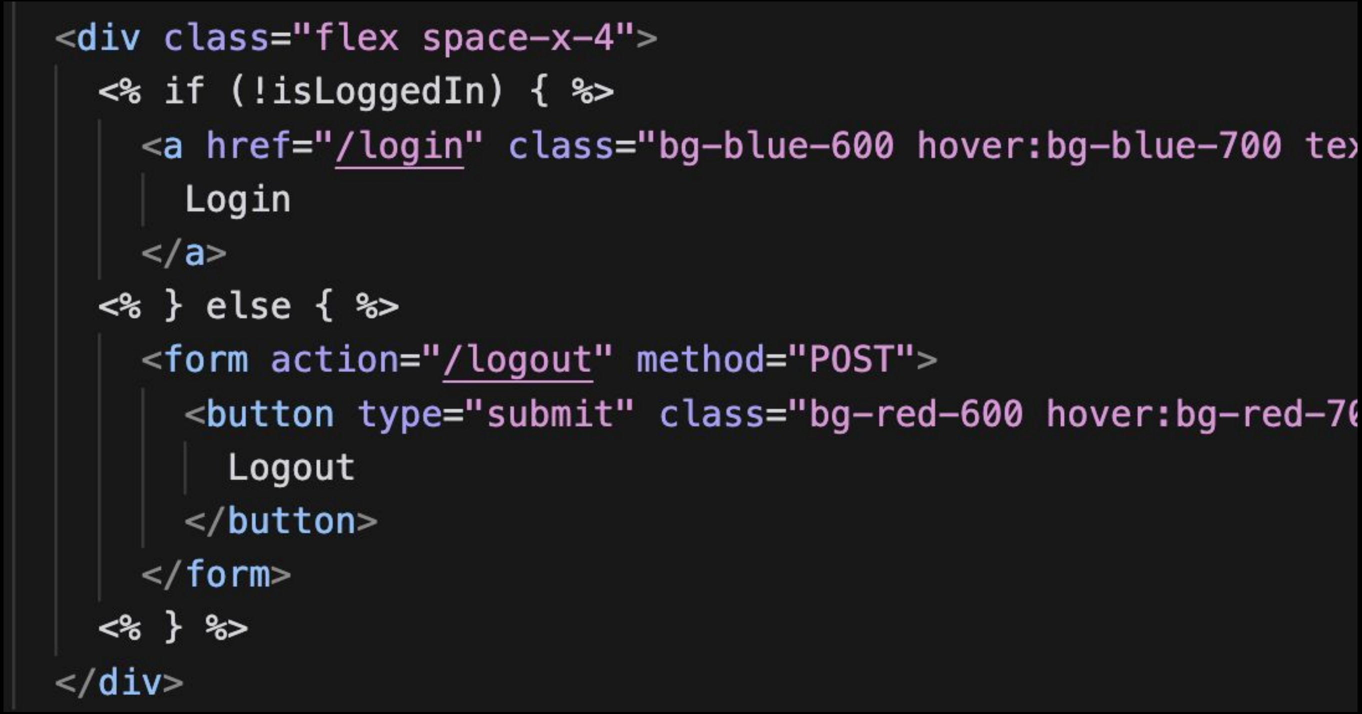
18.5 Define the Logout Feature

1. Define a logout button in nav bar that should come only when user is logged in. Button should be a form that submits to link /logout.
2. Hide the login button in case user is logged in.
3. Handle the logout path and set the isLoggedIn cookie to false.

18.5 Define the Logout Feature

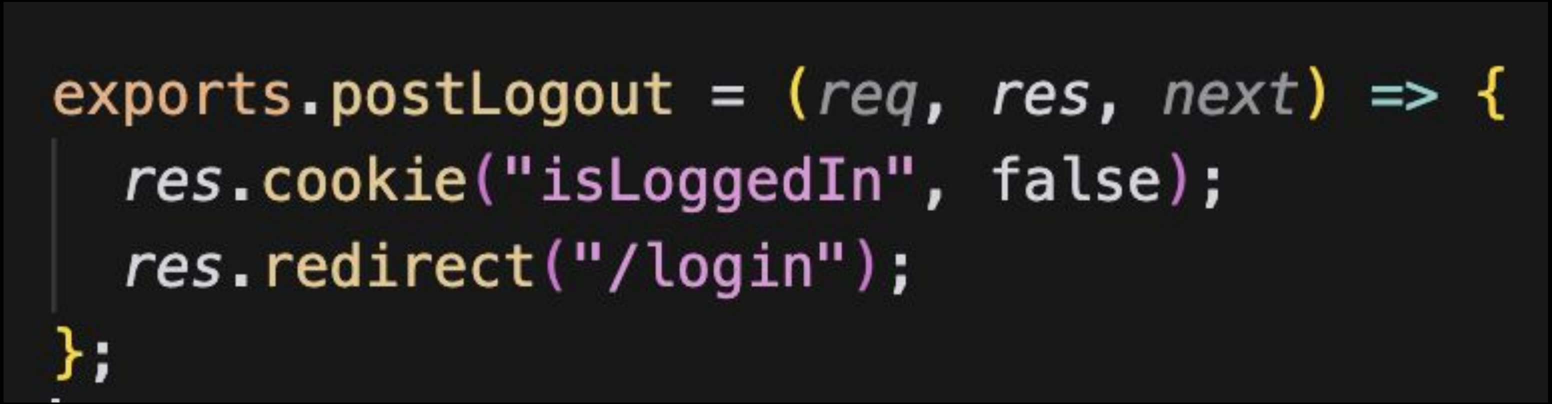
1, 2.



```
<div class="flex space-x-4">
  <% if (!isLoggedIn) { %>
    <a href="/login" class="bg-blue-600 hover:bg-blue-700 te
      Login
    </a>
  <% } else { %>
    <form action="/logout" method="POST">
      <button type="submit" class="bg-red-600 hover:bg-red-7
        Logout
      </button>
    </form>
  <% } %>
</div>
```

18.5 Define the Logout Feature

3.

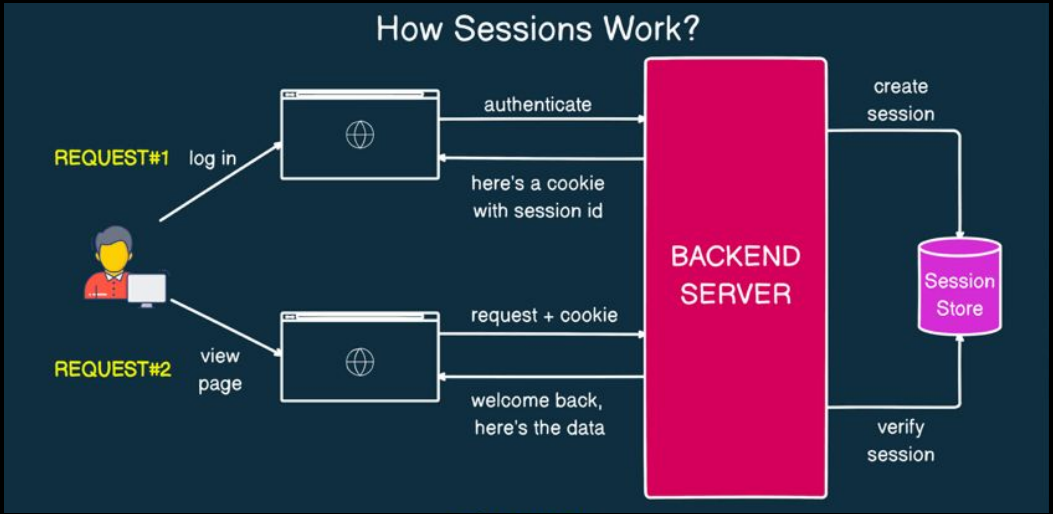


```
exports.postLogout = (req, res, next) => {
  res.cookie("isLoggedIn", false);
  res.redirect("/login");
};
```

