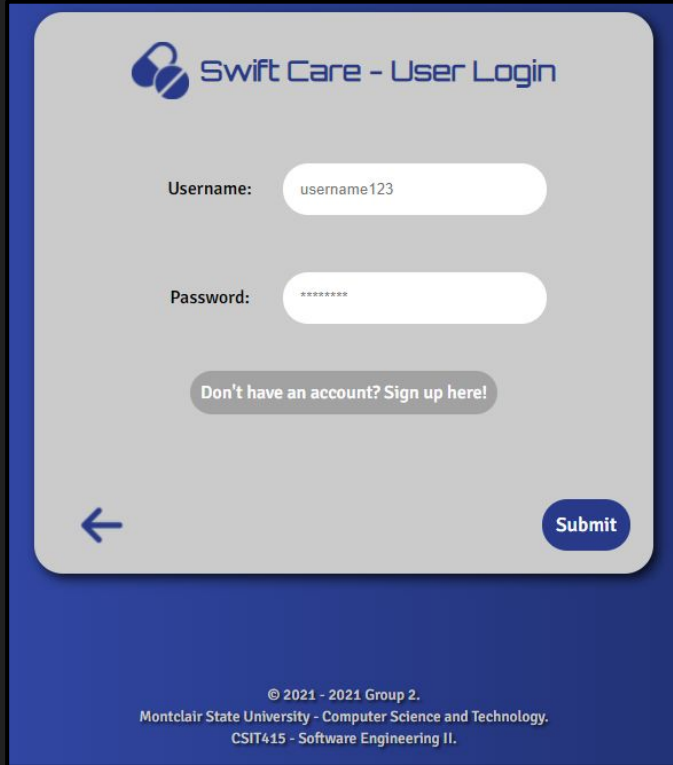# SE2 Group 2 Presentation

By: Arthur Levitsky, Nachi Patil, Joseph Jeetan, Christian Kohlmeyer, Daisy Gonzalez, Lauren Economy, Hope Diamantopoulos

Login Component

User Login

- Front-End:
    - Implemented with HTML and CSS.
    - Username input element: type = text.
    - Password input element: type = password.
    - Placeholders for usability.
    - A link element that looks like a button that sends the user to the Signup page.
    - Error message uses CSS animations to fade away.

User Login

When user first sees the login page
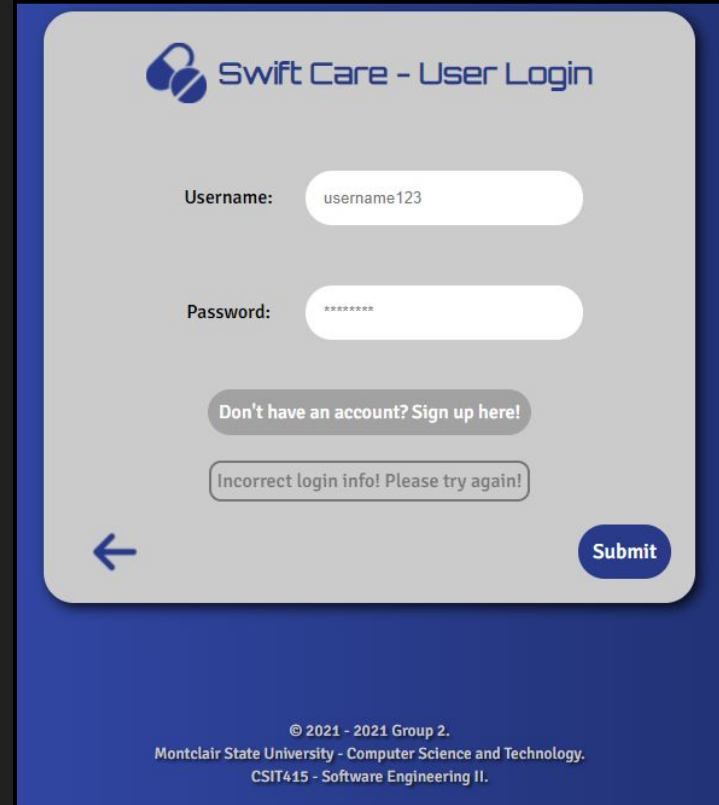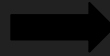
When user types in the wrong information

User Login

- Backend-End:
  - Implemented in PHP using sessions
  - Uses PHPmyadmin database
  - When a user inputs the incorrect credentials, they are presented with an error notice, using echo notice
  - If statement checks were created to ensure...
    - that multiple users could not be logged in at the same time
    - The credentials are indeed in the "users" table in order to log in

"users" table in database ^^^

Error notice ->

User Signup

- Front-End:
    - Email input element: type = email
    - HTML structure of User Login Page was reused and styling from global CSS file.
    - Helped avoid recreation of elements, saved time, and kept things structured and organized.

