**Source code**

**Javascript**

**lib.js 2**

**Navigation.js 5**

**ItemBrowser.js 8**

**BrowserBasket.js 14**

**CMSControls.js 17**

**Admin.js 33**

**PHP**

**API.php 37**

**DBInterface.php 41**

**DELETEResponders.php 44**

**GETResponders.php 47**

**POSTResponders.php 54**

**PUTResponders.php 58**

**index.php (API) 61**

**index.php (CMS) 65**

**index.php (Customer) 66**

**CMSControls.php 67**

**BrowserBasket.php 68**

**index.php (root) 69**

**index.php (Admin) 70**

**HTML**

**Display.html 71**

**Navigation.html 72**

**CSS**

**Global.css 73**

**Infrastructure.css 75**

**ItemBrowser.css 77**

**Navigation.css 80**

**CMSControls.css 81**

**BroswerBasket.css 85**

**SQL**

**CreateTables.sql 87**

**TestData.sql 88**

**lib.js**

var debug = false; // Whether debug information should be output to the console

/\*\*

\* Outputs text to the console only if in debug mode.

\* @param {type} text

\* @returns {undefined}

\*/

function log(text) {

if (debug) {

console.log(text);

}

}

/\*\*

\* Alius for docuemnt.getElementByID

\*

\* @param {String} id

\* @returns {Element}

\*/

function getElem(id) {

return document.getElementById(id);

}

/\*\*

\* Removes element with {id} from the DOM.

\*

\* @param {String} id

\*/

function removeElem(id) {

var elem = document.getElementById(id);

elem.parentNode.removeChild(elem);

}

/\*\*

\* Creates a new DOM element with the given properties.

\*

\* @param {String} type

\* @param {String} className

\* @param {String} placeholder

\* @param value

\* @returns {Element}

\*/

function newElem(type, className, placeholder, value) {

var elem = document.createElement(type);

if (isSet(className)) {

elem.className = className;

}

if (isSet(placeholder)) {

elem.placeholder = placeholder;

}

if (isSet(value)) {

elem.value = value;

}

return elem;

}

/\*\*

\* Creates a new input[type=submit] with the given display text and action.

\*

\* @param {String} name

\* @param {function} action

\* @returns {input}

\*/

function newBtn(name, action) {

var button = document.createElement("input");

button.type = "submit";

button.value = name;

button.onclick = action;

return button;

}

/\*\*

\* Determins whether a variable is both defined and not null.

\*

\* @param {Object} variable

\* @returns {Boolean}

\*/

function isSet(variable) {

return 'undefined' !== typeof variable && variable !== null;

}

/\*\*

\* Sends an ajax request of type {mode} to {uri} with payload {data},

\* then calls {callback} with the response, request status and {parameters}.

\* If {data} is null or undefined, the request will be sent with no payload.

\*

\* @param {String} mode

\* @param {String} uri

\* @param {Object} data

\* @param {Function} callback

\* @param {Object} parameters

\*/

function ajax(mode, uri, data, callback, parameters) {

log(mode + " " + uri);

var request = new XMLHttpRequest();

request.onload = function () {

log("Status: " + request.status);

if (isSet(callback)) {

try {

var response = JSON.parse(request.responseText);

callback(response, request.status, parameters);

} catch (e) {

log("Couldn't parse response!");

log("Recieved: " + request.responseText);

}

}

};

request.open(mode, uri, true);

request.setRequestHeader('Content-Type', 'application/json');

if (isSet(data)) {

request.send(JSON.stringify(data));

} else {

request.send();

}

}

**Navigation.js**

/\*\*

\* Displays a notification with the given text and ID in the item browser.

\* If the ID is 'header' it will instead replace the current header.

\*

\* @param {String} text

\* @param {String} id

\*/

function displayNotification(text, id) {

var note = document.createElement('section');

note.className = 'Notification';

note.innerHTML = text;

if (id === 'header') {

// Replace header

var header = getElem('header');

header.innerHTML = "";

header.appendChild(note);

}

else {

// Insert into Display

note.id = id;

getElem('Display').appendChild(note);

}

}

/\*\*

\* Gets a list of categories from the server and then displayes them in the

\* navigation menu.

\*/

function getCategories() {

// Get categories from server

ajax("GET", "/687691/categories", null, displayCategories);

}

function displayCategories(categoryNames) {

// Clear existing

getElem('tabs').innerHTML = "";

// Add new

categoryNames.forEach(function (element) {

var name = element.categoryName;

addNavTab(name, function () {

// On click, search by category

getProductsByTerm("category", name, 0, true);

});

});

addSearchTab();

}

/\*\*

\* Adds a search tab to the navigation bar.

\*/

function addSearchTab() {

addNavTab("Search");

var searchTab = getElem('Search');

var searchBox = newElem('input', null, "Search", null);

searchBox.onchange = function () {

getProductsByTerm("search", searchBox.value, 0, true);

};

var searchBtn = newBtn("Search", function () {

getProductsByTerm("search", searchBox.value, 0, true);

});

searchTab.appendChild(searchBox);

searchTab.appendChild(searchBtn);

}

/\*\*

\* Creates a new tab and adds it to the navigation bar.

\*

\* @param {String} name

\*/

function addNavTab(name, action) {

if (debug) {

console.log('Adding navigation tab ' + name);

}

var container = newElem('li');

var tab = newElem('a');

tab.id = name;

// Display name (search tab should have no text)

if (name !== "Search") {

tab.textContent = name;

}

// Action on click

tab.onclick = function () {

selectTab(tab);

if (isSet(action)) {

displayNotification("Loading...","header");

action();

}

};

container.appendChild(tab);

getElem('tabs').appendChild(container);

}

/\*\*

\* Makes the given tab the selected tab. The previous tab will no longer be

\* selected.

\*

\* @param {a} tab

\*/

function selectTab(tab) {

if (debug) {

console.log("");

console.log('Clicked on tab ' + tab.id);

}

//deselect previous tab

var currentTab = getElem('currentTab');

if (currentTab !== null) {

currentTab.id = null;

}

//set current tab to list item containing this

tab.parentNode.id = "currentTab";

}

/\*\*

\* Resets the display by removing all items from the Display and removing

\* the footer.

\*/

function clearDisplay() {

getElem("footer").innerHTML = "";

getElem('Display').innerHTML = "";

}

**ItemBrowser.js**

/\*

\* Notes on implementation:

\* A 'browserItem' is the graphical HTML representation of a product visible to

\* the user. BrowserItems can be minimised, displaying only essential

\* information, or expanded, showing the full product information, images and

\* reviews. Batches of products are always loaded in minimised form to save

\* bandwidth; the more expensive full information calls are reserved for

\* products selected by the user.

\*/

var mode; // Required for product interaction

var productsPerQuery = 10; // The number of products to request per search.

/\*\*

\* Sets the opperation mode required for product interaction

\* Accepted modes: Customer - adds 'Add to basket' functionailty to items

\* CMS - adds 'Edit' functionality to items

\*

\* @param {String} value

\*/

function setMode(value) {

mode = value;

}

/\* Navigation \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

window.addEventListener('load', function () {

clearDisplay(); // Remove 'Enable javascript' placholder

displayNotification("Select a category to get started.", 'header');

getCategories();

});

/\* Listings \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*

\* Asynchronusly gets and displays products matching a term.

\* Modes are search methods supported by the server API;

\* These currently include 'category' and 'search'.

\* The existing display can optionally be cleared by setting clearCurrent.

\* The offset is the position into the query results from after which new items

\* will be returned.

\*

\* @param {String} mode

\* @param {String} term

\* @param {integer} offset

\* @param {boolean} clearCurrent

\*/

function getProductsByTerm(mode, term, offset, clearCurrent) {

var uri = "/687691/products/" + mode + "/" + term + "/" + offset + "/" + productsPerQuery;

// Display loading notification

if (clearCurrent) {

clearDisplay();

} else {

// Display after current listings

displayNotification("Loading...", 'loading');

}

var query = {mode: mode, term: term, offset: offset, clearCurrent: clearCurrent};

ajax("GET", uri, null, displayProducts, query);

}

function displayProducts(products, status, query) {

if (!query.clearCurrent) {

removeElem('loading'); // Remove 'loading...' notification

}

if (mode === 'CMS') {

displayNotification("Click on a product for more information and editing options.", 'header');

} else {

displayNotification("Click on products for more information.", 'header');

}

if (status === 404) {

log("No products found");

if (query.clearCurrent) {

displayNotification("No products found", 'header');

} else {

displayNotification("No more products found", 'noProducts');

}

return;

}

// Display products

for (var i = 0; i < products.length; i++) {

var container = createBrowserItem(products[i].productID);

drawBrowserItem(products[i], status, container);

}

// Add 'Go to top' to footer

var footer = getElem("footer");

footer.innerHTML = '<a href="#Top">Go to top</a>';

// Only if not end of results

if (products.length === productsPerQuery) {

// Add 'Load more products' to footer

var offset = query.offset + productsPerQuery;

footer.appendChild(newBtn("Load next " + productsPerQuery + " products",

function () {

getProductsByTerm(query.mode, query.term, offset, false);

}));

}

}

/\*\*

\* Creates a browserItem with no product details and adds it to the document.

\* A handle to the object is returned as a DOM article.

\*

\* @param {integer} productID

\* @returns 'article'

\*/

function createBrowserItem(productID) {

var container = document.createElement('article');

container.className = "ListedProduct Minimised";

container.expanded = false;

container.productID = productID;

//add click listener for expand/minimise

container.onclick = function () {

toggleBrowserItem(container);

};

//Add listing - must be done here to prevent DOM restructure on toggle

getElem('Display').appendChild(container);

return container;

}

/\*\*

\* Creates a HTML representation of the product details inside the container.

\*

\* @param {Product} product

\* @param {article} container

\*/

function drawBrowserItem(product, status, container) {

// Basic product info

container.innerHTML = " \

<img class='ProductThumbnail' src=" + product.thumbnail + "> \

<h1>" + product.productName + "</h1> \

<ul class='ProductDetailsList'> \

<li>Stock: " + product.stock + "</li> \

<li>Price: Â£" + product.price + "</li> \

<li>Delivery: " + product.deliveryType + "</li> \

</ul> \

<p>" + product.description + "</p>";

// Full info if required

if (container.expanded) {

// Display additional images

if (product.images.length !== 0) {

var imageContainer = newElem("section", "Images");

for (var i = 0; i < product.images.length; i++) {

imageContainer.innerHTML += "<img src=" + product.images[i] + ">";

}

container.appendChild(imageContainer);

}

// List reviews

if (product.reviews.length !== 0) {

var reviewsContainer = document.createElement("section");

reviewsContainer.className = "Reviews";

reviewsContainer.innerHTML = "<h1>Reviews</h1>";

for (var i = 0; i < product.reviews.length; i++) {

var review = product.reviews[i];

reviewsContainer.innerHTML += " \

<article> \

<h2>" + review.username + "</h2> \

<p>" + review.review + "</p> \

</article>";

}

container.appendChild(reviewsContainer);

}

// Add interaction button

if (mode === "customer") {

var quantity = null;

var quantityIn = newElem('input', 'TwoButton', "Quantity");

container.appendChild(quantityIn);

// Validate input (must be natural number)

quantityIn.onchange = function () {

var valid = true;

try { // Test if numeric

quantity = parseInt(quantityIn.value);

} catch (e) {

valid = false;

}

valid = valid && quantity > 0;

if (!valid) {

quantityIn.className = 'TwoButton InvalidInput';

quantity = null;

} else {

quantityIn.className = 'TwoButton';

}

};

// Prevent item toggling

quantityIn.onclick = function (e) {

e.stopPropagation();

};

// "Add to Basket" button

var addToBasketButton = newBtn("Add to Basket", function (e) {

// Prevent item toggling

e.stopPropagation();

if (quantity !== null) {

addToBasket(product, quantity);

// Reset input

quantity = null;

quantityIn.value = "";

}

});

addToBasketButton.className = 'TwoButton';

container.appendChild(addToBasketButton);

} else if (mode === "CMS") {

// "Edit" button

container.appendChild(newBtn("Edit", function () {

editItem(product);

}));

}

}

}

/\*\*

\* Toggles a browserItem between being minimised and expanded.

