|  |
| --- |
| Advanced Website Development (BSCTUC-6AWD) My Food Online Order System |
| To: Mike Kimber  From: Andrew Laing  Date: 20/12/2018 |
| CITY OF LIVERPOOL COLLEGE 2018 |

Table of Contents

[Introduction 3](#_Toc533154048)

[Objectives 3](#_Toc533154049)

[Website Design Documentation 6](#_Toc533154050)

[Introduction 7](#_Toc533154051)

[Website Diagrams 8](#_Toc533154052)

[Diagram showing the interaction between Client, PHP Server and Database Server 9](#_Toc533154053)

[Site Navigation Diagram 10](#_Toc533154054)

[Site Folder Structure 11](#_Toc533154055)

[Storyboards 12](#_Toc533154056)

[Database Documentation 34](#_Toc533154057)

[Introduction 35](#_Toc533154058)

[Database Relationship Diagram 36](#_Toc533154059)

[Database Reference 38](#_Toc533154060)

[User Guide 51](#_Toc533154061)

[Testing 58](#_Toc533154062)

[Introduction 59](#_Toc533154063)

[Test Plan 60](#_Toc533154064)

[Test Data 77](#_Toc533154065)

[Test Log 82](#_Toc533154066)

[Example User Feedback Form 147](#_Toc533154067)

[Beta Testing Form (page 1) 148](#_Toc533154068)

[Beta Testing Form (page 2) 149](#_Toc533154069)

[Summary 150](#_Toc533154070)

[Evaluation 151](#_Toc533154071)

[Evaluation 152](#_Toc533154072)

[Has the prototype met the company’s objectives? 152](#_Toc533154073)

[Conclusion 156](#_Toc533154074)

[Bibliography 157](#_Toc533154075)

[Appendix A: Sample Code Listings 1 - Login Functionality 158](#_Toc533154076)

[Implementing ‘Login’ functionality 159](#_Toc533154077)

[Implementing ‘Log Out’ functionality 163](#_Toc533154078)

[Appendix B: Sample Code Listings 2 - Food Ordering Functionality 164](#_Toc533154079)

[Implementing the ‘Order Online’ functionality 165](#_Toc533154080)

[Appendix C: Sample Code Listings 3 - Enforcing Unique User Names 175](#_Toc533154081)

[Enforcing Unique User Names 176](#_Toc533154082)

# Introduction

Mai-Lin's Vegan Takeaway Restaurant is a well-established food outlet situated in the heart of Liverpool. The restaurant has seating for 50 people and has become very popular with local students, artists and office workers, especially during lunchtimes.

The restaurant owner is looking to expand the business by providing an online food delivery service. To demonstrate that he has the advanced web programming skills necessary to produce a working solution for the company, the author of this report has developed a prototype online food shopping website.

The pages which follow document the design, testing and development of the prototype.

# Objectives

Mai-Lin’s restaurant is looking to create an online delivery service where registered customers can order food for their homes or workplaces. Customers will pay in cash upon receipt of their orders, so there is no current need for the solution to include an online payment system such as PayPal, Stripe, or WorldPay.

To justify further expenditure and a future maintenance contract the prototype website must meet certain requirements. These objectives are outlined below with details of how the web technologies chosen by the developer can help to meet them.

1. *The website is to provide relevant and current information about the company’s products. It will also present customers with some background information about the company’s vegan philosophy.*  
   Hyper Text Markup Language (HTML) is the fundamental technology which webpages are built upon, and HTML5 is its most recent version. For the moment placeholder content will be added to the pages.  
     
   Pages will be designed to be easy to update and there are many useful tools and content management systems available which can simplify the process of adding content to these HTML templates.   
     
   Product details will be stored in a MySQL database and added to pages using PHP scripts. This will allow items on offer to be updated according to availability.
2. *The site must adapt to a wide range of devices. Today, customers are just as likely to be ordering food from their mobile phones as from laptops or desktops.*Bootstrap is an open-source framework of HTML and Cascading Style Sheet (CSS) templates which allows developers to create responsive sites which adjust dynamically to the devices they are viewed on. Bootstrap is added to projects by including a couple small files. It is robust, well tested and compatible with most modern browsers.
3. *Pages throughout the website must have a consistent look and feel.*CSS scripts can be used to apply the same colour scheme and typeface to all pages. In addition to Bootstrap’s CSS libraries, custom scripts will be written to organise content in the same manner on all pages.
4. *Content must be easy to find, and visitors must always know what page they are on.*A simple navigation bar shown at the top of the page can allow visitors to switch quickly between pages. The Bootstrap framework contains a selection of attractive examples which adapt to different devices.   
     
   To let site visitors know where they are a header containing a picture and title will be added to the top of each page. This will help to establish the website’s identity and is easy to implement using HTML and CSS.
5. *Visitors should be allowed to send comments, feedback and requests for further information to the website administrator.*PHP can be used to add the information sent from an HTML feedback form to a database. This can be done safely by using a stored MySQL procedure and sanitising the user input. HTML5 offers a variety of simple in-browser validation methods, and JavaScript can be used for further validation and to provide a meaningful message to the user if they make mistakes.
6. *The website will allow new customers to create an account.*
7. *Allow existing customers to edit their account details.*
8. *Allow existing customers to change their password.*  
     
   As with objective six a combination of PHP, HTML5 and JavaScript can be used to implement this functionality securely. Password details will be hashed using the MySQL SHA1 encryption function before being added to the database as a security measure.
9. *The website will allow customers to log into and out of their accounts.*  
     
   Bootstrap provides an attractive modal login form which can be attached to a link on the navigation bar. The form is easy to use and its design will be familiar to users of Google or Facebook. Login checks will be performed using a PHP script, and a JavaScript dialogue can show meaningful error messages to the user if they have completed the form incorrectly.   
     
   Once a user has logged in, a PHP script will alter the navigation bar and allow access to account management functions and the log out button.
10. *The website will allow registered customers to order food online. Customers will be able to make orders with the minimum amount of guidance. Multiple Items will be added to or removed from the shopping basket with a single click. Customers will be able to view the contents of their basket along with a running total before they place their order. The overall shopping experience should be smooth and professional.*  
      
    The order page will first verify that customers have logged in by examining session cookies stored on the server. The food menu can be organised into tabbed submenus by using Bootstrap ‘*nav-pills’* to reveal and hide content.   
      
    Submenus will help visitors to find the items they desire quickly and keep page content short and manageable. As stated above, item details will be stored in a MySQL database and added dynamically using PHP.   
      
    The customer’s order can be created using HTML5 buttons, date selectors and time selectors. These widgets will simplify the ordering process and allow customers to proceed quickly to the checkout.
11. *The website will deal with customer errors and system errors in a friendly manner which does not expose any sensitive information.*JavaScript dialogues will be used to inform customers about their mistakes, and PHP scripts will redirect users to a custom error page when there are problems with database queries instead of showing them the information disclosing error messages produced by PHP.

# Website Design Documentation

## Introduction

This section of the report contains the diagrams and storyboards created during the prototype’s design phase. This documentation was used as a reference during the development phase.

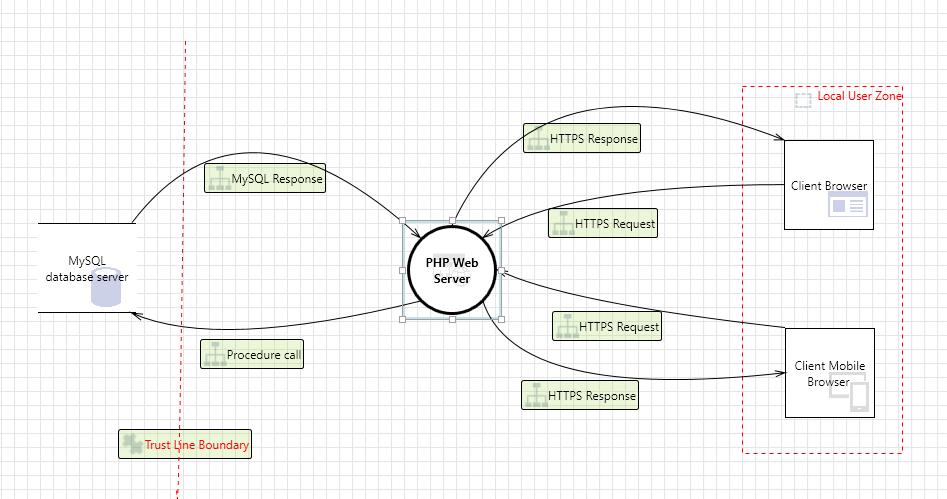
The first diagram shows how the three elements used for the website, the Database Server, PHP Server and the visitor’s browser, interact. This is followed by diagrams of the site’s navigation structure and folder structure for holding webpage files.

The storyboard section contains designs for each page specifying the fonts and colours to be used. A wireframe diagram was created to show each page’s proposed layout.

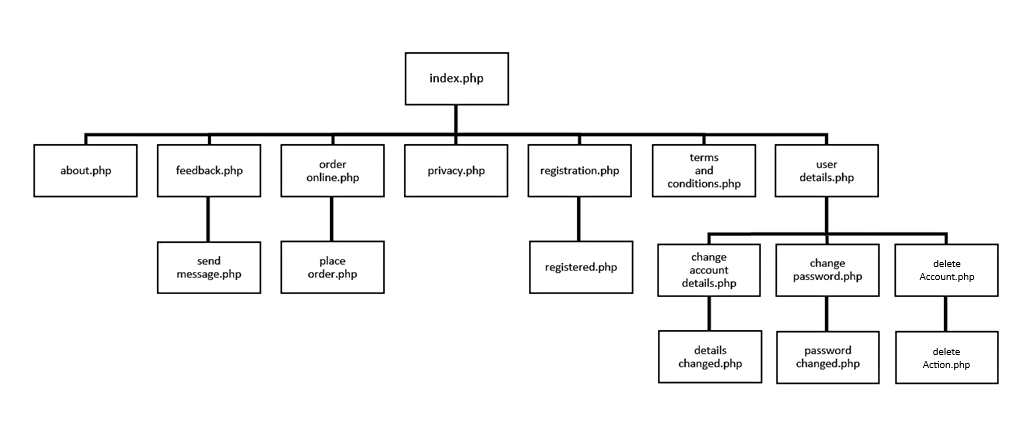
Following the design documentation during development helped to create a consistent look and feel throughout the website. The storyboard will also provide the site’s future maintainers with guidance when extra pages need to be designed.

## Website Diagrams

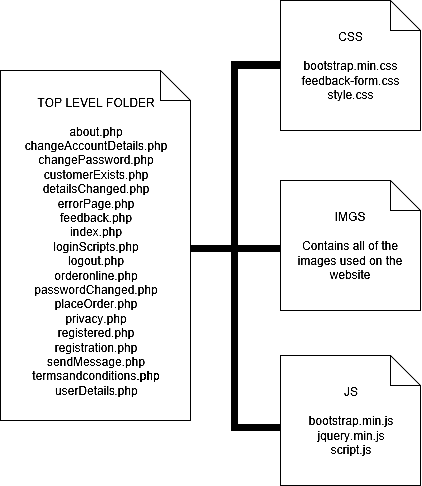
### Diagram showing the interaction between Client, PHP Server and Database Server



### Site Navigation Diagram



### Site Folder Structure



## Storyboards

|  |  |  |
| --- | --- | --- |
| **PROJECT NAME: Mai-Lin's Vegan Takeaway Restaurant Website** | | **PAGE:** about.php |
| **Screen size:**   Flexible **Fonts Used:**  General font: Arial, Helvetica, sans-serif **Background image:**  Body background image: bgimage.jpg   Title background image: mailin.jpg **Text colours:**  General text colour: #000   Header: #000  Footer: #32CD32  Modal header: #FFF **Background colours:**   Header: #222222  Footer: #222222  Modal header: #5CB85C  Modal footer: #F9F9F9  Facebook button: #3B5997  Facebook button hover colour: #2D4373  Twitter button: #55ACEF  Twitter button hover colour: #2795E7 **Additional colours:**   Navbar: Styled with Bootstrap as navbar-inverse |  | |
| **Description:** The About page will contain general details about the chef, veganism, and the restaurant.  **Links:** about.php, feedback.php, index.php, loginScripts.php, logout.php, orderonline.php, privacy.php, termsandconditions.php, userDetails.php, facebook.com, twitter.com. **Included files:** jquery.min.js, bootstrap.min.js, script.js, style.css. | | |
| **PROJECT NAME: Mai-Lin's Vegan Takeaway Restaurant Website** | | **PAGE:** changeAccountDetails.php |
| **Screen size:**   Flexible **Fonts Used:**  General font: Arial, Helvetica, sans-serif **Background image:**  Body background image: bgimage.jpg   Title background image: mailin.jpg **Text colours:**  General text colour: #000   Header: #000  Footer: #32CD32  Modal header: #FFF **Background colours:**   Header: #222222  Footer: #222222  Modal header: #5CB85C  Modal footer: #F9F9F9  Cancel button: #FFAD99  Cancel button hover colour: #FF5C33  Facebook button: #3B5997  Facebook button hover colour: #2D4373  Twitter button: #55ACEF  Twitter button hover colour: #2795E7 **Additional colours:**   Navbar: Styled with Bootstrap as navbar-inverse |  | |
| **Description:** The Change Account Details page will enable customers to update their account details.  **Links:** about.php, feedback.php, index.php, loginScripts.php, logout.php, orderonline.php, privacy.php, termsandconditions.php, userDetails.php, facebook.com, twitter.com. **Included files:** jquery.min.js, bootstrap.min.js, script.js, style.css.  **Notes:** When the submit button is clicked the page will POST to detailsChanged.php. | | |

|  |  |  |
| --- | --- | --- |
| **PROJECT NAME: Mai-Lin's Vegan Takeaway Restaurant Website** | | **PAGE:** changePassword.php |
| **Screen size:**   Flexible **Fonts Used:**  General font: Arial, Helvetica, sans-serif **Background image:**  Body background image: bgimage.jpg   Title background image: mailin.jpg  **Text colours:**  General text colour: #000   Header: #000  Footer: #32CD32  Modal header: #FFF **Background colours:**   Header: #222222  Footer: #222222  Modal header: #5CB85C  Modal footer: #F9F9F9  Cancel button: #FFAD99  Cancel button hover colour: #FF5C33  Facebook button: #3B5997  Facebook button hover colour: #2D4373  Twitter button: #55ACEF  Twitter button hover colour: #2795E7 **Additional colours:**   Navbar: Styled with Bootstrap as navbar-inverse |  | |
| **Description:** The Change Password page will enable customers to change their password.  **Links:** about.php, feedback.php, index.php, loginScripts.php, logout.php, orderonline.php, privacy.php, termsandconditions.php, userDetails.php, facebook.com, twitter.com. **Included files:** jquery.min.js, bootstrap.min.js, script.js, style.css.  **Notes:** When the submit button is clicked the page will POST to passwordChanged.php. | | |

|  |  |  |
| --- | --- | --- |
| **PROJECT NAME: Mai-Lin's Vegan Takeaway Restaurant Website** | | **PAGE:** customerExists.php |
| **Screen size:**   N/A **Fonts Used:**   N/A **Background image:**   N/A **Text colours:**   N/A **Background colours:**   N/A | Page not visible | |
| **Description:** The Customer Exists page will be used to hold a script used to check if a new customer is trying to register on registration.php with an already existing username. It will   be called using AJAX when the customer leaves the username field, and the result used to update the field with an error message if the username already exists.  **Links:** N/A | | |

|  |  |  |
| --- | --- | --- |
| **PROJECT NAME: Mai-Lin's Vegan Takeaway Restaurant Website** | | **PAGE:** deleteAccount.php |
| **Screen size:**   Flexible **Fonts Used:**  General font: Arial, Helvetica, sans-serif **Background image:**  Body background image: bgimage.jpg   Title background image: mailin.jpg  **Text colours:**  General text colour: #000   Header: #000  Footer: #32CD32  Modal header: #FFF **Background colours:**   Header: #222222  Footer: #222222  Modal header: #5CB85C  Modal footer: #F9F9F9  Cancel button: #FFAD99  Cancel button hover colour: #FF5C33  Facebook button: #3B5997  Facebook button hover colour: #2D4373  Twitter button: #55ACEF  Twitter button hover colour: #2795E7 **Additional colours:**   Navbar: Styled with Bootstrap as navbar-inverse |  | |
| **Description:** The Delete Account page will enable customers to delete their accounts. The input field will allow customers to provide feedback as to why they have chosen to delete   their accounts  **Links:** about.php, feedback.php, index.php, loginScripts.php, logout.php, orderonline.php, privacy.php, termsandconditions.php, userDetails.php, facebook.com, twitter.com. **Included files:** jquery.min.js, bootstrap.min.js, script.js, style.css.  **Notes:** When the submit button is clicked the page will POST to deleteAction.php | | |

|  |  |  |
| --- | --- | --- |
| **PROJECT NAME: Mai-Lin's Vegan Takeaway Restaurant Website** | | **PAGE:** deleteAction.php |
| **Screen size:**   N/A **Fonts Used:**  N/A **Background image:**  N/A **Text colours:**  N/A **Background colours:**   N/A | Page not visible | |
| **Description:** The Logout page is a PHP script used to write customer feedback to the database, and remove a customers account from the customer details table.It will show customers a confirmation then redirect them to the Home page.  **Links:** N/A | | |

|  |  |  |
| --- | --- | --- |
| **PROJECT NAME: Mai-Lin's Vegan Takeaway Restaurant Website** | | **PAGE:** detailsChanged.php |
| **Screen size:**   Flexible **Fonts Used:**  General font: Arial, Helvetica, sans-serif **Background image:**  Body background image: bgimage.jpg   Title background image: mailin.jpg **Text colours:**  General text colour: #000   Header: #000  Footer: #32CD32  Modal header: #FFF **Background colours:**   Header: #222222  Footer: #222222  Modal header: #5CB85C  Modal footer: #F9F9F9  Facebook button: #3B5997  Facebook button hover colour: #2D4373  Twitter button: #55ACEF  Twitter button hover colour: #2795E7 **Additional colours:**   Navbar: Styled with Bootstrap as navbar-inverse |  | |
| **Description:** The Details Changed page will be used to confirm to the user that their details have been changed to those entered on the form in the changeAccountDetails.php page.  **Links:** about.php, feedback.php, index.php, loginScripts.php, logout.php, orderonline.php, privacy.php, termsandconditions.php, userDetails.php, facebook.com, twitter.com. **Included files:** jquery.min.js, bootstrap.min.js, script.js, style.css.  **Notes:** The page will redirect to the Home page after 5 seconds if no links are clicked. | | |

|  |  |  |
| --- | --- | --- |
| **PROJECT NAME: Mai-Lin's Vegan Takeaway Restaurant Website** | | **PAGE:** errorPage.php |
| **Screen size:**   Flexible **Fonts Used:**  General font: Arial, Helvetica, sans-serif **Background image:**  Body background image: bgimage.jpg   Title background image: mailin.jpg **Text colours:**  General text colour: #000   Header: #000  Footer: #32CD32  Modal header: #FFF **Background colours:**   Header: #222222  Footer: #222222  Modal header: #5CB85C  Modal footer: #F9F9F9  Facebook button: #3B5997  Facebook button hover colour: #2D4373  Twitter button: #55ACEF  Twitter button hover colour: #2795E7 **Additional colours:**   Navbar: Styled with Bootstrap as navbar-inverse |  | |
| **Description:** The Error page will be shown when anything goes wrong with database interactions. Rather than showing an error message disclosing database details, it will ask  the user to contact the site’s administrator.  **Links:** about.php, feedback.php, index.php, loginScripts.php, logout.php, orderonline.php, privacy.php, termsandconditions.php, userDetails.php, facebook.com, twitter.com. **Included files:** jquery.min.js, bootstrap.min.js, script.js, style.css. | | |