18.6 Problem with Cookies

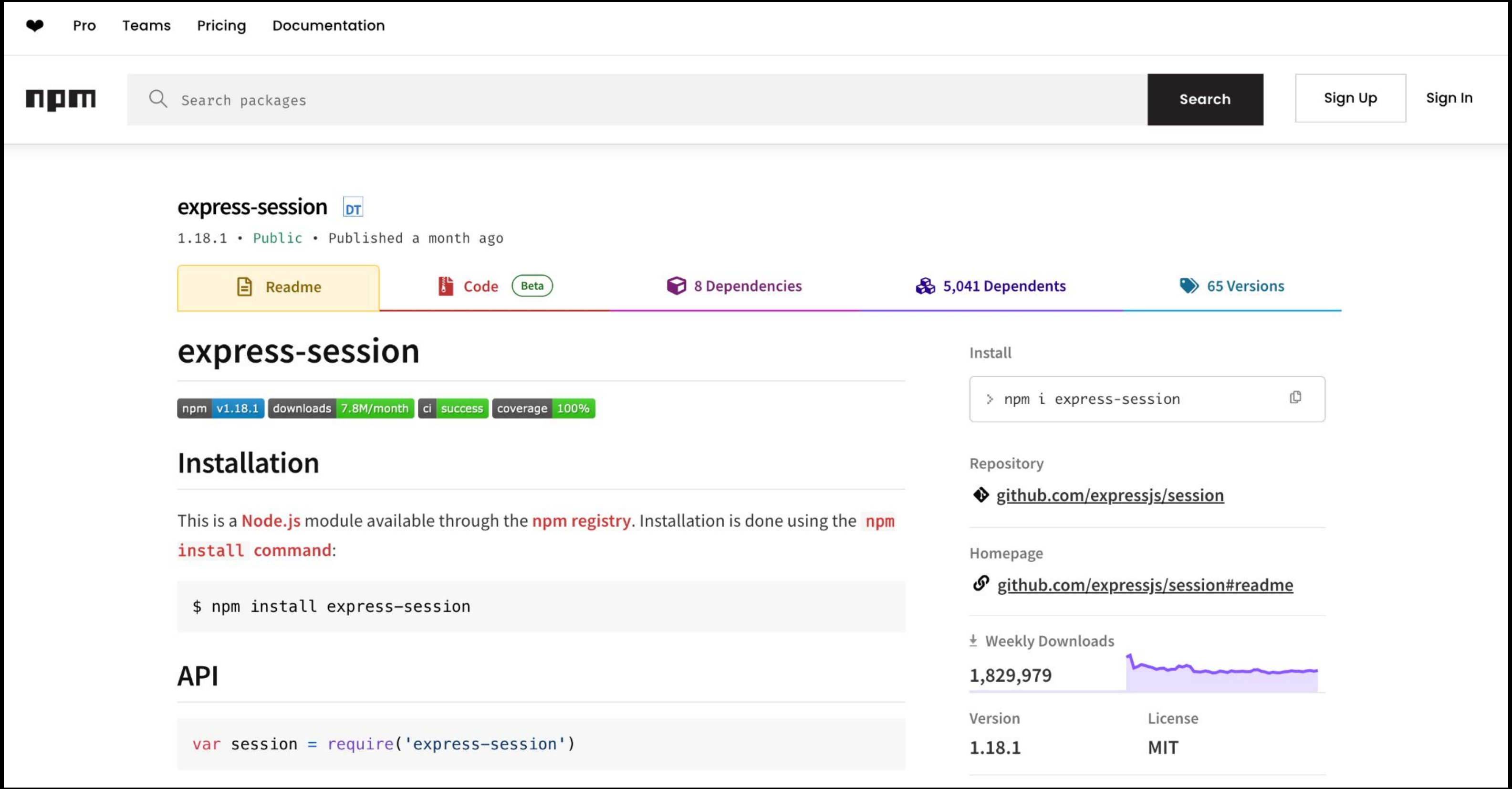
1. Cookies can be intercepted or stolen, posing security risks.
2. They have limited storage capacity (about 4KB).
3. Users can delete or modify cookies, leading to data loss or tampering.
4. Data in cookies is not encrypted, making sensitive information vulnerable.
5. Storing important info in cookies exposes it to client-side attacks.

18.7 What are Sessions



1. Sessions are server-side storage mechanisms that track user interactions with a website.
2. They maintain user state and data across multiple requests in a web application.
3. Sessions enable persistent user experiences by maintaining state between the client and server over stateless HTTP.

18.8 Installing Session Package



Pro Teams Pricing Documentation

Search packages Search Sign Up Sign In

express-session DT

1.18.1 • Public • Published a month ago

Readme | Code Beta | 8 Dependencies | 5,041 Dependents | 65 Versions

express-session

npm v1.18.1 downloads 7.8M/month ci success coverage 100%

Installation

This is a Node.js module available through the npm registry. Installation is done using the `npm install` command:

```
$ npm install express-session
```

