VIETNAMESE – GERMAN UNIVERSITY

FACULTY OF ENGINEERING

COMPUTER SCIENCE DEPARTMENT

PROJECT REPORT

ONLINE LIBRARY MANAGEMENT

Module 21: Programming Exercises – Team 9

1. Nguyen Khoa - 15001
2. Nguyen Van Khanh - 14559
3. Huynh Duong Khai - 16235
4. Tran Huu Hoang Do - 14551
5. Hoang The Vinh - 15382

Lecturer: Huynh Trung Hieu .Ph.D.

Binh Duong, SS2023

ABBREVIATION

Ph.D	Doctor of Philosophy
GUI	Graphic User Interface
DB	Database
CV	Curriculum Vitae

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

I. INTRODUCTION

In this project, we will create an online library to help the learning process for all grades of students and any people who have the urge to read a book or do some research. Since the Covid-19 outbreak, people have limited access to the offline library or to rent books physically. We hope that our project may provide all the necessary books and information for people who need them.

We use HTML, CSS, and JavaScript to create GUI, C# combined with ASP.NET framework for the Application, and MySQL for the Database. Moreover, to create a friendly interface we will design the interface using AdobeXD. Lastly, we will use Markdown for the report of the project.

Administrator:

- Input renting ( )
- Delete renting ( )
- Display renting details ( )
- Keep track about returned book ( )
- Update return ( )
- Add books ( )
- Delete books ( )
- Modify books ( )
- Modify payments ( )

Customer:

- Browse books ( )
- Filter based on category author publication ( )
- Filter based on reading habits ( )
- Like books ( )
- Review books ( )
- Pay for the rent ( )
- Display renting details ( )

II. CLASS ANALYSIS

1. Objects

a. Use Case Diagram

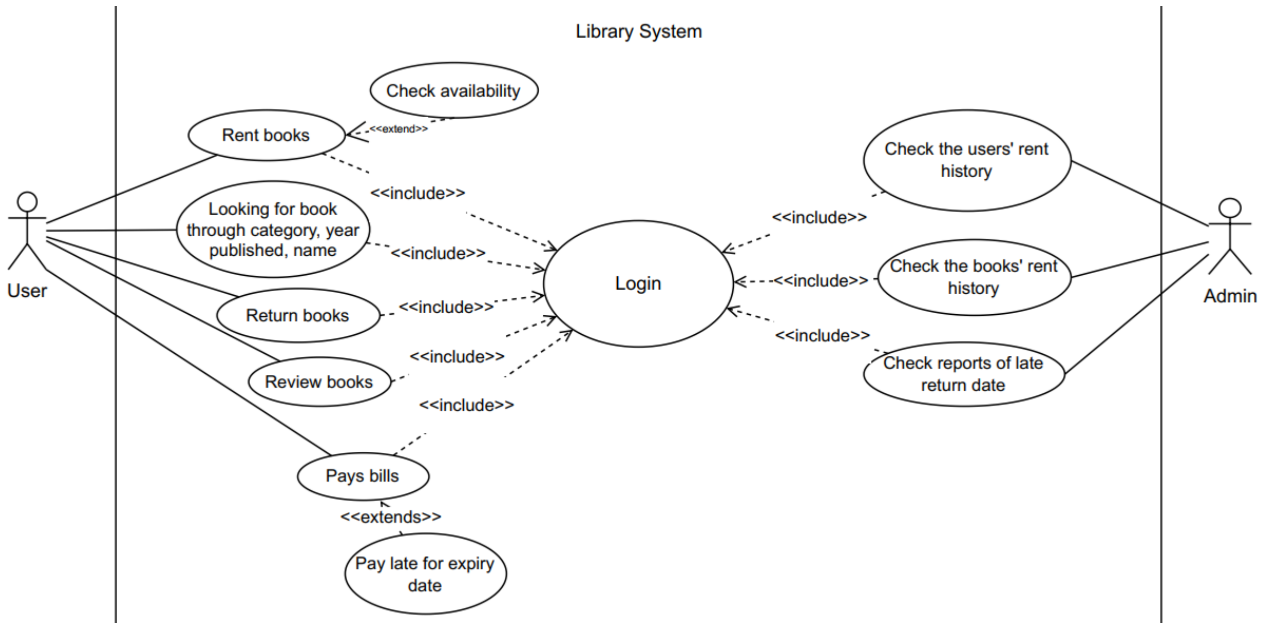

Library System
User
Admin
Check availability
Rent books
<<extend>>
<<include>>
Looking for book through category, year published, name
Return books
Review books
Pays bills
<<extends>>
Pay late for expiry date
Login
Check the users' rent history
Check the books' rent history
Check reports of late return date

Figure 1: *Use Case Diagram*

b. Objects Table

No.	Object Name	State	Behaviors
1.	Author	Author_ID, Author_fname, Author_lname.	Getters and Setters methods
2.	Book	ID, Title, Category_ID, Publication_Year, Quantity, Description.	
3.	Book_Author	BookID, AuthorID.	

4.	Category	Category_ID, Category_name.	
5.	Filter	List<Author>, List<Category>, List<Publication>.	
6.	Fine	SessionID, Fine_days, Fine_amount.	
7.	Lease	SessionID, Book_ID, ISSN, Lease_date, Expiry_date, Status, Name_Book, report.	
8.	Payment	Payment_ID, Customer_ID, Lease_date, Payment_date, Payment_amount.	
9.	Review	ISSNum, IDBook, Review_date, Review_context, Review_star.	
10.	User	ISSN, Name, Email, Address, Phone, Pass.	
11.	ViewBook	Nested classes attributes and methods from Book, Author, Review, Category class. Another attribute is duration, which is of its own.	Getter and Setter methods of its own, and Initiate() to take all data' book.

12.	ViewLeases_User	Nested classes attributes and methods from User and Lease class.	Getter and Setter methods of its own.

Table 1: *Objects Table*

2. Classes

a. Database Design

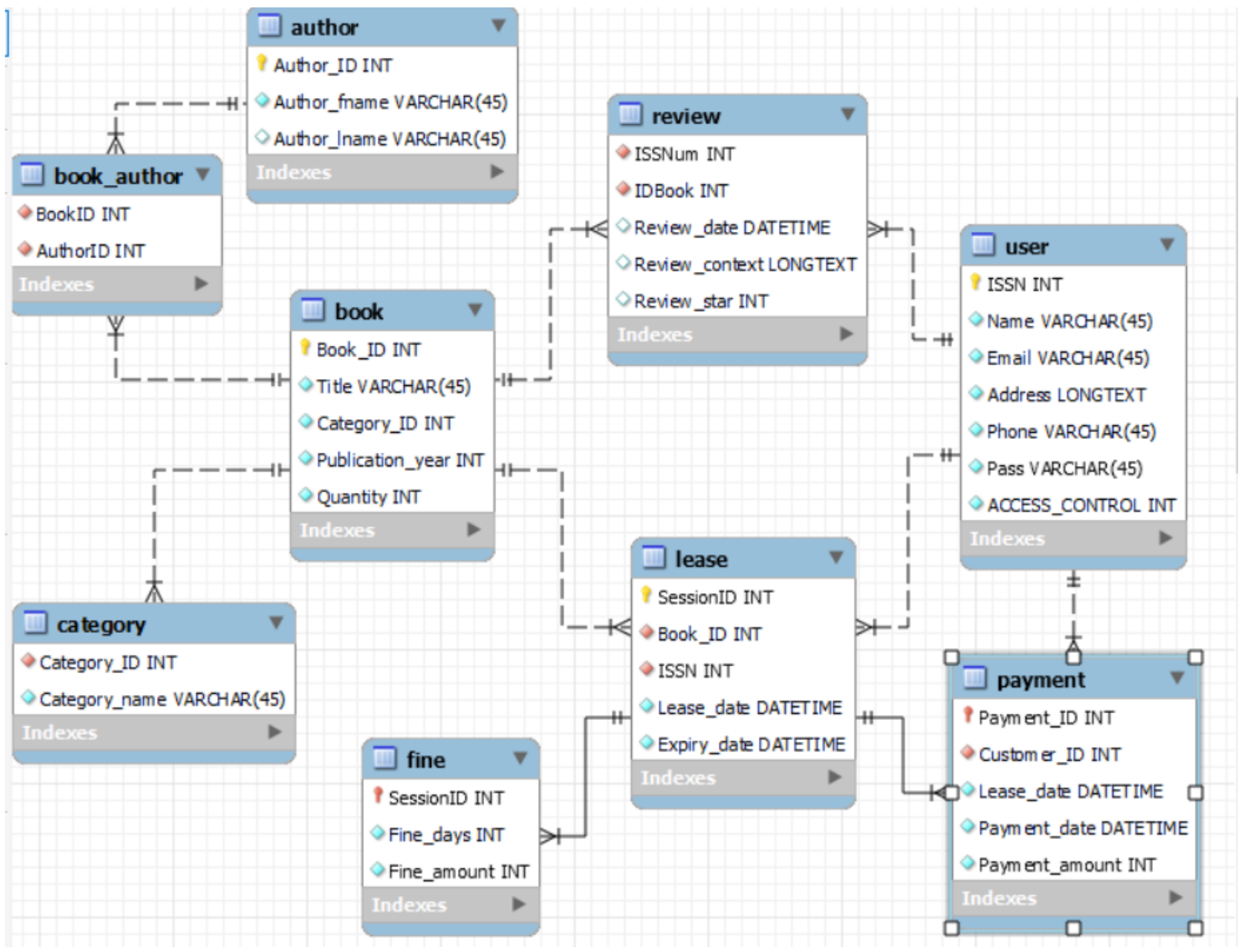


Figure 2: *Database Schema*

b. Communication with Database Layer

No.	**Classes**	**Methods**

1.	AuthorDataAccessLayer	GetAllAuthors(), AddAuthor(), UpdateAuthor(), GetAuthorData(), DeleteAuthor().
2.	BookDataAccessLayer	GetAllBooks(), AddBook(), UpdateBook(), GetBookData(), DeleteBook().
3.	Book_AuthorDataAccessLayer	GetAllBook_Authors(), AddBook_Author(), UpdateBook_Author(), GetBook_1AuthorData(), GetBook_AuthorData(), DeleteBook_Author(), DeleteBook_AllAuthors().
4.	CategoryDataAccessLayer	GetAllCategories(), AddCategory(), UpdateCategory(), GetBook_CategoryData(), GetCategoryData(), GetBooks_Category().
5.	FilterDataAccessLayer	GetFilterBooks().
6.	FineDataAccessLayer	GetAllFines(), AddFine(), UpdateFine(), GetFineData(), DeleteFine(), auto_update().
7.	LeaseDataAccessLayer	GetAllLeases(), AddLease(), UpdateLease(), GetLeaseData(), DeleteLease(), Take_new_SessionID(), Take_num_leases_for_Book().
8.	PaymentDataAccessLayer	GetAllPayments(), AddPayment(), UpdatePayment(), GetPaymentData(), DeletePayment().

9.	ReviewDataAccessLayer	GetAllReviews(), AddReview(), UpdateReview(), GetReviewData(), GetReviewDataForBook(), DeleteReview().
10.	UserDataAccessLayer	GetAllUsers(), AddUser(), UpdateUser(), GetUserData(), DeleteUser().
11.	MySqlConnection	Open(), Close(), CommandType.StoredProcedure(), ExecuteNonQuery(), ExecuteReader().

Table 2: *Communication with Database Layer*

c. Business Logic/Application Layer

No.	Classes	Methods
1.	AuthorDataAccessLayer	GetAllAuthors(), AddAuthor(), UpdateAuthor(), GetAuthorData(), DeleteAuthor().
2.	BookDataAccessLayer	GetAllBooks(), AddBook(), UpdateBook(), GetBookData(), DeleteBook().
3.	Book_AuthorDataAccessLayer	GetAllBook_Authors(), AddBook_Author(), UpdateBook_Author(), GetBook_1AuthorData(), GetBook_AuthorData(), DeleteBook_Author(), DeleteBook_AllAuthors().

4.	CategoryDataAccessLayer	GetAllCategories(), AddCategory(), UpdateCategory(), GetBook_CategoryData(), GetCategoryData(), GetBooks_Category().
5.	FilterDataAccessLayer	GetFilterBooks().
6.	FineDataAccessLayer	GetAllFines(), AddFine(), UpdateFine(), GetFineData(), DeleteFine(), auto_update().
7.	LeaseDataAccessLayer	GetAllLeases(), AddLease(), UpdateLease(), GetLeaseData(), DeleteLease(), Take_new_SessionID(), Take_num_leases_for_Book().
8.	PaymentDataAccessLayer	GetAllPayments(), AddPayment(), UpdatePayment(), GetPaymentData(), DeletePayment().
9.	ReviewDataAccessLayer	GetAllReviews(), AddReview(), UpdateReview(), GetReviewData(), GetReviewDataForBook(), DeleteReview().
10.	UserDataAccessLayer	GetAllUsers(), AddUser(), UpdateUser(), GetUserData(), DeleteUser().
11.	Auto_update_sys	run().
12.	Take_add_img_Book	get_add().
13.	Take_Description_Book	get_add().

14.	ViewBook	Initiate(), show_Filter(), show_all_books(), res_Filter().
15.	ViewLease	ViewLease().
16.	ViewLeases_User	ViewLeases_User().
17.	ViewBook_Author	ViewBook_Author().

Table 3: *Business Logic/Application Layer*

d. UI Layer

No.	View_Page	Methods and Tags
1	_Layout	playBgMusic (), pauseBgMusic (), <iframe src="/audio/silence.mp3" allow="autoplay" id="audio"></iframe>, <audio id="bg_music" src="/audio/silence.mp3" autoplay loop></audio> <form asp-action="Searchtest"> @using ("Searchtest", "Home", FormMethod.Post) {} </form>, <form asp-action="Login"> @using (Html.BeginForm("Login", "Home", FormMethod.Post)) {} </form>, <a asp-action="Register></a>

2	_Layout1	playBgMusic (), pauseBgMusic (),<iframe src="/audio/silence.mp3" allow="autoplay" id="audio"></iframe>, <audio id="bg_music" src="/audio/silence.mp3" autoplay loop></audio>, <form asp-action=“Searchtest”>@using ("Searchtest", "Home", FormMethod.Post) {} </form>,<a asp-action="LeaseIndex"></a>, <a asp-action="UserDetails"></a>, <a asp-action="Logout"></a>, <a asp-controller="Admin" asp-action="Check"></a>
3	_Layout1_cus	playBgMusic (), pauseBgMusic (), <a asp-action="LeaseIndex"></a>, <a asp-action="UserDetails"></a>, <a asp-action="Logout"></a>

4	_Layout2	<a asp-action="BookIndex"></a>, <a asp-action="AuthorIndex"></a>, <a asp-action="Book_AuthorIndex"></a>, <a asp-action="CategoryIndex"> </a>, <a asp-action="FineIndex"> </a>, <a asp-action="LeaseIndex"> </a>, <a asp-action="PaymentIndex"> </a>, <a asp-action="ReviewIndex"> </a>, <a asp-action="UserIndex"> </a>, <a asp-action="Track_Renting_Books"> </a>, <a asp-action="BookIndex" asp-controller="Home"> </a>
5	BookIndex	BookIndex (), <a asp-action="BookDetails" asp-route-id="@item.Book.ID" ></a>
6	BookDetails	BookDetails (), <a asp-action="BookIndex"></a>, <a asp-action="LeaseCreate" asp-route-book_ID="@Model.Book.ID"></a>
7	Filter	Filter (), $(“”).select2(), <select id= “select_box_author”></select>, <select id= “select_box_pubyear”></select>, <select id= “select_box_cat”></select>, btn.addEventListener(‘click’, event()), AddToFilterTest (), myresult(), AjaxFailed ()

8	LeaseIndex	LeaseIndex (), <a asp-action="LeaseDetails" asp-route-id="@item.SessionID"></a>
9	LeaseCreate	LeaseCreate (), <input type="submit" value="Creating Lease" class="btn"/>, <a asp-action="LeaseIndex"> </a>
10	LeaseDetails	LeaseDetails (), <a asp-action="PaymentCreate" asp-route-id_lease="@Model.SessionID> </a>, <a asp-action="LeaseIndex"></a>, <a asp-action="ReviewCreate" asp-route-id="@Model.Book_ID"> </a>

11	Login	Login (), <a asp-action="BookDetails" asp-route-id="@item.Book.ID" ></a>
12	Register	Register (), <button type = “submit”></button>
13	ReviewCreate	ReviewCreate (), <input type="submit" value="Creating Review" class="btn"/>
14	Search	Search (), <a asp-action="BookDetails" asp-route-id="@item.Book.ID" ></a>
15	Searchtest	Searchtest (), <a asp-action="BookDetails" asp-route-id="@item.Book.ID" ></a>

16	AuthorCreate	AuthorCreate (), <a asp-action="AuthorIndex"></a>, <input type="submit" value="Save"/>
17	AuthorDelete	AuthorDelete (), <input type="submit" value="Author Delete" class="btn" />, <a asp-action="AuthorIndex"></a>
18	AuthorDetails	AuthorDetails (), <a asp-action="AuthorEdit" asp-route-id="@Model.Author_ID"></a>
19	AuthorEdit	AuthorEdit (), <a asp-action="AuthorIndex"></a>, <input type="submit" value="Save" class="btn" />

