ECS417U Project Phase 2 Documentation

|  |  |
| --- | --- |
| **Name:** | Rayaan Mubashir Uddin |
| **Student Number:** | 230335705 |

|  |  |
| --- | --- |
| **Assessment Criteria 7**  Each time a new entry is added, the most recent post appears on top, followed by the next most recent post using a PHP-driven technique (writing a sorting algorithm) instead of a SQL Query. | |
| Instruction: use this column to paste the code from the files that have been used to develop this functionality.  **Paste the code directly from visual studio code, do not include a screenshot of the code as this document has a Turnitin requirement and this will lead to an automatic mark of zero. See example below.** | Instruction: use this column to address the following in your own words:   1. Describe the steps you took to develop this feature (including the prompts). 2. Describe how the sorting algorithm works.   Give reference to the code you have written when answering the questions above.  You can use diagrams (e.g. flow charts) or screenshots of your webpage output to illustrate your answer. |
| Filename: database.php  <?php  session\_start();  $db = new mysqli('localhost', 'RayaanUddin', 'Rayaan10', 'portfolio');  $permission\_levels = array(  0 => "User",  1 => "Admin"  );  // Class represents website user  class User {  public $id;  public $fname;  public $lname;  public $email;  private $permission;  public function \_\_construct($id) {  $this->id = $id;  global $db;  $sql = "SELECT \* FROM accounts WHERE accountId = '$id'";  $result = $db->query($sql);  if ($result->num\_rows === 1) {  $row = $result->fetch\_assoc();  $this->fname = $row["fname"];  $this->lname = $row["lname"];  $this->email = $row["email"];  $this->permission = $row['permission'];  }  }  public function getFullname() {  return $this->fname." ".$this->lname;  }  public function isAdmin() {  global $permission\_levels;  if ($permission\_levels[$this->permission] == "Admin") {  return true;  } else {  return false;  }  }  }  // Login to account  function login($email, $password) {  global $db;  $sql = "SELECT \* FROM accounts WHERE email = '".$email."' AND password = '".$password."'";  $result = $db->query($sql);  if ($result->num\_rows === 1) {  $row = $result->fetch\_assoc();  return new User($row['accountId']);  } else {  return false;  }  }  // Register a new account  function register($fname, $lname, $email, $password, $perm) {  global $db;  $sql = "SELECT \* FROM accounts WHERE email = '$email'";  $result = $db->query($sql);  if ($result->num\_rows === 0) { // If account does not exists...  $sql = "INSERT INTO accounts (fname, lname, email, password, permission) VALUES ('$fname', '$lname', '$email', '$password', '$perm')";  if ($db->query($sql)) { // Should always execute unless issue on database  return true;  } else { // Exists for texting purposes  return false;  }  } else { // If account exists...  return false;  }  }  // Abstract class for Comment and Blog Postings.  abstract class Post {  public $id;  public $author;  public $date;  public function \_\_construct($id, $accountId, $date) {  $this->id = $id;  $this->author = new User($accountId);  $this->date = $date;  }  public function timeAgo\_toString() {  $datestr = "";  $yeardiff = date('Y') - date('Y', $this->date);  $monthdiff = date("m") - date("m", $this->date);  if ($monthdiff < 0) {  $yeardiff--;  $monthdiff+=12;  }  $datediff = date("d") - date("d", $this->date);  if ($datediff < 0) {  $monthdiff--;  $datediff+=30;  }  $hourdiff = date("H") - date("H", $this->date);  if ($hourdiff < 0) {  $datediff--;  $hourdiff+=24;  }  $minutediff = date("i") - date("i", $this->date);  if ($minutediff < 0) {  $hourdiff--;  $minutediff+=60;  }  $seconddiff = date("s") - date("s", $this->date);  if ($seconddiff < 0) {  $minutediff--;  $seconddiff+=60;  }  if ($yeardiff > 0) {  $datestr .= $yeardiff." year".($yeardiff > 1 ? "s " : " ");  } elseif ($monthdiff > 0) {  $datestr .= $monthdiff." month".($monthdiff > 1 ? "s " : " ");  } elseif ($datediff > 0) {  $datestr .= $datediff." day".($datediff > 1 ? "s " : " ");  } elseif ($hourdiff > 0) {  $datestr .= $hourdiff." hour".($hourdiff > 1 ? "s " : " ");  } elseif ($minutediff > 0) {  $datestr .= $minutediff." minute".($minutediff > 1 ? "s " : " ");  } elseif ($seconddiff > 0) {  $datestr .= $seconddiff." second".($seconddiff > 1 ? "s " : " ");  }  if ($datestr == "") {  return "Just now";  }  return $datestr."ago";  }  public function getLongDate\_toString() {  return date('dS F Y | H:i', $this->date);  }  }  class Comment extends Post {  public $comment;  public function \_\_construct($comment, $authorId, $date, $id) {  $this->comment = $comment;  parent::\_\_construct($id, $authorId, $date);  }  }  class Blog extends Post {  public $title;  public $content;  public $comments;  private function getComments() {  global $db;  $sql = "SELECT \* FROM comments WHERE blogId = '".$this->id."'";  $result = $db->query($sql);  $comments = array();  if ($result->num\_rows > 0) {  while ($row = $result->fetch\_assoc()) {  $comment = new Comment($row['comment'], $row['accountId'], strtotime($row['date']), $row['commentId']);  array\_push($comments, $comment);  }  }  return $comments;  }  public function \_\_construct($title, $content, $authorId, $date, $id) {  $this->title = $title;  $this->content = $content;  parent::\_\_construct($id, $authorId, $date);  // Get posts comments  $this->comments = orderByDateDesc($this->getComments());  }  }  // AI Generated sorting algorithm  // Start  function mergeSort(&$arr, $left, $right) {  if ($left < $right) {  $mid = (int)(($left + $right) / 2);  mergeSort($arr, $left, $mid);  mergeSort($arr, $mid + 1, $right);  merge($arr, $left, $mid, $right);  }  }  function merge(&$arr, $left, $mid, $right) {  $n1 = $mid - $left + 1;  $n2 = $right - $mid;  $L = [];  $R = [];  for ($i = 0; $i < $n1; $i++)  $L[$i] = $arr[$left + $i];  for ($j = 0; $j < $n2; $j++)  $R[$j] = $arr[$mid + 1 + $j];  $i = 0;  $j = 0;  $k = $left;  while ($i < $n1 && $j < $n2) {  if ($L[$i]->date >= $R[$j]->date) {  $arr[$k] = $L[$i];  $i++;  } else {  $arr[$k] = $R[$j];  $j++;  }  $k++;  }  while ($i < $n1) {  $arr[$k] = $L[$i];  $i++;  $k++;  }  while ($j < $n2) {  $arr[$k] = $R[$j];  $j++;  $k++;  }  }  function orderByDateDesc($objects) {  $length = count($objects);  mergeSort($objects, 0, $length - 1);  return $objects;  }  // End  // Gets all blogs from database  function getAllBlogs() {  global $db;  $blog\_array = array();  $sql = "SELECT \* FROM blog";  $result = $db->query($sql);  if ($result->num\_rows > 0) {  while ($row = $result->fetch\_assoc()) {  $date = strtotime($row['date']);  $blog = new Blog($row['title'], $row['content'], $row['accountId'], $date, $row['blogId']);  array\_push($blog\_array, $blog);  }  }  return $blog\_array;  }  function addBlog($title, $content, $author) {  global $db;  $date = date('Y-m-d H:i:s');  $sql = "INSERT INTO blog (title, content, accountId, date) VALUES ('$title', '$content', '$author', '$date')";  if ($db->query($sql) == true) {  return true;  } else {  return false;  }  }  // Gets a blog by id  function getBlog($id) {  global $db;  $sql = "SELECT \* FROM blog WHERE blogId = '".