User Signup

When user first sees the signup page

When user types in existing information

User Signup

"users" table in database ^^^

- Backend-End:
  - Implemented in PHP using sessions
  - Uses PHPmyadmin database
  - If statement checks were created to ensure...
    - A user with the same email address that already exists in the database can not be created
    - An admin cannot create an account under their username since they are an admin, not user
  - General error message created with echo notice, left purposely general for security purposes

Error notice ->

Admin Login

- Front-End:
    - HTML structure was reused from User Login Page and styling from the global CSS file.
    - Additional CSS file was created used to change the background color to red and scroll wheel to red.
    - There needed to be some distinction between User pages and Admin pages. Changing the color to red helped solve this problem as well as provide more style.

Admin Login

When the admin first sees the Admin Login Page.

When the admin types in the wrong credentials.
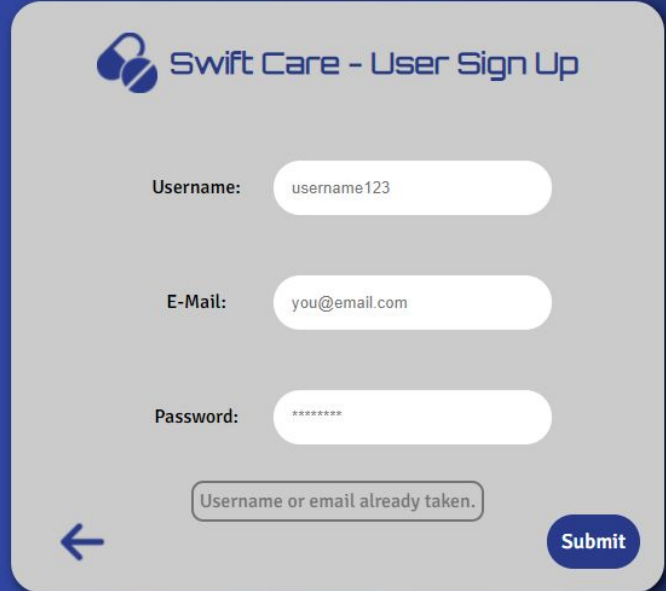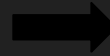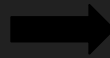
Swift Care - Admin Login
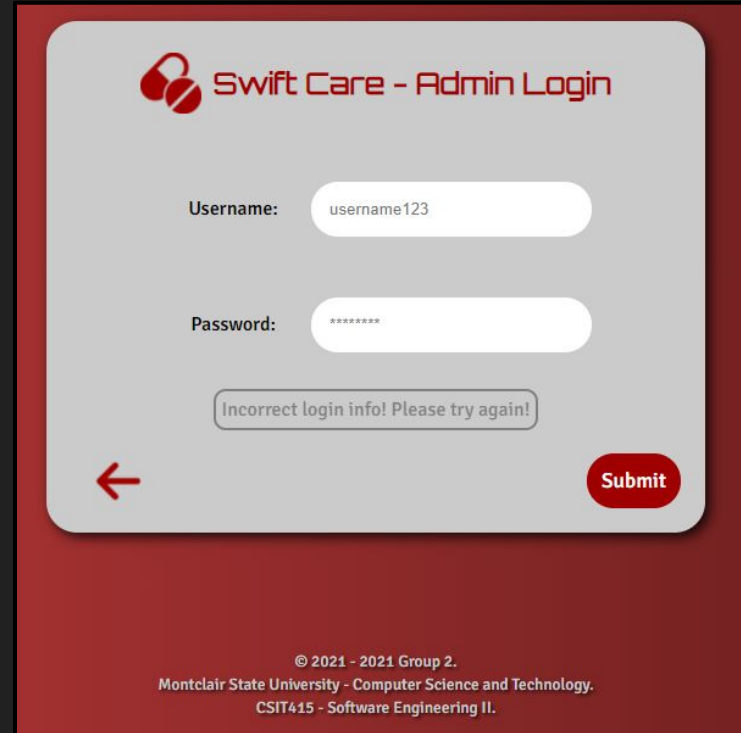
Username: username123

Password: ********

Submit

© 2021 - 2021 Group 2.
Montclair State University - Computer Science and Technology.
CSIT415 - Software Engineering II.

Incorrect login info! Please try again!

