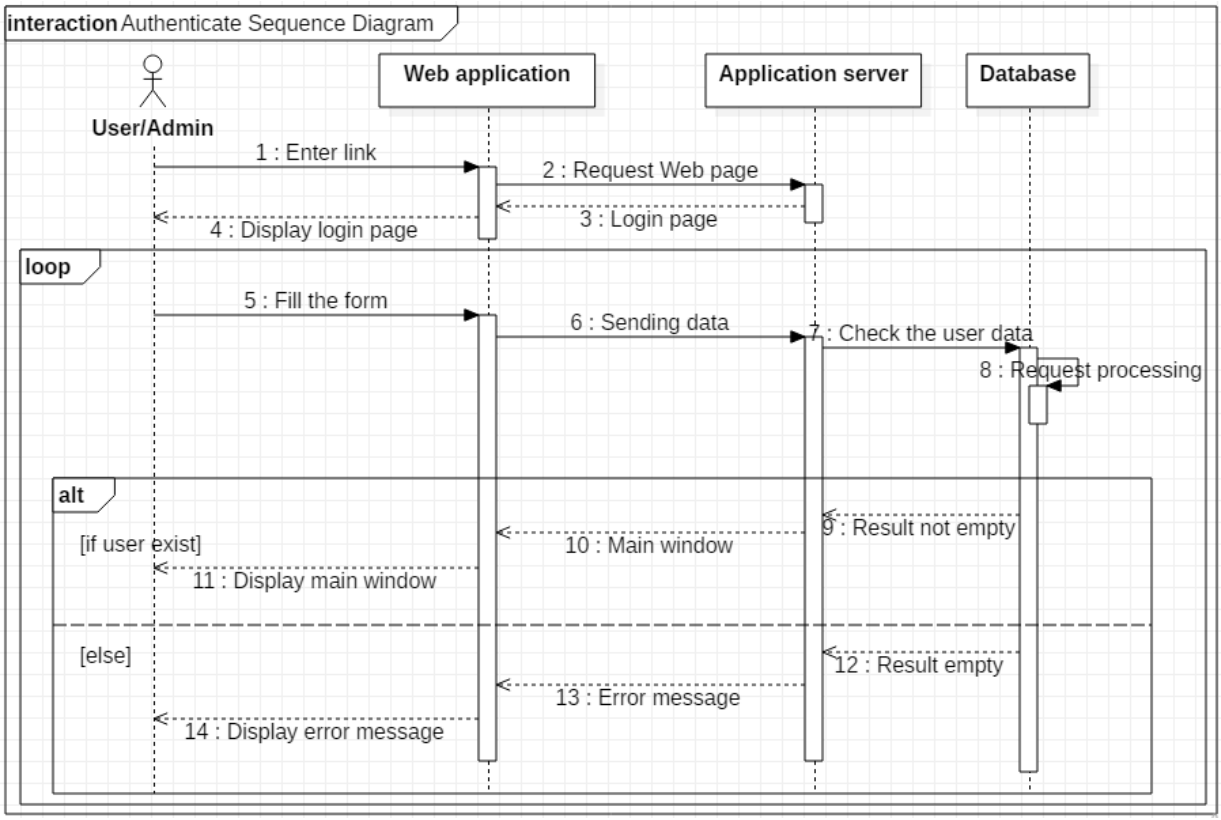


Figure 4.7: Sequence diagram of the use case "Authenticate".

4.7.2 Sequence diagram of the use case "Management of users"

When the admin wants to manage users, he must first authenticate himself, then access the user management window, then he consults the list of users and updates (add, delete, modify) a user and sends the data to the application server, the application server update the information in the Database, if the request is valid, the Web application will be informed and a validation message will be displayed otherwise an error message will be displayed as "Figure 4.8" shows.

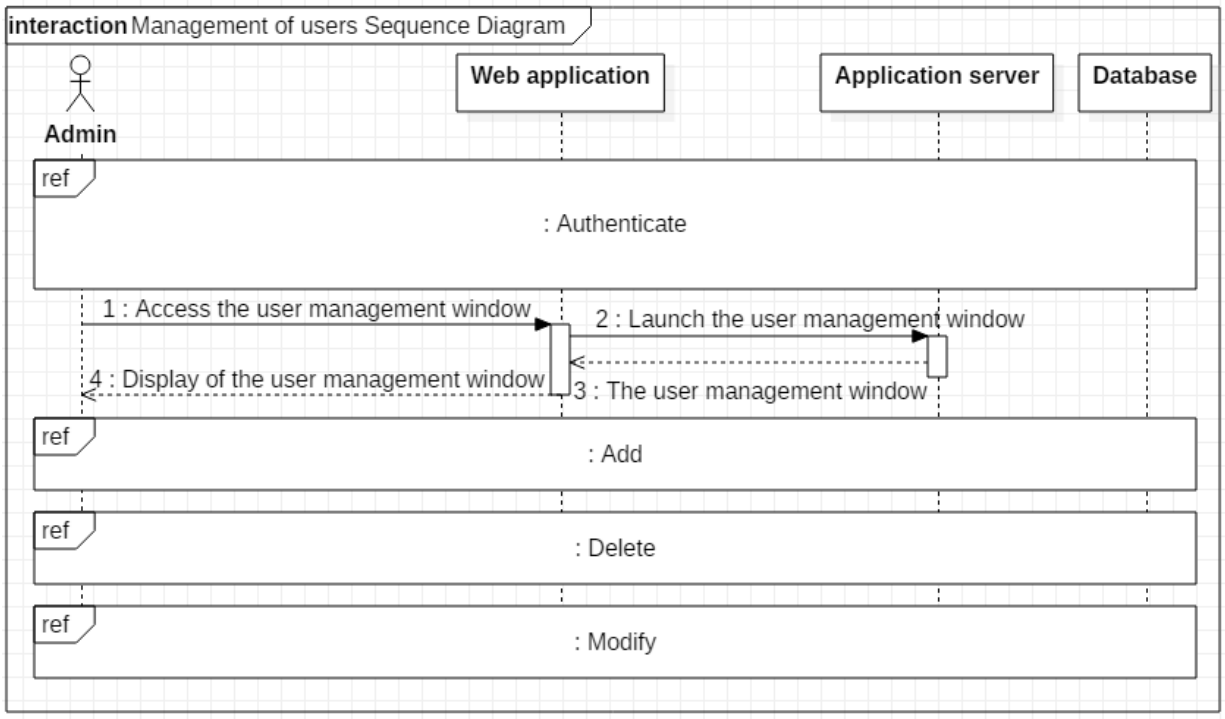


Figure 4.8: Sequence diagram of the use case "Management of users".

"Figure 4.9" shows add operation:

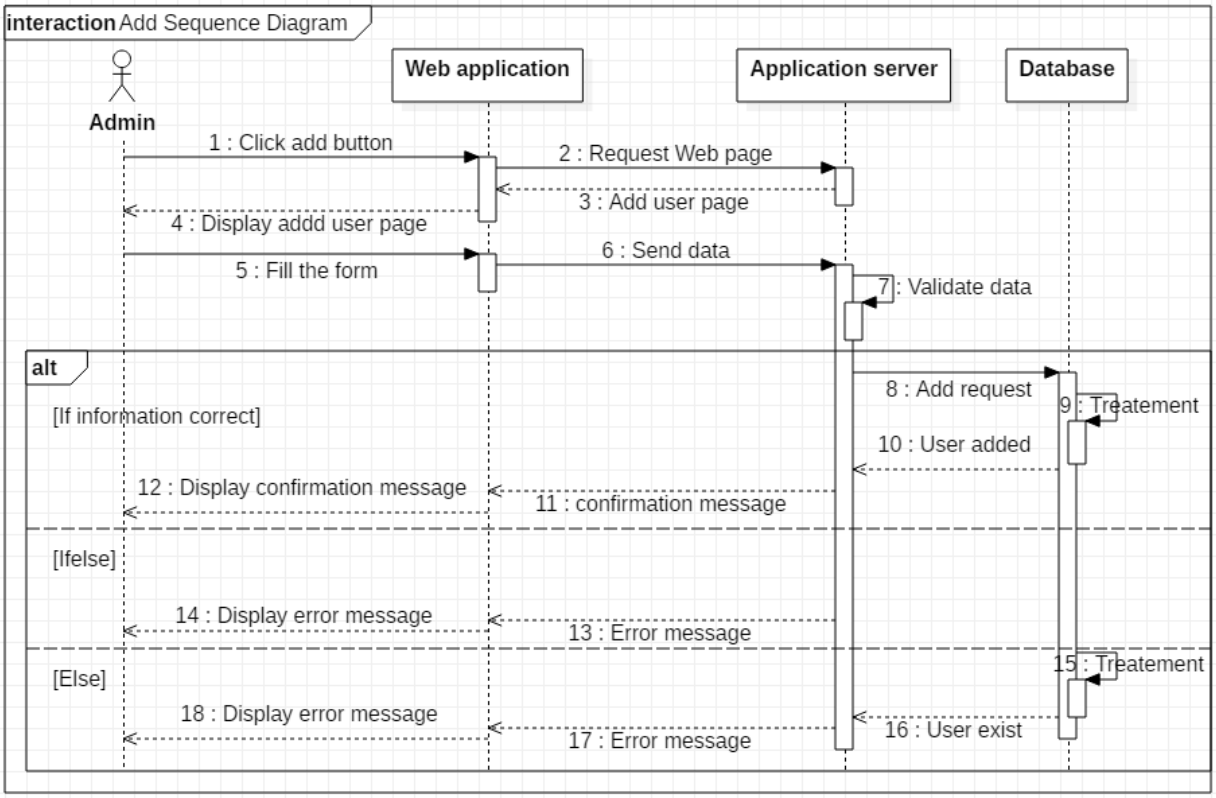


Figure 4.9: Sequence diagram of the operation add in the use case "Management of users".

"Figure 4.10" shows delete operation:

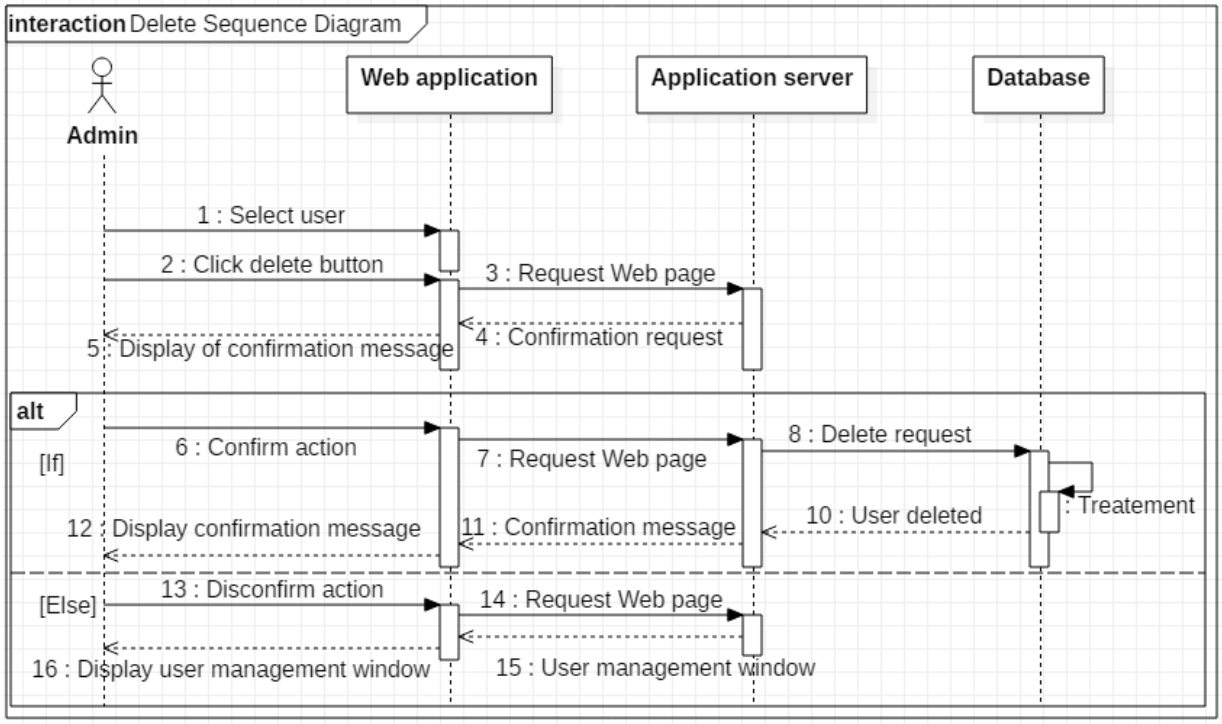


Figure 4.10: Sequence diagram of the operation delete in the use case "Management of users".

