Università degli Studi di Messina

Professor Antonio Celesti

Database MOD A project

Project by Mujibullah Rabin (539269)

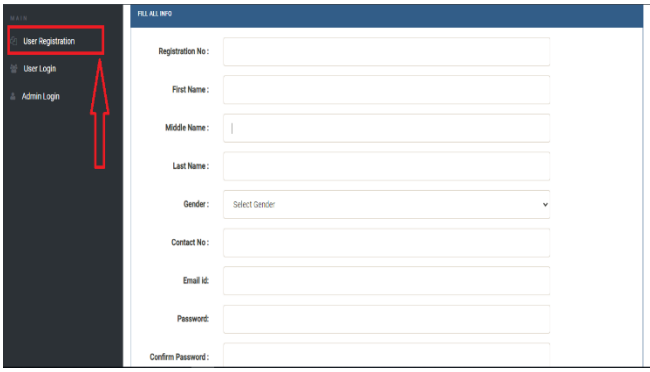
Hostel management website

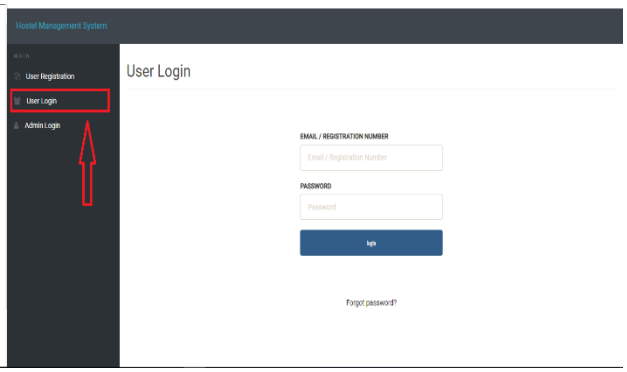
- Introduction of project
- Creating tables in MYSQL database
- Database Diagram
- Creating the website by php and JavaScript
- Explaining the php codes and java script codes
- Using CSS for designing of the website

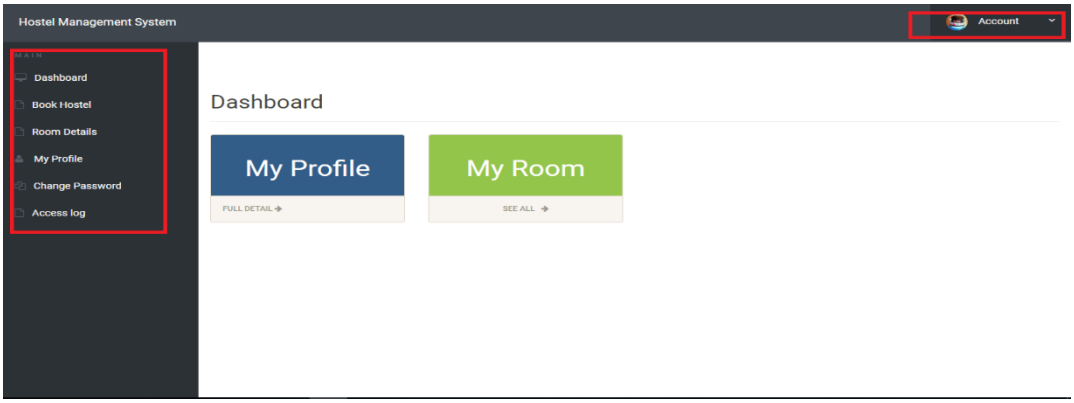
Hostel management MySQL database

Welcome to our Hostel Management System, a user-friendly website designed to streamline the process of hostel bookings for students. This comprehensive platform features two distinct logins – one for users and another for administrators.

