|  |  |  |
| --- | --- | --- |
| 5COSC021W Coursework 2 – Referral/Deferral template June 2023  * YOU MUST USE THIS TEMPLATE FOR THE REFERRAl/DEFERRAL COURSEWORK 2.  The current size of the boxes is not indicating how much you should write; change their size as you need.When you save the file, put your name and registration number in the file name, eg ‘5COSC021W\_cwk2\_group\_Kelly\_Garret\_12345678.doc’.  * A reminder of plagiarism: If you use bits of another’s student’s report in yours or if you give your report to another student to use this will be an academic offence called ‘collusion’. * In order for the tutors to be able to assess your work you must ensure the following for your software submission:   - Submit a zipped project folder of the working files but without the libraries  - Make Sure you include the database you implemented  - You must upload a video describing the work in a google drive given in the submission link. Make sure that you name your video appropriately with your name AND student id number eg 5COSC021w\_video\_Kelly\_Garret\_123456878 | | |
| **Surname** |  | |
| **Forename** |  | |
| **Registration No:** |  | |
| **By submitting this coursework you agree to the following:** | | |
| I confirm that I understand what plagiarism is and have read and understood the section on Assessment Offences in the Essential Information for Students. The work that I have submitted is entirely my own. Any work from other authors is duly referenced and acknowledged. | | I confirm |