\*

\* @param {article} container

\*/

function toggleBrowserItem(container) {

log("");

log("Toggling browser item " + container.productID);

container.expanded = !container.expanded;

if (container.expanded) {

container.className = "ListedProduct Expanded";

// Get expanded product data

var uri = "/687691/products/" + container.productID;

ajax("GET", uri, null, drawBrowserItem, container);

} else {

container.className = "ListedProduct Minimised";

}

}

**BrowserBasket.js**

var username = "Test User 1";

var address = "Test Address 1";

var basket = [];

var totalPrice = 0.0; // Running tally of cost in basket

var orderCost = 0.0; // Calculated cost of purchase after checking stock

window.addEventListener('load', function () {

setMode("customer");

});

/\*\*

\* Adds the given ammount to the total price, negative numbers are acceptable.

\*

\* @param decimal amount

\*/

function updateTotalPrice(amount) {

totalPrice += amount;

getElem("totalPrice").innerHTML = 'Total price Â£' + totalPrice.toFixed(2);

}

/\*\*

\* Adds a quantity of a product to the basket then updates the basket display.

\*

\* @param product product

\* @param integer quantity

\*/

function addToBasket(product, quantity) {

log("");

log("Adding " + quantity + " of '"

+ product.productName + "' to basket");

var cost = product.price \* quantity;

updateTotalPrice(cost);

// Store cut down form of product data in basket

var details = new Object();

details.quantity = quantity;

details.productID = product.productID;

details.productName = product.productName;

details.cost = cost;

var BasketItems = getElem("BasketItems");

// Draw

var item = document.createElement("article");

item.className = "BasketItem";

item.innerHTML = ' \

<h1>' + product.productName + '</h1> \

<ul class="BasketDetails"> \

<li>Quantity: ' + quantity + '</li> \

<li>Price: Â£' + cost.toFixed(2) + '</li> \

</ul> \

';

// // Define edit button

// item.appendChild(newBtn("Edit", function () {

// // TODO should allow editing of quantity

// }));

// Define remove button

item.appendChild(newBtn("Remove", function () {

log("");

log("Removing '" + product.productName + "' from basket");

BasketItems.removeChild(item);

updateTotalPrice(0 - cost);

basket.splice(basket.indexOf(details), 1); // Remove from basket

}));

//Combine elements and add to basket

BasketItems.appendChild(item);

basket.push(details);

orderCost = 0.0; // Must be recalculated when making purchase

}

/\*\*

\* Creates new purchases for each item in the basket, then resets the basket.

\*/

function placeOrder() {

log("");

log("Placing order");

var orderCost = 0;

for (var i = 0; i < basket.length; i++) {

var item = basket[i];

log("Creating purchase for " + item.quantity +

" items with productID: " + item.productID);

// Get details

var purchase = new Object();

purchase.productID = item.productID;

purchase.username = username;

purchase.quantity = item.quantity;

// Send to server

ajax("POST", "/687691/purchases", purchase, purchaseSummary, item);

}

// Reset basket & display

clearDisplay();

getElem("BasketItems").innerHTML = "";

basket = [];

updateTotalPrice(0 - totalPrice); // Reset total cost to 0

}

/\*\*

\* Displays a notification summarising a purchase.

\* Parameter r is an unused placeholder.

\*

\* @param {Object} r

\* @param {int} status

\* @param {Object} item

\*/

function purchaseSummary(r, status, item) {

if (status === 200) { // Purchase made

displayNotification("Successfully purchased " + item.quantity

+ " of '" + item.productName + "'.");

orderCost += item.cost;

displayNotification('Final cost of order: Â£'

+ orderCost.toFixed(2) + ". <br>Products will be delivered to '"

+ address + "'.", 'header');

} else { // Not purchased

displayNotification("Unable to purchase " + item.quantity

+ " of '" + item.productName

+ "', desired quantity is not available.");

}

}

**CMSControls.js**

window.addEventListener('load', function () {

setMode("CMS");

});

/\*\*

\* Displays an editing form for {product}.

\*

\* @param {Object} product

\*/

function editItem(product) {

// Setup editor

var container = document.createElement('article');

container.className = "ListedProduct EditingProduct";

// Create fields

var nameIn = newEditField("Product name", product.productName);

var stockIn = newEditField("Stock", product.stock);

stockIn.id = "stockDisplay"; // To show updates made via stock manager

var priceIn = newEditField("Price", product.price);

var deliveryIn = newEditField("Delivery type", product.deliveryType);

var descIn = newElem('textarea', null, "Product description", product.description);

// Thumbnail

container.innerHTML = "<img id='thumb' class='ProductThumbnail' src='" + product.thumbnail + "'>";

// Product name

var name = newElem('h1');

name.appendChild(nameIn);

container.appendChild(name);

// Product details

var details = newElem('ul', 'ProductDetailsList');

appendDetailInput(details, 'Stock ', stockIn);

appendDetailInput(details, 'Price Â£', priceIn);

appendDetailInput(details, 'Delivery: ', deliveryIn);

container.appendChild(details);

// Description

var description = newElem('p');

description.appendChild(descIn);

container.appendChild(description);

// Save button

var save = newBtn("Save", function () {

// Save only changed values

var changes = new Object();

if (nameIn.value !== product.productName) {

changes.productName = nameIn.value;

}

if (stockIn.value !== product.stock) {

changes.stock = stockIn.value;

}

if (deliveryIn.value !== product.deliveryType) {

changes.deliveryType = deliveryIn.value;

}

if (priceIn.value !== product.price) {

changes.price = priceIn.value;

}

if (descIn.value !== product.description) {

changes.description = descIn.value;

}

saveChanges(product.productID, changes);

});

save.className = "ThreeButton";

// Cancel button

var cancel = newBtn("Cancel", function () {

// Reset display then show unmodified product

clearDisplay();

displayNotification('\

Changes not saved.<br>\

Current state of the product is shown below.', 'header');

toggleBrowserItem(createBrowserItem(product.productID));

});

cancel.className = "ThreeButton";

// Unlist button

var unlist = newBtn("Remove listing for this product", function () {

log("");

log("Unlisting " + product.productName);

// Send unlist request

var uri = "/687691/products/" + product.productID;

ajax("PUT", uri, {listed: false}, unlistResponse, product.productName);

});

unlist.className = "ThreeButton";

// Combine

container.appendChild(save);

container.appendChild(cancel);

container.appendChild(unlist);

attatchImageManager(container, product);

attatchCategoryManager(container, product.productID);

attachStockManager(container, product.productID);

// Display

clearDisplay();

displayNotification("\

All purple text is editable. <br>\

No changes will be made until the save button is pressed. <br>\

Press cancel at any time to discard your changes.", 'header');

getElem('Display').appendChild(container);

}

/\*\*

\* Creates a new editing field with {placeholder} text and intial {value}.

\*

\* @param {String} placeholder

\* @param {String} value

\* @returns {input}

\*/

function newEditField(placeholder, value) {

return newElem("input", "EditField", placeholder, value);

}

/\*\*

\* Appends an li containing both the {text} and {input} field provided to

\* the {parent} container.

\*

\* @param {element} parent

\* @param {String} text

\* @param {input} input

\*/

function appendDetailInput(parent, text, input) {

var detail = newElem('li');

detail.appendChild(document.createTextNode(text));

detail.appendChild(input);

parent.appendChild(detail);

}

/\*\*

\* Displays a notification regarding the success or failure of unlisting

\* a product.

\*

\* @param {String} response

\* @param {int} status

\* @param {String} productName

\*/

function unlistResponse(response, status, productName) {

clearDisplay();

if (status === 200) {

// Reset display then show notification

displayNotification("Unlist sucessful, '" + productName +

"' will no longer be listed for purchase or editing.", "header");

} else {

displayNotification(response, "header");

}

}

/\*\*

\* Saves a collection of {changes} to product with {productID}. Then reloads

\* the editor with the new representation of the product.

\* {changes} should be an object with keys relating to the products field names.

\* Only the specified fields will be changed.

\*

\* @param {int} productID

\* @param {Object} changes

\*/

function saveChanges(productID, changes) {

if (changes === null) { // Don't send empty requests

return;

}

// Products should always be listed after editing

changes.listed = true;

log("");

log("Sending edit request");

// Save changes

ajax("PUT", "/687691/products/" + productID, changes);

// Show in product editor

ajax("GET", "/687691/products/" + productID, null, editItem);

}

/\*\*

\* Posts a new default product to the server.

\*/

function createProduct() {

log("");

log("Creating product");

// Create new

ajax("POST", "/687691/products", null, function (productID) {

// Get for editing

ajax("GET", "/687691/products/" + productID, null, function (product) {

editItem(product);

});

});

}

/\*\*

\* Appends an image manager for {product} to {parent}.

\*

\* @param {section} parent

\* @param {Object} product

\*/

function attatchImageManager(parent, product) {

// Current images

var imageContainer = newElem("section", "Images");

imageContainer.id = 'imageList';

// Display

var container = newElem("section", "ManagerPanel ImageManager");

container.innerHTML = "\

<h1>Images</h1> \

<p>Select an image for additional options or click 'Upload Image' \

to upload more images for this product.</p>";

container.appendChild(imageContainer);

appendImageUploadControls(container, product.productID);

appendSingleImageControls(container, product.productID);

parent.appendChild(container);

displayImages(product.images, 200, imageContainer);

}

/\*\*

\* Displays a response to uploading an image to the server and refreshes the

\* displayed images on the image editor.

\*

\* @param {String} response

\* @param {int} status

\* @param {int} productID

\*/

function imageUploadResponse(response, status, productID) {

// Display server response in image preview container

getElem("imageUploadPreview").innerHTML = "<p>" + response + "</p>";

// Reload image display

var uri = "/687691/products/" + productID + "/images";

ajax('GET', uri, null, displayImages, getElem('imageList'));

}

/\*\*

\* Sets the thumbnail for product with {productID} to {src}, then

\* refreshes the displayed thumbnail on the editor.

\*

\* @param {int} productID

\* @param {String} src

\*/

function changeThumbnail(productID, src) {

var data = {thumbnail: src};

ajax("PUT", "/687691/products/" + productID, data, getThumbnail, productID);

}

/\*\*

\* Refreshed the thumbnail displayed in the editor to that stored for product

\* with {productID} on the server.

\* Parameters {r} and {s} are unused placeholders.