"Figure 4.11" shows modify operation:

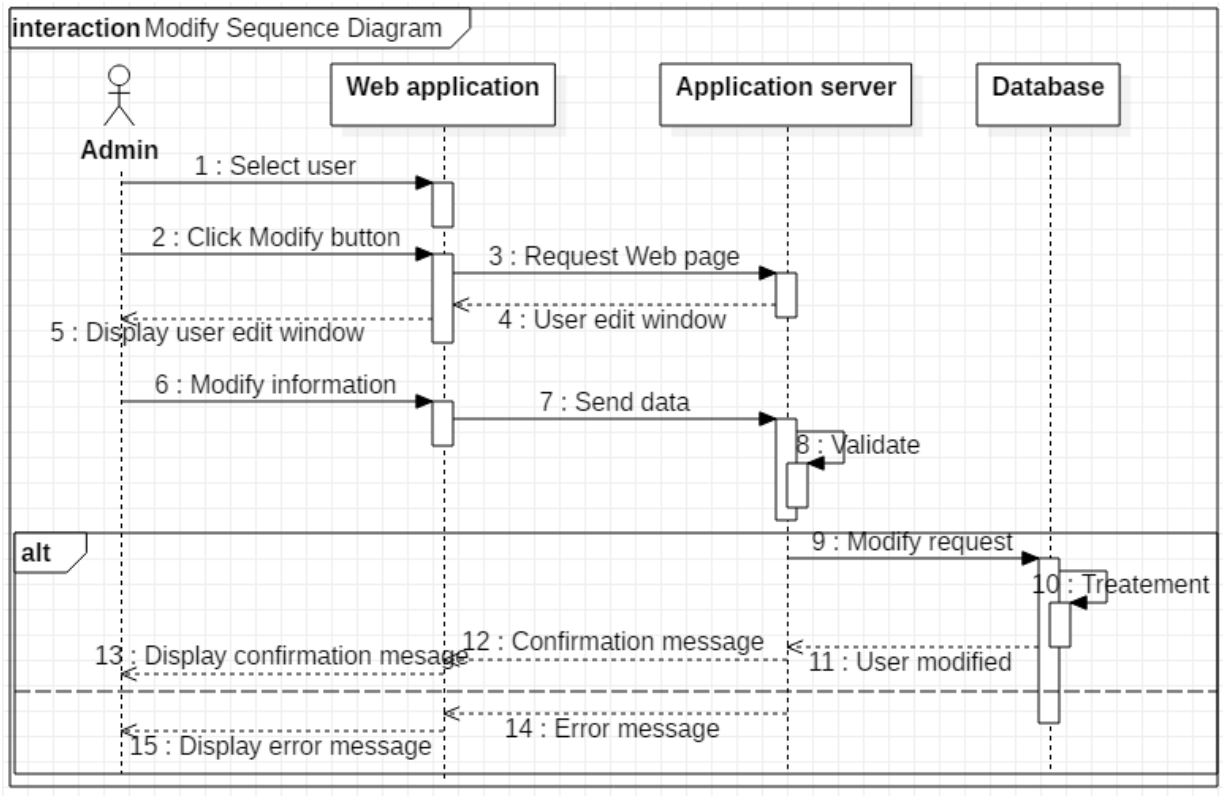


Figure 4.11: Sequence diagram of the operation modify in the use case "Management of users".

4.7.3 Sequence diagram of the use case "Management of COs"

When the admin wants to manage COs, he must first authenticate himself, then access the COs management window, then he consults the list of COs and updates (add, delete, modify) a CO and sends the data in to the application server, the application server updates the information in the Database, if the request is valid, the Web application will be informed and a validation message will be displayed otherwise an error message will be displayed as "Figure 4.12" shows.

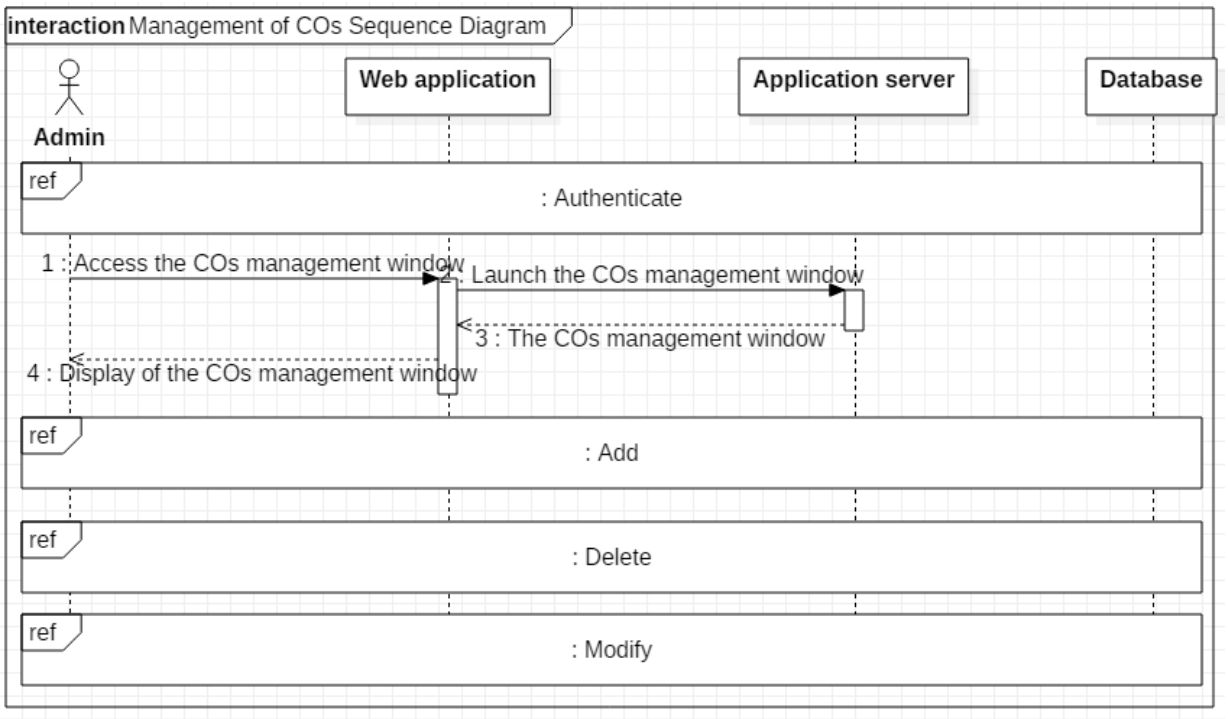


Figure 4.12: Sequence diagram of the use case "Management of COs".

"Figure 4.13 " shows add operation:

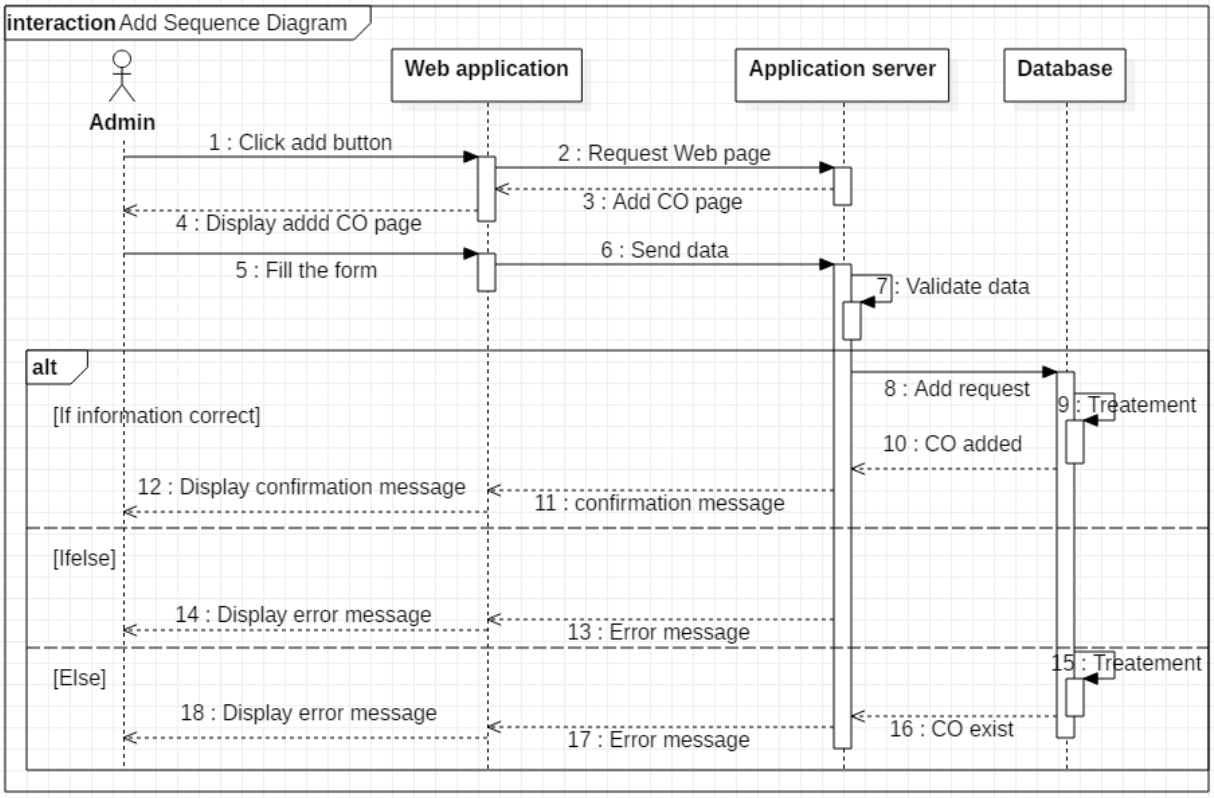


Figure 4.13: Sequence diagram of the operation add in the use case "Management of COs".

"Figure 4.14 " shows delete operation:

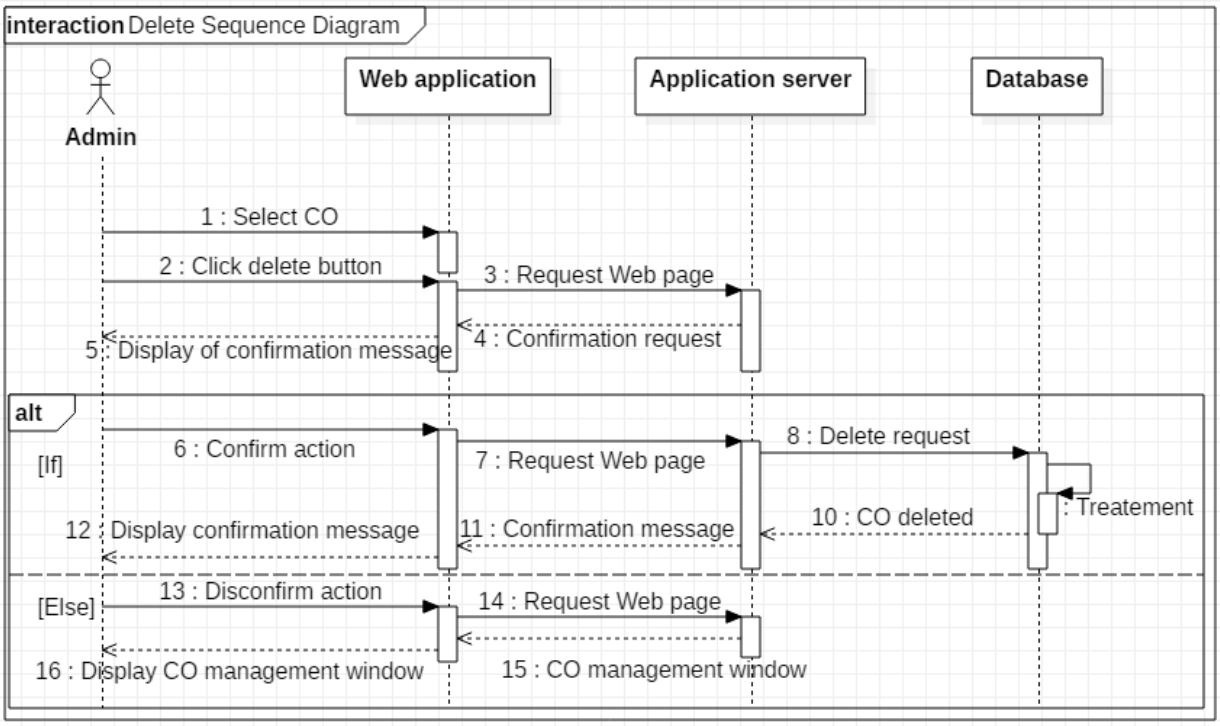


Figure 4.14: Sequence diagram of the operation delete in the use case "Management of COs".