API

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

Install

```
> npm i express-session
```

Repository

github.com/expressjs/session

Homepage

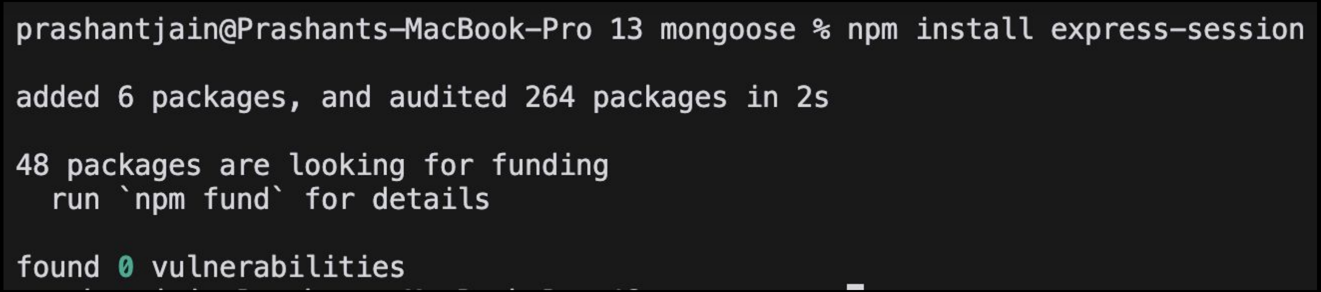
github.com/expressjs/session#readme

Weekly Downloads

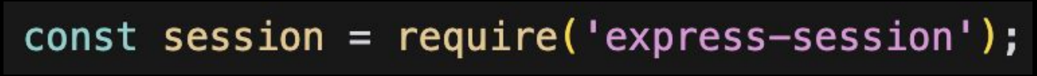
1,829,979

Version	License
1.18.1	MIT

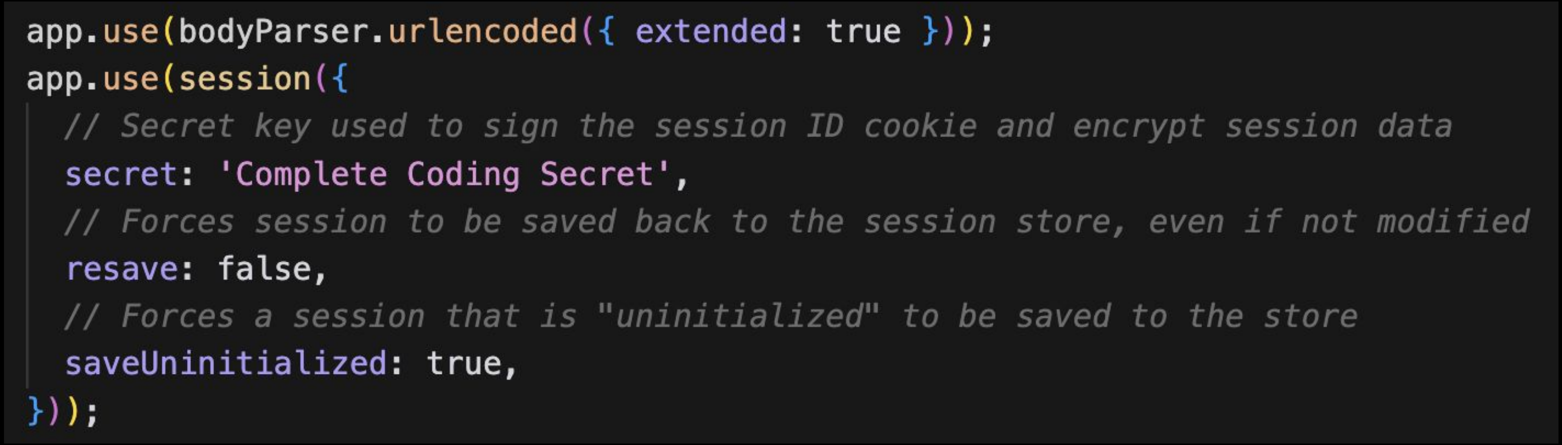
18.8 Installing Session Package



```
prashantjain@Prashants-MacBook-Pro 13 mongoose % npm install express-session

added 6 packages, and audited 264 packages in 2s

48 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```



```javascript
const session = require('express-session');
```



```javascript
app.use(bodyParser.urlencoded({ extended: true }));
app.use(session({
  // Secret key used to sign the session ID cookie and encrypt session data
  secret: 'Complete Coding Secret',
  // Forces session to be saved back to the session store, even if not modified
  resave: false,
  // Forces a session that is "uninitialized" to be saved to the store
  saveUninitialized: true,
}));
```

18.9 Creating a Session

1. Remove setting the cookie and now save the flag in session.
2. Check the browser for cookie changes.
3. Log Session in some get request

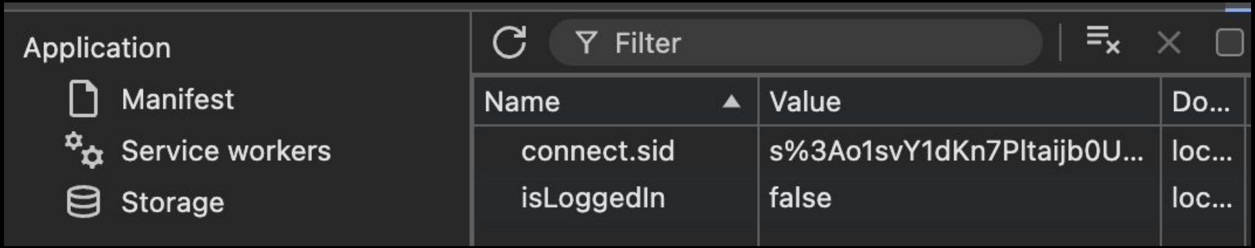
- Sensitive info is stored on server.
- Same session is valid for all requests from one user.
- Try to use a different browser and show that session is different
- Sessions are stored in memory so they reset when server restarts.

18.9 Creating a Sessions

1.

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

2.



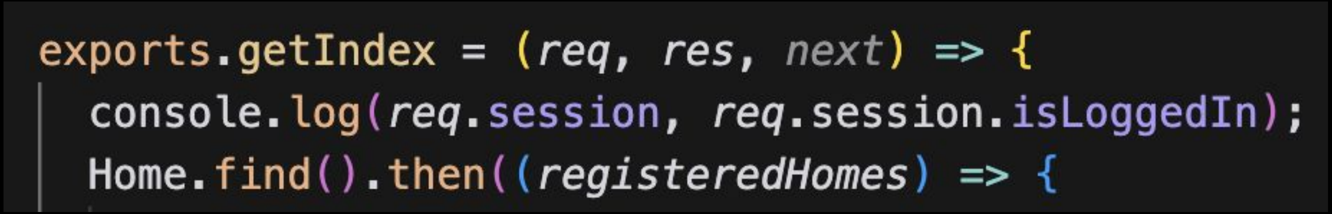
Application
- Manifest
- Service workers
- Storage

Filter

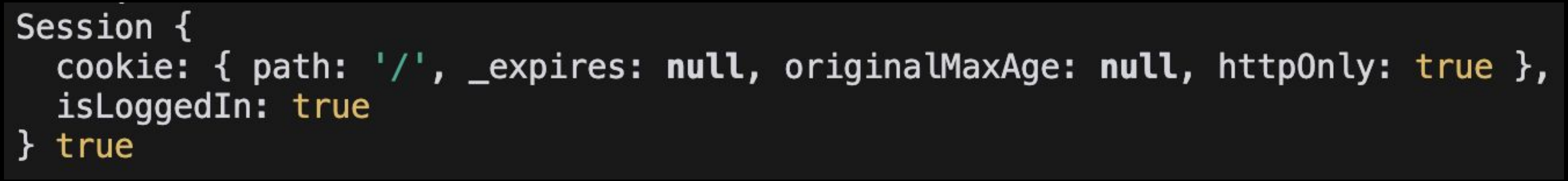
Name	Value	Do...
connect.sid	s%3Ao1svY1dKn7Pltaijb0U...	loc...
isLoggedIn	false	loc...

18.9 Creating a Sessions

3.

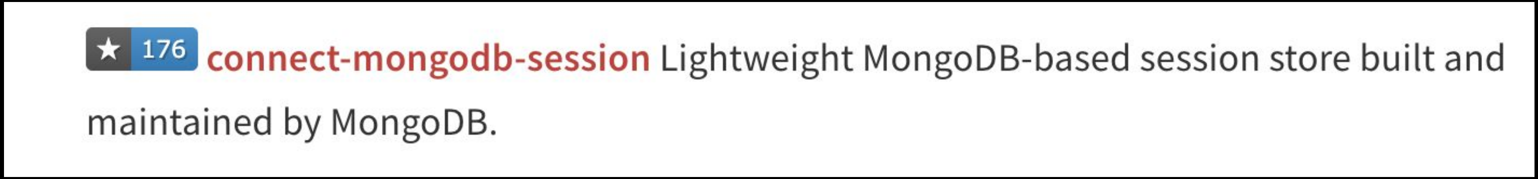


