

**People's Democratic Republic of Algeria**  
**Ministry of Higher Education and Scientific Research**  
**University M'Hamed BOUGARA – Boumerdes**



**Institute of Electrical and Electronic Engineering**  
**Department of Electronics**

Final Year Project Report Presented in Partial Fulfilment of  
the Requirements for the Degree of

**MASTER**

**In Electronics**

**Option: Computer Engineering**

Title:

**Design and Implementation of a Web-based  
Management System. Case of The  
Scientific Council of the “IGEE”.**

Presented By:

- **BAATCHIA Riyadh**
- **ATALLAH Mohamed Nadhir**

Supervisor:

**Dr NAMANE R.**

Registration Number:...../2019

## **Abstract**

This project proposes a technical solution to improve the management for a small institution. The aim of this project is to create an interactive web application (application that can run from a web browser) dedicated to the management of the Scientific Council of the institute (SCI). This web application makes it possible to facilitate the work of the members of the SCI by automating the management of the submissions, meetings, Final decisions, statistics and the members' roles. The application we will develop should be easy to set up and maintain by users with little or no technical expertise.

To improve the management process, this project goal is to analyze the current management processes of a hypothetical scientific council and determine the possible areas of improvement. This work has been carried out using the development process and the modelling language, Unified Modelling Language (UML), for designing the application. As for the implementation, we have chosen to program the application with NodeJS and MongoDB as the Database Management System (DBMS).

Keywords: Web application, NodeJS, Mongo DBMS, and UML.

## Dedication

*I dedicate this modest work to my dear parents  
, my brothers, and to all my family and friends.*

*Nadhir*

*I dedicate this modest work to my dear parents  
, my brothers, and to all my family and friends.*

*Riyadh*

## **Acknowledgements**

We would firstly like to thank our project supervisor, Dr. Namane for his assistance, and valuable feedback in the realization of this senior design project; also, he was our teacher for two years. Secondly, we would like to thank Dr. Zitouni for his valuable course in basics of Object Oriented programming. Dr. Khalifa for his valuable course about databases

Finally, we would like to thank our families and friends for their presence, support, and help.

## Table of contents

<b>Abstract</b> .....	I
<b>Dedication</b> .....	II
<b>Acknowledgement</b> .....	III
<b>Table of contents</b> .....	IV
<b>List of tables</b> .....	VII
<b>List of Figures</b> .....	VIII
<b>List of abbreviations:</b> .....	IX
<b>General Introduction:</b> .....	1
<b>CHAPTER ONE : An overview of project objectives and web applications</b> .....	3
1.1 Introduction: .....	3
1.2 Subject presentation: .....	3
1.2.1 problem statement: .....	3
1.2.2 Objectives: .....	3
1.3 Web application: .....	4
1.3.1 Definition: .....	4
1.3.2 Single-Page Application: .....	4
1.3.3 Multi-Page Application: .....	4
1.3.4 Web client and Web server: .....	5
1.3.5 HTML and HTTP: .....	5
1.3.6 Understanding URL: .....	8
1.3.7 The Communication Procedure: .....	8
1.4 Conclusion: .....	10
<b>CHAPTER TWO : Overview of the Scientific Council</b> .....	11
2.1 Introduction: .....	11
2.2 Scientific Council presentation: .....	11
2.3 Staff role identification: .....	12
2.3.1 SCI President: .....	12
2.3.2 SCI Member: .....	13
2.3.3 Faculty Member / PhD Student: .....	13

2.4 Conclusion: .....	13
<b>CHAPTER THREE : Tools and technologies</b> .....	14
3.1 Introduction: .....	14
3.2 Development tools: .....	14
3.2.1 EDraw Max: .....	14
3.2.2 Visual Studio Code:.....	14
3.2.3 NodeJS:.....	14
3.2.4 ExpressJS:.....	15
3.2.5 ReactJS: .....	15
3.2.6 GraphQL:.....	15
3.2.7 Apollo GraphQL:.....	17
3.2.4 MongoDB: .....	17
3.2.8 Web Browser:.....	17
3.3 Programming languages:.....	18
3.3.1 HTML:.....	18
3.3.2 CSS:.....	18
3.3.3 JavaScript: .....	18
3.4 Conclusion: .....	18
<b>CHAPTER FOUR : Design</b> .....	20
4.1 Introduction: .....	20
4.2 Design Requirements: .....	20
4.2.1 Modelling language: .....	20
4.2.2 Unified Modelling Language (UML):.....	20
Behavioral UML diagrams:.....	20
4.3 Modelling .....	21
4.3.1 System Actors:.....	21
4.3.2 Use Cases: .....	21
4.3.3 Relationships between documents:.....	32

4.3.8 Sequence Diagrams: .....	35
4.4 Introduction to Database: .....	41
4.4.1 Definition:.....	41
4.4.2 Relational Model: .....	41
4.4.3 Non-Relational Model: .....	41
4.4.4 What is MySQL? .....	42
4.4.5 What is MongoDB?.....	42
4.4.6 Benefits of using documents in database: .....	42
4.4.7 Description of our system's database: .....	43
4.5 Conclusion: .....	47
<b>CHAPTER FIVE : Implementation</b> .....	48
5.1 Introduction: .....	48
5.2 Interfacing with the application: .....	48
5.2.1 Authentication interface: .....	48
5.2.2 Homepage interfaces for each user: .....	49
5.2.3 Open submissions interface:.....	50
5.2.4 Submit an application interface:.....	51
5.2.5 Set meeting agenda interface:.....	51
5.2.6 Review applications interface: .....	52
5.2.7 My applications interface: .....	55
5.2.8 Statistics interface:.....	56
5.3 Conclusion: .....	57
General Conclusion .....	58
Future Works .....	58
Webography.....	59
Bibliography .....	60

## **List of tables**

Table 4. 1 use cases of each actor according to their roles .....	22
Table 4. 2 Summary of the used arrows.....	24
Table 4. 3 Textual description for use case "Authentication" .....	29
Table 4. 4 Textual description for use case " Open submissions " .....	29
Table 4. 5 Textual description for use case " Set meeting agenda " .....	30
Table 4. 6 Textual description for use case " Review applications " .....	30
Table 4. 7 Textual description for use case " Final decision " .....	31
Table 4. 8 Textual description for use case " Submissions" .....	31
Table 4. 9 Textual description for use case " Statistics " .....	32
Table 4. 10 used arrows .....	33
Table 4. 11 Basic symbols used in sequence diagram .....	35
Table 4. 12 Message symbols used in sequence diagram .....	36



## List of Figures

Figure 1. 1 HTTP Request Message .....	6
Figure 1. 2 HTTP Response message .....	7
Figure 1. 3 A static client/server communication sequence.....	8
Figure 1. 4 A dynamic client/server communication sequence .....	9
Figure 4. 1 PhD Student use case diagram.....	25
Figure 4. 2 Faculty Member use case diagram .....	26
Figure 4. 3 embedded data relationship .....	32
Figure 4. 4 references relationship .....	33
Figure 4. 5 relationships between documents. ....	34
Figure 4. 6 Sequence diagram “Authentication”.....	36
Figure 4. 7 Sequence diagram “Open submissions” .....	37
Figure 4. 8 Sequence diagram “Submit application” .....	38
Figure 4. 9 Sequence diagram “Set meeting agenda” .....	39
Figure 4. 10 Sequence diagram “Review application” .....	40
Figure 4. 11 database interface.....	43
Figure 4. 12 Announcement collection .....	43
Figure 4. 13 applications collection .....	44
Figure 4. 14 Users collection .....	45
Figure 4. 15 yearly reports collection .....	47
Figure 5. 1 Authentication interface.....	48
Figure 5. 2 Homepage interface when submissions are closed.....	49
Figure 5. 3 Homepage interface when submissions are open .....	50
Figure 5. 4 Open submissions interface .....	50
Figure 5. 5 Submit an application interface .....	51
Figure 5. 6 Set meeting agenda interface .....	51
Figure 5. 7 Review applications interface.....	52
Figure 5. 8 the applicant information Interface.....	52
Figure 5. 9 internship application Interface .....	53
Figure 5. 10 the review status Interface for SC member user .....	53
Figure 5. 11 application assessment Interface.....	53
Figure 5. 12 the review status Interface for SC President user .....	54
Figure 5. 13 the final decision Interface.....	55
Figure 5. 14 My applications interface .....	55

Figure 5. 15 Statistics interface .....	56
Figure 5. 16 pie chart for an application interface .....	56

#### **List of abbreviations:**

- DBMS: Database Management System.
- UML: Unified Modelling Language.
- HTTP: Hypertext Transfer Protocol.
- HTML: Hypertext Mark-up Language.
- URL: Universal Resource Locator.
- MySQL: My Structured Query Language.
- SQL: Structured Query Language
- API: Application Program Interface
- JSON: JavaScript Object Notation
- AJAX: Asynchronous Javascript And XML

## **General Introduction:**

The Internet has taken the world of computers and communications to an extraordinary stage. The invention of phones, radios, and computers opened the way for this special integration of capabilities.

Until now the computers remain the safest way to process and backup information. This feature has made it possible to computerize the data management system of companies, which is an essential part of their development today.

A web application is an application that is accessed by users over a network. Users can easily access the application from any computer connected to the Internet using a standard browser. In another term it is a software system that provides a user interface through a web browser. A web application plays a very important role in every field such as in the field of education, health, and libraries. It can give ease in every field because a web application also saves the time of the clients. Over all, a web application is very useful and convenient for clients when internet is in everyone's reach.

Each university has its scientific council, one of the institutions that computers can help a lot these days. Our project aims to design and implement an interactive, reliable, user-friendly web application to facilitate the work of the scientific council staff.

This report explains the process in which a web application can help to simplify the work of the scientific council members, as well as the management of the meetings and the decisions making, which leads to a very professional job. In addition, our web application helps in the ease of using and maintaining all the information related to the council because it is saved in a database.

Our report is organized in five main chapters:

**The First chapter** introduces web applications, after defining the objectives of this project. **The second chapter** gives a general overview of a hypothetical scientific council; in particular it defines the scientific council of an institute, talks about the management of scientific council tasks, and summarizes its members role's. **The third chapter** introduces the tools and technologies used in the project. **The forth chapter** concerns the design. It brings together all the stages of our process of development using UML modeling language. **The fifth chapter** is devoted to the implementation of the project where we present some interfaces available to the application users. Our report ends with a **general conclusion**.

# ***CHAPTER ONE***

---

***An overview of project objectives  
and web applications***

## **1.1 Introduction:**

In this chapter, we identify the problems that may be faced by the Scientific Council members, then, we discuss our objectives in obtaining a better management of this later. Finally, we provide an introduction to web apps and some generalities related to running the web.

## **1.2 Subject presentation:**

Our objective is to design and implement a web application to manage the scientific council of our institute. this theme is proposed and chosen to solve the problems that face the council members.

### **1.2.1 Problem statement:**

In order to implement our project, we have discussed a lot with a scientific council member and we came up with some of the problems that the members are facing:

- The meetings take time and effort.
- Dealing with a lot of documents and files.
- Documents are stored on shelves, which is an unsecured place; therefore the risk of losing or damaging part or all the documents is very high.

### **1.2.2 Objectives:**

Our objective is to design and implement a web application to solve the problems mentioned above by:

- Automate the process of the scientific council tasks in our web application.
- Review, vote, comment the submitted files and documents from home.
- Registrations and submissions of files for an application will be online.
- Save all documents in a database, which secures and makes an ease access to that documents.
- Finally, a physical meeting will be held to confirm a final decision.

### **1.3 Web application:**

#### **1.3.1 Definition:**

A web application or "web app" is any software program that runs on a web server. Unlike the traditional desktop applications, which are launched by the operating system, web apps must be accessed through a web browser.

Web apps have several advantages over desktop applications. Since web apps run inside a web browser, no complex installation is needed. Web apps also solve some of the "compatibility issues", (Windows, Mac, Linux); all that is needed is a browser. Developers do not need to distribute software updates to users when the web app is updated. By updating the application on the server, all users have access to the updated version. [1]

#### **1.3.2 Single-Page Application:**

A single-page application is an app that works inside a browser and does not require page reloading during use. SPAs are all about serving an outstanding UX(user experience) by trying to imitate a "natural" environment in the browser — no page reloads, no extra wait time. It is just one web page that you visit which then loads all other content using JavaScript — which they heavily depend on. SPA requests the markup and data independently and renders pages straight in the browser.

Single-page sites help keep the user in one, comfortable web space where content is presented to the user in a simple, easy and workable fashion. [2]

#### **1.3.3 Multi-Page Application:**

Multiple-page applications work in a "traditional" way. Every change, for example, displays the data or submit data back to server requests rendering a new page from the server in the browser. These applications are large, bigger than SPAs because they have to be. Due to the amount of content, these applications have many levels of UI (user interface). Luckily, it's not a problem anymore. Thanks to AJAX, we don't have to worry that big and complex applications have to transfer a lot of data between the server and the browser. That solution allows to refresh only particular parts of the application. On the other hand, it adds more complexity and it is more difficult to develop than a single-page application. [2]

### **1.3.4 Web client and Web server:**

**Web client:** it is an application that communicates with a web server. Some of the widely used Web clients are web browsers such as Google chrome, Firefox... etc. When a user requests something from a server (through a URL) the web client takes care of creating a request and sending it to the server and then it parses the server response and presents it to the user.

**Web server:** it is a computer program that accept a request from the client and attempt to reply to it in a meaningful way. The machine the program runs on is usually also called a server. When a Web client sends a request into the internet asking to view the web page found at that address. The web server is the program or machine that responds to that request and delivers the content of the page back to the user.

A web server can usually handle multiple simultaneous connections and when it is not communicating with a client it spends its time listening for an incoming connection. When one arrives, the server sends back a response to confirm its receipt.

### **1.3.5 HTML and HTTP:**

Web Servers and Web Clients are two separate software, so they should have:

- A common language for communication: HTML (Hypertext Mark-up Language) is the common language between a server and a client.
- A common communication protocol: HTTP is the communication protocol between a server and a client.

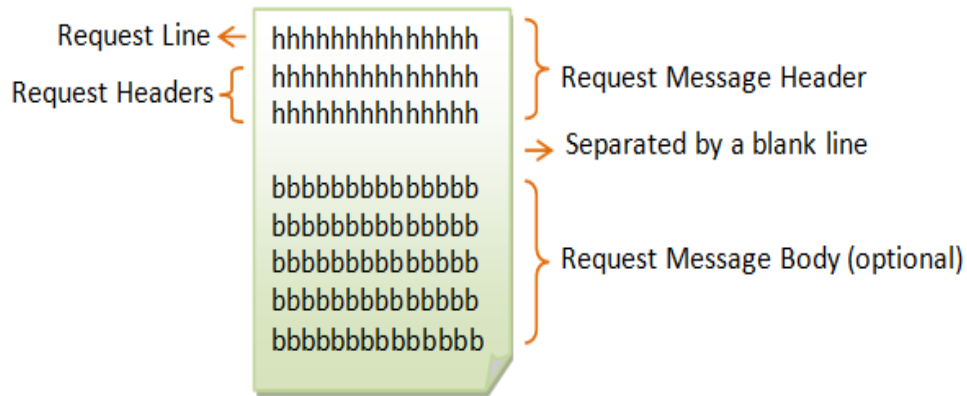
HTTP is a communication standard governing the requests and responses that take place between the browser running on the end user's computer and the web server. The server's job is to accept a request from the client and attempts to reply to it in a meaningful way, usually by serving up a requested web page—that's why the term server is used. The natural counterpart to a server is a client, so that term is applied both to the web browser and the computer on which it's running. [26]



Some of the important parts of HTTP are:

### 1.3.5.1 HTTP Request Message:

The format of an HTTP request message is as follow:



**Figure 1. 1 HTTP Request Message**

- **Request Line:** has the following parameters:
  - **request-method-name:** GET, POST, PUT, and OPTIONS.
  - **request-URI:** specifies the resource requested.
  - **HTTP-version:** Either HTTP/1.0 and HTTP/1.1.
- **Request Headers:**

The request header contains the type, version and capabilities of the browser that is making the request so that server returns compatible data.
- **Request message body:**

Some requests send data to the sever in order to update it, it is often the case with POST requests (containing HTML form data). [3]



### 1.3.6 Understanding URL's:

A Uniform resource locator (URL) is the address of a resource on the Internet. A URL indicates the location of a resource as well as the protocol used to access it. Also known as a Universal Resource Locator (URL) or Web address.

