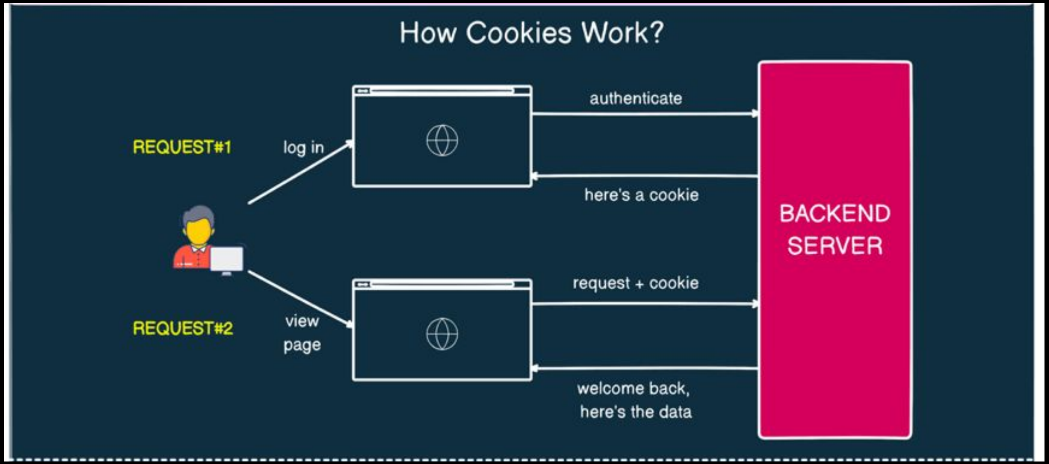
18. Cookies & Sessions

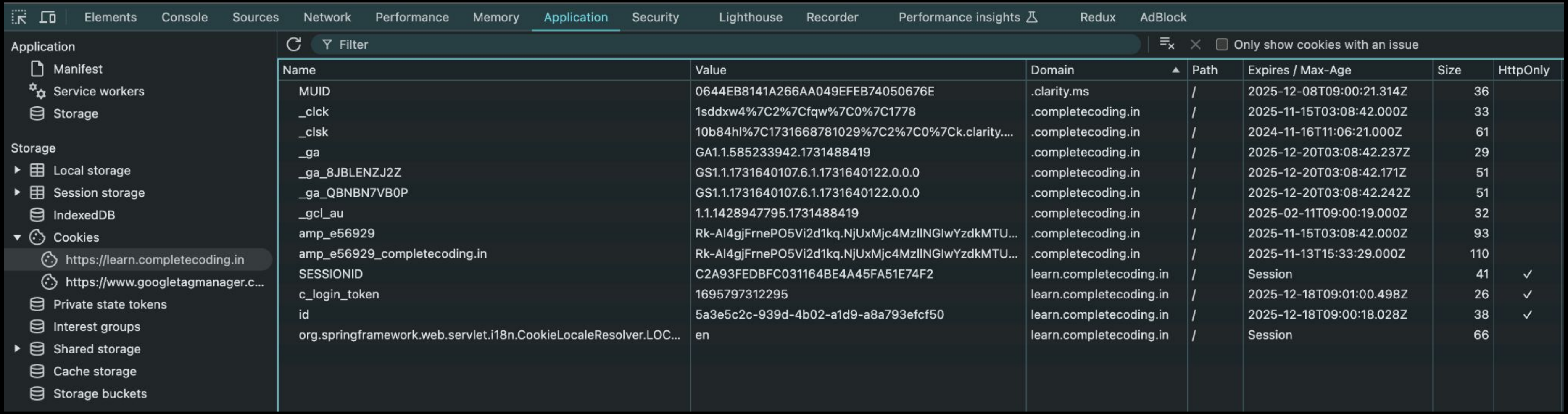
1. What are Cookies
2. Adding Login Functionality
3. Checking Login State
4. Using a Cookie
5. Define the Logout Feature
6. Problem with Cookies
7. What are Sessions
8. Installing Session Package
9. Creating a Sessions
10. Saving Session in DB

18.1 What are Cookies



1. Cookies are small pieces of data stored in the user's browser by server.
2. They help websites remember user information and preferences between page loads or visits.
3. Cookies can manage user sessions and store data for personalized experiences.

