Database Creation

create tables

messages

messageID int NOT NULL AUTO\_INCREMENT,

message char(255) NOT NULL,

userID int NOT NULL,

chatID int NOT NULL,

seenByU1 char(5),

SeenByU2 char(5),

stampDate char(255),

stampTime char(255),

PRIMARY KEY (messageID)

chats

chatID int NOT NULL AUTO\_INCREMENT,

userID1 char(255) NOT NULL,

userID2 char(255),

PRIMARY KEY (chatID)

users

userID int NOT NULL AUTO\_INCREMENT,

userName char(50) NOT NULL,

PRIMARY KEY (userID)

carts

cartID int NOT NULL AUTO\_INCREMENT,

userID int NOT NULL,

purchased bool NOT NULL,

PRIMARY KEY (cartID)

products

itemID int NOT NULL AUTO\_INCREMENT,

itemName char (255) NOT NULL,

itemDesc char (255) NOT NULL,

itemPrice DECIMAL(6,2),

PRIMARY KEY (itemID)

itemCarts

instanceID int NOT NULL AUTO\_INCREMENT,

cartID int NOT NULL,

itemID int NOT NULL,

qty int NOT NULL,

PRIMARY KEY (instanceID)

add foreign keys

messages.userID = users.userID

messages.chatID = chats.chatID

chats.userID1 = users.userID

chats.userID2 = users.userID

carts.userID = users.userID

itemCarts.cartID = carts.cartID

itemCarts.itemID = products.itemID

Database Procedures

bleach(stringIN, cleanString)

-Cleanses any input strings by removing suspect characters that might be used for SQL injection and

-changing them to other characters

make the string lowercase

change = to !#eq

change or to !#0r

change and to !#&

change select to !#slct

change drop to !#drp

change insert to !#ins

change delete to !#del

change alter to !#alt

change create to !#cre

change where to !#whe

change + to !#plu

change - to !#min

change / to !#fsl

change \\ to !#bsl

change \* to !#sta

change ( to !#lper

change ) to !#repScreen

change ; to !#scol

return new variable

contaminate(stringIn)

-undoes everything the bleach procedure did to send it back out to the user.

make the string lowercase

change !#eq to =

change !#0r to or

change !#& to and

change !#slct select

change !#drp to drop

change !#ins insert

change !#del to delete

change !#alt alter

change !#cre to create

change !#whe to where

change !#plu to +

change !#min to -

change !#fsl to /

change !#bsl to \\

change !#sta to \*

change !#lper to (

change !#repScreen to )

change !#scol to ;

return new variable

enterChat(currentChatID, currentUser)

-logs the current user into the chat table as part of a chat

find the current users userID

update the chat record in chats table with current users userID as userID2

lastMessage(currentChatID, currentUser)

-returns the last messages added that the current user has not seen yet

sanitize the username entered with the bleach() procedure

create local variables

currentUserID = find the userID for current userName

user1Check = see if current userID is user1 for current chatID

user2Check = see if current userID is user2 for current chatID

if current user is userID1

concatenate stampDate, stampTime, (contaminate userName, (contaminate message)

for each message that userID1 has not seen yet

update messages.seenByU1 to 'yes' for all messages grabbed

if current user is userID2

concatenate stampDate, stampTime, (contaminate userName, (contaminate message)

for each message that userID2 has not seen yet

update messages.seenByU2 to 'yes' for all messages grabbed

select all messages that have been concatenated together to send off to PHP

newChatMessage(newMessage, userNameIn, chatID, newDate, newTime)

-inserts a new message into the messages table

get the userID for the current bleached userName

sanitize the incoming message with bleach()

insert message, userID, chatID, newDate, newTime into the messages table

newUser(userNameIn)

-create a new user if it does not already exist

sanitize the username in with bleach()

if userName does not exist already

insert new userName into users table

newChatID(userNameIn, newChatID)

-create a new chatID

sanitize the userName entered with bleach() and find the userID for current userName

insert userID into chats as userID1

select the highest chatID from chats for the current user and return that to PHP

checkRequests()

-check to see if there are any user request for a chat that have not already been answered

concatinate users.userName, chats.chatID from chats and users where there is no userID2

select all chats that meet this criteria and send off to PHP

cartStatus()

-pull information about all unordered carts. cartID, username, and cart total price value

find the total amount for all open carts

sum(products.itemPrice) from carts group by carts.cartID

create a table showing information about all open carts

concatenate cartID, userName, totalPrice from carts

send cart information table off to PHP

getProducts()

