UNIVERSITY OF REGINA

Final Project Report:

U of R Buy & Sell

Department of Science, University of Regina

Submitted by : Ibrahim Adogba | Wilbur Dulce | James Sieben

CS 476 - Winter 2023

March 20, 2023

Table of Contents

Problem Definition

The cost of post-secondary education continues to rise, and the cost of textbooks follows suit. Students may find it difficult to browse and locate the books they need on existing online markets, as these are not tailored to their specific needs. For students, the lack of an online marketplace for buying and selling used textbooks could result in increased prices and hassle. We believe students should have an affordable way of acquiring textbooks. Paying a premium for a brand new textbook for an elective class is not ideal. In order to address these issues, our group is creating the "U of R Buy & Sell" web application. This platform will provide students with a simple and practical way to purchase and sell secondhand books, allowing them to have more control over prices. Compared to the school bookstore, which sets its own prices for buying back books, only buys books they deem to be in good condition, and sells books for a premium, the "U of R Buy & Sell" website will be a more cost-effective and eco-friendly solution for obtaining textbooks. This platform will be beneficial for university students, providing them with a convenient way to buy and sell used books.

Application Benefits

This section will discuss the advantages of having a website or application for University of Regina (U of R) students to buy and sell textbooks. By comparing it to more common retail applications, we will list the benefits for students.

Benefits Compared to Facebook Marketplace

Firstly, our application is limited to only U of R students, as users must register with their University of Regina email. This creates a safe environment for students and gives users the confidence that they are interacting with other U of Regina students. Secondly, our application is limited to textbooks and other classroom materials, making transactions easier and more efficient. Lastly, users are given a user ID which is visible to other users. This ID is made up of the first six characters of their university email, so their name and other private information remain anonymous.

Summary of Benefits Compared to Facebook Marketplace

- Limits users to U of R students
- Limits items to textbooks and other classroom materials
- User ID keeps personal information anonymous

Benefits Compared to Kijiji Web Application

Firstly, Kijiji is very aggressive in promoting paid advertisements, which makes the user experience poor. To prevent this, our web application offers an advertisement-free environment, making the user experience more enjoyable. Secondly, the largest age group of Kijiji visitors are aged 24-34. This implies only a small portion of users are in university, which could discourage university students from using Kijiji. By limiting our scope to university students only, our web application becomes more appealing to that demographic.

Summary of Benefits Compared to Kijiji

- More relevant user base
- Less ads
- Safer environment

Benefits Highlighted

Having discussed the benefits of our application in comparison to other notable applications, we can now discuss the benefits this application has to offer University of Regina students.

- **Saving Cost**: One way to lower the cost of textbooks, which can be quite expensive for students, is to create a website that facilitates the buying and selling of books. For example, a student who needs a specific textbook for a class could purchase it used from another student on the website, potentially saving a significant amount of money compared to buying it brand-new from a store.
- **Convenience:** For students, our buying and selling websites can make it much easier to get the textbooks they require. They can utilise the website to quickly identify available books and organise a transaction, saving them from having to visit multiple bookstores or search for vendors on social media or other online platforms.
- **Sustainability**: The website's capacity to enable students to purchase and sell second-hand textbooks can help reduce waste and promote sustainability. It can be beneficial for the environment when students sell their used textbooks to other students, as this keeps them out of the trash or recycling.

- **Community Building & Income Generator**: An application like ours, which facilitates the buying and selling of used books, can create a sense of community among the University of Regina students. These connections built from selling and buying textbooks help students who no longer need their used textbooks to generate extra income. For instance, one student who has recently finished a course could be able to sell their textbook to another student who is going to enrol in the same course.

Conclusion

In conclusion, our service, which allows students to purchase and sell books, provides a variety of advantages to students, from cost savings and convenience to sustainability and community building. The website can help to create a more supportive and collaborative learning environment by allowing students to interact with each other and exchange items.

Requirements Elicitation and Specification

In this section, we will discuss the functional, non-functional, user and system requirements needed to create a website that allows university students to buy and sell used textbooks. We will gather, analyse, and document the requirements for U of R Buy & Sell. We will identify the stakeholders and understand the specific requirements and constraints of our application.

Functional Requirements

In order to satisfy the expectations of stakeholders, a software system must meet certain predefined characteristics and functionalities. These are known as functional requirements and

are often described in terms of inputs, processes, and outputs. They indicate what the system should accomplish. For our application, the functional requirements would be:

Buyers

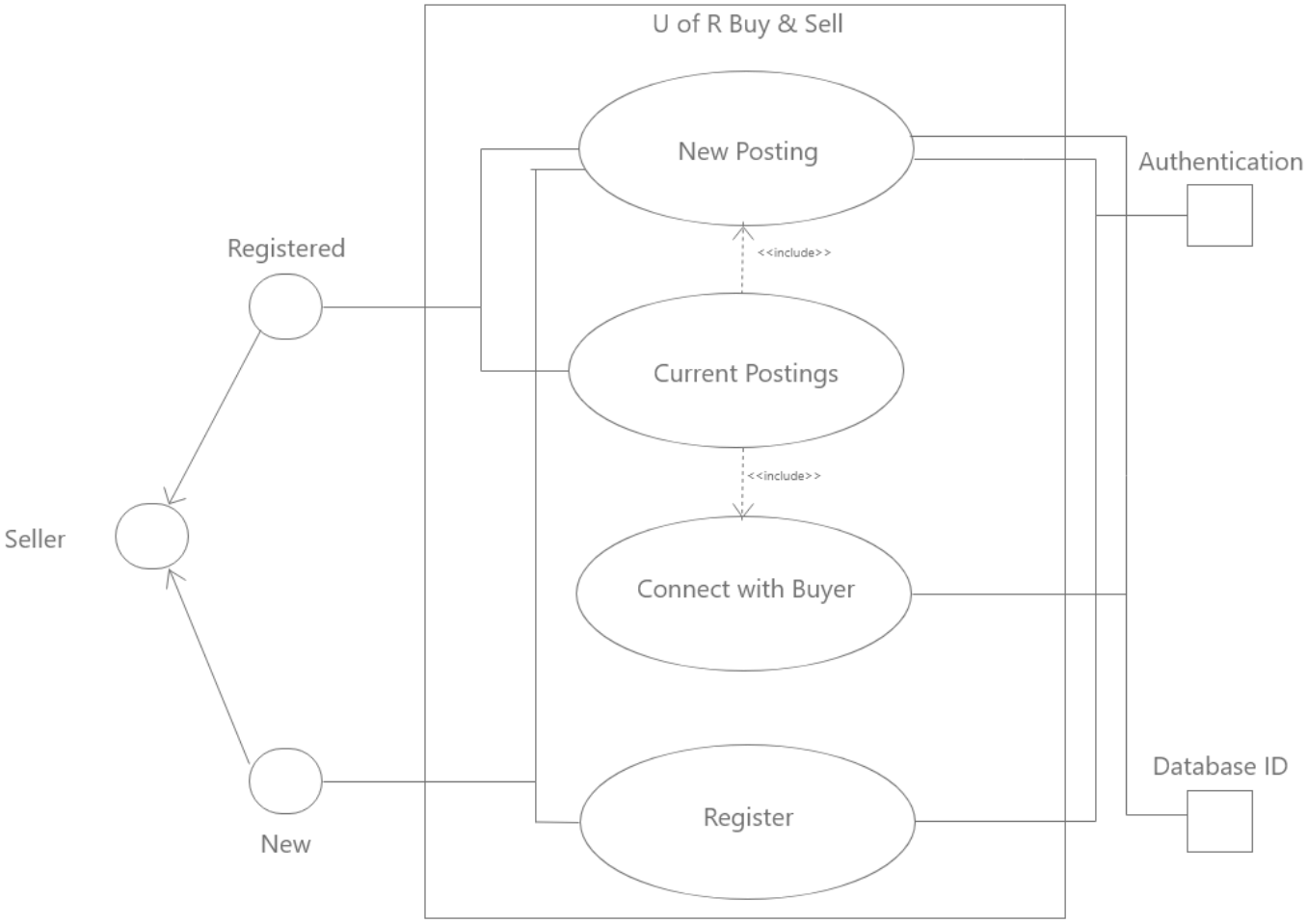
- Users should be able to search for books by title, author, or ISBN.
    - Users should be able to freely navigate any listings as long as they are a registered user.
- Users should be able to view book listings.
    - Users should be able to freely access all the necessary information for specific listings as long as they are a registered user.
- Users can message one another via email.
    - Users should be able to freely access the contact information of the seller for specific listings as long as they are a registered user.
- Users should be able to get notified if they get a message.
    - Users should be able to get notified if they receive a message from another user they have contacted.