20	AuthorIndex	AuthorIndex (), <a asp-action="AuthorEdit" asp-route-id="@item.Author_ID"></a>, <a asp-action="AuthorDetails" asp-route-id="@item.Author_ID"></a>, <a asp-action="AuthorDelete" asp-route-id="@item.Author_ID"></a>, <a asp-action="AuthorCreate"></a>
21	Book_AuthorCreate	Book_AuthorCreate (), <input type="submit" value="Book_Author Create" class="btn" />, <a asp-action="Book_AuthorIndex"></a>
23	Book_AuthorDelete	Book_AuthorDelete (), <input type="submit" value="Delete" class="btn" />, <a asp-action="Book_AuthorIndex></a>
23	Book_AuthorDetails	Book_AuthorDetails (), <a asp-action="Book_AuthorIndex></a>, <a asp-action="Book_AuthorDelete" asp-route-id="@Model.Book.ID" asp-route-id2="@author.Author_ID"></a>

24	Book_AuthorIndex	Book_AuthorIndex (), <a asp-action="Book_AuthorCreate"> </a>, <a asp-action="Book_AuthorDetails" asp-route-id="@item.Book_Authors.BookID"></a>
25	BookCreate	BookCreate (), <a asp-action="BookIndex"></a>, <input type="submit" value="BookCreate" class="btn btn-default"/>
26	BookDelete	BookDelete (), <input type="submit" value="Book Delete" class="btn" />, <a asp-action="BookIndex"></a>
27	BookDetails	BookDetails (), <a asp-action="BookEdit" asp-route-id="@Model.ID" class ="navigate-link"></a>, <a asp-action="BookIndex"></a>
28	BookEdit	BookEdit (), <input type="submit" value="Save" class="btn" />, <a asp-action="BookIndex"></a>

29	BookIndex	BookIndex (), <a asp-action="BookEdit" asp-route-id="@item.ID"></a>, <a asp-action="BookDetails" asp-route-id="@item.ID"></a>, <a asp-action="BookDelete" asp-route-id="@item.ID"></a>, <a asp-action="BookCreate"></a>
30	FineCreate	FineCreate (), <input type="submit" value="Fine Create" class="btn btn-default" />, <a asp-action="FineIndex"></a>
31	FineDelete	FineDelete (), <input type="submit" value="Fine Delete" class="btn" />, <a asp-action="FineIndex"></a>
32	FineDetails	FineDetails (), <a asp-action="FineEdit" asp-route-id="@Model.SessionID"></a>, <a asp-action="FineIndex"> </a>

33	FineEdit	FineEdit (), <input type="submit" value="Save" class="btn" />, <a asp-action="FineIndex"></a>
34	FineIndex	FineIndex (), <a asp-action="FineEdit" asp-route-id="@item.SessionID"></a>, <a asp-action="FineDetails" asp-route-id="@item.SessionID"></a>, <a asp-action="FineDelete" asp-route-id="@item.SessionID"></a>, <a asp-action="FineCreate"></a>
35	LeaseCreate	LeaseCreate (), <input type="submit" value="Create Lease" class="btn" />, <a asp-action="LeaseIndex"></a>
36	LeaseDelete	LeaseDelete (), <input type="submit" value="Lease Delete" class="btn" />, <a asp-action="LeaseIndex"></a>

37	LeaseDetails	LeaseDetails (), <a asp-action="LeaseEdit" asp-route-id="@Model.SessionID"> </a>, <a asp-action="LeaseIndex"></a>
38	LeaseEdit	LeaseEdit (), <input type="submit" value="Save" class="btn" />, <a asp-action="LeaseIndex"></a>
39	LeaseIndex	LeaseIndex (), <a asp-action="LeaseEdit" asp-route-id="@item.SessionID"></a>, <a asp-action="LeaseDetails" asp-route-id="@item.SessionID"></a>, <a asp-action="LeaseDelete" asp-route-id="@item.SessionID"></a>, <a asp-action="LeaseCreate"></a>
40	PaymentCreate	PaymentCreate (), <input type="submit" value="Issue an invoice" class="btn" />, <a asp-action="PaymentIndex”></a>
41	PaymentDelete	PaymentDelete (), <input type="submit" value="Delete" class="btn" />, <a asp-action="PaymentIndex"></a>

42	PaymentDetails	PaymentDetails (), <a asp-action="PaymentEdit" asp-route-id="@Model.Payment_ID"></a>, <a asp-action="PaymentIndex"></a>
43	PaymentEdit	PaymentEdit (), <input type="submit" value="Save" class="btn" />, <a asp-action="PaymentIndex"></a>
44	PaymentIndex	PaymentIndex (), <a asp-action="PaymentEdit" asp-route-id="@item.Payment_ID"></a>, <a asp-action="PaymentDetails" asp-route-id="@item.Payment_ID"></a>, <a asp-action="PaymentDelete" asp-route-id="@item.Payment_ID"></a>, <a asp-action="PaymentCreate"></a>
45	ReviewCreate	ReviewCreate (), <input type="submit" value="ReviewCreate" class="btn" />, <a asp-action="ReviewIndex"></a>
46	ReviewDelete	ReviewDelete (), <input type="submit" value="ReviewDelete" class="btn" />, <a asp-action="ReviewIndex"></a>

47	ReviewDetails	ReviewDetails (), <a asp-action="ReviewEdit" asp-route-id="@Model.ISSNum" asp-route-id2="@Model.IDBook"></a>, <a asp-action="ReviewIndex”> </a>
48	ReviewEdit	ReviewEdit (), <input type="submit" value="Save" class="btn btn-default" />, <a asp-action="ReviewIndex”></a>
49	ReviewIndex	ReviewIndex (), <a asp-action="ReviewEdit" asp-route-id="@item.ISSNum" asp-route-id2="@item.IDBook"></a>, <a asp-action="ReviewDetails" asp-route-id="@item.ISSNum" asp-route-id2="@item.IDBook"></a>, <a asp-action="ReviewDelete" asp-route-id="@item.ISSNum" asp-route-id2="@item.IDBook"></a>, <a asp-action="ReviewCreate"></a>
50	UserCreate	UserCreate (), <input type="submit" value="Create" class="btn" />, <a asp-action="UserIndex"></a>

51	UserDelete	UserDelete (), <input type="submit" value="User Delete" class="btn" />, <a asp-action="UserIndex"></a>
52	UserDetails	UserDetails (), <a asp-action="UserEdit" asp-route-id="@Model.ISSN"></a>, <a asp-action="UserIndex"></a>
53	UserEdit	UserEdit (), <input type="submit" value="Save" class="btn" />, <a asp-action="UserIndex"></a>
54	UserIndex	UserIndex (),<a asp-action="UserEdit" asp-route-id="@item.ISSN"></a>, <a asp-action="UserDetails" asp-route-id="@item.ISSN"></a>, <a asp-action="UserDelete" asp-route-id="@item.ISSN"></a> <a asp-action="UserCreate"></a>

55	Track_Renting_Books	Track_Renting_Books (), filterTitle (),<a asp-action="Track_Renting_Book" asp-route-id_book="@item.Book.ID"> </a>
56	Track_Renting_Book	Track_Renting_Book (), filterISSN () ,<a asp-action="LeaseDetails" asp-route-id="@item.SessionID"></a>, <a asp-action="LeaseEdit" asp-route-id="@item.SessionID"></a>, <a asp-action="Track_Renting_Books"></a>
57	Search	Search (), <a asp-action="BookEdit" asp-route-id="@item.Book.ID"></a>, <a asp-action="BookDetails" asp-route-id="@item.Book.ID"></a>, <a asp-action="BookDelete" asp-route-id="@item.Book.ID"></a>, <a asp-action="BookCreate"></a>

Table 4: *GUI Layer*

e. Data Classes for AdminController

No.	Classes	Methods

1.	Author	AuthorIndex(), AuthorCreate(), AuthorEdit(), AuthorDetails(), AuthorDelete(), AuthorDeleteConfirmed().
2.	Book	BookIndex(), BookCreate(), BookEdit(), BookDetails(), BookDelete(), BookDeleteConfirmed().
3.	Book_Author	Book_AuthorIndex(), Book_AuthorCreate(), Book_AuthorEdit(), Book_AuthorDetails(), Book_AuthorDelete(), Book_AuthorDeleteConfirmed().
4.	Category	CategoryIndex(), CategoryCreate(), Category_BookDetails().
5.	Filter	AuthorIndex(), AuthorCreate(), AuthorEdit(), AuthorDetails(), AuthorDelete(), AuthorDeleteConfirmed().
6.	Fine	FineIndex(), FineCreate(), FineEdit(), FineDetails(), FineDelete(), FineDeleteConfirmed().
7.	Lease	LeaseIndex(), LeaseCreate(), LeaseEdit(), LeaseDetails(), LeaseDelete(), LeaseDeleteConfirmed().

8.	Payment	PaymentIndex(), PaymentCreate(), PaymentEdit(), PaymentDetails(), PaymentDelete(), PaymentDeleteConfirmed().
9.	Review	ReviewIndex(), ReviewCreate(), ReviewEdit(), ReviewDetails(), ReviewDelete(), ReviewDeleteConfirmed().
10.	User	UserIndex(), UserCreate(), UserEdit(), UserDetails(), UserDelete(), UserDeleteConfirmed().
11.	Send mail	Send_Mail_Manually().
12.	Track book	Track_Renting_Books(), Track_Renting_Book().
13.	Security	Check().

Table 5: *Data Classes for AdminController*

f. Data Classes for HomeController

No.	**Classes**	**Methods**
1.	Search	Searchtest(), Search(), SetViewBag().
2.	Filter	Filter().
3.	Author	AuthorIndex().

4.	Book	BookIndex(), BookDetails().
5.	User	Register(), Login(), Logout(), UserDetails(), UserEdit().
6.	Lease	LeaseIndex(), LeaseCreate(), LeaseDetails().
7.	Payment	PaymentCreate(), ProcessPaymentCreate().
8.	Review	ReviewCreate().

Table 6: *Data Classes for HomeController*

- **Some sequence diagrams presenting the HomeController classes:**

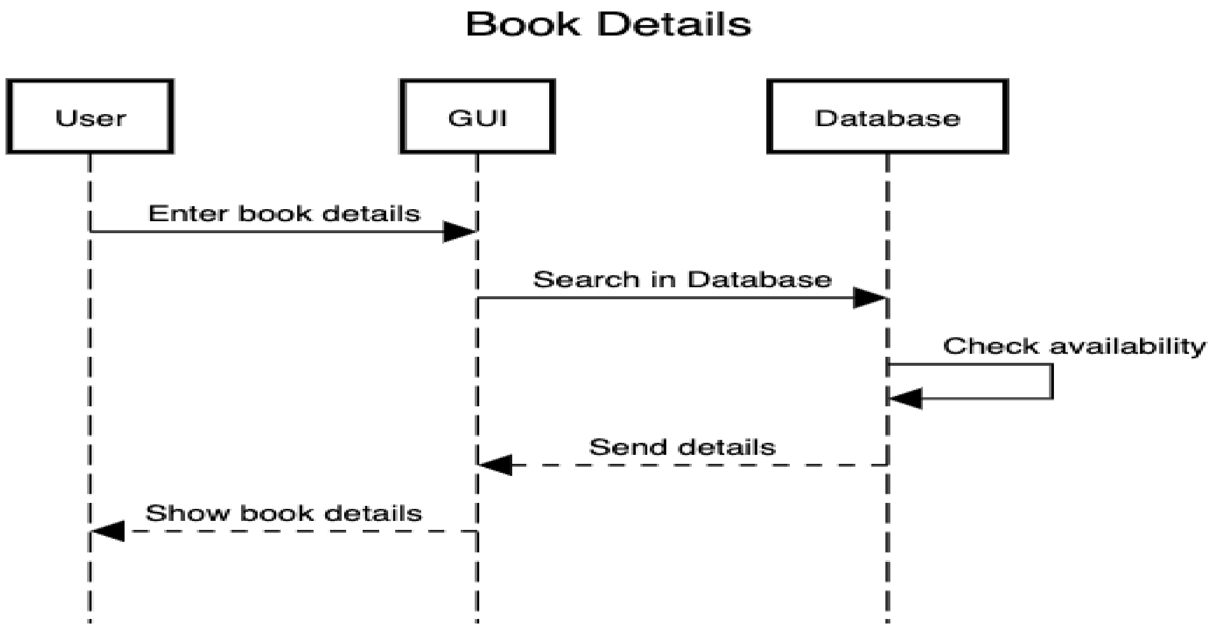


Figure 3: *Book Details Sequence Diagram*

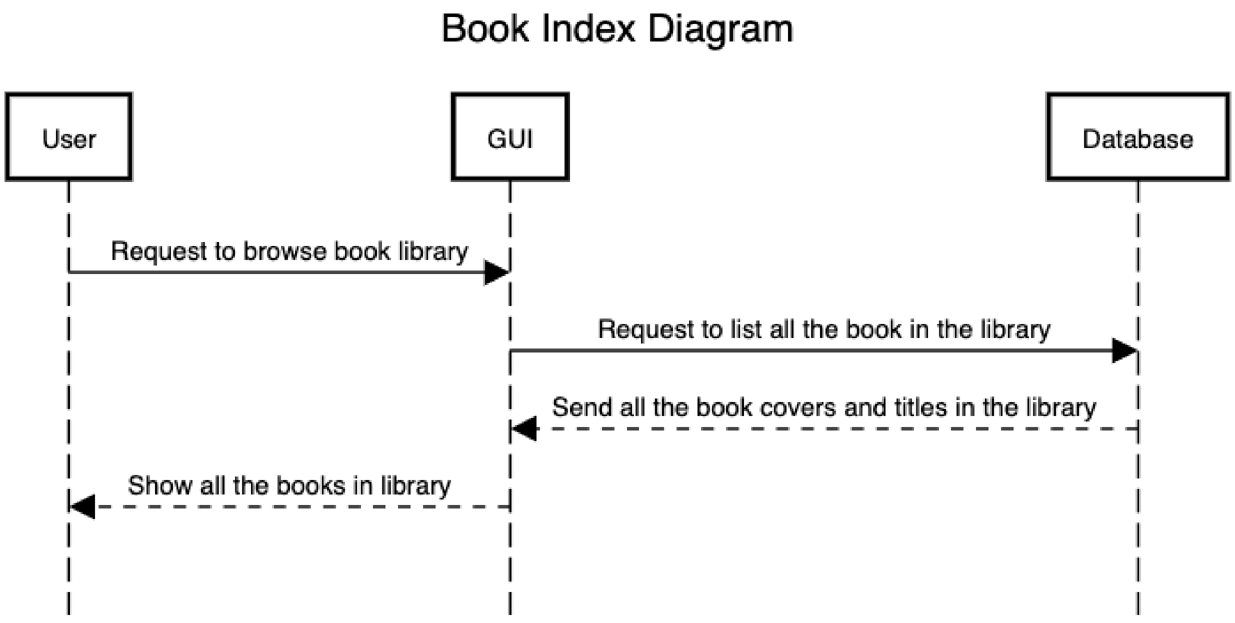

Book Index Diagram
User
GUI
Database
Request to browse book library
Request to list all the book in the library
Send all the book covers and titles in the library
Show all the books in library