$id."'";  $result = $db->query($sql);  if ($result->num\_rows === 1) {  $row = $result->fetch\_assoc();  $date = strtotime($row['date']);  $blog = new Blog($row['title'], $row['content'], $row['accountId'], $date, $row['blogId']);  return $blog;  } else {  return null;  }  }  // Deletes a blog by id  function deleteBlog($id) {  global $db;  if (isset($\_SESSION["user"])) {  if ($\_SESSION["user"]->isAdmin()) { // Only admins can delete blogs  // Delete all comments first from blog post..  $sql = "DELETE FROM comments WHERE blogId = '".$id."'";  if ($db->query($sql) == true) {  // Then delete the blog post..  $sql = "DELETE FROM blog WHERE blogId = '".$id."'";  if ($db->query($sql) == true) {  return true;  } else {  return false;  }  } else {  return false;  }  } else {  return false;  }  } else {  return false;  }  }  // Add Comment to blog  function addComment($id, $content, $author) {  global $db;  $date = date('Y-m-d H:i:s');  $sql = "INSERT INTO comments (comment, blogId, accountId, date) VALUES ('$content', '$id', '$author', '$date')";  if ($db->query($sql) == true) {  echo "Comment added";  } else {  echo "Error: " . $sql . "<br>" . $db->error;  }  }  // Delete comment by id  function deleteComment($id) {  global $db;  if (isset($\_SESSION["user"])) {  if ($\_SESSION["user"]->isAdmin()) {  } else {  $accountId = $\_SESSION["user"]->id;  $sql = "SELECT commentId FROM comments WHERE commentId = '".$id."' AND accountId = '".$accountId."'";  $result = $db->query($sql);  if ($result->num\_rows === 1) {  } else {  return false;  }  }  $sql = "DELETE FROM comments WHERE commentId = '".$id."'";  if ($db->query($sql) == true) {  return true;  } else {  return false;  }  } else {  return false;  }  }  ?>  Filename: viewBlog.php  <?php  include 'scripts/php/database.php';  ?>  <!DOCTYPE html>  <html lang="en">  <?php include 'scripts/php/head.php'; ?>  <body>  <div class="container">  <?php  include 'scripts/php/header.php';  ?>  <section id="blog-error">  <?php  if (isset($\_GET['error'])) {  echo "<p>Error: ".$\_GET['error']."</p>";  }  ?>  </section>  <?php  if (isset($\_GET['id'])) {  $blog = getBlog($\_GET['id']);  if ($blog != null) {  echo "<article id='blog\_full'>";  echo "<nav id='blog-nav'>";  echo "<a href='viewBlog.php#".$\_GET['id']."'><i class='fa fa-arrow-left'></i><p>Back</p></a>";  echo "</nav>";  echo "<time datetime='".$blog->date."'>".$blog->getLongDate\_toString()."</time>";  echo "<h3>".$blog->title."</h3>";  echo "<p class='content'>".$blog->content."</p>";  echo "<p class='author'>Author: ".$blog->author->getFullname()."</p>";  echo "<div id='comments'>";  echo "<form action='scripts/php/addComment.php' method='get'>";  echo "<textarea name='comment' placeholder='Enter comment here'></textarea>";  echo "<input type='hidden' name='id' value='".$blog->id."' required>";  echo "<input type='submit' value='Send'>";  echo "</form>";  echo "<ul id='comments\_list'>";  for ($i = 0; $i < count($blog->comments); $i++) {  echo "<li class='comment'>";  echo "<p class='comment\_author'>".$blog->comments[$i]->author->getFullname()."</p>";  echo "<p class='comment\_content'>".$blog->comments[$i]->comment."</p>";  echo "<time datetime='".$blog->comments[$i]->date."'>".$blog->comments[$i]->timeAgo\_toString()."</time>";  if (isset($\_SESSION['user']) && ($blog->comments[$i]->author->id == $\_SESSION['user']->id || $\_SESSION['user']->isAdmin())) {  echo '<a class="post\_comment\_delete" href="scripts/php/deleteBlog.php?commentId='.$blog->comments[$i]->id.'&blogId='.$\_GET['id'].'"><i class="fa fa-trash"></i></a>';  }  echo "</li>";  }  echo "</ul>";  echo "</div>";  echo "</article>";  }  else {  echo "<p>Blog not found</p>";  }  }  else {  $blogs = orderByDateDesc(getAllBlogs());  if (isset($\_GET['title']) && isset($\_GET['content']) && isset($\_SESSION['user']) && $\_SESSION['user']->isAdmin()) { // Preview Blog  ?>  <nav id="blog-nav">  <a href="addEntry.php?title=<?php echo $\_GET['title']; ?>&content=<?php echo $\_GET['content']; ?>"><i class='fa fa-arrow-left'></i><p>Back</p></a>  <p>Preview</p>  <a href="scripts/php/addPost.php?title=<?php echo $\_GET['title']; ?>&content=<?php echo $\_GET['content']; ?>"><p>Post</p><i class='fa fa-check'></i></a>  </nav>  <a id='addpost\_button' href='addEntry.php'>Add Post</a>  <section id='blog'>  <article id="preview">  <time>Preview</time>  <h3><?php echo $\_GET['title']; ?></h3>  <p class="content"><?php echo $\_GET['content']; ?></p>  <p class="author">Author: <?php echo $\_SESSION['user']->getFullname(); ?></p>  </article>  <?php } else { ?>  <a id='addpost\_button' href='addEntry.php'>Add Post</a>  <section id='blog'>  <?php }  for ($i = 0; $i < count($blogs); $i++) {  echo "<article id=".$blogs[$i]->id." >";  echo "<time datetime='".$blogs[$i]->date."'>".$blogs[$i]->getLongDate\_toString()."</time>";  echo "<h3>".$blogs[$i]->title."</h3>";  echo "<p class='content'>".$blogs[$i]->content."</p>";  if (isset($\_SESSION['user']) && ($\_SESSION['user']->isAdmin())) {  echo '<a class="post\_comment\_delete" href="scripts/php/deleteBlog.php?blogId='.$blogs[$i]->id.'"><i class="fa fa-trash"></i></a>';  }  echo "<p class='author'>Author: ".$blogs[$i]->author->getFullName()."</p>";  echo "<a class='comments\_button' href='viewBlog.php?id=".$blogs[$i]->id."'><i class='fa fa-comments-o'></i><p>".count($blogs[$i]->comments)."</p></a>";  echo "</article>";  }  if (count($blogs) === 0) { ?>  <article id="no\_posts">  <p class="content">No posts found</p>  </article>  <?php }  echo "</section>";  }  include 'scripts/php/footer.php';  ?>  </div>  </body>  </html> | The functions orderByDateDesc, merge and mergeSort was generated using AI Prompt using ChatGPT. The prompt was the following…  “In the programming language php, write a Merge sort sorting algorithm, as a function, called orderByDateDesc, with an object array as parameter. The object has the attribute $date which is of type Time. Order the array in descending order by $date. Then return the array object.”.  The mergesort function is a recursive function that divides the array half by half until each sub-array only contains one element. The array is passed as reference each time.  The merge function merges two sorted sub-arrays, left side and right side. The array is passed as reference each time. It is used at the end of each mergeSort call, hence with the mergeSort function can split the array n number of times, then track back by calling the merge function n number of times to put it back together, till the original call of the function when it ends.  The function objectByDateDesc simply initiates the merge sort by calling mergeSort with the object array passed as a reference along with 0 as start index (left) and length of array – 1 as end index (right). Then the object array is returned.  The function is then used to sort Post read from the database in viewBlog.php, when the blog page outputs the posts, when “($\_GET['id']” is undefined.    Figure 1: Before using sorting algorithm.    Figure 2: After using the sorting algorithm. |