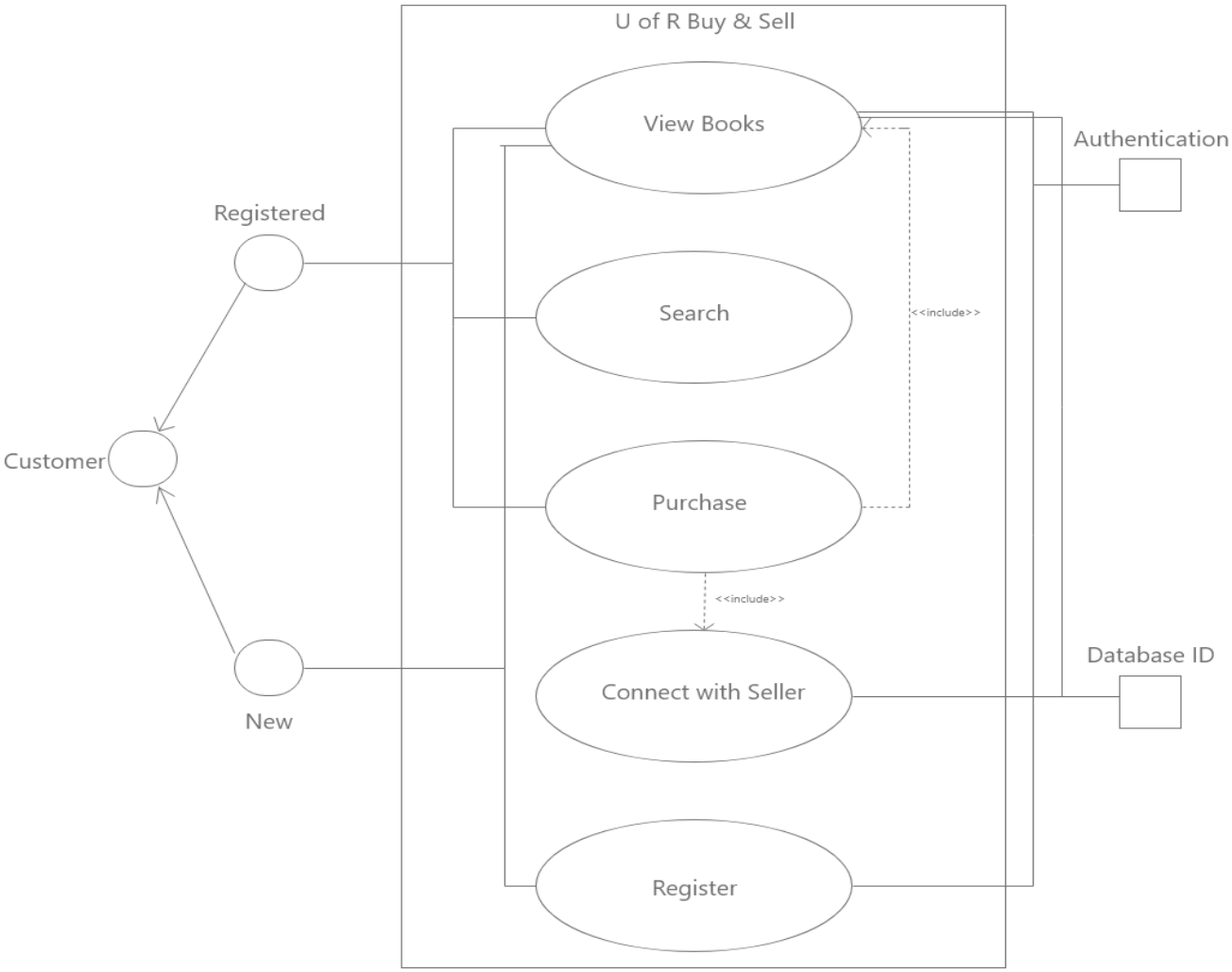
Sellers

- Users should be able to post listings for books they want to sell.
    - Registered users should be able to create listings for books that they want to sell that includes a picture of the book, ISBN, Title, and Author of the book.
- Users should be able to view books they’ve posted.
    - Registered users should be able to view and make changes to their own listings such as changing the status of their listings to available, pending, or sold.

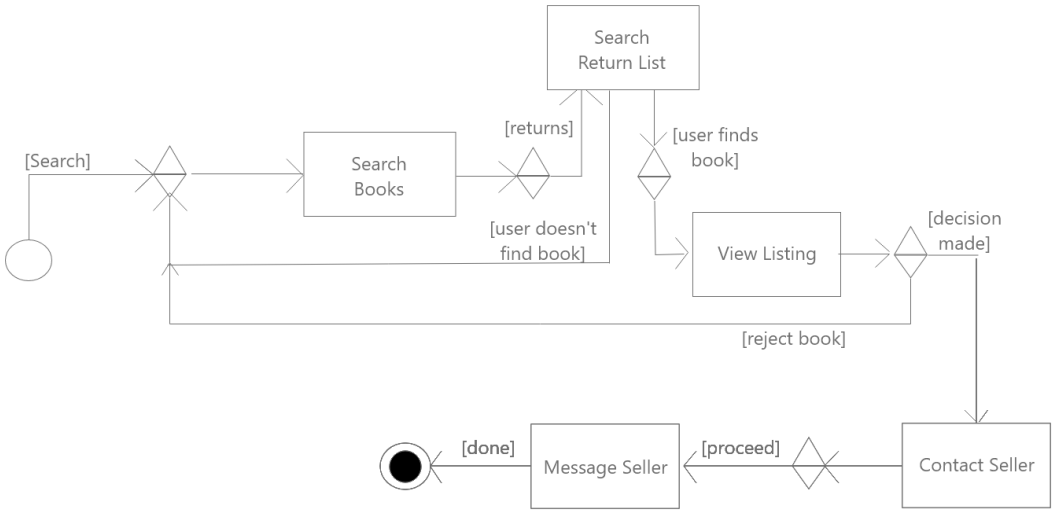
- Users should be able to delete any posts they have created.
    - Registered users should be able to remove their own listings when its no longer available for sale.
- Users should be able to get notified when they get a message.
    - Users should be able to get notified if they receive a message from another user they have contacted.
- Users can message one another via email.

