Project

<Time-Tracker>

CIS-17B

Name: Aleksandar Videv

Table Of Contents

**1.Introduction………………………………….……..……………..……….……3**

**2.** **Overview………………………………….………………………….….…..….3**

**2.1 Project Summary…………………….………………..………………4**

**2.2 Versioning Scheme…………………….……………………………..3**

**3.** **Project Description…………………….…………………………………….…4**

**4. Gantt Chart…………………….………………………………………........…5**

**5. UML Diagrams…………….……………………….…………………………..8**

6.Demonstrating Functionality**………………..………….….……..….….…….9**

6.1 Proof of Concept.**…..................………….….….…….…..…......…..9**

6.2 Testing Results…**…………….………….….….….….….….……...36**

**1.Introduction**

The Time Tracker is an advanced web application designed to assist users in tracking their time efficiently within educational or professional settings. The application offers user authentication, detailed time tracking, and interactive menus to facilitate productivity and accountability. By providing precise tracking functionalities and user management capabilities, Time Tracker ensures that users can focus on their tasks with enhanced efficiency.

**2. Overview**

**2.1 Project Summary**

The Time Tracker application is developed using a variety of web technologies to provide a robust and user-friendly experience. Key metrics and insights about the project are as follows:

* **GitHub Repository**: [Link](https://github.com/AlexanderVidev/Time-Tracker)
* **Total Lines of Code**: 1191
* **PHP**
  + **Variables:** 20 (9 unique member variables, 11 local variables)
  + **Methods:** 13
* **JavaScript**
  + **Variables:** 41 (13 unique member variables, 28 local variables)
  + **Methods:** 27
* **Combined Total**
  + **Variables:** 61 (22 unique member variables, 39 local variables)
  + **Methods:** 40

**Project Insights:** Developed using HTML, CSS, JavaScript, PHP, and MySQL, the Time Tracker leverages various standard libraries to provide robust functionalities such as user data management and timer operations. Approximately 60 hours have been invested in the design and coding phases, excluding additional hours for documentation and debugging. The project is characterized by continuous incremental updates to improve its usability and adapt to the evolving needs of its users.

**2.2 Versioning Scheme**

**Version 10:** Implements the basic structure and functionalities for time tracking, including user and time log management with minimal server interaction.

**Version 11:** Introduces server interaction for fetching existing time, updating logged time dynamically, and allows user account editing.

**Version 12:** Enhances user registration validation with JSON-based responses and improves user management with detailed editing and a better user interface for usability.

**Version 13:** Adds user deletion and creation functionalities, enhancing the admin interface.

**Version 14:** Introduces account creation within the admin interface and improves sorting functionality for user lists.

**Version 15:** Enhances user interface with sorting capabilities and improved validation for user management.

**Version 16:** Implements input validation for user registration and account editing, enhancing security and user data integrity.

**Version 17:** Enhances the user interface with a new dashboard that provides a graphical summary of time logs, including daily, weekly, and monthly summaries.

**Version 18:** Integrates a notification system that alerts users about session start/stop, invalid login attempts, and updates on account changes.

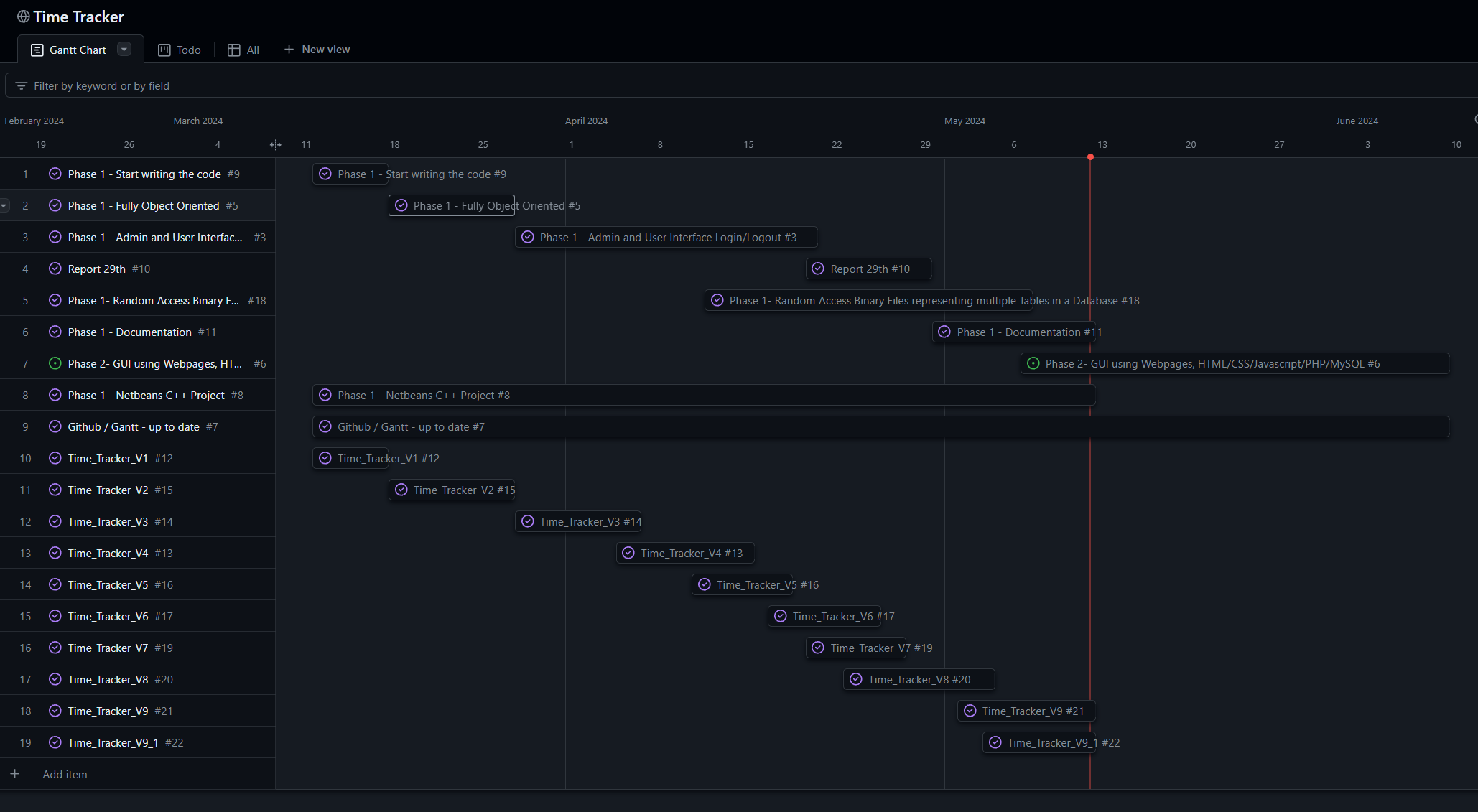
**3. Project Description**

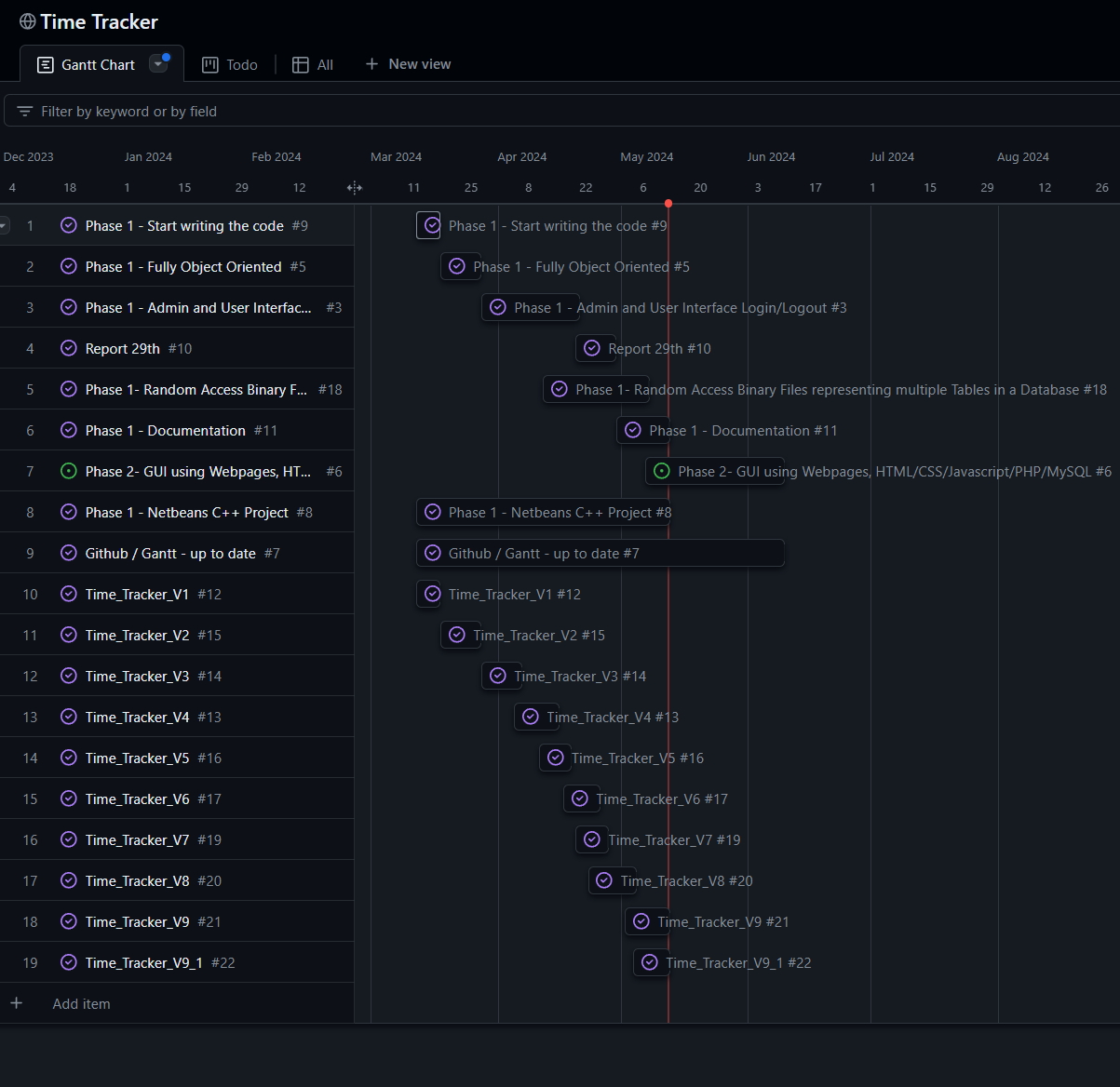
The Time Tracker web application offers several key features and improvements to provide robust functionality and an enhanced user experience:

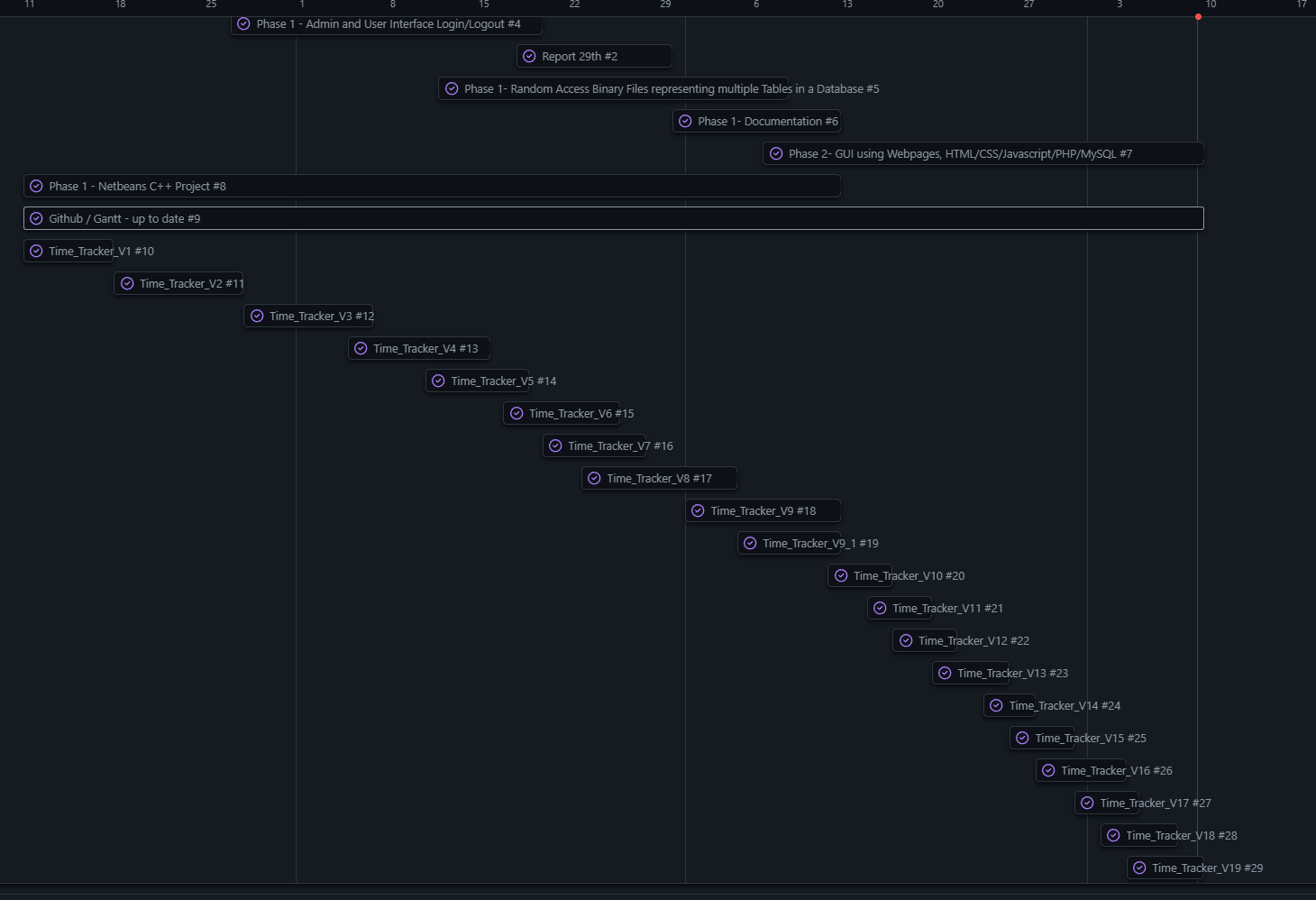
* **Comprehensive Time Tracking:** The application ensures accurate and precise time logging for users, with features that prevent multiple concurrent timers and manage session durations effectively.
* **Advanced User Management**: Administrators can create, edit, and delete user accounts, assign roles, and manage class information seamlessly. This flexibility accommodates diverse user needs and scenarios, such as role changes and class assignments.
* **Intuitive User Interface:** The interface has been designed to be user-friendly, with interactive menus guiding users through various actions such as starting/stopping timers, viewing logs, and editing account details. This reduces the likelihood of user errors and enhances the overall experience.
* **Secure Data Handling:** The application employs encryption for password storage and session management to ensure data security. Robust error handling and form validation techniques are implemented to maintain data integrity and protect against invalid inputs.
* **Database Integration:** The application uses MySQL for persistent data storage, with a well-structured schema that includes user information, time logs, and session data. The database design ensures efficient data retrieval and manipulation.
* **Dynamic Content Management:** Using JavaScript, the application dynamically updates user interfaces based on interactions, providing real-time feedback and data updates without needing full-page reloads. This improves the responsiveness and usability of the application.

**4. Gantt Chart**

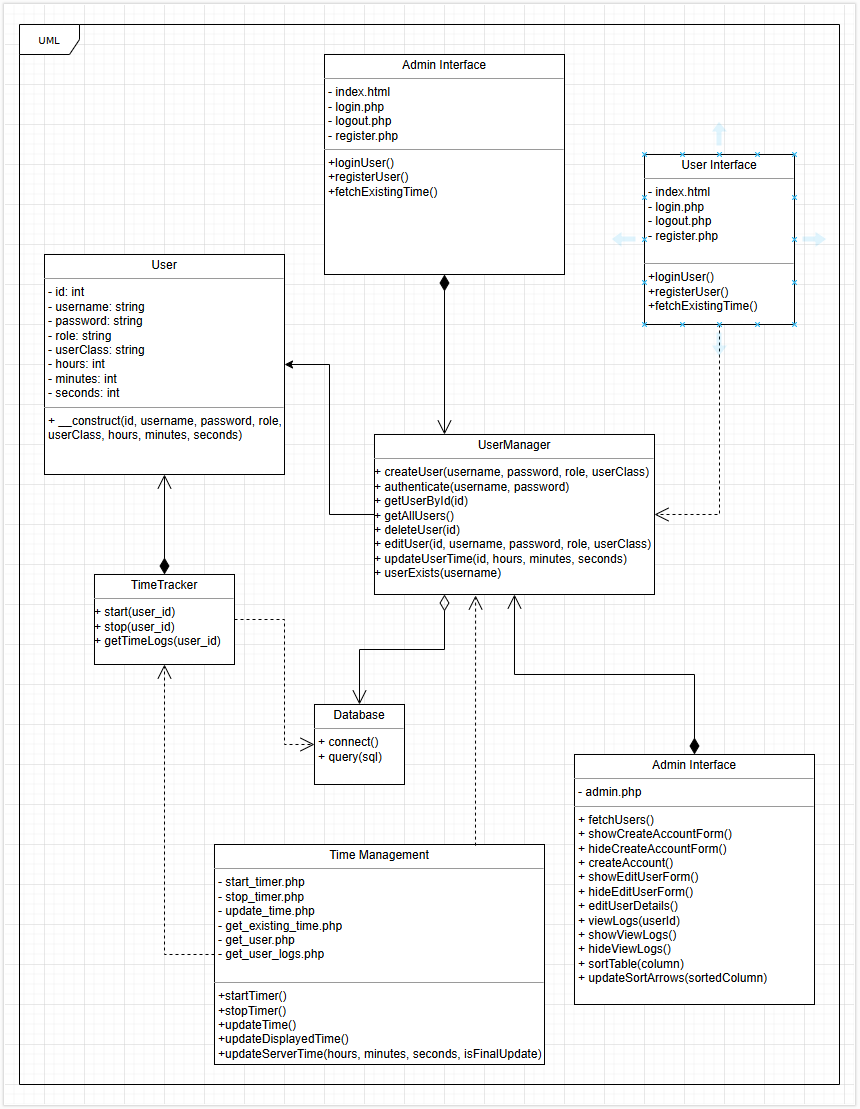
**The Gantt Chart can also be found** [here](https://github.com/users/AlexanderVidev/projects/2/views/1)**.**