\*

\* @param {String} r

\* @param {int} s

\* @param {int} productID

\*/

function getThumbnail(r, s, productID) {

log("");

log("Refreshing displayed thumbnail");

// Get from server

ajax("GET", "/687691/products/" + productID, null, function (product) {

// Display

getElem('thumb').src = product.thumbnail;

});

}

/\*\*

\* Returns the file name (and extension) of the currently selected image in

\* the image selector on the image manager.

\*

\* @returns {String}

\*/

function selectedFileName() {

var filePath = getElem('selectedImage').src;

return filePath.split('/').pop();

}

/\*\*

\* Appends editing controls for a single image of product with {productID}

\* to the {parent} container. Including 'Delete image' and 'Make thumbnail' buttons.

\*

\* @param {section} parent

\* @param {int} productID

\*/

function appendSingleImageControls(parent, productID) {

var container = newElem('section', 'hidden');

container.id = 'singleImageControls';

// Make thumbnail button

container.appendChild(newBtn("Use this image\nas product thumbnail", function () {

// Send update request

changeThumbnail(productID, selectedFileName());

}));

// Delete image button

container.appendChild(newBtn("Remove\nthis image", function () {

// Send delete request

var deluri = "/687691/products/" + productID + "/images/" + selectedFileName();

ajax("DELETE", deluri, null, function () {

// Reload image display

var geturi = "/687691/products/" + productID + "/images";

ajax('GET', geturi, null, displayImages, getElem('imageList'));

});

// Check if deleted image is thumbnail

if (getElem('selectedImage').src === getElem('thumb').src) {

// Remove thumbnail pointer to deleted image

changeThumbnail(productID, null);

}

}));

parent.appendChild(container);

}

/\*\*

\* Displays the images provided in the container provided along with an

\* 'Upload Image' icon which when clicked unhides editing controls.

\* Clicking on any other image in the container will hide upload controls

\* and instead show controls relating to that image.

\* images should be an array of image src values.

\*

\* @param {array(String)} images

\* @param {int} status

\* @param {section} imageContainer

\*/

function displayImages(images, status, imageContainer) {

log("");

log("Displaying images");

log(images);

// Reset

imageContainer.innerHTML = "";

// Display 'UPLOAD IMAGE' image

var uploadImage = clickableImage('../API/Images/res/uploadImage.png',

function () {

// Hide single image controls

getElem('singleImageControls').className = 'controls hidden';

// Unhide upload controls

getElem('imageUploadControls').className = 'controls';

});

uploadImage.id = 'selectedImage';

imageContainer.appendChild(uploadImage);

// Display product images

for (var i = 0; i < images.length; i++) {

imageContainer.appendChild(clickableImage(images[i], function () {

// Hide upload controls

getElem('imageUploadControls').className = 'controls hidden';

// Show single image controls

getElem('singleImageControls').className = 'controls';

}));

}

}

/\*\*

\* Returns an img DOM element that can be selected. Only one clickable image

\* can be selected at a time. When the image is clicked, the action will be run.

\*

\* @param {String} src

\* @param {Function} action

\* @returns {img}

\*/

function clickableImage(src, action) {

var image = newElem('img');

image.src = src;

image.onclick = function () {

// Make this the selected image

selectImage(this);

action();

};

return image;

}

/\*\*

\* Makes the given image the selected image. The previous image will no longer

\* be selected.

\*

\* @param {img} image

\*/

function selectImage(image) {

log("");

log('Clicked on image ' + image.src);

// Deselect previous image

var currentTab = getElem('selectedImage');

if (currentTab !== null) {

currentTab.id = null;

}

// Select the new image

image.id = "selectedImage";

}

/\*\*

\* Appends image upload controls for product with {productID} to {parent}.

\*

\* @param {section} parent

\* @param {int} productID

\*/

function appendImageUploadControls(parent, productID) {

var previewText = "<p>Image to be uploaded:</p>";

var encodedImage;

// Image preview

var preview = newElem('section', 'right');

preview.id = "imageUploadPreview";

preview.innerHTML = previewText;

// File selector

var imageInput = newElem('input');

imageInput.type = 'file';

imageInput.onchange = function () {

encodedImage = null;

var file = imageInput.files[0];

if (file.type.match(/image.\*/)) { // Check if actual image

log("");

log("Image selected for upload:");

log(file.name);

var reader = new FileReader();

reader.onloadend = function () {

// Get image data

encodedImage = reader.result;

// Show preview

preview.innerHTML = previewText + "<img src='" + encodedImage + "'>";

};

reader.readAsDataURL(file);

} else {

preview.innerHTML = previewText + "<p>File not supported!</p>";

}

};

// Upload button

var upload = newBtn("Upload image", function () {

if (encodedImage !== null) {

var image = encodedImage.split(",");

encodedImage = null;

var fileName = imageInput.files[0].name;

log("Uploading image: " + fileName);

// Send to server

var uri = "/687691/products/" + productID + "/images";

ajax("POST", uri, [fileName, image[1]], imageUploadResponse, productID);

} else {

log("No image selected");

preview.innerHTML = previewText + "<p>No image selected</p>";

}

});

// Display

var left = newElem('section', "left");

left.innerHTML = '<p>Select an image to upload</p>';

left.appendChild(imageInput);

left.appendChild(upload);

var controls = newElem('section', 'controls');

controls.id = 'imageUploadControls';

controls.appendChild(left);

controls.appendChild(preview);

parent.appendChild(controls);

}

/\*\*

\* Attatches stock managing controls to the parent container.

\*

\* @param {section} parentContainer

\* @param {int} productID

\*/

function attachStockManager(parentContainer, productID) {

var container = newElem("section", "ManagerPanel stockManager");

container.innerHTML = "<h1>Stock</h1>";

// Current stock display

var stockContainer = newElem('p');

stockContainer.id = "currentStock";

updateStockDisplay(productID);

// Value input

var change = newElem("input", null, "Amount");

change.id = "alterStock";

// Add/subtract buttons

var add = newBtn("Add amount to stock", function () {

updateStock(productID, change.value);

});

var subtract = newBtn("Subtract amount from stock", function () {

updateStock(productID, 0 - parseInt(change.value));

});

change.className = "ThreeButton";

add.className = "ThreeButton";

subtract.className = "ThreeButton";

// Combine

container.appendChild(stockContainer);

container.appendChild(subtract);

container.appendChild(change);

container.appendChild(add);

parentContainer.appendChild(container);

}

/\*\*

\* Updates stock for product with {productID} by {amount}.

\* {amount} can be either positive or negative.

\*

\* @param {int} productID

\* @param {int} amount

\*/

function updateStock(productID, amount) {

log("");

log("Sending stock update");

var uri = "/687691/products/" + productID + "/stock";

ajax("PUT", uri, amount, function () {

updateStockDisplay(productID);

getElem('alterStock').value = "";

});

}

/\*\*

\* Gets the current stock value for product with {productID} and displays it in

\* the 'current stock' display.

\*

\* @param {int} productID

\*/

function updateStockDisplay(productID) {

var uri = "/687691/products/" + productID + "/stock";

ajax("GET", uri, null, function (stock) {

getElem('currentStock').innerHTML = "Current Stock for this product: " + stock;

getElem("stockDisplay").value = stock;

});

}

/\*\*

\* Attaches category managing controls for product with {productID}

\* to {parent} container.

\*

\* @param {section} parent

\* @param {int} productID

\*/

function attatchCategoryManager(parent, productID) {

var manager = newElem("section", "CategoryManager");

manager.innerHTML = "<h1>Categories</h1>";

// Selected categories

var left = newElem('section', 'left');

left.innerHTML = "<h2>Selected categories</h2>";

left.appendChild(swappingSelect('selectedCats', 'availableCats', productID, removeFromCategory));

// Available categories

var right = newElem('section', 'right');

right.innerHTML = "<h2>Available categories</h2>";

right.appendChild(swappingSelect('availableCats', 'selectedCats', productID, addToCategory));

// Transfer buttons

var mid = newElem('section', 'mid');

mid.appendChild(newBtn('<<', function () {

moveSelected('availableCats', 'selectedCats', addToCategory, productID);

}));

mid.appendChild(newBtn('>>', function () {

moveSelected('selectedCats', 'availableCats', removeFromCategory, productID);

}));

// Combine

manager.appendChild(left);

manager.appendChild(mid);

manager.appendChild(right);

parent.appendChild(manager);

// Get data

getProductCategories(productID);

}

/\*\*

\* Creates an HTML select element with {id} for which when an option is double

\* clicked, it is transfered to another select with {partnerID} and the

\* {action} function is run with {productID} as a parameter.

\*

\* @param {String} id

\* @param {String} partnerID

\* @param {int} productID

\* @param {function} action

\* @returns {select}

\*/

function swappingSelect(id, partnerID, productID, action) {

var select = newElem('select');

select.id = id;

select.multiple = true;

select.ondblclick = function () { // Double click transfers selected

moveSelected(id, partnerID, action, productID);

};

return select;

}

/\*\*

\* Moves the selected option from select with {sourceID} to select with

\* {destinationID}, then runs the {action} function with {parmas} as

\* a parameter.

\*

\* @param {String} sourceID

\* @param {String} destinationID

\* @param {function} action

\* @param {Object} params

\*/

function moveSelected(sourceID, destinationID, action, params) {

var source = getElem(sourceID);

var destination = getElem(destinationID);

var selected = source.value;

source.remove(source.selectedIndex);

destination.innerHTML += "<option>" + selected + "</option>";

action(params, selected);

}

/\*\*

\* Asynchronosly gets all categories product with {productID} is in.

\*

\* @param {int} productID

\*/

function getProductCategories(productID) {

log("");

log("Getting data for product category manager");

// Get product categories

var uri = "/687691/products/" + productID + "/categories";

ajax("GET", uri, null, function (productCategories) {

// Get all categories

var uri = "/687691/categories";

ajax("GET", uri, null, displayProductCategories, productCategories);

});

}

/\*\*

\* Displays all categories a product is currently in and all the avalible

\* categories it can be put in on the category manager.

\*

\* @param {array(String)} allCategories

\* @param {int} status

\* @param {array(String)} productCategories

\*/

function displayProductCategories(allCategories, status, productCategories) {

// List current categories

var current = "";

productCategories.forEach(function (elem) {

current += "<option>" + elem.categoryName + "</option>";

});

getElem('selectedCats').innerHTML = current;

// Filter to get only categories product is not in

var availableCategories = filterCategories(allCategories, productCategories);

// List available categories

var availible = "";

availableCategories.forEach(function (elem) {

availible += "<option>" + elem.categoryName + "</option>";

});

getElem('availableCats').innerHTML = availible;

}

/\*\*

\* Returns a filtered list of categories which were in {source} but not {toRemove}.

\*

\* @param {array(Object)} source

\* @param {array(Object)} toRemove

\* @returns {array(Object)}

\*/

function filterCategories(source, toRemove) {

var results = [];

source.forEach(function (elem) {

var keep = true;

toRemove.forEach(function (elem2) {

if (elem.categoryName === elem2.categoryName) {

keep = false;

}

});

if (keep) {

results.push(elem);

}

});

return results;

}

/\*\*

\* Adds product with {productID} to category with {categoryName}.