Use Case Diagrams

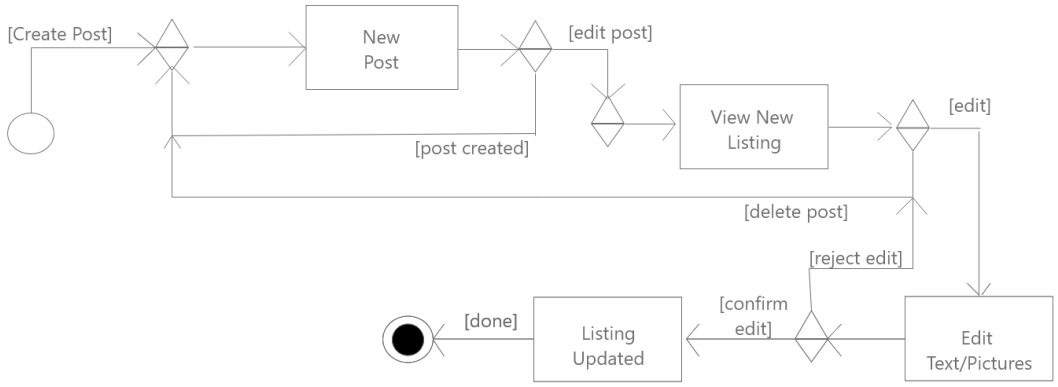




Activity Diagrams



The first diagram illustrates the process when a registered user searches for a book. If the book is not found, the user is taken back to the search page. If the book is located, the user is directed to the listing page for that book. The user then has the option to purchase or decline the post. If the user chooses to buy the book, they are prompted to contact the seller and the web app connects the two parties.



The second diagram depicts the scenario in which a registered user edits a newly uploaded listing. Upon adding the post, the user has the option to edit it. Should they choose to, they will be presented with the listing, enabling them to make changes to it, provided the listing belongs to them. Once the edits are confirmed, the listing is updated.

Non-Functional Requirements

Non-functional requirements are characteristics of exceptional quality that a software system must possess. They delineate the system's expected level of performance in terms of its general behaviour, traits, and features. For our application, some of the non-functional requirements could include:

- **Performance**: Even during periods of high traffic, our website should load quickly and respond promptly to user interaction.
- **Security**: Our website should be secure, incorporating safeguards to protect user information and prevent unauthorised or malicious access. This can be achieved through our secure login and sign-up page.
- **Maintainability**: The website should be created and developed to be easily updatable and maintainable, without compromising the user experience. This will be achieved through well-commented, detailed code and appropriate documentation.
- **Usability**: Our goal is to make our application easy to use by adopting a similar creative process to most ecommerce websites for a familiar experience.

User Requirements

We will ensure that our application allows that:

- Users can view the important information about a book.
- Buyers can communicate with sellers.
- Users have the ability to add items to a favourites tab.

Software Qualities

Correctness

Our website application will only allow University of Regina students to register using their University email, reducing phishing and impersonation risk, and decreasing the load on the web app.

Time-Efficiency

We as developers will ensure our website application only serves one purpose: aiding students to buy and sell used textbooks. We will only implement necessary requirements to improve efficiency, and we won't allow any advertising to reduce server load and improve user experience by reducing clutter. This will ensure users have a smooth, swift experience.

Robustness

We will ensure our website application is robust by putting the general public first, following design software patterns learned throughout the course.

Top & Low Level Software Design

In this section, we will discuss the MVC architecture used on our web project as well as the Factory and Observer design patterns we used. We will also include class diagrams for further clarity.

MVC Architecture

Controller

The Controller is the logic layer of the application, responsible for processing user input and directing the data flow between the Model and View. In our web app, the Controller is responsible for handling requests to view books, upload books, and communicate with other users, as well as managing the database to store and organise user and book information.

When a user buys a book, the Controller updates the Model by removing the book from the database and updates the View so that book isn't shown on the website. Similarly, when a new user is created, the Controller adds the user to the Model and updates the View so the new user has a unique account page and shopping cart.

Model

The Model is the data layer of the application, responsible for storing, retrieving, and manipulating the data stored in the application's database. For our buy and sell web app, the model stores information about the books and users, such as the book titles, authors, descriptions, and usernames, emails, and addresses. Our web app uses MySQL to store data and JSON requests to save or retrieve data.

View

The View is the user interface of the application. This is where the user interacts with the application and can view the data stored in the Model. In our web app, the View is composed of HTML, PHP, and CSS and is responsible for displaying the webpages of the app as well as book information, other user profiles, and the user's saved information like the favourites tab.

Benefits

- Increased flexibility and scalability: By separating our application layers into distinct components, it becomes easier to maintain and modify existing code, as well as add new features to the application.

- Improved code reusability and maintainability: By using the MVC pattern, we can quickly identify and reuse core components that are used in multiple parts of the application. This reduces development time and helps maintain code consistency.
- Separation of concerns: MVC architecture encourages the separation of applications into distinct components, which makes it easier to manage and debug. This separation also helps to ensure that changes in one part of the application doesn't have an unintended consequence on another part.

Design Patterns

Observer

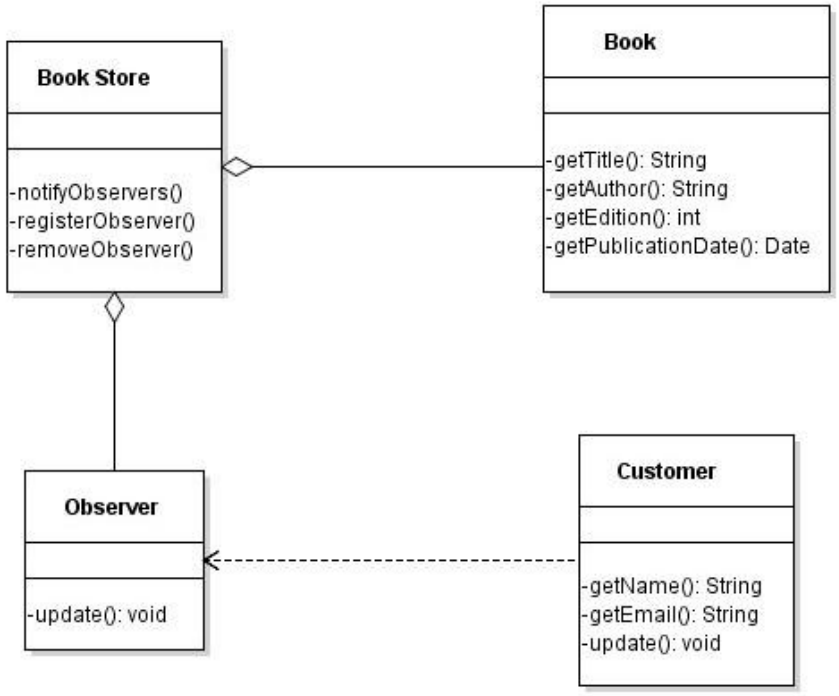
The Observer design pattern is used in this web application to notify users of changes in the database. The Observer pattern allows the application to be more reactive and responsive to user input. For example, when a user uploads a book, all other users that are interested in that book can be notified via email. This way, users can quickly respond to the new book listing. Additionally, the Observer pattern can be used to alert users when a book in their favourites tab has been sold.