```
exports.getIndex = (req, res, next) => {
  console.log(req.session, req.session.isLoggedIn);
  Home.find().then((registeredHomes) => {
```

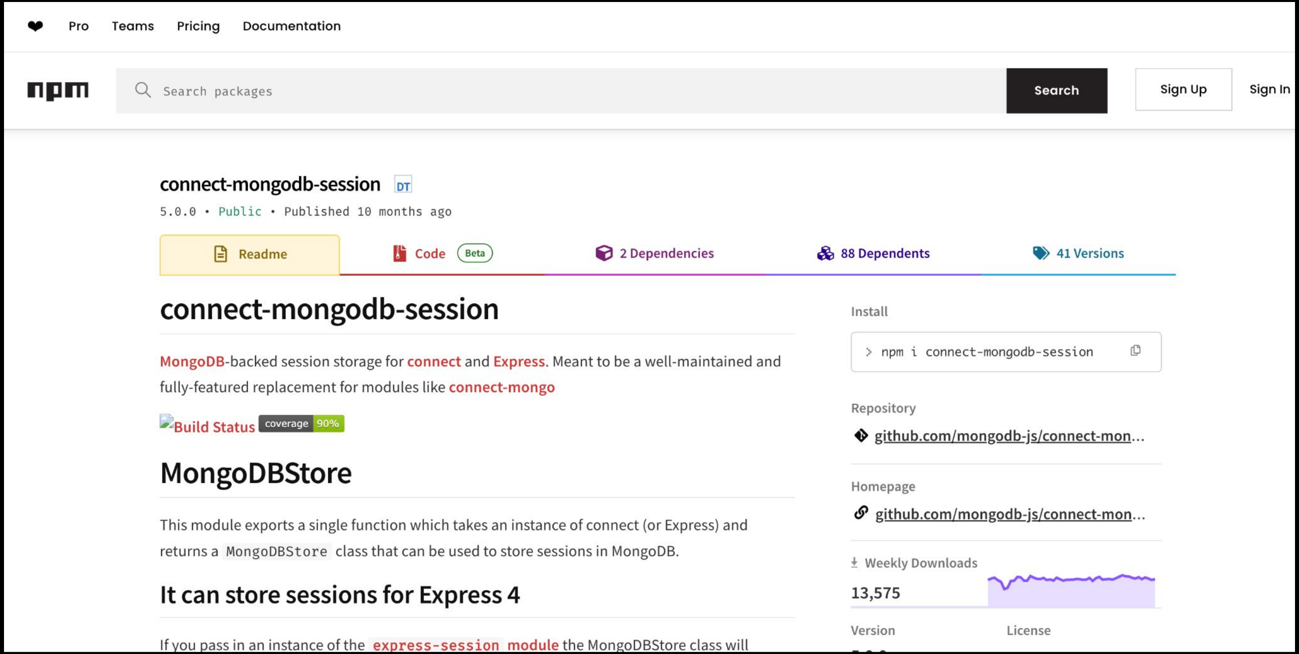


```
Session {
  cookie: { path: '/', _expires: null, originalMaxAge: null, httpOnly: true },
  isLoggedIn: true
} true
```

18.10 Saving Session in DB



★ 176 connect-mongodb-session Lightweight MongoDB-based session store built and maintained by MongoDB.



connect-mongodb-session DT

5.0.0 • Public • Published 10 months ago

connect-mongodb-session

MongoDB-backed session storage for connect and Express. Meant to be a well-maintained and fully-featured replacement for modules like connect-mongo

Build Status coverage 90%

MongoDBStore

This module exports a single function which takes an instance of connect (or Express) and returns a `MongoDBStore` class that can be used to store sessions in MongoDB.

It can store sessions for Express 4

If you pass in an instance of the `express-session` module the MongoDBStore class will

Install

```
> npm i connect-mongodb-session
```

Repository

github.com/mongodb-js/connect-mon...

Homepage

github.com/mongodb-js/connect-mon...

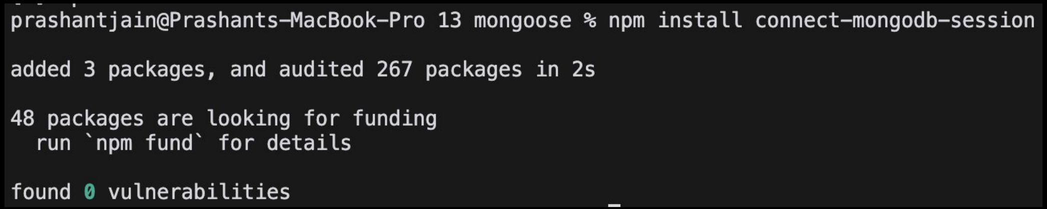
Weekly Downloads

13,575

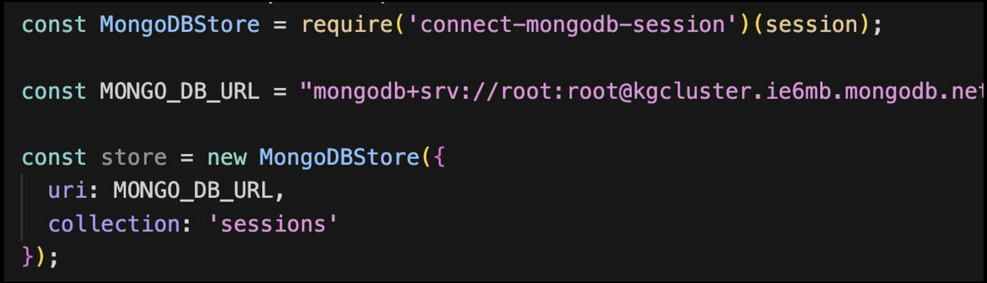
Version

License

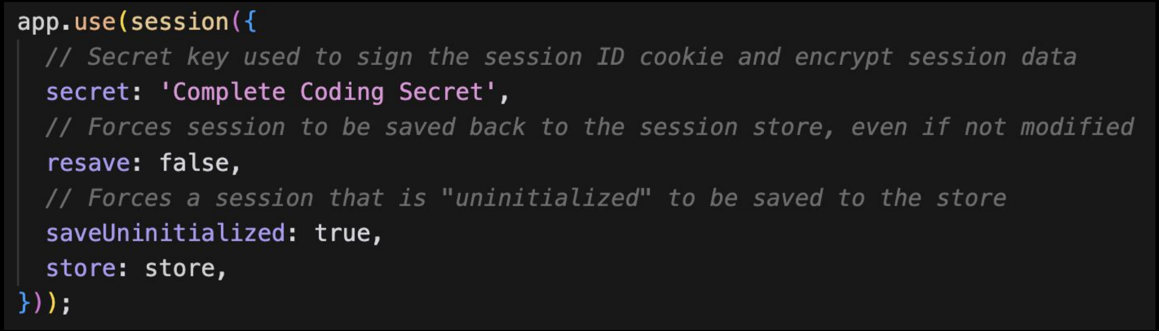
18.10 Saving Session in DB



