كلية الحاسبات والذكاء الإصطناعي
Faculty of Computers & Artificial Intelligence

جامعة حلوان
HELWAN UNIVERSITY

كليـــــة :حاسبات و زكاء اصطناعي

كونترول الفرقة : الثانية

العـــام الجامعـــى 2019 / 2020 – دور مايـــو

الغلاف الخارجى للبحث

أولاً: البيانات الخاصة بالطالب

الفرقة الدراسية	**الثانية**	**التخصص**	**عام**	
اسم القسم	**عام**			
اسم المقرر	**هندسة البرمجيات –1**			
استاذ المقرر	**د / عمرو غنيم**			

ثانياً: البيانات الخاصة بالبحث

عنوان البحث				
طبيعة المشاركة	**بحث فردى**		**بحث جماعى**	√
ارسال البحث	**بواسطة البريد الالكتروني**			

	م	**الاسم رباعى**	**رقم الجلوس**	**الرقم القومى**
اسماء الطلاب المشاركين فى البحث (يكتب الاسم رباعيا)	**1**	**ابانوب رافت وديع كامل**	**2200**	**29809100102458**
	2	**جرجس حنا سمعان حنا**	**2329**	**29910282100717**
	3	**كريم طارق محمد عبد الغني**	**2521**	**29908090100719**
	4			
	5			

تاريخ الإرسال	**10 / 6 / 2020**

ثالثاً: البيانات الخاصة بالكونترول

النتيجة	**ناجح**		**راسب**	

	م	**الاسماء**	**التوقيع**
أعضاء لجنة تقييم البحث	**1**		
	2		
	3		

فى حالة عدم قبول البحث يرجى ذكر الأسباب	- ..... - ..... - .....

PART 1

Overview & Software Requirements Specification

1) Introduction

a) Purpose

Our E-Book System works around make it easier for users to suffer various of books online that has been uploaded by another users, select the required books and order them.

So, we are talking about a way to connect buyers with sellers through the internet which has been nearly used in every home in the world in the last years. The idea is that you can either be a seller and gain money by uploading your books online to be shown by hundreds, thousand or even million of users according to the reach of the website.

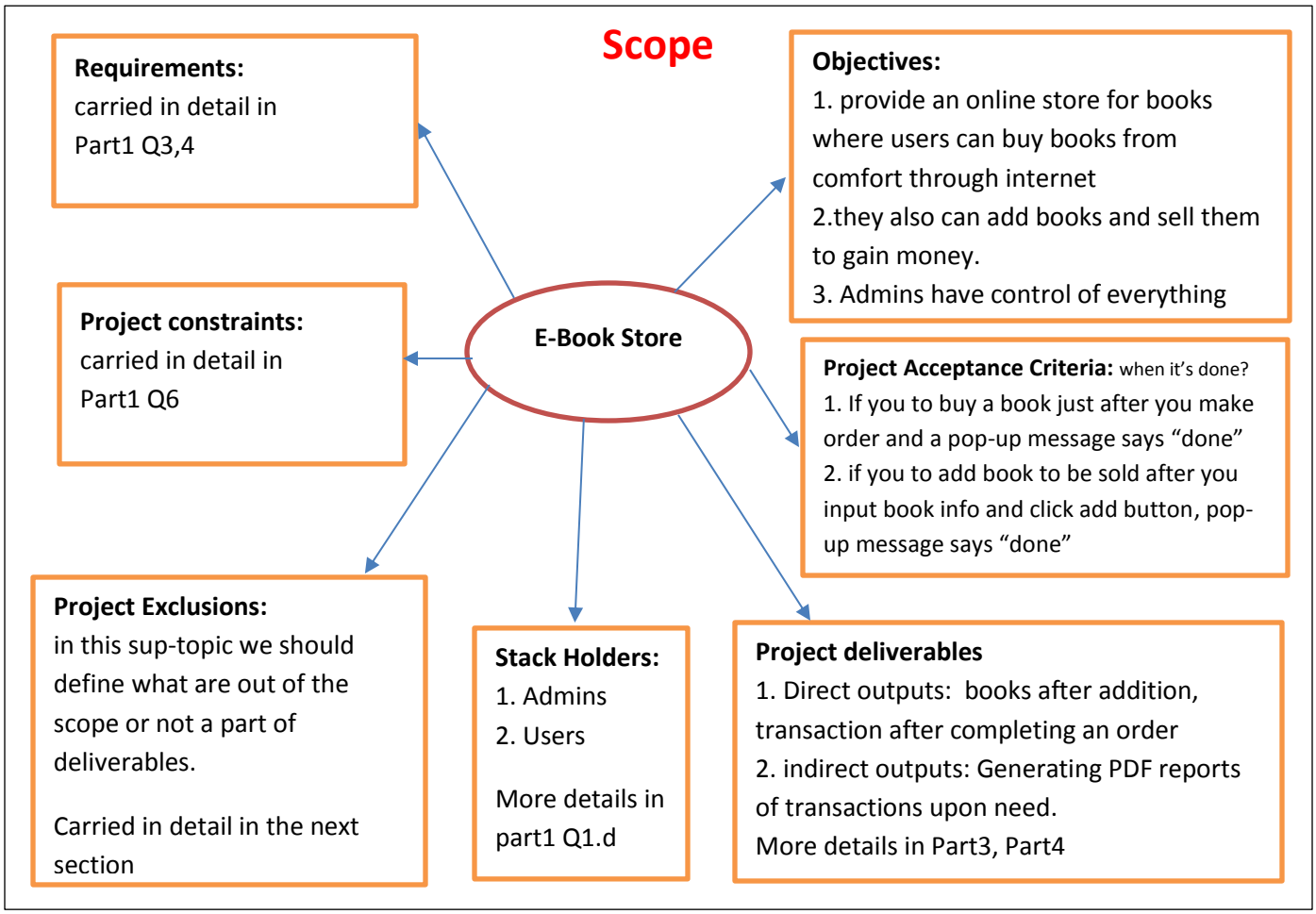
You can be a buyer for sure and select your favorite book out of hundreds of categories where each category contains thousands of books with enough details about each book.

User just need to register an account on the system and with some little clicks the job is done!

That was according to the user view or its purpose to join our website, According to the admins of website it's a way to gain money for sure from collecting taxes on every order and they also have the control to check users on the system, categories and book and can delete any user if he goes upon the site rules. Although admins have information about every user and every transaction that has been made to improve the quality of the system and make it secure.

b) Project Scope

Project Scope



Out of Scope (Project exclusions)

This project **will NOT accomplish or include** the following:	1. Admins Can't register accounts (Add throw database) 2. No insertion or removing for Categories (Add, Remove throw database) 3. Users can't update their profile data 4. No one can access the website except after complete login even if they enter the links of other pages, they will be directed to login page again (made to improve security)

c) Glossary and Abbreviations

1. E-Book → Electronic Book
2. JS → Java Script
3. SRP → Software requirement pattern
4. SE → Software engineering

d) List of the System Stakeholders

1. Admins
2. Users

NOTE: our system is designed that it can hold many stake holders that's is achieved by just change the values of table users in database which contain column user type, can read, can write, can update, can delete. And we later control that in our project with **state design pattern.**
(0,1) acts as Boolean value except for user type column we give value for each new stack holder.

So, for example

User type	0
Can read	1
Can write	0
Can update	0
Can delete	0

User

User type	1
Can read	1
Can write	1
Can update	1
Can delete	1

Admin

User type	2
Can read	1
Can write	1
Can update	0
Can delete	0

Client (new type)

e) References

- Faculty lectures
- Book: Software Engineering 9th edition
- https://www.journaldev.com/1827/java-design-patterns-example-tutorial
- https://businessanalystlearnings.com/blog/2016/8/18/a-list-of-requirements-prioritization-techniques-you-should-know-about
- https://reqtest.com/requirements-blog/functional-vs-non-functional-requirements/

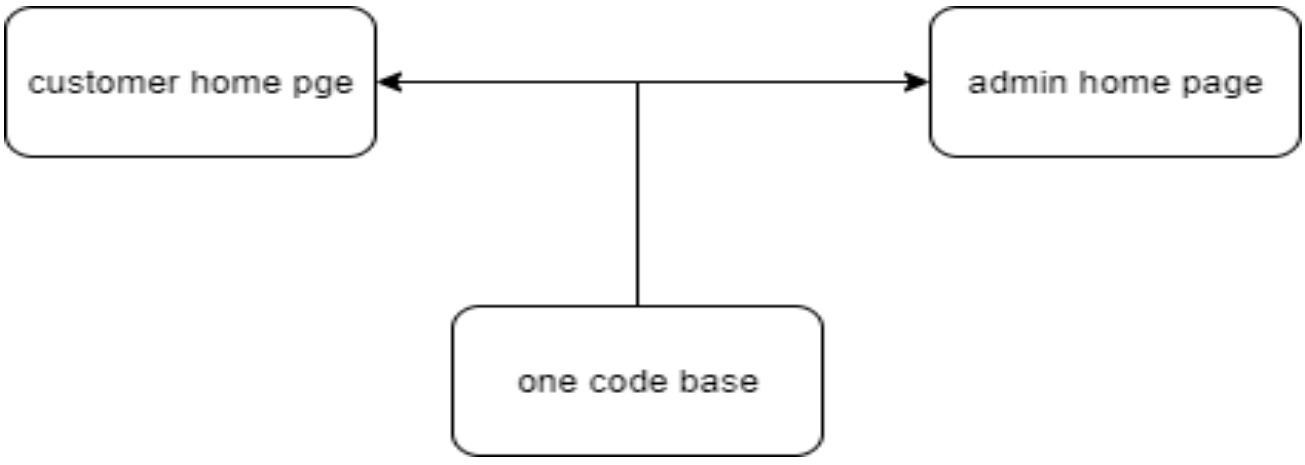
2) What is a Software Requirement Pattern (SRP)?

Software Requirements patterns: the reuse in software applications usually is addressed to the reusing of implementation details (i.e code), but SRP is the reuse of high level abstraction (requirements analysis stage), which comes with many benefits for the overall product and for the customers/developers ,like shortens the reduces the development cost and effort , improves the product quality ,reduces the time needed for the product to be in the market.

Requirement pattern for itself is a way or template used to write a requirement, and SRP is the same of design patterns but in a high-level approach than design patterns.

How SRP affected our architecture?

We used the concept of reusing in SRP and merged the common pages/features for the customer and the admin in a reusable ,customizable way that can implement specific features for each one of them individually, for example, both admin and customer can view the list of books ,login with username and password , so we made one common page for login and use it for both of them , so we didn't need to make two architectural designs for each one of them.



3) Functional Requirements

a) User Requirements Specification

Users FRs:

Requirement ID	Requirement Statement	Comments	Grouping Assignment & Explanation
U01	Customers can register with id, password, name, email	User must use a valid id and password, email	**Critical** - because user can't use our app without it
U02	Customers can login with their email, password	The name can be used for login	**Critical**- because user can't use our app without it
U04	Users can view all the books with their details	details: description, seller if exist, book condition, category, price and name	**Critical**- because it's one of the main features in the system, and let the user see different categories with different books
U05	Users can surf books and add a book to the cart	Books in the cart still not bought yet he need to make order first to buy it	**Moderate**- it's important for the user to make order consist of more than one product
U06	Users can see their cart details and the total amount	Details include: all the added items in that time with their prices and their information	**Critical**- users must see their cart before they make order.
U07	Users can make order to buy the selected books in the cart	The order contains all the (selected) books from the cart in that time	**Critical**- it's the main business of our System
U08	Users can choose the payment method	Cash & Visa	**Optional**- it's nice to have this feature but not important, people can pay cash.

U09	Users can add books to be sold in the website	Upload the name, category, ISBN, publisher, quantity, price, book condition, description, cover image	**Critical**- because it's one of the main features in the app, people must have the ability to add books.
U10	Users can upload books to be sold in the website	They upload a pdf of their book.	**Critical**- because it's one of the main features in the app,
U11	Users can view buyers	If any buyer bought a book from a user, Buyer's details will be displayed to that user	**Optional**- nice feature but it'll not affect the main features of the app
U12	Users can view sellers	If any user wants to buy a book, he can view seller information.	**Optional**- nice feature but it'll not affect the main features of the app
U13	Users can view all the past orders with their details	Details include price of each items with all the information of this item that have been shown before order	**Optional**- it's nice for users to see their past orders, but it'll not affect the main features of the app
U14	Users can view all their added and uploaded books on the system	Book appears with their cover and all information	**Optional**- it's a way that can make seller be in touch with their items
U15	Users can generate reports of their past purchase history	Reports are in form of PDF file that contain all the information of all the past orders of user	**Optional**- user already have that information in purchase page

equation for calculating the total cost of quantity from the same book:

$$Total\ Cost\ of\ Book = \sum (Price\ of\ book\ * Quaninty\ of\ same\ book)$$

equation for calculating the total cost of order:

$$Total\ Cost = \sum_{0}^{number\ of\ items} total\ price\ of\ each\ item$$

Admin FRs:

Requirements ID	Requirement statement	comments	Grouping Assignment & Explanation
A01	Admin can login with his email, password	All data of the admins will be added from the database (not through the website)	**Critical**- the admin can't use our app without it.
A02	Admin can view all the books with their details	Book viewed same as user view	**Critical**- Admins must be in touch of what going inside the site
A03	Admin can display all the registered users in the system with their registration information	Information include their name, email, phone number	**Critical** -the admin must be able to see the users, it's the main feature of the admin
A04	Admin can delete any user when he needs	Users that's goes above site rules or use scamming or cheating must be deleted	**Moderate**- its not a critical requirement nor an optional requirement because system can work without it but it must be included
A05	Admin can see all time transactions with their details	Transactions contains all the information about every order from every single user total price of each order, Total price of all orders	**Critical**- the admin must see the transactions happened in the system and total cost to keep track of their gaining from the website and improve security where everything is recorded
A06	Admin can generate reports of all-time transaction history	Reports are in a form of PDF file which contain all the information of all the past orders	**Moderate-** despite that is an optional requirement of users it highly recommended to be in admins feature because it all allow them to have a regenerated report of the system

b) System Requirement Specification

1. The system can let users register an account after filling the registration information and they can login with email and password.
2. The system shows all books to customers with different categories.
3. The system has search feature which allows users and admins to search for a specific book using search bar.
4. The system has ability to let customers see their cart, total price.
5. The system can let users see their past orders with its details
6. System can generate reports about orders.
7. System can let users email the admins through contact form.
8. System add books for the user in its category when they complete the form of book addition
9. System get the information of the added books of each user and group them in page my books and let each specific user see their added

c) Requirements' Priorities

we have used the numerical grouping technique to prioritize our requirements/features (**shown in the functional requirements**) point 3.a.

Numerical Grouping: This method is based on grouping requirements into different priority groups with each group representing something stakeholders can relate to.

For example, requirements can be grouped into critical priority, moderate priority and optional priority. Stakeholders may also classify requirements as compulsory, *very important*, *rather important*, *not important*, and *does not matte*r in order to describe their importance.

4) Non-Functional Requirements

a) the General types/Categories of non-functional requirements

There are different types/categories of non-functional requirements:

1. Constraints
2. External interface requirements
3. Performance
4. Quality attributes

We implemented some of these categories in our app, we'll discuss them in more detail in the next point.

b, c) Non-Functional Requirements Specification, Fit-Criteria for each one

we gathered the non-functional requirements we considered and how we tested and tried to implement them in the same point).

1. The first non-functional requirement we implemented was the load time of the app, which falls under the performance category, we used google speed test insight for making sure that the load time of our website is efficient and meets the standards for most users.
2. The second one was response time, which falls under the same category as number one (performance), according to Jakob Nielsen, there are 3 main metrics for response time, which we had followed. For more about those 3 metric click here.
3. The last non-functional requirement we had considered was the portability, which falls under the category of quality attributes, we aimed to reduce the size of our app as much as possible, so it can run on old devices, we used TinyPng to reduce the size of the images, which helped decreasing the size of the app dramatically.

d) How would the above-mentioned Non-Functional Requirements affect the overall Architecture of the System?

To implement those non-functional requirements we tried from the beginning to achieve that, so we have used the MVC architecture pattern to ensure high loading, and response time ,as this architecture is used by big companies and many websites, we have models for our data, services to do operations on them , and controller for showing them to users.

5) Domain Requirements Specification

- System can't let users or admin accesses any page without login first even if he tries to write its link in internet browser search bar he will be directed to login page to complete login first.
- The user can add a book on condition that he has intellectual property rights
- user can make order and see all his orders and cancel the orders in any time
- user can buy any book with any quantity included between 1 and the book quantity on the system, if there is stock of the book
- there are ways for payment method and for every order the payment method and details will be stored and displayed and purchase page.
- two users can add and sell the same book if they have intellectual property rights.
- When a quantity of a book is over user can't add it to cart and the button will be disabled and turns to "out of stock"
- admin can see all user and delete any users upon need.

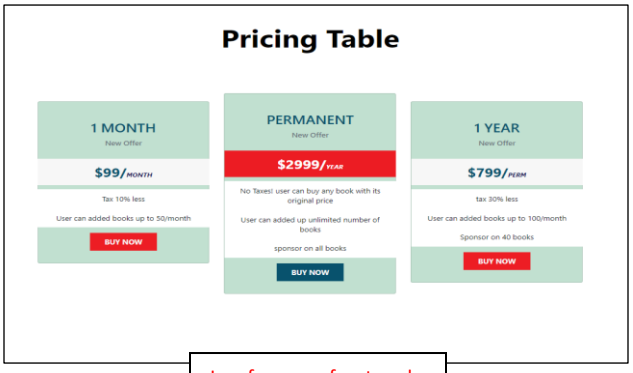
6) Design & Implementation constraints

- System back end implemented using PHP & MySQL.
- System Front End implemented by Html, Css, JavaScript, jQuery, and Bootstrap.
- Design must keep sure that it doesn't affect the system performance badly
- System design must be user friendly and provide preferences
- System must be secured as we got important information about each user this information includes visa information email, phone ...etc.
- We used **State** design pattern in the implementation to differentiate between user, admins and any new stack holder that may be added. these controlling include nav-bar changes and many other pages to be used by user, admin
- We used **factory** design pattern to achieve the re-use of code and make the design better as we grouped functions of Book, order, costumer classes in one class as the shape of factory design pattern.
- We used **MVC** pattern (model, view, control) to make the system easier and more collective as we use the control as an intermediate between view and models. so, when making an action in view model its effect will be made after the system call controller and then go to the model and get the result back
- We used Delegation design pattern to be able to use the methods of a class in another class by creating an object from this class in the method that we want to duplicate and use it better than copy, paste the function or inherit from the class.

7) System Evolution

a) Anticipated changes

1. **Subscribe**
   Soon we will put more features if the user made membership with us, the payment tax on each order will decrease or if he made permanent membership, he can buy books without any tax on any order



Img from our front-end

2. **Speed our site**

if we have many users in the website in the same time then the website will become slow, so we need to make the web faster more details about this how to make it faster in point b

3. **Increase website Storage**

now we work with low storage and will be ended soon

so, we need more storages to keep website work perfectly

4. **Payment methods**

the website has 2 ways to pay (visa or cash).

We will put more types to pay in the future like (PayPal, master, fawry).

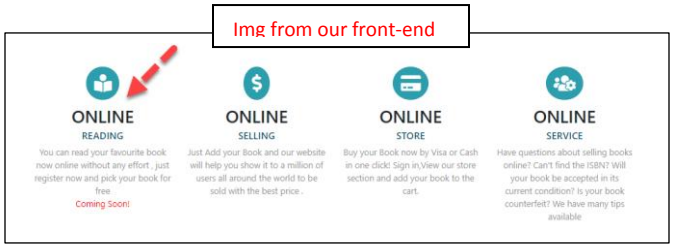
5. **Trade books(product)**

this feature means the customers can exchange books together without buy or sell the book

6. **online reading**

Users can pay for reading and downloading the books online

Img from our front-end



8) What are the requirements discovery approaches that you'll rely on?

The requirements discovery is about collection information for the requirements, collecting user and the system requirements and it used to prepare solution for the problem, example:

Stakeholder.

- User: the customers that will enter the website and select the books to buy or can add book to sell them.
- Admin: the manager that has the ability and responsibility see all the user information and all the transaction process

Interviewing.

- Closed interview: the list of pre-determined required example for user (register, login, view books, buy book, add books, upload books, my order, view buyer) and for admin (login, view books, view transaction, view user).

Scenarios.

- Are example of how the website will be used. so,
  For admin: the admin will login by enter valid credentials so, he can login to the system, he can check and see the books and all its details. he can also view the transaction process and its details. he can also see the registered users' information and can delete any user

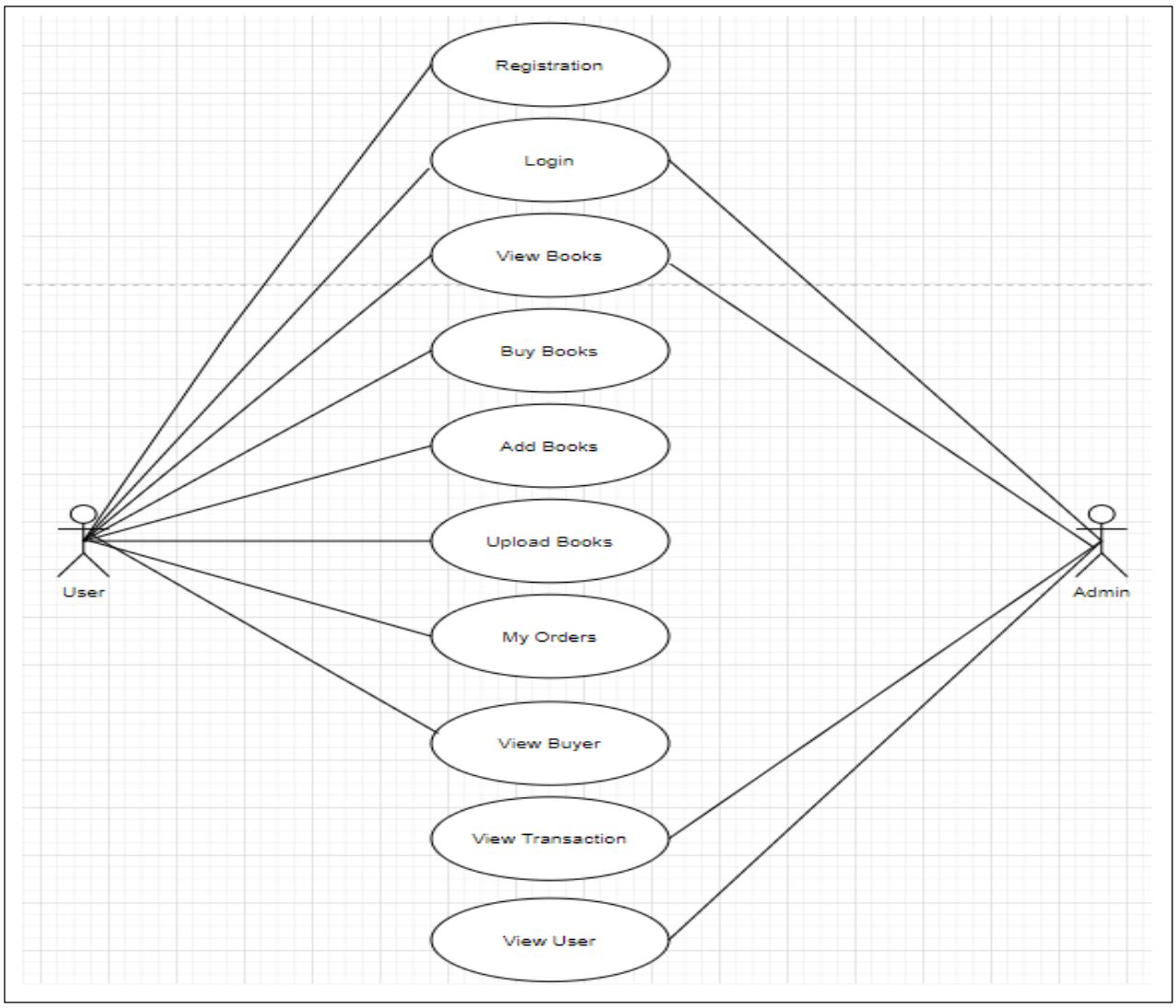
- For user: he will start by register then he will be able to login after that he can search for the category he want which will help to find the book then he can choose any books from different category or from the same category then he can add those to cart and complete order process.
  he can also add or upload books to be sold.

Ethnography.

- This type is get by see how the system work and doesn't need to ask to get function to help the system to become easier to use and more helpful example (we can make function to calculate the total of what the user buy from the website all the time).

Use Case:

- This type used to give the actor an interaction and describe the interface.



9) What are the requirements validation techniques that you'll employ/use?

It is the process of checking that requirements defined for development, it helps us define the system that the customer really needs. In order to check issues related to the requirements, we perform what is called **requirements validation**.

Now there are several techniques which are used either alone or with other techniques to check entire system or just a part of it.

We used a technique called **Prototyping.**

In this technique for validation the prototype of the system is presented to an end user, this end user experiments with the whole system and sees if it is what they need and if there is something missing or if there's an error. This technique is generally used for collecting feedback to help us find what customers need most.

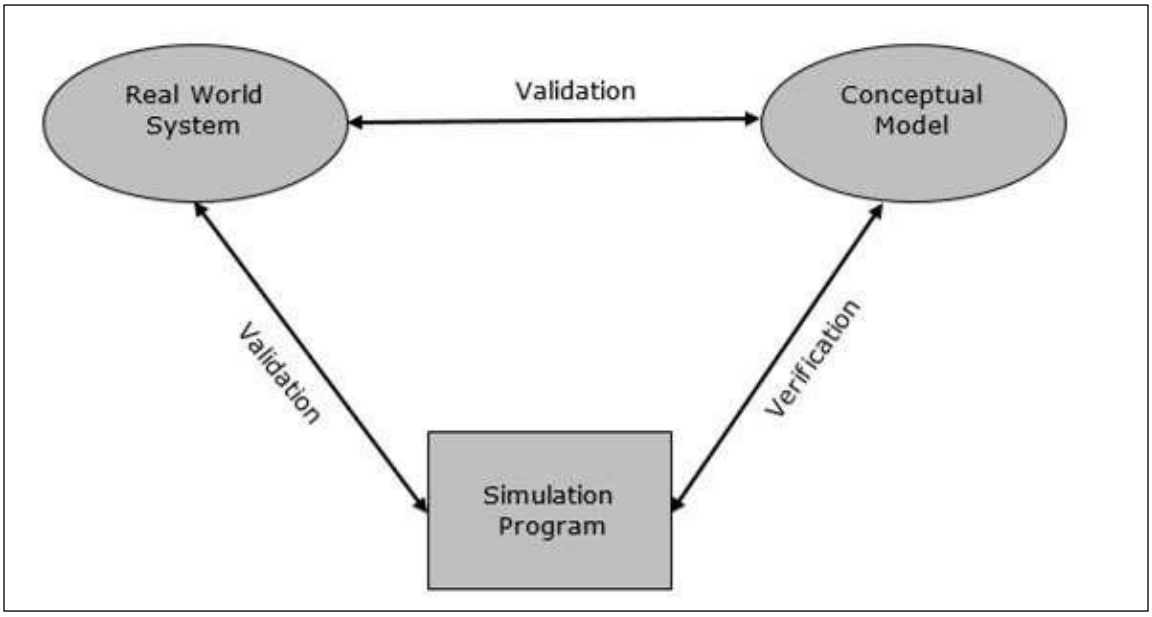
Another technique that we used is **Test Case Generation.**

All requirements are checked and tested to see if they have any errors either in the data the user entered or the requirement field itself. There are many checks in the SRS document which are testable, like Completeness checks, Validity checks, Realism checks and Verifiability.

with coding we've implemented a number of checks from the checks we previously mentioned, for example, in Completeness checks we check if there is missing input data entered by the user such as (Name, email, Credit

Card and other info) and if there is something missing the user is faced with a message explaining what is missing in his data. Another check we used is the Realism check, which checks if the data entered is imaginary or unreal, for example, if the user entered that he was born in the year 2020 then he will receive a message telling him that the data is not real. We also implemented Verifiability checks in order to verify user's data, for example, passwords and usernames, we check if the username and password are stored in our database, or if a user was trying to create an account for the first time he is asked to enter a password and verify it.

Overall, we check every possible requirement and see if there any errors or bugs to help us fix any problems that we face.



