

**CSY2028 WEB PROGRAMMING
ASSIGNMENT REPORT
TERM II**

PHP - Fran's Furniture

Group 'B'
18406547
Diwas Lamsal

Backend username: admin
Backend password: admin

Link to the video demo: [Uploaded to Northampton Media Space](#)

BSc Computing (All Pathways) CSY2028 Web programming			
Date of Issue:	Sunday, 17th February 2019	Date for Submission:	Sunday 5th May 2019 23:59:59
This assignment is weighted as 50% of the Module's assessment			
Assessment Feedback is provided on the rubric via NILE			

Table of Contents

1. INTRODUCTION	1
1.1 Introduction to the report:	1
1.2 Report Structure:	1
2. DATABASE STRUCTURE	2
3. CHECKLIST OF COMPLETED FEATURES	3
4. IDENTIFICATION OF WEAKNESSES	10
4.1 Weaknesses and Solutions:	10
4.2 Other Solutions:	18
5. TESTING	20
5.1 Manual Testing (Black Box Approach)	20
5.1.1 Staff Perspective Tests	20
5.1.2 Customer Perspective Tests	33
5.1.3 Test Report	41
5.2 PHPUnit Testing (White Box Approach)	42
6. EVALUATION	51
6.1 Bugs or Weaknesses and Possible Solutions	51
6.2 Strengths of the Solution	51
7. CONCLUSION	54
REFERENCES	55

List of Figures

Figure 2. 1 - Entity Relationship Diagram	2
Figure 2. 2 - Tables and Their Columns	2
Figure 3. 1 - Copyright Notice.....	5
Figure 3. 2 - FAQs page.....	5
Figure 3. 3 - Furniture Categories	6
Figure 3. 4 – Hide Unavailable Products	6
Figure 3. 5 – Product Condition.....	7
Figure 3. 6 – Users View Products According to Condition.....	7
Figure 3. 7 – Staff User Accounts.....	7
Figure 3. 8 – Upload Photos to Furniture.....	8
Figure 3. 9 – Updates to Home Page	8
Figure 3. 10 – Contact Page Enquiries.....	9
Figure 4. 1 – Example repeated code on about.php and contact.php	10
Figure 4. 2 – Users Template	11
Figure 4. 3 – Provided page CSS path definition	11
Figure 4. 4 – Static date issue solved.....	12
Figure 4. 5 – Static category pages	12
Figure 4. 6 – How categories are dynamically grabbed and sent accordingly	13
Figure 4. 7 – Example database connection code in provided files.....	13
Figure 4. 8 – Database connection file dbconnect.php	14
Figure 4. 9 – SELECT query with category id provided as restriction	14
Figure 4. 10 - Database class member functions.....	15
Figure 4. 11 – Categories in admin and customer area	16
Figure 4. 12 – Add and edit category.....	17
Figure 4. 13 – File Folder Structure	18
Figure 4. 14 – Example MVC method.....	19
Figure 5. 1 – Invalid Login	20
Figure 5. 2 – Add new User.....	22
Figure 5. 3 – Use existing username	23
Figure 5. 4 - Staff attempt to delete or edit another user.....	23
Figure 5. 5 – Admin Delete User	24
Figure 5. 6 – Admin Edit User	24
Figure 5. 7 – Category Errors	26
Figure 5. 8 – Add new Category (valid)	26
Figure 5. 9 – Delete Category attempts Unsuccessful and Successful	27
Figure 5. 10 – Add new Furniture	29
Figure 5. 11 – Furniture Availability	29
Figure 5. 12 – Enquiry State Pending.....	30
Figure 5. 13 – Enquiry State Completed.....	31
Figure 5. 14 – Add Update	32
Figure 5. 15 – Delete Update	32
Figure 5. 16 – Home Page	34
Figure 5. 17 – About Us Page	34
Figure 5. 18 – FAQs Page.....	35
Figure 5. 19 – Contact Page	35
Figure 5. 20 – Our Furniture Page.....	36
Figure 5. 21 – Valid Enquiry Submission	37

Figure 5. 22 – Invalid Enquiry Submission.....	37
Figure 5. 23 – View All Furniture with Pagination	39
Figure 5. 24 – View Second Hand Beds.....	39
Figure 5. 25 – Search Term Red inside Category Sofas.....	40
Figure 5. 26 – Pagination Fourth Page Test.....	40
Figure 5. 27 – PHPUnit test and report generation	42
Figure 5. 28 – PHPUnit code coverage	42
Figure 5. 29 – PHPUnit functions and code coverage	43
Figure 5. 30 – Function checkCategory.....	43
Figure 5. 31 – checkCategory test class.....	43
Figure 5. 32 – Function checkFurniture	44
Figure 5. 33 – checkFurniture test class.....	44
Figure 5. 34 – Function checkUpdate.....	45
Figure 5. 35 – checkUpdate test class.....	45
Figure 5. 36 – Function checkUser.....	46
Figure 5. 37 – checkUser test class.....	46
Figure 5. 38 – Function checkEnquiry	47
Figure 5. 39 – checkEnquiry test class.....	47
Figure 5. 40 – Function checkIsCategoryAvailable.....	48
Figure 5. 41 – checkIsCategoryAvailable test class.....	48
Figure 5. 42 – Function checkIsUsernameAvailable.....	49
Figure 5. 43 – checkIsUsernameAvailable test class.....	49
Figure 5. 44 – Function checkConfirmPassword	50
Figure 5. 45 – checkConfirmPassword test class	50
Figure 6. 1 – Htdocs file and URL example	52
Figure 6. 2 – Super administrator and normal staff.....	52
Figure 6. 3 – Checking username function	53
Figure 6. 4 – Random name generation for images.....	53

1. INTRODUCTION

1.1 Introduction to the report:

The report provides required checklist, tests and other requirements asked in the brief. The sections that follow will show all the necessary content required for a complete assignment.

1.2 Report Structure:

- **Introduction:**
This section introduces the report by providing an introduction to each section and what they contain.
- **Database Structure:**
This section shows the database structure used in the system. It would contain any relevant ERD implemented in the system.
- **Checklist of Completed Features:**
The tabular breakdown of completed features along with the files associated with them. This section also displays what extra work has been done to refine the system other than the ten features provided.
- **Identification of Weaknesses:**
The discovered list of weaknesses that existed within the system provided with the brief. Also contains the proposed fixes to the weaknesses discovered.
- **Testing:**
The testing section includes both the Black Box Testing in the form of test logs and screenshots as well as White Box Testing (PHPUnit) evidences.
- **Evaluation:**
Bugs or weaknesses that exist in the final solution. Any relevant improvements or that could be made are included. This section also includes the strengths of the solution
- **Conclusions:**
This section contains feedbacks about using PHP MVC approach and further recommendations about the project.
- **References:**
Any help taken from anywhere besides the lecture material and assignment brief has been referenced here.

2. DATABASE STRUCTURE



Figure 2. 1 - Entity Relationship Diagram

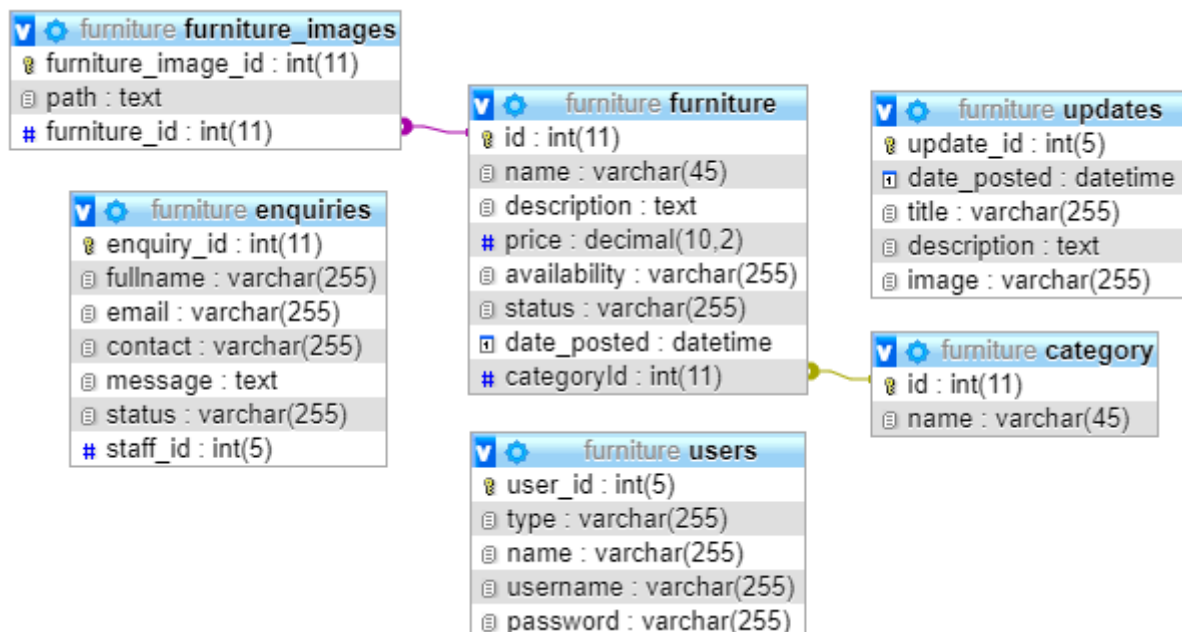


Figure 2. 2 - Tables and Their Columns

3. CHECKLIST OF COMPLETED FEATURES

The following table shows the list of features completed and any information related to them. The screenshots showing proof for each implemented functionality follows the table.

	Feature	Completion	Relevant Files	Notes
1	Copyright Notice	Yes	UserTemplate.php	The copyright notice has been changed to be dynamic and auto display the current year.
2	FAQs page	Yes	FAQ.php (controller), FAQcontent.php (view)	The FAQ page has been added to the users viewing area which displays the necessary message.
3	Furniture Categories	Yes	AdminManage Categories.php, AdminManage Furniture.php and other relevant view files for adding or editing a category or furniture	Categories can be added, edited or deleted from the admin area. The furniture can be assigned to categories. The categories are displayed in the user area. The users can select a category of products to view.
4	Hide Unavailable Products	Yes	AdminManage Furniture.php and other relevant view files for displaying furniture to the users	Products can be marked as available or unavailable from the staff area.
5	Product Condition	Yes	AdminManage Furniture.php and other relevant view files for displaying furniture to the users	Can add condition of product while adding or editing.
6	Users View Products According to Condition	Yes	AdminManage Furniture.php and other relevant view files for displaying furniture to the users	The user can then choose to see only desired kind of products. For example (Second Hand in the category Sofas)

7	Staff User Accounts	Yes	AdminManageUsers.php and relevant form to add/edit users.	The staff accounts can be created, edited or deleted from the admin area. The staff can only edit their own password while the super administrator can perform all different kinds of operations.
8	Upload Photos to Furniture	Yes	AdminManageFurniture.php and relevant view files to display furniture to the user	Furniture can have multiple images uploaded related to them. The uploaded images are stored in a folder and linked to the furniture. Different images can be uploaded while editing to replace the old images.
9	Updates to Home Page	Yes	AdminManageUpdates.php and relevant view files to display the updates to the user	Updates can be added from the administrator area. The updates can contain a photo as well. These updates are displayed in the home page.
10	Contact Page Enquiries	Yes	Contact.php, AdminManageMessages.php and relevant view files	The contact.php file displays a form to post an enquiry. As mentioned, the staff can access the enquiries and respond to them. The staff who responded to the enquiry is displayed.

