BIRZEIT UNIVERSITY

Computer Science Department

Web Application and Technologies (COMP 334)

Second Semester 2023/2024

Project: PHP Script, CSS, and HTML Forms

The due date is 22/06/2024; see the submission instructions below.

Important Notes

- This is an individual task.
- You should not, in any case, use any external web development tool. Any try to use such a development tool will get a ZERO mark.
- Submission is only through the domain that we created for you.
- Since you are submitting to your domain at CSHost, it is your responsibility to ensure that your scripts, database schema, etc., will work properly on the CSHost server. Make sure to upload your scripts and other development files before the submission date so you have time to test and fix any pop-up issues.
- You must define your database connection details in a separate file called “db.php.inc”; it should define the required variables to create a connection and a PDO object. The database connection must be of type **PDO**.
- You should always use ONLY prepared statements with named binding parameters for any database SQL.
- Use only external CSS files; NO embedded or inline CSS is allowed. You must submit the **final style rules**, and all the classes must be used within the HTML pages, so all CSS files must be cleaned from all temporary and not-in-use classes.
- Make sure you use the CSS rules given to you in the **lecture** ONLY! You are not allowed to use frameworks such as Bootstrap or others.

PROJECT DESCRIPTION:

Birzeit Car Rental Agency (BCAR) rents different cars to its customers. Feel free to give your agency a name and logo, but make sure it is fiction. You have been asked to develop a website to help BCAR run its daily business activities. The potential customers visit the BCAR website to search for a car to rent, to rent a car, and to manage their reservations. The following functionality should be provided by the site:

The following functions are provided to all users.

- Search for a car to rent.
- Customer Registration to enable customers to sign up and create an e-account.

The following functions are provided to the signed-up Customer:

- Rent a car.
- Return a car.
- View rented cars.
- View/update customer profile information.

The following functions are provided to the Manager:

- Add a car.
- Return a car
- Cars Inquire.
- Can add a new location.

Car Search

The system should allow the user (any site visitors, whether they have an account or not) to do a filter search. So, the search functionality should be available for all site visitors.

The search can be for a car based on the renting period (from date to date), car type, pick-up location, and price range (min, max). If the fields are left empty, the system should display all the available cars based on default values: renting period (from current system date for three days), car type sedan, pick up location is Birzeit, and price range (200, 1000).

The search result table Specification:

1. The returned list should be displayed within a table. The first column should contain a checked box that the user could use to create a shortlist. So, the header of the first column should contain a button. When clicked, the table should be updated to display only the checked items.
2. The second column should be the price per day.
3. The third column should be car-type.
4. The fourth column should be fuel type (petrol, diesel, electric, hybrid). You should use a CSS class to distinguish the presentation of the cars in the table based on fuel type. *So, you must create a CSS class for each fuel type and write CSS rules to change the background and font colors.*
5. The fifth column should display a car photo.

6. The sixth should be an action. When clicked, a rent button will display a detailed car page, as specified below.

7. The column headers should be clickable so that the data in the table will be sorted according to the column when the user clicks on it, and cookies must be used to remember the user's preference in sorting data.

Car Details Page specification:

1. Car details should include photos of the car on the left and the car description on the right. The following information about the car should be displayed as unordered list to the customer: car reference number, car model, car type, car make, registration year, color, brief description, price per day, the capacity of people, the capacity of suitcases, total price for the renting period, fuel type (petrol, diesel, electric, hybrid), the average consumption of petroleum per 100 kilometers, horsepower, length, width, and gear type (manual or automatic), and any conditions or restriction.
2. A Rent-a-Car button should appear below the car details box. It starts the car-renting process. The car reference number is sent to a PHP script to rent a car by clicking on the button, as described in the Rent-a-Car functionality specified below.
3. On the right side of the page, you can provide some marking information, for instance, how much this car is enjoyable to drive and if there is any discount for a long period.
4. To arrange the layout of the car details page, you should use the positioning, floating, flex, or grid display CSS properties; using an HTML table is **not** acceptable.