|  |  |  |
| --- | --- | --- |
| **PROJECT NAME: Mai-Lin's Vegan Takeaway Restaurant Website** | | **PAGE:** feedback.php |
| **Screen size:**   Flexible **Fonts Used:**  General font: Arial, Helvetica, sans-serif **Background image:**  Body background image: bgimage.jpg   Title background image: mailin.jpg **Text colours:**  General text colour: #000   Header: #000  Footer: #32CD32  Modal header: #FFF **Background colours:**   Header: #222222  Footer: #222222  Modal header: #5CB85C  Modal footer: #F9F9F9  Facebook button: #3B5997  Facebook button hover colour: #2D4373  Twitter button: #55ACEF  Twitter button hover colour: #2795E7 **Additional colours:**   Navbar: Styled with Bootstrap as navbar-inverse |  | |
| **Description:** The feedback page will allow all site visitors to send comments to the company.  **Links:** about.php, feedback.php, index.php, loginScripts.php, logout.php, orderonline.php, privacy.php, termsandconditions.php, userDetails.php, facebook.com, twitter.com. **Included files:** jquery.min.js, bootstrap.min.js, script.js, feedback-form.css, style.css.  **Notes:** When the submit button is clicked the page will POST to sendMessage.php. | | |

|  |  |  |
| --- | --- | --- |
| **PROJECT NAME: Mai-Lin's Vegan Takeaway Restaurant Website** | | **PAGE:** index.php |
| **Screen size:**   Flexible **Fonts Used:**  General font: Arial, Helvetica, sans-serif **Background image:**  Body background image: bgimage.jpg   Title background image: mailin.jpg **Text colours:**  General text colour: #000   Header: #000  Footer: #32CD32  Modal header: #FFF **Background colours:**   Header: #222222  Footer: #222222  Modal header: #5CB85C  Modal footer: #F9F9F9  Facebook button: #3B5997  Facebook button hover colour: #2D4373  Twitter button: #55ACEF  Twitter button hover colour: #2795E7 **Additional colours:**   Navbar: Styled with Bootstrap as navbar-inverse |  | |
| **Description:** The Index page is the Home page of the site, the first page that visitors will see when they arrive.  **Links:** about.php, feedback.php, index.php, loginScripts.php, logout.php, orderonline.php, privacy.php, termsandconditions.php, userDetails.php, facebook.com, twitter.com. **Included files:** jquery.min.js, bootstrap.min.js, script.js, style.css. | | |

|  |  |  |
| --- | --- | --- |
| **PROJECT NAME: Mai-Lin's Vegan Takeaway Restaurant Website** | | **PAGE:** loginScripts.php |
| **Screen size:**   N/A **Fonts Used:**  N/A **Background image:**  N/A **Text colours:**  N/A **Background colours:**   N/A | Page not visible | |
| **Description:** The Login Scripts page is a non-displayed page of PHP scripts used by all other pages to allow site login functionality.  **Links:** N/A | | |

|  |  |  |
| --- | --- | --- |
| **PROJECT NAME: Mai-Lin's Vegan Takeaway Restaurant Website** | | **PAGE:** logout.php |
| **Screen size:**   N/A **Fonts Used:**  N/A **Background image:**  N/A **Text colours:**  N/A **Background colours:**   N/A | Page not visible | |
| **Description:** The Logout page is a PHP script used to log visitors out of their accounts. Once it has logged out customers, it will redirect them to the Home page.  **Links:** N/A | | |

|  |  |  |
| --- | --- | --- |
| **PROJECT NAME: Mai-Lin's Vegan Takeaway Restaurant Website** | | **PAGE:** orderonline.php |
| **Screen size:**   Flexible **Fonts Used:**  General font: Arial, Helvetica, sans-serif **Background image:**  Body background image: bgimage.jpg   Title background image: mailin.jpg **Text colours:**  General text colour: #000   Header: #000  Footer: #32CD32  Modal header: #FFF **Background colours:**   Header: #222222  Footer: #222222  Modal header: #5CB85C  Modal footer: #F9F9F9  Facebook button: #3B5997  Facebook button hover colour: #2D4373  Twitter button: #55ACEF  Twitter button hover colour: #2795E7 **Additional colours:**   Navbar: Styled with Bootstrap as navbar-inverse |  | |
| **Description:** The Order Online page is the site’s shopping page where customers can place delivery orders for food. It will contain tabs of different food categories and a shopping  basket showing the customer’s currently selected items.  **Links:** about.php, feedback.php, index.php, loginScripts.php, logout.php, orderonline.php, privacy.php, termsandconditions.php, userDetails.php, facebook.com, twitter.com. **Included files:** jquery.min.js, bootstrap.min.js, script.js, style.css.  **Notes:** When the submit button is clicked the page will POST to placeOrder.php. | | |

|  |  |  |
| --- | --- | --- |
| **PROJECT NAME: Mai-Lin's Vegan Takeaway Restaurant Website** | | **PAGE:** passwordChanged.php |
| **Screen size:**   Flexible **Fonts Used:**  General font: Arial, Helvetica, sans-serif **Background image:**  Body background image: bgimage.jpg   Title background image: mailin.jpg **Text colours:**  General text colour: #000   Header: #000  Footer: #32CD32  Modal header: #FFF **Background colours:**   Header: #222222  Footer: #222222  Modal header: #5CB85C  Modal footer: #F9F9F9  Facebook button: #3B5997  Facebook button hover colour: #2D4373  Twitter button: #55ACEF  Twitter button hover colour: #2795E7 **Additional colours:**   Navbar: Styled with Bootstrap as navbar-inverse |  | |
| **Description:** The Password Changed page will be used to confirm to the user that their password has been changed to that entered on the form in the changePassword.php page.  **Links:** about.php, feedback.php, index.php, loginScripts.php, logout.php, orderonline.php, privacy.php, termsandconditions.php, userDetails.php, facebook.com, twitter.com. **Included files:** jquery.min.js, bootstrap.min.js, script.js, style.css.  **Notes:** The page will redirect to the Home page after 5 seconds if no links are clicked. | | |

|  |  |  |
| --- | --- | --- |
| **PROJECT NAME: Mai-Lin's Vegan Takeaway Restaurant Website** | | **PAGE:** placeOrder.php |
| **Screen size:**   Flexible **Fonts Used:**  General font: Arial, Helvetica, sans-serif **Background image:**  Body background image: bgimage.jpg   Title background image: mailin.jpg **Text colours:**  General text colour: #000   Header: #000  Footer: #32CD32  Modal header: #FFF **Background colours:**   Header: #222222  Footer: #222222  Modal header: #5CB85C  Modal footer: #F9F9F9  Facebook button: #3B5997  Facebook button hover colour: #2D4373  Twitter button: #55ACEF  Twitter button hover colour: #2795E7 **Additional colours:**   Navbar: Styled with Bootstrap as navbar-inverse |  | |
| **Description:** The Place Order page will be used to confirm to the user that their order entered into the form on the orderOnline.php page has been placed.  **Links:** about.php, feedback.php, index.php, loginScripts.php, logout.php, orderonline.php, privacy.php, termsandconditions.php, userDetails.php, facebook.com, twitter.com. **Included files:** jquery.min.js, bootstrap.min.js, script.js, style.css.  **Notes:** The page will redirect to the Home page after 5 seconds if no links are clicked. | | |

|  |  |  |
| --- | --- | --- |
| **PROJECT NAME: Mai-Lin's Vegan Takeaway Restaurant Website** | | **PAGE:** privacy.php |
| **Screen size:**   Flexible **Fonts Used:**  General font: Arial, Helvetica, sans-serif **Background image:**  Body background image: bgimage.jpg   Title background image: mailin.jpg **Text colours:**  General text colour: #000   Header: #000  Footer: #32CD32  Modal header: #FFF **Background colours:**   Header: #222222  Footer: #222222  Modal header: #5CB85C  Modal footer: #F9F9F9  Facebook button: #3B5997  Facebook button hover colour: #2D4373  Twitter button: #55ACEF  Twitter button hover colour: #2795E7 **Additional colours:**   Navbar: Styled with Bootstrap as navbar-inverse |  | |
| **Description:** The Privacy page will contain details about the company’s data privacy policy.  **Links:** about.php, feedback.php, index.php, loginScripts.php, logout.php, orderonline.php, privacy.php, termsandconditions.php, userDetails.php, facebook.com, twitter.com. **Included files:** jquery.min.js, bootstrap.min.js, script.js, style.css. | | |

|  |  |  |
| --- | --- | --- |
| **PROJECT NAME: Mai-Lin's Vegan Takeaway Restaurant Website** | | **PAGE:** registered.php |
| **Screen size:**   Flexible **Fonts Used:**  General font: Arial, Helvetica, sans-serif **Background image:**  Body background image: bgimage.jpg   Title background image: mailin.jpg **Text colours:**  General text colour: #000   Header: #000  Footer: #32CD32  Modal header: #FFF **Background colours:**   Header: #222222  Footer: #222222  Modal header: #5CB85C  Modal footer: #F9F9F9  Facebook button: #3B5997  Facebook button hover colour: #2D4373  Twitter button: #55ACEF  Twitter button hover colour: #2795E7 **Additional colours:**   Navbar: Styled with Bootstrap as navbar-inverse |  | |
| **Description:** The Registered page will be used to confirm to the user that their information entered into the form on the registration page has been sent, an account has been   created, and they will now be able to log into a personal customer account.  **Links:** about.php, feedback.php, index.php, loginScripts.php, logout.php, orderonline.php, privacy.php, termsandconditions.php, userDetails.php, facebook.com, twitter.com. **Included files:** jquery.min.js, bootstrap.min.js, script.js, style.css.  **Notes:** The page will redirect to the Home page after 5 seconds if no links are clicked. | | |

|  |  |  |
| --- | --- | --- |
| **PROJECT NAME: Mai-Lin's Vegan Takeaway Restaurant Website** | | **PAGE:** registration.php |
| **Screen size:**   Flexible **Fonts Used:**  General font: Arial, Helvetica, sans-serif **Background image:**  Body background image: bgimage.jpg   Title background image: mailin.jpg **Text colours:**  General text colour: #000   Header: #000  Footer: #32CD32  Modal header: #FFF **Background colours:**   Header: #222222  Footer: #222222  Modal header: #5CB85C  Modal footer: #F9F9F9  Facebook button: #3B5997  Facebook button hover colour: #2D4373  Twitter button: #55ACEF  Twitter button hover colour: #2795E7 **Additional colours:**   Navbar: Styled with Bootstrap as navbar-inverse |  | |
| **Description:** The Registration page will contain a form which will allow site visitors to create an account for ordering food.  **Links:** about.php, feedback.php, index.php, loginScripts.php, logout.php, orderonline.php, privacy.php, termsandconditions.php, userDetails.php, facebook.com, twitter.com. **Included files:** jquery.min.js, bootstrap.min.js, script.js, feedback-form.css, style.css.  **Notes:** When the submit button is clicked the page will POST to registered.php. | | |

|  |  |  |
| --- | --- | --- |
| **PROJECT NAME: Mai-Lin's Vegan Takeaway Restaurant Website** | | **PAGE:** sendMessage.php |
| **Screen size:**   Flexible **Fonts Used:**  General font: Arial, Helvetica, sans-serif **Background image:**  Body background image: bgimage.jpg   Title background image: mailin.jpg **Text colours:**  General text colour: #000   Header: #000  Footer: #32CD32  Modal header: #FFF **Background colours:**   Header: #222222  Footer: #222222  Modal header: #5CB85C  Modal footer: #F9F9F9  Facebook button: #3B5997  Facebook button hover colour: #2D4373  Twitter button: #55ACEF  Twitter button hover colour: #2795E7 **Additional colours:**   Navbar: Styled with Bootstrap as navbar-inverse |  | |
| **Description:** The Send Message page will be used to confirm to the user that the information entered into the form on the feedback page has been sent.  **Links:** about.php, feedback.php, index.php, loginScripts.php, logout.php, orderonline.php, privacy.php, termsandconditions.php, userDetails.php, facebook.com, twitter.com. **Included files:** jquery.min.js, bootstrap.min.js, script.js, style.css.  **Notes:** The page will redirect to the Home page after 5 seconds if no links are clicked. | | |

|  |  |  |
| --- | --- | --- |
| **PROJECT NAME: Mai-Lin's Vegan Takeaway Restaurant Website** | | **PAGE:** termsandconditions.php |
| **Screen size:**   Flexible **Fonts Used:**  General font: Arial, Helvetica, sans-serif **Background image:**  Body background image: bgimage.jpg   Title background image: mailin.jpg **Text colours:**  General text colour: #000   Header: #000  Footer: #32CD32  Modal header: #FFF **Background colours:**   Header: #222222  Footer: #222222  Modal header: #5CB85C  Modal footer: #F9F9F9  Facebook button: #3B5997  Facebook button hover colour: #2D4373  Twitter button: #55ACEF  Twitter button hover colour: #2795E7 **Additional colours:**   Navbar: Styled with Bootstrap as navbar-inverse |  | |
| **Description:** The Terms and Conditions page will contain details about the company’s terms and conditions for use of its site and restaurant.  **Links:** about.php, feedback.php, index.php, loginScripts.php, logout.php, orderonline.php, privacy.php, termsandconditions.php, userDetails.php, facebook.com, twitter.com. **Included files:** jquery.min.js, bootstrap.min.js, script.js, style.css. | | |

|  |  |  |
| --- | --- | --- |
| **PROJECT NAME: Mai-Lin's Vegan Takeaway Restaurant Website** | | **PAGE:** userDetails.php |
| **Screen size:**   Flexible **Fonts Used:**  General font: Arial, Helvetica, sans-serif **Background image:**  Body background image: bgimage.jpg   Title background image: mailin.jpg **Text colours:**  General text colour: #000   Header: #000  Footer: #32CD32  Modal header: #FFF **Background colours:**   Header: #222222  Footer: #222222  Modal header: #5CB85C  Modal footer: #F9F9F9  Facebook button: #3B5997  Facebook button hover colour: #2D4373  Twitter button: #55ACEF  Twitter button hover colour: #2795E7 **Additional colours:**   Navbar: Styled with Bootstrap as navbar-inverse |  | |
| **Description:** The User Details page will show the customer their account details.  **Links:** about.php, feedback.php, index.php, loginScripts.php, logout.php, orderonline.php, privacy.php, termsandconditions.php, userDetails.php, facebook.com, twitter.com. **Included files:** jquery.min.js, bootstrap.min.js, script,js, style.css. | | |

# Database Documentation

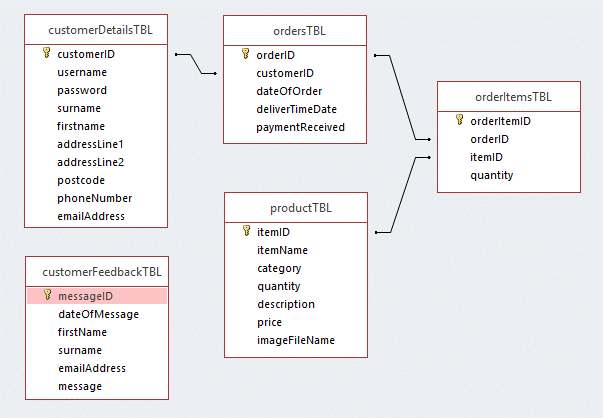
## Introduction

This section of the report contains details of the MySQL database created as the website’s back-end. The first diagram shows the relationships created between the database’s five tables. Tables are normalised to the Boyce-Codd third normal form to reduce data duplication.

The database reference section contains details of these tables, and the many useful MySQL procedures created for the website. Web developers looking to extend the current prototype will find this documentation useful, as will developers of mobile phone applications which need access to the database.

## Database Relationship Diagram

Database Relationship Diagram



## Database Reference

|  |  |  |  |
| --- | --- | --- | --- |
| Mai-Lin Vegan Takeaway Restaurant Database customerDetailsTBL | | | |
| FIELD NAME | DATATYPE | DESCRIPTION | NOTES |
| customerID | INT | Unique ID for a customer. | Primary Key. AUTO\_INCREMENT. |
| username | VARCHAR(35) | Unique username for a customer. | Primary Key. |
| password | VARCHAR(255) | Stored hash of a customer’s password | Required field. Creates a hash of the customers password using the sha1() function. |
| surname | VARCHAR(35) | The customer’s surname. | Required field. |
| firstname | VARCHAR(35) | The customer’s first name. | Required field. |
| addressLine1 | VARCHAR(35) | First line of the customer’s address. |  |
| addressLine2 | VARCHAR(35) | Second line of the customer’s address. |  |
| postcode | VARCHAR(8) | Customer’s postcode. |  |
| phoneNumber | VARCHAR(35) | Customer’s phone number. |  |
| emailAddress | VARCHAR(50) | Customer’s email address |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Mai-Lin Vegan Takeaway Restaurant Database ordersTBL | | | |
| FIELD NAME | DATATYPE | DESCRIPTION | NOTES |
| orderID | INT | Unique ID for an order. | Primary Key. AUTO\_INCREMENT. |
| customerID | INT | Unique ID for a customer. | Required field.  FOREIGN KEY CONSTRAINT fk\_customer\_id  REFERENCES customerDetailsTBL (customerID) |
| dateOfOrder | DATE | The date that the order was placed. | Required field. |
| deliveryTimeDate | DATETIME | The date and time when the delivery should be made. | Required field. |
| paymentReceived | BIT | If payment has been received for the order 1, otherwise 0. | There is no boolean type in MySQL so BIT was used instead. DEFAULT value is 0. |

|  |  |  |  |
| --- | --- | --- | --- |
| Mai-Lin Vegan Takeaway Restaurant Database orderItemsTBL | | | |
| FIELD NAME | DATATYPE | DESCRIPTION | NOTES |
| orderItemID | INT | Unique ID for an item to add to an order. | Primary Key. AUTO\_INCREMENT. |
| orderID | INT | Unique ID for an order. | Required field.  FOREIGN KEY CONSTRAINT fk\_order\_id  REFERENCES ordersTBL (orderID) |
| itemID | INT | Unique ID for a product. | Required field.  FOREIGN KEY CONSTRAINT fk\_item\_id  REFERENCES productTBL (itemID) |
| quantity | TINYINT | The quantity of the item ordered. | Required field. |

|  |  |  |  |
| --- | --- | --- | --- |
| Mai-Lin Vegan Takeaway Restaurant Database  productTBL | | | |
| FIELD NAME | DATATYPE | DESCRIPTION | NOTES |
| itemID | INT | Unique ID for a product. | Primary Key. AUTO\_INCREMENT. |
| itemName | VARCHAR(50) | The name of the product. (e.g., Stir-Fried Tofu) | Required field. |
| category | VARCHAR(20) | The category of the product (e.g., Entree) | Required field. |
| quantity | TINYINT | The quantity of the item counted as one unit (e.g., 5 spring rolls) | Required field. |
| description | VARCHAR(100) | A description of the product. |  |
| price | DECIMAL(13,4) | The price of the product in pounds and pence (e.g., £3.95) | Required field. DECIMAL(13,4) was used because MySQL has no MONEY datatype, and it fits in with *Generally Accepted Accounting Principles* (GAAP) |
| imageFileName | VARCHAR(50) | The name of the image file of the product which will be shown on the website. |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Mai-Lin Vegan Takeaway Restaurant Database  customerFeedbackTBL | | | |
| FIELD NAME | DATATYPE | DESCRIPTION | NOTES |
| messageID | INT | Unique ID for a customer’s feedback message. | Primary Key. AUTO\_INCREMENT. |
| dateOfMessage | DATETIME | The date and time that the message was received. | Required field. |
| firstname | VARCHAR(35) | The message sender’s first name. |  |
| surname | VARCHAR(35) | The message sender’s surname. |  |
| emailAddress | VARCHAR(50) | The message sender’s email address. |  |
| message | VARCHAR(500) | A feedback message sent. |  |