"Figure 4.15 " shows modify operation:

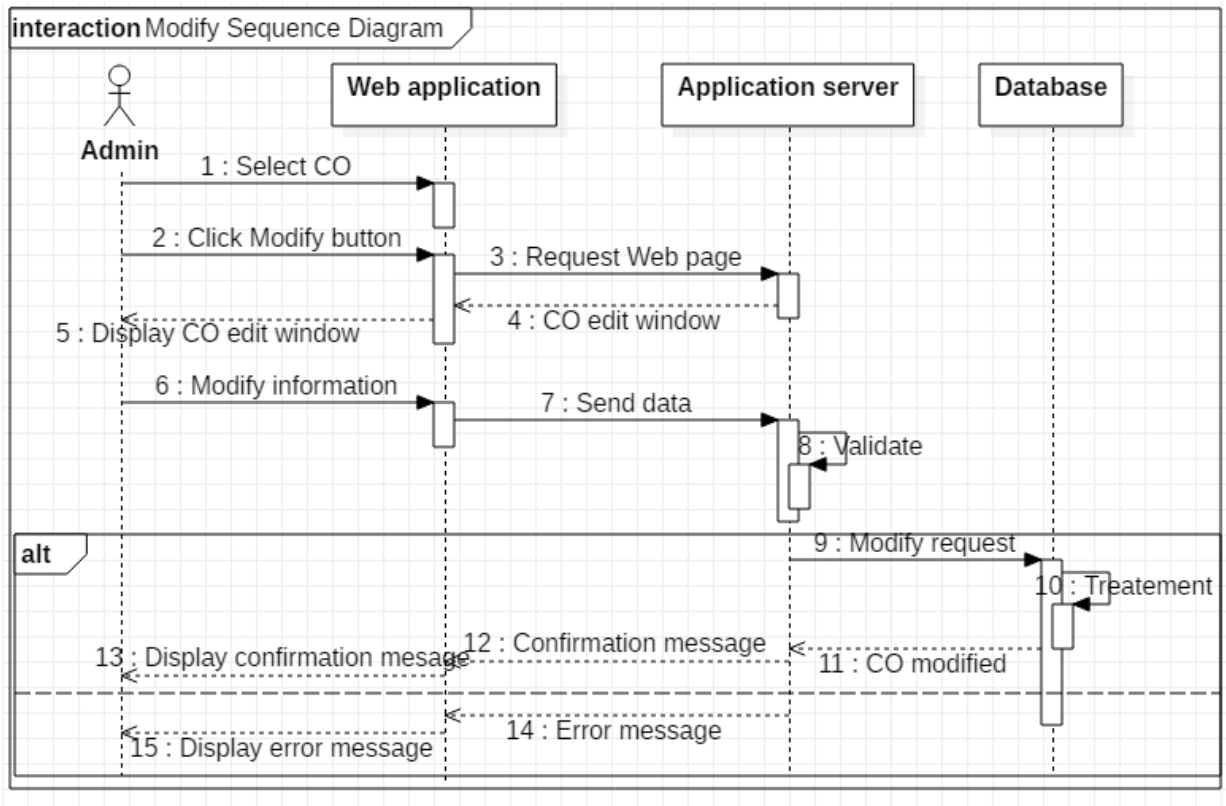


Figure 4.15: Sequence diagram of the operation modify in the use case "Management of COS".

4.7.4 Sequence diagram of the use case "Discover of COs"

When the Admin/User wants to discover COs, he must first authenticate himself, then access the COs management window, then he search for COs and get list of COs to read CO description as "Figure 4.16" shows.

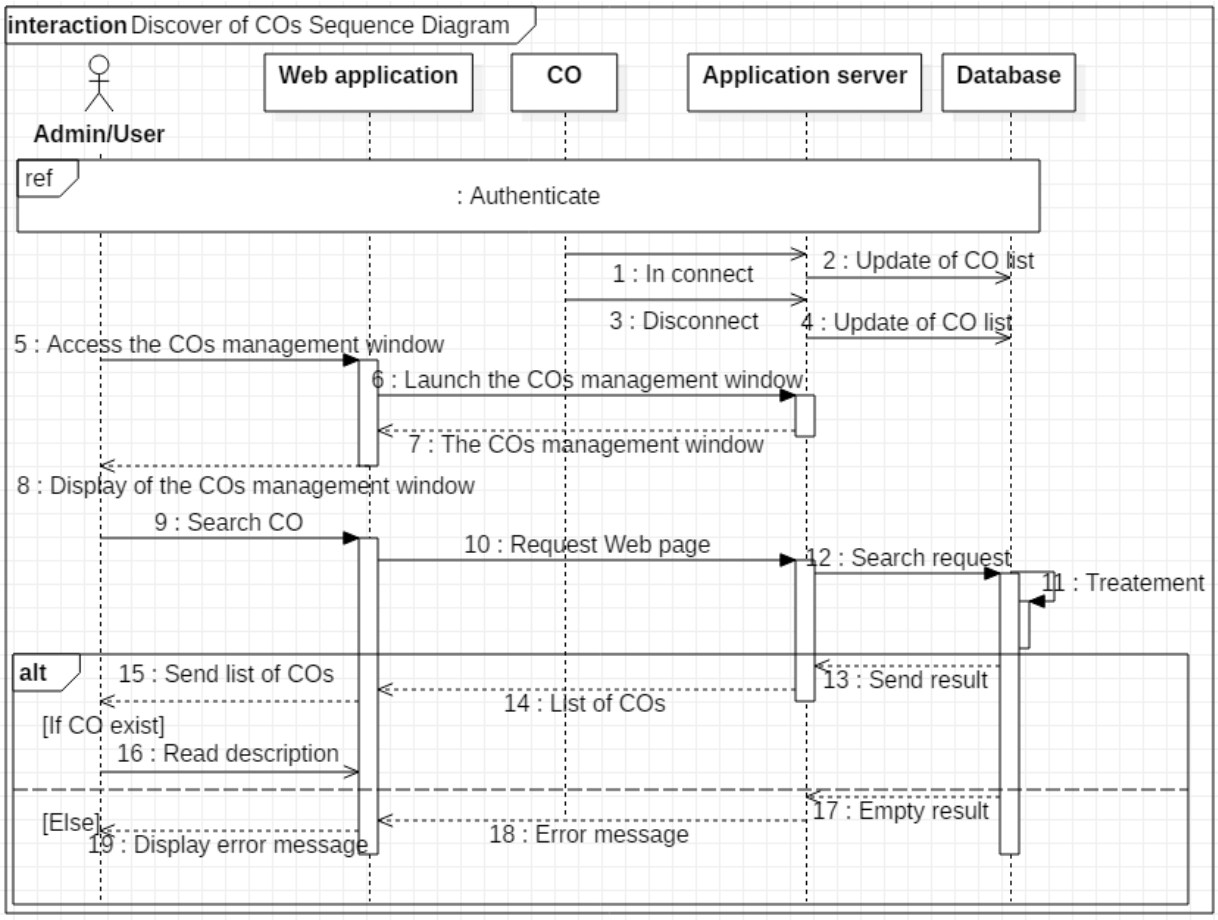


Figure 4.16: Sequence diagram of the use case "Discover of COs".

4.7.5 Sequence diagram of the use case "Invoke action"

When the Admin/User wants to Invoke actions to COs, he must first authenticate himself, then access the COs management window, then he discovers the COs to choose the ideal CO, then he sends the actions parameter (action name and arguments of actions), if CO is free then it performs an action and notifies user, otherwise it records action in FIFO list and notifies user as "Figure 4.17" shows.

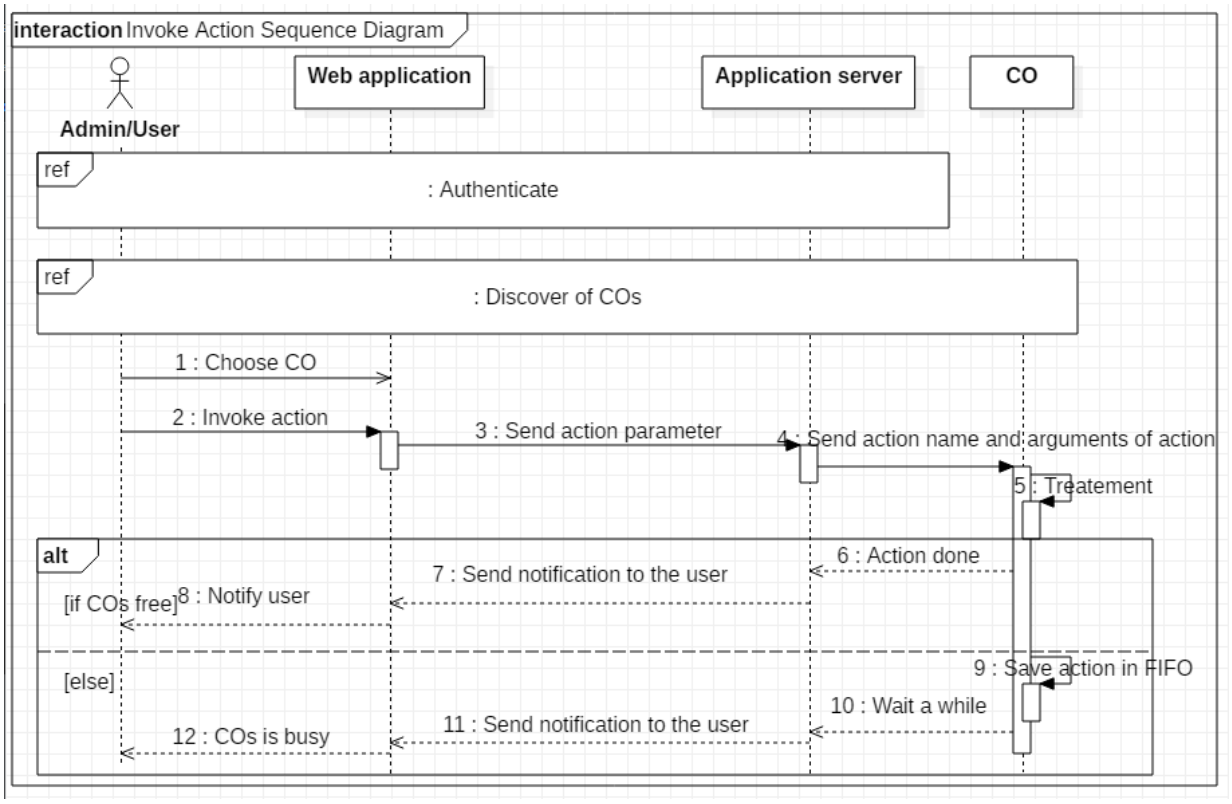


Figure 4.17: Sequence diagram of the use case "Invoke action".

4.7.6 Sequence diagram of the use case "Notify users"

When the CO changes his state and wants to notify users, the user must first authenticate himself, then the CO sends the new state value to the application server, the application server update information of the CO in the Database and sends a notification to the Web application to display to the users as "Figure 4.18" shows.

