INTRODUCTION

The "Gym Management System" project aims to provide an efficient and comprehensive solution for managing gym operations and memberships. The system is designed to be accessible only by administrators, who can utilize a range of features through a user-friendly navigation bar. This centralized platform allows administrators to manage gyms, members, payments, and trainers effectively, reducing manual effort and improving accuracy.

The primary goal of the project is to streamline the management processes within gyms, ensuring smooth operations and better service for members. By integrating various functionalities into one system, the project aims to reduce disputes and administrative burdens, enabling gym administrators to focus more on enhancing the overall member experience.

1.1 OBJECTIVE

The main objective of the "Gym Management System" project is to streamline the management of gym operations, ensuring efficient handling of memberships, payments, and trainer assignments. By centralizing these tasks on an online platform, the project aims to reduce administrative workload, minimize errors, and provide a seamless experience for both gym administrators and members. The goal is to leverage technology to enhance operational efficiency, improve member satisfaction, and ensure accurate management of gym resources and financial records.

1.2 GOALS

- **Streamline Gym Management:** Simplify the process of adding, viewing, updating, and deleting gym information, ensuring accurate and efficient management of multiple gym locations.
- **Enhance Member Management:** Facilitate the registration and management of gym members, allowing administrators to easily add new members, update their details, and view member information**.**

- **Optimize Payment Management**: Provide tools for recording and viewing payments, ensuring accurate financial tracking and reducing the risk of errors and disputes related to member payments.
- **Efficient Trainer Management**: Enable administrators to add and manage trainer details, ensuring that trainer information is accurately recorded and easily accessible.
- **Centralize Operations:** Offer a user-friendly navigation bar with options for managing gyms, members, payments, and trainers, centralizing all key operations in one platform.
- **Reduce Administrative Workload:** Minimize manual effort required for administrative tasks, allowing gym administrators to focus on enhancing member experience and overall gym management.

2.METHODOLOGY

Technology Stack:

Using the following technologies:

2.1 Backend Development:

- PHP:
  - Server-side scripting language used for implementing backend logic, processing form submissions, and interacting with the MySQL database.
  - Responsible for handling user authentication, session management, and business logic related to managing leave requests.

- MySQL:
  - Relational database management system (RDBMS) used for storing and managing data related to students, leave requests, administrators, and any other relevant information.

- Responsible for creating database tables, defining relationships between tables, and executing SQL queries for data retrieval, insertion, update, and deletion operations.

Configuration of PHP and MySQL

- **Install PHP and MySQL:** Install PHP and MySQL on your computer, ensuring PHP version 7.x or higher for compatibility.

- **Create Database:** Use MySQL tools to create a database named leave_letter_collector for your project.

- **Configure PHP:** Enable MySQLi extension in php.ini (extension=mysqli) to connect PHP with MySQL.

- **Connect PHP to MySQL:** Use PHP's MySQLi functions (mysqli_connect) to establish a connection to MySQL in your scripts.

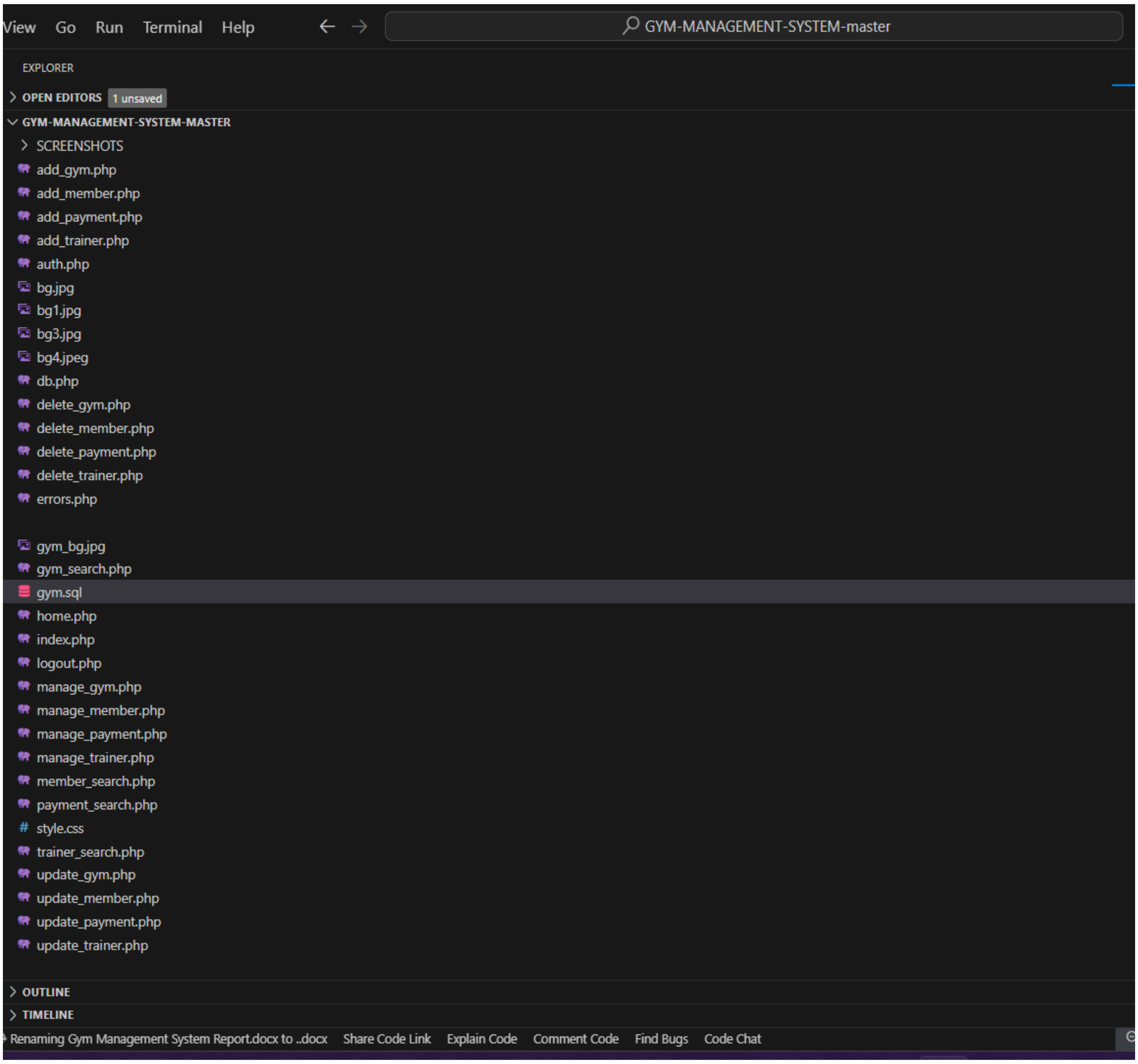
- **Test Connection:** Create a simple PHP script to verify the MySQL connection and run basic queries (SELECT, INSERT, etc.).

- **Develop Your Application:** Start developing your PHP application, handling database operations using MySQL for data storage.

- **Local Development Setup:** Set up a local development environment with tools like XAMPP, WAMP, or MAMP, which include PHP and MySQL.

- **Prepare for Deployment:** Configure server settings and ensure security measures before deploying your PHP and MySQL-based application.

File Structure of project



GYM-MANAGEMENT-SYSTEM-MASTER
- SCREENSHOTS
- add_gym.php
- add_member.php
- add_payment.php
- add_trainer.php
- auth.php
- bg.jpg
- bg1.jpg
- bg3.jpg
- bg4.jpeg
- db.php
- delete_gym.php
- delete_member.php
- delete_payment.php
- delete_trainer.php
- errors.php
- gym_bg.jpg
- gym_search.php
- gym.sql
- home.php
- index.php
- logout.php
- manage_gym.php
- manage_member.php
- manage_payment.php
- manage_trainer.php
- member_search.php
- payment_search.php
- style.css
- trainer_search.php
- update_gym.php
- update_member.php
- update_payment.php
- update_trainer.php

2.2 Frontend Development:

HTML (HyperText Markup Language):

- **Description**:

  HTML is the standard markup language used to create the structure and content of web pages.

- **Key Features:**
  - Semantic Markup: Provides tags like <header>, <footer>, <section>, and <article> for better document structure and accessibility.
  - Forms: Allows creation of input fields (<input>, <textarea>, <select>) and form controls for user interaction.
  - Media Embedding: Supports embedding images (<img>), videos (<video>), and audio (<audio>) for multimedia content.
  - Accessibility: Provides elements and attributes (alt, role, aria-attributes) to improve accessibility for users with disabilities.

CSS (Cascading Style Sheets):

- **Description**:

  CSS is used for styling HTML elements, controlling the visual presentation of web pages.

- **Key Features:**
  - Selectors: Allows targeting of specific HTML elements for styling (class, id, element selectors).
  - Box Model: Defines properties (width, height, margin, padding, border) for controlling layout and spacing.
  - Flexbox and Grid Layout: Modern layout systems for creating flexible and responsive designs.

    - Transitions and Animations: Enables smooth transitions (transition) and animations (@keyframes) for interactive elements.

JavaScript (JS):

- **Description**:

  JavaScript is a scripting language used for creating dynamic and interactive elements within web applications.

- **Key Features:**
    - Event Handling: Allows responding to user actions (click, hover, submit) to trigger functionality.
    - DOM Manipulation: Enables adding, modifying, or deleting HTML elements and attributes dynamically.
    - Asynchronous Programming: Supports AJAX (Asynchronous JavaScript and XML) for making requests to the server without refreshing the page.
    - ES6+ Features: Modern JavaScript syntax and features like arrow functions, let and const declarations, and async/await for handling asynchronous operations.

Advantages in Web Development:

- HTML: Forms the foundation of web pages, providing structure and accessibility features essential for SEO and usability.
- CSS: Enhances user interface design and user experience, ensuring consistency and responsiveness across different devices.
- JavaScript: Powers interactive and dynamic web applications, enabling real-time updates, user input validation, and rich media handling.

2.3 SQL Database:

A structured query language database, such as MySQL within WAMP, plays a role in storing and managing critical data. This includes comprehensive storage of student information, such as their profiles, login credentials, and leave submission history. Additionally, the database handles administrative functions, facilitating the approval or rejection of leave requests and providing robust search capabilities for efficient management. This setup ensures that all operational aspects of leave management are centralized and securely managed, supporting seamless interaction between students and administrators.

