Lab 7

Sessions in Express + MongoDB

# Overview

In this lab, we will implement a session-based system that uses typical cookie-based sessions with the session data stored in a MongoDB database using TTL indexes to automatically clean up session data.

# Database

Create a database called lab7 in your cloud account and create a collection of “UserAccounts”. Create the database in your existing cluster/project.

Each user account should contain the fields:

* Username
* Password – this will be just plain text
* AccountType – either “standard” or “admin”

Create 2 users of each type. The name of the user and the passwords are your choice.

# Session Data

Create a collection called ‘SessionData’ in your mongoDB database. Insert one object like shown in class with the following fields:

* SessionKey – pick any value
* Expiry – pick a time around 2 minutes into the future
* Data – any value or string

Create a TTL index that automatically erases data when the expiry field is more than 0 seconds old. Continue with the next part but check back a few minutes after the expiry date and make sure that your test data vanished. Note that the process of cleaning up expired data might not run instantly and sometimes you have to wait a couple of minutes for it to go. If you get stuck on this, just carry, the rest of the code is more important and you can debug the TTL index later.

One of the most difficult things in this exercise is making the data a correct format. If you use a string for the time, it will not work, it must be a date field.

# Login Screen Handler

Write the handling page for the login page. The handler should work like this:

1. If the username/password is incorrect. Send the user back to the login screen and include some sort of “invalid username/password message”.
2. If the username/password is correct, then you can look at the type of user, create a session, and send the user to either /admin or /standard page.

## To create a session:

* Generate a random session value. Please look online and find out what UUID is and use crypt.randomUUID() to generate one. This will be our “session key”.
* Create a SessionData item with the new session key, expiry time (5 minutes into the future), and put the name of the user and the type of user into the “data” part and store this into the MongoDB collection.
* Send a cookie back to the browser named ‘lab7sessionkey’ with the value of the session key.
* Redirect the user to either the /admin or /standard page.
* You should verify the data in the database as well as the cookie that is sent back to the browser (in the inspect window).

# Complete User Landing Pages

Fix the “User Landing Pages” (/admin and /standard) to check for the presence of a valid session cookie.

I will explain the /admin, but the same thing is done for /standard. Here is the process:

1. Check to see if there is a lab7sessionkey cookie. If not, then send the user to the login page and show an error message.
2. Get the data from the session storage for the key. If there is no session in storage, then send the user out to the login screen.
3. If the type of the user is not “admin” then you send the user to the login page with an appropriate message.
4. If you have made it through the previous items, then the user must be an authorized person for this page. Print a welcome message on the page and list their name such as “Welcome admin user ahmed”. The template already has a placeholder for this.

# Logout

Implement the /logout functionality which should work like this:

1. Remove the session information from MongoDB.
2. Delete the cookie on the browser.
3. Redirect the user to the login page.

# Database Indexes

Create appropriate indexes on your data to make the searches faster. In the persistence.js file please place a comment at the top of the file telling me which collections/attributes you created the collection on.