|  |  |
| --- | --- |
| **Assessment Criteria 8 - Extra Feature 1**  (Note: delete any of the two features below)   * Allow blog viewers to log in and add comments to the entries. You, as the administrator of the blog should be able to delete entries or comments. | |
| Instruction: use this column to paste the code from the files that have been used to develop this functionality.  **Paste the code directly from visual studio code, do not include a screenshot of the code as this document has a Turnitin requirement and this will lead to an automatic mark of zero. See example below.** | Instruction: use this column to address the following in your own words:   1. How did you develop this feature? Describe the steps involved. 2. Describe the inner workings of the approach you developed.   Give reference to the code you have written when answering the questions above.  You can use diagrams (e.g. flow charts) or screenshots of your webpage output to illustrate your answer. |
| Filename: database.php  <?php  session\_start();  $db = new mysqli('localhost', 'RayaanUddin', 'Rayaan10', 'portfolio');  $permission\_levels = array(  0 => "User",  1 => "Admin"  );  // Class represents website user  class User {  public $id;  public $fname;  public $lname;  public $email;  private $permission;  public function \_\_construct($id) {  $this->id = $id;  global $db;  $sql = "SELECT \* FROM accounts WHERE accountId = '$id'";  $result = $db->query($sql);  if ($result->num\_rows === 1) {  $row = $result->fetch\_assoc();  $this->fname = $row["fname"];  $this->lname = $row["lname"];  $this->email = $row["email"];  $this->permission = $row['permission'];  }  }  public function getFullname() {  return $this->fname." ".$this->lname;  }  public function isAdmin() {  global $permission\_levels;  if ($permission\_levels[$this->permission] == "Admin") {  return true;  } else {  return false;  }  }  }  // Login to account  function login($email, $password) {  global $db;  $sql = "SELECT \* FROM accounts WHERE email = '".$email."' AND password = '".$password."'";  $result = $db->query($sql);  if ($result->num\_rows === 1) {  $row = $result->fetch\_assoc();  return new User($row['accountId']);  } else {  return false;  }  }  // Register a new account  function register($fname, $lname, $email, $password, $perm) {  global $db;  $sql = "SELECT \* FROM accounts WHERE email = '$email'";  $result = $db->query($sql);  if ($result->num\_rows === 0) { // If account does not exists...  $sql = "INSERT INTO accounts (fname, lname, email, password, permission) VALUES ('$fname', '$lname', '$email', '$password', '$perm')";  if ($db->query($sql)) { // Should always execute unless issue on database  return true;  } else { // Exists for texting purposes  return false;  }  } else { // If account exists...  return false;  }  }  // Abstract class for Comment and Blog Postings.  abstract class Post {  public $id;  public $author;  public $date;  public function \_\_construct($id, $accountId, $date) {  $this->id = $id;  $this->author = new User($accountId);  $this->date = $date;  }  public function timeAgo\_toString() {  $datestr = "";  $yeardiff = date('Y') - date('Y', $this->date);  $monthdiff = date("m") - date("m", $this->date);  if ($monthdiff < 0) {  $yeardiff--;  $monthdiff+=12;  }  $datediff = date("d") - date("d", $this->date);  if ($datediff < 0) {  $monthdiff--;  $datediff+=30;  }  $hourdiff = date("H") - date("H", $this->date);  if ($hourdiff < 0) {  $datediff--;  $hourdiff+=24;  }  $minutediff = date("i") - date("i", $this->date);  if ($minutediff < 0) {  $hourdiff--;  $minutediff+=60;  }  $seconddiff = date("s") - date("s", $this->date);  if ($seconddiff < 0) {  $minutediff--;  $seconddiff+=60;  }  if ($yeardiff > 0) {  $datestr .= $yeardiff." year".($yeardiff > 1 ? "s " : " ");  } elseif ($monthdiff > 0) {  $datestr .= $monthdiff." month".($monthdiff > 1 ? "s " : " ");  } elseif ($datediff > 0) {  $datestr .= $datediff." day".($datediff > 1 ? "s " : " ");  } elseif ($hourdiff > 0) {  $datestr .= $hourdiff." hour".($hourdiff > 1 ? "s " : " ");  } elseif ($minutediff > 0) {  $datestr .= $minutediff." minute".($minutediff > 1 ? "s " : " ");  } elseif ($seconddiff > 0) {  $datestr .= $seconddiff." second".($seconddiff > 1 ? "s " : " ");  }  if ($datestr == "") {  return "Just now";  }  return $datestr."ago";  }  public function getLongDate\_toString() {  return date('dS F Y | H:i', $this->date);  }  }  class Comment extends Post {  public $comment;  public function \_\_construct($comment, $authorId, $date, $id) {  $this->comment = $comment;  parent::\_\_construct($id, $authorId, $date);  }  }  class Blog extends Post {  public $title;  public $content;  public $comments;  private function getComments() {  global $db;  $sql = "SELECT \* FROM comments WHERE blogId = '".$this->id."'";  $result = $db->query($sql);  $comments = array();  if ($result->num\_rows > 0) {  while ($row = $result->fetch\_assoc()) {  $comment = new Comment($row['comment'], $row['accountId'], strtotime($row['date']), $row['commentId']);  array\_push($comments, $comment);  }  }  return $comments;  }  public function \_\_construct($title, $content, $authorId, $date, $id) {  $this->title = $title;  $this->content = $content;  parent::\_\_construct($id, $authorId, $date);  // Get posts comments  $this->comments = orderByDateDesc($this->getComments());  }  }  // AI Generated sorting algorithm  // Start  function mergeSort(&$arr, $left, $right) {  if ($left < $right) {  $mid = (int)(($left + $right) / 2);  mergeSort($arr, $left, $mid);  mergeSort($arr, $mid + 1, $right);  merge($arr, $left, $mid, $right);  }  }  function merge(&$arr, $left, $mid, $right) {  $n1 = $mid - $left + 1;  $n2 = $right - $mid;  $L = [];  $R = [];  for ($i = 0; $i < $n1; $i++)  $L[$i] = $arr[$left + $i];  for ($j = 0; $j < $n2; $j++)  $R[$j] = $arr[$mid + 1 + $j];  $i = 0;  $j = 0;  $k = $left;  while ($i < $n1 && $j < $n2) {  if ($L[$i]->date >= $R[$j]->date) {  $arr[$k] = $L[$i];  $i++;  } else {  $arr[$k] = $R[$j];  $j++;  }  $k++;  }  while ($i < $n1) {  $arr[$k] = $L[$i];  $i++;  $k++;  }  while ($j < $n2) {  $arr[$k] = $R[$j];  $j++;  $k++;  }  }  function orderByDateDesc($objects) {  $length = count($objects);  mergeSort($objects, 0, $length - 1);  return $objects;  }  // End  // Gets all blogs from database  function getAllBlogs() {  global $db;  $blog\_array = array();  $sql = "SELECT \* FROM blog";  $result = $db->query($sql);  if ($result->num\_rows > 0) {  while ($row = $result->fetch\_assoc()) {  $date = strtotime($row['date']);  $blog = new Blog($row['title'], $row['content'], $row['accountId'], $date, $row['blogId']);  array\_push($blog\_array, $blog);  }  }  return $blog\_array;  }  function addBlog($title, $content, $author) {  global $db;  $date = date('Y-m-d H:i:s');  $sql = "INSERT INTO blog (title, content, accountId, date) VALUES ('$title', '$content', '$author', '$date')";  if ($db->query($sql) == true) {  return true;  } else {  return false;  }  }  // Gets a blog by id  function getBlog($id) {  global $db;  $sql = "SELECT \* FROM blog WHERE blogId = '".