|  |  |
| --- | --- |
| Mai-Lin Vegan Takeaway Restaurant Database customerDetailsTBL procedures | |
| **Procedure Name:** | insertCustomerDetails |
| **Author:** | Andrew Laing |
| **Last Updated:** | 09/11/2018 |
| **Description:** | This procedure inserts a customer details record. |
| **Parameters:** | in\_username – A customer’s chosen username in\_password – A customers chosen password. in\_surname – A customer’s surname in\_firstname – A customer’s given name in\_addressLine1 – The first line of a customer’s address  in\_addressLine2 – The second line of a customer’s address in\_postcode – A customer’s postcode in\_phoneNumber – A customer’s phone number in\_emailAddress – A customer’s email address. |
| **Example Usage:** | CALL insertCustomerDetails ('scotty', 'password123','Scott', 'Randolph', '13 Pleasant Street', 'Liverpool', 'L19 0NE', '0151345987','randyscott@gmail.com'); |
| **Notes:** | The customer’s password is converted to a SHA1 hash before being added to the record. |
|  | |
| **Procedure Name:** | updateCustomerDetails |
| **Author:** | Andrew Laing |
| **Last Updated:** | 09/11/2018 |
| **Description:** | This procedure updates a customer details record. |
| **Parameters:** | in\_customerID – The unique ID for a customer details record. in\_surname – A customer’s surname  in\_firstname – A customer’s given name  in\_addressLine1 – The first line of a customer’s address  in\_addressLine2 – The second line of a customer’s address  in\_postcode – A customer’s postcode  in\_phoneNumber – A customer’s phone number  in\_emailAddress – A customer’s email address. |
| **Example Usage:** | CALL updateCustomerDetails (1, 'Scott', 'Randolph', '13 Pleasant Street', 'Liverpool', 'L19 0NE', '0151345987','realrandyscott@gmail.com'); |
|  | |
| **Procedure Name:** | getCustomerDetails |
| **Author:** | Andrew Laing |
| **Last Updated:** | 09/11/2018 |
| **Description:** | This procedure returns all customer details except for the username customerID and password hash. |
| **Parameters:** | in\_customerID – The unique ID for a customer details record. |
| **Example Usage:** | CALL getCustomerDetails(2); |
|  | |
| **Procedure Name:** | updatePassword |
| Author: | Andrew Laing |
| Last Updated: | 09/11/2018 |
| Description: | This procedure updates the stored hash for a customer’s password. |
| Parameters: | in\_customerID – The unique ID for a customer details record. in\_password – A customer’s chosen password. |
| Example Usage: | CALL updatePassword(2, 'strongerPassword'); |
|  | |
| **Procedure Name:** | deleteCustomerDetails |
| **Author:** | Andrew Laing |
| **Last Updated:** | 09/11/2018 |
| **Description:** | This procedure deletes a customer details record. |
| **Parameters:** | in\_customerID – The unique ID for a customer details record. |
| **Example Usage:** | CALL deleteCustomerDetails(11); |
|  | |
| **Procedure Name:** | checkPasswordIsValid |
| **Author:** | Andrew Laing |
| **Last Updated:** | 09/11/2018 |
| **Description:** | This procedure is used to check whether a username/password combination exists in the customerDetailsTL table. It returns the customerID if a matching record exists otherwise the result will be empty. |
| **Parameters:** | In\_username – A customer’s username in\_password – A customers password. |
| **Example Usage:** | CALL checkPasswordIsValid('scotty', 'strongerPassword'); |
|  | |
| **Procedure Name:** | checkPasswordIDCombo |
| **Author:** | Andrew Laing |
| **Last Updated:** | 09/11/2018 |
| **Description:** | This procedure is used to check whether a username/password combination exists in the customerDetailsTL table. It returns the customerID if a matching record exists otherwise the result will be empty. |
| **Parameters:** | in\_customerID – The unique ID for a customer details record. in\_password – A customers password. |
| **Example Usage:** | CALL checkPasswordIDCombo(2, 'strongerPassword'); |
|  | |
| **Function Name:** | usernameExists |
| **Author:** | Andrew Laing |
| **Last Updated:** | 09/11/2018 |
| **Description:** | This procedure is used to check whether a username exists in the customerDetailsTL table. It returns 0 if it exists otherwise 1. |
| **Parameters:** | in\_username – A customer’s username |
| **Example Usage:** | CALL usernameExists('scotty'); |

|  |  |
| --- | --- |
| Mai-Lin Vegan Takeaway Restaurant Database ordersTBL procedures | |
| **Procedure Name:** | insertOrder |
| **Author:** | Andrew Laing |
| **Last Updated:** | 09/11/2018 |
| **Description:** | This procedure inserts an order record. |
| **Parameters:** | in\_customerID – A customer’s unique ID.  in\_dateOfOrder – The date when the order was placed. in\_deliveryTimeDate – The date and time when the order should be delivered. in\_paymentReceived – Whether payment has been received for the order or not. |
| **Example Usage:** | CALL insertOrder(3, '2018-10-26', '2018-10-27 20:30:00',0); |
|  | |
| **Procedure Name:** | addNewOrder |
| **Author:** | Andrew Laing |
| **Last Updated:** | 09/11/2018 |
| **Description:** | This procedure inserts an order record. It is used as a wrapper method for insertOrder to automatically insert today’s date and zero for payment received. |
| **Parameters:** | in\_customerID – A customer’s unique ID.  in\_deliveryTimeDate – The date and time when the order should be delivered. |
| **Example Usage:** | CALL addNewOrder(3, '2018-10-27 20:30:00'); |
|  | |
| **Procedure Name:** | updateOrder |
| **Author:** | Andrew Laing |
| **Last Updated:** | 09/11/2018 |
| **Description:** | This procedure updates an order record. |
| **Parameters:** | in\_orderID – The unique ID for an order. in\_customerID – A customer’s unique ID.  in\_dateOfOrder – The date when the order was placed. in\_deliveryTimeDate – The date and time when the order should be delivered. in\_paymentReceived – Whether payment has been received for the order or not. |
| **Example Usage:** | CALL updateOrder(1, 3, '2018-10-26', '2018-10-27 19:30:00',0); |
|  | |
| **Procedure Name:** | updateOrderSetPaymentReceived |
| **Author:** | Andrew Laing |
| **Last Updated:** | 09/11/2018 |
| **Description:** | This procedure sets the Payment Received field of an order record to yes (1). |
| **Parameters:** | in\_orderID – The unique ID for an order. |
| **Example Usage:** | CALL updateOrderSetPaymentReceived(1); |
|  | |
| **Procedure Name:** | deleteOrder |
| **Author:** | Andrew Laing |
| **Last Updated:** | 09/11/2018 |
| **Description:** | This procedure deletes an order record. |
| **Parameters:** | in\_orderID – The unique ID for an order. |
| **Example Usage:** | CALL deleteOrder(1); |
|  | |
| **Procedure Name:** | getOrderDeliveryDetails |
| **Author:** | Andrew Laing |
| **Last Updated:** | 09/11/2018 |
| **Description:** | This procedure returns the customer’s delivery details for an order. |
| **Parameters:** | in\_orderID – The unique ID for an order. |
| **Example Usage:** | CALL getOrderDeliveryDetails (2); |
|  | |
| **Procedure Name:** | getOrderItemDetails |
| **Author:** | Andrew Laing |
| **Last Updated:** | 09/11/2018 |
| **Description:** | This procedure returns the details for items attached to an order. |
| **Parameters:** | in\_orderID – The unique ID for an order. |
| **Example Usage:** | CALL getOrderItemDetails (3); |

|  |  |
| --- | --- |
| Mai-Lin Vegan Takeaway Restaurant Database orderItemsTBL procedures | |
| **Procedure Name:** | insertOrderItem |
| **Author:** | Andrew Laing |
| **Last Updated:** | 09/11/2018 |
| **Description:** | This procedure inserts an order item record. |
| **Parameters:** | in\_orderID – A order’s unique ID. in\_itemID – The unique ID for a product. in\_quantity – The quantity of the product ordered. |
| **Example Usage:** | CALL insertOrderItem(2,4,1); |
|  | |
| **Procedure Name:** | updateOrderItem |
| **Author:** | Andrew Laing |
| **Last Updated:** | 09/11/2018 |
| **Description:** | This procedure updates an order item record. |
| **Parameters:** | In\_orderItemID – The unique ID for an order item. in\_orderID – The unique ID for an order. in\_itemID – The unique ID for a product. in\_quantity – The quantity of the product ordered. |
| **Example Usage:** | CALL updateOrderItem(6,3,22,1); |
|  | |
| **Procedure Name:** | deleteOrderItem |
| **Author:** | Andrew Laing |
| **Last Updated:** | 09/11/2018 |
| **Description:** | This procedure deletes an order item record. |
| **Parameters:** | in\_orderItemID – The unique ID for an order. |
| **Example Usage:** | CALL deleteOrderItem(6); |

|  |  |
| --- | --- |
| Mai-Lin Vegan Takeaway Restaurant Database productTBL procedures | |
| **Procedure Name:** | insertProduct |
| **Author:** | Andrew Laing |
| **Last Updated:** | 09/11/2018 |
| **Description:** | This procedure inserts an product record. |
| **Parameters:** | in\_itemName – The name of the product. in\_category – The category of the product. in\_quantity – The quantity of the product that counts as one unit (e.g., 4 spring rolls). in\_description – A description of the product.. in\_price – The cost of the product. in\_imagefilename – The name of the image file representing the product to be added onto the website. |
| **Example Usage:** | CALL insertProduct ('Golden Rolls','Appetisers',3,  'Traditional fried eggless roll filled with taro, cabbage, carrot and tofu',3.95,  'goldenRolls.jpg'); |
|  | |
| **Procedure Name:** | updateProduct |
| **Author:** | Andrew Laing |
| **Last Updated:** | 09/11/2018 |
| **Description:** | This procedure updates an product record. |
| **Parameters:** | in\_itemID – A product’s unique ID. in\_itemName – The name of the product. in\_category – The category of the product. in\_quantity – The quantity of the product that counts as one unit (e.g., 4 spring rolls). in\_description – A description of the product.. in\_price – The cost of the product. in\_imagefilename – The name of the image file representing the product to be added onto the website.. |
| **Example Usage:** | CALL updateProduct (1, 'Golden Rolls','Appetisers',3, 'Traditional deep-fried eggless roll filled with taro, cabbage, carrot and tofu',3.95, 'goldenRolls.jpg'); |
|  | |
| **Procedure Name:** | deleteProduct |
| **Author:** | Andrew Laing |
| **Last Updated:** | 09/11/2018 |
| **Description:** | This procedure deletes an product record. |
| **Parameters:** | in\_itemID – A product’s unique ID. |
| **Example Usage:** | CALL deleteProduct(2); |
|  | |
| **Procedure Name:** | getAllProductDetails |
| **Author:** | Andrew Laing |
| **Last Updated:** | 09/11/2018 |
| **Description:** | This procedure returns all product details, with the price formatted as #.## |
| **Parameters:** | None. |
| **Example Usage:** | CALL getAllProductDetails; |

|  |  |
| --- | --- |
| Mai-Lin Vegan Takeaway Restaurant Database customerFeedbackTBL procedures | |
| **Procedure Name:** | insertFeedback |
| **Author:** | Andrew Laing |
| **Last Updated:** | 09/11/2018 |
| **Description:** | This procedure inserts a customer feedback record. |
| **Parameters:** | in\_firstname – A customer’s given name  in\_surname – A customer’s surname in\_emailAddress – A customers email address in\_message – A feedback message |
| **Example Usage:** | CALL insertFeedback('John', 'Jones', 'jjones@gmail.com', 'I think your website is wonderful'); |
|  | |
| **Procedure Name:** | deleteFeedback |
| **Author:** | Andrew Laing |
| **Last Updated:** | 09/11/2018 |
| **Description:** | This procedure deletes a feedback record. |
| **Parameters:** | In\_messageID – The usinque identifier for a customer feedback record. |
| **Example Usage:** | CALL deleteFeedback(3); |

# User Guide

|  |
| --- |
|  |
| **Website User Guide**  Mai-Lin’s Vegan Takeaway Restaurant |

|  |
| --- |
| Contents |
| Introduction |
| Website Compatibility |
| Disclaimer |
| Frequently Asked Questions |

|  |
| --- |
| Introduction |
| This document will guide users through the common tasks which they will perform when using the Mai-Lin's Vegan Takeaway Restaurant website. It is by no means exhaustive and suggestions for content to be added to future versions are always welcomed.  The website was put online to allow our increasing number of customers to order food to be delivered to their homes. If you have any requests for vegan specialities not on the menu or special dietary requirements please telephone us on 0151 443 7777. |

|  |
| --- |
| Website Compatibility |
| This website is designed to be viewable on all modern laptop, desktop and mobile phone browsers including Chrome, Microsoft Edge, Safari, Firefox, and Internet Explorer versions greater than 8. Some of this site's functionality requires JavaScript. If JavaScript is not enabled for your browser, search Google for instructions on how to turn it on.  If you have any problems using this guide or need help with something not included in the pages below please email the website’s developer at [**parisianconnections@gmail.com**](mailto:parisianconnections@gmail.com)**.** |
| Disclaimer |
| Although we endeavour to keep the information on this website up to date and correct, we make no representations or warranties of any kind, express or implied, about the completeness, accuracy, reliability, suitability or availability with respect to the website or the information, products, services, or related graphics contained on the website for any purpose. Any reliance you place on such information is therefore strictly at your own risk. |
| In no event will we be liable for any loss or damage including without limitation, indirect or consequential loss or damage, or any loss or damage whatsoever arising from loss of data or profits arising out of, or in connection with, the use of this website. |
| Through this website you are able to link to other websites which are not under the control of Mai-Lin’s Vegan Takeaway Restaurant. We have no control over the nature, content and availability of those sites. The inclusion of any links does not necessarily imply a recommendation or endorse the views expressed within them. |
| Every effort is made to keep the website up and running smoothly. However, Mai-Lin’s Vegan Takeaway Restaurant takes no responsibility for, and will not be liable for, the website being temporarily unavailable due to technical issues beyond our control. |

Frequently Asked Questions

|  |  |  |
| --- | --- | --- |
| 1: | How do I create a new account? | |
|  | Click on the ***Sign Up*** link in the Navigation menu at the top of the page. |
|  | Enter details into the Registration form.  *i) A unique user name of 7 characters or longer must be used. ii) Password must be 7 characters or longer. iii) User must supply either a phone number or email address. iv) All other fields except Address Line 2 must be completed.* |
|  | Press the ***Submit*** button. |
|  | User will now be able to login with the credentials provided. |

|  |  |
| --- | --- |
| 2: | How can I log into an existing account? |
|  | Click on the ***Login*** link in the Navigation menu at the top of the page. |
|  | Enter user name and password details into the login form. |
|  | Press the ***Login*** button. |
|  | User will now be logged into their account. |

|  |  |
| --- | --- |
| 3: | How do I log out? |
|  | Click on the ***Log Out*** link in the Navigation menu at the top of the page. |
|  | A dialogue box will appear asking the user to confirm that they want to log out of their account. |
|  | Click ***OK*** on the dialogue box. |
|  | User will now be logged out of their account. |

|  |  |
| --- | --- |
| 4: | How can I recover a lost password? |
|  | Click on the ***Login*** link in the Navigation menu at the top of the page. |
|  | Click on the ***Forgot Password*** link on the login form. |
|  | The user’s mail client will open to allow them to email the website administrator to reset their password. |

|  |  |
| --- | --- |
| 5: | How do I leave feedback/comments? |
|  | Click on the ***Feedback*** link in the Navigation menu at the top of the page. |
|  | Enter details into the Feedback form. |
|  | Press the ***Submit*** button. |
|  | Feedback will be sent to the website administrator. |

|  |  |
| --- | --- |
| 6: | How can I view my user details? |
|  | Log into account. |
|  | Click on the ***Your Account*** link in the Navigation menu at the top of the page. |
|  | User will be able to view all details provided except their password. |

|  |  |
| --- | --- |
| 7: | How do I change my account details? |
|  | Log into account. |
|  | Click on the ***Your Account*** link in the Navigation menu at the top of the page. |
|  | Click on the ***Change your Account Details*** link at the bottom of the page. |
|  | Change account details on the form shown. |
|  | Press the ***Change Details*** button. |
|  | A dialogue box will appear asking the user to confirm that they want to change their details. |
|  | Click ***OK*** on the dialogue box. |
|  | The user’s details will be updated. |

|  |  |
| --- | --- |
| 8: | How can I change my password? |
|  | Log into account. |
|  | Click on the ***Your Account*** link in the Navigation menu at the top of the page. |
|  | Click on the ***Change your Password*** link at the bottom of the page. |
|  | Complete the fields on the Change Password form. |
|  | Press the ***Change Password*** button. |
|  | The user’s will now be able to log in next time with their new password. |
| 9: | How do I order food online? |
|  | Log into account. |
|  | Click on the ***Order Online*** link in the Navigation menu at the top of the page. |
|  | Click on the Menu Item buttons to change the Menu Items displayed on the page. |
|  | Multiple items can be added to the order by clicking on the ***Add to Basket*** button below their images. |
|  | Clicking ***Add to Basket*** again will increase the quantity of the item ordered. |
|  | Clicking the Subtract ( **̶** ) button will reduce the quantity of the item ordered. |
|  | Clicking the Remove (**X**) button will remove all instances of this item from the order. |
|  | Once all items have been chosen add the date and time of delivery to the basket on the right-hand side of the page (below the menu if the site is being viewed on a modile phone.) |
|  | Press the **Place Order** button. |
|  | The order will now be placed and will arrive at the time chosen. |
|  | Food will be paid for by the customer upon delivery. |

# 

# Testing

## Introduction

This section of the report contains details of the website project’s Alpha and Beta testing phases.

For the Alpha tests, a structured plan was created to discover common errors. This includes visual inspections of each page, link checks, boundary checks of input field, plus tests specific to the functionality of each page.

This report contains the Test Plan, Test Data used within the plan, and a Test Log of the Alpha testing results. The Test Log includes Screenshots to show that prototypes worked as expected.

Beta tests were performed at the end of the development cycle. Testers were recruited from a group of the developer’s peers. Each tester was given a two-page form to provide the developer with a mixture of quantitive and qualitative feedback about their experience of using the site. An example of this form is included in the report followed by a brief analysis of test results.

