Acosta, Kobe Andrei R.

November 18, 2025 – Lab Act

1. **What is a Session (in Web)?**

A session is a way for a website to remember information about a user while they browse through different pages. Since web pages do not naturally remember who the user is between requests, a session helps maintain continuity by storing data on the server. For example, when a user logs in, the server creates a session to keep them recognized as they move from one page to another without needing to log in again. Each session is identified by a unique session ID stored in the user's browser, allowing the server to match the user with their stored data. Sessions are temporary and usually expire after some time or when the user logs out.

Sample Code:

```
// Install: npm install express express-session

const express = require("express");

const session = require("express-session");

const app = express();


app.use(session({

   secret: "mysecretkey",

   resave: false,

   saveUninitialized: true

}));

app.get("/", (req, res) => {

   if (!req.session.views) req.session.views = 1;

   else req.session.views++;

  res.send(`Session Views: ${req.session.views}`);

});
```

```
app.listen(3000, () => console.log("Server running on port 3000"));
```

2. **What is Cookie (in Web)?**

   A cookie is a small piece of data saved on a user's browser by a website. It is used to remember information about the user, such as preferences, login details, or items in a shopping cart. Cookies help improve user experience by keeping certain information available even after the user closes the browser or visits the site again. Because they stay in the browser, cookies can persist longer than sessions, unless they are deleted or designed to expire. While useful, cookies must be handled carefully because they can expose personal information if not protected properly.

   ```
   // Create a cookie

   document.cookie = "username=Juan; expires=Fri, 31 Dec 2025 23:59:59 GMT; path=/";



   // Read cookies

   console.log(document.cookie);
   ```

3. **What is Authentication (in Web)?**

   Authentication is the process of verifying the identity of a user before granting access to a website or system. It ensures that the person trying to log in is truly who they claim to be. The most common form of authentication is entering a username and password, but modern systems also use methods such as one-time codes, fingerprints, face recognition, and login via platforms like Google or Facebook. Once a user is authenticated, websites can personalize their experience or secure important sections of the site. Authentication is essential for protecting accounts, personal data, and sensitive information.

   ```
   <form method="POST" action="login.php">

     <input name="username" placeholder="Username">

     <input name="password" type="password" placeholder="Password">

     <button type="submit">Login</button>

   </form>
   ```

```php
<?php
session_start();

$validUser = "admin";
$validPass = "1234";

if ($_POST['username'] === $validUser && $_POST['password'] === $validPass) {
   $_SESSION['user'] = $_POST['username'];
   echo "Login successful!";
} else {
   echo "Invalid credentials!";
}
?>
```

4. **What is RESTFUL?**

RESTFUL refers to an architectural style used in designing web services and APIs. REST stands for Representational State Transfer, and it defines a set of rules that make communication between client and server simple, consistent, and efficient. A RESTful system uses standard HTTP methods—like GET, POST, PUT, and DELETE—to retrieve, create, update, or remove data. The idea is to treat everything as a "resource" that can be accessed through a clean and meaningful URL. RESTful APIs are popular because they are easy to understand, lightweight, and widely supported across different programming languages and platforms.

GET /users → get all users

POST /users → create user

PUT /users/1 → update user

DELETE /users/1 → delete user

5. **What is API**

An API, or Application Programming Interface, is a set of rules that allows two software systems to communicate with each other. It acts like a bridge, enabling applications to request or exchange information in a safe and structured way. APIs are everywhere in modern technology—for example, weather apps use APIs to get weather data, online stores use APIs to handle payments, and social media platforms use APIs so other apps can share posts or retrieve user information. APIs help developers build more powerful and connected applications without having to reinvent existing features.

weather API → returns weather data

facebook API → allows apps to post to user timeline

```
// Install: npm install express

const express = require("express");

const app = express();

app.use(express.json());


// Fake database

let users = [

    { id: 1, name: "Juan" },

    { id: 2, name: "Maria" }

];


// GET - fetch all users

app.get("/users", (req, res) => {
```

```
    res.json(users);
});

// POST - add a new user
app.post("/users", (req, res) => {
    const newUser = { id: users.length + 1, name: req.body.name };
    users.push(newUser);
    res.json(newUser);
});

// PUT - update user
app.put("/users/:id", (req, res) => {
    const id = parseInt(req.params.id);
    users[id - 1].name = req.body.name;
    res.json(users[id - 1]);
});

// DELETE - remove user
app.delete("/users/:id", (req, res) => {
    const id = parseInt(req.params.id);
    users = users.filter(u => u.id !== id);
    res.json({ message: "User deleted" });
});
```

```
app.listen(3000, () => console.log("REST API running on port 3000"));
```