$id."'";  $result = $db->query($sql);  if ($result->num\_rows === 1) {  $row = $result->fetch\_assoc();  $date = strtotime($row['date']);  $blog = new Blog($row['title'], $row['content'], $row['accountId'], $date, $row['blogId']);  return $blog;  } else {  return null;  }  }  // Deletes a blog by id  function deleteBlog($id) {  global $db;  if (isset($\_SESSION["user"])) {  if ($\_SESSION["user"]->isAdmin()) { // Only admins can delete blogs  // Delete all comments first from blog post..  $sql = "DELETE FROM comments WHERE blogId = '".$id."'";  if ($db->query($sql) == true) {  // Then delete the blog post..  $sql = "DELETE FROM blog WHERE blogId = '".$id."'";  if ($db->query($sql) == true) {  return true;  } else {  return false;  }  } else {  return false;  }  } else {  return false;  }  } else {  return false;  }  }  // Add Comment to blog  function addComment($id, $content, $author) {  global $db;  $date = date('Y-m-d H:i:s');  $sql = "INSERT INTO comments (comment, blogId, accountId, date) VALUES ('$content', '$id', '$author', '$date')";  if ($db->query($sql) == true) {  echo "Comment added";  } else {  echo "Error: " . $sql . "<br>" . $db->error;  }  }  // Delete comment by id  function deleteComment($id) {  global $db;  if (isset($\_SESSION["user"])) {  if ($\_SESSION["user"]->isAdmin()) {  } else {  $accountId = $\_SESSION["user"]->id;  $sql = "SELECT commentId FROM comments WHERE commentId = '".$id."' AND accountId = '".$accountId."'";  $result = $db->query($sql);  if ($result->num\_rows === 1) {  } else {  return false;  }  }  $sql = "DELETE FROM comments WHERE commentId = '".$id."'";  if ($db->query($sql) == true) {  return true;  } else {  return false;  }  } else {  return false;  }  }  ?>  Filename: viewBlog.php  <?php  include 'scripts/php/database.php';  ?>  <!DOCTYPE html>  <html lang="en">  <?php include 'scripts/php/head.php'; ?>  <body>  <div class="container">  <?php  include 'scripts/php/header.php';  ?>  <section id="blog-error">  <?php  if (isset($\_GET['error'])) {  echo "<p>Error: ".$\_GET['error']."</p>";  }  ?>  </section>  <?php  if (isset($\_GET['id'])) {  $blog = getBlog($\_GET['id']);  if ($blog != null) {  echo "<article id='blog\_full'>";  echo "<nav id='blog-nav'>";  echo "<a href='viewBlog.php#".$\_GET['id']."'><i class='fa fa-arrow-left'></i><p>Back</p></a>";  echo "</nav>";  echo "<time datetime='".$blog->date."'>".$blog->getLongDate\_toString()."</time>";  echo "<h3>".$blog->title."</h3>";  echo "<p class='content'>".$blog->content."</p>";  echo "<p class='author'>Author: ".$blog->author->getFullname()."</p>";  echo "<div id='comments'>";  echo "<form action='scripts/php/addComment.php' method='get'>";  echo "<textarea name='comment' placeholder='Enter comment here'></textarea>";  echo "<input type='hidden' name='id' value='".$blog->id."' required>";  echo "<input type='submit' value='Send'>";  echo "</form>";  echo "<ul id='comments\_list'>";  for ($i = 0; $i < count($blog->comments); $i++) {  echo "<li class='comment'>";  echo "<p class='comment\_author'>".$blog->comments[$i]->author->getFullname()."</p>";  echo "<p class='comment\_content'>".$blog->comments[$i]->comment."</p>";  echo "<time datetime='".$blog->comments[$i]->date."'>".$blog->comments[$i]->timeAgo\_toString()."</time>";  if (isset($\_SESSION['user']) && ($blog->comments[$i]->author->id == $\_SESSION['user']->id || $\_SESSION['user']->isAdmin())) {  echo '<a class="post\_comment\_delete" href="scripts/php/deleteBlog.php?commentId='.$blog->comments[$i]->id.'&blogId='.$\_GET['id'].'"><i class="fa fa-trash"></i></a>';  }  echo "</li>";  }  echo "</ul>";  echo "</div>";  echo "</article>";  }  else {  echo "<p>Blog not found</p>";  }  }  else {  $blogs = orderByDateDesc(getAllBlogs());  if (isset($\_GET['title']) && isset($\_GET['content']) && isset($\_SESSION['user']) && $\_SESSION['user']->isAdmin()) { // Preview Blog  ?>  <nav id="blog-nav">  <a href="addEntry.php?title=<?php echo $\_GET['title']; ?>&content=<?php echo $\_GET['content']; ?>"><i class='fa fa-arrow-left'></i><p>Back</p></a>  <p>Preview</p>  <a href="scripts/php/addPost.php?title=<?php echo $\_GET['title']; ?>&content=<?php echo $\_GET['content']; ?>"><p>Post</p><i class='fa fa-check'></i></a>  </nav>  <a id='addpost\_button' href='addEntry.php'>Add Post</a>  <section id='blog'>  <article id="preview">  <time>Preview</time>  <h3><?php echo $\_GET['title']; ?></h3>  <p class="content"><?php echo $\_GET['content']; ?></p>  <p class="author">Author: <?php echo $\_SESSION['user']->getFullname(); ?></p>  </article>  <?php } else { ?>  <a id='addpost\_button' href='addEntry.php'>Add Post</a>  <section id='blog'>  <?php }  for ($i = 0; $i < count($blogs); $i++) {  echo "<article id=".$blogs[$i]->id." >";  echo "<time datetime='".$blogs[$i]->date."'>".$blogs[$i]->getLongDate\_toString()."</time>";  echo "<h3>".$blogs[$i]->title."</h3>";  echo "<p class='content'>".$blogs[$i]->content."</p>";  if (isset($\_SESSION['user']) && ($\_SESSION['user']->isAdmin())) {  echo '<a class="post\_comment\_delete" href="scripts/php/deleteBlog.php?blogId='.$blogs[$i]->id.'"><i class="fa fa-trash"></i></a>';  }  echo "<p class='author'>Author: ".$blogs[$i]->author->getFullName()."</p>";  echo "<a class='comments\_button' href='viewBlog.php?id=".$blogs[$i]->id."'><i class='fa fa-comments-o'></i><p>".count($blogs[$i]->comments)."</p></a>";  echo "</article>";  }  if (count($blogs) === 0) { ?>  <article id="no\_posts">  <p class="content">No posts found</p>  </article>  <?php }  echo "</section>";  }  include 'scripts/php/footer.php';  ?>  </div>  </body>  </html>  Filename: createaccount.php  <?php  include("database.php");  session\_unset();  $email = $\_POST['email'];  $password = $\_POST['password'];  $fname = $\_POST['fname'];  $lname = $\_POST['lname'];  if (register($fname, $lname, $email, $password, 0)) {  header('Location: ../../login.php?error=Account created successfully');  exit();  } else {  header('Location: ../../signup.php?error=Account already exists');  exit();  }  ?> | **I did NOT use any Generative AI tools to achieve this.**  Upon loading the posts in viewBlog.php, it is checked if the user is an admin, if so, the user can delete the posts, hence a button with the link to delete the post is displayed to the user. Furthermore, a comment button is added to the left of every post along with a number, representing number of comments existing for that post.  