# 

## Test Plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Plan | | | | |
| **Project Name:**  Mai-Lin's Vegan Takeaway Restaurant Website | | | | |
| **IMPORTANT!**  Before starting the tests ensure that you have created a fresh installation of the database using *createFoodDB.sql* and have ran *addProductDetails.sql* and *addTestData.sql* to insert product details and test data. | | | | |
| **Test No.** | **Date** | **Purpose/Type of Test** | **Test Data** | **Expected Output/Filename** |
| Login Modal | | | | |
| 1 | 13/11/2018 | Functionality test 1. | Click the ‘**Login**’ link on the right-hand of the navigation bar on each page. | The Login Modal should be displayed. |
| 2 | 13/11/2018 | Visual inspection. | Open the Login Modal on each page that the ‘**Login**’ link appears in on the right-hand side of the navigation bar and inspect its content. | Modal is laid out correctly, the colour scheme is correct, spelling and punctuation are correct, and all text is clear and readable. |
| 3 | 13/11/2018 | Character display verification. | Enter any characters into the Password field. | The characters typed into the Password field should not be visible to the user. |
| 4 | 13/11/2018 | Username and Password field presence checks. | Leave both fields blank and press the ‘**Login**’ button. | An error message should be reported. |
| 5 | 13/11/2018 | Username field presence check. | Enter the password ‘**admin**’ into the Password field, leave the username field blank and press the ‘**Login**’ button. | An error message should be reported. |
| 6 | 13/11/2018 | Password field presence check. | Enter the username ‘**admin**’ into the Username field, leave the password field blank and press the ‘**Login**’ button. | An error message should be reported. |
| 7 | 13/11/2018 | Cancel button functionality test. | Press the ‘**Cancel**’ button. | The Login Modal should close. |
| 8 | 13/11/2018 | Invalid User Login test. | Enter Test Data 1 and press the ‘**Login**’ button. | An error message should be reported. |
| 9 | 13/11/2018 | Valid User Login test. | Enter Test Data 2 and press the ‘**Login**’ button. | The navigation bar links on the right-hand side should change to reflect the user’s logged in status showing ‘Your Account’ and ‘Logout’ links instead of ‘Sign up’ and ‘Login’. |
| 10 | 13/11/2018 | Link functionality test 1. | Click on the ‘**Sign Up**’ link on the modal. | The user should be redirected to the Registration page. |
| 11 | 13/11/2018 | Link functionality test 2. | Click on the ‘**Forgot Password**’ link on the modal. | The mail client should open for the user to send a password recovery email to the site administrator. |
| 12 | 13/11/2018 | Close button functionality test. | Click on the ‘**X**’ at the top-right of the Modal. | The Modal should close. |
| about.php | | | | |
| 13 | 13/11/2018 | Visual Inspection of page. | Open the page and inspect its content. | Page is laid out correctly, the colour scheme is correct, spelling and punctuation are correct, and all text is clear and readable. |
| 14 | 13/11/2018 | Page resize test. | Resize the page to portrait format or view on a mobile phone. | The page content should be repositioned to make best use of the available screen. |
| 15 | 13/11/2018 | Navigation bar functionality test. | Click on all the links on the left-hand side of the navigation bar and the ‘**Sign Up**’ link on the right-hand side. | The user should be redirected to the appropriate page. |
| 16 | 13/11/2018 | Social Media links test. | Click on the Facebook and Twitter buttons at the bottom of the page. | The user should be redirected to the appropriate website page. |
| changeAccountDetails.php | | | | |
| 17 | 13/11/2018 | Authorisation check. | Try to open the page without having logged into an account. | An error message will be shown instead of the page content, and the user will be redirected to the Home page after 5 seconds. |
| 18 | 13/11/2018 | Visual Inspection of page. | Log in as user ‘**scotty**’ with password ‘**password123**’. Open the page and inspect its content. | Page is laid out correctly, the colour scheme is correct, spelling and punctuation are correct, and all text is clear and readable. |
| 19 | 13/11/2018 | Page resize test. | Resize the page to portrait format or view on a mobile phone. | The page content should be repositioned to make best use of the available screen. |
| 20 | 13/11/2018 | Navigation bar functionality test. | Click on all the links on the left-hand side of the navigation bar and the ‘**Sign Up**’ link on the right-hand side. | The user should be redirected to the appropriate page. |
| 21 | 13/11/2018 | Social Media links test. | Click on the Facebook and Twitter buttons at the bottom of the page. | The user should be redirected to the appropriate website page. |
| 22 | 13/11/2018 | Cancel button functionality test. | Press the ‘**Cancel**’ button at the bottom of the form. | The user should be redirected to the Home page. |
| 23 | 13/11/2018 | Presence checks 1. | Using Test Data 3 fill in the fields three times and press the ‘**Change Details**’ button leaving one of the surname, first name and postcode fields blank each time. | An error message should be reported each time. |
| 24 | 13/11/2018 | Presence checks 2. | Using Test Data 3 fill in the fields leaving the phone number and email fields empty and press the ‘**Change Details**’ button. | An error message should be reported. |
| 25 | 13/11/2018 | Confirmation dialogue test 1. | Enter Test Data 3 into the fields and press the ‘**Change Details**’ button. | A confirmation dialogue should ask the user to confirm that they want to change their account details. |
| 26 | 13/11/2018 | Confirmation dialogue test 2. | After performing the previous test, when the confirmation dialogue appears choose ‘**Cancel**’. | The user should remain on the Change Account Details page. |
| 27 | 13/11/2018 | Confirmation dialogue test 3. | After performing the previous test, press the ‘**Change Details**’ button again but this time choose ‘OK’. | The user should be redirected to the Details Changed page which will confirm that their account details have been changed. |
| 28 | 13/11/2018 | Update record functionality test. | After performing the previous test, navigate to the ‘**Your Account**’ page, and check the **customerDetailsTBL** database table. | The details held about the logged in user should have changed to those entered. |
| changePassword.php | | | | |
| 29 | 13/11/2018 | Authorisation check. | Try to open the page without having logged into an account. | An error message will be shown instead of the page content, and the user will be redirected to the Home page after 5 seconds. |
| 30 | 13/11/2018 | Visual Inspection of page. | Login as user ‘**scotty**’ using password ‘**password123**’. Open the page and inspect its content. | Page is laid out correctly, the colour scheme is correct, spelling and punctuation are correct, and all text is clear and readable. |
| 31 | 13/11/2018 | Page resize test. | Resize the page to portrait format or view on a mobile phone. | The page content should be repositioned to make best use of the available screen. |
| 32 | 13/11/2018 | Navigation bar functionality test. | Click on all the links on the left-hand side of the navigation bar and the ‘**Sign Up**’ link on the right-hand side. | The user should be redirected to the appropriate page. |
| 33 | 13/11/2018 | Social Media links test. | Click on the Facebook and Twitter buttons at the bottom of the page. | The user should be redirected to the appropriate website page. |
| 34 | 13/11/2018 | Cancel button functionality test. | Press the ‘**Cancel**’ button at the bottom of the form. | The user should be redirected to the Home page. |
| 35 | 13/11/2018 | Presence checks. | Logged in as user ‘**scotty**’ with password ‘**password123**’. Using Test Data 4 fill in the fields three times and press the ‘**Change Password**’ button leaving one of the current password, new password and confirm password fields blank each time. | An error message should be reported each time. |
| 36 | 13/11/2018 | Current Password field length validation check. | Logged in as user ‘**scotty**’ with password ‘**password123**’. Using Test Data 5 fill in the fields and press the ‘**Change Password**’ button. | An error message should be reported. |
| 37 | 13/11/2018 | New Password field length validation check. | Logged in as user ‘**scotty**’ with password ‘**password123**’. Using Test Data 6 fill in the fields and press the ‘**Change Password**’ button. | An error message should be reported. |
| 38 | 13/11/2018 | Password confirm validation check. | Logged in as user ‘**scotty**’ with password ‘**password123**’. Using Test Data 7 fill in the fields and press the ‘**Change Password**’ button. | An error message should be reported. |
| 39 | 13/11/2018 | Invalid current password check. | Logged in as user ‘**scotty**’ with password ‘**password123**’. Using Test Data 8 fill in the fields and press the ‘**Change Password**’ button. | The user should be redirected to **passwordChanged.php** where an error message will show they have entered the current password incorrectly. |
| 40 | 13/11/2018 | Valid Password change test 1. | Logged in as user ‘**scotty**’ with password ‘**password123**’. Using Test Data 9 fill in the fields and press the ‘**Change Password**’ button. | The user should be redirected to **passwordChanged.php** where a message will show that the password has been changed. |
| 41 | 13/11/2018 | Valid Password change test 2. | After completing the previous test, logout then attempt to log back in using the new password. | The user should be able to log in correctly. |
| detailsChanged.php | | | | |
| 42 | 13/11/2018 | Authorisation check. | Try to open the page without having logged into an account. | An error message will be shown instead of the page content, and the user will be redirected to the Home page after 5 seconds. |
| 43 | 13/11/2018 | Visual Inspection of page. | Login with the username ‘**scotty**’ using the new password ‘**password456**’. Change account details to get this page to appear, inspect its content. | Page is laid out correctly, the colour scheme is correct, spelling and punctuation are correct, and all text is clear and readable. |
| 44 | 13/11/2018 | Page resize test. | Resize the page to portrait format or view on a mobile phone. | The page content should be repositioned to make best use of the available screen. |
| 45 | 13/11/2018 | Redirect Functionality test. | Open the page and do nothing. | The page will redirect to the Home page after 5 seconds if no links are clicked. |
| 46 | 13/11/2018 | Navigation bar functionality test. | Click on all the links on the left-hand side of the navigation bar and the ‘**Sign Up**’ link on the right-hand side. | The user should be redirected to the appropriate page. |
| 47 | 13/11/2018 | Social Media links test. | Click on the Facebook and Twitter buttons at the bottom of the page. | The user should be redirected to the appropriate website page. |
| errorPage.php | | | | |
| 48 | 13/11/2018 | Visual Inspection of page. | Open the page and inspect its content. | Page is laid out correctly, the colour scheme is correct, spelling and punctuation are correct, and all text is clear and readable. |
| 49 | 13/11/2018 | Page resize test. | Resize the page to portrait format or view on a mobile phone. | The page content should be repositioned to make best use of the available screen. |
| 50 | 13/11/2018 | Navigation bar functionality test. | Click on all the links on the left-hand side of the navigation bar and the ‘**Sign Up**’ link on the right-hand side. | The user should be redirected to the appropriate page. |
| 51 | 13/11/2018 | Social Media links test. | Click on the Facebook and Twitter buttons at the bottom of the page. | The user should be redirected to the appropriate website page. |
| feedback.php | | | | |
| 52 | 13/11/2018 | Visual Inspection of page. | Open the page and inspect its content. | Page is laid out correctly, the colour scheme is correct, spelling and punctuation are correct, and all text is clear and readable. |
| 53 | 13/11/2018 | Page resize test. | Resize the page to portrait format or view on a mobile phone. | The page content should be repositioned to make best use of the available screen. |
| 54 | 13/11/2018 | Navigation bar functionality test. | Click on all the links on the left-hand side of the navigation bar and the ‘**Sign Up**’ link on the right-hand side. | The user should be redirected to the appropriate page. |
| 55 | 13/11/2018 | Social Media links test. | Click on the Facebook and Twitter buttons at the bottom of the page. | The user should be redirected to the appropriate website page. |
| 56 | 13/11/2018 | Presence check. | With the message field empty, press the ‘**Submit**’ button. | An error message should be reported. |
| 57 | 13/11/2018 | Valid feedback test. | Using Test Data 10 fill in the fields and press the ‘**Submit**’ button. | The user should be redirected to the **sendMessage.php** page where a message will show that the feedback has been sent. The message will also appear in the **customerfeedbackTBL** table |
| index.php | | | | |
| 58 | 13/11/2018 | Visual Inspection of page. | Open the page and inspect its content. | Page is laid out correctly, the colour scheme is correct, spelling and punctuation are correct, and all text is clear and readable. |
| 59 | 13/11/2018 | Page resize test. | Resize the page to portrait format or view on a mobile phone. | The page content should be repositioned to make best use of the available screen. |
| 60 | 13/11/2018 | Navigation bar functionality test. | Click on all the links on the left-hand side of the navigation bar and the ‘**Sign Up**’ link on the right-hand side. | The user should be redirected to the appropriate page. |
| 61 | 13/11/2018 | Social Media links test. | Click on the Facebook and Twitter buttons at the bottom of the page. | The user should be redirected to the appropriate website page. |
| orderonline.php | | | | |
| 62 | 13/11/2018 | Visual Inspection of page. | Open the page and inspect its content. | Page is laid out correctly, the colour scheme is correct, spelling and punctuation are correct, and all text is clear and readable. Product details should be loaded onto the left-hand side of the page. |
| 63 | 13/11/2018 | Page resize test. | Resize the page to portrait format or view on a mobile phone. | The page content should be repositioned to make best use of the available screen. |
| 64 | 13/11/2018 | Navigation bar functionality test. | Click on all the links on the left-hand side of the navigation bar and the ‘**Sign Up**’ link on the right-hand side. | The user should be redirected to the appropriate page. |
| 65 | 13/11/2018 | Social Media links test. | Click on the Facebook and Twitter buttons at the bottom of the page. | The user should be redirected to the appropriate website page. |
| 66 | 13/11/2018 | Navigation pills test. | Click on each of the menu item links. | The details on the page below the menu items buttons should change according to the button pressed. |
| 67 | 13/11/2018 | Add to Basket button functionality test 1 | Click on the ‘**Add to Basket**’ button beneath any item. | The quantity of the item above the button should be updated, the item and its correct price should be added to the subtotal section on the right-hand side of the page and the price for the item added correctly to the total. |
| 68 | 13/11/2018 | Add to Basket button functionality test 2 | After completing the previous test press the same ‘**Add to Basket**’ button several more times. | The quantity of the item above the button should be updated correctly, as should the item’s entry in the subtotal section and the total. |
| 69 | 13/11/2018 | Add to Basket button functionality test 3 | After performing the previous test, click on another ’**Add to Basket**’ button beneath any other item. | The quantity of the item above the button should be updated, the item and its correct price should be added to the subtotal section on the right-hand side of the page and the price for the item added correctly to the total. |
| 70 | 13/11/2018 | Subtract from Basket button functionality test 1. | After performing the previous test press the ‘**Add to Basket**’ button several times more, then press the ‘**Subtract from Basket**’ button (‘**-**’) several times. | Each time the Subtract from Basket button is pressed one should be subtracted from the quantity above the button and the subtotal price for the item and total should be updated correctly. |
| 71 | 13/11/2018 | Subtract from Basket button functionality test 2. | After performing the previous test click the button until the item has zero in the quantity field above the button. | The item should not appear in the subtotal section. |
| 72 | 13/11/2018 | Subtract from Basket button boundary test. | After performing the previous test with quantity set to zero press the ‘**Subtract from Basket**’ button again. | Nothing should happen, and the quantity remain at zero. |
| 73 | 13/11/2018 | Remove from Basket functionality test 1. | Click on the ‘**Add to Basket**’ button beneath any item several times, then press the ‘**Remove from basket**’ button (‘**X**’). | When the Remove from Basket button is pressed, the quantity should be reset to zero, and the item removed from the subtotal with the total updated correctly. |
| 74 | 13/11/2018 | Remove from Basket functionality test 2. | Click on the ‘**Remove from Basket**’ button on any item with quantity shown as zero. | Nothing should happen. |
| 75 | 13/11/2018 | Date of Delivery Calendar functionality test. | Click within the ‘**Date of Delivery**’ field, | A calendar, or other selector depending upon the browser being used, should be shown to allow date selection. |
| 76 | 13/11/2018 | Time of Delivery Selector functionality test. | Click within the ‘**Time of Delivery**’ field. | A selector should be shown to allow time selection. |
| 77 | 13/11/2018 | Time of Delivery presence check. | With the ‘**Time of Delivery**’ field uncompleted, press the ‘**Place Order**’ button. | The order should not be submitted. A selector should be shown to allow time selection in the Time of Delivery field forcing the user to enter a time. |
| 78 | 13/11/2018 | Authorisation check. | Logout if currently logged in. Create an order, fill in a date and time in the future and press the ‘**Place Order**’ button. | The user will be redirected to **placeOrder.php** which will show an error message then redirect to the Home page after 5 seconds. |
| 79 | 13/11/2018 | Order Items presence check. | Fill in a date and time in the future but do not select any items the press the ‘**Place Order**’ button. | The user will be redirected to **placeOrder.php** which will show an error message then redirect back to the Order Online page after 5 seconds. |
| 80 | 13/11/2018 | Valid Order test 1. | Fill in a date and time in the future, select several items then the press the ‘**Place Order**’ button. | The user will be redirected to **placeOrder.php** confirm to the user that their order has been placed then redirect back to the Order Online page after 5 seconds. |
| 81 | 13/11/2018 | Valid Order test 2. | Examine the **ordersTBL** table in the database after completing the previous test. | A new record should be added with the correct details |
| 82 | 13/11/2018 | Valid Order test 3. | Examine the **orderItemsTBL** table in the database after completing the previous test. | New records should have been created for the items added to the order with the correct details. |
| passwordChanged.php | | | | |
| 83 | 13/11/2018 | Authorisation check. | Try to open the page without having logged into an account. | An error message will be shown instead of the page content, and the user will be redirected to the Home page after 5 seconds. |
| 84 | 13/11/2018 | Visual Inspection of page. | Login with the username ‘**scotty**’ using the new password ‘**password456**’. Open the page and inspect its content. | Page is laid out correctly, the colour scheme is correct, spelling and punctuation are correct, and all text is clear and readable. |
| 85 | 13/11/2018 | Page resize test. | Resize the page to portrait format or view on a mobile phone. | The page content should be repositioned to make best use of the available screen. |
| 86 | 13/11/2018 | Functionality test. | Open the page and do nothing. | The page will redirect to the Home page after 5 seconds if no links are clicked. |
| 87 | 13/11/2018 | Navigation bar functionality test. | Click on all the links on the left-hand side of the navigation bar and the ‘**Sign Up**’ link on the right-hand side before the page redirects to the Home page. | The user should be redirected to the appropriate page. |
| 88 | 13/11/2018 | Social Media links test. | Click on the Facebook and Twitter buttons at the bottom of the page. | The user should be redirected to the appropriate website page. |
| placeOrder.php | | | | |
| 89 | 13/11/2018 | Authorisation check. | Try to open the page without having logged into an account. | An error message will be shown instead of the page content, and the user will be redirected to the Home page after 5 seconds. |
| 90 | 13/11/2018 | Visual Inspection of page. | Login with the username ‘**scotty**’ using the new password ‘**password456**’. Place an order on the orderOnline page to open this page and inspect its content. | Page is laid out correctly, the colour scheme is correct, spelling and punctuation are correct, and all text is clear and readable. |
| 91 | 13/11/2018 | Page resize test | Resize the page to portrait format or view on a mobile phone. | The page content should be repositioned to make best use of the available screen. |
| 92 | 13/11/2018 | Redirect Functionality test. | Open the page and do nothing. | The page will redirect to the Home page after 5 seconds if no links are clicked. |
| 93 | 13/11/2018 | Navigation bar functionality test. | Click on all the links on the left-hand side of the navigation bar and the ‘**Sign Up**’ link on the right-hand side before the page redirects to the Home page. | The user should be redirected to the appropriate page. |
| 94 | 13/11/2018 | Social Media links test. | Click on the Facebook and Twitter buttons at the bottom of the page. | The user should be redirected to the appropriate website page. |
| privacy.php | | | | |
| 95 | 13/11/2018 | Visual Inspection of page. | Open the page and inspect its content. | Page is laid out correctly, the colour scheme is correct, spelling and punctuation are correct, and all text is clear and readable. |
| 96 | 13/11/2018 | Page resize test. | Resize the page to portrait format or view on a mobile phone. | The page content should be repositioned to make best use of the available screen. |
| 97 | 13/11/2018 | Navigation bar functionality test. | Click on all the links on the left-hand side of the navigation bar and the ‘**Sign Up**’ link on the right-hand side. | The user should be redirected to the appropriate page. |
| 98 | 13/11/2018 | Social Media links test. | Click on the Facebook and Twitter buttons at the bottom of the page. | The user should be redirected to the appropriate website page. |
| registered.php | | | | |
| 99 | 13/11/2018 | Authorisation check. | Try to open the page without having logged into an account. | An error message will be shown instead of the page content, and the user will be redirected to the Home page after 5 seconds. |
| 100 | 13/11/2018 | Visual Inspection of page. | Create a new account to open this page and inspect its content. | Page is laid out correctly, the colour scheme is correct, spelling and punctuation are correct, and all text is clear and readable. |
| 101 | 13/11/2018 | Page resize test. | Resize the page to portrait format or view on a mobile phone. | The page content should be repositioned to make best use of the available screen. |
| 102 | 13/11/2018 | Redirect Functionality test. | Open the page and do nothing. | The page will redirect to the Home page after 5 seconds if no links are clicked. |
| 103 | 13/11/2018 | Navigation bar functionality test. | Click on all the links on the left-hand side of the navigation bar and the ‘**Sign Up**’ link on the right-hand side before the page redirects to the Home page. | The user should be redirected to the appropriate page. |
| 104 | 13/11/2018 | Social Media links test. | Click on the Facebook and Twitter buttons at the bottom of the page. | The user should be redirected to the appropriate website page. |
| registration.php | | | | |
| 105 | 13/11/2018 | Visual Inspection of page. | Open the page and inspect its content. | Page is laid out correctly, the colour scheme is correct, spelling and punctuation are correct, and all text is clear and readable. |
| 106 | 13/11/2018 | Page resize test. | Resize the page to portrait format or view on a mobile phone. | The page content should be repositioned to make best use of the available screen. |
| 107 | 13/11/2018 | Navigation bar functionality test. | Click on all the links on the left-hand side of the navigation bar and the ‘**Sign Up**’ link on the right-hand side. | The user should be redirected to the appropriate page. |
| 108 | 13/11/2018 | Social Media links test. | Click on the Facebook and Twitter buttons at the bottom of the page. | The user should be redirected to the appropriate website page. |
| 109 | 13/11/2018 | Presence checks 1. | Using Test Data 11 fill in the fields three times and press the ‘**Submit**’ button leaving one of the username, password, surname, first name and address line 1 fields blank each time. | An error message should be reported each time. |
| 110 | 13/11/2018 | Presence checks 2. | Using Test Data 11 fill in the fields leaving the phone number and email fields empty and press the ‘**Submit**’ button. | An error message should be reported. |
| 111 | 13/11/2018 | Username field length validation check. | Using Test Data 12 fill in the fields and press the ‘**Submit**’ button. | An error message should be reported. |
| 112 | 13/11/2018 | Password field length validation check. | Using Test Data 13 fill in the fields and press the ‘**Submit**’ button. | An error message should be reported. |
| 113 | 13/11/2018 | Confirm Password validation check. | Using Test Data 14 fill in the fields and press the ‘**Submit**’ button. | An error message should be reported. |
| 114 | 13/11/2018 | AJAX test 1. | Enter the name ‘**scotty**’ into the username field and press tab. | The username field should be filled with the message ‘USERNAME ALREADY EXISTS!’ |
| 115 | 13/11/2018 | AJAX test 2. | Enter the name ‘**uniqueusername**’ into the username field and press tab. | The username field should remain the same. |
| 116 | 13/11/2018 | Unique username validation check. | Using Test Data 15 fill in the fields, ignoring the ‘**USERNAME ALREADY EXISTS!**’ message in the username field, and press the ‘**Submit**’ button. | An error message should be reported. |
| 117 | 13/11/2018 | Valid registration test 1. | Using Test Data 16 fill in the fields and press the ‘**Submit**’ button. | The user should be redirected to the **registered.php** page where a message will show that their account has been created. |
| 118 | 13/11/2018 | Valid registration test 2. | After completing the previous test, try to login with the credentials supplied. | The user should be able to login with the credentials they have entered. |
| sendMessage.php | | | | |
| 119 | 13/11/2018 | Visual Inspection of page. | Open the page and inspect its content. | Page is laid out correctly, the colour scheme is correct, spelling and punctuation are correct, and all text is clear and readable. |
| 120 | 13/11/2018 | Page resize test. | Resize the page to portrait format or view on a mobile phone. | The page content should be repositioned to make best use of the available screen. |
| 121 | 13/11/2018 | Redirect Functionality test. | Open the page and do nothing. | The page will redirect to the Home page after 5 seconds if no links are clicked. |
| 122 | 13/11/2018 | Navigation bar functionality test. | Click on all the links on the left-hand side of the navigation bar and the ‘**Sign Up**’ link on the right-hand side before the page redirects to the Home page. | The user should be redirected to the appropriate page. |
| 123 | 13/11/2018 | Social Media links test. | Click on the Facebook and Twitter buttons at the bottom of the page. | The user should be redirected to the appropriate website page. |
| termsandconditions.php | | | | |
| 124 | 13/11/2018 | Visual Inspection of page. | Open the page and inspect its content. | Page is laid out correctly, the colour scheme is correct, spelling and punctuation are correct, and all text is clear and readable. |
| 125 | 13/11/2018 | Page resize test. | Resize the page to portrait format or view on a mobile phone. | The page content should be repositioned to make best use of the available screen. |
| 126 | 13/11/2018 | Navigation bar functionality test. | Click on all the links on the left-hand side of the navigation bar and the ‘**Sign Up**’ link on the right-hand side. | The user should be redirected to the appropriate page. |
| 127 | 13/11/2018 | Social Media links test. | Click on the Facebook and Twitter buttons at the bottom of the page. | The user should be redirected to the appropriate website page. |
| userDetails.php | | | | |
| 128 | 13/11/2018 | Authorisation check. | Try to open the page without having logged into an account. | An error message will be shown instead of the page content, and the user will be redirected to the Home page after 5 seconds. |
| 129 | 13/11/2018 | Visual Inspection of page. | Login with the username ‘**scotty**’ using the new password ‘**password456**’. Open the page and inspect its content. | Page is laid out correctly, the colour scheme is correct, spelling and punctuation are correct, and all text is clear and readable. |
| 130 | 13/11/2018 | Page resize test. | Resize the page to portrait format or view on a mobile phone. | The page content should be repositioned to make best use of the available screen. |
| 131 | 13/11/2018 | Navigation bar functionality test. | Click on all the links on the left-hand side of the navigation bar and the ‘**Sign Up**’ link on the right-hand side. | The user should be redirected to the appropriate page. |
| 132 | 13/11/2018 | Social Media links test. | Click on the Facebook and Twitter buttons at the bottom of the page. | The user should be redirected to the appropriate website page. |