Screenshots in Order:

```
<footer>  
  <!-- The updating copyright date. The current year will be displayed -->  
  &copy; Fran's Furniture <?php echo date("Y");?>  
</footer>
```

© Fran's Furniture 2019

Figure 3. 1 - Copyright Notice

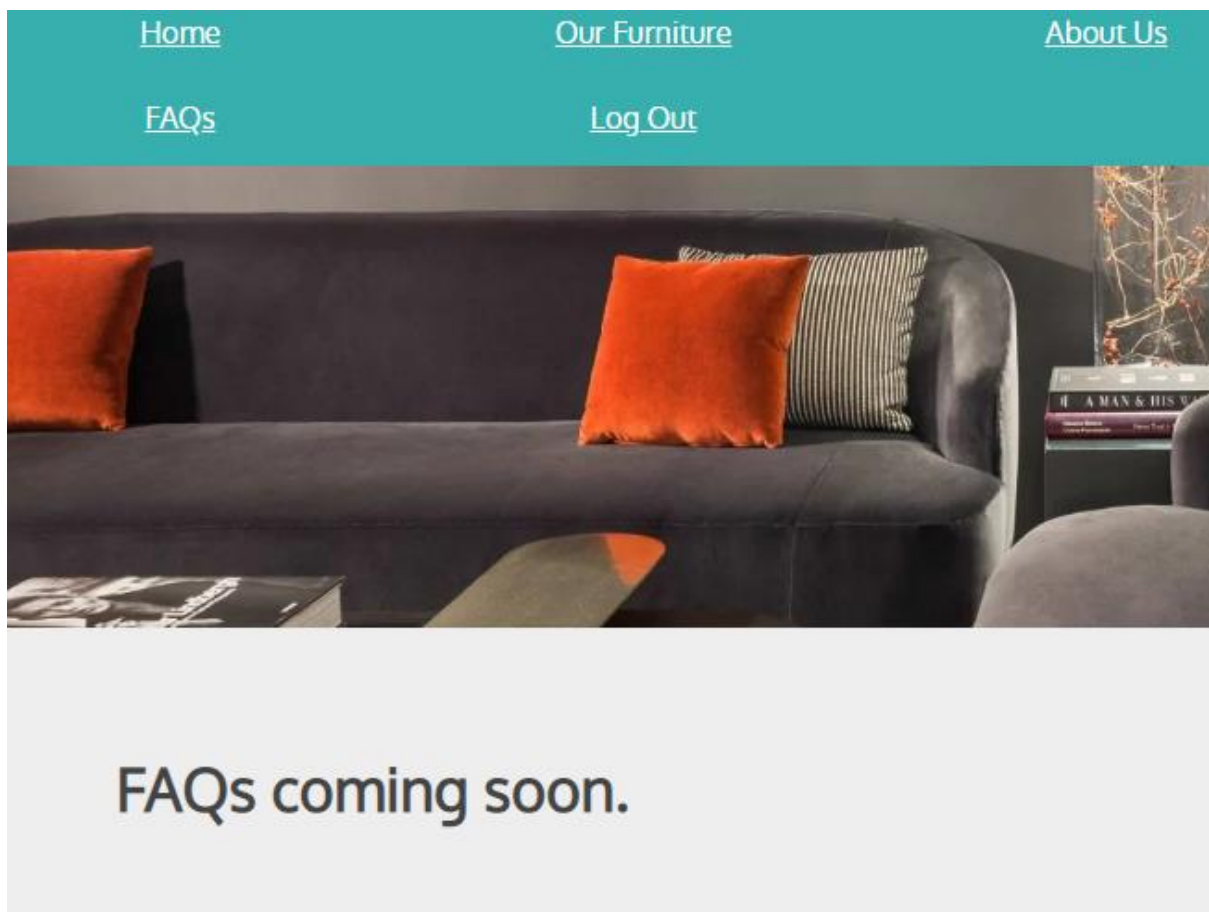



Figure 3. 2 - FAQs page

Manage Categories

	Category Name	Edit	Delete
1	Beds	Edit	Delete
2	Wardrobes	Edit	Delete
3	Sofas	Edit	Delete
4	Dining Tables	Edit	Delete

[Add new Category](#)

Sofas



Yellow Sofa

£899.00
Brand new yellow Sofa.

Item type: Brand New

Date Posted: 2019-02-24 03:31:15

All Furnitures

Beds

Wardrobes



Sofas

Dining Tables

Figure 3. 3 - Furniture Categories

Manage Furnitures

Successfully Changed Availability

	Furniture Name	Price	Availability	Status	Product Description	Images	Category	Edit	Delete	Date Posted
1	Charmcorner Sofas	£1099.00	Unavailable	Brand New	Prepare your senses for the inviting and indulging Charm range. A collection that includes a seat for every situation, it's also a range that gives you the power to create.		Sofas	Edit	Delete	2019-02-23 18:34:22
2	Four Seater Sofas	£999.00	Available	Brand New	Four seater sofas available in wide range of colours. These stay warm and are perfect for sitting in the winter.		Sofas	Edit	Delete	2019-02-23 18:35:44

Note: The Unavailable Products are not displayed in the user area.

Figure 3. 4 – Hide Unavailable Products

Furniture Status:

Furniture Description:

Brand New ▼
Brand New
Second Hand

Figure 3. 5 – Product Condition

Sofas

Filter: Second hand ▼



Second Hand Sofa

£399.00
Second hand sofa set for sale.

Item type: Second Hand

Date Posted: 2019-02-24 03:14:03

Figure 3. 6 – Users View Products According to Condition

Manage Users

	Name	Username	Edit	Delete	Edit Password
1	Administrator	admin	Edit	Delete	Edit Password
2	Anil Upreti	anilupreti	Edit	Delete	Edit Password
3	Ramesh Thapa	rameshthapa	Edit	Delete	Edit Password
4	Diwas Lamsal	diwaslamsal	Edit	Delete	Edit Password



[Add new user](#)

Figure 3. 7 – Staff User Accounts

	Furniture Name	Price	Availability	Status	Product Description	Images	Category
1	Charmcorner Sofas	£1099.00	Available	Brand New	Prepare your senses for the inviting and indulging Charm range. A collection that includes a seat for every situation, it's also a range that gives you the power to create.		Sofas
2	Four Seater Sofas	£999.00	Available	Brand New	Four seater sofas available in wide range of colours. These stay warm and are perfect for sitting in the winter.		Sofas
3	Soft Left Arm Sofa	£500.00	Available	Second Hand	This is a second hand unit but nothing like one. Get this sofa at an unbeatable price from our store.		Sofas
4	Wooden Wardrobe	£599.00	Available	Brand New	Brand new Wooden Wardrobe with three doors and drawers. Will fit in all your clothes and other wardrobe items as it is a big one.		Wardrobes

Figure 3. 8 – Upload Photos to Furniture

Manage Updates


	Title	Description	Image	Date Posted	Edit	Delete
1	First Update	Hello, This is Fran's furniture, I am testing the add update feature and this is the first update I am posting to this system.		2019-02-24 04:21:38	Edit	Delete
2	MEGA SALE	We are offering a mega sale in the coming month. The sale starts from 1st March 2019 and lasts upto 7th. We will be providing heavy discounts on products as much as 70%. You will get at least 25% discount on any product.		2019-02-24 04:32:19	Edit	Delete

[Add new update](#)

Updates

MEGA SALE

Date Posted: 2019-02-24 04:32:19



We are offering a mega sale in the coming month. The sale starts from 1st March 2019 and lasts upto 7th. We will be providing heavy discounts on products as much as 70%. You will get at least 25% discount on any product.

Figure 3. 9 – Updates to Home Page

4. IDENTIFICATION OF WEAKNESSES

The weaknesses in the provided system are discussed in this section in detail. These weaknesses are addressed and solved in the completed assignment. Also, other improvements that need to be made are mentioned.

4.1 Weaknesses and Solutions:

Repeated Code

The first and major difficulty faced was a lot of repeated code. If the system was to be designed the same way as provided, a single change would require changes to be made across all the pages. For example: fixing the path issue (which will be mentioned later) would require fixing it across every php file.

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="styles.css"/>
    <title>Fran's Furniture - Home</title>
  </head>
  <body>
    <header>
      <section>
        <aside>
          <h3>Opening Hours:</h3>
          <p>Mon-Fri: 09:00-17:30</p>
          <p>Sat: 09:00-17:00</p>
          <p>Sun: 10:00-16:00</p>
        </aside>
        <h1>Fran's Furniture</h1>
      </section>
    </header>
    <nav>
      <ul>
        <li><a href="">Home</a></li>
        <li><a href="furniture.php">Our Furniture</a></li>
        <li><a href="about.html">About Us</a></li>
        <li><a href="contact.php">Contact us</a></li>
      </ul>
    </nav>
  </body>
</html>
```

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="styles.css"/>
    <title>Fran's Furniture - Home</title>
  </head>
  <body>
    <header>
      <section>
        <aside>
          <h3>Opening Hours:</h3>
          <p>Mon-Fri: 09:00-17:30</p>
          <p>Sat: 09:00-17:00</p>
          <p>Sun: 10:00-16:00</p>
        </aside>
        <h1>Fran's Furniture</h1>
      </section>
    </header>
    <nav>
      <ul>
        <li><a href="">Home</a></li>
        <li><a href="furniture.php">Our Furniture</a></li>
        <li><a href="about.html">About Us</a></li>
        <li><a href="contact.php">Contact us</a></li>
      </ul>
    </nav>
  </body>
</html>
```

Figure 4. 1 – Example repeated code on about.php and contact.php

Solution:

A template could be used which holds all the repeating information where variables could be used to change the content dynamically. The following screenshot shows how the same template can be used to display multiple kind of information. The content variable would display the contents dynamically sent from different places according to user interactions. For example: the about page and contact page are both loaded from this same template, the only differences are the content and title.

```

<nav>
  <ul>
    <li><a href="/Assignment/public/">Home</a></li>
    <li><a href="/Assignment/public/UsersFurniture">Our Furniture</a></li>
    <li><a href="/Assignment/public/About">About Us</a></li>
    <li><a href="/Assignment/public/Contact">Contact us</a></li>
    <li><a href="/Assignment/public/FAQ">FAQs</a></li>
    <!-- Set the link to Login or Logout depending on whether the user is logged in -->
    <?php if(!isset($_SESSION['loggedin'])){?>
    <li><a href="/Assignment/public/Login">Login</a></li>
    <?php }
    else{?>
    <li><a href="/Assignment/public/Logout">Log Out</a></li>
    <?php } ?>
  </ul>

</nav>


<!-- The content obtained from views -->
<?php echo $content?>

```

Figure 4. 2 – Users Template

Broken Links:

When the provided files were first loaded into the server and executed, it appeared that the CSS was not working. The provided files had CSS path defined as “/styles.css”. This implies that the CSS file is located in the root directory while it is located inside the public directory. This also implies for other links such as navigation.

```

<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="/styles.css"/>
    <title>Fran's Furniture - Home</title>
  </head>
  <body>
    <header>
      <section>
        <aside>

```

Figure 4. 3 – Provided page CSS path definition

Solution:

In this case, simply removing the leading “/” would solve the problem, however, in the completed system, the file structure has been redesigned and this problem is addressed in a different way such that further amendments required very little changes and in a single place.

Static Date in Copyright

The date used in the copyright area was statically set and did not change with change in time. To solve this, the PHP date function was used to display the current date dynamically.

```
33
34
35     <footer>
36         &copy; Fran's Furniture 2016
37     </footer>
38 </body>
39 </html>
40
<footer>
    <!-- The updating copyright date. The current year will be displayed -->
    &copy; Fran's Furniture <?php echo date("Y");?>
</footer>
```

Figure 4. 4 – Static date issue solved

Separate PHP files for furniture categories:

Separate PHP files were used for different kinds of products. Although this could be implemented in a dynamic environment, by creating new php files for each category created, it is not a viable solution and reference to the database should be used for displaying products under each category. The files that come along with the brief include sofas.php, wardrobes.php and beds.php. These files contain the same information and statically set SQL queries.

Project	sofas.php	beds.php	wardrobes.php
public			
admin			
images			
about.html			
beds.php			
contact.php			
furniture.php			
index.php			
sofas.php			
styles.css			
wardrobes.php			

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <link rel="stylesheet" href="styles.css"/>
5     <title>Fran's Furniture - Sofas</title>
6 </head>
7 <body>
8 <header>
9     <section>
10        <aside>
11            <h3>Opening Hours:</h3>
12            <p>Mon-Fri: 09:00-17:30</p>
13            <p>Sat: 09:00-17:00</p>
14            <p>Sun: 10:00-16:00</p>
```

Figure 4. 5 – Static category pages

Solution:

The problem can be solved by creating a single page for displaying all the products according to the value passed for category. The controller can send appropriate category to the view for displaying products under it accordingly.

```
// Create instance of DatabaseTable for categories
$category = new DatabaseTable('category');
$allCategories = $category->findAll();
// Create instance of DatabaseTable for furniture
$furniture = new DatabaseTable('furniture');

/** The createFurnitureContent function
 * Sets up the view to display the furnitures
 */
public function createFurnitureContent($categoryId=""){
    require_once 'UsersFurniture/showFurnitureContent.php';
    $allCategories = $category->findAll();
    //Load furnitures template
    $fileName = '../templates/UsersFurnitureTemplate.php';
    $contents = [
        'allCategories'=>$allCategories,
        'content'=>$content
    ];
    $content = loadTemplate($fileName, $contents);
}
```

Figure 4. 6 – How categories are dynamically grabbed and sent accordingly

Repeating Database Connection Code:

The database connection code that was provided used a different set of hostname, username and password than the one being used locally. As such, changes needed to be made in every single page where database was being used.

```
<?php
$pdo = new PDO('mysql:dbname=furniture;host=127.0.0.1', 'student', 'student');
$furnitureQuery = $pdo->prepare('SELECT * FROM furniture WHERE categoryId = 5');

$furnitureQuery->execute();
```

Figure 4. 7 – Example database connection code in provided files

Solution:

A single file can be used for any connection required to the database. In the submitted solution, the database connection file is placed under the dbconnect directory and looks like this. This file is called from the main single point of entry index.php file.

```
/*
    The dbconnect.php file
    Connects and sets up the $pdo variable
*/

//Set up the variables to be used
$server = 'localhost';
$username = 'root';
$password = '';

//The name of the schema used in the assignment
$schema = 'furniture';

//The pdo variable is initialized
$pdo = new PDO('mysql:dbname=' . $schema . ';host=' . $server, $username, $password,
[PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION]);
```

Figure 4. 8 – Database connection file dbconnect.php

Use of static category IDs:

The provided files contained statically set category ids to find products. For example, as shown in the figure, it is not possible to know which category has 5 as its id. As such, it is difficult to know what the code is really trying to achieve.