```
prashantjain@Prashants-MacBook-Pro 13 mongoose % npm install connect-mongodb-session

added 3 packages, and audited 267 packages in 2s

48 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```



```javascript
const MongoDBStore = require('connect-mongodb-session')(session);

const MONGO_DB_URL = "mongodb+srv://root:root@kgcluster.ie6mb.mongodb.net

const store = new MongoDBStore({
  uri: MONGO_DB_URL,
  collection: 'sessions'
});
```



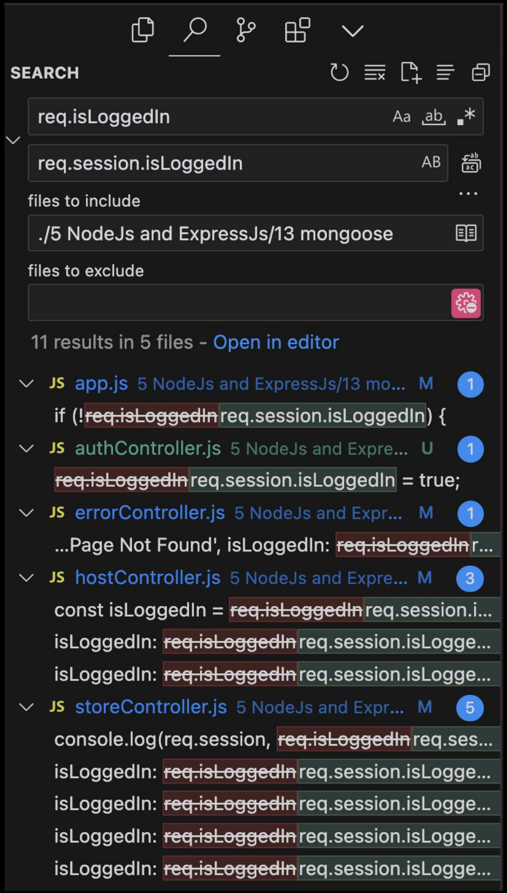
```javascript
app.use(session({
  // Secret key used to sign the session ID cookie and encrypt session data
  secret: 'Complete Coding Secret',
  // Forces session to be saved back to the session store, even if not modified
  resave: false,
  // Forces a session that is "uninitialized" to be saved to the store
  saveUninitialized: true,
  store: store,
}));
```

Practise Milestone

Take your airbnb forward:

- Cleanup cookie code to use session everywhere.
- Remove the cookie middleware.
- Destroy the session on logout.
- Cleanup Logs.
- Understand why leaving the cookie on logout is fine.

Practise Milestone (Solution)



SEARCH

req.isLoggedIn

req.session.isLoggedIn

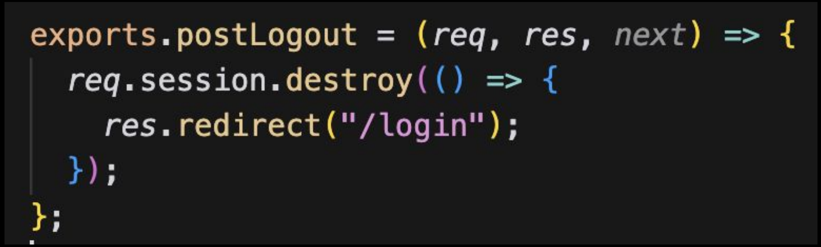
files to include

./5 NodeJs and ExpressJs/13 mongoose

files to exclude

11 results in 5 files - Open in editor

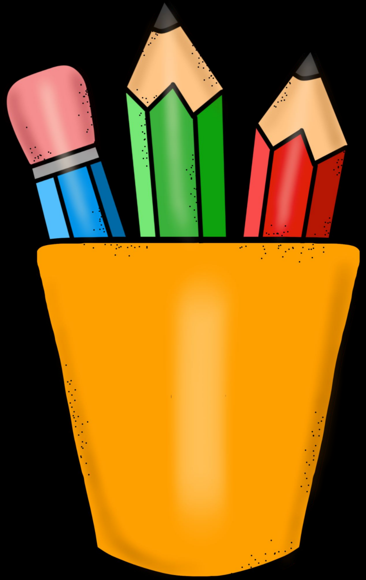
```
app.js 5 NodeJs and ExpressJs/13 mo... M 1
  if (!~~req.isLoggedIn~~req.session.isLoggedIn) {
authController.js 5 NodeJs and Expre... U 1
  ~~req.isLoggedIn~~req.session.isLoggedIn = true;
errorController.js 5 NodeJs and Expr... M 1
  ...Page Not Found', isLoggedIn: ~~req.isLoggedIn~~r...
hostController.js 5 NodeJs and Expre... M 3
  const isLoggedIn = ~~req.isLoggedIn~~req.session.i...
  isLoggedIn: ~~req.isLoggedIn~~req.session.isLogge...
  isLoggedIn: ~~req.isLoggedIn~~req.session.isLogge...
storeController.js 5 NodeJs and Expr... M 5
  console.log(req.session, ~~req.isLoggedIn~~req.ses...
  isLoggedIn: ~~req.isLoggedIn~~req.session.isLogge...
  isLoggedIn: ~~req.isLoggedIn~~req.session.isLogge...
  isLoggedIn: ~~req.isLoggedIn~~req.session.isLogge...
  isLoggedIn: ~~req.isLoggedIn~~req.session.isLogge...
```



```
exports.postLogout = (req, res, next) => {
  req.session.destroy(() => {
    res.redirect("/login");
  });
};
```

Revision

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

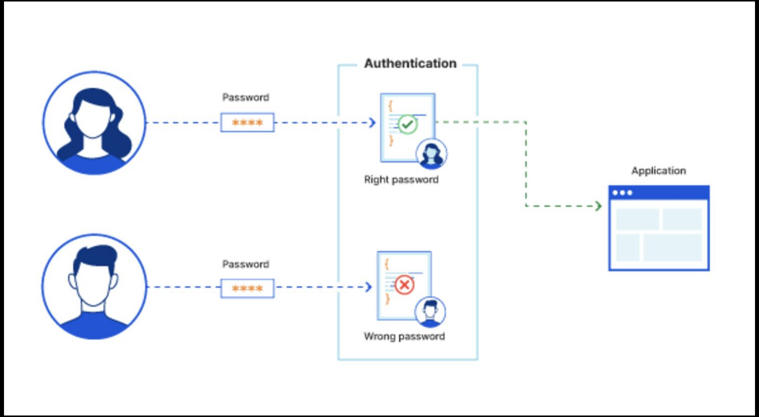


19. Authentication & Authorization

1. What is Authentication
2. What is Authorization
3. Authentication vs Authorization
4. Session based Authentication
5. Signup UI
6. Using Express Validator
7. Adding User Model
8. Encrypting Password
9. Implementing Login

