Курсов проект по дисциплината

„Web програмиране“

Разработващ екип:

|  |  |  |  |
| --- | --- | --- | --- |
| № | Име,презиме и фамилия | Фак. № | Роля |
| 1 | Павел Георгиев Георгиев | 22621621 | Софтуерен разработчик. Ръководител на проекта и тестер. Разработване на ръководство за потребителя. |

Срок за предаване: 07.01.2025г. Водещ преподавател:

/доц.В.Божикова/

2. Задание за разработка:

3. Система за резервации

Когато искате да отидете на почивка, трябва да резервирате хотел, ако искате да се присъедините към събитие, трябва да резервирате място, а ако искате да отидете на зъболекар, трябва да резервирате посещението. И за всички тези дейности има някаква система за резервации, която работи на заден план.

* Функциите, които би било хубаво да имаме в системата за резервации: записване на час с избор на час и дата напомняния с имейл или SMS (можете да опитате да внедрите всеки инструмент на трета страна като Twilio, например)
* плащания
* списък на места/хора, които да резервирате

Описание на изискванията към системата:

Системата за резервации на хотели позволява на потребителите да преглеждат, резервират и управляват ефективно престоя си в хотел. Тя включва функции за регистрация на потребители, избор на стая, потвърждаване на резервации и управление на резервациите. Тази система е проектирана така, че да бъде интуитивна и лесна за използване, като помага на потребителите да вземат информирани решения при резервиране на хотели.

2.1. Функционални изисквания:

* Системата да съхранява данни за:
  + хотели(Hotel)
  + стаи(Room)
  + потребители(User)
  + ревюта(Review)
  + резервации(Booking)
  + плащания(Payment)
* Потребителите трябва да могат да се регистрират с потребителско име, имейл и парола.
* Регистрираните потребители трябва да могат да влизат и излизат от системата по сигурен начин.
* Потребителите трябва да могат да актуализират информацията в профила си (напр. парола, имейл).
* Системата трябва да валидира въведените от потребителя данни по време на регистрацията и влизането в системата.
* Администраторите трябва да могат да добавят, редактират и изтриват данни за хотела (напр. име, местоположение, описание и изображения).
* Потребителите трябва да могат да виждат списък с всички хотели и подробна информация за всеки хотел.
* Администраторите трябва да управляват подробностите за стаите, включително типа, цената и наличността, за всеки хотел.
* Потребителите трябва да могат да преглеждат наличните стаи за конкретен хотел.
* Потребителите трябва да могат да резервират стаи за определени дати.
* Системата трябва да валидира датите на резервациите (напр. датата на напускане трябва да е след датата на настаняване).
* Потребителите трябва да преглеждат историята на резервациите си и статуса на текущите резервации.
* Потребителите трябва да могат да оставят отзиви и оценки за хотелите, в които са отседнали.
* Потребителите трябва да виждат отзиви, оставени от други потребители за конкретен хотел.

2.2. Нефункционални изисквания:

* Системата e проектирана така, че да може да се адаптира към бъдещо разрастване (напр. добавяне на повече хотели, потребители и функции).
* Паролите на потребителите са хеширани с помощта на силен алгоритъм за криптиране.
* Системата предотвратява SQL инжектиране.
* Уебсайтът има чист, удобен за ползване интерфейс с лесна навигация.
* Той е адаптивен и да работи безпроблемно на настолни компютри, таблети и мобилни устройства.
* Системата използва модулен код, който да позволява лесни актуализации и поправки на грешки.
* Разработчиците и администраторите разполагат с подробна документация.

3. Реализация на проекта:

Уебсайтът е изграден с помощта на архитектура Model-View-Controller (MVC), за да се поддържа чисто разделяне на задачите. По-долу са описани основните компоненти на системата:

3.1 Класове

**UserManager:** Обслужва автентификация на потребители, регистрация и управление на пароли.

**HotelManager**: Управлява извличането и съхранението на данни за хотела.

**RoomManager**: Управлява данните за стаите и операциите, свързани с наличността на стаите и резервациите.

**BookingManager**: Управлява процеса на резервиране, включително създаването и промяната на резервации.

**ReviewManager**: Позволява на потребителите да оставят и преглеждат ревюта за хотели.

Всеки от тези класове взаимодейства с базата данни MySQL чрез prepared statements, за да се осигури сигурно и ефективно взаимодействие с базата данни.