Admin Login
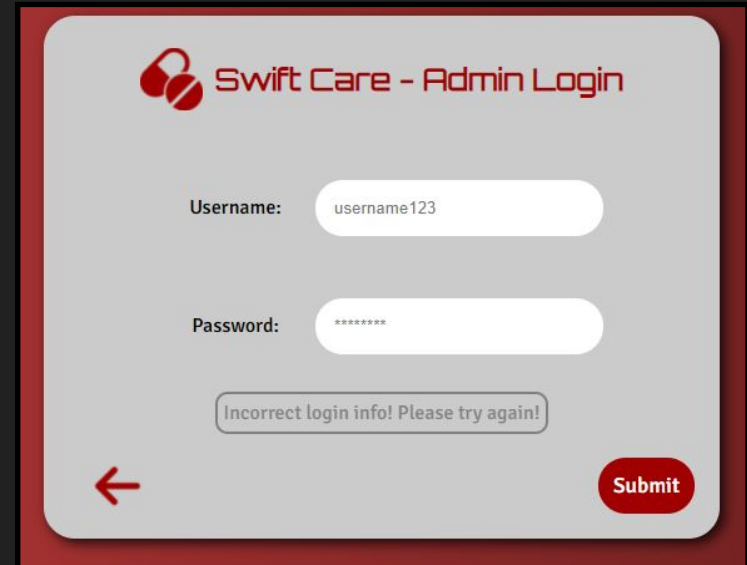
id	username	password
20	super23	$2y$10$4e5B6hMpQJCmwJTE/Xy.4.gcsRTb16IZvg1/adyEaqF...

- Back-End: "admin" table in database ^^^
    - Implemented in PHP using sessions
    - Uses PHPmyadmin database
    - If statement checks were created to ensure...
        - Only an Admin can access the Admin Panel or any pages restricted to an admin
        - A user account cannot login to an Admin Account

Error notice ->

Order Component

View Products

- Front-End:
    - This is considered the main landing page of Swift Care medicine website.
    - The first page that was constructed for this project. (global CSS was being constructed from scratch in reference to this page)
    - Icons were used to serve as links to send the user to different pages in the website. (User Icon, Cart Icon, Logout Icon)

View Products (Design Prototype)
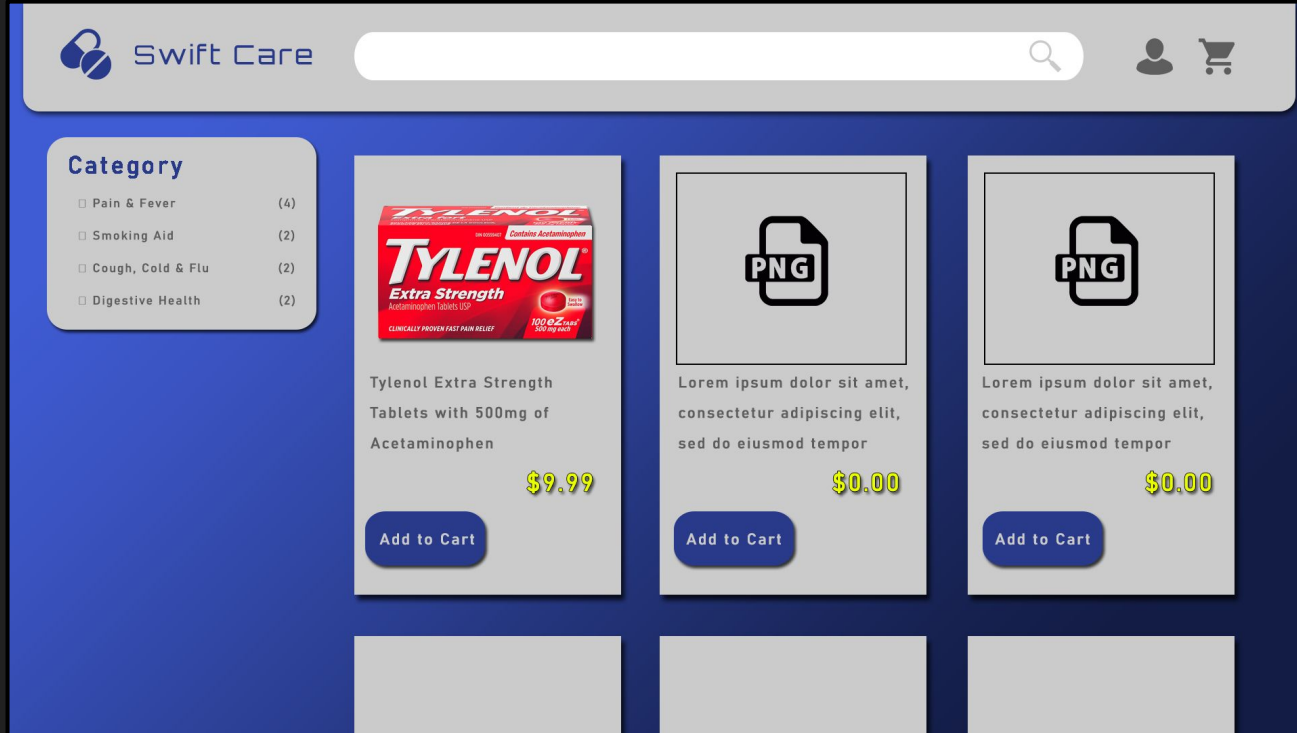
Proposed Design of the website landing page. (Created with Adobe Photoshop)

View Products
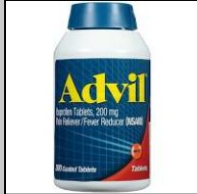
Landing Page of Swift Care medicine website. (User can view all the products on this page)

Category Selection

Once 'Pain Relief' category was selected and submitted, shows relevant products.