When clicking the comment button, user is redirected to viewBlog,php along with a variable of the post id using query string. Everytime the blog page is loaded it is checked if there is a GET variable, “id”, which exists. If so, that post is only displayed along with its comments and an option to add a comment, which any user can if they are logged in.  Any author of a comment or admin can delete the comment(s), by clicking the delete button which I displayed on those conditions.  Deleting posts works by redirecting user to deleteBlog.php with post id using a query string. The deleteBlog.php uses the deletePost function within the database.php to delete the post from the database by sending the post id as an argument. The function first checks that the user is an admin since only admins can delete posts. If the user is an admin, it deletes the posts (returns true) and redirects back to viewBlog.php, else it returns false and the deleteBlog.php displays an error on the viewBlog page which it redirects back to using a query string with the error.  Similarly, the same process is taken using deleting a comment, however the author or an admin can delete it and the comment id is passed along with the post id in the query string. At the start of the deleteBlog.php, it is always checked if the commentId GET exists, since then you are deleting the comment, otherwise its deleting a post.    Figure 3: shows process in action.    Figure 4: Logged in as admin on comment page for a post.  To allow users to add comments, I have also implemented a signup form (signup.php), which takes the required details from a user (email, first & last name, password) and then uses the method within the database to create the account, which first checks if an account with the same email already exists. The method is used by createaccount.php which is what the form POSTs to when the form is submitted.    Figure 5: When the account already exists, it uses a query string to return a result.  The signup form uses html verification to make sure that the inputs are valid.    Figure 6: When an account is created successfully.  Since this feature enables users to create and own accounts, I have also implemented a selection which checks if user is admin or not upon logging in, since if the user is not admin, it is not redirected to addEntry.php. |

|  |  |
| --- | --- |
| **Assessment Criteria 8 - Extra Feature 2**  (Note: delete any of the two features below)   * Add a “preview” button in addpost. When this button is clicked, the new entry is previewed, and you can then decide (via a set of navigational links) whether to upload the entry or go back to edit it. | |
| Instruction: use this column to paste the code from the files that have been used to develop this functionality.  **Paste the code directly from visual studio code, do not include a screenshot of the code as this document has a Turnitin requirement and this will lead to an automatic mark of zero. See example below.** | Instruction: use this column to address the following in your own words:   1. How did you develop this feature? Describe the steps involved. 2. Describe the inner workings of the approach you developed.   Did you use any Generative AI tools to achieve this? If yes, can you provide the prompts you wrote to achieve this?  Give reference to the code you have written when answering the questions above.  You can use diagrams (e.g. flow charts) or screenshots of your webpage output to illustrate your answer. |
| Filename: addEntry.php  <?php  include "scripts/php/database.php";  if (!isset($\_SESSION['user'])) {  header("Location: login.php?error=Required to login first");  exit();  } else {  if (!$\_SESSION['user']->isAdmin()) {  header("Location: viewBlog.php?error=You are not allowed to post a blog. Must be an admin.");  exit();  }  }  ?>  <!DOCTYPE html>  <html lang="en">  <?php include 'scripts/php/head.php'; ?>  <body>  <div class="form-container">  <?php  include 'scripts/php/header.php';  ?>  <form action="scripts/php/addPost.php" method="get">  <h1>Add Blog</h1>  <div class="wrap-input">  <input type="text" name="title" id="title" placeholder="" value="<?php echo (isset($\_GET['title']) ? $\_GET['title'] : "") ?>">  <label for="title">Title</label>  </div>  <textarea name="content" id="content" placeholder="Enter your text here" rows="5"><?php echo (isset($\_GET['content']) ? $\_GET['content'] : "") ?></textarea>  <?php if (isset($\_GET['error'])) {  echo "<div class='error'><p>".$\_GET['error']."</p></div>";  }  ?>  <div id="blog-post-input-container">  <input type="submit" value="Post"/>  <input type="reset" value="Clear"/>  </div>  <button id="blog-post-preview">Preview</button>  </form>  <?php  include 'scripts/php/footer.php';  ?>  </div>  </body>  </html>  Filename: database.php  <?php  session\_start();  $db = new mysqli('localhost', 'RayaanUddin', 'Rayaan10', 'portfolio');  $permission\_levels = array(  0 => "User",  1 => "Admin"  );  // Class represents website user  class User {  public $id;  public $fname;  public $lname;  public $email;  private $permission;  public function \_\_construct($id) {  $this->id = $id;  global $db;  $sql = "SELECT \* FROM accounts WHERE accountId = '$id'";  $result = $db->query($sql);  if ($result->num\_rows === 1) {  $row = $result->fetch\_assoc();  $this->fname = $row["fname"];  $this->lname = $row["lname"];  $this->email = $row["email"];  $this->permission = $row['permission'];  }  }  public function getFullname() {  return $this->fname." ".$this->lname;  }  public function isAdmin() {  global $permission\_levels;  if ($permission\_levels[$this->permission] == "Admin") {  return true;  } else {  return false;  }  }  }  // Login to account  function login($email, $password) {  global $db;  $sql = "SELECT \* FROM accounts WHERE email = '".$email."' AND password = '".$password."'";  $result = $db->query($sql);  if ($result->num\_rows === 1) {  $row = $result->fetch\_assoc();  return new User($row['accountId']);  } else {  return false;  }  }  // Register a new account  function register($fname, $lname, $email, $password, $perm) {  global $db;  $sql = "SELECT \* FROM accounts WHERE email = '$email'";  $result = $db->query($sql);  if ($result->num\_rows === 0) { // If account does not exists...  $sql = "INSERT INTO accounts (fname, lname, email, password, permission) VALUES ('$fname', '$lname', '$email', '$password', '$perm')";  if ($db->query($sql)) { // Should always execute unless issue on database  return true;  } else { // Exists for texting purposes  return false;  }  } else { // If account exists...  return false;  }  }  // Abstract class for Comment and Blog Postings.  