3.2 Схема на базата данни

Таблица User: В нея се съхраняват UserID, Username, Email, PasswordHash и CreatedAt.

Таблица Hotel: Съхранява HotelID, Name, Location, Description, Amenities.

Таблица Room: Съхранява RoomID, HotelID, RoomType, Price, RoomImage и Availability.

Таблица Booking: Съхранява BookingID, UserID, RoomID, CheckInDate, CheckOutDate, TotalPrice и BookingStatus.

Таблица Review:Съхранява ReviewID, UserID, HotelID, Rating, Comment и CreatedAt.

Описание на класовете и техните методи:

**1. Клас UserManager**

Класът UserManager се занимава с регистрацията на потребителите, влизането в системата, валидирането и управлението на профилите.

Методи:

**validateUserRegistration ($username, $email, $password,$new\_password):**

Предназначение: Валидиране на входните данни при регистрация на потребител.

Параметри: $username (низ), $email (низ), $password (низ), $new\_password (низ).

Връща: Успех или съобщение за грешка.

**authenticateUser($username, $password):**

Цел: Удостоверява автентичността на потребител, като проверява потребителското име и хешираната парола.

Параметри: $username (низ), $password (низ).

Връща: Идентификатор на потребител, ако е успешен, или съобщение за грешка, ако данните са невалидни.

**changePassword($new\_password\_input, $confirm\_password\_input, $user\_id, &$new\_password\_err, &$confirm\_password\_err):**

Цел: Позволява на потребител да промени паролата си.

Параметри:$new\_password\_input(string), $confirm\_password\_input(string), $user\_id(int), &$new\_password\_err(string), &$confirm\_password\_err(string)

Връща: Успех или съобщение за грешка.

**createUser($username, $email, $password):**

Цел: Добавяне на потребителски профил в базата данни

Връща: Успех или неуспех

**2. Клас HotelManager**

Класът HotelManager обработва операции, свързани с хотели.

Методи:

**addHotel($name, $location, $description):**

Цел: Добавя нов хотел в базата данни.

Параметри: $name (string), $location (string), $description (string)

Връща: Успех или съобщение за грешка.

**getHotelById($hotelId):**

Цел: Извличане на данни за хотел по неговия идентификатор.

Параметри: $hotelId (int).

Връща: Информация за хотела като асоциативен масив.

**getAllHotels():**

Цел: Извличане на данни за всички хотели.

Параметри: $hotelId (int).

Връща: Информация за всички хотели като асоциативен масив.

**3. Клас RoomManager**

Класът RoomManager управлява операциите, свързани с помещенията.

Методи:

**getRoomsByHotelId($hotelId):**

Предназначение: Извлича всички налични стаи в даден хотел.

Параметри: $hotelId (int).

Връща: Списък на стаите като масив от асоциативни масиви.

**getRoomById($roomId):**

Предназначение: Извличане на данни за стаята по идентификатор на стаята.

Параметри: $roomId (int).

Връща: Подробности за стаята като асоциативен масив.

**4.Клас BookingManager**

Класът BookingManager обработва операции по резервации, като например създаване на резервации, преглед на резервации и потвърждаване на дати.

Методи:

**createBooking($user\_id, $room\_id, $check\_in, $check\_out, $total\_price, $room\_number, $status = "Confirmed")**

Цел: Създава резервация за даден потребител.

Параметри: $userId (int), $roomId (int), $checkInDate (string), $checkOutDate (string),$total\_price(decimal), $status(string).

Връща: Идентификатор на резервацията, ако е успешна, или съобщение за грешка.

**getUserBookings($userId):**

Предназначение: Извличане на всички резервации за конкретен потребител.

Параметри: $userId (int).

Връща: Списък на резервациите.

**validateBookingDates($checkInDate, $checkOutDate):**

Цел: Проверява дали предоставените дати са валидни (напр. напускането не може да бъде по-рано от настаняването).

Параметри: $checkInDate (string), $checkOutDate (string).

Връща:true или false.

**5. Клас ReviewManager**

Класът ReviewManager обработва операции, свързани с ревюта на потребители за хотели.

Методи:

**addReview($userId, $hotelId, $rating, $comment):**

Цел: Добавя ревю за хотел.

Параметри: $userId (int), $hotelId (int), $rating (int), $comment (string).

