Maitri Marg, Dillibazar, Kathmandu

BSC(Hons) Ethical Hacking and Cyber Security

COURSEWORK TYPE:

INDIVIDUAL

MODULE:

ST4056CEM Introduction to Web Development and Database Systems Assignment

Submitted by-Dipesh Rana

Submitted to-Dhirendra Bhatta

Table of Contents

17.2. Video Link

Conclusion

References

Appendix

In the **List of Figures** and **List of Tables** sections, you'll provide a numbered list of all figures (like screenshots, diagrams) and tables used in your document, along with their titles and corresponding page numbers.

Example for List of Figures

Figure No.	Figure Title	Page No.
Figure 1	ER Diagram of the Travel Booking System	15
Figure 2	Normalized Tables (3NF)	18
Figure 3	User Interface for Travel Booking	22
Figure 4	Admin Dashboard Screenshot	25

Example for List of Tables

Table No.	Table Title	Page No.
Table 1	Data Dictionary for Users Table	12
Table 2	Data Dictionary for Bookings Table	13
Table 3	SQL Query to Create the Users Table	16
Table 4	Inserted Data for Bookings Table	20

How to Create the List

- **List of Figures**: Include screenshots of important UI pages, diagrams like the ERD, and other visuals related to your project.
- **List of Tables**: Include tables showing data dictionaries, SQL queries, and sample data inserted into your tables.

Make sure each figure and table has a caption in your document, and update the page numbers once the content is finalized.

Introduction

This travel booking web application is designed to make it easier for users to browse and book travel packages online. The system provides a user-friendly platform where customers can register, log in, and explore various travel options like flights and hotels. After selecting a package, users can complete their booking, receive confirmation, and view their booking history anytime.

The application is built with simplicity and security in mind. It features an intuitive interface, making navigation easy even for users with minimal technical knowledge. For the admin side, the system offers tools to manage bookings, customer data, and available travel packages, ensuring smooth backend operations. User security is prioritized throughout the application, especially during registration, login, and transactions. Proper data validation techniques are applied to prevent errors and ensure a seamless experience.

This project aims to deliver a reliable online platform that meets the needs of users looking to plan and book their trips quickly and conveniently. By combining essential features like secure booking, user management, and admin controls, the travel booking web application serves as a complete solution for both customers and administrators.

Aim

The aim of this project is to create a simple and secure travel booking web application that allows users to easily search for and book travel packages online. The system should make it easy for people to plan their trips and manage their bookings from a single platform.

Objectives

1. **Develop a User-Friendly Interface:** Design an easy-to-use website where users can quickly find and book travel packages.
2. **Implement User Registration and Login:** Allow users to create accounts, log in securely, and manage their personal information.
3. **Enable Travel Package Search and Booking:** Provide a feature for users to search for various travel options, such as flights and hotels, and book their chosen packages online.
4. **Create a Booking History Feature:** Allow users to view their past bookings and check the details of their trips at any time.
5. **Build an Admin Panel:** Develop tools for administrators to manage bookings, update travel package details, and oversee user accounts.
6. **Ensure Data Security:** Protect user information and transaction details to ensure a safe and secure booking process.
7. **Test the System Thoroughly:** Check that all features work correctly and fix any issues to ensure the application runs smoothly.

Problem Statement

Planning and booking a trip can be tricky and time-consuming, especially when using complicated or outdated websites. Many travel booking systems are hard to use, and they might not keep personal and payment information safe, which can be a big concern for users.

This project aims to solve these problems by creating a simple and secure travel booking website. The goal is to make it easy for users to search for flights and hotels, book their trips, and manage their bookings all in one place. The website will also include tools for administrators to keep track of bookings and update travel information.

By making the process easier and safer, this travel booking web application will help users plan their trips without stress and ensure their personal information is protected.