18.1 What are Cookies



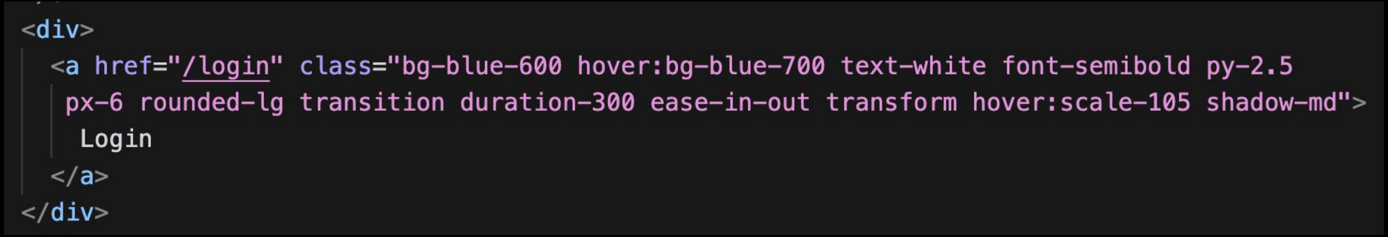
Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly
MUID	0644EB8141A266AA049EFEB74050676E	.clarity.ms	/	2025-12-08T09:00:21.314Z	36	
_clck	1sddxw4%7C2%7Cfqw%7C0%7C1778	.completecoding.in	/	2025-11-15T03:08:42.000Z	33	
_clsk	10b84hl%7C1731668781029%7C2%7C0%7Ck.clarity....	.completecoding.in	/	2024-11-16T11:06:21.000Z	61	
_ga	GA1.1.585233942.1731488419	.completecoding.in	/	2025-12-20T03:08:42.237Z	29	
_ga_8JBLENZJ2Z	GS1.1.1731640107.6.1.1731640122.0.0.0	.completecoding.in	/	2025-12-20T03:08:42.171Z	51	
_ga_QBNBN7VB0P	GS1.1.1731640107.6.1.1731640122.0.0.0	.completecoding.in	/	2025-12-20T03:08:42.242Z	51	
_gcl_au	1.1.1428947795.1731488419	.completecoding.in	/	2025-02-11T09:00:19.000Z	32	
amp_e56929	Rk-Al4gjFrnePO5Vi2d1kq.NjUxMjc4MzlINGIwYzdkMTU...	.completecoding.in	/	2025-11-15T03:08:42.000Z	93	
amp_e56929_completecoding.in	Rk-Al4gjFrnePO5Vi2d1kq.NjUxMjc4MzlINGIwYzdkMTU...	.completecoding.in	/	2025-11-13T15:33:29.000Z	110	
SESSIONID	C2A93FEDBFC031164BE4A45FA51E74F2	learn.completecoding.in	/	Session	41	✓
c_login_token	1695797312295	learn.completecoding.in	/	2025-12-18T09:01:00.498Z	26	✓
id	5a3e5c2c-939d-4b02-a1d9-a8a793efcf50	learn.completecoding.in	/	2025-12-18T09:00:18.028Z	38	✓
org.springframework.web.servlet.i18n.CookieLocaleResolver.LOC...	en	learn.completecoding.in	/	Session	66	

18.2 Adding Login Functionality

1. Add a login button to the nav bar pointing to /login
2. Create a auth router to handle login related routes, register the new router in app.js
3. Create a auth controller to handle GET request for /login and return a UI that does the following:
   a. Accepts email and password
   b. Has a Login button that submits the form to /login with a POST request.
4. Add a post login path in router and handler in controller.
5. Assume the person logged in and redirect them to the home page.

18.2 Adding Login Functionality

1.



```html
<div>
  <a href="/login" class="bg-blue-600 hover:bg-blue-700 text-white font-semibold py-2.5
   px-6 rounded-lg transition duration-300 ease-in-out transform hover:scale-105 shadow-md">
    Login
  </a>
</div>
```

