

# SYSTEMS PROGRAMMING AND SCRIPTING

FLORIAN BERGMANN  
PERSON ID: H00020398

Assessment Three: MACS bookstore

## CONTENTS

---

<b>I DEVELOPMENT OF A BASIC WEB BROWSER</b>	<b>1</b>
<b>1 INTRODUCTION</b>	<b>2</b>
1.1 Document overview . . . . .	2
1.2 Remit . . . . .	2
<b>2 REQUIREMENT'S CHECKLIST</b>	<b>4</b>
<b>3 DESIGN CONSIDERATIONS</b>	<b>5</b>
3.1 Architectural overview . . . . .	5
<b>4 USER GUIDE</b>	<b>7</b>
4.1 Login (and registering) . . . . .	7
4.2 Searching for products . . . . .	9
4.3 Adding items to the shopping cart . . . . .	9
4.4 Editing items in the shopping cart . . . . .	11
<b>5 DEVELOPER GUIDE</b>	<b>12</b>
5.1 Implementation of the requirements . . . . .	12
5.1.1 Homepage-Requirements . . . . .	12
5.1.2 Restrict access to logged-in users . . . . .	12
5.1.3 Provide a link to the search request . . . . .	13
5.1.4 Show user details after logging in . . . . .	13
5.2 Search requirements . . . . .	13
5.2.1 Search functionality . . . . .	13
5.2.2 Links . . . . .	15
5.3 Product requirements . . . . .	15
5.3.1 Display product details . . . . .	15
5.3.2 Provide links . . . . .	15
5.4 Shopping cart requirements . . . . .	15
5.4.1 Updating of already present items . . . . .	15
5.4.2 Adding items . . . . .	16
5.4.3 Provide links . . . . .	16
5.5 Details of the implementation . . . . .	16
5.5.1 Modularity . . . . .	16
5.5.2 Form functions . . . . .	16
<b>6 TESTING</b>	<b>18</b>
<b>7 CONCLUSIONS</b>	<b>20</b>
<b>II APPENDIX</b>	<b>21</b>
<b>A APPENDIX: SOURCE CODE</b>	<b>22</b>
A.1 Websites . . . . .	22

A.2 Includes . . . . .	28
A.3 Tests . . . . .	41
BIBLIOGRAPHY	42

## LIST OF FIGURES

---

Figure 1	Architectural overview . . . . .	6
Figure 2	Login panel . . . . .	7
Figure 3	The header links . . . . .	7
Figure 4	The register form displays errors. . . . .	8
Figure 5	Information to login first . . . . .	9
Figure 6	Searching for products . . . . .	9
Figure 7	Adding a product to the shopping cart. . . . .	10
Figure 8	Product successfully added. . . . .	10
Figure 9	User is informed about incorrect input. . . . .	10
Figure 10	The shopping cart. . . . .	11
Figure 11	Illegal edit is prevented. . . . .	11

## LIST OF TABLES

---

Table 1	Performed tests. . . . .	19
---------	--------------------------	----

## LISTINGS

---

Listing 1	Redirecting the login_form.inc.php to the index page . . . . .	12
Listing 2	Redirecting invalid users . . . . .	12
Listing 3	The search function . . . . .	13
Listing 4	Shopping cart form for editing an existing item. . . . .	15
Listing 5	index.php . . . . .	22
Listing 6	customer_details.php . . . . .	22
Listing 7	logout.php . . . . .	23
Listing 8	product_details.php . . . . .	24
Listing 9	shopping_cart.php . . . . .	26

Listing 10	config.inc.php . . . . .	28
Listing 11	database.inc.php . . . . .	29
Listing 12	footer.php . . . . .	34
Listing 13	form_functions.inc.php . . . . .	34
Listing 14	header.php . . . . .	35
Listing 15	item.php . . . . .	36
Listing 16	login.inc.php . . . . .	36
Listing 17	login_form.inc.php . . . . .	37
Listing 18	shopping_cart.inc.php . . . . .	38
Listing 19	sidebar.php . . . . .	39
Listing 20	user.php . . . . .	40

## ACRONYMS

---

HO	Homepage-Requirements
HTML	HyperText Markup Language
PR	Product-Requirements
SE	Search-Requirements
SHO	Shoppingcart-Requirements

## Part I

### DEVELOPMENT OF A BASIC WEB BROWSER

## INTRODUCTION

---

This chapter will provide an overview over the document and recapture the requested requirements.

### 1.1 DOCUMENT OVERVIEW

The document is divided into seven chapters that will describe different aspects of the developed program:

[chapter 1](#) provides an overview over the document and the specified requirements, alongside certain assumptions that were made during the development.

[chapter 2](#) will give a short outline of the requirements that were fulfilled, as well as mention added functionality that was not requested.

[chapter 3](#) will provide a high level overview over the system's architecture.

[chapter 4](#) is a short user guide that will describe the usage of the program by leading the reader through a selected choice of use cases to accomplish certain tasks.

[chapter 5](#) will be based upon [chapter 3](#) and provide an in-depth explanation of implementation-details.

[chapter 6](#) will outline how testing was performed and which cases have been covered.

[chapter 7](#) will provide a reflection of the development process and the program and highlight areas of interest from the developer's point-of-view.

### 1.2 REMIT

The remit will summarize the requirements provided in the document *Systems Programming & Scripting (2010/2011) Assessment Thee* and list all assumptions made in respect to a certain requirement.

For later reference throughout the document, the requirements will be divided into four groups (Homepage-Requirements ([HO](#)), Search-Requirements ([SE](#)), Product-Requirements ([PR](#)), Shoppingcart-Requirements ([SHO](#))) and a unique identifier will be assigned to each requirement.

H001: Home-page allows user to login<sup>1</sup>.

---

<sup>1</sup> Assumption: The use of a username password combo was used instead of a simple userId to guarantee more safety.

H002: Only logged in users are allowed to use the shop<sup>2</sup>.

H003: Provide a link to the Search Request page.

H004: Display customer details after successful login.

SE01: Allow user to search for Name, Author or Topic.

SE02: Provide link to Product Detail.

SE03: Provide link to Search Request form.

PR01: Display book details.

PR02: Provide book id as parameter.

PR03: Provide link to Search Request form.

PR04: Provide link to Shopping Cart.

SH001: Allow updating (add or update items) of cart.

SH002: Allow adding of new items.

SH003: Allow updating of already present items.

SH004: Provide link to Shopping Cart.

SH005: Provide link to Search Request.

---

<sup>2</sup> This requirement was elicited by communication with the MACS company.



## REQUIREMENT'S CHECKLIST

---

The following list shall provide an overview over the fulfilled requirements. The numbers correspond to those used in [chapter 1](#).

HO01: Fulfilled.

HO02: Fulfilled.

HO03: Fulfilled.

HO04: Fulfilled.

SE01: Fulfilled.

SE02: Fulfilled.

SE03: Fulfilled.

PR01: Fulfilled.

PR02: Fulfilled.

PR03: Fulfilled.

PR04: Fulfilled.

SHO01: Fulfilled.

SHO02: Partially - to guarantee the best user experience adding items is only supported from the search.