Features and Functionalities

Here's a simple explanation of each feature and functionality of the travel booking website:

1. **User Sign-Up and Login:**
    - **Explanation:** Users can create an account to keep track of their personal details. They can log in to access their information and see their past bookings.
2. **Search for Travel Options:**
    - **Explanation:** Users can search for flights and hotels by entering their travel details like destination and dates. This helps them find the best travel deals.
3. **Book Your Trip:**
    - **Explanation:** Once users find the travel options they like, they can book them directly on the website. They'll receive a confirmation to know their booking is successful.
4. **View Booking History:**
    - **Explanation:** Users can look at a list of all their past bookings and current trips. This helps them keep track of where and when they've traveled.
5. **Admin Tools:**
    - **Explanation:** The website has special tools for administrators to manage the travel packages and bookings. This helps them update information and assist users.
6. **Secure Information:**
    - **Explanation:** The website keeps users' personal and payment information safe. This means their details are protected when they sign up, log in, and make payments.

Scope of the Project

Here's a simple explanation of what the travel booking website will include:

1. **Creating User Accounts:**
   - **Explanation:** Users can sign up to make their own accounts and log in to keep track of their personal details and bookings.
2. **Searching for Travel Options:**
   - **Explanation:** Users can look for flights and hotels by entering their travel details like where they want to go and when. This helps them find the best options for their trip.
3. **Booking Travel Packages:**
   - **Explanation:** After finding the right travel options, users can book them through the website. They'll receive a confirmation to know that their booking was successful.
4. **Viewing Booking History:**
   - **Explanation:** Users can see a list of their past trips and current bookings. This helps them remember where they've traveled and what they have planned.
5. **Admin Management:**
   - **Explanation:** The website will have special tools for administrators to manage the travel options, handle bookings, and update user information. This keeps everything running smoothly.
6. **Ensuring Security:**
   - **Explanation:** The website will protect users' personal and payment information to keep it safe from being stolen or misused.
7. **Simple Design:**
   - **Explanation:** The website will be easy to use, with a clear design so users can quickly find what they need and book their trips without any hassle.

Development Methodology

a. Methodology

Explanation: The development methodology is like a plan for building our travel booking website. We'll use a method called the **"Waterfall Methodology"**, which means we follow these steps in order:

1. **Planning:** We first decide what features the website needs and how it should work.
2. **Designing:** We then design how the website will look and what it will do.
3. **Building:** Next, we write the code to create the website based on the design.
4. **Testing:** After building, we check everything to make sure it works properly and fix any issues.
5. **Deployment:** Finally, we launch the website so people can use it.

b. Tools and Technology

Explanation: These are the tools and technologies we'll use to build the website:

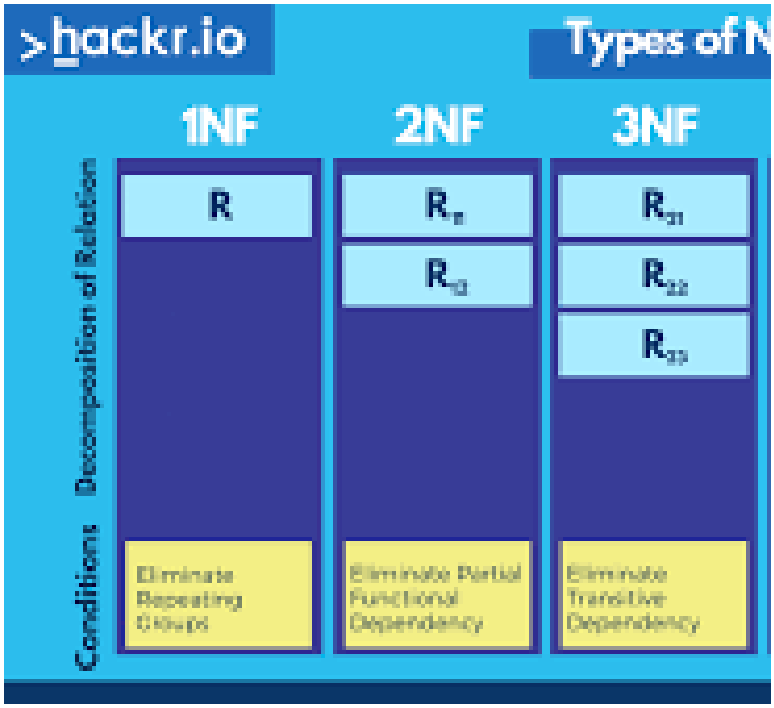
1. **HTML/CSS:** HTML is used to create the structure of the website, while CSS is used to style it and make it look attractive.
2. **PHP:** PHP is a programming language that helps the website interact with users, like handling logins and bookings.
3. **MySQL:** MySQL is a database system where we store important information, such as user details and booking data.
4. **Visual Studio Code:** This is a program where we write and edit the code for the website. It helps us create the site's functions and features.
5. **Web Browser:** A web browser is used to view and test the website to make sure everything works as expected.