Figure 4: *Book Index Sequence Diagram*

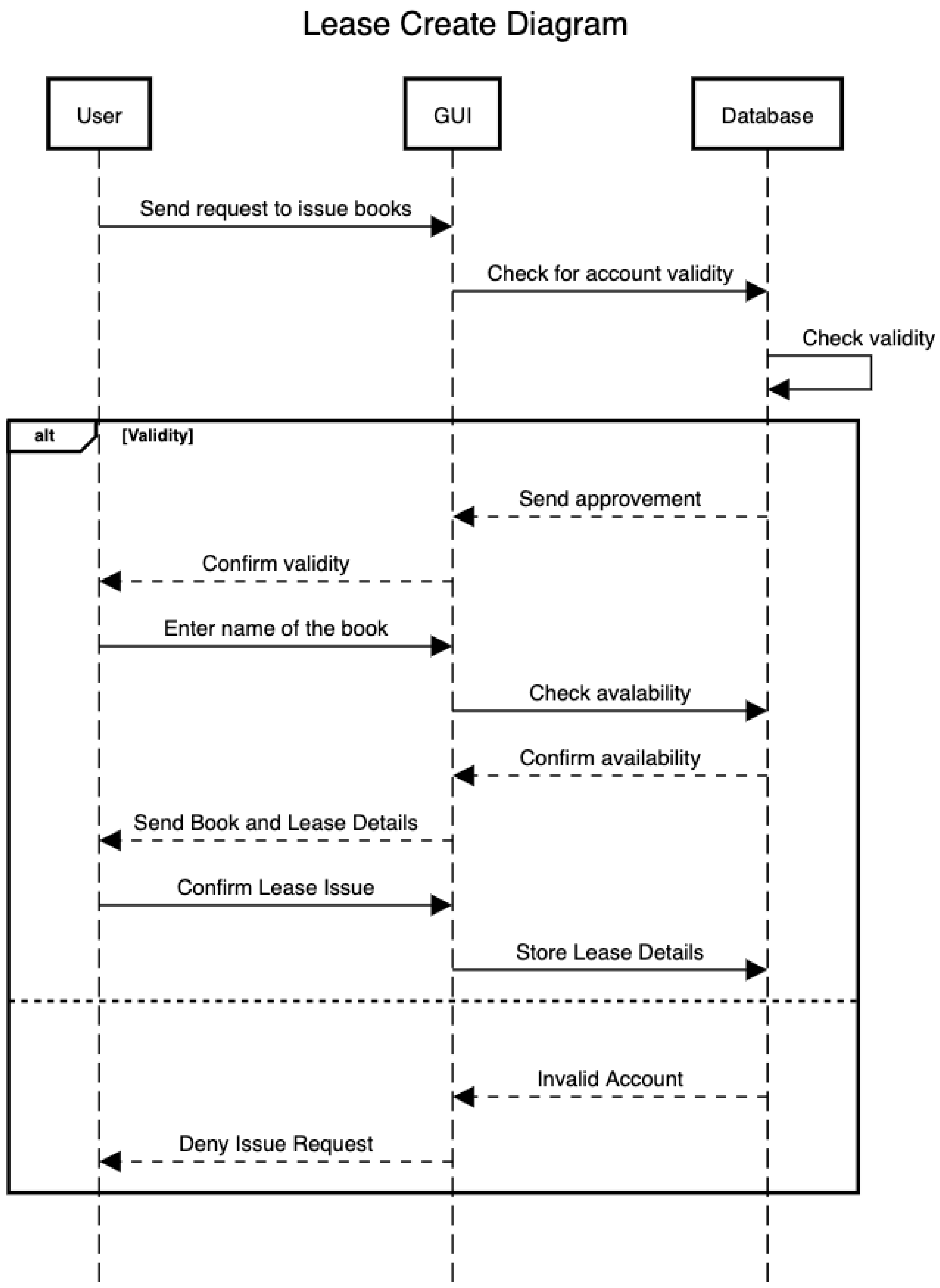


Figure 5: *Lease Create Sequence Diagram*

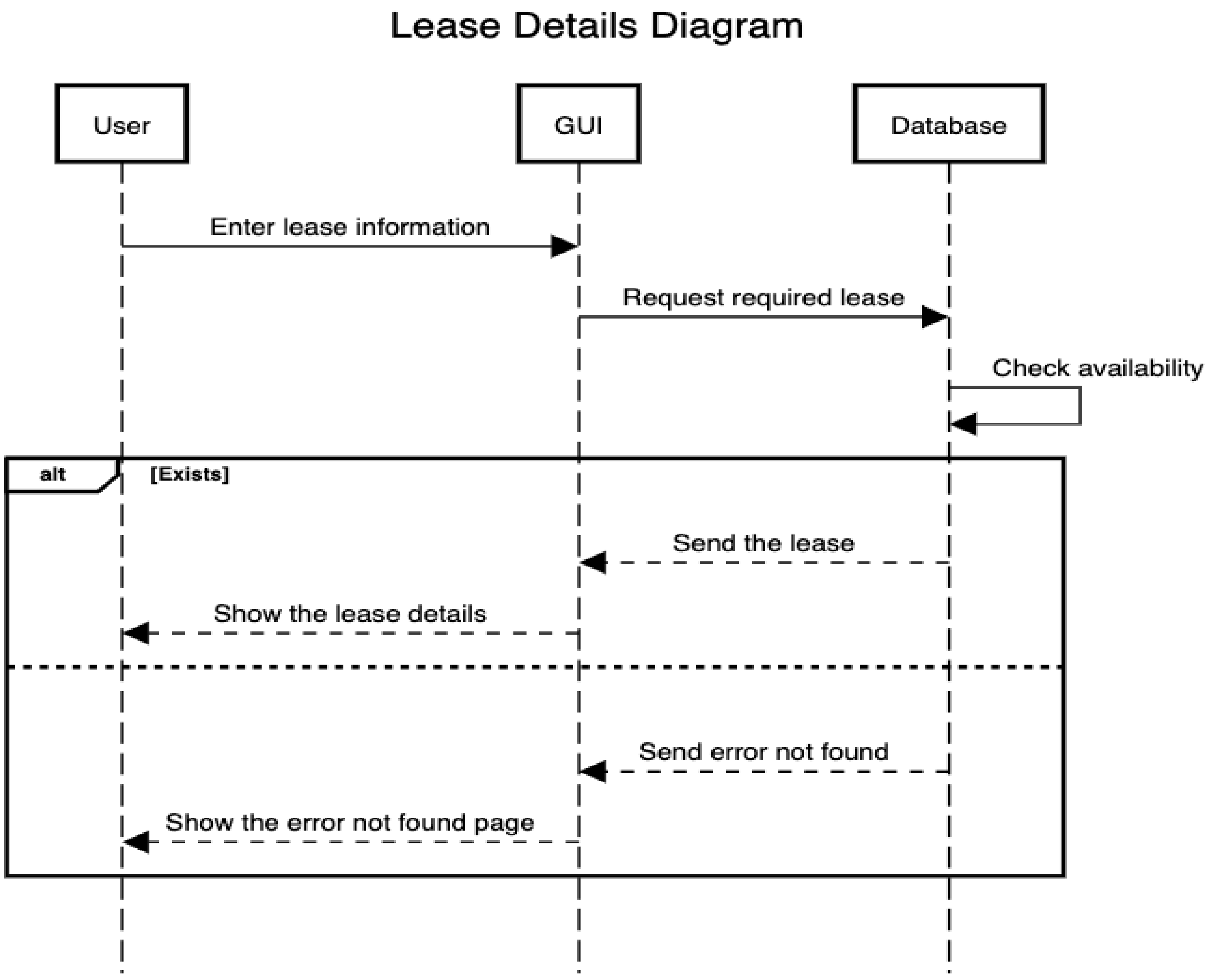


Figure 6: *Lease Details Sequence Diagram*

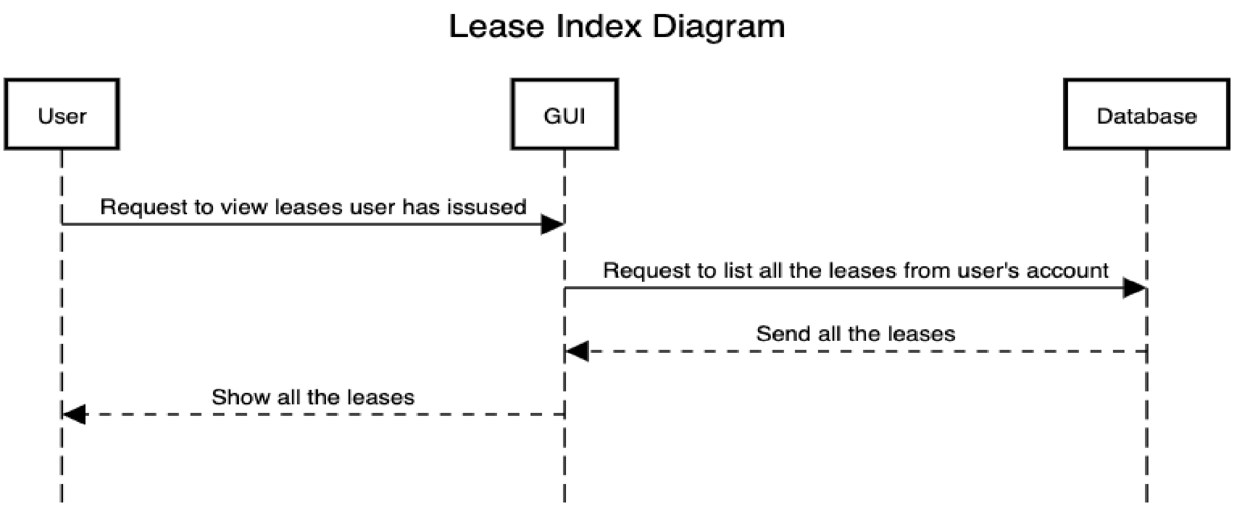

Lease Index Diagram
User
GUI
Database
Request to view leases user has issused
Request to list all the leases from user's account
Send all the leases
Show all the leases

