

INTERNATIONAL BURCH UNIVERSITY

Faculty of Engineering and Natural Sciences

Department of Information Technology



Software Engineering

Bookaholics Anonymous

Proposed to:

Nermina Durmić, Assist. Prof. Dr

Aldin Kovačević, Teaching Assistant

Sarajevo, 14rd of March 2023

Dženeta Džananović, 20002

Adnan Džindo, 20002452

Table of Contents

Table of Contents	2
1. Introduction	3
1.1 About the project	3
1.2 Project Functionalities and Screenshots	4
2. Project Structure	6
2.1 Technologies	6
2.2 Database Entities	6
2.3 Design Patterns	6
2.4 Tests	9
3. Conclusion	10

1. Introduction

1.1 About the project

We embarked on the development of a bespoke internal web application, meticulously designed to cater to the unique needs of a Bookstore. Our paramount objective was to create an intuitive and efficient tool that empowers bookstore employees with the ability to swiftly retrieve specific book information and check its real-time availability. Furthermore, this web application streamlines the process of placing orders for new books, seamlessly integrating them into the existing inventory. By doing so, we aimed to optimize the restocking procedure and ensure a comprehensive and up-to-date inventory.

Throughout the development process, we meticulously crafted a user-centric experience that aligns with the requirements of bookstore operations. Our web application provides a user-friendly interface, enabling bookstore employees to easily search and access detailed information about specific books. They can effortlessly navigate through the system and verify the availability of books in real-time, allowing them to provide accurate and prompt assistance to customers. Moreover, our web application incorporates a seamless order placement feature, facilitating the addition of new books to the inventory. By simplifying and automating this process, bookstore staff can efficiently manage their restocking needs. This functionality eliminates manual interventions, reduces the chances of errors, and ensures that the inventory remains comprehensive and up-to-date.

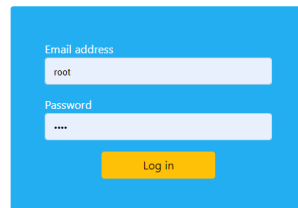
By developing this internal web application tailored for a Bookstore, we aimed to enhance operational efficiency, improve customer service, and streamline inventory management. Our solution empowers bookstore employees with a powerful tool that enables quick access to book information, real-time availability checks, and seamless order integration. Through the implementation of intuitive functionalities, we sought to optimize the day-to-day operations of the bookstore, ultimately providing an enhanced experience for both the staff and customers.

Github Link: <https://github.com/Adnan251/Book-Storte-Web-App>

Web App Link : <https://bookaholic.onrender.com>

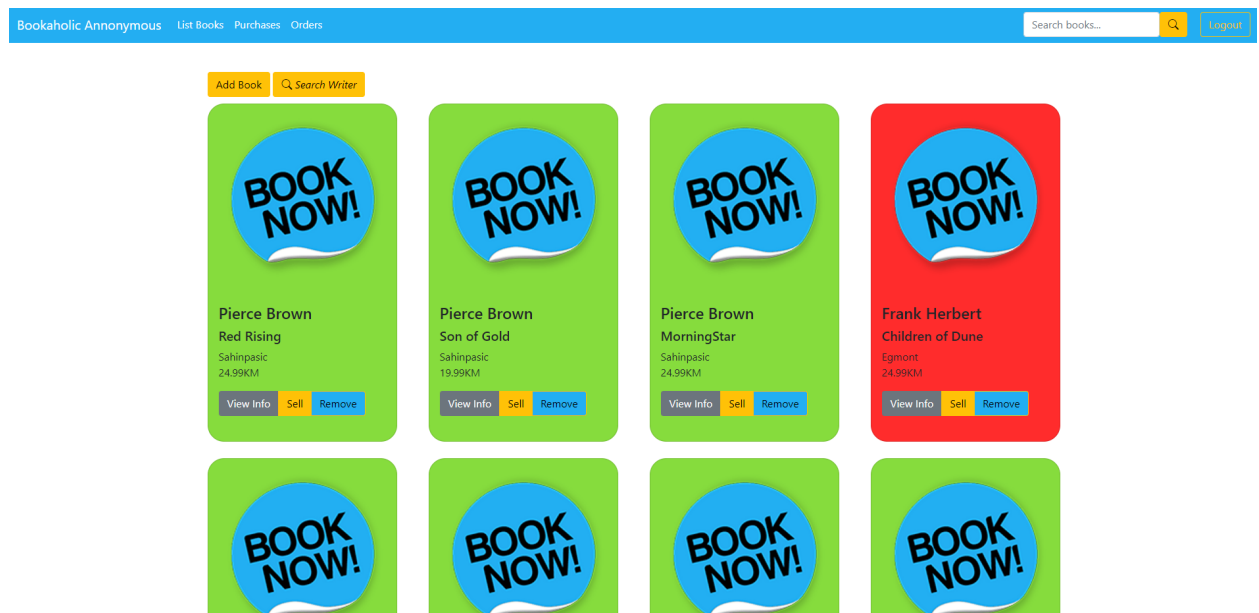
1.2 Project Functionalities and Screenshots

- Login area (email: adnan.dzindo@stu.ibu.edu.ba , password: 12345)



A login form with a blue background. It contains two input fields: 'Email address' with the text 'root' and 'Password' with masked characters '....'. Below the fields is a yellow 'Log in' button.