# 

# 

## Test Data

|  |  |  |
| --- | --- | --- |
| Test Data | | |
| **PROJECT NAME:** Mai-Lin's Vegan Takeaway Restaurant Website | | |
| **Ref No.** | **Date** | **Input Data** |
| 1 | 13/11/2018 | Username: hacker Password: invalid |
| 2 | 13/11/2018 | Username: scotty Password: password123 |
| 3 | 13/11/2018 | Surname: spock First name: admiral  Address line 1: 123 Star Street  Address line 2: Liverpool  Postcode: L18 2BP  Phone number: 1514454567  Email Address: spock@space.com |
| 4 | 13/11/2018 | Current Password: password123  New Password: password456  Confirm Password: password456 |
| 5 | 13/11/2018 | Current Password: pass  New Password: password456  Confirm Password: password456 |
| 6 | 13/11/2018 | Current Password: password123  New Password: pass  Confirm Password: pass |
| 7 | 13/11/2018 | Current Password: password123  New Password: password456  Confirm Password: wrongconfirm |
| 8 | 13/11/2018 | Current Password: notmypassword  New Password: password456  Confirm Password: password456 |
| 9 | 13/11/2018 | Current Password: password123  New Password: password456  Confirm Password: password |
| 10 | 13/11/2018 | First Name: Karlie  Last Name: Kloss  Email Address: cuteklossie@gmail.com  Message: Hey. I think your site is great, and you sell such lovely food. |
| 11 | 13/11/2018 | Username: iamsounique  Password: mypasswordisgreat  Confirm Password: mypasswordisgreat  Surname: McCarrigan  First Name: Sean  Address line 1: 53 Eccles Cake Road  Address line 2: Liverpool  Postcode: L8 2DR  Phone Number: 1516785667 Email Address: seanmac@gmail.com |
| 12 | 13/11/2018 | Username: god  Password: mypasswordisgreat  Confirm Password: mypasswordisgreat  Surname: McCarrigan  First Name: Sean  Address line 1: 53 Eccles Cake Road  Address line 2: Liverpool  Postcode: L8 2DR  Phone Number: 1516785667 Email Address: seanmac@gmail.com |
| 13 | 13/11/2018 | Username: iamsounique  Password: haha  Confirm Password: haha  Surname: McCarrigan  First Name: Sean  Address line 1: 53 Eccles Cake Road  Address line 2: Liverpool  Postcode: L8 2DR  Phone Number: 1516785667 Email Address: seanmac@gmail.com |
| 14 | 13/11/2018 | Username: iamsounique  Password: mypasswordisgreat  Confirm Password: whydoirepeat  Surname: McCarrigan  First Name: Sean  Address line 1: 53 Eccles Cake Road  Address line 2: Liverpool  Postcode: L8 2DR  Phone Number: 1516785667 Email Address: seanmac@gmail.com |
| 15 | 13/11/2018 | Username: scotty  Password: mypasswordisgreat  Confirm Password: mypasswordisgreat  Surname: McCarrigan  First Name: Sean  Address line 1: 53 Eccles Cake Road  Address line 2: Liverpool  Postcode: L8 2DR  Phone Number: 1516785667 Email Address: seanmac@gmail.com |
| 16 | 13/11/2018 | Username: iamsounique  Password: mypasswordisgreat  Confirm Password: mypasswordisgreat  Surname: McCarrigan  First Name: Sean  Address line 1: 53 Eccles Cake Road  Address line 2: Liverpool  Postcode: L8 2DR  Phone Number: 1516785667 Email Address: seanmac@gmail.com |

# 

# 

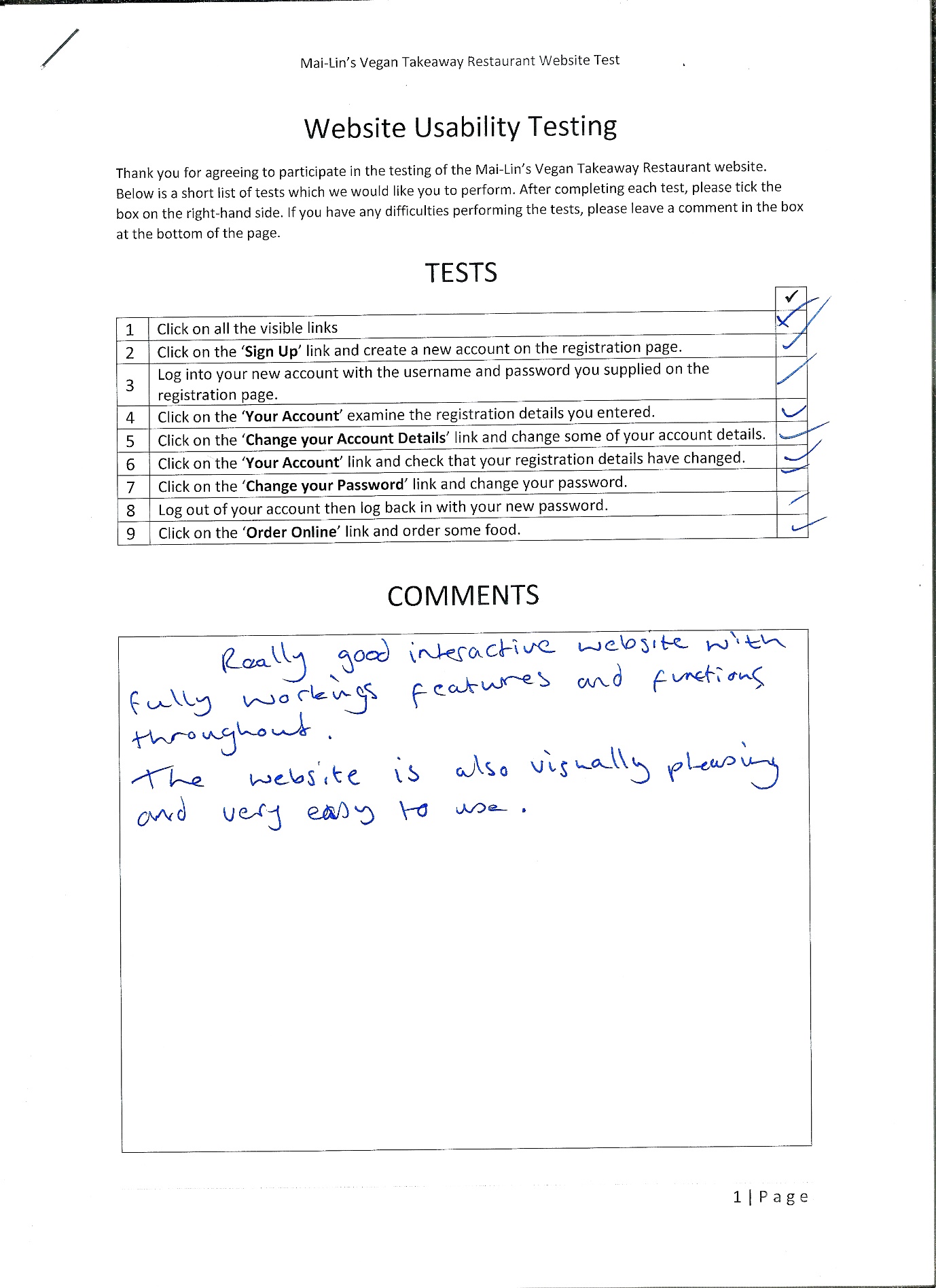
## Test Log

|  |  |  |  |
| --- | --- | --- | --- |
| Test Log | | | |
| **Project Name:** Mai-Lin's Vegan Takeaway Restaurant Website  **Tester:** Andrew Laing | | | |
| **Test No.** | **Date** | **Pass/Fail** | **Comments on discrepancies** |
| Login Modal | | | |
| 1 | 15/11/2018 | PASS |  |
| 2 | 15/11/2018 | PASS | See screenshot above. |
| 3 | 15/11/2018 | PASS |  |
| 4 | 15/11/2018 | PASS |  |
| 5 | 15/11/2018 | PASS |  |
| 6 | 15/11/2018 | PASS |  |
| 7 | 15/11/2018 | PASS | The ‘**Cancel**’ button worked as expected. |
| 8 | 15/11/2018 | PASS |  |
| 9 | 15/11/2018 | PASS |  |
| 10 | 15/11/2018 | PASS |  |
| 11 | 15/11/2018 | PASS |  |
| 12 | 15/11/2018 | PASS |  |
| about.php | | | |
| 13 | 15/11/2018 | PASS |  |
| 14 | 15/11/2018 | PASS | See screenshot above. |
| 15 | 15/11/2018 | PASS | All navigation links redirected to the correct pages. |
| 16 | 15/11/2018 | PASS |  |
| changeAccountDetails.php | | | |
| 17 | 15/11/2018 | PASS |  |
| 18 | 15/11/2018 | PASS |  |
| 19 | 15/11/2018 | PASS |  |
| 20 | 15/11/2018 | PASS | All navigation links redirected to the correct pages. |
| 21 | 15/11/2018 | PASS | The Social Media buttons worked as expected. |
| 22 | 15/11/2018 | PASS | The ‘**Cancel**’ button worked as expected. |
| 23 | 15/11/2018 | PASS |  |
| 24 | 15/11/2018 | PASS |  |
| 25 | 15/11/2018 | PASS |  |
| 26 | 15/11/2018 | PASS |  |
| 27 | 15/11/2018 | PASS |  |
| 28 | 15/11/2018 | PASS |  |
| changePassword.php | | | |
| 29 | 15/11/2018 | PASS |  |
| 30 | 15/11/2018 | PASS |  |
| 31 | 15/11/2018 | PASS |  |
| 32 | 15/11/2018 | PASS | All navigation links redirected to the correct pages. |
| 33 | 15/11/2018 | PASS | The Social Media buttons worked as expected. |
| 34 | 15/11/2018 | PASS | The ‘**Cancel**’ button worked as expected. |
| 35 | 15/11/2018 | PASS |  |
| 36 | 15/11/2018 | PASS |  |
| 37 | 15/11/2018 | PASS |  |
| 38 | 15/11/2018 | PASS |  |
| 39 | 15/11/2018 | PASS |  |
| 40 | 15/11/2018 | PASS |  |
| 41 | 15/11/2018 | PASS |  |
| detailsChanged.php | | | |
| 42 | 15/11/2018 | PASS |  |
| 43 | 15/11/2018 | PASS |  |
| 44 | 15/11/2018 | PASS |  |
| 45 | 15/11/2018 | PASS | The page redirected correctly to the Home page. |
| 46 | 15/11/2018 | PASS | All navigation links redirected to the correct pages. |
| 47 | 15/11/2018 | PASS | The Social Media buttons worked as expected. |
| errorPage.php | | | |
| 48 | 15/11/2018 | PASS |  |
| 49 | 15/11/2018 | PASS |  |
| 50 | 15/11/2018 | PASS | All navigation links redirected to the correct pages. |
| 51 | 15/11/2018 | PASS | The Social Media buttons worked as expected. |
| feedback.php | | | |
| 52 | 15/11/2018 | PASS |  |
| 53 | 15/11/2018 | PASS |  |
| 54 | 15/11/2018 | PASS | All navigation links redirected to the correct pages. |
| 55 | 15/11/2018 | PASS | The Social Media buttons worked as expected. |
| 56 | 15/11/2018 | PASS |  |
| 57 | 15/11/2018 | PASS |  |
| index.php | | | |
| 58 | 15/11/2018 | PASS |  |
| 59 | 15/11/2018 | PASS |  |
| 60 | 15/11/2018 | PASS | All navigation links redirected to the correct pages. |
| 61 | 15/11/2018 | PASS | The Social Media buttons worked as expected. |
| orderonline.php | | | |
| 62 | 15/11/2018 | PASS |  |
| 63 | 15/11/2018 | PASS |  |
| 64 | 15/11/2018 | PASS | All navigation links redirected to the correct pages. |
| 65 | 15/11/2018 | PASS | The Social Media buttons worked as expected. |
| 66 | 15/11/2018 | PASS | The navigation pills for the menu items worked as expected, updating the page content to show the relevant menu items. |
| 67 | 15/11/2018 | PASS |  |
| 68 | 15/11/2018 | PASS |  |
| 69 | 15/11/2018 | PASS |  |
| 70 | 15/11/2018 | PASS |  |
| 71 | 15/11/2018 | PASS |  |
| 72 | 15/11/2018 | PASS |  |
| 73 | 15/11/2018 | PASS |  |
| 74 | 15/11/2018 | PASS |  |
| 75 | 15/11/2018 | PASS |  |
| 76 | 15/11/2018 | PASS |  |
| 77 | 15/11/2018 | PASS |  |
| 78 | 15/11/2018 | PASS |  |
| 79 | 15/11/2018 | PASS |  |
| 80 | 15/11/2018 | PASS |  |
| 81 | 15/11/2018 | PASS |  |
| 82 | 15/11/2018 | PASS |  |
| passwordChanged.php | | | |
| 83 | 15/11/2018 | PASS |  |
| 84 | 15/11/2018 | PASS |  |
| 85 | 15/11/2018 | PASS |  |
| 86 | 15/11/2018 | PASS |  |
| 87 | 15/11/2018 | PASS | All navigation links redirected to the correct pages. |
| 88 | 15/11/2018 | PASS | The Social Media buttons worked as expected. |
| placeOrder.php | | | |
| 89 | 15/11/2018 | PASS |  |
| 90 | 15/11/2018 | PASS |  |
| 91 | 15/11/2018 | PASS |  |
| 92 | 15/11/2018 | PASS | The page redirected to the Home page as expected. |
| 93 | 15/11/2018 | PASS | All navigation links redirected to the correct pages. |
| 94 | 15/11/2018 | PASS | The Social Media buttons worked as expected. |
| privacy.php | | | |
| 95 | 15/11/2018 | PASS |  |
| 96 | 15/11/2018 | PASS |  |
| 97 | 15/11/2018 | PASS | All navigation links redirected to the correct pages. |
| 98 | 15/11/2018 | PASS | The Social Media buttons worked as expected. |
| registered.php | | | |
| 99 | 15/11/2018 | PASS |  |
| 100 | 15/11/2018 | PASS |  |
| 101 | 15/11/2018 | PASS |  |
| 102 | 15/11/2018 | PASS | The page was redirected to the Home page as expected. |
| 103 | 15/11/2018 | PASS | All navigation links redirected to the correct pages. |
| 104 | 15/11/2018 | PASS | The Social Media buttons worked as expected. |
| registration.php | | | |
| 105 | 15/11/2018 | PASS |  |
| 106 | 15/11/2018 | PASS |  |
| 107 | 15/11/2018 | PASS | All navigation links redirected to the correct pages. |
| 108 | 15/11/2018 | PASS | The Social Media buttons worked as expected. |
| 109 | 15/11/2018 | PASS |  |
| 110 | 15/11/2018 | PASS |  |
| 111 | 15/11/2018 | PASS |  |
| 112 | 15/11/2018 | PASS |  |
| 113 | 15/11/2018 | PASS |  |
| 114 | 15/11/2018 | PASS |  |
| 115 | 15/11/2018 | PASS |  |
| 116 | 15/11/2018 | PASS |  |
| 117 | 15/11/2018 | PASS |  |
| 118 | 15/11/2018 | PASS |  |
| sendMessage.php | | | |
| 119 | 15/11/2018 | PASS |  |
| 120 | 15/11/2018 | PASS |  |
| 121 | 15/11/2018 | PASS | The page redirected to the Home page as expected. |
| 122 | 15/11/2018 | PASS | All navigation links redirected to the correct pages. |
| 123 | 15/11/2018 | PASS | The Social Media buttons worked as expected. |
| termsandconditions.php | | | |
| 124 | 15/11/2018 | PASS |  |
| 125 | 15/11/2018 | PASS |  |
| 126 | 15/11/2018 | PASS | All navigation links redirected to the correct pages. |
| 127 | 15/11/2018 | PASS | The Social Media buttons worked as expected. |
| userDetails.php | | | |
| 128 | 15/11/2018 | PASS |  |
| 129 | 15/11/2018 | PASS |  |
| 130 | 15/11/2018 | PASS |  |
| 131 | 15/11/2018 | PASS | All navigation links redirected to the correct pages. |
| 132 | 15/11/2018 | PASS | The Social Media buttons worked as expected. |