```
<?php
$pdo = new PDO('mysql:dbname=furniture;host=127.0.0.1', 'student', 'student');
$furnitureQuery = $pdo->prepare('SELECT * FROM furniture WHERE categoryId = 5');

$furnitureQuery->execute();
```

Figure 4. 9 – SELECT query with category id provided as restriction

Solution:

The category id, when used, should be set dynamically according to the URL using GET variables and not to be used in such a manner as provided.

Hard Coded Database Queries:

The provided files contained hard coded database queries everywhere. This has been addressed in the submitted solution by having a separate Database class which performs almost database related functions. There is also a separate functions file called otherDatabaseFunctions.php which performs other database operations. It would be very easy for someone without any database knowledge to use the documented Database functions and implement them in the actual application.

```
/** Function findSorted
 * Self modified function to find data in a sorted order
 * @param field - The field used to extract data
 * @param value - The value in the field to be used as the criteria to extract data
 * @param key - The field to be used to sort the data
 * @param order - The order could be ASC or Left blank as DESC is the default
 * @return data - Return the selected rows
 */
function findSorted($field, $value, $key, $order = 'DESC') {
    global $pdo;
    $stmt = $pdo->prepare('SELECT * FROM '.$this->table.' WHERE '.$field.'=:value ORDER BY '.$key.' '.$order);
    $criteria = [
        'value' => $value
    ];
    $stmt->execute($criteria);
    return $stmt;
}

////////////////////////////////////

function findAll() {
    global $pdo;
    $stmt = $pdo->prepare('SELECT * FROM ' . $this->table);
    $stmt->execute();
    return $stmt;
}
```

Figure 4. 10 - Database class member functions

Accessing the Admin Area:

There is no way to access the admin area without manually changing the URL in the address bar. From accessibility point of view, this is not a user-friendly way to perform any kind of operation. The brief asks any such kind of problems to be non-existent in the submitted solution. As such, manually, a login link had to be added to the users area in order to solve this problem.

Repeated Login Code:

The same login code was used in every admin page. Because of this, any changes that needed to be made required changes in every file. The password is also statically set in every page as 'letmein', so, any changes required changing this across every page. The username is not asked and thus, allows only a single user account of sort to log in.

```
<?php
if (isset($_POST['submit'])) {
    if ($_POST['password'] == 'letmein') {
        $_SESSION['loggedin'] = true;
    }
}

if (isset($_SESSION['loggedin']) && $_SESSION['loggedin'] == true) {
?>

<section class="left">
    <ul>
        <li><a href="categories.php">Categories</a></li>
        <li><a href="furniture.php">Furniture</a></li>
    </ul>
</section>
```

No way to access categories created in Admin area:

There is no way to access the categories created in the Admin area from the customer's perspective. The only categories displayed are statically set Beds, Wardrobes and Sofas. The Wardrobes and Sofas category do not even exist in the database. This needs to be changed to handle changes dynamically as mentioned earlier.



Figure 4. 11 – Categories in admin and customer area

Separate forms for adding and editing:

A lot of code is repeated with add and edit operations. The same code that could have been used for adding or editing products and categories have been rewritten in separate pages. These add and edit files also contain a repeated template code.

```
<h2>Add Category</h2>

<form action="" method="POST">
  <label>Name</label>
  <input type="text" name="name" />

  <input type="submit" name="submit" value="Add Category" />

</form>

<form action="" method="POST">

  <input type="hidden" name="id" value="<?php echo $currentCategory['id']; ?>" />
  <label>Name</label>
  <input type="text" name="name" value="<?php echo $currentCategory['name']; ?>" />