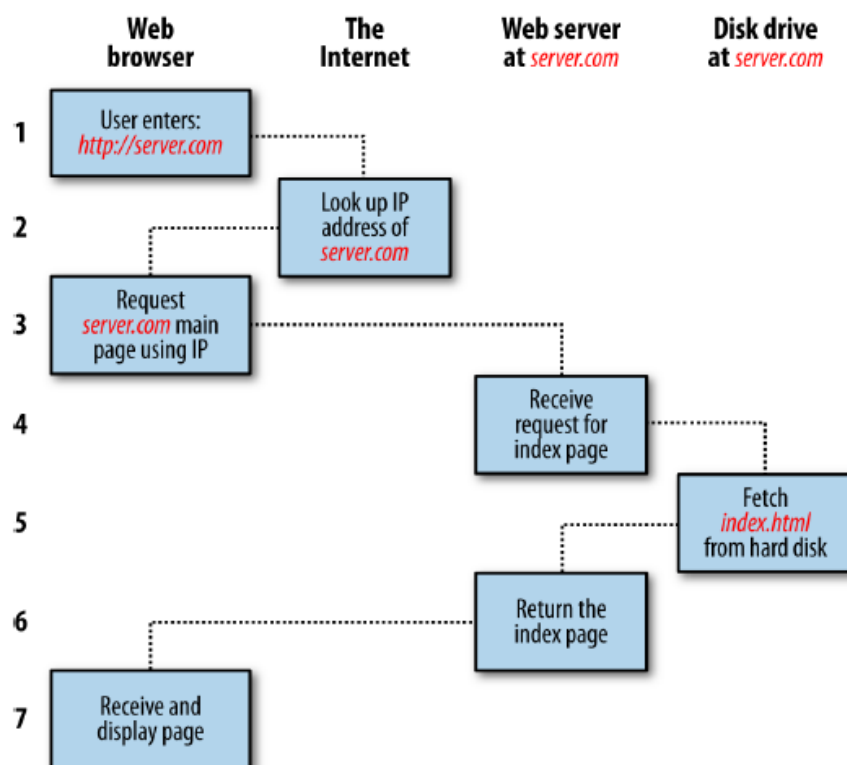
A URL is a type of uniform resource identifier (URI). In common practice, the term URI isn't used, or is used synonymously with URL, even though this is technically incorrect. Every resource has its own unique address. [4]

### 1.3.7 The Communication Procedure:

At its most basic level, the communication process consists of a web browser asking the web server to send it a web page and the server sending back the page. The browser then takes care of displaying the page.

#### 1.3.7.1 Static web pages processing:

A static website consists of a set of HTML pages and files that are hosted on a computer that is running a Web server. The page request is generated when it clicks a page in a Web, a bookmark in a browser, or enters the URL in a URL text box for a browser. The final content of a static web page can be determined by the page designer and does not change when the page is requested.



**Figure 1. 3 A static client/server communication sequence**

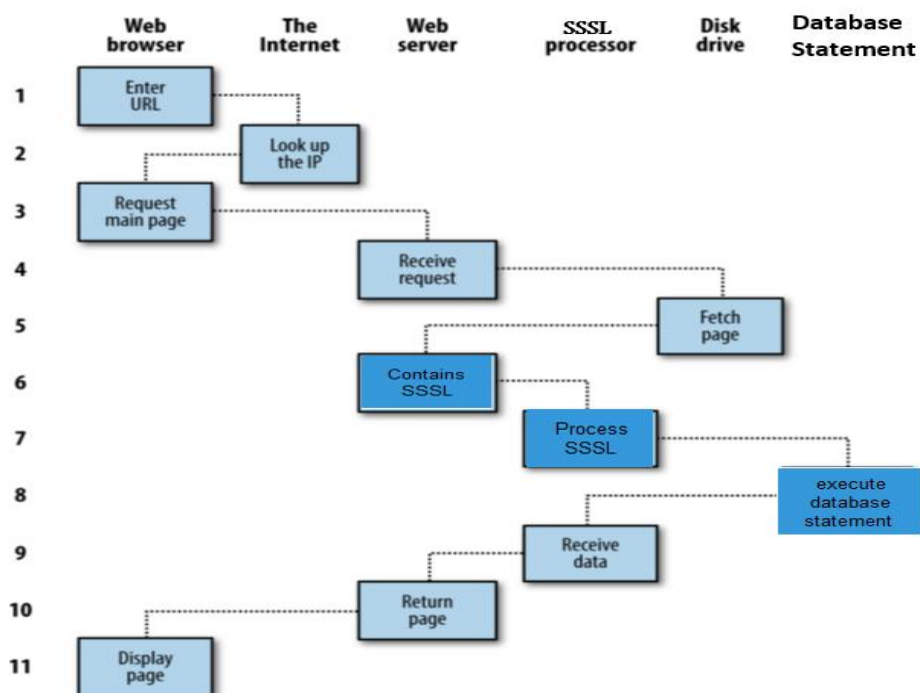
Each step in the request and response sequence is as follows:

1. You enter *http://server.com* into your browser's address bar.
2. Your browser looks up the IP address for *server.com*.
3. Your browser issues a request for the home page at *server.com*.
4. The request crosses the Internet and arrives at the *server.com* web server.
5. The web server, having received the request, looks for the web page on its disk.
6. The web page is retrieved by the server and returned to the browser.
7. Your browser displays the web page.

In step 2, notice that the browser looked up the IP address of *server.com*. Every machine attached to the Internet has an IP address—your computer included. But generally, we access web servers by name, such as *google.com*. As you probably know, the browser consults an additional Internet service called the Domain Name Service (DNS) to find its associated IP address and then uses it to communicate with the computer.

### 1.3.7.2 Dynamic web page processing:

when the web server receives a request for a dynamic web page, the procedure is a little more involved, because it may bring both a server-side scripting language and a database into the mix. Here is a representation of the process:



**Figure 1. 4 A dynamic client/server communication sequence**

1. You enter *http://server.com* into your browser's address bar.
2. Your browser looks up the IP address for *server.com*.
3. Your browser issues a request to that address for the web server's home page.
4. The request crosses the Internet and arrives at the *server.com* web server.
5. The web server, having received the request, fetches the home page from its hard disk.
6. With the home page now in memory, the web server notices that it is a file incorporating server-side scripting language and passes the page to an interpreter.
7. The interpreter executes the scripting code.
8. Some of the scripting code contains a database statements, which the interpreter now passes to the database engine.
9. The database returns the results of the statements to the interpreter.
10. The interpreter returns the results of the executed scripting code, along with the results from the database, to the web server.
11. The web server returns the page to the requesting client, which displays it.

#### **1.4 Conclusion:**

Through this chapter, we specified the problems that the council's members may face and we proposed to help solving them using a web application. In the second part of the chapter we gave a background summary of a web application and some generalities related to running the web.

## ***CHAPTER TWO***

---

### ***Overview of the Scientific Council***

### 2.1 Introduction:

This chapter details a general overview about the scientific council, it provides definition about the scientific council of the Institute, talks about its mission and it also introduces the members of its staff and their respective roles.

### 2.2 Scientific Council presentation: [5]

The Scientific Council of the Institute (SCI) is an advisory body of the Institute that issues advice and recommendations on all aspects related to scientific research and graduate and post-graduate teaching.

The Scientific Council of the Institute includes the following members (Article 67):

- The Chairman of the SCI;
- The Director of the Institute;
- Deputy Directors;
- Heads of Departments;
- Laboratory Director(s);
- The head of the Institute's library;
- Two representatives of the Institute faculty members;
- Two representatives per department for higher-ranking teaching staff (professors or lecturers).

The Chairman of the SCI is elected among the teachers' representatives with the highest ranks for a three-year term renewable once. The list of the members of the scientific council members is laid down by Order of the Minister in charge of higher education. The Scientific Council of the Institute is responsible for issuing advice and recommendations on (Article 68):

- Organization of curricula content;
- Organization of research work;
- Providing proposals for research programs;
- Providing proposals for the creation or removal of departments and/or branches as well as research units and laboratories;
- Providing proposals for the opening, renewal and/or closing of post-graduation programs and the number of vacancies to be filled;
- Monitoring teacher profiles and needs.

## Chapter 2: Overview of the scientific council

It is also in charge of:

- Approving the research themes proposed by post-graduation students and suggest relevant defense juries;
- Proposing university habilitation juries,
- Examining the reports of the Institute pedagogic and scientific activities that, along with the advice and recommendations, are transmitted from the council to the attention of the rector;
- It may be called upon to consider any other educational or scientific question submitted to it by the Director.

The Scientific Council meets in ordinary session once every three (3) months when convened by its Chairman. It may meet in extraordinary session upon the request of its chairman, two-thirds of its members or the director of the Institute (Article 69).

In our work we deal with the following points:

- Approving the research themes proposed by post-graduation students and suggest relevant defense juries
- proposing university habilitation juries,
- Organization of research work;
- Providing proposals for research programs;
- Monitoring teacher profiles and needs.

### 2.3 Staff role identification:

In our work we deal only with the members' roles directly related to the above description of the SCI.

#### 2.3.1 SCI President:

According to the above description of the SCI, the president's roles can be identified as follows:

- **Open submissions:** the president is the one who set a date for the start and end of the submissions, also he should fix a date for the council meeting.
- **Set meeting agenda:** the president has to set a schedule for the meeting.
- **Review applications:** all SCI members including the president should review every submitted application and provide a decision on it.



## Chapter 2: Overview of the scientific council

- **Final decision:** Based on the decisions of all SCI members and after discussion, the president should give the final decision about every submitted application (accepted or rejected)

### 2.3.2 SCI Member:

- **Review applications:** all SCI members including the president should review every submitted application and provide a decision on it.

### 2.3.3 Faculty Member / PhD Student:

All teachers and PhD students are able to submit an application of different types (defense, research, conference...) to be reviewed by the SCI members.

### 2.4 Conclusion:

In this chapter, we have given an overview of the scientific council of the institute. We started by a small presentation of the council, and we finished by identifying the roles of each member of the staff.

# ***CHAPTER THREE***

---

## ***Tools and technologies***

### **3.1 Introduction:**

With the introduction of many popular tools and technologies, a modern web application has really come a long way over the years leading to develop a fully responsive and dynamic web app. In this chapter we define tools and technologies that we are going to use in our project.

### **3.2 Development tools:**

#### **3.2.1 EDraw Max:**

EDraw Max is a comprehensive software program that includes a number of ready-made graphics that facilitate the creation of maps and organizational charts, network charts, business presentations, and even fashion designs and drawings, as well as structures Software, and used by professionals in web design.

#### **3.2.2 Visual Studio Code:**

Visual Studio Code is a source editor developed by Microsoft for Windows, Linux and MacOS. Visual Studio Code has out-of-the-box support for almost every major programming language. Several are included by default, for example, JavaScript, TypeScript, CSS, and HTML but other language extensions can be found and downloaded for free from the VS Code Marketplace.

#### **3.2.3 NodeJS:**

NodeJS is an open source development platform used for developing a server-based application. The traditional JavaScript environment has always been a client-side in a user's browser or in an application that is talking to a server. JavaScript has not been considered when it comes to the server responding to client requests, but that is exactly what node.js provides.

Node.js is not written in JavaScript. It is written in C++ but it uses the JavaScript language as an interpretive language for server-side request/response processing. NodeJS can work with both relational (such as Oracle and MYSQL Server) and non-relational databases (such as MongoDB). [6]

### 3.2.4 ExpressJS:

Express.js is a NodeJS web application server framework, which is specifically designed for building single-page, multi-page, and hybrid web applications. It has become the standard server framework for NodeJS. [7]

The Express.js framework makes it very easy to develop an application which can be used to handle multiple types of requests like the GET, PUT, POST and DELETE requests. And it has the ability to work with databases which are commonly required by most modern-day web applications.

### 3.2.5 ReactJS:

ReactJS basically is an open-source JavaScript library which is used for building user interfaces specifically for single page applications. It's used for handling view layer for web and mobile apps. React also allows us to create reusable UI components.

React allows developers to create large web applications which can change data, without reloading the page. The main purpose of React is to be fast, scalable, and simple. [8]

### 3.2.6 GraphQL:

GraphQL is a query language for API, and a server-side runtime for executing queries by using a type system you define for your data. GraphQL isn't tied to any specific database or storage engine and is instead backed by your existing code and data. A GraphQL service is created by defining types and fields on those types, then providing functions for each field on each type. It is a syntax that describes how to ask for data and is generally used to load data from a server to a client.

GraphQL has three main characteristics:

- It lets the client specify exactly what data it needs.
- It makes it easier to aggregate data from multiple sources.
- It uses a type system to describe data. [9]

#### 3.2.6.1 GraphQL schema:

A GraphQL schema is at the center of any GraphQL server implementation and describes the functionality available to the clients which connect to it.

The core building block within a schema is the "type". Types provide a wide-range of functionality within a schema, including the ability to:

- Create relationships between types.
- Define which data-fetching (querying) and data-manipulation (mutating) operations can be executed by the client.
- If desired, self-explain what capabilities are available to a client via introspection [10].

### 3.2.6.2 Schema Definition Language:

To make it easy to understand the capabilities of a server, GraphQL implements a human-readable schema syntax known as its Schema Definition Language, or "SDL". The SDL is used to express the types available within a schema and how those types relate to each other.

In a simple example involving books and authors, the SDL might declare:

```

1  type Book {
2    title: String
3    author: Author
4  }
5
6  type Author {
7    name: String
8    books: [Book]
9  }
```

It's important to note that these declarations express the *relationships* and the *shape* of the data to return, not where the data comes from or how it might be stored - which will be covered outside the SDL. [10]

### 3.2.6.3 The Query types:

A GraphQL query is for fetching data and compares to the GET verb in REST-based APIs. In order to define what queries are possible on a server, the Query type is used within the SDL. The Query type is one of many root-level types which defines functionality (it doesn't actually trigger a query) for clients and acts as an entry-point to other more specific types within the schema. [10]

### 3.2.6.4 The Mutation types:

Mutations are operations sent to the server to create, update or delete data. These are comparable to the PUT, POST and DELETE verbs on REST-based APIs.

Much like how the Query type defines the entry-points for data-fetching operations on a GraphQL server, the root-level Mutation type specifies the entry points for data-manipulation operations. [10]

### 3.2.6.5 Resolvers:

In order to respond to queries or mutations, a schema needs to have a resolve functions for all fields. This collection of functions is called the "resolver map". This map relates the schema fields and types to a function.