- Users can check the information about a specific book and update it
- Filtering of Books by name and author
- Adding Books



- Adding, Editing and Deleting Orders

Bookaholic Anonymous List Books Purchases Orders							Search books...	🔍	Login
Add Order									
	id	Book	Order amount	Order price	Date of order	Date of delivery	Ordered by		
Edit Delete	19	Harry Potter And the Philosophers Stone	20	499.8 KM	2023-06-09	2023-06-14	Dženeta Džananović		
Edit Delete	27	Son of Gold	15	299.85 KM	2023-06-14	2023-06-21	Adnan Džindo		
Edit Delete	28	Dune Messiah	20	499.8 KM	2023-06-14	2023-06-22	Adnan Džindo		
Edit Delete	35	Death on the Nile	22	329.78 KM	2023-06-08	2023-06-30	Adnan Džindo		

- Checking the purchases made

Bookaholic Anonymous List Books Purchases Orders							Search books...	🔍	Login
	id	Book	Purchase amount	Time of purchase	Date of purchase	Sold by			
	11	Heretics of Dune	24.99KM	2023-06-11 22:16:13	2023-06-11	Dženeta Džananović			
	12	Heretics of Dune	24.99KM	2023-06-11 22:16:52	2023-06-11	Dženeta Džananović			
	13	Heretics of Dune	24.99KM	2023-06-11 22:18:11	2023-06-11	Dženeta Džananović			
	14	Heretics of Dune	24.99KM	2023-06-11 22:18:30	2023-06-11	Dženeta Džananović			
	15	Heretics of Dune	24.99KM	2023-06-11 22:19:14	2023-06-11	Dženeta Džananović			
	16	Heretics of Dune	24.99KM	2023-06-11 22:19:50	2023-06-11	Dženeta Džananović			
	17	Heretics of Dune	24.99KM	2023-06-12 00:20:50	2023-06-12	Dženeta Džananović			
	18	Heretics of Dune	24.99KM	2023-06-12 00:25:10	2023-06-12	Dženeta Džananović			
	19	Red Rising	24.99KM	2023-06-12 22:54:40	2023-06-12	Adnan Džindo			
	20	Red Rising	24.99KM	2023-06-12 22:55:02	2023-06-12	Adnan Džindo			
	21	Red Rising	24.99KM	2023-06-12 23:00:01	2023-06-12	Adnan Džindo			
	22	Red Rising	24.99KM	2023-06-12 23:01:15	2023-06-12	Adnan Džindo			
	23	Son of Gold	19.99KM	2023-06-12 23:11:58	2023-06-12	Adnan Džindo			
	24	Catcher in the Rye	21.99KM	2023-06-12 23:49:15	2023-06-12	Adnan Džindo			
	25	Heretics of Dune	24.99KM	2023-06-13 17:23:35	2023-06-13	Dženeta Džananović			
	26	Red Rising	24.99KM	2023-06-13 17:25:29	2023-06-13	Dženeta Džananović			
	27	Nine stories	21.99KM	2023-06-13 14:19:36	2023-06-13	Adnan Džindo			
	28	Son of Gold	19.99KM	2023-06-13 14:19:36	2023-06-13	Adnan Džindo			
	29	Harry Potter And the Chamber of Secrets	24.99KM	2023-06-13 14:19:36	2023-06-13	Adnan Džindo			

2. Project Structure

2.1 Technologies

In this project we will use PHP for the Backend logic with Flight PHP as the framework, we will also use HTML, CSS and JavaScript for the frontend and we will use Ajax calls that will enable us to update specific parts of the page instead of the whole page. And finally for the database we will use MySQL to store all of our data. We will also use GitHub for version control.

And finally we will use the PSR - 12 coding standard for PHP. We will use the RESTful API to communicate between the frontend and the backend.

2.2 Database Entities

Our DataBase will have the following Entities:

- books - stores all the book information
- orders - stores information about the orders made for new books
- publishers - stores information about the publishing houses
- purchase - stores information about purchases that have been made
- users - stores basic user information that are used for logging in
- writers - stores basic information about the authors of the books
- booksandwriters - this is a table that is used to connect the books table with the writers table because of the many to many relationship between them

2.3 Design Patterns

In this project we decided to use two design patterns:

a) Front Controller Pattern

The Front Controller Pattern serves as an effective solution when we aim to establish a centralized entry point for our web application, efficiently managing all incoming requests. Our team successfully implemented this pattern in our "index.php" file, where we seamlessly consolidated all our code, enabling a modular approach to our application's design.

