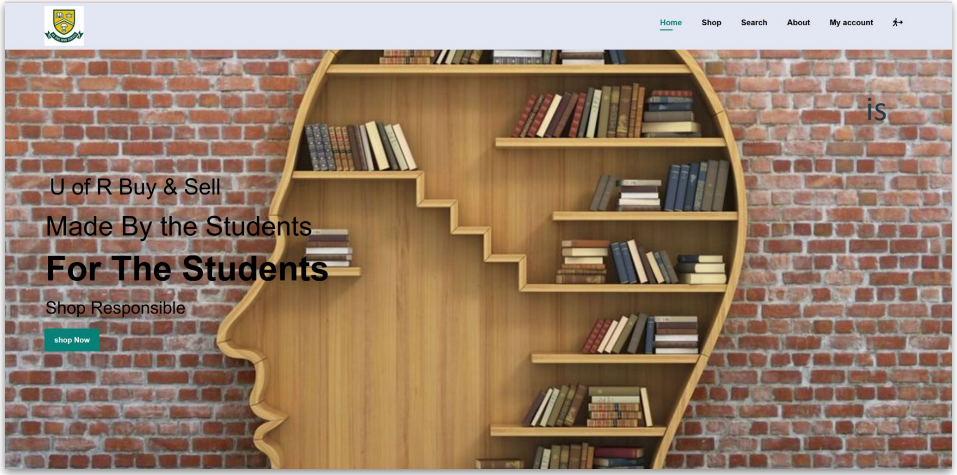
U of R Buy & Sell

CS476 Final Project

Ibrahim Adogba | Wilbur Dulce | James Sieben

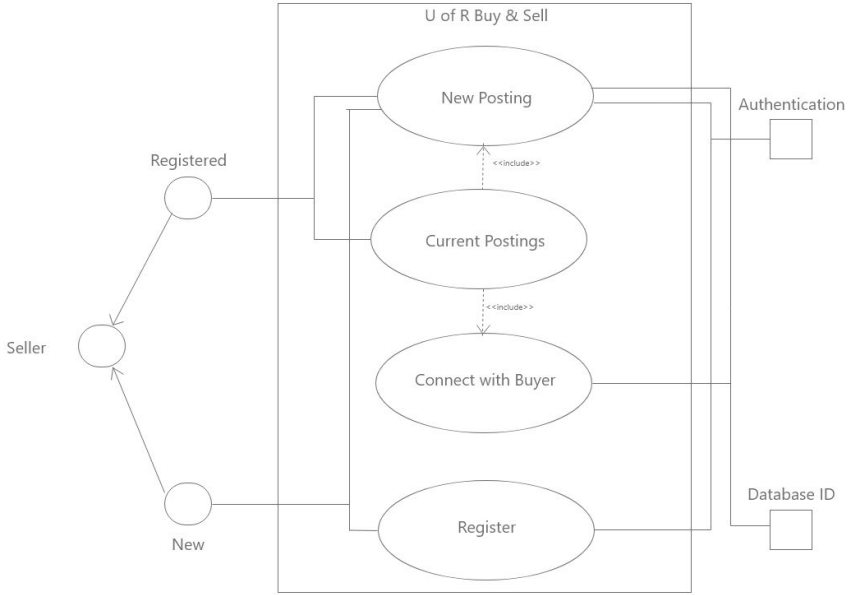
Purpose and Goals

- The purpose of our project is to create a space where students can directly buy and sell textbooks from each other
- Our Buy & Sell website makes it easy for U of R students to upload their old textbooks and connect with other students
- In contrast with Facebook marketplace or Kijiji, our site only allows books and only open to U of R students
- Unlike the school bookstore, books of any quality can be posted and there is no additional markup