**5. UML**

****

**6.Demonstrating Functionality**

**6.1 Proof of Concept**

**Styles.css:**

|  |
| --- |
| body {  **font-family**: Arial, **sans-serif**;  **margin**: **0**;  **padding**: **0**;  **background-color**: **#f4f4f4**;  **display**: flex;  **justify**-**content**: **center**;  align-items: **center**;  **height**: **100**vh;  flex-**direction**: column;  }  h1 {  **color**: **#333**;  }  div {  **margin**: **20px**;  **padding**: **20px**;  **background**: white;  **border**-radius: **5px**;  box-shadow: **0** **0** **10px** rgba(**0**,**0**,**0**,**0**.**1**);  }  input, select, button {  **display**: **block**;  **margin**: **10px** **0**;  **padding**: **10px**;  **width**: **100**%;  **max-width**: **300px**;  box-sizing: **border**-box;  }  table {  **width**: **90**%;  **max-width**: **800px**; /\* Set a maximum width \*/  **margin**: **20px** **auto**; /\* Center the table \*/  **border-collapse**: **collapse**;  }  th, td {  **padding**: **12px** **15px**; /\* Increase padding for more space \*/  **text-align**: **left**;  **border-bottom**: **1px** **solid** **#ddd**;  **white-space**: **nowrap**; /\* Ensure content stays on a single line \*/  }  th {  **background-color**: **#f4f4f4**;  }  **.sort-arrow** {  **cursor**: **pointer**;  **padding-left**: **5px**;  }  **.view-logs-header** {  **display**: flex;  **justify**-**content**: **center**;  align-items: **center**;  **width**: **90**%;  **max-width**: **800px**;  **margin**: **0** **auto** **10px**;  }  **.view-logs-header** h2 {  flex: **1**;  **text-align**: **center**;  }  **.view-logs-header** button {  flex: **1**;  **max-width**: **100px**;  **text-align**: **center**;  **margin**: **0** **auto**;  }  **.form-header** {  **display**: flex;  **justify**-**content**: space-between;  align-items: **center**;  }  button {  **display**: **block**;  **margin**: **10px** **0**;  **padding**: **10px**;  **width**: **100**%;  **max-width**: **300px**;  box-sizing: **border**-box;  }  /\* Specific style for logout button \*/  **.logout-button** {  **width**: **130px**; /\* Adjust the width as needed \*/  **margin**: **10px** **auto**; /\* Center the button \*/  } |