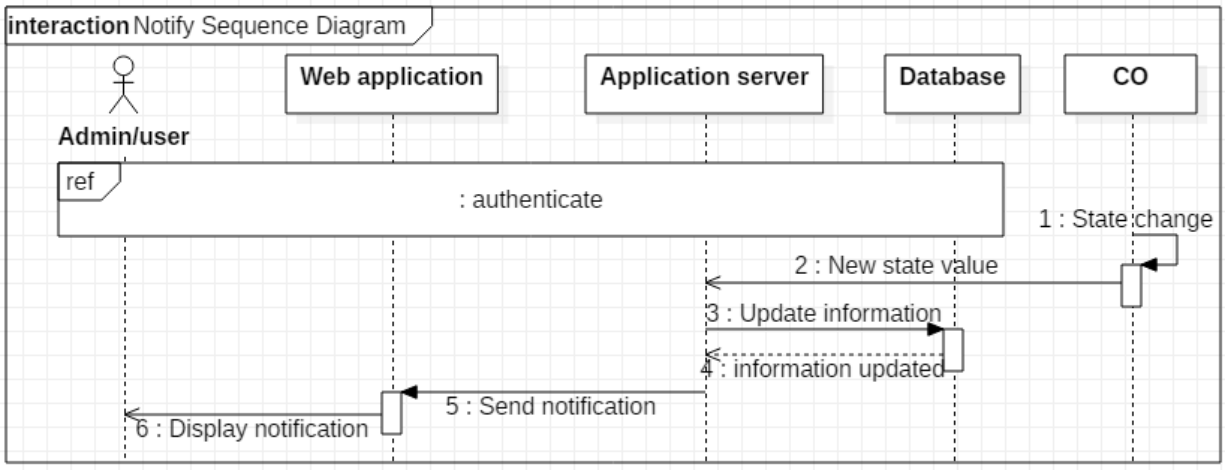


Figure 4.18: Sequence diagram of the use case "Notify users".

4.7.7 Sequence diagram of the use case "Communicate"

When the Admin/User wants to communicate with colleagues, he must first authenticate himself, then access the chat window, then he selects the colleague that he wants to communicate with him, then he chooses to send a message or make an audio/video call (the offer includes all the information about the caller's proposed configuration for the call, answer is the reply of the recipient, which is a description of their end of the call and the candidate is the information about the network connection) as "Figure 4.17" shows.

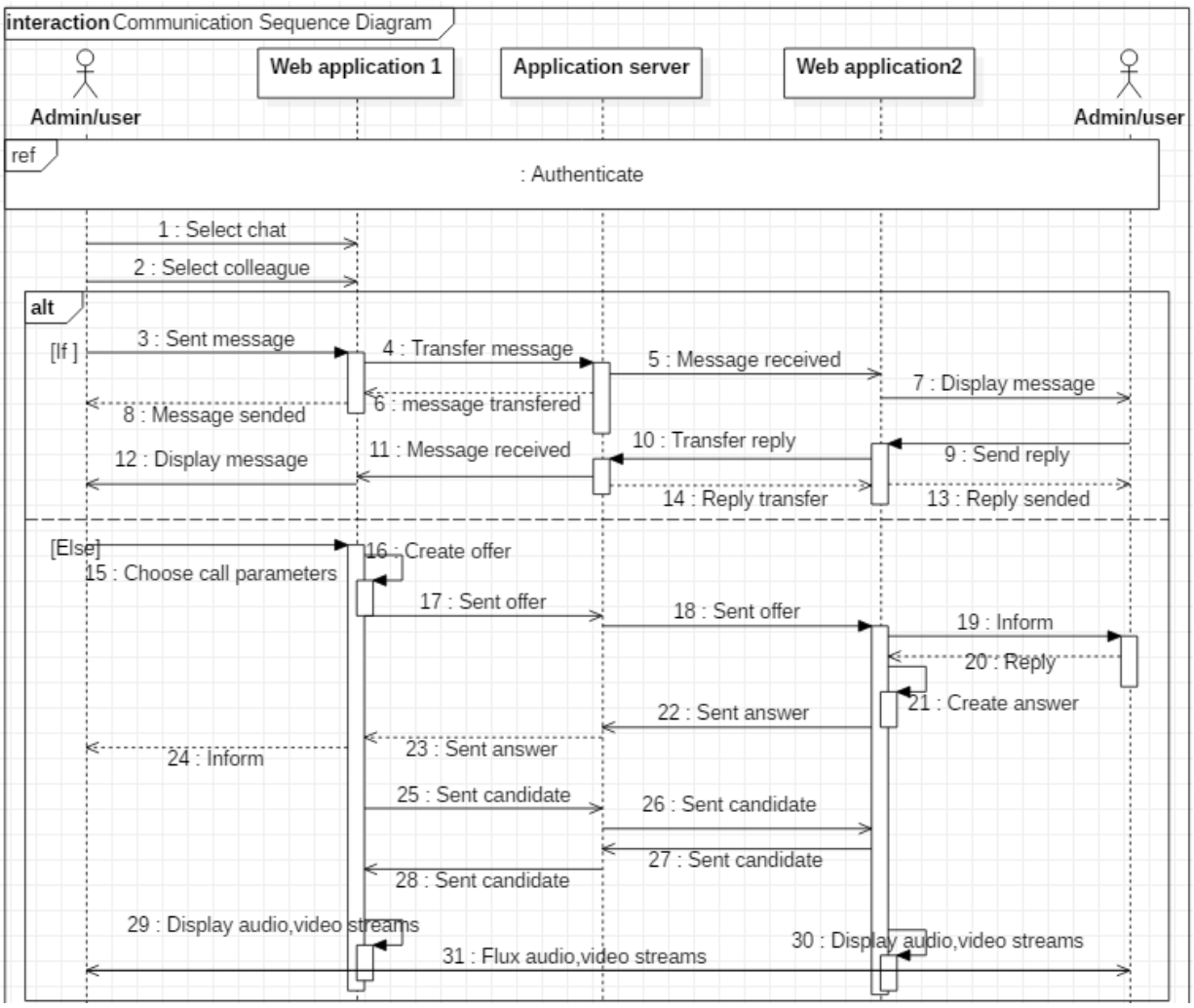


Figure 4.19: Sequence diagram of the use case "Communicate".

4.8 Class diagram

UML class diagram gives an overview of a software system by displaying classes, attributes, operations, and their relationships [42]. "Figure 4.20" shows class diagram for our platform.

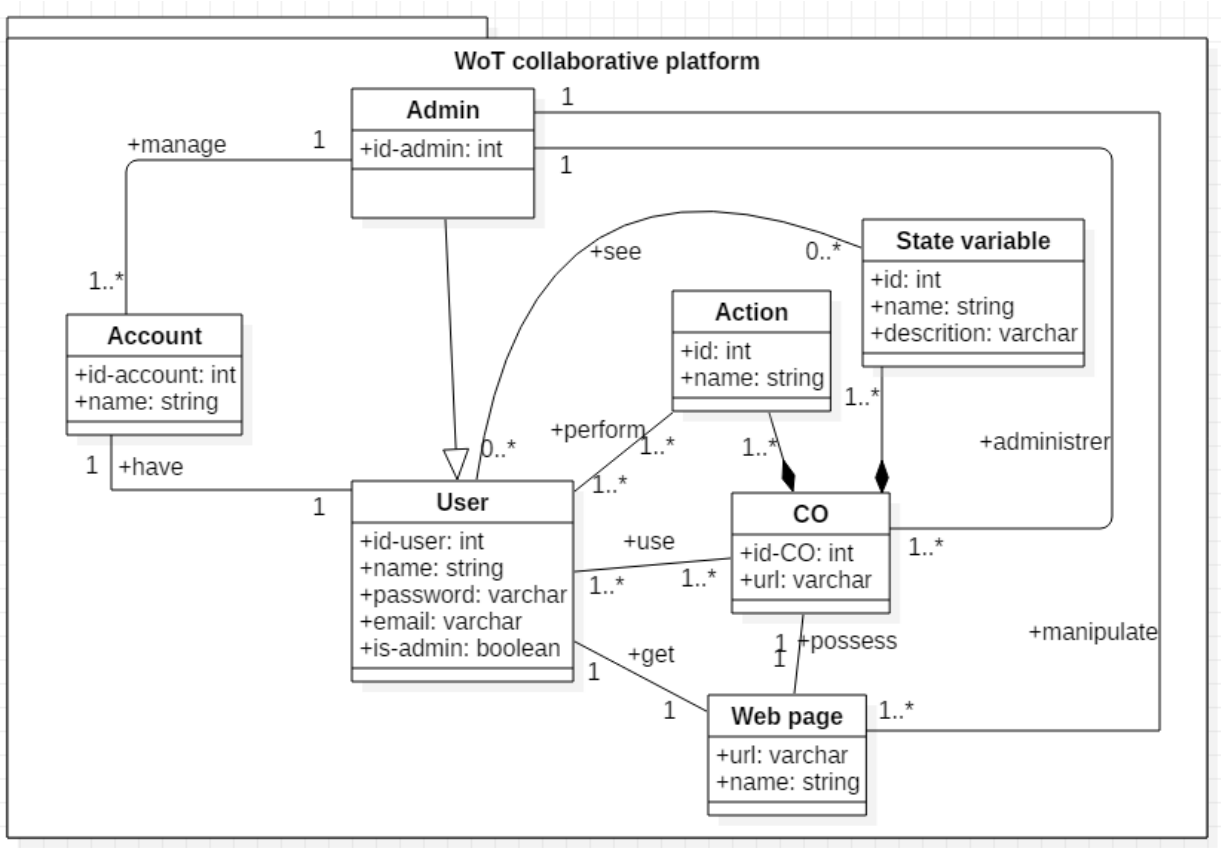


Figure 4.20: Class diagram.

4.9 HMI

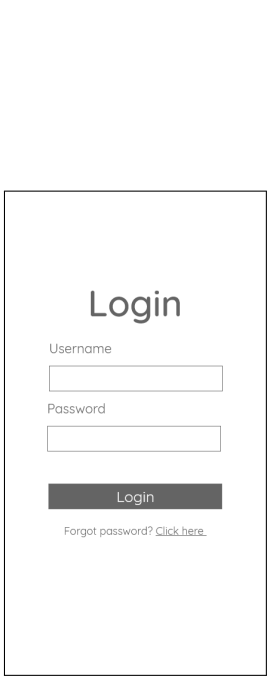
The basic target of the design of human-machine interface is to reduce errors, improve the ability to conduct, and enhance the security and comfortableness of operation in the human-machine interaction. Based on the previous diagrams and *HMI Ergonomic Criteria* we designs the following interfaces :

4.9.1 Authentication (Login)

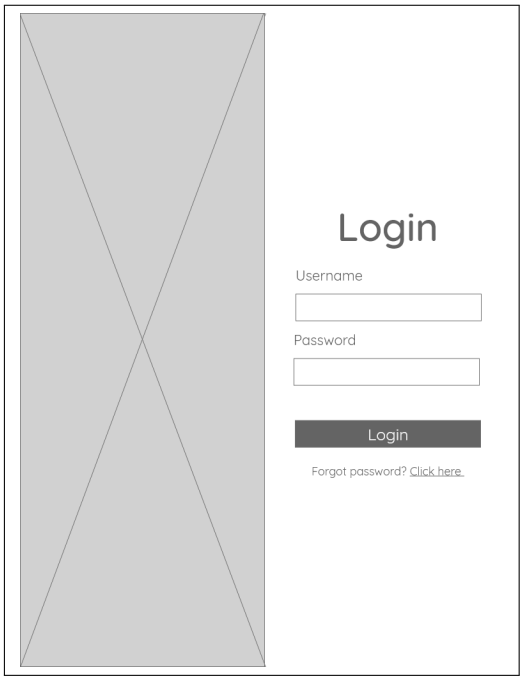
Authentication page is a simple and easy to understand, it contains :

- a username field .
- a password field .
- a clickable link in case of forgetting the password.