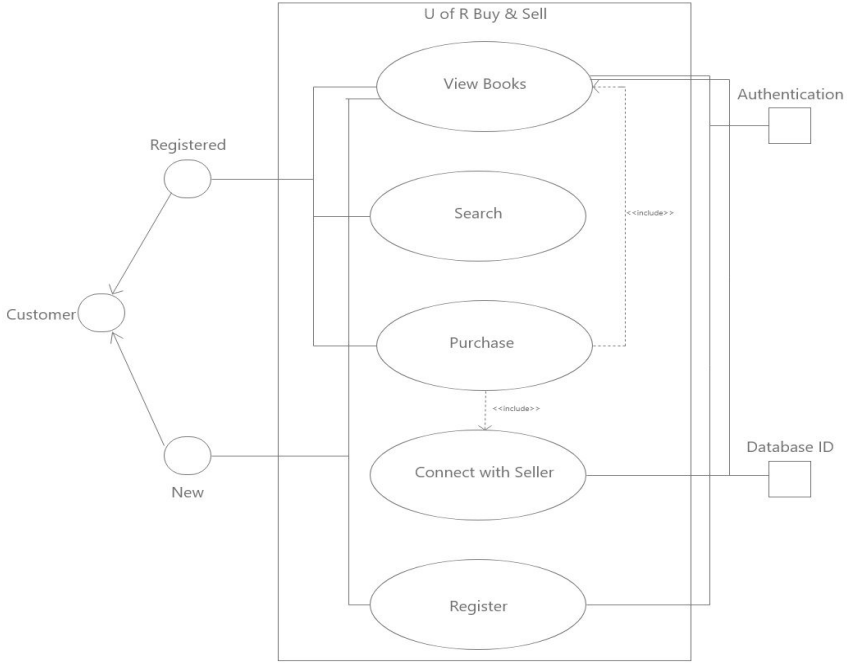
Use Case: Seller

- Sellers must sign up before using our site and must be registered U of R students
- Registered users are asked to login before viewing the site
- Sellers may:
  - Create a new post
  - View their current postings
  - Connect with potential buyers through their student email



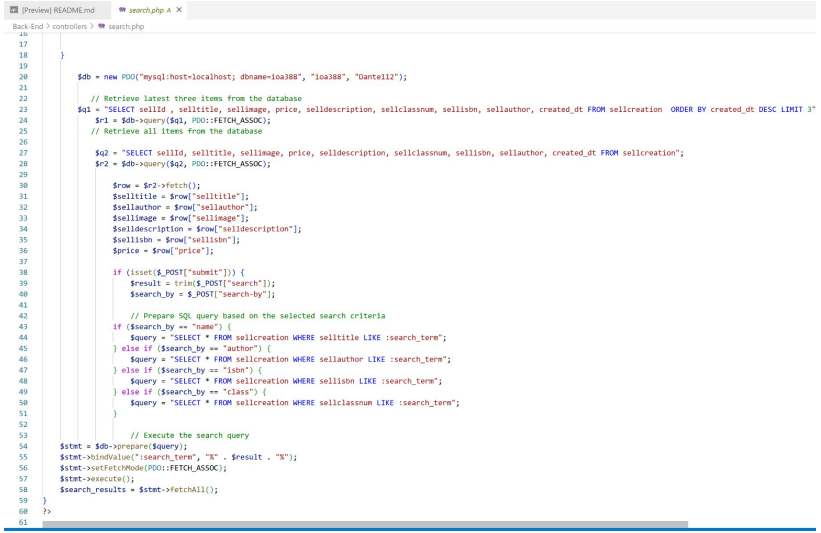
Use Case: Buyer

- Similarly, students shopping for books must login before viewing the site
- Users may:
  - View a specific listing
  - Search the site for books by title, author, genre, or ISBN
  - Purchase books by connecting with sellers
- To buy a book, the user will be shown the seller's email so that they may directly communicate



MVC: Controller

- The Controller is responsible for processing input and directing data flow
- In our web app, it handles 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
- Ex. If a book is bought, the Controller removes it from the database and updates the View so that book isn't shown anymore



- The screenshot shows the code to search the Model and returns results to the View

MVC: Model

- The Model is the data layer, it's responsible for storing, retrieving, and manipulating the data stored in the database
- For our use, it stores info about books and users, such as the book titles, authors, descriptions, and usernames, emails, and addresses
- Our site uses MySQL and JSON requests
- The code shown creates the Users table and the favorites table within the Model