Resolvers provide the instructions for turning a GraphQL operation into data [11]

### 3.2.7 Apollo GraphQL:

- **Apollo Server**

The fastest way to get started with GraphQL is by creating a new server. Apollo Server will set an Express server up for you as long as you provide it with typeDefs, which is a string representing your GraphQL schema, and resolvers, which is a map of functions that implement your schema. [12]

- **Apollo Client**

A sophisticated GraphQL client that manages data and state in an application. Among other benefits, it enables a declarative programming style that lets developers define queries as part of UI components; the client manages all the hairy details of binding query results to the UI, managing consistency, caching, and so on. Apollo Client also supports an exceptionally elegant approach to state management by extending the GraphQL schema inside the client with additional structure. [13]

### 3.2.4 MongoDB:

MongoDB is a powerful, flexible, and scalable general-purpose database management system (DBMS) that uses a document-oriented database model which supports various forms of data. A database is a collection of information that is organized so that it can be easily accessed, managed and updated. Instead of using tables and rows as in relational\_databases, the MongoDB architecture is made up of collections and documents. [14]

### 3.2.8 Web Browser:

A web browser, or simply "browser," is an application used to access and view websites. Common web browsers include Internet Google Chrome, Mozilla Firefox, and Apple Safari. The primary function of a web browser is to render HTML, the code used to design or "markup" webpages. Each time a browser loads a web page, it processes the HTML, which may include text, links, and references to images and other items, such as cascading style sheets and JavaScript functions. The browser processes these items, then renders them in the browser window. [15]

### **3.3 Programming languages:**

#### **3.3.1 HTML:**

Stands for “Hypertext markup” language, it is the major markup language used to display Web pages on the Internet. In other words, Web pages are composed of HTML, which is used to display text, images or other resources through a Web browser.

HyperText refers to the hyperlinks that html page may contain, and “markup language” refers to the way tags are used to define the page layout and elements within the page. [16]

#### **3.3.2 CSS:**

Stands for Cascading Style Sheets, it is a standard (or language) that describes the formatting of markup language pages. CSS enables developers to separate content and visual elements for greater page control and flexibility. A CSS file is normally attached to an HTML file by means of a link in the HTML file. [17]

#### **3.3.3 JavaScript:**

JavaScript (JS) is a scripting language, primarily used on the Web. It is used to enhance HTML pages and is commonly found embedded in HTML code. JavaScript is an interpreted language. Thus, it doesn't need to be compiled.

JavaScript renders web pages in an interactive and dynamic fashion. This allowing the pages to react to events, exhibit special effects, accept variable text, validate data, create cookies, detect a user’s browser, etc. [18]

### **3.4 Conclusion:**

Through this chapter, we gave a simple definition about the tools and the technologies that we used in order to implement our interactive, dynamic and responsive web application.

# ***CHAPTER FOUR***

---

## ***Design***



## 4.1 Introduction:

Web design usually refers to the user experience aspects of web app development rather than software development. A web designer works on the appearance, layout, and, in some cases, content of a web app. We are going to define the roles of each actor that interacts with the system. In addition, we will use UML for modeling and in particular we choose the use case diagram to model these roles. We end the chapter by introducing the database and some of its models.

## 4.2 Design Requirements:

### 4.2.1 Modelling language:

A modelling language is mainly used in the field of computer science and engineering for designing models of new software, systems, devices and equipment. Unified modelling language (UML) is a popular modelling language that is used to build system and object models graphically. [19]

### 4.2.2 Unified Modelling Language (UML):

The Unified Modelling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artefacts of a software-intensive system. It offers a standard way to write a system's blueprints, including conceptual things such as business processes and system functions as well as concrete things such as programming language statements, database schemas, and reusable software components. [28]

The current UML standards call for different types of diagrams. These diagrams are organized into two distinct groups: structural diagrams and behavioral or interaction diagrams, which are as follows:

**Structural UML diagrams:** A type of diagram that depicts the elements of a specification that are irrespective of time. This includes class, composite structure, component, deployment, object, and package diagrams. [20]

**Behavioral UML diagrams:** A type of diagram that depicts behavioral features of a system or business process. This includes activity, Sequence, Use case, State, Communication, Interaction, and Timing diagrams. [20]

## 4.3 Modelling

### 4.3.1 System Actors:

An actor is a person, organization, or an external system that plays a role in one or more interactions with our system (actors are typically drawn as stick figures on UML Use Case diagrams). [20]

We have Two types of actors:

#### 4.3.1.1 Admin:

The only one who have the right to access to the database to manage users (add, update and remove users).

#### 4.3.1.2 User:

A user is the one who will interact with our system. Users can differ from each other according to one of the roles that the admin defines for each one of them. we have four types of roles: SC\_President role, SC\_Member role, Faculty\_Member role and PhD\_Student role.

According to these roles we can have four types of users:

- **SC President user:**

This user can have three types of roles: the SC\_President, Faculty\_Member and the SC\_Member roles.

- **SC Member user:**

This user also can have two types of roles: the SC\_Member role and the Faculty\_Member role.

- **Faculty Member user:**

This user has only one role which is Faculty\_Member role.

- **PhD Student user:**

This user has only one role which is PhD\_Student role.

### 4.3.2 Use Cases:

A use case is a list of actions or event steps typically defining the interactions between an actor and a system to achieve a goal.

### 4.3.2.1 Actors Use Cases Description

In our application, we define the use cases for the actors as shown in Table 4.1.

**Table 4.2 use cases of each actor according to their roles**

Actors	Role	Use cases
Admin		<b>Manage users:</b> adds, edits and removes users
SC President SC Member Faculty member PhD Student		<b>1.Authentication:</b> accessing the application using a valid email and password
SC President	SC_President	<b>1.Open submissions:</b> set a duration for the submissions and a date for the meeting. <b>2.Set meeting agenda:</b> set a program or a schedule for the meeting. <b>3.Review application:</b> see all members reviews, comments and decisions about an application. <b>4.final decision:</b> give a final decision according to the members votes (accepted or rejected). <b>5.statistics:</b> the user can see the status of all submitted applications (accepted, rejected, department...) <b>5. Add Announcement:</b> the user can announce some statements to all users.
	SC_Member	<b>Review application:</b> check the application, comments and votes on it
	Faculty_Member	<b>Submissions:</b> submit an application
SC Member	SC_Member	<b>Review application:</b> check the application, comments and votes on it
	Faculty_Member	<b>Submissions:</b> submit an application
Faculty member	Faculty_Member	<b>Submissions:</b> submit an application
PhD Student	PhD_Student	<b>Submissions:</b> submit an application

#### 4.3.2.2 Use Cases Identification:

- a. Authentication:** Login before accessing the application. Moreover, authentication insures the identity of the user.
- b. Open submissions:** setting a start date and an ending date for the submissions is the job of the president Moreover he also should fix a date for the council meeting.
- c. Set meeting agenda:** the president has to set a schedule for the meeting.
- d. Review applications:** all members including the president should review the application and votes on it and they can also give some comments if they want.
- e. Final decision:** the president should give the final decision about an application (accepted or rejected) according to the members votes, but if there is an equality on the votes the president will decide.
- f. Submissions:** every user have the right to submit an application (internship, conference, research ....).
- g. Statistics:** allow user to see the total number and the number of accepted and refused for each type of submitted applications per year.

#### 4.3.2.3 Use Case Diagram:

**Use case diagrams** are usually referred to as behavior diagrams used to describe 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). Each use case should provide some observable and valuable result to the actors or other stakeholders of the system. [21]


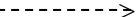
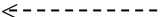

**Functional requirements:** represented as use cases; a verb describing an action, a use case is shown as an ellipse in a use case diagram

**Actors:** they interact with the system; an actor can be a human being, an organization or a system

**Relationships** between actors and use cases, represented using straight arrows derive classes and relate them to actions of the use case activities. Therefore, when we create a sequence diagram it highlights certain aspect of the whole system.

The following table 4.2 summarizes the used arrows:

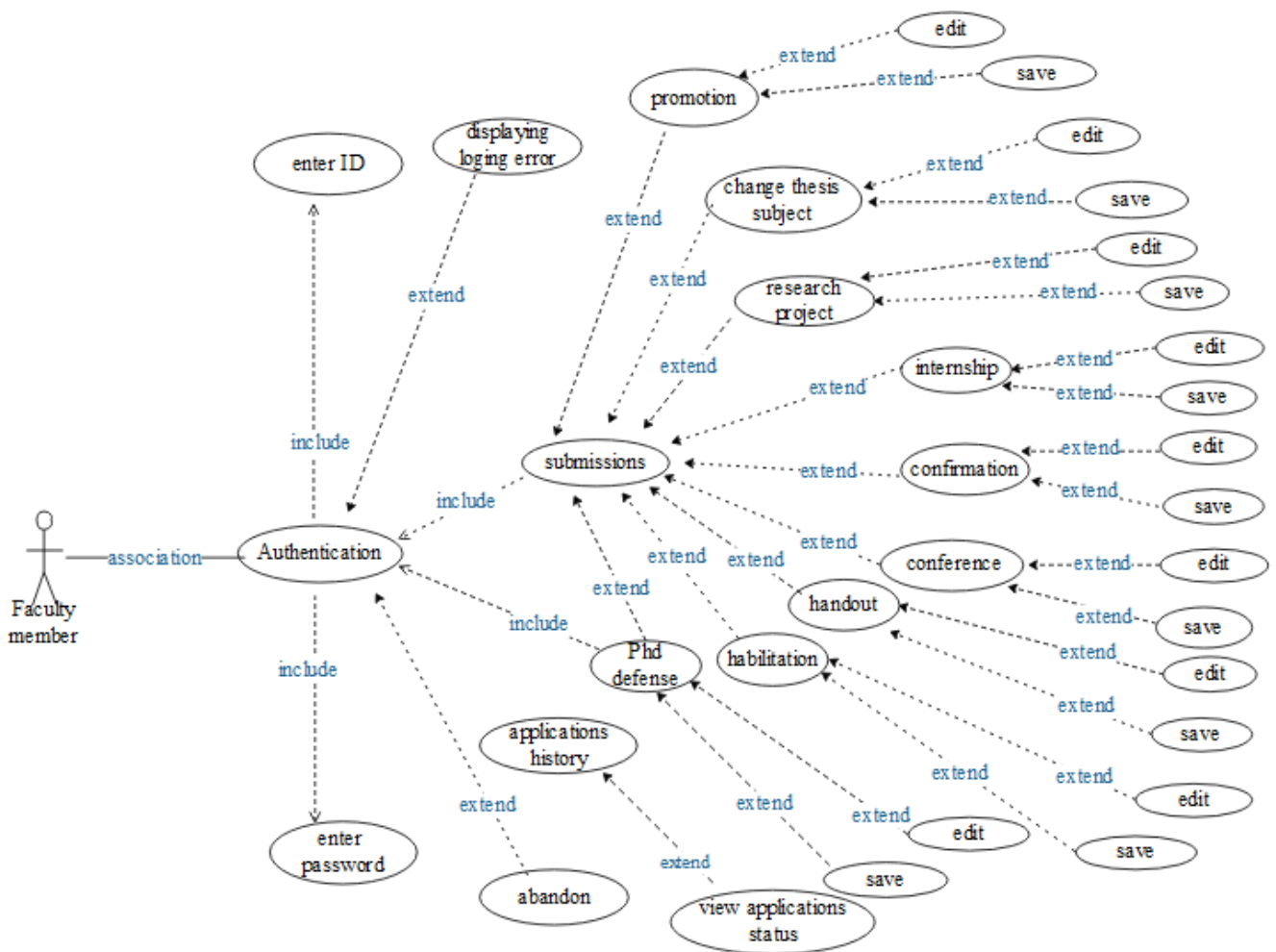
**Table 4.3 Summary of the used arrows.**

Symbols	Description
Association 	<b>Association:</b> Use cases are associated with the actors that perform them. A line is used to link actors to use cases.
Include 	<b>Include:</b> an include relationship shows dependency between a base use case and an included use case Every time the base use case is executed the included use case is executed as well, another way to think of it is that the base use case requires an included use case in order to be complete. When we have an included use case we draw a dashed line with an arrow that points towards the included use case.
Extend 	<b>Extend:</b> it has also a base use case and an extend use case when the base use case is executed the extend use case will happen sometimes but not every time, the extended use case will only happen if certain criteria are met, another way to think of it is that you have the option to extend the behavior of the base use case, when we have an extended use case we draw a dashed line with an arrow that points towards the base use case.
Generalization 	<b>Generalization</b> relationship is also a parent-child relationship between use cases. The child use case has the underlying business process meaning but is an enhancement of the parent use case. The child use case is connected at the base of the arrow. The tip of the arrow is connected to the parent use case.



- **The faculty Member Use Case Diagram:**

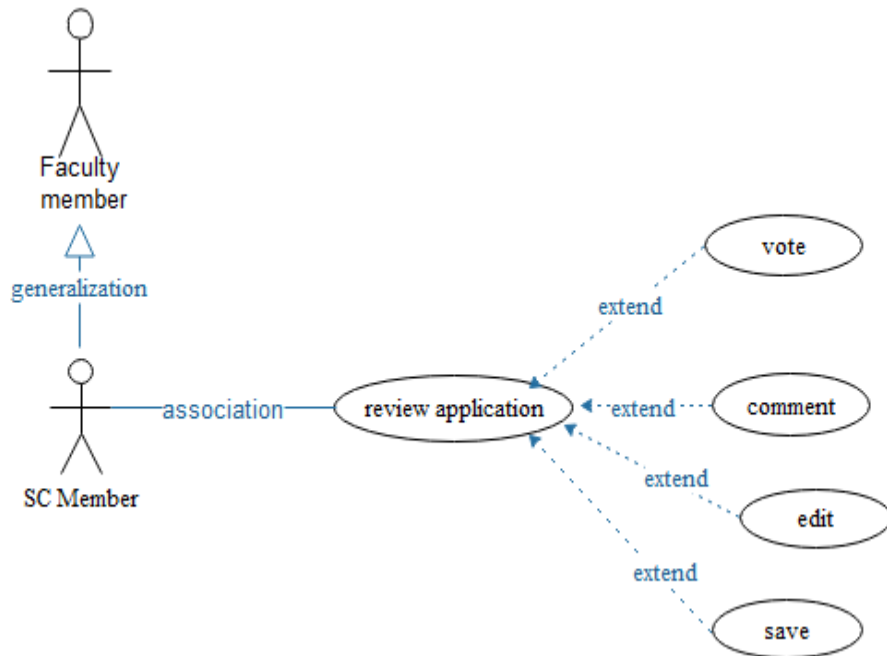
the faculty member user has one role which is Faculty\_Member.



**Figure 4. 2 Faculty Member Use Case Diagram**

- **The SC Member Use case diagram:**

Since the SC Member has the role of Faculty\_Member which makes him in a generalization relationship with the Faculty member user. Moreover, he has the role of SC\_Member.