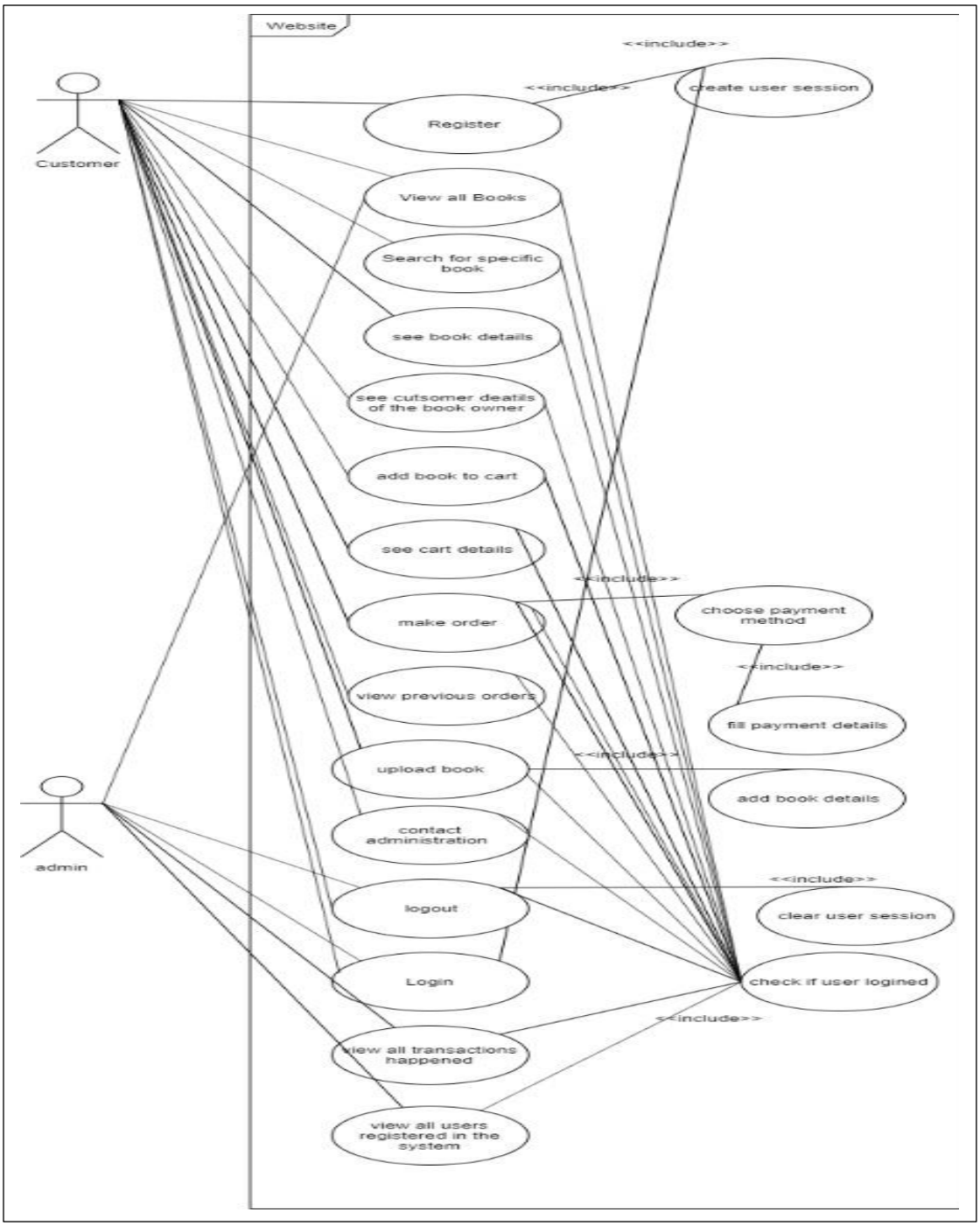
END OF PART 1

PART 2

System Design & Models

10) Functional Diagrams

a) Use-Case Diagrams including all the Use-Cases for the System



b) Detailed Use-Cases Description

- customer registration case

ID:	**01**
Title:	Customer registration
Description:	for someone to use our app, he must register first
Primary Actor:	Customer uses the app for first time
Preconditions:	none
Postconditions:	the user will have an account for our app in order to use it.
Main Success Scenario:	the user enters the required information correctly, press register button , create user session with the user id, and then navigate to the home page
Extensions:	show error message if the user didn't enter correct data
Frequency of Use:	often

- view all books case

ID:	**02**
Title:	view all books
Description:	view all books in the system.
Primary Actor:	customer or admin
Pre-conditions:	must have account and logged in the app
Post-conditions:	have a list with all the books in the system
Main Success Scenario:	user (customer or admin) enters the home page, which requests all the books in the system to see it
Extensions:	if the user isn't logged in, the app will navigate to the login screen
Frequency of Use:	always

- search for a book use case

ID:	**03**
Title:	search for a specific book
Description:	the customer can search for a specific book in the system by its name the result will contain all the books that contains the string he entered
Primary Actor:	customer
Preconditions:	the customer must be logged in the app
Postconditions:	a list will all the books that contains the search string
Main Success Scenario:	the user enters the name he wants, gets the list of the result, find the book he wants
Extensions:	none
Frequency of Use:	often

- book details case:

ID:	04
Title:	show book details
Description:	the customer should be able to see the book details of any book, the details includes data like the book owner name, isbn, price ,category , etc.
Primary Actor:	Customer
Preconditions:	customer must be logged in the system; the book must exist in the system
Postconditions:	the app should be in the book details page with the correct information for the book
Main Success Scenario:	the user opens the home page. the user sees a book and presses on it. the user sees the book details of the book that he pressed on.
Extensions:	error page will appear if the book is not found on the system
Frequency of Use:	often

- Book owner details case:

ID:	**05**
Title:	see the details of the book owner
Description:	the user should be able to see who uploaded or added this book to the app.
Primary Actor:	Customer
Preconditions:	the user should be signed in, and the book should have book owner The book owner should exist.
Postconditions:	the customer should be in the page of the book owner details
Main Success Scenario:	user press on book details. the name of the owner appears in the details. the user presses it. the app navigates to the page of the book owner
Extensions:	if the book owner isn't found the error page will appear.
Frequency of Use:	unlikely

- add book to cart case:

ID:	06
Title:	Add book to cart
Description:	the customer should be able to add book to his cart, if it's already there it will increase the total number of books
Primary Actor:	Customer
Preconditions:	customer should be logged in the system; the book exists on the system
Postconditions:	adding the book to the user's cart, updating the total price of the cart.
Main Success Scenario	the user presses the add to cart button of a specific book. the app adds the book to the cart and updates it the app navigates to the cart page and shows the updates.
Extensions:	checks if the customer logged in or not
Frequency of Use:	usually

- Cart details case:

ID:	**07**
Title:	show Cart details
Description:	the customer should be able to see his cart details, the books inside it , the total price etc..
Primary Actor:	Customer
Preconditions:	the customer should be logged in the system
Postconditions:	the customer should see his cart with its details
Main Success Scenario:	the customer presses the cart button. the customer sees the cart details and total price.
Extensions:	none
Frequency of Use:	usually

- Make order use case:

ID:	**08**
Title:	Creating order
Description:	the customer should be able to create an order with the books inside his cart
Primary Actor:	Customer
Preconditions:	customer must be logged in the system, has at least one book at his cart, fills the order details
Postconditions:	creates an order with the books inside the cart, with a report
Main Success Scenario:	the user presses make order. the customer fills the order details. the customer presses confirm order. the app shows a success message.
Extensions:	checks if the customer logged in the system and has at least one book in his cart
Frequency of Use:	often

- view past orders use case:

ID:	**09**
Title:	view past orders
Description:	the customer can see the past orders he made with its details
Primary Actor:	customer
Preconditions:	the user must be logged in the system
Postconditions:	the user sees his past orders
Main Success Scenario:	the user presses the create order button. the user fills the order detailsinformation. the user presses confirm button. success message appears to the user that the order is created successfully
Extensions:	none
Frequency of Use:	often

- upload book use case:

ID:	**10**
Title:	upload book use case
Description:	the customer uploads the book in pdf format and its details
Primary Actor:	customer
Preconditions:	user logged in the system, has the pdf file of the book, has its details
Postconditions:	the book is uploaded to the system with its details, other customers can view and buy it
Main Success Scenario:	the customer presses upload book. the user chooses the pdf file and uploads it. the user is asked to fill book details. the user presses add.
Extensions:	show validation error messages if the users leave empty or invalid fields that are required
Frequency of Use:	often

- Contact administration use case:

ID:	**11**
Title:	Contact administration
Description:	the customer can contact the app administration for any feedback or issues that they want to submit.
Primary Actor:	customer
Preconditions:	user must be logged in the system
Postconditions:	the message is sent to the administration and success message is shown to the customer
Main Success Scenario:	the customer presses contact button. the app navigates to the contact form. the user fills the contact information. the user presses the sent button. the message is sent successfully.
Extensions:	None
Frequency of Use:	not often.

- View all transactions use case:

ID:	**12**
Title:	view all transactions
Description:	the admin can see all the transactions happened on the system
Primary Actor:	admin
Preconditions:	the admin must be logged in the system
Postconditions:	a list with all transactions with its details is shown to the admin
Main Success Scenario:	the admin presses the history button and sees all the transactions
Extensions:	none
Frequency of Use:	usually

- view all customers use case:

ID:	**13**
Title:	view all customers
Description:	the admin can view all customers that are registered on the system
Primary Actor:	Admin
Preconditions:	admin has to be logged in the system
Postconditions:	a list with all customers in the system
Main Success Scenario:	the admin presses the customers button. a list with all customers is shown to the admin
Extensions:	none
Frequency of Use:	usually

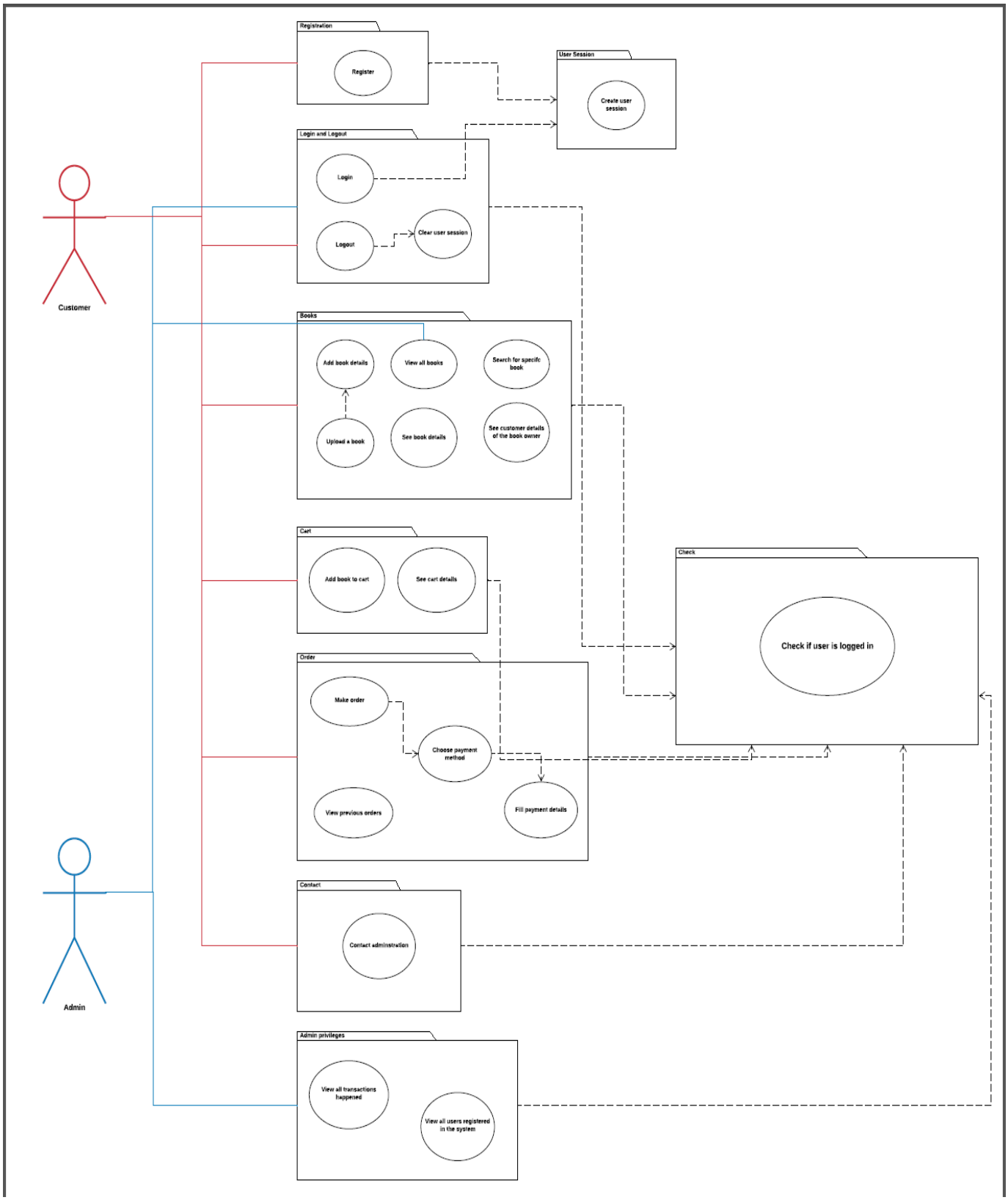
- Delete customer use case:

ID:	**14**
Title:	admin deletes customer
Description:	the admin can select customer from the customers list and deletes him
Primary Actor:	admin
Preconditions:	the admin must be logged in the system. the customer he wants to delete must be registered on the system.
Postconditions:	a success message that the user is deleted. a new list of customers without the deleted one
Main Success Scenario:	the admin opens the list of customers. the admin chooses the customer he wants to delete. the customer is deleted successfully.
Extensions:	none
Frequency of Use:	not often

- login use case

ID:	**15**
Title:	user login
Description:	the user can login with his account into the system
Primary Actor:	admin or customer
Preconditions:	must have an already existed account. there's no open session in the app at the time of login, i.e : he is not logged in yet.
Postconditions:	the app navigates to the home page and creates a session for the user.
Main Success Scenario:	the user (customer or admin) enters his login details. presses login button. the login is success and the app navigate to the home page
Extensions:	checks if the user email and password are valid or not
Frequency of Use:	always

c) Package Diagram grouping relevant Use-Cases into Packages

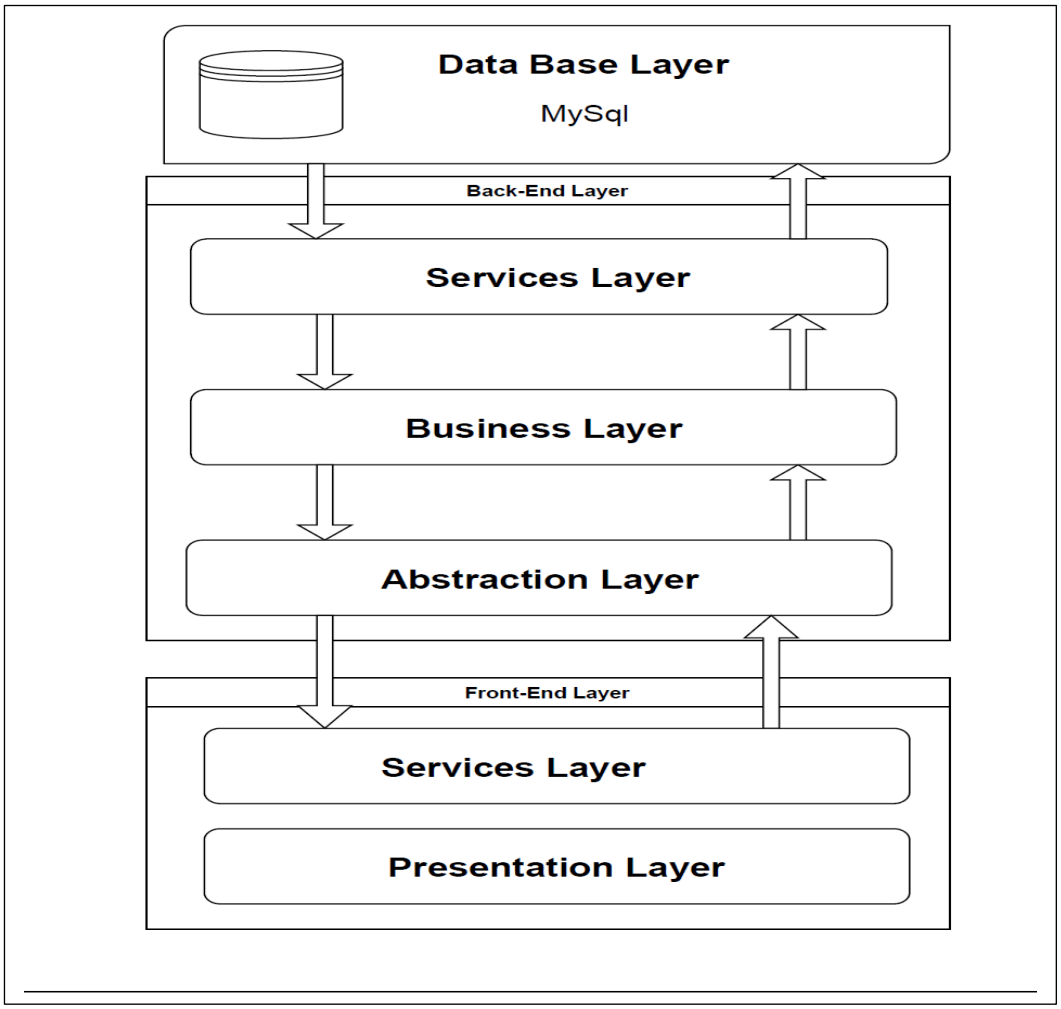


11)Structural & Behavioral Diagram

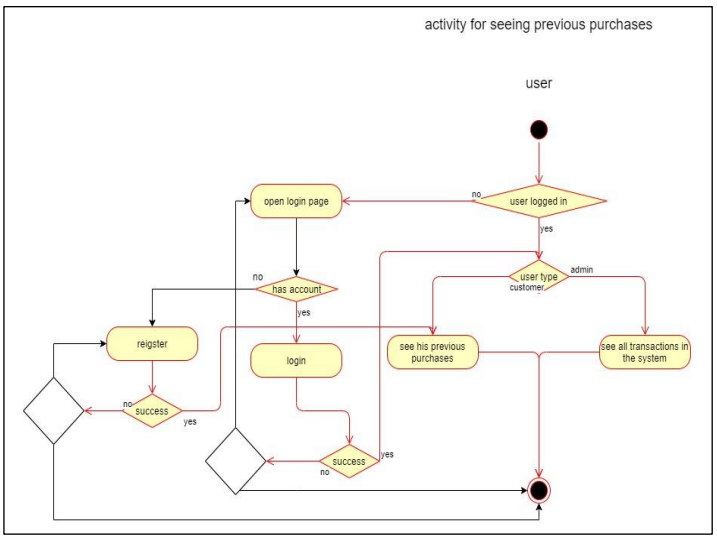
a) System Architecture

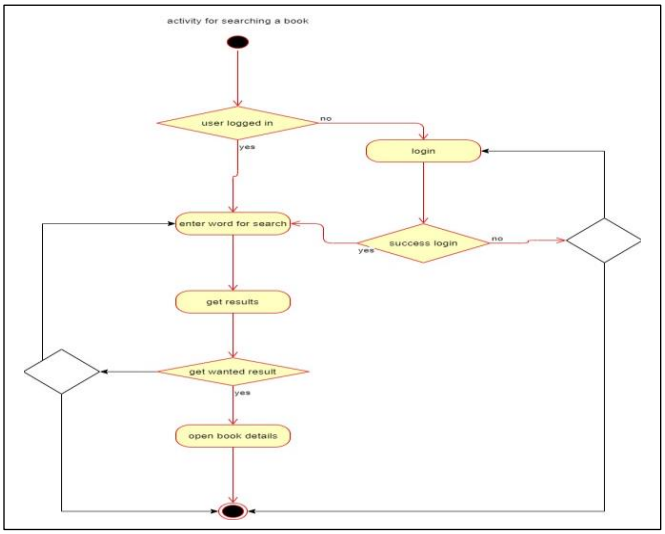
we used the MVC architectural pattern for our app , MVC stands for Model View Controller, models are the data representation that's used in the app , View is the UI that's appeared to the user of the app , Controller is the link between Model and View , it sends that data to the View , and sends actions and mutate the state of the model .

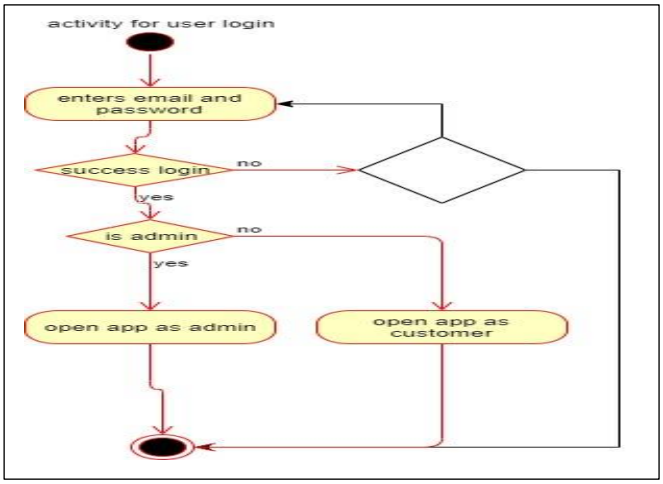
MVC keeps the code clean, easy to maintain and to understand, and its very popular pattern in web development so we adopted this pattern to be consistent with the industry, and any other developer can get familiar with our project as soon as possible.

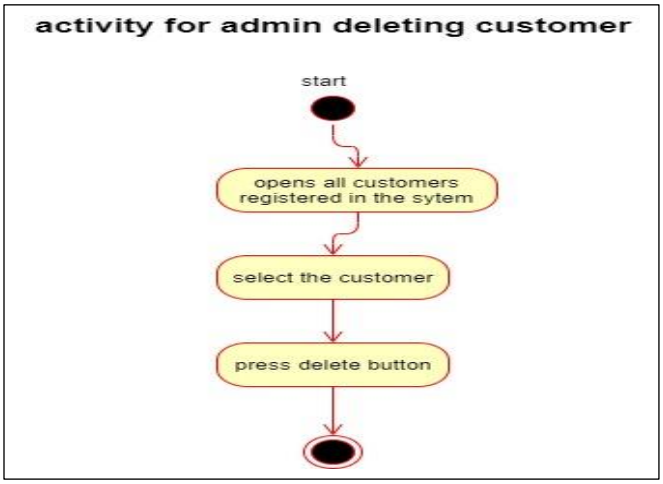


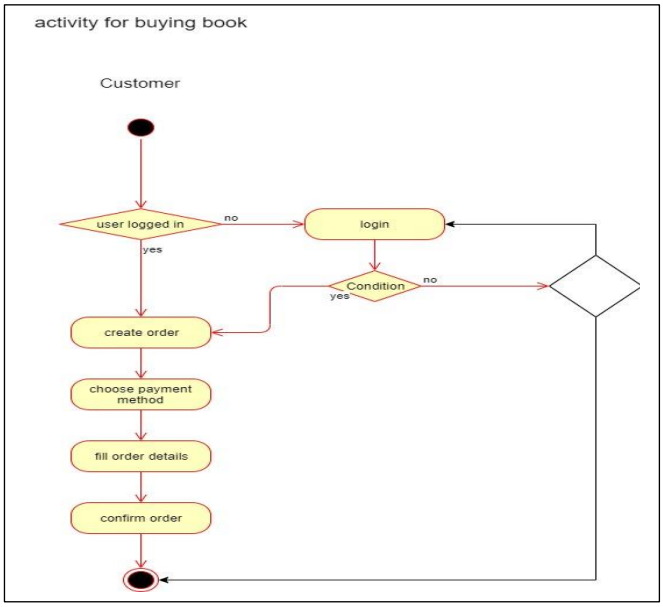
b) Activity Diagrams

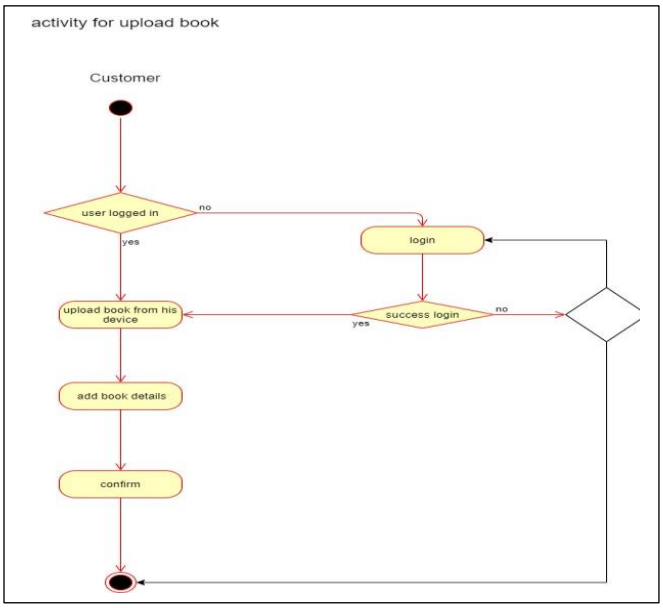






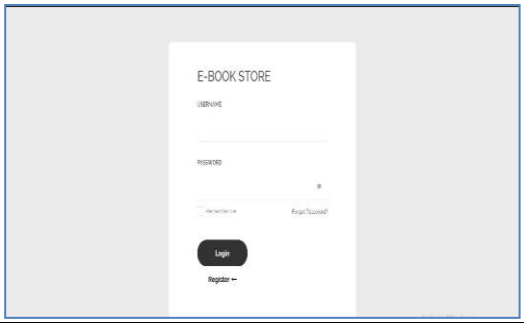
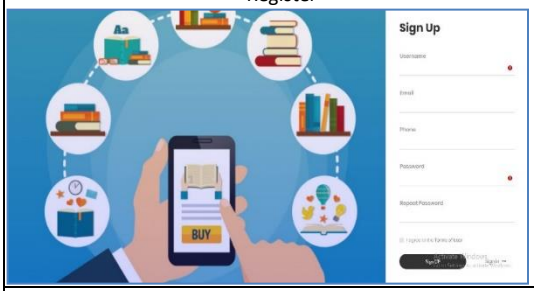
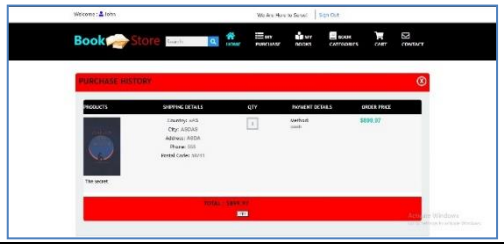
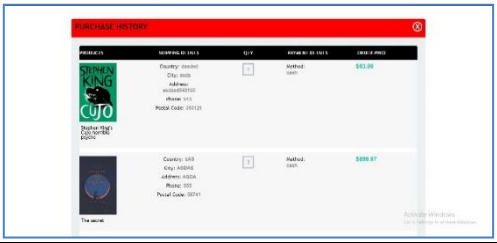




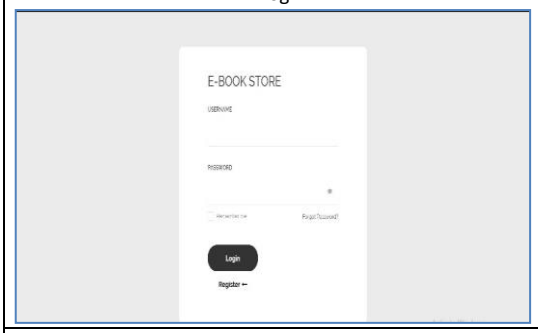
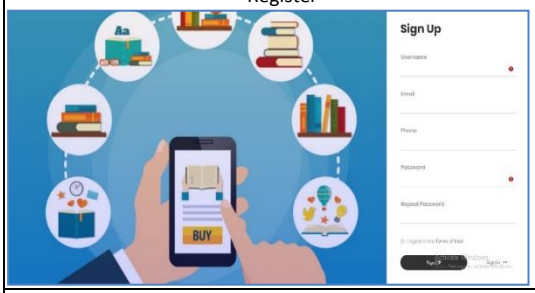
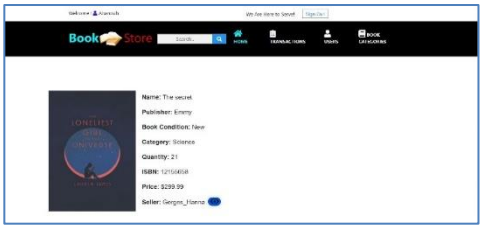


c) Based on the Activity Diagrams, the List of User Interfaces required for the System, and the corresponding users of each interface