Figure 7: *Lease Index Sequence Diagram*

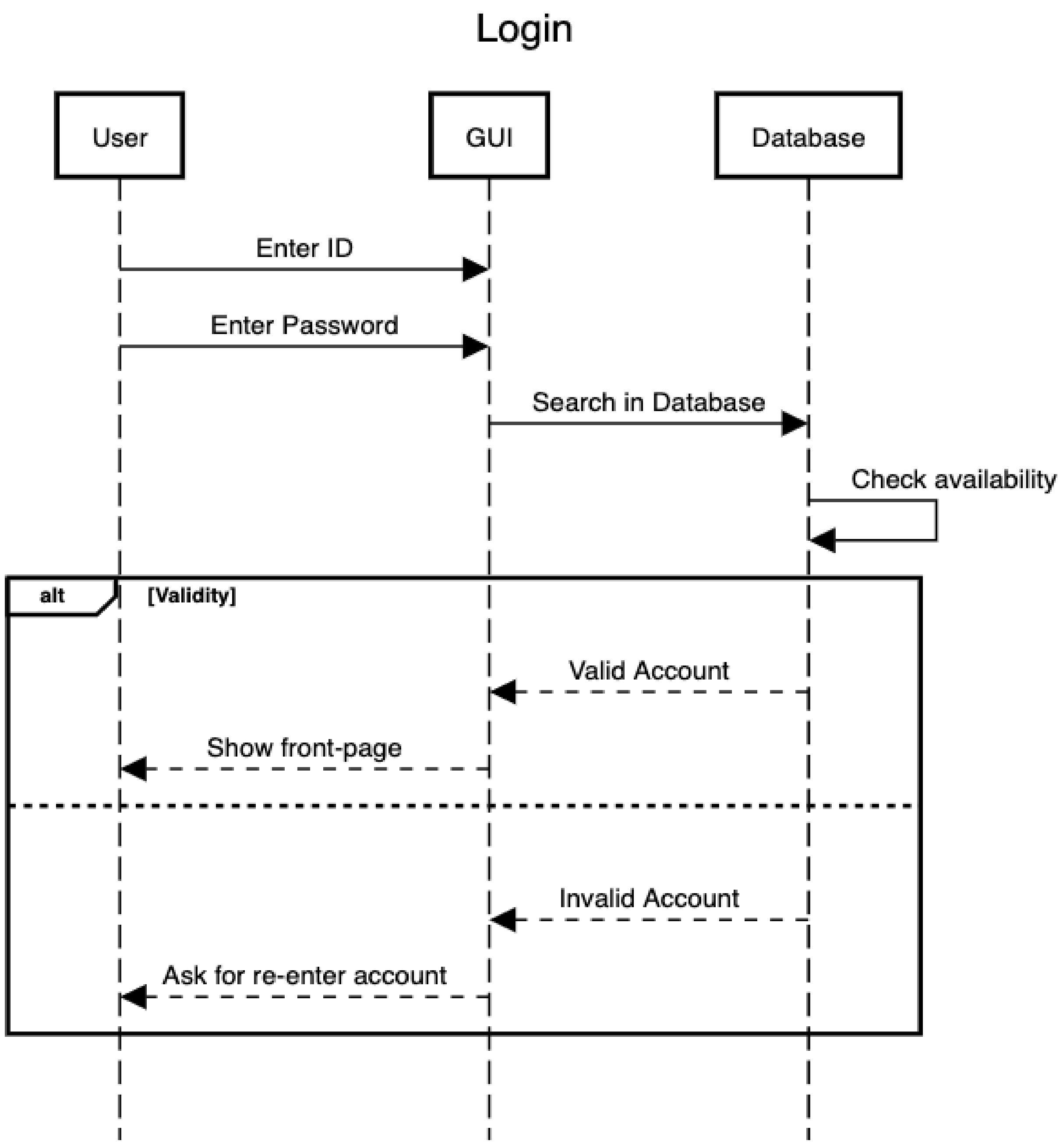


Figure 8: *Login Sequence Diagram*

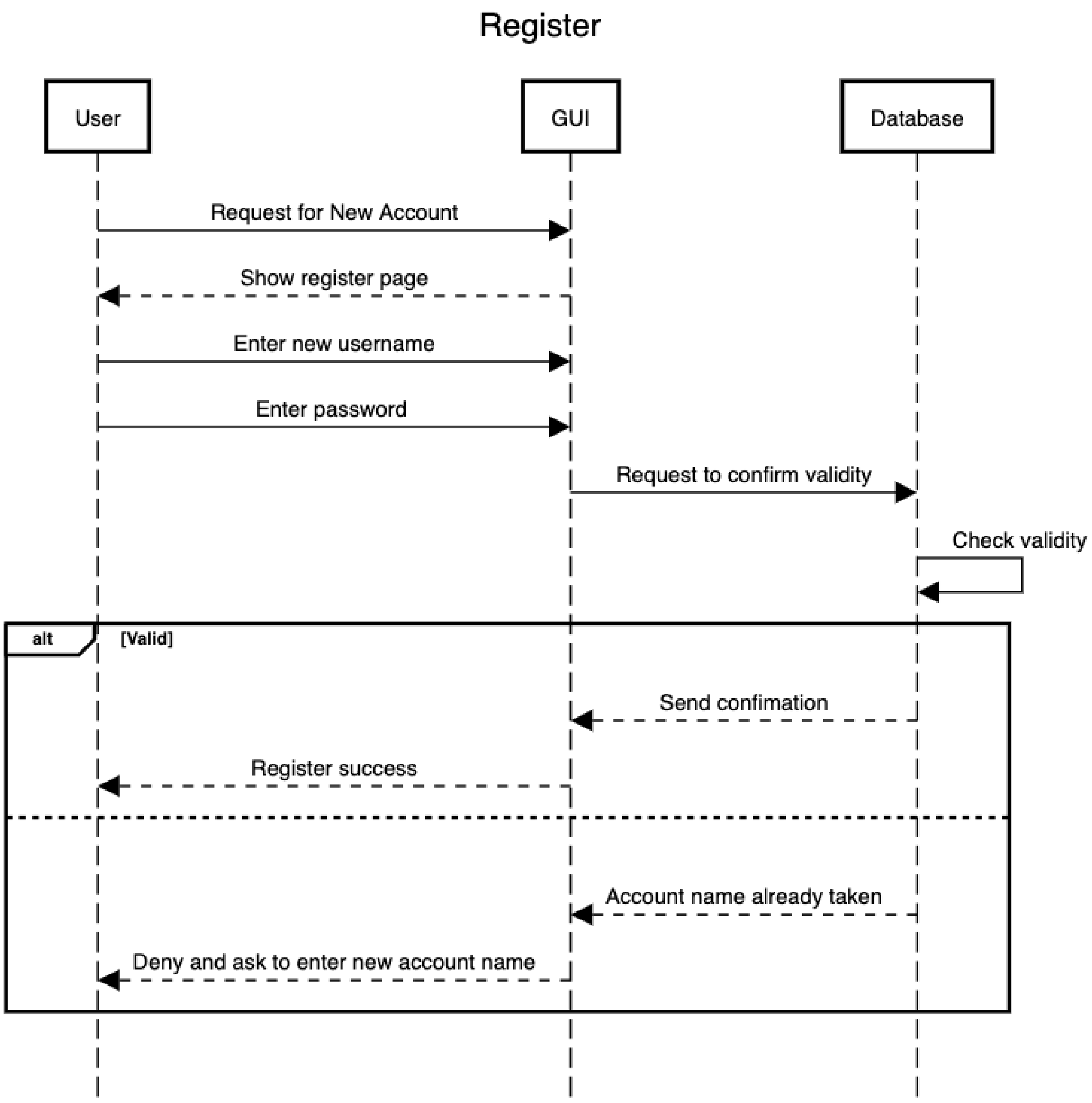


Figure 9: *Register Sequence Diagram*

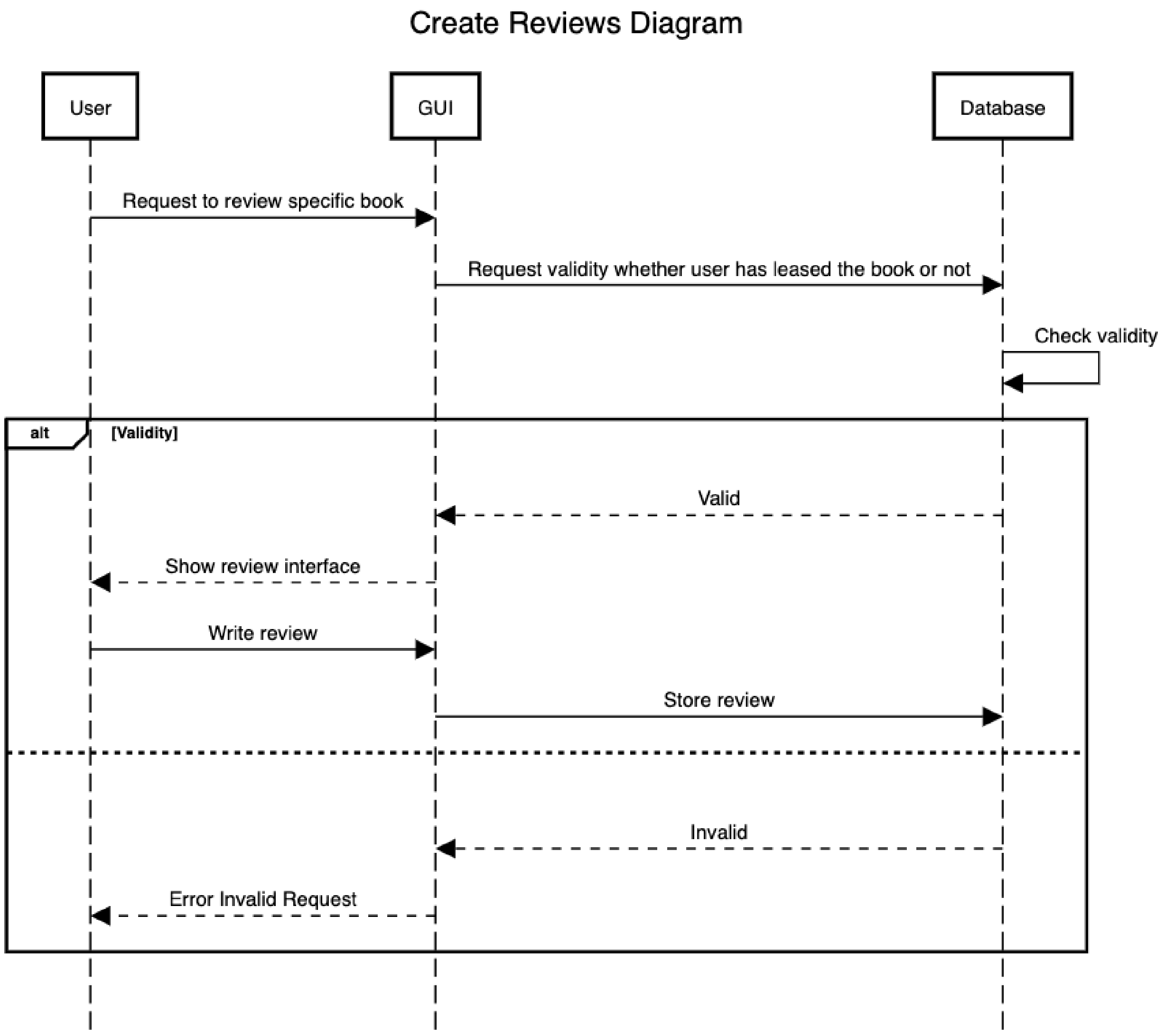


Figure 10: *Review Create Sequence Diagram*

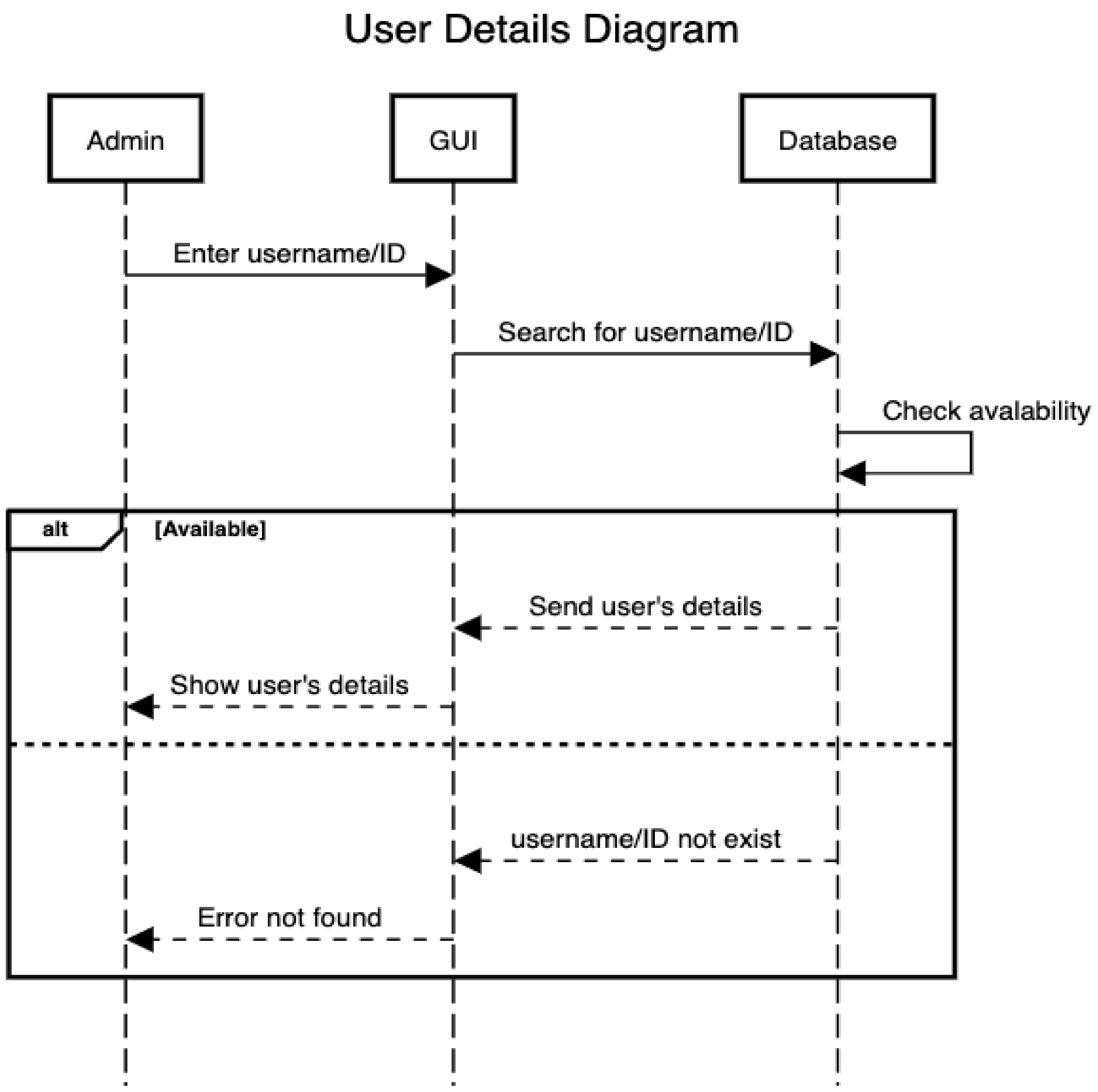


Figure 11: *User Details Sequence Diagram*

III. CLASS DESIGN

1. Details of Database Classes

No.	Classes and description	Instance variable	Methods and functionalities

1.	AuthorDataAccessLayer: it is used to communicate with the DB and send requests to get, update or delete information related to the Author table.	Private:
		
		String connection;
		
		IEnumerable<Author > lstauthor;
		
		Author author
			Public IEnumerable<Author> GetAllAuthors(): will return an array list of Author objects obtained from the DB.
			
			Public void AddAuthor(Author author): will add new Author object to the DB.
			
			Public void UpdateAuthor(Author author): will update Author object existed in the DB.
			
			Public Author GetAuthorData(int? id): will return an Author object obtained from the DB.
			
			Public void DeleteAuthor(int? id): will process to delete data of Author object in the DB.

2.	BookDataAccessLayer: it is used to communicate with the DB and send requests to get, update or delete information related to the Book table.	Private:
		
		String connection;
		
		IEnumerable<Book> lstbook;
		
		Book book
			Public IEnumerable<Book> GetAllBooks(): will return an array list of Book objects obtained from the DB.
			
			Public void AddBook(Book book): will add new Book object to the DB.
			
			Public void UpdateBook(Book book): will update Book object existed in the DB.
			
			Public Book GetBookData(int? id): will return an Book object obtained from the DB.
			
			Public void DeleteBook(int? id): will process to delete data of Book object in the DB.

3.	Book_AuthorDataAccessLayer : it is used to communicate with the DB and send requests to get, update or delete information related to the Book_Author table.	Private:
		
		String connection;
		
		IEnumerable<Book_ Author> lstbauthor;
		
		Book_Author bauthor
			Public IEnumerable<Book_Author> GetAllBook_Authors(): will return an array list of Book_Author objects obtained from the DB.
			
			Public void AddBook_Author(Book_Author bauthor): will add new Book_Author object to the DB.
			
			Public void UpdateBook_Author(Book_Author bauthor): will update Book_Author object existed in the DB.
			
			Public ViewBook_Author GetBook_1AuthorData(int? id, int? id2): will return an Book_Author object obtained from the DB have condition.
			
			Public IEnumerable<Book_Author> GetBook_AuthorData(int? id): will return an Book_Author object obtained from the DB.
			
			Public void DeleteBook_Author(int? id): will process to delete data of Book_Author object in the DB.

4.	CategoryDataAccessLayer: it is used to communicate with the DB and send requests to get, update or delete information related to the Category table.	Private:
		
		String connection;
		
		IEnumerable<Category> lstcategories;
		
		Category category
			Public IEnumerable<Category> GetAllCategories(): will return an array list of Category objects obtained from the DB.
			
			Public void AddCategory(Category category): will add new Category object to the DB.
			
			Public void UpdateCategory(Category category): will update Category object existed in the DB.
			
			Public GetBook_CategoryData(int? id): will return a Category object obtained from the DB which linked with a Book object.
			
			Public Category GetCategoryData(int? id): will return a Category object obtained from the DB.
			
			Public GetBooks_CategoryData(int? idcate): will return a Category object obtained from the DB which linked with Book objects.

5.	FilterDataAccessLayer: it is used to communicate with the DB and send requests to get information related to the ViewBook table.	Private: String connection; IEnumerable<ViewBook> lstViewBook	Public IEnumerable<ViewBook> GetFilterBooks(string category, string author, int publication): will return an array list of ViewBook objects obtained from the DB which following condition's filter.

6.	FineDataAccessLayer: it is used to communicate with the DB and send requests to get, update or delete information related to the Fine table.	Private:
		
		String connection;
		
		IEnumerable<Fine> lstfine;
		
		Fine fine
			Public IEnumerable<Fine> GetAllFines(): will return an array list of Fine objects obtained from the DB.
			
			Public void AddFine(Fine fine): will add new Fine object to the DB.
			
			Public void UpdateFine(Fine fine): will update Fine object existed in the DB.
			
			Public Fine GetFineData(int? id): will return a Fine object obtained from the DB.
			
			Public void DeleteFine(int? id): will process to delete data of Fine object in the DB.
			
			Public void auto_update(): will automatically process data in the DB to ensure that data is up to date.

7.	LeaseDataAccessLayer: it is used to communicate with the DB and send requests to get, update or delete information related to the Lease table.	Private:
		
		String connection;
		
		IEnumerable<Lease> lstlease;
		
		Lease lease
			Public IEnumerable<Lease> GetAllLeases(): will return an array list of Lease objects obtained from the DB.
			
			Public void AddLease(Lease lease): will add new Lease object to the DB.
			
			Public void UpdateLease(Lease lease): will update Lease object existed in the DB.
			
			Public Lease GetLeaseData(int? id): will return a Lease object obtained from the DB.
			
			Public void DeleteLease(int? id): will process to delete data of Lease object in the DB.
			
			Public int Take_new_SessionID(): will return a new id for Lease object (the same function with auto-increment property).
			
			Public int Take_num_leases_for_Book(int id_book): will return a number of Lease objects which is still validated linked with Book object parameter.

8.	PaymentDataAccessLayer: it is used to communicate with the DB and send requests to get, update or delete information related to the Payment table.	Private:
		
		String connection;
		
		IEnumerable<Payment> lstpayment;
		
		Payment payment
			Public IEnumerable<Payment> GetAllPayments(): will return an array list of Payment objects obtained from the DB.
			
			Public void AddPayment(Payment payment): will add new Payment object to the DB.
			
			Public void UpdatePayment(Payment payment): will update Payment object existed in the DB.
			
			Public Payment GetPaymentData(int? id): will return a Payment object obtained from the DB.
			
			Public void DeletePayment(int? id): will process to delete data of Payment object in the DB.

9.	ReviewDataAccessLayer: it is used to communicate with the DB and send requests to get, update or delete information related to the Review table.	Private:
		
		String connection;
		
		IEnumerable<Review> lstreview;
		
		Review review
			Public IEnumerable<Review> GetAllReviews(): will return an array list of Review objects obtained from the DB.
			
			Public void AddReview(Review review): will add new Review object to the DB.
			
			Public void UpdateReview(Review review): will update Fine object existed in the DB.
			
			Public Review GetReviewData(int? id): will return a Review object obtained from the DB.
			
			Public IEnumerable<Review> GetReviewDataForBook(int? id): will return a list of Review objects from the DB linked with the Book object parameter.
			
			Public void DeleteReview(int? id): will process to delete data of Review object in the DB.

10.	UserDataAccessLayer: it is used to communicate with the DB and send requests to get, update or delete information related to the User table.	Private:
		
		String connection;
		
		IEnumerable<User> lstuser;
		
		User user
			Public IEnumerable<User> GetAllUsers(): will return an array list of User objects obtained from the DB.
			
			Public void AddUser(User user): will add new User object to the DB.
			
			Public void UpdateUser(User user): will update User object existed in the DB.
			
			Public User GetUserData(int? id): will return an User object obtained from the DB.
			
			Public void DeleteUser(int? id): will process to delete data of User object in the DB.

11.	MySqlConnection: it is used to establish connection to the DB	Private: String connection	Public void Open(): will connect to DB. Public void Close(): will disconnect to DB. Public Datatype CommandType.StoredProcedure(): will return Datatype object. Public ExecuteNonQuery(): will process data in the DB and return null. Public ExcuteReader(): will process data from the DB and return a SqlDataReader object.

Table 7: *Details of Database Classes*

2. Details of Application Layer

No.	Classes and description	Instance variable	Methods and functionalities

1.	Auto_update_sys: it is used for update the database system.	Private:
		
		AuthorDataAccessLayer objauthor;
		
		BookDataAccessLayer objbook;
		
		Book_AuthorDataAccessLayer objbauthor;
		
		CategoryDataAccessLayer objcategory;
		
		LeaseDataAccessLayer objlease;
		
		PaymentDataAccessLayer objpayment;
		
		UserDataAccessLayer objuser;
		
		FineDataAccessLayer objfine;
		
		ReviewDataAccessLayer objreview
			Public void run(int i=0): will automatically update all data in the DB to ensure that the data is up to date as well as correct the inconsistent data. The parameter is optional if (i ==1) then it will initiate the function send email manually to the customer which lease is expired.

2.	Take_add_img_Book: it is used for take address the image Book cover.	Private: String[,] data;	Public string get_add(int id): will return a string address image of a Book object.

3.	Take_Description_Book: it is used for take description Book.	Private: String[,] data;	Public string get_add(int id): will return a string description of a Book object.

4.	ViewBook: it is used for demonstrate a Book object in details. Using for customers.	Public:
		
		Book? Book;
		
		List<Author>? Authors;
		
		Category? Category;
		
		List<Review>? Review;
		
		Int avgStar;
		
		String add_img_Book;
		
		String Description;
		
		Int available
			Public void Initiate(): will initiate with a parameter Book object, it contain all relevant data.
			
			Public Filter show_Filter(): return list data to filter.
			
			Public List<ViewBook> show_all_books(): will return a list of all Book objects contained in the DB.
			
			Public List<ViewBook> res_Filter(string? category, string? author, string? publication): will return a list Book objects of result search.

5.	ViewLease: it is used for shown the details Lease object linked with Book object.	Public: Lease? Lease; Book? Book	ViewLease(Lease lease): will initiate the class with Lease object parameter.
6.	ViewLeases_User	Public: Lease? Lease; User? User	ViewLease_User(Lease lease): will initiate the class with Lease object parameter.

7.	ViewBook_Author	Public:
		
		String? Book_Title;
		
		Book_Author? Book_Authors;
		
		String? Author_Name
			ViewBook_Author(): will initiate the class with Book_Author object.

Table 8: *Details of Application Layer*

3. Details of Data-Class for AdminController

No.	**Classes and description**	**Instance variable**	**Methods and functionalities**

1.	Author	Public:
		
		Auto_update_sys auto_Update_Sys;
		
		AuthorDataAccessLayer objauthor;
		
		BookDataAccessLayer objbook;
		
		Book_AuthorDataAccessLayer objbauthor;
		
		CategoryDataAccessLayer objcategory;
		
		LeaseDataAccessLayer objlease;
		
		PaymentDataAccessLayer objpayment;
		
		UserDataAccessLayer objuser;
		
		FineDataAccessLayer objfine;
		
		ReviewDataAccessLayer objreview
			Public IActionResult AuthorIndex(): will return a View contains list of Author objects.
			