SHO03: Fulfilled.

SHO04: Fulfilled.

SHO05: Fulfilled.

Apart from these requirements the following features have been added to the application to enhance the user-experience:

REGISTER AS NEW USER: New users are provided with the option to set up a new user account to use the site.

ERROR NOTIFICATION: Upon entering invalid information the user will be informed about the mistakes.

## DESIGN CONSIDERATIONS

---

This chapter provides a general overview of the application's design without explaining implementation details.

Therefore, it will provide a general overview of the application's architecture and point out areas of interest in the design. Concrete implementation details will later be described in [chapter 5](#).

### 3.1 ARCHITECTURAL OVERVIEW

As the application is a web-application, it was necessary to adjust to the behaviour to the stateless nature of HyperText Markup Language ([HTML](#)). This means every page is mainly self-contained. The only lasting data is provided in the form of a database and a user-session.

Pages communicate parameters via POST or GET requests, when the need arises.

The database is used to store the user logins and product information, whereas the user-session stores the current user and the associated shopping cart.

The overall architecture is focused around providing an abstraction layer between the database and the rest of the code:

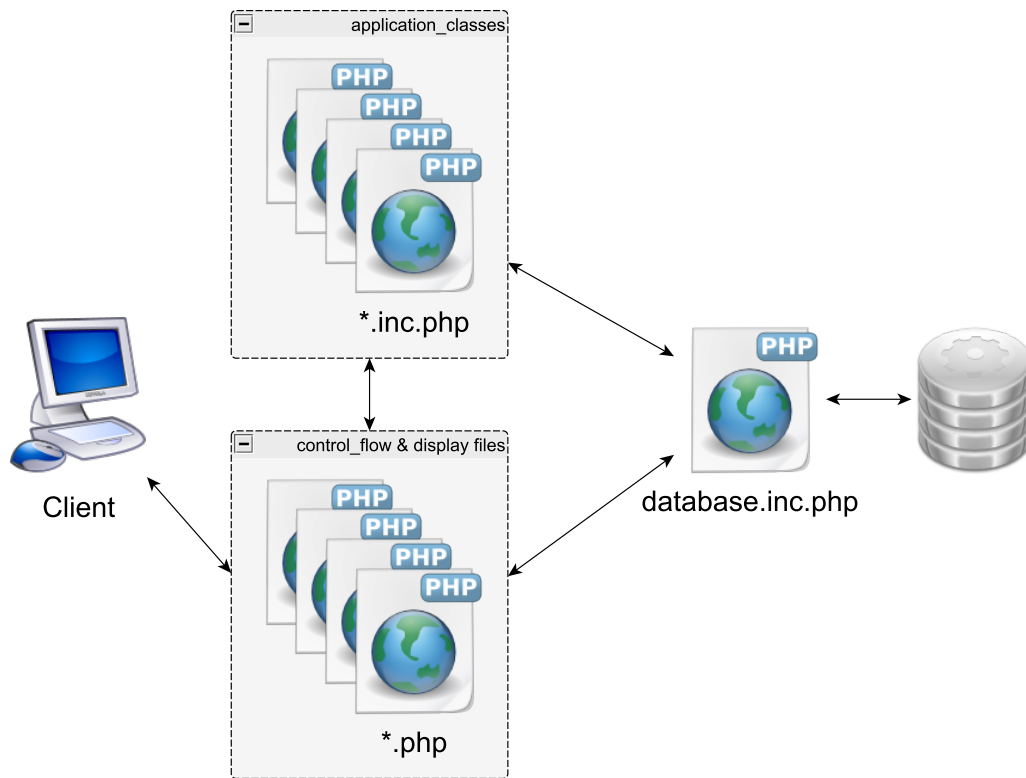


Figure 1: Architectural overview

The `control_flow` classes are the web pages displayed to the user, whereas the `application_classes` are classes that may contain functionality that needs to be reused across different web pages (e.g. the shopping cart)<sup>1</sup>.

This approach was used to ensure the best-practices of *separation of concerns*, as well as *abstraction* and *reuse* of code. This way - in the best case scenario - only one file needs to be changed when a function needs to be changed. Especially when a different database shall be used only the `database.inc.php` file would need to be changed.

<sup>1</sup> Therefore the suffix of `.inc.php` was used to indicate that these files are meant to be *included* in other php files.

## USER GUIDE

---

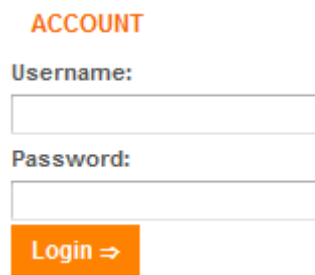
In this chapter an overview over the possible use-cases a user can accomplish will be provided, including pictured tutorials about how to achieve a use case.

The described use cases are:

- Login (and registering).
- Searching for products.
- Adding items to the shopping cart.
- Editing items in the shopping cart.

### 4.1 LOGIN (AND REGISTERING)

When the page is loaded and the user is not yet logged in, a sidebar will provide the necessary form to allow the user to login:



A login panel with the title "ACCOUNT" in orange. Below the title are two labels: "Username:" and "Password:". Each label is followed by a white input field with a thin grey border. Below the password field is an orange button with the text "Login =>" in white.

Figure 2: Login panel

If the user does not yet have a login, he can either click on the Register-tab in the header or use the link provided in the homepage-text.

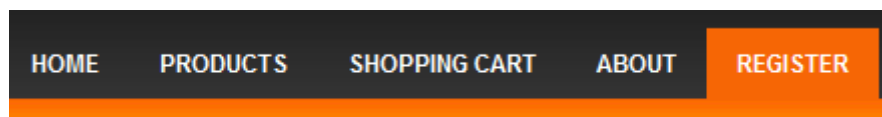


Figure 3: The header links

This will open up the register website, where the user enter his details and create an account. Should the entered details not fulfil the requirements (not entering the same password twice, entering no data in a field), the form will display the errors and urge the user to correct them:

The image shows a registration form with the following fields and error messages:

- First Name:** A text input field with a red error message: "Please enter a first name."
- Last Name:** A text input field with a red error message: "Please enter a last name."
- Username:** A text input field with a red error message: "Please enter a username."
- Address:** A text input field with a red error message: "Please enter an address."
- Password:** A password input field (masked with dots) with no error message.
- Password (repeat):** A password input field with a red error message: "The entered passwords did not match."

At the bottom of the form is an orange button labeled "Next >".

Figure 4: The register form displays errors.