18.2 Adding Login Functionality

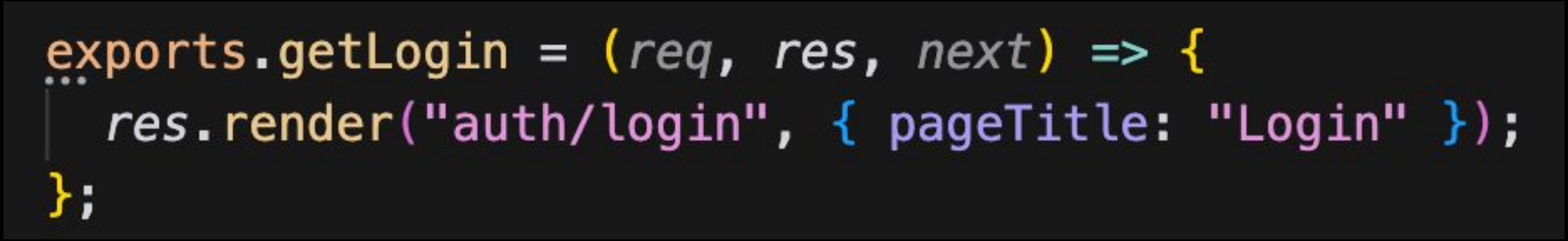
2.

```
const express = require("express");
const authRouter = express.Router();
const authController = require("../controllers/authController");

authRouter.get("/login", authController.getLogin);

exports.authRouter = authRouter;
```

```
app.use("/host", hostRouter);
app.use(authRouter);
```

18.2 Adding Login Functionality

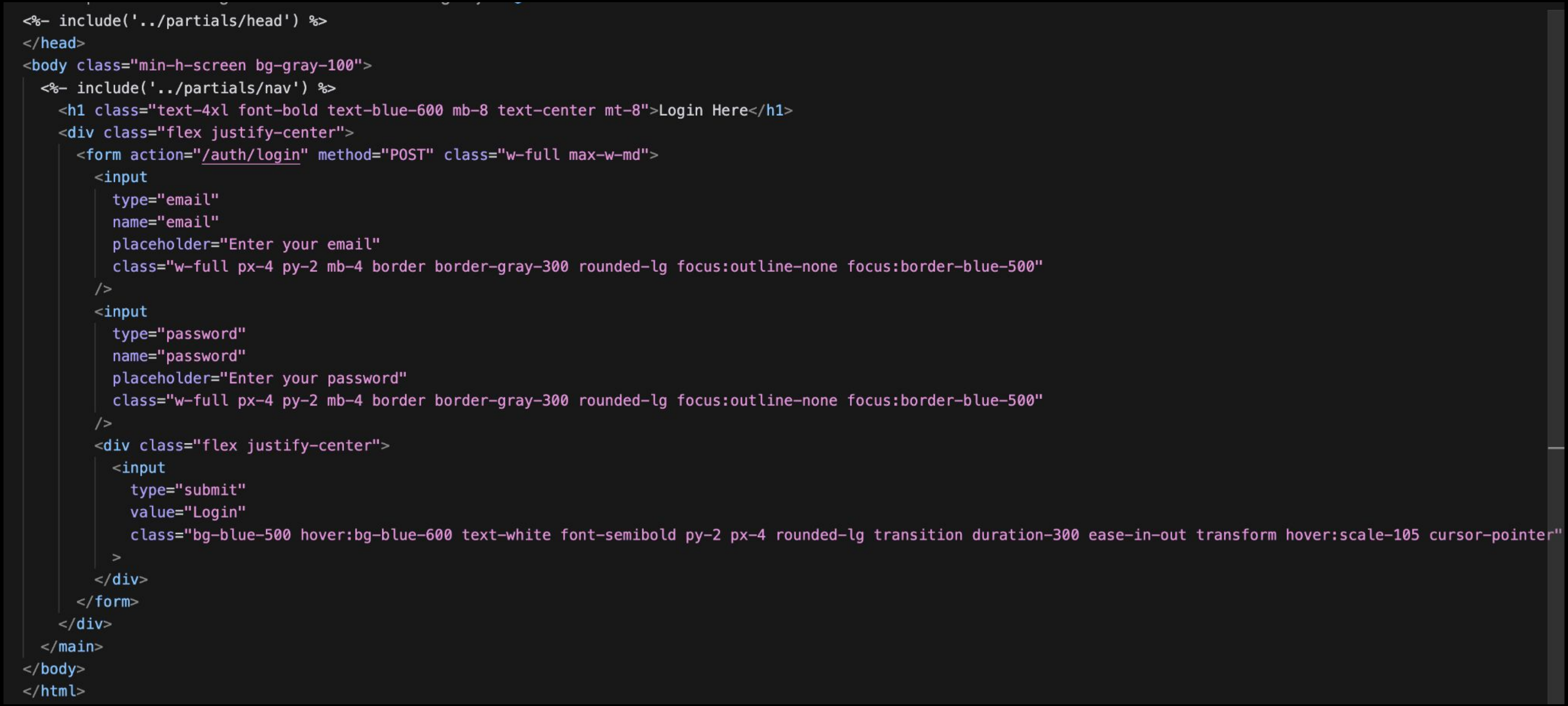
3.



```
exports.getLogin = (req, res, next) => {
  res.render("auth/login", { pageTitle: "Login" });
};
```