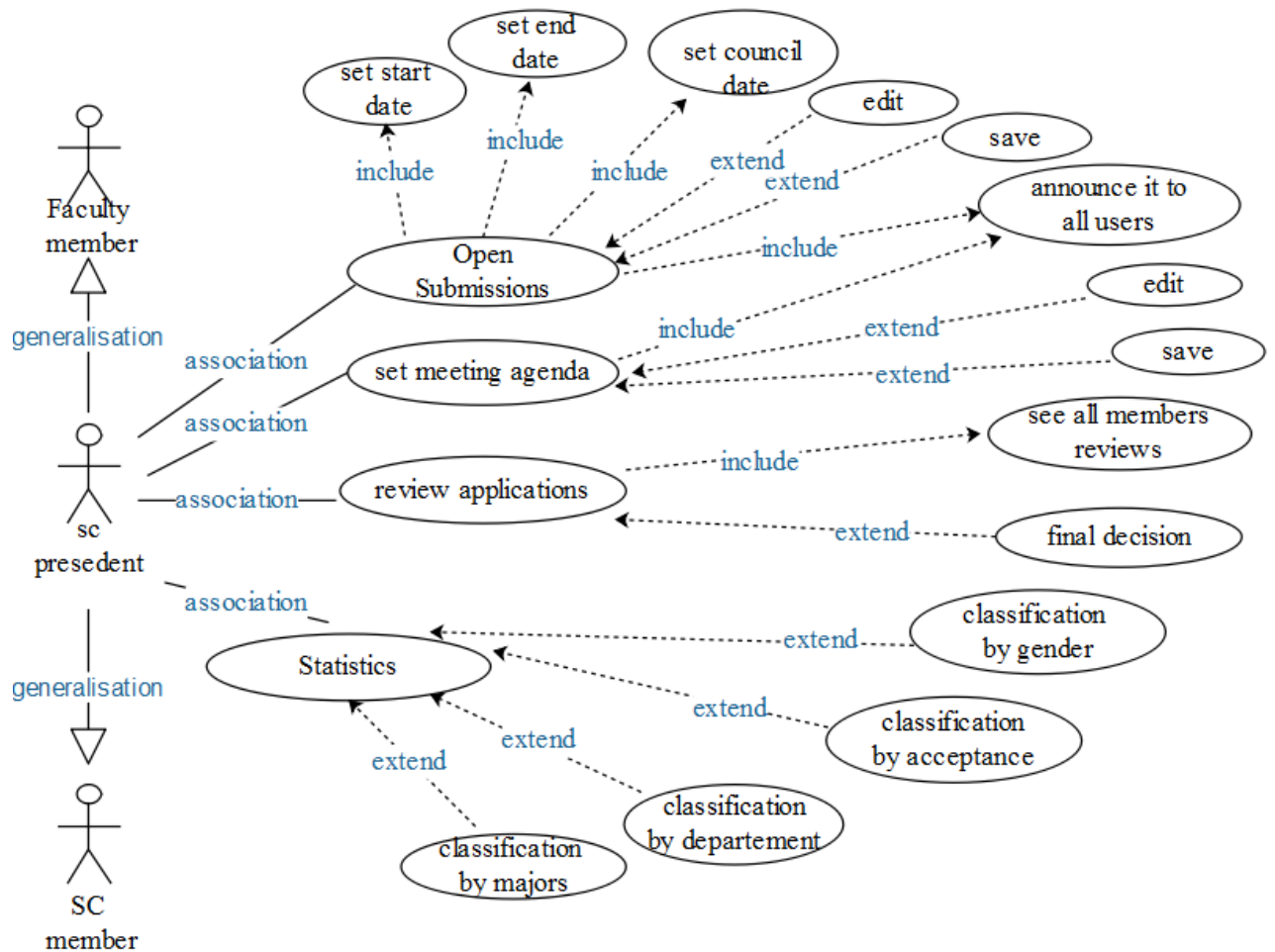


**Figure 4. 3 SC Member Use Case Diagram**



- **The SC President Use case diagram:**

Since the president has the role of Faculty\_Member and SC\_Member he has a generalization relationship with the Faculty Member and the SC\_Member users. Moreover, the president has the role of the SC\_President which adds more use cases to him.



**Figure 4. 4 SC President Use Case Diagram**

#### 4.3.2.4 Textual description of use cases:

##### Use case 01: Authentication

**Table 4. 4 Textual description for use case "Authentication"**

Use case name	Authentication
Actor	Faculty Member, SC President, SC Member, PhD Student
Objective	Authenticate to have access to the application
Precondition	Browser and access to internet
Scenario	<ol style="list-style-type: none"> <li>1. The user lunches the application via a browser.</li> <li>2. The system asks for username and password.</li> <li>3. The user enters his name and password.</li> <li>4. The system checks the conformity of the information entered by sending an authentication query to the server.</li> <li>5. The server verifies the query and send favorable answer.</li> <li>6. The user accesses the application</li> </ol>
Alternative	If the username or password is wrong or missed the system displays an error message or incomplete field (return to 2)

##### Use case 02: Open submissions

**Table 4.5 Textual description for use case " Open submissions "**

Use case name	Open submissions
Actor	SC President
Objective	Set submission duration and meeting date
Precondition	Authentication
Scenario	<ol style="list-style-type: none"> <li>1. The user accesses to Open submissions interface.</li> <li>2. The user enters the starting date and the ending date for the submissions and a meeting date in the date bar and saves.</li> <li>3. The system sends a query to the server for processing.</li> <li>4. The system announces these dates to all users.</li> </ol>
Alternative	<p>Submission start date must not be a past date;          Submission start date must be two months after start date;          Meeting date must be one week after end date.          (return to 2)</p>

**Use case 03: Set meeting agenda****Table 4. 6 Textual description for use case " Set meeting agenda "**

Use case name	Set meeting agenda
Actor	SC President
Objective	Set a program for the council meeting
Precondition	Authentication
Scenario	<ol style="list-style-type: none"> <li>1. The user accesses to SC president interface.</li> <li>2. The user enters a program for each day of the council meeting in the table of days and saves.</li> <li>3. The system sends a query to the server for processing.</li> <li>4. The system sends this data to the SC member interface.</li> </ol>
Alternative	The system displays an error message or missed field (return to 2)

**Use case 04: Review applications****Table 4. 7 Textual description for use case " Review applications "**

Use case name	Review applications
Actor	SC President / SC members
Objective	Review and vote for a submitted application
Precondition	Authentication
Scenario	<ol style="list-style-type: none"> <li>1. The user access to Review applications interface.</li> <li>2. The user checks and reads the submitted application, votes on it, comments (optionally) and clicks save.</li> <li>3. The system updates Review applications interface for the different users.</li> </ol>
Alternative	Connection failed or technical problem (Refresh the page / verify internet connection)

**Use case 05: Final Decision****Table 4. 8 Textual description for use case " Final decision "**

Use case name	Final Decision
Actor	SC President
Objective	A final decision for an application (accepted or rejected)
Precondition	Review applications
Scenario	<ol style="list-style-type: none"> <li>1. The user checks the statistic table and see all the members reviews and their votes.</li> <li>2. The user accept or reject the application according to the members votes.</li> <li>3. The user send a response to the applicant.</li> </ol>
Alternative	Connection failed or technical problem (Refresh the page / verify internet connection)

**Use case 06: Submissions****Table 4. 9 Textual description for use case " Submissions"**

Use case name	Submissions
Actor	Faculty member / PhD Student
Objective	Submit an application.
Precondition	Authentication
Scenario	<ol style="list-style-type: none"> <li>1. The user enters the submissions interface</li> <li>2. The user selects an application.</li> <li>3. The user fills the requested information and saves.</li> <li>4. The system sends a query to the server for processing.</li> </ol>
Alternative	The system displays an error message or missed field (return to 3)

**Use case 07: Statistics****Table 4. 10 Textual description for use case " Statistics "**

Use case name	Statistics
Actor	SC President
Objective	See status of submitted applications per year.
Precondition	Authentication
Scenario	<ol style="list-style-type: none"> <li>1. The user enters the Statistics interface</li> <li>2. The user select an application</li> <li>3. The system sends a query to the server for processing</li> <li>4. The server send a pie chart and the system display it.</li> </ol>
Alternative	Connection failed or technical problem (Refresh the page / verify internet connection)

**4.3.3 Relationships between documents:**

Relationships in MongoDB represent how various documents are logically related to each other. Relationships can be modeled via Embedded and Referenced approaches. Such relationships can be either 1:1, 1: N, N:1 or N: M.

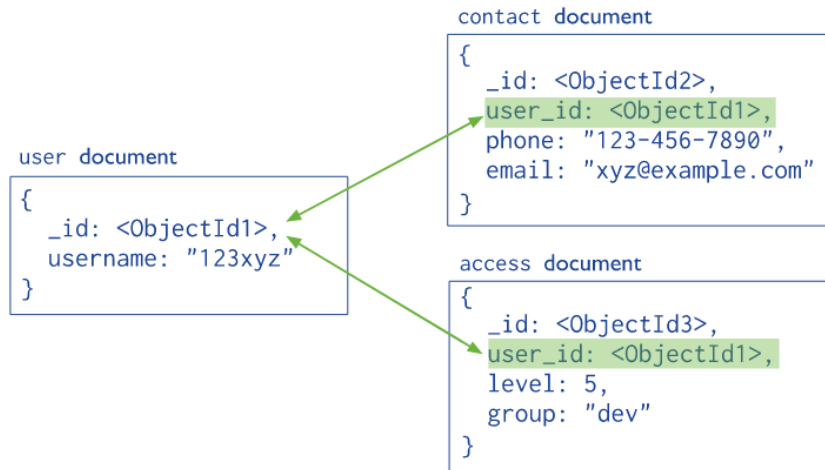
**4.3.3.1 Embedded Relationships**

Embedded documents capture relationships between data by storing related data in a single document structure. MongoDB documents make it possible to embed document structures in a field or array within a document. These allow applications to retrieve and manipulate related data in a single database operation.

**Figure 4. 3 Embedded data relationship**

### 4.3.3.2 Referenced Relationships

References store the relationships between data by including links or references from one document to another. Applications can resolve these references to access the related data.



**Figure 4. 4 References relationship**

The following table (Table 4.10) summarizes the used arrows:

**Table 4. 11 used arrows**

Symbol	Description
	One to many: one document in a collection can be associated with one or more documents in another collection
	One to one: one document in a collection is associated with one and only one document in another collection

Relationships diagram of our system is given in figure 4.5.

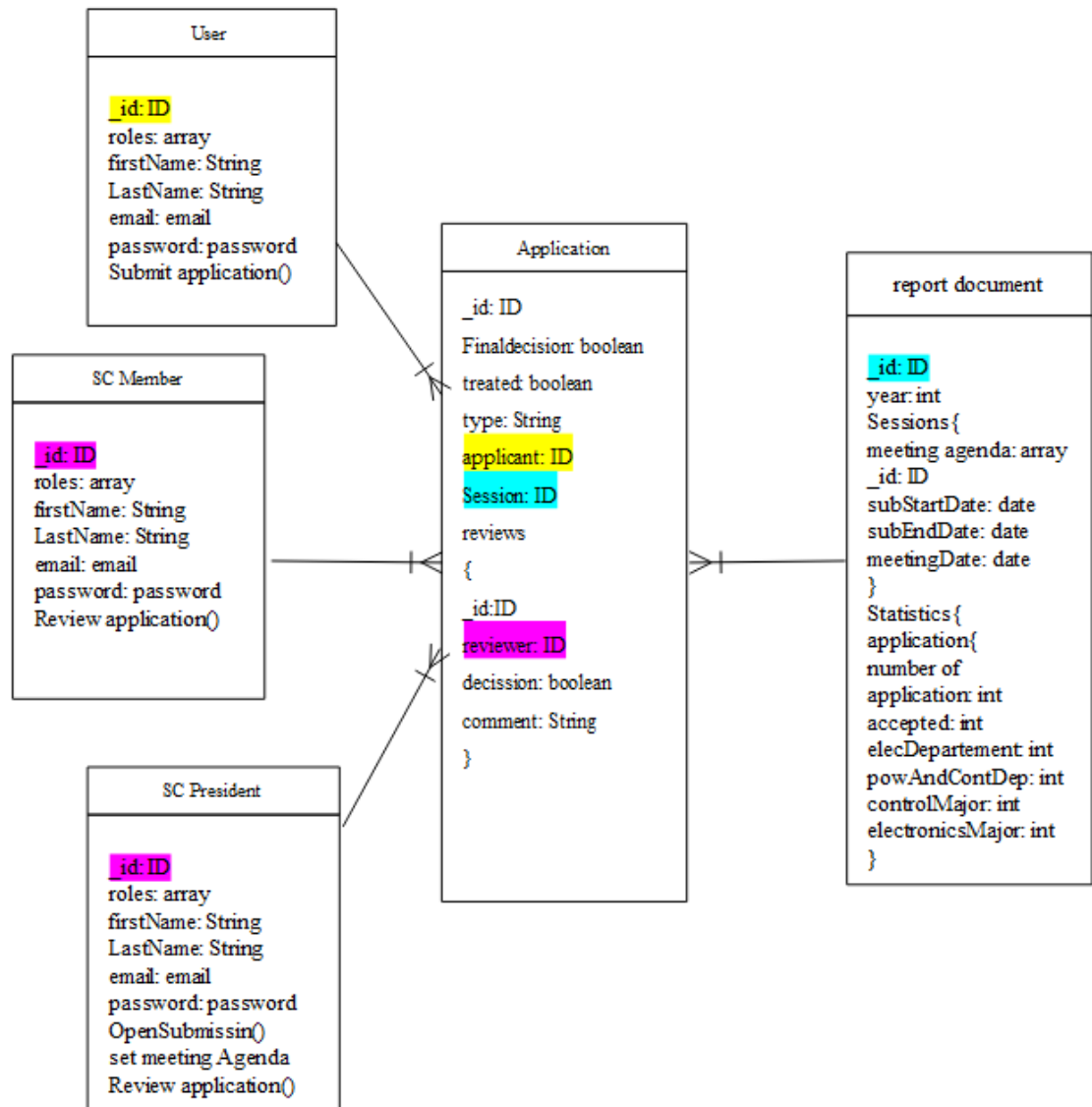




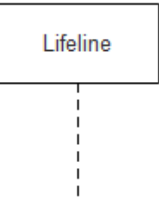

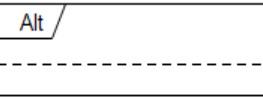

Figure 4. 5 Relationships between documents.

### 4.3.8 Sequence Diagrams:

Sequence diagram is the most common kind of interaction diagram, which focuses on the message interchange between a number of lifelines. Sequence diagram describes an interaction by focusing on the sequence of messages that are exchanged, along with their corresponding occurrence specifications on the lifelines. [22].

Sequence diagrams are made up of the following icons and elements:

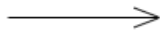
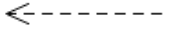
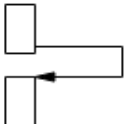
**Table 4. 12 Basic symbols used in sequence diagram**

Symbol	Name	Description
	Actor Symbol	Shows entities that interact with or are external to the system.
	Activation box	Represents the time needed for an object to complete a task. The longer the task will take, the longer the activation box becomes.
	Lifeline Symbol	Represents the passage of time as it extends downward. This dashed vertical line shows the sequential events that occur to an object during the charted process. Lifelines may begin with a labelled rectangle shape or an actor symbol.
	Option loop symbol	Defines that the calls within the fragment may execute multiple times.
	Alternative symbol	Divides fragment into groups and defines condition for each group, only the one whose condition is true will execute.
	Ref symbol	Refers to an interaction defined on another diagram. The frame is drawn to cover the lifelines involved in the interaction.