Should the user try to access a page without logging in first a redirection to the main page will occur and an information will be displayed:

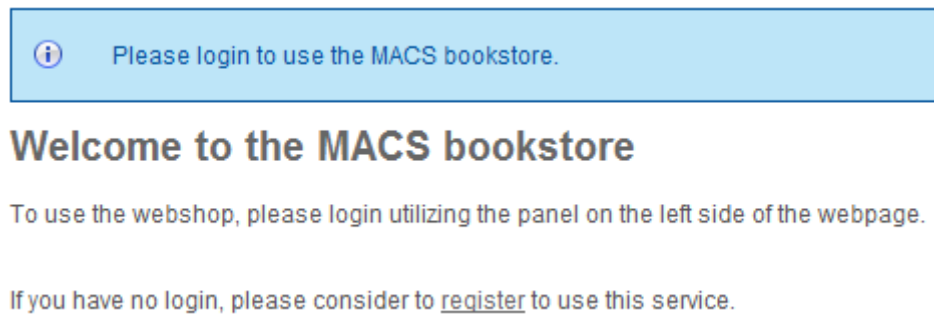


Figure 5: Information to login first

## 4.2 SEARCHING FOR PRODUCTS

After logging in the user can access the search page by clicking on the Products link in the header and the search page will be displayed.

On that page the user can check the category that shall be search, as well as provide a string that shall be searched:

Category:

Title	Subject	Cost	Details
Clean Code	Computer Science	47.99	<a href="#">Details</a>
Code Complete	Computer Science	49.99	<a href="#">Details</a>

Figure 6: Searching for products

The possible categories are: Title, Author, Publisher and Topic. If the user does not select a category and attempts a search then all categories will be used to attempt a match.

If no search string is entered all available products will be displayed.

## 4.3 ADDING ITEMS TO THE SHOPPING CART

Items can be added from the details page of an item. To reach this page a click on the link in the search-results list is enough.

On the the details page the user can fill out a form that add the chosen number of products to the shopping cart:

# Clean Code

Robert Martin

Subject: Computer Science

Cost: 47.99

Available: 5

Quantity:

Add to current quantity? ☐

Add to cart

Figure 7: Adding a product to the shopping cart.

When the product is successfully added to the shopping cart the website will display a message to confirm it:

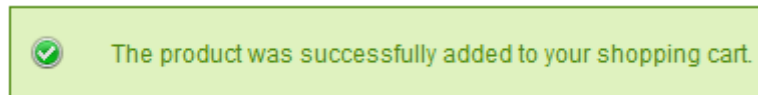


Figure 8: Product successfully added.

Should the user provide an incorrect number of products (e.g. none or not a number) the website will notify the user about the error.

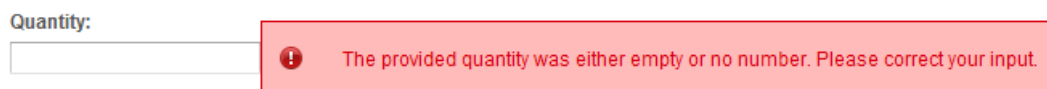


Figure 9: User is informed about incorrect input.

#### 4.4 EDITING ITEMS IN THE SHOPPING CART

When the user clicks on the shopping cart link in the header, this will redirect him to the shopping cart. There all items will be displayed in a table and it is possible to remove items that are currently in the cart, as well as update the current quantity of items:

## Shopping Cart








Item	Quantity		Price
<a href="#">Clean Code</a>	<input type="text" value="1"/>	 Delete  Edit	47.99
<a href="#">Code Complete</a>	<input type="text" value="1"/>	 Delete  Edit	49.99
Sum:			97.98

Figure 10: The shopping cart.

When the user clicks the delete-button () the item that is present in that table row will be removed from the shopping cart.

When the user enters a new quantity and clicks the edit-button () the item's quantity will be updated to the one entered.

Should the user enter an illegal quantity, the web page will display an error and the operation will not be executed.

 Either item was not found, or quantity turned negative.

## Shopping Cart




Item	Quantity		Price
<a href="#">Clean Code</a>	<input type="text" value="1"/>	 Delete  Edit	47.99
<a href="#">Code Complete</a>	<input type="text" value="3"/>	 Delete  Edit	49.99
Sum:			197.96

Figure 11: Illegal edit is prevented.



This chapter will guide the reader through the different requirements and their implementation in the application.

Having finished with the requirements a section highlighting certain special cases that were encountered during the development will be described.

## 5.1 IMPLEMENTATION OF THE REQUIREMENTS

### 5.1.1 Homepage-Requirements

The requirements under the H0-label were fulfilled in the following manner:

#### 5.1.1.1 Provide a login

The graphical representation of the login is provided in the `login_form.inc.php` file, whereas the login-logic (the checking of the username and password) is provided via the `login.inc.php` file.

This way the developer can reuse the login form by including the `login_form.inc.php` file and the page the form will redirect to in it's form field must implement the `login.inc.php` as well.

In the current representation of the web page this page is the index page:

```
1 <form action="index.php" method="post" accept-charset="utf-8">
```

Listing 1: Redirecting the `login_form.inc.php` to the index page

#### 5.1.2 Restrict access to logged-in users

This feature is provided in the `config.inc.php`-file by declaring the function `redirect_invalid_users()`:

```
1 function redirect_invalid_user($check = 'user', $destination = 'index.php', $protocol
   = 'http://') {
2     if (!isset($_SESSION[$check])) {
3         $url = $protocol . BASE_URL . $destination . '?invalid=true';
4         header("Location:$url");
5         exit();
6     }
7 }
```

---

Listing 2: Redirecting invalid users

This function checks if a user is already saved in the Session and will only allow user's where this is the case to proceed. The session-variable is set in the function that checks the login.

### 5.1.3 *Provide a link to the search request*

This requirements is fulfilled by providing a header via the `header.php` file that provides links to the major pages.

### 5.1.4 *Show user details after logging in*

This requirements is fulfilled by a call to the header function with the `customer_details.php` file as destination, after a successful login.

## 5.2 SEARCH REQUIREMENTS

The search requirements were fulfilled in the following manner:

### 5.2.1 *Search functionality*

The search functionality is provided via two files: the form to enter the possible values is declared in the `products.php` file, whereas the real search function is declared in the `database.inc.php` file.

The search function is declared in the database file, as it relies heavily on SQL to perform the search.

Therefore an appropriate string is concatenated that represents the desired search:

```

1 function search_items($search, $category=null) {
2     global $dbc;
3     $items = array();
4     $query;
5     if (isset($category)) {
6         $query = "SELECT i.I_ID, i.I_TITLE, i.I_A_ID, i.I_PUBLISHER, i.
          I_SUBJECT, i.I_COST, i.I_STOCK FROM items i, authors a WHERE
          I_A_ID = A_ID AND ";
7         switch ($category) {
8             case 'Title':
9                 $query.= "i.I_TITLE like '%$search%'";
10                break;
11                case 'Author':

