Web Application

Bako Asaad
510944

25/6/2023

Second-Year Student

Contents

Class Diagram

In this section, I will be presenting a class diagram representing the database structure of the "**Happy Reader**" application. This class diagram describes the relationship between the tables of the database.

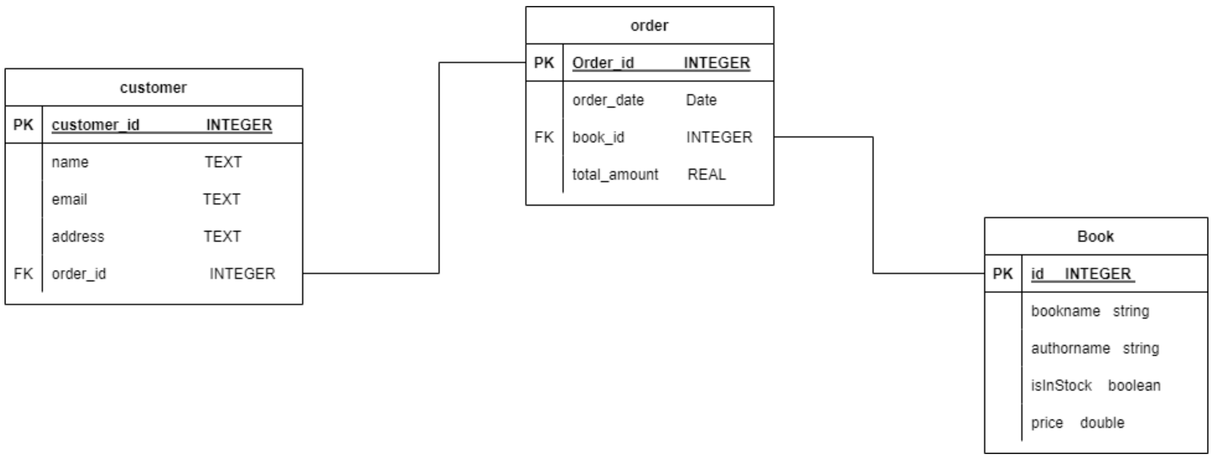
I have created three tables in the database: "**Orders, Books**, and **Customers**". Each table represents an entity and contains specific attributes related to that entity.

The "Orders" table stores the orders that were made by customers. It contains attributes such as '**order_id**', '**order_date**', '**total_amount'**, '**book_id**', and '**customer_id**'.

The "**Books**" table stores the books that are available for ordering. It includes attributes such as '**book_id**', '**book_name**', '**author_name**', '**category'**, and '**price'**.

The "**customers**" table stores the information of those customers of have placed orders. It contains attributes such as '**customer_id**', '**name**', '**email**', and '**address**'.

As it is shown in the diagram below, we can see the relationship between these tables. the "Orders" table is connected to both the "Books" and the "Customers" tables, meaning that each order is associated with a specific book or books and a single customer.



The relationship between the entities:

- **Orders**
  - Has a Many-to-One relationship with Books (many orders can be associated with a single book).
  - Has a One-to-One relationship with Customers (each order is associated with a single customer).
- **Books**
  - Has a One-to-Many relationship with Orders (each book can be associated with multiple orders).
- **Customers**
  - Has a One-to-Many relationship with Orders (each customer can have multiple orders).