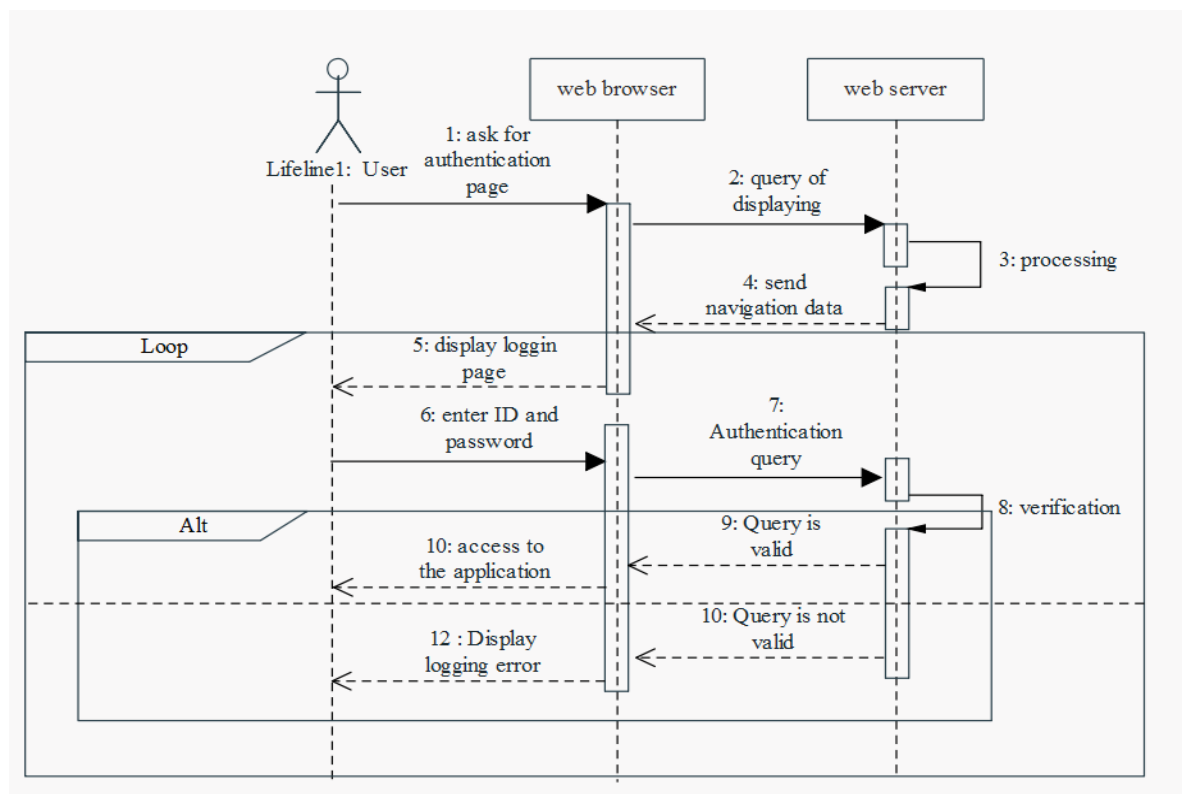
Moreover, the following arrows and message symbols to show how information is transmitted between objects.

**Table 4. 13 Message symbols used in sequence diagram**

Symbol	Name	Description
	Synchronous message Symbol	This symbol is used when a sender must wait for a response to a message before it continues.
	Reply message Symbol	These messages are replies to calls.
	Self-message Symbol	Self-message is a call message which is sent from a lifeline to itself.

### Sequence diagram 01: Authentication

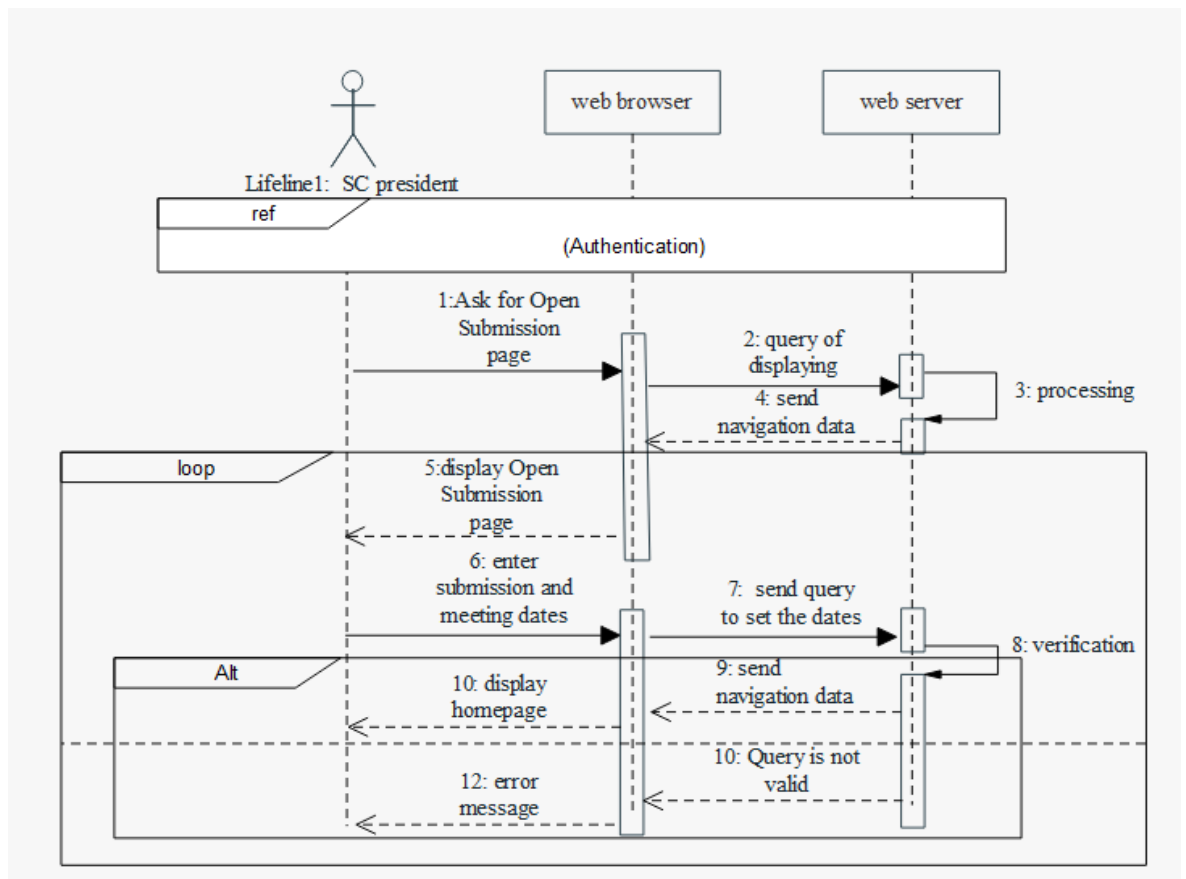
When the user wants to access to the application, he must log-in by entering his ID and password. The system sends a query to the server for checking. If this information exists in the database he can access otherwise an error message and will be displayed.



**Figure 4. 6 Sequence diagram “Authentication”**

**Sequence diagram 02: Open submissions**

After authentication, the SC President user can open the submissions, this is done by accessing the Open submissions page, enter the starting date, the ending date and the meeting date. The browser sends a query to the server for processing and saving. Returning back to homepage or error message will be displayed.



**Figure 4. 7** Sequence diagram “Open submissions”

**Sequence diagram 03: Submit application**

After authentication and opening the submissions, users can submit applications by accessing the application form page and filling the form by entering the requested data. The browser sends query to the server for processing and saving.

The submitted application information or error message will be displayed.

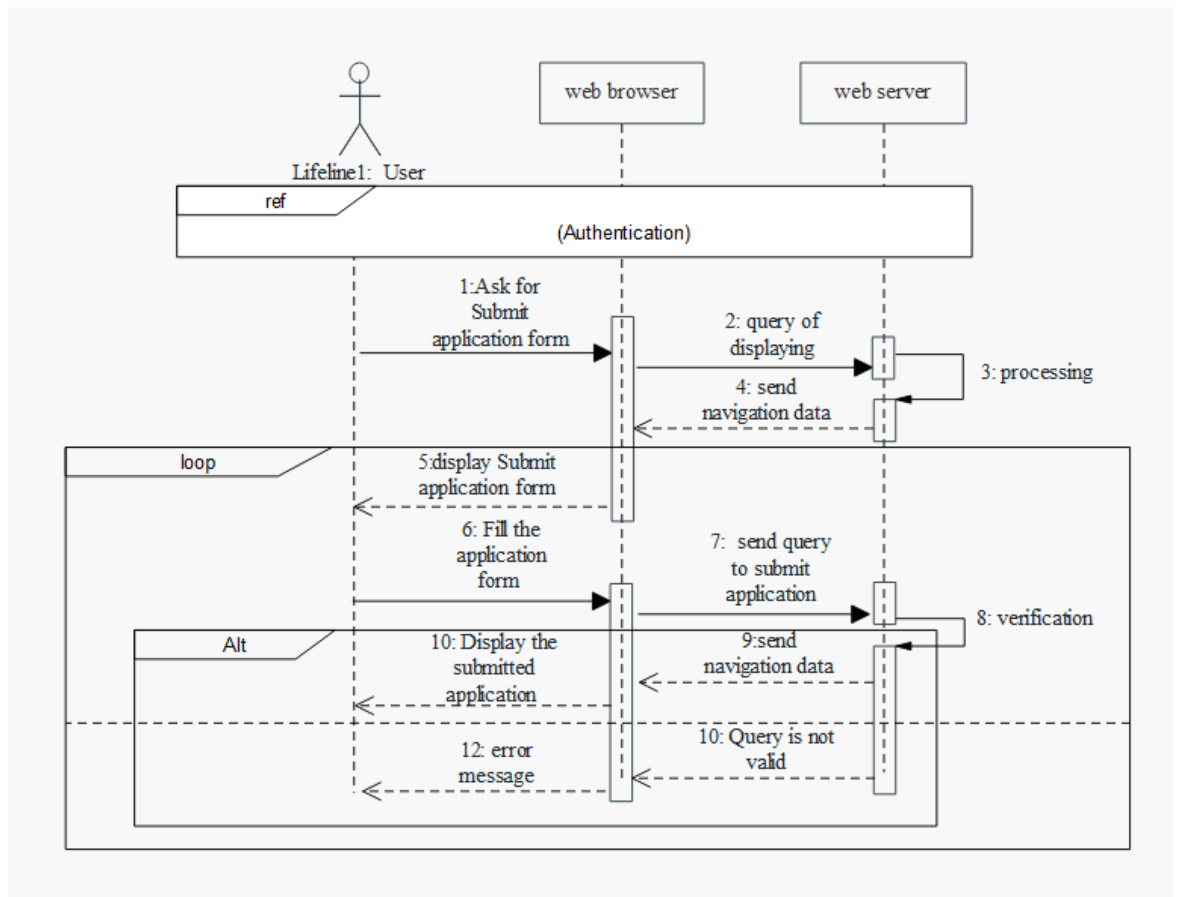
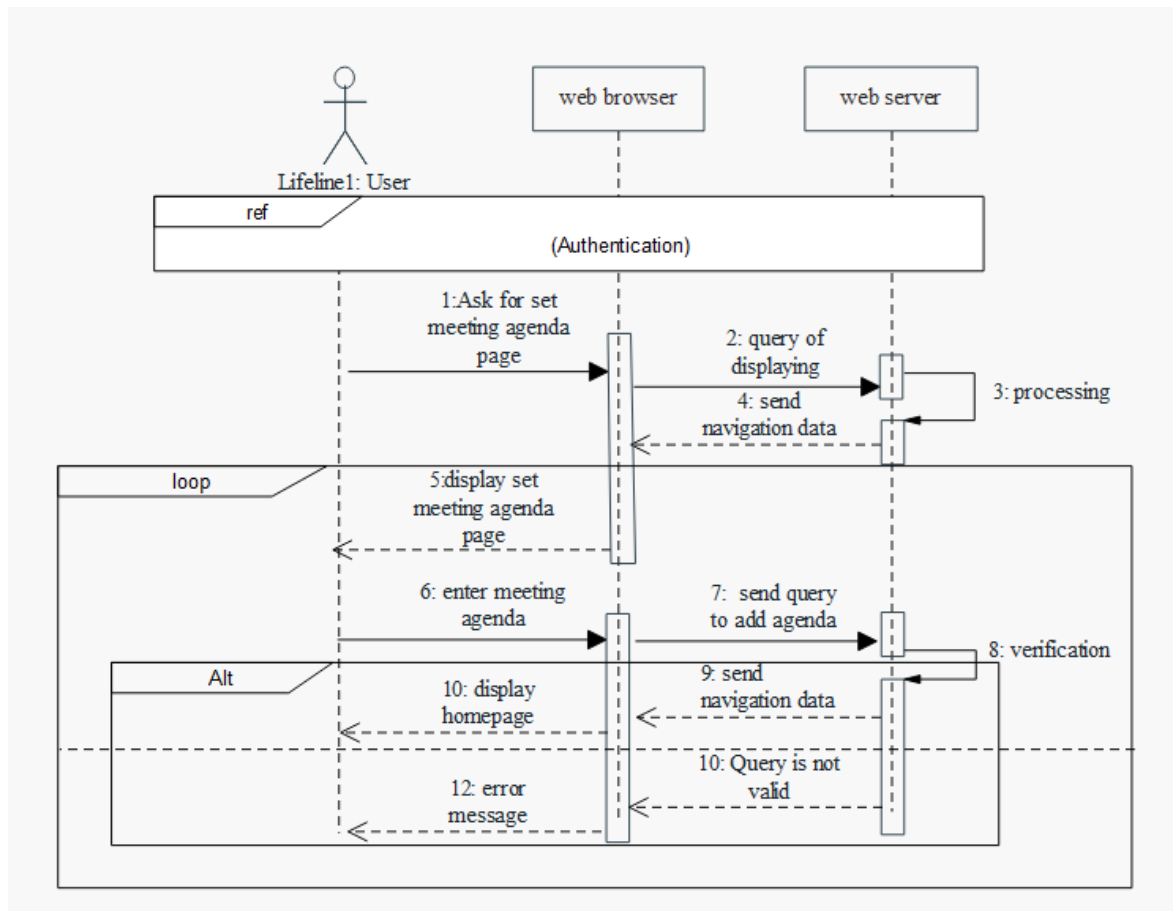


Figure 4. 8 Sequence diagram “Submit application”

**Sequence diagram 04: Set meeting agenda**

After closing the submission, the user can set an agenda for the meeting, this is done by accessing the set meeting agenda page, entering an agenda. The browser sends a query to the server for processing and saving.

Returning back to homepage or error message will be displayed.