```

```

12         $query.= "a.A_FNAME like '%$search%' OR a.A_LNAME like
13             '%$search%'";
14         break;
15     case 'Publisher':
16         $query.= "i.I_PUBLISHER like '%$search%'";
17         break;
18     case 'Topic':
19         $query.= "i.I_SUBJECT like '%$search%'";
20         break;
21     default:
22         $query = "SELECT DISTINCT i.I_ID, i.I_TITLE, i.I_A_ID,
23             i.I_PUBLISHER, i.I_SUBJECT, i.I_COST, i.I_STOCK
24             FROM items i, authors a WHERE I_A_ID = A_ID AND ";
25         $query .= "i.I_TITLE like '%$search%' OR i.I_PUBLISHER
26             like '%$search%' OR i.I_SUBJECT like '%$search%'
27             OR a.A_FNAME like '%$search%' OR a.A_LNAME like '%
28             $search%'";
29     }
30 } else {
31     $query = "SELECT DISTINCT i.I_ID, i.I_TITLE, i.I_A_ID, i.I_PUBLISHER,
32         i.I_SUBJECT, i.I_COST, i.I_STOCK FROM items i, authors a WHERE
33         I_A_ID = A_ID AND I_A_ID = A_ID AND ";
34     $query .= "i.I_TITLE like '%$search%' OR i.I_PUBLISHER like '%$search%'
35         OR i.I_SUBJECT like '%$search%' OR a.A_FNAME like '%$search%'
36         OR a.A_LNAME like '%$search%'";
37 }
38 $result = mysqli_query($dbc, $query);
39 while ($row = $result->fetch_row()) {
40     $itemId = $row[0];
41     $quantity = 0;
42     $title = $row[1];
43     $authorId = $row[2];
44     $publisher = $row[3];
45     $subject = $row[4];
46     $cost = $row[5];
47     $stock = $row[6];
48     $item = new Item($itemId, $quantity, $title, $authorId, $publisher,
49         $subject, $cost, $stock);
50     array_push($items, $item);
51 }
52 return $items;
53 }

```

Listing 3: The search function

This search either just compares one attribute (or two in the case of the author) depending on the category, or it tries to find the provided string in all fields.

### 5.2.2 Links

The links are provided via the header from the header.php file.

## 5.3 PRODUCT REQUIREMENTS

### 5.3.1 Display product details

The product details are displayed via the product\_details.php file that queries the database via the database.inc.php file and the id that was provided from the search form via GET-request.

After that the website is created.

### 5.3.2 Provide links

The links are provided via the header from the header.php file.

## 5.4 SHOPPING CART REQUIREMENTS

The shopping cart requirements were implemented slightly different than requested:

### 5.4.1 Updating of already present items

This functionality is provided by the shopping\_cart.php file. It declares a form in every line of the table that will allow the user to either delete an item or edit its quantity:

```

1 <form action="" method="post" accept-charset="utf-8">
2     <input type="text" <?php echo 'name="quantity" id="quantity" value="' . $value
   ->quantity . "''; ?>/>
3     <button type="submit" <?php echo 'name="delete" id="delete" value="' . $value
   ->itemId . "''; ?> Delete</
   button>
4     <button type="submit" <?php echo 'name="edit" id="edit" value="' . $value->
   itemId . "''; ?> Edit</
   button>
5 </form>

```

Listing 4: Shopping cart form for editing an existing item.

As can be seen the form will call the same page again.

The page will then determine if it was called via a POST-request and, if this is the case, perform the specified action depending on the button that was pressed.

As can be seen from the form, the affected item will be determined via the button's value.

#### 5.4.2 *Adding items*

While implementing the shopping cart the approach to allow a user to add items from the shopping cart page seemed counter-intuitive.

Well known sites like <http://www.amazon.co.uk> only allow the addition of item's from the search. As this seemed a reasonable approach it was copied - why would a user go to the shopping cart to add new items?

#### 5.4.3 *Provide links*

The links are provided via the header from the `header.php` file.

### 5.5 DETAILS OF THE IMPLEMENTATION

#### 5.5.1 *Modularity*

While implementing the bookstore a modular approach was taken to minimise the repetition of code.

Therefore many files are includes: for example the header, footer and sidebar of every web page are implemented in one php file that is included in the page (`footer.php`, `sidebar.php`, `header.php`).

Moreover a central file included by every page was created that holds all information that needs to be accessible to all pages (e.g. access to the database). This file is called `config.inc.php` and can be seen in the appendix.

In this file a variable is defined (`$live`) that decides whether whole error-messages are printed or just a short error-notice. Upon releasing the bookstore this variable should be set to `true`.

#### 5.5.2 *Form functions*

Apart from this the `form_functions.inc.php` file provides a convenience function to create form elements of the type text or password. This way standard form elements can be created easily and with reliable error-handling. Only a reference to an array where errors can be stored in needs to be provided.

```

1 function create_form_input($name, $type, $errors) {
2     $value = null;
3     if (isset($_POST[$name])) {
4         $value = $_POST[$name];

```

```

5         if ($value && get_magic_quotes_gpc()) {
6             $value = stripslashes($value);
7         }
8     }
9     if ($type == 'text' || $type == 'password') {
10        echo '<input type="' . $type . '"name="' . $name . '"id="' . $name . '
        "'';
11    }
12    if ($value) {
13        echo 'value="' . htmlspecialchars($value) . '"';
14    }
15    if (array_key_exists($name, $errors)) {
16        echo '</><span class="error">' . $errors[$name] . '</span>';
17    } else {
18        echo '</>';
19    }
20 }

```

Moreover nearly every form is handled on the same page it is declared on (except the displaying of product details): this approach allows the function that creates the form elements to reuse values in the POST variable, if they are present. This way the form remembers the old input, but will still display the error-messages.

## TESTING

The testing of the application was performed only on the live web page.

The following cases were tested:

Under Test	Input	Output
Login	No username	Display error to enter username.
Login	No password	Display error to enter password.
Login	Username and password do not match	Display error that login failed.
Register	Some field left empty	Display error that field must contain a value.
Register	Passwords do not match	Display error that passwords do not match.
Access products page	Not logged in	Display information that login is needed.
Access shopping cart page	Not logged in	Display information that login is needed.
Search books	Provide no string	All books are returned.
Search books	Provide string	All books matching the search query in any field are returned.
Search books	Provide string and category	Only books matching the search query in the given category are returned.
Add book	Valid input	Book is added to shopping cart.
Add book	Valid input and add to current quantity is checked	Quantity is increased (book was present).

Add book	Valid input and add to current quantity is checked	Book is added with specified quantity (book was not present).
Add book	Negative quantity	Display error message.
Add book	Non-numeric input	Display error message.
Update book	Delete book	Book is removed from shopping cart.
Update book	Edit quantity (valid)	Quantity is changed.
Update book	Edit quantity (invalid)	Quantity is not changed. Error message is shown.
Logout	–	Logout message is displayed and session destroyed.

Table 1: Performed tests.



## CONCLUSIONS

---

To conclude this report a short summary of the achieved goals (apart from fulfilling the requirements) should be provided:

The application was designed with a focus on modularity and a clear separation between the program logic and the database. This approach seems to have been successfully carried out.

Moreover the application tries to notify the user about all changes made (added an item to the shopping cart succeeded or failed) to allow the user a comfortable browsing experience as one would expect from a modern web page.

However, taking into account that this was the first dynamic web page developed by me, a lot of design issues still remain: even though frameworks like Ruby on Rails and Python with Django are able to provide a Model-View-Controller approach in the web world, the try to do the same in my first php project failed.

If I would redo this web page I would probably try it again, now knowing how basic interaction is achieved and how the standard setup of a php page is achieved.

However, another approach would be to use a MVC-framework.

Apart from the design issue, the application should meet the requirements and deliver a modular program.