**Scripts.css:**

|  |
| --- |
| **let** startTime;  **let** timerInterval;  **let** updateInterval;  **let** existingHours = **0**;  **let** existingMinutes = **0**;  **let** existingSeconds = **0**;  **let** isTimerRunning = **false**;  **let** currentSessionSeconds = **0**;  **const** usernamePattern = /^[a-zA-Z0-9\_]{3,20}$/; // Alphanumeric and underscores, 3-20 characters  **const** passwordPattern = /^.{5,}$/; // Minimum 5 characters  **const** classPattern = /^[a-zA-Z0-9\_]{1,20}$/; // Alphanumeric and underscores, 1-20 characters  **function** displayMessage(message, isSuccess) {  **const** messageElement = document.getElementById('message');  messageElement.innerText = message;  messageElement.style.color = isSuccess ? 'green' : 'red';  messageElement.style.display = 'block';  setTimeout(() => {  messageElement.style.display = 'none';  }, **5000**); // Clear the message after 5 seconds  }  **function** validateInput(pattern, input, errorMessage) {  **if** (!pattern.test(input)) {  displayMessage(errorMessage, **false**);  **return** **false**;  }  **return** **true**;  }  **function** startTimer() {  **if** (isTimerRunning) {  displayMessage("Error: Timer is already running.", **false**);  **return**;  }  startTime = **new** Date();  timerInterval = setInterval(updateTime, **1000**);  updateInterval = setInterval(updateServerTime, **1000**); // Update server every 1 second  **const** xhr = **new** XMLHttpRequest();  xhr.open("POST", "php/start\_timer.php", **true**);  xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");  xhr.onreadystatechange = **function** () {  **if** (xhr.readyState === XMLHttpRequest.DONE && xhr.status === **200**) {  **if** (xhr.responseText.includes("success")) {  isTimerRunning = **true**;  } **else** {  displayMessage("Failed to start timer on server.", **false**);  }  }  };  xhr.send();  isTimerRunning = **true**;  }  **function** stopTimer() {  **if** (!isTimerRunning) {  displayMessage("Error: Timer is not running.", **false**);  **return**;  }  clearInterval(timerInterval);  clearInterval(updateInterval);  **const** endTime = **new** Date();  **const** elapsed = Math.floor((endTime - startTime) / **1000**);  currentSessionSeconds += elapsed;  **const** hours = Math.floor(currentSessionSeconds / **3600**);  **const** minutes = Math.floor((currentSessionSeconds % **3600**) / **60**);  **const** seconds = currentSessionSeconds % **60**;  // Update the display with the final logged time  updateDisplayedTime(existingHours + hours, existingMinutes + minutes, existingSeconds + seconds);  // Send final time update to the server  updateServerTime(hours, minutes, seconds, **true**);  **const** xhr = **new** XMLHttpRequest();  xhr.open("POST", "php/stop\_timer.php", **true**);  xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");  xhr.onreadystatechange = **function** () {  **if** (xhr.readyState === XMLHttpRequest.DONE && xhr.status === **200**) {  **const** response = JSON.parse(xhr.responseText);  **if** (response.status === 'success') {  isTimerRunning = **false**;  currentSessionSeconds = **0**; // Reset session time  fetchExistingTime(); // Fetch and display updated time logs  } **else** {  displayMessage("Failed to stop timer on server.", **false**);  }  }  };  xhr.send();  isTimerRunning = **false**;  }  **function** updateTime() {  **const** currentTime = **new** Date();  **const** elapsed = Math.floor((currentTime - startTime) / **1000**);  **const** totalSeconds = existingSeconds + currentSessionSeconds + elapsed;  **const** totalMinutes = existingMinutes + Math.floor(totalSeconds / **60**);  **const** displaySeconds = totalSeconds % **60**;  **const** displayHours = existingHours + Math.floor(totalMinutes / **60**);  **const** displayMinutes = totalMinutes % **60**;  document.getElementById('loggedTime').innerText = `Logged Time: ${displayHours}h ${displayMinutes}m ${displaySeconds}s`;  }  **function** updateDisplayedTime(hours, minutes, seconds) {  **const** totalSeconds = seconds;  **const** totalMinutes = minutes + Math.floor(totalSeconds / **60**);  **const** displaySeconds = totalSeconds % **60**;  **const** displayHours = hours + Math.floor(totalMinutes / **60**);  **const** displayMinutes = totalMinutes % **60**;  document.getElementById('loggedTime').innerText = `Logged Time: ${displayHours}h ${displayMinutes}m ${displaySeconds}s`;  }  **function** updateServerTime(hours, minutes, seconds, isFinalUpdate = **false**) {  **const** xhr = **new** XMLHttpRequest();  xhr.open("POST", "php/update\_time.php", **true**);  xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");  xhr.onreadystatechange = **function** () {  **if** (xhr.readyState === XMLHttpRequest.DONE && xhr.status === **200**) {  console.log(xhr.responseText);  **if** (isFinalUpdate) {  **const** response = JSON.parse(xhr.responseText);  existingHours = response.hours;  existingMinutes = response.minutes;  existingSeconds = response.seconds;  }  }  };  **const** elapsed = isFinalUpdate ? currentSessionSeconds : Math.floor((**new** Date() - startTime) / **1000**);  **const** totalSeconds = existingSeconds + elapsed;  **const** totalMinutes = existingMinutes + Math.floor(totalSeconds / **60**);  **const** newSeconds = totalSeconds % **60**;  **const** newHours = existingHours + Math.floor(totalMinutes / **60**);  **const** newMinutes = totalMinutes % **60**;  xhr.send(`hours=${newHours}&minutes=${newMinutes}&seconds=${newSeconds}&isFinalUpdate=${isFinalUpdate}`);  }  **function** showEditAccount() {  document.getElementById('editAccountForm').style.display = 'block';  }  **function** hideEditAccount() {  document.getElementById('editAccountForm').style.display = 'none';  document.getElementById('editAccount').reset(); // Clear the form fields  }  **function** editAccount() {  **const** newUsername = document.getElementById('newUsername').value;  **const** newPassword = document.getElementById('newPassword').value;  **const** newClass = document.getElementById('newClass').value;  **if** (newUsername && !validateInput(usernamePattern, newUsername, 'Invalid username. Use 3-20 alphanumeric characters or underscores.')) **return**;  **if** (newPassword && !validateInput(passwordPattern, newPassword, 'Password must be at least 5 characters long.')) **return**;  **if** (newClass && !validateInput(classPattern, newClass, 'Invalid class. Use 1-20 alphanumeric characters or underscores.')) **return**;  **const** xhr = **new** XMLHttpRequest();  xhr.open("POST", "php/edit\_account.php", **true**);  xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");  xhr.onreadystatechange = **function** () {  **if** (xhr.readyState === XMLHttpRequest.DONE && xhr.status === **200**) {  displayMessage(xhr.responseText.includes("successfully") ? "Account updated successfully." : "Failed to update account.", xhr.responseText.includes("successfully"));  **if** (xhr.responseText.includes("successfully")) {  hideEditAccount();  }  }  };  xhr.send(`username=${newUsername}&password=${newPassword}&**class**=${newClass}`);  }  **function** editUser(id) {  editUserId = id;  **const** xhr = **new** XMLHttpRequest();  xhr.open("POST", "php/get\_user.php", **true**);  xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");  xhr.onreadystatechange = **function** () {  **if** (xhr.readyState === XMLHttpRequest.DONE && xhr.status === **200**) {  **const** user = JSON.parse(xhr.responseText);  document.getElementById('editUsername').value = user.username;  document.getElementById('editRole').value = user.role;  document.getElementById('editClass').value = user.userClass;  document.getElementById('editHours').value = user.hours;  document.getElementById('editMinutes').value = user.minutes;  document.getElementById('editSeconds').value = user.seconds;  showEditUserForm();  }  };  xhr.send(`id=${id}`);  }  **function** showEditUserForm() {  document.getElementById('editUserForm').style.display = 'block';  }  **function** hideEditUserForm() {  document.getElementById('editUserForm').style.display = 'none';  document.getElementById('editUser').reset(); // Clear the form fields  }  **function** editUserDetails() {  **const** username = document.getElementById('editUsername').value;  **const** password = document.getElementById('editPassword').value;  **const** role = document.getElementById('editRole').value;  **const** userClass = document.getElementById('editClass').value;  **const** hours = document.getElementById('editHours').value;  **const** minutes = document.getElementById('editMinutes').value;  **const** seconds = document.getElementById('editSeconds').value;  **if** (username && !validateInput(usernamePattern, username, 'Invalid username. Use 3-20 alphanumeric characters or underscores.')) **return**;  **if** (password && !validateInput(passwordPattern, password, 'Password must be at least 5 characters long.')) **return**;  **if** (role === 'student' && userClass.trim() === '') {  displayMessage('Error: Students must enter a class.', **false**);  **return**;  }  **if** (userClass && !validateInput(classPattern, userClass, 'Invalid class. Use 1-20 alphanumeric characters or underscores.')) **return**;  **const** xhr = **new** XMLHttpRequest();  xhr.open("POST", "php/edit\_user.php", **true**);  xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");  xhr.onreadystatechange = **function** () {  **if** (xhr.readyState === XMLHttpRequest.DONE && xhr.status === **200**) {  displayMessage(xhr.responseText.includes("successfully") ? "User updated successfully." : "Failed to update user.", xhr.responseText.includes("successfully"));  **if** (xhr.responseText.includes("successfully")) {  hideEditUserForm();  fetchUsers(); // Refresh user list  }  }  };  xhr.send(`id=${editUserId}&username=${username}&password=${password}&role=${role}&userClass=${userClass}&hours=${hours}&minutes=${minutes}&seconds=${seconds}`);  }  **function** registerUser() {  **const** username = document.getElementById('regUsername').value;  **const** password = document.getElementById('regPassword').value;  **const** role = document.getElementById('regRole').value;  **const** userClass = document.getElementById('regClass').value;  **const** usernameRegex = /^[a-zA-Z0-9\_]{3,20}$/;  **const** passwordRegex = /^.{5,}$/;  **if** (!usernameRegex.test(username)) {  displayMessage('Invalid username. Use 3-20 alphanumeric characters or underscores.', **false**);  **return**;  }  **if** (!passwordRegex.test(password)) {  displayMessage('Password must be at least 5 characters long.', **false**);  **return**;  }  **if** (role === 'student' && userClass.trim() === '') {  displayMessage('Error: Students must enter a class.', **false**);  **return**;  }  **const** xhr = **new** XMLHttpRequest();  xhr.open("POST", "php/register.php", **true**);  xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");  xhr.onreadystatechange = **function** () {  **if** (xhr.readyState === XMLHttpRequest.DONE && xhr.status === **200**) {  **const** response = JSON.parse(xhr.responseText);  displayMessage(response.message, response.status === "success");  **if** (response.status === "success") {  document.getElementById('registerForm').reset(); // Clear the form fields  }  }  };  xhr.send(`username=${username}&password=${password}&role=${role}&userClass=${userClass}`);  }  **function** loginUser() {  **const** username = document.getElementById('loginUsername').value;  **const** password = document.getElementById('loginPassword').value;  **const** role = document.getElementById('loginRole').value;  **const** xhr = **new** XMLHttpRequest();  xhr.open("POST", "php/login.php", **true**);  xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");  xhr.onreadystatechange = **function** () {  **if** (xhr.readyState === XMLHttpRequest.DONE && xhr.status === **200**) {  **const** response = JSON.parse(xhr.responseText);  **if** (response.status === 'success') {  window.location.href = response.redirect;  } **else** {  displayMessage('Invalid login credentials', **false**);  }  }  };  xhr.send(`username=${username}&password=${password}&role=${role}`);  }  **function** fetchExistingTime() {  fetch('php/get\_existing\_time.php')  .then(response => response.json())  .then(data => {  existingHours = data.hours;  existingMinutes = data.minutes;  existingSeconds = data.seconds;  updateDisplayedTime(existingHours, existingMinutes, existingSeconds);  **if** (data.duration\_hours !== **undefined** && data.duration\_minutes !== **undefined** && data.duration\_seconds !== **undefined**) {  document.getElementById('loggedTime').innerText += ` (Duration: ${data.duration\_hours}h ${data.duration\_minutes}m ${data.duration\_seconds}s)`;  }  });  }  **function** fetchUsers() {  fetch('php/admin.php')  .then(response => response.json())  .then(users => {  **const** userTableBody = document.getElementById('userTableBody');  userTableBody.innerHTML = '';  users.forEach(user => {  **let** loggedTime = '';  **if** (user.role === 'student') {  loggedTime = `${user.hours}h ${user.minutes}m ${user.seconds}s`;  }  **const** row = document.createElement('tr');  row.innerHTML = `  <td>${user.username}</td>  <td>${user.role}</td>  <td>${user.userClass}</td>  <td>${loggedTime}</td>  <td><button onclick="viewLogs(${user.id})">Logs</button></td>  <td><button onclick="editUser(${user.id})">Edit</button></td>  <td><button onclick="deleteUser(${user.id})">Delete</button></td>  `;  userTableBody.appendChild(row);  });  });  }  **function** deleteUser(id) {  **if** (!confirm("Are you sure you want to delete this user?")) {  **return**;  }  **const** xhr = **new** XMLHttpRequest();  xhr.open("POST", "php/delete\_user.php", **true**);  xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");  xhr.onreadystatechange = **function** () {  **if** (xhr.readyState === XMLHttpRequest.DONE && xhr.status === **200**) {  **const** response = JSON.parse(xhr.responseText);  displayMessage(response.message, response.status === "success");  **if** (response.status === "success") {  fetchUsers(); // Refresh user list  }  }  };  xhr.send(`id=${id}`);  }  **function** showCreateAccountForm() {  document.getElementById('createAccountForm').style.display = 'block';  }  **function** hideCreateAccountForm() {  document.getElementById('createAccountForm').style.display = 'none';  document.getElementById('createAccount').reset(); // Clear the form fields  }  **function** createAccount() {  **const** username = document.getElementById('createUsername').value;  **const** password = document.getElementById('createPassword').value;  **const** role = document.getElementById('createRole').value;  **const** userClass = document.getElementById('createClass').value;  **if** (role === 'student' && userClass.trim() === '') {  displayMessage('Error: Students must enter a class.', **false**);  **return**;  }  **const** xhr = **new** XMLHttpRequest();  xhr.open("POST", "php/register.php", **true**);  xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");  xhr.onreadystatechange = **function** () {  **if** (xhr.readyState === XMLHttpRequest.DONE && xhr.status === **200**) {  **const** response = JSON.parse(xhr.responseText);  displayMessage(response.message, response.status === "success");  **if** (response.status === "success") {  hideCreateAccountForm();  fetchUsers(); // Refresh user list  }  }  };  xhr.send(`username=${username}&password=${password}&role=${role}&userClass=${userClass}`);  }  **function** viewLogs(userId) {  **const** xhr = **new** XMLHttpRequest();  xhr.open("POST", "php/get\_user\_logs.php", **true**);  xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");  xhr.onreadystatechange = **function** () {  **if** (xhr.readyState === XMLHttpRequest.DONE && xhr.status === **200**) {  **const** logs = JSON.parse(xhr.responseText);  **const** logsTableBody = document.getElementById('logsTableBody');  logsTableBody.innerHTML = '';  logs.forEach(log => {  **const** row = document.createElement('tr');  row.innerHTML = `  <td>${log.start\_time}</td>  <td>${log.end\_time}</td>  <td>${log.duration\_hours}h ${log.duration\_minutes}m ${log.duration\_seconds}s</td>  `;  logsTableBody.appendChild(row);  });  showViewLogs();  }  };  xhr.send(`user\_id=${userId}`);  }  **function** showViewLogs() {  document.getElementById('viewLogsForm').style.display = 'block';  }  **function** hideViewLogs() {  document.getElementById('viewLogsForm').style.display = 'none';  document.getElementById('logsTableBody').innerHTML = ''; // Clear the logs table  }  **let** sortOrder = {  username: 'asc',  role: 'asc',  **class**: 'asc',  loggedTime: 'asc'  };  **function** sortTable(column) {  **let** users = Array.from(document.querySelectorAll('#userTableBody tr')).map(tr => {  **return** {  element: tr,  username: tr.children[**0**].innerText,  role: tr.children[**1**].innerText,  **class**: tr.children[**2**].innerText,  loggedTime: tr.children[**3**].innerText  };  });  users.sort((a, b) => {  **if** (sortOrder[column] === 'asc') {  **return** a[column].localeCompare(b[column]);  } **else** {  **return** b[column].localeCompare(a[column]);  }  });  sortOrder[column] = sortOrder[column] === 'asc' ? 'desc' : 'asc';  **const** userTableBody = document.getElementById('userTableBody');  userTableBody.innerHTML = '';  users.forEach(user => {  userTableBody.appendChild(user.element);  });  updateSortArrows(column);  }  **function** updateSortArrows(sortedColumn) {  document.querySelectorAll('.sort-arrow').forEach(span => {  span.innerText = '▲';  });  **const** arrow = document.querySelector(`th[onclick="sortTable('${sortedColumn}')"] .sort-arrow`);  **if** (sortOrder[sortedColumn] === 'asc') {  arrow.innerText = '▲';  } **else** {  arrow.innerText = '▼';  }  }  document.addEventListener('DOMContentLoaded', **function**() {  fetchUsers();  fetchExistingTime(); // Fetch and display existing time on page load  }); |