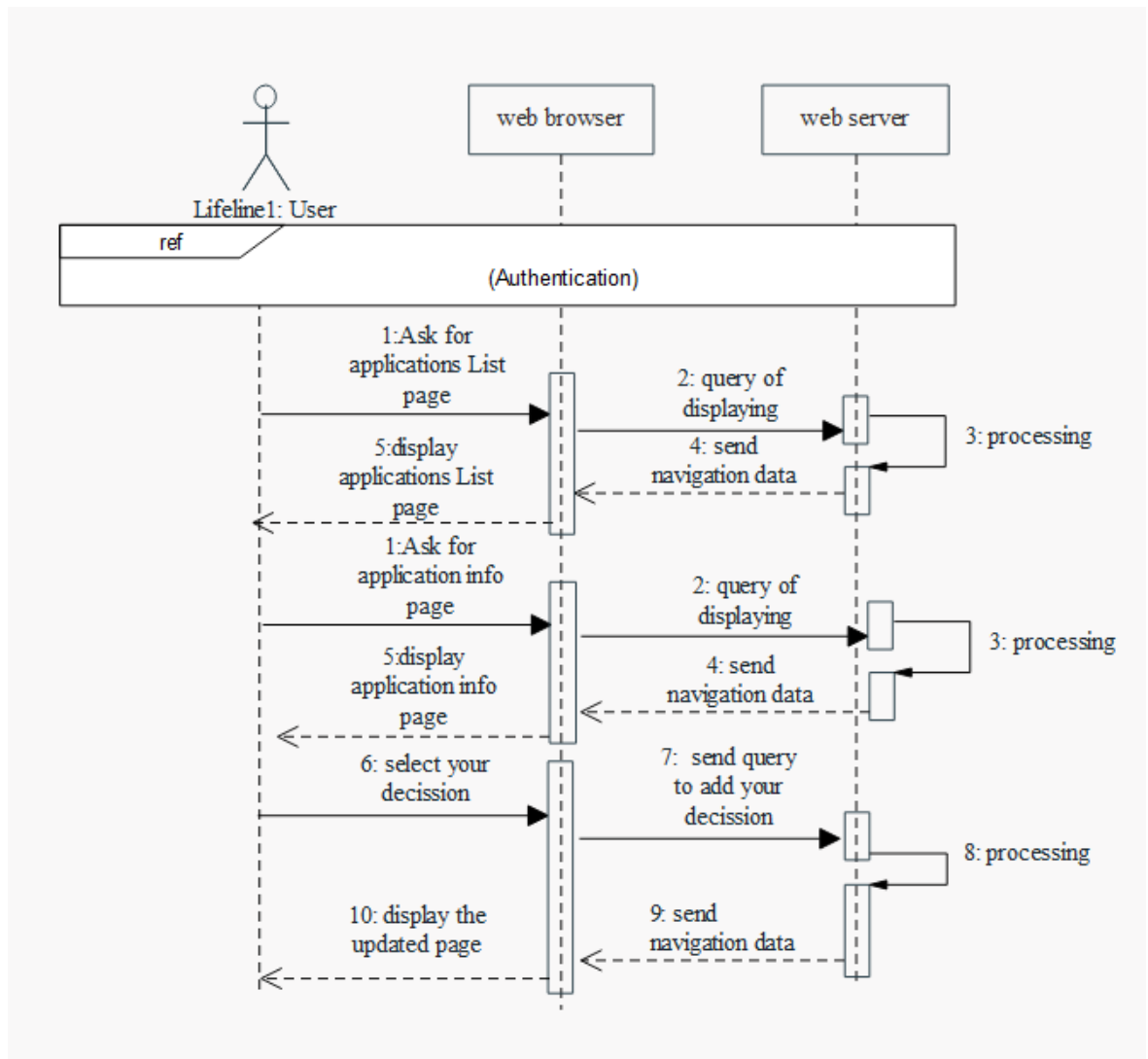


**Figure 4. 9** Sequence diagram “Set meeting agenda”

**Sequence diagram 05: review application**

After closing the submission, the user can review an application, by accessing the application list page, select an application, then select a decision. The browser sends a query to the server for processing and saving.

An updated page (same page contains the decision of the user) will be displayed.



**Figure 4. 10** Sequence diagram “Review application”

For the final decision Sequence diagram, it is the same for the Review application sequence diagram, but instead of giving a decision, the user gives a final decision.

## **4.4 Introduction to Database:**

### **4.4.1 Definition:**

Database is a collection of information that exists over a long period of time. The term database refers to a collection of data that is managed by database management system (DBMS). A DBMS is a collection of programs that enables users to create and maintain a database. The DBMS is a general-purpose software system that facilitates the processes of defining, constructing, manipulating, and sharing databases among various users and applications. [27]

### **4.4.2 Relational Model:**

A relational database is a collection of data items with pre-defined relationships between them. These items are organized as a set of tables with columns and rows. Tables are used to hold information about the objects to be represented in the database. Each column in a table holds a certain kind of data and a field stores the actual value of an attribute. The rows in the table represent a collection of related values of one object or entity. Each row in a table could be marked with a unique identifier called a primary key, and rows among multiple tables can be made related using foreign keys. [23]

### **4.4.3 Non-Relational Model:**

A non-relational database is a database that does not use the tabular schema of rows and columns found in most traditional database systems. Instead, non-relational databases use a ‘Document data stores’ model that is optimized for the specific requirements of the type of data being stored.

A document data store manages a set of named string fields and object data values in an entity referred to as a *document*. These data stores typically store data in the form of JSON documents. Each field value could be a scalar item, such as a number, or a compound element, such as a list or a parent-child collection. The data in the fields of a document can be encoded in a variety of ways, including XML, JSON, BSON, or even stored as plain text.

The fields within documents are exposed to the DBMS, enabling an application to query and filter data by using the values in these fields. [24]

#### 4.4.4 What is MySQL?

It is a popular open-source relational database management system (RDBMS) that is developed, distributed and supported by Oracle Corporation. Like other relational systems, MySQL stores data in tables and uses structured query language (SQL) for database access. In MySQL, you pre-define your database schema based on your requirements and set up rules to govern the relationships between fields in your tables. Any changes in schema necessitates a migration procedure that can take the database offline or significantly reduce application performance. [25]

#### 4.4.5 What is MongoDB?

MongoDB is a non-relational database developed by MongoDB, Inc. MongoDB stores data as documents in a binary representation called BSON (Binary JSON). Related information is stored together for fast query access through the MongoDB query language. Fields can vary from document to document; there is no need to declare the structure of documents to the system – documents are self-describing. If a new field needs to be added to a document, then the field can be created without affecting all other documents in the collection, without updating a central system catalog, and without taking the system offline. Optionally, schema validation can be used to enforce data governance controls over each collection. [Erreur ! Signet non défini.]

A NoSQL database like MongoDB is schema-less. Instead of storing rows in a table you create “documents” that are stored in collections. Each document can be thought of as a kind of very efficient JSON file and a collection is, well, a collection of those documents

#### 4.4.6 Benefits of using documents in database:

- **Documents are natural.** Documents represent data in the same way that applications do. Unlike the tabular rows and columns of a relational database, data can be structured with arrays and subdocuments – in the same way applications represent data, as lists and members / instance variables respectively. This makes it much simpler and faster for developers to model how data in the application will map to data stored in the database.
- **Documents are flexible.** Each document can store data with different attributes from other documents. With JSON documents, we can add new attributes when we need to, without having to alter a centralized database schema.

- **Documents make applications fast.** With data for an entity stored in a single document, rather than spread across multiple relational tables, the database only needs to read and write to a single place. Having all the data for an object in one place also makes it easier for developers to understand and optimize query performance.

#### 4.4.7 Description of our system's database:

Our database system contains four collections announcement, applications, users, and yearly reports.

Collection Name ^	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size	Properties
<a href="#">announcements</a>	4	319.3 B	1.3 KB	1	16.4 KB	
<a href="#">applications</a>	529	4.4 KB	2.3 MB	1	16.4 KB	
<a href="#">users</a>	48	320.4 B	15.4 KB	1	16.4 KB	
<a href="#">yearlyreports</a>	3	2.5 KB	7.5 KB	2	65.5 KB	

Figure 4. 11 database interface

##### 4.4.7.1 Announcement collection:

Documents in that collection contain five attributes:

- **\_id:** it is an identification number that the database inserts by default.
- **Title:** title of the Announcement.
- **Content:** content of the Announcement.
- **createdAt:** date of creation that announcement.
- **updatedAt:** date of update that announcement.

scms-db.announcements

Documents Aggregations Explain Plan Indexes

FILTER

INSERT DOCUMENT VIEW LIST TABLE

```

_id: ObjectId("5d144d3a4342cd07688d7dae")
title: "Ipsam simillique ipsum nemo."
content: "Adipisci a laudantium eligendi. Sit doloremque culpa doloribus nesciun..."
createdAt: 2019-06-27T04:59:38.165+00:00
updatedAt: 2019-06-27T04:59:38.165+00:00
__v: 0

```

Figure 4. 12 Announcement collection



#### 4.4.7.2 applications collection:

Documents in this collection have different attributes according to the type of application, however they have some attributes in common which are:

- **\_id**: it is an identification number that the database inserts by default.
- **finalDecision**: a Boolean attribute (true if the application is accepted, false if not)
- **treated**: a Boolean attribute (true if the application is treated, false if not)
- **submittedAt**: date of application's submission.
- **type**: define the type of an application.
- **session**: an id reference to the yearlyreports collection.
- **applicant**: an id reference to the user who submitted the application.
- **reviews**: an array contains:
  - **\_id**: it is an identification number that the database inserts by default.
  - **reviewer**: id reference to the user who reviewed the application.
  - **decision**: the decision made by that reviewer.
  - **comment**: the comment that added by that reviewer.

```

scms-db.applications
Documents Aggregations Explain Plan Indexes
FILTER
INSERT DOCUMENT VIEW LIST TABLE
{
  "_id": ObjectId("5d144d3a4342cd07688d7db4"),
  "finalDecision": true,
  "treated": true,
  "submittedAt": 2015-03-03T07:00:30.567+00:00,
  "type": "InternshipApplication",
  "laboratoryName": "Adipisci voluptatum non.",
  "laboratoryWebsite": "https://brandt.net",
  "country": "Saudi Arabia",
  "city": "Hollyborough",
  "durationFrom": 2019-07-14T13:18:28.334+00:00,
  "durationTo": 2020-03-04T12:19:25.789+00:00,
  "workPlanLink": "http://karianne.biz",
  "session": ObjectId("5d144d3a4342cd07688d7db0"),
  "applicant": ObjectId("5d144d324342cd07688d7d7f"),
  "reviews": Array
  > 0: Object
    {
      "_id": ObjectId("5d144d3a4342cd07688d7ed6"),
      "reviewer": ObjectId("5d144d394342cd07688d7da2"),
      "decision": true,
      "comment": "Aut pariatur autem nihil et laboriosam corporis eum suscipit. Sed at e..."
    }
  > 1: Object
  > 2: Object
  > 3: Object
}

```

Figure 4. 13 applications collection

#### 4.4.7.3 Users collection:

In this collection, documents have a personal information about the user (first name, last name, gender, department, major, option, email and password).

they also have:

- **\_id**: it is an identification number that the database inserts by default.
- **roles**: an array defines the roles of a user.

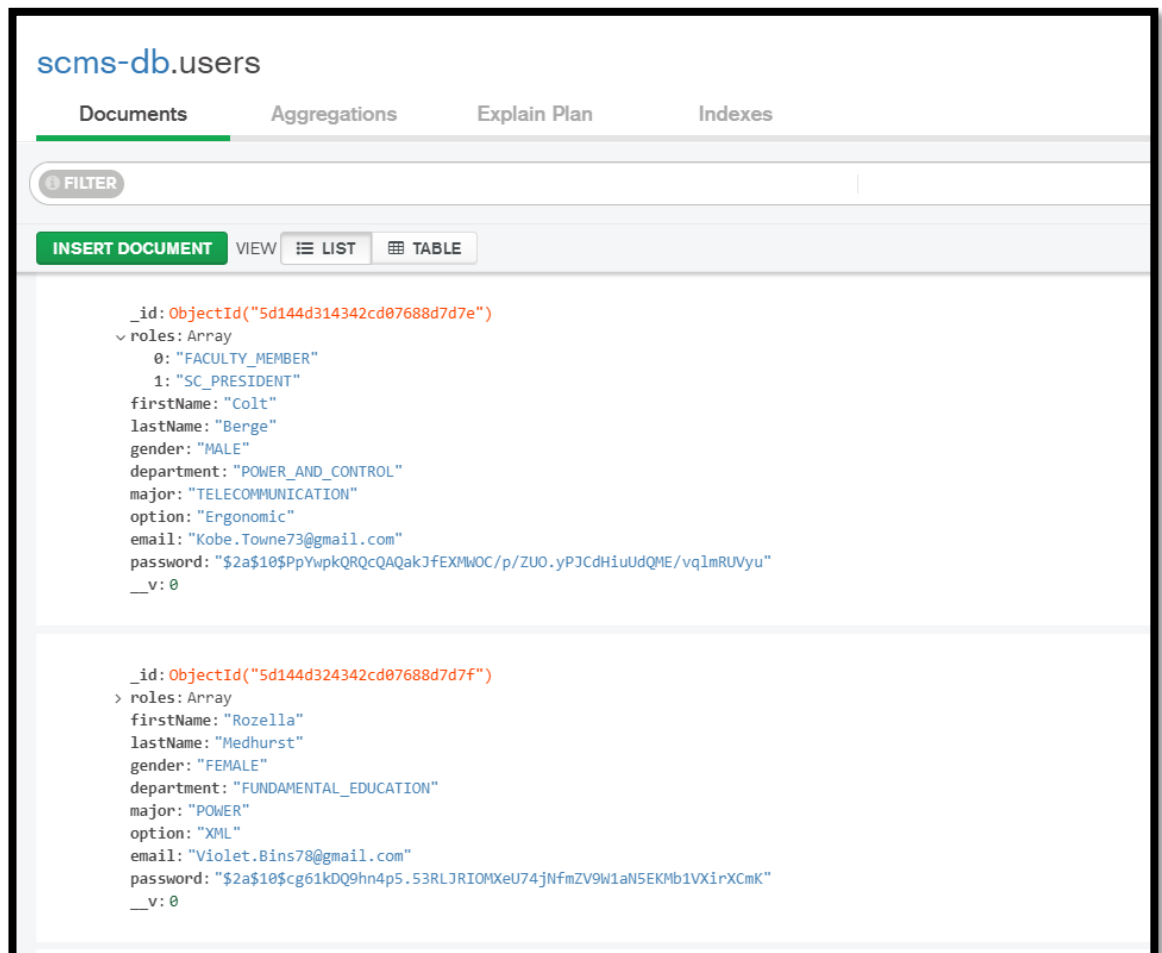


Figure 4. 14 Users collection

#### 4.4.7.3 yearly reports collection:

Documents in this collection have five main attributes:

- **\_id**: it is an identification number that the database inserts by default.
- **year**: decide which year we look for.
- **sessions**: array contains:
  - **meetingAgenda**: the agenda that the president set.
  - **\_id**: it is an identification number that the database inserts by default.
  - **submissionStartDate**: the submission Start Date that the president fixed.

- **submissionEndDate**: the submission End Date that the president fixed.
- **meetingDate**: the meeting Date that the president fixed.
- **statistics**: array contains:
  - **Type of an application**: it is another array contains:
    - **applications**: number of submitted application on this type.
    - **accepted**: number of application accepted in this type.
    - **electronicsDepartement**: number of submitted application that have an applicant belongs to electronics department.
    - **controlMajor**: number of submitted application that have an applicant belongs to control major.
    - **electronicsMajor**: number of submitted application that have an applicant belongs to electronic major.
    - **female**: number of submitted application that have a female applicant.

The screenshot shows the MongoDB Compass interface for the 'scms-db.yearlyreports' collection. The document is displayed in a tree view with the following structure:

```

{
  "_id": ObjectId("5d144d3a4342cd07688d7daf"),
  "year": "2015",
  "sessions": Array
    [
      0: Object
        {
          "meetingAgenda": Array
            [
              0: Object
                {
                  "_id": ObjectId("5d144d3c4342cd07688d8290"),
                  "submissionsStartDate": "2016-04-01T00:00:00.000+00:00",
                  "submissionsEndDate": "2016-06-24T00:00:00.000+00:00",
                  "meetingDate": "2016-06-24T00:00:00.000+00:00"
                }
              1: Object
              2: Object
              3: Object
              4: Object
              5: Object
            ]
          "__v": 1
        }
    ]
  "statistics": Object
    {
      "addThesisCoSupervisorApplication": Object
        {
          "applications": 25,
          "accepted": 15,
          "electronicsDepartment": 5,
          "fundamentalEducationDepartment": 7,
          "powerAndContorlDepartment": 13,
          "controlMajor": 9,
          "electronicsMajor": 5,
          "powerMajor": 4,
          "telecommunicationMajor": 7,
          "female": 9
        }
      "confirmationApplication": Object
    }
  }

```

**Figure 4. 15** yearly reports collection

#### 4.5 Conclusion:

In this chapter, we have used features of UML to model all functionalities involved in our system through use case diagram. Moreover, we modelled our application by providing answers to our modelling and design, based on the analysis of needs of our application. Finally, we introduced the relational model and the non-relational model, and we gave some benefits using non-relational model.

