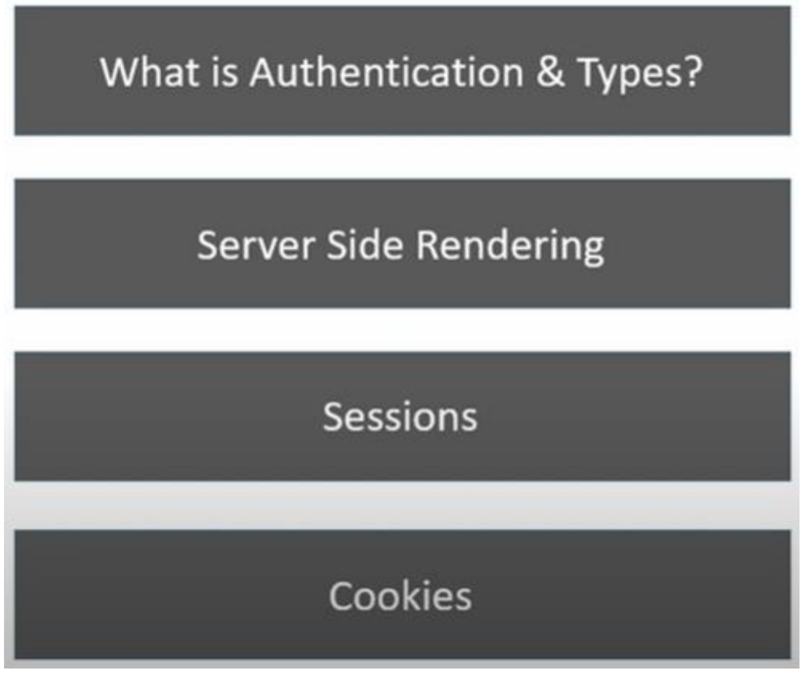
STATEFUL AUTHENTICATION

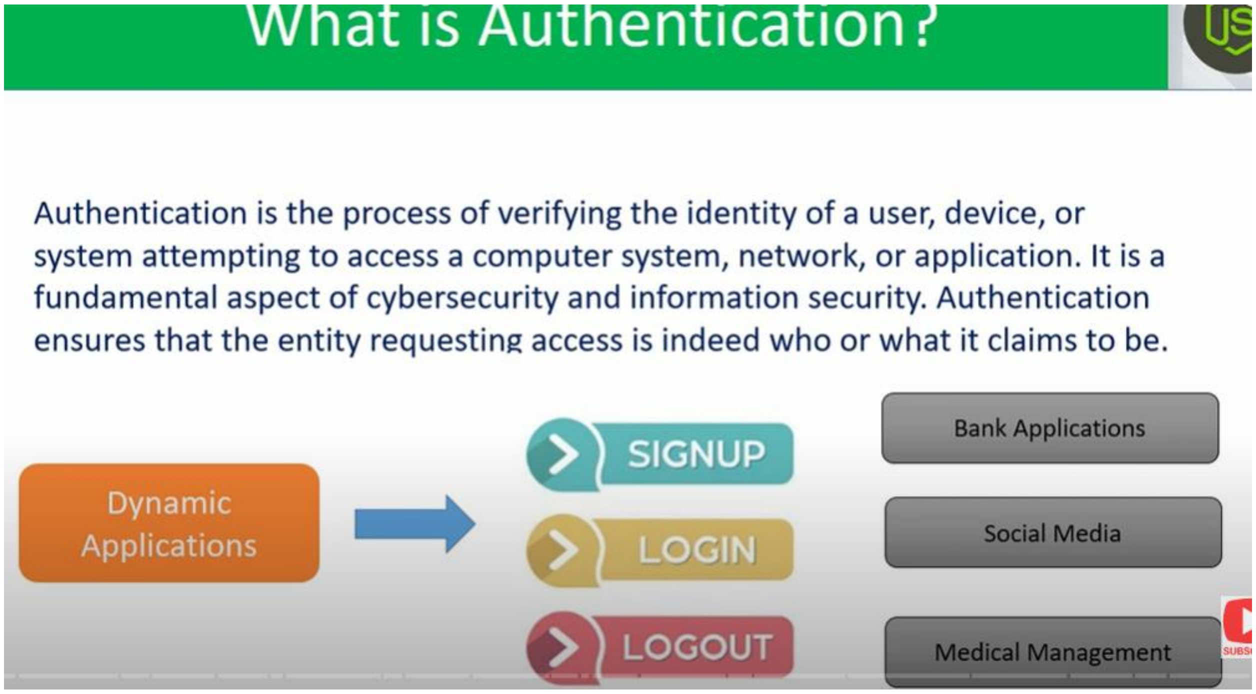


What is Authentication & Types?

Server Side Rendering

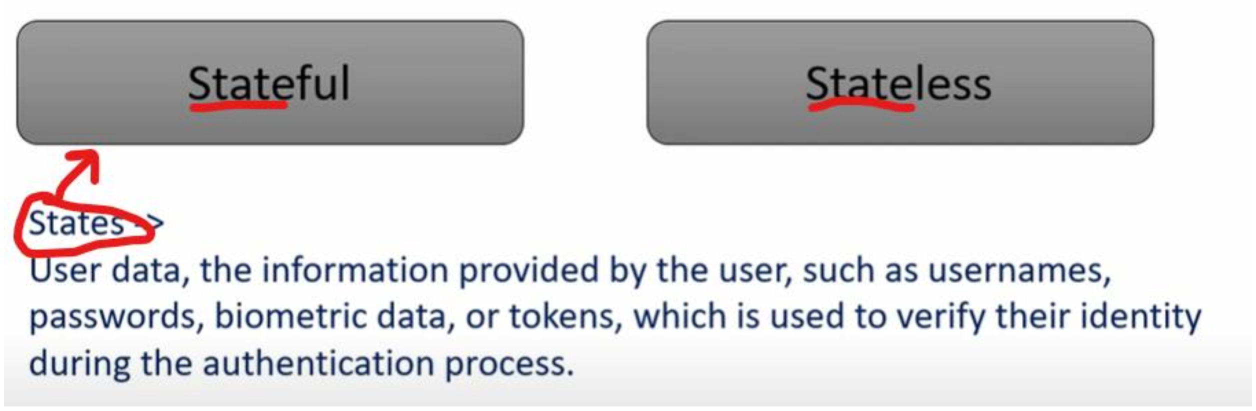
Sessions

Cookies



What is Authentication?

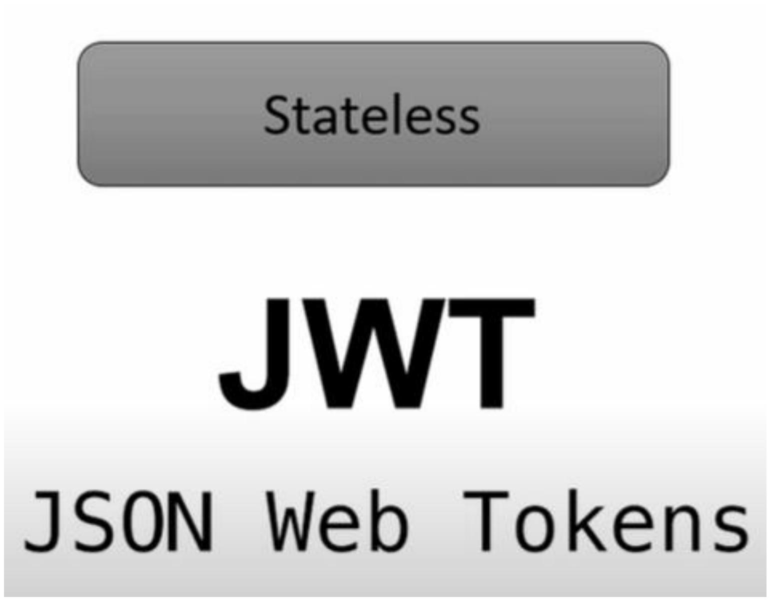
Authentication is the process of verifying the identity of a user, device, or system attempting to access a computer system, network, or application. It is a fundamental aspect of cybersecurity and information security. Authentication ensures that the entity requesting access is indeed who or what it claims to be.