Factory

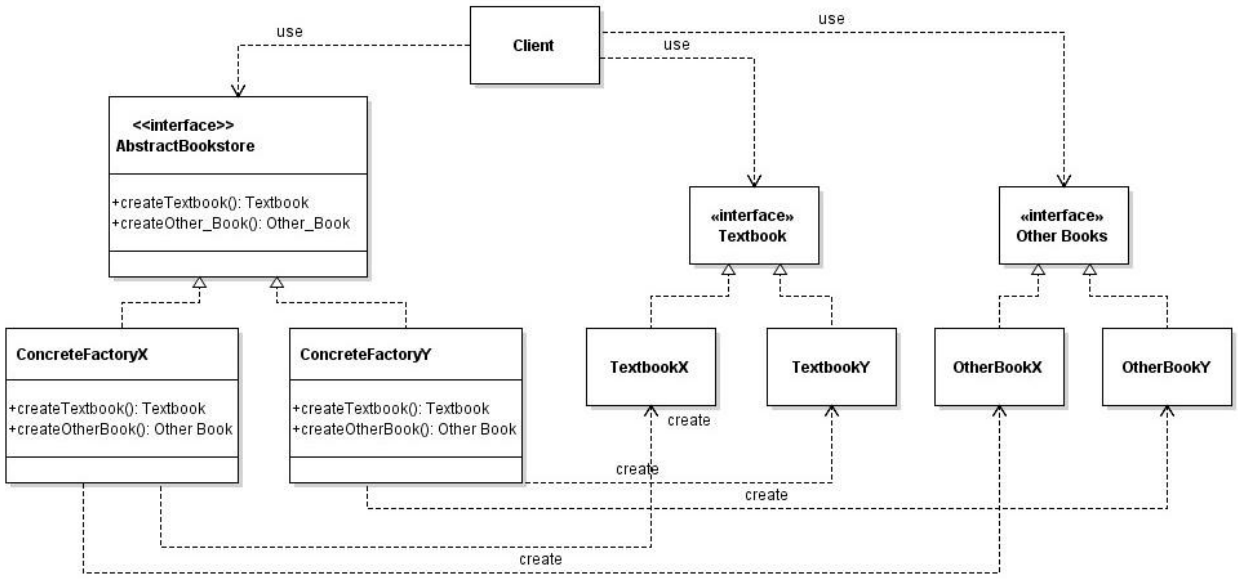
The Factory design pattern is used to create objects from classes It helps to keep the code clean and organised and makes creating and destroying objects simpler and safer. In our web app, this design pattern is applied when creating book and user objects. A book class contains information like the title, author, edition, and ISBN which makes it easier for users to search for

specific books. When a new book is uploaded, a book creation function is called which makes a new book object. Similarly, a user class contains information like the username, password, and email. A new user object is created when a new user signs up and the info is sent to the database.

Class Diagrams

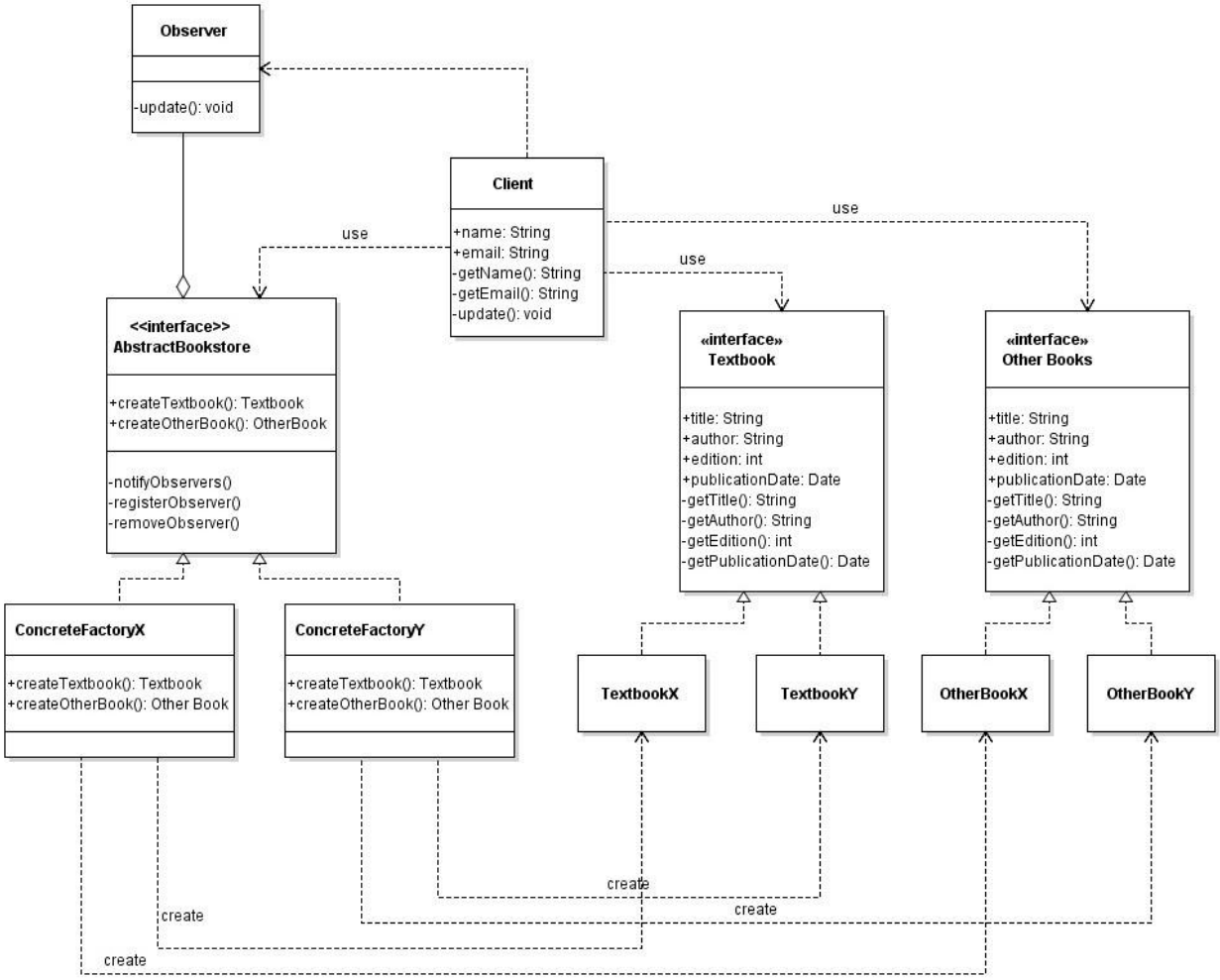


In the Observer method, the pattern's important methods are register/remove observer and notify observer. The corresponding algorithms would be to add a new observer, or remove an old one, to/from the list of observers for an object (book) and to notify all observers for an object when the state of the object has changed. For our website, a state change may be a book becoming available or a book being sold. Sellers automatically have an observer for books they post while buyers can be added to a list of observers by adding a book to their favourites tab.



In the factory method, the pattern's important methods are creating a new entry for the database, removing objects from the database, and allowing buyers to buy books from sellers. The corresponding algorithm is that when a new user signs up or a user uploads a book to sell, a new entry is created in the database so that the user can interact with the website and the new book can be displayed. In addition, when a user deletes their account or a book has been sold, the database can be updated accordingly. Finally, when a buyer buys a book, the database can be parsed to provide the buyer with the seller's email for the sale.

This diagram combines the factory and observer design patterns:



Software Construction

The following section will provide screenshots and information regarding our software structure, hardware configuration, database tables. It will also provide links to our GitHub and individual pages of our website.