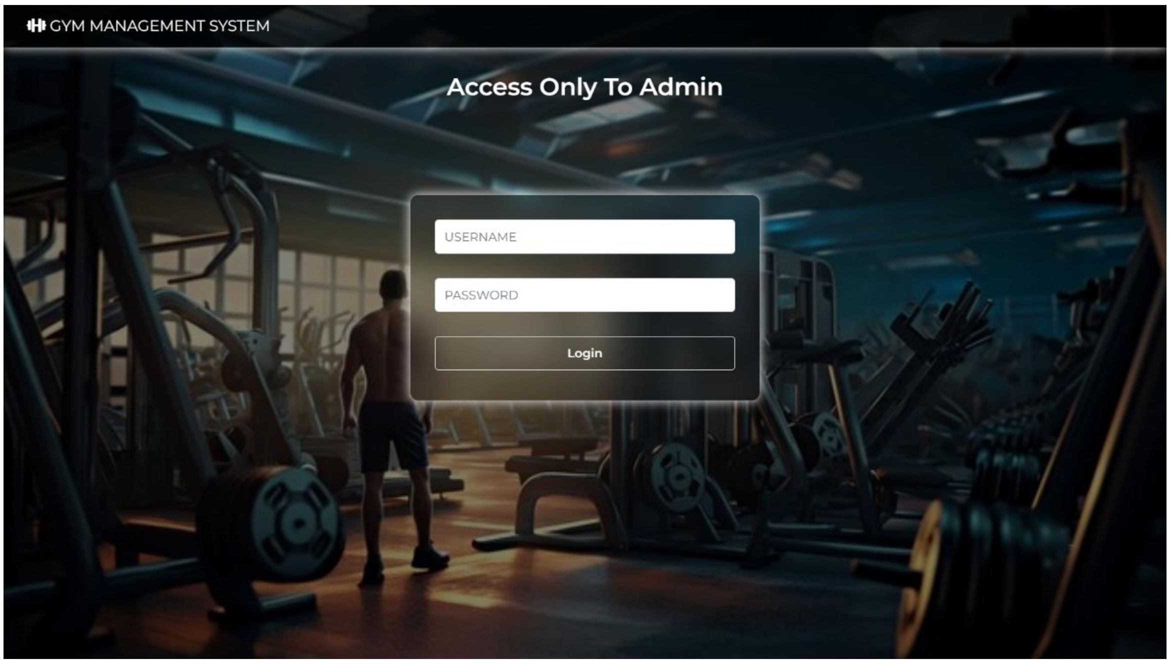
2.4 Development and Deployment:

WAMP Server:

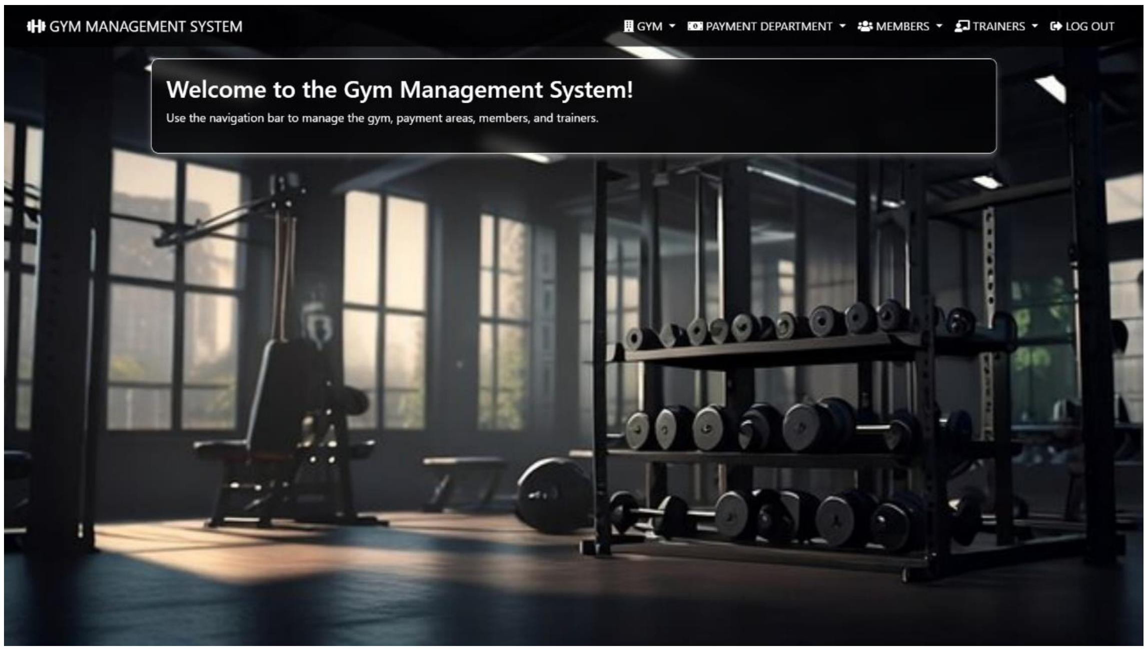
A free and open-source software package, was instrumental in developing your 'GYM MANAGEMENT SYSTEM' project. It includes MySQL database management, PHP, and Perl support, providing a comprehensive local development environment. WAMP served as the foundation for building and testing your application's backend functionalities, such as managing student leave requests and administrative tasks. This setup ensured efficient data handling and robust server-side scripting capabilities before potential deployment to a live server environment.

HERE IS MY WEBSITE INTERFACE:

HOME PAGE:



AFTER ADMIN LOGIN:

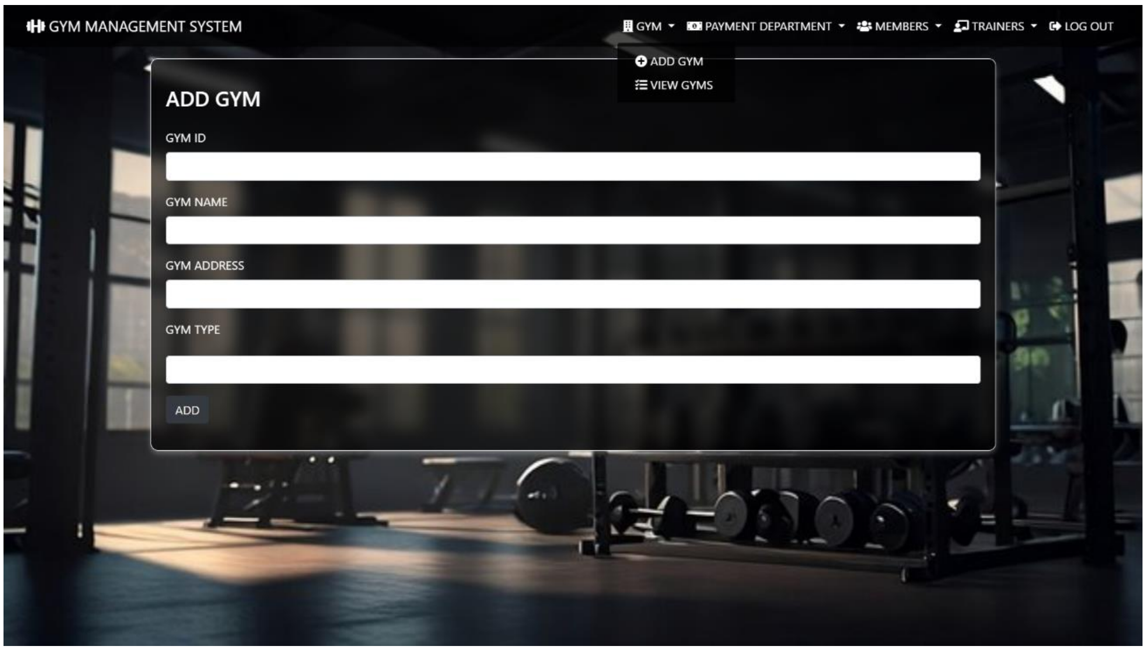


Welcome to the Gym Management System!

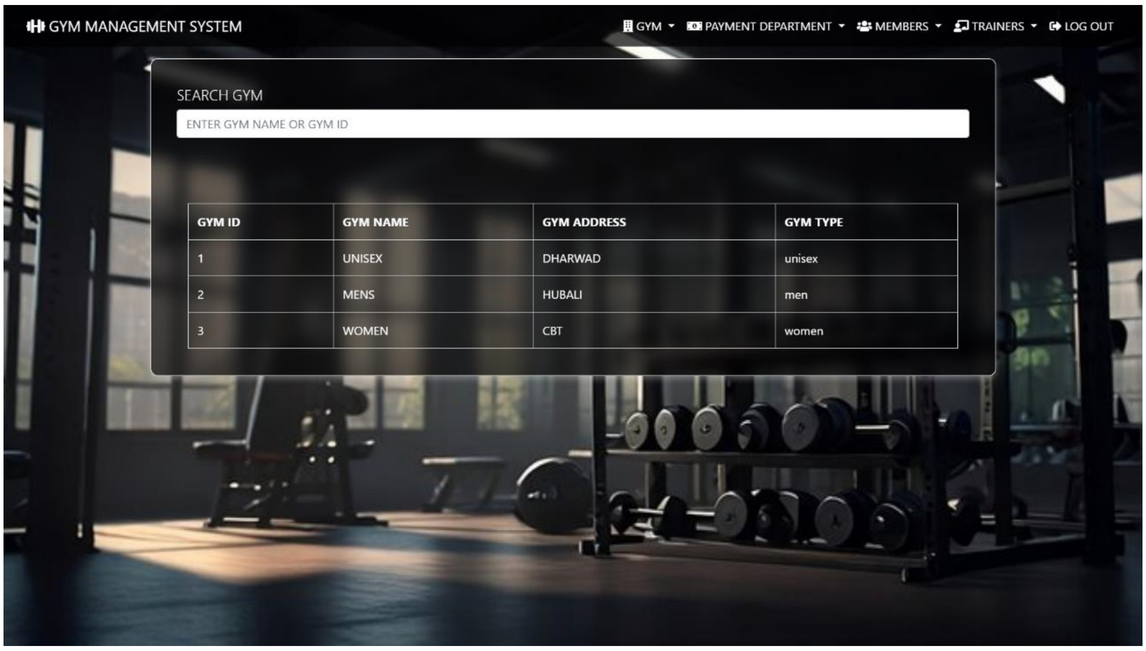
Use the navigation bar to manage the gym, payment areas, members, and trainers.

THE GYM SECTION :

i)ADD GYM



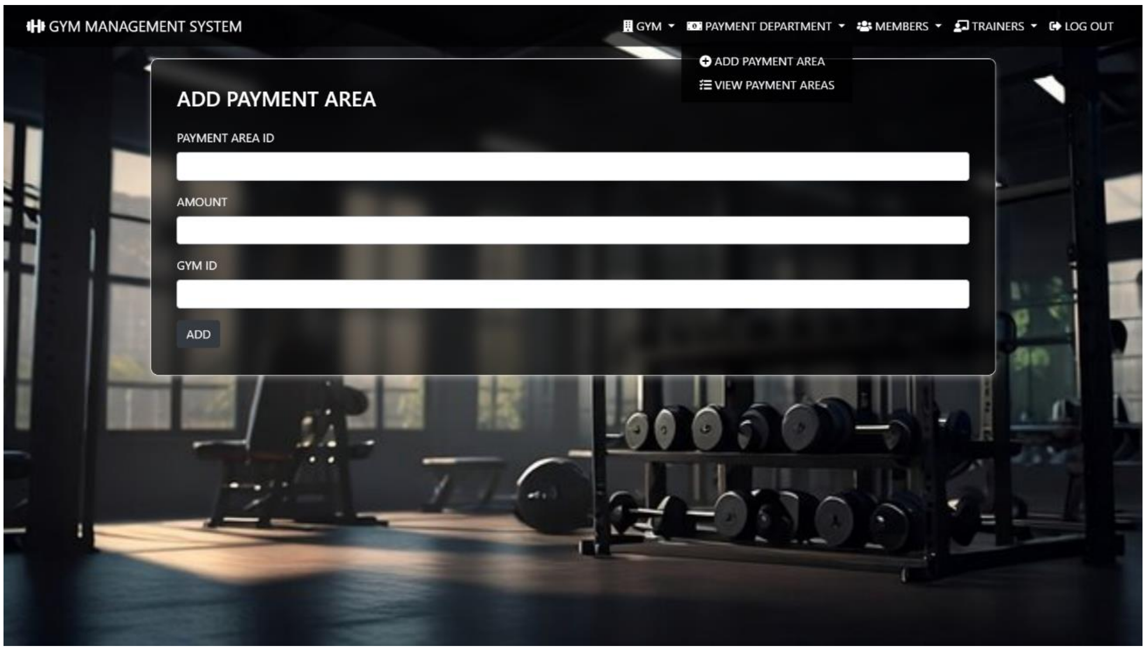
ii)VIEW ADDED GYM



GYM ID	GYM NAME	GYM ADDRESS	GYM TYPE
1	UNISEX	DHARWAD	unisex
2	MENS	HUBALI	men
3	WOMEN	CBT	women

THE PAYMENT DEPARTMENT:

i)ADD PAYMENT



ADD PAYMENT AREA

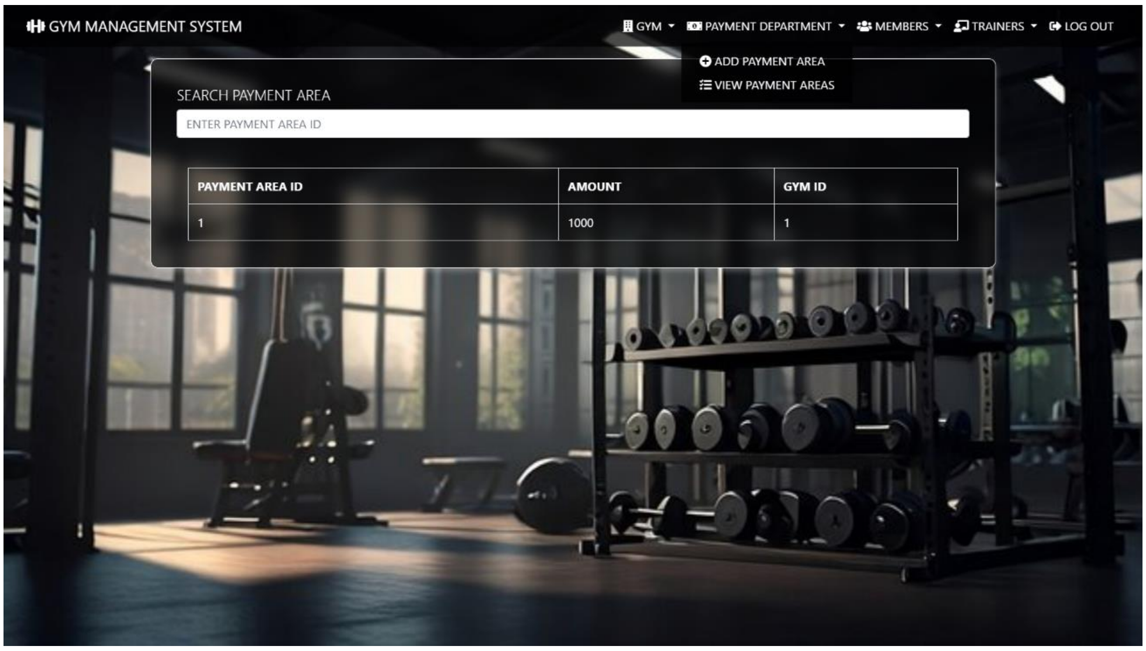
PAYMENT AREA ID

AMOUNT

GYM ID

ADD

ii)VIEW PAYMENT AREA



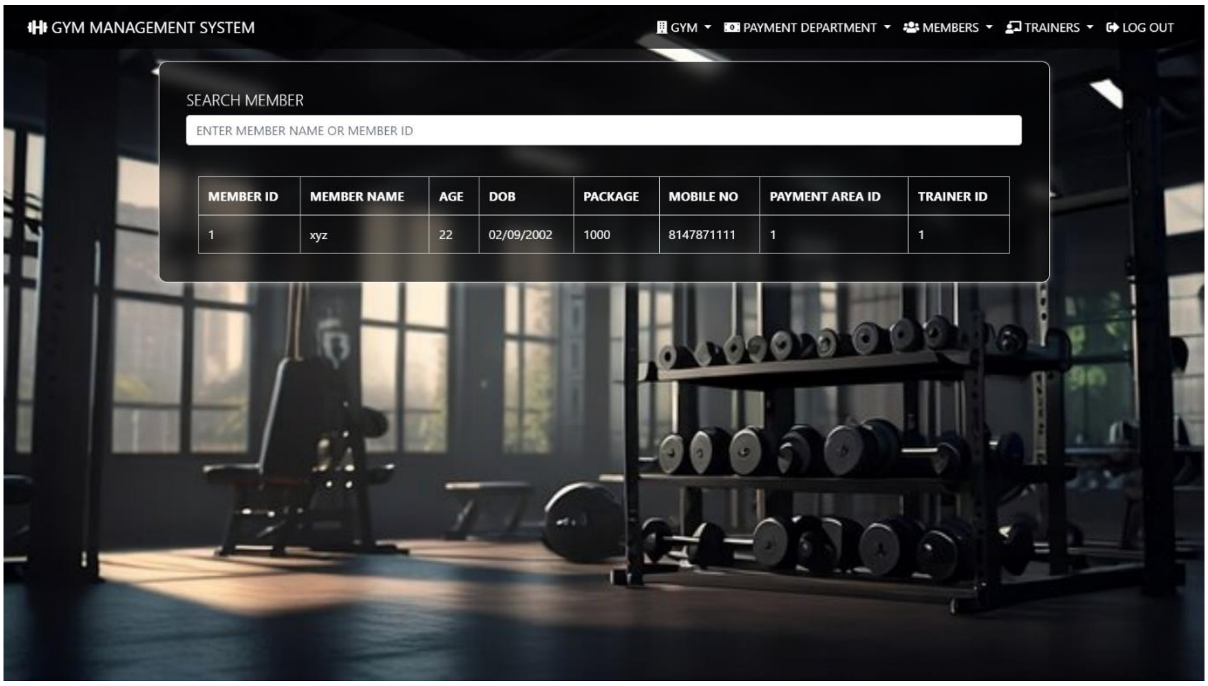
SEARCH PAYMENT AREA

ENTER PAYMENT AREA ID

PAYMENT AREA ID	AMOUNT	GYM ID
1	1000	1

MEMBER SECTION:

i)ADD MEMBER

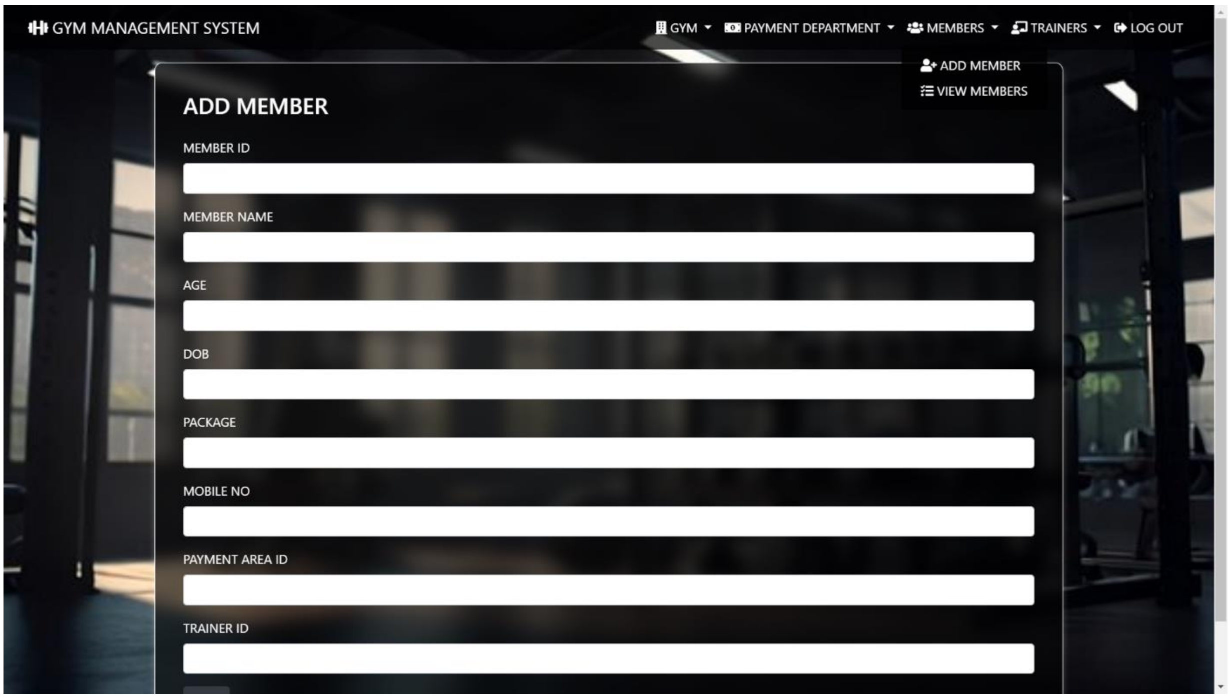


SEARCH MEMBER

ENTER MEMBER NAME OR MEMBER ID

MEMBER ID	MEMBER NAME	AGE	DOB	PACKAGE	MOBILE NO	PAYMENT AREA ID	TRAINER ID
1	xyz	22	02/09/2002	1000	8147871111	1	1

ii)VIEW THE MEMBER AREA



ADD MEMBER

MEMBER ID

MEMBER NAME

AGE

DOB

PACKAGE

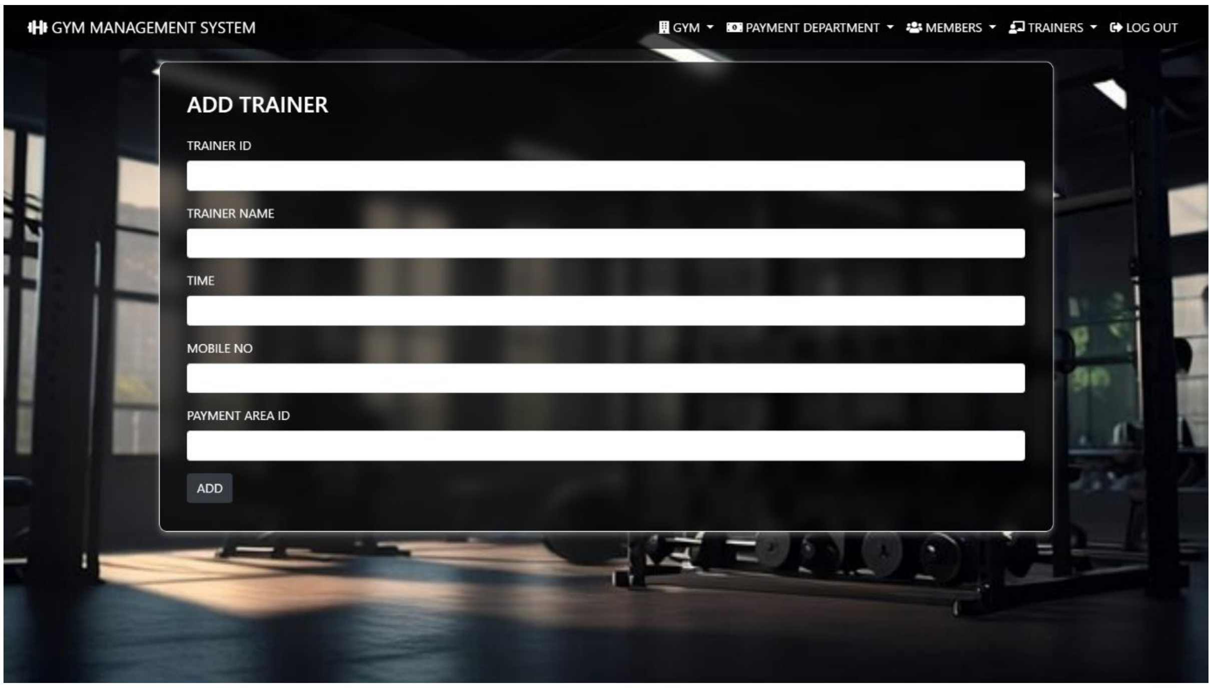
MOBILE NO

PAYMENT AREA ID

TRAINER ID

TRAINER SECTION:

i)ADD TRAINER



ADD TRAINER

TRAINER ID

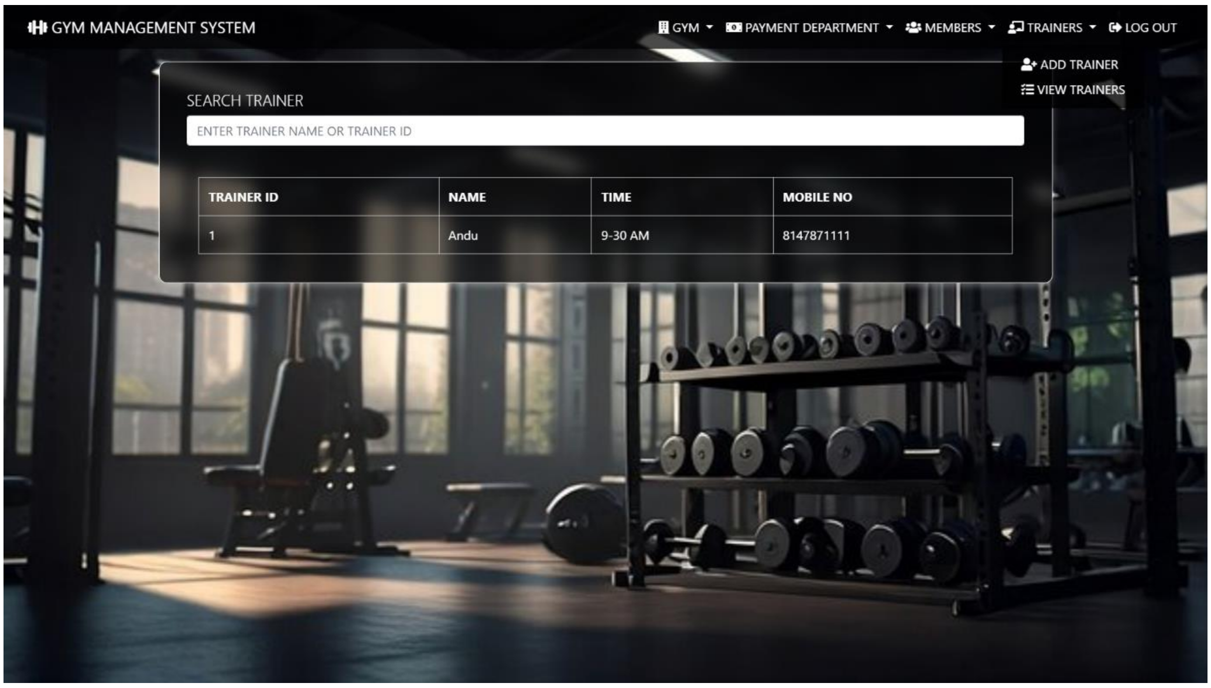
TRAINER NAME

TIME

MOBILE NO

PAYMENT AREA ID

ii)VIEW THE TRAINER AREA



SEARCH TRAINER

ENTER TRAINER NAME OR TRAINER ID

TRAINER ID	NAME	TIME	MOBILE NO
1	Andu	9-30 AM	8147871111

Overview of " Gym Management System " Project:

The "Gym Management System" is a comprehensive software solution designed to optimize operations and enhance member management for gyms catering to diverse clientele. It provides a centralized platform for administrators to efficiently manage gym facilities, memberships, payments, and trainer schedules, ensuring seamless operations and improved member experience.

1.HOME Page:

- **Introduction:** Serves as the initial landing page for administrators, designed to provide a seamless entry point into the Gym Management System. Features a visually compelling background image that reflects the energy and essence of fitness and gym management**.**
- **Navigation:** Clear and user-friendly navigation interface prominently displays options for administrative functions. Intuitive design ensures administrators can swiftly access key features without confusion or unnecessary navigation steps.
- **Purpose Clarity:** The background image effectively communicates the system's primary focus on gym management and administrative efficiency. It aligns administrator expectations with the comprehensive functionalities offered by the Gym Management System.
- **Responsive Design:** Ensures seamless accessibility across various devices (desktops, tablets, mobiles) with a responsive design approach. Maintains consistent user experience and functionality regardless of screen size or platform, optimizing usability for administrators**.**
- **Admin Login:**Authentication: Secure login interface for administrators to access administrative functionalities**.**

- **Design:** User-friendly interface designed for ease of use and efficient navigation within the Gym Management System.

2. Admin Dashboard - GYM Management:

1. GYM Management:

- **Add Gym:** Admin can add new gyms by entering details such as gym name, location, and ID.
- **View Gym Details:** Admin can view the details of all gyms currently registered in the system.
- **Edit Gym:** Allows admin to edit existing gym details, including name, location, and other information.
- **Update Gym:** Admin can update gym information to reflect any changes or corrections.
- **Delete Gym:** Provides the functionality to remove gyms from the system as needed.

3.Payment Department - Payment Management:

1. **Add Payment:**
   - Admin can add new payments by entering details such as payment ID, package details, and associating it with a gym ID.
2. **View Payments:**
   - Admin can view the details of all payments currently registered in the system, including payment ID, package information, and associated gym details.

3. **Edit Payment:**
    - Allows admin to edit existing payment details, including payment ID, package details, and associated gym ID.
4. **Update Payment:**
    - Admin can update payment information to reflect any changes or corrections made to payment details.
5. **Delete Payment:**
    - Provides the functionality to remove payments from the system as needed, ensuring flexibility in managing payment records.

4. Members Management:

1.Add Member:

- Admin can add new members by entering details such as member ID, name, age, date of birth (DOB), selected package, mobile number, payment area ID, and trainer ID.

2.View Members:

- Admin can view the details of all members currently registered in the system, including their ID, name, age, DOB, package, mobile number, payment area, and assigned trainer.

3.Edit Member:

- Allows admin to edit existing member details, including member ID, name, age, DOB, selected package, mobile number, payment area ID, and trainer ID.

4.Update Member:

- Admin can update member information to reflect any changes or corrections made to member details.

5.Delete Member:

- Provides the functionality to remove members from the system as needed, ensuring flexibility in managing member records.

5.Trainer Management:

1.Add Trainer:

- Admin can add new trainers by entering details such as trainer ID, name, availability time, mobile number, and payment area ID**.**

2.View Trainers:

- Admin can view the details of all trainers currently registered in the system, including their ID, name, availability time, mobile number, and assigned payment area.

3.Edit Trainer:

- Allows admin to edit existing trainer details, including trainer ID, name, availability time, mobile number, and payment area ID.

4.Update Trainer:

- Admin can update trainer information to reflect any changes or corrections made to trainer details.

5.Delete Trainer:

- Provides the functionality to remove trainers from the system as needed, ensuring flexibility in managing trainer records.

6. Database and Storage:

- **Structured Data Management:** The Gym Management System uses a relational database to organize gym, member, payment, and trainer data securely and efficiently.
- **Data Integrity Measures:** Robust measures ensure data accuracy and security, including validation checks, transaction logging, and encryption.

Scheduled backups and disaster recovery plans safeguard against data loss and ensure system continuity.

- **Efficient Data Retrieval:** Utilizes indexed fields and optimized queries to ensure quick retrieval of gym, member, payment, and trainer information, enhancing system responsiveness.
- **Scalability:** Designed to handle growing data volumes and user interactions, accommodating future expansion of gym operations and member registrations..
- **User Access Control**: Implements role-based access control (RBAC) to restrict data access based on user roles (admin, trainer, member), ensuring data confidentiality and integrity**.**

.

7. User Interface and Experience:

- **Responsive Design:** "GYM MANAGEMENT SYSTEM" features a responsive user interface design that adapts seamlessly to various devices and screen sizes, providing consistent access and functionality across desktops, tablets, and smartphones.
- **Intuitive Navigation**: The user interface incorporates intuitive navigation menus, clear call-to-action buttons, and logical information architecture, enhancing usability and facilitating ease of use for and administrators.
- **Accessible Design Elements:** User interface elements such as forms, tables, and interactive components are designed with accessibility principles in mind, ensuring usability for users with diverse needs and capabilities.
- **Visual Clarity and Consistency:** Visual design elements, including color schemes, typography, and iconography, are carefully selected to promote clarity, readability, and visual appeal, enhancing the overall user experience.
- **Interactive Features:** The application includes interactive features such as drag-and-drop functionality, real-time data updates, and inline editing capabilities, empowering users to interact dynamically with content and perform tasks efficiently.

- **Performance Optimization:** User interface performance is optimized through techniques such as lazy loading of content, caching of frequently accessed resources, and minification of JavaScript and CSS files, ensuring swift response times and smooth user interactions.
- **Cross-Browser Compatibility:** The user interface is tested and optimized for compatibility across popular web browsers, ensuring consistent functionality and visual presentation regardless of the user's browser preferences.

8. Maintenance and Scalability:

- **Continuous Improvement:** The project is supported by a proactive maintenance strategy that includes regular updates, bug fixes, and feature enhancements based on user feedback and technological advancements.
- **Scalability Planning:** Architecture and infrastructure are designed with scalability in mind, capable of accommodating growth in user base, data volume, and application complexity without compromising performance or user experience.
- **Modular Architecture:** The application is structured with a modular architecture, facilitating the addition of new features, integration of third-party services, and adaptation to evolving business requirements or technological innovations.
- **Performance Monitoring:** Ongoing performance monitoring and optimization efforts ensure that the application maintains optimal performance levels, even during peak usage periods or when handling large datasets.
- **Security Updates:** Security patches and updates are applied regularly to protect against emerging threats and vulnerabilities, safeguarding user data, application integrity, and compliance with regulatory standards.
- **Backup and Recovery:** Robust backup and recovery protocols are implemented to mitigate risks of data loss or system disruptions, ensuring continuity of service and rapid recovery in the event of unforeseen incidents.
- **Compliance and Governance:** The project adheres to industry best practices, regulatory requirements, and internal policies regarding data

privacy, security, and ethical standards, maintaining trust and confidence among users and stakeholders.

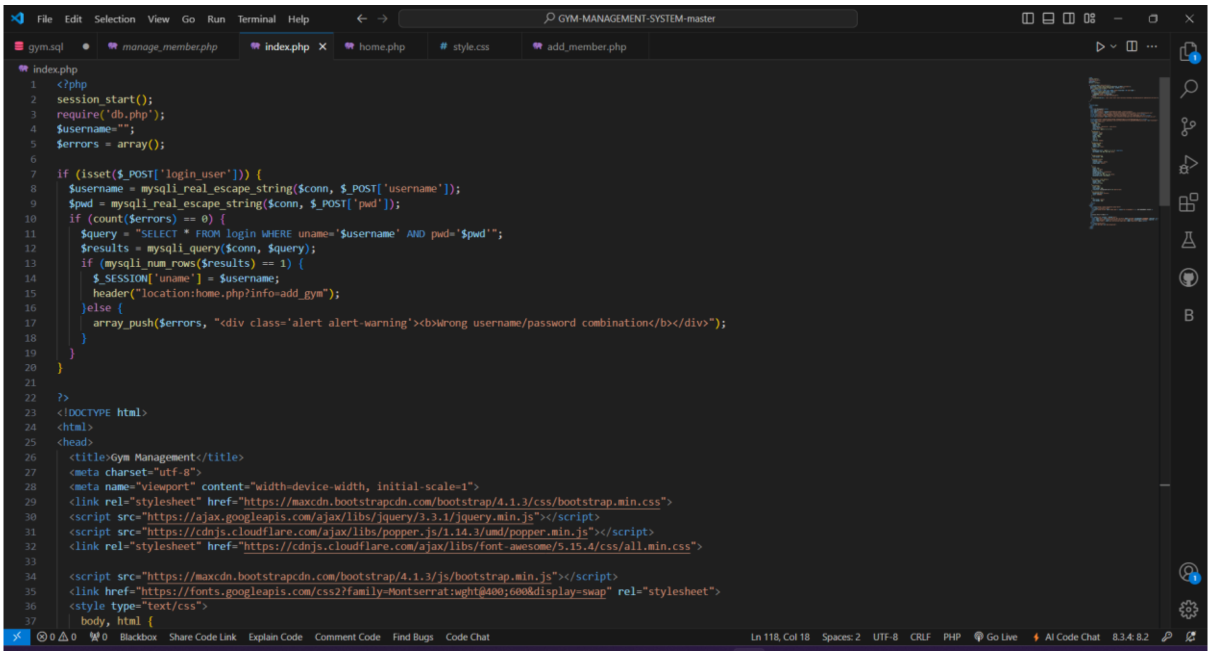
9. Future Enhancements:

- **Mobile App Development**: Create a mobile app for easy member access to schedules, bookings, and notifications.
- **Online Payment Integration:** Enable online payments for memberships, classes, and merchandise for member convenience**.**
- **Enhanced Reporting:** Develop detailed reports on gym performance, member activities, and revenue trends for better decision-making.
- **Biometric Authentication:** Implement biometric login for secure member check-ins and access control.
- **AI-Powered Recommendations:** Offer personalized workout plans and class recommendations based on member preferences and goals.
- **Customer Support Automation:** Integrate chatbots or automated systems for quick member assistance and appointment scheduling.
- **Social and Community Features:** Enhance member engagement with social features like forums, challenges, and leaderboards.
- **Virtual Reality (VR) Experiences:** Explore VR technology for immersive fitness classes and virtual training sessions.