|  |
| --- |
| Code – Code functionality (30 marks) |
| **Guidance:**   * Include here the **names of the files** that make up your application. * Explain what each file does and the functionality of the code. Detail whether all the code in the file is your own product or used components from the lecture notes. Include the names of the other authors where applicable. Authorship information should also be commented at the beginning of each of your files. * Marking of this section will also include assessment on the submitted files and the overall functionality of your work * Provide screen shots of all panels you implemented and the SQLite database * Discuss the front end and back-end implementation * Discuss front-end and back-end error handling |
| **The `db\_connect.php`** file is used to establish a connection with your MySQL database. It specifies the server name, username, password, and database name to connect with the MySQL server using the `mysqli` extension.  Once it creates the connection (`$conn = new mysqli(...)`), it checks if the connection is successful. If there's an error in connecting, it stops the script execution and outputs the error message (`die("Connection failed: " . $conn->connect\_error);`).  Note: In a production setting, avoid showing specific connection errors to users as this can be a potential security risk. Consider logging errors for your review instead of displaying them.  This file is vital because it's included (`require 'db\_connect.php';`) in other PHP files to facilitate communication with your database. In terms of authorship, this is a basic and common pattern for establishing a MySQL database connection in PHP, and is typically customized per the specific application needs.  As per error handling, it uses the built-in error handling provided by PHP and the `mysqli` extension. When the connection fails, the error message gets displayed and the script execution halts.  **Index.php** includes a header with a logo and title, links for registration and login, and a footer with copyright information. The CSS file provides styling. No PHP script is present, except for displaying the current year. The file is serves as the entry point for users to interact with the app.    **login\_staff.php**  is a login page for the web app. It starts a session and checks if the user is already logged in, redirecting them to the appropriate dashboard if so. It includes a form for users to enter their username and password. Upon form submission, it validates the input, retrieves user information from the database, and verifies the password. If the credentials are correct, it creates a session and redirects the user to the corresponding dashboard based on their role. Error messages are displayed if there are any issues. The file also includes HTML markup and styling from the CSS file.    **login.php**  is a login page for web app, specifically designed for patients. It starts a session and checks if the user is already logged in, redirecting them to the patient dashboard if so. It includes a form for users to enter their NHS number and password. Upon form submission, it validates the input, retrieves patient information from the database, and verifies the password. If the credentials are correct, it creates a session and redirects the user to the patient details page. Error messages are displayed if there are any issues. The file also includes HTML markup and styling from the CSS file.    **logout.php** is a logout script for web app. It starts the session, unsets all session variables, destroys the session, and then redirects the user to the login page (index.php). This script is used to log out the user and ensure their session data is cleared.    **home\_patient.php** is the home page for logged-in patients on a GP surgery website. It checks the user's login status, fetches their information from the database, and displays a personalized welcome message. The page also provides a button to navigate to the "patient\_details.php" page for viewing further details.  1. The PHP code begins with session management. It starts the session using `session\_start()` to maintain session data.  2. It checks if the user is already logged in by verifying the existence of the `nhs\_number` session variable. If the variable is not set, indicating that the user is not logged in, it redirects them to the login page (`login.php`) using the `header()` function and then exits the script.  3. If the user is logged in, it includes the `db\_connect.php` file, which is presumably responsible for establishing a connection to the database.  4. The script retrieves user information from the database based on the `nhs\_number` stored in the session. It constructs an SQL query to select the row from the `patients` table where the `nhs\_number` matches the one stored in the session.  5. If the query returns one or more rows, indicating that the user was found in the database, the script fetches the data from the first row and assigns it to variables for further use. The variables `$name`, `$dob`, `$address`, and `$mobilephone` store the patient's name, date of birth, address, and mobile phone number, respectively.  6. If the query returns no rows, implying that the user was not found in the database, the script displays the "User not found." message and exits.  7. The HTML markup starts here. It defines the page title as "GP Surgery" and includes an external CSS file called "styles.css" for styling purposes.  8. The page header section contains a logo and the heading "Home".  9. The main content section is a container with a class `home-content`. It has a height of `80vh` and is styled to display its contents vertically at the center. Inside the container, there's a paragraph that displays a welcome message to the user, making use of the `$name` variable retrieved from the database.  10. There's a button with the text "See Your Details". When clicked, it triggers a JavaScript function (`window.location.href='patient\_details.php'`) that redirects the user to the "patient\_details.php" page, where they can view their details.    **patient\_details.php**  displays patient details in the GP Vaccination Tracker web app. Here's a breakdown of its functionality:  1. It starts the session and checks if the user is logged in. If not, it redirects them to the login page.  2. It includes the database connection file (db\_connect.php) to establish a connection with the database.  3. It fetches the user's information from the database based on their NHS number stored in the session.  4. If the user is found in the database, their details (name, NHS number, date of birth, address, mobile phone) are retrieved and stored in variables.  5. If the user is not found, it displays an error message and exits the script.  6. The retrieved user details are then displayed in an HTML page along with a welcome message.  7. The page also includes a "Logout" link that redirects the user to the logout script (logout.php) when clicked.  Overall, this code retrieves and displays the patient's details based on their NHS number after verifying their login status. It provides a personalized experience for the patient and allows them to log out from the system.    **register\_staff.php it’s**  a registration form for staff in the web app. Here's a breakdown of its functionality:  1. It starts the session and checks if the user is already logged in. If yes, it redirects them to the doctor dashboard.  2. It includes the database connection file (db\_connect.php) to establish a connection with the database.  3. It initializes an empty error variable to store any validation or database-related errors.  4. If the form is submitted ($\_SERVER["REQUEST\_METHOD"] == "POST"), it retrieves the form data (name, last name, phone, username, password) and performs input validation.  5. If any field is empty, it sets an error message.  6. If the username already exists in the database, it sets an error message.  7. If there are no errors, it hashes the password and inserts the doctor's information into the database.  8. If the registration is successful, it creates a session for the doctor and redirects them to the doctor dashboard page.  9. If there is an error during the database insertion, it sets an error message with the corresponding error details.  10. If the page is refreshed after a successful registration (POST request with no error), it redirects the user to the doctor dashboard to prevent duplicate entries.  11. The HTML form includes input fields for name, last name, phone, username, and password.  12. The form data is submitted to the same page (htmlspecialchars($\_SERVER["PHP\_SELF"])), and the values are pre-filled if there was a validation error.  13. If there is an error during form submission, the error message is displayed.  14. The footer section contains the copyright information and a logo.  Overall, this code allows doctors to register by providing their information, which is then stored in the database for future login and usage of the GP Vaccination Tracker system**.**    **register.php** The provided PHP code represents a registration form for patients in the GP Vaccination Tracker web app. Here's a breakdown of its functionality:  1. It starts the session and checks if the user is already logged in. If yes, it redirects them to the patient dashboard.  2. It includes the database connection file (db\_connect.php) to establish a connection with the database.  3. It initializes an empty error variable to store any validation or database-related errors.  4. If the form is submitted ($\_SERVER["REQUEST\_METHOD"] == "POST"), it retrieves the form data (NHS number, name, date of birth, address, mobile phone, username, password) and performs input validation.  5. If any field is empty, it sets an error message.  6. If the NHS number already exists in the database, it sets an error message.  7. If there are no errors, it hashes the password and inserts the patient's information into the database.  8. If the registration is successful, it creates a session for the patient with the NHS number and redirects them to the patient dashboard page.  9. If there is an error during the database insertion, it sets an error message with the corresponding error details.  10. If the page is refreshed after a successful registration (POST request with no error), it redirects the user to the patient dashboard to prevent duplicate entries.  11. The HTML form includes input fields for the NHS number, name, date of birth, address, mobile phone, username, and password.  12. The form data is submitted to the same page (htmlspecialchars($\_SERVER["PHP\_SELF"])), and the values are pre-filled if there was a validation error.  13. If there is an error during form submission, the error message is displayed.    **staff\_dashboard.php** is a registration form for patients in the web app. Here's a breakdown of its functionality:  1. It starts the session and checks if the user is already logged in. If yes, it redirects them to the patient dashboard.  2. It includes the database connection file (db\_connect.php) to establish a connection with the database.  3. It initializes an empty error variable to store any validation or database-related errors.  4. If the form is submitted ($\_SERVER["REQUEST\_METHOD"] == "POST"), it retrieves the form data (NHS number, name, date of birth, address, mobile phone, username, password) and performs input validation.  5. If any field is empty, it sets an error message.  6. If the NHS number already exists in the database, it sets an error message.  7. If there are no errors, it hashes the password and inserts the patient's information into the database.  8. If the registration is successful, it creates a session for the patient with the NHS number and redirects them to the patient dashboard page.  9. If there is an error during the database insertion, it sets an error message with the corresponding error details.  10. If the page is refreshed after a successful registration (POST request with no error), it redirects the user to the patient dashboard to prevent duplicate entries.  11. The HTML form includes input fields for the NHS number, name, date of birth, address, mobile phone, username, and password.  12. The form data is submitted to the same page (htmlspecialchars($\_SERVER["PHP\_SELF"])), and the values are pre-filled if there was a validation error.  13. If there is an error during form submission, the error message is displayed. |