## Part II

### APPENDIX

APPENDIX: SOURCE CODE

---

## A.1 WEBSITES

*Project: Websites*

```
1 <?php
2 require('./includes/config.inc.php');
3 require(MYSQL);
4 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
5     include './includes/login.inc.php';
6 }
7 require('./includes/header.php');
8 require('./includes/sidebar.php');
9 ?>
10 <div id="content" class="column-right">
11     <?php
12         if (isset ($_GET['invalid']))
13         {
14             echo '<div class="info">Please login to use the MACS bookstore.</div>'
15             ;
16         }
17     ?>
18     <h3>Welcome to the MACS bookstore</h3>
19
20     <p>To use the webshop, please login utilizing the panel on the left side of
21         the webpage.</p>
22     <p>If you have no login, please consider to <a href="register.php">register</a>
23         > to use this service.</p>
24 </div>
25 <?php
26 require('./includes/footer.php');
27 ?>
```

Listing 5: index.php

```
1 <?php
2 require('./includes/config.inc.php');
3 require('./includes/header.php');
4 require(MYSQL);
5 require('./includes/sidebar.php');
6 redirect_invalid_user();
7 ?>
8 <div id="content" class="column-right">
```

```

9      <h3>Customer details:</h3>
10
11     <h4>CustomerId:</h4>
12     <p>
13         <?php
14             $userId = $_SESSION['user']->userId;
15             echo "$userId";
16         ?>
17     </p>
18     <h4>Firstname:</h4>
19     <p>
20         <?php
21             $firstname = $_SESSION['user']->firstName;
22             echo "$firstname";
23         ?>
24     </p>
25     <h4>Lastname:</h4>
26     <p>
27         <?php
28             $lastname = $_SESSION['user']->lastName;
29             echo "$lastname";
30         ?>
31     </p>
32     <h4>Address:</h4>
33     <p>
34         <?php
35             $address = $_SESSION['user']->address;
36             echo "$address";
37         ?>
38     </p>
39 </div>

```

Listing 6: customer\_details.php

```

1 <?php
2 include './includes/config.inc.php';
3 redirect_invalid_user();
4 $_SESSION = array();
5 session_destroy();
6 setcookie(session_name(), '', time() - 24800);
7 $page_title = 'Logout';
8 include MYSQL;
9 include './includes/header.php';
10 ?>
11 <div id="content" class="column-right">
12     <h3>
13         Logged out
14     </h3>
15     <p>
16         Thank you for using the MACS bookstore. We hoped you enjoyed your experience.
17     </p>

```

```

18 </div>
19 <?php
20 include './includes/footer.php';
21 ?>

```

Listing 7: logout.php

```

1  <?php
2  include './includes/config.inc.php';
3  include MYSQL;
4  include './includes/form_functions.inc.php';
5
6  redirect_invalid_user();
7
8  include './includes/header.php';
9  include './includes/sidebar.php';
10
11 $order_errors = array();
12
13 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
14     $productid = $_POST['id'];
15     $product = get_product_by_id($productid);
16     $quantity = $_POST['quantity'];
17     if (empty($quantity) || !is_numeric($quantity)) {
18         $order_errors['quantity'] = 'The provided quantity was either empty or
19         no number. Please correct your input.';
20     }
21     if (empty($order_errors)) {
22         try {
23             $product->quantity = $quantity;
24             $shopping_cart = $_SESSION['cart'];
25             if (isset($_POST['add'])) {
26                 $shopping_cart->add_item_increase_quantity($product);
27                 $added = true;
28             } else {
29                 $shopping_cart->add_item($product);
30                 $added = true;
31             }
32         } catch (Exception $e) {
33             $is_error = $e;
34         }
35     }
36     $productid = $_GET['id'];
37     if (isset($productid)) {
38         $product = get_product_by_id($productid);
39     }
40 }
41 <div id="content" class="column-right">
42     <?php
43     if (isset($is_error)) {

```

```

44         echo '<div class="error">Can not enter a negative quantity!</div>';
45     }
46     if (isset($added)) {
47         echo '<div class="success">The product was successfully added to your
48             shopping cart.</div>';
49     }
50     ?>
51     <h1>
52         <?php echo $product->title ?>
53     </h1>
54     <h2>
55         <?php echo $product->authorId ?>
56     </h2>
57     <p>
58         <strong>
59             Subject:
60         </strong>
61         <?php echo $product->subject; ?>
62     </p>
63     <p>
64         <strong>
65             Cost:
66         </strong>
67         <?php echo $product->cost; ?>
68     </p>
69     <p>
70         <strong>
71             Available:
72         </strong>
73         <?php echo $product->stock; ?>
74     </p>
75     <form action="" method="post" accept-charset="utf-8">
76         <p>
77             <label for="quantity"><strong>Quantity:</strong></label>
78             <br/>
79             <?php create_form_input('quantity', 'text', $order_errors) ?>
80         </p>
81         <p>
82             <label for="add">
83                 <strong>Add to current quantity?</strong>
84             </label>
85             <input type="checkbox" name="add" value="add" id="add"/>
86         </p>
87         <p>
88             <input type="hidden" name="id" <?php echo 'value="' .
89                 $productid . '"' ?>/>
90         </p>
91         <input type="submit" name="submit_button" value="Add to cart &
            rarr;" id="submit_button" class="formbutton"/>
92     </p>

```

```

92         </form>
93     </div>
94     <?php
95         include './includes/footer.php';
96     ?>

```

Listing 8: product\_details.php

```

1  <?php
2  include './includes/config.inc.php';
3  include MYSQL;
4
5  redirect_invalid_user();
6
7  include './includes/form_functions.inc.php';
8
9  include './includes/header.php';
10 include './includes/sidebar.php';
11
12 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
13     $quantity = $_POST['quantity'];
14     if (isset($_POST['edit'])) {
15         try {
16             $edit = $_POST['edit'];
17             $product = $_SESSION['cart']->get_item_for_id($edit);
18             $product->set_quantity($_POST['quantity']);
19             $_SESSION['cart']->add_item($product);
20         } catch (Exception $e) {
21             $is_error = $e;
22         }
23     }
24     if (isset($_POST['delete'])) {
25         try {
26             $delete = $_POST['delete'];
27             $product = $_SESSION['cart']->get_item_for_id($delete);
28             $_SESSION['cart']->remove_item($product);
29         } catch (Exception $e) {
30             $is_error = $e;
31         }
32     }
33 }
34
35 $items = $_SESSION['cart']->get_items();
36 $sum = 0;
37 ?>
38 <div id="content" class="column-right">
39     <?php
40     if (isset($is_error)) {
41         echo '<div class="error">Either item was not found, or quantity turned
42         negative.</div>';