Stateful

Stateless

States ->
User data, the information provided by the user, such as usernames, passwords, biometric data, or tokens, which is used to verify their identity during the authentication process.



Stateless

JWT

JSON Web Tokens

** IN STATELESS AUTHETICATION INSTEAD OF STATE WE USE TOKENS IN LECTURE WE STUDY ABOUT STATELESS
HERE WE LEARN ABOUT STATEFUL AUTHETICATION

** STATEFUL AUTHECATION MAINLY FOCUS ABOUT → SESSION AND COOKIE

1. Session :

A session is used to save information on the server momentarily so that it may be utilized across various pages of the website. It is the overall amount of time spent on an activity. The user session begins when the user logs in to a specific network application and ends when the user logs out of the program or shuts down the machine.

Session values are far more secure since they are saved in binary or encrypted form and can only be decoded at the server. When the user shuts down the machine or logs out of the program, the session values are automatically deleted. We must save the values in the database to keep them forever.

2. Cookie :

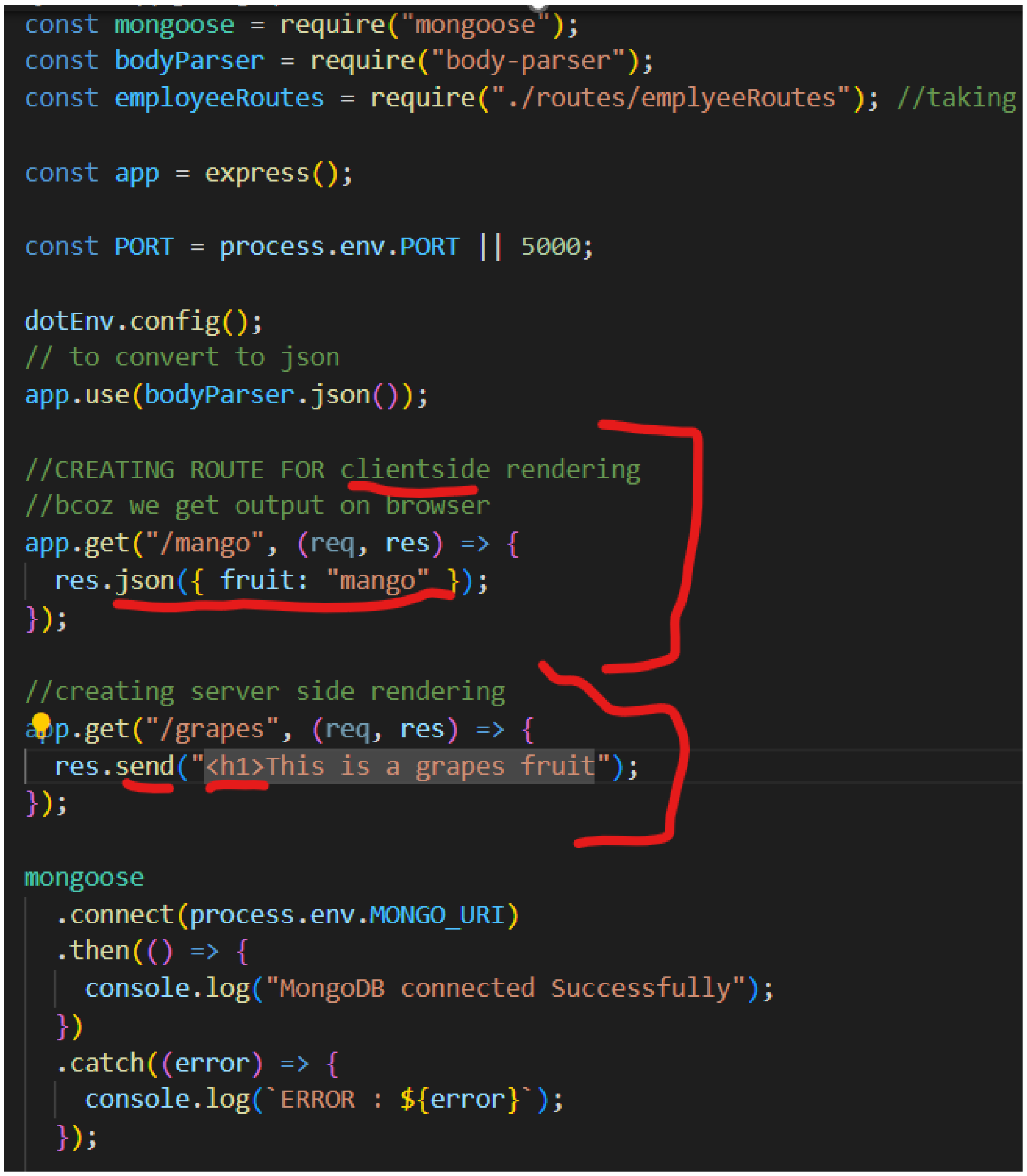
A cookie is a small text file that is saved on the user's computer. The maximum file size for a cookie is 4KB. It is also known as an HTTP cookie, a web cookie, or an internet cookie. When a user first visits a website, the site sends data packets to the user's computer in the form of a cookie.