Add A Car

Only the Manager can enter a new car or update its details. The system should allow the manager to enter a new car by providing the following details: car model, car make that should be selected from a pre-defined list, {BMW, VW,Volvo, etc} and displayed in a combo box to the user, car type that should be selected from a pre-defined list, {Van, Min-Van, state, sedan, SUV, etc} and displayed in a combo box to the user, registration year, brief description, price per day, capacity (people, suitcases), colors, fuel type (petrol, diesel, electric, hybrid), average petroleum consumption per 100 kilometers, horsepower, length, width, and some photos (at least three) of the car, plate number, and any conditions or restrictions. You need to add a button to allow the manager to upload photos. You must upload at least three pictures for each car. You can save the file name in the database. The filename should be like that carIDimgSequenceNo; for instance, if you have a car, its ID is 12345, and then the first image file will be car12345img1.jpeg. The file itself could be saved in a folder named carsImages. *The following file formats are accepted: jpeg, png, and jpg, so you must check for the accepted file type.*

Car ID is a 10-digit number generated by the system. When the manager completes entering the car details, the system must display a confirmation message that confirms the car has been successfully added to the database and displays the car ID.

Rent a Car:

Car rent is a three-step process; you must use *session* to implement this functionality. The process steps are described as follows:

1. Customers can rent a car by clicking the rent button; only logged-in customers can initiate this functionality, so the system should check if the customer is logged in. Otherwise, the customer should be redirected to the login page, and then upon successful login to the system, the customer should be returned to the renting form to continue where she/he was left. The form should have (*loaded from the search step*) the car reference number, model, description, renting period (pick-up date and time, return date and time), pickup location, and the total rent amount. In this step, the customer can ask for some special requirements for an **extra cost**, such as a return to a different location (you should display the available return locations), baby seats, and insurance, and you can add more options. Then, the form is submitted for the next step. The script stores the data in a session until the last step of the renting process.
2. The total amount is re-calculated based on the updated input from the customer. The script should display a rent invoice, including the invoice date and the customer information: customer ID, name, address, and telephone number. In addition to rent details, include car details, car model, type, fuel type, pick up date, time and location, return date, time and location, additional requirements such as insurance, baby seat, etc., and the total amount. The site asks the customer to enter the payment details, which are the credit card details: number, expiration date, holder name, and bank-issued. A credit card number should consist of 9 digits, and the expiration date should be validated. Also, the customer should be able to choose the credit card type {Visa, Master Card, etc} using a radio button. Then, the customer is asked to confirm the rent by checking a box to accept the contract terms and conditions, entering his/her name and date. Finally, the customer confirms the rent by clicking the confirm rent button, which moves to the next step.
3. The Confirm Rent button will send the complete form to the script on the server, which updates the database and responds with the confirmation message. The confirmation message should thank the customer and inform him/her that the car has been successfully rented, as well as inform the customer of the invoice ID for future reference. The invoice ID is a 10-digit number generated by the system.

Add A New Location

Only the manager can add a new location where the customer can pick up or drop off (return) a car. The location should have the following details: location ID, which is a numeric key generated by the system upon successful addition; name, address (property number, street name, city, postal code, country), and telephone number.

Return A Car

Car return is a two-step process, started by the customer and completed by the manager as follows:

- A customer can return a rented car. The System should display a list of active car rents, which are the cars picked up by the customer. The list displayed in the table has the following columns: car reference number, car make, car type, car model, pickup date, return date, and return location. Finally, the action column has a return button that calls a PHP script on the server to update the car state to "returning" and update the pick-up location to become the return location of the rented car.
- The manager clicks on the car return link, which displays all the cars whose state is "returning." The list is the same as the table displayed for the customer with the following differences: the table has an extra column to display the customer name, the return action button displays a car form that is filled with the car details, and the fields are disabled for editing except the following two fields could be changed by the manager: the pickup location and the car status to {available, damaged, repair}.

View Rented Car.

Customers should be able to view a list of rented cars: active, current, or past. The list should be as follows:

- The customer list should be displayed in a table as follows: the list should be sorted to the newest rent at the top, and the first column should contain the invoice ID, invoice date, car type, car model, pick-up date, and location, as well as the return date and location. *CSS classes should be defined and used to change the background color and fonts to distinguish the car rental status in the table clearly using appropriate classes. Three rental statuses are defined:* ***future****, that is, a car has been rented but not been picked up yet;* ***current****, the car has been picked up but not been returned yet; and* ***past****, the car had been picked up and was returned.*

Cars Inquire

- The manager should be able to make a filter search to inquire about the cars. The search can be based on the following criteria: the available cars for a certain period { from date to date}, available for rent in a certain pick-up location, all cars that will be returned on a certain day, return to a certain location, all cars in repair, all cars in damage. The search can combine zero or many search options; for example, a manager can search for all available cars on 06/06/2024 on pick up location Ramallah. If no search option is selected, display all available cars for a week from the current date. The result should be displayed in a table with the car ID, type, model, description, photo, fuel type, and status.

view/update customer profile

Only logged-in customers can view or update their profiles. When the customer clicks on the profile link, the customer details are loaded onto a form; the customer can update all his/her information except the customer ID is read-only.