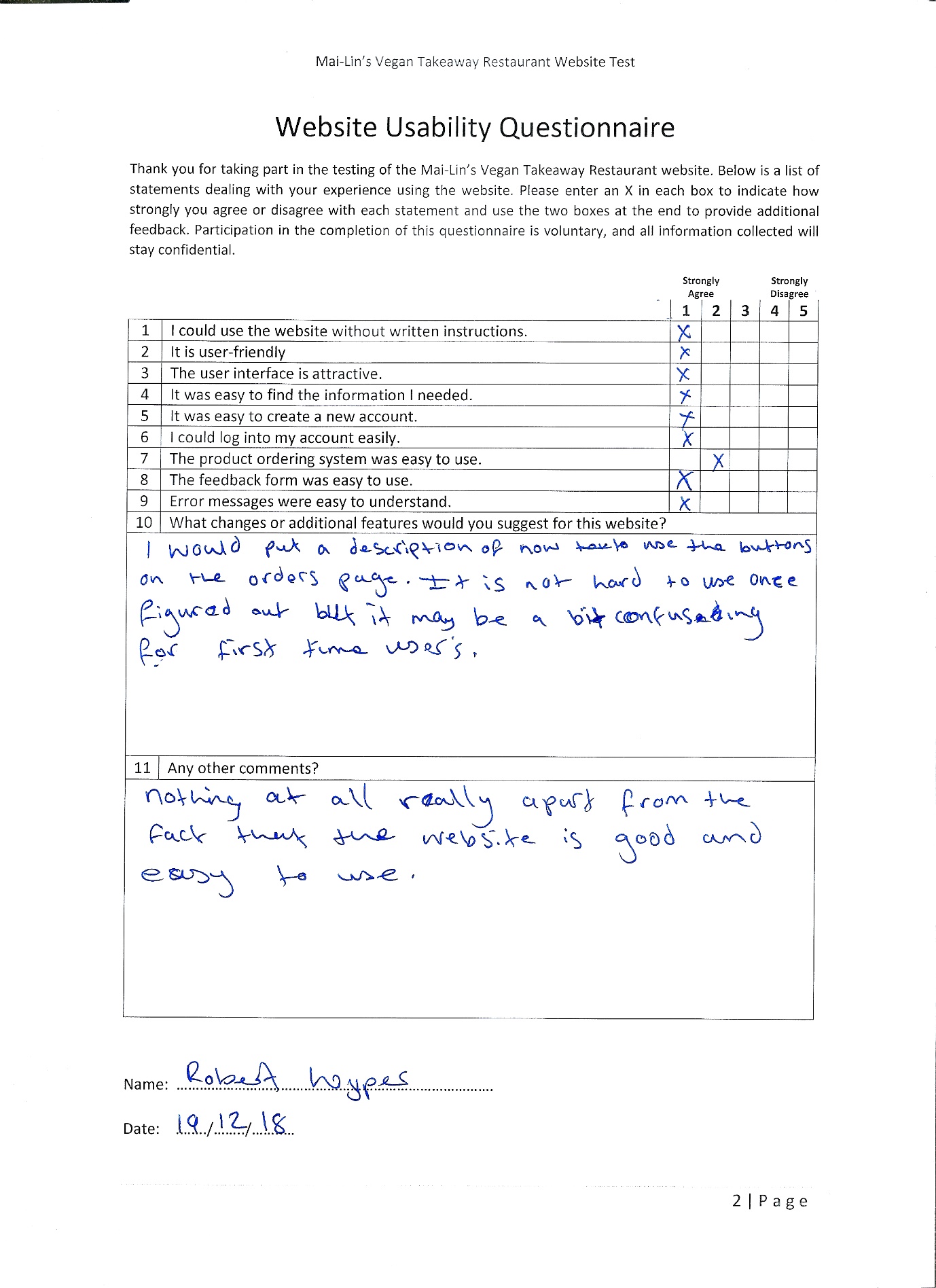
# 

## Example User Feedback Form

### Beta Testing Form (page 1)



### Beta Testing Form (page 2)



## Summary

The Test Plan was devised before actual development started. It was used as a guide during work upon the prototype. Although no formal tests were performed during coding, as new features were added these tests would be run to ensure that errors had not been introduced into the working version.

When the time arrived for formal Alpha testing, each test detailed within the Test Plan had already been performed multiple times. As a result, no errors were found.

The developer feels that the Alpha tests, although comprehensive, lack coverage in certain areas. Unit tests should have been created for the database procedures alongside stress testing and input field fuzzing to look for vulnerabilities.

Vulnerability testing is important to websites that hold customer information. Businesses now have a legal obligation to protect personal data under the General Data Protection Regulation.

However, as the current version of the website is only a prototype, this remains something for future consideration.

Generally, Beta testers provided the website with positive feedback. They found no faults during testing and questionaire scores were consistently high. Comments for improvements focussed mainly upon two areas, the sites lack of multimedia content and difficulties understanding the order form.

A lack of multimedia can lose a visitor’s attention quickly, especially with so many alternative websites to choose from. Choosing multimedia content for a website is something which requires an indepth discussion with business owners. However, adding animations to banners and food images could easily be implemented using JQuery and CSS3. This could make the site a little more interesting to look at.

With regards to the order page, the developer had wrongly assumed that his item selection button designs would be simple enough to use without guidance. Users had to think about what the purpose of the “*Subtract*” and “*Remove*” were and try them before discovering how they worked.

Testers also pointed out that the order total and place order button were difficult to find when the site was visited on a mobile phone. This would perhaps require the creation of a specialised Android or iOS Apple application which could also take advantage of the phones touchscreen capabilities.

Beta tester comments provided the developer with good ideas to incorporate into future projects. These comments were considered carefully and potential solutions included into the following section which evaluates the final prototype.

# 

# Evaluation

## Evaluation

This report began by outlining eleven objectives which the prototype website must meet. The following section will examine whether these objectives were met and provide suggestions for possible improvement.

### Has the prototype met the company’s objectives?

1. *The website is to provide relevant and current information about the company’s products. It will also present customers with some background information about the company’s vegan philosophy.*

An ‘*About Us*’ page was created to meet this objective. About pages are a standard feature of modern websites, so most customers will intuitively use them when they need to find more information about the site’s owners.

The page contains short paragraphs which quickly get across information to even the most hurried of site visitors. This page could nevertheless benefit from the inclusion of multimedia to reach those with shorter attention spans.

HTML5 allows developers to add videos compatible with a wide-range of devices and operating systems using a single line of code. This could be used to add an interview with the chef, giving customers a clearer insight into her background, philosophy, and the reasons why she chose to dedicate her life to cooking vegan food.

Another suggestion is to add a page which contains a list of links to related businesses such as vegan-friendly clothing stores, and websites that explore veganism and the history of Cambodian refugees who came to Britain in greater depth.

1. *The site must adapt to a wide range of devices. Today, customers are just as likely to be ordering food from their mobile phones as from laptops or desktops.*

Using HTML5 and the Bootstrap framework allows pages to adapt to the specifications of many devices. Although a full range of modern devices was not available to the developer, when tested on laptops, desktops and an android mobile phone the prototype website’s pages scaled pictures and texts well.

Testing found that different browsers implement the calendar and time picker components of the Order Online page in their own specific ways. Although most of these implementations were intuitive to use, certain testers found Microsoft Edge’s versions unattractive and harder to understand.

Cross-browser implementations of these features can be implemented using external JQuery or Bootstrap plugins. External libraries and plugins can also be used to add button, link and page loading animations. Certain of these features can also be implemented using customised CSS3 scripts.

1. *Pages throughout the website must have a consistent look and feel.*

Using customised HTML and CSS has ensured that the website’s look will remain consistent throughout. All pages include the *style.css* stylesheet to organise text and image elements. Using this single stylesheet allows changes to be made quickly and consistently across the site from a development environment such as Adobe Dreamweaver.

Although the site’s design is attractive and well laid out, it would definitely benefit from the input of a professional graphic designer. An experienced designer may, for example, incorporate fonts, colours and motifs from the restaurant’s décor into the site. Simple touches added by a trained professional artist could help to convey the business owner’s ideas more directly and stimulate visitor appetites whilst they look at the food images.

1. *Content must be easy to find, and visitors must always know what page they are on.*

The header section of each webpage contains the page’s title and a simple navigation bar. Most modern websites follow this design, so customers will be familiar with using it.

Navigation bar links are changed according to a user’s login status. They are created dynamically by a PHP script so that, for example, when a user logs into their account a link shown to an account administration page.

At present the Order Online page can be viewed by site visitors without accounts although they cannot place orders. This design decision was made so that visitors can view the food and try out the order page before they decide to create an account.

Although this design can encourage new registrations, the process may be stressful to some people. After filling in an order, casual visitors may expect to be redirected towards a credit card payment page.

It would be a good idea to include some information at the top of the page to let visitors and verified customers remember that they need to be logged in before placing orders, or to create a page which shows a simple menu.

1. *Visitors should be allowed to send comments, feedback and requests for further information to the website administrator.*

The website’s current feedback page is simple to use. It allows anonymous feedback to be sent using a MySQL stored procedure which prevents malicious users from using SQL injection attacks against the database server.

Storing feedback inside a MySQL table is not however the best option available as a custom user interface may need to be written for the restaurant’s customer services manager. PHP’s mail function will allow feedback to be sent as an email to a Microsoft Outlook or Google Gmail account instead. Customer feedback can then be dealt with alongside regular mail.

Elsewhere on the website, clicking email links will open the visitor’s email client. Some visitors find that these clients are too difficult to set up, so they abandon the mail sending process. Valuable information could be lost because of this such as notifying the administrator when the website is unavailable within a specific region. A custom error report form could be added for such cases using HTML5, Bootstrap and the PHP mail function.

1. *The website will allow new customers to create an account.*

The account creation procedure needs to be kept as simple as possible so that potential customers do not abandon halfway. The prototype website uses a registration form reached via links on the navigation bar and login modal.

The information gathered has been kept as minimal as possible to comply with the General Data Protection Regulation. Upon submission AJAX and JavaScript validation checks are performed upon the customer’s entered details. Pop up dialogue boxes let customers know when they make common mistakes such as leaving required fields empty or have tried to use a password less than seven characters long.

The AJAX scripts which validate the user name field provide immediate feedback if customers try to register with a duplicate user name. This runs as a background request so that the user is not disturbed.

It may have been a better idea to implement certain field validations using HTML5. These validation methods can be added with a single line of code and will allow developers to use powerful regular expressions to check field contents more closely before submission.

Using HTML5 methods will reduce the need for external JavaScript scripts which users can easily disable within their browsers. HTML5 cannot however implement all functionality required by this website. It cannot determine, for example, if one of two fields had been completed, and the prototype website’s design required that the user would be able to use either their email or phone number for direct communications with the delivery service.

1. *Allow existing customers to edit their account details.*

After logging into their account customers can edit their registration details. This functionality was added using PHP to populate editable input fields with the results of a database query.

The details change form is easy to use. Changes are validated using the same methods as the registration form, so additional suggestions provided above also apply to this form.

It may prove a good idea to add a password field which confirms that account detail changes are being made by an authorised user. Although not entirely necessary, many websites such as Google and Facebook implement changes to details in this manner. Large websites provide users with the standards to expect from online businesses, so by meeting these standards a website’s professionalism will be impressed upon its visitors.

1. *Allow existing customers to change their password.*

Password changes are made via an HTML form which immediately updates a customers’ account details. To confirm that an authorised user is making the change, customers must first enter and confirm their current password.

The form is easy to use, and JavaScript validation helps to ensure that little can go wrong with the process.

1. *The website will allow customers to log into and out of their accounts.*

The login modal is well designed and easy to use. It was implemented using the Bootstrap library and can be accessed on each page via the navigation bar.

When a customer logs into their account, a PHP session is established to give them access to more options on the navigation bar and allows them to make orders for food.

It may be a good idea to display a JavaScript dialogue or welcome page after login. Apart from a change to the navigation bar, there is currently no clear action which informs a customer that their login was successful. Logging out is however more user-friendly as it uses confirmation dialogues.

Overall, login functionality has been well-implemented and using session cookies stored on the server rather than on the user’s machine helps to reduce the website’s vulnerability to session hijacking attacks.

1. *The website will allow registered customers to order food online. Customers will be able to make orders with the minimum amount of guidance. Multiple Items will be added to or removed from the shopping basket with a single click. Customers will be able to view the contents of their basket along with a running total before they place their order. The overall shopping experience should be smooth and professional.*

The shopping basket adapts well to the screen space provided by different devices. It has a clear, tidy design and product images which scale well when the screen is resized.

JQuery animations could be used here to make clicking the ‘*Add to basket*’ buttons more fun to use. Other buttons such as ‘Subtract from Basket’ and ‘Remove from Basket’ could also use tooltips that explain their function. However, their current design should make them intuitive enough for most visitors to understand.

Ordered items and the running total are currently pushed to the bottom of phone screens making it hard to keep track of selections made. For these devices a JQuery plugin which allows a swipe right to show this total followed by a left swipe to move back to the main menu could improve the user’s experience.

Once customers have placed an order, they are unable to change it. Giving customers the ability to update or delete orders will give them more control over their shopping experience and make it more pleasurable.

One possible solution is to add more ‘*Your Account*’ pages which allow orders to be edited or cancelled up to one hour before delivery. A related feature could be implemented to let visitors view previous orders and repeat them, for example every week on the same day.

An unforeseen problem with the order page that could have legal consequences is its lack allergy advice. This could be rectified with a simple search box that allows customers to remove products from the menu which contain allergens such as nuts or sesame seeds.

Before the site is deployed, it would be recommended to add an online payment solution so that customers can make large repeat orders and delivery staff do not have to carry around large amounts of money.

1. *The website will deal with customer errors and system errors in a friendly manner which does not expose any sensitive information.*

The site deals with error messages using JavaScript dialogues and a custom PHP error page which does not expose details about the site’s implementation. Although the information disclosed in such messages is harmless, it can provide attackers with a basis for planning further attacks.

Currently, when non-critical errors occur no information is stored about them. Unless customers are willing to send the website administrator an email they may pass unnoticed.

Errors logging is easily configured database and PHP servers, but if an automatic email could be forwarded to the administrator using PHP’s mail function, this could provide enough timely information to correct common mistakes and catch cyber-attacks in progress.

Conclusion   
Aside from a little remodelling and the addition of a few extra pages the prototype website is fit for purpose. It has room for multimedia content and advertising, and its use of external stylesheets and JavaScript files will ensure that it is easy to extend and maintain.

Adding additional user account management functionality could improve the overall visitor experience as would the inclusion of dynamic content implemented using the JQuery and Bootstrap libraries.

The technologies chosen to create this site were used intelligently, and besides meeting its set objectives the final prototype could well serve as a good template for future website development projects.

# Bibliography

Carey, P. (2017) New Perspectives on HTML5 and CSS3. 7th ed. Boston: Cengage Learning.

Crockford, D. (2011) JavaScript: The Good Parts. 3rd ed. Sebastapol: O'Reilly.

Felke-Morris, T. (2015) Basics of Web Design: HTML5 & CSS3. 3rd ed. London: Pearson.

Franklin, J. & Ferguson, R. (2017) Beginning JQuery. 3rd ed. New York: Apress.

Holzner, S. (2009) AJAX: A Beginner's Guide. 1st ed. New York: McGraw-Hill.

Isherwood, M. (2017) Designing Ecommerce Websites. 1st ed. London: Transmitter Press.

Jorgensen, P. C. (2008) Software Testing: A Craftsman's Approach. 3rd ed. New York: Auerbach Publications.

Lambert, M. (2016) Learning Bootstrap 4. 2nd ed. Birmingham: Packt Publishing.

Moreto, S., Lambert, M., Jakobus, B. & Marah, J. (2016) Bootstrap 4: Responsive Web Design. 1st ed. Birmingham: Packt Publishing.

Schneider, R. D. (2005) MySQL Database Design and Tuning. 1st ed. New York: Pearson.

Welling, L. & Thomson, L. (2009) PHP and MySQL Web Development. 4th ed. New York: Pearson.