```

```

43         if (isset($delete)) {
44             echo '<div class="success">Item successfully deleted.</div>';
45         }
46         if (isset($edit)) {
47             echo '<div class="success">Quantity successfully changed.</div>';
48         }
49     }
50     ?>
51     <h1>
52         Shopping Cart
53     </h1>
54     <?php
55     if (isset($items)) {
56     ?>
57         <table>
58             <thead>
59                 <th>
60                     Item
61                 </th>
62                 <th>
63                     Quantity
64                 </th>
65                 <th>
66                     Price
67                 </th>
68             </thead>
69             <tbody>
70                 <?php foreach ($items as $value) {
71                 ?>
72                     <tr>
73                         <td>
74                             <?php echo '<a href=product_details.php?id=' .
75                                 $value->itemId . '>' . $value->title . '
76                                 </a>'; ?>
77                         </td>
78                         <td>
79                             <form action="" method="post" accept-charset="
80                                 utf-8">
81                                 <input type="text" <?php echo 'name="
82                                     quantity" id="quantity" value="' .
83                                     $value->quantity . '"'; ?>/>
84                                 <button type="submit" <?php echo 'name
85                                     ="delete" id="delete" value="' .
86                                     $value->itemId . '"'; ?><img src=
87                                     "images/shopping-basket--minus.png
88                                     "> Delete</button>
89                                 <button type="submit" <?php echo 'name
90                                     ="edit" id="edit" value="' .
91                                     $value->itemId . '"'; ?><img src=

```



```

81                                     "images/shopping-basket--pencil.
82                                     png"/> Edit</button>
83                                     </form>
84                                     </td>
85                                     <td>
86                                     <?php $sum += $value->cost * $value->quantity;
87                                     echo $value->cost; ?>
88                                     </td>
89                                     </tr>
90                                     <?php
91                                     }
92                                     ?>
93                                     <tr>
94                                     <td>
95                                     <strong>Sum:</strong>
96                                     </td>
97                                     <td></td>
98                                     <td>
99                                     <?php echo $sum; ?>
100                                    </td>
101                                    </tr>
102                                    </tbody>
103                                    </table>
104                                    <?php
105                                    } else {
106                                    echo 'Your shopping cart is empty';
107                                    }
108                                    ?>
109                                    </div>
110 <?php
include './includes/footer.php';
?>

```

Listing 9: shopping\_cart.php

## A.2 INCLUDES

*Project: Includes*

```

1 <?php
2
3 $live = false;
4
5 define('BASE_URI', '/Assessment_Three');
6 // TODO: Change before deployment
7 define('BASE_URL', 'localhost/Assessment_Three/');
8 define('MYSQL', './includes/database.inc.php');
9

```

```

10 include_once('user.php');
11 include_once('item.php');
12 include_once('shopping_cart.inc.php');
13
14 session_start();
15
16 function error_handler($e_number, $e_message, $e_file, $e_line, $e_vars) {
17     global $live;
18     $message = "An error occured in script $e_file on line $e_line: $e_message";
19     $message .= "<pre>" . print_r(debug_backtrace(), true) . "<pre>\n";
20
21     if (!$live) {
22         echo '<div class="error">' . nl2br($message) . '<div>';
23     } else {
24         // TODO: handle error when site was live: send email.
25     }
26     return true;
27 }
28
29 set_error_handler('error_handler');
30
31 /**
32  * Will redirect the user to another page if the check variable is not set in the
33  * session.
34  * @param <type> $check The variable the session should be checked for.
35  * @param <type> $destination The redirection url.
36  * @param <type> $protocol The protocol to use to redirect the user.
37  */
38 function redirect_invalid_user($check = 'user', $destination = 'index.php', $protocol
39     = 'http://') {
40     if (!isset($_SESSION[$check])) {
41         $url = $protocol . BASE_URL . $destination . '?invalid=true';
42         header("Location:$url");
43         exit();
44     }
45 }
46 ?>

```

Listing 10: config.inc.php

```

1 <?php
2
3 include_once 'user.php';
4
5 define('DB_USER', 'fjb2');
6 define('DB_PASSWORD', 'fjb2');
7 define('DB_HOST', 'localhost');
8 define('DB_PORT', 8889);
9 define('DB_NAME', 'fjb2');
10

```

```

11 $dbc = mysqli_connect(DB_HOST, DB_USER, DB_PASSWORD, DB_NAME);
12
13 /**
14  *
15  * @global $dbc
16  * @param <type> $data
17  * @return <type>
18  */
19 function escape_data($data) {
20     global $dbc;
21     return mysqli_real_escape_string($trim($data), $dbc);
22 }
23
24 /**
25  * Creates a password hash for the provided string and returns it.
26  *
27  * @global $dbc
28  * @param <type> $password The password that shall be hashed.
29  * @return <type> The hashed password in binary form.
30  */
31 function get_password_hash($password) {
32     global $dbc;
33     return mysqli_escape_string($dbc, hash_hmac('SHA256', $password, '
34         macsbooksstore', true));
35 }
36
37 /**
38  * Will return an array with the user(s) corresponding to the username and password
39  * provided.
40  *
41  * @global $dbc
42  * @param <type> $username The username that shall be queried.
43  * @param <type> $password The password that shall be queried.
44  * @return array The array with the user(s)
45  */
46 function get_users_by_username_and_password($username, $password=null) {
47     global $dbc;
48     $users = array();
49     $query = "SELECT C_ID, C_USERNAME, C_FNAME, C_LNAME, C_ADD FROM customers
50         WHERE C_USERNAME = '$username'";
51     if (isset ($password))
52     {
53         $query .= "AND C_PASSWORD = '" . get_password_hash($password) . "'";
54     }
55     $result = mysqli_query($dbc, $query);
56     while ($row = $result->fetch_row()) {
57         $user = new User();
58         $user->userId = $row[0];
59         $user->username = $row[1];
60         $user->firstName = $row[2];
61         $user->lastName = $row[3];

```

```

59         $user->address = $row[4];
60         array_push($users, $user);
61     }
62     return $users;
63 }
64
65 /**
66  *
67  * @global <type> $dbc
68  * @return array
69  */
70 function get_all_items() {
71     global $dbc;
72     $items = array();
73     $query = "SELECT I_ID, I_TITLE, I_A_ID, I_PUBLISHER, I_SUBJECT, I_COST,
74             I_STOCK FROM items";
75     $result = mysqli_query($dbc, $query);
76     while ($row = $result->fetch_row()) {
77         $itemId = $row[0];
78         $quantity = 0;
79         $title = $row[1];
80         $authorId = $row[2];
81         $publisher = $row[3];
82         $subject = $row[4];
83         $cost = $row[5];
84         $stock = $row[6];
85         $item = new Item($itemId, $quantity, $title, $authorId, $publisher,
86             $subject, $cost, $stock);
87         array_push($items, $item);
88     }
89     return $items;
90 }
91
92 /**
93  * Will return a list of all items that correspond to the search-criterion.
94  *
95  * @param <type> $search The string that will be matched.
96  * @param <type> $category The category of the matching.
97  * @return A list of items that fulfill the search criterion.
98  */
99 function search_items($search, $category=null) {
100     global $dbc;
101     $items = array();
102     $query;
103     if (isset($category)) {
104         $query = "SELECT i.I_ID, i.I_TITLE, i.I_A_ID, i.I_PUBLISHER, i.
105             I_SUBJECT, i.I_COST, i.I_STOCK FROM items i, authors a WHERE
106             I_A_ID = A_ID AND ";
107         switch ($category) {
108             case 'Title':
109                 $query.= "i.I_TITLE like '%$search%'";