18.2 Adding Login Functionality

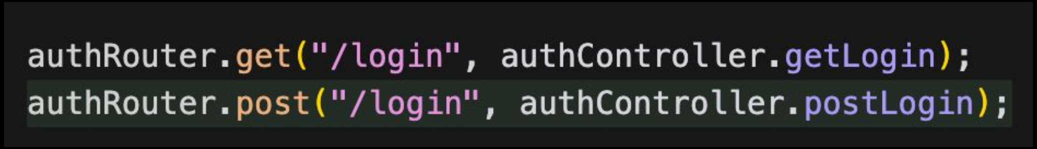
3.



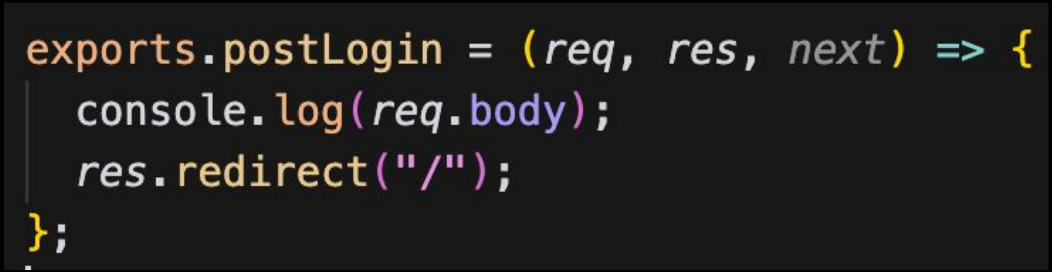
```
<%- include('../partials/head') %>
</head>
<body class="min-h-screen bg-gray-100">
  <%- include('../partials/nav') %>
    <h1 class="text-4xl font-bold text-blue-600 mb-8 text-center mt-8">Login Here</h1>
    <div class="flex justify-center">
      <form action="/auth/login" method="POST" class="w-full max-w-md">
        <input
          type="email"
          name="email"
          placeholder="Enter your email"
          class="w-full px-4 py-2 mb-4 border border-gray-300 rounded-lg focus:outline-none focus:border-blue-500"
        />
        <input
          type="password"
          name="password"
          placeholder="Enter your password"
          class="w-full px-4 py-2 mb-4 border border-gray-300 rounded-lg focus:outline-none focus:border-blue-500"
        />
        <div class="flex justify-center">
          <input
            type="submit"
            value="Login"
            class="bg-blue-500 hover:bg-blue-600 text-white font-semibold py-2 px-4 rounded-lg transition duration-300 ease-in-out transform hover:scale-105 cursor-pointer"
          >
        </div>
      </form>
    </div>
  </main>
</body>
</html>
```

18.2 Adding Login Functionality

4, 5.



```
authRouter.get("/login", authController.getLogin);
authRouter.post("/login", authController.postLogin);
```