  <input type="submit" name="submit" value="Save Category" />

</form>
```

Figure 4. 12 – Add and edit category

4.2 Other Solutions:

Those were the major problems faced on an attempt to rework the same provided system. Because of these persisting problems, the entire system has been redesigned from scratch, using any code from the provided system wherever possible and adding functionality to it. The system has been designed such that any further amendments can be made, and features can be added easily. The solutions that the submission contains that makes it considerably easier to make any changes are listed below.

Redefinition of the file folder structure:

The file folder structure has been reworked to follow an MVC pattern. There is a single-entry point inside the public directory as index.php. Every other page in the system is navigated to from this entry point. The .htaccess file rewrites the URL so that it appears more readable and links can be created in such a way that the succeeding URL subdirectories in the public directory would call class names and other succeeding subdirectories would call the class methods.

















 controllers	3/26/2019 6:57 PM	File folder	
 core	3/26/2019 8:09 AM	File folder	
 dbconnect	3/26/2019 8:09 AM	File folder	
 functions	3/27/2019 11:04 AM	File folder	
 models	3/26/2019 8:09 AM	File folder	
 public	3/26/2019 8:09 AM	File folder	
 report	3/26/2019 3:20 PM	File folder	
 templates	3/26/2019 8:09 AM	File folder	
 tests	3/27/2019 10:56 AM	File folder	
 views	3/26/2019 8:09 AM	File folder	
 .htaccess	3/26/2019 8:09 AM	HTACCESS File	1 KB
 codeCoverageGenerate.txt	3/26/2019 3:25 PM	Notepad++ Docu...	1 KB
 index.php	3/26/2019 8:09 AM	PHP File	1 KB
 loader.php	3/26/2019 8:09 AM	PHP File	1 KB
 phpunit.xml	3/26/2019 3:22 PM	XML File	1 KB
 README.md	3/26/2019 8:09 AM	MD File	1 KB

Figure 4. 13 – File Folder Structure

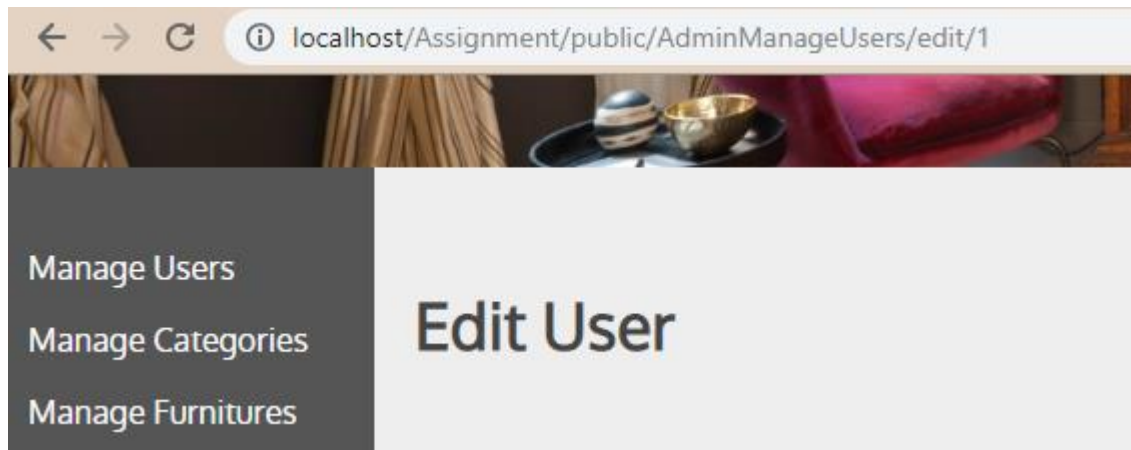


Figure 4. 14 – Example MVC method

For example: in the provided image, the link ‘public/AdminManageUsers/edit’ is calling the edit function of AdminManageUsers class. The edit function has required code to call necessary database operations and display the view to the user.

Separation of Model, View and Controllers:

As seen in the file folder structure screenshot, there are separate folders for model, views and controllers. The model contains only the database class and all the database related operations can be performed simply by creating instances of this class. Also mentioned, the controllers are classes that appear in the URL (as in the xample of AdminManageUsers). These controllers set up the view to display contents to the user. The view folder contains necessary files for displaying data to the user.

Separate Template file:

The template files contain all the common code that is same across the different pages. The only content that changes dynamically is the content and the title. The controller loads all the necessary elements into the content variable and loads it to the template using a loadTemplate function.

Separate Database Connection file:

The database connection problem needing to be edited manually for each page has been addressed by using a separate dbconnect.php file which contains the code to connect to the database. This php file has been called from the entry point index.php file which loads the database connection code for each and every page.

5. TESTING

The tests involve manually working on the expected and actual outcomes from different kind of interactions and automated Unit test provided by PHP Unit. The test logs display what aspects of the program have been tested and the result. The unit test report is displayed as screenshots along with any relevant test codes.

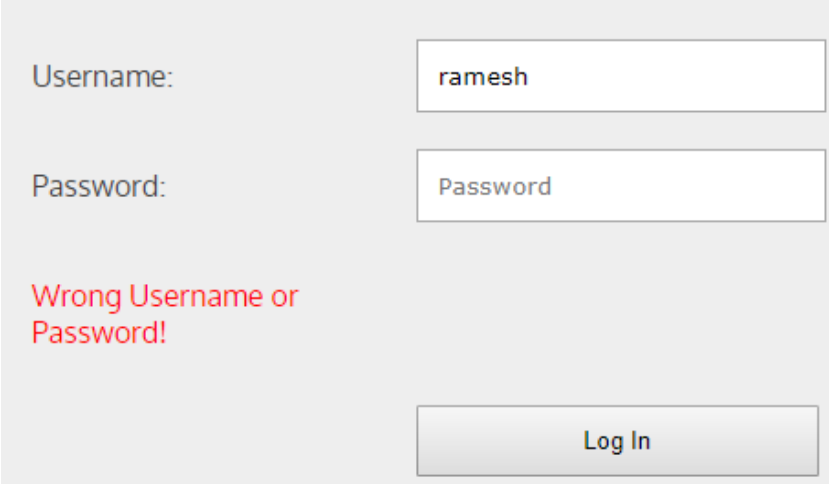
5.1 Manual Testing (Black Box Approach)

5.1.1 Staff Perspective Tests

Any relevant tests for the users that are staff and have access to log in to the system.

Table 5.1 – Login Tests

	Test Case	Steps to Test	Expected Outcome	Actual Outcome
1.	Staff Login using correct credentials	<ul style="list-style-type: none">Staff goes to the login pageStaff enters correct details and clicks login button.	The staff is taken to the administrator dashboard area.	Same as expected.
2.	Staff Login using incorrect credentials.	<ul style="list-style-type: none">Staff goes to the login pageStaff enters incorrect details and clicks login button.	Invalid username or password message is displayed.	Same as expected.



The screenshot shows a login interface with a light gray background. It contains two input fields: 'Username:' with the text 'ramesh' and 'Password:' with the placeholder text 'Password'. Below these fields, a red error message reads 'Wrong Username or Password!'. At the bottom right, there is a 'Log In' button.

Figure 5. 1 – Invalid Login

Table 5.2 – Manage User Tests

	Test Case	Steps to Test	Expected Outcome	Actual Outcome
1.	Add new User by filling all the details correctly	<ul style="list-style-type: none">• Admin goes to the Manage Users area• Admin clicks Add new User button.• Admin fills up all the form correctly• Admin submits the form	The staff is taken to the manage users area showing a success message and the new user is displayed in the list	Same as expected.
2.	Add new User attempt using existing username	<ul style="list-style-type: none">• Admin tries to add new user but with an existing username	Username already exists message is displayed	Same as expected.
3.	Add new User attempt using different values for password and confirm password	<ul style="list-style-type: none">• Admin tries to add new user but with different values for password and confirm password	Passwords do not match message is displayed	Same as expected.
4.	Edit Password attempt by user (valid old password)	<ul style="list-style-type: none">• Staff goes to their edit password area• Staff enters correct old password• Staff enters valid new password and submits the form	The staff's password is changed, and a message is displayed saying so	Same as expected
5.	Edit Password attempt by user (invalid old password)	<ul style="list-style-type: none">• Staff tries to edit the password but with an incorrect old password value	An error message saying incorrect old password is displayed	Same as expected
6.	Staff attempt to delete or edit another user	<ul style="list-style-type: none">• Staff tries to delete or edit another user	The options are not available for staff except the super administrator for	Same as expected

			editing or deleting other user accounts	
7.	Administrator attempt to delete another user	<ul style="list-style-type: none"> Administrator goes to the users area Administrator clicks the delete button 	The user is deleted from the system and a message is displayed saying successfully deleted	Same as expected
8.	Administrator attempt to edit another user	<ul style="list-style-type: none"> Administrator clicks the edit button Administrator changes the user's username or full name and submits the form 	The staff's details are changed	Same as expected

Name:

Username:

Password:

Confirm Password:

	Name	Username	Edit	
1	Administrator	admin	Edit	Delete
2	Anil Upreti	anilupreti	Edit	Delete
3	Ramesh Thapa	rameshthapa	Edit	Delete
4	Diwas Lamsal	diwaslamsal	Edit	Delete
5	Kanchan Upreti	kupreti	Edit	Delete

Figure 5. 2 – Add new User

Name:

Username:

Password:

Confirm Password:

Name:

Username:

Password:

Confirm Password:

Username Already Exists.

Figure 5. 3 – Use existing username

Manage Users					
	Name	Username	Edit	Delete	Edit Password
1	Administrator	admin	No Access	No Access	No Access
2	Anil Upreti	anilupreti	No Access	No Access	No Access
3	Ramesh Thapa	rameshthapa	No Access	No Access	No Access