**TimeTracker.php:**

|  |
| --- |
| <?php  **include** 'config.php';  **class** **TimeTracker** {  **private** $conn;  **public** **function** **\_\_construct**($conn) {  $this->conn = $conn;  }  **public** **function** **start**($user\_id) {  $stmt = $this->conn->prepare("INSERT INTO time\_logs (user\_id, start\_time) VALUES (?, NOW())");  $stmt->bind\_param("i", $user\_id);  **return** $stmt->execute();  }  **public** **function** **stop**($user\_id) {  // Calculate duration and update the end\_time and duration in the same query  $stmt = $this->conn->prepare("  UPDATE time\_logs  SET end\_time = NOW(),  duration\_hours = TIMESTAMPDIFF(HOUR, start\_time, NOW()),  duration\_minutes = MOD(TIMESTAMPDIFF(MINUTE, start\_time, NOW()), 60),  duration\_seconds = MOD(TIMESTAMPDIFF(SECOND, start\_time, NOW()), 60)  WHERE user\_id = ?  AND end\_time IS NULL  ORDER BY start\_time DESC  LIMIT 1");  $stmt->bind\_param("i", $user\_id);  **return** $stmt->execute();  }  **public** **function** **getTimeLogs**($user\_id) {  $stmt = $this->conn->prepare("SELECT \* FROM time\_logs WHERE user\_id = ?");  $stmt->bind\_param("i", $user\_id);  $stmt->execute();  **return** $stmt->get\_result()->fetch\_all(MYSQLI\_ASSOC);  }  }  ?> |

**User.php:**

|  |
| --- |
| <?php  **class** **User** {  **public** $id;  **public** $username;  **public** $password;  **public** $role;  **public** $userClass;  **public** $hours;  **public** $minutes;  **public** $seconds;  **public** **function** **\_\_construct**($id, $username, $password, $role, $userClass, $hours, $minutes, $seconds) {  $this->id = $id;  $this->username = $username;  $this->password = password\_hash($password, PASSWORD\_DEFAULT);  $this->role = $role;  $this->userClass = $userClass;  $this->hours = $hours;  $this->minutes = $minutes;  $this->seconds = $seconds;  }  }  ?> |

**UserManager.php:**

|  |
| --- |
| <?php  **include** 'config.php';  **include** 'User.php';  **class** **UserManager** {  **private** $conn;  **public** **function** **\_\_construct**($conn) {  $this->conn = $conn;  }  **public** **function** **createUser**($username, $password, $role, $userClass = "") {  $stmt = $this->conn->prepare("INSERT INTO users (username, password, role, userClass) VALUES (?, ?, ?, ?)");  $hashed\_password = password\_hash($password, PASSWORD\_DEFAULT);  $stmt->bind\_param("ssss", $username, $hashed\_password, $role, $userClass);  **return** $stmt->execute();  }  **public** **function** **authenticate**($username, $password) {  $stmt = $this->conn->prepare("SELECT \* FROM users WHERE username = ?");  $stmt->bind\_param("s", $username);  $stmt->execute();  $result = $stmt->get\_result();  $user = $result->fetch\_assoc();  **if** ($user && password\_verify($password, $user['password'])) {  **return** $user;  }  **return** **null**;  }  **public** **function** **getUserById**($id) {  $stmt = $this->conn->prepare("SELECT \* FROM users WHERE id = ?");  $stmt->bind\_param("i", $id);  $stmt->execute();  **return** $stmt->get\_result()->fetch\_assoc();  }  **public** **function** **getAllUsers**() {  $stmt = $this->conn->prepare("SELECT \* FROM users");  $stmt->execute();  **return** $stmt->get\_result()->fetch\_all(MYSQLI\_ASSOC);  }  **public** **function** **deleteUser**($id) {  // Delete user time logs  $stmt = $this->conn->prepare("DELETE FROM time\_logs WHERE user\_id = ?");  $stmt->bind\_param("i", $id);  $stmt->execute();    // Delete user record  $stmt = $this->conn->prepare("DELETE FROM users WHERE id = ?");  $stmt->bind\_param("i", $id);  **return** $stmt->execute();  }  **public** **function** **editUser**($id, $username, $password, $role, $userClass) {  **if** ($password === **null**) {  $stmt = $this->conn->prepare("UPDATE users SET username = ?, role = ?, userClass = ? WHERE id = ?");  $stmt->bind\_param("sssi", $username, $role, $userClass, $id);  } **else** {  $hashed\_password = password\_hash($password, PASSWORD\_DEFAULT);  $stmt = $this->conn->prepare("UPDATE users SET username = ?, password = ?, role = ?, userClass = ? WHERE id = ?");  $stmt->bind\_param("ssssi", $username, $hashed\_password, $role, $userClass, $id);  }  **return** $stmt->execute();  }  **public** **function** **updateUserTime**($id, $hours, $minutes, $seconds) {  $stmt = $this->conn->prepare("UPDATE users SET hours = ?, minutes = ?, seconds = ? WHERE id = ?");  $stmt->bind\_param("iiii", $hours, $minutes, $seconds, $id);  **return** $stmt->execute();  }  **public** **function** **userExists**($username) {  $stmt = $this->conn->prepare("SELECT \* FROM users WHERE username = ?");  $stmt->bind\_param("s", $username);  $stmt->execute();  $result = $stmt->get\_result();  **return** $result->num\_rows > **0**;  }  }  ?> |

**admin.php:**

|  |
| --- |
| <?php  session\_start();  **include** 'UserManager.php';  **if** (!isset($\_SESSION['user\_id']) || $\_SESSION['role'] !== 'instructor') {  header("Location: ../index.html");  **exit**();  }  $userManager = **new** UserManager($conn);  $users = $userManager->getAllUsers();  **echo** json\_encode($users);  ?> |

**config.php:**

|  |
| --- |
| <?php  $servername = "localhost";  $username = "root";  $password = "";  $dbname = "time\_manager";  $conn = **new** mysqli($servername, $username, $password, $dbname);  **if** ($conn->connect\_error) {  **die**("Connection failed: " . $conn->connect\_error);  }  ?> |

**delete\_user.php:**

|  |
| --- |
| <?php  **include** 'config.php';  **include** 'UserManager.php';  $userManager = **new** UserManager($conn);  $id = $\_POST['id'];  **if** ($userManager->deleteUser($id)) {  **echo** json\_encode(["status" => "success", "message" => "User deleted successfully"]);  } **else** {  **echo** json\_encode(["status" => "error", "message" => "Failed to delete user"]);  }  ?> |

**edit\_account.php:**

|  |
| --- |
| <?php  session\_start();  **include** 'config.php';  **include** 'UserManager.php';  **if** (!isset($\_SESSION['user\_id'])) {  **echo** "User not logged in.";  **exit**();  }  $user\_id = $\_SESSION['user\_id'];  $newUsername = $\_POST['username'];  $newPassword = $\_POST['password'];  $newClass = $\_POST['class'];  $userManager = **new** UserManager($conn);  $user = $userManager->getUserById($user\_id);  **if** ($user) {  // Update user details  $newUsername = **empty**($newUsername) ? $user['username'] : $newUsername;  $newPassword = **empty**($newPassword) ? **null** : $newPassword;  $newClass = **empty**($newClass) ? $user['userClass'] : $newClass;  **if** ($userManager->editUser($user\_id, $newUsername, $newPassword, $user['role'], $newClass)) {  **echo** "Account updated successfully.";  } **else** {  **echo** "Failed to update account.";  }  } **else** {  **echo** "User not found.";  }  ?> |

**edit\_user.php:**

|  |
| --- |
| <?php  **include** 'config.php';  **include** 'UserManager.php';  $userManager = **new** UserManager($conn);  $id = $\_POST['id'];  $username = $\_POST['username'];  $password = **empty**($\_POST['password']) ? **null** : $\_POST['password'];  $role = $\_POST['role'];  $userClass = $\_POST['userClass'];  $hours = intval($\_POST['hours']);  $minutes = intval($\_POST['minutes']);  $seconds = intval($\_POST['seconds']);  **if** ($role === 'student' && **empty**(trim($userClass))) {  **echo** "Failed to update user. Students must have a class.";  **exit**();  }  **if** ($userManager->editUser($id, $username, $password, $role, $userClass)) {  $userManager->updateUserTime($id, $hours, $minutes, $seconds);  **echo** "User updated successfully.";  } **else** {  **echo** "Failed to update user.";  }  ?> |

**get\_existing\_time.php:**

|  |
| --- |
| <?php  session\_start();  **include** 'config.php';  **include** 'UserManager.php';  **if** (!isset($\_SESSION['user\_id'])) {  **echo** json\_encode(['hours' => **0**, 'minutes' => **0**, 'seconds' => **0**]);  **exit**();  }  $user\_id = $\_SESSION['user\_id'];  $userManager = **new** UserManager($conn);  $user = $userManager->getUserById($user\_id);  **if** ($user) {  **echo** json\_encode([  'hours' => $user['hours'],  'minutes' => $user['minutes'],  'seconds' => $user['seconds']  ]);  } **else** {  **echo** json\_encode(['hours' => **0**, 'minutes' => **0**, 'seconds' => **0**]);  }  ?> |

**get\_user.php:**

|  |
| --- |
| <?php  **include** 'config.php';  **include** 'UserManager.php';  $userManager = **new** UserManager($conn);  $user\_id = $\_POST['id'];  $user = $userManager->getUserById($user\_id);  **echo** json\_encode($user);  ?> |

**get\_user\_logs.php:**

|  |
| --- |
| <?php  **include** 'config.php';  **include** 'TimeTracker.php';  $user\_id = $\_POST['user\_id'];  $timeTracker = **new** TimeTracker($conn);  $logs = $timeTracker->getTimeLogs($user\_id);  **echo** json\_encode($logs);  ?> |

**login.php:**

|  |
| --- |
| <?php  session\_start();  **include** 'UserManager.php';  $userManager = **new** UserManager($conn);  $username = $\_POST['username'];  $password = $\_POST['password'];  $role = $\_POST['role'];  $user = $userManager->authenticate($username, $password);  **if** ($user && $user['role'] === $role) {  $\_SESSION['user\_id'] = $user['id'];  $\_SESSION['role'] = $user['role'];  **if** ($role === 'instructor') {  **echo** json\_encode(['status' => 'success', 'redirect' => 'admin.html']);  } **else** {  **echo** json\_encode(['status' => 'success', 'redirect' => 'user.html']);  }  } **else** {  **echo** json\_encode(['status' => 'error', 'message' => 'Invalid login credentials']);  }  ?> |