Task 1 – Normalization

Explanation: Normalization is a process used to organize a database so that it is efficient and free from unnecessary duplication. It helps to make sure that the database is easy to manage and reduces the chances of errors. Here's a simple breakdown of the steps:

1. **First Normal Form (1NF):**
    - **Definition:** This step involves making sure that each table in the database has a clear and unique way to identify each record, and that each cell in the table contains only one piece of information.
    - **Example:** If we have a table for travel bookings, each cell should contain only one value (like a single travel date, not a list of dates).
2. **Second Normal Form (2NF):**
    - **Definition:** This step builds on the first by ensuring that all the data in the table is dependent on the whole primary key, not just part of it. It means breaking down tables into smaller ones to remove redundant data.
    - **Example:** If a table includes customer information and booking details, we might split it into two tables: one for customer details and one for bookings.
3. **Third Normal Form (3NF):**

- **Definition:** This step ensures that all data in a table is directly related to the primary key and that there are no other dependencies. It means further breaking down tables to ensure that they only contain data related to the primary key.
- **Example:** In our booking system, if a table contains customer info and their address, we might split the address into a separate table if it's used in other parts of the database.



Task 2:

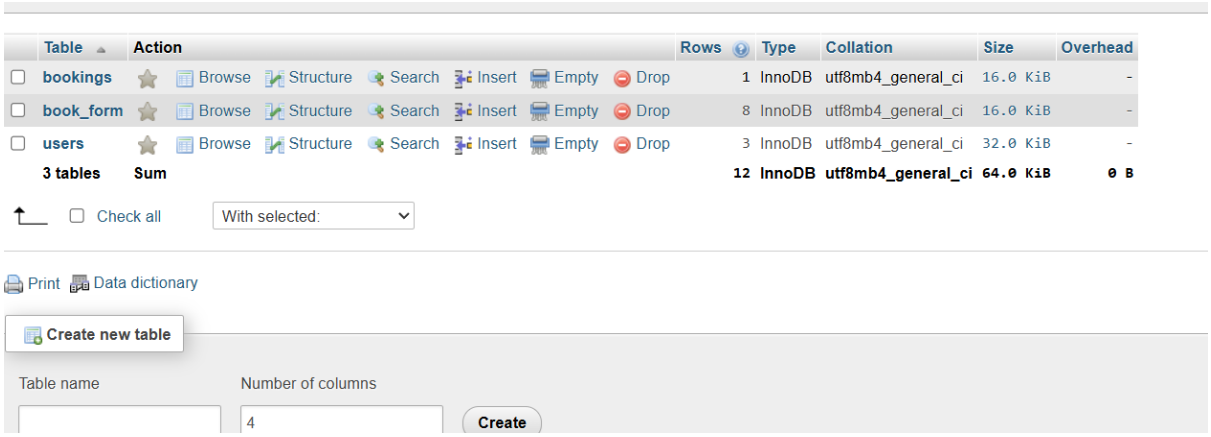
Definition: An ER diagram is a visual representation of how different entities (like tables) in a database are related to each other. It shows the structure and connections between data elements.

Steps to Create an ER Diagram:

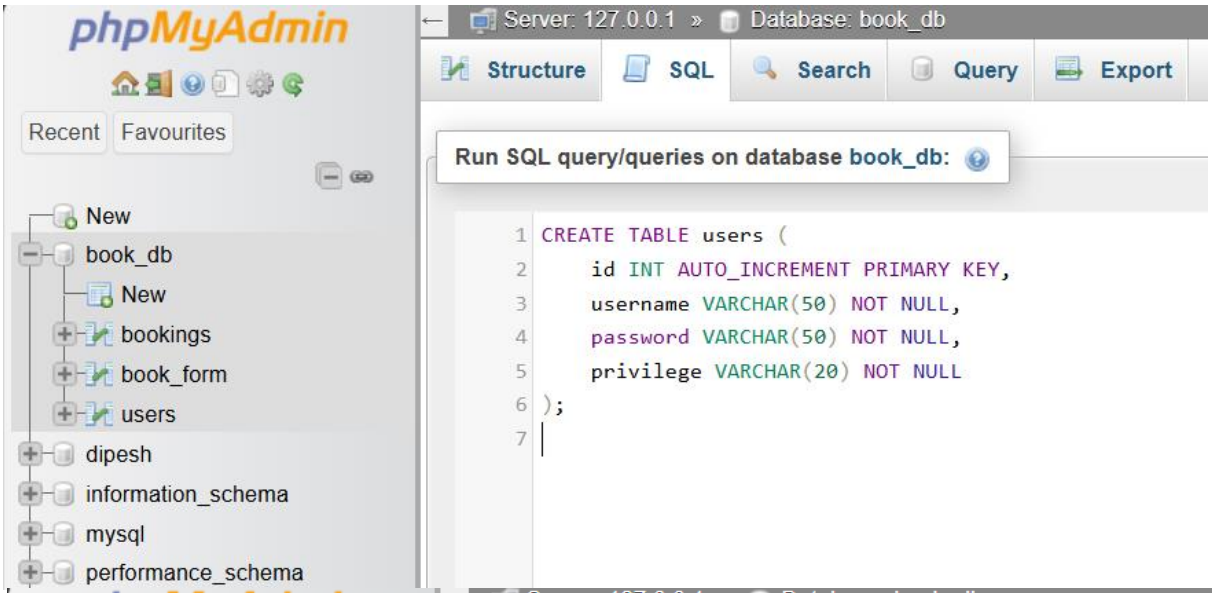
To create an ER diagram, start by identifying the main entities that will be part of your database, such as users, bookings, and travel packages. Next, determine how these entities are related; for example, a user might make multiple bookings, and each booking is linked to a specific travel package. Draw rectangles to represent each entity and connect them with lines to show their relationships. Label each entity and relationship clearly. Use tools like Visual Paradigm to create and refine your ER diagram, making sure it accurately reflects how data interacts and is organized in your database.

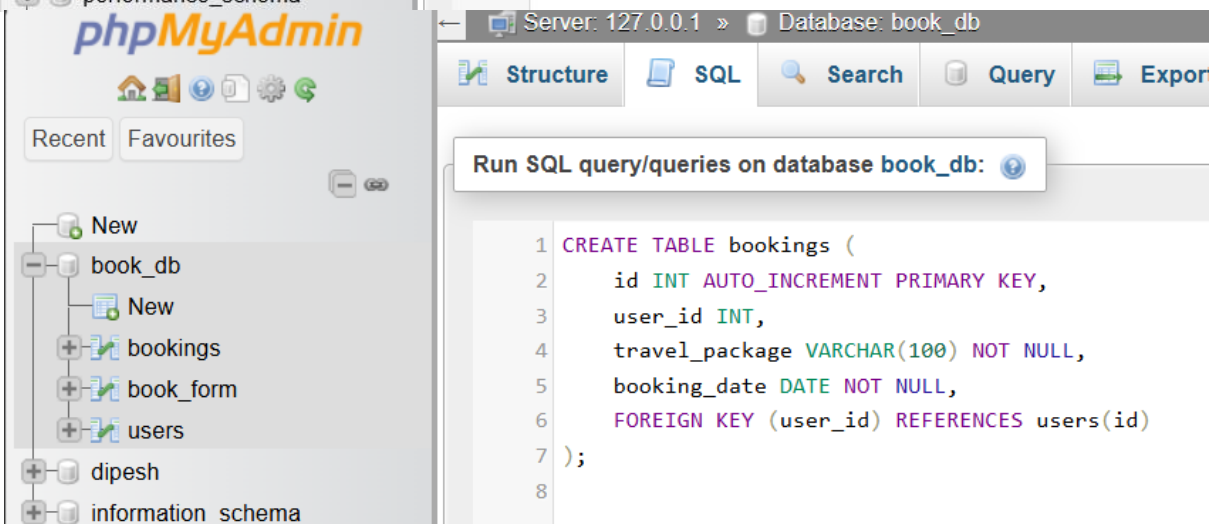
Task 3:

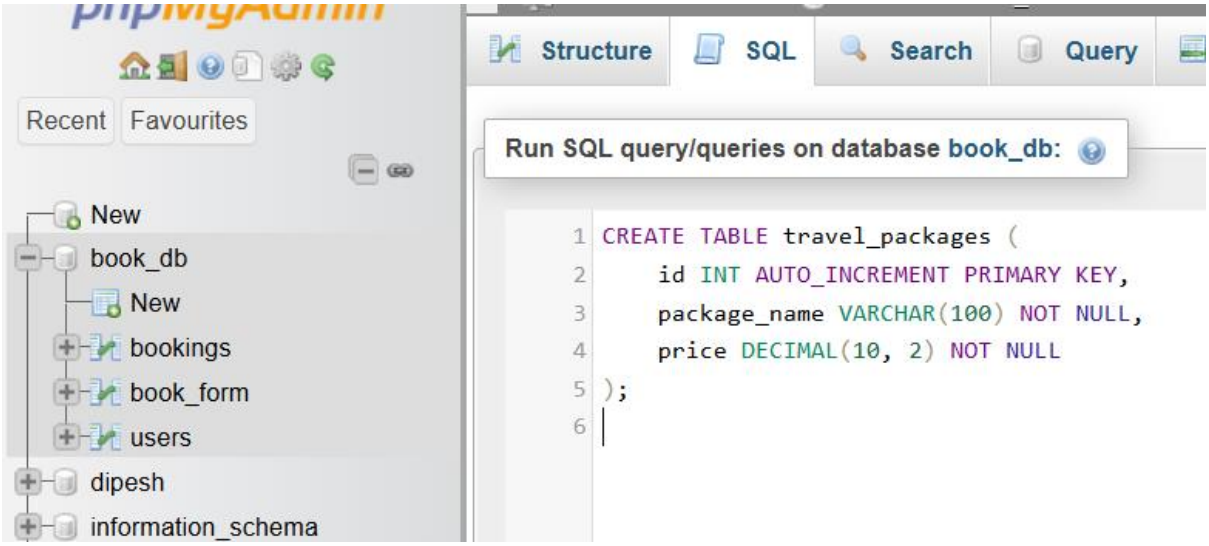
Definition: A data dictionary is like a reference guide that explains what each piece of data in a database means. It shows the names, types, and purposes of all the data elements, helping us understand how the information is organized.



Task 4:






phpMyAdmin
Recent
Favourites
New
book_db
New
bookings
book_form
users
dipesh
information_schema
Structure
SQL
Search
Query
Run SQL query/queries on database book_db:
1 CREATE TABLE travel_packages (
2 id INT AUTO_INCREMENT PRIMARY KEY,
3 package_name VARCHAR(100) NOT NULL,
4 price DECIMAL(10, 2) NOT NULL
5);
6

Task 6 – Reflection

Understanding the Requirements:

For our travel booking website, we needed a way to keep track of:

1. **User Information:** This includes details like usernames, passwords, and what kind of access they have.
2. **Travel Packages:** We had to store information about different travel options, like vacation packages and their prices.
3. **Bookings:** We needed to connect users to the trips they book and record the details of these bookings.

Designing the Database:

1. **Organizing Data:**
    - **Normalization:** We split the data into different tables to make sure we don't store the same information more than once. This makes the database easier to manage and more efficient.
2. **Creating Tables:**
    - **Users Table:** This table holds information about each user, like their username and password. Each user has a unique ID.
    - **Travel Packages Table:** This table keeps track of all the travel options available, including their names and prices. Each package also has a unique ID.
    - **Bookings Table:** This table connects users with their bookings. It records which user booked which travel package and the date of the booking. We use special keys to link this table to the Users and Travel Packages tables.
3. **Why This Design is Good:**
    - **Primary and Foreign Keys:** We use primary keys to identify each piece of data uniquely, and foreign keys to link related information. For example, we use these keys to connect a user with their bookings and travel packages.
    - **Inserting Data:** We added example data to test if everything works correctly, like linking users to their bookings and travel packages.

Task 7 – Database Technologies

1. Local Databases

Description: Local databases are stored on a single computer or server. They are used to manage data on that specific machine.