4	Diwas Lamsal	diwaslamsal	No Access	No Access	Edit Password
5	Kanchan Upreti	kupreti	No Access	No Access	No Access

Figure 5. 4 - Staff attempt to delete or edit another user

Manage Users					
Successfully Deleted User					
	Name	Username	Edit	Delete	Edit Password
1	Administrator	admin	Edit	Delete	Edit Password
2	Anil Upreti	anilupreti	Edit	Delete	Edit Password
3	Ramesh Thapa	rameshthapa	Edit	Delete	Edit Password
4	Diwas Lamsal	diwaslamsal	Edit	Delete	Edit Password
Add new user					

Figure 5. 5 – Admin Delete User

Manage Users					
Successfully Edited User					
	Name	Username	Edit	Delete	Edit Password
1	Administrator	admin	Edit	Delete	Edit Password
2	Anil Upreti	anilupreti	Edit	Delete	Edit Password
3	Ramesh Thapa	<u>rthapa123</u>	Edit	Delete	Edit Password
4	Diwas Lamsal	diwaslamsal	Edit	Delete	Edit Password
Add new user					

Figure 5. 6 – Admin Edit User

Table 5.3 – Manage Categories Tests

	Test Case	Steps to Test	Expected Outcome	Actual Outcome
1.	Create new Category with valid name	<ul style="list-style-type: none">• Staff goes to the Manage Categories area• Staff clicks the add new category button• Staff enters a valid category name and submits	Successfully added category message is displayed, also showing the updated list of categories	Same as expected.
2.	Create new Category with Empty Name	<ul style="list-style-type: none">• Staff tries to add new category with empty name	Error message is displayed	Same as expected.
3.	Create new Category with Existing Category's Name	<ul style="list-style-type: none">• Staff tries to add new category with duplicate name	Error message is displayed	Same as expected.
4.	Delete empty Category	<ul style="list-style-type: none">• Staff tries to delete a category which has no products under it	Successfully deleted category message is displayed	Same as expected
5.	Delete Category with Furniture under them	<ul style="list-style-type: none">• Staff tries to delete a category which has products under it	Error message is displayed	Same as expected

Enter Category Name:

Cannot have empty name.

Submit

Enter Category Name:

Category Name Already Exists.

Submit

Figure 5. 7 – Category Errors

Enter Category Name:

Submit

Successfully Added Category

	Category Name	Edit	Delete
1	Beds	Edit	Delete
2	Wardrobes	Edit	Delete
3	Sofas	Edit	Delete
4	Dining Tables	Edit	Delete
5	Valid Category	Edit	Delete

Add new Category

Figure 5. 8 – Add new Category (valid)

Cannot Delete Category With Products			
	Category Name	Edit	Delete
1	Beds	Edit	Delete
2	Wardrobes	Edit	Delete
3	Sofas	Edit	Delete
4	Dining Tables	Edit	Delete
5	Valid Category	Edit	Delete

Successfully Deleted Category			
	Category Name	Edit	Delete
1	Beds	Edit	Delete
2	Wardrobes	Edit	Delete
3	Sofas	Edit	Delete
4	Dining Tables	Edit	Delete

Figure 5. 9 – Delete Category attempts Unsuccessful and Successful

Table 5.4 – Manage Furniture Tests

	Test Case	Steps to Test	Expected Outcome	Actual Outcome
1.	Add new Furniture with Valid Details	<ul style="list-style-type: none"> Staff goes to the Manage Furniture area Staff clicks the add new furniture button Staff enters valid furniture details and submits the form 	Successfully added furniture message is displayed, also showing the updated list of furniture	Same as expected.
2.	Add new Furniture with some empty fields	<ul style="list-style-type: none"> Staff tries to add new furniture with one or more empty fields 	Error message is displayed (same message as in category or user test)	Same as expected.
3.	Add images to the furniture	<ul style="list-style-type: none"> Staff selects images to add when filling the form to add new furniture 	Images are added for the furniture and displayed in the furniture display list	Same as expected.

4.	Change availability of furniture	<ul style="list-style-type: none"> Staff clicks the Available or Unavailable button for a furniture 	The availability is changed for the product and a message is displayed	Same as expected
5.	Edit Furniture (without providing images)	<ul style="list-style-type: none"> Staff updates details about the furniture and leaves the image upload file chooses empty 	The furniture details are updated, and the previous images are displayed for the furniture.	Same as expected
6.	Edit Furniture (also providing images)	<ul style="list-style-type: none"> Staff updates details about the furniture and selects images to upload for the furniture 	The old images are removed, and the latest uploaded images are displayed for the furniture	Same as expected

Note: There is a separate test section to test how the furniture and other updates are displayed from the customer's perspectives.

Furniture Name:

Furniture Price:

Furniture Status:




Furniture Description:

This is a test for new Furniture

Furniture Category:

Select Multiple Images: 2 files

Note*: Images larger than 2MB will not be uploaded and only valid image file types will be uploaded.

14	Wardrobe	£299.00	Available	Second Hand	Second hand wardrobes for sale at just £299.		Wardrobes	Edit	Delete	2019-02-24 04:04:22
15	Test New Furniture	£1200.00	Available	Second Hand	This is a test for new Furniture	 	Sofas	Edit	Delete	2019-03-27 12:29:29

[Add new Furniture](#)

Figure 5. 10 – Add new Furniture





Manage Furnitures										
Successfully Changed Availability										
	Furniture Name	Price	Availability	Status	Product Description	Images	Category	Edit	Delete	Date Posted
1	Charmcorner Sofas	£1099.00	Available	Brand New	Prepare your senses for the inviting and indulging Charm range. A collection that includes a seat for every situation, it's also a range that gives you	 	Sofas	Edit	Delete	2019-02-23 18:34:22
15	Test New Furniture	£1200.00	Unavailable	Second Hand	This is a test for new Furniture	 	Sofas	Edit	Delete	2019-03-27 12:29:29

Figure 5. 11 – Furniture Availability

Table 5.4 – Manage Enquiries Tests

	Test Case	Steps to Test	Expected Outcome	Actual Outcome
1.	Set Enquiry to Pending	<ul style="list-style-type: none"> Staff goes to the Manage Enquiries area Staff clicks the pending button for an enquiry 	Enquiry status is changed to pending and a message is displayed. The staff associated is changed to the one who caused the change.	Same as expected.
2.	Set Enquiry to Completed	<ul style="list-style-type: none"> Staff clicks the completed button for an enquiry. 	Enquiry status is changed to completed and a message is displayed. The staff associated is changed to the one who caused the change.	Same as expected.
3.	Delete Enquiry	<ul style="list-style-type: none"> Staff deletes an enquiry. 	The enquiry is deleted, and a message is displayed.	Same as expected.

Manage Enquiries							
Enquiry Status Changed To Pending							
	Customer Name	Email	Phone	Enquiry	Status	Manage	Staff Associated
1	Diwas Lamsal	diwaslamsalturbo@gmail.com	+9779808066424	asdasdasdasdasdasdasdasdasd asdasdasdasdasdasdasdasdasd	Pending	Pending Completed Delete	Administrator
2	Ramesh Thakur	rameshthapa@example.com	123456	ramesh thakur wants wardrobe ramesh thakur wants wardrobe ramesh thakur wants wardrobe	Completed	Pending Completed Delete	Administrator
3	Arjun Upreti	arjun@upreti.thapa	123456789	Hello I am arjun upreti I want to know about the kinds of wardrobes that you have.	Pending	Pending Completed Delete	Diwas Lamsal

Figure 5. 12 – Enquiry State Pending

Manage Enquiries

Enquiry Status Changed To Completed

	Customer Name	Email	Phone	Enquiry	Status	Manage	Staff Associated
1	Diwas Lamsal	diwaslamsalturbo@gmail.com	+9779808066424	asdasdasdasdasdasdasdasdasd asdasdasdasdasdasdasdasdasd	Completed	Pending Completed Delete	Diwas Lamsal
2	Ramesh Thakur	rameshthapa@example.com	123456	ramesh thakur wants wardrobe ramesh thakur wants wardrobe ramesh thakur wants wardrobe	Completed	Pending Completed Delete	Administrator