**logout.php:**

|  |
| --- |
| <?php  session\_start();  session\_destroy();  header("Location: ../index.html");  ?> |

**register.php:**

|  |
| --- |
| <?php  **include** 'config.php';  **include** 'UserManager.php';  $userManager = **new** UserManager($conn);  $username = $\_POST['username'];  $password = $\_POST['password'];  $role = $\_POST['role'];  $userClass = $\_POST['userClass'];  **if** (!preg\_match('/^[a-zA-Z0-9\_]{3,20}$/', $username)) {  **echo** json\_encode(["status" => "error", "message" => "Invalid username. Use 3-20 alphanumeric characters or underscores."]);  **exit**();  }  **if** (strlen($password) < **5**) {  **echo** json\_encode(["status" => "error", "message" => "Password must be at least 5 characters long."]);  **exit**();  }  // Check if the username already exists  **if** ($userManager->userExists($username)) {  **echo** json\_encode(["status" => "error", "message" => "Account already exists"]);  } **else** {  **if** ($userManager->createUser($username, $password, $role, $userClass)) {  **echo** json\_encode(["status" => "success", "message" => "User registered successfully"]);  } **else** {  **echo** json\_encode(["status" => "error", "message" => "Failed to register user"]);  }  }  ?> |

**start\_timer.php:**

|  |
| --- |
| <?php  session\_start();  **include** 'config.php';  **include** 'TimeTracker.php';  **if** (!isset($\_SESSION['user\_id'])) {  **echo** "User not logged in.";  **exit**();  }  $user\_id = $\_SESSION['user\_id'];  $timeTracker = **new** TimeTracker($conn);  **if** ($timeTracker->start($user\_id)) {  **echo** "Timer started successfully.";  } **else** {  **echo** "Failed to start timer.";  }  ?> |

**stop\_timer.php:**

|  |
| --- |
| <?php  session\_start();  **include** 'config.php';  **include** 'TimeTracker.php';  **if** (!isset($\_SESSION['user\_id'])) {  **echo** json\_encode(['status' => 'error', 'message' => 'User not logged in.']);  **exit**();  }  $user\_id = $\_SESSION['user\_id'];  $timeTracker = **new** TimeTracker($conn);  **if** ($timeTracker->stop($user\_id)) {  **echo** json\_encode(['status' => 'success', 'message' => 'Timer stopped successfully.']);  } **else** {  **echo** json\_encode(['status' => 'error', 'message' => 'Failed to stop timer.']);  }  ?> |

**update\_time.php:**

|  |
| --- |
| <?php  session\_start();  **include** 'config.php';  **include** 'UserManager.php';  **include** 'TimeTracker.php';  **if** (!isset($\_SESSION['user\_id'])) {  **echo** json\_encode(['status' => 'error', 'message' => 'User not logged in.']);  **exit**();  }  $user\_id = $\_SESSION['user\_id'];  $hours = intval($\_POST['hours']);  $minutes = intval($\_POST['minutes']);  $seconds = intval($\_POST['seconds']);  $isFinalUpdate = isset($\_POST['isFinalUpdate']) ? $\_POST['isFinalUpdate'] === 'true' : **false**;  $userManager = **new** UserManager($conn);  $timeTracker = **new** TimeTracker($conn);  $user = $userManager->getUserById($user\_id);  **if** ($user) {  // Update user's logged time in the database  $new\_hours = $hours;  $new\_minutes = $minutes;  $new\_seconds = $seconds;  // Normalize time  $new\_minutes += intdiv($new\_seconds, **60**);  $new\_seconds %= **60**;  $new\_hours += intdiv($new\_minutes, **60**);  $new\_minutes %= **60**;  **if** ($userManager->updateUserTime($user\_id, $new\_hours, $new\_minutes, $new\_seconds)) {  **if** ($isFinalUpdate) {  $timeTracker->stop($user\_id);  }  **echo** json\_encode([  'status' => 'success',  'hours' => $new\_hours,  'minutes' => $new\_minutes,  'seconds' => $new\_seconds,  'message' => 'Time updated successfully.'  ]);  } **else** {  **echo** json\_encode(['status' => 'error', 'message' => 'Failed to update time.']);  }  } **else** {  **echo** json\_encode(['status' => 'error', 'message' => 'User not found.']);  }  ?> |

**admin.html:**

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Admin Interface</title>  <link rel="stylesheet" href="css/styles.css">  </head>  <body>  <h1>Admin Interface</h1>  <h2>Manage Users</h2>  <div id="message" style="display: none;"></div>  <button onclick="showCreateAccountForm()">Create Account</button>  <table>  <thead>  <tr>  <th onclick="sortTable('username')">Username <span class="sort-arrow">▲</span></th>  <th onclick="sortTable('role')">Role <span class="sort-arrow">▲</span></th>  <th onclick="sortTable('class')">Class <span class="sort-arrow">▲</span></th>  <th onclick="sortTable('loggedTime')">Logged Time <span class="sort-arrow">▲</span></th>  <th>Logs</th>  <th>Edit</th>  <th>Delete</th>  </tr>  </thead>  <tbody id="userTableBody">  <!-- User rows will be populated by JavaScript -->  </tbody>  </table>  <button class="logout-button" onclick="location.href='php/logout.php'">Logout</button>  <div id="createAccountForm" style="display: none;">  <h2>Create Account</h2>  <form id="createAccount">  <input type="text" id="createUsername" placeholder="Username" required>  <input type="password" id="createPassword" placeholder="Password" required>  <select id="createRole">  <option value="student">Student</option>  <option value="instructor">Instructor</option>  </select>  <input type="text" id="createClass" placeholder="Class (if student)">  <button type="button" onclick="createAccount()">Create Account</button>  <button type="button" onclick="hideCreateAccountForm()">Close</button>  </form>  </div>  <div id="editUserForm" style="display: none;">  <h2>Edit User</h2>  <form id="editUser">  <input type="text" id="editUsername" placeholder="New Username">  <input type="password" id="editPassword" placeholder="New Password">  <select id="editRole">  <option value="student">Student</option>  <option value="instructor">Instructor</option>  </select>  <input type="text" id="editClass" placeholder="New Class">  <input type="number" id="editHours" placeholder="Hours">  <input type="number" id="editMinutes" placeholder="Minutes">  <input type="number" id="editSeconds" placeholder="Seconds">  <button type="button" onclick="editUserDetails()">Save Changes</button>  <button type="button" onclick="hideEditUserForm()">Close</button>  </form>  </div>  <div id="viewLogsForm" style="display: none;">  <div class="view-logs-header">  <h2>View User Logs</h2>  <button type="button" onclick="hideViewLogs()">Close</button>  </div>  <table>  <thead>  <tr>  <th>Start Time</th>  <th>End Time</th>  <th>Duration</th>  </tr>  </thead>  <tbody id="logsTableBody">  <!-- Logs will be populated by JavaScript -->  </tbody>  </table>  </div>  <script src="js/scripts.js"></script>  </body>  </html> |