\*

\* @param {int} productID

\* @param {String} categoryName

\*/

function addToCategory(productID, categoryName) {

log("");

log("Adding product to category");

ajax("POST", "/687691/products/" + productID + "/categories", categoryName);

}

/\*\*

\* Removes product with {productID} from category with {categoryName}.

\*

\* @param {int} productID

\* @param {String} categoryName

\*/

function removeFromCategory(productID, categoryName) {

log("");

log("Removing product from category");

ajax("DELETE", "/687691/products/" + productID + "/categories/" + categoryName);

}

/\*\*

\* Refreshes the displayed list of all categories on the allCategories manager.

\*/

function updateCatList() {

log("updating category list");

ajax("GET", "/687691/categories", null, function (catNames) {

getElem('alterCat').value = "";

var list = getElem('catList');

list.innerHTML = "";

catNames.forEach(function (element) {

var elem = newElem('li');

elem.innerHTML = element.categoryName;

elem.onclick = function () {

getElem('alterCat').value = element.categoryName;

};

list.appendChild(elem);

});

//reload nav bar

getCategories();

});

}

/\*\*

\* Displays the allCategories manager which allows creation and deletion

\* of categories.

\*/

function manageAllCategories() {

clearDisplay();

displayNotification("\

All categories are listed below.<br>\

To delete a category, enter it's name and click delete.<br>\

To create a new category, enter it's name an click create.", 'header');

var container = newElem('article', "ListedProduct Manager");

container.innerHTML = "<h1>Manage Categories</h1>";

var catList = newElem('ul');

catList.id = "catList";

updateCatList();

var change = newElem("input", "ThreeButton", "Category name");

change.id = "alterCat";

container.appendChild(catList);

container.appendChild(change);

var add = newBtn("Create category", function () {

createCategory(change.value);

});

add.className = 'ThreeButton';

container.appendChild(add);

var remove = newBtn("Delete category", function () {

deleteCategory(change.value);

});

remove.className = 'ThreeButton';

container.appendChild(remove);

getElem('Display').appendChild(container);

}

/\*\*

\* Creates a new category with {categoryName}.

\*

\* @param {String} categoryName

\*/

function createCategory(categoryName) {

ajax("POST", "/687691/categories", categoryName, updateCatList);

}

/\*\*

\* Deletes category of {categoryName}.

\*

\* @param {String} categoryName

\*/

function deleteCategory(categoryName) {

ajax("DELETE", "/687691/categories/" + categoryName, null, updateCatList);

}

**Admin.js**

var lowStockThreashold = 30;

/\* Navigation \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

window.addEventListener('load', function () {

clearDisplay(); // Remove 'Enable javascript' placholder

displayNotification("Select a tab to get started.", 'header');

displayAdminTabs();

});

function displayAdminTabs() {

// Clear existing

getElem('tabs').innerHTML = "";

// Add new

addNavTab("Stock overview", function () {

// Show stock manager

ajax("GET", "/687691/stock", null, manageStock);

});

addNavTab("Deliveries overview", function () {

// Show delivery manager

ajax("GET", "/687691/purchases/pending", null, manageDeliveries);

});

// addNavTab("Site options", manageSite);

}

function manageStock(results) {

// Display all products ordered by stock low to high, highlight

// products with stock be low optional threashold in red.

// Display instruction text

displayNotification("\

The current stock for all products is listed below. <br>\

Products highlighted in red are out of stock,\

products in orange may soon be out of stock.\

", "header");

// Create table and column headers

var table = newElem('table');

table.innerHTML = "\

<tr> \

<th>Product name</th> \

<th>Current stock</th> \

</tr>";

for (var i = 0; i < results.length; i++) {

// Get properties of product

var name = results[i].productName;

var stock = results[i].stock;

// Highlight row based on stock level

var className = 'InStock';

if (stock === 0) {

className = 'OutOfStock';

} else if (stock <= lowStockThreashold) {

className = 'LowStock';

}

// Add to table

table.appendChild(tableRow([name, stock], className));

}

// Display

clearDisplay();

getElem('Display').appendChild(table);

}

/\*\*

\* Returns a table row element of {className} containing {values}.

\*

\* @param {array} values

\* @param {String} className

\* @returns {tr}

\*/

function tableRow(values, className) {

log("");

log("Adding table row:");

log(values + " class: " + className);

// Create row

var row = newElem('tr', className);

// Insert values

for (var i = 0; i < values.length; i++) {

var value = values[i];

var col = newElem('td');

// Check if value is simple or object

if (typeof value === "object") {

// Supports DOM elements in table

col.className = 'ObjectCell';

col.appendChild(value);

} else {

col.innerHTML = value;

}

row.appendChild(col);

}

return row;

}

function manageDeliveries(results) {

// List all pending purchases

// Display instruction text

displayNotification("\

All pending orders awaiting dispatch are listed below.", "header");

// Create table and column headers

var table = newElem('table', 'Striped');

table.innerHTML = "\

<tr> \

<th>Product name</th> \

<th>Quantity ordered</th> \

<th>Delivery address</th> \

<th class='ObjectCell'></th>\

</tr>";

for (var i = 0; i < results.length; i++) {

// Get properties of product

var name = results[i].productName;

var quantity = results[i].quantity;

var address = results[i].address;

var resolveButton = newBtn("Mark as dispatched",

markDispatched.bind(null, results[i].purchaseID));

// Add to table

table.appendChild(tableRow([name, quantity, address, resolveButton]));

}

// Display

clearDisplay();

getElem('Display').appendChild(table);

}

/\*\*

\* Marks a purchase as dispatched then reloads the delivery manager.

\*

\* @param {int} purchaseID

\*/

function markDispatched(purchaseID) {

var uri = "/687691/purchases/" + purchaseID;

ajax("PUT", uri, {processed: true}, function () {

// Refresh delivery manager

ajax("GET", "/687691/purchases/pending", null, manageDeliveries);

});

}

**API.php**

<?php

require\_once 'DBInterface.php';

include\_once 'GETResponders.php';

include\_once 'POSTResponders.php';

include\_once 'PUTResponders.php';

include\_once 'DELETEResponders.php';

// Root image directory relative to JS

define('ImgRoot', '../API/Images/');

// Removes URL encoding from string (eg. %20 for spaces)

$fullURI = urldecode(getenv('REQUEST\_URI'));

//URI starts with /API/ so uri[0] = "" and uri[1] = "API"

$uri = array\_slice(explode("/", $fullURI), 2); //array representing URI

$method = getenv('REQUEST\_METHOD');

$response = null;

$requestBody;

if ($method === "POST" || $method === "PUT") {

$requestBody = json\_decode(file\_get\_contents('php://input'));

}

if ($uri[0] === "categories") {

if (count($uri) === 1) {

//URI format /categories/

if ($method === "GET") {

$response = getCategories();

} elseif ($method === "POST") {

$response = addCategory($requestBody);

}

}

//URI format /categories/$name

elseif (count($uri) === 2 && $method === "DELETE") {

$response = deleteCategory($uri[1]);

}

} elseif ($uri[0] === "products") {

switch (count($uri)) {

case 1:

//URI format /products

if ($method === "POST") {

$response = addProduct($requestBody);

}

break;

case 2:

//URI format /products/$id

if ($method === "GET") {

$response = getItem($uri[1], true);

} elseif ($method === "PUT") {

$response = editProduct($uri[1], $requestBody);

} elseif ($method === "DELETE") {

$response = deleteProduct($uri[1]);

}

break;

case 3:

//URI format /products/$id/reviews

if ($uri[2] === "reviews" && $method === "POST") {

$response = addReview($uri[1], $requestBody);

}

//URI format /products/$id/images

elseif ($uri[2] === "images") {

if ($method === "GET") {

$response = getImages($uri[1]);

} elseif ($method === "POST") {

$response = addImage($uri[1], $requestBody);

}

}

//URI format /products/$id/stock

elseif ($uri[2] === "stock") {

if ($method === "GET") {

$response = getStock($uri[1]);

} elseif ($method === "PUT") {

$response = addStock($uri[1], $requestBody);

}

}

//URI format /products/$id/categories

elseif ($uri[2] === "categories") {

if ($method === "GET") {

$response = getProductCategories($uri[1]);

} elseif ($method === "POST") {

$response = addToCategory($uri[1], $requestBody);

}

}

break;

case 4:

//URI format /products/$id/reviews/$username

if ($uri[2] === "reviews") {

if ($method === "PUT") {

$response = editReview($uri[1], $uri[3], $requestBody);

} elseif ($method === "DELETE") {

$response = deleteReview($uri[1], $uri[3]);

}

}

//URI format /products/$id/categories/$name

elseif ($uri[2] === "categories" && $method === "DELETE") {

$response = removeFromCategory($uri[1], $uri[3]);

}

//URI format /products/$id/images/$fileName

elseif ($uri[2] === "images" && $method === "DELETE") {

$response = deleteImage($uri[1], $uri[3]);

}

break;

case 5:

//URI format /products/category/$term/$offset/$number

if ($uri[1] === "category") {

if ($method === "GET") {

$response = getProductsByCategory($uri[2], $uri[3], $uri[4]);

}

}

//URI format /products/search/$term/$offset/$number

elseif ($uri[1] === "search") {

if ($method === "GET") {

$response = getProductsBySearch($uri[2], $uri[3], $uri[4]);

}

}

break;

}

} elseif ($uri[0] === "purchases") {

//URI format /purchases

if (count($uri) === 1) {

if ($method === "POST") {

$response = addPurchase($requestBody);

}

} elseif (count($uri) === 2) {

//URI format /purchases/pending

if ($uri[1] === "pending") {

if ($method === "GET") {

$response = getPendingPurchases();

}

}

//URI format /purchases/$id

else if ($method === "PUT") {

$response = editPurchase($uri[1], $requestBody);

}

}

} elseif ($uri[0] === "stock") {

//URI format /stock

if (count($uri) === 1) {

if ($method === "GET") {

$response = getAllStock();

}

}

}

if ($response === null) {

header("Content-Type:text/plain; charset=UTF-8");

echo "Invalid request!";

} else {

header("Content-Type: application/json; charset=UTF-8");

echo json\_encode($response);

}

**DBInterface.php**

<?php

class DBConnection {

// Settings

const DBName = "CMS687691";

const DBServername = "127.0.0.1";

const DBUsername = "root";

const DBPassword = "";

const UseTestData = true;

private $db; // Database handle

/\*\*

\* Constructs a DBConnection object by establishing a database connection,

\* or triggering the creation of a new database if none exists matching the

\* database name specified above.

\*/

public function \_\_construct() {

// Establish connection

$this->db = new PDO("mysql:host=" . self::DBServername, self::DBUsername, self::DBPassword);

// Create database if not exists

if (!$this->dbExists()) {

$this->createDB(self::UseTestData);

}

// Configure connection & select database for use

$this->db->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

$this->db->setAttribute(PDO::ATTR\_EMULATE\_PREPARES, false); // for LIMIT

$this->db->exec("USE " . self::DBName . ";");

}

/\*\*

\* Returns a boolean, whether a database matching the database name

\* specified above exists.

\*

\* @return boolean

\*/

public function dbExists() {

$matches = $this->db->query("SHOW DATABASES LIKE '" . self::DBName . "';");

return $matches->rowCount() > 0;

}

/\*\*

\* Prepairs and executes a query on the database and returns the results as

\* an array of objects.