Връща: Успех или съобщение за грешка.

**getReviewsByHotelId($hotelId):**

Предназначение: Извлича всички ревюта за конкретен хотел.

Параметри: $hotelId (int).

Връща: Списък на ревютата като масив от асоциативни масиви.

Някои по-важни фрагменти код:

**public function authenticateUser($username, $password) {**

**$sql = "SELECT UserID, Username, PasswordHash FROM User WHERE Username = ?";**

**if ($stmt = $this->conn->prepare($sql)) {**

**$stmt->bind\_param("s", $username);**

**if ($stmt->execute()) {**

**$stmt->store\_result();**

**if ($stmt->num\_rows === 1) {**

**$stmt->bind\_result($user\_id, $db\_username, $hashed\_password);**

**if ($stmt->fetch() && password\_verify($password, $hashed\_password)) {**

**return ['user\_id' => $user\_id, 'username' => $db\_username];**

**}**

**}**

**}**

**$stmt->close();**

**}**

**return false;**

**}**

Тази функция проверява потребителските данни по време на влизане. Тя извлича съхранения хеш на паролата за дадено потребителско име и използва password\_verify() функцията, за да го сравни с предоставената парола. Ако данните съвпадат, тя връща данните на потребителя. В противен случай връща false. Този метод осигурява сигурно удостоверяване чрез съхраняване само на хеширани пароли и предотвратява неоторизиран достъп.

**public function changePassword($new\_password\_input, $confirm\_password\_input, $user\_id, &$new\_password\_err, &$confirm\_password\_err) {**

**$new\_password = $confirm\_password = "";**

**if (empty(trim($new\_password\_input))) {**

**$new\_password\_err = "Please enter the new password.";**

**} elseif (strlen(trim($new\_password\_input)) < 6) {**

**$new\_password\_err = "Password must have at least 6 characters.";**

**} else {**

**$new\_password = trim($new\_password\_input);**

**}**

**if (empty(trim($confirm\_password\_input))) {**

**$confirm\_password\_err = "Please confirm the password.";**

**} else {**

**$confirm\_password = trim($confirm\_password\_input);**

**if (empty($new\_password\_err) && ($new\_password !== $confirm\_password)) {**

**$confirm\_password\_err = "Passwords do not match.";**

**}**

**}**

**if (empty($new\_password\_err) && empty($confirm\_password\_err)) {**

**$sql = "UPDATE User SET PasswordHash = ? WHERE UserID = ?";**

**try {**

**if ($stmt = $this->conn->prepare($sql)) {**

**$hashed\_password = password\_hash($new\_password, PASSWORD\_DEFAULT);**

**$stmt->bind\_param("si", $hashed\_password, $user\_id);**

**if ($stmt->execute()) {**

**session\_destroy();**

**header("Location: login.php");**

**exit;**

**} else {**

**throw new Exception("Failed to update the password. Please try again later.");**

**}**

**} else {**

**throw new Exception("Database query failed.");**

**}**

**} catch (Exception $e) {**

**throw new Exception($e->getMessage());**

**}**

**}**

**}**

Този метод позволява на потребителите да променят паролите си по сигурен начин. Той включва валидиране, за да се гарантира, че паролата отговаря на изискванията за минимална дължина и потвърждава, че новата парола съответства на въведената от потребителя парола, и актуализира хеша на паролата в базата данни. Използването на password\_hash гарантира, че паролата се съхранява по сигурен начин, а сесията се унищожава, за да се подкани потребителят да влезе отново, което повишава сигурността.

**public function calculateRoomNumber($room\_id, $check\_in, $check\_out) {**

**$query = "**

**SELECT RoomNumber**

**FROM Booking**

**WHERE RoomID = ?**

**AND NOT (CheckOutDate <= ? OR CheckInDate >= ?)**

**ORDER BY RoomNumber";**

**if ($stmt = $this->conn->prepare($query)) {**

**$stmt->bind\_param('iss', $room\_id, $check\_in, $check\_out);**

**$stmt->execute();**

**$result = $stmt->get\_result();**

**$used\_numbers = [];**

**while ($row = $result->fetch\_assoc()) {**

**$used\_numbers[] = $row['RoomNumber'];**

**}**

**$stmt->close();**

**$room\_number = 1;**

**while (in\_array($room\_number, $used\_numbers)) {**

**$room\_number++;**

**}**

**return $room\_number;**

**}**

**throw new Exception("Unable to calculate room number.");**

**}**

Този метод динамично изчислява следващия свободен номер на стая за даден тип стая и период. Той проверява таблицата за резервации за припокриващи се резервации и събира списъка на вече използваните номера на стаи. Метода идентифицира първия свободен номер на стая, която не е резервирана. По този начин се гарантира, че системата ефикасно присвоява уникални номера на стаи, без да се получава прекомерно резервиране.