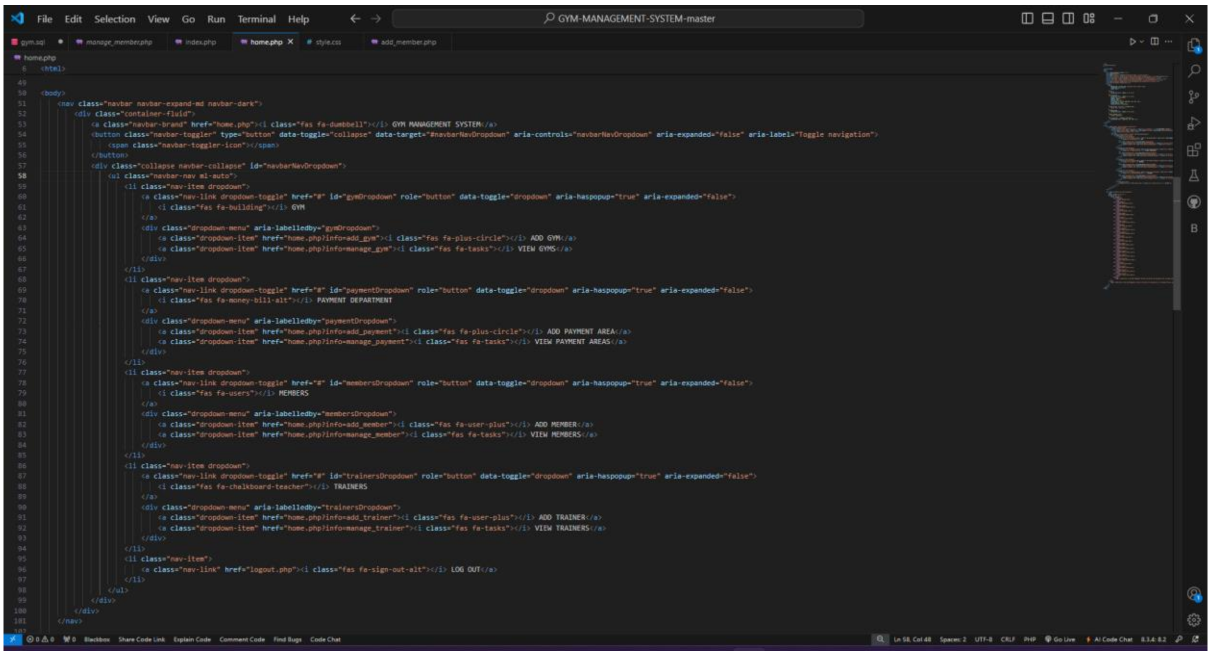
10.Continuous Integration and Deployment (CI/CD):

- **Automated Testing:** Implement automated testing frameworks (e.g., Selenium, PHPUnit) to validate code changes and ensure application functionality remains intact throughout development cycles.
- **Continuous Deployment:** Adopt CI/CD pipelines to automate build, testing, and deployment processes, enabling rapid and reliable delivery of updates and bug fixes to production environments**.**

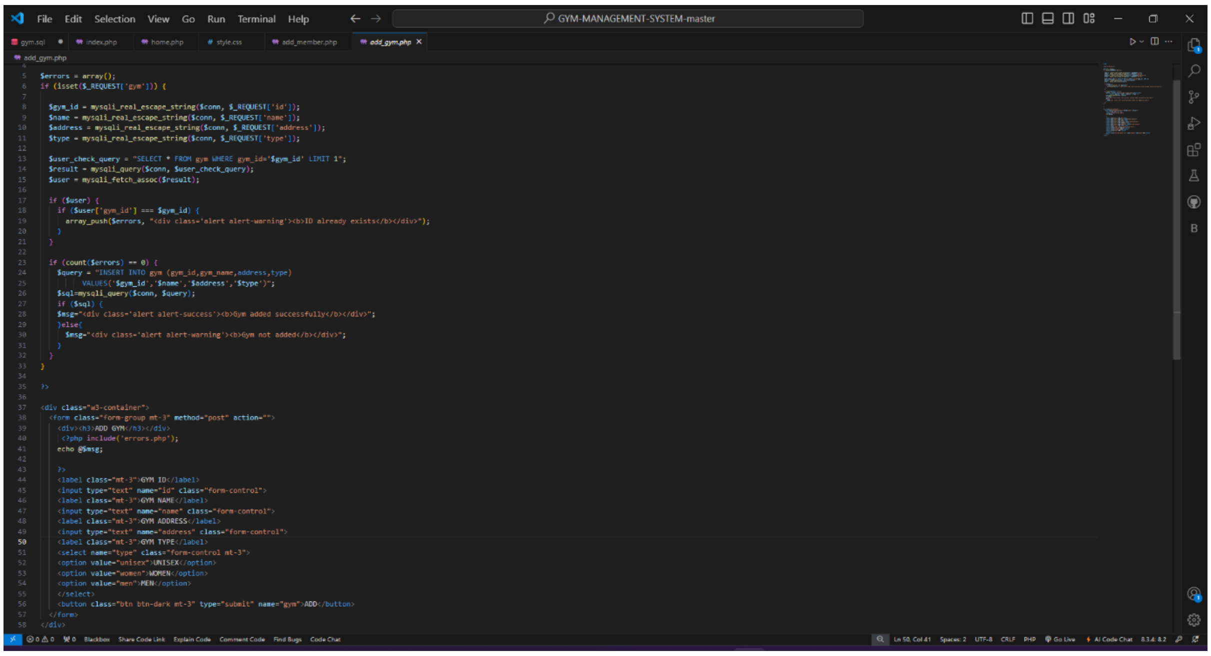
HERE IS MY CODE:



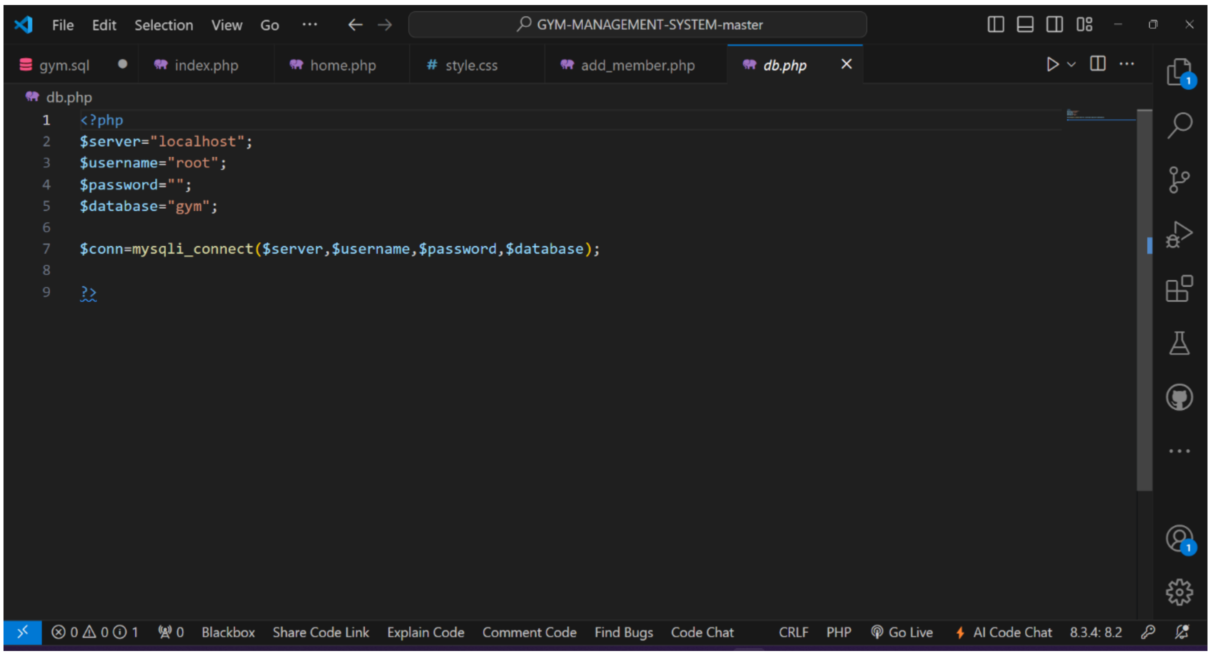
```
<?php
session_start();
require('db.php');
$username="";
$errors = array();

if (isset($_POST['login_user'])) {
  $username = mysqli_real_escape_string($conn, $_POST['username']);
  $pwd = mysqli_real_escape_string($conn, $_POST['pwd']);
  if (count($errors) == 0) {
    $query = "SELECT * FROM login WHERE uname='$username' AND pwd='$pwd'";
    $results = mysqli_query($conn, $query);
    if (mysqli_num_rows($results) == 1) {
      $_SESSION['uname'] = $username;
      header("location:home.php?info=add_gym");
    }else {
      array_push($errors, "<div class='alert alert-warning'><b>Wrong username/password combination</b></div>");
    }
  }
}

?>
<!DOCTYPE html>
<html>
<head>
  <title>Gym Management</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js"></script>
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.4/css/all.min.css">

  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js"></script>
  <link href="https://fonts.googleapis.com/css2?family=Montserrat:wght@400;600&display=swap" rel="stylesheet">
  <style type="text/css">
    body, html {
```



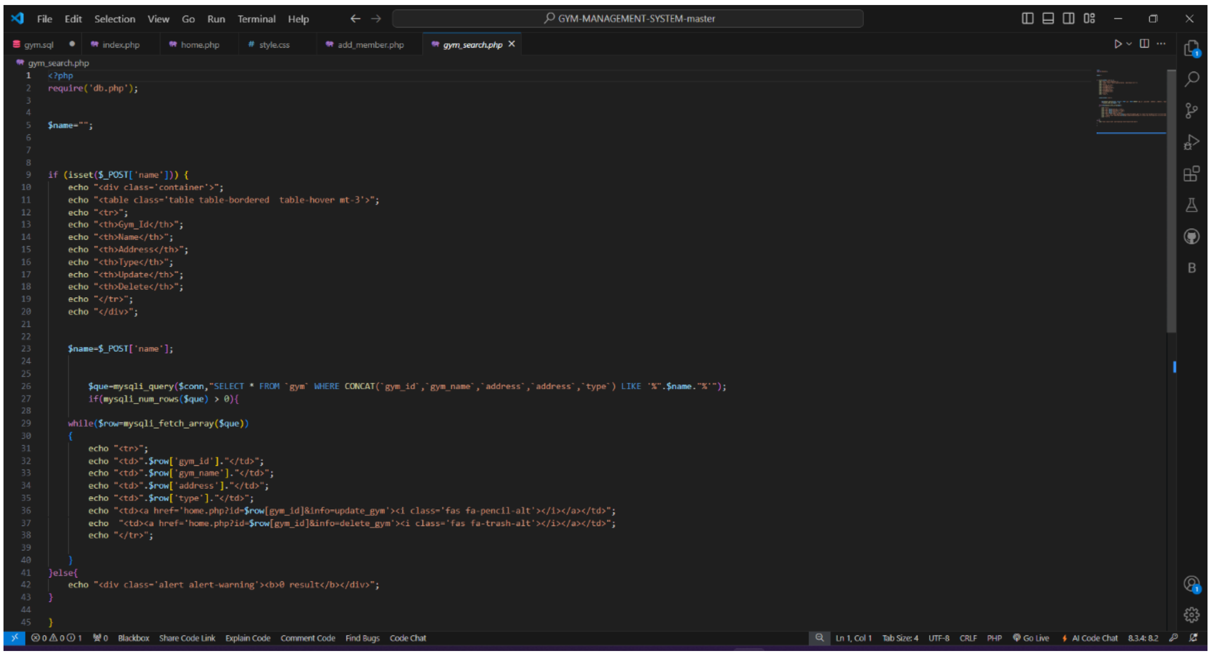
```
<html>

<body>
  <nav class="navbar navbar-expand-md navbar-dark">
    <div class="container-fluid">
      <a class="navbar-brand" href="home.php"><i class="fas fa-dumbbell"></i> GYM MANAGEMENT SYSTEM</a>
      <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNavDropdown" aria-controls="navbarNavDropdown" aria-expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarNavDropdown">
        <ul class="navbar-nav ml-auto">
          <li class="nav-item dropdown">
            <a class="nav-link dropdown-toggle" href="#" id="gymDropdown" role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
              <i class="fas fa-building"></i> GYM
            </a>
            <div class="dropdown-menu" aria-labelledby="gymDropdown">
              <a class="dropdown-item" href="home.php?info=add_gym"><i class="fas fa-plus-circle"></i> ADD GYM</a>
              <a class="dropdown-item" href="home.php?info=manage_gym"><i class="fas fa-tasks"></i> VIEW GYMS</a>
            </div>
          </li>
          <li class="nav-item dropdown">
            <a class="nav-link dropdown-toggle" href="#" id="paymentDropdown" role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
              <i class="fas fa-money-bill-alt"></i> PAYMENT DEPARTMENT
            </a>
            <div class="dropdown-menu" aria-labelledby="paymentDropdown">
              <a class="dropdown-item" href="home.php?info=add_payment"><i class="fas fa-plus-circle"></i> ADD PAYMENT AREA</a>
              <a class="dropdown-item" href="home.php?info=manage_payment"><i class="fas fa-tasks"></i> VIEW PAYMENT AREAS</a>
            </div>
          </li>
          <li class="nav-item dropdown">
            <a class="nav-link dropdown-toggle" href="#" id="membersDropdown" role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
              <i class="fas fa-users"></i> MEMBERS
            </a>
            <div class="dropdown-menu" aria-labelledby="membersDropdown">
              <a class="dropdown-item" href="home.php?info=add_member"><i class="fas fa-user-plus"></i> ADD MEMBER</a>
              <a class="dropdown-item" href="home.php?info=manage_member"><i class="fas fa-tasks"></i> VIEW MEMBERS</a>
            </div>
          </li>
          <li class="nav-item dropdown">
            <a class="nav-link dropdown-toggle" href="#" id="trainersDropdown" role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
              <i class="fas fa-chalkboard-teacher"></i> TRAINERS
            </a>
            <div class="dropdown-menu" aria-labelledby="trainersDropdown">
              <a class="dropdown-item" href="home.php?info=add_trainer"><i class="fas fa-user-plus"></i> ADD TRAINER</a>
              <a class="dropdown-item" href="home.php?info=manage_trainer"><i class="fas fa-tasks"></i> VIEW TRAINERS</a>
            </div>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="logout.php"><i class="fas fa-sign-out-alt"></i> LOG OUT</a>
          </li>
        </ul>
      </div>
    </div>
  </nav>
```



