# Car Rental System

# Login:

This page is used for user authentication. It first establishes a database connection and retrieves the email and password from the POST request. It then prepares a SQL statement to select a user based on the provided email and password (either as plain text or hashed). If a user is found, their information is stored in the session, including their role (admin or user). If the user is an admin, they are redirected to the admin dashboard (admin.php); otherwise, they are redirected to the user dashboard (user.php). If no user is found, an error message is passed as a URL parameter to the login page (Login.php) for display.

This page allows users to log in to the system. It includes a form with fields for email and password. On form submission, the data is sent to Result/result.php for processing. The page also includes links for forgot password functionality and account creation (Registration/Email.php). If an error message is received as a URL parameter (e.g., from a failed login attempt), it is displayed in an alert and then removed from the URL to prevent it from being displayed again on page refresh.

# Sign up (Register):

This page handles user registration. It first establishes a database connection and retrieves the username, email, password, first name, and last name from the POST request. It then prepares a SQL statement to check if the email already exists in the database. If the email exists, the user is redirected back to the registration page (Email.php) with an error message.

If the email does not exist, a new user is inserted into the database with the provided information. If the insertion is successful, the user is logged in using session variables and redirected to the user dashboard (user.php). If there is an error during insertion, the user is redirected back to the registration page with an error message.

Note: Using md5() for password hashing is not recommended for security reasons. It's better to use more secure hashing algorithms such as password\_hash() provided by PHP.

# All Cars:

This page displays a list of available cars based on user input filters. It includes a search form that allows users to filter cars by model, year, location, base rate, and status. Users can also show/hide the search form using a button. The page includes a table to display the filtered cars with columns for model, year, plate ID, base rate, location name, status, and an action column for reserving cars.

The PHP code at the bottom of the page handles the form submission and constructs a WHERE clause based on the user's input. It then fetches data from the database (Cars table joined with Locations table) using the constructed WHERE clause to filter the results. The fetched data is displayed in the table, and the status column is colored based on the status of the car (green for available, blue for rented, red for out of service). Additionally, a reservation button is displayed for available cars, allowing users to reserve a car.

Overall, this page provides a user-friendly interface for browsing and reserving available cars from a car dealership.

# Reserve Car:

This page allows users to reserve a car by providing pick-up and drop-off dates, drop-off location, and payment method. The page displays details of the selected car (model, year, plate ID, base rate, location, and status) based on the data passed in the URL parameters ($\_GET).

The PHP code at the beginning of the page checks if all required parameters are set in the URL. If not, it displays an error message. If the parameters are set, it displays the car details and retrieves the CarID and LocationID from the database based on the plate ID of the car.

The page includes a form for users to input their reservation details. It calculates the total amount based on the selected dates and the base rate of the car. The drop-off location and payment method are selected from dropdown menus. The form is submitted to reserve\_submit.php for processing.

A JavaScript function is used to calculate the total amount dynamically based on the selected dates and the base rate. Another function is used to validate the form before submission, ensuring that valid dates are selected and the total amount is calculated.

Overall, this page provides a user-friendly interface for reserving a car from the car dealership.

# Payment:

This page displays a list of cars that have been reserved by the current user, along with their payment history. Here's a breakdown of its functionality:

User Authentication Check:

The page starts by checking if the user is logged in. If not, it redirects them to the login page.

Fetching Reserved Cars:

It queries the database to retrieve information about cars reserved by the current user.

The SQL query selects the model, year, plate ID, and base rate of the reserved cars, ordered by the start date of the reservations.

If reserved cars are found, it displays them in a table with columns for model, year, plate ID, base rate, and an action button to return the car.

Fetching Payment History:

It queries the database to retrieve the payment history of the current user.

The SQL query selects the amount, payment date, and payment method of the payments made by the user for their reservations.

If payment records are found, it displays them in a separate table with columns for amount, payment date, and payment method.

Handling No Records Found:

If no reserved cars or payment records are found, appropriate messages are displayed in the respective tables.

Closing Database Connection:

Finally, the database connection is closed to free up resources.

Overall, this page provides users with visibility into their reserved cars and payment history, enhancing their experience with the car dealership platform.

# Return Car:

### Session Check:

It starts by checking if the user is logged in by verifying the existence of the user\_id session variable. If not, it redirects the user to the login page.

### Database Connection:

It includes the database connection file (db\_connect.php) to establish a connection to the database.

### Handling Plate ID Parameter:

It checks if the plateID parameter is set in the URL (via GET method). The plateID is used to identify the specific car that the user wants to return.

### Updating Reservation Status:

If the plateID parameter is set, it constructs and executes an SQL query to update the reservation status of the car associated with the provided plate ID to "Ended". This marks the end of the reservation for that car.

### Updating Car Status:

It then updates the status of the car with the provided plate ID to "Available", indicating that the car is now available for new reservations.

### Redirecting:

After updating the reservation and car statuses, it redirects the user back to the payments.php page, where they can view their payment history and any remaining reserved cars.