Merits:

- **Easy to Set Up:** Simple to install and use on just one computer.
- **Fast Access:** Data is quickly available because it's all on the same machine.
- **Low Cost:** No need for extra equipment or expensive services.

Demerits:

- **Limited Access:** Only the computer where the database is installed can use the data, which makes sharing difficult.
- **Risk of Data Loss:** If the computer breaks or has problems, the data might be lost unless it's regularly backed up.
- **Hard to Expand:** It's tough to handle more data or users if needed.

Examples: SQLite, Microsoft Access

2. Distributed Databases

Description: Distributed databases spread the data across several computers or servers that work together to manage and access the information.

Merits:

- **Scalable:** You can add more computers or servers to handle more data or users.
- **Reliable:** If one computer fails, the others keep working, so there's less risk of losing data.
- **Faster:** Data can be accessed from different places, which can speed things up.

Demerits:

- **Complex Setup:** Setting up and managing multiple computers or servers can be complicated.
- **Data Consistency:** Keeping all the data synchronized across different machines can be tricky.
- **Higher Cost:** More equipment and maintenance mean higher costs.

Examples: Apache Cassandra, Google Bigtable

3. Cloud Databases

Description: Cloud databases are stored on the internet, not on a single computer. You access them through online services provided by companies.

Merits:

- **Accessible Anywhere:** You can get to your data from any place with an internet connection, which is great for working with others.
- **Automatic Backups:** The service often takes care of saving and backing up your data, so you don't have to worry about losing it.
- **Easily Scalable:** It's simple to handle more data or users without needing extra hardware.

Demerits:

- **Needs Internet:** You must have a good internet connection to access the data.
- **Security Concerns:** Storing data online can raise worries about privacy and security.
- **Ongoing Costs:** There are usually regular fees to use cloud services, which can add up.

Coding and Implementation (or Development):

Code of the home.php

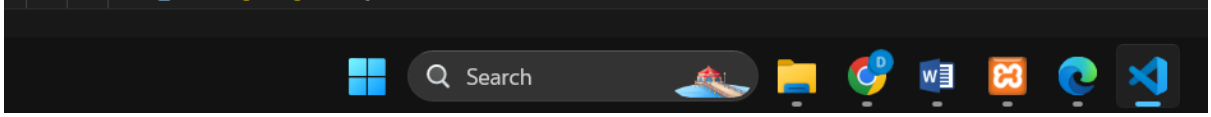
```php
// Check if the user is logged in
if (!isset($_SESSION['username'])) {
    $is_logged_in = false;
} else {
    $is_logged_in = true;
}

// Get the welcome message from the query string
$welcome_message = isset($_GET['welcome']) ? "Welcome, " . htmlspecialchars($_GET['welcome']) . "!" : "Welcome back!";

// Optional: Unset the welcome message after it is displayed to prevent showing it multiple times
if (isset($_SESSION['welcome_message'])) {
    $welcome_message = $_SESSION['welcome_message'];
    unset($_SESSION['welcome_message']);
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Home - Travel Agency</title>
    <!-- swiper css link  -->
    <link rel="stylesheet" href="https://unpkg.com/swiper@7/swiper-bundle.min.css" />
    <!-- font awesome cdn link  -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.4/css/all.min.css">
    <!-- custom css file link  -->
    <link rel="stylesheet" href="css/style.css">
    <style>
        @keyframes fadeIn {
            0% { opacity: 0; }
            100% { opacity: 1; }
        }

        .welcome-message {
            background: #f4f4f4;
            padding: 20px;
            margin: 20px 0;
            border-radius: 10px;
            text-align: center;
```

code of admin panel:

```php
<?php
session_start();

// Check if the user is logged in and is an admin
if (!isset($_SESSION['loggedin']) || $_SESSION['privilege'] !== 'admin') {
    header("Location: login.php");
    exit;
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Admin Panel</title>
    <!-- Bootstrap CSS -->
    <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">
    <!-- Font Awesome CSS -->
    <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.4/css/all.min.css" rel="stylesheet">
    <style>
        body {
            font-family: Arial, sans-serif;
        }
        .navbar {
            margin-bottom: 20px;
        }
        .sidebar {
            background-color: #343a40;
            height: 100vh;
            padding-top: 20px;
        }
        .sidebar a {
            color: #fff;
            display: block;
            padding: 10px 20px;
            text-decoration: none;
        }
        .sidebar a:hover {
            background-color: #495057;
        }
        .content {
            padding: 20px;
        }
        .welcome-message {
```