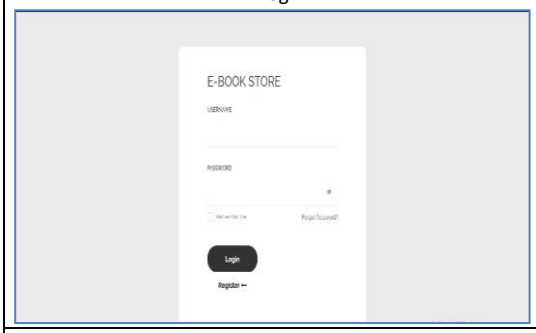
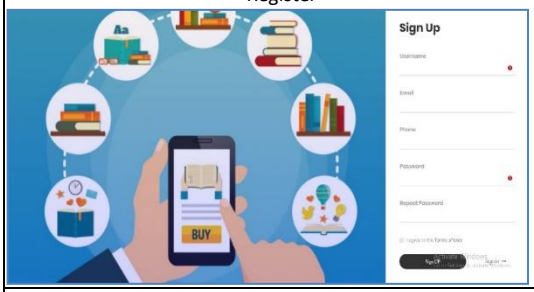
View Purchase History Activity

Login 	**User & Admin**
Register 	**User**
View previous Purchase 	**User**
View all transactions: 	**Admin**

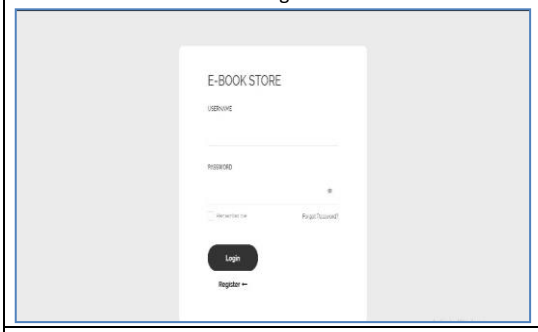
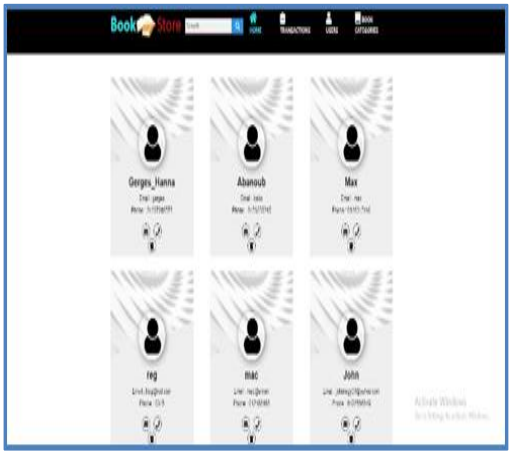
Searching Book Activity

Login 	**User & Admin**
Register 	**User**
Search by search bar 	**User & Admin**
Result of search 	**User & Admin**
See Book Details 	**User & Admin**

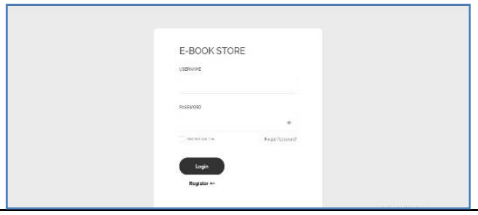
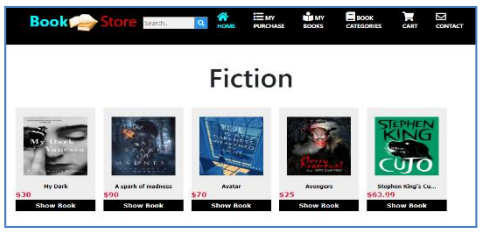
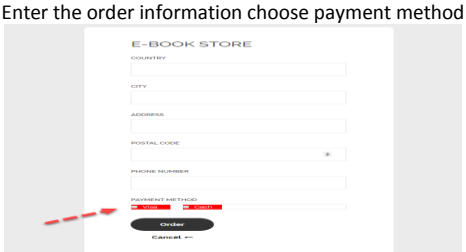
User Login Activity (check admin or user

Login 	**User & Admin**
Register 	**User**
login as user 	**User**
login as Admin 	**Admin**

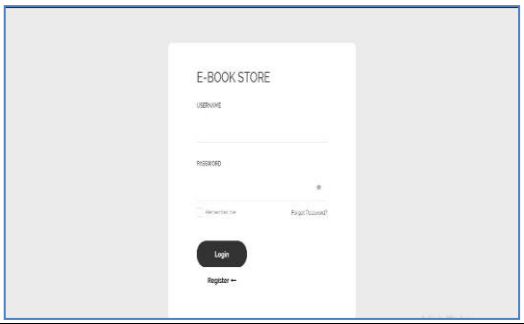
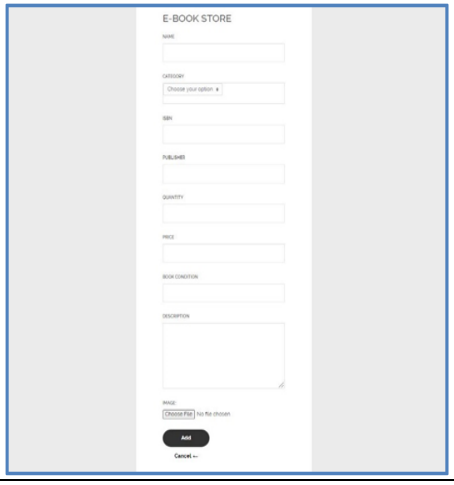
Delete User Activity

Login 	**User & Admin**
Log in as Admin 	**Admin**
Select the user and click on trash button 	**Admin**

Buying Book Activity

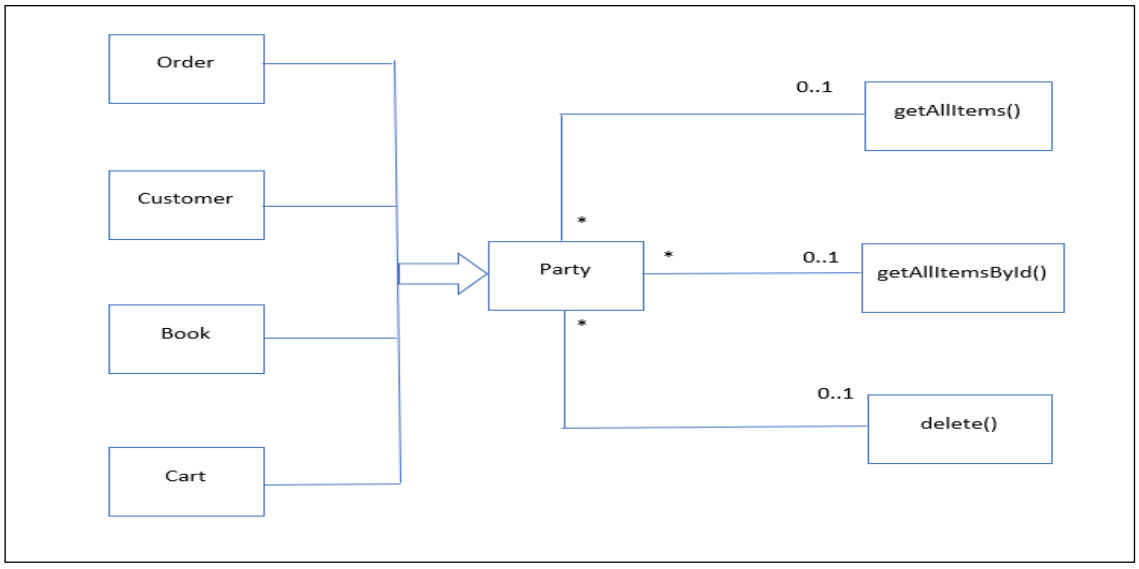
Login 	**User & Admin**
Register 	**User**
Select Book Category or show all books 	**User**
Select Book 	**User**
Required quantity click add to cart 	**User**
Go to Cart and Check the details then click Order 	**User**
Enter the order information choose payment method 	**User**

Add, Upload Book Activity

Login 	**User & Admin**
Register 	**User**
Click on add book on categories page 	**User**
Enter The book information, Click Add 	**User**

d) What is a Software Analysis Pattern? Apply at least one analysis pattern while analyzing and designing your system

- The analysis pattern is group of concepts that represents the common construction in the system.
- The type of analysis pattern:
  1. Accountability: this type used to describe the relation and responsibilities between the parties.
  2. Observation and Measurements: this type used to the facts.
  3. Referring to objects: this type used when referring to object.
  4. Inventory and Accounting: this type used to record the way how the money and goods are moved between the parts and to make sure to record in the properly way.
  5. Planning: this type used to describe the plan and protocol that used to record the resources.
  6. Trading: this type used in the situation where the price is Changeable, and we need to understand how the price will affect the profit.
- The analysis that will be used is Accountability and the pattern called party:
  - The problem: we have many classes have the same the function.
  - The solution: create common class has this function for this classes.

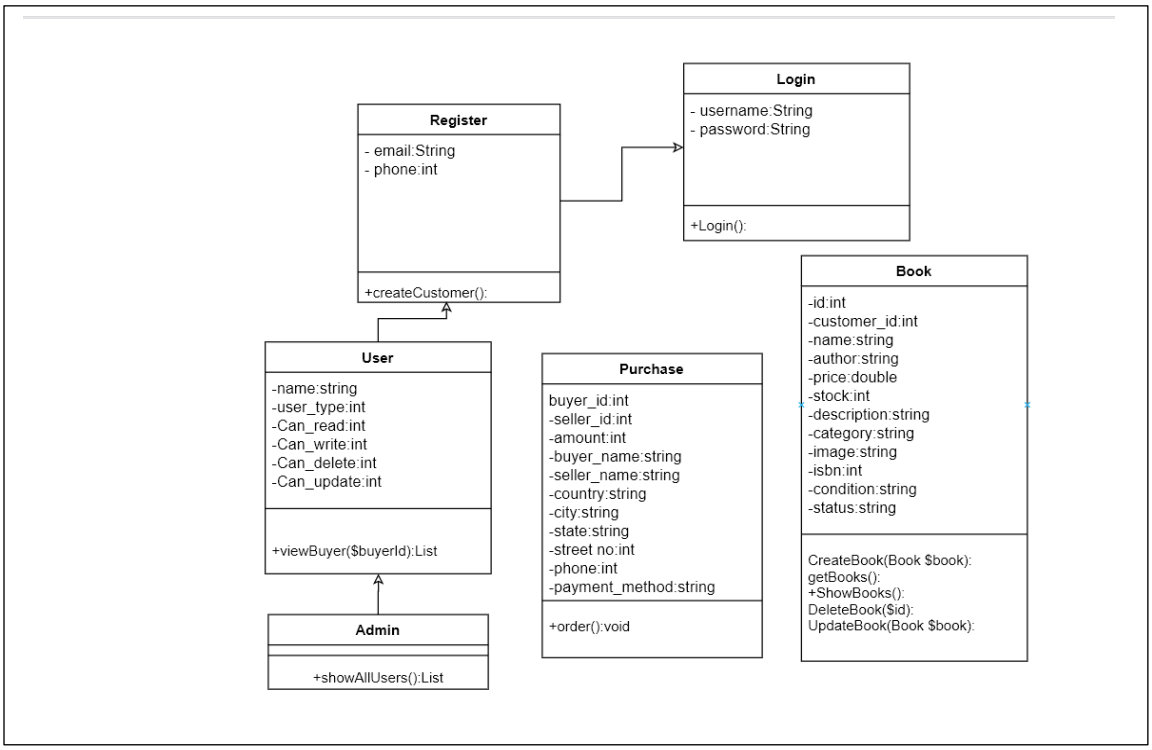


NOTE: We used factory design pattern to achieve this in our code

e) Class Diagram 1: An initial version based on the requirements and Use Case/Activity diagrams

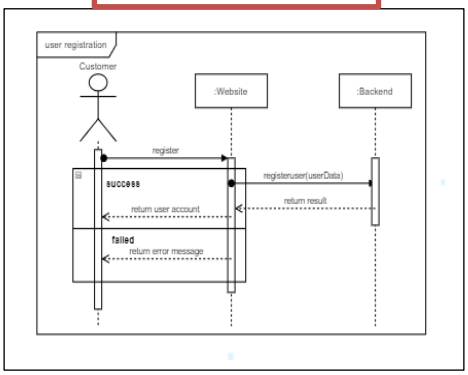
we created the first-class diagram initially based on requirements so, it's very simple where it only consist of the initial attributes and basic operation needed to achieve the requirements

Class Diagram V1

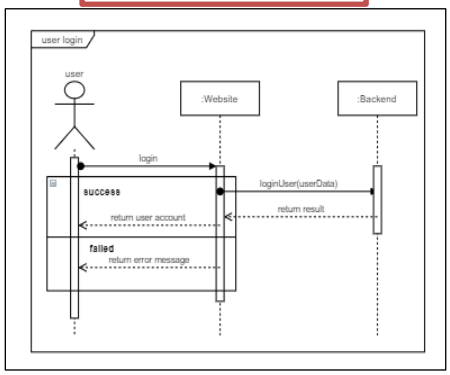


f) Sequence Diagram(s)

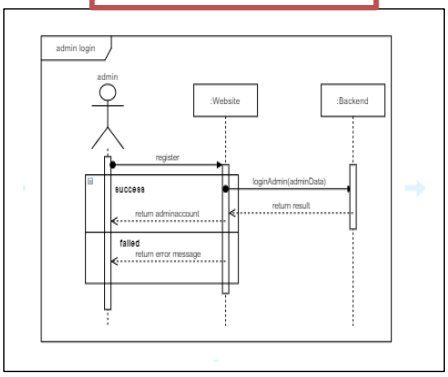
User Registration



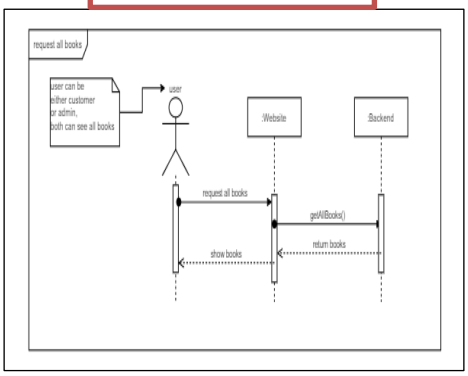
User Login



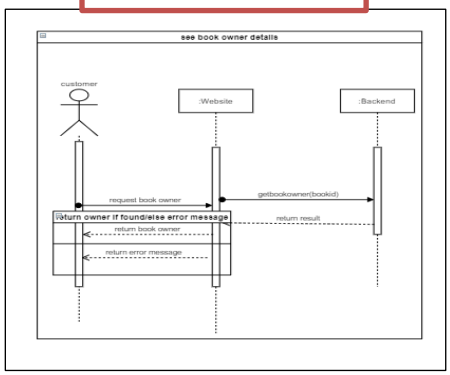
Admin Login



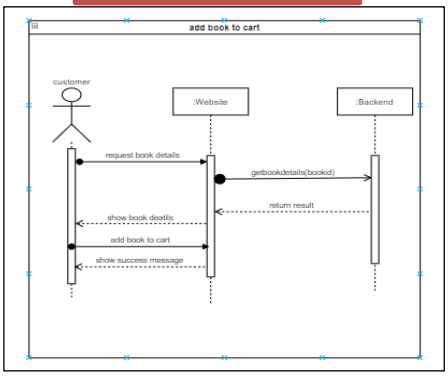
Request All Books



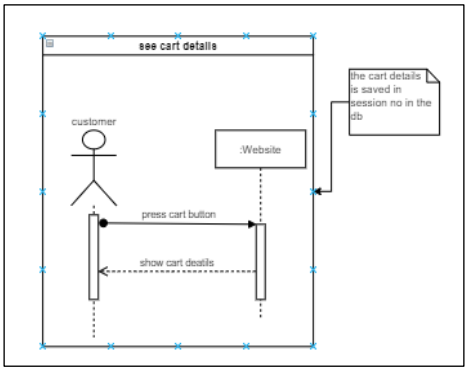
See Book Owner Details



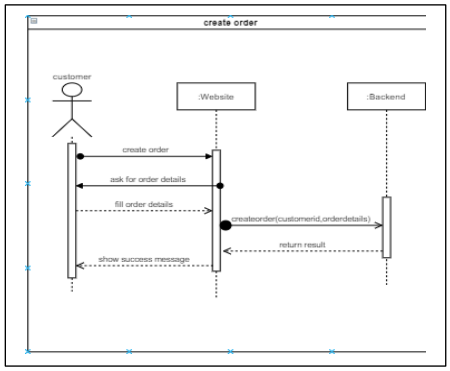
Added Book to Cart



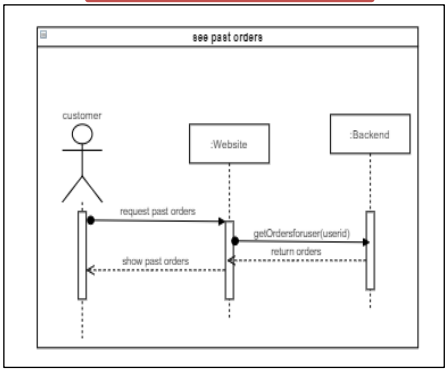
See Cart Details



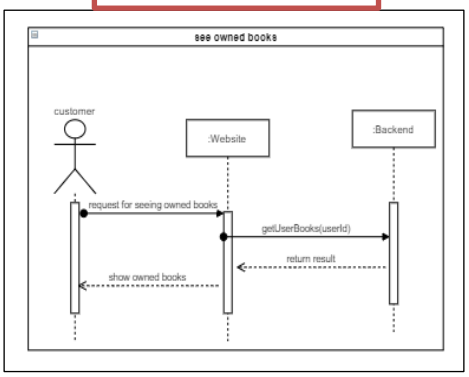
Create Order



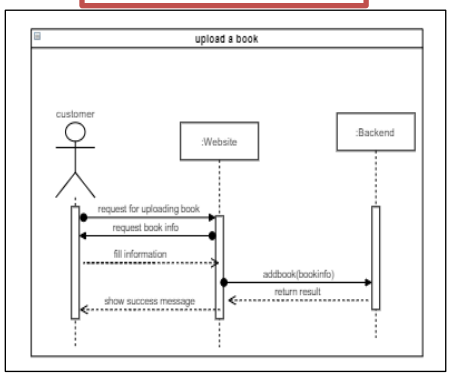
See Past Offers



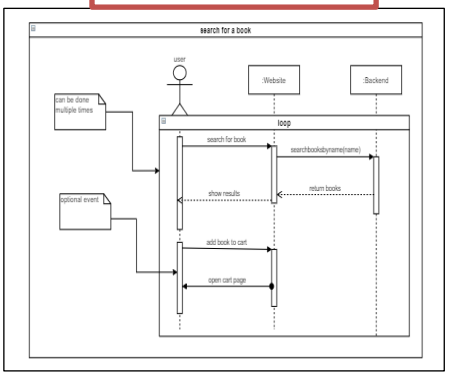
See Owned Books



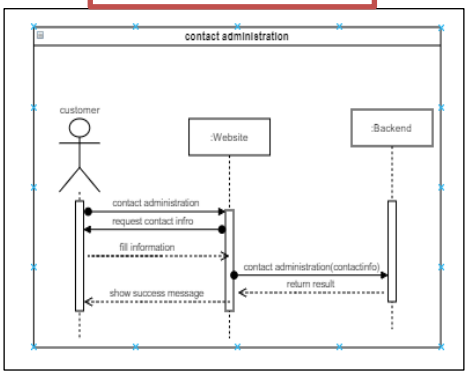
Upload a Book



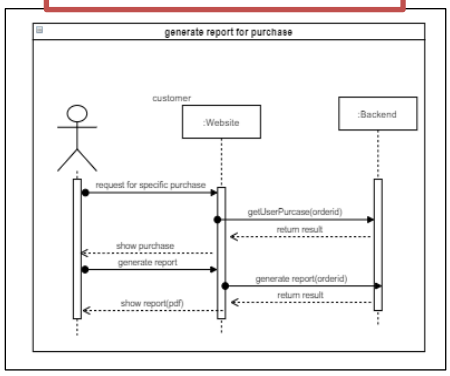
Search for a Book



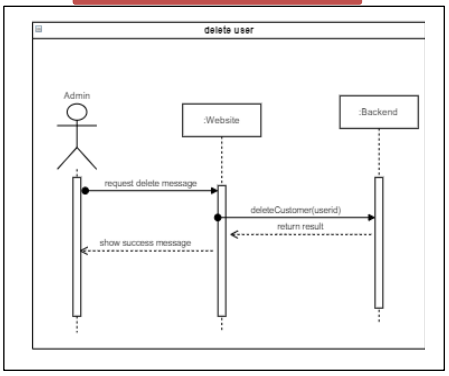
Contact Administration



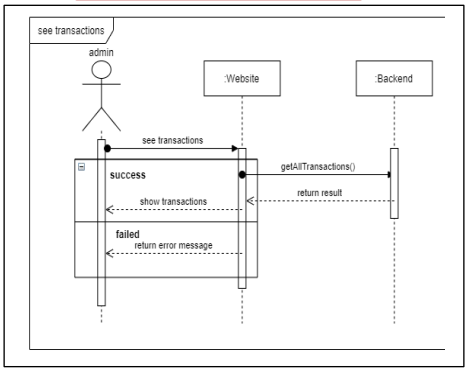
Generate Report for Purchase



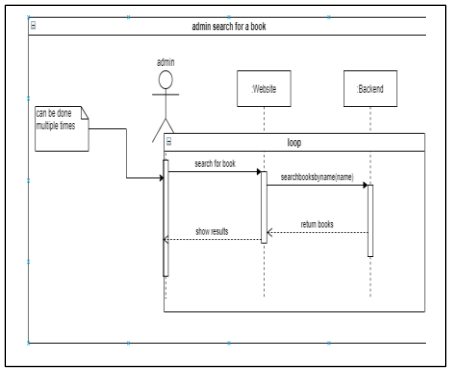
Delete User



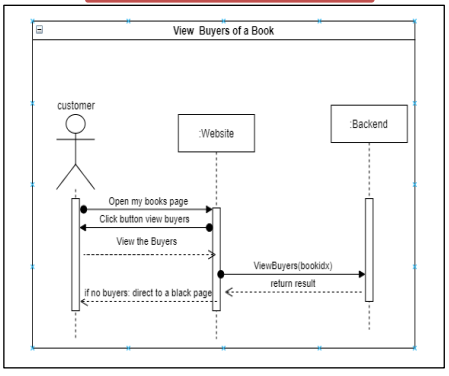
See Transactions



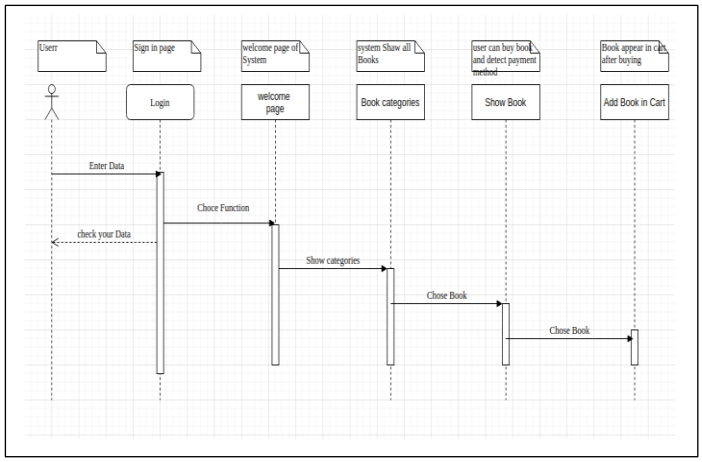
Admin Search for a Book

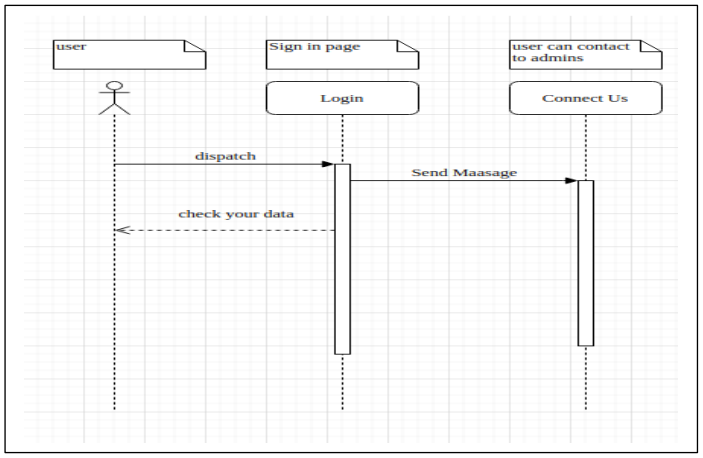


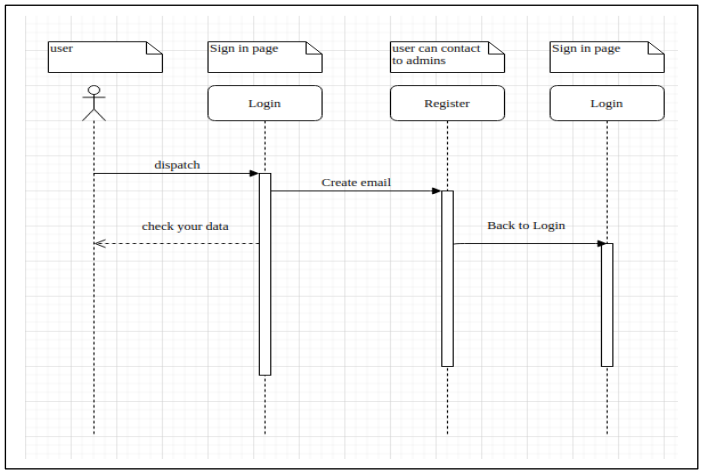
View Buyers of a Book

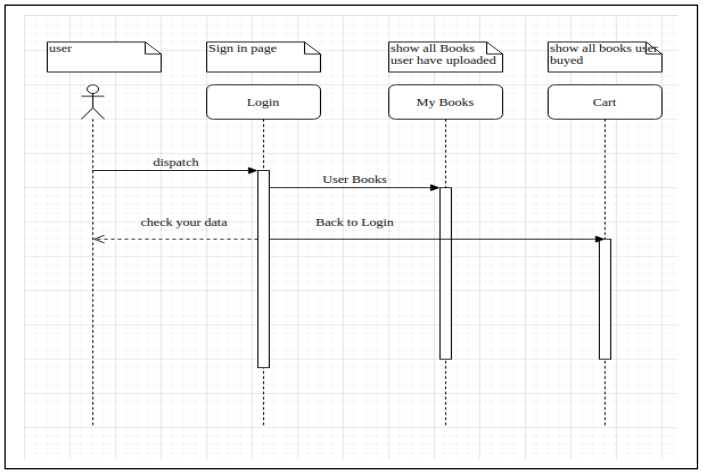


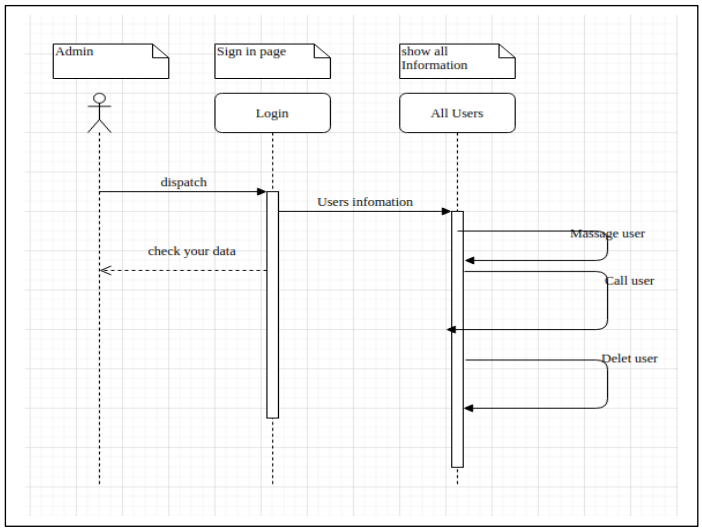
g) System Sequence Diagrams (SSDs)