# ***CHAPTER FIVE***

---

## ***Implementation***

## 5.1 Introduction:

After designing the application and specifying the roles of each user, this last chapter is concerned with the implementation. We give a presentation of the application and its functionalities accompanied with some user interfaces.

## 5.2 Interfacing with the application:

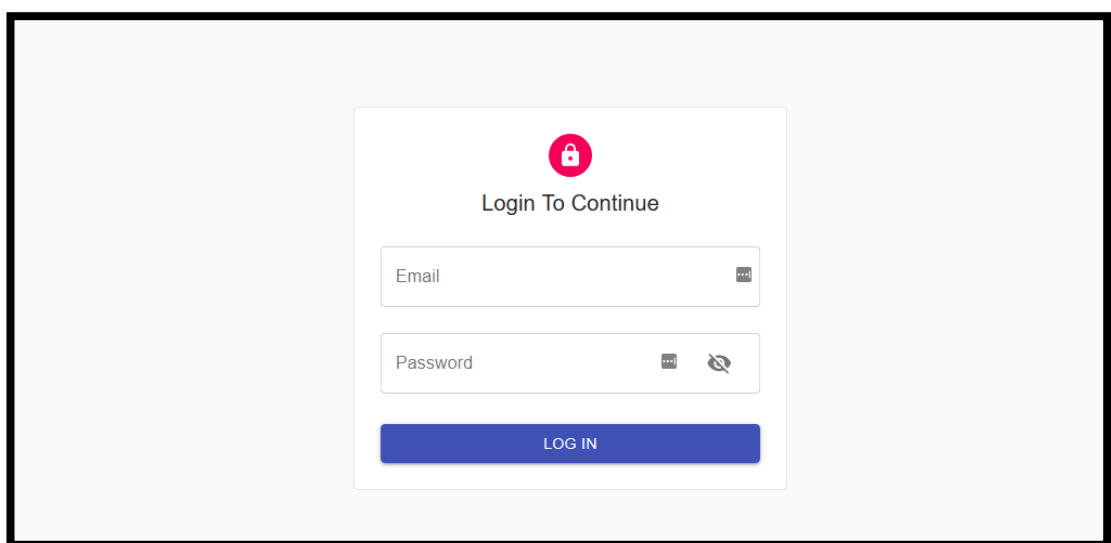
After implementing the application with the tools specified in the previous chapters, in this paragraph, we present the different interfaces, the user will be dealing with.

**Note:** It is to be noted that all the filled data that are saved in the database was generated by a faker library that give as a random string.

### 5.2.1 Authentication interface:

As all secured applications, this following interface is the first one that will be displayed when the user wants to access the application.

Firstly, to sign up in the application you need to send an email to the admin accompanied with the required data which is first name, last name, gender, and roles. The admin then verifies this information before creating the requested account and sending the email and password to the user. When the user enters email and password, the system sends a query to the server for checking. If this information exists in the database he can access otherwise an error message will be displayed. Figure 5.1 shows the authentication interface.



**Figure 5. 1 Authentication Interface**

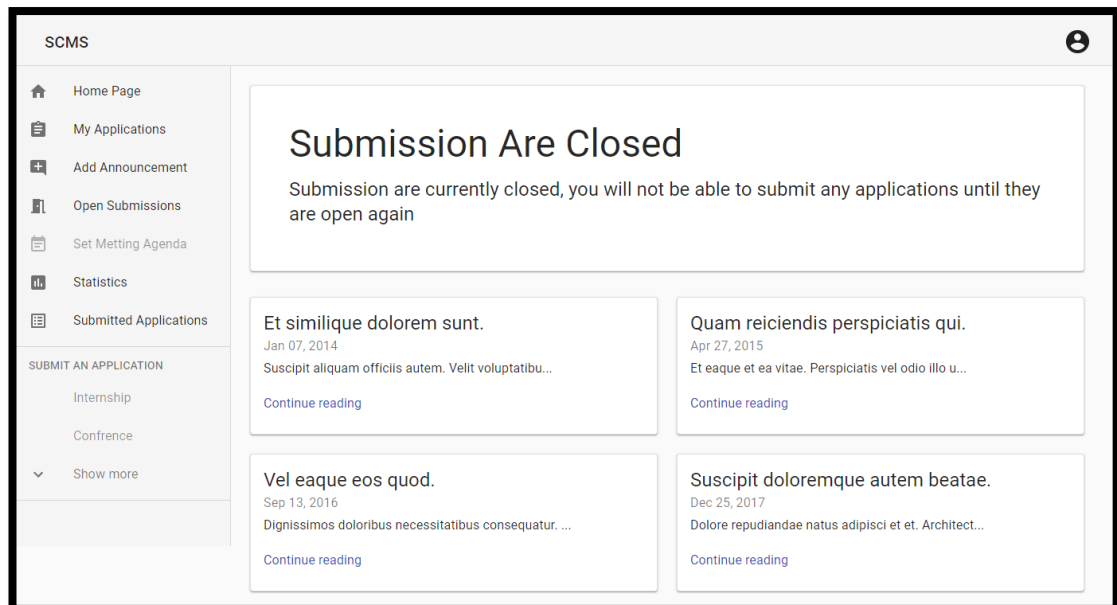
### 5.2.2 Homepage interfaces for each user:

All users have two parts in their homepage interface, a main area and a sidebar. The main area is shared between all users. When submissions are open, users can see the ending date of the submissions duration, the meeting date and agenda, otherwise a statement of ‘Submissions closed’ is appearing instead.

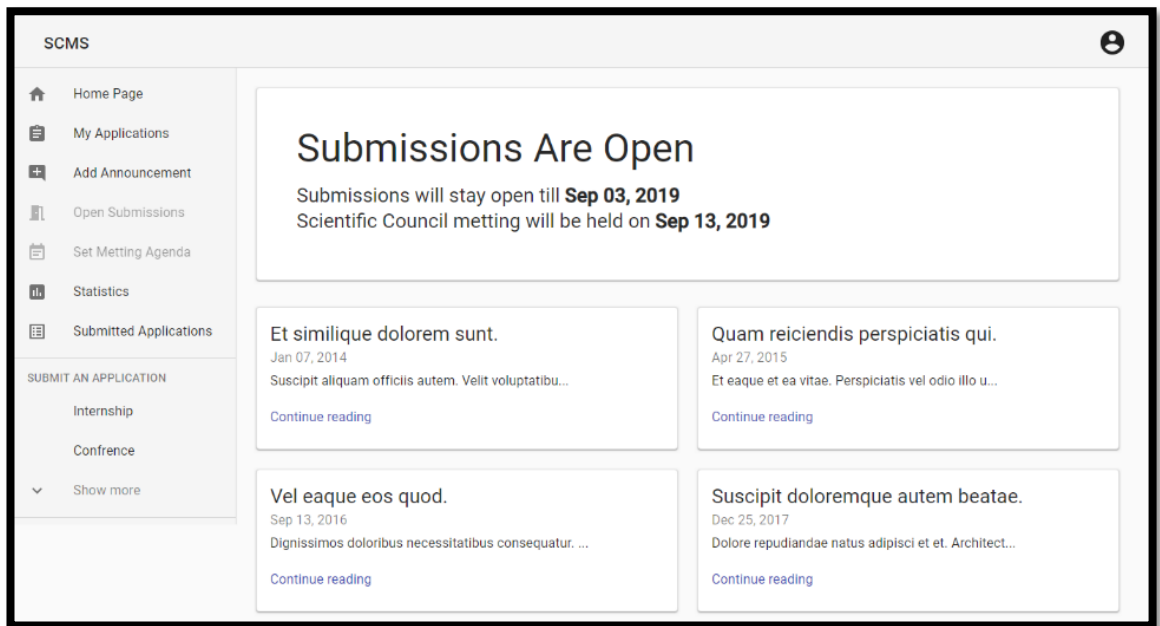
Under this part users can see some announcement that have been added via the SC President.

The sidebar differs from a user to another:

- For the SC president, it consists of Home page, My applications, Add announcement, Open submissions, Set meeting agenda, Review application, Submit an application, and Statistics.
- For the SC members it consists of Home page, My applications, Review application, and Submit an application.
- For the Faculty member and PhD student it consists of Home page, My applications, and Submit an application.



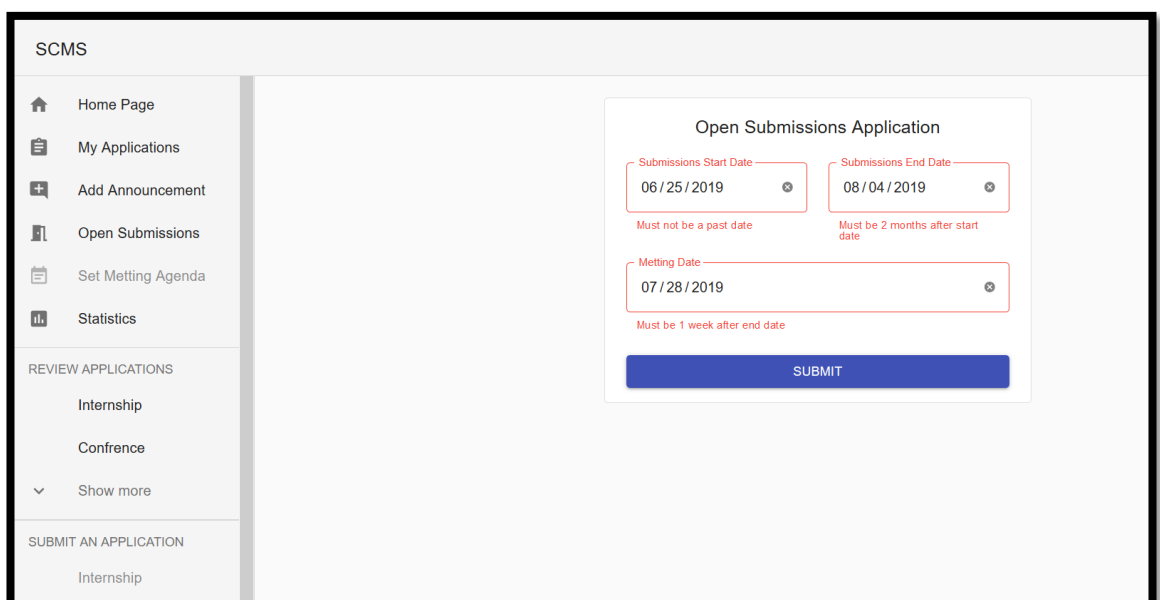
**Figure 5. 2 Homepage interface when submissions are closed**



**Figure 5. 3 Homepage interface when submissions are open**

### 5.2.3 Open submissions interface:

When the SC president enters the Open submissions interface, he can set the submissions duration and the meeting date. The system announces this information on the interfaces of all users. The starting date cannot be a past date, the ending date should be at least two months after the starting date and the meeting date should be one week after the ending date.



**Figure 5. 4 Open submissions interface**



### 5.2.4 Submit an application interface:

After opening the submissions, users can use the submission interface to submit an application. After choosing an application user should fill the form below and submit. The user can edit the information of the submitted application before the duration ends.

The screenshot shows the SCMS interface for submitting an application. On the left, there is a sidebar menu with sections: 'REVIEW APPLICATIONS' (listing Internship, Conference, Confirmation, Promotion, Thesis, Registration, Registration Renewal, Defense, Research Sumition, Course Handout, Show less) and 'SUBMIT AN APPLICATION' (listing Internship). The main content area is titled 'Internship Application' and contains the following form fields:

- Laboratory Name (text input)
- Laboratory Website (text input)
- Laboratory Location:
  - Country (text input)
  - City (text input)
- Internship Duration:
  - From: 05 / 13 / 2019 (date picker)
  - To: 06 / 13 / 2019 (date picker)
- Selected Work Plan File: S057082-2.pdf (file upload)

A blue 'SUBMIT' button is located at the bottom of the form.

**Figure 5. 5 Submit an application interface**

### 5.2.5 Set meeting agenda interface:

After the submissions duration ends, the president should set an agenda for the meeting then the system will announce it to all users. This interface is deactivated when the submissions are open.

The screenshot shows the SCMS interface for setting a meeting agenda. On the left, there is a sidebar menu with sections: 'Home Page', 'My Applications', 'Add Announcement', 'Open Submissions', 'Set Metting Agenda', 'Statistics', 'REVIEW APPLICATIONS' (listing Internship, Conference, Show more), and 'SUBMIT AN APPLICATION' (listing Internship, Conference, Show more). The main content area is titled 'Metting Agenda' and contains the following form fields:

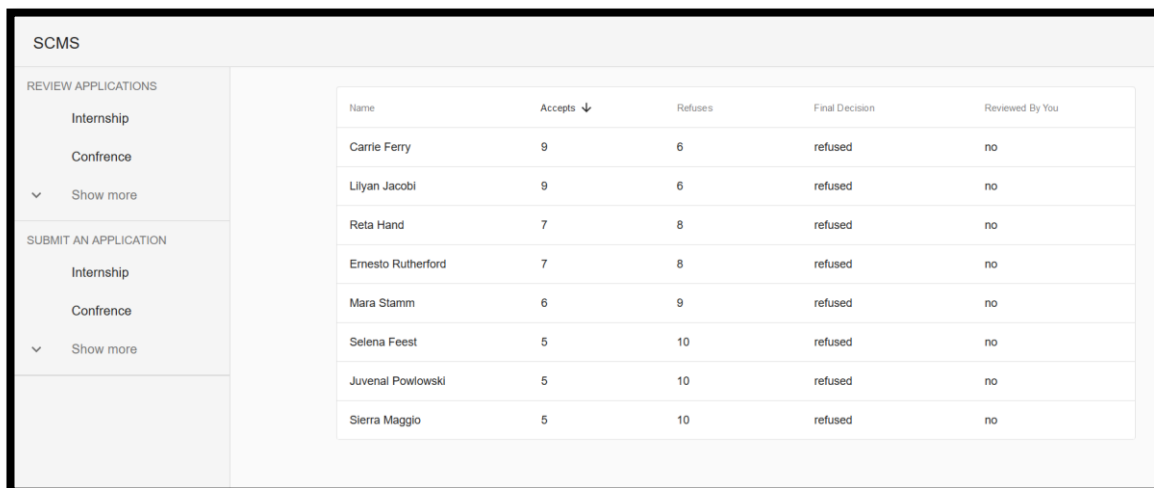
- Metting Agenda Topic #1 (text input with a red minus icon)
- Metting Agenda Topic #2 (text input with a red minus icon)
- Metting Agenda Topic #3 (text input with a red minus icon)
- Metting Agenda Topic #4 (text input with a red minus icon)

Below the topics is a grey 'ADD A TOPIC' button and a blue 'SUBMIT' button.