abstract class Post {  public $id;  public $author;  public $date;  public function \_\_construct($id, $accountId, $date) {  $this->id = $id;  $this->author = new User($accountId);  $this->date = $date;  }  public function timeAgo\_toString() {  $datestr = "";  $yeardiff = date('Y') - date('Y', $this->date);  $monthdiff = date("m") - date("m", $this->date);  if ($monthdiff < 0) {  $yeardiff--;  $monthdiff+=12;  }  $datediff = date("d") - date("d", $this->date);  if ($datediff < 0) {  $monthdiff--;  $datediff+=30;  }  $hourdiff = date("H") - date("H", $this->date);  if ($hourdiff < 0) {  $datediff--;  $hourdiff+=24;  }  $minutediff = date("i") - date("i", $this->date);  if ($minutediff < 0) {  $hourdiff--;  $minutediff+=60;  }  $seconddiff = date("s") - date("s", $this->date);  if ($seconddiff < 0) {  $minutediff--;  $seconddiff+=60;  }  if ($yeardiff > 0) {  $datestr .= $yeardiff." year".($yeardiff > 1 ? "s " : " ");  } elseif ($monthdiff > 0) {  $datestr .= $monthdiff." month".($monthdiff > 1 ? "s " : " ");  } elseif ($datediff > 0) {  $datestr .= $datediff." day".($datediff > 1 ? "s " : " ");  } elseif ($hourdiff > 0) {  $datestr .= $hourdiff." hour".($hourdiff > 1 ? "s " : " ");  } elseif ($minutediff > 0) {  $datestr .= $minutediff." minute".($minutediff > 1 ? "s " : " ");  } elseif ($seconddiff > 0) {  $datestr .= $seconddiff." second".($seconddiff > 1 ? "s " : " ");  }  if ($datestr == "") {  return "Just now";  }  return $datestr."ago";  }  public function getLongDate\_toString() {  return date('dS F Y | H:i', $this->date);  }  }  class Comment extends Post {  public $comment;  public function \_\_construct($comment, $authorId, $date, $id) {  $this->comment = $comment;  parent::\_\_construct($id, $authorId, $date);  }  }  class Blog extends Post {  public $title;  public $content;  public $comments;  private function getComments() {  global $db;  $sql = "SELECT \* FROM comments WHERE blogId = '".$this->id."'";  $result = $db->query($sql);  $comments = array();  if ($result->num\_rows > 0) {  while ($row = $result->fetch\_assoc()) {  $comment = new Comment($row['comment'], $row['accountId'], strtotime($row['date']), $row['commentId']);  array\_push($comments, $comment);  }  }  return $comments;  }  public function \_\_construct($title, $content, $authorId, $date, $id) {  $this->title = $title;  $this->content = $content;  parent::\_\_construct($id, $authorId, $date);  // Get posts comments  $this->comments = orderByDateDesc($this->getComments());  }  }  // AI Generated sorting algorithm  // Start  function mergeSort(&$arr, $left, $right) {  if ($left < $right) {  $mid = (int)(($left + $right) / 2);  mergeSort($arr, $left, $mid);  mergeSort($arr, $mid + 1, $right);  merge($arr, $left, $mid, $right);  }  }  function merge(&$arr, $left, $mid, $right) {  $n1 = $mid - $left + 1;  $n2 = $right - $mid;  $L = [];  $R = [];  for ($i = 0; $i < $n1; $i++)  $L[$i] = $arr[$left + $i];  for ($j = 0; $j < $n2; $j++)  $R[$j] = $arr[$mid + 1 + $j];  $i = 0;  $j = 0;  $k = $left;  while ($i < $n1 && $j < $n2) {  if ($L[$i]->date >= $R[$j]->date) {  $arr[$k] = $L[$i];  $i++;  } else {  $arr[$k] = $R[$j];  $j++;  }  $k++;  }  while ($i < $n1) {  $arr[$k] = $L[$i];  $i++;  $k++;  }  while ($j < $n2) {  $arr[$k] = $R[$j];  $j++;  $k++;  }  }  function orderByDateDesc($objects) {  $length = count($objects);  mergeSort($objects, 0, $length - 1);  return $objects;  }  // End  // Gets all blogs from database  function getAllBlogs() {  global $db;  $blog\_array = array();  $sql = "SELECT \* FROM blog";  $result = $db->query($sql);  if ($result->num\_rows > 0) {  while ($row = $result->fetch\_assoc()) {  $date = strtotime($row['date']);  $blog = new Blog($row['title'], $row['content'], $row['accountId'], $date, $row['blogId']);  array\_push($blog\_array, $blog);  }  }  return $blog\_array;  }  function addBlog($title, $content, $author) {  global $db;  $date = date('Y-m-d H:i:s');  $sql = "INSERT INTO blog (title, content, accountId, date) VALUES ('$title', '$content', '$author', '$date')";  if ($db->query($sql) == true) {  return true;  } else {  return false;  }  }  // Gets a blog by id  function getBlog($id) {  global $db;  $sql = "SELECT \* FROM blog WHERE blogId = '".$id."'";  $result = $db->query($sql);  if ($result->num\_rows === 1) {  $row = $result->fetch\_assoc();  $date = strtotime($row['date']);  $blog = new Blog($row['title'], $row['content'], $row['accountId'], $date, $row['blogId']);  return $blog;  } else {  return null;  }  }  // Deletes a blog by id  function deleteBlog($id) {  global $db;  if (isset($\_SESSION["user"])) {  if ($\_SESSION["user"]->isAdmin()) { // Only admins can delete blogs  // Delete all comments first from blog post..  $sql = "DELETE FROM comments WHERE blogId = '".$id."'";  if ($db->query($sql) == true) {  // Then delete the blog post..  $sql = "DELETE FROM blog WHERE blogId = '".$id."'";  if ($db->query($sql) == true) {  return true;  } else {  return false;  }  } else {  return false;  }  } else {  return false;  }  } else {  return false;  }  }  // Add Comment to blog  function addComment($id, $content, $author) {  global $db;  $date = date('Y-m-d H:i:s');  $sql = "INSERT INTO comments (comment, blogId, accountId, date) VALUES ('$content', '$id', '$author', '$date')";  if ($db->query($sql) == true) {  echo "Comment added";  } else {  echo "Error: " . $sql . "<br>" . $db->error;  }  }  // Delete comment by id  function deleteComment($id) {  global $db;  if (isset($\_SESSION["user"])) {  if ($\_SESSION["user"]->isAdmin()) {  } else {  $accountId = $\_SESSION["user"]->id;  $sql = "SELECT commentId FROM comments WHERE commentId = '".$id."' AND accountId = '".$accountId."'";  $result = $db->query($sql);  if ($result->num\_rows === 1) {  } else {  return false;  }  }  $sql = "DELETE FROM comments WHERE commentId = '".$id."'";  if ($db->query($sql) == true) {  return true;  } else {  return false;  }  } else {  return false;  }  }  ?>  Filename: viewBlog.php  <?php  include 'scripts/php/database.php';  ?>  <!DOCTYPE html>  <html lang="en">  <?php include 'scripts/php/head.php'; ?>  <body>  <div class="container">  <?php  include 'scripts/php/header.php';  ?>  <section id="blog-error">  <?php  if (isset($\_GET['error'])) {  echo "<p>Error: ".$\_GET['error']."</p>";  }  ?>  </section>  <?php  if (isset($\_GET['id'])) {  $blog = getBlog($\_GET['id']);  if ($blog != null) {  echo "<article id='blog\_full'>";  echo "<nav id='blog-nav'>";  echo "<a href='viewBlog.php#".$\_GET['id']."'><i class='fa fa-arrow-left'></i><p>Back</p></a>";  echo "</nav>";  echo "<time datetime='".$blog->date."'>".$blog->getLongDate\_toString()."</time>";  echo "<h3>".$blog->title."</h3>";  echo "<p class='content'>".$blog->content."</p>";  echo "<p class='author'>Author: ".$blog->author->getFullname()."</p>";  echo "<div id='comments'>";  echo "<form action='scripts/php/addComment.php' method='get'>";  echo "<textarea name='comment' placeholder='Enter comment here'></textarea>";  echo "<input type='hidden' name='id' value='".$blog->id."' required>";  echo "<input type='submit' value='Send'>";  echo "</form>";  echo "<ul id='comments\_list'>";  for ($i = 0; $i < count($blog->comments); $i++) {  echo "<li class='comment'>";  echo "<p class='comment\_author'>".