			Public IActionResult AuthorCreate(): will return a View contains form to fill about the Author object details.
			
			Public IActionResult AuthorCreate([Bind] Author author): will process a [Bind] data of Author object and adding to the DB if ModelState is valid, then return to AuthorIndex function.
			
			Public IActionResult AuthorEdit(int? id): will return a View contains form to edit the data of this Author object parameter.
			
			Public IActionResult AuthorEdit(int id, [Bind] Author author): will check data, if it is consistent and ModelState is valid then it will use method update to the DB and return to AuthorIndex function.

			Public IActionResult AuthorDetails(int? id): will return a View of details of this Author object parameter. Public IActionResult AuthorDelete(int? id): will return a View of details of this Author object parameter to confirm this action delete. Public IActionResult AuthorDeleteConfirmed(int? id): will process the deleting progress and return to AuthorIndex function.

2.	Book		Public IActionResult BookIndex(): will return a View contains list of Book objects. Public IActionResult BookCreate(): will return a View contains form to fill about the Book object details. Public IActionResult BookCreate([Bind] Book book): will process a [Bind] data of Book object and adding to the DB if ModelState is valid, then return to BookIndex function. Public IActionResult BookEdit(int? id): will return a View contains form to edit the data of this Book object parameter. Public IActionResult BookEdit(int id, [Bind] Book book): will check data, if it is consistent and ModelState is valid then it will use method update to the DB and return to BookIndex function.

			Public IActionResult BookDetails(int? id): will return a View of details of this Book object parameter. Public IActionResult BookDelete(int? id): will return a View of details of this Book object parameter to confirm this action delete. Public IActionResult BookDeleteConfirmed(int? id): will process the deleting progress and return to BookIndex function.

3.	Book_Author		Public IActionResult Book_AuthorIndex(): will return a View contains list of Book_Author objects. Public IActionResult Book_AuthorCreate(): will return a View contains form to fill about the Book_Author object details. Public IActionResult Book_AuthorCreate([Bind] Book_Author bauthor): will process a [Bind] data of Book_Author object and adding to the DB if ModelState is valid, then return to Book_AuthorIndex function. Public IActionResult Book_AuthorEdit(int? id): will return a View contains form to edit the data of this Book_Author object parameter. Public IActionResult Book_AuthorEdit(int id, [Bind] Book_Author bauthor): will check data, if it is consistent and ModelState is valid then it

			will use method update to the DB and return to Book_AuthorIndex function. Public IActionResult Book_AuthorDetails(int? id): will return a View of details of this Book_Author object parameter. Public IActionResult Book_AuthorDelete(int? id, int? id2): will return a View of details of this Book_Author object parameter to confirm this action delete. Public IActionResult Book_AuthorDeleteConfirmed(int? id, int? id2): will process the deleting progress and return to Book_AuthorIndex function.

4.	Category		Public IActionResult CategoryIndex(): will return a View contains list of Category objects. Public IActionResult CategoryCreate(): will return a View containing a form to fill about the Category object details. Public IActionResult CategoryCreate([Bind] Category category): will process a [Bind] data of Category object and add to the DB if ModelState is valid, then return to CategoryIndex function. Public IActionResult Category_BookDetails(int? id): will return a View of details of the Book object following this Category object parameter.

5.	Filter		Public IActionResult Search(string search): will return View of list ViewBook objects which contains a string search in its Book_Title. Public IActionResult Search(string category, string author, string publication): will return View of listing ViewBook objects which is the result of function ViewBook.res_Filter().

6.	Fine		Public IActionResult FineIndex(): will return a View contains list of Fine objects. Public IActionResult FineCreate(): will return a View contains form to fill about the Fine object details. Public IActionResult FineCreate([Bind] Fine fine): will process a [Bind] data of Fine object and adding to the DB if ModelState is valid, then return to FineIndex function. Public IActionResult FineEdit(int? id): will return a View contains form to edit the data of this Fine object parameter. Public IActionResult FineEdit(int id, [Bind] Fine fine): will check data, if it is consistent and ModelState is valid then it will use method update to the DB and return to FineIndex function.

			Public IActionResult FineDetails(int? id): will return a View of details of this Fine object parameter. Public IActionResult FineDelete(int? id): will return a View of details of this Fine object parameter to confirm this action delete. Public IActionResult FineDeleteConfirmed(int? id): will process the deleting progress and return to FineIndex function.

7.	Lease		Public IActionResult LeaseIndex(): will return a View contains list of Lease objects. Public IActionResult LeaseCreate(): will return a View contains form to fill about the Lease object details. Public IActionResult LeaseCreate([Bind] Lease lease): will process a [Bind] data of Book object and adding to the DB if ModelState is valid, then return to LeaseIndex function. Public IActionResult LeaseEdit(int? id): will return a View contains form to edit the data of this Lease object parameter. Public IActionResult LeaseEdit(int id, [Bind] Lease lease): will check data, if it is consistent and ModelState is valid then it will use method update to the DB and return to LeaseIndex function.

			Public IActionResult LeaseDetails(int? id): will return a View of details of this Lease object parameter. Public IActionResult LeaseDelete(int? id): will return a View of details of this Lease object parameter to confirm this action delete. Public IActionResult LeaseDeleteConfirmed(int? id): will process the deleting progress and return to LeaseIndex function.

8.	Payment		Public IActionResult PaymentIndex(): will return a View contains list of Payment objects. Public IActionResult PaymentCreate(): will return a View contains form to fill about the Payment object details. Public IActionResult PaymentCreate([Bind] Payment payment): will process a [Bind] data of Payment object and adding to the DB if ModelState is valid, then return to PaymentIndex function. Public IActionResult PaymentEdit(int? id): will return a View contains form to edit the data of this Payment object parameter. Public IActionResult PaymentEdit(int id, [Bind] Payment payment): will check data, if it is consistent and ModelState is valid then it will use method update to the DB and return to PaymentIndex function.

			Public IActionResult PaymentDetails(int? id): will return a View of details of this Payment object parameter. Public IActionResult PaymentDelete(int? id): will return a View of details of this Payment object parameter to confirm this action delete. Public IActionResult PaymentDeleteConfirmed(int? id): will process the deleting progress and return to PaymentIndex function.

9.	Review		Public IActionResult ReviewIndex(): will return a View contains list of Review objects. Public IActionResult ReviewCreate(): will return a View contains form to fill about the Review object details. Public IActionResult ReviewCreate([Bind] Review review): will process a [Bind] data of Review object and adding to the DB if ModelState is valid, then return to ReviewIndex function. Public IActionResult ReviewEdit(int? id, int? id2): will return a View contains form to edit the data of this Review object linked with User object parameter. Public IActionResult ReviewEdit(int id, [Bind] Review review): will check data, if it is consistent and ModelState is valid then it will use method update to the DB

			and return to ReviewIndex function. Public IActionResult ReviewDetails(int? id, int? id2): will return a View of details of this Review object linked with User object parameter. Public IActionResult ReviewDelete(int? id, int? id2): will return a View of details of this Review object parameter and User object parameter2 to confirm this action delete. Public IActionResult ReviewDeleteConfirmed(int? id, int? id2): will process the deleting progress and return to ReviewIndex function.

10.	User		Public IActionResult UserIndex(): will return a View contains list of User objects. Public IActionResult UserCreate(): will return a View contains form to fill about the User object details. Public IActionResult UserCreate([Bind] User user): will process a [Bind] data of User object and adding to the DB if ModelState is valid, then return to UserIndex function. Public IActionResult UserEdit(int? id): will return a View contains form to edit the data of this User object parameter. Public IActionResult UserEdit(int id, [Bind] User user): will check data, if it is consistent and ModelState is valid then it will use method update to the DB and return to UserIndex function.

			Public IActionResult UserDetails(int? id): will return a View of details of this User object parameter. Public IActionResult UserDelete(int? id): will return a View of details of this User object parameter to confirm this action delete. Public IActionResult UserDeleteConfirmed(int? id): will process the deleting progress and return to UserIndex function.

11.	Send mail		Public IActionResult Send_Mail_Manually(): will process the auto_Update_Sys.run with the parameter (i=1) to send the announce mail to customers follow the expired leases list in the DB. Then return to Track_Renting_Books function.
12.	Track book		Public IActionResult Track_Renting_Books(): will return a View of list ViewBook objects and its status and remaining available. Public IActionResult Track_Renting_Book(int id_book): will return a View of list Lease objects linked with a certain Book object parameter.
13.	Security		Public RedirectToActionResult Check(): will identify admin by using session to compare the access-control in the DB as well as the cookies key by function ValidateAntiForgeryToken.

Table 9: *Details of Data-Class for AdminController*

4. Details of Data-Class for HomeController

No.	Classes and description	Instance variable	Methods and functionalities

1.	Search	Public:
		Auto_update_sys auto_Update_Sys;
		AuthorDataAccessLayer objauthor;
		BookDataAccessLayer objbook;
		Book_AuthorDataAccessLayer objbauthor;
		CategoryDataAccessLayer objcategory;
		LeaseDataAccessLayer objlease;
		PaymentDataAccessLayer objpayment;
		UserDataAccessLayer objuser;
		FineDataAccessLayer objfine;
		ReviewDataAccessLayer objreview;
		Public:
		private readonly ILogger<HomeController> _logger
			Public IActionResult Searchtest(string search): will return View of list ViewBook objects which contains a string search in its Book_Title.
			Public IActionResult Search(string category, string author, string publication): will return View of list ViewBook objects which is result of function ViewBook.res_Filter().

2.	Filter		Public IActionResult Filter(): will return View of filter function to select the input. Public ActionResult SetViewBag(string? s1, string? s2, string? s3): will process these data from the Filter page-view and using method PartialView to illustrate the list of ViewBook objects result in that page_view.

3.	Author		Public IActionResult AuthorIndex(): will return a View contains list of Author objects.
4.	Book		Public IActionResult BookIndex(): will return a View contains list of Book objects. Public IActionResult BookDetails(int? id): will return a View of details of this Book object parameter.

5.	User		Public ActionResult Register(): will clear all session data and return a View of form register to fill for customer. Public ActionResult Register(User _user): will check the submitted form by ModelState. If it valid to create a new account, then it will grant a key session (for insstance, the User.ISSN) by ValidateAntiForgeryToken function and a new account will be created for customer. After all, it redirect to the BookIndex function. Public ActionResult Login(string ISSN, string password): will check the submitted form by ModelState. If it valid to login, then it will grant a key session (for insstance, the User.ISSN) by ValidateAntiForgeryToken function and it will return the mainpage View with the Layout for customer or admin depending on the account's access-control.

			Public RedirectToActionResult Logout() : will clear all session key and return the mainpage View with the access of anonymous. Public IActionResult UserDetails(): will return a View of the user's account data. Public IActionResult UserEdit(): will return a View contains form to edit the data of this User object parameter by using session key to identify. Public IActionResult UserEdit([Bind] User user): will check data, if it is consistent and ModelState is valid then it will use method update to the DB and return to UserDetails function.

6.	Lease		Public IActionResult LeaseIndex(): will take the session key be the parameter and return the View of list Lease objects including the parameter with the status “active” for customer. Public IActionResult LeaseCreate(int book_ID): will return a View contains form to fill about the Lease object details linked with this Book object. Public IActionResult LeaseCreate([Bind] Lease lease): will process a [Bind] data of Book object and adding to the DB if ModelState is valid, then it automatically correct the imput data and return to LeaseDetails function, else will display a View of error. After all, it also run the auto_Update_Sys.run() function. Public IActionResult LeaseDetails(int? id): will return a View of details of this Lease object parameter.

7.	Payment		Public IActionResult PaymentCreate(int id_lease): will check "is this Lease object parameter already paid?" and return a View contains form about the Payment object details. Public IActionResult PaymentCreate([Bind] Payment payment): will process a [Bind] data of Payment object and adding to the DB by auto_Update_Sys.run() function, then return to LeaseDetails function.
8.	Review		Public IActionResult ReviewCreate(): will return a View contains form to fill about the Review object details. Public IActionResult ReviewCreate([Bind] Review review): will process a [Bind] data of Riview object and adding to the DB if ModelState is valid, then return to BookDetails function.

Table 10: *Details of Data-Class for HomeController*

5. Details of Behavior HTML Tags in UI Page

No.	Pages and Description	Behavior HTML Tags
1	Layout (Home): This template is used as the default Layout on first attempt entering the website.	<iframe src="/audio/silence.mp3" allow="autoplay" id="audio"></iframe>: This tag is used to initialize one second of silence. <audio id="bg_music" src="/audio/silence.mp3" autoplay loop></audio>: This tag is used to play the music. <form asp-action=" Searchtest"></form>: This tag redirects the form to the Searchtest page. <form asp-action=" Login"></form>: This tag redirects the form to the Login page. <a asp-action="Register></a>: This tag redirects to the Register page.

2	_Layout1(Home): This template is used as the default Layout when the User Admin log in.	<iframe src="/audio/silence.mp3" allow="autoplay" id="audio"></iframe>: This tag is used to initialize one second of silence. <audio id="bg_music" src="/audio/silence.mp3" autoplay loop></audio>: This tag is used to play the music. <form asp-action=" Searchtest"></form>: This tag redirects the form to the Searchtest page. <a asp-action="UserDetails"></a>: This tag redirects to the UserDetails page. <a asp-action="Logout"></a>: This tag redirects the form to the Searchtest page. <a asp-controller="Admin" asp-action="Check"></a>: This tag checks and logs into the Admin section if the user is an Admin.

3	_Layout1_cus (Home): This template is used as the default Layout when other default User/Customer log in	<iframe src="/audio/silence.mp3" allow="autoplay" id="audio"></iframe>: This tag is used to initialize one second of silence. <audio id="bg_music" src="/audio/silence.mp3" autoplay loop></audio>: This tag is used to play the music. <a asp-action="LeaseIndex"></a>: This tag redirects to the LeaseIndex page. <a asp-action="Logout"></a>: This tag logs out the user’s session. <a asp-action="UserDetails"></a>: This tag redirects to the UserDetails page.

4	_Layout_2 (Admin): This template is used as the default Layout for the Admin functions.	<a asp-action="BookIndex"></a>: This tag redirects to the BookIndex page in the Admin Controller. <a asp-action="AuthorIndex"></a>: This tag redirects to the AuthorIndex page in the Admin Controller. <a asp-action="Book_AuthorIndex"></a>: This tag redirects to the Book_AuthorIndex page in the Admin Controller. <a asp-action="CategoryIndex"> </a>: This tag redirects to the CategoryIndex page in the Admin Controller. <a asp-action="FineIndex"> </a>: This tag redirects to the FineIndex page in the Admin Controller. <a asp-action="LeaseIndex"> </a>: This tag redirects to the LeaseIndex page in the Admin Controller. <a asp-action="PaymentIndex"> </a>: This tag redirects to the PaymentIndex page in the Admin Controller.

		<a asp-action="ReviewIndex"> </a>: This tag redirects to the ReviewIndex page in the Admin Controller.
		<a asp-action="UserIndex"> </a>: This tag redirects to the UserIndex page in the Admin Controller.
		<a asp-action="Track_Renting_Books"> </a>: This tag redirects to the Track_Renting_Books page in the Admin Controller.
		<a asp-action="BookIndex" asp-controller="Home"> </a>: This tag redirects to the current BookIndex page in the Admin Controller.

5	BookIndex (Home): This page displays the main index.	<a asp-action="BookDetails" asp-route-id="@item.Book.ID" ></a>: This tag routes the book’s content to their corresponding BookDetails page.
6	BookDetails (Home): This page displays the book’s details.	<a asp-action="BookIndex"></a>: This tag redirects to the BookIndex page. <a asp-action="LeaseCreate" asp-route-book_ID="@Model.Book.ID"></a>: This tag routes the BookDetails page to their corresponding LeaseCreate page.
7	Filter (Home): This page displays the Filter that is used to search for a specific book.	<select id= “select_box_author”></select>: This tag provides Javascript with the variable author_val. <select id= “select_box_pubyear”></select>: This tag provides Javascript with the variable pubyear_val. <select id= “select_box_cat”></select>: This tag provides Javascript with the variable cat_val.

8	LeaseIndex (Home): This page displays the lease table.	<a asp-action="LeaseDetails" asp-route-id="@item.SessionID"></a>: This tag routes the elements to their corresponding LeaseDetails page.
9	LeaseCreate (Home): This page is used for creating leases.	<input type="submit" value="Creating Lease" class="btn"/>: This tag creates a button for submitting lease. <a asp-action="LeaseIndex"> </a>: This tag redirects to the LeaseIndex page.
10	LeaseDetails (Home): This page displays the lease details.	<a asp-action="PaymentCreate" asp-route-id_lease="@Model.SessionID> </a>: This tag routes the LeaseDetails page to their corresponding PaymentCreate page. <a asp-action="LeaseIndex"> </a>: This tag redirects to the LeaseIndex page. <a asp-action="ReviewCreate" asp-route-id="@Model.Book_ID"> </a>: This tag routes the Payment page to their corresponding ReviewCreate page.