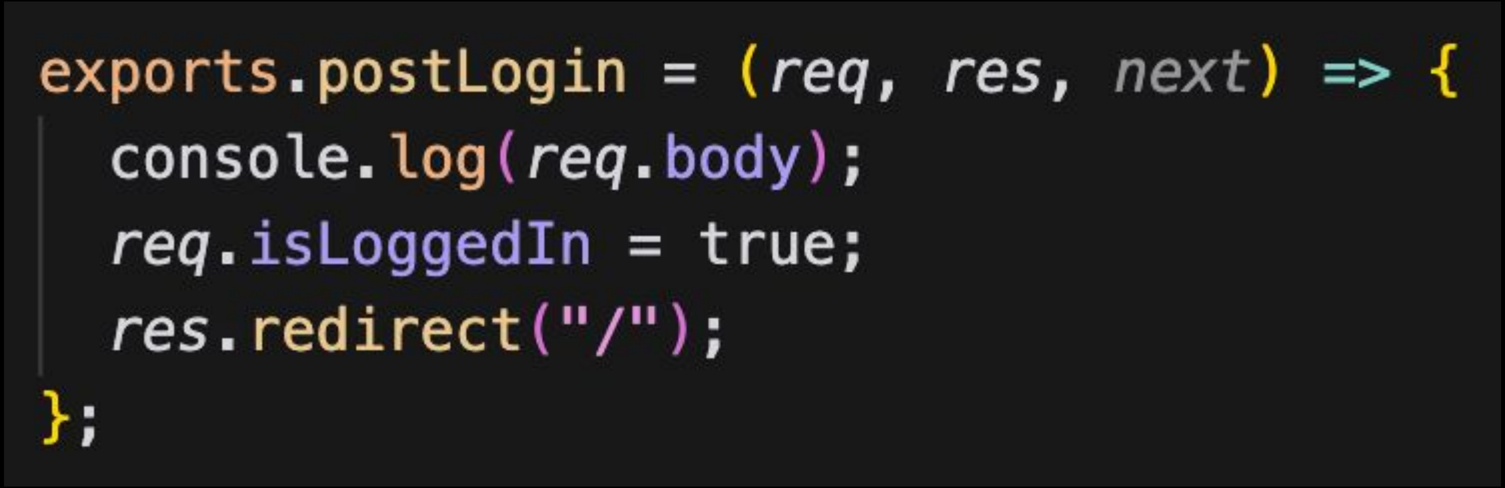
```
exports.postLogin = (req, res, next) => {
  console.log(req.body);
  res.redirect("/");
};
```

18.3 Checking Login State

1. Add a isLoggedIn field in the req object and use it everywhere else.
2. Add a condition in the navigation ejs file that no path other than home and login should be visible until a user has logged in.
3. Fix all the render calls to send the flag
4. Also add a middleware for host routes that if the user is not logged in they should be redirected to the login page.

18.3 Checking Login State

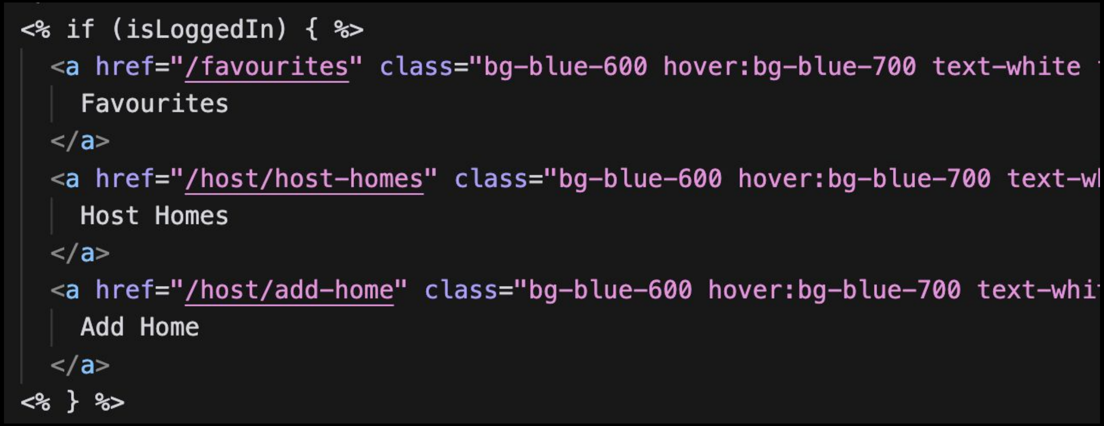
1.



```javascript
exports.postLogin = (req, res, next) => {
  console.log(req.body);
  req.isLoggedIn = true;
  res.redirect("/");
};
```