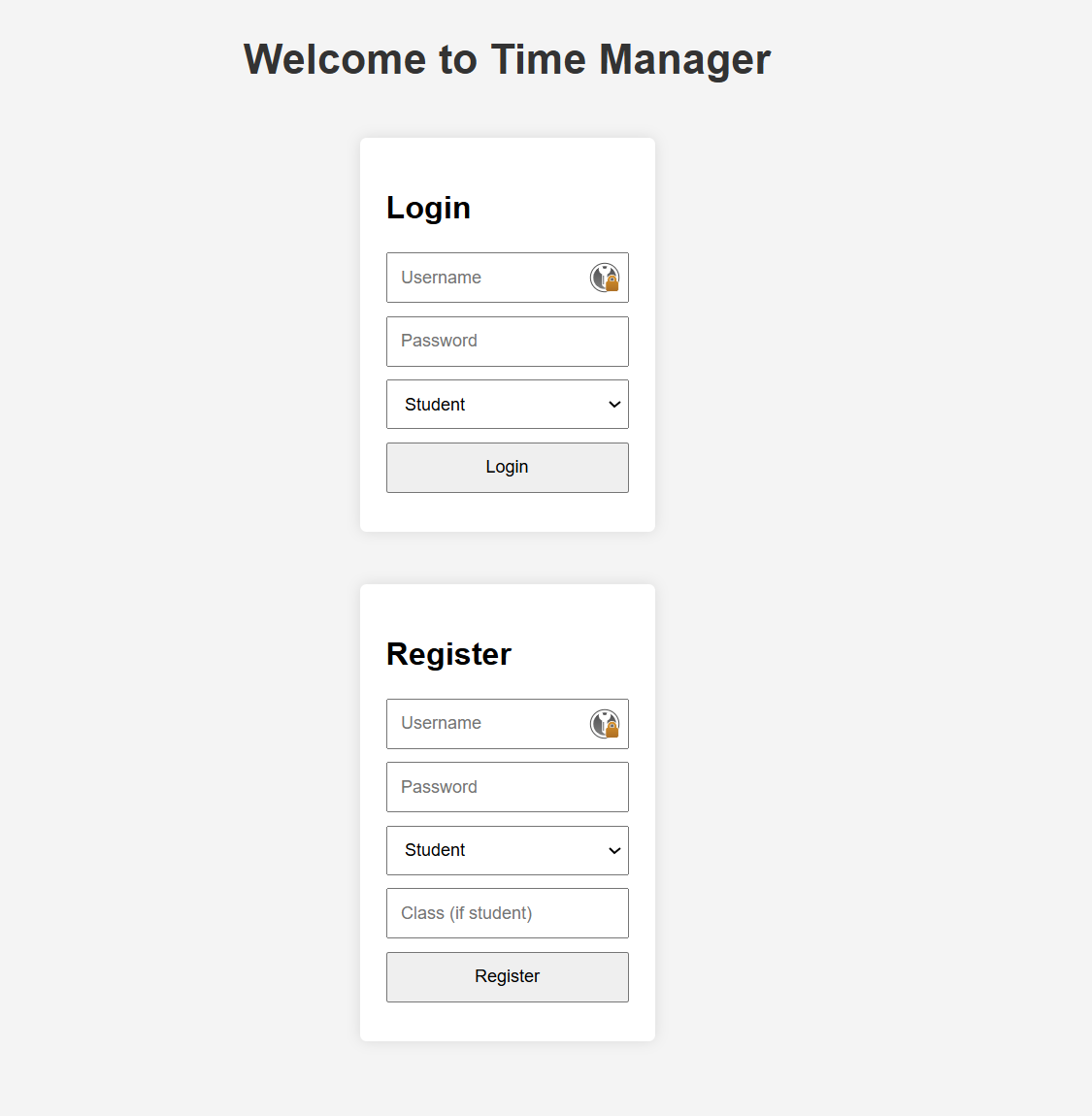
**index.html:**

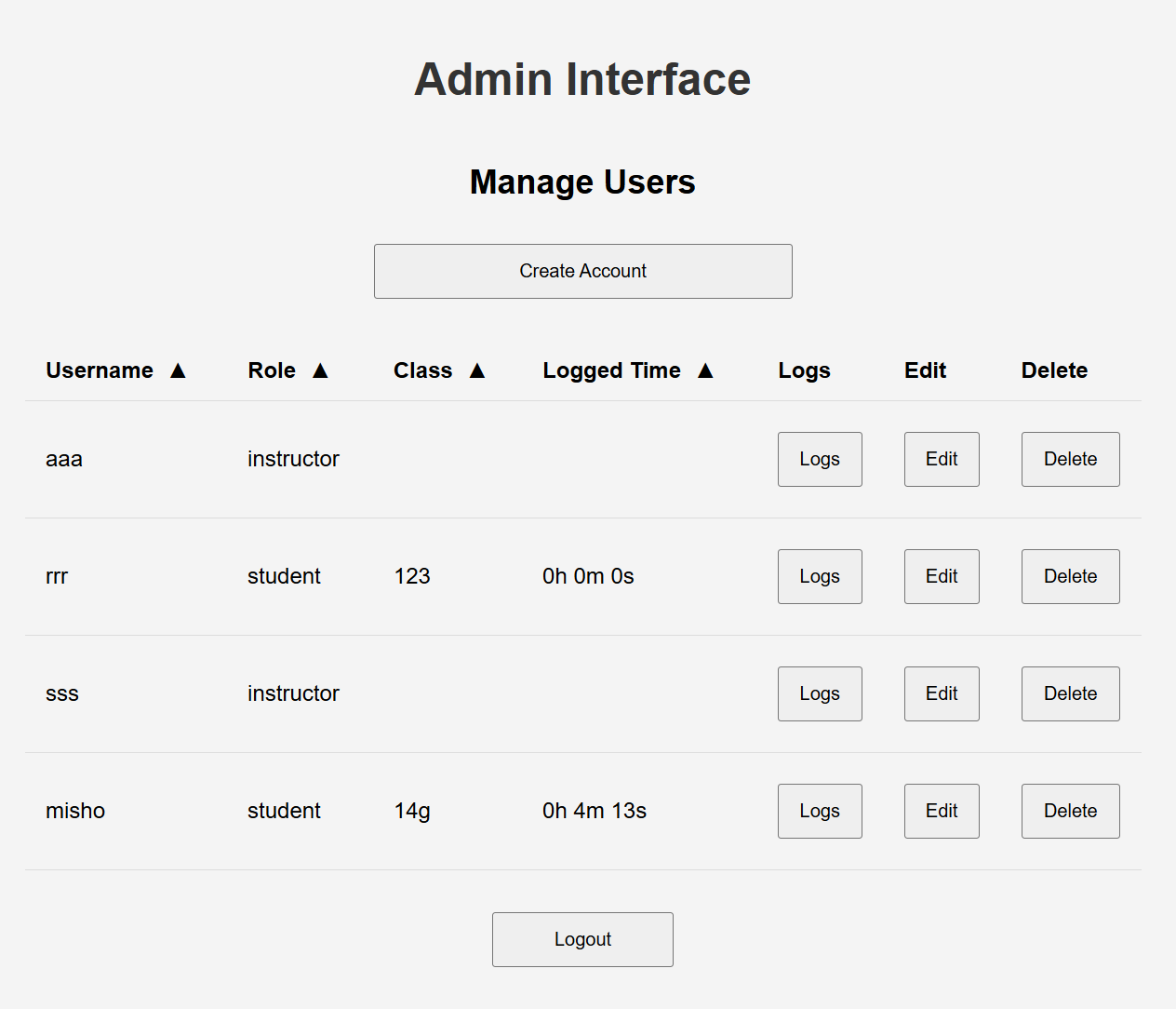
|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Time Manager</title>  <link rel="stylesheet" href="css/styles.css">  </head>  <body>  <h1>Welcome to Time Manager</h1>  <div id="message" style="display: none;"></div>  <div>  <h2>Login</h2>  <form id="loginForm">  <input type="text" id="loginUsername" name="username" placeholder="Username" required>  <input type="password" id="loginPassword" name="password" placeholder="Password" required>  <select id="loginRole" name="role">  <option value="student">Student</option>  <option value="instructor">Instructor</option>  </select>  <button type="button" onclick="loginUser()">Login</button>  </form>  </div>  <div>  <h2>Register</h2>  <form id="registerForm">  <input type="text" id="regUsername" name="username" placeholder="Username" required>  <input type="password" id="regPassword" name="password" placeholder="Password" required>  <select id="regRole" name="role">  <option value="student">Student</option>  <option value="instructor">Instructor</option>  </select>  <input type="text" id="regClass" name="userClass" placeholder="Class (if student)">  <button type="button" onclick="registerUser()">Register</button>  </form>  </div>  <script src="js/scripts.js"></script>  </body>  </html> |