**if ($check\_out\_date <= $check\_in\_date) {**

**$error\_message = "Check-out date must be after the check-in date.";**

**} elseif ($check\_in\_date < $current\_date) {**

**$error\_message = "Check-in date cannot be in the past.";**

**} else {**

**try {**

**$room\_number = $bookingManager->calculateRoomNumber($room\_id, $check\_in, $check\_out);**

**$bookingManager->validateRoomAvailability($room\_id, $room\_number);**

**$num\_nights = ($check\_out\_date - $check\_in\_date) / (60 \* 60 \* 24);**

**$total\_price = $room['Price'] \* $num\_nights;**

**if (isset($\_POST['confirm\_booking']) && isset($\_SESSION['user\_id'])) {**

**$user\_id = $\_SESSION['user\_id'];**

**if ($bookingManager->createBooking($user\_id, $room\_id, $check\_in, $check\_out, $total\_price, $room\_number)) {**

**$\_SESSION['booking\_details'] = [**

**'hotel\_name' => $hotel['Name'],**

**'room\_type' => $room['RoomType'],**

**'check\_in' => $check\_in,**

**'check\_out' => $check\_out,**

**'total\_price' => $total\_price,**

**'room\_number' => $room\_number,**

**];**

**header("Location: thank\_you.php");**

**exit();**

**} else {**

**$error\_message = "Failed to create the booking. Please try again.";**

**}**

**}**

**} catch (Exception $e) {**

**$error\_message = $e->getMessage();**

**}**

**}**

Валидиране на датите на резервация:

Първото условие if гарантира, че датата на напускане е по-късна от датата на настаняване. Ако това не е така, се задава съобщение за грешка.

Условието elseif проверява дали датата на настаняване не е в миналото спрямо текущата дата. Ако датата на настаняване е по-ранна от днешната, се подава сигнал за грешка.

Присвояване на номер на стая:

След като датите са потвърдени, системата изчислява следващия наличен номер на стая за дадения тип стая и дати на резервация, като използва метода calculateRoomNumber от BookingManager. По този начин се гарантира, че няма конфликт със съществуващи резервации.

Валидиране на наличността на стаите:

Методът validateRoomAvailability гарантира, че изчисленият номер на стая действително е наличен за резервация през посочения период. Това е допълнителна предпазна мярка срещу свръхрезервиране.

Изчисляване на общата цена:

Общата цена се изчислява, като се определи броят на нощувките и се умножи по цената на стаята.

Създаване на резервация:

Системата проверява дали резервацията се потвърждава и дали потребителят е влязъл в системата.

Ако и двете условия са изпълнени, се извиква методът createBooking, за да се въведат данните за резервацията в базата данни. При успех:

Масивът $\_SESSION['booking\_details'] се попълва с данните за резервацията.

Потребителят се пренасочва към страница „Благодаря“.

Ако създаването на резервация е неуспешно, се извежда съобщение за грешка.

4. Кратко ръководство за потребителя. Възможности за развитие.

1. Регистрация

За да използват системата, потребителите трябва първо да си направят профил.

Отидете на страницата register.php.

Въведете потребителско име, имейл адрес и парола (паролата трябва да съдържа поне 6 символа).

Потвърдете паролата си и изпратете формата, за да създадете акаунта си.

При успешна регистрация ще бъдете пренасочени към страницата за вход.

2. Вход

Преминете към страницата login.php.

Въведете регистрираните си потребителско име и парола.

Ако данните са верни, ще влезете в профила си и ще бъдете пренасочени към началната страница.

3. Разглеждане на хотели

След като влезете в системата, можете да разглеждате наличните хотели, като посетите началната страница (index.php).

Кликнете върху всеки хотел, за да видите подробна информация, включително местоположение, описание и налични стаи.