3	Arjun Upreti	arjun@upreti.thapa	123456789	Hello I am arjun upreti I want to know about the kinds of wardrobes that you have.	Completed	Pending Completed Delete	Diwas Lamsal

Figure 5. 13 – Enquiry State Completed

Table 5.4 – Manage Updates Tests

	Test Case	Steps to Test	Expected Outcome	Actual Outcome
1.	Add new Update with Valid Details	<ul style="list-style-type: none"> Staff goes to the Manage Updates area Staff clicks the add new update button Staff enters valid details and submits the form 	Successfully added update message is displayed, also showing the updated list of updates	Same as expected.
2.	Add new Update with some empty fields	<ul style="list-style-type: none"> Staff tries to add new update with one or more empty fields 	Error message is displayed (same message as in category or user test)	Same as expected.
3.	Add image to the update	<ul style="list-style-type: none"> Staff selects image to add when filling the form to add new update 	Image is added for the update and displayed in the update display list	Same as expected.
4.	Delete update	<ul style="list-style-type: none"> Staff clicks the delete button for an update 	The update is deleted, and a message is displayed.	Same as expected

Add Update

Title:

Description:

This is my new Update its a test

Image: 2476826...8_7.jpg





2	SALE	7th. We will be providing heavy discounts on products as much as 70%. You will get at least 25% discount on any product.		02-24 04:32:19	Edit	Delete
3	My New Update	This is my new Update its a test		2019-03-27 13:56:43	Edit	Delete

Figure 5. 14 – Add Update

Manage Updates

Succesfully Deleted Update

	Title	Description	Image	Date Posted	Edit	Delete
1	First Update	Hello. This is Fran's furniture. I am testing the add update feature and this is the first update I am posting to this system.		2019-02-24 04:21:38	Edit	Delete
2	MEGA SALE	We are offering a mega sale in the coming month. The sale starts from 1st March 2019 and lasts upto 7th. We will be providing heavy discounts on products as much as 70%. You will get at least 25% discount on any product.		2019-02-24 04:32:19	Edit	Delete

[Add new update](#)

Figure 5. 15 – Delete Update

5.1.2 Customer Perspective Tests

Any relevant tests for the users that are customers and do not have access to log in to the system.

Table 5.1 – Navigation Tests

	Test Case	Steps to Test	Expected Outcome	Actual Outcome
1.	Home Page (Updates)	<ul style="list-style-type: none">User reaches the system home page	The updates posted in the staff area are displayed along with their information such as when the update was posted.	Same as expected.
2.	About Us Page	<ul style="list-style-type: none">User navigates to the About Us page from the home page	The provided about us content is displayed.	Same as expected.
3.	FAQs Page	<ul style="list-style-type: none">User navigates to the FAQs page from the home page	The FAQs coming soon message is displayed.	Same as expected
4.	Contact Us page	<ul style="list-style-type: none">User navigates to the Contact Us page from the home page	A form to post an enquiry is displayed.	Same as expected
5.	Our Furniture Page	<ul style="list-style-type: none">User navigates to the Our Furniture page from the home page	The furniture added to the system along with their images are displayed. Also displays the list of available categories to choose from, the search bar and the selector to select the type of furniture	Same as expected

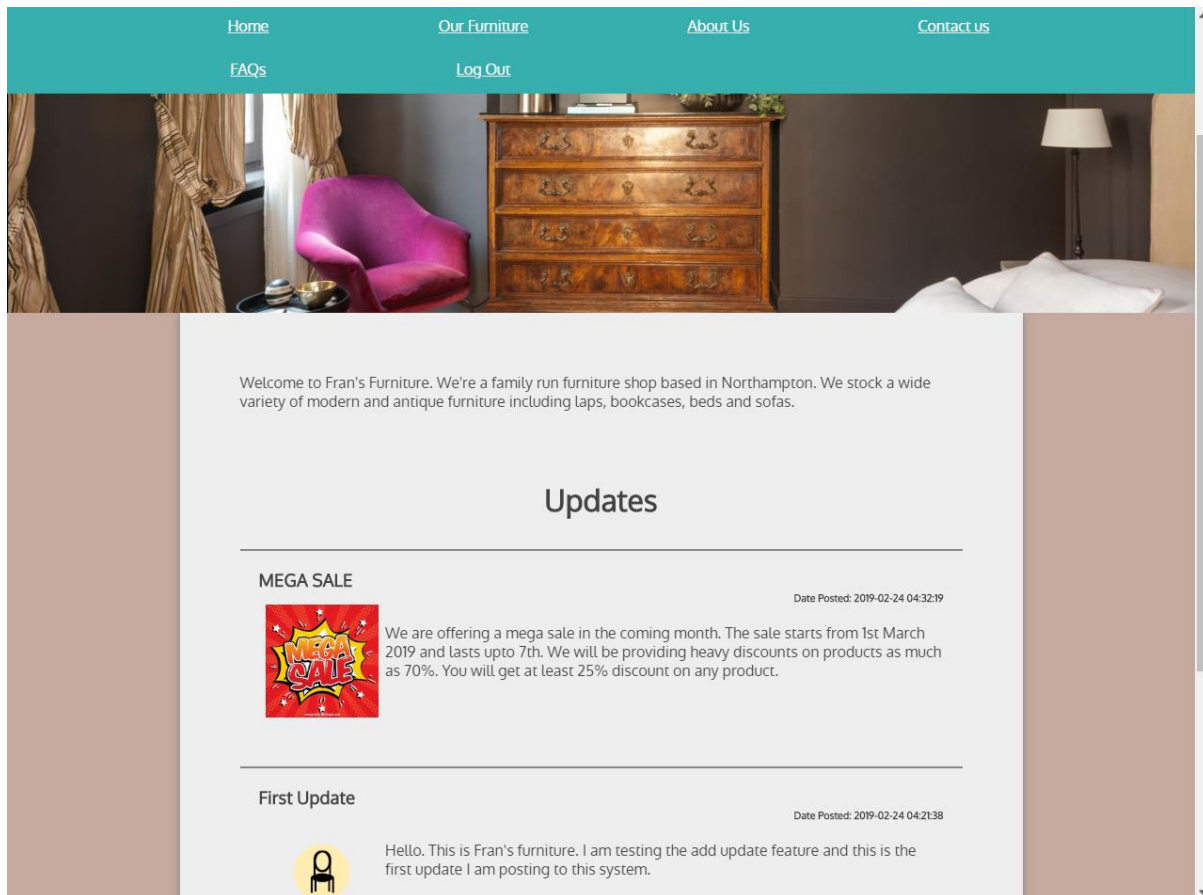


Figure 5. 16 – Home Page

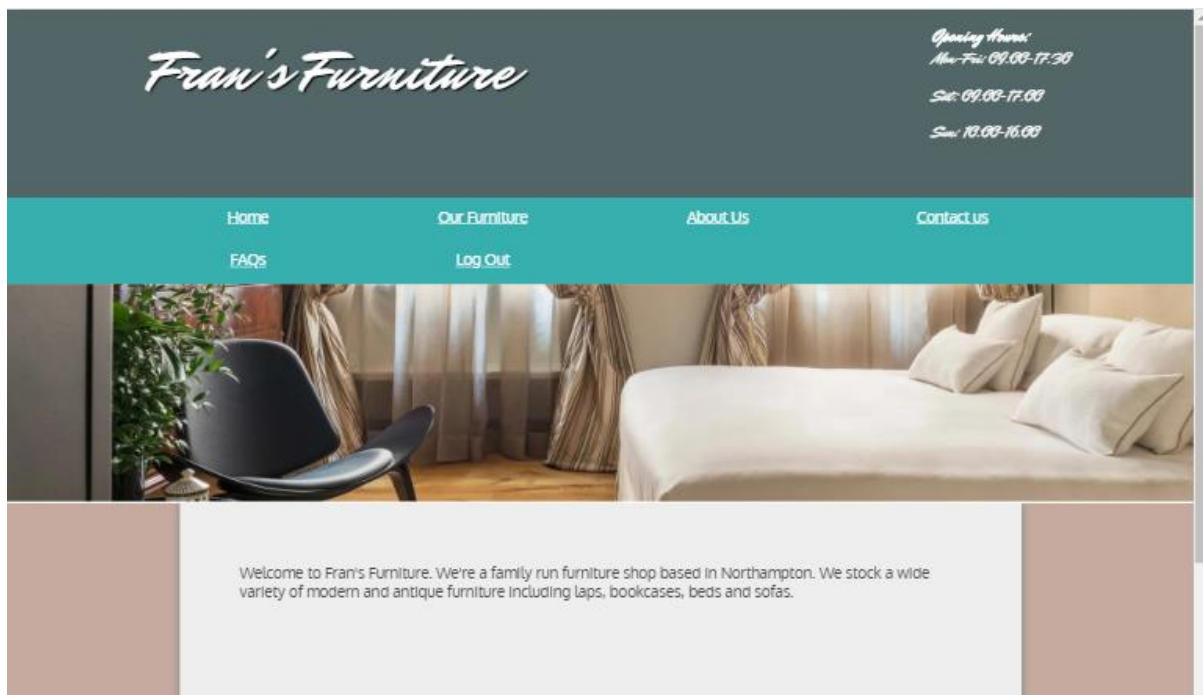


Figure 5. 17 – About Us Page

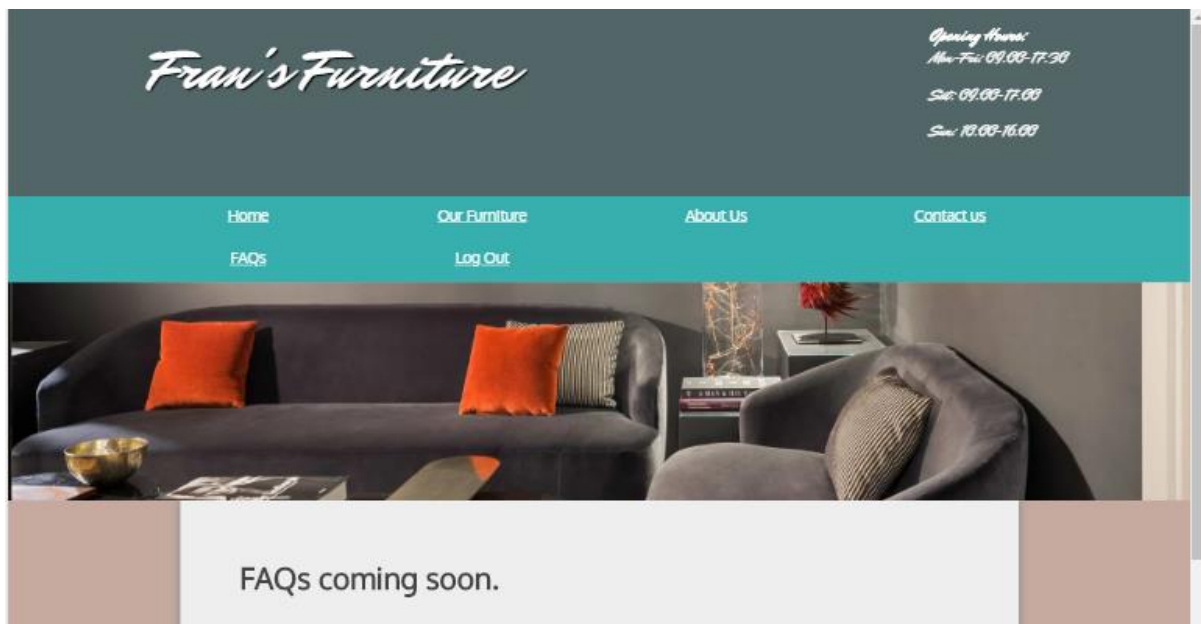


Figure 5. 18 – FAQs Page

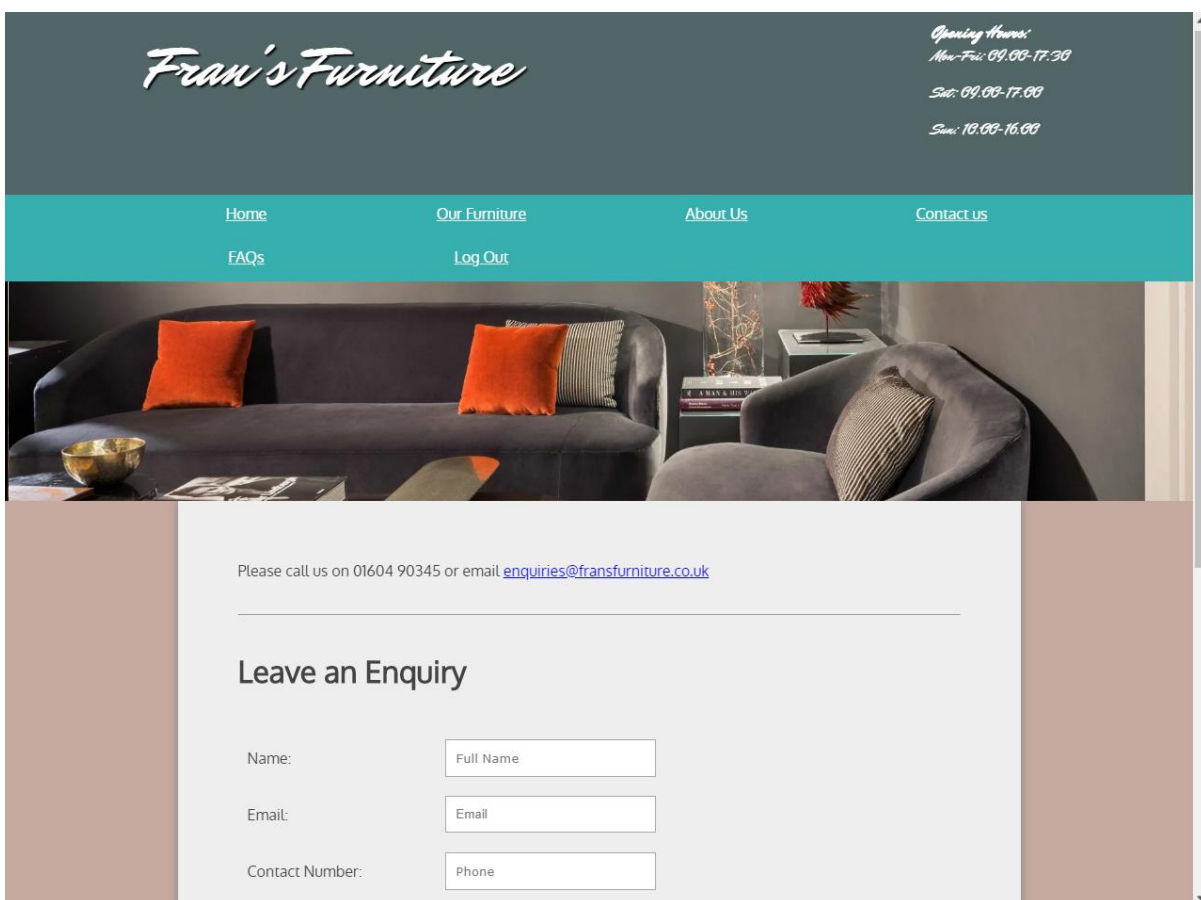


Figure 5. 19 – Contact Page

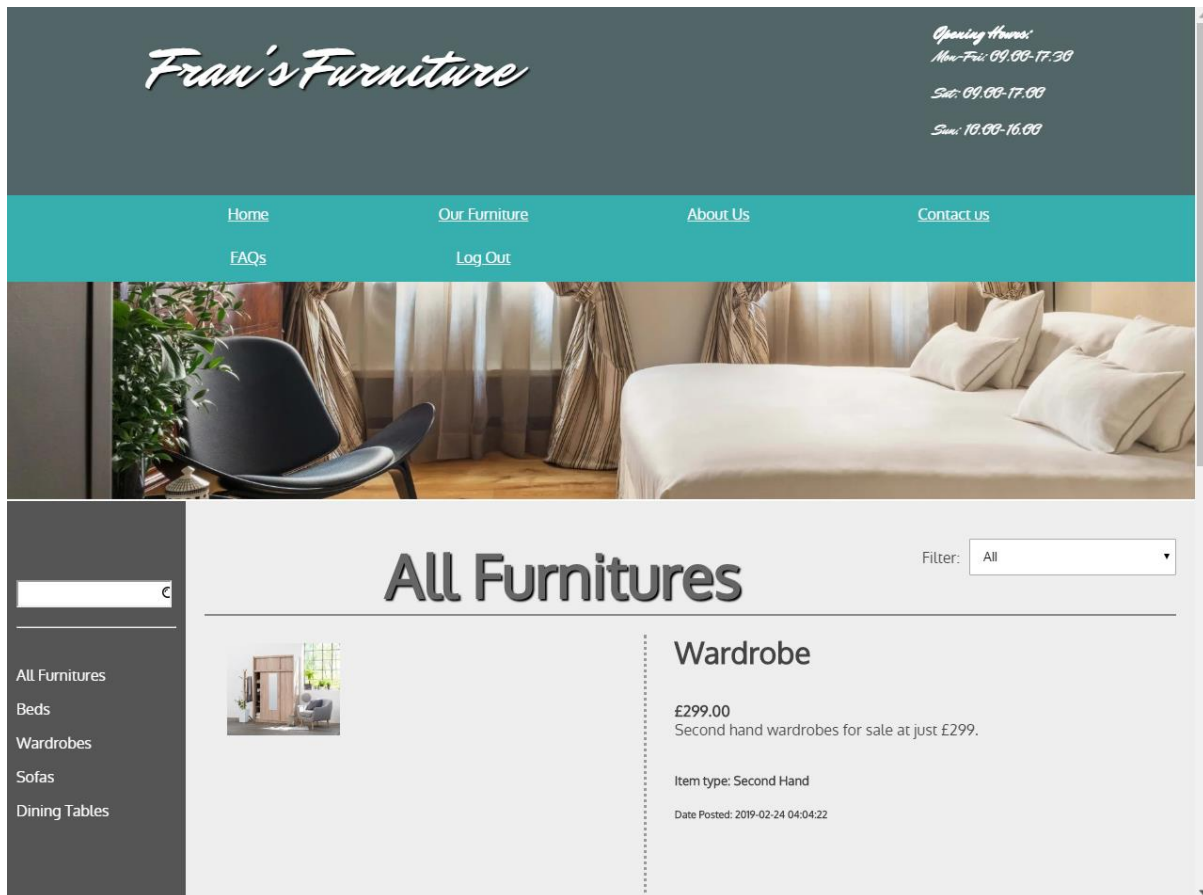


Figure 5. 20 – Our Furniture Page

Table 5.1 – Customer Post Enquiry Test

	Test Case	Steps to Test	Expected Outcome	Actual Outcome
1.	Add Enquiry (valid)	<ul style="list-style-type: none"> User navigates to the contact page User fills up the details in the form and submits 	A message is displayed. The enquiry can be managed from the staff area	Same as expected.
2.	Add Enquiry (invalid)	<ul style="list-style-type: none"> User fills up the details in the form partially and submits 	An error message is displayed	Same as expected.

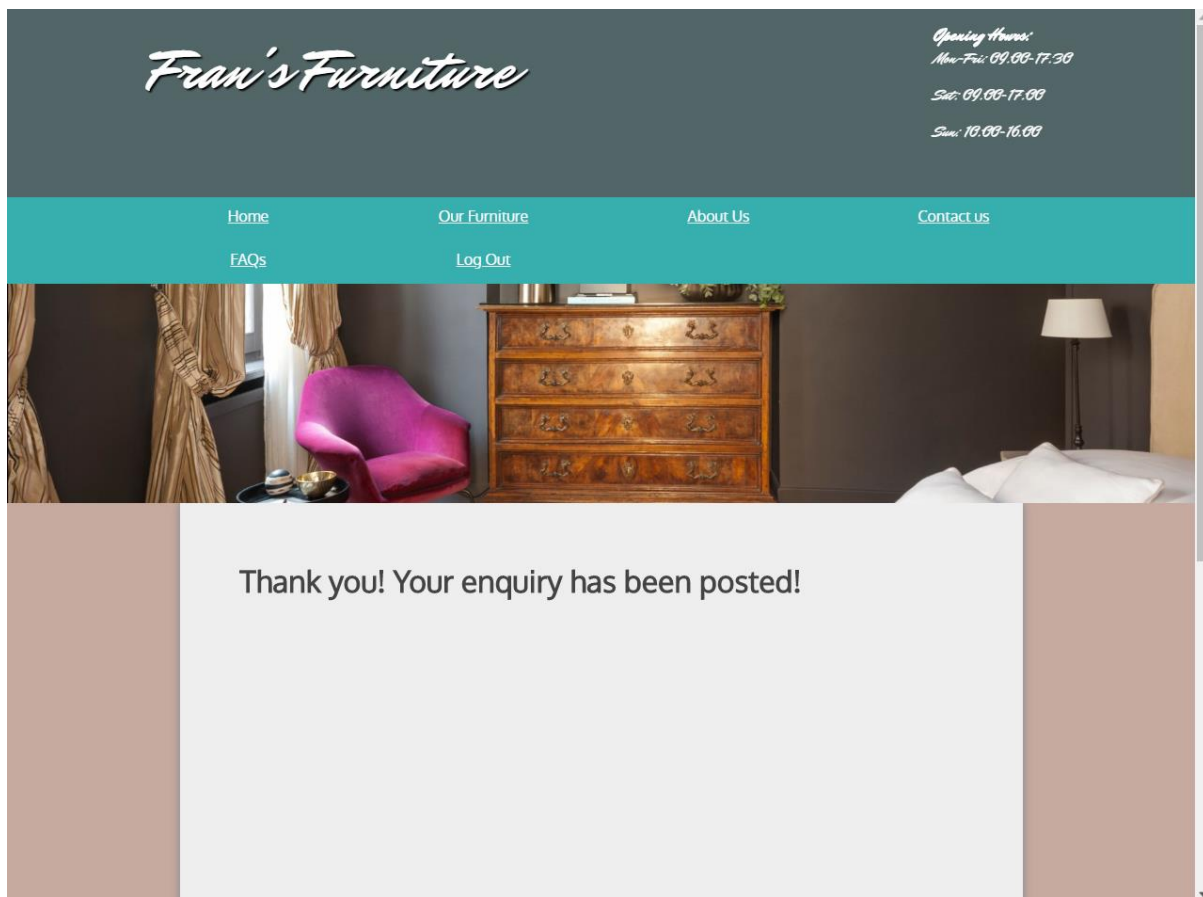


Figure 5. 21 – Valid Enquiry Submission

The screenshot shows the 'Leave an Enquiry' form on the website. The form is titled 'Leave an Enquiry' and contains the following fields: 'Name' (filled with 'Ram Bahadur'), 'Email' (filled with 'Email'), 'Contact Number' (filled with 'Phone'), and 'Message' (filled with 'Hello this is a test enquir'). Below the form, a red error message 'Cannot have empty fields' is displayed. A 'Submit' button is located at the bottom of the form.

Figure 5. 22 – Invalid Enquiry Submission

Table 5.1 – Customer Furniture Test

	Test Case	Steps to Test	Expected Outcome	Actual Outcome
1.	View All Furniture	<ul style="list-style-type: none"> User navigates to the Our Furniture section 	All the furniture is displayed, with use of pagination, a maximum of 3 products are displayed per page.	Same as expected.
2.	View Second Hand Furniture	<ul style="list-style-type: none"> User selects the second-hand option from the filter selection 	All the second-hand products are displayed but the pagination is broken.	Same as expected. (weakness – pagination breaks on doing so)
3.	View all Beds	<ul style="list-style-type: none"> User selects the Beds category 	All the products under Beds category are displayed in pagination.	Same as expected.
4.	View Second Hand Beds	<ul style="list-style-type: none"> User selects the beds category then also selects the second-hand option 	All the second-hand beds are displayed but the pagination is broken.	Same as expected.
5.	Search for a product	<ul style="list-style-type: none"> User types in a name for product 	The product related to the search term.	Same as expected.
6.	Search inside a category	<ul style="list-style-type: none"> User selects a category then types in a search term 	The product within the category and related to the search term is displayed.	Same as expected.
7.	Navigate to other pages in pagination	<ul style="list-style-type: none"> User clicks on other numbered pages in the pagination 	The products according to their position in the queue are displayed in respective pages	Same as expected.

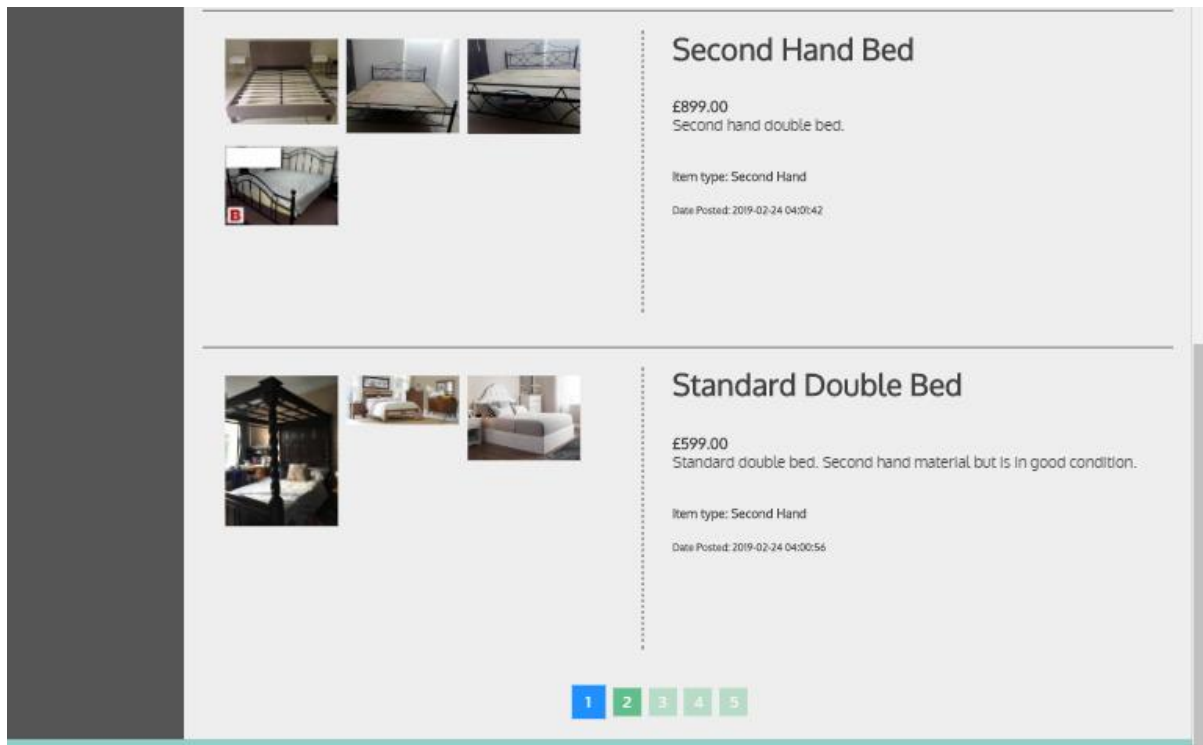


Figure 5. 23 – View All Furniture with Pagination

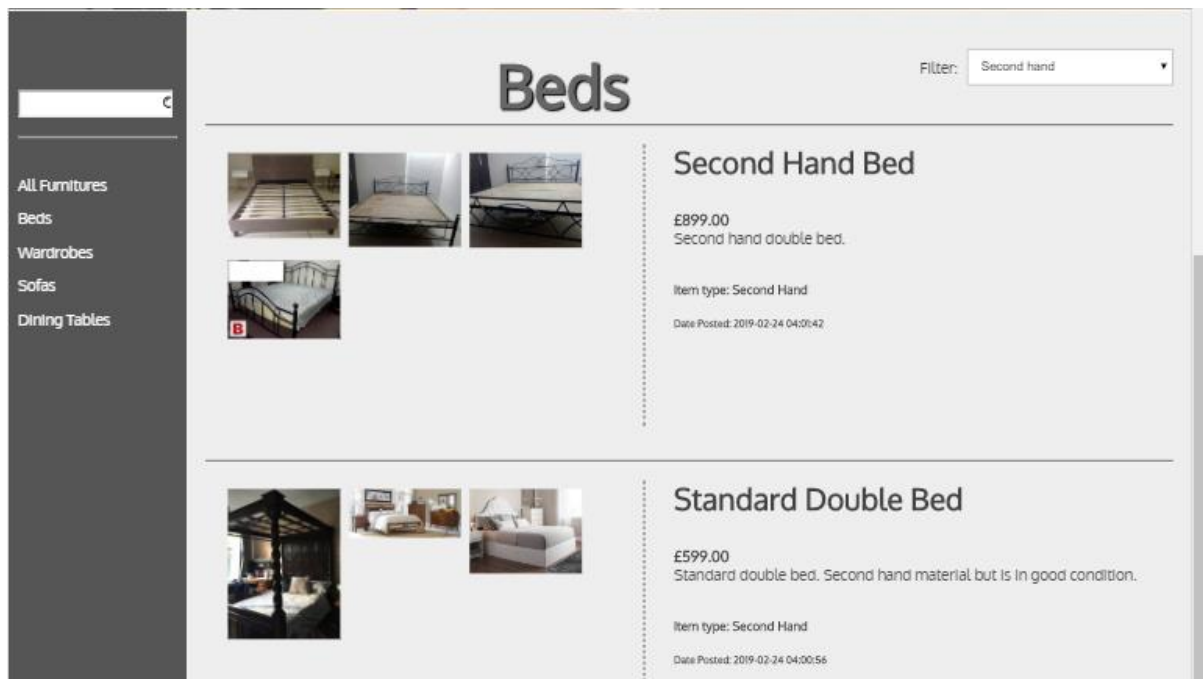


Figure 5. 24 – View Second Hand Beds

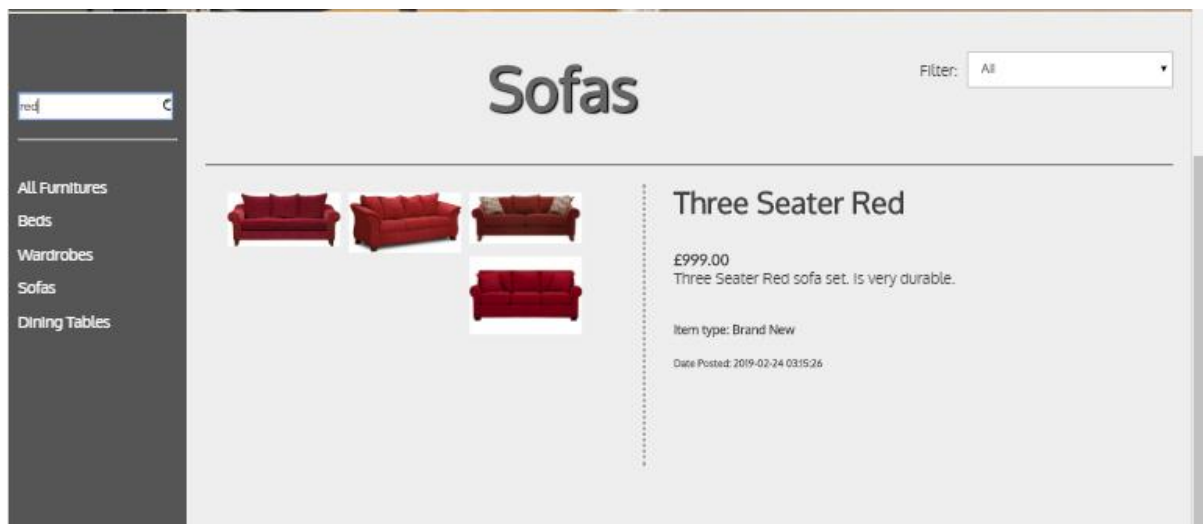


Figure 5. 25 – Search Term Red inside Category Sofas

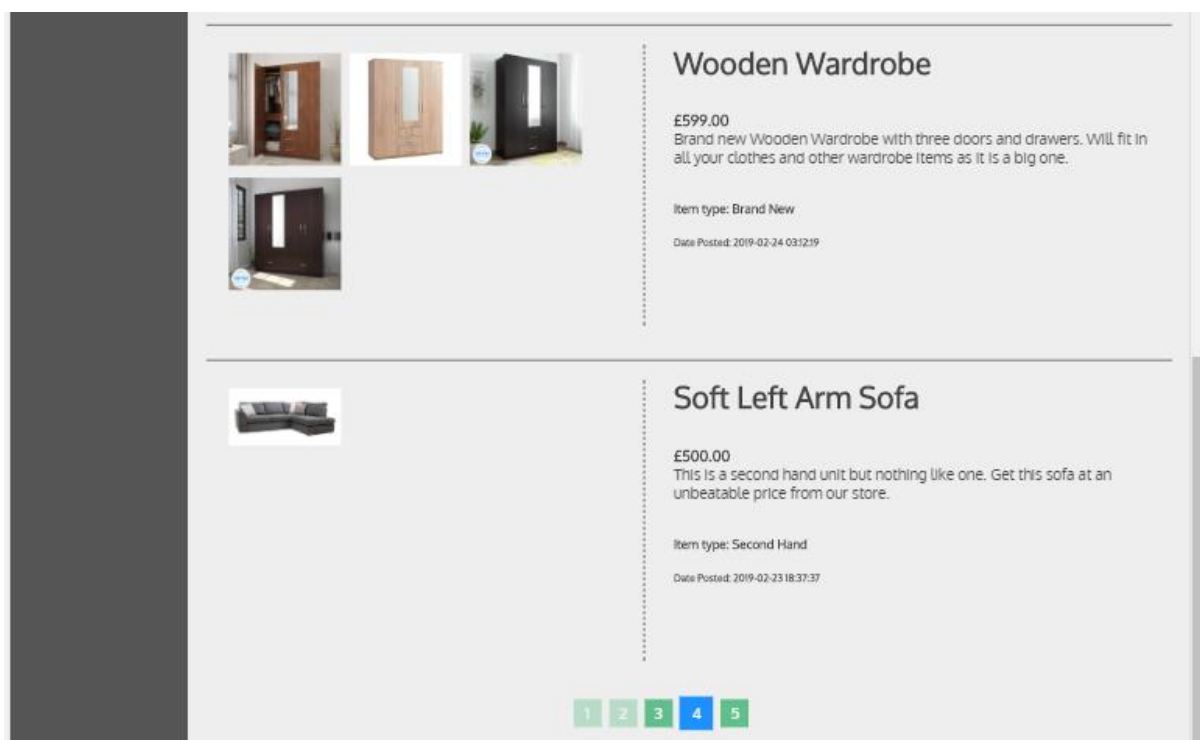


Figure 5. 26 – Pagination Fourth Page Test

5.1.3 Test Report

Weaknesses/Bugs Found: 1

Test Case: Filter Second-Hand or Brand-New products

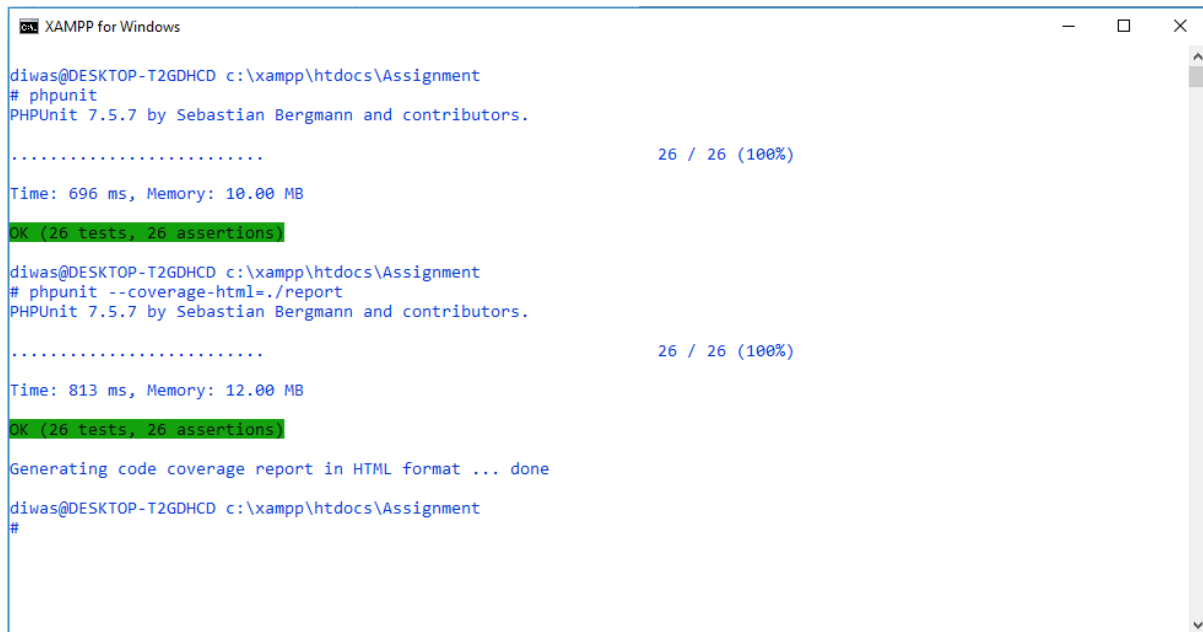
Bug/Weakness: When the user chooses a filter from the selection menu, the pagination is removed, and all the products are displayed in the page.

Proposed Solution: The pagination should be integrated to work with the filter. This can be done in the controller where if the filter is set, the products should be displayed in the pagination accordingly.

Any other bugs/weaknesses and their solutions are broken down in the evaluation section.

5.2 PHPUnit Testing (White Box Approach)

PHPUnit test has been performed to test data validation functions and some other functions written in the submission. These might include empty field submissions, or database checking functions. All the aspects of the tested functions have been tested getting a code coverage of 100%.