Searchbar Selection
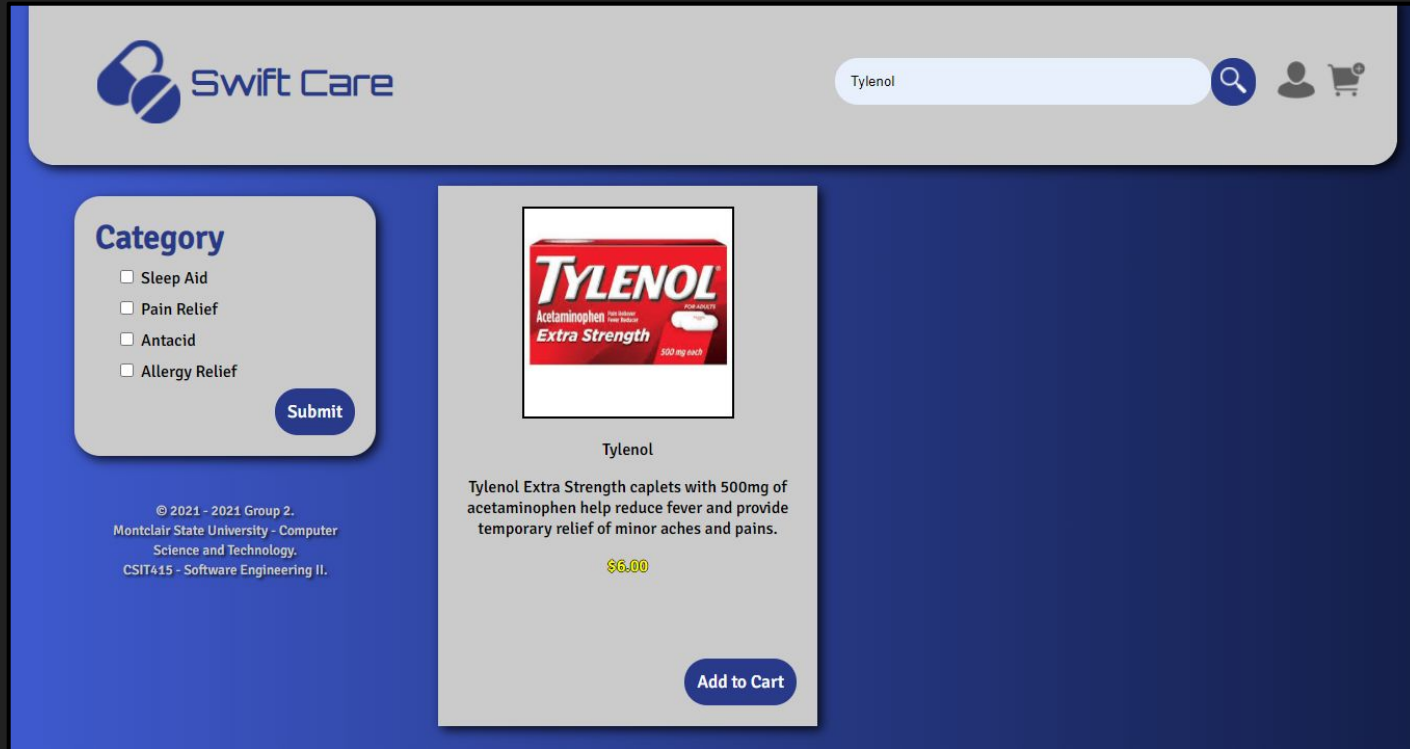
Once 'Tylenol' was typed into the search bar.

Add to Cart

User adds an item to cart. (Notice 'Item successfully added to cart!' and logout icon)

User Cart

- Front-End:
    - Reused the User Login Page to obtain the container and the title with icon.
    - Internal components held a table that was three columns wide. Rows of the table were dependent on what the user added to their cart. (Dynamic PHP and HTML)
    - Table header elements contained white text and blue background to go with the style of the page.
    - Additional table below that held the Total Price with a blue border for only the top of the table.
- Back-End:
    - Cart implemented with a MySQL database
    - Backend functionality implemented with PHP
    - User can add items to cart, remove items from cart, or update quantity of an item in cart
    - User can view items currently in cart before purchasing
    - Items in cart are automatically deleted after 7 days

User Cart

The User Cart Page with all the selected items. (Notice subtotal changed for Melatonin based on quantity)

Empty Cart

The User Cart Page when user has removed all the items or had no items in the cart.

Payment Component

Place Order and Payment API:

Checkout Page presented after user selects 'Continue to Checkout'
from Cart Page.