4. Избор на стая и резервация

На страницата room\_selection.php можете да изберете хотел, за да видите наличните стаи.

За всяка стая можете да видите цената на нощувка и вида на стаята.

Щракнете върху бутона „Book This Room“ (Резервирай тази стая), за да инициирате резервация.

След като изберете стая, потвърдете данните за резервацията и преминете към плащане.

5. Управление на резервации

Преглеждайте резервациите си на страницата my\_bookings.php.

Тук можете да видите състоянието на резервациите си и техните подробности, включително дати на настаняване и напускане, обща цена и име на хотела.

6. Ревюта

Можете да оставите ревю за хотела.

Посетете страницата на хотела и дайте оценка заедно с коментар.

Възможности за развитие:

* Многоезична поддръжка
* Групови резервации
* Начини за плащане
* Подобрено търсене и филтриране
* Известия чрез имейл или СМС
* Двуфакторна автентификация (2FA)
* Администраторско табло
* Влизане с акаунт от социалните медии

5. Използвана литература:

<https://www.php.net/manual/en/function.date.php>

<https://stackoverflow.com/questions/325933/determine-whether-two-date-ranges-overlap>

<https://stackoverflow.com/questions/2545947/check-overlap-of-date-ranges-in-mysql>

[PHP Pass variable to next page - Stack Overflow](https://stackoverflow.com/questions/871858/php-pass-variable-to-next-page)

[PHP number\_format() Function](https://www.w3schools.com/php/func_string_number_format.asp)

[sql - How to specify decimal precision and scale number in MySQL using phpMyAdmin - Stack Overflow](https://stackoverflow.com/questions/23877487/how-to-specify-decimal-precision-and-scale-number-in-mysql-using-phpmyadmin)

[How to Design a Database for Online Hotel Booking App Like MakeMyTrip - GeeksforGeeks](https://www.geeksforgeeks.org/how-to-design-a-database-for-online-hotel-booking-app-like-makemytrip/)

<https://getbootstrap.com/docs/5.3/components/card/>

[https://getbootstrap.com/docs/5.3/examples/headers/#](https://getbootstrap.com/docs/5.3/examples/headers/)

<https://getbootstrap.com/docs/5.3/examples/buttons/>

<https://www.tutorialrepublic.com/php-tutorial/php-mysql-login-system.php>

6. Приложение

**Клас BookingManager**

<?php

class BookingManager {

private static $instance = null;

private $conn;

private function \_\_construct($conn) {

$this->conn = $conn;

}

public static function getInstance($conn) {

if (self::$instance === null) {

self::$instance = new BookingManager($conn);

}

return self::$instance;

}

public function calculateRoomNumber($room\_id, $check\_in, $check\_out) {

$query = "

SELECT RoomNumber

FROM Booking

WHERE RoomID = ?

AND NOT (CheckOutDate <= ? OR CheckInDate >= ?)

ORDER BY RoomNumber";

if ($stmt = $this->conn->prepare($query)) {

$stmt->bind\_param('iss', $room\_id, $check\_in, $check\_out);

$stmt->execute();

$result = $stmt->get\_result();

$used\_numbers = [];

while ($row = $result->fetch\_assoc()) {

$used\_numbers[] = $row['RoomNumber'];

}

$stmt->close();

$room\_number = 1;

while (in\_array($room\_number, $used\_numbers)) {

$room\_number++;

}

return $room\_number;

}

throw new Exception("Unable to calculate room number.");

}

public function validateRoomAvailability($room\_id, $room\_number) {

$query = "SELECT MaxNumber FROM Room WHERE RoomID = ?";

if ($stmt = $this->conn->prepare($query)) {

$stmt->bind\_param("i", $room\_id);

$stmt->execute();

$stmt->bind\_result($max\_number);

$stmt->fetch();

$stmt->close();

if ($room\_number > $max\_number) {

throw new Exception("No available rooms for the selected dates.");

}

} else {

throw new Exception("Failed to validate room availability.");

}

}

public function createBooking($user\_id, $room\_id, $check\_in, $check\_out, $total\_price, $room\_number, $status = "Confirmed") {

$sql = "INSERT INTO Booking (UserID, RoomID, RoomNumber, CheckInDate, CheckOutDate, TotalPrice, BookingStatus)