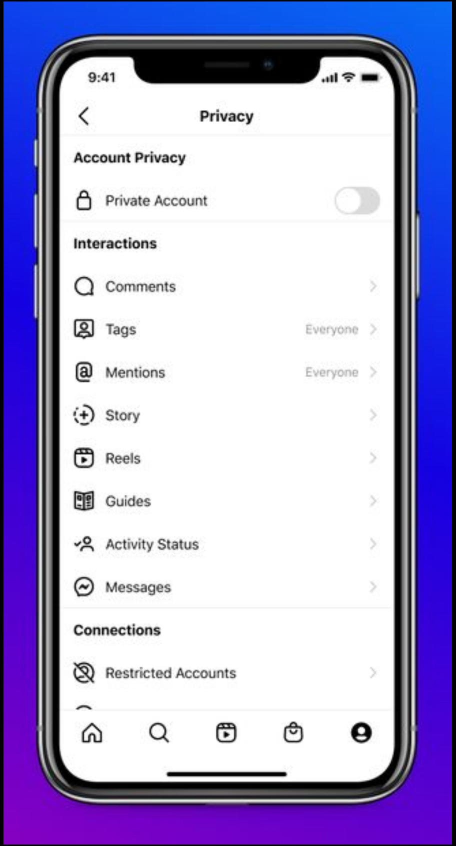
19.1 What is Authentication

1. Authentication is the process of verifying the identity of a user or system accessing an application.
2. It ensures that only authorized users can access protected resources and features.
3. In ReactJS (Frontend), authentication handles user input for login forms and manages authentication states.
4. In NodeJS (Backend), authentication checks user credentials against a database and issues tokens or sessions.
5. Authentication is crucial for security, protecting data, and providing personalized experiences in web applications.



19.2 What is Authorization

1. Authorization is the process of determining what actions a user is permitted to perform within an application.
2. It ensures that users can access only the resources and functionalities they have permission for.
3. In ReactJS (Frontend), authorization controls the display of UI elements and routes based on user roles or permissions.
4. In NodeJS (Backend), authorization involves middleware or logic that checks user permissions before processing requests.
5. Authorization enhances security by restricting access to sensitive data and operations, complementing the authentication process.