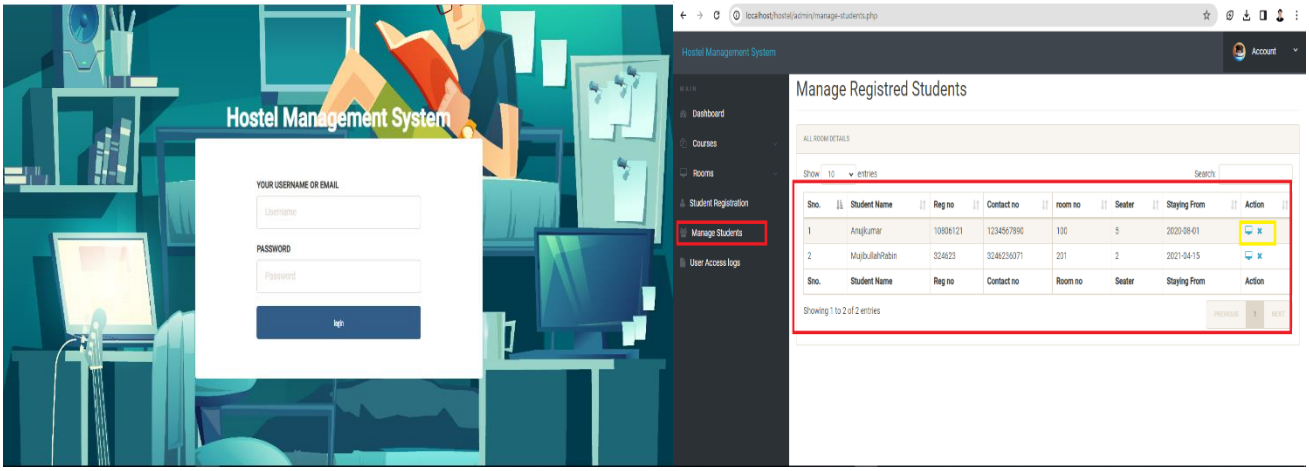
For users, the registration and login process are straightforward, allowing them to create an account, manage personal information, and seamlessly book rooms. Users can also access their payment receipts, modify their details, and view room availability. The platform aims to simplify the online booking experience, providing real-time information on room availability and enabling users to make reservations at their convenience.







Admin login, on the other hand, grants administrators' full control over the system. Admins can manage courses, add, or remove courses, oversee room details, add new rooms, and register users. The administrative login allows for efficient management of registered students, including the ability to modify registration details and remove users when necessary. Furthermore, administrators can access user login data to monitor user activity.



In summary, our Hostel Management System offers a user-friendly interface for students seeking hostel accommodations. The system's simplicity extends to the administrator's end, providing easy management of hostel facilities and user data through a clear and concise database structure.

Creating the MYSQL database

The provided SQL script creates a MySQL database named (hostel) with tables to manage hostel-related information, user registrations, courses, and administrative tasks. The script employs PhpMyAdmin, and the database design appears to support efficient data management.

The first code in MySQL These statements is often found at the beginning of SQL scripts or as part of initialization routines to ensure that the database session is configured with specific settings.

```sql
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET AUTOCOMMIT = 0;
START TRANSACTION;
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
```

So, we must create several tables according to our database diagram. So here is the explanation of the tables.

1. Admin ‘table’:

- **Purpose:** This table, named **admin**, is designed to store information about administrators who have access to the hostel management system.
- **FIELDS:**
- **id:** An integer serving as a unique identifier for each administrator.
- username: A varchar field to store the username of the administrator.
- email: A varchar field to store the email address of the administrator.
- **password:** A varchar field to store the password of the administrator (Note: In real-world applications, it is recommended to use secure password storage mechanisms like hashing).
- **reg_date:** A timestamp indicating the registration date of the administrator.
- **updation_date:** A date field representing the last update date of the administrator's information.
- **Sample Data:** The table includes an initial admin account with the username 'admin', email 'mujib.rabin@gmail.com', and a hashed password'rabin123'.

```sql
--

CREATE TABLE `admin` (
  `id` int(11) NOT NULL,
  `username` varchar(255) NOT NULL,
  `email` varchar(255) NOT NULL,
  `password` varchar(300) NOT NULL,
  `reg_date` timestamp NOT NULL DEFAULT current_timestamp(),
  `updation_date` date NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `admin`
--

INSERT INTO `admin` (`id`, `username`, `email`, `password`, `reg_date`, `updation_date`) VALUES
(1, 'admin', 'mujib.rabin@gmail.com', 'rabin123', '2023-11-04 20:31:45', '2023-11-17');

-- -------------------------------------------------------
```

2. Admin log table:

- **Purpose:** The adminlog table is responsible for recording login activities of administrators**.**
- **Fields:**
  - **id:** An integer serving as a unique identifier for each log entry.
  - adminid**:** A reference to the id field in the admin table, linking each log entry to a specific administrator.
  - **ip:** A varbinary field storing the IP address of the administrator during login.
  - **logintime**: A timestamp indicating the time of login.

```sql
CREATE TABLE `adminlog` (
  `id` int(11) NOT NULL,
  `adminid` int(11) NOT NULL,
  `ip` varbinary(16) NOT NULL,
  `logintime` timestamp NOT NULL DEFAULT current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- -------------------------------------------------------

--
```