Screenshots

Database tables:

mysql> DESCRIBE favorites;

Field	Type	Null	Key	Default	Extra
favoriteId	int	NO	PRI	NULL	auto_increment
sellId	int	NO	MUL	NULL	
userId	int	NO	MUL	NULL	
created_dt	datetime	NO		NULL	

4 rows in set (0.01 sec)

mysql> DESCRIBE Users;

Field	Type	Null	Key	Default	Extra
userId	int	NO	PRI	NULL	auto_increment
email	varchar(50)	NO		NULL	
photo_img	varchar(10000)	NO		NULL	
pswd	varchar(50)	NO		NULL	
username	varchar(50)	NO		NULL	

5 rows in set (0.00 sec)

mysql> DESCRIBE sellcreation;

Field	Type	Null	Key	Default	Extra
sellId	int	NO	PRI	NULL	auto_increment
selltitle	varchar(100)	YES		NULL	
sellauthor	varchar(100)	YES		NULL	
sellisbn	varchar(100)	YES		NULL	
sellclassnum	varchar(100)	YES		NULL	
sellimage	varchar(10000)	YES		NULL	
selldescription	varchar(600)	YES		NULL	
price	varchar(100)	YES		NULL	
created_dt	datetime	YES		NULL	
userId	int	NO	MUL	NULL	

10 rows in set (0.00 sec)

mysql> SELECT * FROM Users;

userId	email	photo_img	pswd	username
1	jobbench@ngitservices.com	uploads/LAAPPY.JPG	Dante12@	ioa388
2	olayinkaadogba@gmail.com	uploads/goblet.JPG	Olayin1@	favou1
16	ibrahimadogba@yahoo.com	uploads/p1.jpeg	L8@lohj6	Drake
17	ibrobaski112@yahoo.com	uploads/p2.jpg	K5#tghp7	Lewis
18	adogbaibrahim@yahoo.com	uploads/p4.png	J9^fghl2	Lee
19	ioa388@uregina.ca	uploads/p4.jpg	K5#tghp7	Tayler

6 rows in set (0.00 sec)

mysql> SELECT * FROM favorites;

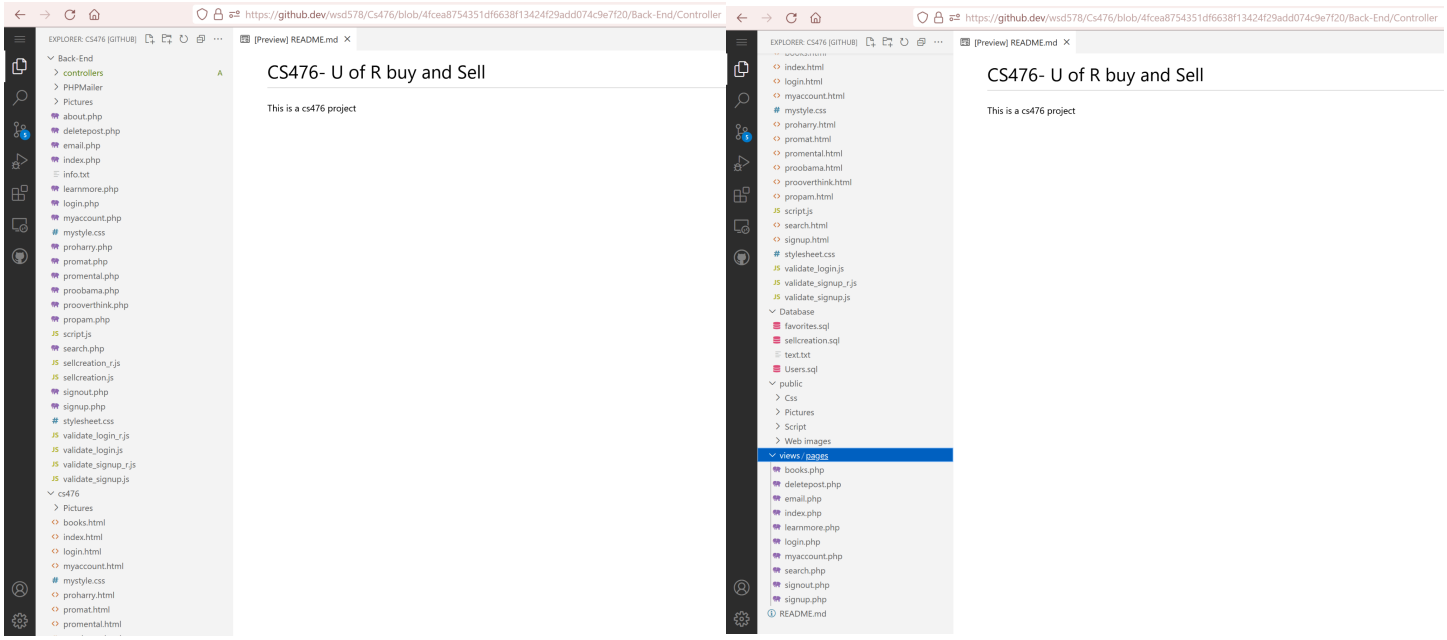
favoriteId	sellId	userId	created_dt
1	14	1	2023-03-17 18:24:03
2	14	1	2023-03-17 18:26:33
3	13	1	2023-03-17 18:26:44
4	14	1	2023-03-18 11:10:30
5	14	1	2023-03-18 12:49:26
6	14	16	2023-03-18 12:55:05
7	13	16	2023-03-18 12:55:44
8	14	16	2023-03-18 12:56:07
9	12	16	2023-03-18 12:57:11

9 rows in set (0.00 sec)

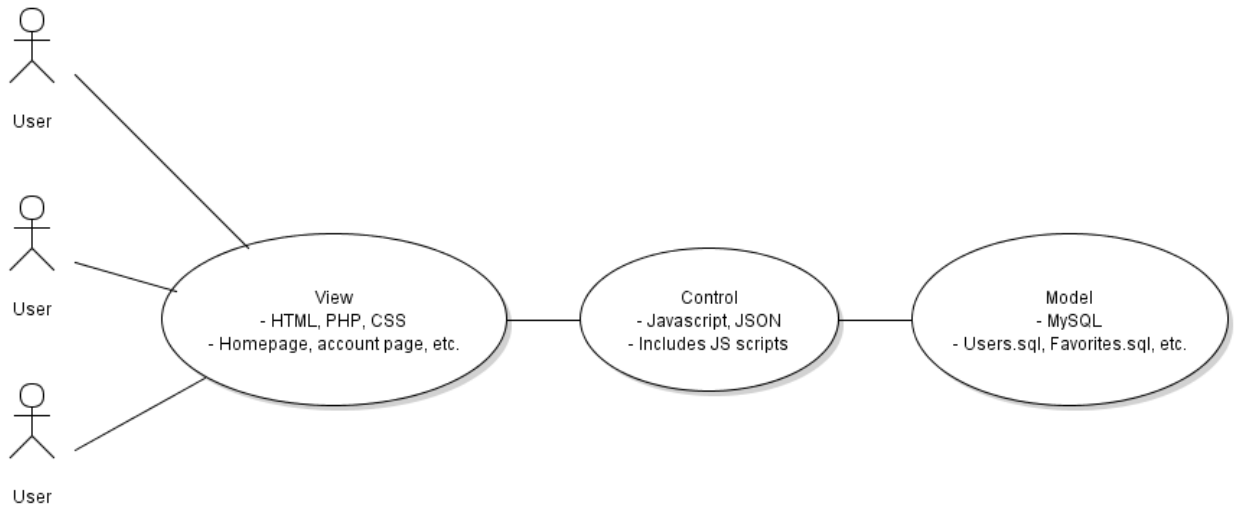
mysql> SELECT * FROM sellcreation;

