

Design and Implementation of a Responsive Web-Based Library Management System for Educational Purposes

Hasan Özçelik

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science
in
Information and Communication Technologies in Education

Eastern Mediterranean University
September 2020
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Ali Hakan Ulusoy
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science in Information and Communication Technologies in Education.

Prof. Dr. Ersun İşçiođlu
Chair, Department of Computer
Education and Instructional
Technologies

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Information and Communication Technologies in Education.

Asst. Prof. Dr. Hüsnu Bayramođlu
Supervisor

Examining Committee

1. Asst. Prof. Dr. Hüsnu Bayramođlu

2. Asst. Prof. Dr. Emre Özen

3. Asst. Prof. Dr. Kamil Yurtkan

ABSTRACT

This study was conducted for building a responsive web application for managing the Eastern Mediterranean University library, both for library users and librarians. The followed research method for the project is Design Base Implementation Research. The main objective of the project is to provide the mentioned user base a web application that they can use for managing their library online usage, easily and efficiently. This is aimed to be achieved by building a responsive application, meaning that the user can use the application on any smart device, such as a computer, a tablet, or a mobile, also, mostly automating the borrowing, reserving and returning resources operations Thus, decreasing the time and effort spent to carry out these processes both for library users and the librarian.

The developed project was built with the client-server approach, consisting of two applications, a backend application (service) and a frontend application (consumer). The backend application handles receiving requests from the frontend application, process data and respond the according information to the frontend application. Whereas, the frontend application provides the user a graphical user interface for using and interacting with the system, both sending requests to the backend application, and displaying information responded from the backend application.

This project was developed by using a range of technologies and methodologies. The backend application was built by using NodeJS, JavaScript, and ExpressJS. Moreover, the frontend application was built with a modern, responsive approach, by using HTML, CSS, JavaScript, ReactJS and Material-UI.

Overall, this project provides a modern, responsive web application in order to carry out general library management operations online, faster, easier and more efficient than the current methods that are used in the Eastern Mediterranean University library.

Keywords: Library management, responsive design, client-server model, React, NodeJS

ÖZ

Bu çalışma, hem öğrenciler ve kütüphaneciler için Doğu Akdeniz Üniversitesi kütüphane yönetimi için esnek web uygulama oluşturmak için yapılmıştır. Proje için takip edilen araştırma yöntemi, Tasarım Temel Uygulama Araştırması'dır. Projenin temel amacı, belirtilen kullanıcılara, kütüphane kullanımlarını çevrimiçi, kolay ve verimli bir şekilde yönetmek için kullanabilecekleri bir web uygulaması sunmaktır. Bunun, duyarlı bir uygulama oluşturarak elde edilmesi amaçlanmaktadır, yani kullanıcının uygulamayı bilgisayar, tablet veya mobil gibi herhangi bir akıllı cihazda kullanabilmesi, ayrıca genel olarak ödünç alma, ayırma ve iade işlemlerini otomatikleştirmesi, böylece hem öğrenciler hem de kütüphaneci için bu işlemleri gerçekleştirmek için harcanan zamanı ve çabayı azaltacaktır.

Geliştirilen proje bir bütün olarak istemci-sunucu yaklaşımı ile oluşturulmuş, arka uç uygulama ve ön uç uygulama olarak iki alt uygulamadan oluşturulmuştur. Arka uç uygulaması, ön uç uygulamasından gelen istekleri ele alır, verileri işler ve ön uç uygulamasına uygun bilgileri yanıtlar. Bunun yanında ön uç uygulaması, kullanıcıya hem arka uç uygulamasına istek göndererek hem de arka uç uygulamasından yanıtlanan bilgileri görüntüleyerek sistemi kullanmak ve onunla etkileşim kurmak için bir grafik kullanıcı arabirimi sağlar.

Bu proje, bir dizi teknoloji ve metodoloji kullanılarak geliştirilmiştir. Arka uç uygulaması NodeJS, JavaScript ve ExpressJS kullanılarak oluşturulmuştur. Ön uç uygulaması ise, HTML, CSS, JavaScript, ReactJS ve Material-UI kullanılarak modern, duyarlı bir yaklaşımla oluşturulmuştur.

Genel olarak, bu proje çevrimiçi genel kütüphane yönetim işlemlerini yürütmek amacıyla modern, duyarlı, daha hızlı, daha kolay ve Doğu Akdeniz Üniversitesi kütüphanesinde kullanılan güncel yöntemlere göre daha verimli bir web uygulaması sağlar.

Anahtar Kelimeler: Kütüphane yönetimi, duyarlı tarasım, istemci-sunucu modeli, React, NodeJS

To My Family and Beloved Ones

ACKNOWLEDGEMENT

I happily want to state that I am very lucky to have my family members, my girlfriend, and my supervisor besides me at all times, during my postgraduate studies.

I am greatly thankful for the support and guidance that was provided to me by my supervisor Asst. Prof. Dr. Hüsnü Bayramođlu, always with patience and care, regardless of how tough the times was during my studies.

Also, I would like to thank my family for their endless encouragement and support along the way. They have always been by my side, and I will always appreciate that they have always been and will be supporting of me with any path I want to follow in life.

Additionally, I am thankful to my co-workers at Analiz Systems for their care, help and support.

At last, I would like to give special thanks to my girlfriend, for her endless support, help, attention, and always believing in me, regardless of time or place, providing me strength throughout my study.

TABLE OF CONTENTS

ABSTRACT	iii
ÖZ	v
DEDICATION	vii
ACKNOWLEDGEMENT	viii
LIST OF TABLES	xi
LIST OF FIGURES	xii
1 INTRODUCTION	1
2 RELATED WORKS	6
3 PROPOSED SYSTEM	12
3.1 Frontend Technologies and Methodologies	13
3.1.1 Responsive Web Design	13
3.1.2 Hypertext Mark-up Language Version 5	15
3.1.3 Cascading Style Sheets Version 3	16
3.1.4 JavaScript	17
3.1.5 React	19
3.1.6 Material-UI	21
3.2 Backend Technologies and Methodologies	22
3.2.1 NodeJS	22
3.2.2 ExpressJS	23
3.2.3 MySQL	24
3.2.4 Design-Based Implementation Design	24
3.3 Software Development Architecture	25
3.3.1 Development Life Cycle	25

3.3.2 Iterative Model	25
3.3.3 Scrum.....	26
3.4 Proposed Library Management System Design Model.....	27
3.4.1 Backend Application	31
3.4.1.1 Database Tables	31
3.4.1.2 Routes.....	34
3.4.1.3 Base Route	35
3.4.1.4 Modelled Routes	37
3.4.1.5 Application Programming Interface.....	37
3.4.1.6 Authentication	37
3.4.1.7 Security	41
3.4.2 Frontend Application	42
3.4.2.1 View Models	43
3.4.2.2 Forms	45
3.4.2.3 Search.....	46
3.4.2.4 View Tables	46
3.4.2.5 Authentication.....	46
3.4.3 Testing	47
3.4.4 Development Environment.....	49
3.4.5 Replacing the Already Existing System	50
4 CONCLUSION	51
4.1 Future Work	54
REFERENCES.....	56
APPENDIX.....	62

LIST OF TABLES

Table 1: LMS, Existing System, and Related Works Comparison Table 52

LIST OF FIGURES

Figure 1: Responsive Web Applications on Multiple Devices (Napper, 2020).....	13
Figure 2: JavaScript Frontend Framework Use in 2020 (Makhija, 2020)	20
Figure 3: Iterative Model Cycle (Powell-Morse, 2016).....	26
Figure 4: Scrum Framework Visualization (Job, 2015).....	27
Figure 5: EMU Library Management System Use Case Diagram.....	29
Figure 6: Client-Server Model Visualization (Christensson, 2016).....	30
Figure 7: EMU LMS Database Relationship Diagram	34
Figure 8: An example of Base Route and Modelled Route Relation.....	35
Figure 9: Login Activity Diagram	38
Figure 10: Verify Token Activity Diagram	39
Figure 11: Check Library User Role Activity Diagram.....	40
Figure 12: Check Librarian Role Activity Diagram	40
Figure 13: An example of Base View Model and View Model Relation	44
Figure 14: Home Page (Library Users/Guest)	63
Figure 15: Search Results Page.....	63
Figure 16: Author Search Result.....	64
Figure 17: Floor Search Result	64
Figure 18: Shelf Search Result.....	65
Figure 19: Resource Details Page as a Guest.....	65
Figure 20: Login Required Dialog	66
Figure 21: Login Page.....	66
Figure 22: Resource Details Page as a Student/Lecturer	67
Figure 23: Borrow Dialog	67

Figure 24: Reservable Resource.....	68
Figure 25: Reserve Dialog	68
Figure 26: Current Library User Processes	69
Figure 27: Librarian Home Page.....	70
Figure 28: Resources Page (Librarian)	70
Figure 29: Add Resource (Librarian).....	71
Figure 30: Edit Resource (Librarian)	72
Figure 31: Resource Types Page (Librarian)	72
Figure 32: Add Resource Type Page (Librarian).....	73
Figure 33: Edit Resource Type Page (Librarian)	73
Figure 34: Authors Page (Librarian)	74
Figure 35: Add Author Page (Librarian).....	74
Figure 36: Edit Author Page (Librarian).....	75
Figure 37: Add Resource Item Page (Librarian).....	75
Figure 38: Floors Page (Librarian).....	76
Figure 39: Add Floor Page (Librarian)	76
Figure 40: Edit Floor Page (Librarian).....	77
Figure 41: Shelves Page (Librarian)	77
Figure 42: Add Shelf Page (Librarian).....	78
Figure 43: Edit Floor Page (Librarian).....	78
Figure 44: Library User Processes (Librarian)	79
Figure 45: Borrow Request Dialog (Librarian).....	79
Figure 46: Return Borrow Page (Librarian).....	80
Figure 47: Return Borrow Request	80

LIST OF ABBREVIATIONS

BLM	Bluetooth Library Manager
CRUD	Create, Read, Update, Delete
CSS	Cascading Style Sheets
DOM	Document Object Model
EMU	Eastern Mediterranean University
GSM	Global System for Mobile Communications
GUI	Graphical User Interface
HTML	Hypertext Mark-up Language
HTTP	Hypertext Transfer Protocol
I/O	Input/Output
IoT	Internet of Things
JSON	JavaScript Option Notation
JSX	JavaScript XML
LMS	Library Management System
MVC	Model-View-Controller
NFC	Near-Field Communication
OFET	Organic Field Effect Transistor
RFID	Radio-frequency Identification
SDLC	Software Development Life Cycle
SMS	Short Message Service
SPICE	Simulation Program with Integrated Circuit Emphasis
SQL	Structured Query Language
UI	User interface

URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WSN	Wireless Sensor Network
XML	Extensible Markup Language
XSS	Cross-site Scripting

Chapter 1

INTRODUCTION

In a university, the library is a large part in providing the students a solid environment to access resources of many kinds. Thus, students spend a considerable amount of time in libraries to use these resources and to gain or enhance their knowledge while studying. Besides of physical resources held in a library, there are also a large amount of digital resources that are provided as well. So, it is evident that a tool for browsing and managing a library is almost essential (Chang, 2013).

Over the years, there have been different ways of managing libraries. In the past, we would often see libraries being managed in more traditional ways, without the use of internet, but nowadays this is possible to be mostly achieved online. Even so, there are many types of online solutions ranging from open source software to mobile applications or systems that are integrated with sensors and different networking technologies. For such reasons, today nearly all universities have utilized their university library with a dedicated online library management system.

Library management systems are mostly managed or used online by logged in library users and sometimes even by guest users as well. It is clear why that is the case; overall, online systems of any other type are also a consistent part of our lives and they are being used for many other activities out of school as well such as entertainment, shopping, communication as in social media and so forth (Torres-Díaz, Duart, Gómez-

Alvarado, Marín-Gutiérrez, & Segarra-Faggioni, 2016). Also, for students, online systems are used frequently if not every day in school for managing or carrying out their school activities during or out of their classes. These online activities typically are or similar to student portals, learning management systems, websites that are dedicated to their lectures, or even classes that are held online.

Nowadays, most of these mentioned systems are developed as responsive web applications, which are websites that are built with Hypertext Mark-up Language version 5 (known as HTML) and Cascading Style Sheets version 3 (known as CSS) and have user interfaces (UI) that are adaptive among many different devices and screen sizes (Giurgiu & Gligorea, 2017). So, responsive web applications are especially very popular, because they essentially give students a wide range of freedom and the ability to adapt their daily needs, either being used on a computer or a mobile device. In brief, most people are familiar with online management systems as they are using it for different activities in their daily and school lives. Thus, factors such as these make a well built, fully responsive online library management system a significantly useful tool for all library users.

In recent times, it is apparent that most of the online library management systems are also shifting towards mobile friendly environments, similar to other systems. In some cases, even new systems are being developed specifically for mobile devices to replace existing systems. So, it is clear that mobile friendly online applications are more preferable overall when it comes to library management systems.

Most of these applications that are replacing existing systems focus on making processes that take place in a library quicker and easier. Mostly, these processes are about locating and borrowing books.

The proposed study adopts the research methodology which is called Design Base Implementation Research which is also known as DBIR. DBIR is an effective research methodology where an emphasis is taken on focusing on a solution for a problem, following certain conditions (LeMahieu, Nordstrum, & Potvin, 2017). Nowadays, DBIR is often used by researchers and guides them to develop and implement, and also test innovations in accurate environments. So, this study follows the DBIR methodology by including a wide literature review of similar systems that have similar aims and also related works.

This study aims to develop a system which is a responsive and user-friendly web application that will be used both by library users or clients and library staff. By using the system, library users (students or lecturers) will have the ability to search any resource in the library, see the details of the respective resource, see borrowing or reserving details, as well as carrying out the borrowing process of a resource online. The library staff will also have the ability to search any resource in the library, see the details of a resource similar to library users, but besides of that, the library staff will also have the ability to carry out such as borrowing or reserving processes, seeing detailed information about borrowing or returning processes, as well as general administrative tasks given to the library staff.

This study focuses on developing a responsive web application, which aims to be adaptive to different devices and screen sizes ranging from personal computers to

mobile devices. This will be ensured by testing the application on different devices and platforms. Likewise, the project will be designed focusing on user centric UI design in order to make the developed application easy to use and user friendly. For achieving the desired aims and objectives, several techniques such as user centric design, responsive design and technologies will be used. Alongside HTML5 and CSS3 which were previously mentioned, other frameworks and libraries are also used that are based on JavaScript both on client side (frontend) and also on server side (backend).

As a fully responsive web application, the project contains two sections that work in correlation, those sections being backend and frontend. Backend section being responsible to handle server-side operations of the application such as the database connection and frontend section being responsible for the UI implementation. For backend development, a JavaScript framework called ExpressJS is used, which is a popular framework for building application programming interfaces (API) for web applications that runs in NodeJS environment. Subsequently for frontend development, another popular JavaScript library called ReactJS (also known as React) is used, which is a library that is used for creating UIs and is known for being highly flexible and scalable.

Additionally, my personal experiences with online library management systems also made the benefit of a responsive and user-friendly one very clear. When I was studying abroad in the United Kingdom, I was a frequent user of the university library and the library web application. This application was a responsive, a very user-friendly application, which helped me well and saved me a lot of time, thus affected my learning vastly. Then when I came to study at EMU and tried to use the university's

library web application, I was surprised to see that it was not very easy to use and responsive. The idea that EMU students or lecturers could also benefit from such an application touched me and made me think. Hence, I was greatly motivated to conduct a study in this field, as well as keeping me motivated during my study.

The rest of this thesis is arranged as follows: Chapter 2 reviews other similar studies that have been conducted in recent years, Chapter 3 explains the developed system in technical details, also giving information about the technologies and methodologies that were used, and finally Chapter 4 makes a general conclusion to the study, as well as making a brief statement about possible future work.

Chapter 2

RELATED WORKS

Over the years and especially since the early 2010s, it can be said that there have been a considerable growth on online library management system development (Londhe & Patil, 2015). As a part of that, literature growth in the implementation of online library management systems also is apparent overall. Within this growth, even though the aims and the objectives are really similar, there are some different approaches that are taken when providing a solution. These different approaches could be seen on factors such as different technologies, platforms, principles, and so on.

Moreover, a study by authors Suda and Rani, proposes a study which aims to design a system where the identification of a large number of books at a time is possible (Suda & Rani, 2013). The study briefly gives information about the existing system which uses barcode technology to identify books. Even though similar systems also often use barcodes, this project aims to replace the use barcodes with Radio-frequency Identification (RFID) technology. This is because of the main objective; allowing identification of a large amount of items at the same time is required, the study breaks down both the barcode and RFID technologies and their advantages and disadvantages to achieve this objective. Overall, RFID is preferred as it is more advantageous in such a case, so the system preferably utilizes RFID technology.

In a different study by Bhattacharya, a project was built called “Bluetooth Library Manager” (BLM) with the objective of reducing the heavy work force that is required for maintaining the daily work of a more traditional, non-online library system (Bhattacharya, 2014). But, this project has a different approach to this problem. Bluetooth technology with SQLite was used for connecting to the database. Thus, the connection between the server and the client is made only inside the library, as Bluetooth has a limited connectivity range compared to other network technologies. Besides, this enables users to also have functionalities such as phone calls, short message service (SMS) and email within the library with each other.

In a similar study, a project was developed with the aim of being used as a tracking system within the library, tracking books on shelves, to simplify similar processes and decrease the possibility of any mistakes made by the library staff, as well as to save time both for the library staff and the library users (Sangavi, Deepa, Surya, Vinumadhi, & Brindha, 2016). However, RFID is used alongside two models called Simulation Program with Integrated Circuit Emphasis (SPICE) and Organic Field Effect Transistor (OFET) essentially for both scanning and finding the books. Thus, overall provides flexibility and acts as “building blocks” with several other hardware components according to the author.