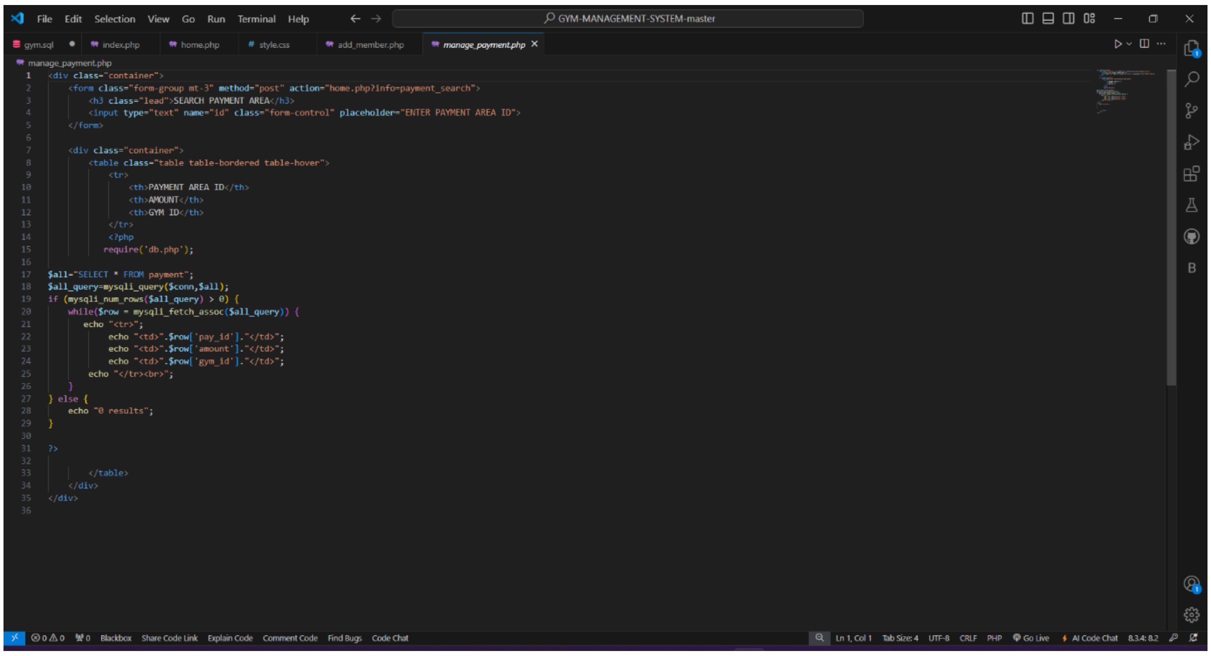
```php
$errors = array();
if (isset($_REQUEST['gym'])) {

  $gym_id = mysqli_real_escape_string($conn, $_REQUEST['id']);
  $name = mysqli_real_escape_string($conn, $_REQUEST['name']);
  $address = mysqli_real_escape_string($conn, $_REQUEST['address']);
  $type = mysqli_real_escape_string($conn, $_REQUEST['type']);

  $user_check_query = "SELECT * FROM gym WHERE gym_id='$gym_id' LIMIT 1";
  $result = mysqli_query($conn, $user_check_query);
  $user = mysqli_fetch_assoc($result);

  if ($user) {
    if ($user['gym_id'] === $gym_id) {
      array_push($errors, "<div class='alert alert-warning'><b>ID already exists</b></div>");
    }
  }

  if (count($errors) == 0) {
    $query = "INSERT INTO gym (gym_id,gym_name,address,type)
          VALUES('$gym_id','$name','$address','$type')";
    $sql=mysqli_query($conn, $query);
    if ($sql) {
    $msg="<div class='alert alert-success'><b>Gym added successfully</b></div>";
    }else{
      $msg="<div class='alert alert-warning'><b>Gym not added</b></div>";
    }
  }
}

?>

<div class="w3-container">
  <form class="form-group mt-3" method="post" action="">
    <div><h3>ADD GYM</h3></div>
     <?php include('errors.php');
    echo @$msg;

    ?>
    <label class="mt-3">GYM ID</label>
    <input type="text" name="id" class="form-control">
    <label class="mt-3">GYM NAME</label>
    <input type="text" name="name" class="form-control">
    <label class="mt-3">GYM ADDRESS</label>
    <input type="text" name="address" class="form-control">
    <label class="mt-3">GYM TYPE</label>
    <select name="type" class="form-control mt-3">
    <option value="unisex">UNISEX</option>
    <option value="women">WOMEN</option>
    <option value="men">MEN</option>
    </select>
    <button class="btn btn-dark mt-3" type="submit" name="gym">ADD</button>
  </form>
</div>
```



```php
<?php
$server="localhost";
$username="root";
$password="";
$database="gym";

$conn=mysqli_connect($server,$username,$password,$database);

?>
```


File Edit Selection View Go Run Terminal Help
GYM-MANAGEMENT-SYSTEM-master
gym.sql
index.php
home.php
style.css
add_member.php
gym_search.php
Blackbox Share Code Link Explain Code Comment Code Find Bugs Code Chat
Ln 1, Col 1 Tab Size: 4 UTF-8 CRLF PHP Go Live AI Code Chat

```php
<?php
require('db.php');


$name="";



if (isset($_POST['name'])) {
    echo "<div class='container'>";
    echo "<table class='table table-bordered  table-hover mt-3'>";
    echo "<tr>";
    echo "<th>Gym_Id</th>";
    echo "<th>Name</th>";
    echo "<th>Address</th>";
    echo "<th>Type</th>";
    echo "<th>Update</th>";
    echo "<th>Delete</th>";
    echo "</tr>";
    echo "</div>";


    $name=$_POST['name'];


        $que=mysqli_query($conn,"SELECT * FROM `gym` WHERE CONCAT(`gym_id`,`gym_name`,`address`,`address`,`type`) LIKE '%".$name."%'");
        if(mysqli_num_rows($que) > 0){

    while($row=mysqli_fetch_array($que))
    {
        echo "<tr>";
        echo "<td>".$row['gym_id']."</td>";
        echo "<td>".$row['gym_name']."</td>";
        echo "<td>".$row['address']."</td>";
        echo "<td>".$row['type']."</td>";
        echo "<td><a href='home.php?id=$row[gym_id]&info=update_gym'><i class='fas fa-pencil-alt'></i></a></td>";
        echo  "<td><a href='home.php?id=$row[gym_id]&info=delete_gym'><i class='fas fa-trash-alt'></i></a></td>";
        echo "</tr>";

    }
}else{
    echo "<div class='alert alert-warning'><b>0 result</b></div>";
}

}
```


File Edit Selection View Go Run Terminal Help
GYM-MANAGEMENT-SYSTEM-master
gym.sql
index.php
home.php
style.css
add_member.php
manage_payment.php
Blackbox Share Code Link Explain Code Comment Code Find Bugs Code Chat
Ln 1, Col 1 Tab Size: 4 UTF-8 CRLF PHP Go Live AI Code Chat

```php
<div class="container">
    <form class="form-group mt-3" method="post" action="home.php?info=payment_search">
        <h3 class="lead">SEARCH PAYMENT AREA</h3>
        <input type="text" name="id" class="form-control" placeholder="ENTER PAYMENT AREA ID">
    </form>

    <div class="container">
        <table class="table table-bordered table-hover">
            <tr>
                <th>PAYMENT AREA ID</th>
                <th>AMOUNT</th>
                <th>GYM ID</th>
            </tr>
            <?php
           require('db.php');

$all="SELECT * FROM payment";
$all_query=mysqli_query($conn,$all);
if (mysqli_num_rows($all_query) > 0) {
    while($row = mysqli_fetch_assoc($all_query)) {
       echo "<tr>";
            echo "<td>".$row['pay_id']."</td>";
            echo "<td>".$row['amount']."</td>";
            echo "<td>".$row['gym_id']."</td>";
        echo "</tr><br>";
    }
} else {
    echo "0 results";
}

?>

        </table>
    </div>
</div>
```

SYSTEM REQUIREMENTS

Software Requirement:

1. Front End: Chrome or Any Search Engine
2. Wamp x64 bit
3. Back End: Visual Studio Code or any Text Editor

Hardware Requirement:

1. Processor- Intel i3 or more
2. RAM- 2 GB or more
3. Hard Disk- 500GB or more

3.DATABASE DESIGNE

3.1 Designing Relational Databases:

Schema Design:

- **Enhanced Schema Design:**

  Enhanced schema design focuses on creating efficient and normalized database structures tailored for the "GYM MANAGEMENT SYSTEM" project. This involves:

  - **Tables and Columns:** Designing tables such as gym, login, member, payment, and trainer with appropriate columns (id, username,, password, etc.) to store student and administrator details securely.

    Data Types and Constraints: Choosing suitable data types (e.g., varchar, int, date) and defining constraints (NOT NULL, UNIQUE) to ensure data accuracy and integrity.

  - **Normalization:** Applying normalization techniques to minimize redundancy and maintain data consistency, such as breaking down data into smaller, related tables and using foreign keys to establish relationships.

- **Data Modeling:**

  Data modeling for the project involves:

  - **Entity-Relationship Diagrams (ERDs):** Creating ERDs to visualize and define relationships between entities like gym, login, member, payment, and trainer, clarifying how data flows and interacts within the system.

  - **Optimization:** Optimizing data models to support efficient querying and retrieval of information, ensuring that the database can handle operations smoothly as the application scales.

Process:

Application:

Applied normalization techniques within the "GYM MANAGEMENT SYSTEM" project to eliminate data redundancy and enhance data integrity. This involved:

- **Normalization Techniques**: Identified functional dependencies and decomposed tables to achieve higher normal forms (e.g., 3NF). This process ensured that each table in the database was well-structured and free from redundant data, optimizing storage and improving overall database performance.