```

```

106         break;
107     case 'Author':
108         $query.= "a.A_FNAME like '%$search%' OR a.A_LNAME like
109             '%$search%'";
110         break;
111     case 'Publisher':
112         $query.= "i.I_PUBLISHER like '%$search%'";
113         break;
114     case 'Topic':
115         $query.= "i.I_SUBJECT like '%$search%'";
116         break;
117     default:
118         $query = "SELECT DISTINCT i.I_ID, i.I_TITLE, i.I_A_ID,
119             i.I_PUBLISHER, i.I_SUBJECT, i.I_COST, i.I_STOCK
120             FROM items i, authors a WHERE I_A_ID = A_ID AND ";
121         $query .= "i.I_TITLE like '%$search%' OR i.I_PUBLISHER
122             like '%$search%' OR i.I_SUBJECT like '%$search%'
123             OR a.A_FNAME like '%$search%' OR a.A_LNAME like '%"
124             $search%'";
125     }
126 } else {
127     $query = "SELECT DISTINCT i.I_ID, i.I_TITLE, i.I_A_ID, i.I_PUBLISHER,
128         i.I_SUBJECT, i.I_COST, i.I_STOCK FROM items i, authors a WHERE
129         I_A_ID = A_ID AND I_A_ID = A_ID AND ";
130     $query .= "i.I_TITLE like '%$search%' OR i.I_PUBLISHER like '%"
131         $search%' OR i.I_SUBJECT like '%"
132         $search%' OR a.A_FNAME like '%"
133         $search%' OR a.A_LNAME like '%"
134         $search%'";
135 }
136 $result = mysqli_query($dbc, $query);
137 while ($row = $result->fetch_row()) {
138     $itemId = $row[0];
139     $quantity = 0;
140     $title = $row[1];
141     $authorId = $row[2];
142     $publisher = $row[3];
143     $subject = $row[4];
144     $cost = $row[5];
145     $stock = $row[6];
146     $item = new Item($itemId, $quantity, $title, $authorId, $publisher,
147         $subject, $cost, $stock);
148     array_push($items, $item);
149 }
150 return $items;
151 }
152 }
153 /**
154  * Will return the product corresponding to the provided id.
155  *
156  * @global $dbc The connection to the database.
157  * @param <type> $productId The productId of the product that shall be retrieved.
158  * @return Item The product that was retrieved from the database.

```

```

146  */
147  function get_product_by_id($productId) {
148      global $dbc;
149      $item;
150      $query = "SELECT I_ID, I_TITLE, A_FNAME, A_LNAME, I_PUBLISHER, I_SUBJECT,
               I_COST, I_STOCK FROM items, authors WHERE items.I_A_ID = authors.A_ID AND
               items.I_ID = '$productId'";
151      $result = mysqli_query($dbc, $query);
152      $row = $result->fetch_row();
153      $itemId = $row[0];
154      $quantity = 0;
155      $title = $row[1];
156      $authorId = $row[2] . ' ' . $row[3];
157      $publisher = $row[4];
158      $subject = $row[5];
159      $cost = $row[6];
160      $stock = $row[7];
161      $item = new Item($itemId, $quantity, $title, $authorId, $publisher, $subject,
                      $cost, $stock);
162      return $item;
163  }
164
165  /**
166   * Inserts a user into the database with the provided parameters.
167   *
168   * @param <type> $first_name The first name.
169   * @param <type> $last_name The last name.
170   * @param <type> $address The address.
171   * @param <type> $username The username.
172   * @param <type> $password The password.
173   * @return <type> Returns true if insert successful, false otherwise.
174   */
175  function insert_user_into_database($first_name, $last_name, $address, $username,
                                   $password){
176      global $dbc;
177      $query = "INSERT INTO customers(C_FNAME, C_LNAME, C_ADD, C_USERNAME,
               C_PASSWORD) VALUES ('$first_name', '$last_name', '$address', '$username',
               '" . get_password_hash($password) . "')";
178      $result = mysqli_query($dbc, $query);
179      if (mysqli_affected_rows ($dbc) == 1)
180      {
181          return true;
182      }
183      else {
184          return false;
185      }
186  }
187
188  ?>

```

Listing 11: database.inc.php

```

1 </div>
2 <div id="footer" class="clear">
3     <div class="footer-box">
4     </div>
5
6     <div class="footer-box">
7     </div>
8
9     <div class="footer-box">
10    </div>
11
12    <div class="footer-box end-footer-box">
13    </div>
14 </div>
15
16 <div id="footer-links">
17     <p>
18         &copy; 2009 sitename. Design by <a href="http://www.spyka.net"
19         >Free CSS Templates</a> and <a href="http://www.
20         justfreetemplates.com">Free Web Templates</a>
21     </p>
22 </div>
23 </div>
24 </body>
</html>

```

Listing 12: footer.php

```

1 <?php
2
3 /**
4  * Creates a form-element with the given inputs.
5  *
6  * @param <type> $name
7  * @param <type> $type
8  * @param <type> $errors
9  */
10 function create_form_input($name, $type, $errors) {
11     $value = null;
12     if (isset($_POST[$name])) {
13         $value = $_POST[$name];
14         if ($value && get_magic_quotes_gpc()) {
15             $value = stripslashes($value);
16         }
17     }
18     if ($type == 'text' || $type == 'password') {
19         echo '<input type="' . $type . '" name="' . $name . '" id="' . $name . ' '
20             . '"';
21     }
22 }

```

```

21     if ($value) {
22         echo 'value="' . htmlspecialchars($value) . '"';
23     }
24     if (array_key_exists($name, $errors)) {
25         echo '/><span class="error">' . $errors[$name] . '</span>';
26     } else {
27         echo '/>';
28     }
29 }

```

Listing 13: form\_functions.inc.php

```

1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/
   DTD/xhtml1-strict.dtd">
2  <html xmlns="http://www.w3.org/1999/xhtml">
3  <head>
4  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
5  <title>
6      <?php if (isset ($page_title))
7          {
8              echo $page_title;
9          }
10         else {
11             echo 'Macs bookstore';
12         }
13     ?>
14 </title>
15 <link rel="stylesheet" type="text/css" href="css/styles.css" />
16 </head>
17 <body>
18 <div id="wrapper">
19     <?php
20     $pages = array(
21         'Home' => 'index.php',
22         'Products' => 'products.php',
23         'Shopping Cart' => 'shopping_cart.php',
24         'About' => 'about.php',
25         'Register' => 'register.php'
26     );
27     $this_page = basename($_SERVER['PHP_SELF']);
28     echo '<div id="top" class="clear">';
29     echo '<h1>Macs Bookstore</h1>';
30     echo '<ul>';
31     foreach ($pages as $k => $v)
32     {
33         echo '<li ' ;
34         if ($this_page == $v)
35         {
36             echo 'class="selected"';
37         }
38         echo '><a href="' . $v . '"><span>', $k . '<span></a></li>';

```



```

39     }
40     echo '</ul>';
41     echo '</div>'
42     ?>
43     <div id="body" class="clear">