$blog->comments[$i]->author->getFullname()."</p>";  echo "<p class='comment\_content'>".$blog->comments[$i]->comment."</p>";  echo "<time datetime='".$blog->comments[$i]->date."'>".$blog->comments[$i]->timeAgo\_toString()."</time>";  if (isset($\_SESSION['user']) && ($blog->comments[$i]->author->id == $\_SESSION['user']->id || $\_SESSION['user']->isAdmin())) {  echo '<a class="post\_comment\_delete" href="scripts/php/deleteBlog.php?commentId='.$blog->comments[$i]->id.'&blogId='.$\_GET['id'].'"><i class="fa fa-trash"></i></a>';  }  echo "</li>";  }  echo "</ul>";  echo "</div>";  echo "</article>";  }  else {  echo "<p>Blog not found</p>";  }  }  else {  $blogs = orderByDateDesc(getAllBlogs());  if (isset($\_GET['title']) && isset($\_GET['content']) && isset($\_SESSION['user']) && $\_SESSION['user']->isAdmin()) { // Preview Blog  ?>  <nav id="blog-nav">  <a href="addEntry.php?title=<?php echo $\_GET['title']; ?>&content=<?php echo $\_GET['content']; ?>"><i class='fa fa-arrow-left'></i><p>Back</p></a>  <p>Preview</p>  <a href="scripts/php/addPost.php?title=<?php echo $\_GET['title']; ?>&content=<?php echo $\_GET['content']; ?>"><p>Post</p><i class='fa fa-check'></i></a>  </nav>  <a id='addpost\_button' href='addEntry.php'>Add Post</a>  <section id='blog'>  <article id="preview">  <time>Preview</time>  <h3><?php echo $\_GET['title']; ?></h3>  <p class="content"><?php echo $\_GET['content']; ?></p>  <p class="author">Author: <?php echo $\_SESSION['user']->getFullname(); ?></p>  </article>  <?php } else { ?>  <a id='addpost\_button' href='addEntry.php'>Add Post</a>  <section id='blog'>  <?php }  for ($i = 0; $i < count($blogs); $i++) {  echo "<article id=".$blogs[$i]->id." >";  echo "<time datetime='".$blogs[$i]->date."'>".$blogs[$i]->getLongDate\_toString()."</time>";  echo "<h3>".$blogs[$i]->title."</h3>";  echo "<p class='content'>".$blogs[$i]->content."</p>";  if (isset($\_SESSION['user']) && ($\_SESSION['user']->isAdmin())) {  echo '<a class="post\_comment\_delete" href="scripts/php/deleteBlog.php?blogId='.$blogs[$i]->id.'"><i class="fa fa-trash"></i></a>';  }  echo "<p class='author'>Author: ".$blogs[$i]->author->getFullName()."</p>";  echo "<a class='comments\_button' href='viewBlog.php?id=".$blogs[$i]->id."'><i class='fa fa-comments-o'></i><p>".count($blogs[$i]->comments)."</p></a>";  echo "</article>";  }  if (count($blogs) === 0) { ?>  <article id="no\_posts">  <p class="content">No posts found</p>  </article>  <?php }  echo "</section>";  }  include 'scripts/php/footer.php';  ?>  </div>  </body>  </html> | **I did NOT use any Generative AI tools to achieve this.**  I created a preview button and styled it using html and CSS. Using JavaScript, upon a click event of the button preview (when the button is pressed) the script checks that the title and content is entered and not an empty string or just whitespaces.  If not, it alerts the site user to fill in all fields and highlights each input field to fill in.  If it is then it proceeds in redirecting the current page to viewBlog.php with the title and content variables using a query string. The viewBlog.php always checks upon loading if there is such case where title and content variables are given. If so it displays a preview of the comment along with a navigation section on top with options to go back to editing or posting. The preview post is highlighted. To go back to editing, it adds the same variables it GETs from the query string, within the hyperlink to addEntry.php, to allow the user to edit the current content and not retype it all. When pressed post, it adds the content and title to the URL query string to the page addPost.php, same way the submit within the form in addEntry.php uses the method GET, to the page addPost.php, to add the entry to the blog.    Figure 7: A image of the preview in use.    Figure 8: A image of the preview in use when back button is pressed.    Figure 9: A image of the preview in use when the post button is pressed. |

|  |
| --- |
| **Assessment Criteria 14 and 15 - Identification of semantic HTML5 elements and importance of semantic markup**  What semantic HTML5 elements have been used within your portfolio site?  Why is semantic markup important?  You will need to provide examples related to your project to illustrate your answer. (max. 250 words) |
| Semantic markup is important for the content of your html rather than the visual structure/ style. It defines the meaning of the content placed within the tags. It also makes the markup language much more readable by humans, as it should be.  One semantic element I used was “header”. It is used to demonstrate the top of the page, like a banner, such as the title and navigation. Navigation, included within header, is also has a semantic element “nav”. It holds all the links to other pages to navigate through the portfolio.  Another semantic element used is the “footer”. Like the header, footer element which represents the bottom of the page content. This generally is to do with odd information such as, contacting, the copyright license and more.  Another semantic element used is “section”, which is used to group together related content. These would typically include a heading. An example is when I was creating my index.php, my home page, I have a section with the heading education and a section with the heading license and certifications, both group the content related to the headings.  Another semantic element used is “article”, which is used to contain standalone pieces of content. An example is within the viewBlog.php, article is used for each blog post.  Another semantic element used is “aside”, which contains content somewhat related to the block it is contained within, however it is either not as important or does not constitute to the main content. |

|  |
| --- |
| **Assessment Criteria 16 - Content organisation**  Explain your portfolio site's structure and navigation.  Justify the arrangement of the content and how it enhances user experience.  You should also include a site navigation diagram. |