VALUES (?, ?, ?, ?, ?, ?, ?)";

if ($stmt = $this->conn->prepare($sql)) {

$stmt->bind\_param("iiissds", $user\_id, $room\_id, $room\_number, $check\_in, $check\_out, $total\_price, $status);

return $stmt->execute();

}

return false;

}

public function getUserBookings($user\_id) {

$query = "

SELECT b.BookingID, b.CheckInDate, b.CheckOutDate, b.TotalPrice, b.BookingStatus,

b.RoomNumber, r.RoomType, h.Name AS HotelName

FROM Booking b

JOIN Room r ON b.RoomID = r.RoomID

JOIN Hotel h ON r.HotelID = h.HotelID

WHERE b.UserID = ?

ORDER BY b.Timestamp DESC

";

if ($stmt = $this->conn->prepare($query)) {

$stmt->bind\_param('i', $user\_id);

$stmt->execute();

$result = $stmt->get\_result();

$bookings = $result->fetch\_all(MYSQLI\_ASSOC);

$stmt->close();

return $bookings;

}

return [];

}

}

**Клас HotelManager**

<?php

class HotelManager {

private static $instance = null;

private $conn;

private function \_\_construct($conn) {

$this->conn = $conn;

}

public static function getInstance($conn) {

if (self::$instance === null) {

self::$instance = new HotelManager($conn);

}

return self::$instance;

}

public function addHotel($name, $location, $description) {

$sql = "INSERT INTO Hotel (Name, Location, Description) VALUES (?, ?, ?)";

if ($stmt = $this->conn->prepare($sql)) {

$stmt->bind\_param("sss", $name, $location, $description);

return $stmt->execute();

}

return false;

}

public function getHotelById($hotel\_id) {

$sql = "SELECT \* FROM Hotel WHERE HotelID = ?";

if ($stmt = $this->conn->prepare($sql)) {

$stmt->bind\_param("i", $hotel\_id);

$stmt->execute();

$result = $stmt->get\_result();

return $result->fetch\_assoc();

}

return null;

}

public function getAllHotels() {

$sql = "SELECT \* FROM Hotel";

$result = $this->conn->query($sql);

$hotels = [];

if ($result->num\_rows > 0) {

while ($row = $result->fetch\_assoc()) {

$hotels[] = $row;

}

}

return $hotels;

}

}

**Клас ReviewManager**

<?php

class ReviewManager {

private static $instance = null;

private $conn;

private function \_\_construct($conn) {

$this->conn = $conn;

}

public static function getInstance($conn) {

if (self::$instance === null) {

self::$instance = new ReviewManager($conn);

}

return self::$instance;

}

public function getReviewsByHotelId($hotel\_id) {

$sql = "SELECT r.Rating, r.Comment, r.Timestamp, u.Username

FROM review r

JOIN user u ON r.UserID = u.UserID

WHERE r.HotelID = ?

ORDER BY r.Timestamp DESC";

if ($stmt = $this->conn->prepare($sql)) {

$stmt->bind\_param("i", $hotel\_id);

$stmt->execute();

$result = $stmt->get\_result();

return $result->fetch\_all(MYSQLI\_ASSOC);

}

return [];

}

public function addReview($user\_id, $hotel\_id, $rating, $comment) {

$sql = "INSERT INTO review (UserID, HotelID, Rating, Comment) VALUES (?, ?, ?, ?)";

if ($stmt = $this->conn->prepare($sql)) {

$stmt->bind\_param("iiis", $user\_id, $hotel\_id, $rating, $comment);

return $stmt->execute();

}

return false;

}

}

**Клас RoomManager**

<?php

class RoomManager {

private static $instance = null;

private $conn;

private function \_\_construct($conn) {

$this->conn = $conn;

}

public static function getInstance($conn) {

if (self::$instance === null) {

self::$instance = new RoomManager($conn);

}

return self::$instance;

}

public function getRoomById($room\_id, $hotel\_id) {

$sql = "SELECT \* FROM Room WHERE RoomID = ? AND HotelID = ?";

if ($stmt = $this->conn->prepare($sql)) {

$stmt->bind\_param("ii", $room\_id, $hotel\_id);

$stmt->execute();

$result = $stmt->get\_result();

return $result->fetch\_assoc();

}

return null;