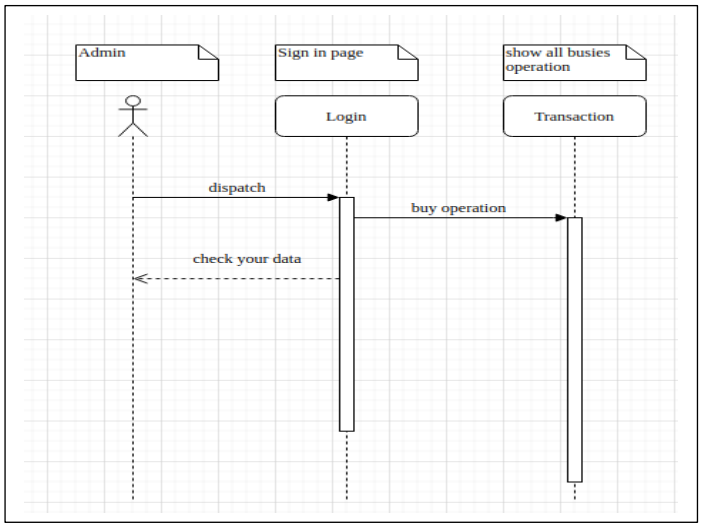




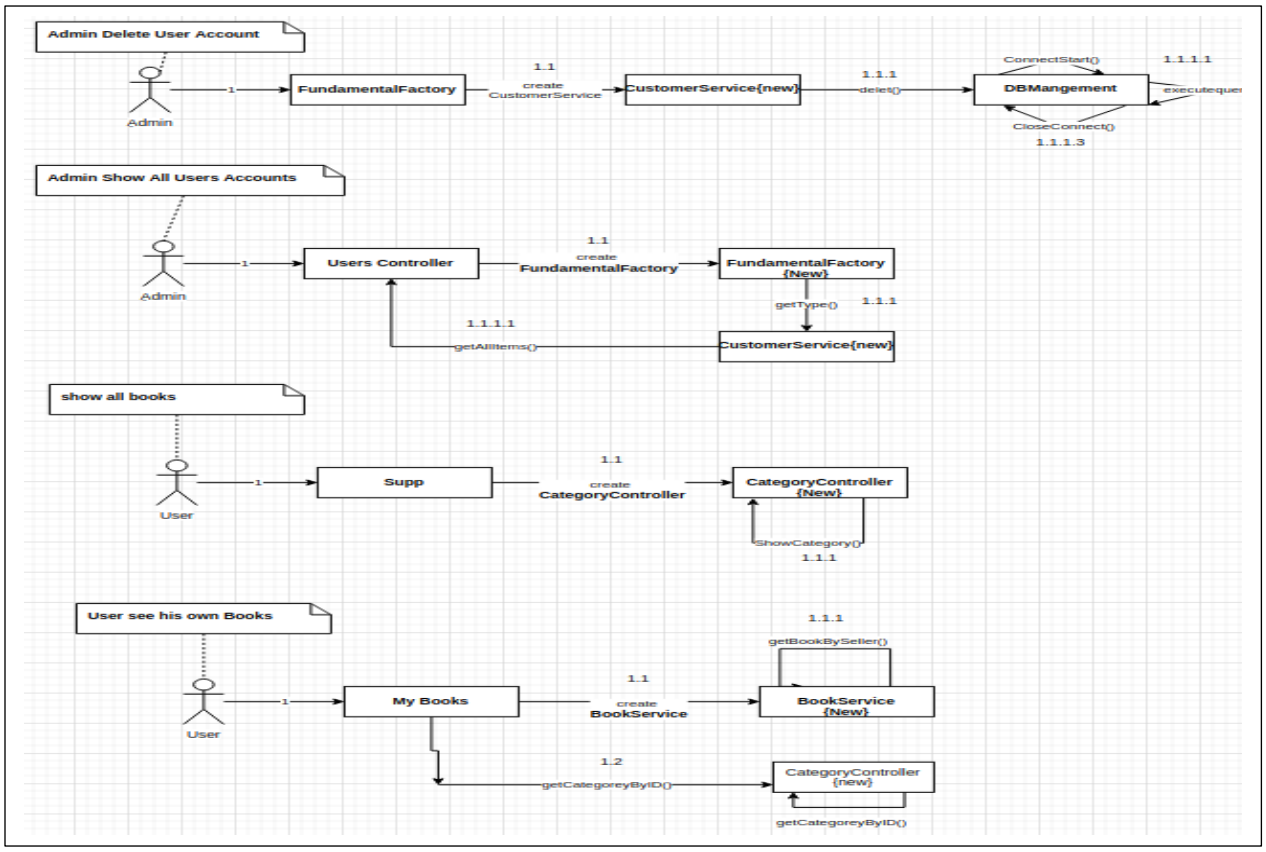








h) Collaboration/Communication Diagram(s)



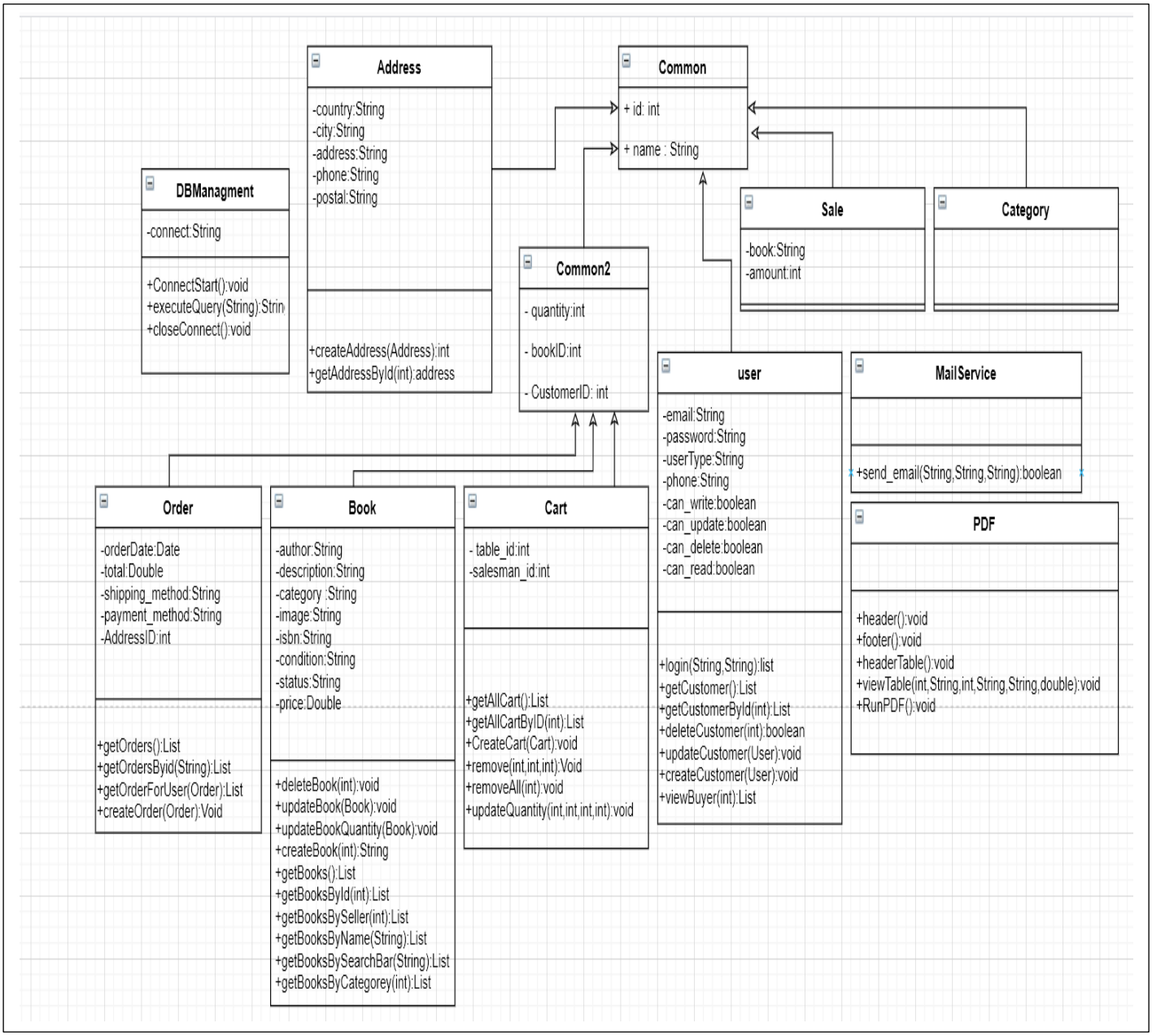
i) Which strategy (or strategies) did you use to implement the use-cases: One Central Class, Actor Class, or Use-Case class.

we used the user case owner strategy in our use case implementation ,we wanted the use case diagram to be clear for the customers, we made it for them, so it should be from their perspective.

its advantages:	Its disadvantages:
1. Customer friendly 2. Easy to understand 3. straightforward	1. Not clear to developers 2. Lake of data representation or visualization in some cases

j) Class Diagram 2: An intermediate version based on the interaction diagrams

We have created class diagram 2 before we apply design patterns and the use the OOP properties so, it contains attributes and classes, some more operations added after version 1.



k) Four Design Patterns Applied

Definition: Design pattern is a well-described and good solution for the common software problems, and it's very popular between developers.

We use design patterns to save time and reduce effort because it's already defined before in java and it provides industry standard approach for a specific software problem to be solved, it depends on reusability of what we have to reduce total cost, and because they are already defined they make code easy to understand and make it highly maintainable.

Categories of design patterns: Creational DP, Structural DP, Behavioral DP

In our project, we used four patterns that will be discussed in the next lines

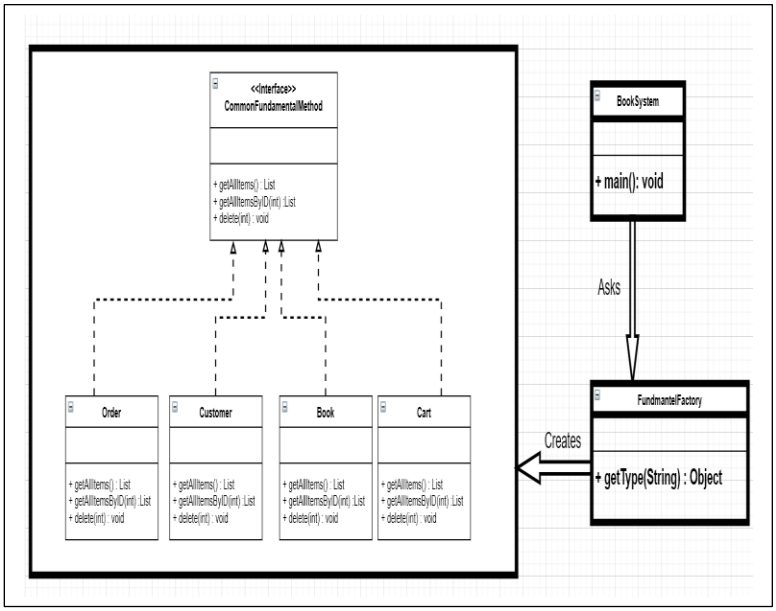
Factory pattern

Description: the aim of this pattern is to get a copy or instantiation from the client program to the factory class.

Problem it solves used when we want to return one sub-class from multiple sub-classes that is grouped in one super class based on input.

Affection on my system design:

We used factory pattern factory pattern to get a copy from the methods in the figure and group them in a CommonFundemntalMethod class to share them with the sub classes Order, Customer, Book, Cart. better than declare them in each class and this how to make a re-usable code

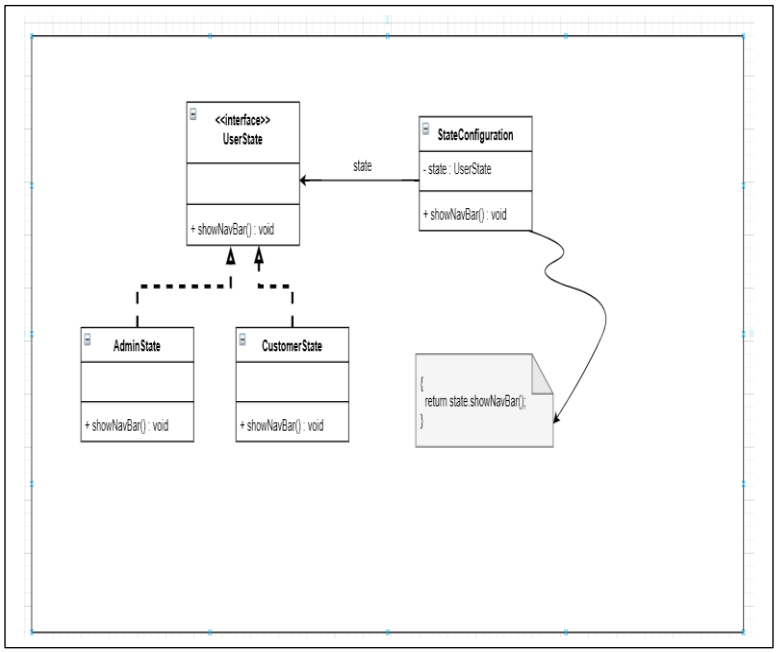


State pattern

Description: State design pattern is used if we have to change the behavior of an object or variable based on its state, we can have a state variable in the Object and use if-else condition block to perform different actions based on the state. State pattern is used to provide a systematic and loosely coupled way to achieve this through Context and State implementations.

Problem it solves very useful at facing problems that related to a state of an object in a part of runtime , the most common example on it is the Tv remote that when we click on its button state of tv changed to turn on , then if we click it again state will back to turn off . with this concept it can help in so many problems that the system want to change a part or a state in it.

Affection on my system design:



We used State design pattern in differentiate between system stack holders (admin, user) as we have a column name “user-type” in the user table in our database this column hold a value for each one who use our system we set the value of user-type of admin equal to 1 , and 0 for users .

So, we used state design pattern to hold the value of object usType and change project functionality according to this value

as example: the navbar of the pages changes according admin or user as shown in the figure. Also, we disable add to cart button for admin using usType value.

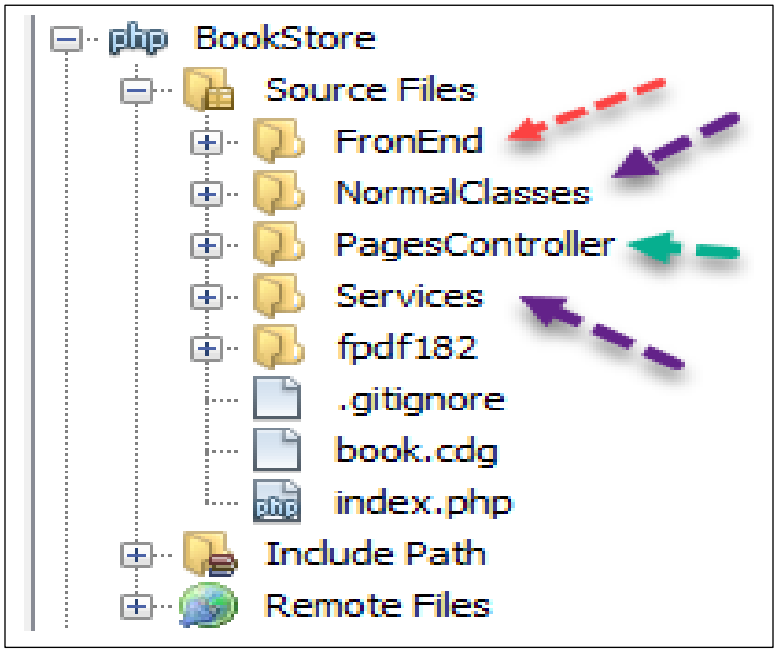
MVC pattern

Description: The MVC architecture pattern is used from a long time ago in software engineering, where almost used by all languages with a little bit different technique.
MVC stands for Model, View, Control where it separates the application into different three parts.

View is the UI that's appeared to the user of the app, Controller is the link between Model and View, it sends that data to the View , and sends actions and mutate the state of the model .

Problem it solves MVC keeps the code clean, easy to maintain and to understand, and it's very popular pattern in web development

Affection on my system design:



We used MVC as shown in the figure we make the package Normal Classes, Services represent the Model
and Front end package represent the view as it holds everything related to shape of the system
and package Pages Controller represent the Controller of the MVC we made this to make the system easier and more collective as we use the control as an intermediate between view and models also, to ensure high loading, and response time ,as this architecture is used by big companies and many websites

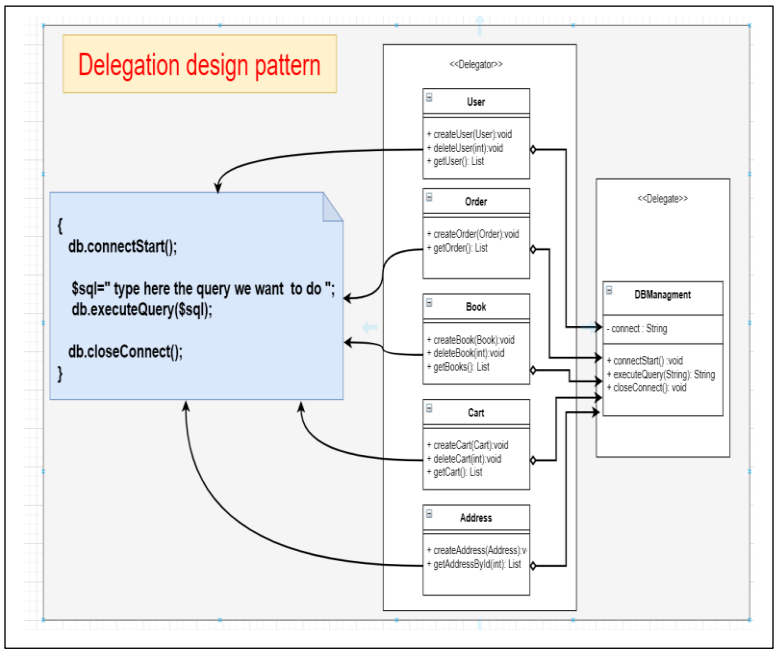
Delegation pattern

Description: Delegation Pattern abstract methods and functionality of a class into another class, not only this but the real power of this pattern comes when there are multiple delegates. The delegator typically has a method for each delegate that will convert the delegator to use that delegate.

It is useful for understanding to compare the delegation pattern to inheritance. Both are powerful reuse techniques with a few of key differences; inheritance is directly supported by today's object-oriented programming languages and enables the use of polymorphism, whereas the delegation pattern allows the delegate to be changed at run-time.

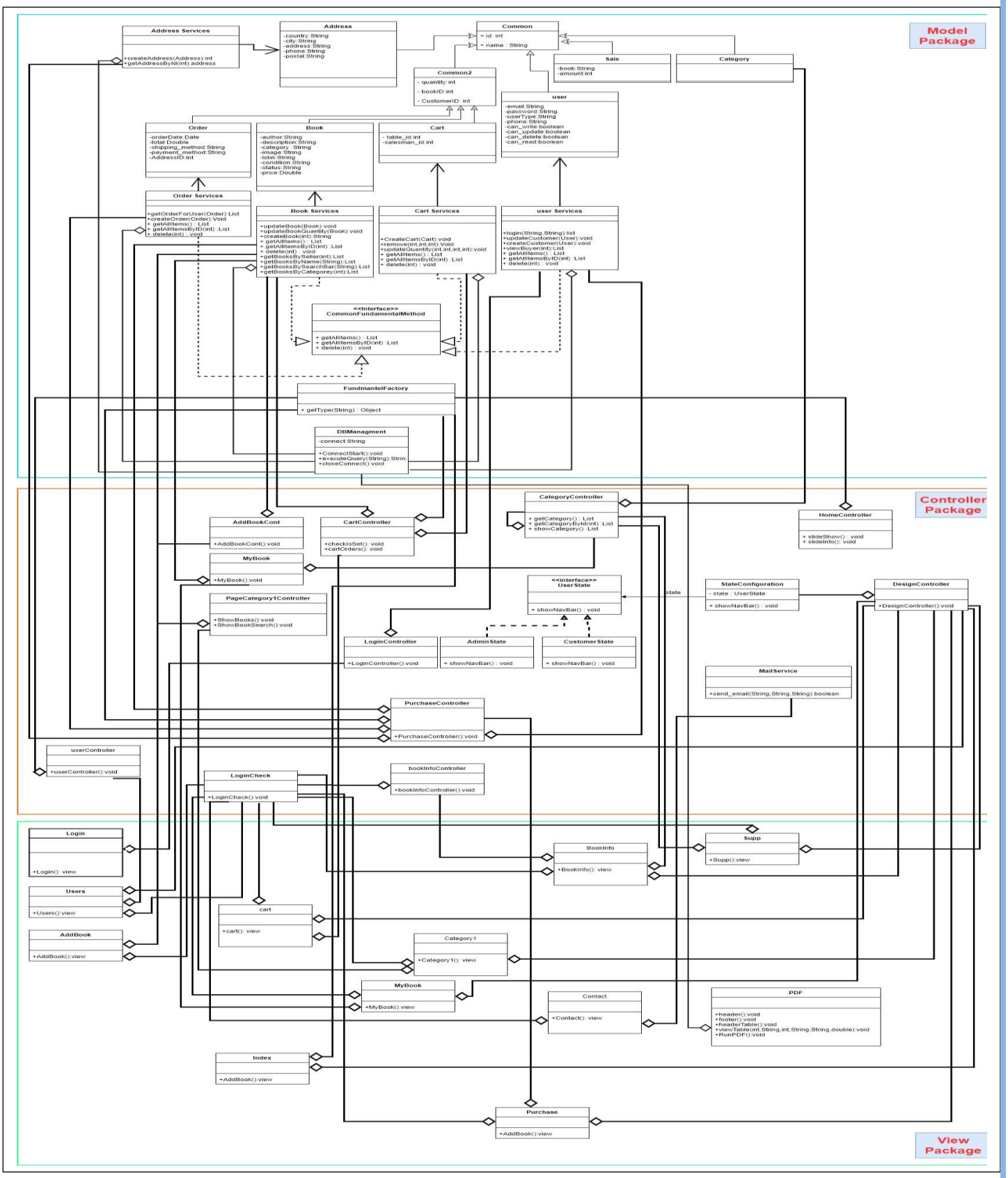
Problem it solves very useful to decrease system run time, increase its flexibility and separates the different set of methods

Affection on my system design:



From the description its clear that we use this pattern when we want to abstract methods of a class so we create object from this class into the classes where we want to use those methods so, as shown in the figure we have class DB management which open connection with database so, this method is very useful and nearly used in every part of the project that why we used delegation pattern here to use these method in another classes better than re-typing method at every class or copy-paste the method.

l) Class Diagram 3: The final version, after applying the design pattern(s) and any other modifications



PART 3 *please, zoom in this part*

Development phase (Implementation details)

1) Create and document a Front-End Design for all Functions (HTML, Bootstrap)

Register Form

•The Register Form [**frontend for register function**]:
this form designed for the users to register an account on the system.
The form has validations using Java script.

Back-End

Back-End

Front-End

Login Page

•The Login page [**frontend for register function**]:

this page is designed for the users, admins to login into the system .

The page has validations using Java script.

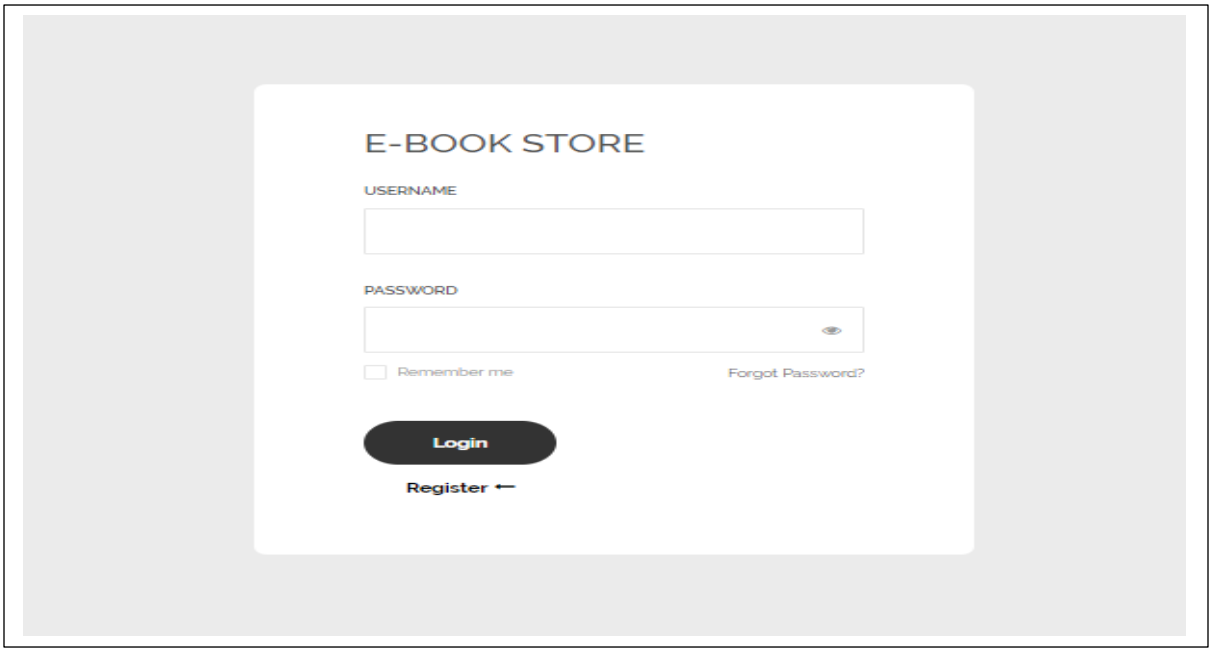
Back-End

```
<div class="limiter">
  <div class="container-login100">
<div class="wrap-login100 p-l-85 p-r-85 p-t-55 p-b-55">
  <form class="login100-form validate-form flex-sb flex-w" method="POST" action="">
        ../../Services/loginService.php-->
          <span class="login100-form-title p-b-32">
                E-Book Store
          </span>
          <span class="txt1 p-b-11">
                Username
          </span>
          <div class="wrap-input100 validate-input m-b-36" data-validate = "Username is required">
                <input class="input100" type="text" name="username" >
                <span class="focus-input100"></span>
          </div>
          <span class="txt1 p-b-11">
                Password
          </span>
          <div class="wrap-input100 validate-input m-b-12" data-validate = "Password is required">
                <span class="btn-show-pass">
                      <i class="fa fa-eye"></i>
                </span>
                <input class="input100" type="password" name="password" >
                <span class="focus-input100"></span>
          </div>
          <div class="flex-sb-m w-full p-b-48">
                <div class="contact100-form-checkbox">
                  <input class="input-checkbox100" id="ckb1" type="checkbox" name="remember-me">
```

Back-End

```
                  <div class="contact100-form-checkbox">
                    <input class="input-checkbox100" id="ckb1" type="checkbox" name="remember-me">
                        <label class="label-checkbox100" for="ckb1">
                              Remember me
                        </label>
                  </div>
                  <div>
                        <a href="#" class="txt3">
                              Forgot Password?
                        </a>
                  </div>
            </div>
            <div class="container-login100-form-btn">
                <input type="submit" name="submit" value="Login"  class="login100-form-btn"/>

            </div>
      </form>
      <div class="Reg-btn" >
          <a class="Register-form-btn"  href="Register.php">
                  Register
                  <i class="fa fa-long-arrow-left m-l-5"></i>
          </a>
      </div>
    </div>
  </div>
</div>

<div id="dropDownSelect1"></div>
```

Front-End



NAVBAR

•The Navbar [**fronted for role management function**]: the header is used in all the pages except the form pages and the buttons inside it will be different depending on the User Type [admin or user].