A different study was conducted with the aim to be a guide towards building a framework for employing the Internet of Things (IoT) in renovating the conventional library systems and to become “smart” online library schemes (Bayani, Segura, Alvarado, & Loaiza, 2018). Overall, many technologies are discussed such as Near-Field Communication (NFC), RFID and Wireless Sensor Network (WSN). NFC and RFID, which are two similar technologies in that they are both used for identifying

books and grant operation control to libraries and both achieve this by scanning items. With WSN technology, a non-centralized network is created between nodes in order to build a line of communication between them. Besides of the mentioned technologies, it is also stated in this study that when designing a smart library system, two components are in focus which are hardware architecture and software development, as the approach to these two should be well balanced.

In their study, similarly, authors Monali and Gaikwad designed and built a project with the objective of reducing the efforts of users when finding books and locations in the library (Monali & S.A., 2017). So, the system was developed using Global System for Mobile Communications (commonly known as GSM) and RFID technologies. This system also has a distinguishable approach on what network technology it uses and unlike many other similar studies, it uses GSM. The aim on using these specific technologies is stated by the authors as having an essential role on reducing human effort. Overall, the complete system works by sending messages to a GSM modem, then with a link of communication between GSM and RFID technologies, any search by a user about a book results on locating the relative book and informing the user accordingly.

A similar project was also built to replace an existing system (Karanth, Castelino, Nireeksha, Nazareth, & K, 2017). The authors briefly give information about how previously the librarian kept track of books physically in a library, but with the advancements of many technologies, nowadays these processes care carried out much easier and efficiently. In this study the system which is aimed to be replaced, was being operated manually by the librarian. The developed project aims to provide a mobile solution, thus it is developed for Android mobile devices. The objectives of the project

is to provide the information of books, download and borrow books upon admin permission and notify the users about their upcoming book return dates, as well as giving the librarian the ability to manage books and book information. Besides of this, technical information is given such as that the system is built using Java programming language and SQLite database management system, which are both popular due to Java being multiplatform and SQLite being free of charge.

On the other hand, in this study authors discusses the IoT principles in detail, mentioning the architecture, elements and layers of IoT (P., Bhattacharjee, & Gupta, 2017). According to the authors, IoT provides opportunities such as direct interaction with the physical world and that it increases accuracy and efficiency with reduced human intervention, which makes it important for a smart online library system. Besides of IoT, the authors make an emphasis on RFID, as well as other sensors that are used and their communication through IoT. Finally, it is stated by the authors that the developed system is connected through a central processor and a common, simple graphical user interface.

In a more recent research, a study was conducted which aims to make locating and borrowing books and similar processes quicker (Patil, Karande, Desai, & Pereira, 2017). Mainly, this project is designed around the IoT principles and besides of this, uses NFC technology instead of RFID or barcode technologies for scanning and identifying books. In addition, the details about hardware implementation that is required for the system is specified, which in summary includes NFC cards, hand held NFC card reader, an Arduino board and other components. Overall, makes an emphasis on the importance of hardware-software syncing in such systems and some

problems that might appear when creating a library system as well, which often most of them are related with hardware-software relationship and syncing.

In another study, the authors discuss the aim and functionality of a smart library system (Baryshev, Verkhovets, & Babina, 2017). Overall, this study by itself aims to develop the required set of theory and techniques which will fuse the smart library system with information technology and into educational environments. States that in time, it might be required to rethink the library's comprehensive aim with focusing on new upcoming technology. Besides of this, according to this study, the individual user is the centre of a smart library of a university. So, an online survey is carried out as a questionnaire in order to define user needs. Making a conclusion that a library also always has to be adjustable with quickly changing modern technologies and changing different needs.

Similarly, another project was also developed to replace an existing, already being-used system (Pandey, Kazmi, Hayat, & Ahmed, 2017). Even though the existing system can accomplish most of the required functionality and objectives such as tracking books, tracking book transactions, editing book information, etc. The existing system is mentioned to be a slow-working one, limiting the overall efficiency of the library and also decreasing the potential users' interest in using the system. In order to define a main objective, a methodology is followed which includes methods such as defining and refining difficulties, framing proposition and getting conclusions, and at last carefully testing the conclusions. Thus, the main objective of the project was determined by approaching the project with qualitative and quantitative analyses and conducting a survey was a big part of capturing general user needs and requirements. Other than that, the project was built accordingly and was built around RFID technology.

Another study makes an emphasis on what is meant by a “smart library”, what makes a library system “smart” and how can traditional library systems be transformed to smart ones (Cao, Liang, & Li, 2018). According to the authors, the “smartness” in a smart library refers to the ability of a system in which capturing needs, providing resources and services can be automated, in order to fulfil those needs. With this, the dimensions of a smart library are discussed, which are stated as; technology (IoT, data mining, sensors such as RFID or NFC, etc.), service (user-centred services), user-oriented (i.e. the user and the librarian) dimensions and each are considered to be concepts that are associated with smart libraries. Besides of this, it is implied that to make a library system smart, it is often required to integrate and interact a mixture of technologies. Finally, study discusses the importance of providing smart services, which includes both the users as the library user and the librarian, alongside a smart system. Meaning that, alongside the technology, the human factor is also important in making the system smart and work as intended.

The following study is developed for implementing a graphical user interface for an Android mobile library management system application (Shada & Ayu, 2018). It aims to enhance and improve the library with the application, by considering user needs. Besides, mentioned development techniques are requirement gathering, system analysis, graphical user interface design, implementation and unit testing. As well as this, the study also aims to understand user interface elements of a mobile application that affects user experience. So, the study considers user feedback and user testing results, and states that the development will be considered as the final system if it satisfies all mentioned feedback and testing.

Chapter 3

PROPOSED SYSTEM

The proposed system is a library management system (LMS) that is built for Eastern Mediterranean University (EMU) library. This system is built as a responsive web application, which can both used by students, lecturers and librarians. As a modern responsive web application, this system was built using a range of technologies and methodologies.

As mentioned, the LMS web application was built as a responsive web application which is dynamic, generally with languages such as HTML, CSS, JavaScript and NodeJS runtime environment, alongside with other frameworks or libraries. Alongside these technologies, a responsive design approach and design patterns were keenly followed.

In this section, the proposed system will be discussed from the technical perspective, explaining the technologies that were used and how the project was built in detail. Firstly, the technologies and methodologies that were used will be explained, then section by section how the application was built, and the project structure will be explained.

3.1 Frontend Technologies and Methodologies

3.1.1 Responsive Web Design

Previously, browsing the web was mostly carried on computers, viewed on a larger screen by comparison to smaller, mobile devices. On the other hand, an application was necessary to be developed for mobile devices. However, as the usage of mobile devices of many kinds such as smart phones and tablets has been increasing, nowadays almost every website or application is being developed with responsive web design approach (Giurgiu & Gligorea, 2017). Responsive web design is a concept where a website is designed with a purpose to visually adapt to the screen size of the device which the website is being browsed on (Wiener, Ekholm, & Haller, 2017). This is very important, because mobile development on itself can require developing the same application for multiple other operating systems, such as Apple's iOS and Google's Android. This approach mainly eliminates the need to develop more than one application for different device screen sizes. So, the same website can be browsed on a computer, tablet, mobile phone, etc., but with a completely adapted graphical user interface (GUI) when necessary on any type of device.



Figure 1: Responsive Web Applications on Multiple Devices (*Napper, 2020*)

There are many techniques and methods that are used to make a website responsive. One of the most known out of these is the use of grid-based layouts. These layouts are based on the use of flexible grids, which instead of having a layout fixed width, or designed to be placed on a fixed section of the screen, they are set with percentage sizes which are calculated according to the screen size and they are scaled accordingly. So, regardless of the screen size, with the help of a grid-based layout, the GUI elements can be displayed widely on a higher screen resolution but shrink and align on a lower screen resolution. Another common technique is the use of media queries (Jin, 2017). They are basically conditions that are followed depending on the current screen resolution, providing ways to implement different styling for different screen sizes and resolutions.

Currently, HTML, CSS and JavaScript are the essentials of frontend web development. Therefore, all previously mentioned techniques that are used on responsive web development are based on HTML, CSS and JavaScript. Because in core, HTML with the help of CSS, which is used to style the page, and a scripting language such as JavaScript, is all that is needed. So basically, with the combination of these technologies, developers have everything that they need in order to develop responsive web application frontend. However beside of that, these essential technologies bring a solid foundation for many other frameworks developing robust responsive web frontend, such as Bootstrap, React, Angular and Vue.js, which provide easier to maintain, a faster development and scalability for more complex and large applications.

3.1.2 Hypertext Mark-up Language Version 5

Hypertext Mark-up Language –better known as HTML- is often called the “publishing language of the World Wide Web” and it is currently the standard way of creating hypertext documents to be published on the internet (W3C, 2018). One of the main reasons that it became a standard for years is that it is developed to be cross-platform. Meaning that regardless of a specific platform, or a specific operating system, there are no requirements to view HTML documents with the use of a web browser.

Originally, HTML was developed in the early 1990s. But it was not published to the public until the mid-1990s, as HTML 2.0. As same as we know it today, HTML 2.0 brought the core HTML document structure, consisting of main elements such as head and body, but furthermore, included other now staple HTML elements such as the headings (h1 to h6), paragraph, ordered and unordered lists, hyperlinks, forms, with fields and form submission. In the following years, HTML versions 3 and 4 were released right after another in the same year.

HTML4 was released by the World Wide Web Consortium (W3C) and was a very successful version for a long time compared to HTML3. Early after it was released, Document Object Model Level 2 HTML (simply known as DOM) was developed and published by W3C. DOM provides both the access and the ability to update the elements or structure of HTML documents with programs and/or scripts.

After a decade of popular HTML4 use by web developers all around the world, HTML5 was published in 2010s. HTML5 is currently the latest version of HTML, expanding HTML even further than version 4. It brings many new features to web development, which are oriented more towards modern web applications. Some of

these features vary from new elements such as header or footer, video embedding, local storage to enhanced web forms, which overall are all vastly common nowadays. These features are very significant and important, because previously some of these functionalities were provided by third party plug-ins. Even though in perspective to responsive web design HTML is highly dependent on CSS and JavaScript, it is a proven language that has the greatest impact and usefulness on current modern web development.

3.1.3 Cascading Style Sheets Version 3

Cascading Style Sheets (better known as CSS) is a styling description for HTML documents (W3C, 2019). It is almost an essential part of HTML, thus an essential piece in web development. So alongside of HTML, web developers have been building web sites utilizing CSS since the earlier days of web development. By using CSS, HTML documents are supported visual-wise very extensively. Basically, almost every element of an HTML document can be styled by a CSS specification, such as colouring, lay outing, text styling, and so on.

At the beginning, CSS initially was published in mid-1990s by W3C. Even though there were other proposed methods and technologies to style HTML documents at the time, with the requirement of grouping many styling elements and even with multiple documents, CSS became more outstanding and popular. One of the requirements which was a main focus of CSS was to support inclusion to HTML externally. Meaning that besides of internal CSS implementation (directly in an HTML document), implementing and including an external CSS file to an HTML document was also made possible. Apart from that in CSS version 1, the base of its essential features were included, such as styling fonts, texts, colours, backgrounds and box

properties as in margins, paddings etc., but in more of an elementary sense, as well as the general standard CSS document structure.

With a rapid growth in web usage and development, CSS3 was published in the early 2000s. Similar to HTML, it had a clearer step forward on a more modern approach to web design. This is all due to a several new features of CSS3, such as tables, columns and media queries. In these features, one stands out as one of the most important features that affected responsive web design; media queries. Media queries in CSS3 are conditions that check a browser's size, in order to take actions in styling according to it. With media queries, properties such as a browser's current width, height and even orientation can be checked, thus, different styling can be created for different captured properties (device screen sizes). With these mentioned features of CSS3, developers are given a great range of options to develop and optimize fluid and responsive web applications.

3.1.4 JavaScript

The birth of JavaScript goes back to mid-1990s. In the beginning, it was developed to work on a specific web browser, called Netscape Navigator. But, as its popularity grew quickly, it has been adapted to other web browsers as well and got the nickname "the language of the web" (Miller, 2018). JavaScript is essentially an object-oriented programming language, normally, it is most known by being used as a scripting language on client-side web development. So, in this situation, it runs on a browser and it is used to interact with the GUI. This is in order to achieve more dynamic and interactive websites, dynamically adapting the display of a page controlling what appear on the page in certain conditions. So, this use of JavaScript is more known as client-side JavaScript.

JavaScript is known to be similar to other popular object-oriented programming languages such as C++ and Java. Thus, JavaScript brings many features that are expected to be on an object-oriented programming language, however in a lighter sense. But, principally, many concepts are supported by JavaScript. Various datatypes such as integers, strings, Boolean, as well as arrays, logical operations such as comparisons, conditional executions such as if statements and loops. So, the features that are brought to web development by JavaScript allow for broader web applications, especially for better user interaction. Many solutions could be provided to modern web development problems; handling user interaction with HTML forms and form elements, handling data distribution, even provide connectivity to servers for data submission and fetching.

On the other hand, even though there are some structural close similarities between JavaScript and high-level programming languages previously mentioned such as C++ and Java, there are some limitations to JavaScript in comparison. First of all, JavaScript has a lighter syntax, also known as 'Java-lite', not being as strict about syntax rules overall then the mentioned higher-level languages. Other than that, for example, unlike those high-level programming languages, JavaScript does not have capabilities that interact with the client device's operating system, such as the access to client I/O facilities, launching programs and reading or writing files.

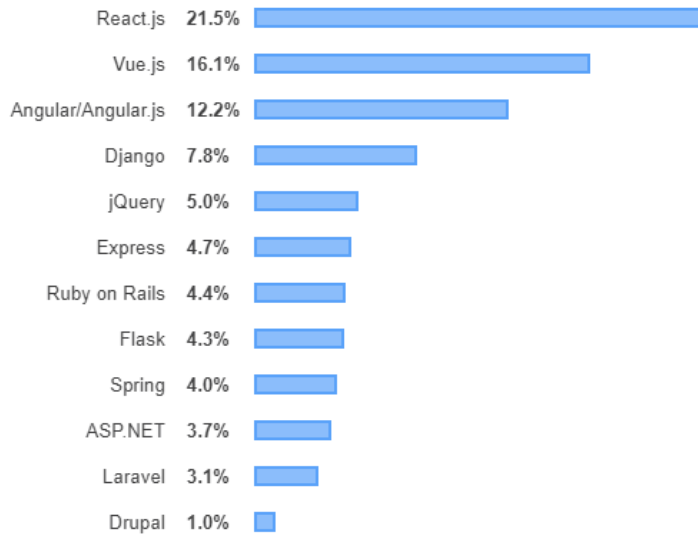
As JavaScript became the most common programming language for web development over the years, frameworks written in JavaScript also have been appearing as frequently. The need for a framework emerges as web applications become more complex and difficult to maintain. Definition of a framework can be stated as a group of written utilities, functions or libraries in a programming language, in order to

produce and contain reusable code for solving specific development problems. Nowadays a few examples as some of the most popular UI frameworks for JavaScript are React, Angular and Vue. Frameworks such as these, give developers the ability to quickly and easily create and maintain modern, complex, responsive UI for web applications.

Although JavaScript is mostly known as being a client-side programming language and to be run on a browser, developers sought ways of running JavaScript on other environments. One of the most apparent reasons for this, is so that developers could develop server-side applications using JavaScript as well. With that, a full web application would be possible to be developed with JavaScript, without developing the server application with another language such as PHP, Perl, or Python. So throughout, there have been ways to create environments, or engines which allowed JavaScript to run on other development environments. Nowadays, one of the most runtime environments for JavaScript is NodeJS and is widely used on many server-side applications.

3.1.5 React

As mentioned previously, building UI for web applications using a frontend JavaScript framework is very common nowadays. ReactJS (more commonly known as React) is one of the most used ones. React was developed by and currently being maintained by Facebook, initially to solve a UI problem their development team had recently faced at the time, then it was released as a library back in 2013 (Wieruch, 2019). The main objective of React is to create interactive and reactive UI, easily and quickly. This is aimed to be achieved by the creation of reusable fragment of UI elements, called “components”.



% of developers who are not developing with the language or technology but have expressed interest in developing with it

Figure 2: JavaScript Frontend Framework Use in 2020 (Makhija, 2020)

React controls an element that is called “The Virtual DOM”, a copy of the HTML DOM. Whenever there is a state change in a React component subtrees of nodes are rendered by this Virtual DOM accordingly instead of re-rendering the HTML DOM. The difference between the virtual DOM and the HTML DOM is compared and only the section or sections that contain changes are updated in the HTML DOM. This results in a smaller amount of interference to HTML DOM, thus making significantly lighter than some other similar frontend frameworks.

React components can be written in normal native JavaScript syntax. However, there is a special syntax that, can be used to create React components, which is called JavaScript XML (JSX). This syntax is similar to any mark-up language syntax, such as XML or HTML. So, as this enables creating components in a syntax that is similar to HTML, it is easier for both to write and read overall. Even though React components can be written without using the JSX syntax, it is not as popular and mostly is not

recommended. Because, besides of it being easier to read and visualize a mark-up structure, the lines of code that is written also reduced. Then, React has the ability to convert the JSX syntax into native JavaScript syntax. This conversion is carried out by a JSX transformer, which transforms JSX into JavaScript in the browser. Thus, the conversion does not occur at runtime, but before the application deployment, by the JSX transformer. An example of a JSX transformer is Babel.

Previously, it was mentioned that the reusable fragments of UI elements that can be created by React and are called “components”, elements called “components”, have many functions and properties of their own, ranging from controlling their data, display states and even can be used for debugging. Data that are contained in a component can be provided both internally, and externally. Meaning that the data could be initialized and be accessed within a component, which is called a state, but could also from outside of the component, which is called a prop. Besides of these, a component has a required method called “render”, mostly returning a single JSX layout, which is the components display.

3.1.6 Material-UI

Google keenly follow a consistent UI design language all across their developed platforms from Android, Google Drive, Gmail and even to YouTube. This concept in UI design is also often called a “UI design pattern language” (Doosti, Dong, & Deka, 2018). This refers to the consistency of design choices and following standards, which naturally causes an identifiability to users, making an application more user friendly. The overall design language and philosophy adopted by Google is called Material design, introduced by Google back in 2014. Google maintains an open source library for React, providing components that are designed as a part of Material Design.