\*

\* @param String $query

\* @param Array $vars

\* @return Array(Object)

\*/

public function select($query, $vars) {

try {

$result = $this->db->prepare($query);

$result->execute($vars);

return $result->fetchAll(PDO::FETCH\_OBJ);

} catch (PDOException $e) {

return ("Failed to run query: " . $e->getMessage());

}

}

/\*\*

\* Prepairs and executes a query on the database and returns a messege,

\* either 'Success' or an error log as a string.

\*

\* @param String $query

\* @param Array $vars

\* @return String

\*/

public function run($query, $vars) {

try {

$result = $this->db->prepare($query);

$result->execute($vars);

return "Success";

} catch (PDOException $e) {

return ("Failed to run query: " . $e->getMessage());

}

}

/\*\*

\* Returns the last ID inserted to the database

\*

\* @return String

\*/

public function lastID() {

return $this->db->lastInsertId();

}

/\*\*

\* Creates a new database of the name specified above. Optionally test data

\* can be added to the database upon creation. If creation is unsucessful,

\* an error report will be returned as a string.

\*

\* @param boolean $useTestData

\* @return String

\*/

public function createDB($useTestData) {

try {

// Initialise database

$this->db->exec("CREATE DATABASE " . self::DBName . ";");

$this->db->exec("USE " . self::DBName . ";");

// Setup tables

$this->db->exec(file\_get\_contents('DB\_setup/CreateTables.sql'));

if ($useTestData) {

// Insert test data if desired

$this->db->exec(file\_get\_contents('DB\_setup/TestData.sql'));

}

} catch (PDOException $e) {

return("Failed to create database: " . $e->getMessage());

}

}

/\*\*

\* Terminates the database connection.

\*/

public function close() {

$this->db = null;

}

}

**DELETEResponsers.php**

<?php

/\*\*

\* Removes all products from, then deletes category of $categoryName.

\*

\* @param {String} $categoryName

\* @return {String}

\*/

function deleteCategory($categoryName) {

$categoryID = getCategoryID($categoryName);

if (!is\_numeric($categoryID)) {

return $categoryID;

}

$db = new DBConnection;

$query = "

DELETE FROM ProductCategory

WHERE categoryID = ?";

$result = $db->run($query, array($categoryID));

$query = "

DELETE FROM Category

WHERE categoryID = ?";

$result = $db->run($query, array($categoryID));

$db->close();

if ($result !== "Success") {

http\_response\_code(500); // Internal Server Error

}

return $result;

}

/\*\*

\* Removes product with $productID from category of $categoryName.

\*

\* @param {int} $productID

\* @param {String} $categoryName

\* @return {String}

\*/

function removeFromCategory($productID, $categoryName) {

$categoryID = getCategoryID($categoryName);

if (!is\_numeric($categoryID)) {

return $categoryID;

}

$db = new DBConnection;

$query = "DELETE FROM ProductCategory WHERE productID = ? AND categoryID = ?";

$result = $db->run($query, array($productID, $categoryID));

$db->close();

if ($result !== "Success") {

http\_response\_code(500); // Internal Server Error

}

return $result;

}

/\*\*

\* Deletes image of product with $productID of name $fileName.

\*

\* @param {int} $productID

\* @param {String} $fileName

\* @return {String}

\*/

function deleteImage($productID, $fileName) {

if (unlink(ImgRoot . "$productID/$fileName")) {

return "Successfully deleted $fileName";

} else {

http\_response\_code(500); // Internal Server Error

return "Failed to delete $fileName";

}

}

/\*\*

\* Deletes all images of product with $productID and it's image directory.

\*

\* @param {int} $productID

\* @return {String}

\*/

function deleteAllImages($productID) {

$imgDir = ImgRoot . $productID;

if (file\_exists($imgDir)) {

// Delete all images of product

$images = getImages($productID);

foreach ($images as $filePath) {

unlink($filePath);

}

// Delete products image directory

unlink($imgDir);

}

}

/\*\*

\* Deletes product with $productID and all associated images,

\* reviews and purchases.

\*

\* @param {int} $productID

\* @return {String}

\*/

function deleteProduct($productID) {

deleteAllImages($productID);

$db = new DBConnection;

$query = "

DELETE FROM Purchase

WHERE Product.productID = ?";

$result = $db->run($query, array($productID));

$query = "

DELETE FROM ProductCategory

WHERE Product.productID = ?";

$result = $db->run($query, array($productID));

$query = "

DELETE FROM Product

WHERE Product.productID = ?";

$result = $db->run($query, array($productID));

$db->close();

if ($result !== "Success") {

http\_response\_code(500); // Internal Server Error

}

return $result;

}

/\*\*

\* Deletes review of product with $productID by user with $username.

\*

\* @param {int} $productID

\* @param {String} $username

\* @return {String}

\*/

function deleteReview($productID, $username) {

//TODO

http\_response\_code(501); // Not Implemented

return "'delete review' not implemented yet";

}

**GETResponders.php**

<?php

// Details to retrive when selecting products from the database

define('ProductDetails', 'Product.productID, productName, price, deliveryType, stock, description, thumbnail');

/\*\*

\* Generates a hard-coded test product with either minimal or expanded data.

\*

\* @param {boolean} $expanded

\* @return {product}

\*/

function getTestItem($expanded) {

$imgDir = ImgRoot . 'TestProduct/';

$product = new stdClass();

$product->productName = 'Test product';

$product->price = 99.99;

$product->deliveryType = 'Free UK';

$product->stock = 9;

$product->description = 'A hard coded test product loaded through JSON.';

$product->thumbnail = $imgDir . 'teapot1.jpg';

if ($expanded) {

$product->images = array(

$imgDir . 'teapot2.jpg',

$imgDir . 'teapot3.jpg',

$imgDir . 'teapot4.jpg'

);

$product->reviews = array(

array("User1", 'Review 1 text'),

array("User2", 'Review 2 text'),

array("User3", 'Review 3 text, this is a moderate length review')

);

}

header("Content-Type: application/json; charset=UTF-8");

return(json\_encode($product));

}

/\*\*

\* Gets either the minimal or expanded data for a product.

\* Expanded includes reviews and additional images (besides the thumbnail).

\*

\* @param {int} $productID

\* @param {boolean} $expanded

\* @return {product}

\*/

function getItem($productID, $expanded) {

if ($productID === "TestProduct") {

return(getTestItem($expanded));

}

$db = new DBConnection;

// Get basic product info (required for both minimised and expanded)

$products = $db->select("

SELECT " . ProductDetails . "

FROM Product

WHERE productID = ?

", array($productID));

if ($products == null) {

$db->close();

http\_response\_code(404); // Not found

return("Product not found");

}

$products = updateThumbnailPaths($products);

$product = $products[0];

if ($expanded) {

// Get images

$product->images = getImages($productID);

// Get reviews

$product->reviews = $db->select("

SELECT username, review

FROM Purchase

WHERE productID = ?

AND review IS NOT NULL

", array($productID));

}

$db->close();

return($product);

}

/\*\*

\* Gets an array of all categories in the database.

\*

\* @return {array(String)}

\*/

function getCategories() {

$query = "SELECT categoryName FROM Category";

$db = new DBConnection;

$categories = $db->select($query, null);

return($categories);

}

/\*\*

\* Gets up to $number products from a category starting from $offset.

\*

\* @param {String} $categoryName

\* @param {integer} $offset

\* @param {integer} $number

\* @return {array(product)}

\*/

function getProductsByCategory($categoryName, $offset, $number) {

$query = "

SELECT " . ProductDetails . "

FROM Product, ProductCategory, Category

WHERE Category.categoryName = ?

AND Category.categoryID = ProductCategory.categoryID

AND Product.productID = ProductCategory.productID

AND Product.listed = true

LIMIT ?, ?";

$db = new DBConnection;

$results = $db->select($query, array($categoryName, $offset, $number));

$db->close();

if ($results == null) {

http\_response\_code(404); // Not found

return("No products found");

}

$products = updateThumbnailPaths($results);

return $products;

}

/\*\*

\* Gets up to $number products matching $searchTerm starting from $offset.

\* Matches are by (in order) : product name, category, description

\*

\* @param {String} $searchTerm

\* @param {integer} $offset

\* @param {integer} $number

\* @return {array(product)}

\*/

function getProductsBySearch($searchTerm, $offset, $number) {

// Add % for matching using SQL LIKE

$term = "%$searchTerm%";

$query = "

SELECT " . ProductDetails . "

FROM Product

LEFT JOIN ProductCategory ON ProductCategory.productID = Product.productID

LEFT JOIN Category ON Category.categoryID = ProductCategory.categoryID

WHERE (productName LIKE ?

OR Category.categoryName LIKE ?

OR description LIKE ?)

AND Product.listed = true

LIMIT ?, ?";

$db = new DBConnection;

$results = $db->select($query, array($term, $term, $term, $offset, $number));

$db->close();

if ($results == null) {

http\_response\_code(404); // Not found

return("No products found");

}

$products = updateThumbnailPaths($results);

return $products;

}

/\*\*

\* Prepends the corisponding image directory to the thumbnail path for each

\* product or sets the thumbnail to the default image if none is set.

\*

\* @param {array(product)} $products

\* @return {array(product)}

\*/

function updateThumbnailPaths($products) {

foreach ($products as $product) {

$thumbnail = $product->thumbnail;

if ($thumbnail != null) {

$product->thumbnail = ImgRoot . $product->productID . '/' . $thumbnail;

} else {

$product->thumbnail = ImgRoot . "res/noImage.png";

}

}

return $products;

}

/\*\*

\* Returns the current stock for product with $productID.

\*

\* @param {int} $productID

\* @return {int}

\*/

function getStock($productID) {

$query = "SELECT stock FROM Product WHERE productID = ?";

$db = new DBConnection;

$results = $db->select($query, array($productID));

$db->close();

$result = $results[0]->stock;

return $result;

}

/\*\*

\* Returns an array of all categories product with $productID is in.

\*

\* @param {int} $productID

\* @return {array(String)}

\*/

function getProductCategories($productID) {

$query = "

SELECT categoryName FROM Category, ProductCategory

WHERE productID = ?

AND Category.categoryID = ProductCategory.categoryID";

$db = new DBConnection;

$results = $db->select($query, array($productID));

$db->close();

return $results;

}

/\*\*

\* Returns the ID of category with $categoryName.

\*

\* @param {String} $categoryName

\* @return {int}

\*/

function getCategoryID($categoryName) {

$db = new DBConnection;

$query = "SELECT categoryID FROM Category WHERE categoryName = ?";

$results = $db->select($query, array($categoryName));

if (sizeOf($results) === 0) {

http\_response\_code(404); // Not Found

return $results;

}

$result = $results[0]->categoryID;

$db->close();

return $result;

}

/\*\*

\* Returns an array of the paths to all images of product with $productID.

\*

\* @param {int} $productID

\* @return {array(String)}

\*/

function getImages($productID) {

return glob(ImgRoot . "$productID/\*.\*");

}

/\*\*

\* Returns an array of objects with the properties productName and stock,

\* describing the current stock level for all products.

\*

\* @return {array(Object)}

\*/

function getAllStock() {

$query = "

SELECT productName, stock

FROM Product

WHERE listed = true

ORDER BY stock";

$db = new DBConnection;

$results = $db->select($query, null);

$db->close();

return $results;

}

/\*\*

\* Returns an array of objects with the properties purchaseID, productName,

\* quantity and address, describing all currently pending purchases.