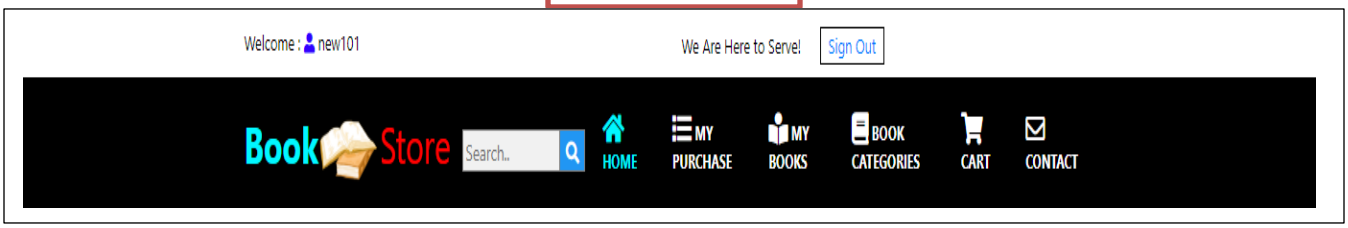
Back-End

```
art navbar -->
<nav class="navbar navbar-expand-lg navbar-light">
        <div class="container">
            <a class="navbar-brand" href="index.php">
            <span>Book</span><img src="img/logo.png" style="height: 80px; margin-bottom: 15px;"><span>Store</span>
            </a>
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent"
            aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"  style="background-color: white;"    ></span>
        </button>
            <form class="search" action="../FronEnd/Category1.php" method="GET" style="margin:auto;max-width:400px">
                <input type="text" placeholder="Search.." name="search">
                <button type="submit"><i class="fa fa-search"></i></button>
                </form>
            <div class="collapse navbar-collapse" id="navbarSupportedContent">
            <ul class="navbar-nav ml-auto">
            <li class="nav-item active">
                <a class="nav-link" href="index.php"><span class="sr-only">(current)</span><i class="fas fa-home"
                style="font-size:25px "></i>Home</a>
            </li>
```

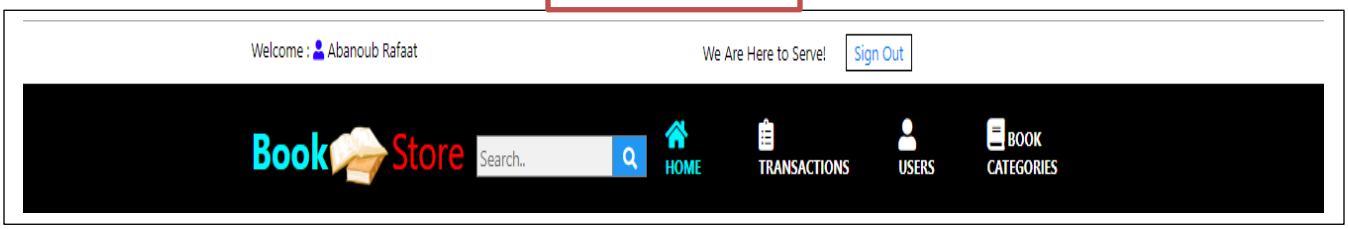
Back-End

```
<?php
  require_once '../PagesController/StateConfigurations.php';
  $ob=new StateConfigurations($_SESSION['usType']);
  $ob->showNavBar();
  echo '<li class="nav-item">
  <a class="nav-link" href="Supp.php"  style="color: white;"><i class="fas fa-book" style="font-size:25px" ></i> book Categories</a>
</li>';
if($_SESSION['usType']!=1)
  {
  echo ' <li class="nav-item">
  <a class="nav-link" href="Cart.php" style="color: white;" ><i class="fas fa-shopping-cart" style="font-size:25px "></i>CART</a>
</li>
  <li class="nav-item">
  <a class="nav-link" href="Contact.php" style="color: white;" ><i class="far fa-envelope" style="font-size:25px"></i>Contact</a>
</li>';
  }
```

Front-End AS **USER**



Front-End AS **Admin**



Footer

•The Footer is the bottom used in all the pages except the form pages which contain our social media accounts to be user friendly and keep a touch between system and users

Back-End

```
<!--Footer-->
<div id="footer">
    <div class="text-center" style="padding-top: 20px;">
        <ul class="list-unstyled list-inline" style=" text-transform: uppercase; letter-spacing: 15px;">
            <i onclick="window.open('https://www.facebook.com/xXlBebolXx','_blank')"
               class="fab fa-facebook fa-2x " aria-hidden="true" style ="color:#3b5999;"></i>
            <i onclick="window.open('https://www.instagram.com/illll_bebo_illll/','_blank')"
               class="fab fa-instagram fa-2x"aria-hidden="true" style ="color:#cd486b"></i>
            <i onclick="window.open('https://web.whatsapp.com/','_blank')"
               class="fab fa-whatsapp fa-2x"aria-hidden="true" style ="color:greenyellow"></i>
            <i onclick="window.open('https://www.youtube.com/channel/UC5ykJ9vGoQIfsV8DK-3jnBg/featured','_blank')"
               class="fab fa-youtube fa-2x"aria-hidden="true" style ="color:#cd201f" ></i>
        </ul>
        </div>
    </div>
    <div class="footer-copyright py-3 text-center">
        <small>© 2020 Copyright:
            <a href="https://www.facebook.com/xXlBebolXx">Abanoub Rafaat</a></small>            </div>
</div>
<!--End of Footer-->
```

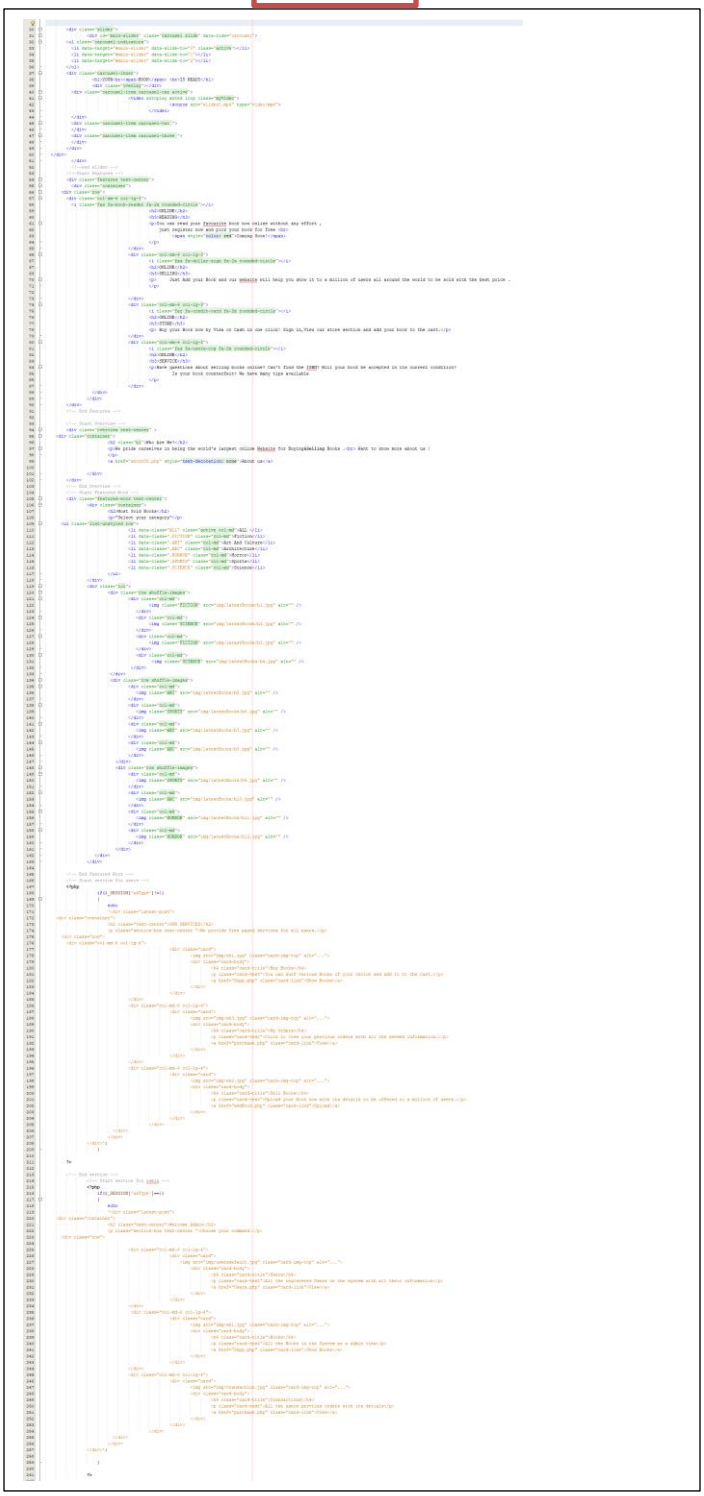
Front-End

© 2020 Copyright: Abanoub Rafaat

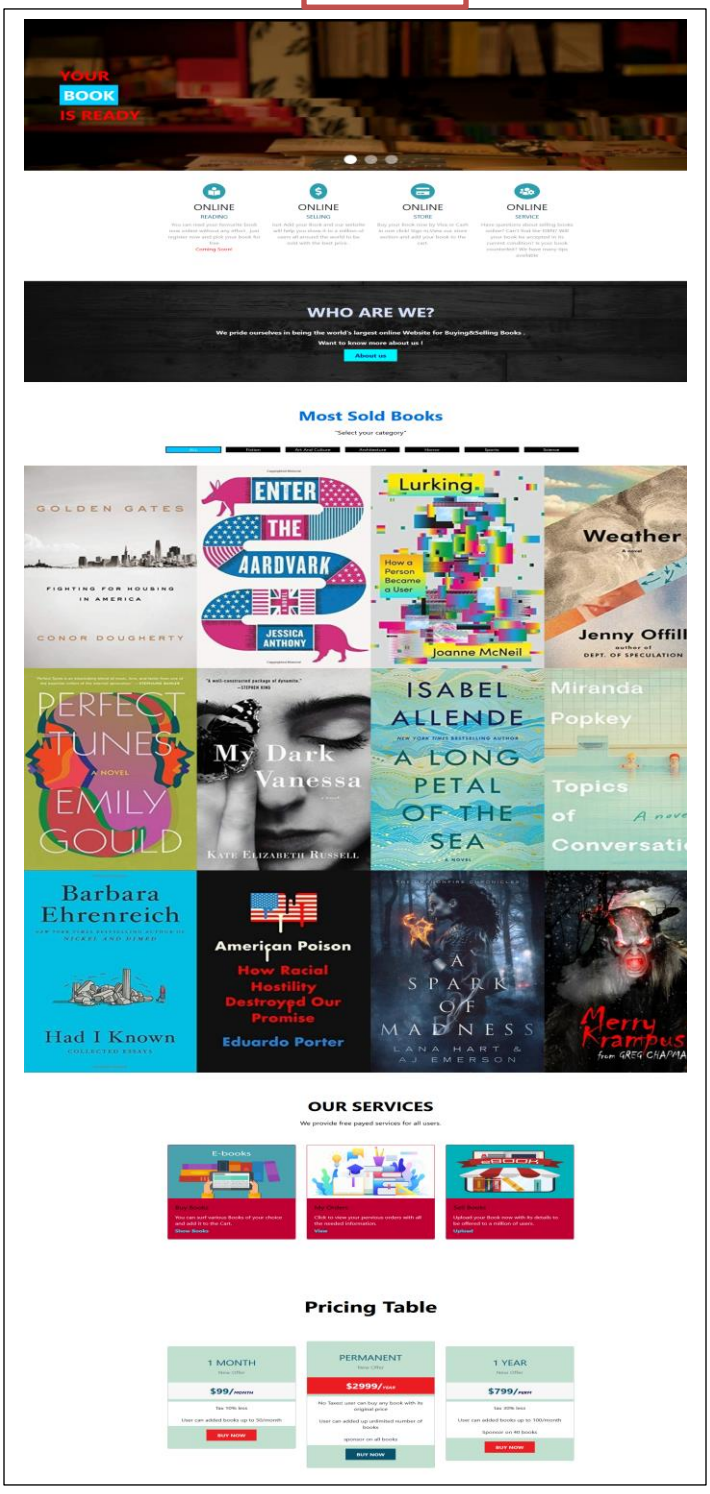
Home Page

•The Home page is the first page the user/admin will see when they login in it have information about our site slider video, our services, who are we the home page can direct you to all this thing it's just a container. I removed header, footer from photo.

Back-End



Front End



Category Page

• Category Page [**fronted for get category function**]: this page shows all the categories in the system which are added in database with its details (category name, category image) admins and users can view this page a button “Add book” appears is user was logged in. show all books button

Back-End

```
    <!--Contet-->
<div class="Supp">
    <h1 style="bottom: 150px"><strong>Categories</strong></h1>
<div class="Supp" style="bottom: 150px">
    <?php
        if($_SESSION['usType']!=1)
        {
   echo '<div class="addbook"> <a href="addBook.php">Add <i class="fas fa-book fa-1x"'
            . ' style=" color: black;"></i> Book</a> </div>';
        }
    ?>
    <br>
    <br>
    <br>
    <br>
    <div class="allbooks"> <a href="Category1.php?cat=0&catname=All Books">Show All</a> </div>
    <ul style="list-style-type: none;text-align: center;">

        <li>
            <?php
                include_once '../PagesController/CategoryController.php';
                $t=new CategoryController();
                $t->ShowCategory();
            ?>
        </li>
    </ul>
</div>
</div>
```

Back-End

```
Function ShowCategory()
{
    $t=new CategoryController();
    $arr=$t->getCategorey();
    for($i=0;$i<count($arr);$i++)
    {
        $img=$i+1;
        echo '<a href="Category1.php?cat='.$arr[$i]->getId().'&catname='.$arr[$i]->getName().'" > '
        . '<img title="Gainers" src="img/C'.$img.'.png" style="width:200px;'
                . 'height:250px" class="zoom"></a>';

    }
}
```

Front-End AS **Admin**

Front-End AS **User**

Add Book Form

- The Add, upload Book Form [**fronted for add book function**]: this is used by users to add books contain validations using java script.

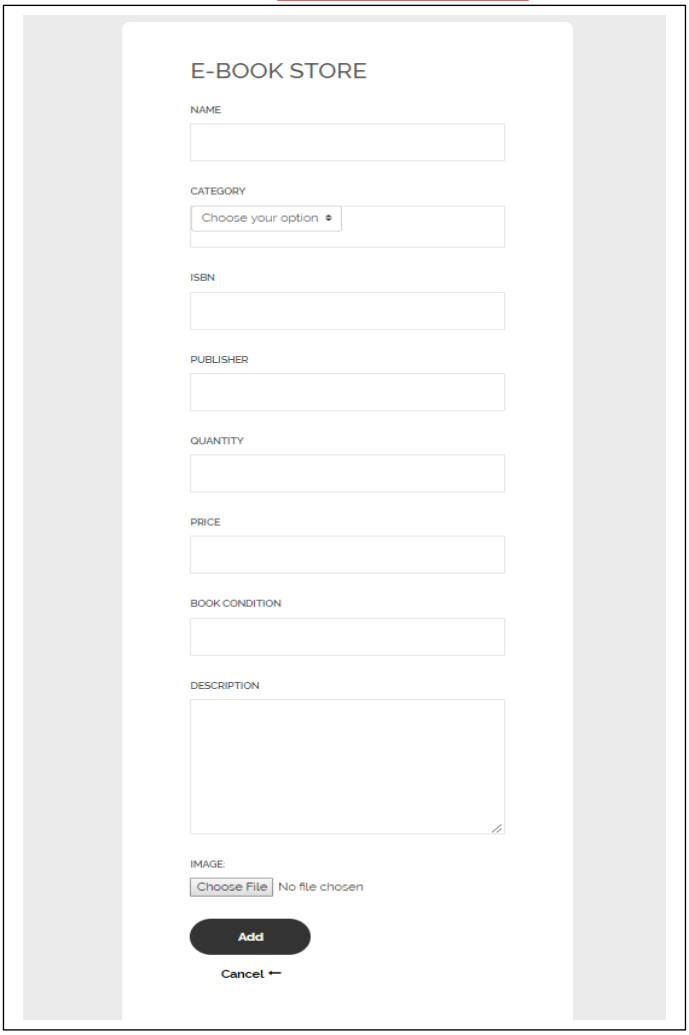
Back-End

```
<div class="limiter">
    <div class="container-login100">
<div class="wrap-login100 p-l-85 p-r-85 p-t-55 p-b-55">
    <form class="login100-form validate-form flex-sb flex-w" method="Get" action="">
            <span class="login100-form-title p-b-32">
                E-Book Store
            </span>
            <span class="txt1 p-b-11">
                Name
            </span>
            <div class="wrap-input100 validate-input m-b-36" data-validate = "Name is required">
                <input type="text" id="textInput" class="input100" name="name" >
                    <span class="focus-input100"></span>
            </div>
            <span class="txt1 p-b-11">
                Category
            </span>
<div class="wrap-input100 m-b-36" >
    <select class="browser-default custom-select mb-4" id="select" name="cat">
        <option  selected="" value="Empty">Choose your option</option>
                <option value="op1">Option 1</option>
                <option value="op2">Option 2</option>
                <option value="op3">Option 3</option>-->
                <?php
                for($i=0;$i<count($catArr);$i++)
                {
                    echo '<option value="'.$catArr[$i]->getId().'">'.$catArr[$i]->getName().'</option>';
                }
                ?>
```

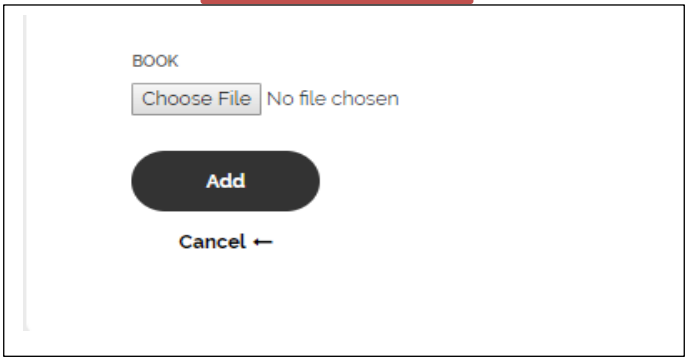
```
    </div>
            <span class="txt1 p-b-11" >
                ISBN
            </span>
            <div class="wrap-input100 validate-input m-b-36" data-validate = "ISBN is required">
                <input type="text" id="textInput" class="input100" name="isbn" >
                    <span class="focus-input100"></span>
            </div>
             <span class="txt1 p-b-11">
                Publisher
            </span>
            <div class="wrap-input100 validate-input m-b-36" data-validate = "Price is required">
                <input type="text" id="textInput" class="input100" name="publisher">
                    <span class="focus-input100"></span>
             </div>
            <span class="txt1 p-b-11" >
                QUANTITY
            </span>
            <div class="wrap-input100 validate-input m-b-36" data-validate = "Quantity is required">
                <input type="text" id="textInput" class="input100" name="quantity" >
                    <span class="focus-input100"></span>
            </div>
            <span class="txt1 p-b-11">
                Price
            </span>
            <div class="wrap-input100 validate-input m-b-36" data-validate = "Price is required">
                <input type="text" id="textInput" class="input100" name="price">
                    <span class="focus-input100"></span>
```

```
                    <span class="focus-input100"></span>
             </div>
             <span class="txt1 p-b-11">
                Description
            </span>
            <div class="wrap-input100  m-b-36">
                <textarea type="text" id="textarea" class="input100" name="desc" style="height: 200px" ></textarea>
                    <span class="focus-input100"></span>
             </div>
                    <div class="custom-file" style="margin-bottom: 50px;">
                        <label for="myfile" class="txt1 p-b-11">IMAGE:</label>
                        <input type="file"  id="myfile" name="myfile"  >
                    </div>
                <div class="container-login100-form-btn">
                    <input class="login100-form-btn" value="Add" name="submit" type="submit">
                </div>
        </form>
        <div class="Reg-btn" >
            <a class="Register-form-btn" href="Supp.php">
                Cancel
                <i class="fa fa-long-arrow-left m-l-5"></i>

            </a>
        </div>
    </div>
</div>
 </div>

<div id="dropDownSelect1"></div>
```

Front-End **ADD BOOK**



Front-End **UPLOAD BOOK**



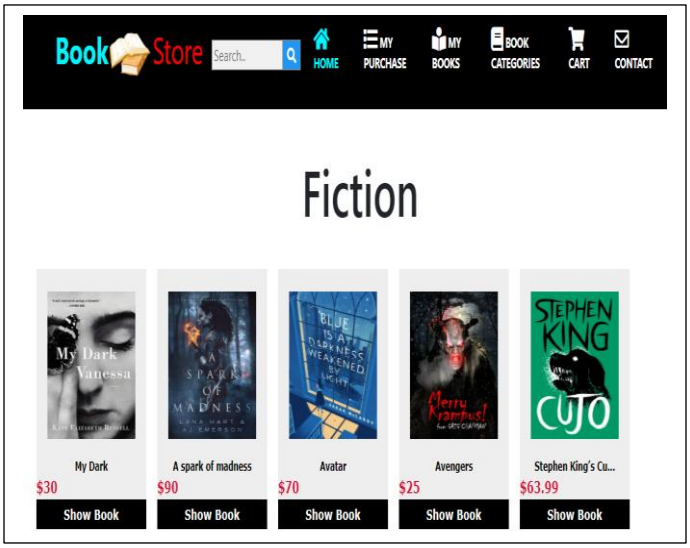
Books Page

- Show Book Page [**fronted for show book and show book by search bar function**]: this page used to show the books on the system either if the user was selecting a specific category or using search bar admins and users can view this page.

Back-End Show Books

```
<h1 style="text-align: center;font-weight: 3000;font-size: 70px;margin-bottom: 20px;">
    <?php
    if(isset($_GET['cat']))
        {
            echo $_GET['catname'];
        }elseif ($_GET['search']) {

            echo 'Search ('.$_GET['search'].')';
        }
?>
</h1>
<br>
<div class="products" style="margin-bottom: 70px; font-family: verdana;">
<div class="container">
    <div class="row" >
        <ul>
            <?php
                $bookCat=new PageCategorylController();

                if(isset($_GET['cat']))
                {
                    $bookCat->ShowBooks();
                }elseif ($_GET['search']) {
                    $bookCat->ShowBooksSearch($_GET['search']);
                }
            ?>
</ul>
</div>
</div>
</div>
```

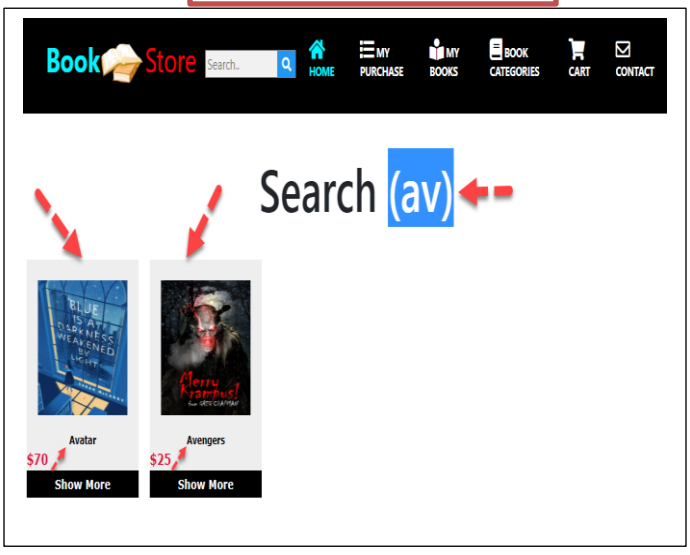
Front-End Show Books




```
function ShowBooks()
{
    $book2=new BookService();
    $arr=$book2->getBookByCat($_GET['cat']);
    for($i=0;$i<count($arr);$i++)
    {
        if(strlen($arr[$i]->getName())>19)
        {
            $name=substr($arr[$i]->getName(), 0,19)."...";
        } else {
            $name =$arr[$i]->getName();
        }

        echo '<li>
                    <div class="Product">
                    <a style="width=200px;height=200px;" href="Bookinfo.php?cat='.$_GET['cat'].'&catname='.
                $_GET['catname'].'&book='.$arr[$i]->getId().'" class="img"><img src="img/'.$arr[$i]->getImage().'"/></a>
                    <a href="Bookinfo.php?cat='.$_GET['cat'].'&catname='.$_GET['catname'].
                '&book='.$arr[$i]->getId().'" class="name" >'.$name.'</a>
                    <div>   $'.$arr[$i]->getPrice().'</div>
                        <a href="Bookinfo.php?cat='.$_GET['cat'].'&catname='.$_GET['catname'].
                '&book='.$arr[$i]->getId()."class="cart">Show Book</a>
                    </div>
                    </li>';
    }
}
```

Front-End Show Books by Search bar

```
function ShowBooksSearch($search)
{
    $book2=new BookService();
    $arr=$book2->getBookBySearchBar($search);
    for($i=0;$i<count($arr);$i++)
    {
        if(strlen($arr[$i]->getName())>25)
        {
            $name=substr($arr[$i]->getName(), 0,25)."...";
        } else {
            $name =$arr[$i]->getName();
        }
        echo '<li>
                    <div class="Product ">
                    <a href="Bookinfo.php?search='.$search.'&book='.$arr[$i]->getId().'"
                . ' class="img"><img src="img/'.$arr[$i]->getImage().'"/></a>
                    <a href="Bookinfo.php?search='.$search.'&book='.$arr[$i]->getId().'"
                . ' class="name" >'.$name.'</a>
                    <div>   $'.$arr[$i]->getPrice().'</div>
                        <a href="Bookinfo.php?search='.$search.'&book='.$arr[$i]->getId().
                '"class="cart">Show More</a>
                    </div>
                    </li>';
    }
}
```

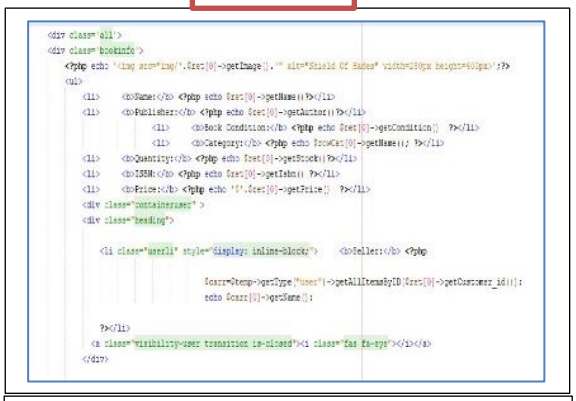


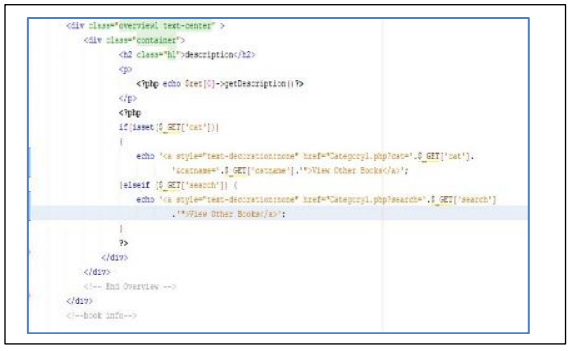
Book information Page

• When a user clicks a specific book in book pages, he is directed to this page which have 2 main responsibilities

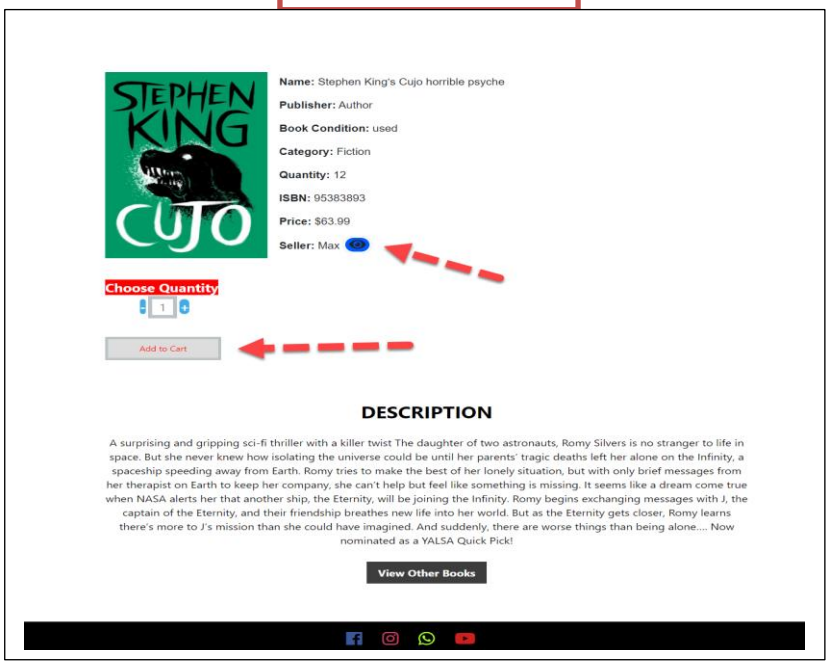
first: it shows the details of the book to the user which include (book name, publisher, book condition, category, Quantity, ISBN, price, book seller)
second: contain a button with eye icon to show the information of the seller and ability to deal his number if he has built in software which allows
third: contain a text field with scroll pane to enter the required quantity of the books and a button add to cart so if the user want to buy this book he just choose the quantity and it to his cart where he can order it upon need

Back-End

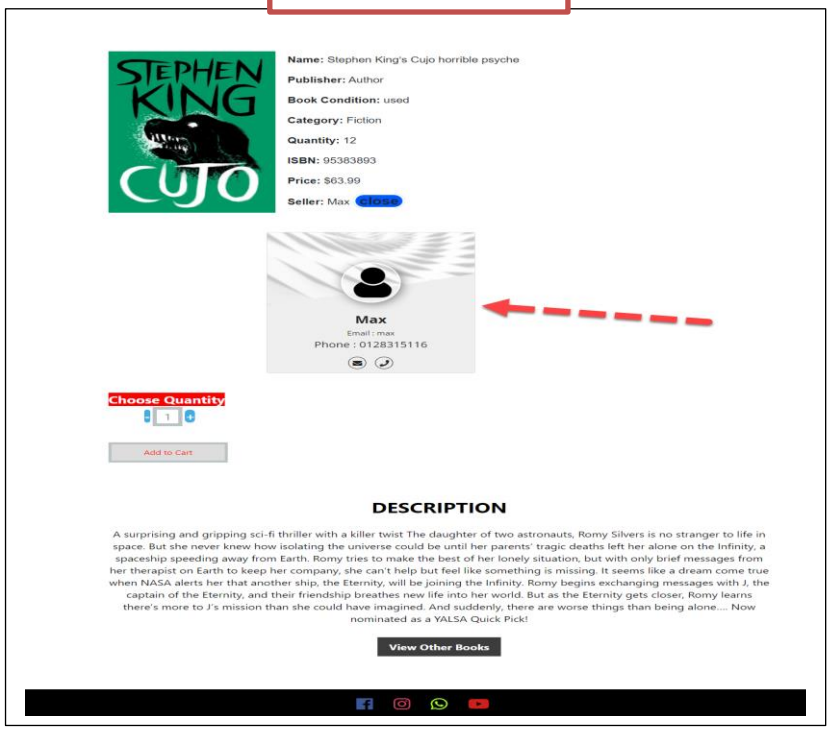




Front-End Before view seller



Front-End After view seller



Cart Page

This page is used by users only, they put the books they want to buy in the cart and then click order now to create order and buy the books