The information stored in cookies is not safe since it is kept on the client-side in a text format that anybody can see. We can activate or disable cookies based on our needs.

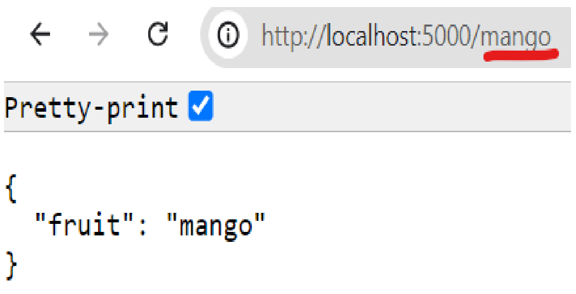
STATE FULL --→ MAINLY BASED ON SERVER SIDE

STEP-1 -→SERVERSIDE RENDERING

OPEN SERVER.JS ON API

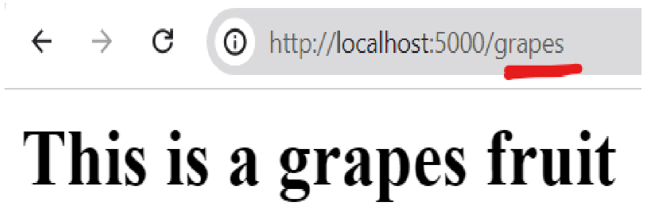


```javascript
const mongoose = require("mongoose");
const bodyParser = require("body-parser");
const employeeRoutes = require("./routes/emplyeeRoutes"); //taking

const app = express();

const PORT = process.env.PORT || 5000;

dotEnv.config();
// to convert to json
app.use(bodyParser.json());

//CREATING ROUTE FOR clientside rendering
//bcoz we get output on browser
app.get("/mango", (req, res) => {
  res.json({ fruit: "mango" });
});

//creating server side rendering
app.get("/grapes", (req, res) => {
  res.send("<h1>This is a grapes fruit");
});

mongoose
  .connect(process.env.MONGO_URI)
  .then(() => {
    console.log("MongoDB connected Successfully");
  })
  .catch((error) => {
    console.log(`ERROR : ${error}`);
  });
```


http://localhost:5000/mango

Pretty-print ✓

```
{
  "fruit": "mango"
}
```


http://localhost:5000/grapes

This is a grapes fruit

In client side rendering→ we getting out in json format

In serverside rendering→ we getting output as html css direct out put because we are keeping html and css code directly in route