Material UI aids a simpler, faster development, with well designed and maintained, also easily customizable UI components. This library contains a large range of UI elements; layout components such as containers and grid lists, input components such as buttons, checkboxes, radio buttons and text fields, navigation components such as menus, tabs and breadcrumbs, and many others such as dialogs, avatars, badges, icons, lists, or dividers.

3.2 Backend Technologies and Methodologies

3.2.1 NodeJS

Today, languages such as HTML5, JavaScript and frameworks that are being used for developing web applications allow developing complex applications. Also, with a responsive design approach, it is possible to develop applications that are similar to mobile native applications. Thus, providing and receiving a large data-stream in a stable nature over to the client-side application is essential. With this current complexity in current standard web applications, the concept of client-server side development integration, meaning that developing server-side applications with JavaScript is highly desirable.

NodeJS was first introduced in 2009, as a platform to run JavaScript on servers. Since then, it has been a solid choice for developers to develop server-side applications for complex web applications. NodeJS as a framework provides an environment for building scalable and fast back-end applications, along with good performance, running on Google's V8 JavaScript engine (Hota & Prabhu, 2014).

JavaScript Option Notation, also known as JSON, is a notation that is used for providing data to an endpoint. JSON is a subset of JavaScript and is very beneficial in

providing the UI application with data that is interpreted with JSON. Therefore, on the client-side application, the received data can be easily parsed and integrated. NodeJS works as a single thread application. Meaning that each request is operated on a single thread, instead of creating a thread for each operation like in other similar back-end environments. By this, a non-blocking I/O model is provided by NodeJS with asynchronous programming through JavaScript's asynchronous tools such as Promises and `async/await`. When a server is created with NodeJS, it listens a port for receiving incoming requests, when a request is received, it later is placed and handled in the NodeJS event loop. Thus, with NodeJS, an event-driven, asynchronous server-side environment is created.

3.2.2 ExpressJS

ExpressJS is a framework that is used on the NodeJS server-side applications and provides an easier use for NodeJS's main functionality (Hahn, 2016). With that, one of the primary objective of it is to be minimal, flexible, fast, as well as easy to setup and configure for back-end development.

One of the features of ExpressJS is called a "middleware" and is used for carrying out requests and responses to/from the server. A middleware can be received as services, which technically are functions. These middleware functions are in charge of managing both HTTP requests and responses. So with these middlewares, the requests are broken down into smaller pieces, handling a single section accordingly.

Another feature of ExpressJS which is called "routing", enables different requests to be handled with different request handlers. Specifically, specific routes are created, followed with a middleware function. When one of these routes are visited, the according function is executed. As a simple example, we can create a route such as

“/hello”, with a function which returns the value “Hello World”, when a HTTP.GET request is made to this route, the value “Hello World” is received. So, these routes can be used as endpoints which the client-side application sends requests to the server.

3.2.3 MySQL

MySQL is an open source, relational database system, which is relatively fast, stable and easy to learn, thus very popular. It has a wide range of features which are apparent in a quality, standard database system. As the name implies, MySQL has Structured Query Language (SQL) support, which is a standard language of querying a database administration.

As many others, MySQL is actually a client/server system, meaning that the transaction between the database and the user is carried out between the client application and the server where the data is stored. Thus, database transactions such as data queries, data saves or edits are sent from the client and are handled on the MySQL server. Besides, MySQL is easily integrated with NodeJS as well.

3.2.4 Design-Based Implementation Design

Similar to other ways of design research, DBIR has the objective to develop practical solutions to continuous educational problems. It aims to achieve this objective by designing and studying innovations in the context of their implementation. The difference that separates DBIR from other different forms of design research is that it focuses on implementation. Also, the use of implementation research for improving iterative design. An essential focus of DBIR is not about what works but is more about how effective programs could be made to work in a variety of context and for diverse groups of people in or out of school. As an iterative, collaborative and practice focused

method, the developers are brought together as equals, in order to develop, test and scale programs to enhance teaching and learning.

3.3 Software Development Architecture

3.3.1 Development Life Cycle

In software engineering, the set of different processes that are carried out in order to successfully and efficiently design, develop, test and maintain software is called software development life cycle (SDLC) (S, 2017). In a SDLC, there often are specific rules that are followed in each step, or an overall idea that is followed. Some examples of these steps are requirements gathering, design, implementation, verification, testing, maintenance, etc. Even though the followed steps are similar, how they are carried out differentiates the SDLCs from each other.

Over the years, there have been various theoretical SDLCs, which software developers and software development teams have been using. Some of the most popular SDLCs are the waterfall model, V model, iterative model, spiral model, agile model, etc. Waterfall model is the most traditional SDLC, but it is a model that has very strict rules which are not very suitable for most of the modern software development practices. So, nowadays software development teams mostly use SDLCs such as agile model or iterative model.

3.3.2 Iterative Model

As the name implies, iterative development as a method focuses on developing software in iterations, as divided in smaller parts (Nugroho, Waluyo, & Hakim, 2017). This model does not depend on requirement gathering. Instead, with a small amount of requirements, development can be started. With each iteration of development, the developed application expands in small versions, with steps such as design,

implementation and testing being repeated. So, with each iteration, new functionalities is added to the application. The development is continued as such until the final version of the application is fully completed.

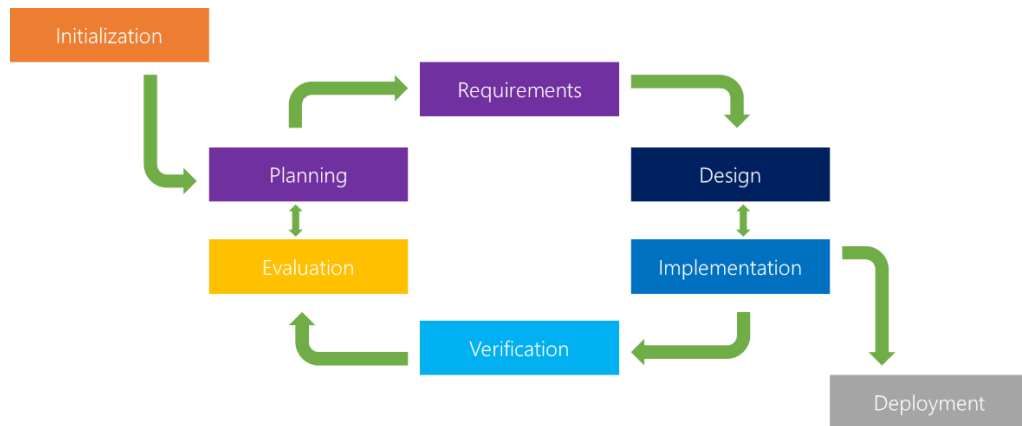


Figure 3: Iterative Model Cycle (*Powell-Morse, 2016*)

There are many scenarios where the iterative model can be preferred. The most apparent scenario is whenever the initial requirements are not very clear or the most important requirements are defined however smaller requirements can be developed in following iterations. But besides these, there are other suitable scenarios for the iterative model where there is no time constraints, or when the required skill set is not met yet but being learnt during the development. Hence, the iterative model provides a work flow where working, usable versions of the project can be produced from the earliest completed iterations.

3.3.3 Scrum

In agile development environments such as incremental model, scrum is seen as an incremental and iterative method, or also referred as compression algorithm (Sutherland & Schwaber, 2011). In scrum, the development is structured in cycles that are called sprints. These sprints are aimed to be completed in a specific time-frame,

with no pauses. As they are timed, at the end of that specific time-frame, they end regardless of the task being completed or not and they are never extended. Thus, scrum is a method that can be used to control and structure iteration cycles when following the iterative model.

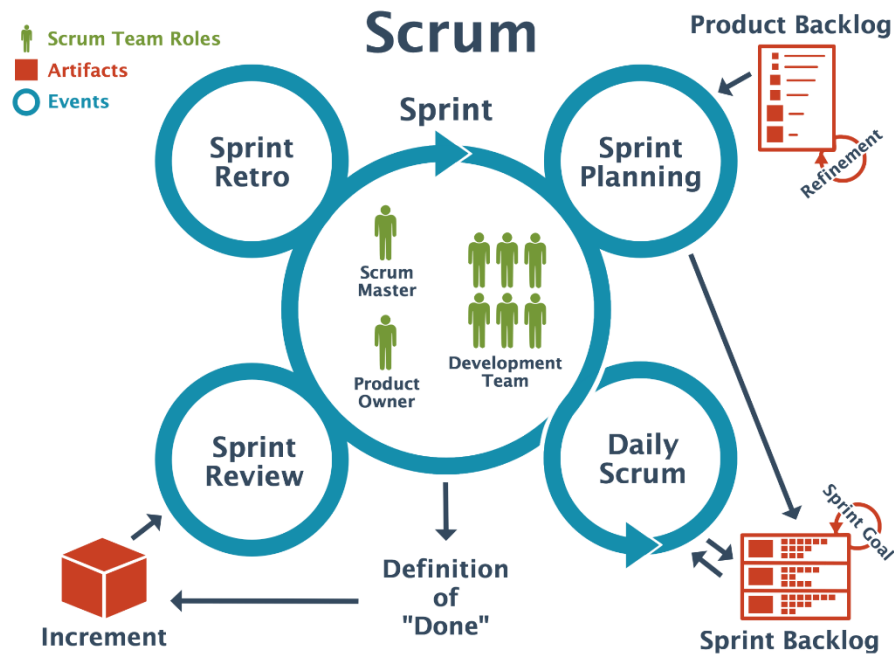


Figure 4: Scrum Framework Visualization (*Job, 2015*)

On a project management sense, the control structures that the scrum methodology provides are very beneficial. Hence, managing the project from the perspectives of time and resource management becomes more extensive. So, overall, scrum as a methodology is very flexible by design.

3.4 Proposed Library Management System Design Model

The Library Management System, also called LMS (standing for 'library management system'), is a responsive web application that is designed and developed for managing and browsing the EMU library, in the educational setting, both for library users and the library staff. And as many people know, such an application will enhance the

quality of education in students' or lecturers' busy everyday lives and it is equally vital that library staff carry out their work easily and efficiently.

Guest users, which are non registered users, can search for a resource (book, article, magazine, news paper, etc), view the details of a resource, search for an author, view resources by an author and search the resources on a shelf or a floor. Additionally, library users users can login to their LMS account to make a borrow request to a resource, borrow the resource, reserve a resource if it is currently borrowed by another user and also, track any of their borrow or reserve processes that are currently held by them.

On the other hand, the application provides more functionalities for the library staff. Besides searching and veiwing resources, resource types, authors, floors and shelves, the library staff also have administration privileges in order to manage these items by adding, editing and deleting resources, resource types, authors, floors and shelves. The library staff users can also check and approve/reject borrow requests, as well as viewing any borrow or reserving process.

There is a user manual prepared for the system that can be checked in the appendix section of this thesis, where all the mentioned functionalities are stated with screenshots provided.

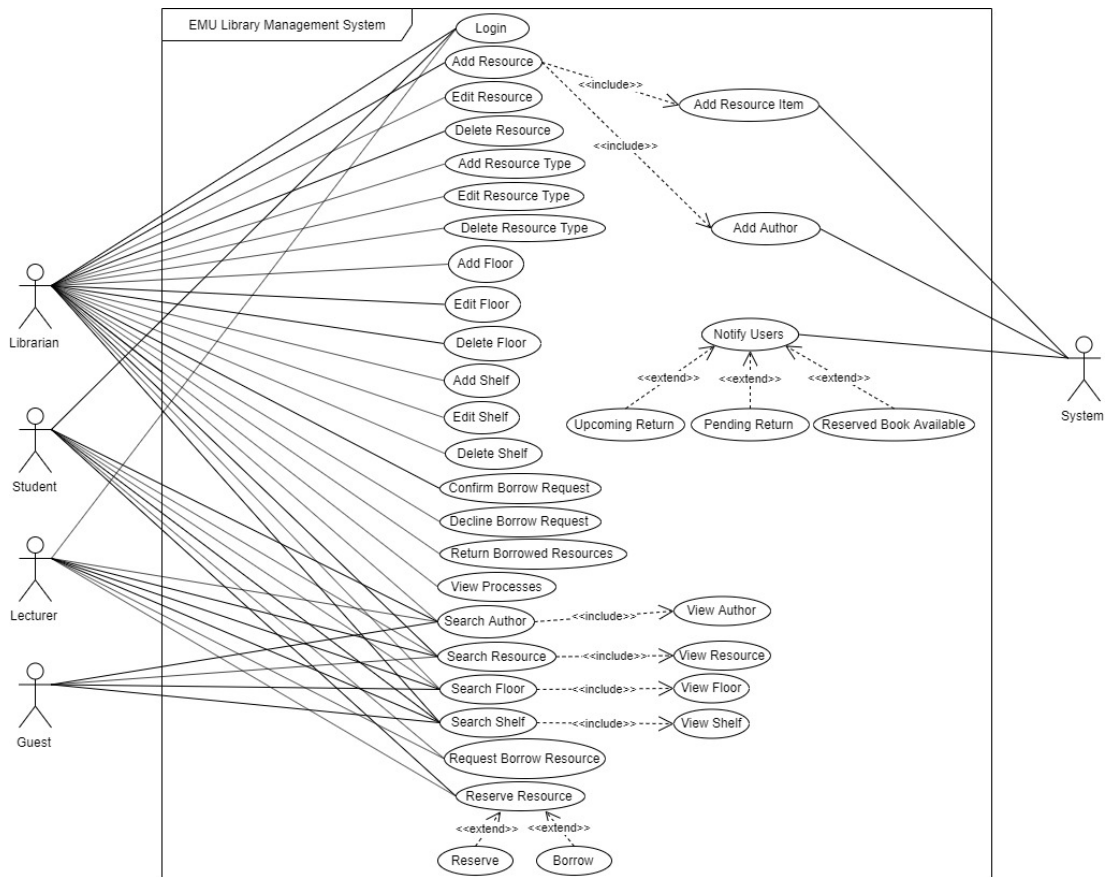


Figure 5: EMU Library Management System Use Case Diagram

Nowadays, dynamic web applications based on client-server model that work with displaying and managing information consist of two sections, which can be seen as two different applications. Those applications are called frontend or client side and backend or server side applications. Frontend applications run on the user's internet browser and mainly handles displaying the GUI. But besides, it also handles requesting the backend application in order to carry out any process or receive information and display the information that is received from the backend application. Contrarily, the backend application has communication to the database, where query requests are sent through, then process and provide the information that is received by the frontend application.

Client-Server Model

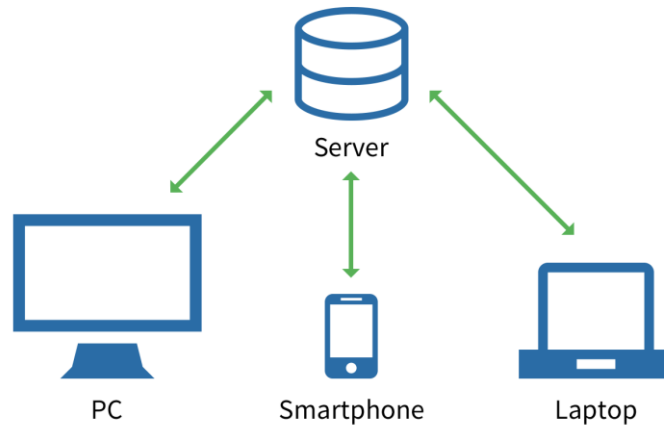


Figure 6: Client-Server Model Visualization (*Christensson, 2016*)

The developed LMS was built by using many different technologies and approaches. The frontend application was build using technologies such as HTML, CSS, JavaScript and React framework, with a responsive design approach. Whereas, the backend application was built by using JavaScript in NodeJS version 10.16.3 environment and ExpressJS framework version 4.17.1 alongside MySQL rational database management system version 8.0.19.

Overall, the LMS were built with a framework approach in mind, both for frontend and backend applications. In this study, the framework approach refers to have a certain way of structuring the project which aims to have a faster, easier and robust development. To achieve this aim, both of the applications are structured in a way that have many processes and functionalities automated and reused. The details of this structure and approach will be discusses further for both applications as mentioned before, in their respective sections.

3.4.1 Backend Application

In the LMS project, the backend application consists the technologies and methodologies that are necessary for processing requests and then producing and providing a response to the client. The code that handles this process runs on a server, which includes the logic that processes the request and provides the response accordingly. Since MySQL is used in LMS, the database is located on a cluster, and the backend application has a connection to that MySQL database, that the LMS data is stored.

The backend application consists all the logic that defines how a request is responded. With the use of special ExpressJs functions, routes are created where the client can send requests to. These functions that are run on the backend are called middlewares. A middleware is usually a application as mentioned, which runs whenever the server receives a request or is returning a response. By using these middleware functions many processes can be handled on backend. To give a few examples, the request objects can be modified, the database can be queried, or process the information that will be sent as a response.

The LMS backend application runs on a server which is created with ExpressJS framework on NodeJS. The server handles the base routing of the application, which are also handled by using ExpressJS. The LMS database is created using MySQL. Then, a connection is established to this database from the backend application.

3.4.1.1 Database Tables

The database tables of the LMS is explained as follows:

- **Resource:** This table represents the items within the library. These resources can range for many types, such as books, journals, magazines, etc. The resource

table has the fields 'id', 'isbn', 'name', 'language', 'publisher', 'publish_date', 'page', 'summary', 'subject', 'cover_image', 'back_cover_image', 'author', and 'shelf'. The 'author' and 'shelf' fields are foreign keys, pointing to author and shelf ids.

