Authentication and authorisation

Welcome to this lab activity

In this lab activity, you will implement session management in your Node.js application. You will build a login form to authenticate the user again previously registered user credentials. And you will protect the /addbook route so that only authenticated users can access it.

Task 1: Make a copy of your previous lab

You will start with your previous lab code.

1. Make a copy of the folder `24_password_hashing` from your previous lab and call it `25_authentication_and_authorisation`.
2. Open the new folder in Visual Studio Code.
3. Run the code and confirm that it works as expected.

Task 2: Set up session management

4. You will use a module called express-session to make it easy to manage sessions in your web application.
5. Install the **express-session** module by running the following command from a terminal session:

```
npm install express-session
```

For more information related to express-session refer to expressjs/session.

6. Import this module in your `index.js` file, after the other 'require's in your code:

```
var session = require ('express-session')
```

7. Add the following piece of code to your to `index.js` file to enable sessions. Add it before you define the database connection:

```
// Create a session
app.use(session({
    secret: 'my special secret text',
    resave: false,
    saveUninitialized: false,
    cookie: {
        maxAge: 600000
```

```
    }
}))
```

The secret value is used with the session id.

What do you think the secret parameter does? Read:

- Singleton, J. 'One line of code that compromises your server', *martinFowler.com* (2017).

Task 3: Authentication: Login

You will need to provide a mechanism for users to log into your application. Once they are logged in, you can allow them to access the /addbook route.

8. Create a new file, login.ejs, in the views directory. Add the following code, which provides a login form:

```
<!doctype html>
<html>
  <head>
    <title>Login</title>
  </head>
  <body>
    <h1>Login to My Bookshop</h1>
    <form method="POST" action="loggedin">
        <p>Username: <input id="username" type="text" name="username"/></p>
        <p>Password: <input id="password" type="text" name="password"/></p>
        <input type="submit" value="Login" />
      </form>
  </body>
</html>
```

9. Show the login page to the user when they access the /login route by adding the following code to main.js:

```
router.get('/login', function (req, res, next) {
    res.render('login.ejs')
})
```

When a user completes the login form and presses the Login button, you need a route handler to handle the login process. This route handler needs to check the username and password entered against the database. If there is a matching set of credentials in the database, you can allow the user to login and set up the session.

10. Add this code to main.js:

```
router.post('/loggedin', function (req, res, next) {
    // Select the hashed password for the user from the database
    let sqlquery = "SELECT id, hashed_password FROM users WHERE username =?"
    db.query(sqlquery, [req.body.username], (err, result) => {
        if (err) {
            next(err)
        } else {
            // Extract the matching row information
            hashedPassword = result[0].hashed_password
            userId = result[0].id

            // Compare the password supplied with the password retrieved from the db
            bcrypt.compare(req.body.password, hashedPassword, function(err, result) {
                if (err) {
                    next(err)
                }
                else if (result == true) {
                    // Save user session here, when login is successful
                    req.session.userId = req.body.username;

                    res.send("Logged in successfully!")
                }
                else {
                    res.send("Login failed: passwords don't match")
                }
            })
        }
    })
})
```

You also need to provide a logout page. This will destroy the current session.

11. Add logout page handler to main.js:

```
router.get('/logout', redirectLogin, (req,res) => {
    req.session.destroy(err => {
        if (err) {
        return res.redirect('./')
        }
        res.send('You are now logged out');
    })
})
```

Task 4: Authorisation – protect the /addbook route

Now that you have the ability for users to log into your application, you can protect the routes that you only want your logged in users to access. You have the following routes in your application:

/

/about

/search

/register

/list

/addbook

/login

/logout

Which of these should you only allow logged in users to access? Probably not /list, otherwise customers coming to your website won't be able to see the books that you sell. But /addbook should be protected. You don't want the general public to be able to add books to your bookshop! So let's protect this route.

12. Add the following piece of code to your `main.js` file:

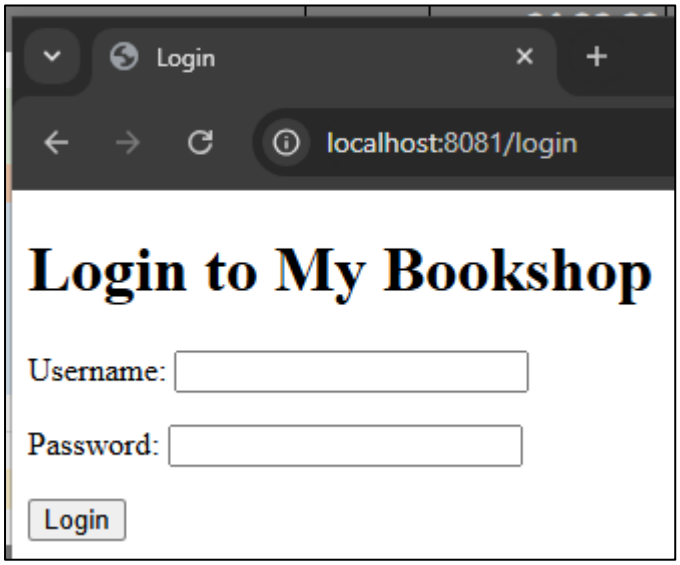
```
const redirectLogin = (req, res, next) => {
    if (!req.session.userId ) {
      res.redirect('./login') // redirect to the login page
    } else {
        next (); // move to the next middleware function
    }
}
```