**in server side rendering-→we are keeping html css code directly in the routes → so better to use→templete engines →EJS

*** EJS → used for render styling from server →to write code of html css

-→to install EJS→npm install ejs

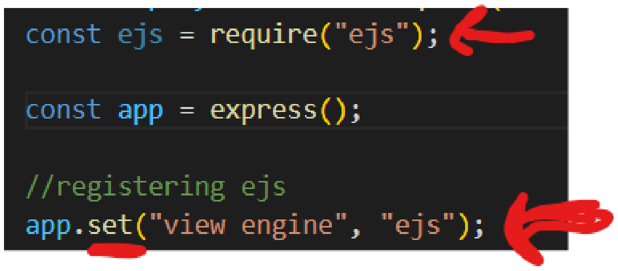
→ we mainly follows mvc rule

model→to create schema

controller→to create logic of each record

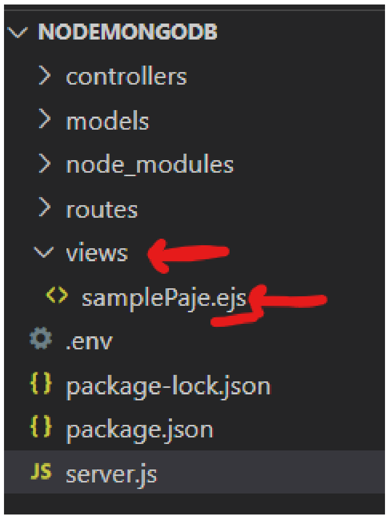
view → to show the output

**now import and register ejs in server.js

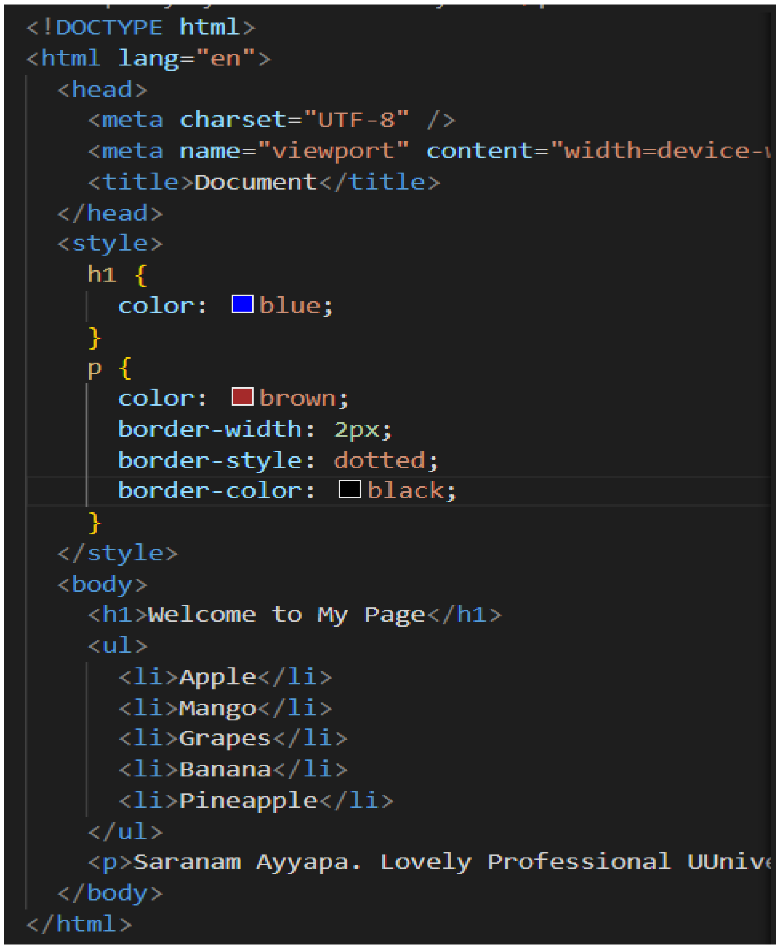


```
const ejs = require("ejs");

const app = express();

//registering ejs
app.set("view engine", "ejs");
```