- Resource item: This table represents the resource items within the library. For example, a book could have two copies within the library and a digital version. The resource item table has the fields 'id', 'type' and 'resource'. Both the 'type' and 'resource' fields points to resource type and resource ids.
- Resource type: This table represents the resource types, as previously mentioned in th resource table, types such as a book, journal, magazine, etc. This table has the fields 'id', 'name' and 'hard_copy', the 'hard_copy' field representing if the resource is a hard copy or a digital copy.
- Shelf: This table represents the shelves within the library. It has the fields 'id', 'name' and 'floor', the 'floor' field pointing to the floor id.
- Floor: This table represents the floors within the library. It has the fields 'id', 'name' and 'number.
- Author: This table represents the authors that can be saven in LMS. It has the fields 'id', 'full_name' and 'date_of_birth'.
- Process: This table represents the processes that can be taken within the LMS. These processes can be borrowing or reserving a book. It has the fields 'id', 'date', 'start_date', 'end_date', 'status', and 'type'. The 'date' field is the date that the process is issued, 'start_date' is the date that the borrow or reserve starts and the 'end_date' is the date that the borrow or reserve ends. The 'status' field states the status of the process such as 'Approved', 'Declined', or 'Returned', and the 'type' fields points to the process type id.

- Process type: This table represents the process types within the LMS, as recently mentioned, the process types are reserve and borrow. It has the fields 'id' and 'name'.
- User process: This table represents the processes that the user processes that are currently taking place. It has the fields 'id', 'user', 'process' and 'resource', all pointing to the respective ids.
- Notification: This table represents the notifications that are shown to users about their past due dates, upcoming due dates, or recent borrows. It has the fields 'content', 'title', 'type', 'user' and 'process'. The 'content' field contains the message of the notification, the 'title' field contains the title of the notification, the 'type' field contains the type of the notification such as 'error', 'warning', or 'success' and it is used for showing the according notification icon. The 'user' field points to the user id, and the 'process' field point to the process id.
- User: This table represents the users that use the LMS. It has the fields 'id', 'name', 'surname', 'username', 'password', 'user_id', 'date_of_birth', 'department', and 'type'. The 'type' field points to the user role id.
- User role: This table represents the user roles within the LMS. It has the field 'id' and 'name'.

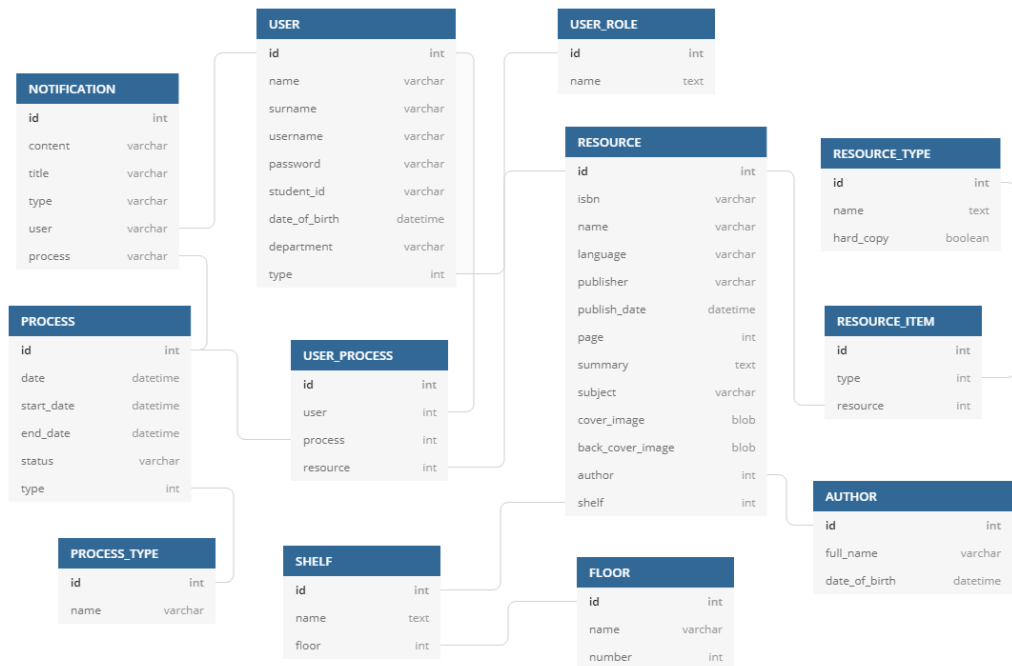


Figure 7: EMU LMS Database Relationship Diagram

3.4.1.2 Routes

The routes that are created by the help of ExpressJS can be visited with a URI that is defined. These are structured exactly as a URL for example. By that, the client sends the requests and receive responds by pointing to these routes.

In the LMS project, the use of these routes are structured, that are similar to models in the Model-View-Controller software architecture, which is commonly known as MVC. To briefly explain, the ‘model’ in the MVC architecture corresponds to the data structure, that is dynamic and independent from the other sections of the application, such as the GUI.

Thus, each table in the database has a route that is created for it, which includes the information that is needed to handle the base create, read, update, delete operations

(also known as CRUD operations) as well as other helper operations such as getting view tables data, searching query responds and creating database tables automatically.

3.4.1.3 Base Route

The LMS routes are inherited from a class called 'Base Route'. The Base Route defines a data structure for the inheriting classes, which are called 'Modelled Routes'. Also, the methods previously mentioned, CRUD operations and other helper operations are defined in the Base Route. These base operations in the Base Route are all general and they are used by all the Modelled Routes. The defined data structure is used both in the base operation methods, and the specified methods in the Modelled Routes.

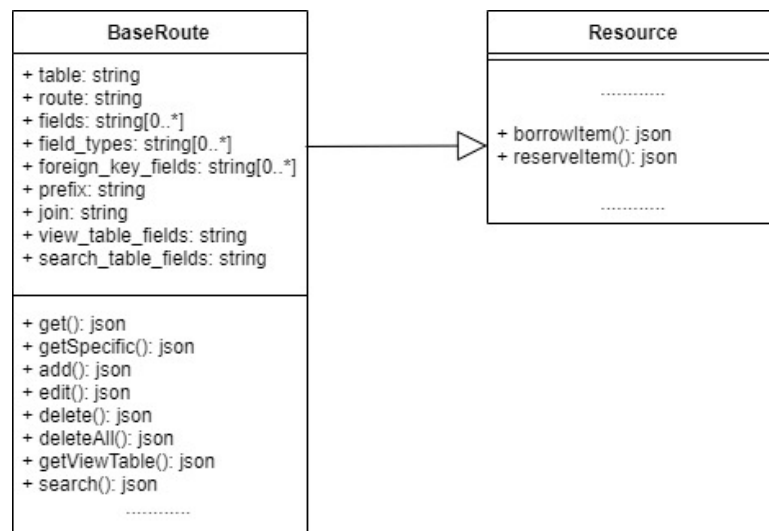


Figure 8: An example of Base Route and Modelled Route Relation

The data structure of the Base Route contains the following data; table, route, fields, field types, foreign key fields, prefix, join, view table fields and search table fields.

- Table: The 'table' field is the table name of the route in the database. This field is used by the CRUD operations and the helper operations.
- Route: The 'route' field is the name that is given to the route, which used in the route URI, similar to the example <http://server/route-name>.

- Fields: The 'fields' field is an array of strings that contain the field names of the route's database table. These fields are used by the CRUD operations and the helper operations.
- Field types: The 'field types' field is an array of strings that contain the field types of the route's database table. These fields are used in database table creation.
- Foreign key fields: The 'foreign key fields' field is an array of strings that contain the field names of the foreign key fields that are in the route's database table. These fields are used when searching for items, in such where they are added to the 'SELECT' section of a select-from SQL query. By this, the foreign keys get included when searching for items, producing a more throughout search result.
- Prefix: As a part of the structure, the database table fields begin with a prefix. Thus, the 'prefix' field stands for the prefix of the route's database table fields. This field is used when getting a specific value, where the ID of a field is obtained as 'prefix_ID', similar to 'USER_ID' as an example.
- Join: The 'join' field is the join statement that can be passed to a SQL query. This field is used by the helper operations, such as search and getting view tables.
- View table fields: The 'view table fields' field is a string that contains the field names that will be returned in a view table response. By this field, what fields are shown on a view table can be customized.
- Search table fields: The 'search table fields' field is a string that contains the field names that will be use returned in a search response. By this field, what fields are returned in a search response can be customized.

3.4.1.4 Modelled Routes

The Modelled Routes which all inherit the Base Route, include the data structure, CRUD operation methods and helper methods which are inherited from the Base Route. Thus, a Modelled Route can be created for representing any table in the database. Then, any CRUD operation, or helper operation can be carried out. Apart from the base operations, any required specific method is written in a Modelled Route's own class. So, by following this general structure, the development phase can be completed quicker, more efficiently and as well organized.

3.4.1.5 Application Programming Interface

An "Application Programming Interface" (API) is defined as a group of methods and procedures, that serves the information to the client application from the backend application. So, in LMS, all the methods that are defined in both the Base Route and the Modelled Routes represent the API as a whole. Every method is represented in a specific structure, that can be accessed from the client application and send requests to or receive responds from. These specific address structures are stated as `http://server/route-name/method`. So, an example can be given as `http://server/BaseRoute/USER/getAll`, which would return all the data in the 'USER' table as a response. Any other specific method that is aside from the base operations could be reached such as `http://server/USER/getAllUserDepartments`, which reside from the Modelled Route's address, instead of the Base Route's. Thus, the collection of these addresses represent the LMS API.

3.4.1.6 Authentication

In LMS, there are restricted processes that has to be authenticated in order to proceed. Such as borrowing a book, which a user has to be logged in. Or, viewing all the borrow/reserve application processes, which a user has to be logged in as a librarian

user. So, logging in the user and checking the user role authentication operations are handled on the backend application. To carry out these authentication operations, middlewares are created and used respectively. To any authenticated operation similar to what is given as an example previously, these authentication middlewares are used. These authentication middlewares are defined as in the following:

- Login the user: The database is checked if the user exists, then it is checked if the password matches. If the user exist and the password is correct, a token is created for the user and is returned to the client. This token is generated from the user id and a specifically defined secret, which means that each token is unique in each login.

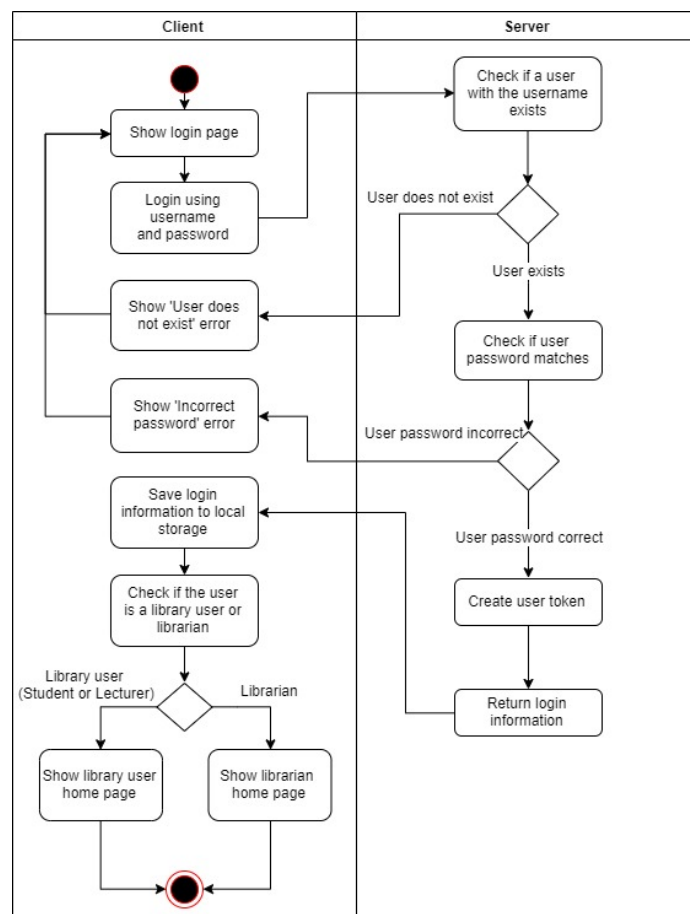


Figure 9: Login Activity Diagram

- Verify token: Any operation that requires the user to be logged in requires a token to be provided, that should be generated on user login. This token then has to be verified in order to proceed to a login restricted operation. If a token is not provided or fails to be verified, the operation does not carry on and an according respond is sent to the client.

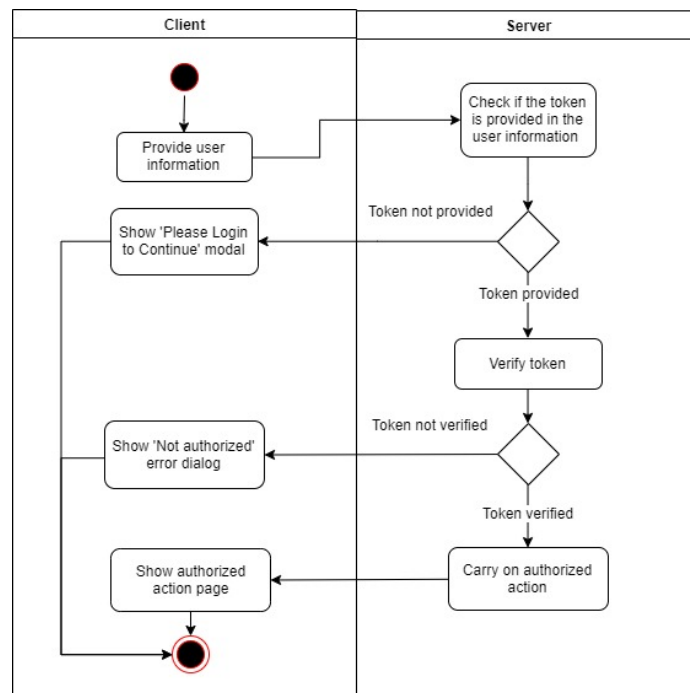


Figure 10: Verify Token Activity Diagram

- Is student or lecturer: The user role is checked in the database, if the user role is not student or lecturer, a response is returned accordingly and the operation is not carried out. But, if the user role is student or lecturer, the operation proceeds.

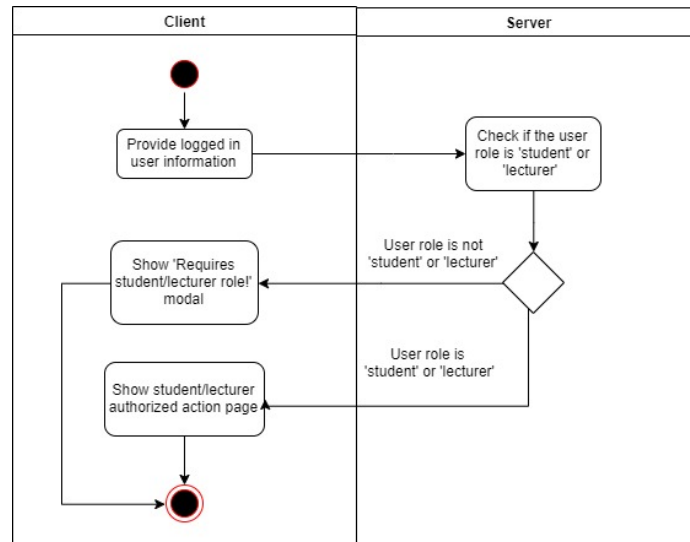


Figure 11: Check Library User Role Activity Diagram

- Is librarian: Exactly as the student/lecturer authentication, the user role is checked in the database, if the user role is not librarian, a response is returned accordingly and the operation is not carried out. But, if the user role is librarian, the operation proceeds.

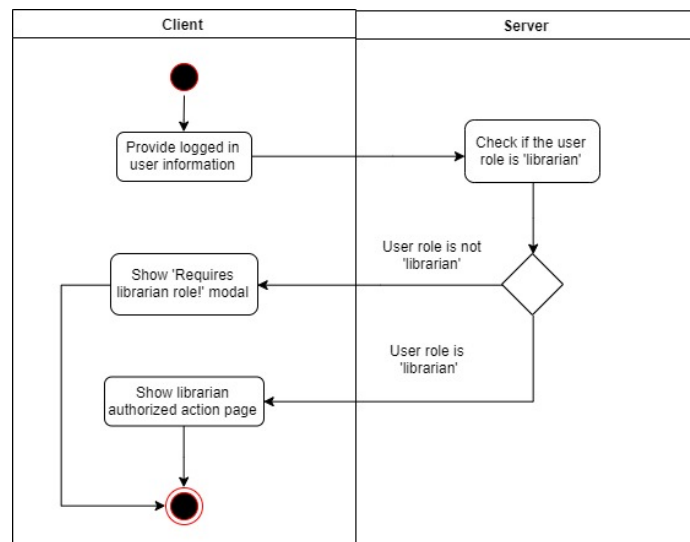


Figure 12: Check Librarian Role Activity Diagram

In conclusion, when performing authenticated operations, these actions are followed by using the respective middlewares depending on the authentication that is required. As an example, these actions are taken in the following order when a borrow request is received; “verify token, check if the user is a student or a lecturer, perform the borrowing operation”.

3.4.1.7 Security

For any web application, security is an important aspect. If an application is not secure, the application can be exposed and open to attacks. The ‘Authentication’ section that was recently mentioned, provides a base level of security as the users must be authenticated whenever they are carrying out operations that are confidential and has to be authenticated.

Apart from that, the backend applications are mostly targetted by the attackers especially by the HTTP headers. The reason for this, is the HTTP headers’ potential vulnerability, as they can leak sensitive information. These sensitive information can be used by the attacker to attack to a web application with attacks such as cross-site scripting (also known as XSS) and other attacks called ‘click-jacking’ or ‘sniffing’ attack (Vogt, et al., 2007) (Huang, Moshchuk, Wang, Schechter, & Jackson, 2012) (Thakur & Chaudhary, 2013). To secure the application against such attacks, a module called 'helmet' of NodeJS module is used in the LMS backend application, which handles the securing of the HTTP headers. Helmet provides many security measures, a few most important of these being:

- X-Frame-Options: Used for preventing the ‘click-jacking’ attack.
- X-Content-Type-Options: Used for preventing the ‘sniffing’ attack.
- X-XSS-Protection: Used for adding protection against XSS attacks.

Another general measure of security in LMS is provided by encrypting the user password. The encryption is done by bcrypt password hashing, using the NodeJS 'bcrypt' module. Bcrypt is a password hashing algorithm, based on the Blowfish cipher (Sriramya & Karthika, 2015). On the backend application, this module is used for hashing the password, before saving it to the database. So, the user password is not saved to the database as being exposed. When logging in, the bcrypt module is used again, for comparing and verifying that the user password is matched.