Place Order and Payment API:
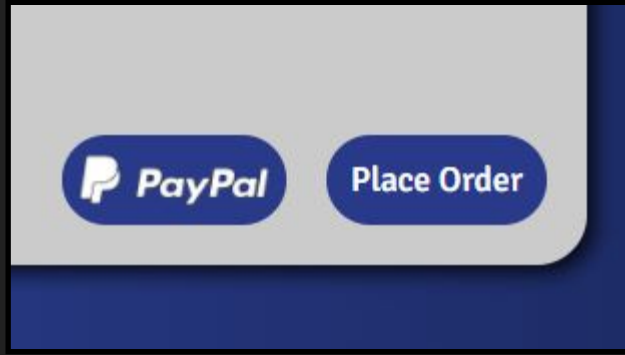
Paypal Button that a user can press in the Checkout page.
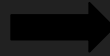
Redirects to the Paypal Login page asking user to input their credentials. (FAKE API)

Place Order

- Front-End:
    - Styling from the global main CSS file and user CSS (blue)
    - HTML table styling for shipping address on the right and payment method on the left
    - Input fields with specific input types to different fields such a month and state drop down table
    - Hidden password for security purposes
- Back-End:
    - SQL statements to place order details in orders and orderitems tables.
    - Orders table stores user information.
    - Orderitems table stores actual item details.

Order History (User)

- **Back-End:**
  - Users can view any past orders on their account.
  - Order cancellation is possible from this page as well.
  - Cancelled orders are removed from the orders and orderitems database.

View and Order Details/Cancel Order Front End (User)

- Used HTML and CSS
- Same format as other User pages
    - Blue Background
    - Grey Container
- Blue "View details" button will direct user to order details page
- Once user on order details page customer can Cancel order using button in addition to viewing order information
- Blue back arrow redirects customer to Control Panel

User View and Order Details/Cancel Order

Shows all User's orders
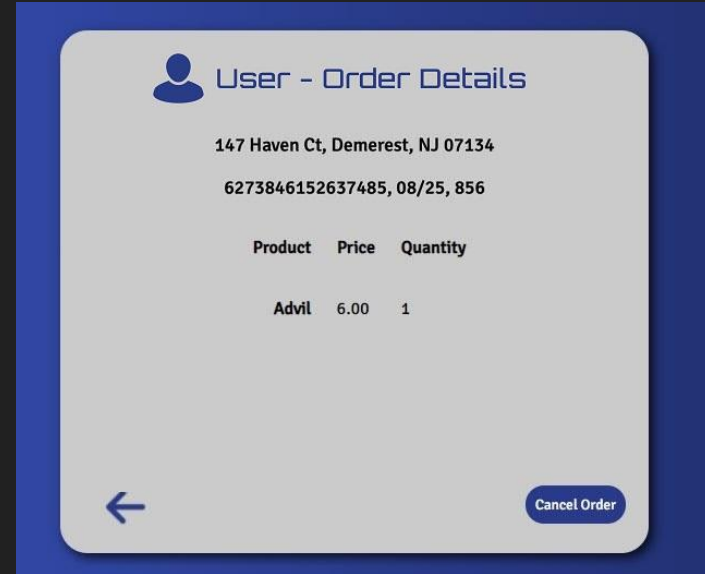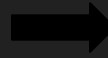
Shows all User's order details along with cancel order option

Managerial Component

View and Process Orders Front End (Admin)

- Used HTML and CSS
- Same format as other Admin pages
    - Red Background
    - Grey Container
- Red Authorize button will redirect order from "Unprocessed Orders" to "Processed Orders"
- Red "View details" button will direct Admin to Order details page
- Red back arrow redirects Admin back to control panel

Admin View of Placed Orders

Admin can authorize and view details

Shows User order details

View and Process Orders (Admin)

- Admin can view information about all orders placed by users
- Admin can authorize orders to be shipped to customer
- Unprocessed orders are displayed in a priority table above processed ones

Admin View of Order Details
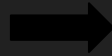
Admin - Order Details - #29

Product	Price	Quantity
Tums	$3.00	1
Advil	$6.00	1
Melatonin	$5.00	1

Admin View of Product Entry, Update, and Delete

Admin - Add/Update Product

Would you like to create or update a product? [Create ▾]

ID:

Name:

Category:

Price:

Description:

Image: [Choose File] No file chosen

← [Submit]

Admin - Remove Product

Product ID	Product Name	Product Category	Product Price
1	Melatonin	Sleep Aid	$5.00
2	Advil	Pain Relief	$6.00
3	Tums	Antacid	$3.00
4	Zyrtec	Allergy Relief	$9.00
5	Benadryl	Allergy Relief	$12.00
6	Tylenol	Pain Relief	$6.00
7	Acetaminophen	Pain Relief	$6.00
8	Pepcid	Antacid	$20.00
9	Pepto Bismol	Antacid	$7.00
10	ZzzQuil	Sleep Aid	$7.00