[Preview] README.md | Users.sql

Database > Users.sql

```sql
DROP TABLE IF EXISTS Users;

CREATE TABLE Users (
    userId int(11) NOT NULL AUTO_INCREMENT,
    email varchar (50) NOT NULL,
    photo_img  varchar(10000) NOT NULL,
    pswd varchar (50)NOT NULL,
    username varchar (50) NOT NULL,
    PRIMARY KEY (userId)
    );
```

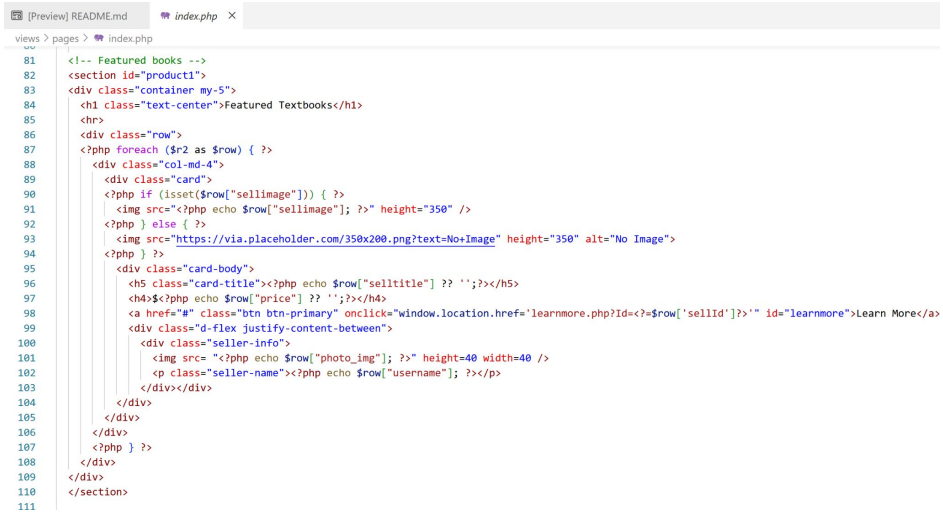
[Preview] README.md | favorites.sql

Database > favorites.sql

```sql
DROP TABLE IF EXISTS favorites;


CREATE TABLE favorites (
    favouriteId int(50) NOT NULL AUTO_INCREMENT,
    sellId int NOT NULL,
    userId int NOT NULL,
    created_dt datetime,
    PRIMARY KEY (favouriteId),
    FOREIGN KEY (sellId) REFERENCES sellcreation (sellId),
    FOREIGN KEY (userId) REFERENCES Users (userId)
    );

    insert into favorites (favouriteId, sellId, userId, created_dt)
    values ('1','14','1', '2023-03-17 18:24:03');

```

MVC: View

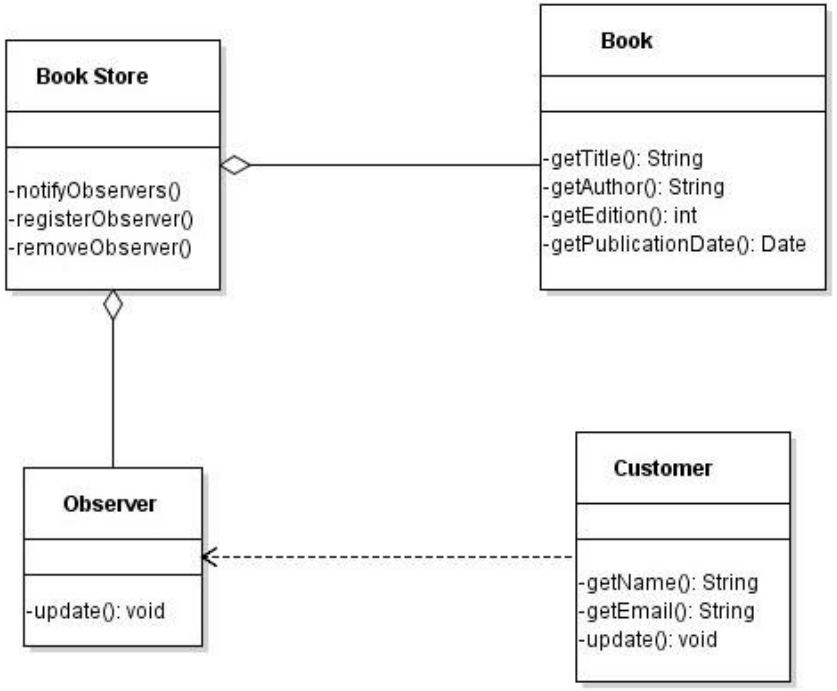
- The View is the user interface of the app
- We used HTML, PHP, and CSS to display our website to users
- Updated by the Controller so if a book is uploaded or deleted the View will display the correct info
- This code displays the featured books on the homepage - it relies on the Controller to pull titles and images from the Model


