Q1(a). Install Node.js and create a basic HTTP server using the built-in http module.

1. Setting Up a Node.js Server

Topic: Install Node.js and create a basic HTTP server using the built-in http module.

What is Node.js?

Node.js is a runtime environment that allows you to run JavaScript outside the browser (on the server).

Why use Node.js?

- Fast
- Lightweight
- Good for real-time apps
- Uses JavaScript everywhere (frontend + backend)

Steps to Set Up a Node.js Server

Step 1: Install Node.js

1. Go to **https://nodejs.org**
2. Download **LTS version**
3. Install by clicking Next → Next → Finish
4. Check version:

   node -v
   npm -v

Step 2: Create a Project Folder

Example:

FSD/

Inside it, create a file:

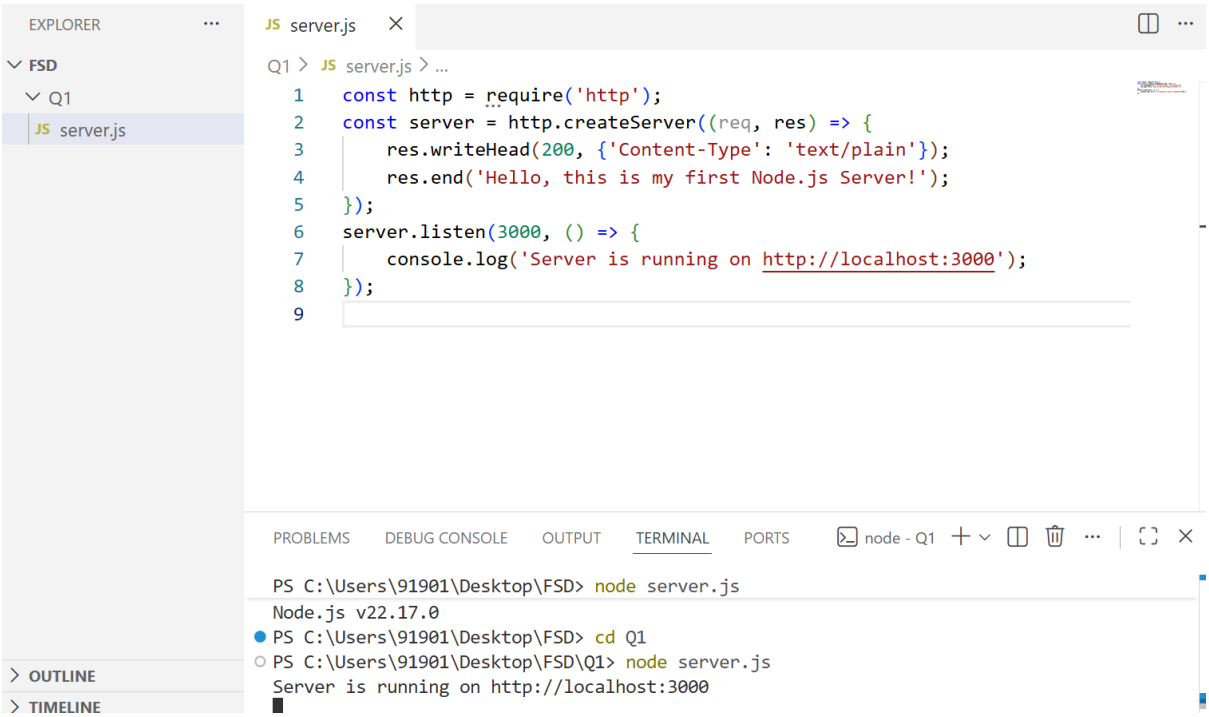
Q1>

   server.js

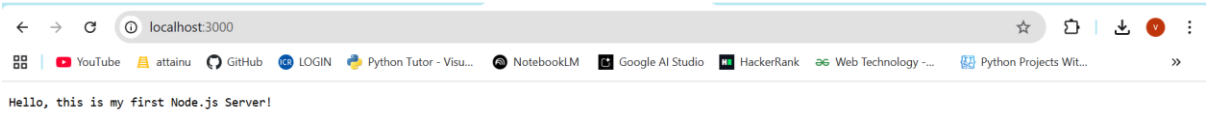
Step 3: Write Basic HTTP Server Using http Module

```
// Importing the 'http' module that allows us to create a web server
const http = require('http');
// Creating the server
// req = request (data coming from browser)
// res = response (data we send back to browser)
const server = http.createServer((req, res) => {
   // Setting the response headers
   // 200 = OK (successful response)
   // 'Content-Type': 'text/plain' means we are sending plain text
   res.writeHead(200, {'Content-Type': 'text/plain'});
   // Sending the response message to the browser
   res.end('Hello, this is my first Node.js Server!');
});
```

```
// Making the server listen on port 3000
server.listen(3000, () => {

    // Message printed in the terminal to tell server is running
    console.log('Server is running on http://localhost:3000');
});
```



Open any Browser and type and search http://localhost:3000



Q1(b) Experiment: Routing and Handling HTTP Methods (GET, POST, PUT, DELETE) in Node.js

(using only the built-in http module — no Express)

What is Routing?