sellId	selltitle	sellauthor	sellisbn	sellclassnum	sellimage	selldescription	price	created_dt	userId
1	Prince Harry Spare	Prince Harry	1234356781	NULL	uploads/Harry.jpg	It was one of the most searing images of the twentieth century: two young boys, two princes, walking behind their mothers coffin as the world watched in sorrow—and horror. As Princess Diana was laid to rest, billions wondered what Prince William and Prince Harry must be thinking and feeling—and how their lives would play out from that point on.	135	2023-03-14 13:24:17	1
2	The Menatal Toughness	F. Scott Fitzgerald	978-0141182636	CLASS123	Pictures/books/mental toughness.jpg	A classic novel about the decadence of the Jazz Age	9.99	2023-03-14 13:24:17	2
8	Freakonomics: A Rogue Economist	Steven Levitt	9780060555665	NULL	uploads/51TtLwUETPL._AC_UL900_SR615,900_.jpg	A non-fiction book that uses economic principles to explain a variety of social phenomena.	78	2023-03-15 11:50:21	1
9	Biology: Concepts and Connections	Campbell and Reece	978-0321489845	NULL	uploads/biology..jpg	A comprehensive textbook that covers everything from the basics of cellular biology to the complexities of ecology and biodiversity. A must-have for any biology student or enthusiast.	97.86	2023-03-16 20:44:42	16
10	The Norton	Kelly J. Mays	9780393911640	NULL	uploads/norton.jpg	A classic introduction to literary theory and analysis, featuring a wide selection of poems, plays, and short stories from around the world. Perfect for English majors and anyone interested in exploring the power of language and storytelling.	50	2023-03-16 20:45:52	16
11	Calculus: Early Transcendentals	James Stewart	978-1305616691	NULL	uploads/calculus.jpg	A clear and concise guide to calculus, featuring numerous real-world applications and examples. Essential for anyone studying math, physics, or engineering.	40	2023-03-16 20:48:15	17
12	American Government:	James Q. Wilson and John J. DiIulio	978-1305500062	NULL	uploads/american.jpg	A comprehensive guide to macroeconomic theory and policy, featuring real-world examples and case studies. A must-read for anyone interested in the economy and how it affects our daily lives.	20	2023-03-16 20:50:44	17
13	Organic Chemistry	Paula Yurkanis Bruice	9780321803221	NULL	uploads/organic.jpg	A detailed and thorough guide to the principles of organic chemistry, featuring a strong focus on real-world applications and problem-solving. A great resource for anyone studying chemistry or pursuing a career in science.	35	2023-03-16 20:54:05	18

Structure of web framework:



Deployment Diagram



The software has been tested and is compatible with Chrome, Edge, and Firefox. We did not test other browsers. It makes use of the University of Regina's WebDev server using our student accounts. This is fine for the short term but the web app will need to be moved to a

different server after we graduate if we want it to continue to work. The database solution used is MySQL as it was the one we were most familiar with.

Links

- GitHub repository: https://github.com/wsd578/Cs476
- Webpages:
    - Login: http://www.webdev.cs.uregina.ca/~ioa388/cs476/login.php
    - Sign-up: http://www.webdev.cs.uregina.ca/~ioa388/cs476/signup.php
    - Home: http://www.webdev.cs.uregina.ca/~ioa388/cs476/index.php
    - Book List: http://www.webdev.cs.uregina.ca/~ioa388/cs476/books.php
    - Search: http://www.webdev.cs.uregina.ca/~ioa388/cs476/search.php
    - About: http://www.webdev.cs.uregina.ca/~ioa388/cs476/about.php
    - Account: http://www.webdev.cs.uregina.ca/~ioa388/cs476/myaccount.php
    - Shopping cart: http://www.webdev.cs.uregina.ca/~ioa388/cs476/cart.php
- Additional note: An account needs to be made before logging in and viewing pages.

Technical Documentation

This section will list the programming languages, reused code, software tools, and environments used during the creation of our website. It will also briefly discuss the benefits of our choices.

Programming Languages

Our website uses HTML, PHP, and Javascript. It also utilises JSON to convert data in and out of the database.

Reused Code

Our code is adapted from assignments done in CS 215 at the University of Regina. This mostly applies to the structure of the website layout and the login/signup features. Link to CS 215 homepage: http://www2.cs.uregina.ca/~hoeber/teaching/cs215/2021W/

Software Tools & Environments

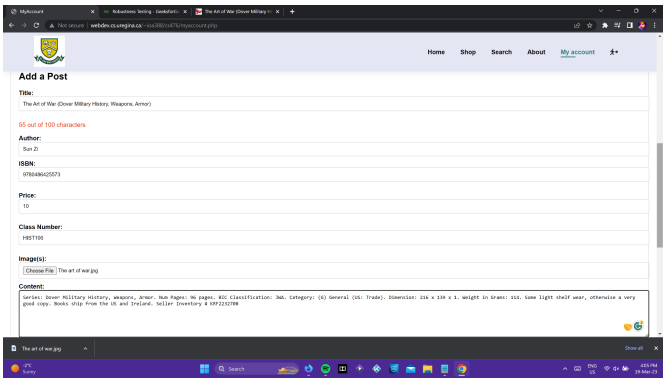
- We are hosting our application on the University of Regina's WebDev server as this is the cheapest and most familiar option for us. The main benefit of WebDev is that it allows us to host our website for free on secure and reliable University servers.
- MySQL is used for the database. This was the most familiar option for us and also provides the benefits of being easy to work with and reliable.
- Visual Studio Code was used for programming. It allows us to code in all languages listed earlier.

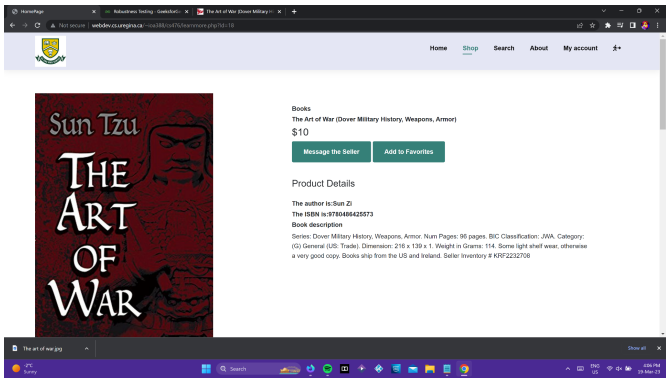
Acceptance Testing

The testing section provides screenshots of the use of our website to ensure correctness, robustness, and time-efficiency. These qualities make a website more user-friendly and enjoyable to use.