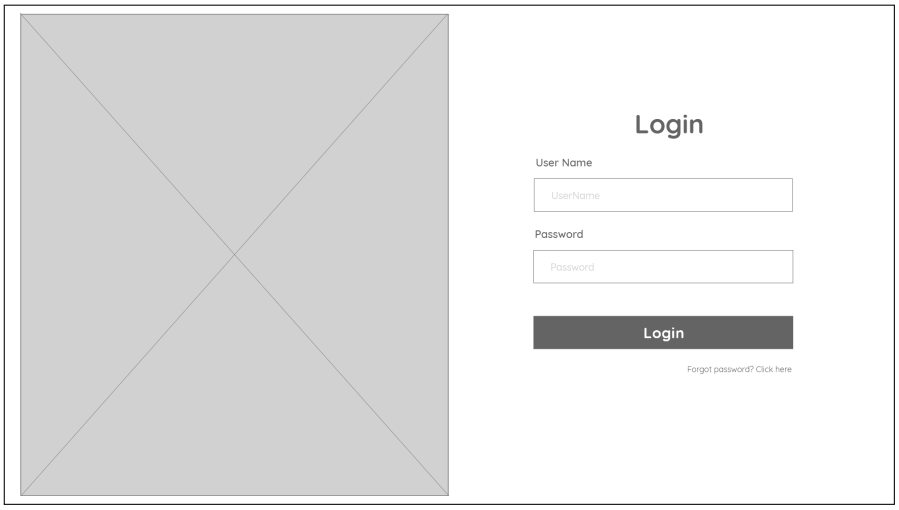
"Figure 4.21" shows the Authentication pages.



(a) login page mobile



(b) login page tablet



(c) login page desktop

Figure 4.21: Authentication

4.9.2 Navigation

In order to reach all sections of the platform, we have designed a navbar that enables users navigate easily. The navbar contains five buttons (icons) and user's profile image as "Figure 4.22 " shows:

1- Notifications.

2- Mesaages chat.

3- Home page for user dashboard for admin.

4- Video call.

5- Settings.

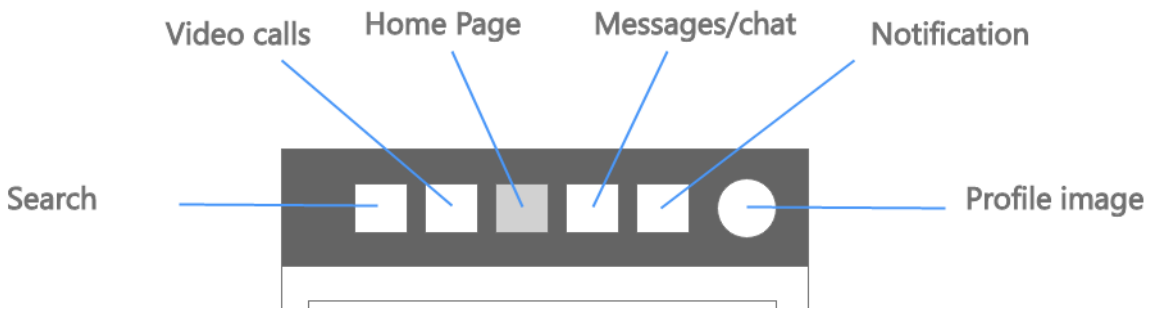


Figure 4.22: Mobile Navbar

When user tap on profile image it will show a drop down menu contains :

- User profile.

- Settings.

- Logout.

4.9.3 Customization

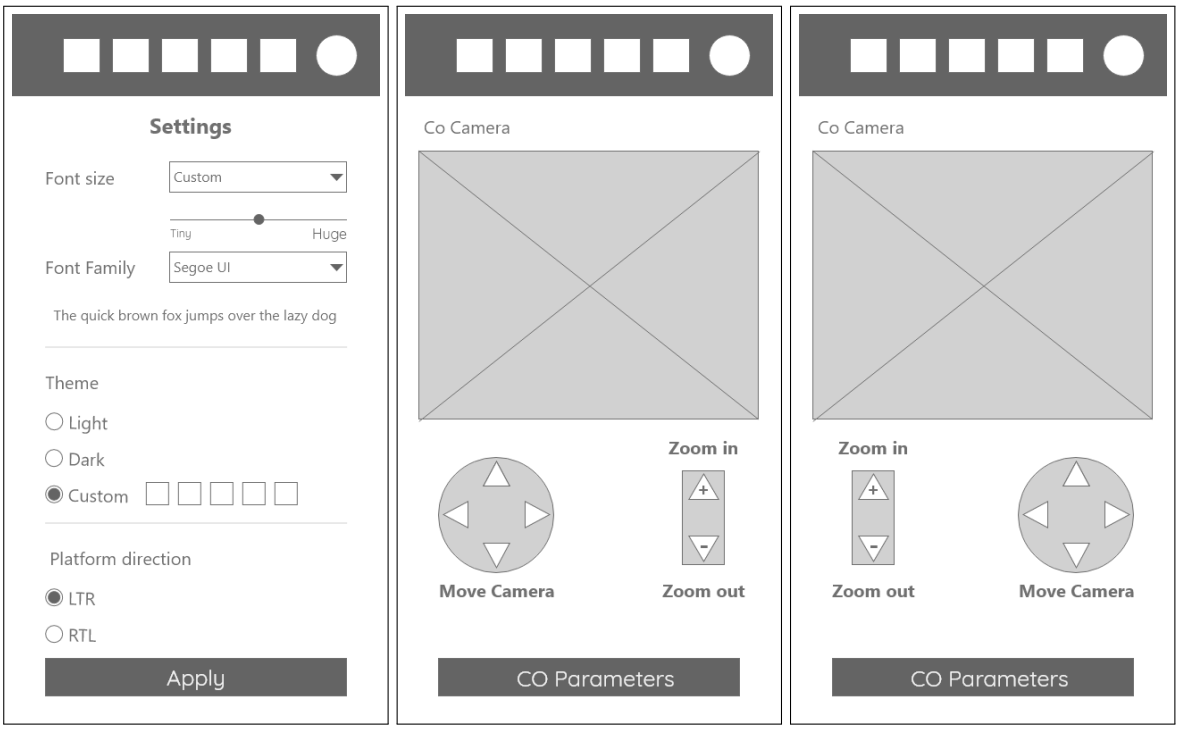
Customization is so important for many users to make their preferable settings :

- user can change font size , font family , direction LTR (left to right) or RTL (right to left).

- user can change buttons positions in CO control page.

- user can choose light Theme , dark theme or even custom colors theme .

"Figure 4.23" shows the Authentication pages.



(a) Settings (b) LTR interface (c) RTL interface

Figure 4.23: Customization

4.9.4 Communication

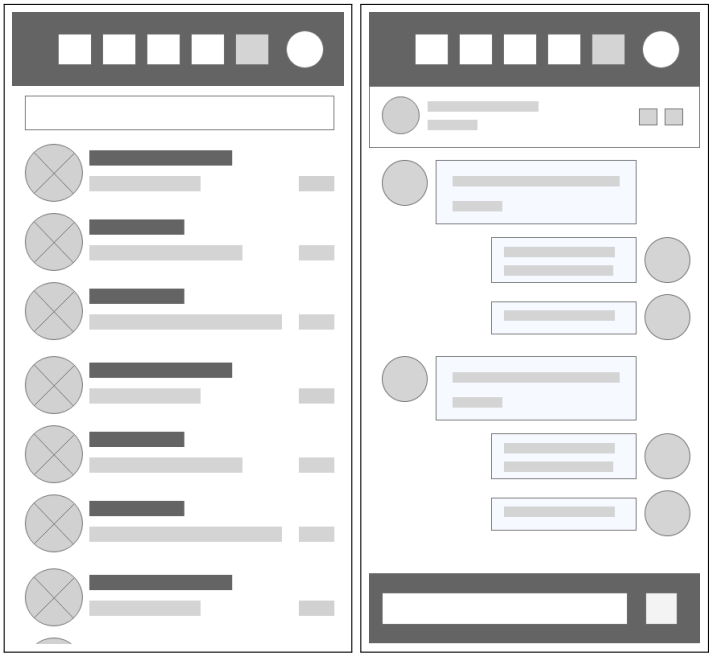
Communication is divided into two parts (messages / video calls).

4.9.4.1 Messages

Tapping messages icon will show you members and there last messages "Figure 4.24(a)". Choosing a member will redirect you to messages chat "Figure 4.24(b)".

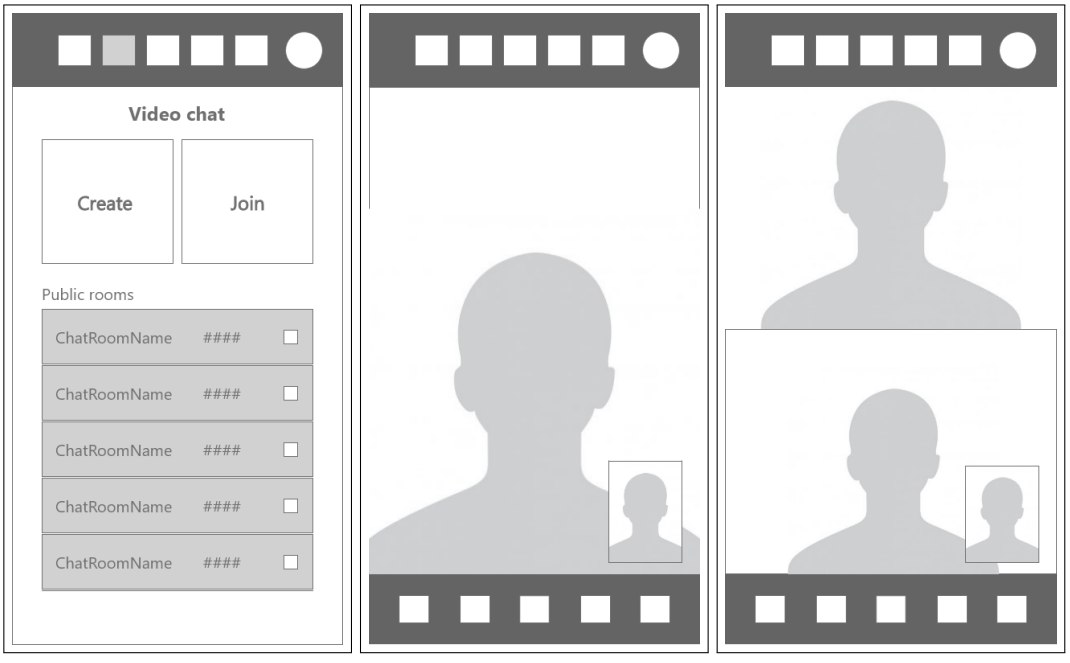
4.9.4.2 Video calls

Video calls main page is easy to understand it contains 2 big buttons where user can create or join a video call rooms or even brows public rooms as shown in "Figure 4.25(c)" .