Apart from these, MySQL has backup and restore functionalities. One of those is to exporting the database to an SQL file. Thus, LMS database can be backed up by exporting it to a file, which can be easily restored by importing that exported file. The exporting process can also be automated by using many different tools. Also for automating the backup import, MySQL's own 'mysqldump' command can be used alongside with the Linux 'cron' utility which is a time-base job scheduling utility, to import the backed up file with defined intervals such as every 24 hours (Kirch & Dawson, 2000).

3.4.2 Frontend Application

In the LMS project, the frontend application composed of technologies and methodologies that are necessary for building the GUI, that aims to display the information that is provided from the backend application, as well as submitting forms for adding and editing. The code that handles these processes all run on the client side, which is the user's internet browser. Overall, React is used for routing client-side and rendering pages that are written with JSX, and requesting the API is handled by JavaScript fetch interface. Thus, all the pages that the user navigate and interact with the LMS is built with React, version 16.13.1.

Similar to the backend application, the frontend application is also structured in a way where models are utilized. With a structure utilizing models, it is aimed to handle sending the base CRUD operation requests to the API, also automatically create forms and view tables.

3.4.2.1 View Models

Frontend models, which are called 'View Models' are inherited from the 'Base View Model'. In order to utilize the model structure, the models are created representing each Modelled Route in the backend application. By doing this, the URI that is created in the backend application can be requested from the API. In the Base View Model, a specific data structure is defined. With this data structure, it is possible to create requests for the API accordingly.

In the Base View Model, basic methods are created that can be used for generating requests to the CRUD operations. These requests are generated with the use of the specific data fields that is defined for each View Model. Thus, with any View Model that is created, CRUD operations can be carried out quickly and easily. Apart from the Base View Model and the base operations, more specific operations can be defined in any View Model.

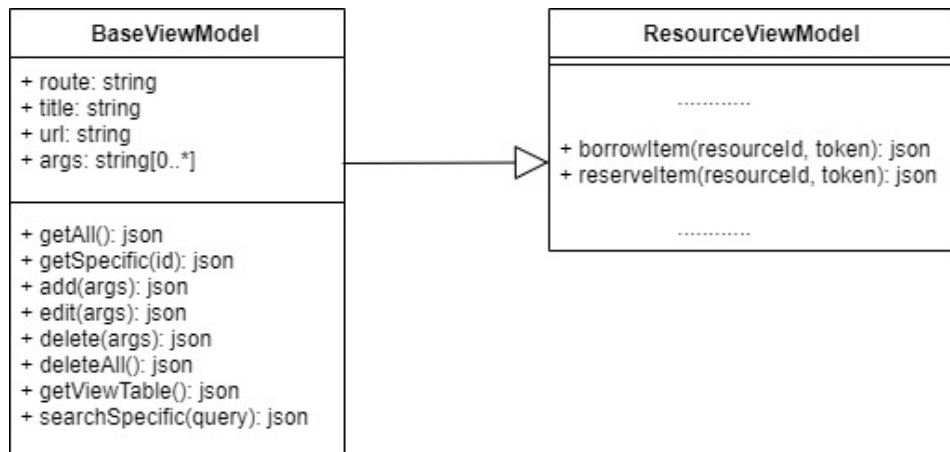


Figure 13: An example of Base View Model and View Model Relation

In a page where an action is required to be taken related with a View Model, for example seeing the table of resources within the library, a View Model should be defined accordingly in order to use the required action. So as in our example, in a page similar to ‘View Resources’, a Resource View Model should be defined, where from that View Model the ‘get all’ method is called, and all the resources are received from the backend application. Received as a JSON object, this information is set to a React state and applied to the GUI. Similar to this idea, any base operations can be used for the View Models, which are forms, search and view tables. Apart from the base operations, any other specific operation can be used after being defined in the according View Model.

The Base View Model contains the data fields ‘route’, ‘title’, ‘url’ and ‘arguments’.

These fields can be explained as follows:

- **Route:** This field is a string which represents the route that is defined in the backend application. This field is used when creating the URL that will be requested.

- Title: This field is a string which will be used for titling forms, search results, and view tables.
- URL: This field is a string which represents the URL, and it is created by concatenating the server name with the route.
- Arguments: This field, is an array of JavaScript objects, which represents each field of a Modelled Route in the View Model. These ‘arguments’ are used when automatically creating forms and view tables. In each JavaScript object, a field name, label and value, alongside other helper fields. These fields are not mandatory and used whenever in need, in such operations such as forms and view tables.

3.4.2.2 Forms

Forms are a fundamental functionality in many web applications. And also as important in LMS, forms are a base functionality in the frontend application. Thus all the operations of generating, validating, and submitting the form are done in the frontend application.

The forms are generated by checking each argument defined in a View Model. For each argument, related helper fields are checked in order to both generate and validate the form. The ‘hidden’ field is checked to determine if the argument is included in the form as a field, the ‘required’ field is checked to determine if the field is allowed to be submitted empty or not. Also, a boolean field called ‘number’ is used to set a field into only allow numeric values, and a field called ‘digit’ is used for setting a maximum digit. When the form is submitted, the base ‘add’ operation is called for the respective View Model, with the according fields.

3.4.2.3 Search

Similar to a search engine, the user can make a search within the library with the search functionality in LMS. This search is usually made for a resource, but authors, shelves and floors can also be searched. Thus, the user can view any resource, author, shelf, and floor information by searching. A specific response is received from the backend application containing the information of each search result. So, views are built specifically according to this request. The search functionality is a base operation in the LMS frontend application, which means that it can be used for any View Model.

3.4.2.4 View Tables

View tables are an effective way of displaying information to the user. This is another base functionality of the LMS frontend application, which is implemented by using the base structure and Material-UI tables. Thus, view tables can be created for any item within LMS, such as resources, floors, shelves, etc.

By requesting the table view data from the API for a View Model, data is responded according to Modelled Route's view table fields from the backend application. With the received information, the view table is generated automatically, using the Material-UI table component. These view tables also have features such as column sorting, increasing pagination, increasing the number of rows to display, as well as searching and filtering any row with a search in real time.

3.4.2.5 Authentication

As mentioned previously, there are tasks in LMS which require authentication. Even though some of the tasks such as searching a resource, checking resource details, etc can be taken as a guest, other tasks such as borrowing or reserving a resource requires the user to be a student or a lecturer.

Authentication on the frontend application is handled by as described; when the user logs in successfully, the backend application responds with the user information which contains the user id, name, surname, student id, username, birth date, department, role, and the access token.

The information that is received from the backend application is then stored locally. When an action is taken that requires authentication, the access token is also included to the request as a parameter to be matched and verified on the backend application. For example, the user information is checked before borrowing a book, if the user is not logged in, he/she is prompted to login to their account and retry borrowing.

Any page that authentication is required for access also is checked by the system if the user is logged in or not by from the local storage, if not, the user is redirected to another page such as the login page. As an example, all the librarian user pages can only be seen if the user is logged in as a librarian, otherwise they will be redirected to the login page. On the other hand, because the student and lecturer users can view some pages as a guest, if a guest tries to see page which requires authentication such as their processes page, guests are redirected to the previous page that they were in.

3.4.3 Testing

In software engineering, testing is conducted in order to gather or provide information about a software system's quality. There are several ways and techniques on how software testing is carried out. These techniques are generally guided towards both finding bugs or errors in the system and measuring the usability of the system. In brief, it involves running a section of the software, aiming to make sure that it meets some certain factors such as meeting the requirements, responding correctly to inputs, performance and proper usability.

Some of the most used software testing techniques are called black-box testing and white-box testing. Black-box testing is carried out without accessing the source code, meaning that the tester is aware of what the software is supposed to do, but does not have any technical knowledge about it. On the other hand, white-box testing is carried out by testers that have the knowledge of inner workings of the software, thus conducting tests in a unit level. As a combination of these two techniques, there is another technique called gray-box testing. In grey-box testing, the test is conducted by a tester who have the knowledge of the inner workings of the software, defining test cases based on that knowledge. This enables the tester to execute those test cases similar to black-box testing, but test the qualities of the software more thoroughly.

As mentioned previously, LMS was developed iteratively. Where in each iteration, it is aimed to bring a new feature to the system. As I worked alone on the project, I did not have the chance to conduct black-box testing. Thus, I carried out grey-box testing at the end of each iteration before advancing to a new one, making sure that the features of the iteration meet the requirements, works as intended, and there are no apparent bugs or errors.

During grey-box testing, my strategy was to identify inputs, expected outputs, user scenarios and defining comprehensive test cases regarding these factors. Then, each test case was tested by me, either passing if there were no problems, or failing when faced any problems as result. At the end, any problem, error or bug that found by failing tests was fixed. The goal of this strategy was to make sure there were no bugs before moving on to a new development iteration, as stated earlier.

3.4.4 Development Environment

In web development, the tools to be used are generally not very strict. It often depends on what the developer would prefer to use. For writing JavaScript code, only a text editor is necessary when developing web applications, as it does not require compiling. For that reason, even a lightweight, simple text editor can be used, however nowadays there are many integrated development environment and editor choices such as VSCode, Sublime Text, or WebStorm, each having different options such as keyboard shortcuts, error detecting, code formatting, and so forth. But as mentioned, it is completely up to what to developer prefers to use as a code editor.

Besides of an editor, other tools might also be used in order to make the development easier, and more efficient during web development. One of the most important of these tools is a version control tool. A version control tool is a tool that is used for managing the changes in development, such as Git.

Tools such as a visual database design tool or an API client are other tools that could be used during web development. A visual database design tool is a tool that is easily used to build, change, query databases and update data through a GUI. Whereas an API client is a tool that is used for testing API calls, by sending calls, then receiving and presenting the response. These tools make sending complex or repeated test API calls easier and efficient, as requests can be saved, or edited.

While developing the LMS, I used WebStorm IDE, Git with GitHub for version control, MySQL Workbench database design tool for MySQL, Postman for testing the backend API, on Windows 10 operating system.

3.4.5 Replacing the Already Existing System

In software engineering, systems which are outdated, but still in use are called legacy systems. Over the time, already existing systems become outdated by current time standarts, which is both difficult to use by the users and dificult to maintain by the developers. Thus, techniques and tools exist that are used when migration and modernization needed.

Legacy system migration is a multistep process, which is implemented by using many techniques and tools. Some of the legacy system migration techniques are legacy system understanding, target system understanding, target system development and deployment and provisioning of target systems (Ganesan, T.Chithralekha, & Rajapandian, 2018). The tools that are used in legacy system migration are often for data analysis and reverse engineering, such as Software Refinery, DB-MAIN, Apache NiFi (Bisbal, et al., 1997). By utilizing these techniques and tools, a sufficient and through migration can be achieved.

In order to migrate the already existing system into LMS, data such as resources and users has to be transferred to the LMS database. Even though the data and database structure is not exactly the same, they are similar, since library systems mostly have a commonly known structure. The process of integrating the required data to the LMS database can be achieved either by using previously mentioned tools or by writing simple, specific scripts by further analysis of the data.

Chapter 4

CONCLUSION

Today the advantages of a modern, responsive web application are evident. As an application that can be accessed through a web browser, can both be used on a computer, or on any other mobile device with full compatibility and this provides more flexibility than a traditional non-responsive web application or a native mobile application. Combined with a straightforward, easy to use, user centric design, a library application in an educational setting can be improved tremendously.

It is aimed to develop an application with an objective to ease the use of the EMU library, with more of a user centric and responsive design, both for students, lecturers and the library staff. Overall, the library users can mainly search resources within the library, can view the details of the resources, and carry out borrow or reserve requests on resources easily, where the library staff can respond to borrow or reserve requests and check the current borrow and reserve processes.

The LMS aims to provide the library users a versatile use as part of their education and busy everyday lives. Thus, it is focused to be designed with a modern approach, methodologies, and technologies. In addition to this, the LMS also provides a solution to the tedious process of locating, borrowing or reserving a resource, keeping track of the borrowing or the reservation using a modern responsive web application, offering an overall simpler way to use and manage the EMU library. The LMS brings many

advantages both for students, lecturers and librarians of EMU. Users of EMU library can view and use the whole system anywhere, anytime and with any device, thus making these processes fully online. Besides, the library users get notified about any reserve or borrowed resource whenever they have a closing pick-up or return date and keeping track of their reserve or borrow processes easily online. On the other hand, librarians can manage keeping track of the library stock fully online, carry out borrowing operation fully online as well without the need of paperwork, and have a safer environment as only the EMU students and lecturers can reserve or borrow resources. Below, a table of comparison is shown briefly comparing LMS to the existing system, and some of the mentioned related works.

Table 1: LMS, Existing System, and Related Works Comparison Table

	Library Management System	Current EMU Library Catalogue	LIBKART	Bluetooth Library Manager
Technologies	HTML, CSS, JavaScript, ReactJS, NodeJS, ExpressJS, MySQL	JavaScript, PrototypeJS, script.aculo.us, Apache Web Server	Java	Python, SL4A, Bluetooth, SQLite
Platforms	Web application, any computer or smart device with an internet browser	Web application, any computer or smart device with an internet browser	Android	Android
Responsiveness	Fully responsive, along computers and smart devices	Responsive, along computers and smart devices, has several bugs/errors	Only on mobile devices	Only on mobile devices

Hardware Implementation	No hardware implementation, stated as a future work further in chapter 4	No hardware implementation	No hardware implementation	No hardware implementation
Search for resources	Search for resources all within the system	Search for resources all within the system	Search for resources all within the system	Search for resources all within the system
Locate resources	View resource details, containing the resource floor and shelf information	View resource details, containing the resource shelf number	N/A	View resource details, containing the resource rack
Borrow/Reserve online	Carried out fully within the system, without applying or paperwork	Requires applying for a membership and borrowed by filling a form	Only borrow request within the system	N/A
Keep track of borrows/reserves	Within the system	Within the system, if the user is a member	N/A	N/A
Notifications	Within the system	Via e-mail	N/A	N/A

As a web application, the LMS contains two applications, and works as a combination of these two applications. The backend application, which receives requests from the client, and provides response to the user, alongside with the frontend application, where the user interacts with the system through the GUI. Focused on modern methodologies and technologies, the backend application was built using NodeJS, ExpressJS and MySQL, then the frontend application was built using HTML, CSS, JavaScript and React alongside with Material-UI.

4.1 Future Work

In recent years, many libraries around the world is adapting their systems both with online and hardware technologies combined. As also mentioned in the ‘Related Works’ section of this study, it is clear that many studies are conducted to bring both hardware technologies and other methodologies such RFID, NFC, WSN, OFET, Bluetooth, IoT, etc., besides of online technologies. The recent progress and the availability of such technologies enable a library management system to be a more complete, coherent, and efficient system as a whole with the combination of both hardware and online technologies.

Therefore, the LMS can also be easily integrated with such technologies, as a part of an even larger system. A system where all the resources within the library is equipped with RFID cards, and all the shelve racks with RFID readers. That way, the books could dynamically be tracked within the library, automatically showing the book location according to those RFID readings. Besides of that, special borrowing stations physically could also be placed within the library, where the borrowing can be carried out by scanning the RFID cards on these resources, automatically without any further support from the librarian. Amongst these borrowing stations, there could also be returning stations for returning the borrowed resources, carrying out the returning resource process, again without any further assist from the librarian. Both on borrowing and returning stations, a web service could be written with NodeJS, for carrying out the borrowing and returning actions, by making related calls to the LMS API.

Often, resources within the library go under maintenance, in order to keep them in a good physical condition. So, certain resources inside the library may not be accessible from time to time until they are replaced or fixed, if they need and undergo maintenance. To identify if a resource is currently under maintenance, a 'maintenance' field could be kept in the 'resource_item' database table. Thus, the number items that are currently available to borrow could be controlled more accurately. For example, if there are multiple copies available of an item, and only a single book undergoes maintenance, the total number of available copies should be decreased by one. But if there is only a single copy of the book, and it undergoes maintenance, a message could be shown to the user on the resource details page, stating that the book is currently undergoing maintenance.

Another useful feature for library users, would be having a rating system for resources. By keeping a rating for resources, library users could get information about how useful or helpful a resource is more clearly. This feature could be implemented by adding a rating picker on the resource details page and a field called 'rating' to the 'resource' database table, keeping the average rating of a resource.

In addition, digital versions of resources could also be provided for library users to view online. To implement this, a field can be added to the 'resource_item' database table called 'digital_file', for the scanned, digital version of the resource, such as a PDF file. When adding a resource item, the digital version could be added by this field. Then on the resource details page, a button called 'View Digitally' could be shown, if the digital version of the book is available within the system. That way, if any of the users prefer to view a digital version of a resource, they can view it online.