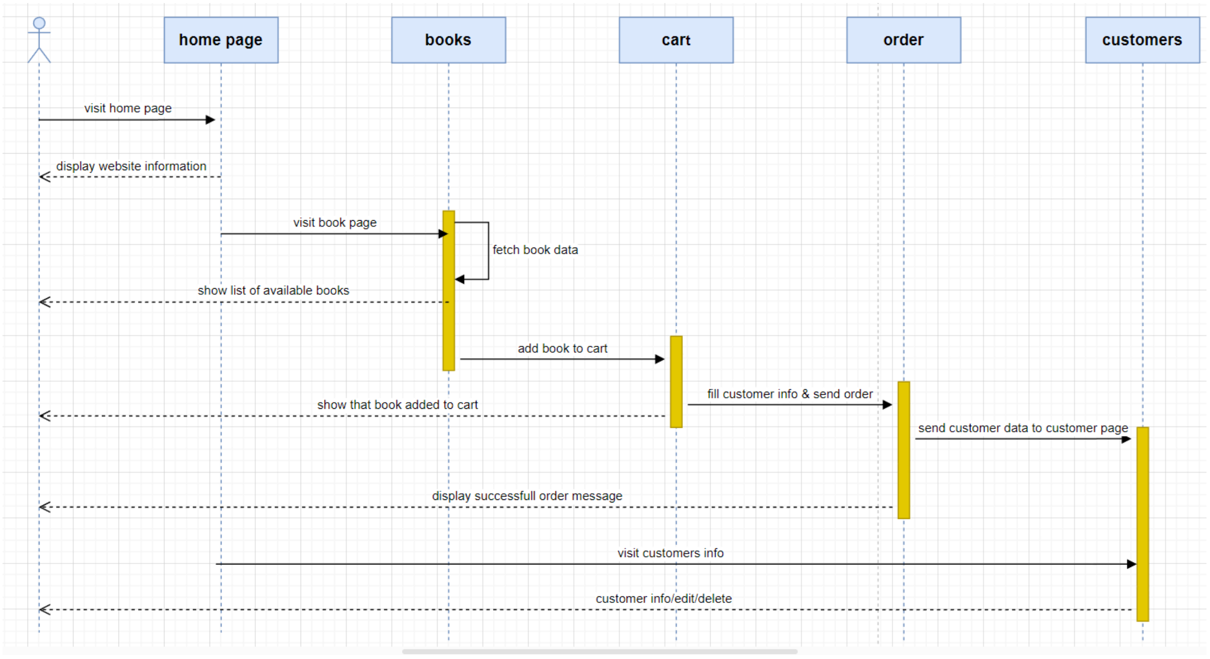
Sequence Diagram

In this section, I will present a sequence diagram along with the steps that a customer takes as they navigate through the pages of the website when ordering their desired books.

1. **Customer visits the Home page:**
   The customer starts by accessing the website's home page, where they can view information about the website.
2. **Customer navigates to the Books page:**
   From the home page, a customer clicks on the letter "books" that is on the header, which takes them to the books page, where customers can see the available books that were fetched from the database.
3. **Customer adds books to the Shopping Cart:**
   On the books page, the customer selects their desired books and adds them to their shopping cart. When clicking on add to cart button, a green icon will be displayed to the customer as a confirmation.
4. **Customer navigates to the Shopping Cart:**
   After selecting the books, the customer navigates to the shopping cart page. Here, they can view the list of books they have added and see the total amount.
5. **Customer places Order:**
   In the shopping cart, the customer clicks on the "Place Order" button. this action will send the customer to the order page.