By adopting the Front Controller Pattern, we achieved several benefits. Firstly, it provided a single, unified location to handle all incoming requests, ensuring a streamlined and consistent processing flow. This centralization facilitated easier maintenance and allowed for the implementation of cross-cutting concerns, such as security checks and logging, at a single point.

Moreover, the pattern empowered us to create a modular codebase, as we could neatly integrate different components and functionalities into the front controller. This modular structure enhanced code organization and reusability, making it simpler to add or modify features in our web application without the need for extensive modifications across multiple files.

Overall, the Front Controller Pattern's adoption in our "index.php" file demonstrated its effectiveness in establishing a centralized control point for our web application, fostering modularity, maintainability, and a coherent request handling mechanism.

b) Singleton Pattern

The Singleton Pattern is a fundamental design pattern that provides a powerful mechanism to ensure the existence of only one instance of a specific class throughout an application. By restricting the instantiation of a class to a single object, the Singleton Pattern offers numerous advantages in terms of control, resource management, and consistency.

In our development project, we recognized the immense utility of the Singleton Pattern, particularly in our dao (data access object) layer. By applying this pattern, we achieved a central repository of database entities, where each entity class had a single instance. This singleton approach allowed us to efficiently manage and reuse these instances whenever they were needed, promoting optimal resource utilization.

One of the notable benefits of the Singleton Pattern is its ability to maintain a consistent state across the application. Since there is only one instance of the class, any changes made to its internal state are instantly reflected throughout the entire system. This consistency eliminates the risk of having multiple instances with conflicting data and ensures that all parts of the application operate on the same set of information.

Furthermore, the Singleton Pattern promotes encapsulation by encapsulating the creation and management of the single instance within the class itself. This encapsulation not only simplifies the usage of the class but also reduces the risk of creating multiple instances unintentionally. It provides a clear and standardized approach to access the instance, ensuring that all parts of the codebase interact with the same object.

In our case, the Singleton Pattern enabled us to establish a reliable and efficient connection between our application and the underlying database. By having a single instance of each database entity, we minimized the overhead associated with establishing and tearing down connections repeatedly. This optimization significantly improved the performance and responsiveness of our application, especially during high-traffic scenarios.

In summary, the Singleton Pattern proved to be an invaluable asset in our project, facilitating the creation and management of a single instance of a specific class. By adopting this pattern in our dao layer, we reaped the benefits of improved resource utilization, consistent data access, encapsulation, and enhanced performance. The Singleton Pattern continues to be a widely used and highly regarded pattern in software development, delivering robustness and efficiency to applications across various domains.

2.4 Tests

During the course of our project, our testing approach primarily relied on manual testing rather than extensive automated testing. We found manual testing to be advantageous because it provided us with immediate visibility into the changes made to the web application as soon as the code was modified. This allowed us to quickly observe and verify the behavior and visual aspects of the application in real-time.

While manual testing served as our primary testing method, we did incorporate Postman into our testing workflow. Postman, a popular API testing tool, proved to be invaluable for testing the functionality and correctness of our routes. With Postman, we could conveniently send requests to our web application's API endpoints and validate that they performed as intended, returning the expected responses and behaving appropriately.

While automated testing is typically recommended for ensuring comprehensive test coverage and enabling regression testing, our project's specific circumstances and requirements favored manual testing. The nature of our web application, coupled with the need for immediate feedback on changes, made manual testing a practical choice. Additionally, the usage of Postman for API route testing allowed us to verify critical functionalities efficiently.

It is worth noting that although manual testing has its benefits, it does have limitations. It heavily relies on the tester's expertise and may not cover all edge cases or scenarios. Introducing automated testing, such as unit tests and integration tests, can complement manual testing, providing a more systematic and thorough verification of the application's functionality.

In future projects or iterations, considering the advantages of automated testing, we might explore implementing a more robust test suite that includes both automated and manual testing approaches. This would help ensure the reliability, stability, and overall quality of our web application while still benefiting from the immediacy and visual feedback provided by manual testing.

3. Conclusion

In conclusion, our project involved the development of an intuitive internal web application tailored for a Bookstore. The application successfully addressed the need for efficient book information retrieval, real-time availability checks, seamless order placement, and integration into the existing inventory. Through the adoption of appropriate technologies, such as PHP, Flight PHP framework, HTML, CSS, JavaScript, and MySQL, we were able to deliver a functional and user-friendly web application.

During the development process, we encountered several challenges, including difficulties in connecting the frontend and backend components. Ensuring smooth communication and data exchange between these two layers required careful coordination and troubleshooting. Despite these challenges, we were able to overcome them and establish a functional and cohesive integration.

In conclusion, our project successfully delivered a straightforward internal web application for the Bookstore, providing essential functionalities and addressing the specific needs of the business. The adoption of the Front Controller and Singleton Patterns, along with our diligent problem-solving efforts, allowed us to create a well-structured, modular, and efficient application. As with any development project, we acknowledge the importance of continuous learning and improvement, and we remain open to future enhancements and advancements in our application to further enhance its functionality, performance, and user experience.