REFERENCES

- Baryshev, R. A., Verkhovets, S. V., & Babina, O. I. (2017). The smart library project Development of information and library services for educational and scientific activity. *The Electronic Library*, 36(3).
- Bayani, M., Segura, A., Alvarado, M., & Loaiza, M. (2018). IoT-Based Library Automation and Monitoring system: Developing an Implementation framework of Implementation. *e-Ciencias de la Información*, 8(1).
- Bhattacharya, S. (2014). Blue-Droid: An Intelligent Library Management System on Android Platform. *IOSR Journal of Computer Engineering*, 16(4).
- Bisbal, J., Lawless, D., Wu, B., Grimson, J., Wade, V., Richardson, R., & O'Sullivan, D. (1997). *A Survey of Research into Legacy System Migration*.
- Cao, G., Liang, M., & Li, X. (2018). How to make the library smart? The conceptualization of the smart library. *The Electronic Library*, 36(5).
- Chang, C.-C. (2013). Library mobile applications in. *Emeral Insight*.
- Christensson, P. (2016, June 17). *Client-Server Model Definition*. (TechTerms) Retrieved August 19, 2020, from https://techterms.com/definition/client-server_model

- Doosti, B., Dong, T., & Deka, B. (2018). *A Computational Method for Evaluating UI Patterns*.
- Ganesan, A., T.Chithralekha, & Rajapandian, M. (2018). A Formal Model for Legacy System Understanding. *I.J. Intelligent Systems and Applications*.
- Giurgiu, L., & Gligorea, I. (2017). Responsive Web Design Techniques. *De Gruyter Open*.
- Hahn, E. M. (2016). *Express in Action: Writing, building, and testing Node.js applications*. New York: Manning Publications.
- Hota, A., & Prabhu, D. M. (2014). Node.js: Lightweight, Event driven I/O web development. *Informatics*.
- Huang, L.-S., Moshchuk, A., Wang, H. J., Schechter, S., & Jackson, C. (2012). *Clickjacking: Attacks and Defenses*.
- Jin, K. H. (2017). *Teaching Responsive Web Design to Novice Learners*. New York: Association for Computing Machinery.
- Job, J. (2015, December 7). *My Scrum Diagram*. (jordanjob.me) Retrieved August 19, 2020, from <https://jordanjob.me/blog/scrum-diagram/>

- Karant, S., Castelino, J., Nireeksha, Nazareth, F., & K, A. (2017). An Advanced Library Management System Using Android Device. *International Journal of Latest Technology in Engineering, Management & Applied Science*, VI(IV).
- Kirch, O., & Dawson, T. (2000). *Linux Network Administrator's, 2nd Edition*. O'Reilly.
- LeMahieu, P. G., Nordstrum, L. E., & Potvin, A. S. (2017). Design-based implementation research. *Quality Assurance in Education*, 25(1).
- Londhe, M. N., & Patil, D. S. (2015). Open Source Library Management Systems: A Survey and Present Developmental Status. *International Journal of Library and Information Science*.
- Makhija, R. (2020). *Top Front-End Frameworks in 2020*. (Guru TechnoLabs) Retrieved August 19, 2020, from <https://www.gurutechnolabs.com/top-front-end-frameworks/>
- Miller, D. (2018). *The Power of JavaScript*. New York: Cavendish Square Publishing.
- Monali, M. S., & S.A., P. G. (2017). GSM and RFID Based Library Book Availability and Location Finder System. *International Advanced Research Journal in Science, Engineering and Technology*, 4(Special Issue 2).

- Napper, T. M. (2020, APRIL 7). *Responsive Web Design*. (MY LATEST NEWS)
Retrieved August 19, 2020, from <https://mylatestnews.org/responsive-web-design/>
- Nugroho, S., Waluyo, S. H., & Hakim, L. (2017). Comparative Analysis of Software Development Methods between Parallel, V-Shaped and Iterative. *International Journal of Computer Applications*, 169(11).
- Pandey, J., Kazmi, S. I., Hayat, M. S., & Ahmed, I. (2017). *A Study on Implementation of Smart Library Systems using IoT*.
- Patil, N., Karande, P., Desai, J., & Pereira, S. (2017). Internet of Things for library Management System. *International Journal of Engineering Science and Computing*, 7(4).
- Powell-Morse, A. (2016, December 15). *Iterative Model: What Is It And When Should You Use It?* (Airbrake) Retrieved August 19, 2020, from <https://airbrake.io/blog/sdlc/iterative-model>
- Sangavi, S., Deepa, C. S., Surya, S., Vinumadhi, C. P., & Brindha, M. S. (2016). Advanced Library Management System Using RFID. *International Journal for Trends in Engineering & Technology*, 15(1).
- Shada, G. S., & Ayu, M. A. (2018). *Designing Android User Interface for University Mobile Library*.

- Shylesh, S. (2017). *A Study of Software Development Life Cycle Process Models*. Mangalore: Srinivas Institute of Management Studies.
- Sriramya, P., & Karthika, R. A. (2015). Providing Password Security by Salted Password Hashing Using Bcrypt Algorithm. *ARPJN Journal of Engineering and Applied Sciences*, 10(13).
- Suda, K. A., & Rani, N. S. (2013). Radio Frequency Identification for Efficient Library Management. *International Journal of Business and Social Science*, 4(15).
- Sutherland, J., & Schwaber, K. (2011). *The Scrum Papers: Nut, Bolts, and Origins of an Agile Framework*. Paris.
- Thakur, B. S., & Chaudhary, S. (2013). Content Sniffing Attack Detection in Client and Server Side: A Survey. *International Journal of Advanced Computer Research*, 3(10).
- Torres-Díaz, D. J.-C., Duart, D. J., Gómez-Alvarado, D. H.-F., Marín-Gutiérrez, D. I., & Segarra-Faggioni, V. (2016). Internet Use and Academic Success in University Students. *Media Education Research Journal*.
- Vandana, C. P., Bhattacharjee, A., & Gupta, A. (2017). Library Management system based on IoT. *Journal of Computer Science and Engineering*, 3(4).

Vogt, P., Nentwich, F., Jovanovic, N., Kirda, E., Kruegel, C., & Vigna, G. (2007). *Cross-Site Scripting Prevention with Dynamic Data Tainting and Static Analysis*.

W3C. (2018, March 27). *HTML 4.01 Specification*. Retrieved from <https://www.w3.org/TR/2018/SPSD-html401-20180327/>

W3C. (2019, January 22). *CSS Snapshot 2018*. Retrieved from <https://www.w3.org/TR/css3-roadmap/>

Wiener, L., Ekholm, T., & Haller, P. (2017). *Modular Responsive Web Design: An Experience Report*.

Wieruch, R. (2019). *The Road to Learn React*. Leanpub.

APPENDIX

User Manual

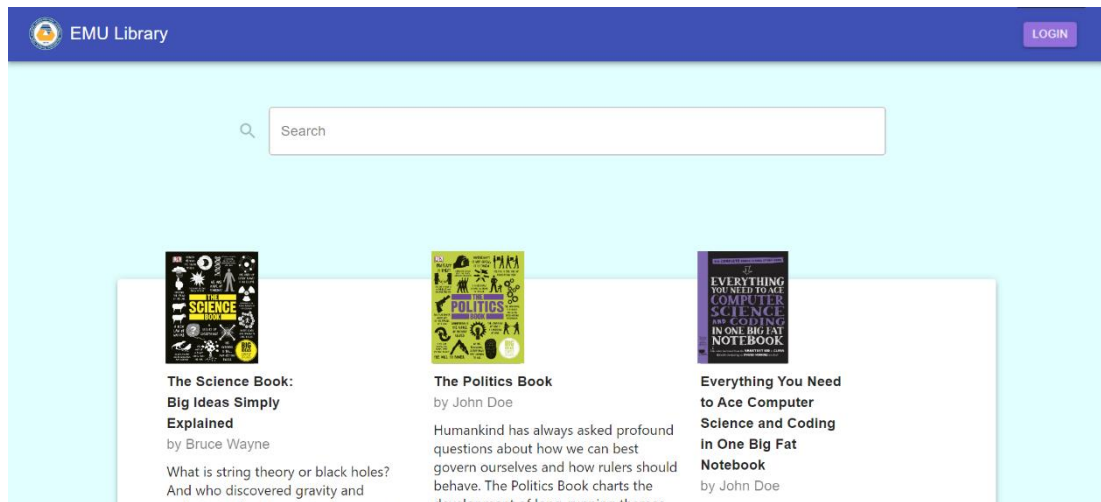


Figure 14: Home Page (Library Users/Guest)

From the home page, the users can search for a resource, author, shelf or floor.

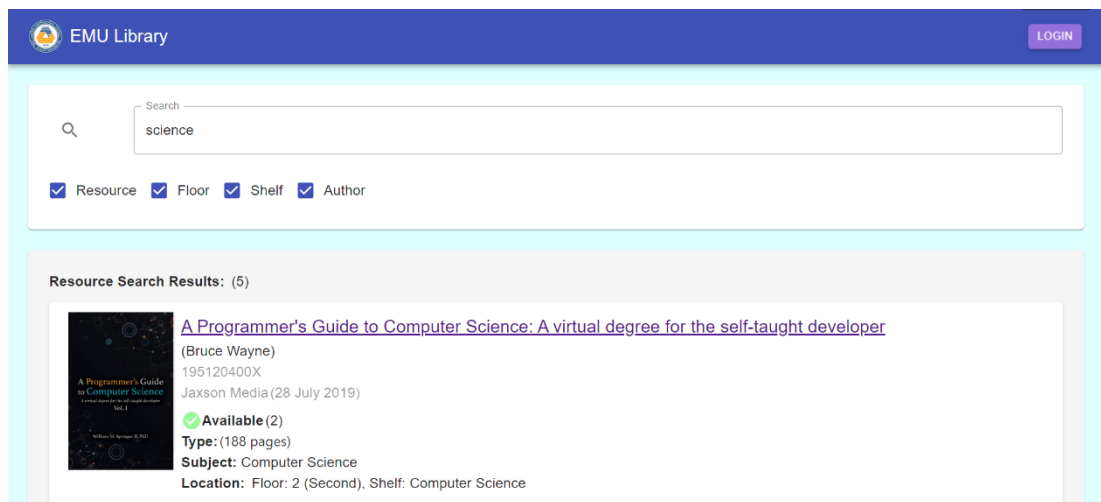


Figure 15: Search Results Page

When a search is made, the user is taken to the search page to check the search results.

In this page, the related items with the query are listed, according to the search criteria.

The search results here can be narrowed down by resources, floors, shelves or authors.

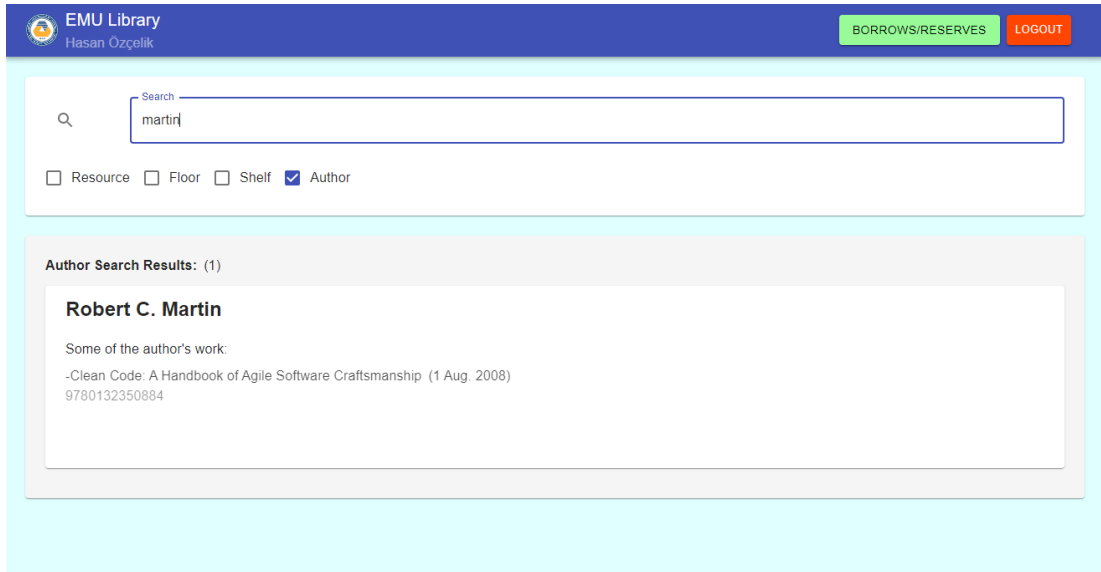


Figure 16: Author Search Result

When a search result includes an author, the author name and some work of the author is displayed in the author search results.

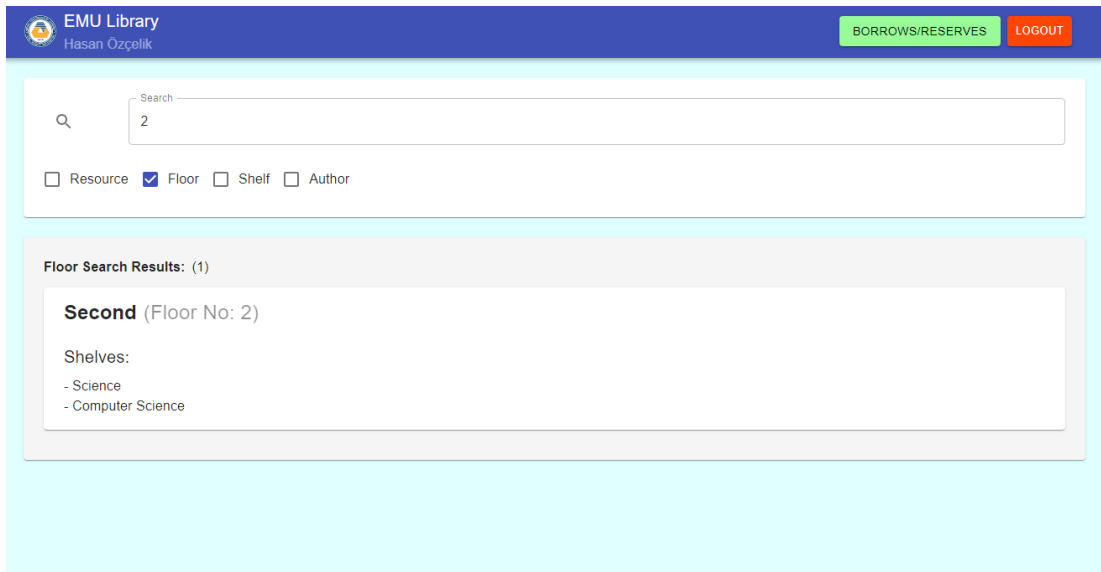


Figure 17: Floor Search Result

When a search result includes a floor, the floor name, floor number, and some of the shelves that are in that floor is displayed in the floor search results.

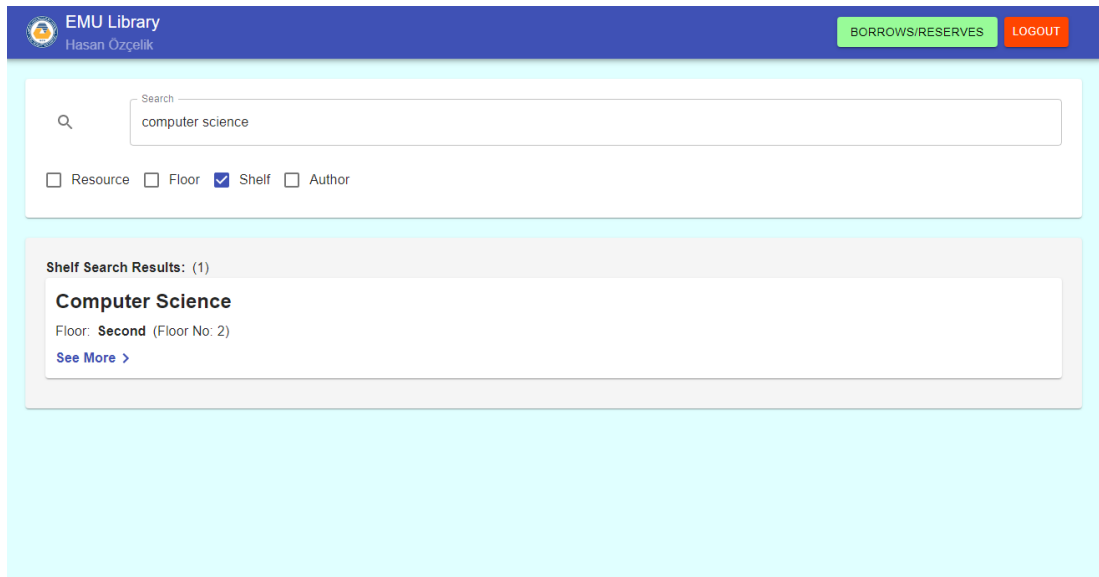


Figure 18: Shelf Search Result

When a search result includes a shelf, the shelf name and the floor that the shelf is in is displayed in the shelf search results.

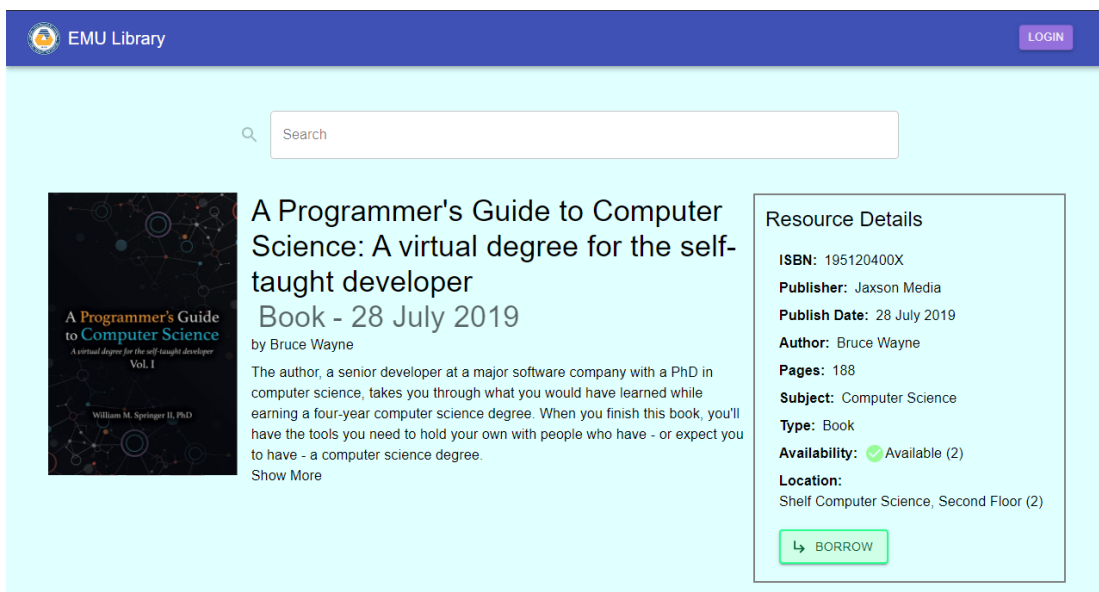


Figure 19: Resource Details Page as a Guest

Upon clicking the resource title in search results, the user is taken to the resource details page. In this page, the user can see the related resource information such as the title, type, release date, author, summary, ISBN, publisher, publish date, author, page

number, availability and location. In this page, any resource that is available can be asked to be borrowed by clicking the “Borrow” button.