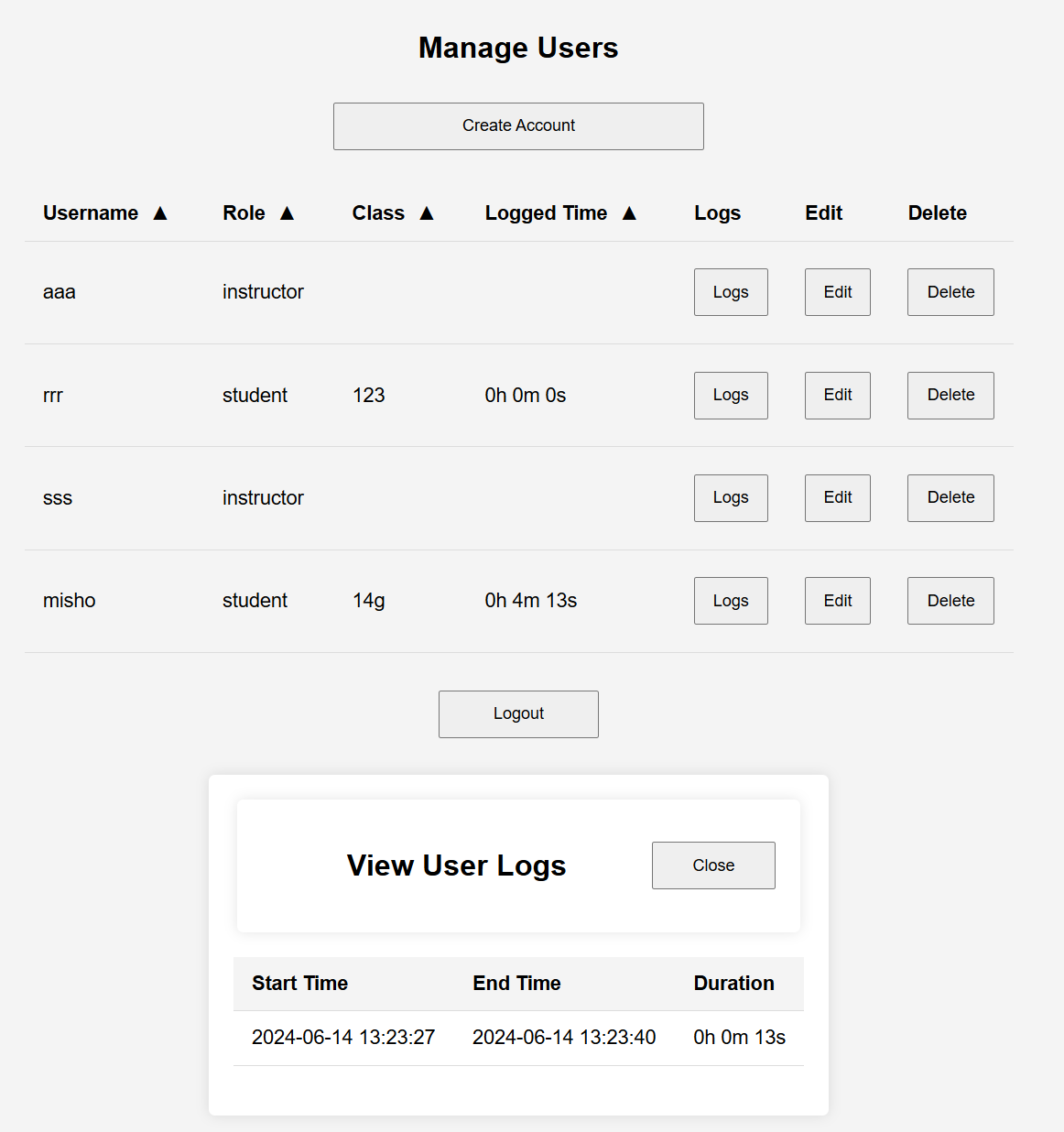
**user.html:**

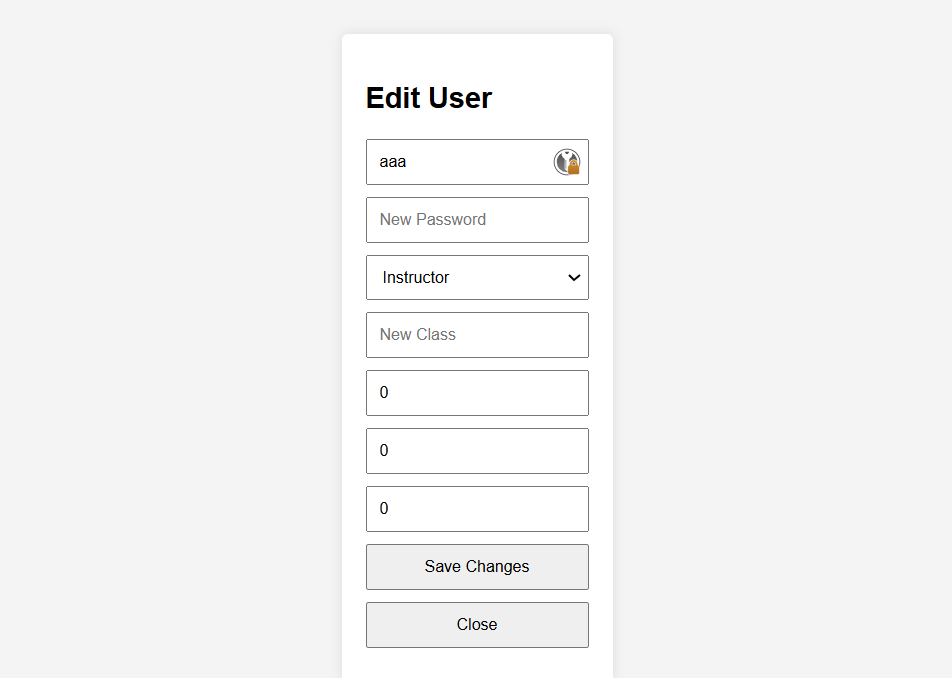
|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>User Interface</title>  <link rel="stylesheet" href="css/styles.css">  </head>  <body>  <h1>User Interface</h1>  <div id="message" style="display: none;"></div>  <div>  <h2>Time Tracking</h2>  <button onclick="startTimer()">Start Timer</button>  <button onclick="stopTimer()">Stop Timer</button>  <button onclick="showEditAccount()">Edit Account</button>  <div id="loggedTime">Logged Time: 0h 0m 0s</div>  </div>  <button class="logout-button" onclick="location.href='php/logout.php'">Logout</button>    <div id="editAccountForm" style="display: none;">  <h2>Edit Account</h2>  <form id="editAccount">  <input type="text" id="newUsername" placeholder="New Username">  <input type="password" id="newPassword" placeholder="New Password">  <input type="text" id="newClass" placeholder="New Class">  <button type="button" onclick="editAccount()">Save Changes</button>  </form>  <button type="button" onclick="hideEditAccount()">Close</button>  </div>  <script src="js/scripts.js"></script>  </body>  </html> |

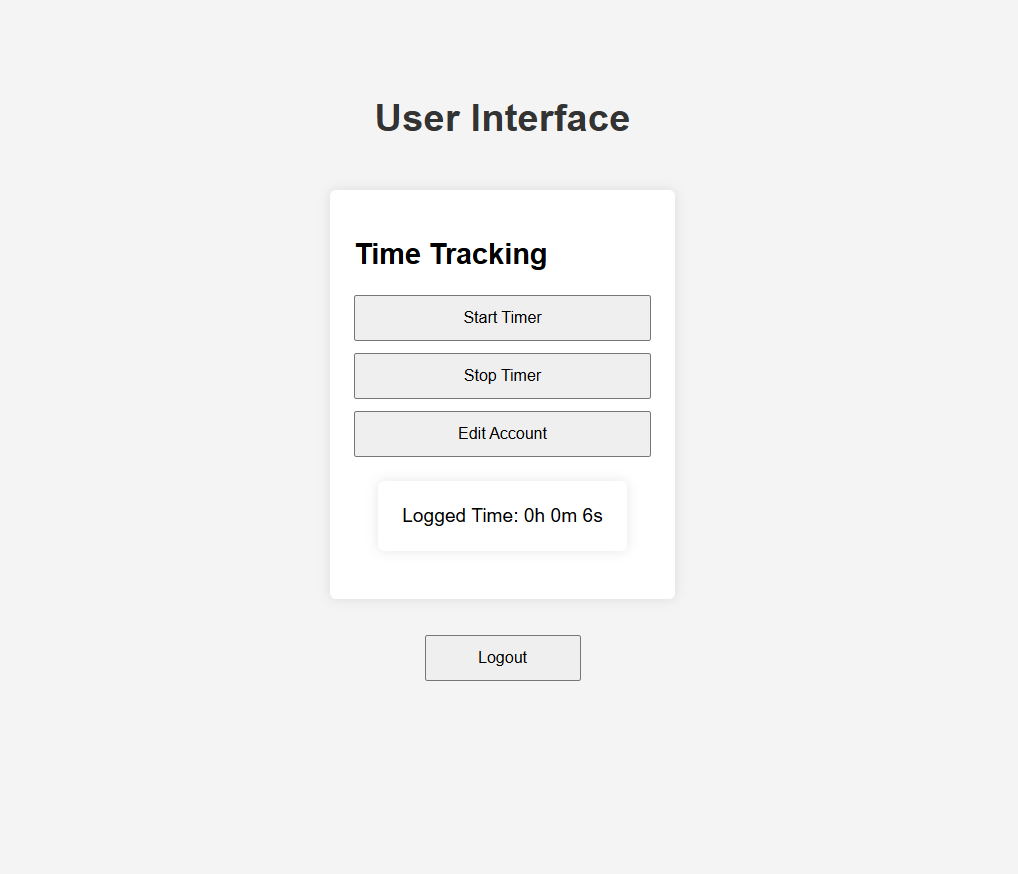
**6.2 Testing Results**

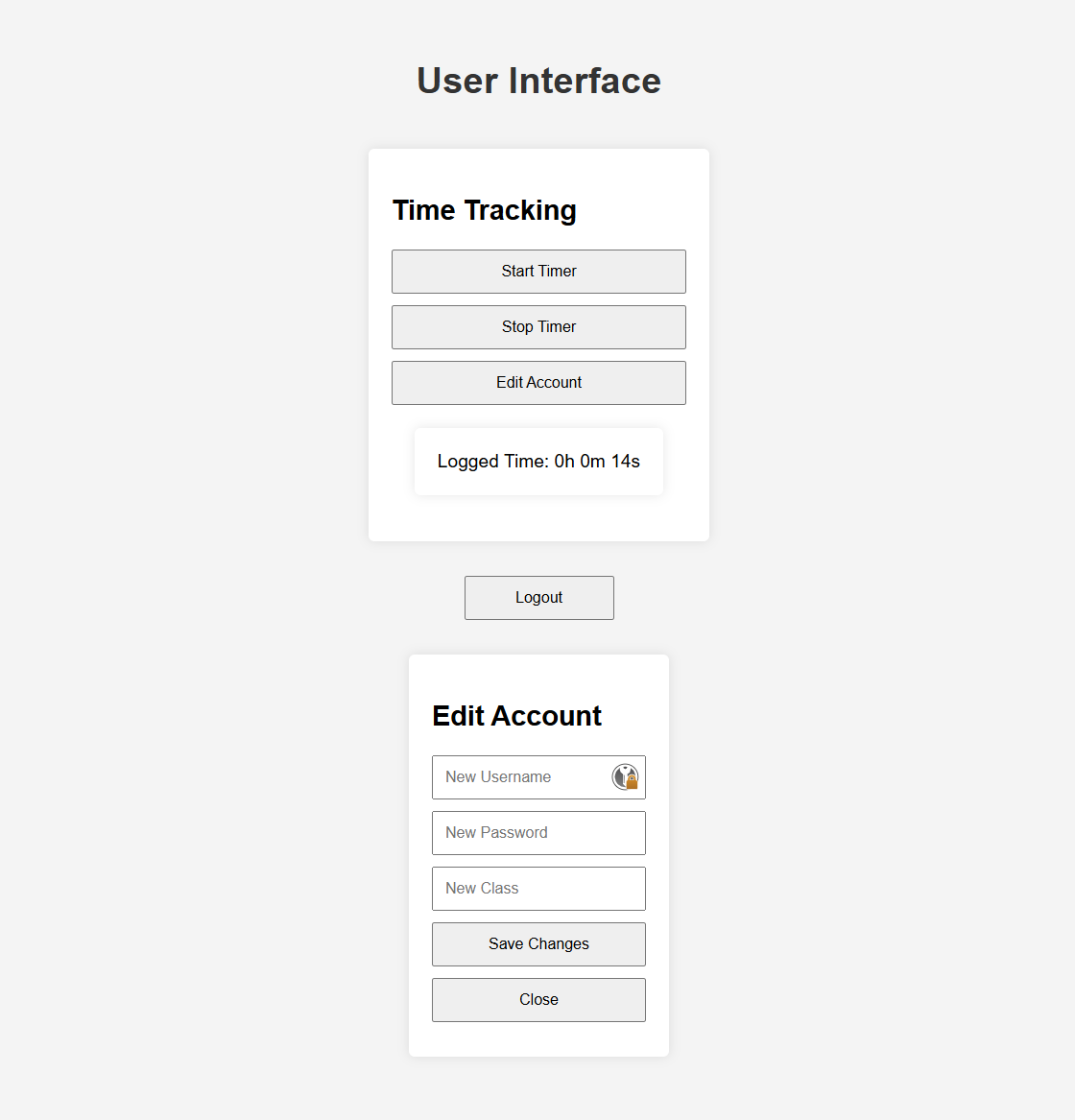
****

****

****

****

****

****