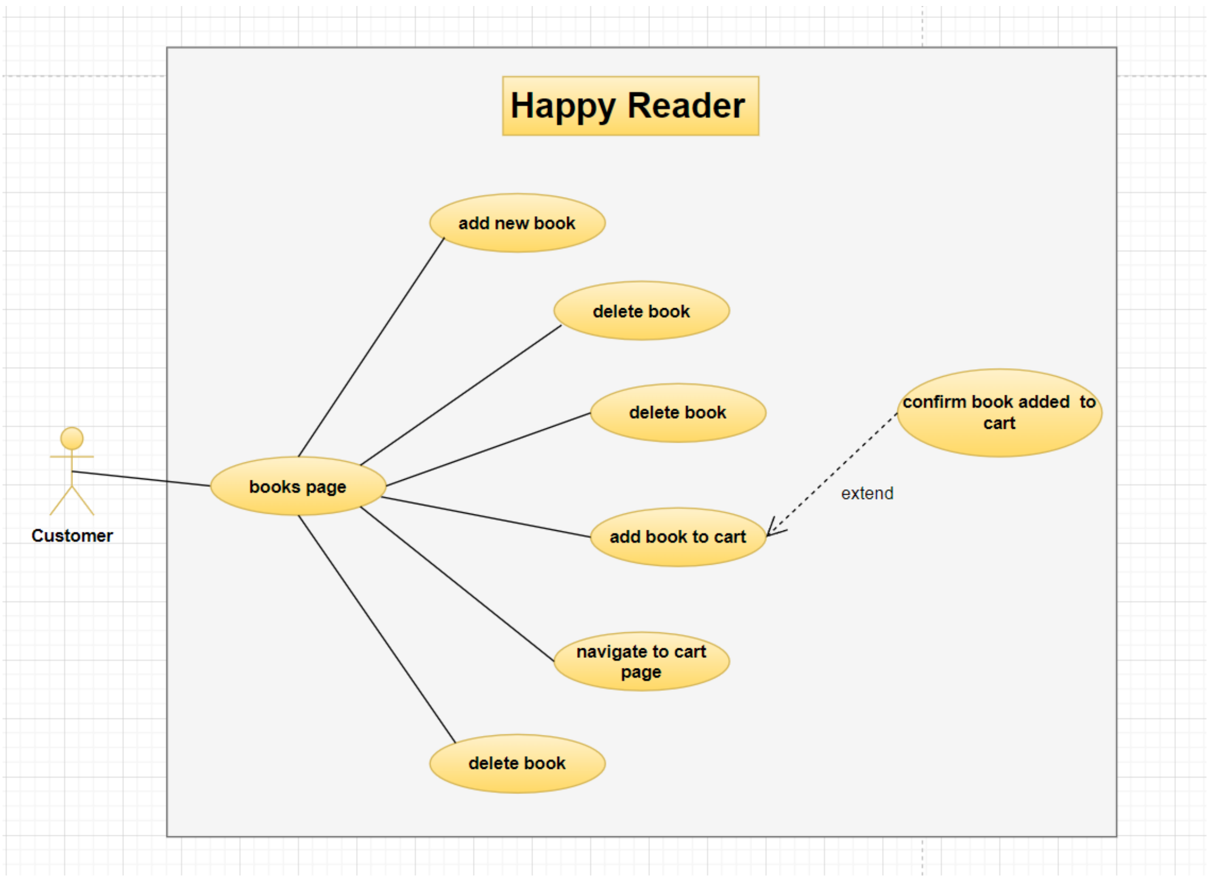
6. **Customer fills in their Information:**
   On the order page, the customer fills in their information and clicks on the "Send" button to complete the order. Next, a confirmation message will be displayed to the customer showing the customer's information and the address where the books will be sent to. Then after two seconds, the customer will be redirected to the home page.
7. **Customer visits Customer page:**
   From the home page, the customer can visit the customers page, where they can see the customers information and have the option to delete or edit those information.

By following these steps, the customer can easily navigate through the website, select books, add them to the shopping cart, review the cart, place the order, and provide the necessary information to finalize the order.


home page
books
cart
order
customers
visit home page
display website information
visit book page
fetch book data
show list of available books
add book to cart
fill customer info & send order
show that book added to cart
send customer data to customer page
display successfull order message
visit customers info
customer info/edit/delete

Use Case Diagram

The use case diagram below focuses on the features of the "**Books**" page on the website. Users can add new books, delete existing books, add books to their shopping cart, and navigate to the cart page, home page, and customers page. The diagram provides a visual representation of these actions.



Rest API

Note:

In the tables below, if an attribute creates automatically by the database and the server, then I haven't included in the POST tables.

Books API

GET requests

GET	**/books**		
Retrieves all books from the database.			
Parameters:	**Name**	**Type**	**Description**

Responses:	Code	Description/example if successful
	200	OK
	500	Failed to retrieve books from the database.
	404	No books were found in the database

GET	/books/{book_id}		
Retrieve a book by ID.			
Parameters:	**Name**	**Type**	**Description**
	Book_Id*	Integer	The ID of the book to retrieve
Responses:	**Code**	**Description/example if successful**	
	200	OK	
	500	Failed to retrieve the book from the database.	
	404	Book was not found	

POST requests

POST	/books		
Create a new book			
Parameters:	**Name**	**Type**	**Description**
	Book_name*	text	The name of the book

	Author_name*	text	The name of the author
	Category*	Text	The category of the book
	Price*	Real	The price of the book
Responses:	**Code**	**Description / example if successful**	
	201	CREATED	
	400	Required fields are missing	
	500	Failed to create a book	