Levels:

Achieved appropriate normal forms, primarily up to the third normal form (3NF), within the database design for the " GYM MANAGEMENT SYSTEM" project. This ensured:

- **Data Consistency**: Structured data in such a way that updates and modifications maintained consistency across related tables, preventing anomalies such as insertion, update, and deletion anomalies.
- **Reduction of Redundancy**: Minimized redundant data by organizing tables into normalized forms, thereby reducing storage requirements and improving the efficiency of data retrieval operations.

Outcome:

Efficient Database:

Ensured the creation of a well-structured and efficient database system for the " GYM MANAGEMENT SYSTEM" project. This outcome:

- **Enhanced Data Consistency**: Facilitated reliable and accurate data management by maintaining consistency throughout the database schema.

- **Optimized Storage**: Reduced storage overhead by eliminating redundant data and organizing tables according to normalization principles.

3.2 Normalization

Normalization Techniques:

- **Applied Normalization:**
  Applied normalization techniques within the " GYM MANAGEMENT SYSTEM" project to systematically organize data and reduce redundancy. By decomposing tables and identifying functional dependencies, the database design was structured up to the third normal form (3NF). This approach ensures:

- Minimized Redundancy: Elimination of redundant data storage, leading to optimized storage utilization and reduced risk of data inconsistencies.

- Improved Data Integrity: Structuring data to prevent anomalies during insertion, update, and deletion operations, maintaining data integrity across the database.

- **Performance Optimization**:
  Implemented performance optimization strategies to enhance database efficiency and responsiveness:

- Indexing: Created indexes on frequently queried columns (e.g., gym_id from the gym table, id from the login table, mem_id from the member table, pay_id from the payment table, and trainer_id from the trainer table.) to speed up data retrieval and improve query performance.

- Query Optimization: Optimized query execution plans to streamline data access pathways, ensuring quick and efficient database operations.

3.3 ER Diagrams:

Creation:

Development: Developed Entity-Relationship (ER) diagrams for the " GYM MANAGEMENT SYSTEM " project to visually represent data structures and relationships. This involved:

- Defining Entities: Identified and defined entities such as gym, login, member, payment, and trainer ,along with their attributes.
- Relationships: Gyms are associated with payments, allowing for efficient membership management and financial tracking. Each payment is linked to a member, ensuring clear membership status and benefits management.

Usage:

Understanding:

Utilized these ER diagrams as foundational blueprints during database implementation and development phases. This ensured:

- Consistent Implementation: Guided the development team in aligning database schema with the intended application functionality, fostering a shared understanding of data relationships and dependencies.

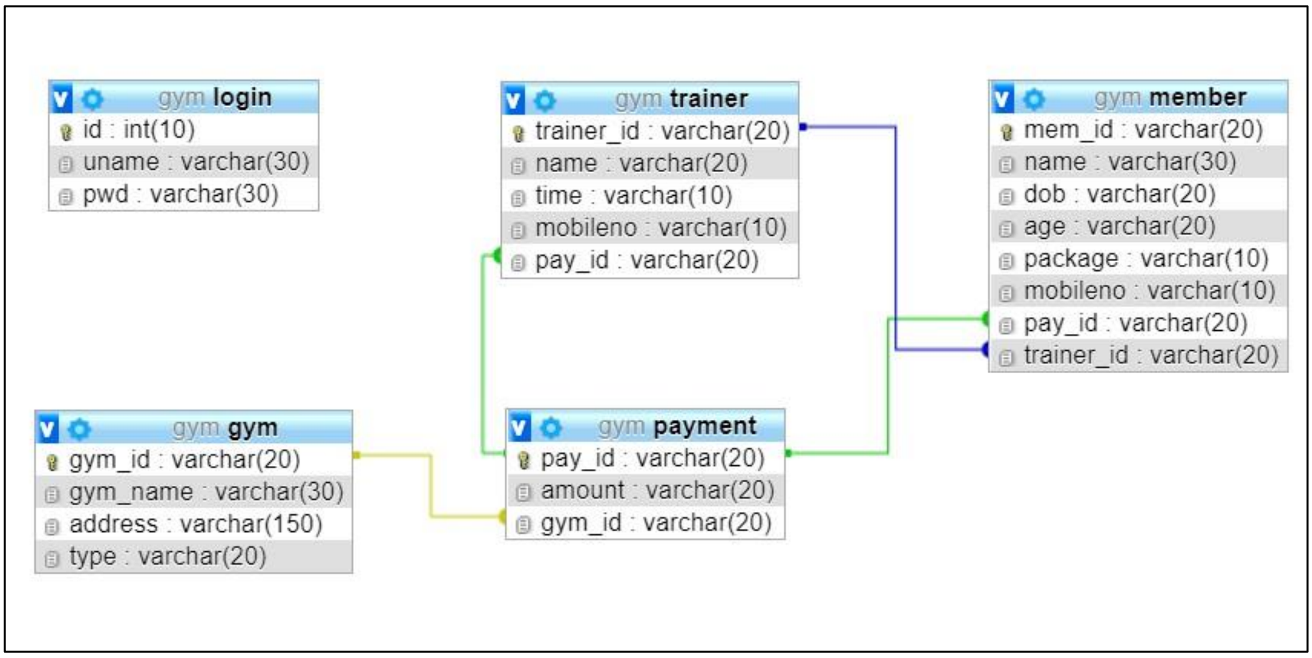
Outcome:

Clear Representation:

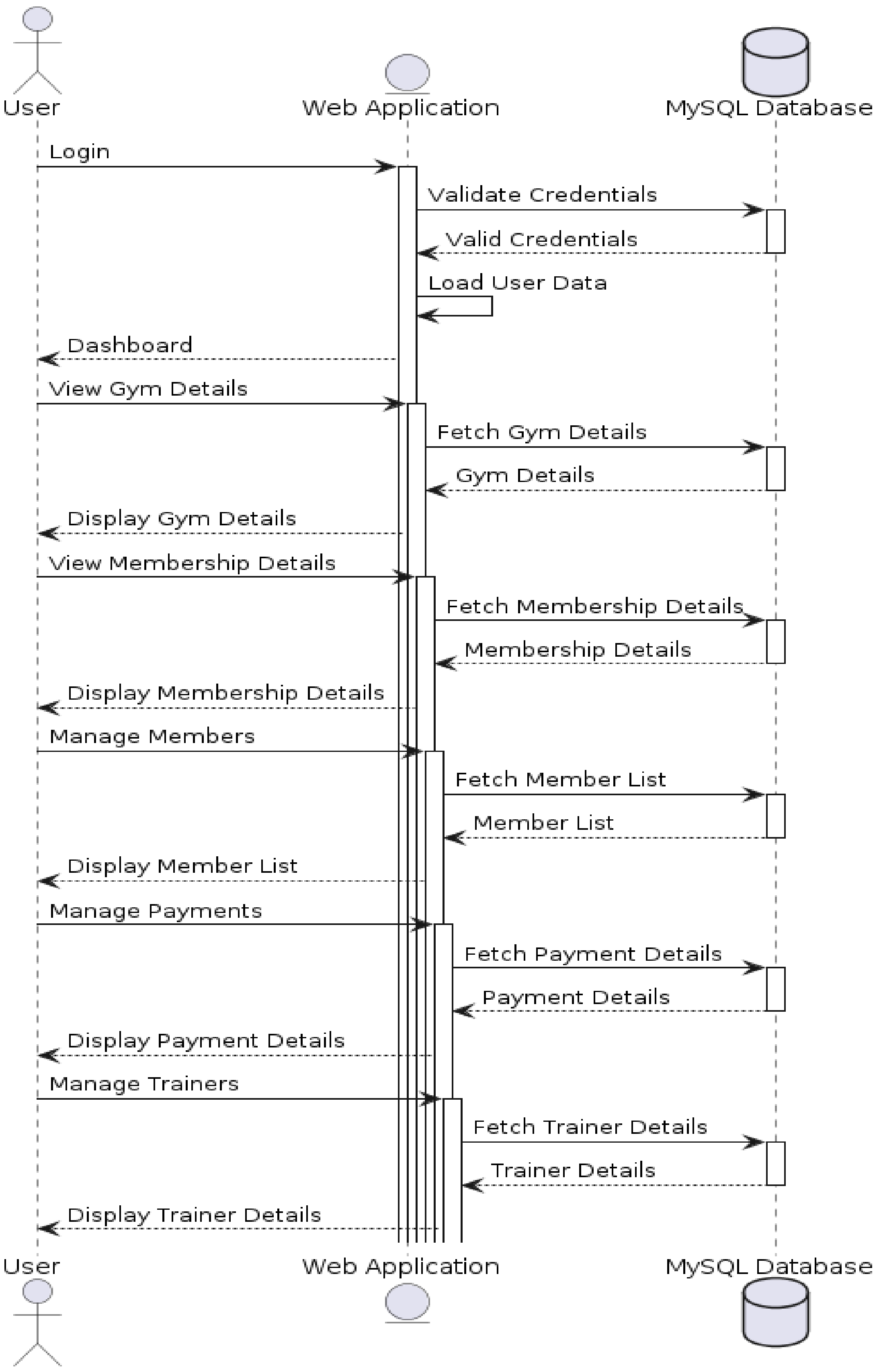
Provided a clear and precise representation of the database schema through ER diagrams. This outcome:

- Facilitated Implementation: Supported effective database implementation by clearly illustrating tables, attributes, and their relationships, enhancing communication and alignment among project stakeholders.

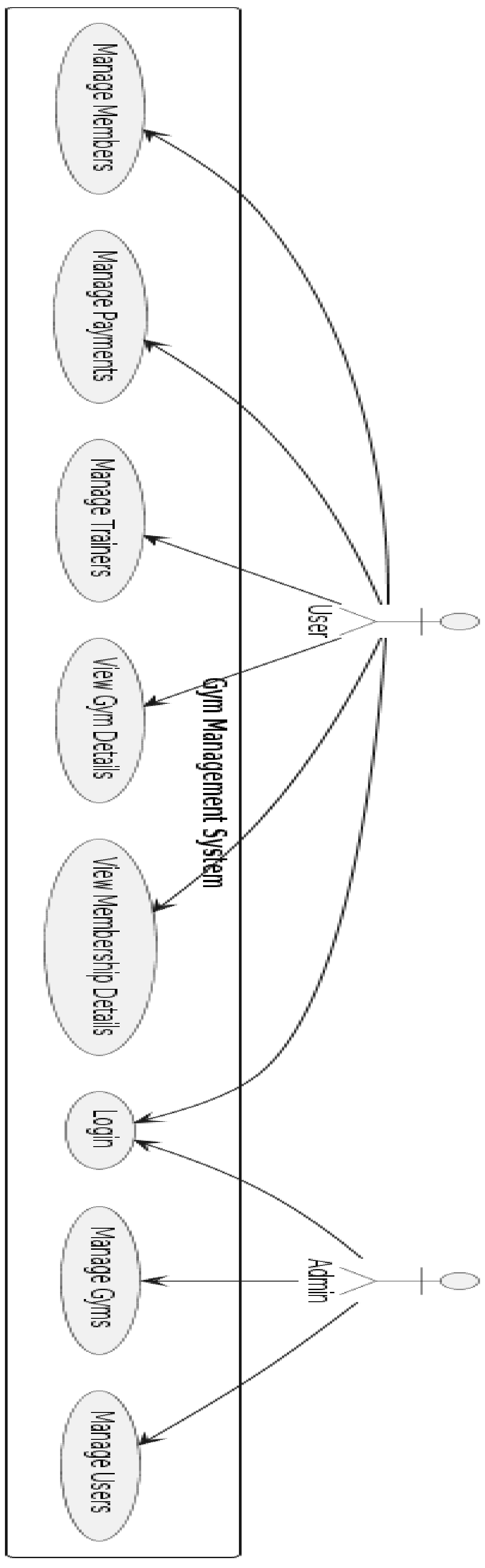
ER-Diagrams:


gym login
id : int(10)
uname : varchar(30)
pwd : varchar(30)
gym trainer
trainer_id : varchar(20)
name : varchar(20)
time : varchar(10)
mobileno : varchar(10)
pay_id : varchar(20)
gym member
mem_id : varchar(20)
name : varchar(30)
dob : varchar(20)
age : varchar(20)
package : varchar(10)
mobileno : varchar(10)
pay_id : varchar(20)
trainer_id : varchar(20)
gym gym
gym_id : varchar(20)
gym_name : varchar(30)
address : varchar(150)
type : varchar(20)
gym payment
pay_id : varchar(20)
amount : varchar(20)
gym_id : varchar(20)

SEQUENCE DAIGRAM:



USE CASE DAIGRAM:



3.4 Login and Database connection

Ensure you have a MySQL database (your_database) set up with a users table containing username and password fields. Replace your_username, your_password, and your_database with your actual MySQL credentials and database name in both login.php and login.html.