# 

# Appendix A Sample Code Listings 1 Login Functionality

## Implementing ‘Login’ functionality

Login functionality is implemented on each page by adding a link to the navigation bar dynamically with PHP when the user is logged into their account;

<ul class=**"nav navbar-nav navbar-right"**>

<!-- Generate links with PHP according to the user’s login status -->

<?php

**if**( !(**isset**($\_SESSION['login']) && $\_SESSION['login'] != '') ) {

**echo** '<li><a href="registration.php">Sign Up</a></li>';

**echo** '<li><a href="#" id="loginBtn">Login</a></li>';

**session\_unset**();

}

**else** {

**echo** '<li><a href="userDetails.php">Your Account</a></li>';

**echo** '<li><a href="javascript:confirm\_logout();">  
 Log out</a></li> ';

}

?>

When the link is clicked an anonymous JavaScript function written using the syntax of the JQuery library will be called;

<script>

$**(**document**).**ready**(*function*(){**

$**(**"#loginBtn"**).**click**(*function*(){**

$**(**"#loginModal"**).**modal**();**

**});**

**});**

</script>

This function will reveal a hidden modal login form included on each page;

<!-- Login Modal -->

<div class=**"modal fade"** id=**"loginModal"** role=**"dialog"**>

<div class=**"modal-dialog"**>

<!-- Modal content-->

<div class=**"modal-content"**>

<div class=**"modal-header"** style=**"padding:35px 50px;"**>

<!-- Close button -->

<button type=**"button"** class=**"close"**   
 data-dismiss=**"modal"**>*&times;*</button>

<h3>**Login to your Account** </h3>

</div>

<!—Login form -->

<div class=**"modal-body"** style=**"padding:40px 50px;"**>  
 <!—When submit is clicked POST to this page -->

<form role=**"form"** name=**"frmLogin"** method=**"post"**   
 action=**"**<?php **echo** **htmlspecialchars**($\_SERVER["PHP\_SELF"]); ?>**"**>

<div class=**"form-group"**>

<label for=**"username"**>**Username**</label>

<input type=**"text"** class=**"form-control"** id=**"username"**   
 name=**"username"** placeholder=**"Enter username"**>

<br />

<label for=**"password"**></span> **Password**</label>

<input type=**"password"** class=**"form-control"** id=**"pwd"**   
 name=**"pwd"** placeholder=**"Enter password"**>

</div>

<div class=**"modal-login-btn"**>  
 <!-- The submit button validates form details before 🡪  
 <!—POSTing using the JavaScript validate\_login() function -->

<button type=**"submit"** class=**"btn btn-success btn-block"**   
 name=**"LoginBtn"** value=**"LoginBtn"**   
 onClick=**"return validate\_login()"**>

**Login**</button>

</div>

</form>

</div>

<div class=**"modal-footer"**>

<button type=**"submit"** class=**"btn btn-danger btn-default pull-left"**   
 data-dismiss=**"modal"**>**Cancel**</button>  
 <!-- Link to registration and mailto for forgotten password -->

<p>**Not a member?** <a href=**"registration.php"**>**Sign Up**</a></p>

<p>**Forgot** <a href=**"mailto:admin@mailinsrestaurant.com?Subject=Forgotten%20Password"**>  
 **Password?**</a>  
 </p>

</div>

</div><!-- End of .modal-content -->

</div><!-- End of .modal-dialog -->

</div><!-- End of #loginModal -->

When the user presses submit a JavaScript function will first check that the username and password fields have been completed;

/\* The function validate\_login check that both login field have   
 \* been completed \*/

**function** validate\_login**()**

**{**

**var** usr **=** **document.forms.**frmLogin**.**username**.**value**;**

**var** pwd **=** **document.forms.**frmLogin**.**pwd**.**value**;**

**var** usrLen **=** usr**.**length**;**

**var** pwdLen **=** pwd**.**length**;**

**if(**usrLen**===**0 **||** pwdLen**===**0**)** **{**

**alert(**"Error - both fields must be completed!"**);**

**return** **false;**

**}**

**return** **true;**

**}**

If both field have been completed correctly a PHP script included at the top of each page will check that a valid username/password combination has been entered. If all is correct session variables can be set, otherwise a JavaScript error message dialogue will be shown;

<?php

/\* The function checkPasswordIsValid checks whether a username/password

\* combination exists in the database. If yes it returns the users   
 \* customerID, otherwise it returns -1 \*/

**function** checkPasswordIsValid($usr, $pwd) {

/\* Connect to the DB \*/

$mysqli = **new** mysqli("localhost","webuser","webuser","fooddb");

$res = -1;

/\* check connection \*/

**if** (**mysqli\_connect\_errno**()) {  
 /\* If there was an error redirect to the error page \*/

**header**('Location: errorPage.php');

**exit**();

}

**else** {  
 /\* call the stored MySQL procedure to check the user has entered  
 \* a valid username/password combination \*/

$SQL = $mysqli->prepare('CALL checkPasswordIsValid(?,?)');

$SQL->bind\_param('ss', $usr, $pwd);

$SQL->execute();

$result = $SQL->get\_result();

**if**($result->num\_rows == 1) {

$db\_field = $result->fetch\_assoc();

/\* If a valid username/password combination has been sent then  
 \* set res to the returned customer ID \*/

$res = $db\_field['customerID'];

}

**else** { /  
 /\* Show an error message dialogue and destroy the session \*/

**echo** ' <script>alert("Error: Unable to login with the credentials   
 supplied.");</script>';

}

}

$mysqli->close();

**return** $res;

}

/\* If there is not an active session start one \*/

**if** (**session\_status**() == **PHP\_SESSION\_NONE**) {

**session\_start**();

}

/\* If the user is not logged in clear session variables \*/

**if**( !(**isset**($\_SESSION['login']) && $\_SESSION['login'] != '') ) {

/\* Clear session variables if set \*/

**session\_unset**();

}

/\* If the user has pressed the submit button to reload the page   
 \* perform the login check \*/

**if**( $\_SERVER["REQUEST\_METHOD"] == "POST" ) {

/\* If the login button was pressed, check for a valid userrname/password combination \*/

**if** (**isset**($\_POST['LoginBtn']) &&   
 **isset**($\_POST['username']) &&   
 **isset**($\_POST['pwd'])) {

/\* get the login details entered \*/

$usr1= $\_POST['username'];

$pwd = $\_POST['pwd'];

/\* check that the details are valid \*/

$customerID = checkPasswordIsValid($\_POST['username'],$pwd);

**if**($customerID != -1) {

/\* Set the Session variables for a valid login \*/

**session\_unset**();

$\_SESSION['login'] = "1";

$\_SESSION['uname'] = $usr1;

$\_SESSION['customerID'] = $customerID;

**header**($\_SERVER["PHP\_SELF"]);

}

**else** {

// remove all session variables

**session\_unset**();

**session\_destroy**();

**header**($\_SERVER["PHP\_SELF"]);

}

}

**else** {

**echo** ' <script>alert("Another button was pressed!!!!");</script>';

}

}

?>

The checkPasswordIsValid stored procedure called in the above script first converts the password sent to it into a SHA1 hash so that it can compare it against the one stored for the user in the database. It will then return all rows which match the username/password combination;

-- Create the checkPasswordIsValid procedure

**DROP** **PROCEDURE** **IF** **EXISTS** checkPasswordIsValid**;**

DELIMITER $$

**CREATE** **PROCEDURE** checkPasswordIsValid **(** **IN** in\_username **VARCHAR(**35**),**

**IN** in\_password **VARCHAR(**255**)** **)**

**BEGIN**

**SET** @password\_hash **=** sha1**(**in\_password**);**

**SELECT** customerID **FROM** customerDetailsTBL

**WHERE** username**=**in\_username

**AND** password**=**@password\_hash**;**

**END** $$

DELIMITER **;**

## Implementing ‘Log Out’ functionality

Log out functionality is implemented on each page by adding a link to the navigation bar dynamically with PHP when the user is logged into their account;

<ul class=**"nav navbar-nav navbar-right"**>

<!-- Generate links with PHP according to the user’s login status -->

<?php

**if**( !(**isset**($\_SESSION['login']) && $\_SESSION['login'] != '') ) {

**echo** '<li><a href="registration.php">Sign Up</a></li>';

**echo** '<li><a href="#" id="loginBtn">Login</a></li>';

**session\_unset**();

}

**else** {

**echo** '<li><a href="userDetails.php">Your Account</a></li>';

**echo** '<li><a href="javascript:confirm\_logout();">  
 Log out</a></li> ';

}

?>

When the link is clicked a JavaScript function will be called to confirm that the user wishes to log out of their account. If the user clicks ‘***OK’*** the browser will redirect them to logout.php.

/\* The function confirm\_logout is used by each page’s log out link to confirm that the user wishes to log out of their account. \*/

**function** confirm\_logout**()**

**{**

/\* Confirm from the user that they want to log out \*/

**var** r **=** **confirm(**"Are you sure you want to log out of your account."**);**

**if** **(**r **==** **true)**

**{** /\* Show the user a message dialogue\*/

**alert(**"You are now logged out."**);** /\* Redirect the user to logout.php \*/

**window.location** **=** "logout.php"**;**

**}**

/\* Else just return to the current page without logging off \*/

**else**

**{**

**return** **false;**

**}**

**}**

The logout.php page contains a short script which will destroy the current session and redirect the user back to the home page.

<?php

/\* If there is no active session start one \*/

**if** (**session\_status**() == **PHP\_SESSION\_NONE**) {

**session\_start**();

}

/\* Unset Session cookie values \*/

**session\_unset**();

/\* Return to the home page \*/

$headerText = 'Refresh:0; url="index.php"';

**header**($headerText);

**exit**;

?>

# 

# Appendix B Sample Code Listings 2 Food Ordering Functionality

## Implementing the ‘Order Online’ functionality

When orderOnline.php is opened it calls a stored MySQL procedure to get all of the product details and adds them into an array;

<?php

/\* Connect to the DB, get the product details and add them to an array \*/

$mysqli = **new** mysqli("localhost","webuser","webuser","fooddb");

**if** ($mysqli->connect\_errno) {

/\* Redirect to error page here \*/

**header**('Location: errorPage.php');

**exit**;

}

/\* Run the query \*/

**if** (! ($query = $mysqli->query("CALL getAllProductDetails()"))) {

/\* Redirect to error page here \*/

**header**('Location: errorPage.php');

**exit**;

}

**else** {

$result = [];

$counter = 1;

/\* Add product details to an array \*/

**while**($row = $query->fetch\_assoc()) {

$result[$counter] = $row;

$counter++;

}

}

$mysqli->close();

?>

The getAllProductDetails procedure refered to above is a simple SELECT statement which also formats the product price to two decimal places.

-- Create the getAllProductDetails procedure

**DROP** **PROCEDURE** **IF** **EXISTS** getAllProductDetails**;**

DELIMITER $$

**CREATE** **PROCEDURE** getAllProductDetails **(** **)**

**BEGIN**

**SELECT** itemID**,** itemName**,** category**,** quantity**,** description**,**

**FORMAT(**price**,**2**)** **AS** "price"**,**imageFileName

**FROM** productTBL**;**

**END** $$

DELIMITER **;**

A PHP script then uses the product details stored in the $result array to create JavaScript arrays which will be used for menu items.

<?php

**function** createArrays($queryresult) {

$arraysToAdd = '<script type="text/javascript">';

$itemsArray = 'var items = ["';

$pricesArray = ' var prices = [0';

**foreach**($queryresult **as** $row) {

$itemsArray = $itemsArray . '", "' . $row['itemName'];

$pricesArray = $pricesArray . ', ' . $row['price'];

}

$itemsArray = $itemsArray . '"];';

$pricesArray = $pricesArray . '];';

$arraysToAdd = $arraysToAdd . $itemsArray .   
 $pricesArray . '</script>';

/\* Add the code to the page to create the JS arrays \*/

**echo** $arraysToAdd;

}

?>

The items can then be added into submenus. These are put onto the page within Bootstrap tab-panes. Switching between panes is done using nav-pills at the top of the menu.

<!-- Food items section. This div spans 8 columns of the grid. -->

<div class=**"col col-sm-8 content"**>

<br />

<h1>**Menu Items**</h1>

<!-- Add the nav pills -->

<ul class=**"nav nav-pills"**>

<li class=**"active"**><a data-toggle=**"pill"**   
 href=**"#menu1"**>**Appetisers**</a></li>

<li><a data-toggle=**"pill"** href=**"#menu2"**>**Salads**</a></li>

<li><a data-toggle=**"pill"** href=**"#menu3"**>**Noodles**</a></li>

<li><a data-toggle=**"pill"** href=**"#menu4"**>**Rice Plates**</a></li>

<li><a data-toggle=**"pill"** href=**"#menu5"**>**Entrees**</a></li>

<li><a data-toggle=**"pill"** href=**"#menu6"**>**Specialities**</a></li>

<li><a data-toggle=**"pill"** href=**"#menu7"**>**Claypots**</a></li>

<li><a data-toggle=**"pill"** href=**"#menu8"**>**Side Orders**</a></li>

<li><a data-toggle=**"pill"** href=**"#menu9"**>**Desserts**</a></li>

<li><a data-toggle=**"pill"** href=**"#menu10"**>**Beverages**</a></li>

</ul>

<br />

<!-- Add menu items to the tab content -->

<div class=**"tab-content"**>

<div id=**"menu1"** class=**"col col-sm-12 tab-pane fade in active"**>

<h2>**Appetisers**</h2>

<br />

<?php createItemHTML("Appetisers", $result); ?>

</div>

<div id=**"menu2"** class=**"col col-sm-12 tab-pane fade"**>

<h2>**Salads**</h2>

<br />

<?php createItemHTML("Salads", $result); ?>

</div>

<div id=**"menu3"** class=**"col col-sm-12 tab-pane fade"**>

<h2>**Noodles**</h2>

<br />

<?php createItemHTML("Noodles", $result); ?>

</div>

<div id=**"menu4"** class=**"col col-sm-12 tab-pane fade"**>

<h2>**Rice Plates**</h2>

<br />

<?php createItemHTML("RicePlates", $result); ?>

</div>

<div id=**"menu5"** class=**"col col-sm-12 tab-pane fade"**>

<h2>**Entrees**</h2>

<br />

<?php createItemHTML("Entrees", $result); ?>

</div>

<div id=**"menu6"** class=**"col col-sm-12 tab-pane fade"**>

<h2>**Specialities**</h2>

<br />

<?php createItemHTML("Specialities", $result); ?>

</div>

<div id=**"menu7"** class=**"col col-sm-12 tab-pane fade"**>

<h2>**Claypots**</h2>

<br />

<?php createItemHTML("Claypots", $result); ?>

</div>

<div id=**"menu8"** class=**"col col-sm-12 tab-pane fade"**>

<h2>**Side Orders**</h2>

<br />

<?php createItemHTML("SideOrders", $result); ?>

</div>

<div id=**"menu9"** class=**"col col-sm-12 tab-pane fade"**>

<h2>**Desserts**</h2>

<br />

<?php createItemHTML("Desserts", $result); ?>

</div>

<div id=**"menu10"** class=**"col col-sm-12 tab-pane fade"**>

<h2>**Beverages**</h2>

<br />

<?php createItemHTML("Beverages", $result); ?>

</div>

</div><!-- End of tab-content -->

</div>

Each submenu is created using the PHP *createItemHTML* function declared at the top of the page. This function gets the product details for a specific category of food stored in the *$results* array and creates the HTML code necessary to create an entry for each item in the submenu

<?php

/\*

\* The function createItemHTML is used to add the menu items to the tabs

\*/

**function** createItemHTML($category, $queryresult) {

**if**($queryresult==**null**) {

**echo** "Unable to load menu items";

**exit**;

}

/\* These variables are used to keep the code readable and manageable \*/

$htmlPart1 = '<div class="col col-sm-4">  
 <img class="img-responsive foodImgs" src="imgs/';

$htmlPart2 = '" alt="';

$htmlPart3 = '" title="';

$htmlPart4 = '"><br /><h2>';

$htmlPart5 = '</h2><p>';

$htmlPart6 = '</p><p>Price: &pound;';

$htmlPart7 = '<br />Quantity: <a id="' ;

$htmlPart8 = 'numberOfItems">0</a></p>  
 <button type="button" class="btn btn-info btn-add" value="';

$htmlPart9 = '" onClick="onClickAdd(this)">Add to basket</button>  
 <button type="button" class="btn btn-info btn-subtract"   
 value="';

$htmlPart10 = '" onClick="onClickSubtract(this)">&mdash;</button>  
 <button type="button" class="btn btn-info btn-remove"   
 value="';

$htmlPart11 = '" onClick="onClickRemove(this)">X</button><br />';

$htmlPart12 = '<input type="hidden" id="';

$htmlPart13 = 'hidden" name="';

$htmlPart14 = 'hidden" value="0"></div>';

/\* Add the item details stored in the $results array to the page \*/

$count = 1;

**foreach**($queryresult **as** $row) {

**if**($row['category']==$category) {

**echo** $htmlPart1 . $row['imageFileName'] . $htmlPart2 . $row['itemName'];

**echo** $htmlPart3 . $row['itemName'] . $htmlPart4 . $row['itemName'];

**echo** $htmlPart5 . $row['description'] . $htmlPart6 . $row['price'];

**echo** $htmlPart7 . $row['itemID'] . $htmlPart8 . $row['itemID'];

**echo** $htmlPart9 . $row['itemID'] . $htmlPart10 . $row['itemID'].   
 $htmlPart11;

**echo** $htmlPart12 . $row['itemID'] . $htmlPart13 .   
 $row['itemID']. $htmlPart14;

/\* Add a 'newline' after every 3 items to keep the page tidy \*/

**if**($count%3==0) {

**echo** '<div id="food" class="col col-sm-12"><br /><br /></div>';

}

$count++;

}

}

}

?>

The ‘*Add to Basket*’, ‘*Subtract from Basket*’, and ‘*Remove from Basket*’ buttons within the submenus each use a JavaScript event handler to implement their functionality. When each is pressed the order array, quantity text for the item, and the submenu texts and total on the right-hand side of the screen will be updated.

/\* The function onClickAdd adds 1 to the total number of the item clicked,

\* updates its quantity on the page, then updates the subtotal on the page

\* and total on the page. \*/

**function** onClickAdd**(element)**

**{**

// If the item has not yet been added to the order create an entry

**if(**order**[parseInt(element.**value**)]** **==** **null)** **{**

order**[parseInt(element.**value**)]** **=** 1**;**

**}**

**else** **{**

order**[parseInt(element.**value**)]** **+=** 1**;**

**}**

// Create the number of items text

**var** numberOfItemsField **=** **element.**value **+** "numberOfItems"

**var** numberOfItemsHiddenField **=** **element.**value **+** "hidden"

**var** numberOfItems **=** order**[parseInt(element.**value**)];**

// Update the number of items element on the page

**document.**getElementById**(**numberOfItemsField**).**innerHTML **=** numberOfItems**;**

**document.**getElementById**(**numberOfItemsHiddenField**).**value **=**   
 numberOfItems**;**

// update the subtotal and total on the page

updateSubtotal**();**

updateOrderTotal**();**

**};**

/\* The function onClickRemove removes the item from the order list

\* updates its quantity on the page, then updates the subtotal and total

\* on the page. \*/

**function** onClickRemove**(element)**

**{**

// If the item has not yet been added to the order nothing to do here.