← [Enter Product ID] [Delete]

Admin View User Details

			id	username	password	email
☐ 🖊 Edit ⧉ Copy ⊘ Delete			3	test	$2y$10$g68OfbONaakZ7p8xGs0oy.89nz1wWXD72Je/bTfRjo....	test@email.com
☐ 🖊 Edit ⧉ Copy ⊘ Delete			2	christian	$2y$10$doWIrntVUwIH9HnrYWn41On7ua4Cm8vBjuv2.YUXZSw...	christian@test.com
☐ 🖊 Edit ⧉ Copy ⊘ Delete			4	arthur	$2y$10$MeLtuLGQoaPePSA269BqRuKzKt1V04yxAvRvned8K9r...	art@gmail.com
☐ 🖊 Edit ⧉ Copy ⊘ Delete			5	joe	$2y$10$DM38XERL9LAxUJqYn7mzOufOyBB/w1ZqTRAIkT4XqUB...	joej@aol.com
☐ 🖊 Edit ⧉ Copy ⊘ Delete			6	hope	$2y$10$delPijd/AhM9F5wQQKsi7uixtP/Go6Ad5up.zJh1ahg...	hope@gmail.com

- Fetched the user details from the "users" table in the backend
  - Selects all the data from the table
  - Queries the data in order to make it available to display in the front end
  - Echos the results

"users" table in database ^^^

Results echoed ->

Admin - View User Details

User ID	User Name	User Email
3	test	test@email.com
2	christian	christian@test.com
4	arthur	art@gmail.com
5	joe	joej@aol.com
6	hope	hope@gmail.com
7	asdasda	dassdasd@gmail.com

←

© 2021 - 2021 Group 2.
Montclair State University - Computer Science and Technology.
CSIT415 - Software Engineering II.

Interface Requirements

Interface Requirements
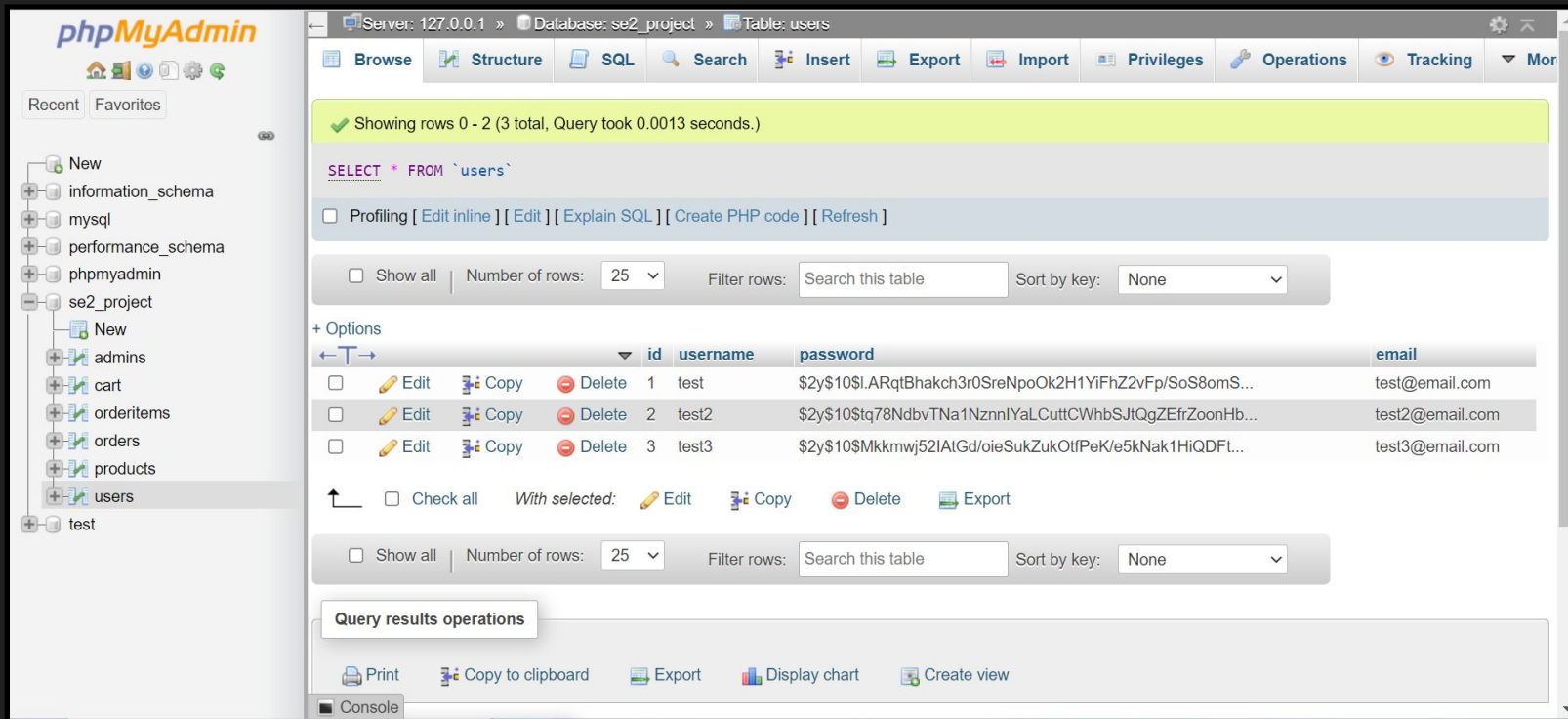
- Application Type:
    - *Swift Care* is a web-based application.
- Front-end Software:
    - Notepad, Sublime Text, and Visual Studio.
    - The front-end was written solely in HTML and CSS.
- Back-end Software:
    - phpmyadmin for database management and MySQL for the actual database.
    - The website was also created on a real public server which was setup with LAMP software.
    - Individual local testing of the database files and web-page files were done using XAMPP software.
    - The back-end was written solely in PHP.
- Operating System:
    - The website is operational on Windows, Linux, and MacOS.
- Supported Browsers:
    - The website is supported on browsers such as Brave, Chrome, and Firefox.
- Editor and Compiler Used:
    - No compilers were used in the production of the *Swift Care* medicine website.