18.3 Checking Login State

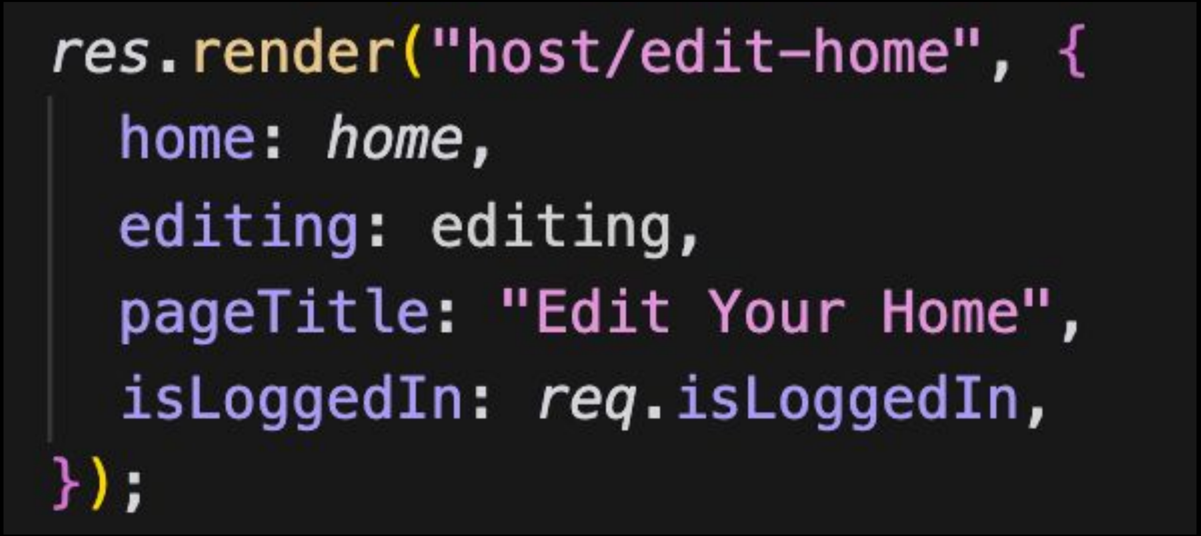
2.



```
<% if (isLoggedIn) { %>
  <a href="/favourites" class="bg-blue-600 hover:bg-blue-700 text-white
    Favourites
  </a>
  <a href="/host/host-homes" class="bg-blue-600 hover:bg-blue-700 text-wh
    Host Homes
  </a>
  <a href="/host/add-home" class="bg-blue-600 hover:bg-blue-700 text-whi
    Add Home
  </a>
<% } %>
```

18.3 Checking Login State

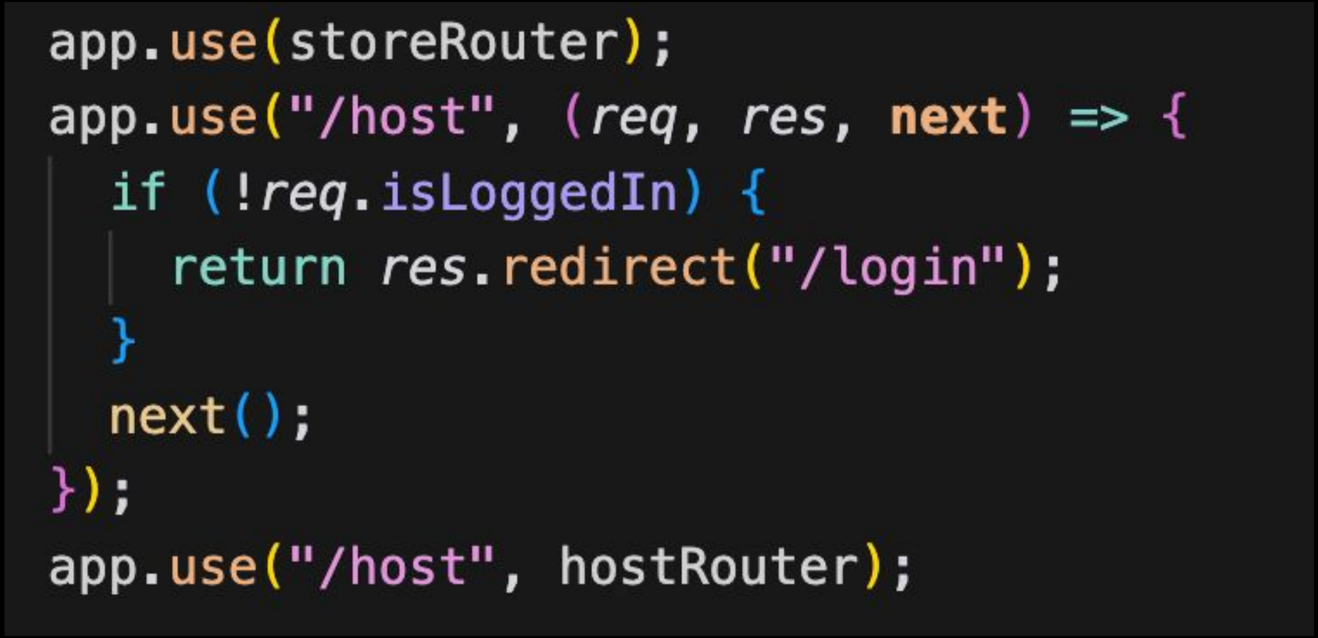
3.



```
res.render("host/edit-home", {
  home: home,
  editing: editing,
  pageTitle: "Edit Your Home",
  isLoggedIn: req.isLoggedIn,
});
```