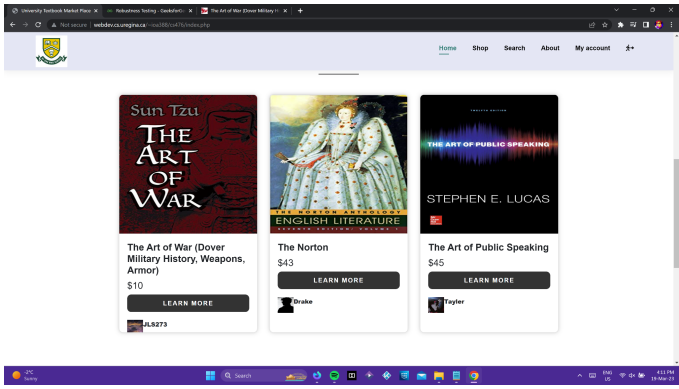
Correctness

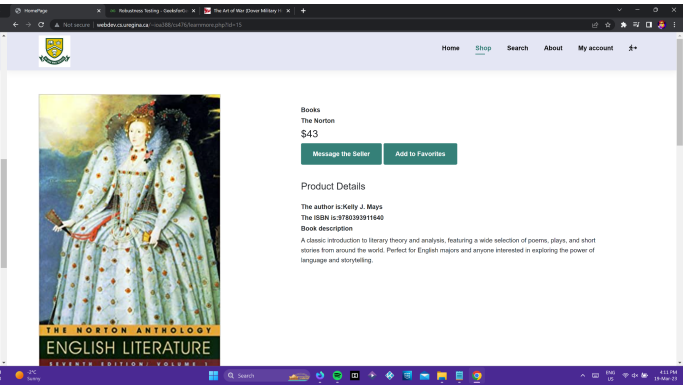
Uploading a book to sell



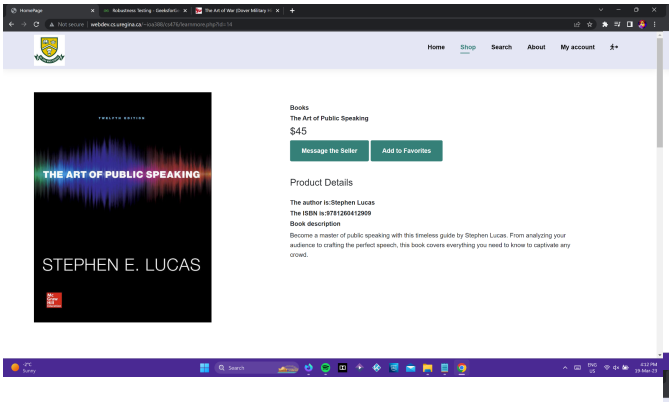


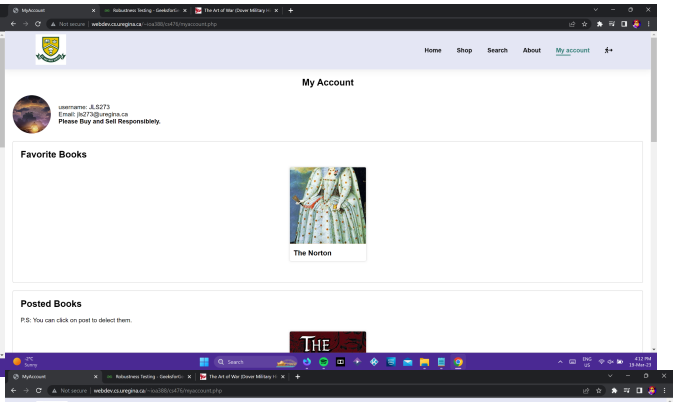
Clicking on a book to get to its listing