(a) Messages page (b) chat page

Figure 4.24: Chat section

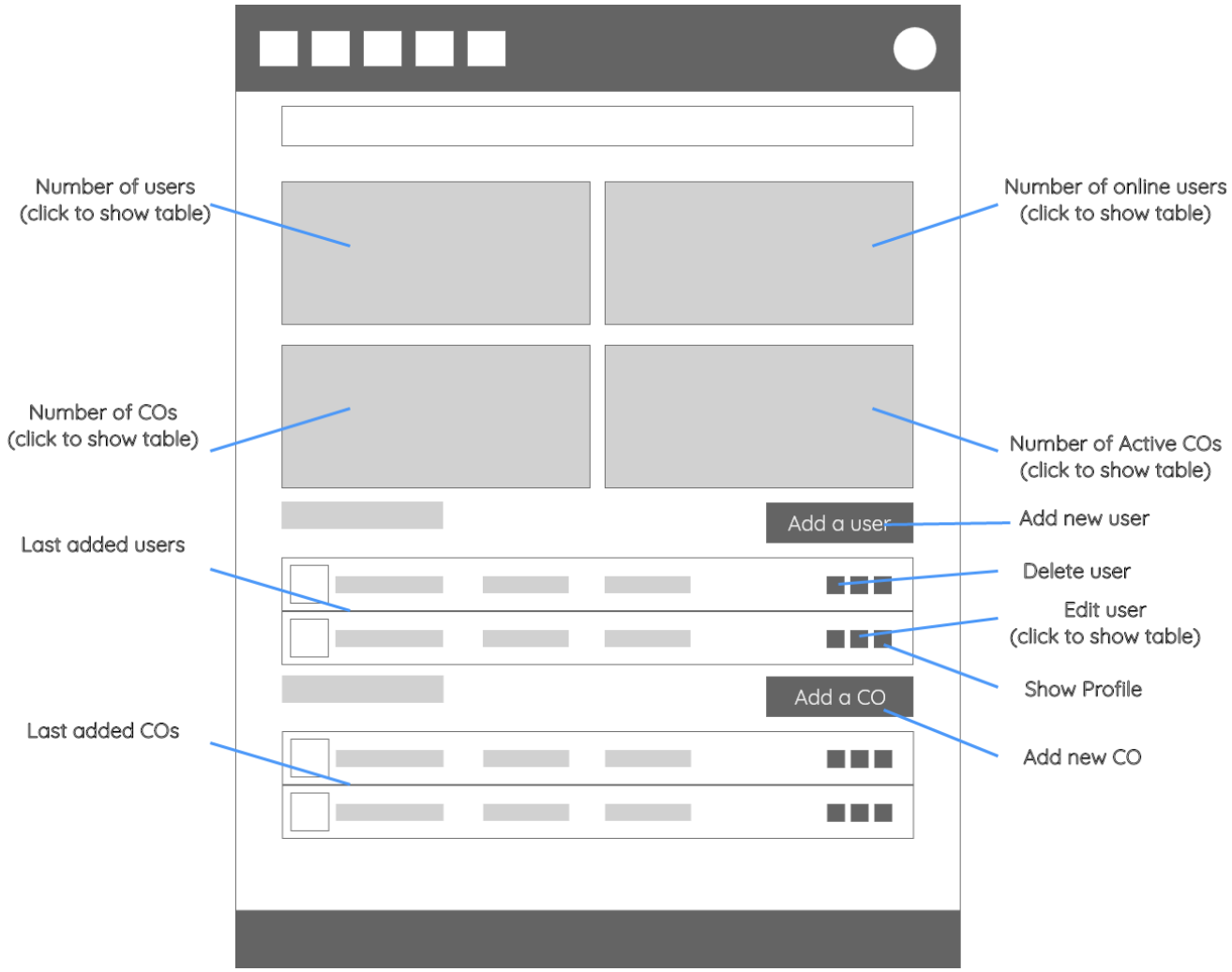


(a) video call main page (b) video call 2 users (c) video call 3 users

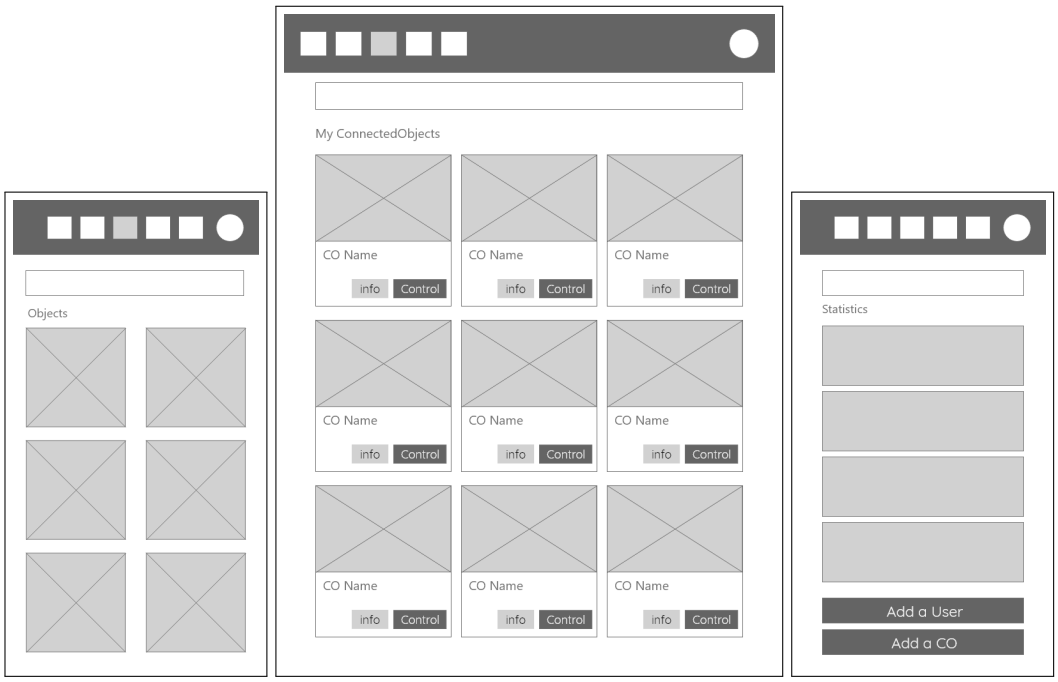
Figure 4.25: Video Call Section

4.9.5 Home Pages

There are Two types of Home Pages , user home page and Admin Home Page (dashboard) as shown in "Figure 4.26(b)".



(a) Admin page Tablet



(b) Home page contains COs

(c) home page Tablet

(d) admin page mobile

Figure 4.26: Home/Admin pages

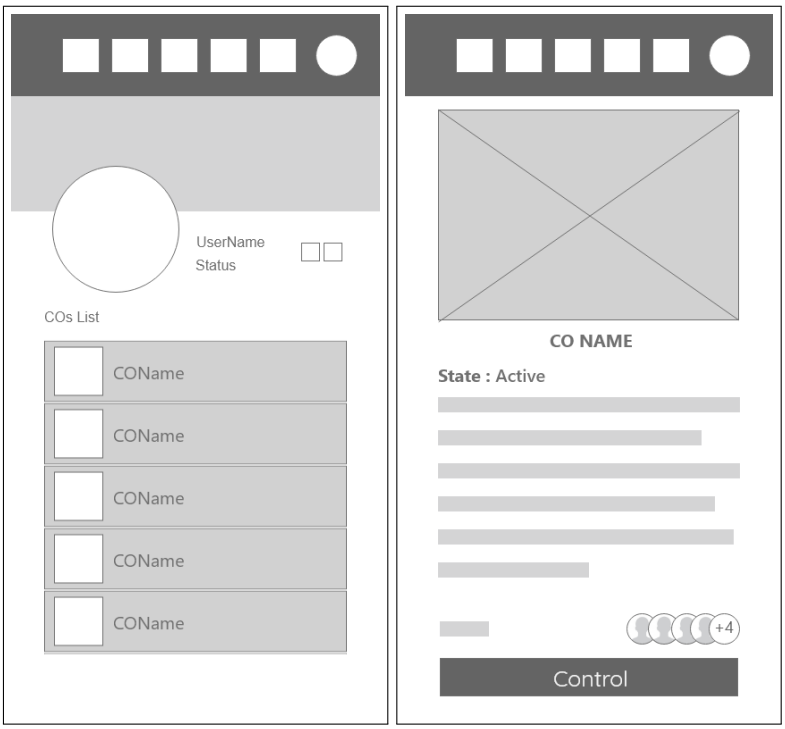
4.9.6 Profiles

There are Two types of profiles (User/Admin profile and CO profile) :

- User/Admin profile contains their information such as images,status(online/offline) and COs list that can control.

- CO profile contain CO info such as images and description.

"Figure 4.27" shows types of profiles pages.



(a) User profile (b) CO profile

Figure 4.27: profiles pages

4.10 Conclusion

This chapter is very important, especially to understand the context of our project, and its content, we made a general presentation and an architecture of our platform, then we added the development methode. In addition, the general design was made based on the standards of UML modeling language (use case diagram, sequence diagram and class diagram) and finally we designed HMI.

The following chapter will focus on the implementation of our application.

Chapter V

Implementation

5.1 Introduction

In this chapter we will present our implementation of our application, we will start by mention all the used development tools, materials and the different interfaces allowing interaction between the users and the system. Finally, we will end with a conclusion.

5.2 Software tools

The software tools contain diagram development tool, programming language, servers and editors.

5.2.1 Languages

Several languages are used to describe our platform.

5.2.1.1 HTML5

HTML5 (Hyper Text Markup Language 5) is designed to make the development of web applications easier, more natural, and more logical, where developers can design and build once, and deploy anywhere. HTML5 makes web applications more usable, as well, as it removes the need for plugins [18].

5.2.1.2 CSS3

Stands for "Cascading Style Sheet." Cascading style sheets are used to format the layout of Web pages. They can be used to define text styles, table sizes, and other aspects of Web pages that previously could only be defined in a page's HTML.

CSS helps Web developers create a uniform look across several pages of a Web site. Instead of defining the style of each table and each block of text within a page's HTML, commonly used styles need to be defined only once in a CSS document [43].

5.2.1.3 JavaScript

JavaScript is a programming language commonly used in web development. It was originally developed by Netscape as a means to add dynamic and interactive elements to websites. While JavaScript is influenced by Java, the syntax is more similar to C and is based on ECMAScript, a scripting language developed by Sun Microsystems [44].

5.2.2 Servers

We use different servers :

5.2.2.1 Node.js

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices [45].

5.2.2.2 MySQL

MySQL is a freely available open source Relational Database Management System (RDBMS) that uses Structured Query Language (SQL) [46]. We used Mysql Ver 15.1 Distrib 10.4.13-MariaDB.

5.2.3 Editors

We use different types of editors

5.2.3.1 Visual Studio Code

Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages (such as C++, C#, Java, Python, PHP, Go) and runtimes (such as .NET and Unity) [47].

5.2.3.2 StarUML

To realize the UML diagrams we used StarUML softaware.

StarUML is an open source software modeling tool that supports the UML framework for system and software modeling. It is based on UML version 1.4, provides eleven different types of diagram and it accepts UML 2.0 notation. It actively supports the MDA (Model Driven Architecture) approach by supporting the UML profile concept and allowing to generate code for multiple languages [48].

5.2.3.3 Adobe XD

Adobe XD is a tool for helping user experience and user interface designers prototype and wireframe new mobile and web applications [49], and collaborate on everything from prototypes to mockups to full designs [50].

5.3 Hardware tools

In this section, we use different hardware tools.

5.3.1 Laptop

we use modele 'ASUS Zenbook UX305U' core i7 6500U.

5.3.2 Arduino Mega 2560

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560.

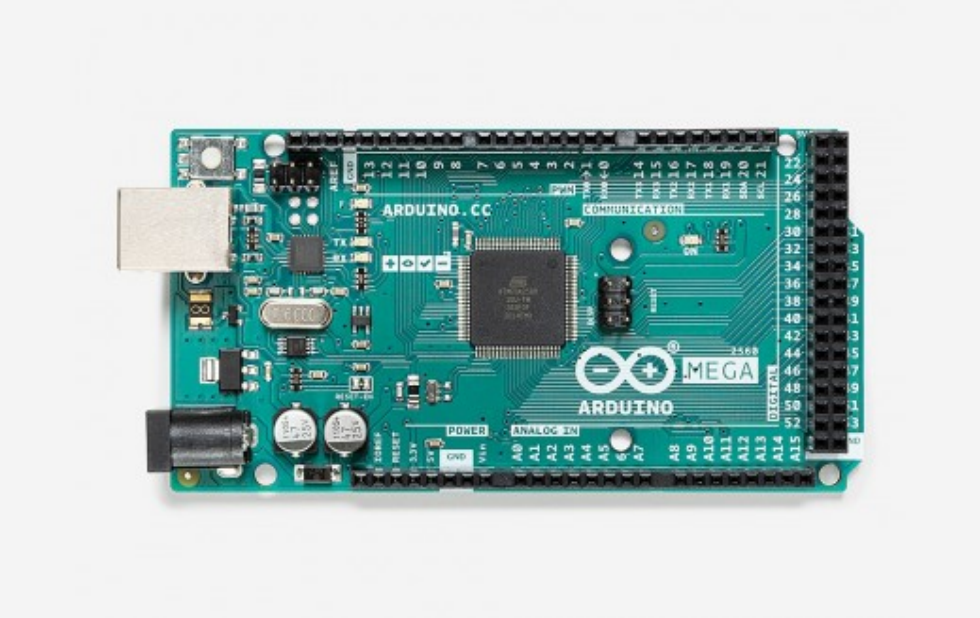


Figure 5.1: Arduino Mega 2560

5.3.3 Router

We use D-Link DSL-2750U Wireless N 300 ADSL2+ Modem Router , to connect our Local Network.

5.4 Platform interface

Based on previous chapter we built the following interface.