3. courses Table

- **Purpose:** The **courses** table manages information about academic courses available within the hostel.
- **Fields:**
  - **id**: An integer serving as a unique identifier for each course.
  - **course_code**: A varchar field for the course code.
  - **course_sn**: A varchar field for the short name of the course.
  - **course_fn**: A varchar field for the full name of the course.
  - **posting_date**: A timestamp indicating the date the course information was added.
- **Sample Data:** The table includes entries for various courses such as B.Tech, B.Com, BSC, BCA, MCA, MBA, and BE.

```sql
CREATE TABLE `courses` (
  `id` int(11) NOT NULL,
  `course_code` varchar(255) DEFAULT NULL,
  `course_sn` varchar(255) DEFAULT NULL,
  `course_fn` varchar(255) DEFAULT NULL,
  `posting_date` timestamp NULL DEFAULT current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

After creating the tables, we will insert our data in that for example data of variables as id, course code and else:

```sql
INSERT INTO `courses` (`id`, `course_code`, `course_sn`, `course_fn`, `posting_date`) VALUES
(1, 'B10992', 'B.Tech', 'Bachelor  of Technology', '2020-07-04 19:31:42'),
(2, 'BCOM1453', 'B.Com', 'Bachelor Of commerce ', '2020-07-04 19:31:42'),
(3, 'BSC12', 'BSC', 'Bachelor  of Science', '2020-07-04 19:31:42'),
(4, 'BC36356', 'BCA', 'Bachelor Of Computer Application', '2020-07-04 19:31:42'),
(5, 'MCA565', 'MCA', 'Master of Computer Application', '2020-07-04 19:31:42'),
(6, 'MBA75', 'MBA', 'Master of Business Administration', '2020-07-04 19:31:42'),
(7, 'BE765', 'BE', 'Bachelor of Engineering', '2020-07-04 19:31:42');

-- ----------------------------------------------------------
```

We will continue making our database tables according to the diagram after creating the table**.**

The registration table is one of the most important tables in our database for users.

4. registration Table

- **Purpose:** The **registration** table stores detailed information about users who register for hostel accommodation.
- **Fields:**
  - Numerous fields such as **roomno**, **seater**, **feespm**, **foodstatus**, **stayfrom**, **duration**, and various personal details capturing the user's information and stay preferences.
- **Sample Data:** The table includes a sample registration entry with details like room number, seater capacity, fees per month, food status, stay duration, and extensive personal information of the user.

```sql
CREATE TABLE `registration` (
  `id` int(11) NOT NULL,
  `roomno` int(11) DEFAULT NULL,
  `seater` int(11) DEFAULT NULL,
  `feespm` int(11) DEFAULT NULL,
  `foodstatus` int(11) DEFAULT NULL,
  `stayfrom` date DEFAULT NULL,
  `duration` int(11) DEFAULT NULL,
  `course` varchar(500) DEFAULT NULL,
  `regno` int(11) DEFAULT NULL,
  `firstName` varchar(500) DEFAULT NULL,
  `middleName` varchar(500) DEFAULT NULL,
  `lastName` varchar(500) DEFAULT NULL,
  `gender` varchar(250) DEFAULT NULL,
  `contactno` bigint(11) DEFAULT NULL,
  `emailid` varchar(500) DEFAULT NULL,
  `egycontactno` bigint(11) DEFAULT NULL,
  `guardianName` varchar(500) DEFAULT NULL,
  `guardianRelation` varchar(500) DEFAULT NULL,
```

Here we created the table for registration, and we will do the same process inserting the data in the table.

5. rooms Table

- **Purpose:** The **rooms** table contains information about different hostel rooms.
- **Fields:**
  - **id**: An integer serving as a unique identifier for each room.
  - **seater**: An integer representing the capacity (number of occupants) of the room.
  - **room_no**: An integer indicating the room number.
  - **fees**: An integer indicating the fees associated with the room.
  - **posting_date**: A timestamp indicating when the room information was added.
- **Sample Data:** The table includes entries for various rooms, each with its unique room number, capacity, fees, and posting date.