Non-Functional Requirements

Security

- Passwords for accounts were hashed when stored in the database; PHP's default password hashing function was used
- Outputs when the user entered an invalid username/email while signing up were left vague in order to make it more difficult for bad actors to try and guess credentials of existing users
- An invalid username/password also returns a generic response, once again to prevent the guessing of credentials of existing users

Security

Hashed passwords stored in database

Security

Output for incorrect login info

Security

Output for signing up with invalid credentials

Usability

- User can resize the screen without breaking elements. The purpose is to allow users with any size screen to use the website comfortably.
- The overall design of the website is very simplistic as well as aesthetically more universal.
- The use of icons provide interface metaphors so that when users enter the website, they will not be required to learn anything, rather just be required to recognize what the icons mean in order to operate the website.
- The use of displaying messages back to the user is also an extremely important concept to usability within our website. Helps the user understand what to do next.

Usability
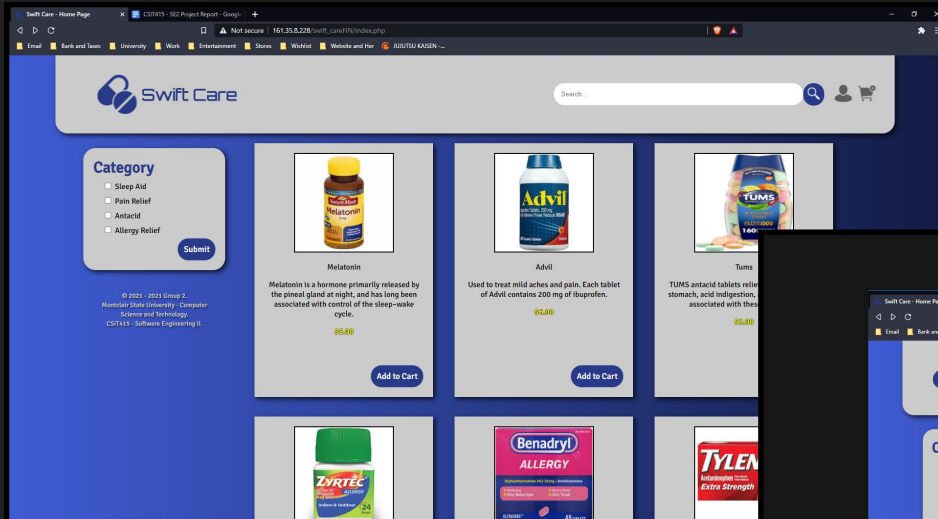
Display of the website in fullscreen (notice three products per row).
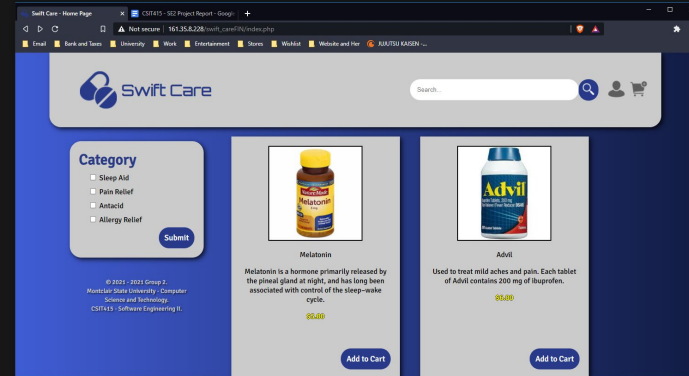
Display of the website not in fullscreen (notice two products are displayed)