PUT requests

PUT	/books/{book_id}		
Updates the information of a book with a provided ID			
Parameters:	**Name**	**Type**	**Description**
	Book_name	Text	The name of the book
	Author_name	Text	The author of the book
	Category	Text	The category of the book
	Price	Real	The updated price of the book
Responses:	**Code**	**Description / example if successful**	
	201	OK	
	500	Failed to update book	
	404	Book was not found	
	400	Cannot update the book ID	

DELETE requests

DELETE	**/books/{book_id}**		
Deletes the book with the provided id from the database			
Parameters:	**Name**	**Type**	**Description**
	Book_id*	Integer	The id of the book
Responses:	**Code**	**Description / example if successful**	
	204	NO CONTENT	
	404	Book was not found	
	500	Failed to delete the book	

Customers API

GET requests

GET	**/customers**		
Retrieves all customers from the database.			
Parameters:	**Name**	**Type**	**Description**

Responses:	**Code**	**Description/example if successful**
	200	OK
	500	Failed to retrieve customers from the database.
	404	No customers were found in the database

GET	**/customers/{customer_id}**		
Retrieve a customer by ID.			
Parameters:	**Name**	**Type**	**Description**
	Customer_id*	Integer	The ID of the customer to retrieve
Responses:	**Code**	**Description/example if successful**	
	200	OK	
	500	Failed to retrieve the customer from the database.	
	404	customer was not found	

POST requests

POST	**/customers**		
Create a new customer			
Parameters:	**Name**	**Type**	**Description**
	name*	text	The name of the book

	email*	text	The name of the author
	address*	Text	The category of the book
	Price*	Real	The price of the book
Responses:	**Code**	**Description / example if successful**	
	201	CREATED	
	400	Required fields are missing	
	500	Failed to create a book	

PUT requests

PUT	/customer/{customer_id}		
Updates the information of a customer with a provided ID			
Parameters:	**Name**	**Type**	**Description**
	Name	text	The name of the customer
	email	Text	The email of the customer
	Address	Text	The address of the customer
Responses:	**Code**	**Description / example if successful**	
	200	OK	
	500	Failed to update customer	
	404	customer was not found	
	400	Cannot update the customer ID	

DELETE requests

DELETE	/customer/{customer_id}		
Deletes the customer with the provided id from the database			
Parameters:	**Name**	**Type**	**Description**
	customer_id*	Integer	The id of the customer
Responses:	**Code**	**Description / example if successful**	
	204	NO CONTENT	
	404	customer was not found	
	500	Failed to delete the customer	

Orders API

GET requests

GET	/orders		
Retrieves all orders from the database.			
Parameters:	**Name**	**Type**	**Description**

Responses:	Code	Description/example if successful
	200	OK
	500	Failed to retrieve orders

GET	/orders/{order_id}		
Retrieve an order by ID.			
Parameters:	**Name**	**Type**	**Description**
	Id*	Integer	The ID of the order
Responses:	**Code**	**Description/example if successful**	
	200	OK	
	500	Failed to retrieve the order from the database.	
	404	order was not found	

POST requests

POST	/orders		
Create a new order			
Parameters:	**Name**	**Type**	**Description**
	Total_amount*	Real	The total amount of the books

	Book_id *	Integer	the id of the book
Responses:	**Code**	**Description / example if successful**	
	201	CREATED	
	400	Invalid book id	
	500	Failed to create an order	
	500	Failed to retrieve book	

PUT requests

PUT	**/books/{book_id}**		
Updates the information of a book with a provided ID			
Parameters:	**Name**	**Type**	**Description**
	Order_date	Date	The date of the order
	Total_amount	Real	The total amount of books price
	Book_id	Integer	The id of the book
Responses:	**Code**	**Description / example if successful**	
	204	NO_CONTENT	
	500	Failed to update the order	
	404	order was not found	
	400	Cannot update Order ID	

DELETE requests

DELETE	**/orders/{order_id}**		
Deletes the order with the provided id from the database			
Parameters:	**Name**	**Type**	**Description**
	order_id*	Integer	The id of the order
Responses:	**Code**	**Description / example if successful**	
	204	NO_CONTENT	
	404	order was not found	
	500	Failed to delete the order	