```sql

CREATE TABLE `rooms` (
  `id` int(11) NOT NULL,
  `seater` int(11) DEFAULT NULL,
  `room_no` int(11) DEFAULT NULL,
  `fees` int(11) DEFAULT NULL,
  `posting_date` timestamp NULL DEFAULT current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `rooms`
--

INSERT INTO `rooms` (`id`, `seater`, `room_no`, `fees`, `posting_date`) VALUES
(1, 5, 100, 8000, '2020-04-11 22:45:43'),
(2, 2, 201, 6000, '2020-04-12 01:30:47'),
(3, 2, 200, 6000, '2020-04-12 01:30:58'),
(4, 3, 112, 4000, '2020-04-12 01:31:07'),
(5, 5, 132, 2000, '2020-04-12 01:31:15');

```

We will continue creating the tables according to the diagram. How many tables do we need, and which data should we add in the table.

4. We will create state table and insert the data.

And after admin log in tables finished here, we go with user log in tables.

7. userlog Table

- **Purpose:** The **userlog** table is responsible for recording login activities of users.
- **Fields:**
  - **id**: An integer serving as a unique identifier for each log entry.
  - **userId**: A reference to the user's identifier, linking each log entry to a specific user.
  - **userEmail**: A varchar field storing the email address of the user.
  - **userIp**: A varbinary field storing the IP address of the user during login.
  - **city** and **country**: Varchar fields capturing the user's city and country.
  - **loginTime**: A timestamp indicating the time of user login.

```sql
CREATE TABLE `userlog` (
  `id` int(11) NOT NULL,
  `userId` int(11) NOT NULL,
  `userEmail` varchar(255) NOT NULL,
  `userIp` varbinary(16) NOT NULL,
  `city` varchar(255) NOT NULL,
  `country` varchar(255) NOT NULL,
  `loginTime` timestamp NOT NULL DEFAULT current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `userlog`
--

INSERT INTO `userlog` (`id`, `userId`, `userEmail`, `userIp`, `city`, `country`, `loginTime`
(6, 3, '10806121', 0x3a3a31, '', '', '2020-07-20 14:56:45');
```

And as following these we will create the other tables as registration insertions until our creation of table is finished.

We start using alter table function to allow database to get modified in the website to add or remove to modify it without changing it from the code.

Here are the codes of allowing it to modify.

```sql
ALTER TABLE `admin`
  ADD PRIMARY KEY (`id`);

--
-- Indexes for table `courses`
--
ALTER TABLE `courses`
  ADD PRIMARY KEY (`id`);

--
-- Indexes for table `registration`
--
ALTER TABLE `registration`
  ADD PRIMARY KEY (`id`);