**if(**order**[parseInt(element.**value**)]** **==** **null)** **{**

**return;**

**}**

// Remove this item from the order list

order**[parseInt(element.**value**)]** **=** **null;**

// Create the number of items text

**var** numberOfItemsField **=** **element.**value **+** "numberOfItems"

**var** numberOfItemsHiddenField **=** **element.**value **+** "hidden"

**var** numberOfItems **=** 0**;**

// Update the number of items element on the page

**document.**getElementById**(**numberOfItemsField**).**innerHTML **=** numberOfItems**;**

**document.**getElementById**(**numberOfItemsHiddenField**).**value **=**   
 numberOfItems**;**

// update the subtotal and total on the page

updateSubtotal**();**

updateOrderTotal**();**

**};**

/\* The function onClickSubtract subtracts 1 from the total number of the   
 \* item clicked updates its quantity on the page, then updates the subtotal   
 \* and total on the page. \*/

**function** onClickSubtract**(element)**

**{**

// If the item has not yet been added nothing to do here.

**if(**order**[parseInt(element.**value**)]** **==** **null)** **{**

**return;**

**}**

// If the item is no longer selected remove it from the subtotal

**if(**order**[parseInt(element.**value**)]** **==** 1**)** **{**

order**[parseInt(element.**value**)]** **=** **null;**

// Update the number of items element on the page

**var** numberOfItemsField **=** **element.**value **+** "numberOfItems"

**document.**getElementById**(**numberOfItemsField**).**innerHTML **=** "0"**;**

**}**

// Otherwise update the item details

**else** **{**

order**[parseInt(element.**value**)]** **-=** 1**;**

// Create the number of items text

**var** numberOfItemsField **=** **element.**value **+** "numberOfItems"

**var** numberOfItemsHiddenField **=** **element.**value **+** "hidden"

**var** numberOfItems **=** order**[parseInt(element.**value**)];**

// Update the number of items element on the page

**document.**getElementById**(**numberOfItemsField**).**innerHTML **=**   
 numberOfItems**;**

**document.**getElementById**(**numberOfItemsHiddenField**).**value **=**   
 numberOfItems**;**

**}**

// update the subtotal and total on the page

updateSubtotal**();**

updateOrderTotal**();**

**};**

/\* The function updateSubtotal creates the text for the subtotal

\* and adds it to the webpage.

\*/

**function** updateOrderTotal**()**

**{**

**var** orderTotal **=** 0.0**;**

**var** subtotalText**=**""**;**

// Calculate and display the order total

**for(var** key **in** order**)** **{**

orderTotal **+=** prices**[parseInt(**key**)]** **\*** order**[parseInt(**key**)];**

**}**

**document.**getElementById**(**"ordertotal"**).**innerHTML **=** "&pound;" **+**  orderTotal**.**toFixed**(**2**);**

**}**

/\* The function updateSubtotal creates the text for the subtotal

\* and adds it to the webpage.

\*/

**function** updateSubtotal**()**

**{**

**var** itemTotal **=** 0.0**;**

**var** subtotalText**=**""**;**

// Add each item ordered to the subtotal element

**for(var** key **in** order**)** **{**

**var** totalText**=**key **+** "total"**;**

// If the item has been ordered add it to the subtotal

**if(**order**[**key**]!=null)** **{**

itemTotal **=** prices**[parseInt(**key**)]** **\*** order**[parseInt(**key**)];**

subtotalText **+=** items**[**key**]**

subtotalText **+=** " x" **+** order**[parseInt(**key**)];**

subtotalText **+=** " - &pound;"**;**

subtotalText **+=** itemTotal**.**toFixed**(**2**);**

subtotalText **+=** "<br />"**;**

**}**

**}**

**document.**getElementById**(**"subtotal"**).**innerHTML **=** subtotalText**;**

**}**

Pressing the ‘*Place Order*’ button will send a POST request for *placeOrder.php*. The PHP script in *placeOrder.php* will first verify that the user has selected items from the basket. If not, an error message will be shown then the user will be automatically redirected back to *orderonline.php* after 5 seconds.

<?php

/\* Ensure that the customer has ordered at least one item before   
 \* proceeding \*/

$countItems = 0;

**foreach** ($\_POST **as** $key => $value)

{

**if**(endswith($key, 'hidden') && $value!=0) {

$countItems++;

}

**if**($countItems>0) {

**break**;

}

}

**if**($countItems<1) {

**echo** '<h1>ERROR</h1>';

**echo** '<p>You have not selected any items!</p>';

**echo** '<p>You will be redirected back to the previous page in 5   
 seconds</p>';

/\* Get the address of the previous page \*/

$previous = "javascript:history.go(-1)";

**if**(**isset**($\_SERVER['HTTP\_REFERER'])) {

$previous = $\_SERVER['HTTP\_REFERER'];

}

/\* Redirect back to the previous page \*/

$headerText = 'Refresh:5; url="orderonline.php"';

**header**($headerText);

**exit**;

}

?>

Otherwise the PHP script will extract the users order from the details sent with the POST request, connect to the database, create a new order record, then add the ordered items to the database.

<?php

**function** endswith($string, $test) {

$strlen = **strlen**($string);

$testlen = **strlen**($test);

**if** ($testlen > $strlen) {

**return** **false**;

}

**return** **substr\_compare**($string,   
 $test,  
 $strlen - $testlen,   
 $testlen) === 0;

}

$deliveryDatetime = '';

$deliveryDate = '';

$deliveryTime = '';

$customerID = $\_SESSION['customerID'];

**foreach** ($\_POST **as** $key => $value) {

**if** ($key== 'deliveryDate') {

//echo 'Delivery date = ' . $value . '<br />';

$deliveryDatetime = $value . ' ';

$deliveryDate = $value;

}

**else** **if** ($key== 'deliveryTime') {

//echo 'Delivery time = ' . $value . '<br />';

$deliveryDatetime = $deliveryDatetime . $value;

$deliveryTime = $value;

}

}

/\* Connect to the DB, get the product details and add them to an array \*/

$mysqli = **new** mysqli("localhost","webuser","webuser","fooddb");

**if** ($mysqli->connect\_errno) {

**echo** "Failed to connect to MySQL: (" . $mysqli->connect\_errno . ") " .   
 $mysqli->connect\_error;

/\* Redirect to error page here \*/

**header**('Location: errorPage.php');

**exit**;

}

$orderId = -1; /\* Initialise the orderId \*/

/\* create a prepared statement \*/

**if** ($stmt = $mysqli->prepare("SELECT addNewOrder(?,?)")) {

/\* add parameters to the statement \*/

$stmt->bind\_param("is", $customerID, $deliveryDatetime);

/\* execute the addNewOrder procedure call \*/

$stmt->execute();

/\* bind result from function call to orderID \*/

$stmt->bind\_result($orderID);

/\* fetch value \*/

$stmt->fetch();

/\* close statement and connection \*/

$stmt->close();

$mysqli->close();

}

**if** ($orderID==-1) {

**echo** '<h1>ERROR</h1><p>Unable to process your order.   
 Please telephone the store to inform the  
 website&apos;s administrator!</p><br />';

**exit**();

}

/\* Connect to the database, get the ordered item details and insert   
 \* them into the database \*/

$mysqli = **new** mysqli("localhost","webuser","webuser","fooddb");

**if** ($mysqli->connect\_errno) {

**echo** "Failed to connect to MySQL: (" . $mysqli->connect\_errno . ") " .   
 $mysqli->connect\_error;

/\* Redirect to error page here \*/

**header**('Location: errorPage.php');

**exit**;

}

/\* Prepare the insert statement \*/

**if** ($stmt = $mysqli->prepare("CALL insertOrderItem(?,?,?)")) {

$stmt->bind\_param("iii",$param1, $param2, $param3);

/\* Loop through POST array for item details \*/

**foreach** ($\_POST **as** $key => $value)

{

**if**(endswith($key, 'hidden') && $value!=0) {

/\* Add a new order item \*/

$param1 = $orderID;

/\* get the number from the key for the item id\*/

$param2 = **str\_replace**("hidden","",$key);

$param3 = $value;

$stmt->execute();

}

}

/\* End of loop here \*/

/\* close statement and connection \*/

$stmt->close();

$mysqli->close();

}

The addNewOrder procedure used in the code above is a wrapper method for the insertOrder procedure which includes the current date.

-- Create the addNewOrder function

**DROP** **FUNCTION** **IF** **EXISTS** addNewOrder**;**

DELIMITER $$

**CREATE** **FUNCTION** addNewOrder **(** in\_customerID **INT,**

in\_deliveryTimeDate **DATETIME** **)**

**RETURNS** **INT**

**BEGIN**

**DECLARE** today **DATE;**

**DECLARE** todaynow **DATETIME;**

**SET** today **=** curdate**();**

**SET** todaynow **=** now**();**

-- check first that delivery date is not before today and current time

**IF** todaynow **>** in\_deliveryTimeDate **THEN**

**RETURN** -1**;**

**END** **IF;**

-- insert the new order and return the id of the record created

**CALL** insertOrder**(**in\_customerID**,** today**,** in\_deliveryTimeDate**,** 0**);**

**RETURN** LAST\_INSERT\_ID**();**

**END** $$

DELIMITER **;**

The *insertOrderItem* procedure adds ordered items individually to the orderItemsTBL table.

-- Create the insertOrderItem procedure

**DROP** **PROCEDURE** **IF** **EXISTS** insertOrderItem**;**

DELIMITER $$

**CREATE** **PROCEDURE** insertOrderItem **(** **IN** in\_orderID **INT,**

**IN** in\_itemID **INT,**

**IN** in\_quantity **TINYINT** **)**

**BEGIN**

**INSERT** **INTO** orderItemsTBL **(** orderID**,**

itemID**,**

quantity**)**

**VALUES** **(** in\_orderID**,**

in\_itemID**,**

in\_quantity **);**

**END** $$

DELIMITER **;**

If all has gone well, the user will be informed that their order has been placed then redirected to the home page after 5 seconds.

<?php

/\* Write the confirmation message to the page \*/

**echo** '<h1 class="indentedText">Confirmation</h1><p class="indentedText">  
 Your order has been placed and will arrive on ' .   
 $deliveryDate . ' at ' .   
 $deliveryTime . '</p>';

**echo** '<br />';

**echo** '<p class="indentedText">You will be redirected back to the home page   
 in 5 seconds</p>';

/\* Redirect back to the home page \*/

$headerText = 'Refresh:5; url="index.php"';

**header**($headerText);

?>

A future application or webpage for restaurant staff will allow them to view customer orders as they are received. Procedures have already been written to provide the details required by cooks and delivery personnel.

The procedure *getOrderItemDetails* was written so to retrieve a list of all order items. This procedure will calculate the subtotal price of each item and format it into a user-friendly decimal format.

-- Create the getOrderItemDetails procedure

**DROP** **PROCEDURE** **IF** **EXISTS** getOrderItemDetails**;**

DELIMITER $$

**CREATE** **PROCEDURE** getOrderItemDetails **(** **IN** in\_orderID **INT** **)**

**BEGIN**

**SELECT** producttbl**.**itemName**,**

producttbl**.**category**,**

producttbl**.**description**,**

orderitemstbl**.**quantity**,**

**FORMAT(**producttbl**.**price **\*** orderitemstbl**.**quantity**,**2**)** "ordercost"

**FROM** orderitemstbl

**INNER** **JOIN** producttbl **ON** **(**orderitemstbl**.**itemID**=**producttbl**.**itemID**)**

**WHERE**

orderitemstbl**.**orderID**=**in\_orderID**;**

**END** $$

DELIMITER **;**

Another procedure *getOrderDeliveryDetails* was written to retrieve customer details for the order.

-- Create the getOrderDeliveryDetails procedure

**DROP** **PROCEDURE** **IF** **EXISTS** getOrderDeliveryDetails**;**

DELIMITER $$

**CREATE** **PROCEDURE** getOrderDeliveryDetails **(** **IN** in\_orderID **INT** **)**

**BEGIN**

**SELECT** customerdetailstbl**.**firstname**,**

customerdetailstbl**.**surname**,**

customerdetailstbl**.**addressLine1**,**

customerdetailstbl**.**addressLine2**,**

customerdetailstbl**.**postcode**,**

customerdetailstbl**.**phoneNumber

**FROM** orderstbl

**INNER** **JOIN** customerdetailstbl **ON** **(**orderstbl**.**customerID**=**customerdetailstbl**.**customerID**)**

**WHERE**

orderstbl**.**orderID **=** in\_orderID**;**

**END** $$

DELIMITER **;**

# 

# Appendix C Sample Code Listings 3 Enforcing Unique User Names

## Enforcing Unique User Names

Although customer records are identified using a unique number, customers will log into their accounts with a user name and password combination. User names chosen by customers need to be unique so that their login credentials do not clash with other users.

The first stage in implementing this is to declare the user name as unique in the creation of the *customerDetailsTBL* table used to hold customer details.

**CREATE** **TABLE** customerDetailsTBL **(**

customerID **INT** AUTO\_INCREMENT **PRIMARY** **KEY,**

username **VARCHAR(**35**)** **UNIQUE)** **NOT** **NULL,**

password **VARCHAR(**255**)** **NOT** **NULL,**

surname **VARCHAR(**35**)** **NOT** **NULL,**

firstname **VARCHAR(**35**)** **NOT** **NULL,**

addressLine1 **VARCHAR(**35**),**

addressLine2 **VARCHAR(**35**),**

postcode **VARCHAR(**8**)** **NOT** **NULL,**

phoneNumber **VARCHAR(**35**),**

emailAddress **VARCHAR(**50**)**

**);**

If a customer now tries to enter a duplicate user name into the database, it will throw an error. This is not however the most user-friendly way of implementing this functionality. The developer wanted to let the user know that they had entered a duplicate user name immediately after they had entered it into the registration form.

Implenetation required a combination of JavaScript, AJAX and a MySQL procedure. First a MySQL function was created check whether a user name is unique;

-- Create the usernameExists function

-- If the user name sent doesn’t exist this will return 0.

**DROP** **FUNCTION** **IF** **EXISTS** usernameExists**;**

DELIMITER $$

**CREATE** **FUNCTION** usernameExists**(** in\_username **VARCHAR(**35**)** **)**

**RETURNS** **INT**

**BEGIN**

**DECLARE** out\_var **INT;**

**SELECT** **COUNT(\*)**

**INTO** out\_var

**FROM** customerDetailsTBL

**WHERE** username**=**in\_username**;**

**RETURN** out\_var**;**

**END** $$

DELIMITER **;**

Next, an event handler was added to the registration form for the user name field. This calls a JavaScript function each time the user leaves the field.

<label for=**"usrname"**>**Username**</label>

<input type=**"text"** id=**"usrname"** name=**"usrname"**   
 placeholder=**"Choose a unique username ..."**

maxlength=**"35"** onchange=**"checkUsername(this.value)"**>

The JavaScript function called by the event handler makes a background GET request for a PHP script using AJAX. This request checks if the user name already exists. The value created by the request is then added to the user name field on the form.

<script>

***function*** checkUsername**(**str**)** **{**

***if*** **(**str **==** ""**)** **{**

document.getElementById**(**"usrname"**).**value **=** ""**;**

***return*;**

**}**

***else*** **{**

***if*** **(**window.XMLHttpRequest**)** **{**

// code for IE7+, Firefox, Chrome, Opera, Safari

xmlhttp **=** ***new*** XMLHttpRequest**();**

**}**

***else*** **{**

// code for IE6, IE5

xmlhttp **=** ***new*** ActiveXObject**(**"Microsoft.XMLHTTP"**);**

**}**

xmlhttp.onreadystatechange **=** ***function*()** **{**

***if*** **(**this.readyState **==** 4 **&&** this.status **==** 200**)** **{**

document.getElementById**(**"usrname"**).**value **=** this.responseText**;**

**}**

**};**

xmlhttp.open**(**"GET"**,**"customerExists.php?q="**+**str**,*true*);**

xmlhttp.send**();**

**}**

**}**

</script>

The *customerExists.php* script connects to the database and calls the *usernameExists* function to check the user name. If it already exists the script will create an error message for the field, otherwise the user name entered will be reinserted.

<?php

$usrName = $\_GET['q'];

/\* Connect to the DB, check if the username exists\*/

$mysqli = **new** mysqli("localhost","webuser","webuser","fooddb");

**if** ($mysqli->connect\_errno) {

**echo** "Failed to connect to MySQL: (" . $mysqli->connect\_errno . ") " .   
 $mysqli->connect\_error;

/\* Redirect to error page here \*/

**header**('Location: errorPage.php');

**exit**;

}

$customerIdExists = -1;

/\* create a prepared statement

\* Note: usernameExists returns 0 if a username is not already   
 \* in the database \*/

**if** ($stmt = $mysqli->prepare("SELECT usernameExists(?)")) {

/\* add parameters to the statement \*/

$stmt->bind\_param("s", $usrName);

/\* execute query \*/

$stmt->execute();

/\* bind result from function call to orderID \*/

$stmt->bind\_result($customerIdExists);

/\* fetch value \*/

$stmt->fetch();

/\* close statement and connection \*/

$stmt->close();

$mysqli->close();

}

/\* This text will be added to the username input field \*/

**if**(!$customerIdExists==0) {

**echo** 'USERNAME ALREADY EXISTS!';

}

**else** {

**echo** $usrName;

}

?>

Sometimes a customer may miss the error message, so a further JavaScript check was added to prevent the error message from being used as the customer’s new user name. The function *validate\_registration* is called when the SUBMIT button is pressed on the registration form. It checks for empty fields and ensures that certain fields are the correct length.

**function** validate\_registration**()**

**{**

/\* Store values from the input fields into variables \*/

**var** usrname **=** **document.forms.**frmRegister**.**usrname**.**value**;**

**var** userLen **=** **document.forms.**frmRegister**.**usrname**.**value**.**length**;**

**var** pass **=** **document.forms.**frmRegister**.**pwd**.**value**;**

**var** passLen **=** pass**.**length**;**

**var** pass1 **=** **document.forms.**frmRegister**.**pwdconfirm**.**value**;**

/\* Get the lengths of field values \*/

**var** lnameLen **=** **document.forms.**frmRegister**.**lname**.**value**.**length**;**

**var** fnameLen **=** **document.forms.**frmRegister**.**fname**.**value**.**length**;**

**var** phoneLen **=** **document.forms.**frmRegister**.**phoneno**.**value**.**length**;**

**var** emailLen **=** **document.forms.**frmRegister**.**email**.**value**.**length**;**

**if(** userLen **==** 0 **)**

**{**

**alert(**"Please choose a Username "**);**

**return** **false;**

**}**

**if(** userLen **<** 7 **)**

**{**

**alert(**"Invalid Username - minimum length is 7 characters "**);**

**return** **false;**

**}**

/\* If the user has not entered a unique user name show message \*/

**if(** usrname**.**includes**(**'USERNAME ALREADY EXISTS!'**)** **)**

**{**

**alert(**"Error: You must choose a unique username."**);**

**return** **false;**

**}**

**if(** pass **==** ""**)**

**{**

**alert(**"Please choose a Password "**);**

**return** **false;**

**}**

**if(** passLen **<** 7 **)**

**{**

**alert(**"Invalid Password - minimum length is 7 characters "**);**

**return** **false;**

**}**

**if(** pass1 **==** "" **||** pass **!=** pass1 **)**

**{**

**alert(**"Passwords entered do not match "**);**

**return** **false;**

**}**

**if(** lnameLen **==** 0 **)**

**{**

**alert(**"Please enter your surname."**);**

**return** **false;**

**}**

**if(** fnameLen **==** 0 **)**

**{**

**alert(**"Please enter your first name."**);**

**return** **false;**

**}**

**if(document.forms.**frmRegister**.**addr1**.**value **==** ""**)**

**{**

**alert(**"Please enter the first line of your address."**);**

**return** **false;**

**}**

**if(**phoneLen **+** emailLen **==** 0**)** **{**

**alert(**"Please enter either your phone number or email address   
 for us to be able to contact you."**);**

**return** **false;**

**}**

**}**