-get all product information from the products table

concatenate itemID, itemName, itemDesc, itemPrice

send information off to PHP

addtoCart(itemNum, currentUser)

-add and item to a users shopping cart

get the userID for bleached userName entered

find if there is an unordered cartID for current userID

if there is not an unordered cart

insert a new record for a cart in carts table with current userID

grab the unordered cartID for the current user

add the itemNum to current users cartID

getCart(userName)

-grab item list from current users open shopping cart

sanitize userName entered with bleach() and get the userID for it

find the unordered cartID for current userID

find all items for current cartID from items

send items off to PHP

getCartNum(currentUser)

-grabs the cart number for an unpurchased cart for the current user

sanitize the input username with bleach()

get the current users userID

find out if there is a current unpurchased cart for current userID

if there is no unpurchased cart

insert a new record into carts for a new cart

send the new cartID off to PHP

removeFromCart(instanceID)

-remove an item from a cart

delete a record from itemCarts matching the instanceID entered

Main Pages

index.php

-Temporary main page holding links to the framework website

Login.php

-main login page for customers

php:

start a session

include chat\_pdo.php

unset session variables

if POST user\_name is posted

save posted username as local variable

call the add user sql procedure with mysqli\_query

set the posted username as a session variable user\_name

send the user off to productlist.php

html:

foxy photography title

body

form name: loginForm action: login.php method POST

input type: text name: user\_name

input type: submit name: submit value: submit

repLogin.php

-representative login page

php:

start session

include chat\_pdo.php

unset session variables

if POST user\_name is set

save the post variable to a local variable

call the newUser sql function

set the session user\_name variable to the POST user\_name variable

send the user to repScreen.php

html:

foxy photography title

body

form name: loginForm action: rep:login.php method: POST

input type: text name: user\_name

input type: submit name: submit value: submit

repScreen.php

-representative screen that shows user chat requests, and open shopping carts

php:

start session

include chat\_pdo.php

show the user\_name variable on the screen

html:

Foxy Photography title

head

link to chat.css stylesheet

body

JavaScript:

include checkRequests.js

include cartStatus.js

pull the username from PHP's session variable

run every 3 seconds with setInterval function:

call checkRequests() JavaScript Function

call cartStatus() JavaScript Function

end JavaScript

p: users that are requesting to chat with a rep

div id: userRequests class: scrollbar

p: current unordered shopping carts

div id: cartStatus class: scrollbar

chatLogin.php

-log into a chat window either as a customer or a representative

php:

start session

include chat\_pdo.php

show the user\_name varialbe on the screen

if chatID is POSTed

save post variables to local variables

call sql stored procedures to get a new chat id from the database

if chatID was left at '1'

call newChatID() SQL stored procedure with mysqli\_query

grab the array from PHP and change it to a string and save as SESSION variable

else

save the chatID from POST to SESSION

call the enterChat(chatID, user\_Name) SQL procedure to log that user is in the chat

send the user off to chat.php

html:

Foxy Photography title

form name: loginForm action: chatLogin.php method: POST

input type: text name: chatID value: '1'

input type: submit name: submit id: submit value: submit

chat.php

-chat screen that shows messages saved in the message table of the database in a window and add new messages

php:

start session

include chat\_pdo.php

if chatID is REQUESTED

set REQUESTED chatID to SESSION chatID

display userName, and chatID on the screen

html:

Foxy Photography title

head

link to chat.css stylesheet

body

JavaScript:

include newMessage.js

include updateScreen.js

pull the userName and ChatID from SESSION variables to JavaScript variables using PHP

create a 3 second timer with setInterval()

run updateScreen() JavaScript function every 3 seconds asynchronously

end JavaScript

div id: message class: scrollbar

ul: id: messageList

label: type a question and click submit

input type: text id: enteredText

button: id: submit onclick: addNodeToDatabase() JavaScript function

JavaScript:

tell the form to submit if the enter key is pressed while in the text box

grab the text box id from DOM

add event listener function when "enter" key pressed for text box

click the submit button

end JavaScript

productList.php

-screen that shows all products that are available to be ordered, grabbed from the products table of the database

php:

start session

include chat\_pdo.php

display the SESSION user\_name on the screen

query the database for products using getProducts() SQL function using mysqli\_query

save the array grabbed as a string to send to JavaScript

html:

Foxy Photography Title

body

JavaScript:

include addtocart.js

end JavaScript

link to shoppingCart.php

link to chatLogin.php

div id: productList class: scrollbar

JavaScript:

grab the username from the SESSION variable through PHP

grab the allProducts string from PHP and save as JS variable

split the allProducts string back into an array

for each row in allProducts array:

slice the item number out of the front of the row and save as new variable

create a string variable with the row data wrapped in html code and a button

button links to addtoCart(itemNum) JavaScript function

add the string variable to the DOM using document.getElementById("productList").innerHTML

increase the count variable for the loop by 1

end JavaScript

shoppingCart.php

-screen that shows the contents of the users shopping cart, grabbed from the itemCarts table of the database

php:

start session

include chat\_pdo.php

display the SESSION user\_name on the screen

query the database for items in shoppingcart using getCart() SQL function using mysqli\_query

save the array grabbed as a string to send to JavaScript

html:

Foxy Photography Title

body

JavaScript:

include removefromCart.js

end JavaScript

link to shoppingCart.php

link to chatLogin.php

div id: cartContents class: scrollbar

JavaScript:

grab the username from the SESSION variable through PHP

grab the allItemsInCart string from PHP and save as JS variable

split the allItemsInCart string back into an array

for each row in allItemsInCart array:

slice the instance number out of the front of the row and save as new variable

create a string variable with the row data wrapped in html code and a button

button links to removeFromCart(instanceNum) JavaScript function

add the string variable to the DOM using document.getElementById("productList").innerHTML

increase the count variable for the loop by 1

end JavaScript

JavaScript functions

checkRequests.js

-shows all chat requests that have not been answered

make an ajax request to the database to find user chat requests

split the string grabbed into separate requests

remove all current nodes from the div element before adding new information

for each request in array grabbed:

add a child node to the userRequests DOM element ID

increase the count variable for the loop by 1

newMessage.js

-adds a message to the database

pull the value of the text box using document.getElementById("enteredText").value

create date and time stamps using Date() function

call addMessage.php to update the database with new message and date and time variablesusing xmlhttp

updateScreen.js

-shows new messages that the user has not seen yet on the screen

make an ajax request to the database to find new messages using xmlhttp to call checkDatabase.php

split the string grabbed from PHP into an array

while rows in array:

append row to DOM using document.getElementById("messageList").appendchild(node)

increase the count variable for the loop by 1

addToCart.js

-adds a product to a customers cart

use xmlhttp to send a request to addtoCart.php with the itemNum and jsUser variables

cartStatus.js

-shows all unordered carts with user name, cartID, and total price of cart

make an ajax request to the database to find user chat requests

use xmlhttp to send a request to cartStatus

split the string grabbed from PHP into an array

remove all current nodes from the div element before new ones are added

for each request in the array:

append the row as a child to the cartStatus div element

increase the count variable for the loop by 1

removeFromCart.js

-removes a line item from a users shopping cart

create an ajax request to removeFromCart.php with the instanceNum variable

wait 1 second for the ajax request to complete before reloading the page

PHP Functions

addMessage.php

-sends an SQL request to the database

include chat\_pdo.php

save all REQUEST variables as local variables

create a string for the SQL request using the newChatMessage() stored procedure

use mysqli\_query to send the request to the database

chat\_pdo.php

-saves the username, password, and address to the database

save a mysqli\_connect object as $pdo variable

save address, username, password, database name as $pdo2 variable

checkDatabase.php

-calls the lastMessage procedure from the database to pull all unseen messages from the database

include chat\_pdo.php

save REQUEST variables as local variables

create a new array called $messages

create a string to call the lastMessage SQL stored procedure

send the request with mysqli\_query

while row in array fetched back

push each row into $messages array

combine each row of the array into a string

send the string to JavaScript

checkRequests.php

-calls the checkRequests procedure from the database to display any chat request that have not been answered

include chat\_pdo.php

create an array to hold the requests

create a string variable to hold the SQL request for checkRequests()

send the request with mysqli\_fetch\_array

for each row in the request:

save each row in an array

convert the array into a string

send the string to JavaScript

addtoCart.php

-adds a product to the customers cart using SQL's addtoCart() function

include chat\_pdo.php

save REQUEST variables as local variables

create a variable to hold the SQL request for addtoCart() SQL stored procedure

send request using mysqli\_query

cartStatus.php

-calls the cartStatus() SQL procedure on the database to show all unordered carts

include chat\_pdo.php

create an array to hold the cart information

create a string to hold the SQL query to call cartStatus()

send the query using mysqli\_fetch\_array

for each row in result

push the result into the $carts array

convert the array to a string

send the string off to JavaScript

removeFromCart.php

-calls the removeFromCart() SQL stored procedure to remove a line item from a users cart

include chat\_pdo.php

save REQUEST variable as local variable

create a string variable to hold the SQL call to removeFromCart() procedure

send request with mysqli\_query