--
-- Indexes for table `rooms`
--
ALTER TABLE `rooms`
  ADD PRIMARY KEY (`id`),
  ADD KEY `room_no` (`room_no`);
```

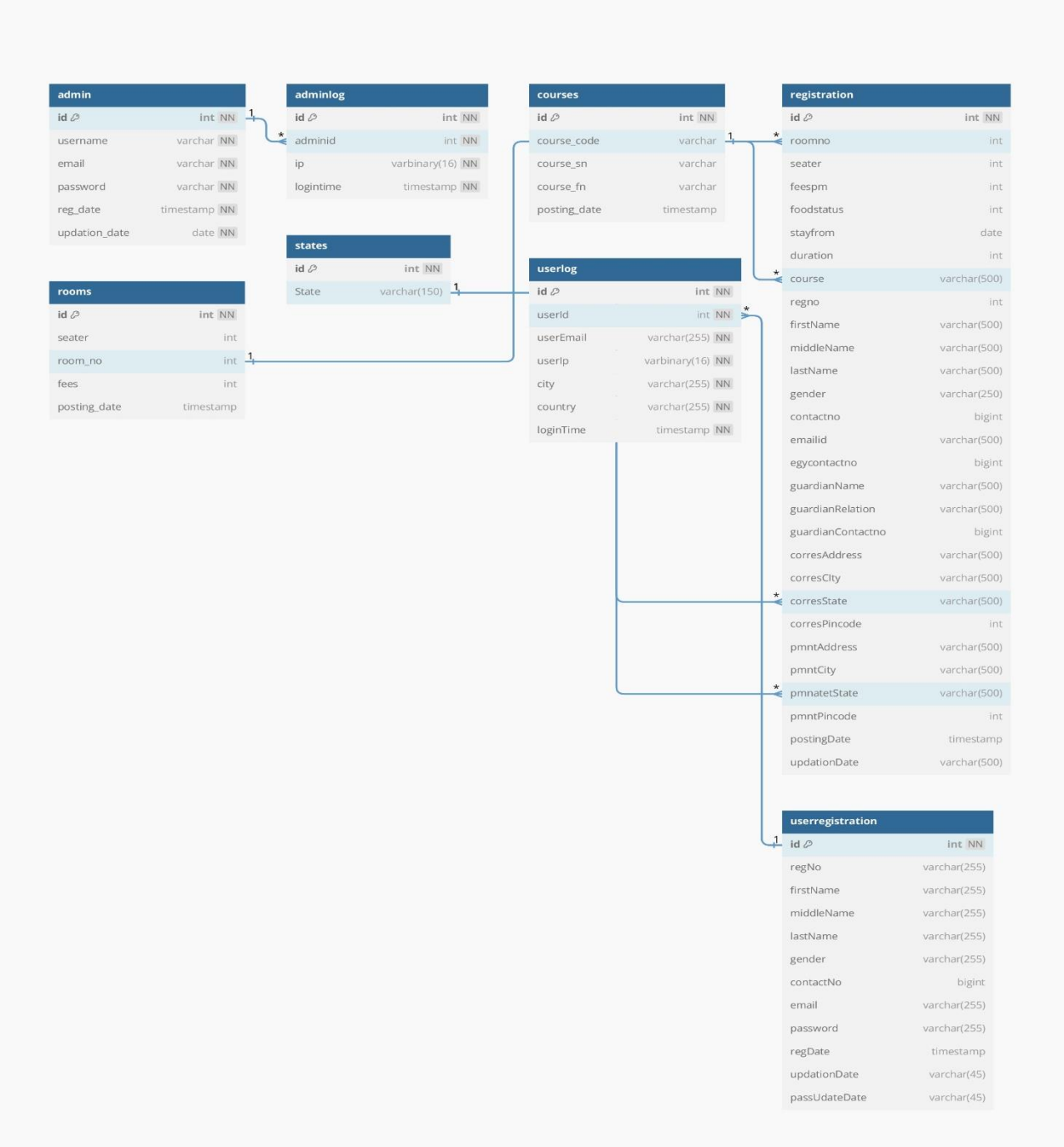
```sql
--
ALTER TABLE `rooms`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=6;

--
-- AUTO_INCREMENT for table `states`
--
ALTER TABLE `states`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=37;

--
-- AUTO_INCREMENT for table `userlog`
--
ALTER TABLE `userlog`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=7;
a
--
-- AUTO_INCREMENT for table `userregistration`
--
ALTER TABLE `userregistration`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;
```

This statement adds a primary key index to the **id** column and an additional index on the **email** column of the **user registration** table. The **email** index can expedite searches based on email address.

MYSQL database Diagram:



Functionalities of PHP, JS, CSS, HTML

1. **PHP (Hypertext Preprocessor):**

   - **File Handling:** PHP is likely used to handle the communication between your HTML pages and the MySQL database. This involves executing SQL queries to retrieve, insert, update, or delete data from the database.

   - **User Authentication:** The **admin** and **userregistration** tables suggest user authentication functionality. PHP is likely used to handle user login and registration processes, checking credentials against the database.

access-log.php	6/10/2019 9:25 AM	PHP Source File	4 KB
book-hostel.php	7/19/2020 7:33 PM	PHP Source File	15 KB
change-password.php	6/27/2016 6:22 PM	PHP Source File	6 KB
check_availability.php	7/7/2020 11:43 AM	PHP Source File	2 KB
dashboard.php	6/10/2019 7:05 AM	PHP Source File	4 KB
forgot-password.php	6/26/2016 9:27 PM	PHP Source File	4 KB
get_seater.php	7/19/2020 7:35 PM	PHP Source File	1 KB
hostel.css	3/23/2011 1:52 AM	Cascading Style S...	5 KB
index.php	7/7/2020 11:30 AM	PHP Source File	5 KB
login.php	6/27/2016 6:08 PM	PHP Source File	5 KB
logout.php	4/13/2016 11:25 AM	PHP Source File	1 KB
my-profile.php	7/19/2020 3:40 PM	PHP Source File	7 KB
registration.php	7/7/2020 11:19 AM	PHP Source File	7 KB
room-details.php	7/19/2020 3:30 PM	PHP Source File	6 KB

2. **HTML (Hypertext Markup Language):**

   - **User Interface:** HTML is used to structure the content of your web pages. It defines the layout, forms, and other elements that users interact with.

   - **Form Handling:** HTML forms are probably used for user input (e.g., registration forms). PHP likely handles form submissions, validates input, and interacts with the database accordingly.