5.4.1 Login pages

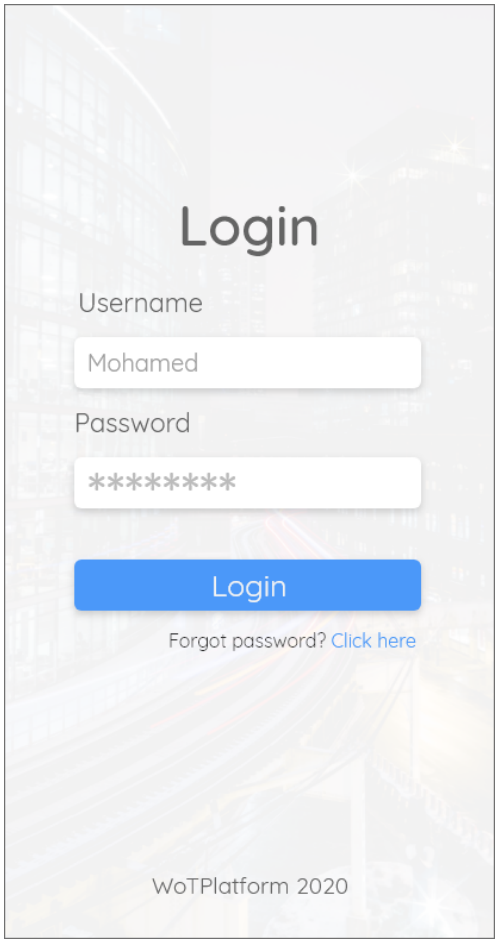


Figure 5.2: Login page mobile Web-application

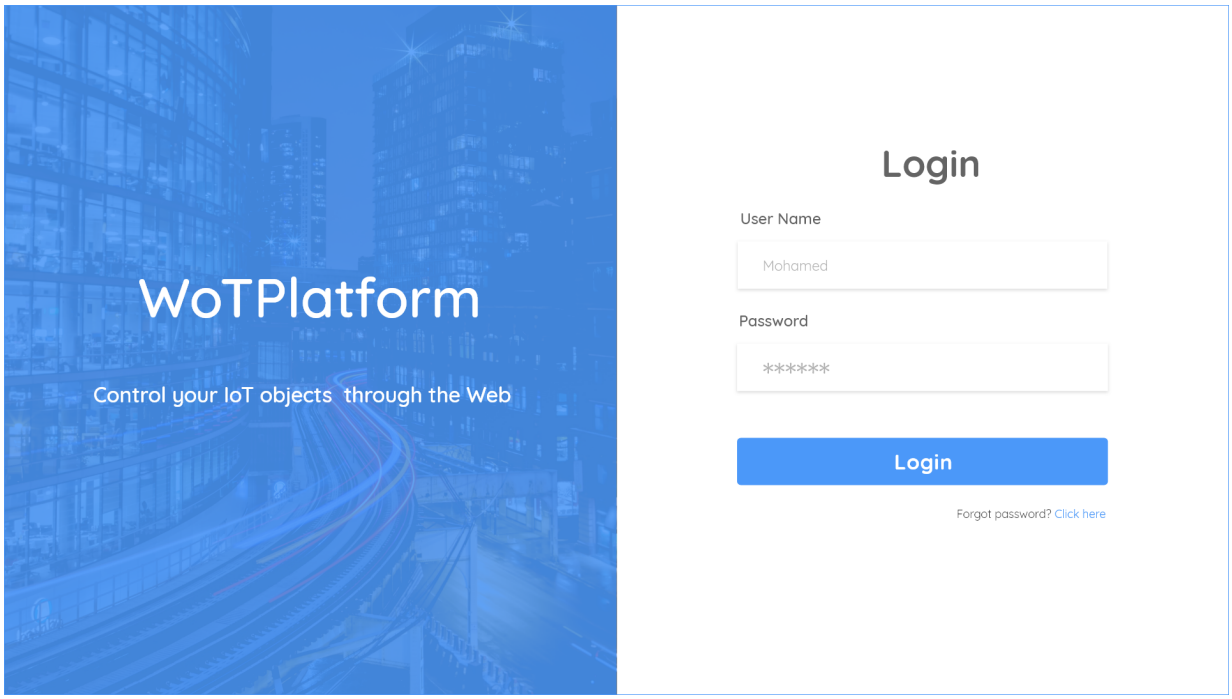


Figure 5.3: Login page disktop Web-application

5.4.2 Home/Dashboard pages



Figure 5.4: Dashboard page mobile Web-application

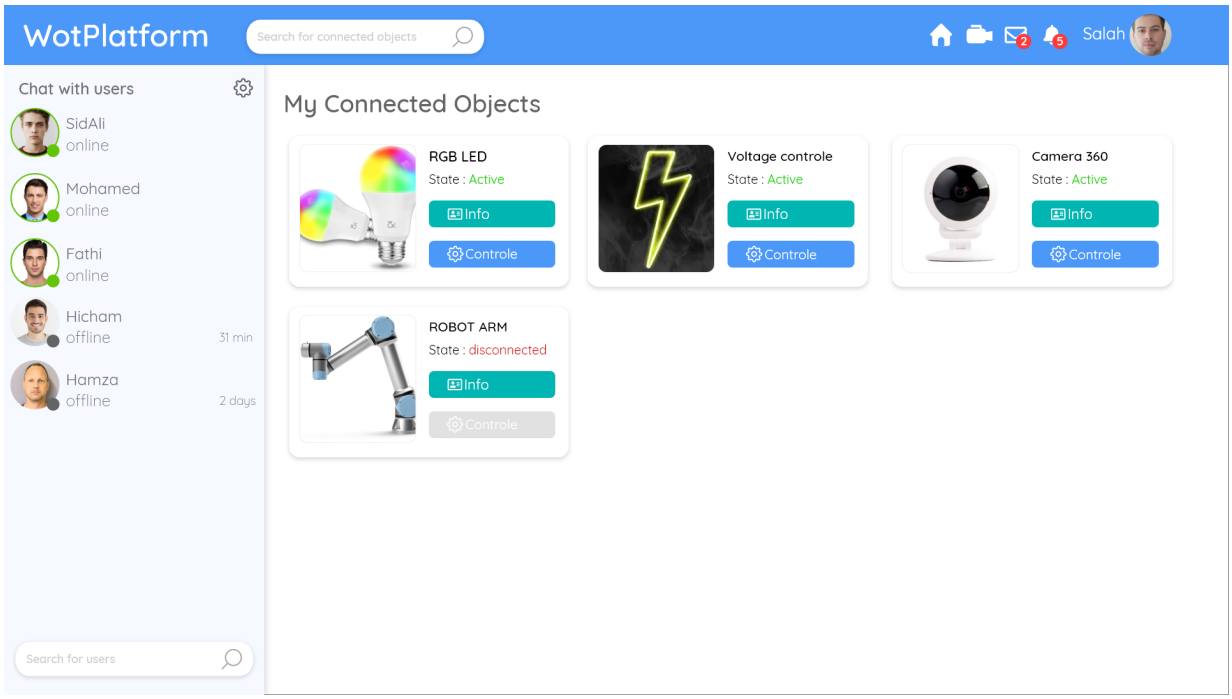


Figure 5.5: Home page desktop

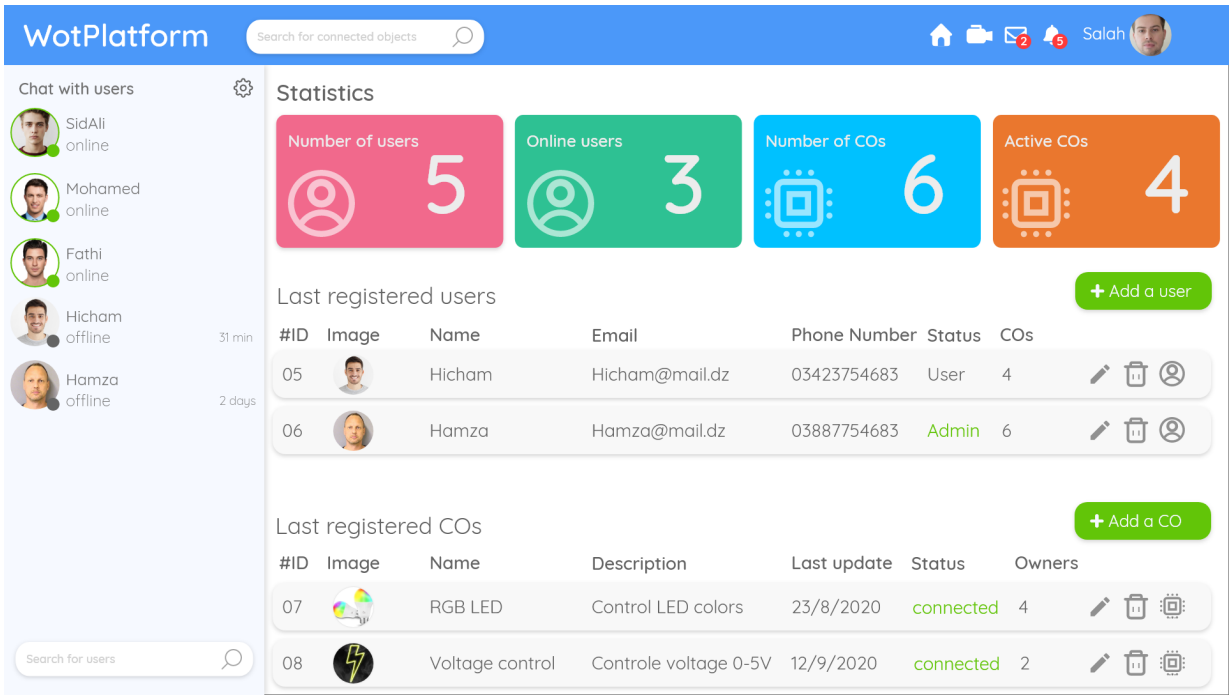
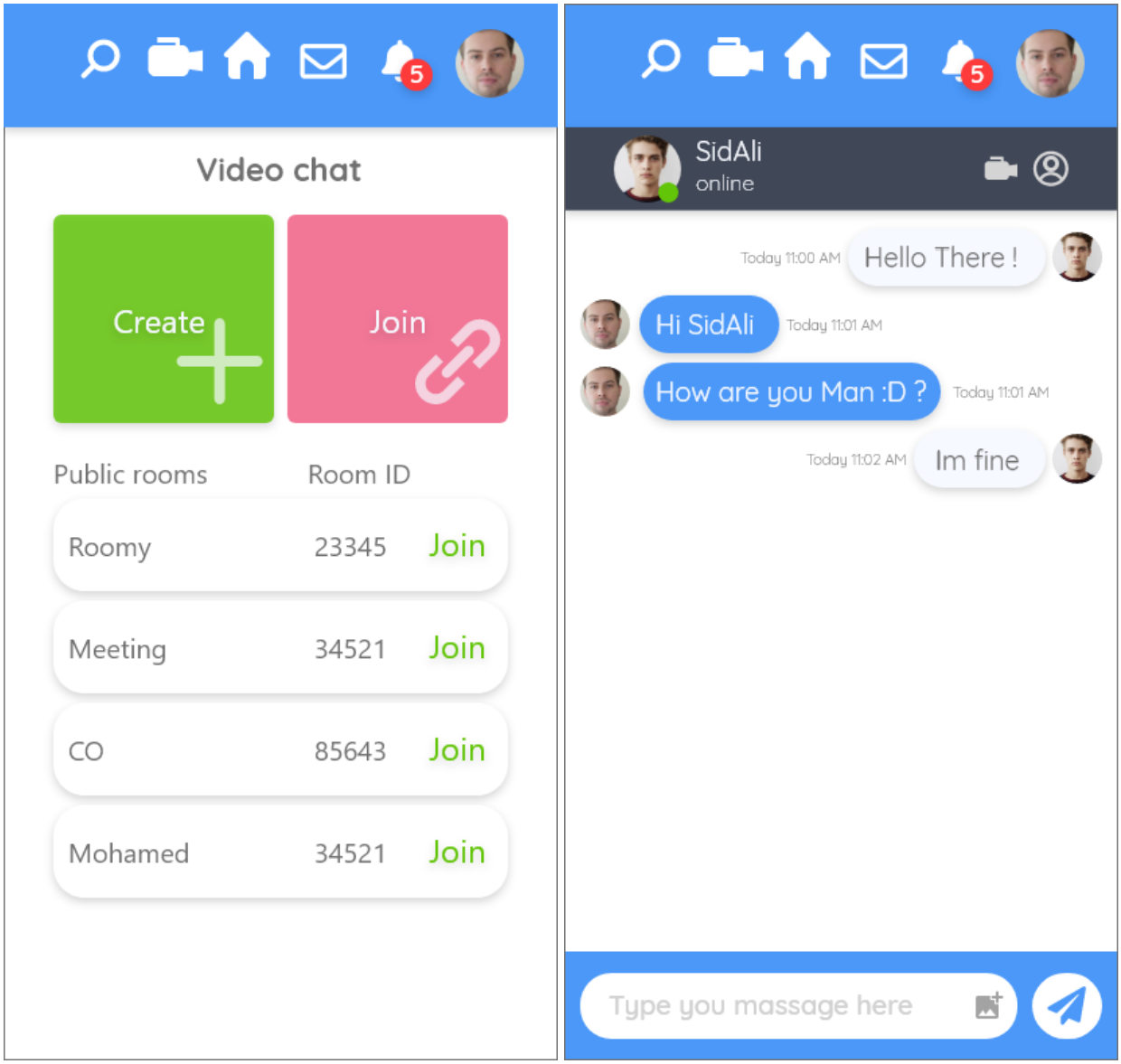