|  |
| --- |
| Your code – Code Quality – Maintainability (10 marks) |
| **Guidance:**   * Comment on the maintainability of your code. * **Support your statements with evidence from your code** * Marking of this section will also include assessment on the submitted files and the overall functionality of your work |
|  |

|  |
| --- |
| Your code – Code Quality – version control (10 marks) |
| **Guidance:**   * Comment on your version control of your code. * **Support your statements with evidence from by providing screenshots (eg Github or google drive or whatever method you used)** |
|  |

|  |
| --- |
| Your code – Code Quality – output of test plans (10 marks) |
| **Guidance:**   * List your **test plan and the output** of your program * Marking of this section will also include assessment on the submitted files and the overall functionality of your work |
|  |

|  |
| --- |
| Application Front End (HCI) (10 marks) |
| **Guidance:** Attach here a screenshot of the front end of your application, incorporating both login panel and the graphical display of data   * Sum up all the main **HCI** issues of the application you have addressed and how they were addressed. Include any HCI issues remaining. * Comment on how well all parts of the application are linked |

|  |
| --- |
| Application Front End (SECURITY) (10 marks) |
| **Guidance:**   * Sum up all the main **security** issues of the application and how they were addressed.   and any security risks remaining. |

|  |
| --- |
| Professional conduct: Legal (10 marks) |
| **Guidance:** With the aid of a table list here the legal issues that would affect both the development and the use of your application. You need to support this work with research. The marks in this section also include marks for references (see end of document). |
|  |

|  |
| --- |
| Professional conduct: Ethical (10 marks) |
| **Guidance:** With the aid of a table list here the ethical issues that would affect both the development and the use of your application. You need to support this work with research. The marks in this section also include marks for references (see end of document). |
|  |

|  |
| --- |
| References (marks included in each of the main sections) |
| Sections 7 and 8 must be supported by research.List below your sources, using Harvard referencing. Make sure that your references are referred to correctly from the relevant text of your work. **If you are not clear how to reference read:**  **https://www.westminster.ac.uk/library-and-it/support-and-study-skills/guides-and-tutorials/referencing-your-work** Here’s how we’ll assess it:  * No research sources: that’s very bad for level 5 work * There is one source with all information, copied directly as if it’s your own text: that is plagiarism * There is one source with all information, referenced and discussed: that is bad research * There are a few different sources, referenced and discussed in the text: this is getting better * There are quite a few good sources from many different places, referenced and discussed in the text: this gets good marks. |
| **Section 7 references** |
|  |
| **Section 8 references** |
|  |