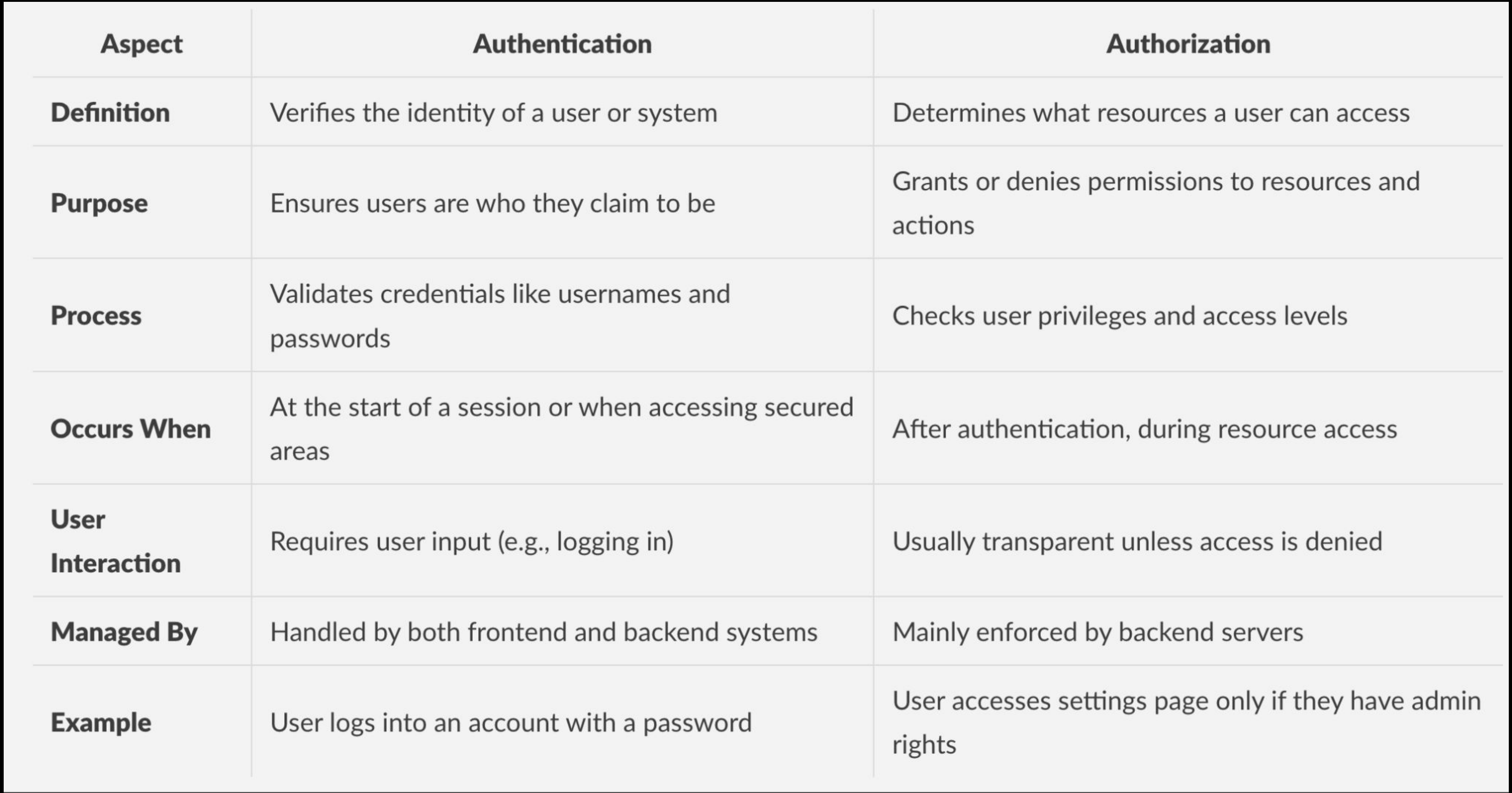
19.3 Authentication vs Authorization

Authentication
Authorization
user name
******
LOGIN

Who are you?
Validate a system is accessing by the right person

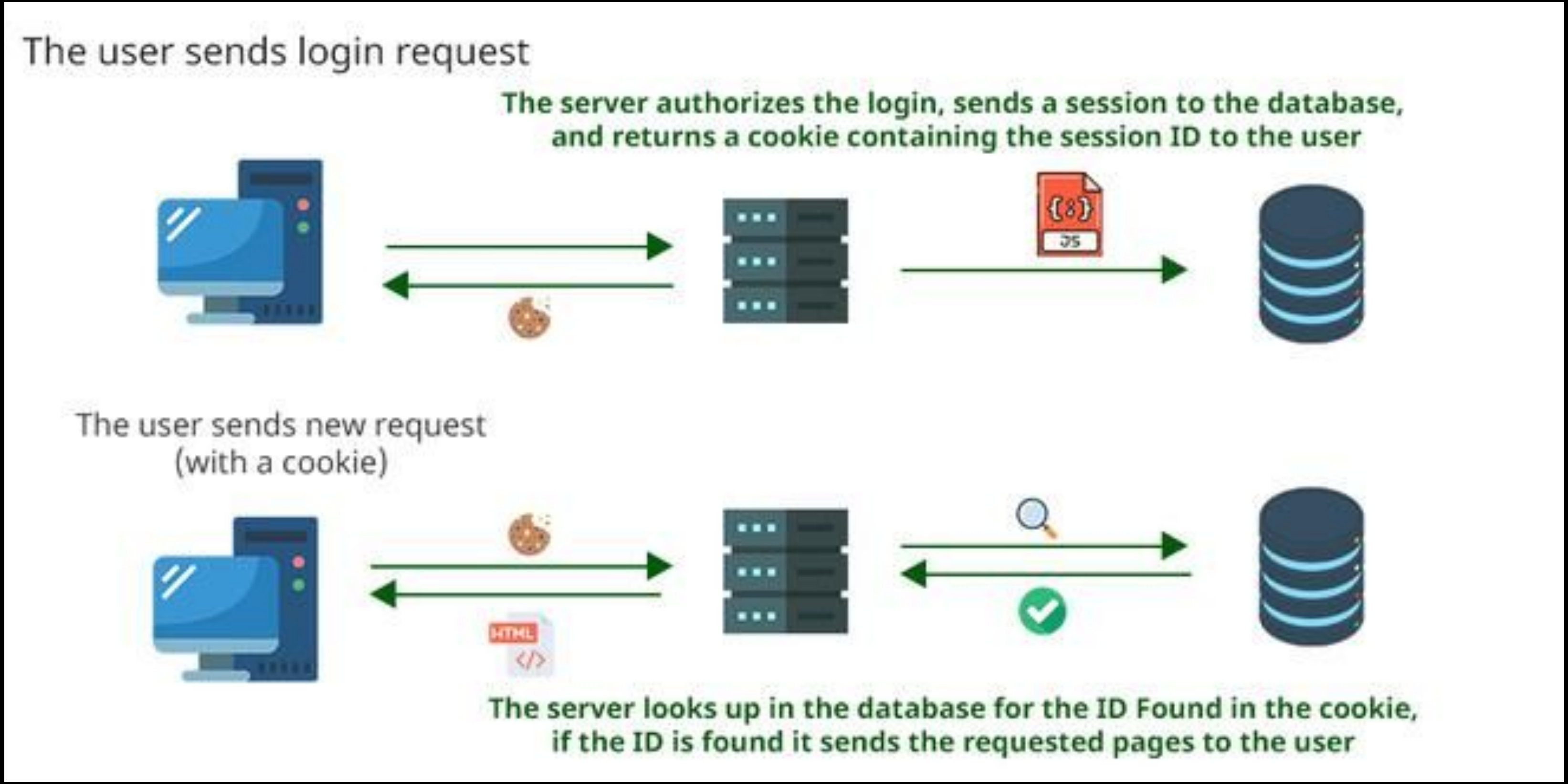
Are you allowed to do that?
Check users' permissions to access data

19.3 Authentication vs Authorization



Aspect	Authentication	Authorization
Definition	Verifies the identity of a user or system	Determines what resources a user can access
Purpose	Ensures users are who they claim to be	Grants or denies permissions to resources and actions
Process	Validates credentials like usernames and passwords	Checks user privileges and access levels
Occurs When	At the start of a session or when accessing secured areas	After authentication, during resource access
User Interaction	Requires user input (e.g., logging in)	Usually transparent unless access is denied
Managed By	Handled by both frontend and backend systems	Mainly enforced by backend servers
Example	User logs into an account with a password	User accesses settings page only if they have admin rights

19.4 Session based Authentication

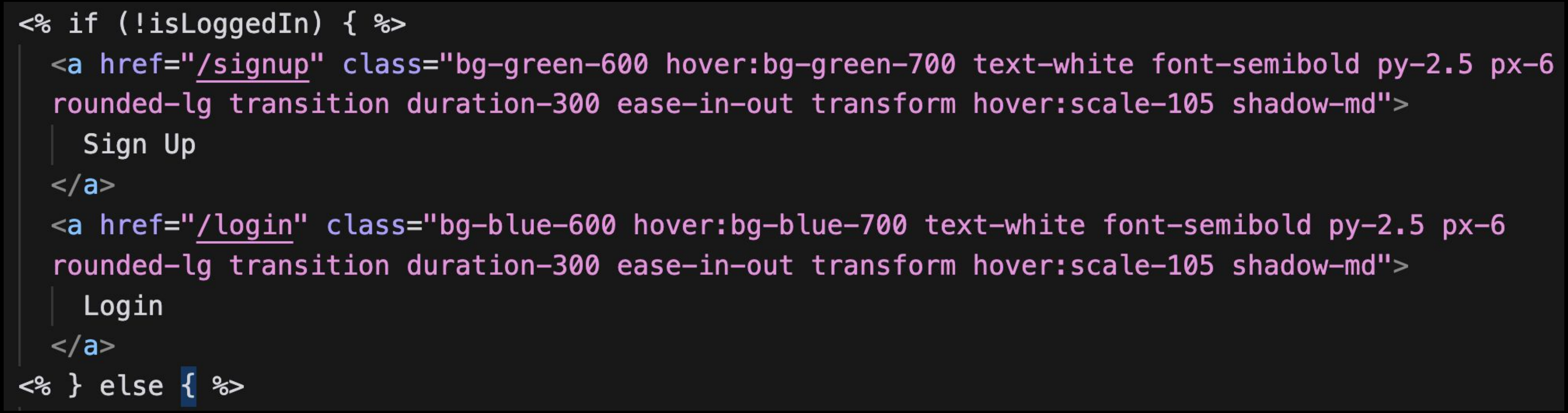

The user sends login request
The server authorizes the login, sends a session to the database, and returns a cookie containing the session ID to the user
JS
The user sends new request (with a cookie)
HTML
The server looks up in the database for the ID Found in the cookie, if the ID is found it sends the requested pages to the user

19.5 Signup UI

1. Define a signup button in navigation bar along with sign-in. It should point to a link /signup.
2. Define a auth/signup.ejs file that has email, password and confirm password fields and submits POST request to /signup
3. Define routes in authRouter and behaviour in authController.
4. Fix the UI of the app to look pretty.