Explanation:
HTML Form (login.html): Provides a basic structure for the login page with fields for username and password, and a submit button that posts the form data to login.php.

PHP Script (login.php):

Database Connection: Establishes a connection to MySQL using mysqli.
Form Handling: Retrieves and sanitizes user inputs (username and password).
Authentication: Executes a SQL query to check if a user with provided credentials exists in the database.

Error Handling: Redirects back to the login page with an error message if credentials are invalid

Database connection

Start WAMP: Launch Apache and MySQL services through the WAMP Control Panel.
Create Database: Use phpMyAdmin to manage MySQL databases, ensuring it matches your PHP application settings.
Configure PHP: Create auth.php to securely store database credentials and establish a connection using PHP's mysqli extension.
Execute SQL Queries: Use create_tables.php to run SQL queries that define database tables and their structure.
Verify Setup: Confirm tables are created correctly in phpMyAdmin to ensure a stable database foundation for PHP applications.

Initial admin creation

- **Database Setup**: Ensure your MySQL database is set up and running in XAMPP.
- **Admin Table Design**: Similarly, define the structure of your admins table if admins have separate privileges or data requirements.
- **Create Admin Account**: Insert an initial admin account into the admins table using PHP to manage the SQL query execution.

1. BACKEND DEVELOPMENT

4.1 Process:

Utilization of ORM: Employed PHP's PDO or MySQLi extension to define models mirroring database tables, enabling streamlined interactions and ensuring data consistency.

- **Model Classes Definition**: Defined PHP classes representing database tables, specifying attributes and relationships as per application requirements.

Outcome:

- **Efficient Data Handling**: Enabled efficient data handling and operations through PHP's object-oriented approach and database abstraction layers.
- **Enhanced Application Performance**: Facilitated improved application performance and data integrity by leveraging structured models for database interactions.

4.2 View Functions

Development:

- Function Implementation: Created PHP functions to manage user actions like creating, reading, updating, and deleting (CRUD) data. This involved writing logic to process user requests and interact with the MySQL database.

- Request Handling: Managed HTTP requests and formulated appropriate responses using PHP's capabilities.

Outcome:

- Core Logic Implementation: Successfully implemented core functionalities of the application, ensuring smooth and dynamic user interactions.
- Responsive Application: Achieved responsiveness in handling user inputs and database interactions, enhancing overall user experience and application reliability.

4.3 Templates

Integration:

Usage:

- Utilization of PHP Templates: Integrated PHP templates to dynamically render HTML content based on MySQL database data. This process involved creating .php template files, passing context data fetched from the database, and utilizing PHP tags for dynamic content generation.

Outcome:

- Dynamic User Interfaces: Achieved dynamic and responsive user interfaces that display data fetched from the MySQL database. This approach enhanced user interaction by presenting real-time information effectively within the application's interface.

2. FRONTEND DEVELOPMENT

Frontend Technologies:

- **HTML, CSS, and JavaScript:** Developed skills in frontend technologies to create responsive and interactive web interfaces. This involved designing user interfaces, styling web pages, and adding interactivity to enhance user experience.

5.1 Frontend Frameworks:

Bootstrap:

- **Responsive Design:** Used Bootstrap to create responsive designs, ensuring the application looks good on various devices. Bootstrap's grid system and
- pre-built components facilitate the development of mobile-friendly web applications.
- The Bootstrap starter template is a basic HTML document pre-configured with Bootstrap. It provides a starting point for quickly building web pages using Bootstrap's components and styles.

```
<!doctype html>

<html lang="en">

  <head>

    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-
scale=1">

    <title>Bootstrap demo</title>

    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/boo
tstrap.min.css" rel="stylesheet" integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY
6hW+ALEwIH" crossorigin="anonymous">
```

```
  </head>

  <body>

    <h1>Hello, world!</h1>

<p>this is bootstrap example</p>

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bu
ndle.min.js"                                          integrity="sha384-
YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jI
eHz" crossorigin="anonymous"></script>

  </body>

</html>
```

6. TESTING & DEBUGGING

6.1 Unit Testing:

- **Writing Tests:** Wrote unit tests to verify the functionality of individual components, ensuring they work as intended. This involved using Django's testing framework to create test cases and assertions.

Outcome:

- **Early Bug Identification:** Identified and resolved bugs early in the development process, improving the quality and stability of the application.

6.2 Integration Testing:

Process:

- **Testing:** Conducted tests to ensure seamless interaction between system components, validating their integration. This involved testing interactions between different modules and verifying data flow.

Outcome:

- **Module Cohesion:** Verified that different modules worked together correctly, ensuring the system's cohesiveness and functionality.

6.3 System Testing:

Scope:

- **End-to-End Testing:** Performed end-to-end testing of the entire system to validate its overall functionality and reliability under various conditions. This involved simulating real-world scenarios and user interactions.

Outcome:

- **System Reliability:** Ensured the system met all specified requirements and performed reliably in different scenarios.

6.4 Debugging:

- **Efficient Bug Management:** Managed and resolved bugs efficiently, maintaining system quality and performance.

7.Challenges and Considerations:

7.1 Challenges and Learnings:

Challenges Faced

1. **Database Design Complexity**: Initially, grappling with the complexity of relational database design posed challenges in mapping entities and establishing effective relationships between tables.
2. **User Authentication and Authorization**: Implementing secure user authentication and authorization mechanisms required careful consideration of best practices and potential vulnerabilities, such as SQL injection and session management.
3. **Integration of Payment Systems**: Integrating a robust payment system that ensures security and reliability while adhering to industry standards presented technical and regulatory challenges.

Learnings and Solutions

1. **Structured Database Design**: Through iterative processes and collaboration with mentors, I gained proficiency in designing and optimizing database schemas, ensuring efficient data retrieval and maintenance.
2. **Security Best Practices**: Implementing multi-layered security measures, including parameterized queries and encryption techniques, improved the application's resilience against security threats.
3. **Agile Development Practices**: Adopting agile methodologies facilitated continuous improvement and responsiveness to stakeholder feedback, enabling timely adjustments and feature enhancements.

7.2 Future Enhancements:

1. Enhanced User Experience

- **Responsive Design Optimization**: Implement responsive design techniques to ensure seamless usability across various devices, enhancing accessibility for users accessing the system from smartphones and tablets.
- **Interactive Dashboard**: Develop an interactive dashboard with personalized widgets for members and administrators, providing real-time insights into gym activities, membership status, and financial metrics.

2. Expanded Functionality

- **Class Scheduling and Booking System**: Introduce a class scheduling and booking feature that allows members to view available classes, book sessions in advance, and receive reminders, enhancing engagement and participation.
- **Nutrition and Fitness Tracking**: Integrate nutrition and fitness tracking functionalities that enable members to log their dietary intake, track workout routines, and monitor progress towards fitness goals.

3. Operational Efficiency

- **Automated Membership Renewal**: Implement automated membership renewal reminders and processing, streamlining administrative tasks and improving member retention through seamless renewal processes.
- **Advanced Reporting and Analytics**: Enhance reporting capabilities with advanced analytics tools to generate comprehensive reports on membership trends, revenue forecasts, and class attendance patterns, empowering administrators to make informed decisions.

4. Security and Compliance

- **Enhanced Security Measures**: Continuously update security protocols and conduct regular audits to ensure compliance with data protection regulations and safeguard member information against potential cyber threats.

5. Integration with CRM and Marketing Tools

- **CRM Integration**: Integrate with Customer Relationship Management (CRM) systems to manage member relationships effectively, automate communication workflows, and personalize marketing campaigns based on member preferences and behavior.

7.3 References

Websites:

- **PHP Documentation**: Visit PHP.net for official PHP documentation and guides.[https://www.php.net/]
- **MySQL Documentation**: Explore MySQL Documentation for detailed MySQL guides and references.[https://dev.mysql.com/doc/]
- **Bootstrap Documentation**: Access Bootstrap Documentation for comprehensive information on using Bootstrap for frontend development.
  Links:
  1.<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.4/css/all.min.css" integrity="sha512-ABCDEF123456=" crossorigin="anonymous" />
  2.<scriptsrc="https://stackpath.bootstrapcdn.com/bootstrap/5.3.0/js/bootstrap.bundle.min.js"></script>
  3.<linkhref="https://stackpath.bootstrapcdn.com/bootstrap/5.3.0/css/bootstrap.min.css" rel="stylesheet">
- **W3Schools Bootstrap Tutorial**: Visit W3Schools Bootstrap Tutorial for tutorials on building responsive layouts with Bootstrap.. [https://www.w3schools.com/bootstrap/]

8. CONCLUSION

The Gym Management System project focused on improving how fitness centers operate and how their members experience their services. Throughout our development journey, we successfully achieved our main goals and gained valuable insights:

Our main aim was to streamline gym operations and enhance member satisfaction. By creating efficient systems for managing memberships, scheduling trainers, and handling payments, we aimed to make gym management easier and more effective. Along the way, we encountered challenges that taught us valuable lessons in problem-solving and teamwork. These experiences have not only strengthened our technical skills in database management, frontend design using

Bootstrap, and backend development with PHP and MySQL but also deepened our understanding of user needs and expectations in a digital environment.

Looking ahead, we are excited about the potential to apply our newfound knowledge and skills to future projects. We are committed to continuing our journey of learning and innovation, striving to create impactful solutions that benefit both businesses and their customers in the fitness industry and beyond.

8.1 Key Achievements

1. **Functional System Implementation:** Successfully implemented core functionalities such as member management, payment processing, and trainer scheduling, providing a robust platform for gym administrators to streamline operations**.**
2. **User-Centric Design**: Designed with a focus on user experience, incorporating responsive layouts and intuitive interfaces that facilitate easy navigation and interaction for both members and administrators.
3. **Security Enhancements**: Implemented stringent security measures to safeguard sensitive member data and ensure compliance with data protection regulations, enhancing trust and confidentiality.

Challenges Overcome

1. **Technical Complexity:** Overcame challenges related to database design, integration of payment systems, and ensuring seamless functionality across different devices and browsers.
2. **User Authentication and Authorization**: Addressed complexities in user authentication and authorization, implementing secure login mechanisms to protect user accounts from unauthorized access.

Future Directions

Looking ahead, there are several opportunities to further enhance and expand the Gym Management System:

1. **Feature Expansion:** Introduce additional features such as class scheduling and booking systems, nutrition tracking, and advanced reporting capabilities to cater to evolving user needs.
2. **Integration with IoT:** Explore integration possibilities with IoT devices to monitor equipment usage, optimize energy consumption, and enhance the overall gym experience**.**
3. **Continuous Improvement:** Commit to ongoing refinement based on user feedback and emerging technologies, ensuring the system remains innovative and competitive in the fitness industry**.**

8.2 Conclusion:

Throughout the Gym Management System project, I've gained invaluable technical skills and personal insights. On a technical level, I've learned database design, frontend development using Bootstrap 5, and backend development with PHP and MySQL. Implementing robust security measures and integrating third-party APIs further deepened my understanding of web application security and functionality. Personally, I've honed problem-solving abilities, improved project management skills, and strengthened teamwork through collaborative efforts. This experience has not only equipped me with a solid foundation in web development but also instilled a passion for continuous learning and innovation. I look forward to applying these skills in future projects, contributing to impactful solutions in the tech industry.