[Preview] README.md
index.php
views > pages > index.php

```
<!-- Featured books -->
<section id="product1">
<div class="container my-5">
  <h1 class="text-center">Featured Textbooks</h1>
  <hr>
  <div class="row">
  <?php foreach ($r2 as $row) { ?>
    <div class="col-md-4">
      <div class="card">
      <?php if (isset($row["sellimage"])) { ?>
        <img src="<?php echo $row["sellimage"]; ?>" height="350" />
      <?php } else { ?>
        <img src="https://via.placeholder.com/350x200.png?text=No+Image" height="350" alt="No Image">
      <?php } ?>
        <div class="card-body">
          <h5 class="card-title"><?php echo $row["selltitle"] ?? '';?></h5>
          <h4>$<?php echo $row["price"] ?? '';?></h4>
          <a href="#" class="btn btn-primary" onclick="window.location.href='learnmore.php?Id=<?=$row['sellId']?>'" id="learnmore">Learn More</a>
          <div class="d-flex justify-content-between">
            <div class="seller-info">
              <img src= "<?php echo $row["photo_img"]; ?>" height=40 width=40 />
              <p class="seller-name"><?php echo $row["username"]; ?></p>
            </div></div>
          </div>
        </div>
      </div>
      <?php } ?>
    </div>
  </div>
</section>
```

Design Pattern: Observer

- The observer design pattern is focused on the interactivity of the site
- User's are assigned "Observers" which notify them of state changes
    - For sellers, an Observer notifies them if a user has added their book to the user's favourites list
    - For buyers, an Observer notifies them if a book in their favourites has been sold or if a book they want has become available

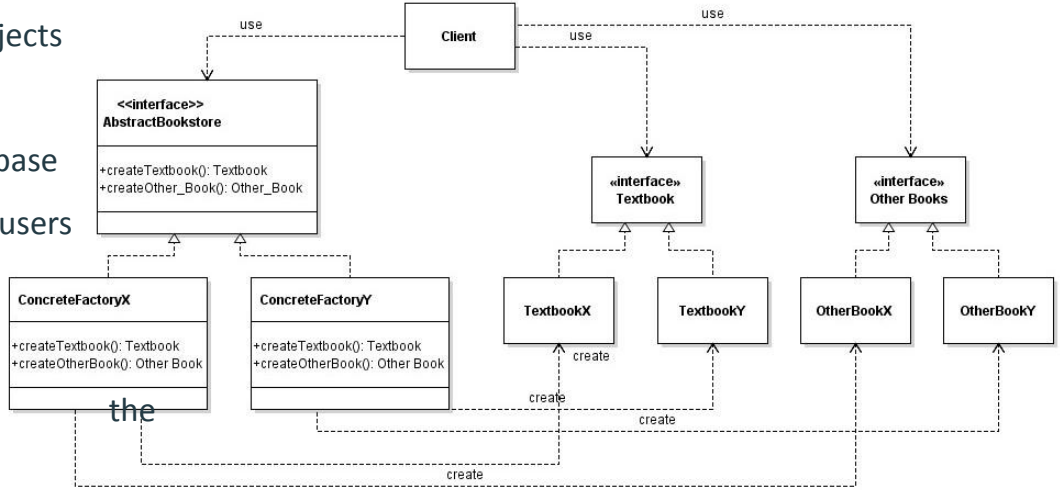


Design Pattern: Observer Cont.