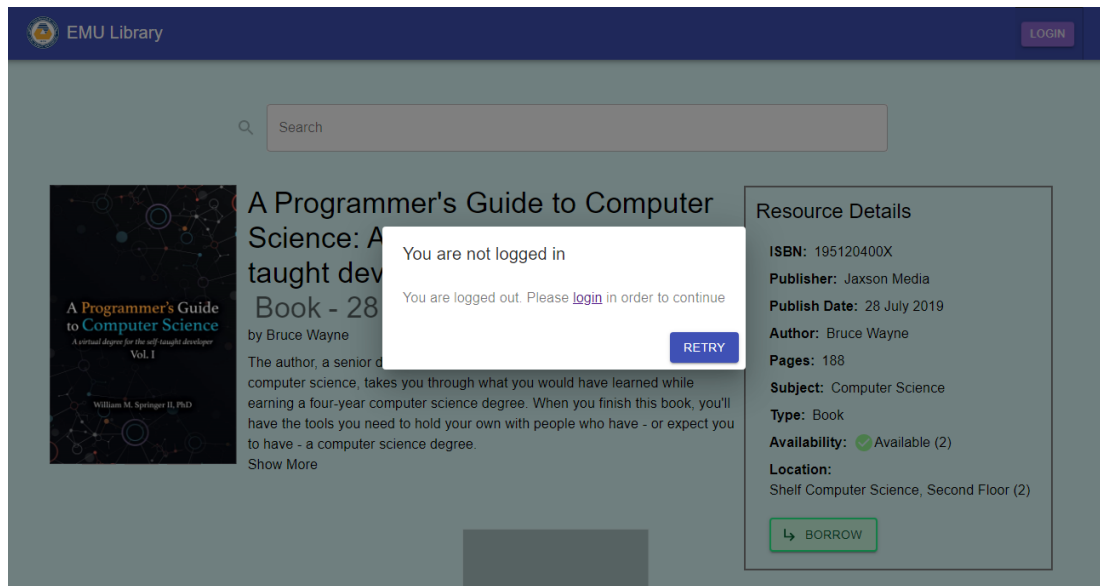


Figure 20: Login Required Dialog

If users wish to borrow a resource, they have to login to the system with their student/lecturer id and password. The users can login by clicking the ‘login’ button either on the login warning dialog, or on the right side of the top bar.

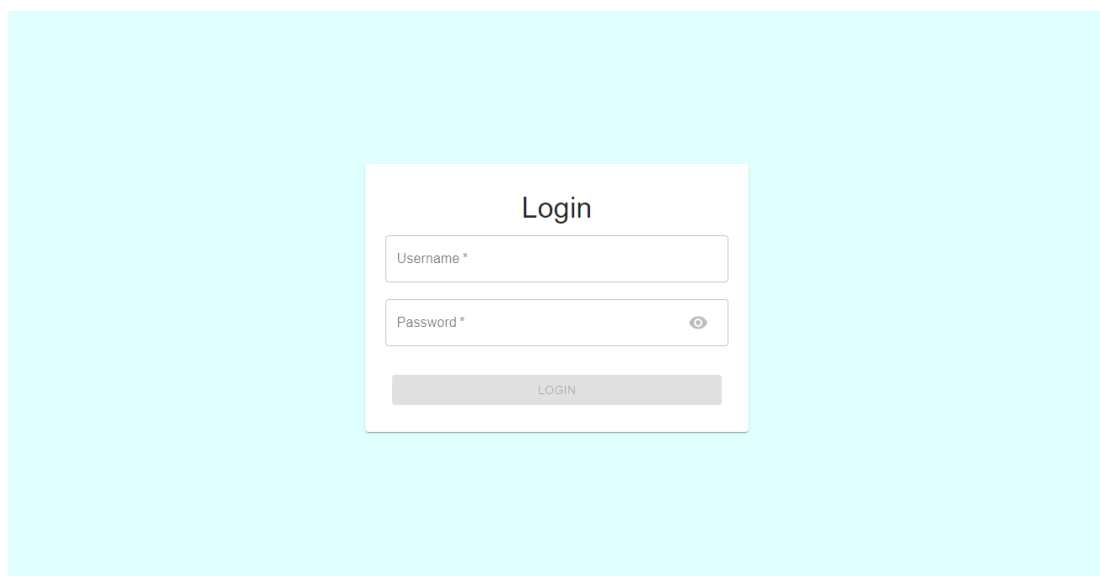


Figure 21: Login Page

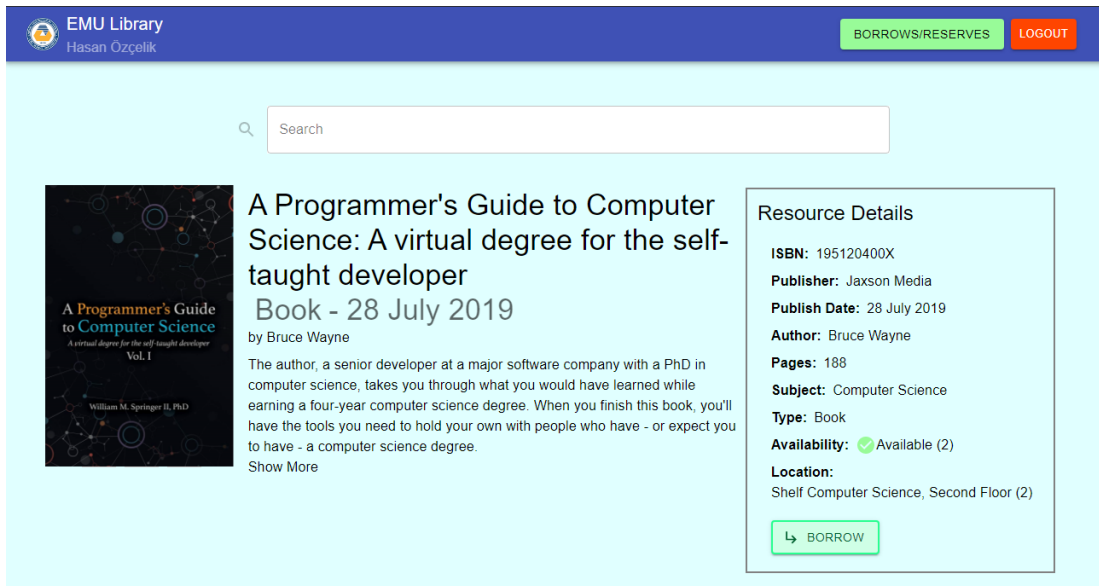


Figure 22: Resource Details Page as a Student/Lecturer

After logging in, the users can now make a borrow request to any resource that is available, by clicking the 'Borrow' button.

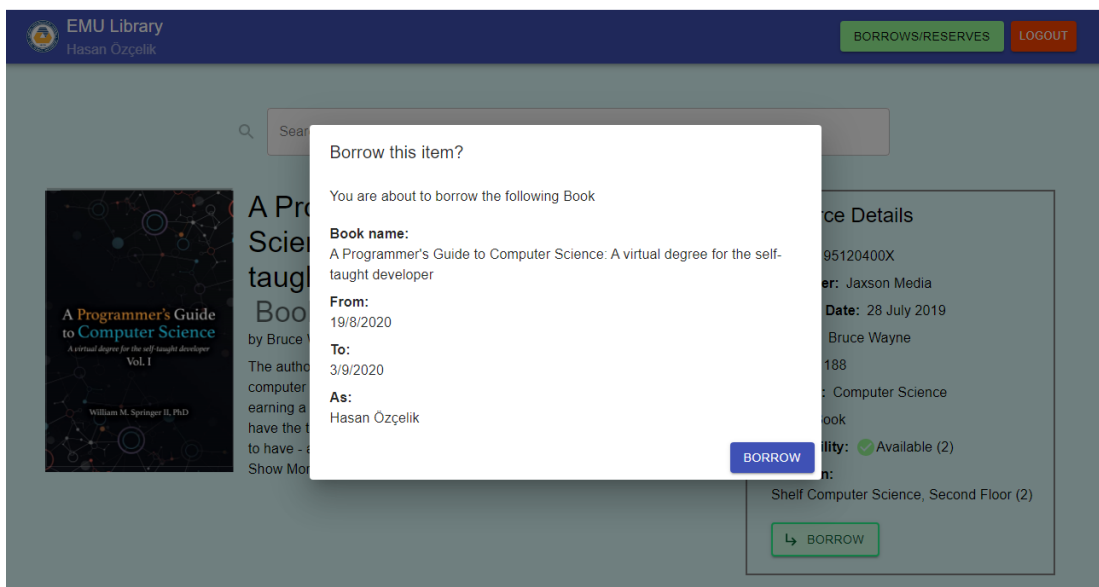


Figure 23: Borrow Dialog

When the 'Borrow' button is clicked, the user can see the borrow details on the borrow dialog. In this dialog, the book name, borrow starting date, borrow ending that, and

user full name is displayed. The user can confirm the borrow request by clicking the ‘Borrow’ button on this dialog.



Figure 24: Reservable Resource

If a resource is not currently available, meaning that all the resource copies are borrowed, the user can reserve the resource by clicking the ‘Reserve’ button.

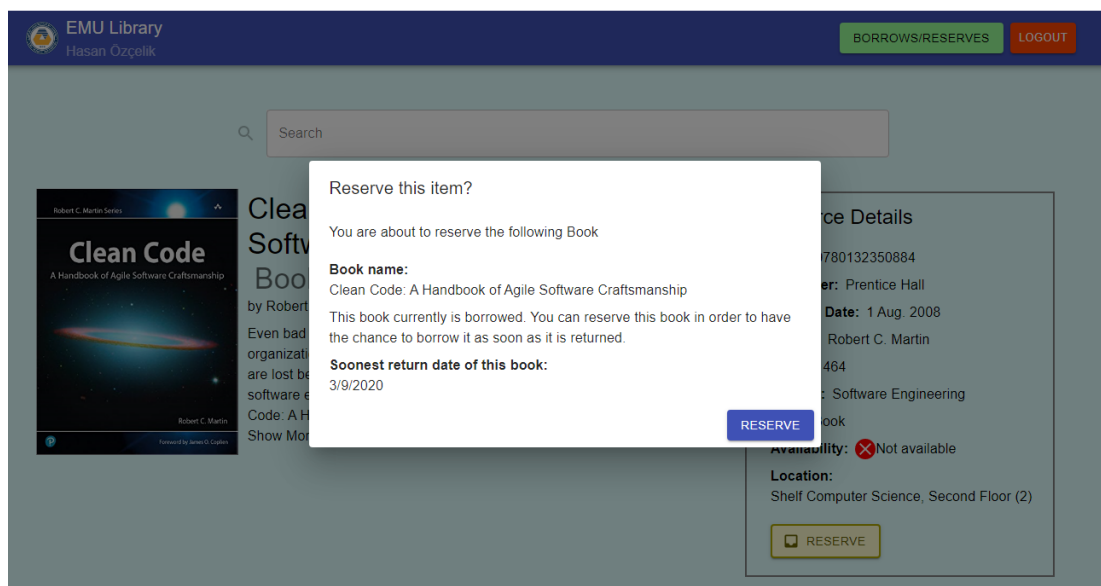


Figure 25: Reserve Dialog

When the 'Reserve' button is clicked, the user can see the reserve details on the borrow dialog. In this dialog, the book name and the soonest return date is displayed. The user can confirm the reserve by clicking the 'Reserve' button on this dialog.

#	Process	Borrower Name	Borrower Surname	Borrower Student ID	Borrowed Resource ISBN	Borrowed Resource Name	Borrow Date	Return Date
1	Reserve	Hasan	Özçelik	18500368	195120400X	A Programmer's Guide to Computer Science: A virtual degree for the self-taught developer	2020-08-03T21:40:01.000Z	2020-08-18T21:40:01.000Z
2	Reserve	Hasan	Özçelik	18500368	9780132350884	Clean Code: A Handbook of Agile Software Craftsmanship	2020-08-19T19:48:27.000Z	2020-09-03T19:48:27.000Z

Figure 26: Current Library User Processes

By clicking the 'Borrows/Reserves' button on the right side of the app bar, users can view the current ongoing borrow or reserve processes as a table. On this table, the process type, borrower name, borrower surname, borrower student/lecturer id, borrowed resource ISBN, borrowed resource name, borrow date and return date are displayed.



Figure 27: Librarian Home Page

The librarian section of the LMS can only be accessed by librarian users, thus no guest is allowed on the librarian application. From the home page, the librarian can view the resources (books), resource types, authors, processes, floors and shelves. From each respective view page for these, new items can be added, edited or deleted.

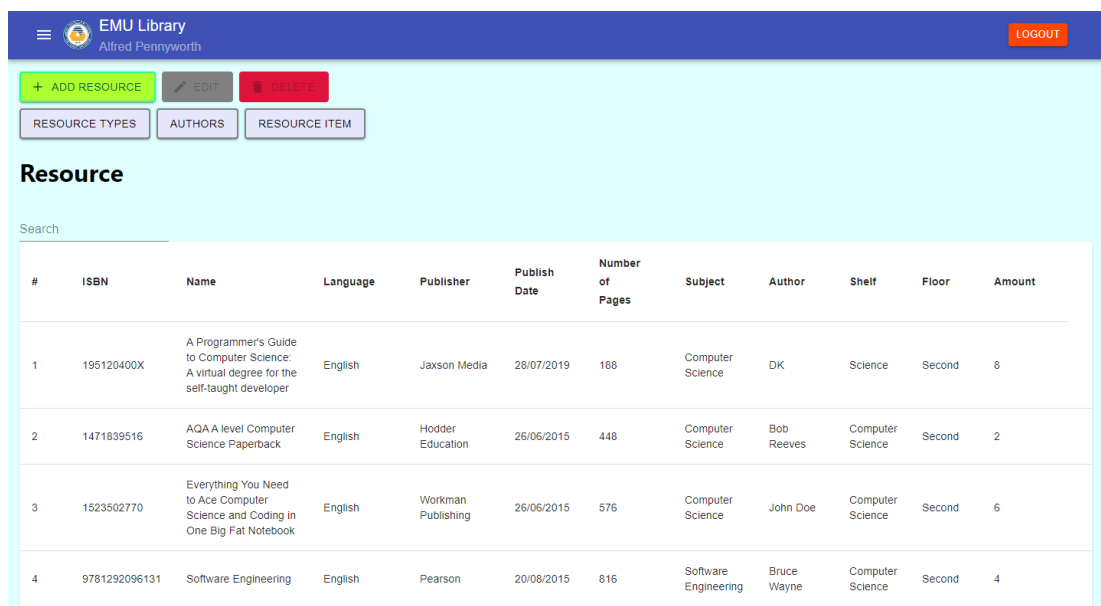
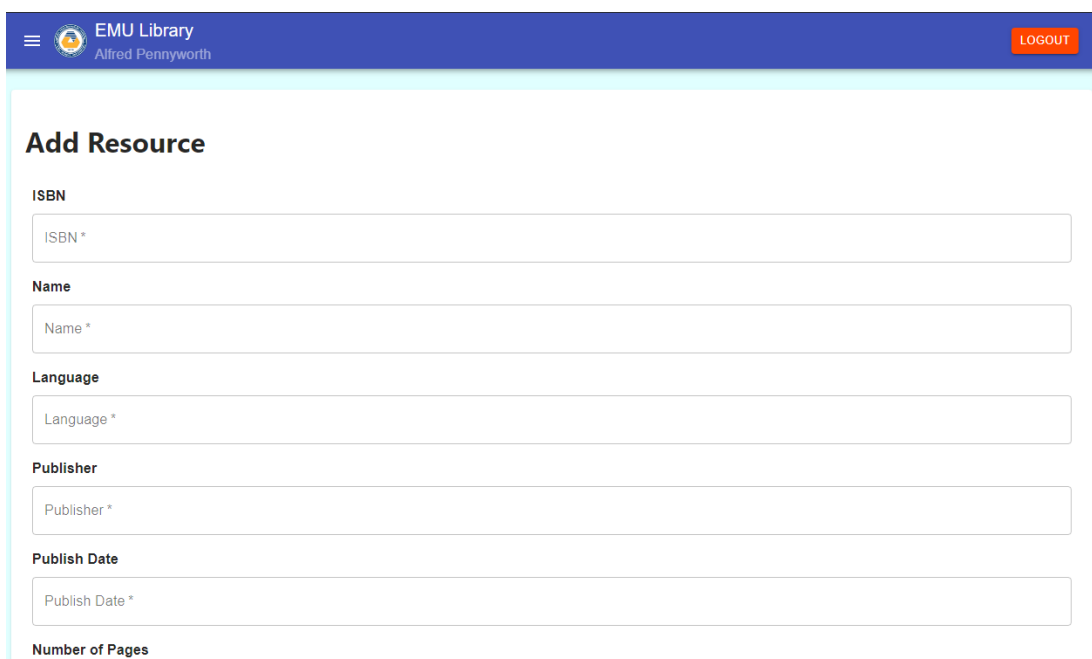


Figure 28: Resources Page (Librarian)

In this page, all the resources within the library. The ISBN, name, language, publisher, publish date, number of pages, subject, author, shelf, floor and the current amount of the resource are displayed in the view table. The librarian can search items the view

table by using the 'Search' text field on top of the view table. Any row can also be sorted ascending and descending. A new resource can be added by clicking the 'Add New' button on top, an item can be edited by choosing it on the view table and clicking the 'Edit' button, and any item can be deleted by choosing it on the view table and clicking 'Delete' button. Also, in the Resources Page, the librarian can view resource types and authors, by clicking to the 'Resource Types' button and the 'Authors' button and resource items can be added by clicking the 'Resource Item' button.



The screenshot shows the 'Add Resource' form in the EMU Library system. The header includes the EMU Library logo and name, and a 'LOGOUT' button. The form contains the following fields:

- ISBN**: ISBN *
- Name**: Name *
- Language**: Language *
- Publisher**: Publisher *
- Publish Date**: Publish Date *
- Number of Pages**: (label visible, input field not fully shown)

Figure 29: Add Resource (Librarian)

On the 'Add Resource' page, the user can add a new resource by filling the form. The form contains these fields; ISBN, name, language, publisher, publish date, number of pages, summary, subject, cover image, back cover image, author and shelf.