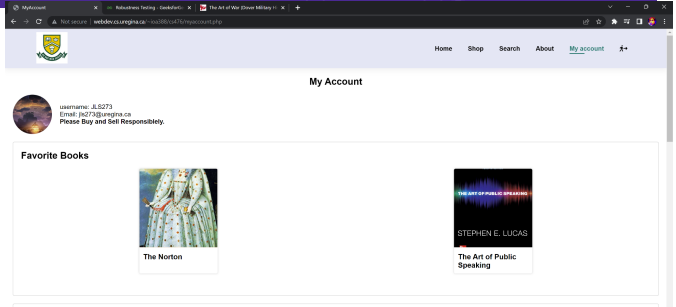




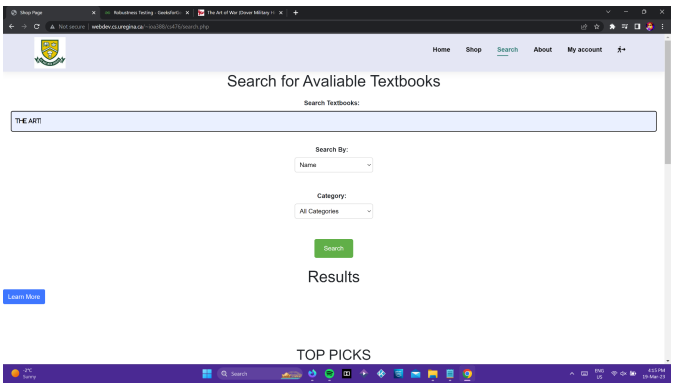
Adding a book to favourites

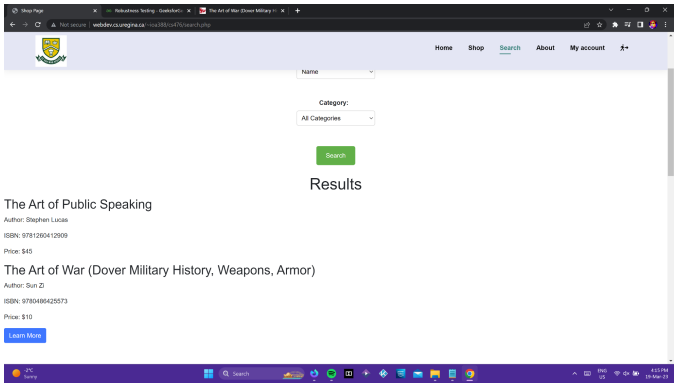






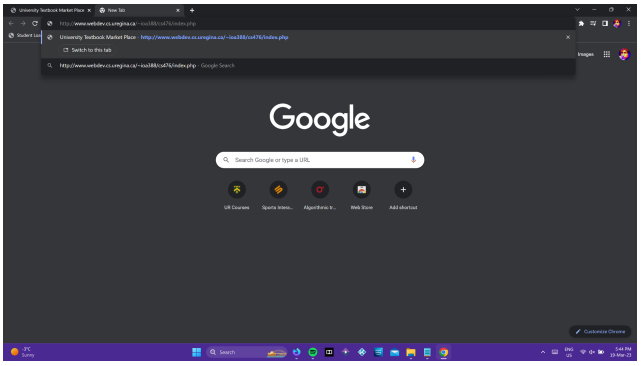
Searching for a book

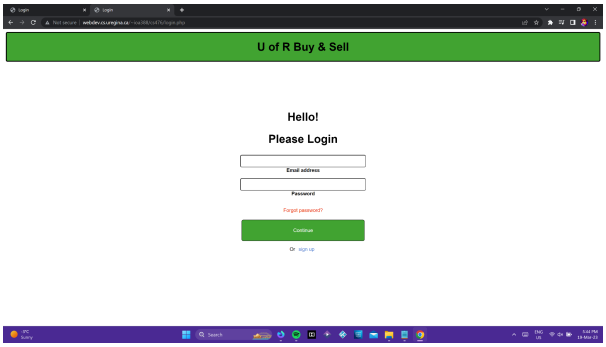




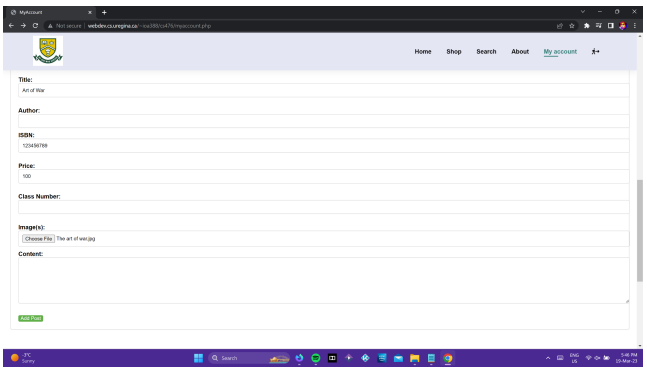
Robustness

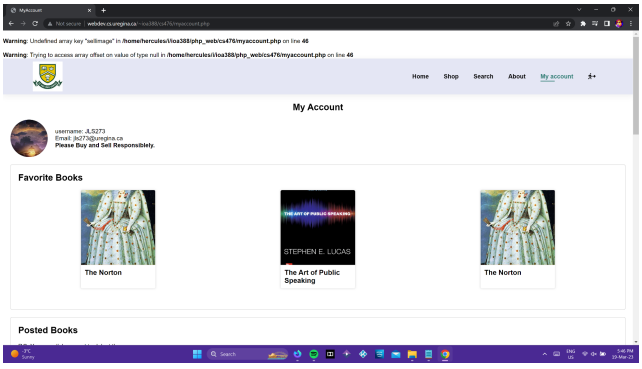
Trying to access the site without logging in



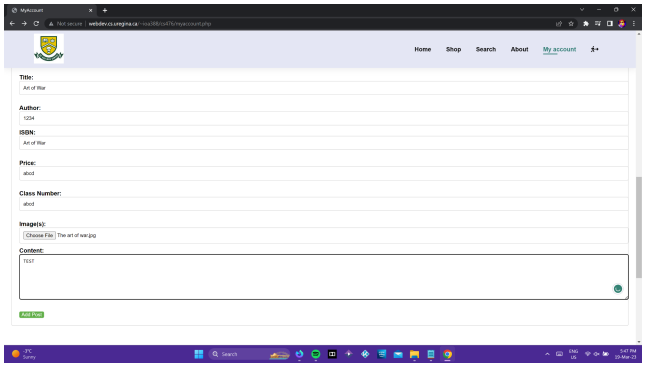


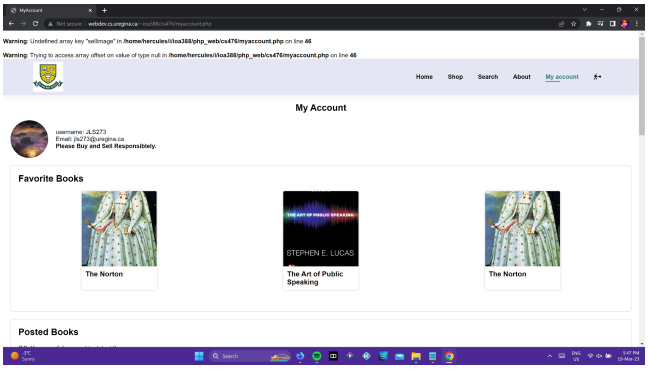
New post with missing details



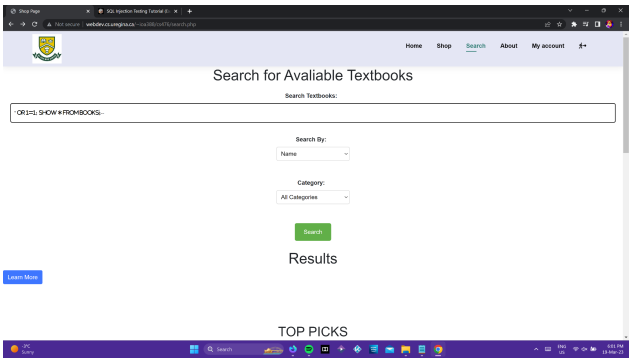


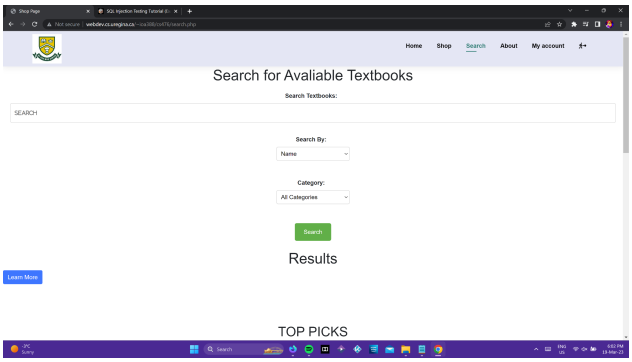
New post with incorrect details



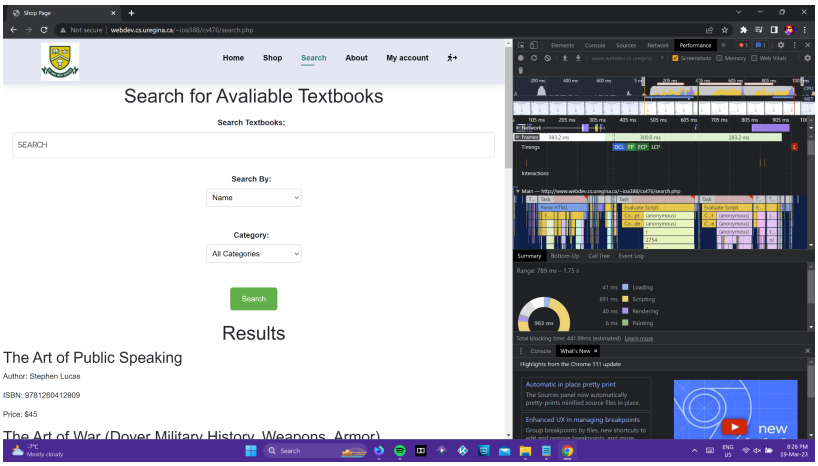


SQL injection in search bar

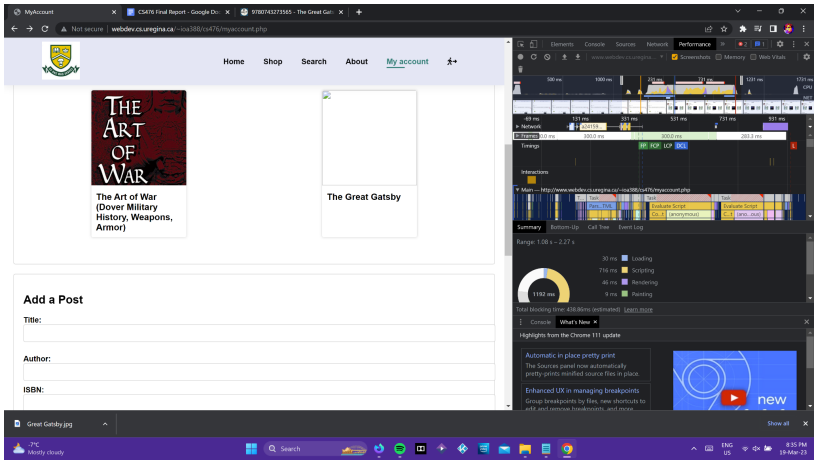




Time-Efficiency



Searching for books took about 205ms to display results.



Adding a new post took about 231ms.

References

"CS 476 Winter 2023." *Miro.com*, miro.com/app/board/uXjVP1NDmkg=/. Accessed 17 Feb. 2023.

Hoeber, O. (2021). CS 215 :: University of Regina. Retrieved March 21, 2023.

Sadaoui, S. (n.d.). *CS 476: Software Development and Demonstration Requirements*. UR Courses. Accessed 17 Feb. 2023.