- Implementation of this design pattern was limited due to complexity and time constraints
- At the moment, user's can add books to a favourites list or upload new posts but they will not be notified if any state change occurs
- The code shown pulls the user's favourite books from the database to be displayed. Ideally, an Observer would be here to occasionally check for changes and notify the user if there is

Design Pattern: Factory

- Makes creating or deleting book objects simpler and keeps code organised
- Similar for adding users to the database
- A book can be uploaded by normal users or by administrators - then the create function is called and a new book object is created with info like the title, author, and ISBN
- The client is only able to see the interfaces for the bookstore and books



Design Pattern: Factory Cont.

- Sellers can be thought of as the factories creating new books by making new posts
- Buyers only interact with the View which uses the Controller to display the bookstore website and books
- The code shown takes user-entered info to make a new post and saves it in the database - this can be thought of as the create function in the factory

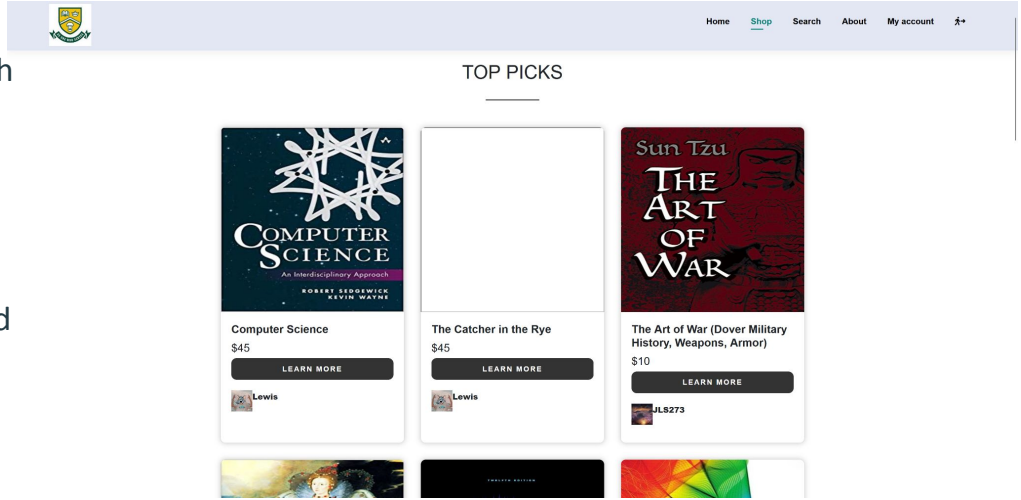
[Preview] README.md | myaccount.php

views > pages > myaccount.php

```php
if (isset($_POST["Createsell"]) && $_POST["Createsell"])
{
  $selltitle = trim($_POST["book-title"]);
  $sellauthor = trim($_POST["book-author"]);
  $sellisbn = trim($_POST["book-isbn"]);
  $sellclassnum = trim($_POST["book-class"]);
  $selldescription = trim($_POST["post-content"]);
  $price = trim($_POST["book-price"]);
  $currenttime=time();
  try {

    if($selltitle == null || $selltitle == "" || $selltitle == false) {
        $validate = false;
        $error .= "Book Title Is empty.\n<br />";
    }

if($validate == true) {
    $created_dt=date("Y-m-d H:i:s",$currenttime);
    $q2 = "INSERT INTO sellcreation (selltitle, sellauthor, sellisbn, sellimage, selldescription, created
    VALUES ('$selltitle', '$sellauthor', '$sellisbn', '$target_file', '$selldescription', '$created_dt',

    $r2 = $db->exec($q2);

    if ($r2 != false) {
      header("Location: myaccount.php");

      $r2 = null;
      $db = null;
      exit();

    } else {
      $r2 = null;
      $validate = false;
      $error .= "Trouble adding product to database!\n<br />";
    }
  }
```

Software Behaviour

- The site functions in a similar way to Kijiji or Facebook marketplace
- Sellers upload their own books to sell and buyers browse books and message sellers directly
- Transactions do not take place on our website, instead users are expected to work out payment methods and book exchanges themselves after connecting through our site
- The site allows users to search, view top picks, view a specific book, add a book to favourites, and upload



Thanks!