Edit Resource

ISBN
ISBN *
195120400X

Name
Name *
A Programmer's Guide to Computer Science: A virtual degree for the self-taught developer

Language
Language *
English

Publisher
Publisher *
Jaxson Media

Publish Date
Publish Date *
28 July 2019

Number of Pages

Figure 30: Edit Resource (Librarian)

On the 'Edit Resource' page, the user can edit the resource details by filling the form. The form contains the same fields as the add page; ISBN, name, language, publisher, publish date, number of pages, summary, subject, cover image, back cover image, author and shelf.

Resource Type

Search

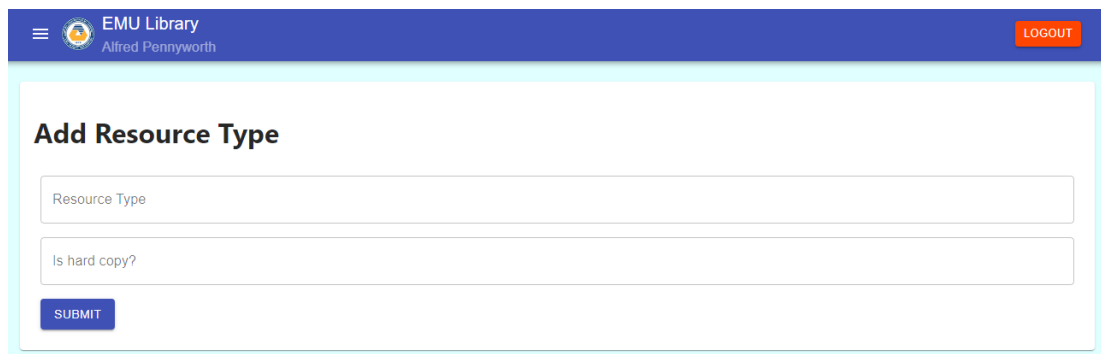
#	Resource Type	Is hard copy?
1	Book	Yes
2	E-Book	No
3	Thesis	Yes
4	Journal	Yes
5	Magazine	Yes

Rows per page: 5 1-5 of 7

Figure 31: Resource Types Page (Librarian)

In this page, all the resource types within the library. The resource type name and is the resource type hard copy or not are displayed in the view table. The librarian can

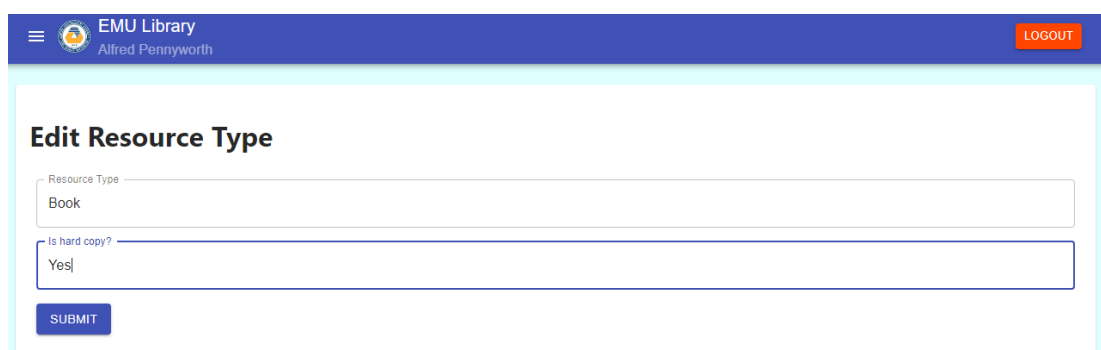
search items the view table by using the ‘Search’ text field on top of the view table. Any row can also be sorted ascending and descending. A new resource type can be added by clicking the ‘Add New’ button on top, an item can be edited by choosing it on the view table and clicking the ‘Edit’ button, and any item can be deleted by choosing it on the view table and clicking ‘Delete’ button.



The screenshot shows the 'Add Resource Type' page. At the top, there is a blue navigation bar with the EMU Library logo and the name 'Alfred Pennyworth'. A 'LOGOUT' button is in the top right corner. The main content area is titled 'Add Resource Type' and contains two text input fields. The first field is labeled 'Resource Type' and is empty. The second field is labeled 'Is hard copy?' and is also empty. Below the second field is a blue 'SUBMIT' button.

Figure 32: Add Resource Type Page (Librarian)

On the ‘Add Resource Type’ page, the user can add a new resource type by filling the form. The form contains these fields; resource type name and is the resource type hard copy or not.



The screenshot shows the 'Edit Resource Type' page. At the top, there is a blue navigation bar with the EMU Library logo and the name 'Alfred Pennyworth'. A 'LOGOUT' button is in the top right corner. The main content area is titled 'Edit Resource Type' and contains two text input fields. The first field is labeled 'Resource Type' and contains the text 'Book'. The second field is labeled 'Is hard copy?' and contains the text 'Yes'. Below the second field is a blue 'SUBMIT' button.

Figure 33: Edit Resource Type Page (Librarian)

On the 'Edit Resource Type' page, the user can edit the resource details by filling the form. The form contains the same fields as the add page; resource type name and is the resource type hard copy or not.



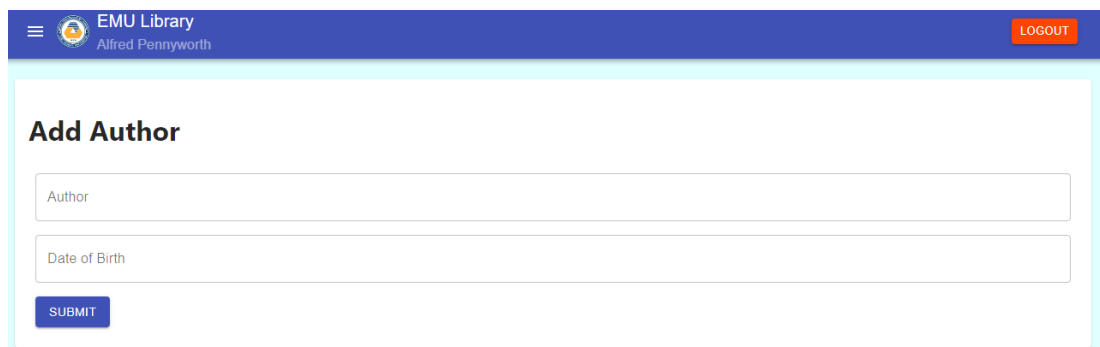
The screenshot shows the EMU Library interface for the 'Author' page. At the top, there is a navigation bar with the EMU Library logo and name, and a 'LOGOUT' button. Below the navigation bar, there are three buttons: 'ADD NEW' (green), 'EDIT' (gray), and 'DELETE' (red). The main content area is titled 'Author' and features a search bar. Below the search bar is a table with the following data:

#	Author	Date of Birth
1	John Doe	13 September 1972
2	Bruce Wayne	26 July 1983
3	Richard Grayson	1 June 1993
4	Robert C. Martin	5 December 1952

At the bottom right of the table, there is a pagination control showing 'Rows per page: 5' and '1-4 of 4'.

Figure 34: Authors Page (Librarian)

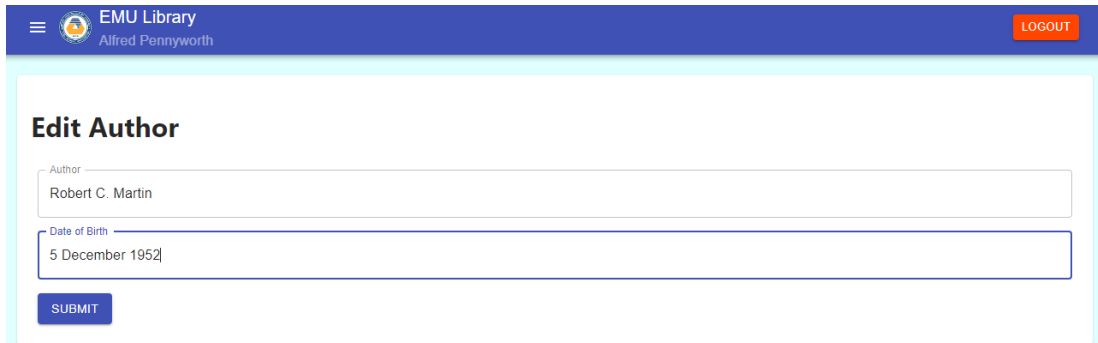
In this page, all the authors within the library. The author name and the author date of birth are displayed in the view table. The librarian can search items the view table by using the 'Search' text field on top of the view table. Any row can also be sorted ascending and descending. A new author can be added by clicking the 'Add New' button on top, an item can be edited by choosing it on the view table and clicking the 'Edit' button, and any item can be deleted by choosing it on the view table and clicking 'Delete' button.



The screenshot shows the EMU Library interface for the 'Add Author' page. At the top, there is a navigation bar with the EMU Library logo and name, and a 'LOGOUT' button. Below the navigation bar, there is a form titled 'Add Author'. The form contains two input fields: 'Author' and 'Date of Birth'. Below the input fields is a 'SUBMIT' button.

Figure 35: Add Author Page (Librarian)

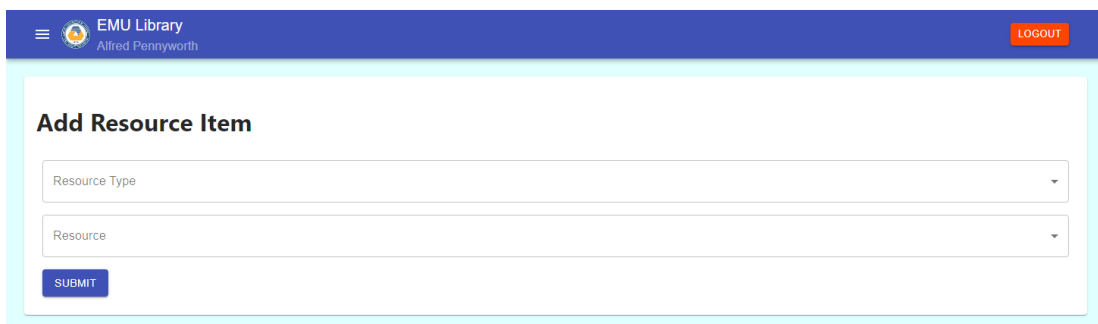
On the 'Add Author' page, the user can add a new resource type by filling the form. The form contains these fields; author name and the author date of birth.



The screenshot shows the 'Edit Author' page. At the top, there is a blue navigation bar with the EMU Library logo and the name 'Alfred Pennyworth' on the left, and a 'LOGOUT' button on the right. The main content area is titled 'Edit Author' and contains two text input fields. The first field is labeled 'Author' and contains the text 'Robert C. Martin'. The second field is labeled 'Date of Birth' and contains the text '5 December 1952'. Below these fields is a blue 'SUBMIT' button.

Figure 36: Edit Author Page (Librarian)

On the 'Edit Author' page, the user can edit the author details by filling the form. The form contains the same fields as the add page; author name and the author date of birth.



The screenshot shows the 'Add Resource Item' page. At the top, there is a blue navigation bar with the EMU Library logo and the name 'Alfred Pennyworth' on the left, and a 'LOGOUT' button on the right. The main content area is titled 'Add Resource Item' and contains two dropdown menus. The first dropdown is labeled 'Resource Type' and the second is labeled 'Resource'. Below these dropdowns is a blue 'SUBMIT' button.

Figure 37: Add Resource Item Page (Librarian)

On the 'Add Resource Item' page, the user can add a new resource item by choosing the resource item type, and the resource.

#	Floor Name	Floor Number	Shelves in floor
1	First	1	1
2	Second	2	2
3	3	3	0

Figure 38: Floors Page (Librarian)

In this page, all the floors within the library. The floor name, floor number, number of shelves on the floor are displayed in the view table. The librarian can search items the view table by using the ‘Search’ text field on top of the view table. Any row can also be sorted ascending and descending. A new floor can be added by clicking the ‘Add New’ button on top, an item can be edited by choosing it on the view table and clicking the ‘Edit’ button, and any item can be deleted by choosing it on the view table and clicking ‘Delete’ button.

Figure 39: Add Floor Page (Librarian)

On the ‘Add Floor’ page, the user can add a new floor by filling the form. The form contains these fields; floor name and floor number.

Figure 40: Edit Floor Page (Librarian)

On the ‘Edit Floor’ page, the user can edit the floor details by filling the form. The form contains the same fields as the add page; floor name and floor number.

#	Shelf Name	Shelf Floor Name	Shelf Floor Number	Books in shelf
1	Politics	First	1	1
2	Science	Second	2	1
3	Computer Science	Second	2	5

Figure 41: Shelves Page (Librarian)

In this page, all the shelves within the library. Shelf name, shelf floor name, shelf floor number, and number of books on the shelf are displayed in the view table. The librarian can search items the view table by using the ‘Search’ text field on top of the view table. Any row can also be sorted ascending and descending. A new shelf can be added by clicking the ‘Add New’ button on top, an item can be edited by choosing it on the view table and clicking the ‘Edit’ button, and any item can be deleted by choosing it on the view table and clicking ‘Delete’ button.

EMU Library
Alfred Pennyworth

LOGOUT

Add Shelf

Shelf Name

Shelf Floor Name

SUBMIT

Figure 42: Add Shelf Page (Librarian)

On the ‘Add Shelf’ page, the user can add a new shelf by filling the form. The form contains these fields; shelf name and the floor.

EMU Library
Alfred Pennyworth

LOGOUT

Edit Shelf

Shelf Name
Science

Shelf Floor Name
Second

SUBMIT

Figure 43: Edit Floor Page (Librarian)

On the ‘Edit Shelf’ page, the user can edit the shelf details by filling the form. The form contains the same fields as the add page; shelf name and the floor.

The screenshot shows the EMU Library interface with a 'Student Processes' section. A search bar is located at the top left of the table area. The table contains four rows of data, each representing a library process. The columns are: #, Process, Borrower Name, Borrower Surname, Borrower Student ID, Borrowed Resource ISBN, Borrowed Resource Name, Borrow Date, and Return Date. The data is as follows:

#	Process	Borrower Name	Borrower Surname	Borrower Student ID	Borrowed Resource ISBN	Borrowed Resource Name	Borrow Date	Return Date
1	Borrow	Jane	Doe	11223344	195120400X	A Programmer's Guide to Computer Science: A virtual degree for the self-taught developer	2020-08-03T21:39:34.000Z	2020-08-18T21:39:34.000Z
2	Reserve	Hasan	Özçelik	18500368	195120400X	A Programmer's Guide to Computer Science: A virtual degree for the self-taught developer	2020-08-03T21:40:01.000Z	2020-08-18T21:40:01.000Z
3	Borrow	Jane	Doe	11223344	9780132350884	Clean Code: A Handbook of Agile Software Craftsmanship	2020-08-19T19:46:58.000Z	2020-09-03T19:46:58.000Z
4	Reserve	Hasan	Özçelik	18500368	9780132350884	Clean Code: A Handbook of Agile Software Craftsmanship	2020-08-19T19:48:27.000Z	2020-09-03T19:48:27.000Z

At the bottom right of the table, there is a pagination control showing 'Rows per page: 5' and '1-4 of 4'.

Figure 44: Library User Processes (Librarian)

In this page, all the library user processes within the library. Process type, borrower name, borrower surname, borrower student/lecturer id, borrowed resource ISBN, borrowed resource name, borrow date, borrow return date are displayed in the view table. The librarian can search items the view table by using the ‘Search’ text field on top of the view table. Any row can also be sorted ascending and descending.

The screenshot shows the same 'Student Processes' page as Figure 44, but with a 'Borrow Request' dialog box overlaid on top. The dialog box contains the following information:

- Borrow Request**
- Borrow request for the following book
- Book name:** A Programmer's Guide to Computer Science: A virtual degree for the self-taught developer
- From:** 19/8/2020
- To:** 3/09/2020
- As:** Hasan Özçelik

At the bottom of the dialog box, there are two buttons: 'CONFIRM' (blue) and 'DECLINE' (red). The background table is dimmed, showing the same data as in Figure 44.

Figure 45: Borrow Request Dialog (Librarian)

By clicking a process on the Library User Processes view table, the librarian can confirm or decline a borrow request. On this dialog, the book name, borrow start date, borrow return date, and the user full name is displayed. The borrow request can be

confirmed by clicking the ‘Confirm’ button and declined by clicking the ‘Decline’ button.

Process	Process Status	Borrower Name	Borrower Surname	Borrower Student ID	Borrowed Resource ISBN	Borrowed Resource Name	Request Date	Return Date
Borrow	Approved	Hasan	Özçelik	18500368	9781292096131	Software Engineering	19/08/2020	03/09/2020
Borrow	Approved	Hasan	Özçelik	18500368	195120400X	A Programmer's Guide to Computer Science: A virtual degree for the self-taught developer	25/08/2020	09/09/2020
Borrow	Approved	Hasan	Özçelik	18500368	1523502770	Everything You Need to Ace Computer Science and Coding in One Big Fat Notebook	07/09/2020	22/09/2020
Borrow	Approved	Hasan	Özçelik	18500368	0241257107	Art: The Definitive Visual Guide	07/09/2020	22/09/2020

Figure 46: Return Borrow Page (Librarian)

On the ‘Process Requests’ page the librarian can change the process status of returned resource items by choosing an item, then clicking the ‘Return’ button.

Return Request

Return the following book borrowed book

Book name:
Software Engineering
9781292096131

Borrow date:
19/08/2020

Return date:
22/09/2020

As:
Hasan Özçelik
18500368

CANCEL
CONFIRM RETURN

Figure 47: Return Borrow Request

By clicking the ‘Return’ button, the librarian can confirm or decline a return. On this dialog, the book name, borrow start date, borrow return date, the user full name and

the user ID is displayed. The return request can be confirmed by clicking the 'Confirm Return' button and declined by clicking the 'Cancel' button.