Customer Registration:

For a customer to rent a car, the customer should have an account in the system. So, the system should allow a customer to sign up and create an e-account. Customer sign-up is a three steps process described as follow:

a) The customer fills out the registration form by providing the following information. Customer info. (all the information is required.)

- Name
- Address, {Flat/House No, Street, City, Country}
- Date of Birth
- ID number {رقم الهوية}
- E-mail address
- Telephone
- Credit card details: The customer should enter his/her Credit card details: number, expiration date, name, and bank issued it.

Once this step is validated, all the above fields should be filled in, and then the customer can proceed to the next step. You must use session management to implement this functionality.

b) The customer can proceed to the next step, which is creating an e-account. Your site should display an E-account form that allows the customer to enter a username, which should be between 6 and 13 characters. The user should also enter the password and password confirmation. The valid password should be between 8 and 12 characters.

c) The last step is the confirmation step, in which the system displays all the information entered by the user in the previous two steps in a form. The form has a confirm button that sends the complete form to the server. Then, the server stores the data in the database and replies to the customer with a confirmation message. The confirmation message should have the following information: the customer ID is a 10-digit number (the system will generate the customer ID) and a link to the login page.

User login clicking the link open login form which asks the user to enter his username and password.

User Logout clicking the link logs the user outside the system and destroys the session.

On the right side of the header section of the website, the following links should always be available:

- User Profile (display customer name as hypertext, click on it, and the customer details will be displayed).
- The link for the shopping basket shows the ongoing car rentals, which are the rents that the customer has started but did not make final confirmation.
- And sign-up/ login logout/ link. Login for returning users; clicking the link opens the login form. Sign-up to allow new customers to create an account register as described in user registration above. The user logs in to the system, and the user menu is generated based on the user type, which could be a customer or a manger.

CSS Requirements:

All your web pages and CSS code must be validated correctly. All your formatting should be done using CSS, using the appropriate CSS selector, and all CSS rules must be stored in one external CSS file. The layout of your pages should be similar to the layout shown in Figure 1 below. You must use a flex container to create the page layout.

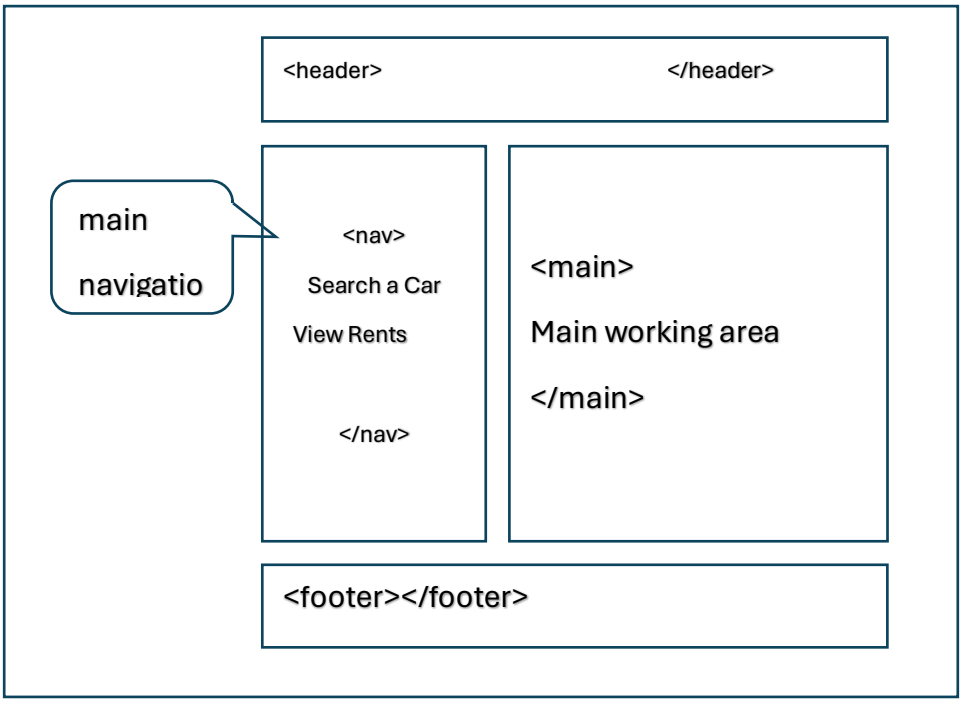

Figure 1: Page Layout

2. Each **page** should contain four sections called **"head," "nav," "main," and "footer."** The header section should extend horizontally across the top of the entire page. The navigation and display sections should run vertically below the header section. The navigation section should be on the left side and 15% of the width of the screen. The display section should consume the remainder of the screen.
3. The **header section** should contain:

    - The agency name
    - logo
    - About us page link.
    - User Profile (user name and username)
    - The shopping basket.
    - And register/ login logout/ link.
4. **Footer section**: smaller logo and copy write text, such as an address, contact email, and telephone number for customer support and a link for the Contact Us page.