### Error Handling:

If no plateID parameter is provided in the URL, it redirects the user back to the payments.php page to prevent any unexpected behavior.

Overall, this script ensures that when a user returns a reserved car, the appropriate database records are updated to reflect the change in status, and the user is redirected to the relevant page afterward.  
  
  
Dashboard:

Session Management: The code starts by checking if the user is logged in as an admin. It verifies the session variables to ensure the user is authenticated. If not, it redirects them to the login page.

Logout Functionality: If the user clicks on the logout link, the session variables are unset and destroyed, and the user is redirected to the landing page.

Dashboard Navigation: The sidebar provides links to different sections of the admin panel, including Dashboard, Cars, Users, Reservations, Search, and Logout.

Report Generation: The admin panel allows the admin to generate various reports:

All Reservations within a Specified Period: Admin can input a start date and an end date to retrieve all reservations made within that time frame.

All Reservations of a Car within a Specified Period: Admin can input a car ID, a start date, and an end date to retrieve all reservations made for that specific car within the specified time frame.

Status of All Cars on a Specific Day: Admin can input a date to view the status of all cars on that particular day, whether they are available or reserved.

All Reservations of a Specific Customer: Admin can input a customer ID to view all reservations made by that specific customer.

Daily Payments within a Specific Period: Admin can input a start date and an end date to view all payments made within that time frame.

Data Retrieval Functions: Several PHP functions are defined to interact with the database and retrieve the required data based on the admin's input.

Presentation of Reports: The reports are displayed on the page in an organized manner, allowing the admin to easily view the relevant information.

Refresh Prevention: JavaScript is used to prevent the page from being refreshed after form submission, ensuring that the reports stay visible to the admin without interruption.

Overall, this admin panel provides comprehensive functionality for managing car reservations, viewing user data, and generating various reports to monitor the system's activity.

# Car:

### Login Functionality:

* The session starts, and it checks if the user is logged in as an admin.
* If the user is not logged in as an admin, they are redirected to the user page.
* It provides a logout mechanism that destroys the session and redirects the user to the landing page.

### Admin Panel Functionality:

* Displays a sidebar with navigation links for Dashboard, Cars, Users, Reservations, Search, and Logout.
* The main content area displays a list of all cars with options to add, update, and delete cars.
* Allows admins to add a new car by redirecting them to the add car page.
* Retrieves car data from the database, including CarID, Model, Year, Plate ID, Status, Base Rate, and Location Name.
* Displays the status of each car using badges with different colors (green for available, yellow for rented, red for out of service).
* Provides update and delete buttons for each car, allowing admins to edit or remove cars from the system.
* Uses a modal for confirming car deletion, preventing accidental deletions.
* Implements JavaScript to handle the deletion process by redirecting to a delete car script with the car ID as a parameter.

Overall, the admin panel provides a user-friendly interface for admins to manage cars, view user data, and perform various actions related to car reservations.

# User:

* Add New User: Links to a page for adding a new user.
* Update User: Links to a page for updating a user's information.
* Delete User: Deletes a user from the database.

# Reservation:

* Validation: Uses JavaScript to validate the form before submission, ensuring that valid dates are selected and the total amount is calculated.
* Data Retrieval: Retrieves car details (model, year, plate ID, base rate, location, and status) from the query string ($\_GET variables) and displays them on the page.
* Database Query: Retrieves the CarID and LocationID from the database based on the PlateID of the selected car.
* Location Dropdown: Populates the drop-off location dropdown with options retrieved from the database.
* Total Amount Calculation: Calculates the total amount based on the selected dates and the car's base rate.
* Submission: Posts the form data to reserve\_submit.php for processing the reservation.

# Search:

The search logic in this PHP script allows the user to perform an advanced search based on different criteria (car, user, or reservation) selected from a dropdown menu. Here's a breakdown of how it works:

1. Form Setup: The form contains a dropdown (<select>) element where the user can select the search type (car, user, or reservation). Depending on the selected type, different input fields are displayed dynamically using JavaScript.
2. JavaScript Function (showFormFields): This function is triggered when the user selects a search type. It hides all the input field groups (carFields, userFields, reservationFields) and then shows the input fields relevant to the selected search type.
3. Form Submission and Processing: When the user submits the form, the PHP script checks the $\_GET['searchType'] variable to determine the type of search being performed. Based on the selected type, the script constructs a SQL query to retrieve data from the database.
4. SQL Query Building: Depending on the selected search type, the script dynamically adds conditions to the SQL query to filter the search results. For example, if the search type is 'car', it adds conditions to match the input values for car model, year, plate ID, and base rate. Similarly, for 'user', it adds conditions for user's first name, last name, email, and username. For 'reservation', it adds conditions for the start date.
5. Search Results Display: After executing the SQL query, the script checks if any results are returned. If results are found, it displays them in a table format. If no results are found, it displays a message indicating that no results were found.
6. Logout and Access Control: The script also includes logic to check if the user is logged in and has the 'Admin' role. If not, it redirects the user to the user.php page. Additionally, it includes a logout functionality that destroys the session and redirects the user to the landing.php page.