11	Login (Home): This page displays the main index after login	<a asp-action="BookDetails" asp-route-id="@item.Book.ID" ></a>: >: This tag routes the book’s content to their corresponding BookDetails page.
12	Register (Home): This page displays the register section.	<button type = “submit”></button>: This tag creates a button for registering.
13	ReviewCreate (Home): This page is used for creating reviews.	<input type="submit" value="Creating Review" class="btn"/>: This tag determines a button for creating reviews.
14	Search (Home): This page is used for a successful search session in the Filter page.	<a asp-action="BookDetails" asp-route-id="@item.Book.ID" ></a>: This tag routes the book’s content to their corresponding BookDetails page.
15	Searchtest (Home): The user will be redirected to this page for each successful search session in the BookIndex page.	<a asp-action="BookDetails" asp-route-id="@item.Book.ID" ></a>: This tag routes the book’s search results to their corresponding BookDetails page.

16	AuthorCreate (Admin): This page is used to create Authors.	<a asp-action="AuthorIndex"></a>: This tag redirects to the AuthorIndex page. <input type="submit" value="Save"/>: This tag determines a button for creating Authors.
17	AuthorDelete (Admin): This page is used to delete Authors.	<input type="submit" value="Author Delete" class="btn" />: This tag determines a button for deleting Authors. <a asp-action="AuthorIndex"></a>: This tag redirects to the AuthorIndex page.
18	AuthorDetails (Admin): This page is used to view Authors' details.	<a asp-action="AuthorEdit" asp-route-id="@Model.Author_ID"></a>: This tag routes the Authors to their corresponding AuthorEdit page. <a asp-action="AuthorIndex"></a>: This tag redirects to the AuthorIndex page.

19	AuthorEdit (Admin): This page is used to edit Authors' details.	<a asp-action="AuthorIndex"></a>: This tag redirects to the AuthorIndex page. <input type="submit" value="Save" class="btn" />: This tag determines a button for editing Authors.
20	AuthorIndex (Admin): This page displays the report table of the Author section.	<a asp-action="AuthorEdit" asp-route-id="@item.Author_ID"></a>: This tag routes the Author to their corresponding AuthorEdit page. <a asp-action="AuthorDetails" asp-route-id="@item.Author_ID"></a>: This tag routes the Authors to their corrsponding AuthorDetails page. <a asp-action="AuthorDelete" asp-route-id="@item.Author_ID"></a>: This tag routes the Authors to their corrsponding AuthorDeletepage. <a asp-action="AuthorCreate"></a>: This tag redirects to the AuthorCreate page.

21	Book_AuthorCreate: This page is used to create new Book_Authors.	<input type="submit" value="Book_Author Create" class="btn" />: This tag determines a button for creating Book_ Authors. <a asp-action="Book_AuthorIndex"></a>: This tag redirects to the Book_AuthorIndex page.
23	Book_AuthorDelete: This page is used to delete Book_Authors	<input type="submit" value="Delete" class="btn" />: This tag determines a button for deleting Book_ Authors. <a asp-action="Book_AuthorIndex></a>: This tag redirects to the Book_AuthorIndex page.
23	Book_AuthorDetails: This page is used to view Book_Authors details.	<a asp-action="Book_AuthorIndex></a>: This tag redirects to the Book_AuthorIndex page. <a asp-action="Book_AuthorDelete" asp-route-id="@Model.Book.ID" asp-route-id2="@author.Author_ID"></a>: This tag routes the Book_Author to their corresponding Book_AuthorDelete page.

24	Book_AuthorIndex: This page displays the report table of the Book_Author section.	<a asp-action="Book_AuthorCreate"> </a>: This tag redirects to the Book_AuthorCreate page. <a asp-action="Book_AuthorDetails" asp-route-id="@item.Book_Authors.BookID"></a>: This tag routes the Book_Author to their corresponding Book_AuthorDetails page.
25	BookCreate (Admin): This page is used to create Books.	<a asp-action="BookIndex"></a>: This tag redirects to the BookIndex page. <input type="submit" value="BookCreate" class="btn btn-default"/>: This tag determines a button for creating Books.

26	BookDelete (Admin): This page is used to delete Books.	<input type="submit" value="Book Delete" class="btn" />: This tag determines a button for deleting Books. <a asp-action="BookIndex"></a>: This tag redirects to the AuthorIndex page.
27	BookDetails (Admin): This page is used to view Books' details.	<a asp-action="BookEdit" asp-route-id="@Model.ID" class ="navigate-link"></a>: This tag routes the Book to their corresponding BookEdit page. <a asp-action="BookIndex"></a>: This tag redirects to the AuthorIndex page.

28	BookEdit (Admin): This page is used to edit Books' details.	<input type="submit" value="Save" class="btn" />: This tag determines a button for editing Books. <a asp-action="BookIndex"></a>: This tag redirects to the AuthorIndex page.
29	BookIndex (Admin): This page displays the report table of the Book section.	<a asp-action="BookEdit" asp-route-id="@item.ID"></a>: This tag routes the Book to their corresponding BookEdit page. <a asp-action="BookDetails" asp-route-id="@item.ID"></a>: This tag routes the Book to their corresponding BookDetails page. <a asp-action="BookDelete" asp-route-id="@item.ID"></a>: This tag routes the Book to their corresponding BookDelete page. <a asp-action="BookCreate"></a>: This tag redirects to the BookCreate page.

30	FineCreate (Admin): This page is used to create Fines.	<input type="submit" value="Fine Create" class="btn btn-default" />: This tag determines a button for creating Fines. <a asp-action="FineIndex"></a>: This tag redirects to the FineIndex page.
31	FineDelete (Admin): This page is used to delete Fines.	<input type="submit" value="Fine Delete" class="btn" />: This tag determines a button for deleting Fines. <a asp-action="FineIndex"></a>: This tag redirects to the FineIndex page.
32	FineDetails (Admin): This page is used to view Fines' details.	<a asp-action="FineEdit" asp-route-id="@Model.SessionID"></a>: This tag routes the Fine to their corresponding Fine_Edit page. <a asp-action="FineIndex"> </a>: This tag redirects to the FineIndex page.

33	FineEdit (Admin): This page is used to edit Fines' details.	<input type="submit" value="Save" class="btn" />: This tag determines a button for editing Fines. <a asp-action="FineIndex"></a>: This tag redirects to the FineIndex page.
34	FineIndex (Admin): This page displays the report table of the Fine section.	<a asp-action="FineEdit" asp-route-id="@item.SessionID"></a>: This tag routes the Fine to their corresponding FineEdit page. <a asp-action="FineDetails" asp-route-id="@item.SessionID"></a>: This tag routes the Fine to their corresponding FineDetails page. <a asp-action="FineDelete" asp-route-id="@item.SessionID"></a>: This tag routes the Fine to their corresponding FineDelete page. <a asp-action="FineCreate"></a>: This tag redirects to the FineCreate page.

35	LeaseCreate (Admin): This page is used to create Leases.	<input type="submit" value="Create Lease" class="btn" />: This tag determines a button for creating Leases. <a asp-action="LeaseIndex"></a>: This tag redirects to the LeaseIndex page.
36	LeaseDelete (Admin): This page is used to delete Leases.	<input type="submit" value="Lease Delete" class="btn" />: This tag determines a button for deleting Leases. <a asp-action="LeaseIndex"></a>: This tag redirects to the LeaseIndex page.
37	LeaseDetails (Admin): This page is used to view Leases' details.	<a asp-action="LeaseEdit" asp-route-id="@Model.SessionID"></a>: This tag routes the Lease to their corresponding LeaseEdit page. <a asp-action="LeaseIndex"></a>: This tag redirects to the LeaseIndex page.

38	LeaseEdit (Admin): This page is used to edit Leases' details.	<input type="submit" value="Save" class="btn" />: This tag determines a button for editing Lease. <a asp-action="LeaseIndex"></a>: This tag redirects to the LeaseIndex page.
39	LeaseIndex (Admin): This page displays the report table of the Lease section.	<a asp-action="LeaseEdit" asp-route-id="@item.SessionID"></a>: This tag routes the Lease to their corresponding LeaseEdit page. <a asp-action="LeaseDetails" asp-route-id="@item.SessionID"></a>: This tag routes the Lease to their corresponding LeaseDetailspage. <a asp-action="LeaseDelete" asp-route-id="@item.SessionID"></a>: This tag routes the Lease to their corresponding LeaseDelete page. <a asp-action="LeaseCreate"></a>: This tag redirects to the LeaseCreate page.

40	PaymentCreate (Admin): This page is used to create Payments.	<input type="submit" value="Issue an invoice" class="btn" />: This tag determines a button for creating Payments. <a asp-action="PaymentIndex”></a>: This tag redirects to the PaymentIndex page.
41	PaymentDelete (Admin): This page is used to delete Payments.	<input type="submit" value="Delete" class="btn" />: This tag determines a button for deleting Payments. <a asp-action="PaymentIndex"></a>: This tag redirects to the PaymentIndex page.
42	PaymentDetails (Admin): This page is used to view Payments’ details.	<a asp-action="PaymentEdit" asp-route-id="@Model.Payment_ID"></a>: This tag routes the Payment to their corresponding PaymentEdit page. <a asp-action="PaymentIndex"></a>: This tag redirects to the PaymentIndex page.

43	PaymentEdit (Admin): This page is used to edit Payments’ details.	<input type="submit" value="Save" class="btn" />: This tag determines a button for editing Payments. <a asp-action="PaymentIndex"></a>: This tag redirects to the PaymentIndex page.
44	PaymentIndex (Admin): This page displays the report table of the Payment section.	<a asp-action="PaymentEdit" asp-route-id="@item.Payment_ID"></a>: This tag routes the Payment to their corresponding PaymentEdit page. <a asp-action="PaymentDetails" asp-route-id="@item.Payment_ID"></a> <a asp-action="PaymentDelete" asp-route-id="@item.Payment_ID"></a>: This tag routes the Payment to their corresponding PaymentDelete page. <a asp-action="PaymentCreate"></a>: This tag redirects to the PaymentCreate page.

45	ReviewCreate (Admin): This page is used to create Reviews.	<input type="submit" value="ReviewCreate" class="btn" />: This tag determines a button for creating Reviews. <a asp-action="ReviewIndex"></a>: This tag redirects to the ReviewIndex page.
46	ReviewDelete (Admin): This page is used to delete Reviews.	<input type="submit" value="ReviewDelete" class="btn" />: This tag determines a button for deleting Reviews. <a asp-action="ReviewIndex"></a>: This tag redirects to the ReviewIndex page.
47	ReviewDetails (Admin): This page is used to view Reviews' details.	<a asp-action="ReviewEdit" asp-route-id="@Model.ISSNum" asp-route-id2="@Model.IDBook"></a>: This tag routes the Review to their corresponding ReviewEdit page. <a asp-action="ReviewIndex"> </a>: This tag redirects to the ReviewIndex page.

48	ReviewEdit (Admin): This page is used to edit Reviews’ details.	<input type="submit" value="Save" class="btn btn-default" />: This tag determines a button for editing Reviews. <a asp-action="ReviewIndex”></a>: This tag redirects to the ReviewIndex page.
49	ReviewIndex (Admin): This page displays the report table of the Review section.	<a asp-action="ReviewEdit" asp-route-id="@item.ISSNum" asp-route-id2="@item.IDBook"></a>: This tag routes the Review to their corresponding ReviewEdit page. <a asp-action="ReviewDetails" asp-route-id="@item.ISSNum" asp-route-id2="@item.IDBook"></a>: This tag routes the Review to their corresponding ReviewDetails page. <a asp-action="ReviewDelete" asp-route-id="@item.ISSNum" asp-route-id2="@item.IDBook"></a>: This tag routes the Review to their corresponding ReviewDelete page. <a asp-action="ReviewCreate"></a>: This tag redirects to the ReviewCreatepage.

50	UserCreate (Admin): This page is used to create Users.	<input type="submit" value="Create" class="btn" />: This tag determines a button for creating Users. <a asp-action="UserIndex"></a>: This tag redirects to the UserIndex page.
51	UserDelete (Admin): This page is used to delete Users.	<input type="submit" value="User Delete" class="btn" />: This tag determines a button for deleting Users. <a asp-action="UserIndex"></a>: This tag redirects to the UserIndex page.
52	UserDetails (Admin): This page is used to view Users’ details.	<a asp-action="UserEdit" asp-route-id="@Model.ISSN"></a>: This tag routes the User to their corresponding UserEdit page. <a asp-action="UserIndex"></a>: This tag redirects to the UserIndex page.

53	UserEdit (Admin): This page is used to edit Users’ details.	<input type="submit" value="Save" class="btn" />: This tag determines a button for editing Users. <a asp-action="UserIndex"></a>: This tag redirects to the UserIndex page.
54	UserIndex (Admin): This page displays the report table of the User section.	<a asp-action="UserEdit" asp-route-id="@item.ISSN"></a>: This tag routes the User to their corresponding UserEdit page. <a asp-action="UserDetails" asp-route-id="@item.ISSN"></a>: This tag routes the User to their corresponding UserDetails page. <a asp-action="UserDelete" asp-route-id="@item.ISSN"></a>: This tag routes the User to their corresponding UserDelete page. <a asp-action="UserCreate"></a>: This tag redirects to the UserCreate page.

55	Track_Renting_Books (Admin):	<a asp-action="Track_Renting_Book" asp-route-id_book="@item.Book.ID"> </a>: This tag routes the renting books to their corresponding list leases page.
56	Track_Renting_Book (Admin):	<a asp-action="LeaseDetails" asp-route-id="@item.SessionID"></a>: This tag routes the renting books to their corresponding LeaseDetails page. <a asp-action="LeaseEdit" asp-route-id="@item.SessionID"></a>: This tag routes the renting books to their corresponding LeaseEdit page. <a asp-action="Track_Renting_Books"></a>: This tag redirects to the Track_Renting_Books page.

57	Search (Admin): This page displays the Search result.	<a asp-action="BookEdit" asp-route-id="@item.Book.ID"></a>: This tag routes the Book to their corresponding BookEdit page. <a asp-action="BookDetails" asp-route-id="@item.Book.ID"></a>: This tag routes the Book to their corresponding BookDetails page. <a asp-action="BookDelete" asp-route-id="@item.Book.ID"></a>: This tag routes the Book to their corresponding BookDelete page. <a asp-action="BookCreate"></a>: This tag routes the Book to their corresponding BookCreate page.

Table 11: *Details of Behavior HTML Tags in UI Page*

6. Details of Function UI Page

No.	**Pages and Description**	**Variables and Scopes**	**Methods and Functionalities**

1	_Layout (Home): This template is used as the default Layout on first attempt entering the website.	Local Scope: Audio audio Variables: ActionName Searchtest Login ControllerName Home	function playBgMusic () {audio.play ()}: This function controls the “Play” button. function pauseBgMusic (audio.pause()) {}: This function controls the “Pause” button. @using (“Searchtest”, “Home”, FormMethod.Post) {}: This function creates a form method redirects to Searchtest for the Controller Home. @using (Html.BeginForm(“Login”, “Home”, FormMethod.Post)) {}: This function creates a form method redirects to Login for the Controller Home.

2	_Layout1(Home): This template is used as the default Layout when the User Admin log in.	Local Scope: Audio audio Variables: ActionName Searchtest Login ControllerName Home	function playBgMusic () {audio.play ()}: This function controls the "Play" button. function pauseBgMusic (audio.pause()) {}: This function controls the "Pause" button. @using ("Searchtest", "Home", FormMethod.Post) {}: This function creates a form method redirects to Searchtest for the Controller Home. @using (Html.BeginForm("Login", "Home", FormMethod.Post)) {}: This function creates a form method redirects to Login for the Controller Home.

3	_Layout1_cus (Home): This template is used as the default Layout when other default User/Customer log in	Local Scope: Audio audio	function playBgMusic () {audio.play ()}: This function controls the "Play" button. Function pauseBgMusic (audio.pause()) {}: This function controls the "Pause" button.
4	_Layout_2 (Admin): This template is used as the default Layout for the Admin functions.	Null	Null
5	BookIndex (Home): This page displays the main index.	Variables: URL url, string Description, int Publication_Year	BookIndex (URL url, string Description, int Publication_Year): This function displays the list books and their corresponding images based on their url, description and publication year.

6	BookDetails (Home): This page displays the book’s details.	Variables: string Title, string Category, int Publication_Year, string Author_fname, Date Review_date, string Review_context, URL url	BookDetails (string Title, string Category, int Publication_Year, string Author_fname, Date Review_date, string Review_context, URL url): This function displays the books details and their corresponding images based on their title, categories, publication year, first name, date review(s) and context(s).