```php
check_login();
?>
<!doctype html>
<html lang="en" class="no-js">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1, minimum-
    <meta name="description" content="">
    <meta name="author" content="">
    <meta name="theme-color" content="#3e454c">
    <title>Access Log</title>
    <link rel="stylesheet" href="css/font-awesome.min.css">
    <link rel="stylesheet" href="css/bootstrap.min.css">
    <link rel="stylesheet" href="css/dataTables.bootstrap.min.css">
    <link rel="stylesheet" href="css/bootstrap-social.css">
    <link rel="stylesheet" href="css/bootstrap-select.css">
    <link rel="stylesheet" href="css/fileinput.min.css">
    <link rel="stylesheet" href="css/awesome-bootstrap-checkbox.css">
    <link rel="stylesheet" href="css/style.css">
</head>
```

3. **JavaScript:**

- **User Interaction:** JavaScript may be used for client-side validation and dynamic behavior on the user interface. For example, ensuring that certain form fields are filled out before submission.
- **AJAX (Asynchronous JavaScript and XML):** JavaScript might be used to make asynchronous requests to the server, allowing for dynamic updates without refreshing the entire page.

bootstrap.js	4/13/2016 11:25 AM	JavaScript Source ...	68 KB
bootstrap.min.js	4/13/2016 11:25 AM	JavaScript Source ...	37 KB
bootstrap-select.js	4/13/2016 11:25 AM	JavaScript Source ...	58 KB
bootstrap-select.min.js	4/13/2016 11:25 AM	JavaScript Source ...	31 KB
Chart.min.js	4/13/2016 11:25 AM	JavaScript Source ...	56 KB
chartData.js	4/13/2016 11:25 AM	JavaScript Source ...	4 KB
dataTables.bootstrap.min.js	4/13/2016 11:25 AM	JavaScript Source ...	2 KB
fileinput.js	4/13/2016 11:25 AM	JavaScript Source ...	105 KB
jquery.dataTables.min.js	4/13/2016 11:25 AM	JavaScript Source ...	81 KB
jquery.min.js	4/13/2016 11:25 AM	JavaScript Source ...	91 KB
jquery-1.11.3-jquery.min.js	4/13/2016 11:25 AM	JavaScript Source ...	94 KB
main.js	4/13/2016 11:25 AM	JavaScript Source ...	1 KB
validation.min.js	4/13/2016 11:25 AM	JavaScript Source ...	21 KB

```javascript
(function ($) {
  'use strict';

  //<editor-fold desc="Shims">
  if (!String.prototype.includes) {
    (function () {
      'use strict'; // needed to support `apply`/`call` with `undefined`/`nu
      var toString = {}.toString;
      var defineProperty = (function () {
        // IE 8 only supports `Object.defineProperty` on DOM elements
        try {
          var object = {};
          var $defineProperty = Object.defineProperty;
          var result = $defineProperty(object, object, object) && $definePro
        } catch (error) {
        }
        return result;
      }());
      var indexOf = ''.indexOf;
      var includes = function (search) {
        if (this == null) {
          throw TypeError();
```

4. **CSS (Cascading Style Sheets):**
    - **Styling:** CSS is used for styling HTML elements, providing a visually appealing layout for your web pages.

Now, let's look at specific features based on the database tables:

- **admin and adminlog:** These tables suggest an admin system with login functionality. PHP likely handles admin login and tracks login activity (as seen in **adminlog**).
- **courses:** This table seems to store information about different courses. PHP could handle displaying this information on the website, and JavaScript might be used to make the display more interactive.
- **registration and rooms:** These tables suggest a system for managing room registrations. PHP is likely used for processing user registrations, and JavaScript could be involved in form validation or enhancing the user experience.
- **states:** This table might be related to user addresses or some other location-based functionality.
- **userlog and userregistration:** These tables suggest user login functionality. PHP likely handles user login and registration processes.

In summary, PHP handles server-side logic, interacts with the MySQL database, and generates dynamic content based on user input. HTML provides the structure of your web pages, JavaScript enhances user interactivity, and CSS styles the visual presentation. Together, they create a full-stack web application for managing admin, courses, registrations, rooms, and user-related activities.