\*

\* @return {array(Object)}

\*/

function getPendingPurchases() {

$query = "

SELECT purchaseID, productName, quantity, address

FROM Product, Purchase, Customer

WHERE processed = false

AND Product.productID = Purchase.productID

AND Customer.username = Purchase.username";

$db = new DBConnection;

$results = $db->select($query, null);

$db->close();

return $results;

}

/\*\*

\* Returns the total quantity sold of product with $productID.

\*

\* @param {int} $productID

\* @return {int}

\*/

function quantitySold($productID) {

$query = "

SELECT SUM(quantity) AS value

FROM Purchase

WHERE productID = ?";

$db = new DBConnection;

$results = $db->select($query, array($productID));

$db->close();

// Return 0 if no results

if (!isset($results[0]->value)) {

return 0;

} else {

return $results[0]->value;

}

}

**POSTResponders.php**

<?php

/\*\*

\* Creates a new category with the name provided.

\*

\* @param {String} $data

\* @return {boolean}

\*/

function addCategory($data) {

// Check if already exists

if (sizeOf(getCategoryID($data)) !== 0) {

http\_response\_code(500); // Internal Server Error

return "Category already exists";

} else {

http\_response\_code(200); // OK

}

$query = "INSERT INTO Category (categoryName) VALUES (?)";

$db = new DBConnection;

$result = $db->run($query, array($data));

$db->close();

return $result;

}

/\*\*

\* Creates a new product listing.

\* The $data parameter is optional, if not set, default properties will be used.

\* Otherwise, $data should be an object including the properties:

\* {listed, productName, price, deliveryType, stock, description, thumbnail}

\*

\* @param {Object} $data

\* @return {String}

\*/

function addProduct($data) {

if (!isSet($data)) {

$data = defaultProduct();

}

$details = [];

foreach (get\_object\_vars($data) as $value) {

$details[] = $value;

}

$query = "INSERT INTO

Product (listed, productName, price, deliveryType, stock, description, thumbnail)

VALUES (?, ?, ?, ?, ?, ?, ?)";

$db = new DBConnection;

$result = $db->run($query, $details);

if ($result !== "Success") {

http\_response\_code(500); // Internal Server Error

$db->close();

return $result;

}

// Get productID of new product

$result = $db->lastID();

$db->close();

return $result;

}

/\*\*

\* Creates a product object with default values.

\*

\* @returns {Object}

\*/

function defaultProduct() {

$product = new stdClass();

$product->listed = false;

$product->productName = "Enter product name";

$product->price = 0;

$product->deliveryType = "Choose pricing";

$product->stock = 0;

$product->description = "Enter a description for this product.";

$product->thumbnail = null;

return $product;

}

/\*\*

\* Records a new purchase.

\* $data should be an object including the properties {productID, username, quantity}

\*

\* @param {Object} $data

\* @return {String}

\*/

function addPurchase($data) {

// TODO needs to validate

$productID = $data->productID;

$quantity = $data->quantity;

// Ensure quantity can be supplied and update database

$result = addStock($productID, (0 - $quantity));

if ($result === "Success") {

// Create purchase

$query = "

INSERT INTO Purchase (productID, username, quantity, processed)

VALUES (?, ?, ?, false)";

$db = new DBConnection;

$result = $db->run($query, array($data->productID, $data->username, $data->quantity));

$db->close();

}

if ($result !== "Success") {

http\_response\_code(500); // Internal Server Error

}

return $result;

}

/\*\*

\* Adds product with $productID to category of name $categoryName

\*

\* @param {int} $productID

\* @param {String} $categoryName

\* @return {String}

\*/

function addToCategory($productID, $categoryName) {

$categoryID = getCategoryID($categoryName);

if (!is\_numeric($categoryID)) {

return $categoryID;

}

$db = new DBConnection;

$query = "INSERT INTO ProductCategory VALUES (?, ?)";

$result = $db->run($query, array($productID, $categoryID));

$db->close();

if ($result !== "Success") {

http\_response\_code(500); // Internal Server Error

}

return $result;

}

/\*\*

\* Creates file with $image data in the folder for product with $productID.

\* $image should be an array of the form [filename, base64 encoded image data]

\* Images with duplicate names will be renamed automatically.

\* (eg. image.png, image(0).png, image(1).png, ....)

\*

\* @param {int} $productID

\* @param {array} $image

\* @return {String}

\*/

function addImage($productID, $image) {

$fileName = $image[0];

$imageFile = base64\_decode($image[1]);

$imageDir = ImgRoot . "$productID/";

$filePath = $imageDir . $fileName;

$fileType = pathinfo($filePath, PATHINFO\_EXTENSION);

// Restrict file format

if (!in\_array($fileType, ["jpg", "jpeg", "png", "gif"])) {

http\_response\_code(415);

return "Only JPG, JPEG, PNG & GIF files allowed, recieved $fileType";

}

// Create directory if not exists

if (!file\_exists($imageDir)) {

mkdir($imageDir, 0777, true);

}

// Check if file already exists

if (file\_exists($filePath)) {

$fileNoExtension = rtrim($filePath, ".$fileType");

$newName = $filePath;

$i = 0; // Find index to append to filename

while (file\_exists($newName)) {

$newName = "$fileNoExtension($i).$fileType";

$i++;

}

$filePath = $newName;

}

file\_put\_contents($filePath, $imageFile);

return "Image uploaded successfully";

}

/\*\*

\* Adds a review containing $data for product with $productID.

\*

\* @param {int} $productID

\* @param {Object} $data

\* @return {String}

\*/

function addReview($productID, $data) {

//TODO

http\_response\_code(501); // Not Implemented

return "'add review' not implemented yet";

}

**PUTResponders.php**

<?php

/\*\*

\* Updates data for product with $productID to match changes specified by $data.

\* $data should be an object with key-value pairs for each field to be modified.

\*

\* @param {int} $productID

\* @param {Object} $data

\* @return {String}

\*/

function editProduct($productID, $data) {

$keys = [];

$values = [];

// Get values from request

$properties = get\_object\_vars($data);

foreach ($properties as $key => $value) {

// If unlisting a product with no purchases, delete instead

if ($key === 'listed' && $value === false) {

if (quantitySold($productID) === 0) {

return deleteproduct($productID);

}

}

// catch negative price/stock

if (!(($key === 'price' || $key === 'stock') && $value < 0)) {

$keys[] = $key;

$values[] = $value;

}

}

// Combine query variables

$values[] = $productID;

$names = join(" = ?, ", $keys) . " = ?";

$query = "UPDATE Product SET $names WHERE productID = ?";

// Run query

$db = new DBConnection;

$result = $db->run($query, $values);

$db->close();

if ($result !== "Success") {

http\_response\_code(500); // Internal Server Error

}

return $result;

}

/\*\*

\* Adds an amount to the stock for a product, negative amounts are permitted.

\* If stock would become negative or product cannot be found, a failure messege

\* is returned and changes are not made.

\*

\* @param integer $productID

\* @param integer $amount

\* @return string

\*/

function addStock($productID, $amount) {

$currentStock = getStock($productID);

if (!is\_numeric($currentStock)) {

http\_response\_code(404); // Not Found

return "Failed: Cannot find product";

}

$quantity = $currentStock + $amount;

if ($quantity < 0) {

return "Failed: Insufficient stock";

}

$db = new DBConnection;

$result = $db->run("

UPDATE Product SET stock = ?

WHERE productID = ?", array($quantity, $productID));

$db->close();

return $result;

}

/\*\*

\* Updates data for purchase with $purchaseID to match changes specified

\* by $data. $data should be an object with key-value pairs for each field

\* to be modified.

\*

\* @param {int} $purchaseID

\* @param {Object} $data

\* @return {String}

\*/

function editPurchase($purchaseID, $data) {

$keys = [];

$values = [];

// Get values from request

$properties = get\_object\_vars($data);

foreach ($properties as $key => $value) {

// catch negative quantity

if (!($key === 'quantity' && $value < 0)) {

$keys[] = $key;

$values[] = $value;

}

}

// Combine query variables

$values[] = $purchaseID;

$names = join(" = ?, ", $keys) . " = ?";

$query = "UPDATE Purchase SET $names WHERE purchaseID = ?";

// Run query

$db = new DBConnection;

$result = $db->run($query, $values);

$db->close();

if ($result !== "Success") {

http\_response\_code(500); // Internal Server Error

}

return $result;

}

function editReview($productID, $username) {

//TODO

http\_response\_code(501); // Not Implemented

return "'edit review' not implemented yet";

}

**index.php (API)**

<!doctype html>

<html>

<head>

<title>WebShop - API</title>

<meta charset="UTF-8">

<link rel="stylesheet" href="../Shared/CSS/Global.css">

<link rel="stylesheet" href="../Shared/CSS/Infrastructure.css">

</head>

<body>

<p class='Notification'>

Below is the full RESTful server API.<br>

Cells highlighted in yellow are incomplete and will return 501 not

implemented, these are intended as points for further expansion.

</p>

<table>

<tr>

<th>Path</th>

<th>GET</th>

<th>POST</th>

<th>PUT</th>

<th>DELETE</th>

</tr>

<tr>

<td>/categories</td>

<td>List all categories</td>

<td>Add new category</td>

<td></td>

<td></td>

</tr>

<tr>

<td>/categories/{name}</td>

<td></td>

<td></td>

<td></td>

<td>Removes all items from and deletes category of {name}</td>

</tr>

<tr>

<td>/products/category/{term}/{offset}/{number}</td>

<td>Returns next {number} products in {category} after {offset}</td>

<td></td>

<td></td>

<td></td>

</tr>

<tr>

<td>/products /search/{term}/{offset}/{number}</td>

<td>Returns next {number} products matching {term} after {offset}</td>

<td></td>

<td></td>

<td></td>

</tr>

<tr>

<td>/products</td>

<td></td>

<td>Creates a new product and returns the productID</td>

<td></td>

<td></td>

</tr>

<tr>

<td>/products/{id} </td>

<td>Gets full data for product with {id}</td>

<td></td>

<td>Updates data for product with {id}</td> <!--should archive previous version-->

<td>Deletes product with {id} and all associated references</td>

</tr>

<tr>

<td>/products/{id}/stock</td>

<td>Gets the current stock of product with {id}</td>

<td></td>

<td>Changes the current stock of product with {id} by the given amount (can be positive or negative)</td>

<td></td>

</tr>

<tr>

<td>/products/{id}/images</td>

<td>Gets all images of product with {id}</td>

<td>Uploads a new image of product with {id}</td>

<td></td>

<td></td>

</tr>

<tr>

<td>/products/{id}/images/{fileName}</td>

<td></td>

<td></td>

<td></td>

<td>Deletes image with {fileName} of product with {id}</td>

</tr>

<tr>

<td>/products/{id}/reviews</td>

<td></td>

<td class='ToDo'>Adds a review for product with {id}</td>

<td></td>

<td></td>

</tr>

<tr>

<td>/products/{id}/reviews/{username}</td>

<td></td>

<td></td>

<td class='ToDo'>Edits review for product with {id} by user with {username}</td>

<td class='ToDo'>Deletes review for product with {id} by user with {username}</td>