```

Listing 14: header.php

```

1  <?php
2
3  class Item {
4
5      public $itemId;
6      public $quantity;
7      public $title;
8      public $authorId;
9      public $publisher;
10     public $subject;
11     public $cost;
12     public $stock;
13
14     function __construct($itemId, $quantity, $title, $authorId, $publisher,
15         $subject, $cost, $stock) {
16         $this->itemId = $itemId;
17         $this->quantity = $quantity;
18         $this->title = $title;
19         $this->authorId = $authorId;
20         $this->publisher = $publisher;
21         $this->subject = $subject;
22         $this->cost = $cost;
23         $this->stock = $stock;
24     }
25
26     function set_quantity($param) {
27         if ($param > 0) {
28             $this->quantity = $param;
29         } else {
30             throw new Exception("Item quantity can not be empty");
31         }
32     }
33 }
34
35 ?>

```

Listing 15: item.php

```

1  <?php
2
3  require_once 'shopping_cart.inc.php';

```

```

4
5 $login_errors = array();
6
7 $username = $_POST['username'];
8 $password = $_POST['password'];
9 if (!empty($username)) {
10     if (!empty($password)) {
11         $result = get_users_by_username_and_password($username, $password);
12         if (count($result) == 1) {
13             $user = $result[0];
14             $_SESSION['user'] = $user;
15             $_SESSION['cart'] = new ShoppingCart();
16             header("Location:customer_details.php");
17         } else {
18             $login_errors['login'] = 'The username and password do not
19                                     match';
20         }
21     } else {
22         $login_errors['password'] = 'Please enter a password';
23     }
24 } else {
25     $login_errors['username'] = 'Please enter a username';
26 }
27 ?>

```

Listing 16: login.inc.php

```

1 <?php
2 if (!isset($login_errors)) {
3     $login_errors = array();
4 }
5 require_once './includes/form_functions.inc.php';
6 ?>
7 <div class="title">
8     <form action="index.php" method="post" accept-charset="utf-8">
9         <p>
10             <?php
11             if (array_key_exists('login', $login_errors)) {
12                 echo '<span class="error">' . $login_errors['login'] .
13                     '</span><br/>';
14             }
15             ?>
16             <label for="username">
17                 <strong>Username:</strong>
18             </label>
19             <br/>
20             <?php create_form_input('username', 'text', $login_errors); ?>
21             <br/>
22             <label for="password">
23                 <strong>Password:</strong>
24             </label>

```

```

24         <br/>
25         <?php create_form_input('password', 'password', $login_errors)
           ; ?>
26         <br/>
27         <input type="submit" value="Login &rArr;" class="formbutton"
           />
28     </p>
29 </form>
30 </div>

```

Listing 17: login\_form.inc.php

```

1 <?php
2
3 /**
4  * A shopping cart holding the products a user wants to buy.
5  */
6 class ShoppingCart {
7
8     private $items;
9
10    function __construct() {
11        $this->items = array();
12    }
13
14    /**
15     *
16     * @param <type> $item
17     */
18    function add_item($item) {
19        $itemId = $item->itemId;
20        if ($item->quantity > 0) {
21            if (array_key_exists($itemId, $this->items)) {
22                unset($this->items[$itemId]);
23            }
24            $this->items[$itemId] = $item;
25        } else {
26            throw new Exception("No negative quantity permitted.");
27        }
28    }
29
30    function add_item_increase_quantity($item) {
31        $itemId = $item->itemId;
32        if (array_key_exists($itemId, $this->items)) {
33            $currentQuantity = $this->items[$itemId]->quantity;
34            if (($item->quantity > 0) || (($currentQuantity - $item->
35                quantity) > 0)) {
36                $this->items[$itemId]->quantity = $currentQuantity +

```

```

37         throw new Exception("No negative quantity permitted.")
38         ;
39     } else {
40         $this->add_item($item);
41     }
42 }
43
44 /**
45  *
46  * @param <type> $item
47  */
48 function remove_item($item) {
49     $itemId = $item->itemId;
50     if (array_key_exists($itemId, $this->items)) {
51         unset($this->items[$itemId]);
52     } else {
53         throw new Exception("Element $item not found in shopping cart.
54             ");
55     }
56 }
57
58 /**
59  * @return <type>
60  */
61 function get_items() {
62     return $this->items;
63 }
64
65 function get_item_for_id($itemId) {
66     if (array_key_exists($itemId, $this->items)) {
67         return $this->items[$itemId];
68     } else {
69         throw new Exception("Item does not exist");
70     }
71 }
72
73 }
74
75 ?>

```

Listing 18: shopping\_cart.inc.php

```

1 <div id="sidebar" class="column-left">
2     <ul>
3         <li>
4             <h4>Account</h4>
5             <?php if (isset($_SESSION['user'])) {
6                 ?>
7                 <div class="title">

```

```

8         <h4>Manage your account</h4>
9     </div>
10    <ul>
11        <li>
12            <a href="customer_details.php" title="
                Show your account details">Account
                details</a>
13        </li>
14        <!-- TODO: add change password function
15    </li>
16            <a href="change_password.php" title="
                Change password">Change password</
                a>
17        </li>-->
18        <li>
19            <a href="logout.php" title="Logout">
                Logout</a>
20        </li>
21    </ul>
22    <?php
23    } else {
24        require('includes/login_form.inc.php');
25    }
26    ?>
27    </li>
28 </ul>
29 </div>

```

Listing 19: sidebar.php

```

1 <?php
2
3 class User {
4
5     public $userId;
6     public $firstName;
7     public $lastName;
8     public $address;
9     public $username;
10    public $password;
11
12    /*function __construct($userId, $firstName, $lastName, $address, $username,
        $password) {
13        $this->userId = $userId;
14        $this->firstName = $firstName;
15        $this->lastName = $lastName;
16        $this->address = $address;
17        $this->username = $username;
18        $this->password = $password;
19    }*/
20

```

```
21 |}  
22 |  
23 |?>
```

Listing 20: user.php

### A.3 TESTS

## BIBLIOGRAPHY

---

- Balzert, Helmut (2009). *Lehrbuch der Software-Technik: Basiskonzepte und Requirements Engineering*. 3. Heidelberg: Spektrum. ISBN: 9783827417053.
- Freeman, Eric and Elisabeth Freeman (2004). *Head First - Design Pattern*. Ed. by Mike Loukides. O'Reilly. ISBN: 0-596-00712-4.
- Martin, Robert C. (2008). *Clean Code: A Handbook of Agile Software Craftsmanship*. 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR. ISBN: 0132350882, 9780132350884.
- McConnell, Steve (2004). *Code Complete, Second Edition*. Redmond, WA, USA: Microsoft Press. ISBN: 0735619670.
- Powers, David (2010). *PHP Solutions: Dynamic Web Design Made Easy*. Friends of ED. ISBN: 978-1-4302-3249-0.
- Sommerville, Ian (2006). *Software Engineering*. 8th ed. Addison Wesley. ISBN: 9780321210265.
- Ullman, Larry (2010). *Effortless E-Commerce with PHP and MySQL*. New Riders. ISBN: 9780321656223.