Figure 5.6: Dashboard page desktop Web-application

5.4.3 Chat pages



(a) Video chat (b) Messages chat

Figure 5.7: Chat pages mobile Web-application

5.4.4 CO pages



Figure 5.8: Camera as a CO

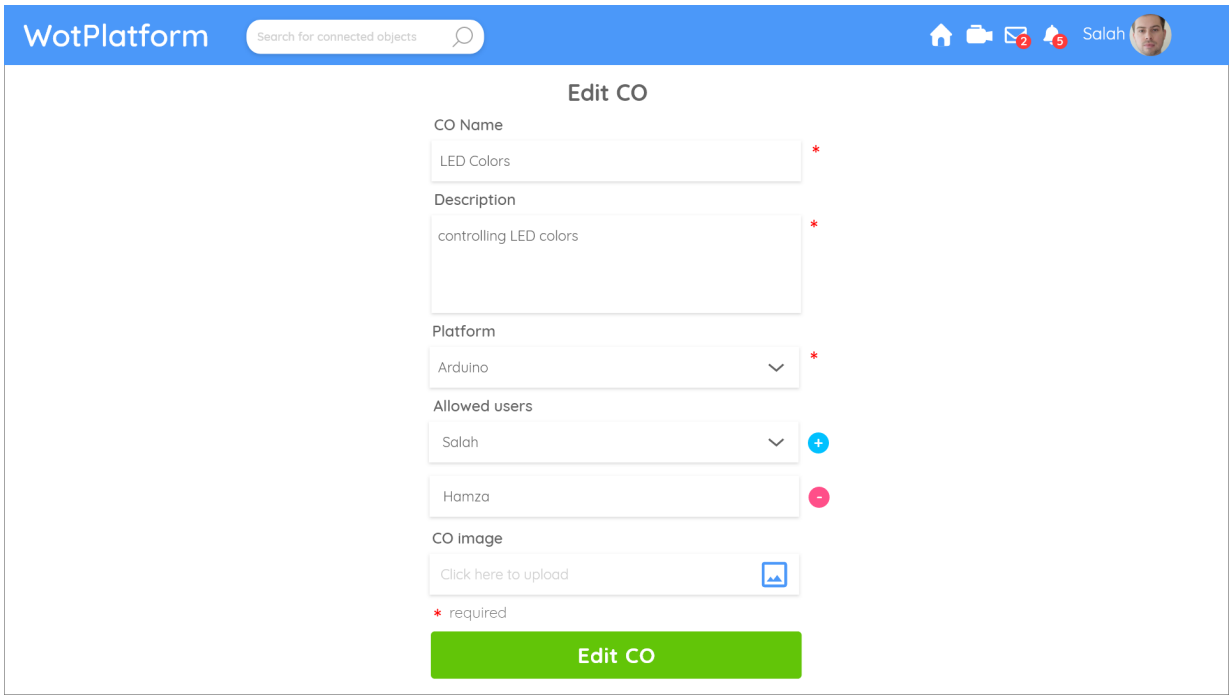


Figure 5.9: Edit CO page Desktop

5.5 Conclusion

In this last chapter, We wanted to respond to what we presented in the design chapter and give a clearer idea, also describe the different components and interfaces of our application.

General conclusion

WoT is a technology that has imposed it self on topical research thanks to their applications and domains, such as smart domestic, transport and health care. This last enables CO to speak the same language, such as to communicate and interoperate freely on the Web. This work aims to propose a collaborative platform respecting the ergonomic criteria of the HMI which allow to manage several connected objects in collaboration.

Every user has a web page that contains all COs that he can use , also he can communicate and collaborate with other users through it, and every CO object managed by the users, show its status and notify them. There is also an admin Web page, who can grant access to users and manage COs.

These web pages respect a number of HMI ergonomic standards to make them easier to use

In the future work, we want to improve the security and simplicity of this platform and use it in different areas to bring benefits.

Bibliography

[1] Linus Schwab and Arnaud Durand. Universal explorer for the web of things. *Master's thesis. Faculty of Science University of Bern*, 2018.

[2] Deze Zeng, Song Guo, and Zixue Cheng. The web of things: A survey. *JCM*, 6(6):424–438, 2011.

[3] Mahmoud Elkhodr, Qusay F Hassan, and Seyed Shahrestani. *Networks of the Future: Architectures, Technologies, and Implementations*. CRC Press, 2017.

[4] Knud Lasse Lueth. Why the internet of things is called internet of things: Definition, history, disambiguation. *IoT Analytics*, 19, 2014.

[5] Dominique Guinard. What is the web of things? `https://webofthings.org/2017/04/08/what-is-the-web-of-things/`, April 2017.

[6] Dominique D Guinard and Vlad M Trifa. *Building the web of things*, volume 3. Manning Publications Shelter Island, 2016.

[7] Keyur K Patel, Sunil M Patel, et al. Internet of things-iot: definition, characteristics, architecture, enabling technologies, application & future challenges. *International journal of engineering science and computing*, 6(5), 2016.

[8] Nitin Naik. Choice of effective messaging protocols for iot systems: Mqtt, coap, amqp and http. In *2017 IEEE international systems engineering symposium (ISSE)*, pages 1–7. IEEE, 2017.

[9] Jaap Haartsen. Bluetooth-the universal radio interface for ad hoc, wireless connectivity. *Ericsson review*, 3(1):110–117, 1998.

[10] What is wi-fi? `https://www.cisco.com/c/en/us/products/wireless/what-is-wifi.html`.

[11] Marilyn Wolf. *Embedded System Interfacing: Design for the Internet-of-Things (IoT) and Cyber-Physical Systems (CPS)*. Morgan Kaufmann, 2019.

[12] mark liles. Arduino tutorial part 1 – overview and hardware. http://play.karlssonrobotics.com/tutorials/arduino/arduino-tutorial-part-1-overview-hardware/, february 2016.

[13] Arduino getting started guide. https://www.arduino.cc/en/guide/introduction.

[14] Nick Health. What are the best raspberry pi alternatives? everything you need to know about pi rivals. https://www.zdnet.com/article/what-are-the-best-raspberry-pi-alternatives-everything-you-need-to-know-about-pi#:~:text=The%20%2410%20Raspberry%20Pi%20Zero,Wi%2DFi%20and%20Bluetooth%204.0., july 2019.

[15] Charles Bell. *Beginning sensor networks with Arduino and Raspberry Pi*. Apress, 2014.

[16] Patrick John McGee. The web of things with mozilla open badges. https://www.slideshare.net/PatrickJohnMcGee/the-web-of-things-with-mozilla-open-badges-40521825, October 2014.

[17] Dominique Guinard. *A web of things application architecture: Integrating the real-world into the web*. PhD thesis, ETH Zurich, 2011.

[18] Vanessa Wang, Frank Salim, and Peter Moskovits. *The definitive guide to HTML5 WebSocket*, volume 1. Springer, 2013.

[19] Christian Nwamba. Websockets vs server-sent events. https://www.telerik.com/blogs/websockets-vs-server-sent-events, February 2019.

[20] Andrea Arcuri. Restful api automated test case generation. In *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, pages 9–20. IEEE, 2017.

[21] Salvatore Loreto and Simon Pietro Romano. *Real-time communication with WebRTC: peer-to-peer in the browser*. " O'Reilly Media, Inc.", 2014.

[22] M Bîzoi, AM Suduc, and FG Filip. Using collaborative platforms for decision support. In *Proceedings of the 17th International Conference on Control Systems and Computer Science*, pages 349–352, 2009.

[23] What is a collaboration platform? https://www.comidor.com/knowledge-base/enterprise-collaboration-kb/collaboration-platform-definition/, October 2017.

[24] Types of collaboration software: 5 groups you should know about. https://collaboration-software.financesonline.com/types-of-collaboration-software-5-groups-you-should-know-about/.

[25] Greg Digneo. 64 online collaboration tools to help your team be more productive. https://biz30.timedoctor.com/online-collaboration-tools/.

[26] Margaret Rouse. Dropbox. https://searchmobilecomputing.techtarget.com/definition/Dropbox.

[27] Trello. https://www.computerhope.com/jargon/t/trello.htm, 2019.

[28] Proofhub review. https://reviews.financesonline.com/p/proofhub/.

[29] MELLOUL Besma. Plateforme collaborative pour contrôler IdO en utilisant le WebRTC, 2019.

[30] Christian Broser, Christoph Fritsch, Oliver Gmelch, and Günther Pernul. Impact analysis of collaborative platforms-the case of spike. *IFAC Proceedings Volumes*, 42(25):24–29, 2009.

[31] Roland Schulz Hanna Zieschang Marlen Cosmar, Peter Nickel. Human machine interface. https://oshwiki.eu/wiki/Human_machine_interface?fbclid=IwAR076TeeESopWv0rg692ag_X4Le3XkVE9gNtv5CFW9c--nsadgef1HB9CO8.

[32] inductiveautomation. What is hmi? https://www.inductiveautomation.com/resources/article/what-is-hmi, August 2018.

[33] Dominique L Scapin and JM Christian Bastien. Ergonomic criteria for evaluating the ergonomic quality of interactive systems. *Behaviour & information technology*, 16(4-5):220–231, 1997.

[34] Ivar Jacobson, Grady Booch, and James Rumbaugh. The unified process. *Ieee Software*, 16(3):96, 1999.

[35] BrainKart. Unified software development process or unified process. http://www.brainkart.com/article/Unified-Software-Development-Process-or-UnifiedProcess_9971/.

[36] scott W. Ambler. The agile unified process (aup) home page. http://www.ambysoft.com/unifiedprocess/agileUP.html, May 2006.

[37] sciencedirect. Software designing with unified modeling language driven approaches. https://www.sciencedirect.com/topics/computer-science/unified-process#:~:text=Each%20phase%20and%20its%20iteration,the%20activities%20within%20requirements%20analysis).

[38] Keng Siau and Qing Cao. Unified modeling language: A complexity analysis. *Journal of Database Management (JDM)*, 12(1):26–34, 2001.

[39] AMIT. Uml diagrams. https://tallyfy.com/uml-diagram/#:~:text=A%20UML%20diagram%20is%20a,document%20information%20about%20the%20system.

[40] uml diagrams.org. Uml 2.5 diagrams overview. https://www.uml-diagrams.org/uml-25-diagrams.html.

[41] Sequence diagram. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/.

[42] Uml class diagram tutorial with examples. https://www.guru99.com/uml-class-diagram.html.

[43] CSS Definition. techterms.com. https://techterms.com/definition/css.

[44] Techterms. Javascript. https://techterms.com/definition/javascript, august 2014.

[45] tutorialspoint. Node.js - introduction. https://www.tutorialspoint.com/nodejs/nodejs_introduction.htm.

[46] sitegrounds. What is mysql tutorial. https://www.siteground.com/tutorials/php-mysql/mysql/.

[47] visualstudio. visual studio code. https://code.visualstudio.com/docs.

[48] Franco Martinig. Staruml - open source uml tool. https://www.methodsandtools.com/tools/staruml.php, 2011.

[49] Frederic Lardinois. Adobe's xd prototyping and wireframing tool is now out of beta. https://techcrunch.com/2017/10/18/adobe-xd-designing-at-the-speed-of-thought/, October 2017.

[50] Carrie Cousins. What is adobe xd? a 101 intro. https://designshack.net/articles/software/what-is-adobe-xd/, November 2019.