Created view folder → ejs file //to add html css for serverside rendering

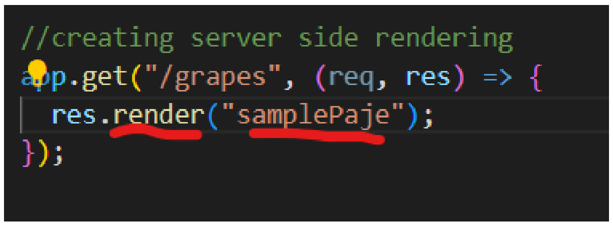

```
NODEMONGODB
  > controllers
  > models
  > node_modules
  > routes
  v views
     samplePaje.ejs
  .env
  package-lock.json
  package.json
  server.js
```

In samplePage.ejs file

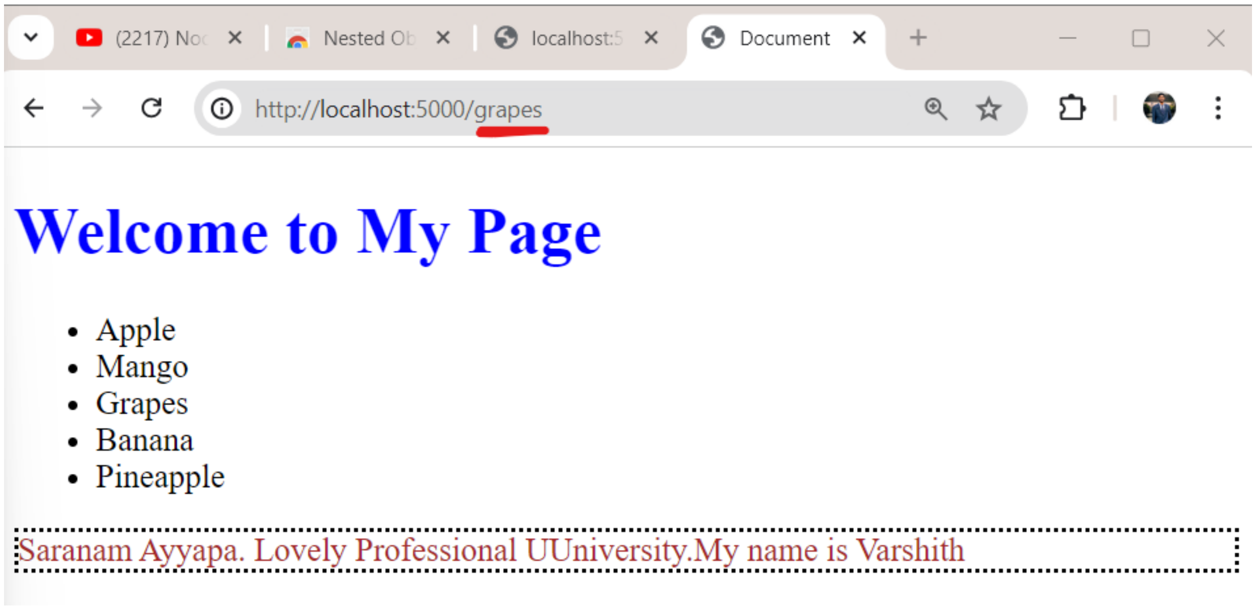

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-
    <title>Document</title>
  </head>
  <style>
    h1 {
      color: blue;
    }
    p {
      color: brown;
      border-width: 2px;
      border-style: dotted;
      border-color: black;
    }
  </style>
  <body>
    <h1>Welcome to My Page</h1>
    <ul>
      <li>Apple</li>
      <li>Mango</li>
      <li>Grapes</li>
      <li>Banana</li>
      <li>Pineapple</li>
    </ul>
    <p>Saranam Ayyapa. Lovely Professional UUnive
  </body>
</html>
```

Now we are link this file to serverside rendering

We will use →res.render()



```
//creating server side rendering
app.get("/grapes", (req, res) => {
  res.render("samplePaje");
});
```



Welcome to My Page

- Apple
- Mango
- Grapes
- Banana
- Pineapple

Saranam Ayyapa. Lovely Professional UUniversity.My name is Varshith

This is called ServerSide rendering -→
creating html css files and sending them as request to server called as serverside rendering