}

public function getRoomsByHotelId($hotel\_id) {

$sql = "SELECT \* FROM Room WHERE HotelID = ?";

if ($stmt = $this->conn->prepare($sql)) {

$stmt->bind\_param('i', $hotel\_id);

$stmt->execute();

$result = $stmt->get\_result();

return $result->fetch\_all(MYSQLI\_ASSOC);

}

return [];

}

}

**Клас UserManager**

<?php

class UserManager {

private static $instance = null;

private $conn;

private function \_\_construct($conn) {

$this->conn = $conn;

}

public static function getInstance($conn) {

if (self::$instance === null) {

self::$instance = new UserManager($conn);

}

return self::$instance;

}

public function changePassword($new\_password\_input, $confirm\_password\_input, $user\_id, &$new\_password\_err, &$confirm\_password\_err) {

$new\_password = $confirm\_password = "";

if (empty(trim($new\_password\_input))) {

$new\_password\_err = "Please enter the new password.";

} elseif (strlen(trim($new\_password\_input)) < 6) {

$new\_password\_err = "Password must have at least 6 characters.";

} else {

$new\_password = trim($new\_password\_input);

}

if (empty(trim($confirm\_password\_input))) {

$confirm\_password\_err = "Please confirm the password.";

} else {

$confirm\_password = trim($confirm\_password\_input);

if (empty($new\_password\_err) && ($new\_password !== $confirm\_password)) {

$confirm\_password\_err = "Passwords do not match.";

}

}

if (empty($new\_password\_err) && empty($confirm\_password\_err)) {

$sql = "UPDATE User SET PasswordHash = ? WHERE UserID = ?";

try {

if ($stmt = $this->conn->prepare($sql)) {

$hashed\_password = password\_hash($new\_password, PASSWORD\_DEFAULT);

$stmt->bind\_param("si", $hashed\_password, $user\_id);

if ($stmt->execute()) {

session\_destroy();

header("Location: login.php");

exit;

} else {

throw new Exception("Failed to update the password. Please try again later.");

}

} else {

throw new Exception("Database query failed.");

}

} catch (Exception $e) {

throw new Exception($e->getMessage());

}

}

}

public function authenticateUser($username, $password) {

$sql = "SELECT UserID, Username, PasswordHash FROM User WHERE Username = ?";

if ($stmt = $this->conn->prepare($sql)) {

$stmt->bind\_param("s", $username);

if ($stmt->execute()) {

$stmt->store\_result();

if ($stmt->num\_rows === 1) {

$stmt->bind\_result($user\_id, $db\_username, $hashed\_password);

if ($stmt->fetch() && password\_verify($password, $hashed\_password)) {

return ['user\_id' => $user\_id, 'username' => $db\_username];

}

}

}

$stmt->close();

}

return false;

}

public function validateUserRegistration($username, $email, $password, $confirm\_password) {

$errors = [];

if (empty(trim($username))) {

$errors['username\_err'] = "Please enter a username.";

} else {

$sql = "SELECT UserID FROM User WHERE Username = ?";

if ($stmt = $this->conn->prepare($sql)) {

$stmt->bind\_param("s", $param\_username);

$param\_username = trim($username);

$stmt->execute();

$stmt->store\_result();

if ($stmt->num\_rows > 0) {

$errors['username\_err'] = "This username is already taken.";

}

$stmt->close();

}

}

if (empty(trim($email))) {

$errors['email\_err'] = "Please enter an email address.";

} else {

$errors['email'] = $email;

}

if (empty(trim($password))) {

$errors['password\_err'] = "Please enter a password.";

} elseif (strlen(trim($password)) < 6) {

$errors['password\_err'] = "Password must have at least 6 characters.";

}

if (empty(trim($confirm\_password))) {

$errors['confirm\_password\_err'] = "Please confirm password.";

} elseif ($password !== $confirm\_password) {

$errors['confirm\_password\_err'] = "Password did not match.";

}

return $errors;

}

public function createUser($username, $email, $password) {

$sql = "INSERT INTO User (Username, Email, PasswordHash, CreatedAt) VALUES (?, ?, ?, NOW())";

if ($stmt = $this->conn->prepare($sql)) {

$stmt->bind\_param("sss", $param\_username, $param\_email, $param\_password);

$param\_username = $username;

$param\_email = $email;

$param\_password = password\_hash($password, PASSWORD\_DEFAULT);

return $stmt->execute();

}

return false;

}

}