18.3 Checking Login State

4.



```
app.use(storeRouter);
app.use("/host", (req, res, next) => {
  if (!req.isLoggedIn) {
    return res.redirect("/login");
  }
  next();
});
app.use("/host", hostRouter);
```

18.4 Using a Cookie

1. Understand why a global variable would not work.
2. Set a cookie on successful login. See it in storage, also on the next request.
3. Read the cookie using syntax and Define a middleware to set this value to the request object.

```
console.log(req.get('Cookie').split('=')[1]);
```

4. Change the Cookie from the browser and see the result.

18.4 Using a Cookie

2.

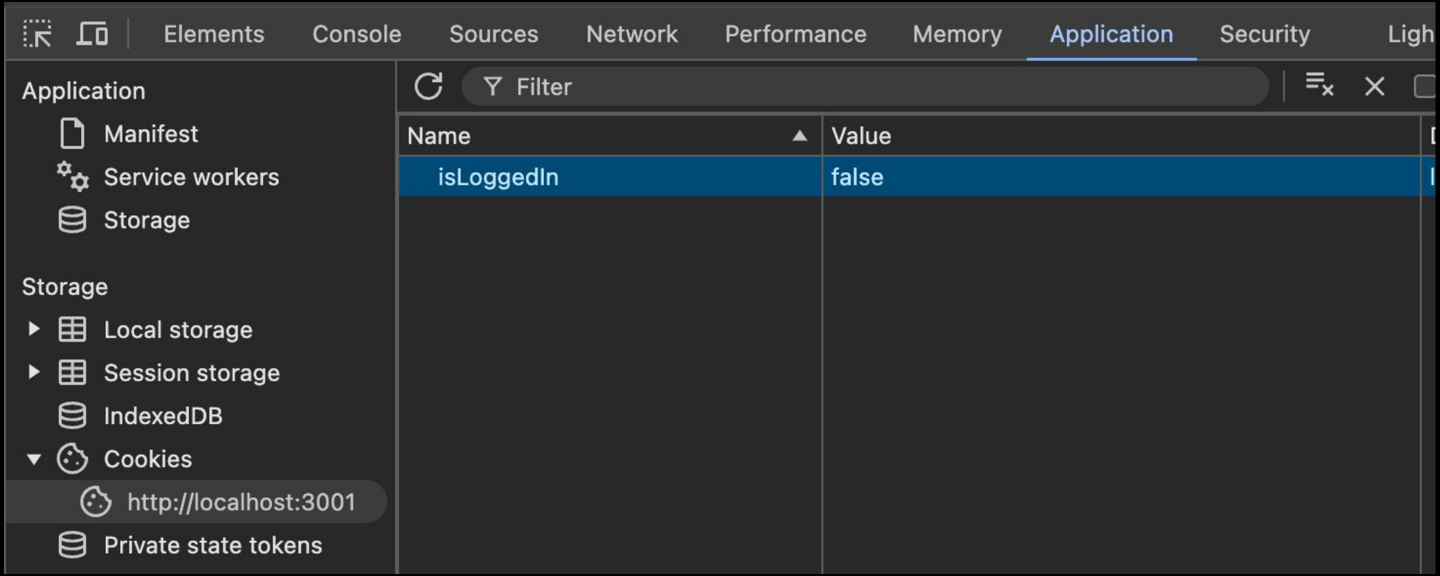
```
exports.postLogin = (req, res, next) => {
  console.log(req.body);
  res.cookie("isLoggedIn", true);
  res.redirect("/");
};
```

3.

```
app.use((req, res, next) => {
  req.isLoggedIn = req.get('Cookie')?.split('=')[1] || false;
  console.log(req.isLoggedIn);
  next();
});
```

18.4 Using a Cookie

4.



Name	Value
isLoggedIn	false