5. The **main section** changed from one page to another depending on the task being performed, i.e., all pages have the same header, navigation, and footer sections and differ in the content of the main section. For instance, the search form is displayed when the user selects search, and the table result appears underneath. In the navigation section, the activated link, if the user selects search, the search link is activated and should be displayed in a different style in the navigation section. When the user selects the view order link, it is activated by changing its style in the navigation section and loading the appropriate page to the main section. The header, navigation, and footer sections should not change. *The links in the navigation section should have a style different than other links on the site*. The **main section** should be used to display the results of the links chosen from the navigation section. Upon initial load, this section should be loaded with promotional materials such as new arrivals cars and offers of the day.

css requirements

- All the HTML files should have a consistent presentation format and layout. Page Layout should be like Figure 1, consisting of a navigation section on the left, a header section on the top, a footer section on the button, and the main section on the center.
- Unless specified otherwise, it must use the Flex property to implement the side-by-side layouts.
- Use the correct element types, for instance, nav for the navigation bar, etc. You must *use semantic elements*. General elements such as div are allowed **unless** there is no

available semantic element. Use instead of <i>, instead of <b>, and provide style.

- You must use classes and IDs correctly for your solution. For example, if the user forgets to enter the required field, the system should mark the input field with a special background color and display an appropriate error message near that error input field. So, create a class called “error” to give a style error message that will be displayed to the user when they forget to fill in a mandatory field.
- All required fields should have a special background color in the user forms. So, use CSS classes for this option. You should provide a style for the **required** fields and must be consistent with all forms. All mandatory inputs should be clearly marked. Even *if all the input fields in a form are required, they must be clear for the user*. In other words, you should write appropriate CSS rules that do not depend on the browser to deal with the required fields.
- Change default font properties in 4 locations, headers, table, .., etc
- Change box model properties (border, padding, margin) in four different locations on your website
- You must use the figure element to display the images and change the position properties to display the images and the captions.
- All input text should change the background on the focus.
- All **input labels** should be surrounded by a border in a color different from the text label color.
- The **table header** should have a different background and text color than the table data. Header data should be centralized.
- The table result should have a border and consume 100% of the containing element's width.
- The **table rows** should have a different background for even rows than for odd rows.
- **Links outside** your website should have a different color schema (color for visited, non-visited, and hovering) than those within your website. The difference should be clear enough to be disguised. Also, they should be different in terms of text decoration.
- **Lists:** unordered list, select an image to use instead of the bullet point.
- **Submission button**: use images (metaphors) that indicate the task, provide a style to add a border, change the background color upon click, and change the default text color.
- *You must submit the final style rules, all the classes must be used within the HTML pages. All CSS files must be cleaned from all temporary and not-in-use classes.*

- *Any work taken from the internet must be documented by referencing the source. The code {HTML, CSS, and PHP} taken from the internet or somewhere else should not exceed more than 10% of your work and* ***should not be a complete functionality****.*

Submission instructions

The Deadline for submitting this task is Saturday, 22/06/2024. At 16:00 at the CShost, the server will be closed on Saturday, 22/06/2024, at 17:00. In addition to your submission to the CShost, you must send me your project to ITC by the deadline, which is Saturday, 22/06/2024, at 17:00. What you must submit to ITC: your project files should be compressed, and the compressed file should be extracted to a Folder named stdID (where std is your name and ID is your student ID). Make sure when you extract the compressed file to the localhost and type the following URL (http://localhost/webprojects/stdID) into the browser address bar, the index page of your project will be loaded correctly, and all the links will work fine.
The folder should have the following: -

- All the PHP scripts, HTML files, CSS files, images, ..etc.
- The database Schema is exported as SQL; the file name should be "dbschema_yourNumber.sql."
- Index.html; the index page should have:
  - ✓ Name and Student ID.
  - ✓ A link to the Main/index page of your project
  - ✓ You should create two users, one customer, and one manager, and provide their details for testing: username and password.
  - ✓ You should load your database with some products for testing.