Code of login:

```php
<?php
session_start(); // Start the session

// Enable error reporting for debugging
ini_set('display_errors', 1);
ini_set('display_startup_errors', 1);
error_reporting(E_ALL);

// Database connection details
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "book_db";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$error_message = ""; // Initialize error message

// Check if form was submitted
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $input_username = trim($_POST['username']);
    $input_password = trim($_POST['password']);

    // Prepare and bind
    if ($stmt = $conn->prepare("SELECT id, username, password, privilege FROM users WHERE username = ?")) {
        $stmt->bind_param("s", $input_username);

        // Execute the statement
        $stmt->execute();

        // Bind the result variables
        $stmt->bind_result($id, $db_username, $db_password, $privilege);
        $stmt->fetch();

        // Check if username exists and verify the password
        if ($db_username && password_verify($input_password, $db_password)) {
            // Password is correct, start session and set user information
            $_SESSION['loggedin'] = true;
            $_SESSION['id'] = $id;
```



Code of signup:

signup.php

```php
<?php
session_start();

// Database connection details
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "book_db";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Handle form submission
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $input_username = trim($_POST['username']);
    $input_password = password_hash(trim($_POST['password']), PASSWORD_DEFAULT);

    // Prepare and bind
    if ($stmt = $conn->prepare("INSERT INTO users (username, password, privilege) VALUES (?, ?, 'user')")) {
        $stmt->bind_param("ss", $input_username, $input_password);

        if ($stmt->execute()) {
            $_SESSION['username'] = $input_username; // Set session variable for username

            // Redirect to home.php with a welcome message
            header("Location: home.php?welcome=" . urlencode($input_username));
            exit;
        } else {
            $message = "Error signing up. Please try again.";
        }

        $stmt->close();
    } else {
        $message = "Database error: Could not prepare statement.";
    }
}

$conn->close();
```

Code of book.php.

```php
<?php
session_start();

// Database connection details
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "book_db"; // Keep the database name as 'book_db'

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Handle form submission
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $name = $conn->real_escape_string($_POST['name']);
    $email = $conn->real_escape_string($_POST['email']);
    $phone = $conn->real_escape_string($_POST['phone']);
    $address = $conn->real_escape_string($_POST['address']);
    $location = $conn->real_escape_string($_POST['location']);
    $guests = $conn->real_escape_string($_POST['guests']);
    $arrivals = $conn->real_escape_string($_POST['arrivals']);
    $leaving = $conn->real_escape_string($_POST['leaving']);

    // Insert booking data into the database
    $sql = "INSERT INTO bookings (name, email, phone, address, location, guests, arrivals, leaving)
            VALUES ('$name', '$email', '$phone', '$address', '$location', '$guests', '$arrivals', '$leaving')";

    if ($conn->query($sql) === TRUE) {
        // Redirect to home.php without a success message
        header("Location: home.php");
        exit;
    } else {
        echo "Error: " . $sql . "<br>" . $conn->error;
    }
}

$conn->close();
?>

<!DOCTYPE html>
```

This is package.php.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Travel Agency :: Best Agency</title>

    <!-- swiper css link  -->
    <link rel="stylesheet" href="https://unpkg.com/swiper@7/swiper-bundle.min.css" />

    <!-- font awesome cdn link  -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.4/css/all.min.css">

    <!-- custom css file link  -->
    <link rel="stylesheet" href="css/style.css">

    <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
    <script>
        $(document).ready(function(){
            $(".scroll-top").click(function() {
                $("html, body").animate({
                    scrollTop: 0
                }, "slow");
                return false;
            });
        });
    </script>

    <style>
        .signup-btn {
            display: inline-block;
            padding: 10px 20px;
            background: #ff6347;
            color: white;
            text-decoration: none;
            border-radius: 5px;
            font-size: 1rem;
            margin-left: 15px;
            transition: background 0.3s ease;
        }

        .signup-btn:hover {
            background: #ff4500;
        }
```