19.5 Signup UI

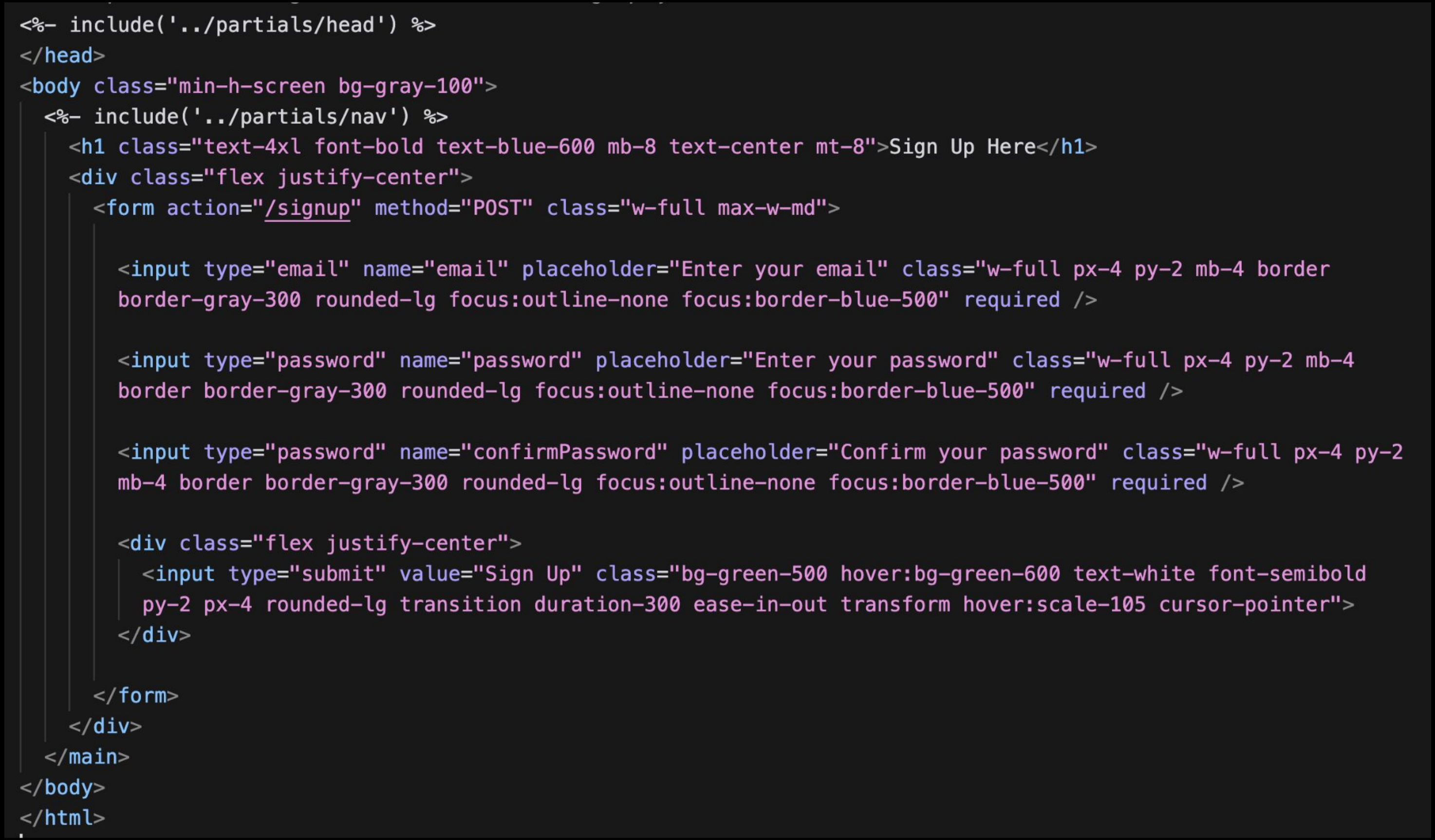
1.



```
<% if (!isLoggedIn) { %>
  <a href="/signup" class="bg-green-600 hover:bg-green-700 text-white font-semibold py-2.5 px-6
  rounded-lg transition duration-300 ease-in-out transform hover:scale-105 shadow-md">
    Sign Up
  </a>
  <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>
<% } else { %>
```

19.5 Signup UI

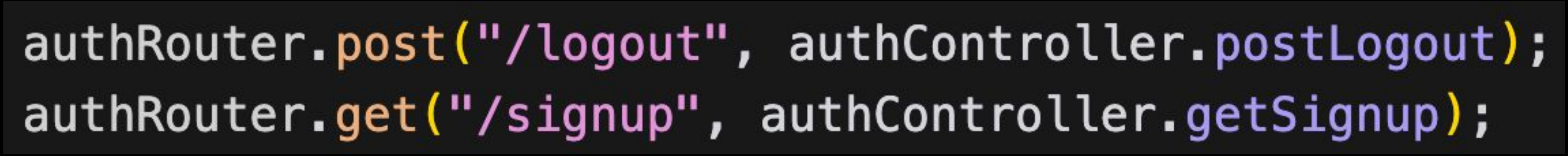
2.



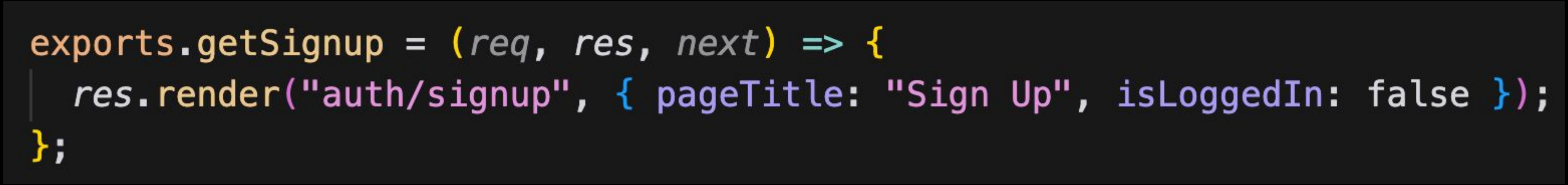
```html
<%- include('../partials/head') %>
</head>
<body class="min-h-screen bg-gray-100">
  <%- include('../partials/nav') %>
    <h1 class="text-4xl font-bold text-blue-600 mb-8 text-center mt-8">Sign Up Here</h1>
    <div class="flex justify-center">
      <form action="/signup" method="POST" class="w-full 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" required />

        <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" required />

        <input type="password" name="confirmPassword" placeholder="Confirm your password" class="w-full px-4 py-2
        mb-4 border border-gray-300 rounded-lg focus:outline-none focus:border-blue-500" required />

        <div class="flex justify-center">
          <input type="submit" value="Sign Up" class="bg-green-500 hover:bg-green-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>
```

19.5 Signup UI

3, 4.



```
authRouter.post("/logout", authController.postLogout);
authRouter.get("/signup", authController.getSignup);
```



```
exports.getSignup = (req, res, next) => {
  res.render("auth/signup", { pageTitle: "Sign Up", isLoggedIn: false });
};
```