7	Filter (Home): This page displays the Filter that is used to search for a specific book.	Local Scope: array[] search Variable: string Category_name, string Author_fname, int Publication, Button btn, Value cat_val, Value author_val, Value pubyear_val	$(“”).select2(): This method initializes the select2 library, which is used for filtering data on any random jQuery selector. Filter (string Category_name, string Author_fname, int Publication): This function display three filters: category name, author name and publication year. function AddToFilterTest(cat_val, author_val, pubyear_val): This function is used to print out the search’s results. function myresult(): This function logs the result into a section below the filters. function AjaxFailed(catVal, authorVal, pubYearVal): This function is used to logout the error text.

			btn.addEventListener('click', event()){search.push(catVal, authorVal, pubyearVal)}: This function adds a click event to the 'btn' button. The category, author and publication year which is the user's input are also added into an array.

8	LeaseIndex (Home): This page displays the lease table.	Variables: int SessionID, int Book_ID, int ISSN, Date Lease_date, Date Expiry_date	LeaseIndex (int SessionID, int Book_ID, int ISSN, Date Lease_date, Date Expiry_date): This function displays the lease table based on their session ID, book ID, ISSN, lease date and expiry date.
9	LeaseCreate (Home): This page is used for creating leases.	Variables: Date Lease_date	LeaseCreate (Date Lease_date): This function displays the lease date inputs for User/Customer.
10	LeaseDetails (Home): This page displays the lease details.	Variables: int SessionID, int Book_ID, int ISSN, Date Lease_date, Date Expiry_date, string status	LeaseDetails (int SessionID, int Book_ID, int ISSN, Date Lease_date, Date Expiry_date, string status): This function displays the lease details.

11	Login (Home): This page displays the main index after login	Variables: URL url, string Description, int Publication_Year	Login (URL url, string Description, int Publication_Year): This function displays the list books and their corresponding images based on their url, description and publication year.
12	Register (Home): This page displays the register section.	Variables: int ISSN, string Name, Email email, Address address, int Phone, Password pass	Register (int ISSN, string Name, Email email, Address address, int Phone, Password pass): This function displays the values that are required for registering.
13	ReviewCreate (Home): This page is used for creating reviews.	Variables: Context Review_context, int Review_star	ReviewCreate (): This function displays the review create inputs for User/Customer.

14	Search (Home): This page is used for a successful search session in the Filter page.	Variables: URL url, string Description, int Publication_Year	Search (URL url, string Description, int Publication_Year): This function displays the search results and their corresponding images based on their url, description and publication year.
15	Searchtest (Home): The user will be redirected to this page for each successful search session in the BookIndex page.	Variables: URL url, string Description, int Publication_Year	Searchtest (URL url, string Description, int Publication_Year): This function displays the search results and their corresponding images based on their url, description and publication year.
16	AuthorCreate (Admin): This page is used to create Authors.	Variables: int Author_ID, string Author_fname, string Author_lname	AuthorCreate (int Author_ID, string Author_fname, string Author_lname): This function displays the Author create inputs for Admin.

17	AuthorDelete (Admin): This page is used to delete Authors.	Variables: int Author_ID, string Author_fname, string Author_lname	AuthorDelete (int Author_ID, string Author_fname, string Author_lname): This function displays the Author delete page for Admin.
18	AuthorDetails (Admin): This page is used to view Authors' details.	Variables: int Author_ID, string Author_fname, string Author_lname	AuthorDetails (int Author_ID, string Author_fname, string Author_lname): This function displays the Author's details for Admin.
19	AuthorEdit (Admin): This page is used to edit Authors' details.	Variables: int Author_ID, string Author_fname, string Author_lname	AuthorEdit (int Author_ID, string Author_fname, string Author_lname): This function displays the Author edit inputs for Admin.
20	AuthorIndex (Admin): This page displays the report table of the Author section.	Variables: int Author_ID, string Author_fname, string Author_lname	A uthorIndex (int Author_ID, string Author_fname, string Author_lname): This function displays a table that summarizes all the functions related to Author for Admin.

21	Book_AuthorCreate: This page is used to create new Book_Authors.	Variables: int BookID, int AuthorID	Book_AuthorCreate (int BookID, int AuthorID): This function displays the Book_Author create inputs for Admin.
23	Book_AuthorDelete: This page is used to delete Book_Authors	Variables: string Book_Title, string Author_Name	Book_AuthorDelete (string Book_Title, string Author_Name): This function displays the Book_Author deletes page for Admin.
23	Book_AuthorDetails: This page is used to view Book_Authors details.	Variables: int ID, string Title, string Authors	Book_AuthorDetails (int ID, string Title, string Authors): This function displays the Book_Author details for Admin.
24	Book_AuthorIndex: This page displays the report table of the Book_Author section.	Variables: int BookID, int AuthorID, string Author_name	Book_AuthorIndex (int BookID, int AuthorID, string Author_name): This function displays a table that summarizes all the functions related to Book_Author for Admin.

25	BookCreate (Admin): This page is used to create Books.	Variables: int ID, string Title, int Category_ID, int Publication_Year, int Quantity	BookCreate (int ID, string Title, int Category_ID, int Publication_Year, int Quantity): This function displays the Book create inputs for Admin.
26	BookDelete (Admin): This page is used to delete Books.	Variables: string Title, int Category_ID, int Publication_Year, int Quantity	BookDelete (string Title, int Category_ID, int Publication_Year, int Quantity): This function displays the Book deletes page for Admin.
27	BookDetails (Admin): This page is used to view Books' details.	Variables: string Title, int Category_ID, int Publication_Year, int Quantity	BookDetails (string Title, int Category_ID, int Publication_Year, int Quantity): This function displays the Book details for Admin.
28	BookEdit (Admin): This page is used to edit Books' details.	Variables: string Title, int Category_ID, int Publication_Year, int Quantity	BookEdit (string Title, int Category_ID, int Publication_Year, int Quantity): This function displays the Book edit inputs for Admin.

29	BookIndex (Admin): This page displays the report table of the Book section.	Variables: int ID, string Title, int Category_ID, int Publication_Year, int Quantity	BookIndex (int ID, string Title, int Category_ID, int Publication_Year, int Quantity): This function displays a table that summarizes all the functions related to Book for Admin.
30	FineCreate (Admin): This page is used to create Fines.	Variables: int Session_ID, int Fine_days, int Fine_amount	FineCreate (int Session_ID, int Fine_days, int Fine_amount): This function displays the Fine create inputs for Admin.
31	FineDelete (Admin): This page is used to delete Fines.	Variables: int Session_ID, int Fine_days, int Fine_amount	FineDelete (int Session_ID, int Fine_days, int Fine_amount): This function displays the Fine deletes page for Admin.
32	FineDetails (Admin): This page is used to view Fines’ details.	Variables: int Session_ID, int Fine_days, int Fine_amount	FineDetails (int Session_ID, int Fine_days, int Fine_amount): This function displays the Fine details for Admin.

33	FineEdit (Admin): This page is used to edit Fines' details.	Variables: int Fine_days, int Fine_amount	FineEdit (int Fine_days, int Fine_amount): This function displays the Fine edit inputs for Admin.
34	FineIndex (Admin): This page displays the report table of the Fine section.	Variables: int Session_ID, Date Fine_days, Date Fine_amount	FineIndex (int Session_ID, Date Fine_days, Date Fine_amount): This function displays a table that summarizes all the functions related to Fine for Admin.
35	LeaseCreate (Admin): This page is used to create Leases.	Variables: int SessionID, int Book_ID, int ISSN, Date Lease_date	LeaseCreate (int SessionID, int Book_ID, int ISSN, Date Lease_date): This function displays the Book create inputs for Admin.
36	LeaseDelete (Admin): This page is used to delete Leases.	Variables: int SessionID, int Book_ID, int ISSN, Date Lease_date, Date Expiry_date, string Status	LeaseDelete (int SessionID, int Book_ID, int ISSN, Date Lease_date, Date Expiry_date, string Status): This function displays the Book deletes page for Admin.

37	LeaseDetails (Admin): This page is used to view Leases' details.	Variables: int SessionID, int Book_ID, int ISSN, Date Lease_date, Date Expiry_date, string Status	LeaseDetails (int SessionID, int Book_ID, int ISSN, Date Lease_date, Date Expiry_date, string Status): This function displays the Book details for Admin.
38	LeaseEdit (Admin): This page is used to edit Leases' details.	Variables: int Book_ID, int ISSN, Date Lease_date, Date Expiry_date, string Status	LeaseEdit (int Book_ID, int ISSN, Date Lease_date, Date Expiry_date, string Status): This function displays the Book edit inputs for Admin.
39	LeaseIndex (Admin): This page displays the report table of the Lease section.	Variables: int SessionID, int Book_ID, int ISSN, Date Lease_date, Date Expiry_date, string Status	LeaseIndex (int SessionID, int Book_ID, int ISSN, Date Lease_date, Date Expiry_date, string Status): This function displays a table that summarizes all the functions related to Book for Admin.

40	PaymentCreate (Admin): This page is used to create Payments.	Variables: int Payment_ID, int Customer_ID	PaymentCreate (int Payment_ID, int Customer_ID): This function displays the Payment create inputs for Admin.
41	PaymentDelete (Admin): This page is used to delete Payments.	Variables: int Payment_ID, int Customer_ID, Date Lease_date, Date Payment_date, int Payment_amount	PaymentDelete (int Payment_ID, int Customer_ID, Date Lease_date, Date Payment_date, int Payment_amount): This function displays the Payment deletes page for Admin.
42	PaymentDetails (Admin): This page is used to view Payments' details.	Variables: int Payment_ID, int Customer_ID, Date Lease_date, Date Payment_date, int Payment_amount	PaymentDetails (int Payment_ID, int Customer_ID, Date Lease_date, Date Payment_date, int Payment_amount): This function displays the Payment details for Admin.

43	PaymentEdit (Admin): This page is used to edit Payments' details.	Variables: int Customer_ID, Date Lease_date, Date Payment_date, int Payment_amount	PaymentEdit (int Customer_ID, Date Lease_date, Date Payment_date, int Payment_amount): This function displays the Payment edit inputs for Admin.
44	PaymentIndex (Admin): This page displays the report table of the Payment section.	Variables: int Payment_ID, int Customer_ID, Date Lease_date, Date Payment_date, int Payment_amount	PaymentIndex (int Payment_ID, int Customer_ID, Date Lease_date, Date Payment_date, int Payment_amount): This function displays a table that summarizes all the functions related to Payment for Admin.
45	ReviewCreate (Admin): This page is used to create Reviews.	Variables: int ISSNum, int IDBook, Date Review_date, string Review_context, int Review_star	ReviewCreate (int ISSNum, int IDBook, Date Review_date, string Review_context, int Review_star): This function displays the Review create inputs for Admin.

46	ReviewDelete (Admin): This page is used to delete Reviews.	Variables: int ISSNum, int IDBook, Date Review_date, string Review_context, int Review_star	ReviewDelete (int ISSNum, int IDBook, Date Review_date, string Review_context, int Review_star): This function displays the Review deletes page for Admin.
47	ReviewDetails (Admin): This page is used to view Reviews’ details.	Variables: int ISSNum, int IDBook, Date Review_date, string Review_context, int Review_star	ReviewDetails (int ISSNum, int IDBook, Date Review_date, string Review_context, int Review_star): This function displays the Review details for Admin.
48	ReviewEdit (Admin): This page is used to edit Reviews’ details.	Variables: int IDBook, Date Review_date, string Review_context, int Review_star	ReviewEdit (int IDBook, Date Review_date, string Review_context, int Review_star): This function displays the Review edit inputs for Admin.

49	ReviewIndex (Admin): This page displays the report table of the Review section.	Variables: int ISSNum, int IDBook, Date Review_date, string Review_context, int Review_star	ReviewIndex (int ISSNum, int IDBook, Date Review_date, string Review_context, int Review_star): This function displays a table that summarizes all the functions related to Review for Admin.
50	UserCreate (Admin): This page is used to create Users.	Variables: int ISSN, string Name, string Email, string Address, string Phone, string Pass	UserCreate (int ISSN, string Name, string Email, string Address, string Phone, string Pass): This function displays the User create inputs for Admin.
51	UserDelete (Admin): This page is used to delete Users.	Variables: string Name, string Email, string Address, string Phone	UserDelete (string Name, string Email, string Address, string Phone): This function displays the User deletes page for Admin.

52	UserDetails (Admin): This page is used to view Users’ details.	Variables: string Name, string Email, string Address, string Phone	UserDetails (string Name, string Email, string Address, string Phone): This function displays the User details for Admin.
53	UserEdit (Admin): This page is used to edit Users’ details.	Variables: int ISSN, string Name, string Email, string Address, string Phone, string Pass	UserEdit (int ISSN, string Name, string Email, string Address, string Phone, string Pass): This function displays the User edit inputs for Admin.
54	UserIndex (Admin): This page displays the report table of the User section.	Variables: int ISSN, string Name, string Email, string Address, string Phone	UserIndex (int ISSN, string Name, string Email, string Address, string Phone, string Pass): This function displays a table that summarizes all the functions related to User for Admin.

55	Track_Renting_Books (Admin):	Variables: string Title, int Category_ID, int Publication_Year, int Quantity, int available Local Scope (filterTitle ()): var input, filter, table, tr, td, i, txtValue	Track_Renting_Books (string Title, int Category_ID, int Publication_Year, int Quantity, int available): This function displays a table that tracks the books’ status for Admin. filterTitle (): This function provides a search filter that loops through all rows and find the search query based on the “Title” column. Those who don’t match will be temporarily hidden.

56	Track_Renting_Book (Admin):	Variables: int Session_ID, int ISSN, Date Expiry_date, string Status Local Scope (filterISSN ()): var input, filter, table, tr, td, i, txtValue	Track_Renting_Book (int Session_ID, int ISSN, Date Expiry_date, string Status): This function displays the current book's status and its functions. filterTitle (): This function provides a search filter that loops through all rows and find the search query based on the "ISSN" column. Those who don't match will be temporarily hidden.
57	Search (Admin): This page displays the Search result.	Variables: int ID, string Title, int Category_ID, int Publication_Year, int Quantity	Search (int ID, string Title, int Category_ID, int Publication_Year, int Quantity): This function displays a table that summarizes all the functions related to Book for Admin.

Table 12: *Details of Function UI Page*

7. Details of Function Library MVC

No.	Methods and functionalities	Description

1.	[HTTPGET]	These functions return the data and it is called page-load.
2.	[HTTPPOST]	These functions return the data afther a posting form or ajax.
3.	[ACTIONNAME]	It allows to specify a different action name than the method name.
4.	[VALIDATEANTIFORGERYTOKEN]	The function support writes a unique vlue to an HTTP-only cookie and then the same value is written to the form. When the page is submitted, an error is raised if the cookie value doesn't match the form value. The feature doesn't prevent any other type of data forgery or tampering based attacks.
5.	IActionResult	The return type is appropriate when multiple ActionResult return types are possible in an action.

6.	ActionResult	The return type is represented the result of an action method.
7.	RedirectToActionResult	An ActionResult that returns a Found(302), Moved Permanently(301), Temporary Redirect(307) or Permanent Redirect(308) response with a Location header. Targets a controller action.
8.	PartialView	A partial view is a Razor markup file (.cshtml) without a page directive that renders HTML output within another markup file's rendered output. The term partial view is used when developing either an MVC app, where markup files are called views, or a Razor Pages app, where markup files are called pages. This topic generically refers to MVC views and Razor Pages as markup files.
9.	NotFound()	A page view display an error not found page 404.

Table 13: Details of Function Library MVC

IV. NUGET PACKAGE

1. Devart.Data.MySql.EFCore

dotConnect for MySQL is an enhanced database connectivity solution built over ADO.NET architecture and a development framework with advanced support for ORMs, such as Entity Framework and EF Core, and offers a complete solution for developing DB-related applications and web sites. It introduces new approaches for designing a data access layer and boosts the productivity of database application development. It is the best alternative to Connector/NET for working with MySQL data.

dotConnect for MySQL supports MySQL, including Embedded servers, MariaDB, Percona, and Amazon RDS Aurora.

This package contains the .NET Standard/.NET Core compatible assemblies with Entity Framework Core-related functionality of dotConnect for MySQL. It includes support for Entity Framework Core 1.1, 3.1, 5.0 and 6.0. Note that for Entity Framework Core 1.1 it does not include support for database-first approach via the Scaffold-DbContext command, which is available via the separate Devart.Data.MySql.EFCore.Design package.

This package contains only runtime features of dotConnect for MySQL. dotConnect for MySQL is also provided as an installation package (exe), which installs runtime assemblies for Full .NET Framework and a set of design-time tools, integrated into Visual Studio - Server Explorer integration, DataSet tools, Windows Forms components with powerful design-time, etc. It also includes visual ORM designer for Entity Framework, Entity Framework Core, and LinqConnect ORM models. Besides, this installer generates the trial key files required for using this package on a trial basis. You can download it at www.devart.com.

2. Microsoft.AspNet.MVC

This package contains the runtime assemblies for ASP.NET MVC. ASP.NET MVC gives you a powerful, patterns-based way to build dynamic websites that enables a clean separation of concerns and that gives you full control over markup.

3. Microsoft.AspNet.Session

ASP.NET 5 session state middleware.

4. Microsoft.Extensions.Caching.Memory

In-memory cache zone, the implementation of Microsoft.Extensions.Caching.Memory.IMemoryCache. used by Microsoft.AspNet.MVC library.