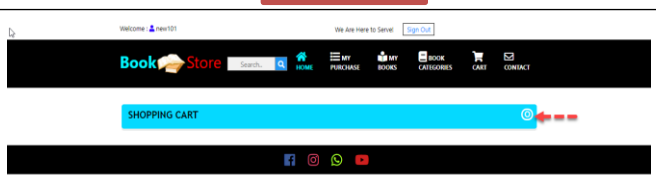
NOTE: user can delete any item from the cart. Page shows the information of items price of each row ,total prince and the tax

Back-End

```
<div class="containercart">
        <div class="heading">
          <h1>
              <span class="shopper"></span> Shopping Cart
          </h1>
    <a href="#" class="visibility-cart transition is-open">X</a>
        </div>
        <div class="cart transition is-open">
                        <button class="btn btn-update" onclick="document.
          <div class="table">
              <div class="layout-inline row th">
                <div class="col col-pro">Product</div>
                <div class="col col-price align-center ">
                      Seller
                </div>
                <div class="col col-qty align-center">Quantity</div>
                <div class="col">TAX</div>
                <div class="col">Price</div>
              </div>
            <?php
              $cc->cartOrders();
            ?>
        </div>
            <a href="order.php" class="btn btn-update">Order Now</a>
  </div>
</div>
```

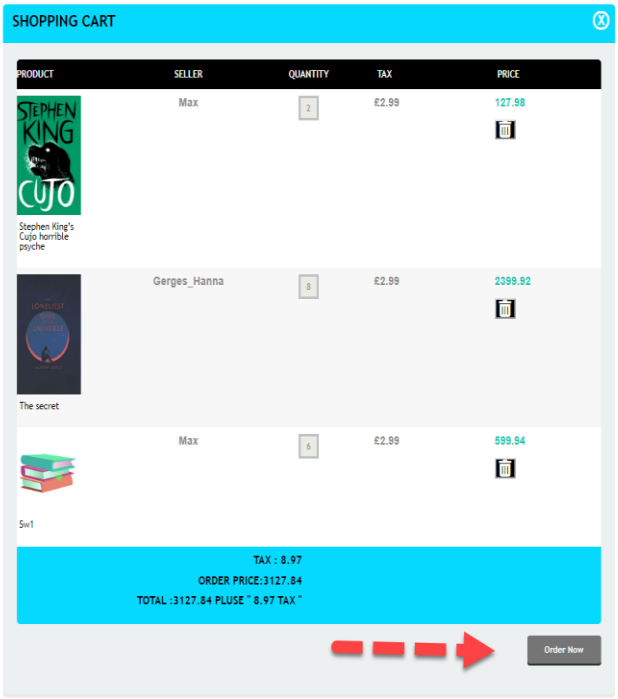
```
function cartOrders()
{
    $factorydp=new FundamentalFactory();
        $arr=$factorydp->getType("cart")->getAllItemsByID($_SESSION['usId']);
        $total=NULL;
        for($i=0;$i<count($arr);$i++)
        {
            $id=$arr[$i]->getBook_id();
            $barr=$factorydp->getType("book")->getAllItemsByID($id);
            $carr=$factorydp->getType("user")->getAllItemsByID($arr[$i]->getSalesman_id());
            if($i%2==0)
            {
                echo '<div class="layout-inline row">

          <div class="col col-pro layout-inline">
        <a href=""><img src="img/'.$barr[0]->getImage().'" alt="'.$barr[0]->getName().'" /></a>
            <p style="color:black">'.$barr[0]->getName().'</p>
          </div>

          <div class="col col-price col-numeric align-center ">
              <p>'.$carr[0]->getName().'</p>
          </div>

          <div class="col col-qty layout-inline">

              <input type="numeric" disabled="" value="'.$arr[$i]->getQuantity().'" />

          </div>

          <div class="col col-vat col-numeric">
              <p>£'.$this->tax.'</p>
          </div>
          <div class="col col-total col-numeric">
            <p>'.$arr[$i]->getQuantity()*$barr[0]->getPrice().'</p>
            <button onclick="document.location = \'Cart.php?bookiddel='.$arr[$i]->getBook_id().
                    '&seller='.$arr[$i]->getSalesman_id().'&customer='.$arr[$i]->getCustomer_id()
                    .'&quantityb='.$arr[$i]->getQuantity().'\'"
                        style="background-color: black;color: rgb(255, 255, 255); font-size: 25px;
                        " class="fas fa-trash-alt"></button>
          </div>

        </div>';
            } else {
              echo '<div class="layout-inline row row-bg2">

          <div class="col col-pro layout-inline">
              <a href=""><img src="img/'.$barr[0]->getImage().'" alt="'.$barr[0]->getName().'" /></a>
              <p style="color:black">'.$barr[0]->getName().'</p>
          </div>

          <div class="col col-price col-numeric align-center ">
              <p>'.$carr[0]->getName().'</p>
          </div>

          <div class="col col-qty layout-inline">

              <input type="numeric" disabled="" value="'.$arr[$i]->getQuantity().'" />

          </div>

          <div class="col col-vat col-numeric">
              <p>£'.$this->tax.'</p>
          </div>
          <div class="col col-total col-numeric">
              <p>'.$arr[$i]->getQuantity()*$barr[0]->getPrice().'</p>
              <button style="background-color: black;color: rgb(255, 255, 255);
              font-size: 25px;" class="fas fa-trash-alt" onclick="document.location =
              \'Cart.php?bookiddel='.$arr[$i]->getBook_id().'&seller='.
                      $arr[$i]->getSalesman_id().'&customer='.$arr[$i]->getCustomer_id().
                      '&quantityb='.$arr[$i]->getQuantity().'\'"></button>
          </div>

        </div>';
            }
            $total=$total+($arr[$i]->getQuantity()*$barr[0]->getPrice());
        }

        echo '<div class="tf">
            <div class="row layout-inline">
                <div class="col">
                <p>TAX : '.$this->tax*count($arr) .'</p>
                <p>Order Price:'.$total.' </p>
                  <p>Total :'.$total.' pluse " '.$this->tax*count($arr).' tax "</p>
                </div>
                <div class="col"></div>
            </div>
        </div> ';

r code here
```

Front-End



The O button show the cart using JS

The X button hide the cart using JS



Order Form Page

this page includes a form to complete order processes, form contain the required information that must be found to keep sure that the product will be sent correctly and securely.
the information contains (country, city, address, postal code, phone number, payment method)
the payment method is a check box to make user select between cash or visa
if the user selects visa check box a drop-down list which contain more details about visa will appear
validation and drop-down list made by java script

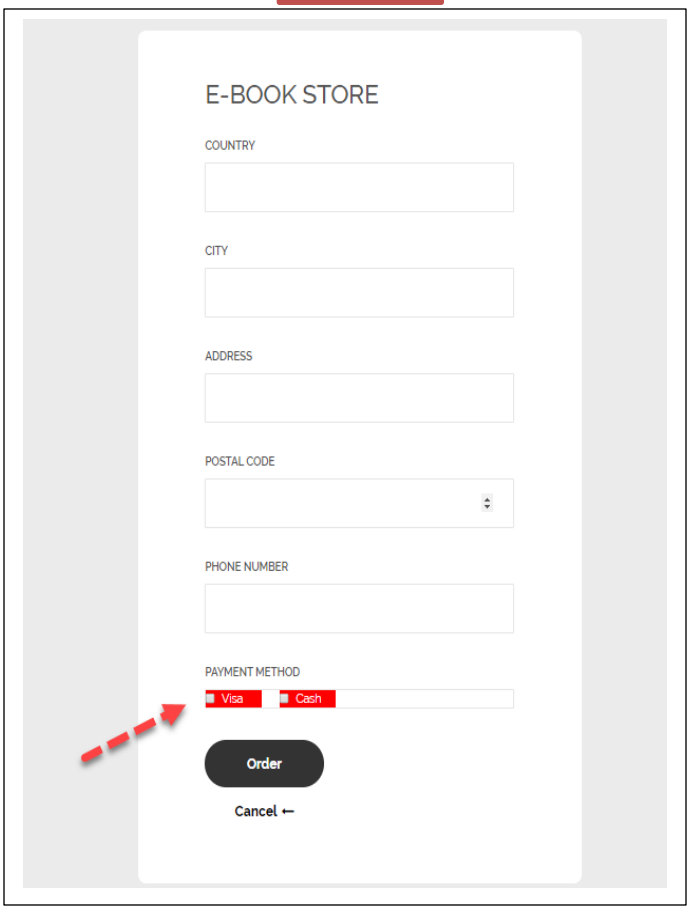
Back-End

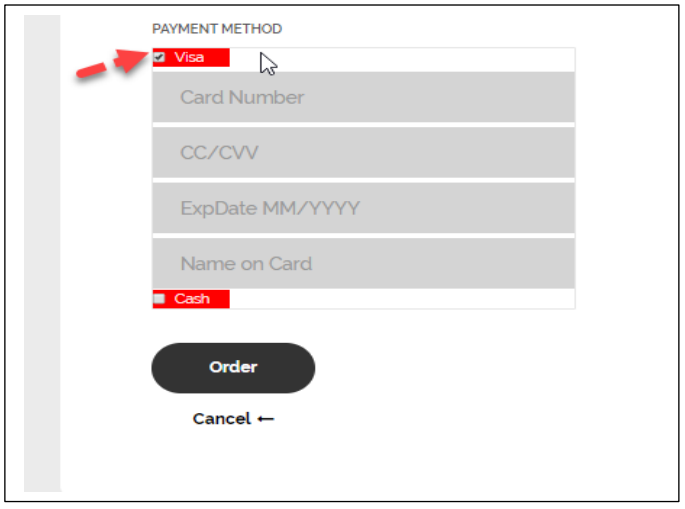
```
<div class="limiter">
        <div class="container-login100">
                <div class="wrap-login100 p-l-85 p-r-85 p-t-55 p-b-55">
                    <form class="login100-form validate-form flex-sb flex-w" method="POST" action="">
                                <span class="login100-form-title p-b-32">
                        E-Book Store
                    </span>
                    <span class="txt1 p-b-11" >
                            Country
                    </span>
                    <div class="wrap-input100 validate-input m-b-36" data-validate = "Country is required">
                        <input type="text" id="textInput" class="input100" name="country" >
                            <span class="focus-input100"></span>
                    </div>
                      <span class="txt1 p-b-11">
                            City
                    </span>
                    <div class="wrap-input100 validate-input m-b-36" data-validate = "City is required">
                        <input type="text" id="textInput" class="input100" name="city">
                            <span class="focus-input100"></span>
                     </div>
                     <span class="txt1 p-b-11">
                            Address
                    </span>
                    <div class="wrap-input100 validate-input m-b-36" data-validate = "Address is required">
                        <input type="text" id="textInput" class="input100" name="address" >
                            <span class="focus-input100"></span>
```

```
        <input type="text" id="textInput" class="input100" name="address" >
            <span class="focus-input100"></span>
    </div>
    <span class="txt1 p-b-11" >
            Postal code
    </span>
    <div class="wrap-input100 validate-input m-b-36" data-validate = "Posatal Code is required">
        <input type="number" id="number" class="input100" name="postal" >
            <span class="focus-input100"></span>
    </div>
    <span class="txt1 p-b-11">
            Phone Number
    </span>
    <div class="wrap-input100 validate-input m-b-36" data-validate = "Phone is required">
        <input type="number" id="number" class="input100" name="phone">
            <span class="focus-input100"></span>
     </div>
    <span class="txt1 p-b-11">
            Payment Method
    </span>
    <div class="wrap-input100 validate-input m-b-36" data-validate ="Method is required">

        <label class="checkbox-inline"
               style="background-color: red;color: white;width: 70px;
               margin-right: 20px;"><input type="checkbox" id="myCheck" name="type"
                value="Visa"  onclick="myFunction1()">Visa
            </label>
```

```
    <input id="text1" type="number" class="input100" name="numCard" placeholder="Card Number"
         style="display:none;background-color: rgb(212, 212, 212);margin-top: 5px;">
        <input id="text2" type="number" class="input100" name="cc" placeholder="CC/CVV"
               style="display:none;background-color: rgb(212, 212, 212);margin-top: 5px;">
        <input id="text3" type="text" class="input100" name="exp" placeholder="ExpDate MM/YYYY"
               style="display:none;background-color: rgb(212, 212, 212);margin-top: 5px;">
        <input id="text4" type="text" class="input100" name="nameCard" placeholder="Name on Card"
               style="display:none;background-color: rgb(212, 212, 212);margin-top: 5px;">

        <label class="checkbox-inline" style="background-color: red;color: white;width: 70px;">
              <input type="checkbox" id="checkBox" onclick="myFunction2()" name="type" value="cash" >Cash
          </label>
                    <span class="focus-input100"></span>
                </div>

                <div class="container-login100-form-btn">
                    <input class="login100-form-btn" value="Order" name="submit" type="submit">
                </div>
        </form>
        <div class="Reg-btn" >
            <a class="Register-form-btn" href="index.php">
                    Cancel
                    <i class="fa fa-long-arrow-left m-l-5"></i>
                </a>
        </div>
        </div>
    </div>
```

Front-End





Purchase, Transaction Page

This page shows the past purchase, transaction of the logged in user

if the user type was admin the page will show all time transactions with enough details about each order and each item in each order as well as it will also show a button to generate a PDF report with all this transaction.

if the user type was normal user the page will show his all-time purchase history since he registers an account with enough details about each purchase as well as a button to generate a PDF report with all his purchases

Back-End

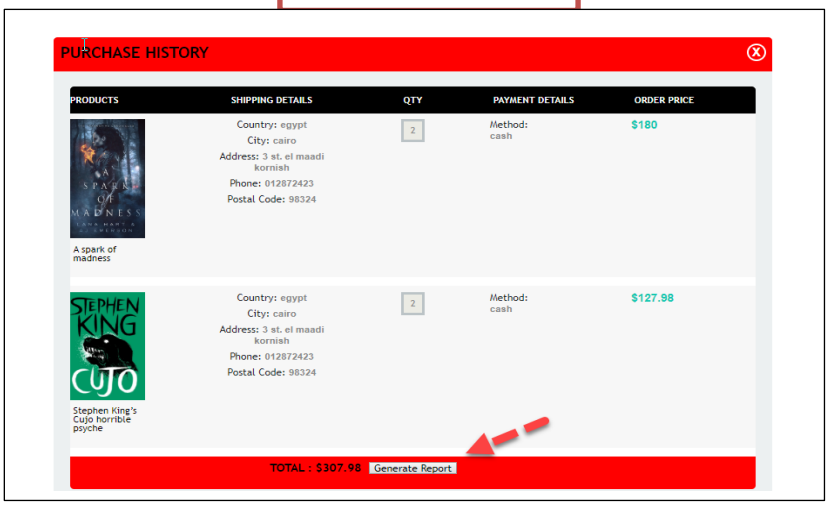
```
<div class="containercart">
        <div class="heading">
          <h1>
              <span class="shopper"></span>Purchase History
          </h1>

          <a href="#" class="visibility-cart transition is-open">X</a>
        </div>

        <div class="cart transition is-open">

          <button class="btn btn-update">Clear History</button>-->


          <div class="table">

              <div class="layout-inline row th">
                <div class="col col-pro">Products</div>
                <div class="col col-price align-center ">
                    Shipping Details
                </div>
                <div class="col col-qty align-center">QTY</div>
                <div class="col">Payment Details</div>
                <div class="col">Order Price</div>
              </div>
            <!--          from                    -->
```

```
    <!--          from                    -->
    <?php require_once '../PagesController/PurchaseController.php'; ?>
    <!--                to here-->
<div class="tf">
              <div class="row layout-inline">
                <div class="col">
                    <p>Total : $<?php echo $count; ?></p>

                </div>
                <button onclick="window.open('PDF.php')" target="_blank"
                  style="font-size: 16px; height:20px; margin-top: 10px; ">Generate Report</button>
                <div class="col"></div>
              </div>
            </div>


          </div>

      </div>
    </div>
```

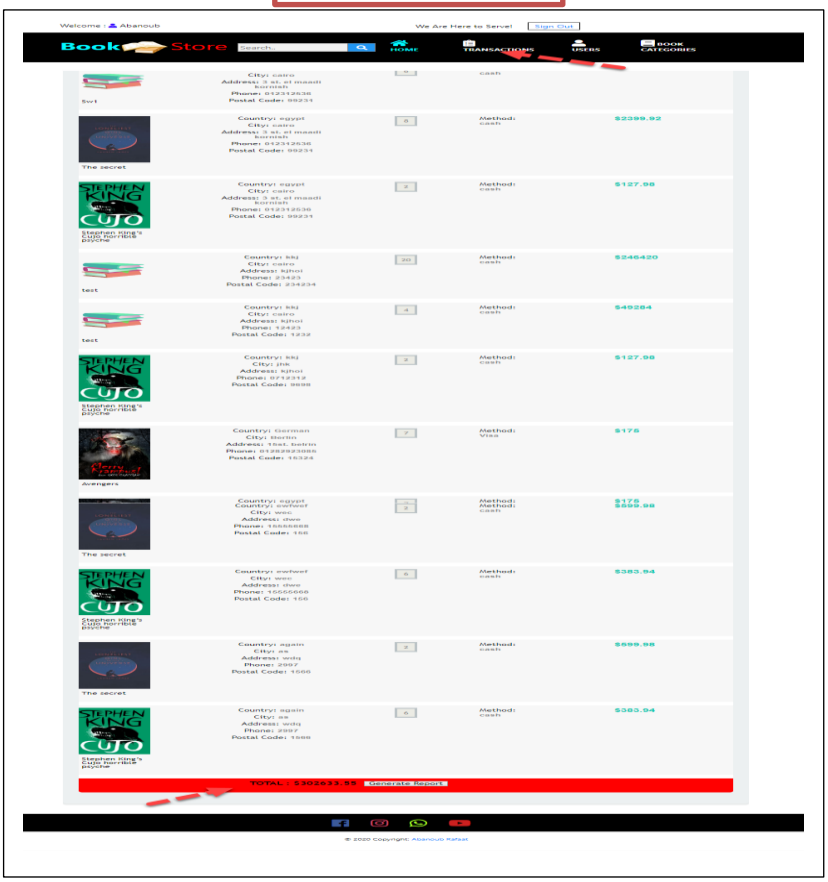
```
$temp=new FundamentalFactory();
$or=new OrderService();
$book=new BookService();
$address=new AddressService();
$cs=new CustomerService();
if($_SESSION['usType']==0)
{
    $OrArr=$temp->getType("order")->getAllItemsByID($_SESSION['usId']);
}  else if($_SESSION['usType']==1) {
    $OrArr=$temp->getType("order")->getAllItems();
}

$count=0;
for($i=0;$i<count($OrArr);$i++)
{
            $count=$count+$OrArr[$i]->getTotal();
            $boArr=$temp->getType("book")->getAllItemsByID($OrArr[$i]->getBook_id());
            $adArr=$address->getAddressById($OrArr[$i]->getAddressID());
            $csArr=$temp->getType("user")->getAllItemsByID($boArr[0]->getCustomer_id());
            $csArr2=$temp->getType("user")->getAllItemsByID($OrArr[0]->getCustomerID());

            echo
            '<div class="layout-inline row row-bg2">

            <div class="col col-pro layout-inline">
                <a ><img src="img/'.$boArr[0]->getImage().'" alt="'.$boArr[0]->getName().'" /></a>
                <p style="color:black">'.$boArr[0]->getName().'</p>
            ';
```

```
<div class="col col-shipping col-numeric align-center">
        <div class="col-md">
            <h6>Country: <p style="display: inline;font-size: 15px;">'.$adArr[0]->getCountry().'</p></h6>
            <h6>City: <p style="display: inline;font-size: 15px;">'.$adArr[0]->getCity().'</p></h6>
            <h6>Address: <p style="display: inline;font-size: 15px;">'.$adArr[0]->getAddress().'</p></h6>
            <h6>Phone: <p style="display: inline;font-size: 15px;">'.$adArr[0]->getPhone().'</p></h6>
            <h6>Postal Code: <p style="display: inline;font-size: 15px;">'.$adArr[0]->getPostal().'</p></h6>
        </div>
</div>

<div class="col col-qty layout-inline">
    <input type="numeric" disabled="" value="'.$OrArr[$i]->getQuantity().'" />
</div>

<div class="col col-val col-numeric">
    <h6>Method:<br><p style="display: inline;font-size: 15px;">'.$OrArr[$i]->getPayment_method().'</p></h6>
</div>
<div class="col col-total col-numeric">
    <p> $'.$OrArr[$i]->getTotal().'</p>

</div>

</div>

';
```

Front-End as user



Front-End as admin



PDF GENERATED

This is a front-end design of the generated PDF as a result of clicking generate report button in purchase, transaction to be well-organized and having a good style

Back-End

```php
class TestPDF extends FPDF{
    function header() {
        $this->Image('logo.png',80,3);
        $this->SetFont('Arial', 'B', 30);
        $this->Cell(276,18,'Book Store',0,0,'C');
        $this->Ln();
    }
    function footer()
    {
        $this->SetY(-15);
        $this->SetFont('Arial', '', 8);
        $this->Cell(0,10,'Page '.$this->PageNo().'/{nb}',0,0,'C');
    }
    function headerTsble(){
        $this->SetFont('Times','B',18);
        $this->Ln();
        $this->Cell(40,10,'ID',1,0,'C');
        $this->Cell(80,10,'Book Name',1,0,'C');
        $this->Cell(30,10,'Quantity',1,0,'C');
        $this->Cell(45,10,'Payment Type',1,0,'C');
        $this->Cell(40,10,'Seller',1,0,'C');
        $this->Cell(50,10,'Total',1,0,'C');
        $this->Ln();

    }
    function viewTable($id,$name,$qty,$type,$seller,$total)
    {
            $this->SetFont('Times','',10);
            $this->Cell(40,10,$id,1,0,'L');
            $this->Cell(80,10,$name,1,0,'L');
            $this->Cell(30,10,$qty,1,0,'L');
            $this->Cell(45,10,$type,1,0,'L');
            $this->Cell(40,10,$seller,1,0,'L');
            $this->Cell(50,10,$total,1,0,'L');
            $this->Ln();

    }
```

Front-End

PDF.php 1 / 2

Book Store

ID	Book Name	Quantity	Payment Type	Seller	Total
26	A spark of madness	2	cash	new101	180
25	Stephen Kingâ€™s Cujo horrible psyche	2	cash	Max	127.98
24	Sw1	6	cash	Max	599.94
23	The secret	8	cash	Gerges_Hanna	2399.92
22	Stephen Kingâ€™s Cujo horrible psyche	2	cash	Max	127.98
21	test	20	cash	new101	246420
20	test	4	cash	new101	49284
19	Stephen Kingâ€™s Cujo horrible psyche	2	cash	Max	127.98
18	Avengers	7	Visa	Max	175
17	Avengers	7	cash	Max	175
16	Stephen Kingâ€™s Cujo horrible psyche	1	cash	Max	63.99
15	The secret	2	cash	Gerges_Hanna	599.98
14	Stephen Kingâ€™s Cujo horrible psyche	6	cash	Max	383.94

Page 1/2

User page

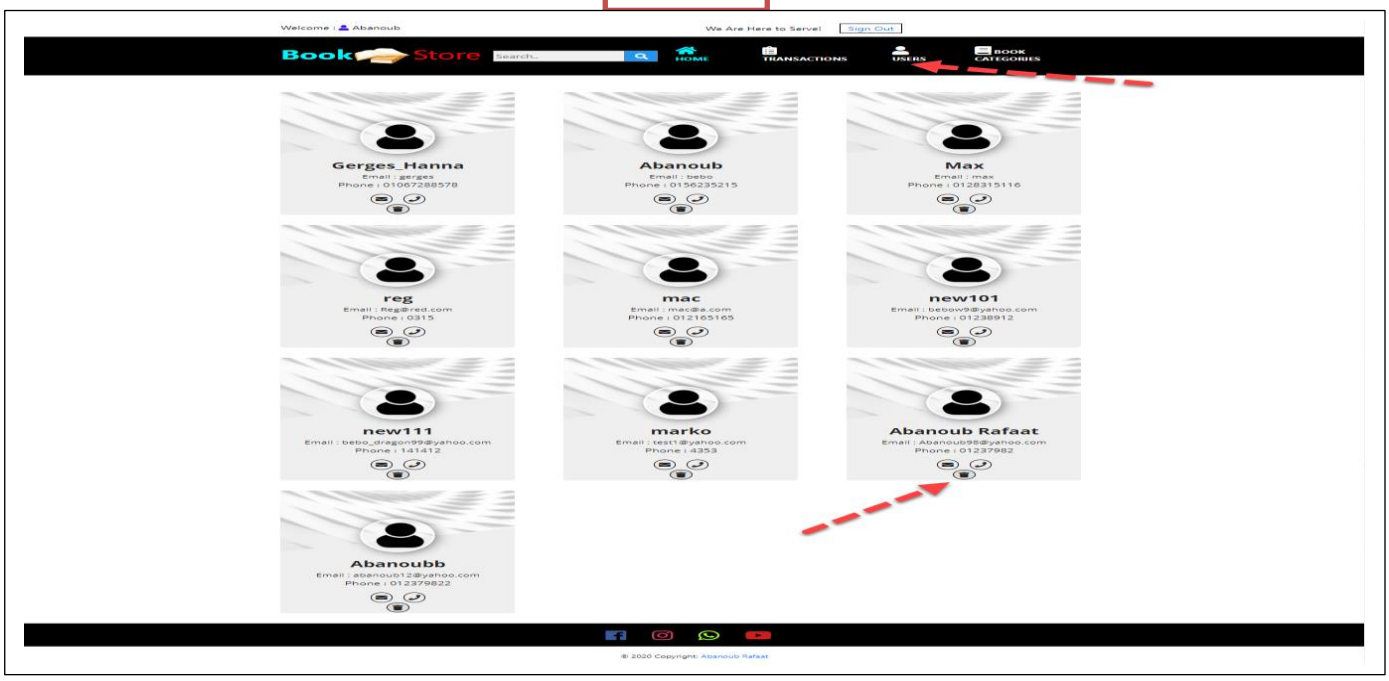
The page is used by admins to manage the registered register admin can view their information and delete any user upon need

Back-End

```
<!--Card --><!--Card -->
<div class="users">
    <div class="container">
    <div class="row">

        <?php
            $us->getUsers();
        ?>

    </div>
</div>
    </div>
    <!--Card --><!--Card -->
```

```
function getUsers()
{
    $temp=new FundamentalFactory();
    $carr=$temp->getType("user")->getAllItems();
    for($i=0;$i<count($carr);$i++)
        {
        echo '<div class="col-md-4">
      <div class="card profile-card-3">
        <div class="background-block">
            <img src="img/userbck.jpg" alt="profile-sample1" class="background"/>
        </div>
        <div class="profile-thumb-block">
            <img src="img/defuserimg.png" alt="profile-image" class="profile"/>
        </div>
        <div class="card-content">
                    <h2>'.$carr[$i]->getName().'<small>Email : '.$carr[$i]->getEmail().'</small></h2
                        <span>Phone : '.$carr[$i]->getPhone().'</span>
                    <div class="icon-block"><a href="#"><i class="fas fa-envelope"></i>
                    </a><a href="tel:'.$carr[$i]->getPhone().'"> <i class="fas fa-phone"></i></a>
                        <form  method="POST">
                        <button class="fas fa-trash" name="trash" value="'
                .$carr[$i]->getID().'"></button>
                    </form>
                    </div>
                    </div>
                </div>
                    </div>';
                    }
```

Front-End



My Books Page

This page includes the all the added or uploaded books for a specific user with their details with ability to see the buyers of this book if you clicked on **view buyer** button