13. In the /addbook route, add the redirectLogin function to the list of middleware functions, as the first one in the list:

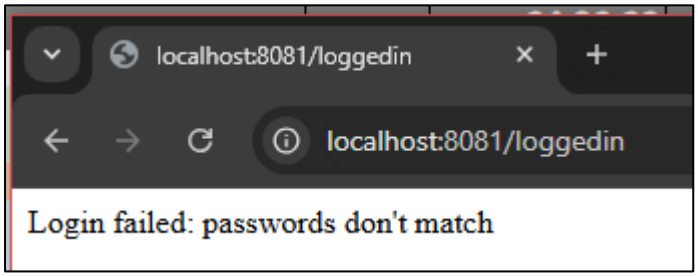
```
router.get("/addbook", redirectLogin, (req,res) => {
    res.render("addbook.ejs");
});
```

Task 5: Test your application

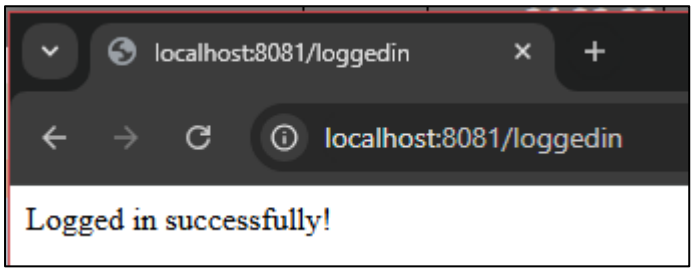
14. Browse to the /addbook route. It should redirect you to the login page:



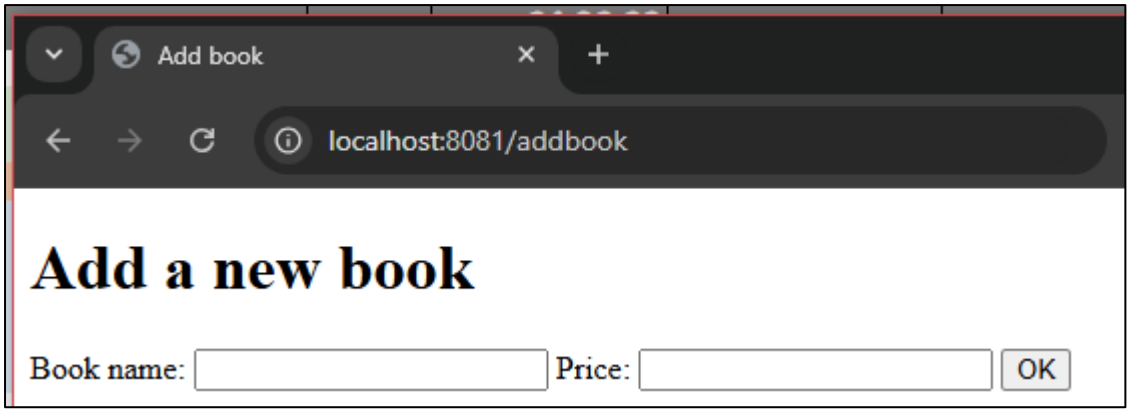
15. Enter an incorrect username and password and confirm you get notified of an incorrect login:



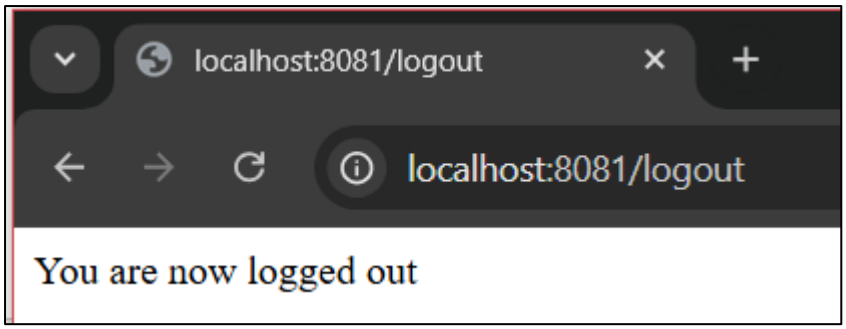
16. Enter a correct username and password and confirm that you get the confirmation:



17. Now browse to the /addbook route again and confirm that you can access it:



18. Browse to the /logout route and confirm you get the logout message:



19. Now check the /addbook route again to confirm that you are prevented from accessing it.

Task 6: Explore further

When tackling these lab activities, it's always good to stretch yourself by doing some research and attempting some changes on your own.

Try logging in with an invalid username. What happens? Your program throws an error! Can you fix it?

Typing in the URLs each time is tiresome! Can you add links to all the pages from the homepage? And links from each page back to the homepage?

End of lab

Congratulations on completing this lab.

You have successfully implemented authentication and authorisation in your web application!

If these links are broken, please let us know via the Student Portal.