5. Microsoft.VisualStudio.Azure.Containers.Tools.Targets

To target files to enable the Visual Studio Tools for Containers.

6. MySql.Data

MySql.Data.MySqlClient .Net Core Class Library.

7. MySqlConnector

MySqlConnector is an ADO.NET data provider for MySQL, MariaDB, Amazon Aurora, Azure Database for MySQL, and other MySQL-compatible databases.

Key Features

- Full support for async I/O
- High performance
- Supports .NET Framework, .NET Core, and .NET 5.0+

Main Types

The main types provided by this library are:

- MySqlConnection (implementation of DbConnection)
- MySqlCommand (implementation of DbCommand)
- MySqlDataReader (implementation of DbDataReader)
- MySqlBulkCopy
- MySqlBulkLoader
- MySqlConnectionStringBuilder
- MySqlConnectorFactory
- MySqlDataAdapter
- MySqlException
- MySqlTransaction (implementation of DbTransaction)

8. System.Data.SqlClient

Provides the data provider for SQL Server. These classes provide access to versions of SQL Server and encapsulate database-specific protocols, including tabular data stream (TDS).

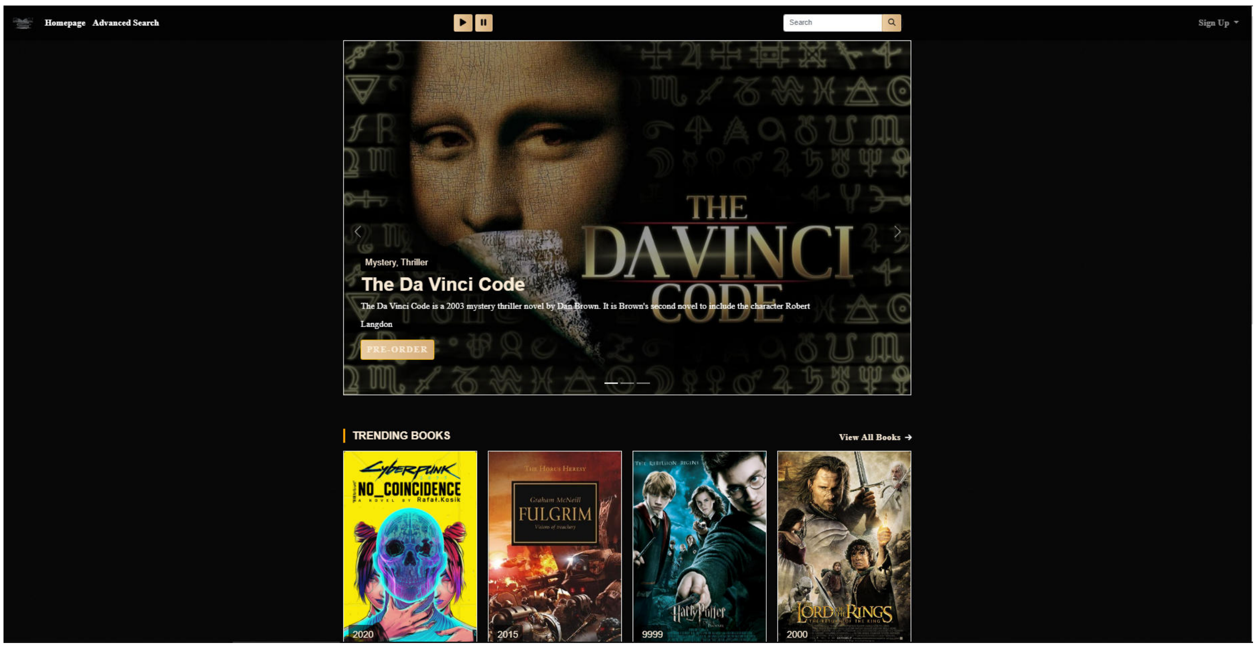
Commonly Used Types:

- System.Data.SqlClient.SqlConnection
- System.Data.SqlClient.SqlException
- System.Data.SqlClient.SqlParameter
- System.Data.SqlDbType
- System.Data.SqlClient.SqlDataReader
- System.Data.SqlClient.SqlCommand
- System.Data.SqlClient.SqlTransaction
- System.Data.SqlClient.SqlParameterCollection
- System.Data.SqlClient.SqlClientFactory

9. xunit.assert

Includes the assertion library from xUnit.net (xunit.assert.dll) supports .NET Standard 1.1.

V. INTERFACE DESIGN

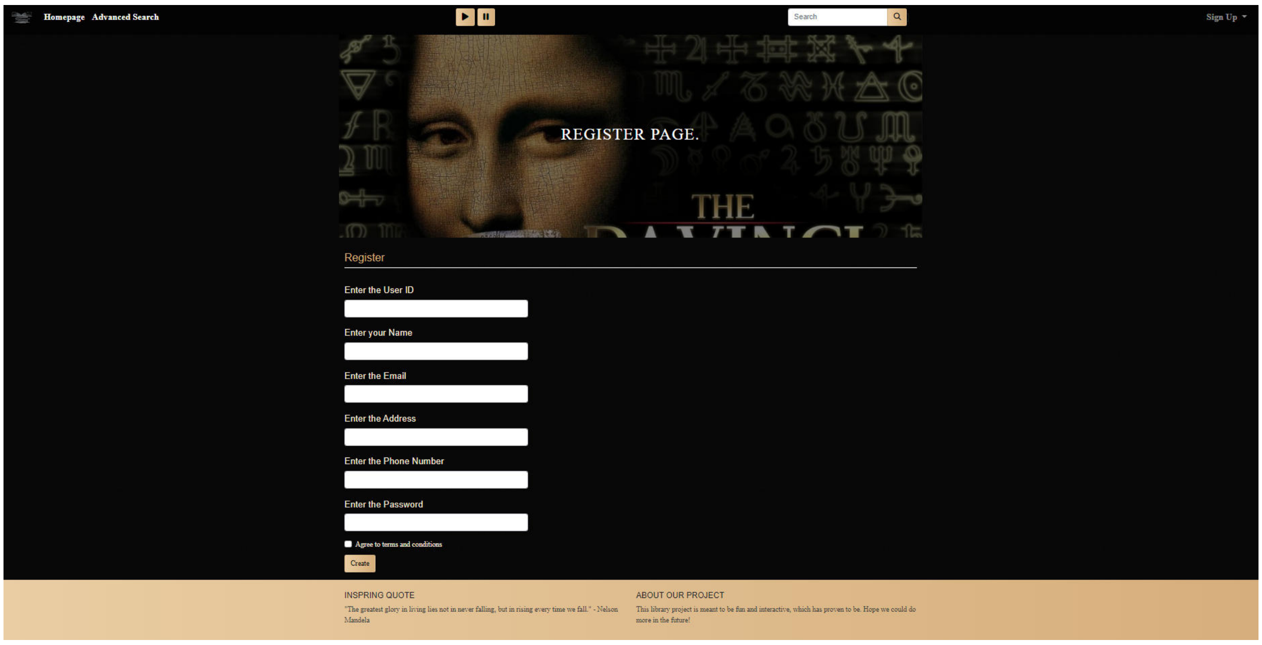


Mystery, Thriller

The Da Vinci Code

The Da Vinci Code is a 2003 mystery thriller novel by Dan Brown. It is Brown's second novel to include the character Robert Langdon

***Figure 12**: Main Page*



Register

Enter the User ID

Enter your Name

Enter the Email

Enter the Address

Enter the Phone Number

Enter the Password

Agree to terms and conditions

Create

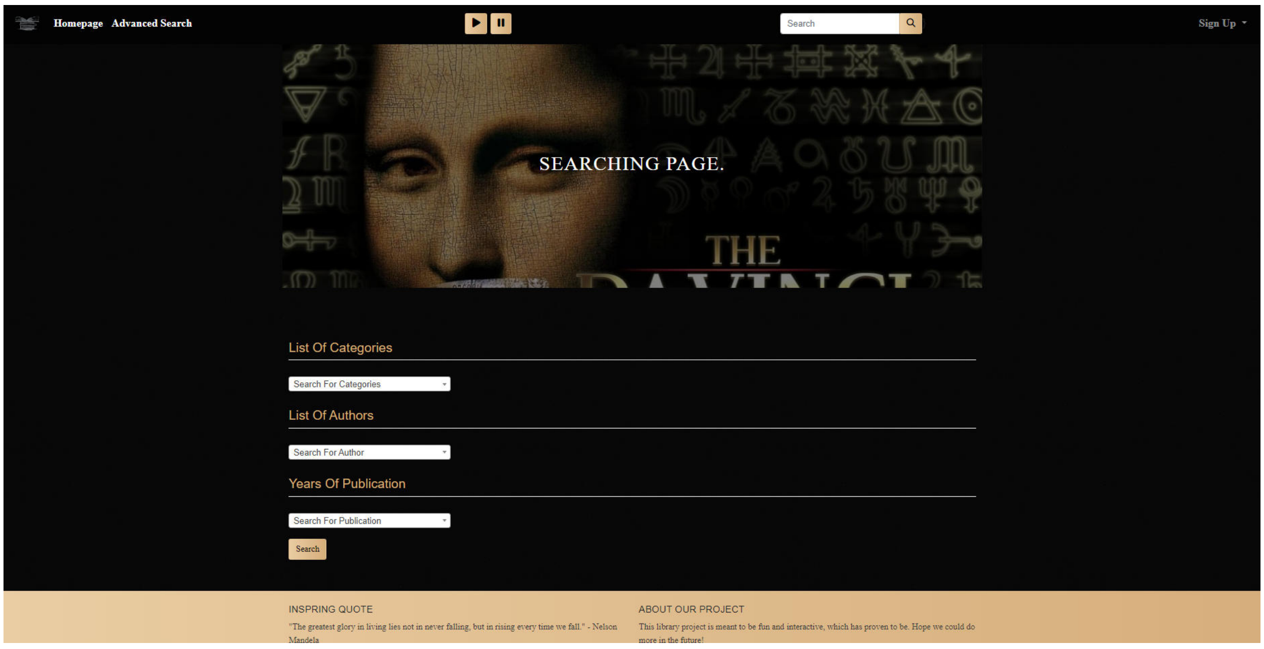
INSPRING QUOTE

"The greatest glory in living lies not in never falling, but in rising every time we fall." - Nelson Mandela

ABOUT OUR PROJECT

This library project is meant to be fun and interactive, which has proven to be. Hope we could do more in the future!

Figure 13: *Register Page*



List Of Categories

Search For Categories

List Of Authors

Search For Author

Years Of Publication

Search For Publication

Search

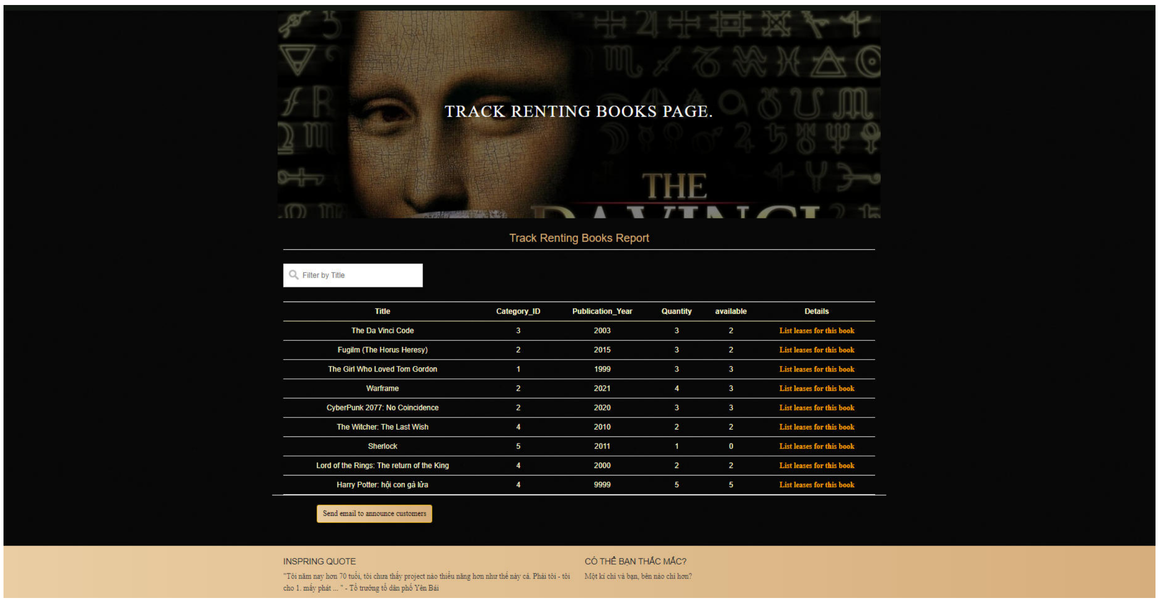
INSPRING QUOTE

"The greatest glory in living lies not in never falling, but in rising every time we fall." - Nelson Mandela

ABOUT OUR PROJECT

This library project is meant to be fun and interactive, which has proven to be. Hope we could do more in the future!

Figure 14: *Filter - Search Page*



Track Renting Books Report

Filter by Title

Title	Category_ID	Publication_Year	Quantity	available	Details
The Da Vinci Code	3	2003	3	2	List leases for this book
Fugilm (The Horus Heresy)	2	2015	3	2	List leases for this book
The Girl Who Loved Tom Gordon	1	1999	3	3	List leases for this book
Warframe	2	2021	4	3	List leases for this book
CyberPunk 2077: No Coincidence	2	2020	3	3	List leases for this book
The Witcher: The Last Wish	4	2010	2	2	List leases for this book
Sherlock	5	2011	1	0	List leases for this book
Lord of the Rings: The return of the King	4	2000	2	2	List leases for this book
Harry Potter: hội con gà lửa	4	9999	5	5	List leases for this book

Send email to announce customers

INSPRING QUOTE

"Tôi năm nay hơn 70 tuổi, tôi chưa thấy project nào thiếu năng hơn như thế này cả. Phải tôi - tôi cho 1. mấy phát ... " - Tổ trưởng tổ dân phố Yên Bái

CÓ THỂ BẠN THẮC MẮC?

Một kí chì và bạn, bên nào chì hơn?

Figure 15: *Track Renting Page*

VI. EXPERIMENT

- Advantages of the project:
  - It helps us to learn C# and JavaScript with hands-on experience.
  - It teaches us to work as a small team.
  - It helps us know what steps we should take when developing an application from scratch.
  - Although it is a small project, we can still put it in our CVs as a proper project and experience.
  - It helps us to realize what our capabilities are because there are other functionalities we would want to develop but we are limited in terms of time and experience.
- Disadvantages of the project:
  - Other than being physically and mentally drained sometimes, it was a great experience for us to know a bit about how a project should be done industrially.
- Personally, I believe our project can be done with higher quality if we had more time. In terms of functionality, the application does not allow the users to edit existing records, if they want to do that, they would have to delete that record and create a new one with the edited fields of their preference. In terms of OOP when developing the project, because we did not plan it properly, hence, the OOP techniques used in the project were not many.
- If we had more time and planned the design properly, we believe that we would be able to add a feature to let the users edit an existing record and also produce better code to optimize OOP properties that C# gives us.

VII. REFERENCE

Since we use new programming languages as well as new frameworks and other things, we decided to list our references here for you in case you want to give this C# and .NET platform a try.

.NET documentation | Microsoft Docs
C# docs - get started, tutorials, reference. | Microsoft Docs
ADO.NET | Microsoft Docs
Get started with ASP.NET Core MVC | Microsoft Docs
CRUD Operation With ASP.NET Core MVC Using ADO.NET and Visual Studio 2017 - Ankit Sharma's Blog (ankitsharmablogs.com)
jQuery.ajax() | jQuery API Documentation
The Razor _Layout.cshtml file | Learn Razor Pages
Hướng dẫn tự học lập trình ASP.NET Core toàn tập | Tự học ICT (tuhocict.com)
NuGet Gallery | Home

DUTY ROSTER

ID	Task	In Charge	Start	End	State	Note
1	MySQL Database	Nguyen Khoa	8-Mar-23	17-Mar-23	Done	
2	Design Class Diagram	Nguyen Khoa Hoang The Vinh	28-Feb-23	21-Mar-23	Done	
3	Design Interface Classes	Nguyen Van Khanh Huynh Duong Khai	23-Mar-23	8-May-23	Done	
4	Design Application Class	Tran Huu Hoang Do Hoang The Vinh	23-Mar-23	8-May-23	Done	
5	Design Database Classes	Tran Huu Hoang Do Hoang The Vinh	23-Feb-23	27-Apr-23	Done	

6	Report Section I	Nguyen Khoa Hoang The Vinh	13-Apr-23	11-Feb-23	Done	
7	Report Section II	Huynh Duong Khai Nguyen Van Khanh	12-Feb-23	12-Jun-23	Done	
8	Testing and Report Section III	Nguyen Khoa Nguyen Van Khanh Huynh Duong Khai Tran Huu Hoang Do Hoang The Vinh	17-Feb-23	20-Jun-23	Done	