```
XAMPP for Windows

diwas@DESKTOP-T2GDHCD c:\xampp\htdocs\Assignment
# phpunit
PHPUnit 7.5.7 by Sebastian Bergmann and contributors.

.....                                     26 / 26 (100%)

Time: 696 ms, Memory: 10.00 MB
OK (26 tests, 26 assertions)

diwas@DESKTOP-T2GDHCD c:\xampp\htdocs\Assignment
# phpunit --coverage-html=./report
PHPUnit 7.5.7 by Sebastian Bergmann and contributors.

.....                                     26 / 26 (100%)

Time: 813 ms, Memory: 12.00 MB
OK (26 tests, 26 assertions)

Generating code coverage report in HTML format ... done

diwas@DESKTOP-T2GDHCD c:\xampp\htdocs\Assignment
#
```

Figure 5. 27 – PHPUnit test and report generation

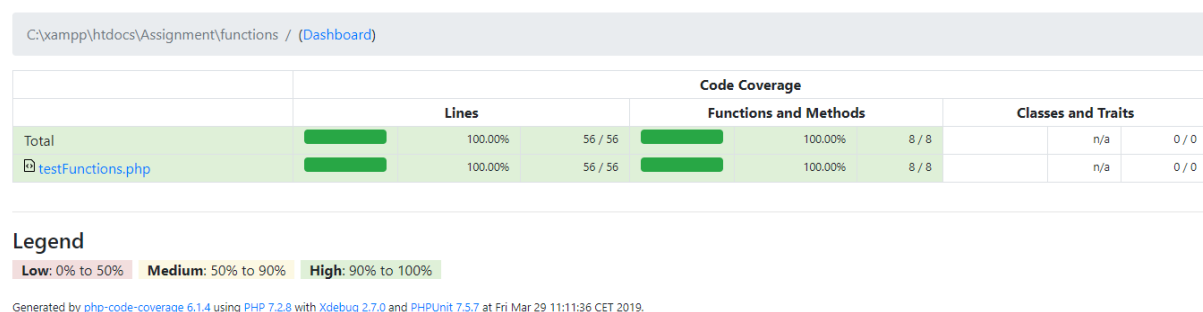


Figure 5. 28 – PHPUnit code coverage



















	Code Coverage								
	Classes and Traits			Functions and Methods				Lines	
Total		n/a	0 / 0		100.00%	8 / 8	CRAP		100.00% 56 / 56
checkCategory					100.00%	1 / 1	2		100.00% 4 / 4
checkFurniture					100.00%	1 / 1	4		100.00% 8 / 8
checkUpdate					100.00%	1 / 1	3		100.00% 6 / 6
checkUser					100.00%	1 / 1	5		100.00% 10 / 10
checkEnquiry					100.00%	1 / 1	5		100.00% 10 / 10
checkIsCategoryAvailable					100.00%	1 / 1	2		100.00% 6 / 6
checkIsUserNameAvailable					100.00%	1 / 1	2		100.00% 6 / 6
checkConfirmPassword					100.00%	1 / 1	3		100.00% 6 / 6

Figure 5. 29 – PHPUnit functions and code coverage

The tests that were performed:

Function checkCategory:

The function is used for confirming that a blank input field is not submitted for the category name. The function returns false if the field is left empty and true when not.

```

////////////////////////////////////

// Function to check category validation
function checkCategory($categoryName){
    $valid = true;
    if ($categoryName == '') {
        $valid = false;
    }
    return $valid;
}

////////////////////////////////////

```

Figure 5. 30 – Function checkCategory

```

use PHPUnit\Framework\TestCase;

// Test Whether the provided category name is empty
class CategoryTest extends TestCase {

    public function testEmptyName() {
        $emptyName = '';
        $valid = checkCategory($emptyName);
        $this->assertFalse($valid);
    }
}

```

Figure 5. 31 – checkCategory test class

Function checkFurniture:

The function is used for confirming that a blank input field is not submitted for the furniture details. The function returns false if any of the field is left empty and true if all the fields are not empty.

```
////////////////////////////////////  
  
// Function to check Furniture for Validation  
function checkFurniture($furniture){  
    $valid = true;  
    if($furniture['name']==''){  
        $valid = false;  
    }  
    if($furniture['description']==''){  
        $valid = false;  
    }  
    if($furniture['price']==''){  
        $valid = false;  
    }  
    return $valid;  
}  
  
////////////////////////////////////
```

Figure 5. 32 – Function checkFurniture

```
use PHPUnit\Framework\TestCase;  
  
// Test Whether the provided furniture contains missing fields  
class FurnitureTest extends TestCase {  
  
    public function testEmptyName() {  
        $invalidFurniture = [  
            'name' => '',  
            'description'=> 'Some Description',  
            'price'=> '500'  
        ];  
        $valid = checkFurniture($invalidFurniture);  
        $this->assertFalse($valid);  
    }  
}
```

Figure 5. 33 – checkFurniture test class

Function checkUpdate:

The function is used for confirming that a blank input field is not submitted for the update details. The function returns false if any of the field is left empty and true if all the fields are not empty.

```
////////////////////////////////////  
  
// Function to check Update for Validation  
function checkUpdate($update){  
    $valid = true;  
    if($update['title']==''){  
        $valid = false;  
    }  
    if($update['description']==''){  
        $valid = false;  
    }  
    return $valid;  
}  
  
////////////////////////////////////
```

Figure 5. 34 – Function checkUpdate

```
use PHPUnit\Framework\TestCase;  
  
// Test Whether the provided update contains missing fields  
class UpdateTest extends TestCase {  
  
    public function testEmptyName() {  
        $invalidUpdate = [  
            'title' => '',  
            'description'=> 'Some Description'  
        ];  
        $valid = checkUpdate($invalidUpdate);  
        $this->assertFalse($valid);  
    }  
}
```

Figure 5. 35 – checkUpdate test class

Function checkUser:

The function is used for confirming that a blank input field is not submitted for the user details. The function returns false if any of the field is left empty and true if all the fields are not empty.