</tr>

<tr>

<td>/products/{id}/categories</td>

<td>Gets all categories product with {id} is a member of</td>

<td></td>

<td></td>

<td></td>

</tr>

<tr>

<td>/products/{id}/categories/{name}</td>

<td></td>

<td>Adds product with {id} to category of {name}</td>

<td></td>

<td>Removes product with {id} from category of {name}</td>

</tr>

<tr>

<td>/purchases</td>

<td></td>

<td>Adds a new purchase and updates stock for product</td>

<td></td>

<td></td>

</tr>

<tr>

<td>/purchases/{id}</td>

<td></td>

<td></td>

<td>Edits properties of purchase with {id}</td>

<td></td>

</tr>

<tr>

<td>/purchases/pending</td>

<td>Gets a list of the product names, quantity and delivery address for all pending purchases</td>

<td></td>

<td></td>

<td></td>

</tr>

<tr>

<td>/stock</td>

<td>Gets a list of the names and stock of all listed products</td>

<td></td>

<td></td>

<td></td>

</tr>

</table>

</body>

</html>

**index.php (CMS)**

<!doctype html>

<html>

<head>

<title>WebShop - Content management system</title>

<meta charset="UTF-8">

<link rel="stylesheet" href="../Shared/CSS/Global.css">

<link rel="stylesheet" href="../Shared/CSS/Navigation.css">

<link rel="stylesheet" href="../Shared/CSS/Infrastructure.css">

<script src='../Shared/lib/lib.js'></script>

<script src='Admin.js'></script>

</head>

<body>

<?php include '../Shared/Include/Navigation.html'; ?>

<section id="SingleSection">

<?php include '../Shared/Include/Display.html'; ?>

</section>

</body>

</html>

**index.php (Customer)**

<!doctype html>

<html>

<head>

<title>WebShop - Content management system</title>

<meta charset="UTF-8">

<link rel="stylesheet" href="../Shared/CSS/Global.css">

<link rel="stylesheet" href="../Shared/CSS/ItemBrowser.css">

<link rel="stylesheet" href="../Shared/CSS/Navigation.css">

<link rel="stylesheet" href="CMSControls.css">

<script src='../Shared/lib/lib.js'></script>

<script src='../Shared/lib/ItemBrowser.js'></script>

</head>

<body>

<?php include '../Shared/Include/Navigation.html'; ?>

<a class="anchor" id="Top"></a>

<section id="RightSection">

<?php include 'CMSControls.php'; ?>

</section>

<section id="LeftSection">

<?php include '../Shared/Include/Display.html'; ?>

</section>

</body>

</html>

**CMSControls.php**

<script src='CMSControls.js'></script>

<nav class="Controls">

<h1>Controls</h1>

<section>

<input type="submit" value="Create product" onclick="createProduct()">

<input type="submit" value="Manage categories" onclick="manageAllCategories()">

</section>

</nav>

**BrowserBasket.php**

<script src='BrowserBasket.js'></script>

<nav class="Basket">

<h1>Your Basket</h1>

<section>

<p id="totalPrice">

Total price Â£0.00

</p>

<input id="checkoutButton" type="submit" value="Place order" onclick="placeOrder()">

</section>

<section id="BasketItems"></section>

</nav>

**index.php (root)**

<!doctype html>

<html>

<head>

<title>WebShop</title>

<meta charset="UTF-8">

<link rel="stylesheet" href="../Shared/CSS/Global.css">

<link rel="stylesheet" href="../Shared/CSS/ItemBrowser.css">

<link rel="stylesheet" href="../Shared/CSS/Navigation.css">

<link rel="stylesheet" href="BrowserBasket.css">

<script src='../Shared/lib/lib.js'></script>

<script src='../Shared/lib/ItemBrowser.js'></script>

</head>

<body>

<?php include '../Shared/Include/Navigation.html'; ?>

<a class="anchor" id="Top"></a>

<section id="RightSection">

<?php include 'BrowserBasket.php'; ?>

</section>

<section id="LeftSection">

<?php include '../Shared/Include/Display.html'; ?>

</section>

</body>

</html>

**index.php (Admin)**

<!doctype html>

<html>

<head>

<title>WebShop - Admin</title>

<meta charset="UTF-8">

<link rel="stylesheet" href="../Shared/CSS/Global.css">

<link rel="stylesheet" href="../Shared/CSS/Navigation.css">

<link rel="stylesheet" href="../Shared/CSS/Infrastructure.css">

<script src='../Shared/lib/lib.js'></script>

<script src='Admin.js'></script>

</head>

<body>

<?php include '../Shared/Include/Navigation.html'; ?>

<section id="SingleSection">

<?php include '../Shared/Include/Display.html'; ?>

</section>

</body>

</html>

**Display.html**

<header id="header"></header>

<section id="Display">

<article id="Notification"><h1>You must have Javascript enabled to use this site</h1></article>

</section>

<footer id="footer"></footer>

**Navigation.html**

<script src='../Shared/lib/Navigation.js'></script>

<nav class="navBar">

<h1>WebShop</h1>

<ol id="tabs">

<li>Loading...</li>

</ol>

</nav>

**Global.css**

body{

background-color: #CCE6FF;

font-family: "Oxygen","Sans-serif";

margin: 0;

padding: 0;

}

.anchor{

display: block;

position: relative;

top: -5em;

visibility: hidden;

}

ul{

padding: 0;

margin: 0;

}

#LeftSection{

margin: 16em 0;

min-width:31em;

max-width:60em;

margin-top:5em;

margin-left: auto;

margin-right: auto;

position:relative;

right:7em;

}

#RightSection{

position:fixed;

right:1em;

top:5em;

width: 15em;

}

#SingleSection{

min-width:31em;

max-width:60em;

margin-top:5em;

margin-left: auto;

margin-right: auto;

position:relative;

}

@media all and (max-width: 80em) {

#LeftSection{

right:0;

}

#RightSection{

position:relative;

margin-left: auto;

margin-right: auto;

margin-bottom: 6em;

}

}

footer{

margin: 0.8em;

font-size: 1.3em;

background-color: silver;

text-align: center;

position: relative;

border-radius: 1em;

}

.Notification{

margin:1em;

text-align: center;

background-color: #F0F0F0;

font-size: 1.5em;

border-radius: 1em;

padding: 1em;

font-family:Verdana,"Sans-serif";

}

textArea{

font: inherit; /\* For IE \*/

}

p{

text-align: justify;

font-family:Verdana,"Sans-serif";

}

.InvalidInput{

border-color: red;

}

.hidden{

display: none;

}

.TwoButton{

width:49.5% !important;

margin:0%;

height:2em;

font-size: 1em;

text-align: center;

}

.ThreeButton{

width:32% !important;

height:2.5em;

font-size: 1em;

text-align: center;

}

.ThreeButton + input[type=submit]{

margin-left: 1.5%;

}

.ThreeButton + input{

margin-left: 1.5%;

}

**Infrastructure.css**

#Interlinks li{

text-align: center;

background-color: #F0F0F0;

font-size: 3em;

border-radius: 1em 1em 1em 1em;

padding: 1em;

margin: 1em;

list-style-type: none;

}

table {

font-family:Verdana,"Sans-serif";

margin: 2%;

width: 96%;

border-width: 1px;

border-collapse: collapse;

background-color: white;

}

th {

border-width: 1px;

padding: 8px;

border-style: solid;

background-color: #dedede; /\* Light grey \*/

}

td {

border-width: 1px;

padding: 8px;

border-style: solid;

}

.Striped tr:nth-child(odd) {

background: #F0F0F0;

}

.ToDo{

background-color: #FAF882; /\* Light orange \*/

}

.OutOfStock{

background-color: #FF9C9C; /\*Light red\*/

}

.LowStock{

background-color: #FFDC9C; /\* Light orange \*/

}

.InStock{

background-color: #BAFFC2; /\* Light green \*/

}

.ObjectCell{

padding:0;

}

.ObjectCell input[type=submit]{

width:100%;

height:100%;

font-size: 1em;

}

**ItemBrowser.css**

.ProductThumbnail{

height:11em;

width:11em;

margin:0 0.5em 0.5em 0;

float:left;

}

.ProductDetailsList{

float:right;

}

.ProductDetailsList li{

font-size: 1.3em;

background-color: #B9B9B9;

margin: 0.6em;

padding: 0.2em 0.5em 0.25em 0.5em;

border-radius: 0.3em;

list-style-type: none;

}

.ListedProduct{

font-size: 1em;

background-color: #F0F0F0;

padding:0.5em;

margin-top: 1.2em;

position: relative;

z-index: 1;

box-shadow: 0 0 0.5em grey;

overflow: hidden;

}

.ListedProduct input[type=submit]{

font-size: 1.2em;

width: 100%;

}

.ListedProduct:hover{

box-shadow: 0 0 1.5em #CC9900;

}

.ListedProduct h1{

font-size: 1.6em;

color: white;

background-color: #B9B9B9;

margin: 0;

padding: 0;

padding-left:0.5em;

border-radius: 0.3em;

}

.ListedProduct p{

margin-top:1em;

}

.Minimised{

max-height: 11em;

-webkit-transition: max-height 1s;

transition: max-height 1s;

}

.Minimised p{

min-height: 8em;

max-height: 7.3em;

overflow: hidden;

}

.Expanded{

max-height:100em;

-webkit-transition: max-height 2s;

transition: max-height 2s;

}

.Expanded p{

min-height: 8em;

}

@media all and (max-width: 850px) {

.ListedProduct p{

min-height:3.5em;

margin-top:10em;

}

.Minimised{

max-height: 15.5em;

}

.Minimised p{

max-height:3.5em;

}

}

.Reviews{

border-radius: 0.5em;

margin: 1em 0 1em 0;

padding-bottom: 1px;

background-color: #D4D4D4;

}

.Reviews article{

background-color: #F0F0F0;

border-radius: 0.5em;

margin: 0.5em;

}

.Reviews h1{

border-radius: 0.3em 0.3em 0 0;

}

.Reviews h2{

padding-left:0.5em;

margin-bottom:0;

border-radius: 0.3em 0.3em 0 0;

background-color: #B9B9B9;

}

.Reviews p {

min-height:0;

margin-top:0;

padding: 0.5em;

}

.Images{

width:100%;

margin:0.5em 0 0.5em 0;

overflow-x: scroll;

overflow-y:hidden;

white-space:nowrap;

}

.Images img{

height: 12em;

margin:0.3em;

}

footer input[type=submit]{

margin-left:2em;

font-size: 0.8em;

}

**Navigation.css**

.navBar{

min-width:10em;

font-size: 1.3em;

top:0;

margin: 0;

padding: 0;

position: fixed;

width:100%;

z-index: 2;

}

.navBar h1{

text-align: center;

margin: 0;

padding: 0;

padding-bottom:0;

width:100%;

background-color: grey;

color:white;

font-size: 1.6em;

}

.navBar ol{

background-color: silver;

margin: 0;

padding: 0;

text-align: center;

white-space:nowrap;

overflow-x:hidden;

overflow-y:hidden;

}

.navBar li{

display: inline;

list-style-type: none;

padding-right: 2%;

padding-left: 2%;

overflow-y: scroll;

}

.navBar li:hover{

text-shadow: 0 0 1em white, 0 0 1em white, 0 0 1em #CC9900;

}

.navBar input{

margin: 0;

margin-bottom:0.3em;

font-size:0.7em;