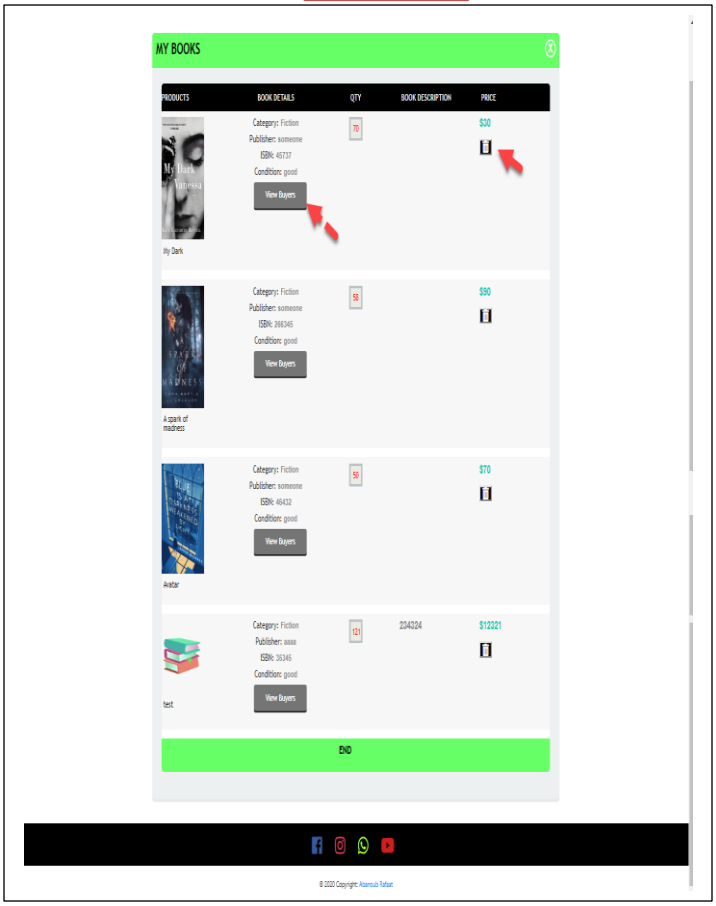
Back-End

```
        <div class="containercart">
        <div class="heading">
          <h1>
                <span class="shopper"></span>My Books
          </h1>
          <a href="#" class="visibility-cart transition is-open">X</a>
        </div>
        <div class="cart transition is-open">
          <div class="table">
                <div class="layout-inline row th">
                  <div class="col col-pro">Products</div>
                  <div class="col col-price align-center ">
                        Book Details
                  </div>
                  <div class="col col-qty align-center">QTY</div>
                  <div class="col">Book Description</div>
                  <div class="col">Price</div>
                </div>
<?php require_once '../PagesController/MyBooks.php';  ?>
        <div class="tf">
                        <div class="row layout-inline">
                         <div class="col">
                             <p>End</p>
                         </div>
                         <div class="col"></div>
                    </div>
                          </div>
                    </div>
```

```
require_once '../Services/OrderService.php';
    require_once '../Services/BookService.php';
    require_once '../Services/CustomerService.php';
    require_once '../Services/AddressService.php';
    require_once '../PagesController/CategoryController.php';

    $book=new BookService();
    $bArr=$book->getBookBySeller($_SESSION['usId']);

    $cat=new CategoryController();
    $rowCat=$cat->getCategoreyByID($bArr[0]->getCategory());
    $count=0;
    for($i=0;$i<count($bArr);$i++)
    {
                echo
            '
        <div class="layout-inline row row-bg2">

        <div class="col col-pro layout-inline">
                <a ><img src="img/'.$bArr[$i]->getImage().'" alt="'.$bArr[$i]->getName().'" /></a>
                <p style="color:black">'.$bArr[$i]->getName().'</p>
            </div>
    <div class="col col-shipping col-numeric align-center">
          <div class="col-md">
                <h6>Category: <p style="display: inline;font-size: 15px;">'.$rowCat[0]->getName(). '</p></h6>
                <h6>Publisher: <p style="display: inline;font-size: 15px;">'.$bArr[$i]->getAuthor().'</p></h6>
                <h6>ISBN: <p style="display: inline;font-size: 15px;">'.$bArr[$i]->getIsbn().'</p></h6>
                <h6>Condition: <p style="display: inline;font-size: 15px;">'.$bArr[$i]->getCondition().'</p></h6>
```

```
                <a ><img src="img/'.$bArr[$i]->getImage().'" alt="'.$bArr[$i]->getName().'" /></a>
                <p style="color:black">'.$bArr[$i]->getName().'</p>
            </div>
    <div class="col col-shipping col-numeric align-center">
          <div class="col-md">
                <h6>Category: <p style="display: inline;font-size: 15px;">'.$rowCat[0]->getName(). '</p></h6>
                <h6>Publisher: <p style="display: inline;font-size: 15px;">'.$bArr[$i]->getAuthor().'</p></h6>
                <h6>ISBN: <p style="display: inline;font-size: 15px;">'.$bArr[$i]->getIsbn().'</p></h6>
                <h6>Condition: <p style="display: inline;font-size: 15px;">'.$bArr[$i]->getCondition().'</p></h6>
              </div>
    </div>
    <div class="col col-qty layout-inline">
            <input type="numeric" disabled="" value="'.$bArr[$i]->getStock().'" />
        </div>
    <div class="col col-desc col-numeric">
            <p>'.$bArr[$i]->getDescription().'</p>
        </div>
    <div class="col col-total col-numeric">
          <p> $'.$bArr[$i]->getPrice().'</p>
          <button style="background-color: black;color: white; font-size: 20px;"
          class="fas fa-trash-alt"></button>
    </div>

    </div>
      ';
}
```

Front-End



Buyers page

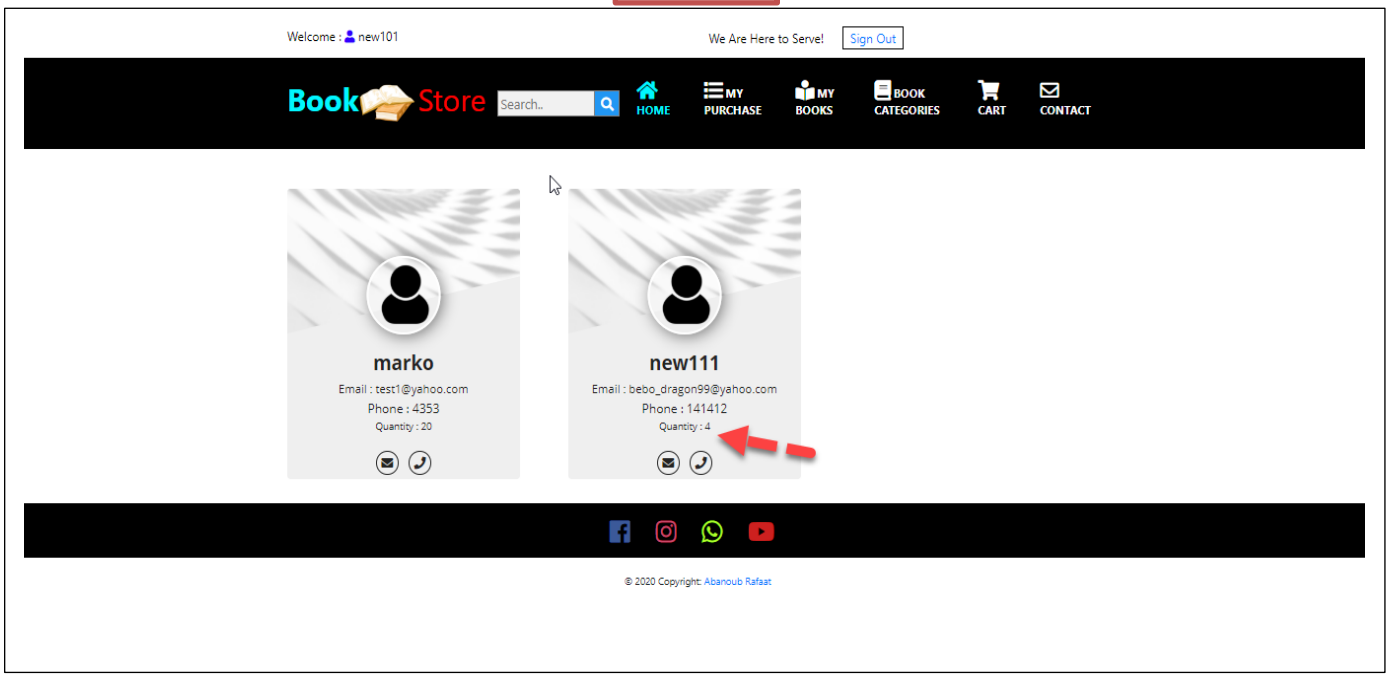
When a user enters **view buyers' button** for specific book in My Book page, he will be directed to this page which shows the users who bought this book with his details name, email, phone and quantity of the book he bought.

NOTE: Front end of this page looks like users page of admin but its not its just to keep the shape of the website organized

Back-End

```
<div class="users">
    <div class="container">
        <div class="row">

<?php
                    $or=new OrderService();
                    $orArr=$or->getOrderbyBookId($_GET['book']);
                    for($i=0;$i<count($orArr);$i++)
                    {
                        $cs=new CustomerService();
                        $csArr=$cs->getAllItemsByID($orArr[$i]->getCustomerID());
                        echo '
        <div class="col-md-4">
            <div class="card profile-card-3">
                <div class="background-block">
                    <img src="img/userbck.jpg" alt="profile-sample1" class="background"/>
                </div>
                <div class="profile-thumb-block">
                    <img src="img/defuserimg.png" alt="profile-image" class="profile"/>
                </div>
                <div class="card-content">

                        <h2>'.$csArr[0]->getName().'<small>Email : '.$csArr[0]->getEmail().' </small></h2>
                            <span>Phone : '.$csArr[0]->getPhone().'</span>
                            <h6><small>Quantity : '.$orArr[$i]->getQuantity().' </small></h6>
                        <div class="icon-block"><a href="#"><i class="fas fa-envelope"></i></a><a href="tel:'.$csArr[0]->getPhone().'"> <i class="fas fa-phone"></i></a>
                            <form  method="POST">
                        </form>
                        </div>
            </div>
        </div>
            </div>
';
```

Front-End



Contact Page

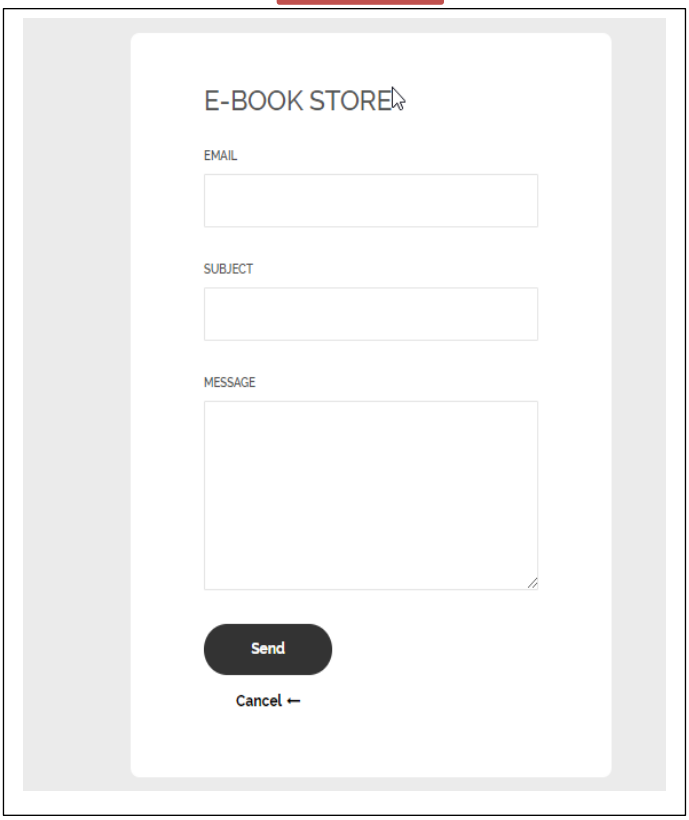
The Contact Form [**fronted for email function**]: this form done has validation using java script, used for sending emails which is used by user to contact admins.

Back-End

```
<div class="limiter">
    <div class="container-login100">
        <div class="wrap-login100 p-l-85 p-r-85 p-t-55 p-b-55">
            <form class="login100-form validate-form flex-sb flex-w" method="Get" action="">
                    <span class="login100-form-title p-b-32">
                        E-Book Store
                    </span>
                    <span class="txt1 p-b-11" >
                        Email
                    </span>
                    <div class="wrap-input100 validate-input m-b-36" data-validate = "Valid email is required: ex@abc.xyz">
                        <input type="email" id="email" class="input100" name="email" >
                            <span class="focus-input100"></span>
                    </div>
                    <span class="txt1 p-b-11" >
                        Subject
                    </span>
                    <div class="wrap-input100 validate-input m-b-36" data-validate = "Subject is required">
                            <input type="text" id="textInput" class="input100" name="subject" >
                        <span class="focus-input100"></span>
                </div>
                    <span class="txt1 p-b-11" >
                        Message
                    </span>
                    <div class="wrap-input100 validate-input m-b-36" data-validate = "Message is required">
                    <textarea type="text" id="textarea" class="input100" name="message" style="height: 200px;" ></textarea>
                        <span class="focus-input100"></span>
                        </div>
```

```
                    <div class="wrap-input100 validate-input m-b-36" data-validate = "Message is required">
                    <textarea type="text" id="textarea" class="input100" name="message" style="height: 200px;" ></textarea>
                        <span class="focus-input100"></span>
                        </div>
                        <div class="container-login100-form-btn">
                            <input class="login100-form-btn" value="Send" name="submit" type="submit">
                        </div>
                </form>
                <div class="Reg-btn" >
                    <a class="Register-form-btn" href="index.php">
                            Cancel
                            <i class="fa fa-long-arrow-left m-l-5"></i>
                        </a>
                </div>
        </div>
    </div>
</div>
 <div id="dropDownSelect1"></div>
```

Front-End



Notes

1-we tried to make the front-end documentation as simple as possible. so, we just put the following code which shows the main part in each page or for using specific function, and some of them with the way of how they related to back-end classes and functions because their front end was included in this linking.

2-there are other pages of front end we did not mention, that because those pages are a not a design for any function and they have no connection with back end

Examples on these pages is: About us Page which shows only information about the system and system creators

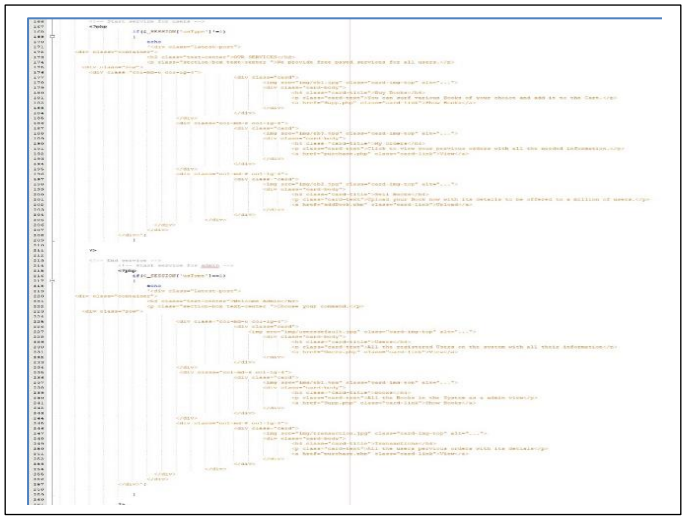
3- Pictures are HD so, if they are zoomed out so please zoom in it will appear very clearly

END OF QUESTION 12

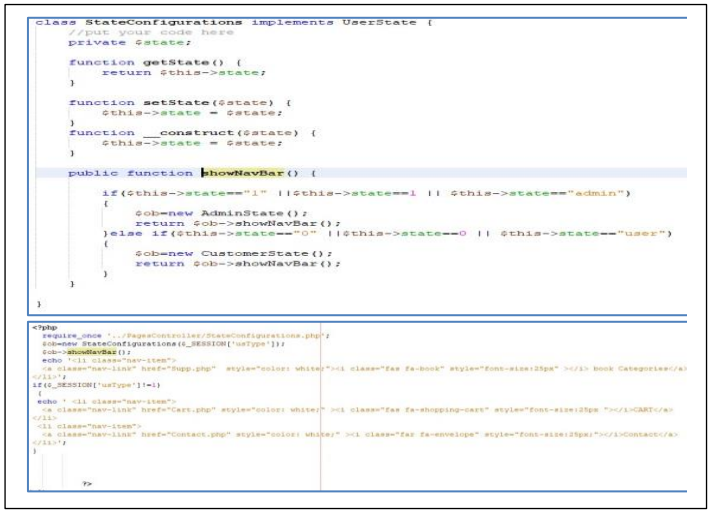
13) Document an Implementation based on the abovementioned Requirements & Design *(it should include all the following modules, in addition of course to modules specific to your individual projects)*:

a) User Role Management Module

•Home Page a have a section for services that the website provide, this section changes depending on usType object value we set the usType for admins equal 1 while for users equal 0 and we check the value and change the Role management as with this feature we used state design pattern to get the state of the object either it was 1 or 0, or any other value if we add new stock holders



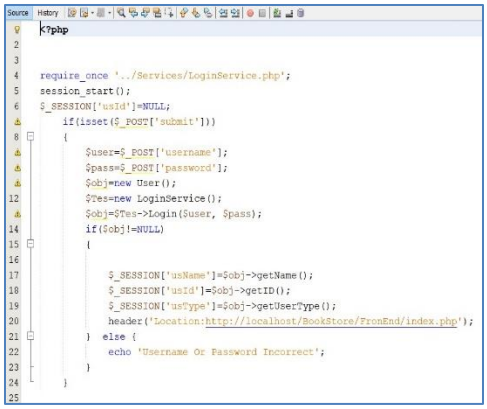
•The showNavBar function shows the way how we used state design pattern as we also change the navbar shape and buttons as the type of the user so, again if there usType was 1 it means admin we show home, books, users, transactions for him. else if usType was 0 it means user we show Home, Books, purchase, my books, cart and contact pages for him

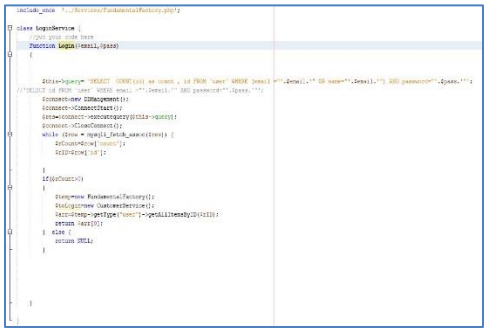


•Also, at books information page we make a role management in that, if a user opens the page so its usType is not equal 1, he will see Add to cart button to buy the book if he want, else if he was admin this section will disappear.

b) User manipulation Module *(Login, Add / Delete / Update / Search, List)*.

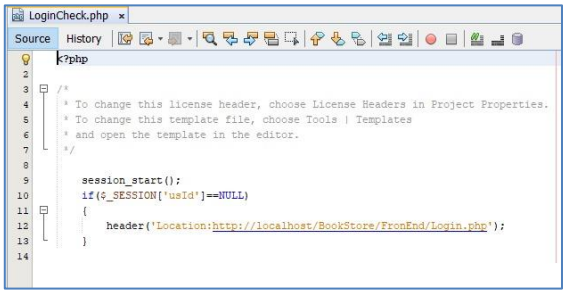
•The login page: this is used to take the username and the password from the login form and save it in object and then used the function login and after that check if it equal null or not, if the object not equal null so then it will enter the home page but if it equal null this mean that there information not correct.





•The login function: this is used to check the information in the database if it found or not found and check the UserType for this information if it 1 or not.

•The Login Check: this used in all pages and it used when anyone want to enter the page of the site without login then it will checking if this user id is found or if it found it will open the page if not found it will go to the login page so the user can enter his information to enter the website.



•The register page: this is used to add user and it work by when the user enters his information and then click submit it will check if the password and the confirm password are the same and then it will create object from create user function to use to check if this object is equal 1 so it will print done else it will return wrong information.

•The create user function: this working by enter the information and check if the information is not repeated and correct then the function will return

•The delete: this done when the delete button clicked it will send in the post “trash” then we will see if there is any “trash” in the post then we will use the get type to delete the user and the post “trash” which is the user id.

```php
<?php
    require_once '../PagesController/LoginCheck.php';
    require_once '../PagesController/userController.php';
    require_once '../Services/FundamentalFactory.php';
    $us=new userController();
    $temp=new FundamentalFactory();

    if(isset($_POST['trash']))
    {
        $temp->getType("user")->delete($_POST['trash']);
    }


?>
```

•The delete function: this work by take the id and from the post “trash” which get it from the front end.

```php
public function delete($id) {
    $db = new  DBMangement();
  $db->ConnectStart();
  $query = "DELETE FROM user WHERE id = $id ";
  $res = $db->executequery($query);
  $db->CloseConnect();
  return $res;//this return bool
}
```

•The update function: this is used to update the user information and it work by using the id to know which user want to change his information and then user can change any of his information.

```php
function updateCustomer(User $user){
    $con= new DBMangement();
    $con->ConnectStart();
    $query="UPDATE user SET name= "'.$user->getName().'" ,email= "'.$user->getEmail().'"'
            . ' ,password= "'.$user->getPassword().'" ,phone= "'.$user->getPhone().'" ,user_type= "'.$user->getUserType().'" '
            . ',can_read= "'.$user->getCan_read().'" ,can_write= "'.$user->getCan_write().'" '
            . ',can_update= "'.$user->getCan_update().'" ,can_delete= "'.$user->getCan_delete().'" where id= "'.$user->getID().'" ';
    $res= $con->executequery($query);
    $con->CloseConnect();
    if($res!=1)
    {
        $this->function_alert("Error");
    }
}
```

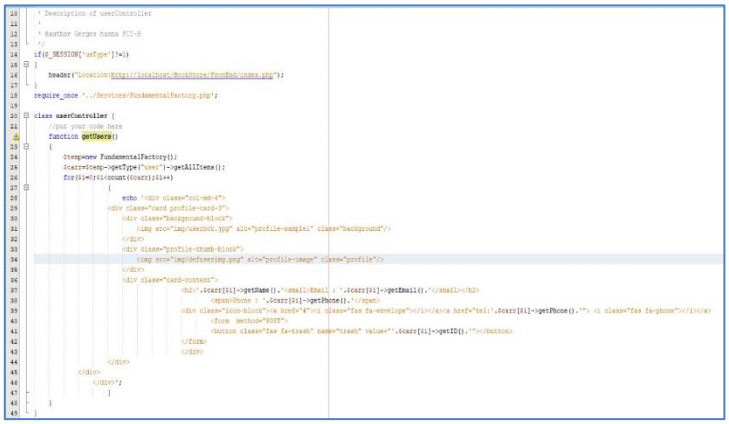
•The search function: this is used to search for user and this work by using the id when we want to find the buyer or the seller or information about any user.

```php
    // to search about users
    public function getAllItemsByID($id) {

      $db = new  DBMangement();
      $db->ConnectStart();
      $query = "SELECT * FROM user WHERE id = $id ";
      $res = $db->executequery($query);
       $arr = array() ;

        while ($row=  mysqli_fetch_assoc($res))
            {

                $cu=new User();
                $cu->setID($row['id']);
                $cu->setName($row['name']);
                $cu->setEmail($row['email']);
                $cu->setPassword($row['password']);
                $cu->setPhone($row['phone']);
                $cu->setUserType($row['user_type']);
                $cu->setCan_read($row['can_read']);
                $cu->setCan_write($row['can_write']);
                $cu->setCan_update($row['can_update']);
                $cu->setCan_delete($row['can_delete']);
                $arr[] = $cu ;

            }
      $db->CloseConnect();
      return $arr;//return data of one customer
    }

}
```

•The list user: this is used to show the information of the user and it found the user page and this page was shown by the admin only and it work by call the function get users to get the information of all users.

```
34          <!--start navbar -->
35          <?php
36            require_once '../PagesController/DesignController.php';
37          ?>
38          <!--end navbar -->
39          <!--Card --><!--Card -->
40  <div class="users">
41          <div class="container">
42          <div class="row">
43
44                  <?php
                        $us->getUsers();
46                  ?>
47
48          </div>
49  </div>
```

•The get users function: this done by create the html formation that will show by the admin but before this done it call the get all item from [type users] from the function get type and then after the information return all the user will be go on loop to create the html template for them.



•The get all items function: this is work by return all the information of users in array.

```
97      //list all users
99      public function getAllItems() {
100         $db = new  DBMangement();
101       $db->ConnectStart();
102       $query = "SELECT * FROM user  ";
103       $res = $db->executequery($query);
104        $arr = array() ;
105
106          while ($row=  mysqli_fetch_assoc($res))
107              {
108
109                  $cu=new User();
110                  $cu->setID($row['id']);
111                  $cu->setName($row['name']);
112                  $cu->setEmail($row['email']);
113                  $cu->setPassword($row['password']);
114                  $cu->setPhone($row['phone']);
115                  $cu->setUserType($row['user_type']);
116                  $cu->setCan_read($row['can_read']);
117                  $cu->setCan_write($row['can_write']);
118                  $cu->setCan_update($row['can_update']);
119                  $cu->setCan_delete($row['can_delete']);
120                  $arr[] = $cu ;
121
122             }
123             $db->CloseConnect();
124       return $arr;//return data of one customer
125     }
```

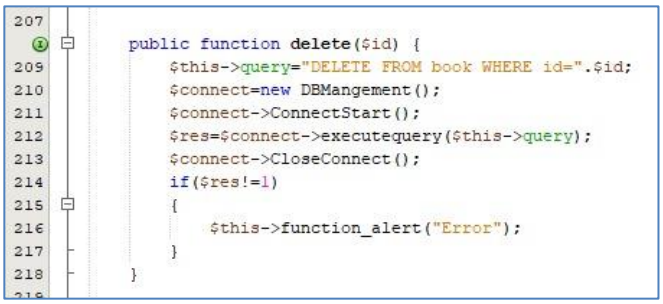
c) Controlling Resources Module *(Rooms, Orders, Products, ... etc.)*.