**Figure 5. 6 Set meeting agenda interface**

### 5.2.6 Review applications interface:

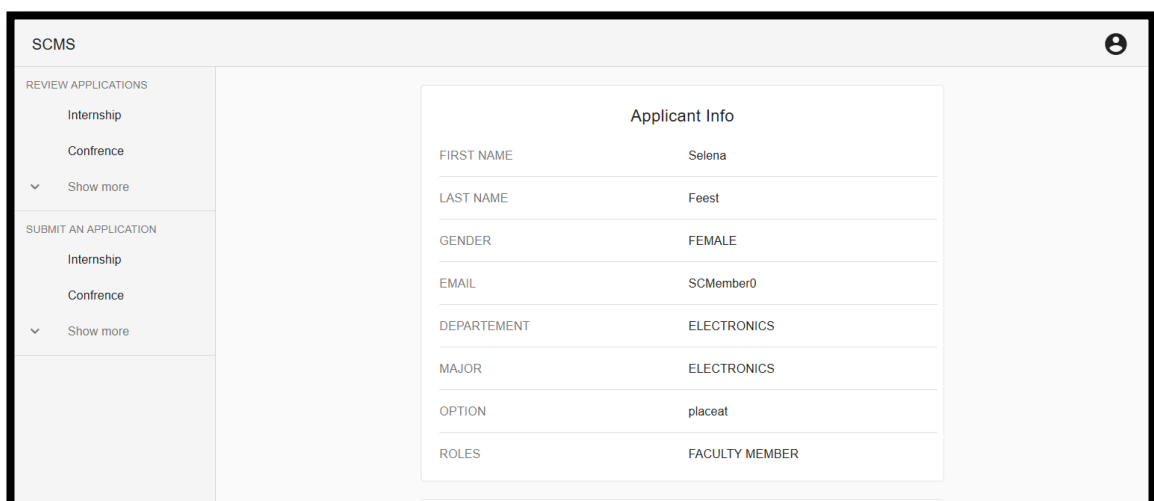
In this interface user can find a list of users who have submitted an application. It contains also the number of reviewers who accept the application and the ones who refuse it. In addition, it is mentioned if that reviewer has already reviewed the application or not (see figure 5.7).



Name	Accepts ↓	Refuses	Final Decision	Reviewed By You
Carrie Ferry	9	6	refused	no
Lilyan Jacobi	9	6	refused	no
Reta Hand	7	8	refused	no
Ernesto Rutherford	7	8	refused	no
Mara Stamm	6	9	refused	no
Selena Feest	5	10	refused	no
Juvenal Powlowski	5	10	refused	no
Sierra Maggio	5	10	refused	no

**Figure 5. 7 Review applications interface**

Once the reviewer selects one of the listed users, he can view the applicant information as shown in figure 5.8.



Applicant Info	
FIRST NAME	Selena
LAST NAME	Feest
GENDER	FEMALE
EMAIL	SCMember0
DEPARTEMENT	ELECTRONICS
MAJOR	ELECTRONICS
OPTION	placeat
ROLES	FACULTY MEMBER

**Figure 5. 8 the applicant information Interface**

The detailed information related to the submitted application can be seen too by the reviewer (figure 5.9).

Internship Application	
LABORATORY NAME	Sint aliquam iure.
LABORATORY WEBSITE	http://joseph.name
<b>Location</b>	
CITY	Lake Maximo
COUNTRY	Botswana
<b>Duration</b>	
From	Thu Feb 13 2020
To	Wed Sep 18 2019
WORK PLAN	<a href="#">DOWNLOAD</a>

**Figure 5. 9 internship application Interface**

In addition, the reviewer can see the others' reviews decisions for the current application. It is to be noted here that for SC members (reviewers), they can only see the number of votes, without seeing the viewers' names as mentioned in figure 5.10.

Review	
Accepts	5
Refuses	10
Final Decision	refused

**Figure 5. 10 The review status Interface for SC member user**

Finally, the reviewer can submit his comments about the submitted application followed by his decision about it as shown in figure 5.11.

Application Assessment	
Your Decision	Approve
Comment	
<a href="#">SUBMIT</a>	

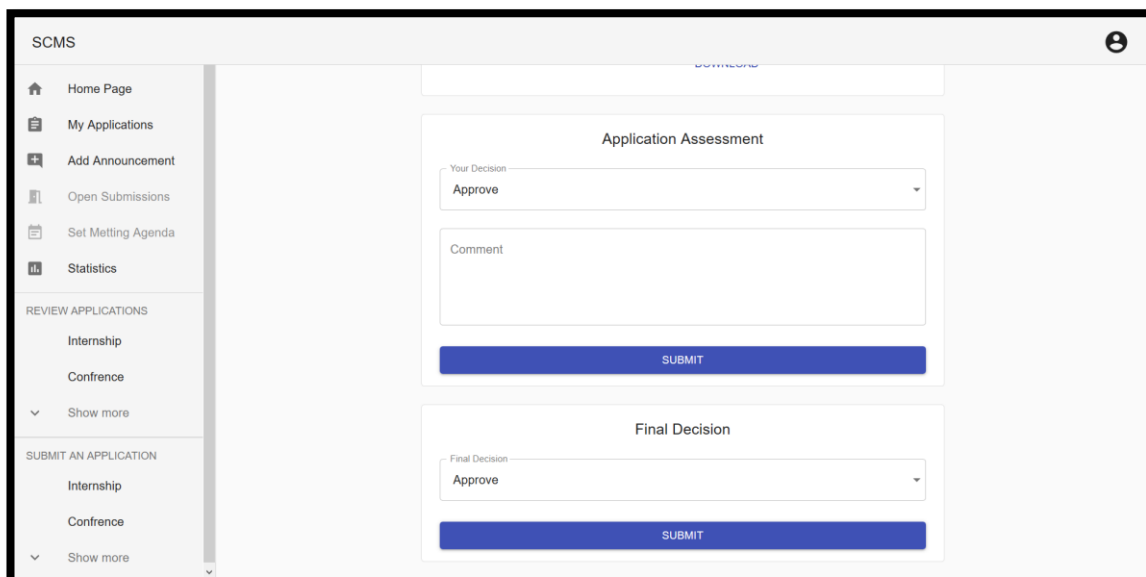
**Figure 5. 11 application assessment Interface**

As far as the SC President is concerned, his review application interface contains more features: he can see the number of votes, the reviewers' names, their comments and their decisions. Furthermore, he can also check the previous submitted applications of that applicant.

15 Reviews	5 Accepts	10 Refuses
<p>REVIEWER</p> <hr/> <p>DECISION</p> <hr/> <p>COMMENT</p> <p>Qui nulla voluptatum. Qui commodi officia est nihil. Nisi optio reprehenderit animi omnis.</p>	<p>Odell McDermott</p> <hr/> <p>Refused</p>	
<p>REVIEWER</p> <hr/> <p>DECISION</p> <hr/> <p>COMMENT</p> <p>Nostrum sed quas illum quis. Quia tenetur eius doloribus. Minima adipisci laboriosam blanditiis deserunt amet. Placeat quo quod voluptatem. Deserunt iure et rerum iure labore quas itaque rerum.</p>	<p>Alda Koepp</p> <hr/> <p>Refused</p>	
<p>REVIEWER</p> <hr/> <p>DECISION</p> <hr/> <p>COMMENT</p> <p>Praesentium odit quo reiciendis illum. Asperiores ad atque quia est ut harum repudiandae sunt. Voluptatum fuga reiciendis eveniet placeat. Occaecati culpa id qui et quas saepe voluptate id consectetur.</p>	<p>Amani Reichel</p> <hr/> <p>Accepted</p>	

**Figure 5. 12 the review status Interface for SC President user**

The SC President has to set the final decision about the submitted application. The final decision is set once the meeting is held and the application is discussed with all SC members based on the different submitted reviews. Therefore, this interface won't appear until the date of the meeting. Figure 5.13 shows the interface provided for the SC President to set the final decision for the submitted application.



**Figure 5. 13 the final decision Interface**

**5.2.7 My applications interface:**

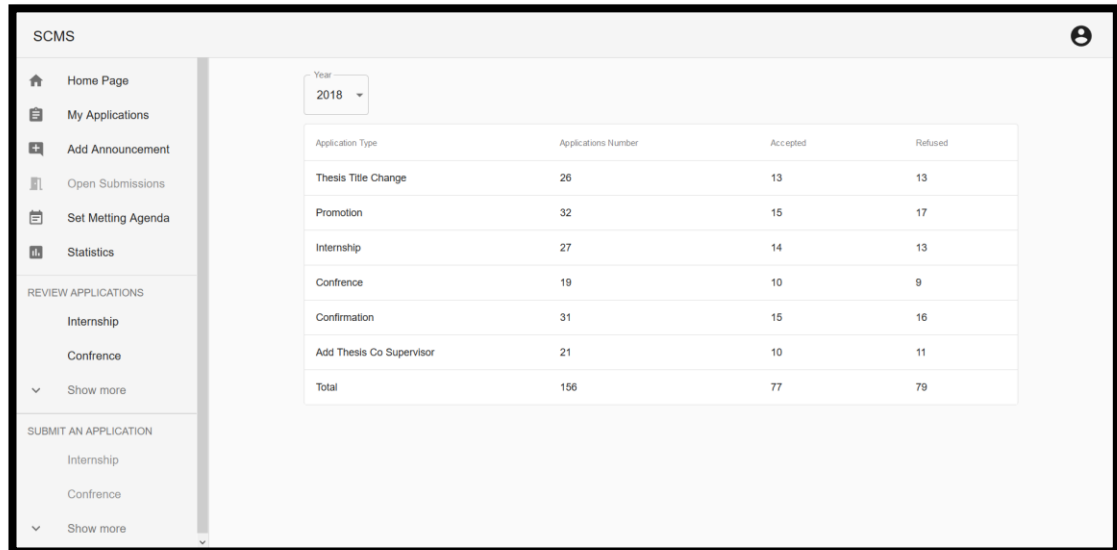
In this interface users can see and review all their submitted applications.

Application Type	Submission Date	Treated	Final Decision
Promotion	Feb 25, 2015	yes	accepted
Confrence	Mar 23, 2015	yes	accepted
Internship	Mar 13, 2015	yes	accepted
Confirmation	Feb 06, 2015	yes	accepted
Add Thesis Co Supervisor	Jan 19, 2015	yes	accepted
Thesis Title Change	Mar 17, 2015	yes	refused
Confirmation	Apr 29, 2016	yes	accepted
Add Thesis Co Supervisor	May 05, 2016	yes	refused
Confrence	Jun 22, 2016	yes	refused
Internship	Jun 07, 2016	yes	refused
Promotion	Jun 20, 2016	yes	refused
Thesis Title Change	Apr 13, 2016	yes	accepted

**Figure 5. 14 My applications interface**

### 5.2.8 Statistics interface:

The president has a statistics part which is a list of all submitted applications with the number of accepted and refused ones and the total number for each type of application per year. Figure 5.15 is an example of some statistics retrieved from our filled database.



**Figure 5. 15 Statistics interface**

In this statistics interface, when the president selects one of the listed items (types of submission) he can view a pie chart containing some statistic information related to this item. A classification is provided by different criteria including gender, acceptance, department, and major. Figure 5.16 shows an example of that interface.



**Figure 5. 16 pie chart for a type of application interface**

### **5.3 Conclusion:**

In this chapter, we have given a brief explanation about the interfaces of the different users interacting with our system and how can they deal with them to fulfill their desired tasks. As we have seen, the user interfaces are simple to use and provide a rapid access to all necessary information requested in the different scenarios that may occur.

## **General Conclusion**

In this report, we have presented the different steps to design and develop a web application intended for managing a hypothetical scientific council of the institute. In order to realize this project, we have gathered and analyzed problems encountered by the staff. For that purpose, several discussions have been realized with a member of the scientific council.

The major objective of our project is to facilitate the work of the users of the scientific council (SCI president, SCI member, Faculty member, and PhD students) including the electronic storage of data and the rapid access to it, and to achieve a better decision making about every submitted. In addition, data now is saved in a secured and centralized database. We have implemented a web application, so it can be accessed from anywhere at any time using internet.

This project, which falls in the field of the design and implementation of an information system, was very interesting and allowed us to become familiar with new concepts, and to improve our knowledge and skills in the field of programming.

## **Future Works**

The first version of the SCI management system is achieved. This version may be enhanced more by including other functionalities. Among those functionalities, is to let the administrative staff (Director, Post-Graduate department head, and Department heads) interact with our system to submit their pedagogical issues and related ones. In addition, the application should allow the automatic generation of the transcript of the meeting.



## Webography

- [1] [https://techterms.com/definition/web\\_application](https://techterms.com/definition/web_application). Consulted (2 May 2019)
- [2] <https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58>. Consulted (2 May 2019)
- [3]  
[https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP\\_Basics.html](https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html). Consulted (5 May 2019)
- [4] <https://www.techopedia.com/definition/1352/uniform-resource-locator-url>. Consulted (5 May 2019)
- [5] Executive Decree No. 03-279 of 23 August 2003 setting out the specific missions and rules pertaining to the organization and operation of the university.
- [6] <https://www.techopedia.com/definition/27927/nodejs>. Consulted (17 May 2019)
- [7] <https://www.guru99.com/node-js-express.html>. Consulted (17 May 2019)
- [8] <https://www.c-sharpcorner.com/article/what-and-why-reactjs/>. Consulted (17 May 2019)
- [9] <https://graphql.org/learn/>. Consulted (19 May 2019)
- [10] <https://www.apollographql.com/docs/apollo-server/essentials/schema/>. Consulted (24 June 2019)
- [11] <https://www.apollographql.com/docs/apollo-server/essentials/data/>. Consulted (24 June 2019)
- [12] <https://www.apollographql.com/docs/apollo-server/essentials/server/>. Consulted (24 June 2019)
- [13] <https://www.apollographql.com/docs/intro/platform/>. Consulted (24 June 2019)
- [14] <https://searchdatamanagement.techtarget.com/definition/MongoDB>. Consulted (02 June 2019)
- [15] [https://techterms.com/definition/web\\_browser](https://techterms.com/definition/web_browser). Consulted (01 June 2019)
- [16] <https://www.techopedia.com/definition/1892/hypertext-markup-language-html>. Consulted (15 May 2019).
- [17] <https://www.techopedia.com/definition/26268/cascading-style-sheet-css>. Consulted (15 May 2019).
- [18] <https://www.techopedia.com/definition/3929/javascript-js>. Consulted (15 May 2019).

- [19] <https://www.techopedia.com/definition/20810/modeling-language>. Consulted (13 June 2019).
- [20] <http://www.agilemodeling.com/essays/umlDiagrams.htm>. Consulted (13 June 2019).
- [21] <https://www.uml-diagrams.org/use-case-diagrams.html>. Consulted (13 June 2019).
- [22] <https://www.uml-diagrams.org/sequence-diagrams.html>. Consulted (29 June 2019)
- [23] <https://aws.amazon.com/relational-database/>. Consulted (20 June 2019).
- [24] <https://docs.microsoft.com/en-us/azure/architecture/data-guide/big-data/non-relational-data>. Consulted (20 June 2019).
- [25] <https://www.mongodb.com/compare/mongodb-mysql>

### **Bibliography**

- [26] Robin Nixon Learning PHP, MySQL & JavaScript 4th Edition
- [27] Ramez Elmasri, Shamkant B. Navathe, Fundamentals of Database Systems, 6th Ed. (2010)
- [28] G. Booch, J. Rumbaugh, I. Jacobson, The Unified Modelling Language Reference Manual, 2nd ed. 2005