```
////////////////////////////////////  
  
// Function to check User for Validation  
function checkUser($user){  
    $valid = true;  
    if($user['name']=='){  
        $valid = false;  
    }  
    if($user['username']=='){  
        $valid = false;  
    }  
    if($user['password']=='){  
        $valid = false;  
    }  
    if($user['confirmpassword']=='){  
        $valid = false;  
    }  
    return $valid;  
}  
  
////////////////////////////////////
```

Figure 5. 36 – Function checkUser

```
// Test Whether the provided user contains missing fields  
class UserTest extends TestCase {  
  
    public function testEmptyName() {  
        $invalidUser = [  
            'name' => '',  
            'username'=> 'username',  
            'password'=> 'password',  
            'confirmpassword'=> 'password2'  
        ];  
        $valid = checkUser($invalidUser);  
        $this->assertFalse($valid);  
    }  
}
```

Figure 5. 37 – checkUser test class

Function checkEnquiry:

The function is used for confirming that a blank input field is not submitted for the user details. The function returns false if any of the field is left empty and true if all the fields are not empty.

```
////////////////////////////////////  
  
// Function to check Enquiry for Validation  
function checkEnquiry($enquiry){  
    $valid = true;  
    if($enquiry['fullname']==''){  
        $valid = false;  
    }  
    if($enquiry['email']==''){  
        $valid = false;  
    }  
    if($enquiry['contact']==''){  
        $valid = false;  
    }  
    if($enquiry['message']==''){  
        $valid = false;  
    }  
    return $valid;  
}  
  
////////////////////////////////////
```

Figure 5. 38 – Function checkEnquiry

```
use PHPUnit\Framework\TestCase;  
  
// Test Whether the provided furniture contains missing fields  
class FurnitureTest extends TestCase {  
  
    public function testEmptyName() {  
        $invalidFurniture = [  
            'name' => '',  
            'description'=> 'Some Description',  
            'price'=> '500'  
        ];  
        $valid = checkFurniture($invalidFurniture);  
        $this->assertFalse($valid);  
    }  
}
```

Figure 5. 39 – checkEnquiry test class

Function checkIsCategoryAvailable:

The function is used for checking whether the provided category name already exists in the database. This function is used so that it does not allow duplicate categories with same names which would cause conflicts when adding products.

```
////////////////////////////////////  
  
// Function to check if category name is available  
function checkIsCategoryAvailable($catName){  
    $valid = true;  
    $categoryClass = new DatabaseTable('category');  
    $category = $categoryClass->find('name', $catName);  
    if($category->rowCount()>0)  
        $valid = false;  
    return $valid;  
}  
  
////////////////////////////////////
```

Figure 5. 40 – Function checkIsCategoryAvailable

```
require_once 'functions/testFunctions.php';  
require_once 'models/databasetable.php';  
require_once 'dbconnect/dbconnect.php';  
  
use PHPUnit\Framework\TestCase;  
  
// Test Whether the provided category name already exists in database  
class CategoryAvailabilityTest extends TestCase {  
  
    public function testExistingName() {  
        $name = 'Beds';  
        $valid = checkIsCategoryAvailable($name);  
        $this->assertFalse($valid);  
    }  
}
```

Figure 5. 41 – checkIsCategoryAvailable test class

Function checkIsUsernameAvailable:

The function is used for checking whether the provided username already exists in the database. This function is used so that it does not allow duplicate users with same names which would cause conflicts when logging in.

```
////////////////////////////////////  
  
// Function to check if username is available  
function checkIsUserNameAvailable($username){  
    $valid = true;  
    $userClass = new DatabaseTable('users');  
    $users = $userClass->find('username', $username);  
    if($users->rowCount()>0)  
        $valid = false;  
    return $valid;  
}  
  
////////////////////////////////////
```

Figure 5. 42 – Function checkIsUsernameAvailable

```
use PHPUnit\Framework\TestCase;  
  
// Test Whether the provided username already exists in database  
class UsernameAvailabilityTest extends TestCase {  
  
    public function testExistingName() {  
        $name = 'admin';  
        $valid = checkIsUserNameAvailable($name);  
        $this->assertFalse($valid);  
    }  
  
    public function testNonExistingName() {  
        $name = 'Astronaut';  
        $valid = checkIsUserNameAvailable($name);  
        $this->assertTrue($valid);  
    }  
}
```

Figure 5. 43 – checkIsUsernameAvailable test class

Function checkConfirmPassword:

The function is used for checking whether the provided two passwords match. Additionally, the function also checks if the provided password is less than 8 characters and returns false if it is.

```
////////////////////////////////////  
  
// Function to check whether provided passwords match and password is valid  
function checkConfirmPassword($user){  
    $valid = true;  
    if($user['password']!=$user['confirmpassword'])  
        $valid = false;  
    if(strlen($user['password'])<8)  
        $valid = false;  
    return $valid;  
}  
  
////////////////////////////////////
```

Figure 5. 44 – Function checkConfirmPassword

```
use PHPUnit\Framework\TestCase;  
  
// Test Whether the provided passwords match and password is valid  
class ConfirmPasswordTest extends TestCase {  
  
    public function testInvalidMatch() {  
        $user = [  
            'password'=>'somepassword',  
            'confirmpassword'=>'otherpassword'  
        ];  
        $valid = checkConfirmPassword($user);  
        $this->assertFalse($valid);  
    }  
  
    public function testInvalidLength() {  
        $user = [  
            'password'=>'somepassword',  
            'confirmpassword'=>'otherpassword'  
        ];  
        $valid = checkConfirmPassword($user);  
        $this->assertFalse($valid);  
    }  
}
```

Figure 5. 45 – checkConfirmPassword test class

6. EVALUATION

6.1 Bugs or Weaknesses and Possible Solutions

Table 6.1 – Bugs or weaknesses and possible solutions

Bug/Weakness	Possible Solution
Bug found from test - When the user chooses a filter from the selection menu, the pagination is removed, and all the products are displayed in the page.	The pagination should be integrated to work with the filter. This can be done in the controller where if the filter is set, the products should be displayed in the pagination accordingly.
Absolute path is used in many cases because of CSS, JavaScript or other resource routing failures with the MVC framework followed. As such, the directory of the solution needs to be placed directly under the root folder.	Relative paths could be used, or a constant could be introduced by using 'define' keyword with absolute path to the root.
No confirmation for deleting data. Sometimes, it can be a mistake on the user's part and no confirmation dialog boxes or prompts are shown before removing the data.	Confirmation modal box could have been added which would have made the system more user friendly.
All the furniture are displayed in the same page in the staff and administrator section.	Pagination and search could be added to the back end of the system as well.

6.2 Strengths of the Solution

- Proper routing using htaccess mod rewrite because of which the users do not see long unreadable URLs
- Inclusion of a super administrator who is the only one allowed to add, edit and delete users, while the rest can only edit their own passwords.
- Cannot delete category with furniture under them
- Functions to check duplicate usernames or category names before allowing to add them.
- Features are easily accessible
- AJAX search implemented, and the products can be searched without refreshing the page
- The images that are uploaded are given unique names so that another image with same name can be uploaded without any problems.
- Pagination implemented, and all the products are not displayed in a single page, instead, products are displayed as three per page.


```

1
2 #htaccess file for root index.php
3 #Manages links so that they appear in a user friendly way without hard to read symbols
4
5 Options -MultiViews
6 RewriteEngine On
7
8 RewriteBase /Assignment/public
9
10 RewriteCond %{REQUEST_FILENAME} !-d
11 RewriteCond %{REQUEST_FILENAME} !-f
12
13 RewriteRule ^(.*)$ index.php?url=$1 [QSA]
14 RewriteRule ^([^\s]+)/Products/([^\s+])$ index.php?url=$1&category=$2 [QSA]
15 RewriteRule ^([^\s]+)/ParticularProduct/([^\s+])$ index.php?url=$1&productid=$2 [QSA]
16

```

Figure 6. 1 – Htdocs file and URL example

Manage Users					
	Name	Username	Edit	Delete	Edit Password
1	Administrator	admin	Edit	Delete	Edit Password
2	Anil Upreti	anilupreti	Edit	Delete	Edit Password
3	Ramesh Thapa	rthapa123	Edit	Delete	Edit Password
4	Diwas Lamsal	diwaslamsal	Edit	Delete	Edit Password

Manage Users					
	Name	Username	Edit	Delete	Edit Password
1	Administrator	admin	No Access	No Access	No Access
2	Anil Upreti	anilupreti	No Access	No Access	No Access
3	Ramesh Thapa	rthapa123	No Access	No Access	No Access
4	Diwas Lamsal	diwaslamsal	No Access	No Access	Edit Password

Figure 6. 2 – Super administrator and normal staff

```

////////////////////////////////////
/** Function isUserNameAvailable
 * Checks whether provided username already exists
 * @param username -The username selected
 * @param currentUserNme - The existing username(Used when editing)
 * @return flag - Boolean value according to whether username is available
 */
function isUserNameAvailable($username, $currentUserNme=""){
    $userClass = new DatabaseTable('users');
    $users = $userClass->find('username', $username);
    if($users->rowCount()>0 && $username!=$currentUserNme)
        return false;
    else
        return true;
}
////////////////////////////////////

```

Figure 6. 3 – Checking username function

```

// Random name is generated for unique image name
$dir = "images/furniture/";
$image = $dir.microtime(true).basename($name);
$_POST['image']['path']=$image;

```

The figure displays a collection of furniture images arranged in a grid. Each image is accompanied by a unique filename generated by the script, which combines a directory path, a timestamp, and the image's basename. The images include various types of furniture such as sofas, wardrobes, and beds. The filenames are: charmcorner.jpg, charmcorner2.jpg, ofaleftarm.jpg, 10.jpg, classywardrobe.jpg, classywardrobe2.jpg, 1553525897.723412.jpg, edsdfa1.jpg, edsdfa2.jpg, edsdfa3.jpg, edsdfa4.jpg, eds.jpg, and 1553525907.8812r.jpg.

Figure 6. 4 – Random name generation for images

7. CONCLUSION

The CSY 2028 Web Programming Term II teaches about dealing with industry level ways to use PHP and implementing it to produce working web applications. It comes as a package of MVC framework, structuring the PHP code, separating database code from the rest of the program, using AJAX and PHP, using mod rewrite to redesign URL looks and the way they can be implemented to the MVC pattern, and lastly, implementing PHPUnit to perform white box automated tests. The module provides necessary knowledge to implement it right at the industry level in making any server-side web applications using PHP including mobile applications, browser applications, or even games. Transitioning the acquired knowledge to other widely used real life applications such as Laravel, Node.js, python, java servlets, etc. would not be that difficult with some research as many aspects are common.

While the application completes all the requirements, it cannot be said as a perfect application. Some weaknesses could be improved, and it could have been made more efficient. If the work was to be done from scratch, a cleaner MVC approach would be used, also addressing the weaknesses discovered during this first attempt.

Other features that could be included if more time was available or the system was to be redesigned from scratch:

- Pagination could be implemented for the back end as well.
- System could be upgraded to support mailing features directly from the application which would help in replying the enquiries.
- A forgot password feature could be added which would allow the super administrator to be notified of which user lost their password and change or reset it accordingly.
- AJAX could be implemented on deleting data or other aspects of the application such as loading the different sections in the backend.
- JavaScript validations could be added to let the uses know about invalid data before they submit the form and trigger a refresh.
- Popup modal boxes could be used to warn users whenever they try to edit or delete data from the system.
- The routing would have been redesigned to make relative pathing work throughout the application as well as for further refinements in the URL.
- The code could be structured more efficiently so that the model and controllers be further separated.

REFERENCES

PHP MVC Application

Codecourse – YouTube (2014), Build a PHP MVC Application Parts (1-9). Available from: <https://www.youtube.com/watch?v=OsCTzGASImQ> [Accessed 01/03/2019]

How to add whitespace between word

Mark Elliot – Stackoverflow (2011), How do I add white space between word. Available from: <https://stackoverflow.com/questions/4701143/how-do-i-add-white-space-between-word> [Accessed 08/03/2019]

PHP Multiple Image Upload

Kaplesh Rajai – JavaScript Hive (2017), PHP Multiple Files Upload with Validation. Available from: <http://www.javascripthive.info/php/php-multiple-files-upload-validation/> [Accessed 07/03/2019]

PHP Pagination

Mike Dalisay – Code of Ninja (2013), Simple PHP Pagination to Handle Your Growing Data. Available from: <https://www.codeofaninja.com/2013/06/simple-php-pagination-script.html> [Accessed 05/03/2019]

PHP Pagination

Codecourse – YouTube (2015), Super Simple PHP Pagination. Available from: <https://www.youtube.com/watch?v=8WoxPWVxXHI> [Accessed 09/03/2019]

File Upload

W3Schools (2019), PHP 7 File Upload. Available from: https://www.w3schools.com/php7/php7_file_upload.asp [Accessed 02/03/2019]

Install Xdebug on Windows

Burhan Nasir (2017), How to Install Xdebug on Windows. Available from: <https://burhandodhy.me/2017/08/29/how-to-install-xdebug-on-windows/> [Accessed 15/03/2019]

Using htaccess

Noah Hendrix (2009), Using htaccess Files for pretty URLs. Available from: <https://code.tutsplus.com/tutorials/using-htaccess-files-for-pretty-urls--net-6049> [Accessed 09/03/2019]

PHP Get time in Milliseconds

Laurent – Stackoverflow (2012), How to get current time in milliseconds in PHP? Available from: <https://stackoverflow.com/questions/3656713/how-to-get-current-time-in-milliseconds-in-php> [Accessed 12/03/2019]