Usability
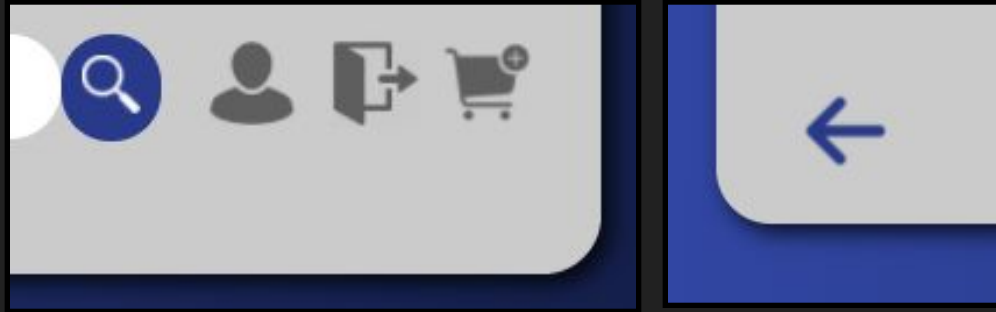
Representation of Search Icon, User Icon, Back Arrow Icon, Logout Icon, and Cart Icon.
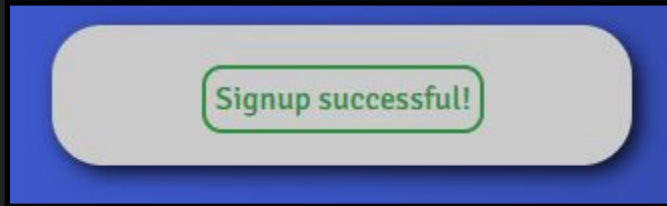
"Signup successful!" message after user creates an account.
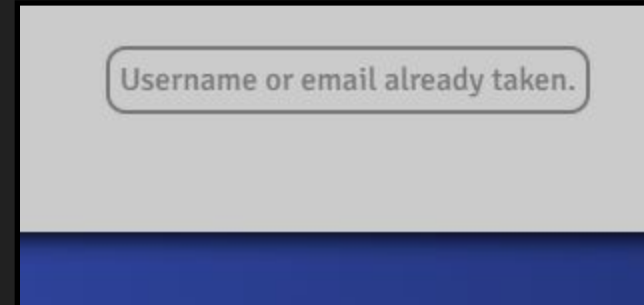
When a user attempts to create an account and fails, the message "Username or email already taken." must be displayed to notify them.

Roles of Team Members and Conclusion

Team Members

- **Nachi:**
    - Set up the server to host and deploy the website, set up product search (by name and by category), set up cart and cart functionality (add/remove items, update quantity to items), set up admin view of orders placed and admin ability to authorize orders.
- **Arthur:**
    - Largely responsible for the creation of the front-end portion of the website. Created the style of the website using a wireframe (Adobe Photoshop). Created the user interface for Admin Login page, User Login page, User Signup page, View Products page, User View Order page, Admin Process Order page. Provided assistance to the back-end of the User Login functionalities. Responsible for creating and modularizing most of the CSS for easy reusability of styling in all the front-end modules. Provided assistance and fixes to the rest of the front-end modules. Created Google Sheets module schedule.
- **Christian:**
    - Organized and deployed initial database tables, in addition to creating the back end files for Admins to execute Product Entry, Updates, and Deletion, as well as User functionality for Checkout, Viewing Order Details, and Canceling Orders.
- **Hope:**
    - Completed the backend for Admin Login, User Login, User Signup and Admin: View User Details. Set up most of the zoom meetings outside of class and helped troubleshoot other backend modules.
- **Lauren:**
    - Using HTML and CSS, created front-end page Place Order for users and View Order for admin side. Utilized the reusable CSS to maintain continuity in the pages I worked on.
- **Joseph:**
    - Created front-end pages for Admin View and Process Order, User View Order, and Cancel Order for User. Was done using HTML and CSS. Also recommended some style changes for simplicity reasons.
- **Daisy:**
    - Created front-end pages for Admin: View User Details and Product Entry, Update and Delete using HTML and CSS. Was done using reusable CSS so that the pages would have a cohesive design.

Conclusion

- As a result of the pandemic, not everyone was able to meet up in person. Much of the work had to be done remotely. This made sharing code and integrating modules difficult. Also, everyone having different levels of knowledge when it came to front-end and back-end development made it difficult to figure out who should complete each section. This led to group members having to learn new features they hadn't worked with previously and therefore encouraged pair programming.

- Organization also played a much bigger role in the project than expected, and more time should have been put into it before starting development. A uniform naming convention was not used at first, which made it difficult to integrate some pages and functions with each other. Methods of testing and location for storing and sharing code could have been made more clear in the beginning as well. These factors should be considered very early on upon starting another group project.