Assignment 6

This assignment assesses your understanding of:

- Using sessions to implement authentication and authorization
- Using MongoDB in a server side application

Submission Instructions

When you are ready to submit, follow these instructions to create your submission:

1. At the top of your server.js file, add the following declaration

```
/********************************************************************************
*  WEB322 – Assignment 06
*
*  I declare that this assignment is my own work and completed based on my
*  current understanding of the course concepts.
*
*  The assignment was completed in accordance with:
*  a. The Seneca's Academic Integrity Policy
*  https://www.senecacollege.ca/about/policies/academic-integrity-policy.html
*
*  b. The academic integrity policies noted in the assessment description
*
*  I did NOT use generative AI tools (ChatGPT, Copilot, etc) to produce the code
*  for this assessment.
*
*  Name: ______________________ Student ID: ______________
*
********************************************************************************/
```

2. Rename your project folder: `A6_FIRSTNAME.` Replace `FIRSTNAME` with your preferred name. Example: `A5_MIKA`

3. Create a zip file of the folder. Name the zip file: `A5_FIRSTNAME.zip`. Replace `FIRSTNAME` with your preferred name. Example: `A5_MIKA.zip`

4. Submit your zip file to the dropbox by the specified due date.

Important Notes

- **NO LATE SUBMISSIONS** for assignments. Late assignment submissions will not be accepted and will receive a grade of zero (0).

- **Submitted assignments must run locally,** ie: start up errors causing the assignment/app to fail on startup will result in a grade of zero (0) for the assignment.

Academic Integrity

Please familiarize yourself with the college's Academic Integrity Policy

This is an individual assessment.

Permitted activities:
- Using course materials or other internet sources to clarify course concepts.
- Using the internet to lookup HTML, CSS, or Javascript syntax

Not permitted:
- Reposting any part of the assessment to online forums or homework help websites
- Contract plagiarism: Purchasing a solution, or completing a solution for compensation
- Sharing or receiving source code, references, or assistance from others

Usage of Artificial Intelligence (AI) tools:
- AI usage to generate solutions is NOT permitted. Examples: ChatGPT, Copilot,the AI tools built into VSCode.
- This assignment should be created based on your current knowledge of the course concepts, NOT based on what an AI knows.

Problem Description

Using the techniques described in class, create a web app that simulates a simple car rental system. The app must consist of the following pages:

1. Login page:	Used to login a user or create a new user account
2. Cars List page:	Displays a list of cars to rent
3. Booking page:	Displays a <form> for the user to book a car

You may use the assignment starter code located in the assignment folder on the course webpage. The starter code contains a sample user interface. You may modify the user interface as needed.

1. Technical Requirements

- The database must be implemented using **MongoDB**.
- Authentication and authorization must be implemented using **express-session**

2. Database

The app's database must consist of a cars and users collection.

Relationships

- A car can be rented by 1 user (1:1 relationship)

Cars collection

- Your server must prepopulate the database with 5 cars of your choice.

- Every car has these properties:

☐ `model`	Example: Tesla Model Y, Honda Civic, etc
☐ `imageUrl`	Url to an image of the car
☐ `returnDate`	A string representing the date the car will be returned. If the car is not rented, then set this to an empty string

Users collection

- Every user has:

☐ email	*string*
☐ password	*string*

3. Implement Pages

The app consists of the following pages

1.	Login page:	Used to login a user or create a new user account
2.	Cars List page:	Displays a list of cars to rent
3.	Booking page:	Displays a <form> for the user to book a car

You are responsible for creating the necessary endpoints to implement these pages and their associated functionality.

Login Page

This is the initial screen of the application.

This page must provide form fields for a user to enter an email and password. Here's how the login form behaves:

Scenario	***Result***
Correct email and password	Navigate user to the **Cars List** screen.
Correct email, but wrong password	Do not proceed. An error message is nice to have, but not required.
Email provided that does not exist in the users collection	1. Create a new User in the database with the provided email and password 2. Navigate the user to the **Cars List** page..

Cars List Page

The page must display:

1. A navigation bar with a logout button. Pressing the logout button will log the user out and return them to the **Login Page**.
2. A list of cars in the database.

1. List of Cars

For each car, display:

- The car's **model** and **image**

- **If the CAR is not rented**, then show a BOOK CAR link. Clicking on the link navigates the user to the **Booking Page**

- **If the car is rented by the currently logged in user**, then show a return date and a RETURN button. Clicking on the RETURN button will:

    a. remove the relationship between the car and the renter (HINT: set the appropriate property to null)
    b. set the return date back to an empty string

- **If the car is rented by someone else**, then show an "Unavailable until xxxx", where xxx is the return date of the car

2. Logout

- Clicking on the Logout link will log the user out of the app and return them to the **Home Page**

Booking Page

- This page is displayed when the user attempts to book a specific car.
- The page must display a form for the user to enter the date they will return the car.
- When the form data is submitted, the server must update the specified car with the logged in user and the provided return date.

4. Authorization

Your app must enforce the following:

Anyone can:	Only logged in users can
● View the Login page	● View the Cars List Page ● View the Booking Page ● Book a car ● Return a car If a non-logged in user attempts to do any of the above operations, then redirect them to the Login page. An error message is nice to have, but not required.

4. User Interface

A sample user interface is provided in the assignment starter code. You may modify the UI to your liking

- The app's user interface must be implemented using EJS templates.

- Pages should be reasonably pretty and styled with Tailwind/DaisyUI. You can use the HTML code from the sample user interface as a starting point for relevant Tailwind/DaisyUI CSS classes.

- A fully responsive design is **not** required.

5. Deployment

Deploy the application to Vercel.

END OF ASSESSMENT