}

#currentTab{

background-color: #CCE6FF;

padding-bottom:0.3em;

border-top-left-radius: 0.3em;

border-top-right-radius: 0.3em;

}

**CMSControls.css**

.Controls{

background-color: silver;

text-align: center;

border-radius: 0.5em;

}

.Controls h1{

border-radius: 0.3em 0.3em 0 0;

background-color: grey;

font-size: 1.8em;

margin:0 0 1% 0;

padding: 0.1em;

text-align: center;

color: white;

}

.Controls input{

font-size: 1.2em;

width: 96%;

margin:1% 2% 2% 1%;

}

.Controls input:hover{

box-shadow: 0 0 0.5em #CC9900;

}

.EditField{

border-style:none;

color:purple;

background-color: #B9B9B9;

font-size: 0.9em;

width:9em;

}

.EditingProduct h1 input{

width: calc(100% - 8em);

}

.EditingProduct textarea{

background-color: #F0F0F0;

color:purple;

font-size:1.2em;

min-height:6.2em;

width:47%;

}

.EditingProduct section{

margin-top:2em;

background-color: #D4D4D4;

}

.EditingProduct input[type=submit]:active{

background-color: #78A0FF;

}

.EditingProduct:hover{

box-shadow: 0 0 0.5em grey;

}

.EditingProduct section h1{

border-radius: 0;

}

.Manager ul li{

font-size:1.3em;

display:inline;

margin:1em;

}

.Manager ul{

margin:0.5em;

border-radius: 0.3em 0.3em 0 0;

}

.CategoryManager {

width:100%;

overflow: auto;

text-align: center;

}

.CategoryManager h2{

margin:0;

}

.CategoryManager section{

margin-top: 1em;

}

.left select{

width:100%;

padding:5px;

}

.right select{

width:100%;

padding:5px;

}

.left{

float: left;

width:40%;

margin:1%;

}

.right{

float: right;

width:40%;

margin:1%;

}

.CategoryManager > .mid {

float: left;

width:16%;

margin-top: 2.6em;

}

.mid input[type=submit]{

width:100%;

margin:1%;

font-size: 1.6em;

}

.ManagerPanel{

text-align: center;

padding-bottom: 0.5em;

}

.ManagerPanel p{

text-align: center;

}

.ManagerPanel input[type=submit]{

width:90%;

}

.ManagerPanel input{

font-size:1em;

text-align: center;

margin: 0.5em;

width:90%;

}

.ManagerPanel input[type=file]{

background-color: white;

width: 90%;

}

.ImageManager input[type=submit]{

width: 90%;

height:2em;

}

.ImageManager .left{

width:50%;

margin-left:0;

}

.ImageManager .right{

margin:0;

}

.ImageManager .Images img:hover{

box-shadow: 0 0 1.5em grey;

}

.ImageManager .controls{

height:16em;

margin:0;

}

#imageUploadPreview img{

height:12em;

}

#imageUploadPreview{

height:100%;

border-style: dashed;

border-width: 3px;

}

#singleImageControls input[type=submit]{

height:80%;

width:40%;

margin:1em;

font-size: 1.5em;

white-space:pre;

}

#selectedImage{

box-shadow: 0 0 1.5em #CC9900;

}

#catList{

margin:2em 0 1.5em 0;

}

#catList li{

background-color: #D4D4D4;

padding:0.3em;

border-radius: 0.3em;

margin:0.5em;

}

**BrowserBasket.css**

.Basket {

background-color: grey;

text-align: center;

border-radius: 0.5em;

padding-bottom: 0.1em;

}

.Basket h1{

font-size: 1.8em;

margin:0;

margin-top: 0.1em;

padding: 0.1em;

text-align: center;

color: white;

}

.BasketItem h1{

font-size: 1em;

background-color: silver;

margin: 0;

margin-top: 0.1em;

padding: 0.1em;

text-align: center;

border-top-left-radius: 0.3em;

border-top-right-radius: 0.3em;

}

.Basket p{

background-color: silver;

font-size:1.3em;

margin:0.4em;

margin-top:0;

}

.BasketDetails h1{

font-size: 1em;

}

.BasketDetails li{

list-style-type: none;

}

.Basket input{

font-size: 0.8em;

width:100%;

}

.BasketItem{

font-size: 1.3em;

background-color: #F0F0F0;

padding:0;

margin:0.4em;

border-radius: 0.3em;

}

.BasketItem:hover{

box-shadow: 0 0 0.5em #CC9900;

}

.BasketItem h1{

color: black;

}

#totalPrice{

margin-bottom:0;

text-align: center;

}

#checkoutButton{

margin-top:0;

margin-bottom:0.5em;

width:94%;

font-size: 1.3em;

background-color: #EBE000;

}

**CreateTables.sql**

CREATE TABLE Customer (

username VARCHAR(30) PRIMARY KEY ,

address VARCHAR(200) NOT NULL

);

CREATE TABLE Category (

categoryID INTEGER AUTO\_INCREMENT PRIMARY KEY,

categoryName VARCHAR(30) NOT NULL

);

CREATE TABLE Product (

productID INTEGER AUTO\_INCREMENT PRIMARY KEY,

listed BOOLEAN NOT NULL,

productName VARCHAR(30) NOT NULL,

price DECIMAL(12,2) NOT NULL,

deliveryType VARCHAR(30) NOT NULL,

stock INTEGER NOT NULL,

description TEXT NOT NULL,

thumbnail VARCHAR(30)

);

CREATE TABLE ProductCategory (

productID INTEGER,

categoryID INTEGER,

INDEX (categoryID),

INDEX (productID),

CONSTRAINT PK\_ProductCategory PRIMARY KEY(productID, categoryID),

CONSTRAINT FK\_categoryID FOREIGN KEY(categoryID) REFERENCES Category(categoryID),

CONSTRAINT FK\_productID FOREIGN KEY(productID) REFERENCES Product(productID)

);

CREATE TABLE Purchase (

purchaseID INTEGER AUTO\_INCREMENT PRIMARY KEY,

productID INTEGER NOT NULL,

username VARCHAR(30) NOT NULL,

quantity INTEGER NOT NULL,

processed BOOLEAN NOT NULL,

rating INTEGER,

review TEXT,

INDEX (username),

INDEX (productID),

CONSTRAINT FK\_purchaseUsername FOREIGN KEY(username) REFERENCES Customer(username),

CONSTRAINT FK\_purchaseproductID FOREIGN KEY(productID) REFERENCES Product(productID)

);

**TestData.sql**

-- username, address

INSERT INTO Customer VALUES ('Test User 1', 'Test Address 1');

INSERT INTO Customer VALUES ('Test User 2', 'Test Address 2');

INSERT INTO Customer VALUES ('Test User 3', 'Test Address 3');

INSERT INTO Customer VALUES ('Test User 4', 'Test Address 4');

-- categoryName

INSERT INTO Category (categoryName) VALUES ('Test category 1');

INSERT INTO Category (categoryName) VALUES ('Test category 2');

INSERT INTO Category (categoryName) VALUES ('Test category 3');

INSERT INTO Category (categoryName) VALUES ('Test category 4');

-- listed, productName, price, deliveryType, stock, description, thumbnail

INSERT INTO Product (listed, productName, price, deliveryType, stock, description, thumbnail) VALUES (true, 'Test product 1', 20.30, 'Free global', 10, 'This is the first test product, it costs Â£20.30 and has free global delivery. There are initially 10 in stock. It belongs in Test category 1.', 'Teapot1.jpg');

INSERT INTO Product (listed, productName, price, deliveryType, stock, description, thumbnail) VALUES (true, 'Test product 2', 12.99, 'Free UK', 32, 'This is the second test product, it costs Â£12.99 and has free UK delivery. There are initially 32 in stock. It belongs in Test categories 1 and 3.', 'Teapot2.jpg');

INSERT INTO Product (listed, productName, price, deliveryType, stock, description, thumbnail) VALUES (true, 'Test product 3', 1.20, 'Location dependent', 1256,'This is the third test product, it costs Â£1.20 and has location dependent delivery. There are initially 1256 in stock. It belongs in Test category 1.', 'Teapot3.jpg');

INSERT INTO Product (listed, productName, price, deliveryType, stock, description, thumbnail) VALUES (true, 'Test product 4', 0.01, 'Location dependent', 124, 'This is the fourth test product, it costs Â£0.01 and has location dependent delivery. There are initially 124 in stock. It belongs in Test category 2.', 'Teapot4.jpg');

INSERT INTO Product (listed, productName, price, deliveryType, stock, description, thumbnail) VALUES (true, 'Test product 5', 6.72, 'Location dependent', 56, 'This is the fith test product, it costs Â£6.72 and has location dependent delivery. There are initially 56 in stock. It belongs in Test category 2.', 'Teapot4.jpg');

INSERT INTO Product (listed, productName, price, deliveryType, stock, description, thumbnail) VALUES (true, 'Test product 6', 120.78, 'Free UK', 12, 'This is the sixth test product, it costs Â£120.78 and has free UK delivery. There are initially 12 in stock. It belongs in no categories', 'Teapot3.jpg');

INSERT INTO Product (listed, productName, price, deliveryType, stock, description, thumbnail) VALUES (true, 'Test product 7', 1265.02, 'Free global', 2, 'This is the seventh test product, it costs Â£1265.02 and has free global delivery. There are initially 2 in stock. It belongs in Test category 3. It has no thumbnail', null);

-- productID, categoryID

INSERT INTO ProductCategory VALUES (1, 1);

INSERT INTO ProductCategory VALUES (2, 1);

INSERT INTO ProductCategory VALUES (2, 3);

INSERT INTO ProductCategory VALUES (3, 2);

INSERT INTO ProductCategory VALUES (4, 2);

INSERT INTO ProductCategory VALUES (5, 2);

INSERT INTO ProductCategory VALUES (7, 3);

-- productID, username, quantity, processed, rating, review

INSERT INTO Purchase (productID, username, quantity, processed, rating, review) VALUES (1, 'Test user 1', 2, true, 5, 'Bought 2, perfect 5/5, will buy more' );

INSERT INTO Purchase (productID, username, quantity, processed, rating, review) VALUES (1, 'Test user 2', 1, true, 3, 'Bought 1, was alright 3/5' );

INSERT INTO Purchase (productID, username, quantity, processed, rating, review) VALUES (2, 'Test user 2', 3, true, 4, 'Pretty good, bought 3, 4/5' );

INSERT INTO Purchase (productID, username, quantity, processed, rating, review) VALUES (3, 'Test user 3', 5, true, 1, 'Terrible, should not have bought 5, 1/5' );

INSERT INTO Purchase (productID, username, quantity, processed, rating, review) VALUES (1, 'Test user 4', 2, true, 3, null);

INSERT INTO Purchase (productID, username, quantity, processed, rating, review) VALUES (4, 'Test user 4', 7, true, null, null);

INSERT INTO Purchase (productID, username, quantity, processed, rating, review) VALUES (1, 'Test user 1', 3, false, null, null);

INSERT INTO Purchase (productID, username, quantity, processed, rating, review) VALUES (5, 'Test user 2', 2, false, null, null);

INSERT INTO Purchase (productID, username, quantity, processed, rating, review) VALUES (7, 'Test user 3', 1, false, null, null);

COMMIT;