•The Add: this is used to add book, and this is work by take the book [name, price, description, quantity, isbn, authoname, status, category-id, customer-id, image-URL, book-condition].
and add them to database

•The List: this is used to get all the books in the page and it work by show all the information of the book [name, price, description, quantity, isbn, authoname, status, category-id, customer-id, image-URL, book-condition].

```php
public function getAllItems() {
    $connect=new DBMangement();
    $connect->ConnectStart();
    $this->query = 'select * from book ORDER BY id DESC;';
    $res=$connect->executequery($this->query);
    $array = array();
    while ($row=  mysqli_fetch_assoc($res))
        {
            $book=new Book();
            $book->setId($row['id']);
            $book->setCustomer_id($row['customer_id']);
            $book->setName($row['name']);
            $book->setAuthor($row['autho_name']);
            $book->setPrice($row['price']);
            $book->setStock($row['quantity']);
            $book->setDescription($row['description']);
            $book->setCategory($row['category_id']);
            $book->setImage($row['image_url']);
            $book->setIsbn($row['isbn']);
            $book->setCondition($row['book_condition']);
            $book->setStatus($row['status']);
            $array[] = $book ;
        }
    $connect->CloseConnect();
    return $array ;
}
```

•The Delete: this is used to delete book and it work by take the id of the book and will search for this and delete it.



```php
public function delete($id) {
    $this->query="DELETE FROM book WHERE id=".$id;
    $connect=new DBMangement();
    $connect->ConnectStart();
    $res=$connect->executequery($this->query);
    $connect->CloseConnect();
    if($res!=1)
    {
        $this->function_alert("Error");
    }
}
```

•The Search By Seller-id: this is used to find a book and it work by using the id of the seller of the book to show the book information [name, price, description, quantity, isbn, authoname, status, category-id, customer-id, image-URL, book-condition].

```php
Function getBookBySeller($id)
{
    $connect=new DBMangement();
    $connect->ConnectStart();
    $this->query = 'select * from book where customer_id = ' . $id.' ORDER BY id DESC;';
    $res=$connect->executequery($this->query);
    $array = array();
    while ($row=  mysqli_fetch_assoc($res))
        {
            $book=new Book();
            $book->setId($row['id']);
            $book->setCustomer_id($row['customer_id']);
            $book->setName($row['name']);
            $book->setAuthor($row['autho_name']);
            $book->setPrice($row['price']);
            $book->setStock($row['quantity']);
            $book->setDescription($row['description']);
            $book->setCategory($row['category_id']);
            $book->setImage($row['image_url']);
            $book->setIsbn($row['isbn']);
            $book->setCondition($row['book_condition']);
            $book->setStatus($row['status']);
            $array[] = $book ;
        }
    $connect->CloseConnect();
    return $array;
}
```

•The Search By Book-id: this is used to find a book and it work by using the id of the book of the book to the book information [name, price, description, quantity, isbn, authoname, status, category-id, customer-id, image-URL, book-condition].

```php
public function getAllItemsByID($id) {
    $connect=new DBMangement();
    $connect->ConnectStart();
    $this->query = 'select * from book where id = ' . $id.' ORDER BY id DESC;';
    $res=$connect->executequery($this->query);
    $array = array();
    while ($row=  mysqli_fetch_assoc($res))
        {
            $book=new Book();
            $book->setId($row['id']);
            $book->setCustomer_id($row['customer_id']);
            $book->setName($row['name']);
            $book->setAuthor($row['autho_name']);
            $book->setPrice($row['price']);
            $book->setStock($row['quantity']);
            $book->setDescription($row['description']);
            $book->setCategory($row['category_id']);
            $book->setImage($row['image_url']);
            $book->setIsbn($row['isbn']);
            $book->setCondition($row['book_condition']);
            $book->setStatus($row['status']);
            $array[] = $book ;
        }
    $connect->CloseConnect();
    return $array;
}
```

•The Search By category-id: this is used to find a book and it work by using the id of the category of the book to show the book information [name, price, description, quantity, isbn, authoname, status, category-id, customer-id, image-URL, book-condition].

```php
Function getBookByCat($catID)
{
    $connect=new DBMangement();
    $connect->ConnectStart();
    if($catID==0)
    {
        $this->query = 'select * from book ORDER BY id DESC;';
    }else
    {
        $this->query = 'select * from book where category_id = ' . $catID.' ORDER BY id DESC;';
    }

    $res=$connect->executequery($this->query);
    $array = array();
    while ($row=  mysqli_fetch_assoc($res))
        {
            $book=new Book();
            $book->setId($row['id']);
            $book->setCustomer_id($row['customer_id']);
            $book->setName($row['name']);
            $book->setAuthor($row['autho_name']);
            $book->setPrice($row['price']);
            $book->setStock($row['quantity']);
            $book->setDescription($row['description']);
            $book->setCategory($row['category_id']);
            $book->setImage($row['image_url']);
            $book->setIsbn($row['isbn']);
            $book->setCondition($row['book_condition']);
            $book->setStatus($row['status']);
            $array[] = $book ;
        }
    $connect->CloseConnect();
    return $array;
}
```

•The Search By Book-name: this is used to find a book and it work by using the name of the book to show the book information [name, price, description, quantity, isbn, authoname, status, category-id, customer-id, image-URL, book-condition].

```php
Function getBookByName($name)
{
    $connect=new DBMangement();
    $connect->ConnectStart();
    $this->query = 'select * from book where name = ' . $name.' ORDER BY id DESC;';
    $res=$connect->executequery($this->query);
    $array = array();
    while ($row=  mysqli_fetch_assoc($res))
        {
            $book=new Book();
            $book->setId($row['id']);
            $book->setCustomer_id($row['customer_id']);
            $book->setName($row['name']);
            $book->setAuthor($row['autho_name']);
            $book->setPrice($row['price']);
            $book->setStock($row['quantity']);
            $book->setDescription($row['description']);
            $book->setCategory($row['category_id']);
            $book->setImage($row['image_url']);
            $book->setIsbn($row['isbn']);
            $book->setCondition($row['book_condition']);
            $book->setStatus($row['status']);
            $array[] = $book ;
        }
    $connect->CloseConnect();
    return $array;

}
```

•The Search by Search bar: this is used to find a book and it work by using the search bar to show the book information [name, price, description, quantity, isbn, authoname, status, category-id, customer-id, image-URL, book-condition]

```php
Function getBookBySearchBar($search)
{
    $connect=new DBMangement();
    $connect->ConnectStart();
    $this->query = 'SELECT * FROM book WHERE name LIKE "%".$search."%" OR price LIKE "%".$search."%" OR isbn LIKE "%".$search."%" ORDER BY id DESC;';
    $res=$connect->executequery($this->query);
    $array = array();
    while ($row=  mysqli_fetch_assoc($res))
        {
            $book=new Book();
            $book->setId($row['id']);
            $book->setCustomer_id($row['customer_id']);
            $book->setName($row['name']);
            $book->setAuthor($row['autho_name']);
            $book->setPrice($row['price']);
            $book->setStock($row['quantity']);
            $book->setDescription($row['description']);
            $book->setCategory($row['category_id']);
            $book->setImage($row['image_url']);
            $book->setIsbn($row['isbn']);
            $book->setCondition($row['book_condition']);
            $book->setStatus($row['status']);
            $array[] = $book ;
        }
    $connect->CloseConnect();
    return $array;
}
```

•The Update: this is used to update the information of the book and it work by using the id of the book to update the information of the book [name, price, description, quantity, isbn, authoname, status, category-id, customer-id, image-URL, book-condition].

```
Function UpdateBook(Book $book)
{
    $this->query='UPDATE book SET '
            . 'name="'.$book->getName().'",'
            . 'description="'.$book->getDescription().'",'
            . 'price='.$book->getPrice().','
            . 'quantity='.$book->getStock().','
            . 'isbn='.$book->getIsbn().','
            . 'image_url="'.$book->getImage().'",'
            . 'autho_name="'.$book->getAuthor().'",'
            . 'book_condition="'.$book->getCondition().'",'
            . 'customer_id='.$book->getCustomer_id().','
            . 'status="'.$book->getStatus().'",'
            . 'category_id='.$book->getCategory().''
            . ' WHERE id = '.$book->getId().';';
    $connect=new DBMangement();
    $connect->ConnectStart();
    $res=$connect->executequery($this->query);
    $connect->CloseConnect();
    if($res!=1)
    {
        $this->function_alert("Error");
    }
    else {
            echo 'Done';
    }
}
```

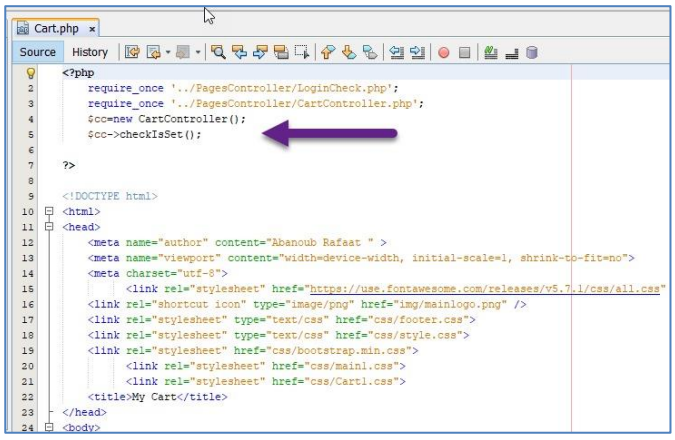
- The Update-Quantity: this is used to update the quantity of the book and it work by using the id of the book to update the quantity only of the book.

```
Function UpdateBookQuantity(Book $book)
{
    $this->query='UPDATE book SET quantity=quantity+'.$book->getStock().' WHERE id = '.$book->getId().';';
    $connect=new DBMangement();
    $connect->ConnectStart();
    $res=$connect->executequery($this->query);
    $connect->CloseConnect();
    if($res!=1)
    {
        $this->function_alert("Error");
    }

}
```

d) Reservation and Rescheduling Module.

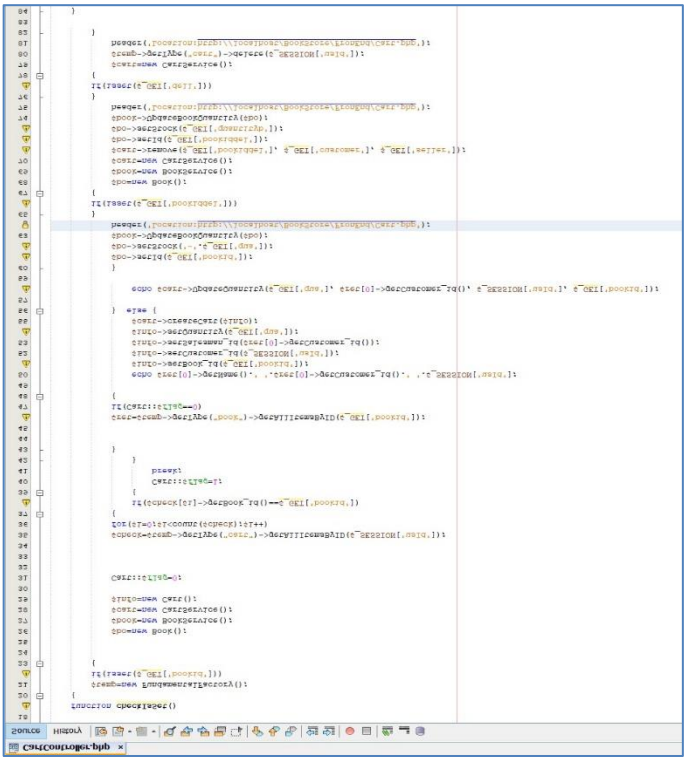
•First, if we want to buy book and then we will click submit button [this button will have the information of the user and the quantity of the book that the user want to buy] and also this button was be enable only if the book in stock if it not in the stock the site will disable this button.

•Second, this check is used to see if the quantity that the user enter is in stock or out of stock if it in stock then it will go to the cart page if not it will return message for the user.



•Third, then it will call function is set after that if the is set is run correct it will save this book in the cart.

•The isset function: this is used to check if this book-id is found or not then if the id found it will decrease this amount from the book quantity that the user enters from the site to avoid error of buying book not in the site.



```
            </div>

                <a href="order.php" class="btn btn-update">Order Now</a>

    </div>
```

•Then the order button used after the above process done and this button found in the cart page.

•After click the order button it will take some information form the user [country, city, address, postal, phone] and after that it will create object from the order to get the payment-method, shipping-method, customer-id, id and this done after the user click order now and after that it will decrease this quantity of book from the site.

order.php

```php
<?php
    require_once '../PagesController/LoginCheck.php';
    require_once '../Services/AddressService.php';
    require_once '../Services/FundamentalFactory.php';

    $address=new Address();
    $cart=new Cart();
    $orSer=new OrderService();
    $adSer=new AddressService();
    $temp=new FundamentalFactory();


    if(isset($_POST['submit']))
    {
        $address->setCountry($_POST['country']);
        $address->setCity($_POST['city']);
        $address->setAddress($_POST['address']);
        $address->setPostal($_POST['postal']);
        $address->setPhone($_POST['phone']);

        $arr=$temp->getType("cart")->getAllItemsByID($_SESSION['usId']);

        $order=new Order();
        $order->setPayment_method($_POST['type']);
        $order->setSipping_method(1);
        $order->setAddressID($adSer->CreateAddress($address));
        $order->setCustomerID($_SESSION['usId']);

        for($i=0;$i<count($arr);$i++)
        {
            $order->setBook_id($arr[$i]->getBook_id());
            $order->setQuantity($arr[$i]->getQuantity());
            $bArr=$temp->getType("book")->getAllItemsByID($arr[$i]->getBook_id());
            $qua=$bArr[0]->getPrice()*$arr[$i]->getQuantity();
            $order->setTotal($qua);
            if($orSer->createOrder($order)==1)
            {
                echo 'Done';
                $temp->getType("cart")->delete($_SESSION['usId']);
            }

        }
```

e) Generating Reports Module *(PDFs, … etc.)*.

•This is done in the purchase page in the frontend we will find button called “generate report” which is used to generate the pdf.

•The PDF page: first the header, footer and header table are just used to design the shape of the PDF, then the view table used to view the information of the PDF on it and finally, the Run PDF is used to get the information from the database so then check if the usType if equal 0 or not if it equal 0 it will run the PDF as the user type but if it equal 1 this means that it will run the PDF for admin.

```
<div class="tf">
        <div class="row layout-inline">
         <div class="col">
             <p>Total : $<?php echo $count; ?></p>
              <button onclick="document.location = 'PDF.php'">PDF</button>
         </div>
         <div class="col"></div>
    </div>
</div>
```

PDF.php

Source History

```
<?php

require_once '../fpdf182/fpdf.php';
require_once '../Services/OrderService.php';
require_once '../Services/BookService.php';
require_once '../Services/CustomerService.php';
require_once '../Services/FundamentalFactory.php';
class TestPDF extends FPDF{
    function header() {
        $this->Image('logo.png',80,8);
        $this->SetFont('Arial', 'B', 30);
        $this->Cell(276,18,'Book Store',0,0,'C');
        $this->Ln();
    }
    function footer()
    {
        $this->SetY(-15);
        $this->SetFont('Arial', '', 8);
        $this->Cell(0,10,'Page '.$this->PageNo().'/{nb}',0,0,'C');
    }
    function headerTable(){
        $this->SetFont('Times','B',18);
        $this->Ln();
        $this->Cell(40,10,'ID',1,0,'C');
        $this->Cell(80,10,'Book Name',1,0,'C');
        $this->Cell(30,10,'Quantity',1,0,'C');
        $this->Cell(45,10,'Payment Type',1,0,'C');
        $this->Cell(40,10,'Seller',1,0,'C');
        $this->Cell(50,10,'Total',1,0,'C');
        $this->Ln();

    }
    function viewTable($id,$name,$qty,$type,$seller,$total)
    {
            $this->SetFont('Times','',10);
            $this->Cell(40,10,$id,1,0,'L');
            $this->Cell(80,10,$name,1,0,'L');
            $this->Cell(30,10,$qty,1,0,'L');
            $this->Cell(45,10,$type,1,0,'L');
            $this->Cell(40,10,$seller,1,0,'L');
            $this->Cell(50,10,$total,1,0,'L');
            $this->Ln();

    }
    function RunPDF()
    {
        session_start();
        $ord=new OrderService();
        $book=new BookService();
        $cs=new CustomerService();
        $temp=new FundamentalFactory();
        if($_SESSION['usType']==0)
        {
            $OrArr=$temp->getType("order")->getAllItemsByID($_SESSION['usId']);
            //$OrArr=$ord->getOrderById($_SESSION['usId']);
        } else if($_SESSION['usType']==1) {
            //$OrArr=$ord->getOrders();
             $OrArr=$temp->getType("order")->getAllItems();
        }
        $pdf=new TestPDF();
        $pdf->AliasNbPages();
        $pdf->AddPage('L', 'A4', 0);
        $pdf->headerTable();
        for($i=0;$i<  count($OrArr);$i++)
        {

            $boArr= $temp->getType("book")->getAllItemsByID($OrArr[$i]->getBook_id());
                    //$book->getBookById($OrArr[$i]->getBook_id());
            $csArr=$temp->getType("user")->getAllItemsByID($boArr[0]->getCustomer_id());
                    // $cs->getCustomerById($boArr[0]->getCustomer_id());
            $pdf->viewTable($OrArr[$i]->getID(),$boArr[0]->getName(),$OrArr[$i]->getQuantity()
                    ,$OrArr[$i]->getPayment_method(),$csArr[0]->getName(),$OrArr[$i]->getTotal());
        }
        $pdf->Output();
    }
}


// this is to run the function and execute the pdf
$q=new TestPDF();
$q->RunPDF();
```

f) Sending Emails or Notifications Module.

```php
<?php
class EmailSender{

public Function send_email($from,$to ,$subject ,$message)
{
   $header = "From:".$from;
  $isMailSent =  mail($to ,$subject,$message,$header);
  if($isMailSent)
  {
     echo "email sent successfuly";
  }
  return $isMailSent;
}
}
```

---

php.ini file
SMTP=smtp.gmail.com
smtp_port=587
sendmail_from = abanoubkamal@gmail.com
sendmail_path =
"\"C:\xampp\sendmail\sendmail.exe\" -t" ----------
---

sendmail.ini file
smtp_server=smtp.gmail.com
smtp_port=587
error_logfile=error.log
debug_logfile=debug.log
auth_username=abanoub82@gmail.com
auth_password=bebo512348
force_sender=abanoub82@gmail.com

---

php has built in function called mail that has 4 parameters the sender mail and receiver mail and subject and massage.

Here we set the configuration to send all massage from one mail.

user will write his message and his mail, and the function will send massage to our mail by the detected stander mail.

Later we can replay to user by the detected mail to his mail

g) File Uploaders

•Image uploader: we used this function in choosing cover book image from the file so when the user clicks on choose image the name and the format of the image will be taken to be saved in database and the cover book photo will appear. Ex: test.jpg when the user chooses this now image section carries it.

```
<div class="custom-file" style="margin-bottom: 50px;">
        <label for="myfile" class="txtl p-b-11">IMAGE:</label>
        <input type="file"  id="myfile" name="myfile"  >
</div>
```

AddBookCont.php

```
<?php
include_once '../Services/BookService.php';
include '../PagesController/CategoryController.php';

$cat=new CategoryController();
$catArr=$cat->getCategorey();
if(isset($_GET['submit']))
{
    if( $_GET['cat']=="Empty")
    {
        echo 'Please Select Category';
    }
    else {

     $book=new Book();
     $book->setName($_GET['name']);
     $book->setDescription($_GET['desc']);
     $book->setPrice($_GET['price']);
     $book->setStock($_GET['quantity']);
     $book->setIsbn($_GET['isbn']);
     $book->setImage($_GET["myfile"]);
     $book->setAuthor($_GET['publisher']);
     $book->setCondition($_GET['bookCon']);
     $book->setCustomer_id($_SESSION['usId']);
     $book->setStatus("Available");
     $book->setCategory($_GET['cat']);

       $bookServ=new BookService();
       $ret=$bookServ->CreateBook($book);
       echo $ret;


    }
}
```

- This function upload Books as a pdf it takes the source of the pdf book if error found it printed if not it checks if the file already exist it printed this out else it uploads it and print its information

```
if ( isset( $_FILES['pdfFile'] ) ) {
    if ($_FILES['pdfFile']['type'] == "application/pdf") {
            $source_file = $_FILES['pdfFile']['tmp_name'];
            $dest_file = "upload/".$_FILES['pdfFile']['name'];

            if (file_exists($dest_file)) {
                print "The file name already exists!!";
            }
            else {
                move_uploaded_file( $source_file, $dest_file )
                or die ("Error!!");
                if($_FILES['pdfFile']['error'] == 0) {
                    print "Pdf file uploaded successfully!";
                    print "<b><u>Details : </u></b><br/>";
                    print "File Name : ".$_FILES['pdfFile']['name']."<br.>"."<br/>";
                    print "File Size : ".$_FILES['pdfFile']['size']." bytes"."<br/>";
                    print "File location : upload/".$_FILES['pdfFile']['name']."<br/>";
                }
            }
    }
    else {
            if ( $_FILES['pdfFile']['type'] != "application/pdf") {
                print "Error occured while uploading file : ".$_FILES['pdfFile']['name']."<br/>";
                print "Invalid  file extension, should be pdf !!"."<br/>";
                print "Error Code : ".$_FILES['pdfFile']['error']."<br/>";
            }
    }
```

Notes

1-we tried to gather all the possible needed modules but there are additional functions in our project as its very large and we made so many requirements that wasn't needed in SWE-1 Projects, so we didn't mention those above.

2-Some images may be duplicated with those if front end question as the fact that the front section was included internal the functions as echo

3- Pictures are HD so, if they are zoomed out so please zoom in it will appear very clearly

END OF QUESTION 13

14) Design some User Interfaces

The Interaction Design is used to test if the interface of the system is friendly user or not by using the 5 dimensions of it:

1. **The Words:**

•This means that the word used in the system must be easy to understand and give information about example [home page, back button] but also doesn't contain much information just the information that gives the meaning.

2.**The Visual Representations:**

•This type means that the icon or the image that used in the system must be easy to understand because this element used to give the information to the user so, they must be easy and friendly.

3.**The Physical Object or Space**:

•This type describes how the user physically interact with this system example [using laptop, mobile phone] and where the spaces that the user use this system example [supermarket, school] so all this information will affect this dimension.

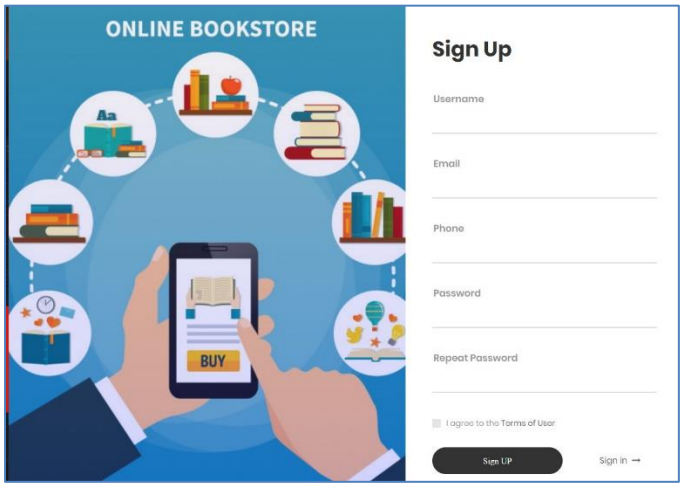
4.**The Time**:

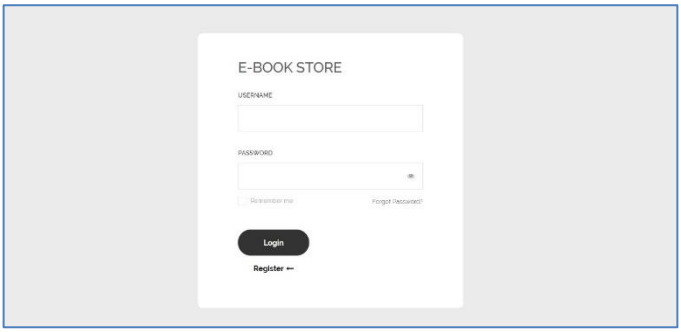
•This type used to define how much does the system take to get what the user need and also how much time the animation take in the system which consider as important major for all the user to make the system doesn't need to much time to obtain what they need.

5.**The Behavior**:

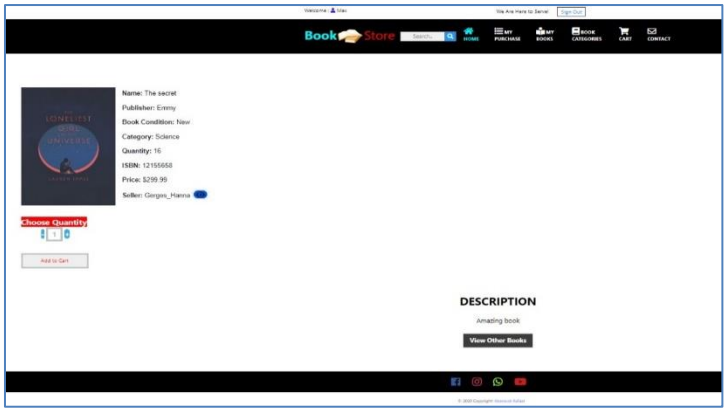
•This type used to show the user can perform his action in the system and what the feedback from the user about this system and it depends on the previous dimension.

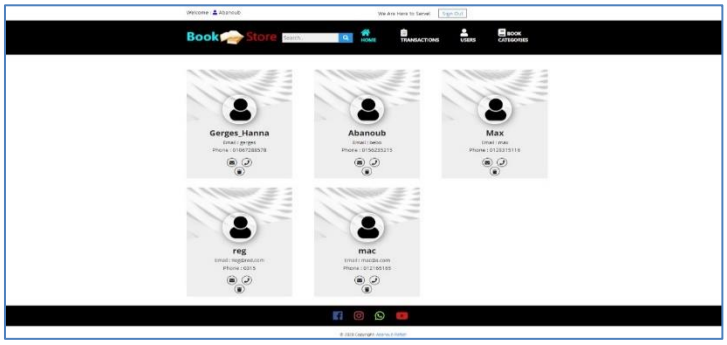
•The first dimension [**words**]: in this dimension we use the word (login, register) in first page to help the user to register if doesn't have account, also use the words (home, my purchase, my book, book category, cart, contact) in the navbar if user use the site, also use word (order now) in cart page, but if admin use the site he will find another words to help him (home, transaction, users, book-category).





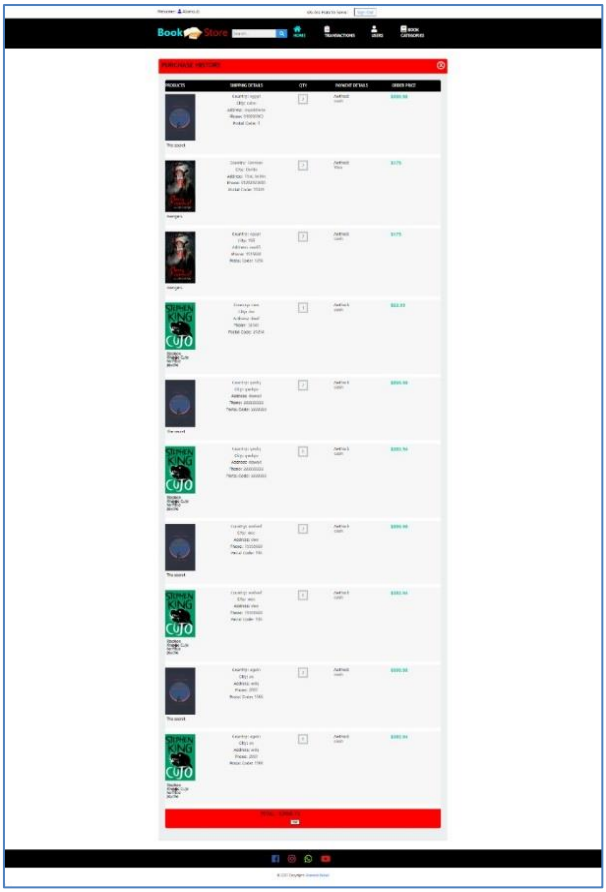
•The second dimension [**Visual Representations**]: in this dimension we use some image to help the user example in category page we use the shape of book to represent the book category, also in book page we use the (+ an -) to use in decrease and increase the number of books to order, in the cart page we use the shape of trash basket to represent the delete and in if the admin use the site and open the user page he will found the information of all the user and found some shape to help to contact or delete user.

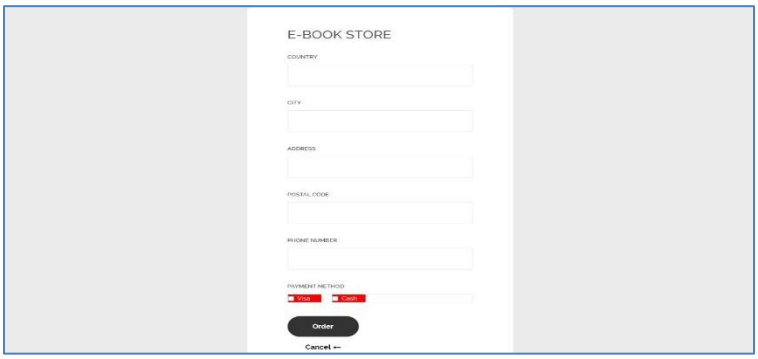




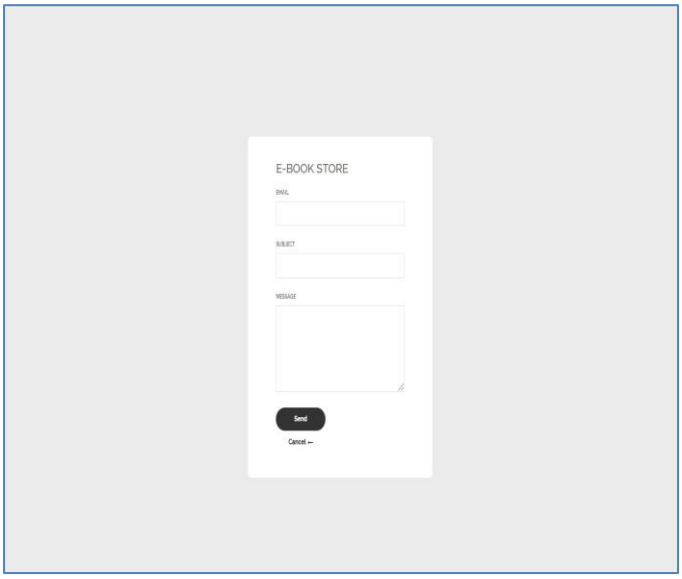
•The third dimension [**Physical Object or Space**]: in this dimension we make our system can work with laptop with different operation system [mac, windows] and also the user can use this system from home or any place doesn't require some special places to browse the site.

•The fourth dimension [**Time**]: in this dimension represent how the user can use the site example if he tries to register and login it doesn't need much time and also if he tries to make order or add to cart doesn't need much time after that he can cancel the order or order now and after order now he will fill some information and also if the admin wants to see all the process in the site he will click the transaction button.





•The fifth dimension [**Behavior**]: in this dimension it shows how the user can use the system and, we can get his feedback or any question from him in the contact page.



END OF PART 3