| This is a site navigation diagram (Top-Down structure), which consists of all the pages within my portfolio in PURPLE, and its key content in grey. The website starts from Home Page.  Home Page: Contains main information, such as about me, experiences, licenses and certifications, and my current post (currently student). This is a good introduction to a user to my site to tell them all about me and my career goals.  Education: Contains all institutions I visited and gained qualifications from, stating which qualifications, what I have personally learnt and why, grades achieved, skills that I have gained and start/ end year of course. This is a page intended for users more interested about me, by clicking on the page in navigation bar.  Projects: Contains any projects and dates of when I have been working on it since and till. Presents my skills and abilities further along with interests within my work of field. Each project has a direct link to its repository on GitHub making it easier for user to access.  Blog: A blog page with posts to my users about my recent activities. This is at the end of the navigation on left most to make it stand out more. The add post page is placed on the page since it correlates for its function; it has no purpose anywhere else. On this page, users can also post comments on each blog post by opening the post.  Login: Allows user to login to add comments. It also has a link to access signup page since it correlates for its function; it has no purpose anywhere else. Login button is placed in the corner right most of the navigation bar as traditionally seen on other websites online to make it visible by the user.  The site is structured this way to present most important data “about me” first along then more personal information later. It has the blog at the end of the navigation bar as it is a page with a different concept compared to others. Each page includes contact information within its footer. |

|  |
| --- |
| **Assessment Criteria 17 and 18 – Design concept and rationale**  Describe the website design, including the theme and colour scheme.  What CSS style rules did you write to achieve this?  Did you use any Generative AI tools to achieve this? If yes, can you provide the prompts you wrote to achieve this?  Finally, provide a rationale behind the chosen design and how it reflects your personal brand. |
| **No AI tools were used to produce the design for the portfolio.**  The website has a clean design with the background primarily being back along with text being beige or white. The website uses a flexbox structure for all pages with a gap of 1rem between header, … and footer. This has been primarily achieved by using the declarations background-color, display, and color.  The design is consistent between all pages and the header stays the same between all. When active on a page, its name on the header is highlighted. This is done by applying the class to do so the relevant “li” (list) element of the unsorted list, which represents the pages of the portfolio, the navigation system, which makes it easier for the user to go through the website and know which page they are on.  The website consists of a scroll bar, which is beige when hovered or active, else grey. This is to make the scroll bar look the same on all devices and browsers, and allow to a professional look to the website. CSS can be used to do this by using the pseudo element ::-webkit-scrollbar.  The font styles are generally matching throughout the website where a nice soft round, and easy to read font style has been imported from google fonts. They are sans-serif fonts, Ubuntu & Varela Round.  **FORM**  The form pages (login.php, addEntry.php, signup.php) uses a white background to give a back shadow of the form. It also has interactive labels and input boxes to show where the user is selected and when there is an input within a box.  The CSS styles to display a back shadow of the form was box-shadow declared for the form elements.  For imperativeness, a transition declaration was used for the input boxes to for smooth transitions like the “border-bottom” declaration when the focus is upon the input box. Each button/ input type submit/ reset also use transition declaration to have a slow transition between the background changing upon hover.    **CONTENT LIST**  Both education and portfolio content are structured in a list with each section being a box of its own, using display flex, which allows for a one-dimensional list structure. Each content is displayed as a grid, with title, information, date, skills gained and more displayed accordingly in a two-dimensional state using the declaration display and grid-template-areas. There is a gap between each of 0.5 rem. Each content item also has its own background, which is done using its unique id and calling the declaration, background-image.  **INDEX (HOME) PAGE**  For my home page, I have decided to go for a different styling for the main heading, where it stands out more and is bigger since it’s my name and to show that it is the main page of the site, where in other pages, the heading is part of the head with the same styling as others. The home page includes a picture of myself with a short article about me. This is structured using a flexbox (display: flex; declaration) with the direction being horizontal using the declaration flex-direction with value row. The article is scrollable vertically, using the declaration overflow-y with value scroll.  The index page also includes experiences and certifications. The two are displayed within a flexbox display where each of them are represented by a list. There is a border between each list item to separate each item, using the declaration border-right set to a solid white with width of 0.2rem. The last element does not have the border. This is done by using a CSS pseudo class selector, “: first-child”. Note the flex-direction is set to row reversed which reverses the order of the li elements, hence the first child is the last child. This is done since we want the most recent experience or certificate/ license to be displayed at the front. The list items are viewable by scrolling. This is done by the declaration overflow-x: scroll.      The design is created this way for it to have a modern professional appearance, but also focusing on functionality of the use of this portfolio. Personally, dark, and off-white shades are what I like the most, in terms of even when wearing clothes, hence making my portfolio personal to me.  It is important to have a professional and minimalistic design to allow my skills to be outlined for the user, to have no distractions, and making it as easy as possible for the user to use the website. |

|  |
| --- |
| **Assessment Criteria 19 - Web technologies/framework**  For this project, you have made use of HTML, CSS, JavaScript and PHP. If you were to conduct this project again, which other web technologies/frameworks would you use to develop your portfolio site? Justify your choice. (max. 250 words) |
| If I was to conduct this project again, I would use React.js for frontend development due to its reusability of its components and excellent documentation. Moreover, it offers better performance compared to JavaScript, for UI, due to its virtual DOM (Document Object Model) rendering, where it compares the differences and updates necessary parts of DOM tree rather than all of it, which is what JavaScript does and is inefficient.  For server-side development, I would use Django since it is more secure with its built-in security. In terms if syntax, there are some things which I prefer in PHP compared to Django, however PHP can sometimes be a bit tedious. Moreover, Django offers much more built-in features. |