Routing means deciding **what response to give** based on the **URL** the user requests.

Example:

- / → Home page
- /about → About page
- /contact → Contact page

What are HTTP Methods?

These are different types of requests sent by the client:

Method	Meaning	Example
GET	Only to fetch data	Open website
POST	Send data to server	Login form
PUT	Update data	Edit profile
DELETE	Delete data	Remove product

2. Sample Code (Simple Routing + GET & POST)

Create file **server.js**:

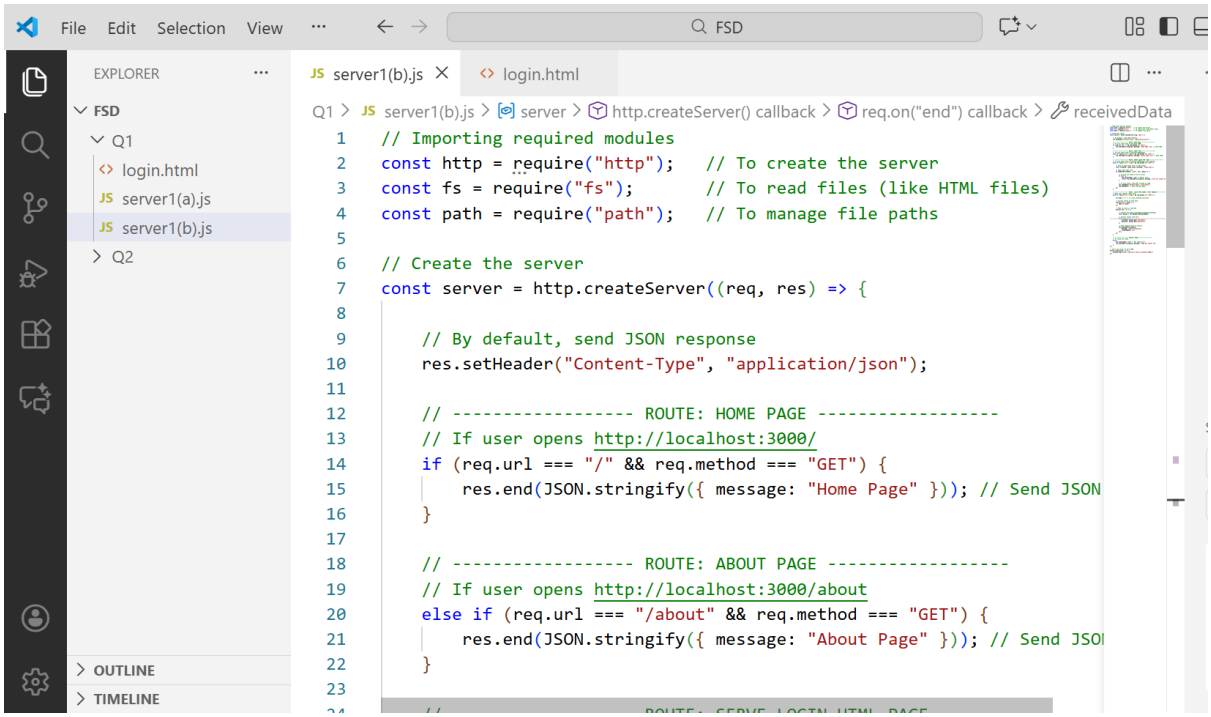
```
// Importing required modules
const http = require("http");   // To create the server
const fs = require("fs");       // To read files (like HTML files)
const path = require("path");   // To manage file paths

// Create the server
const server = http.createServer((req, res) => {

    // By default, send JSON response
    res.setHeader("Content-Type", "application/json");

    // ------------------ ROUTE: HOME PAGE ------------------
    // If user opens http://localhost:3000/
    if (req.url === "/" && req.method === "GET") {
        res.end(JSON.stringify({ message: "Home Page" })); // Send JSON
    }

    // ------------------ ROUTE: ABOUT PAGE ------------------
    // If user opens http://localhost:3000/about
    else if (req.url === "/about" && req.method === "GET") {
        res.end(JSON.stringify({ message: "About Page" })); // Send JSON
    }

    // ------------------ ROUTE: SERVE LOGIN HTML PAGE ------------------
    // If user opens http://localhost:3000/login (GET request)
    else if (req.url === "/login" && req.method === "GET") {

        // Path to login.html file in same folder
        const filePath = path.join(__dirname, "login.html");

        // Read login.html file
        fs.readFile(filePath, "utf8", (err, data) => {
```

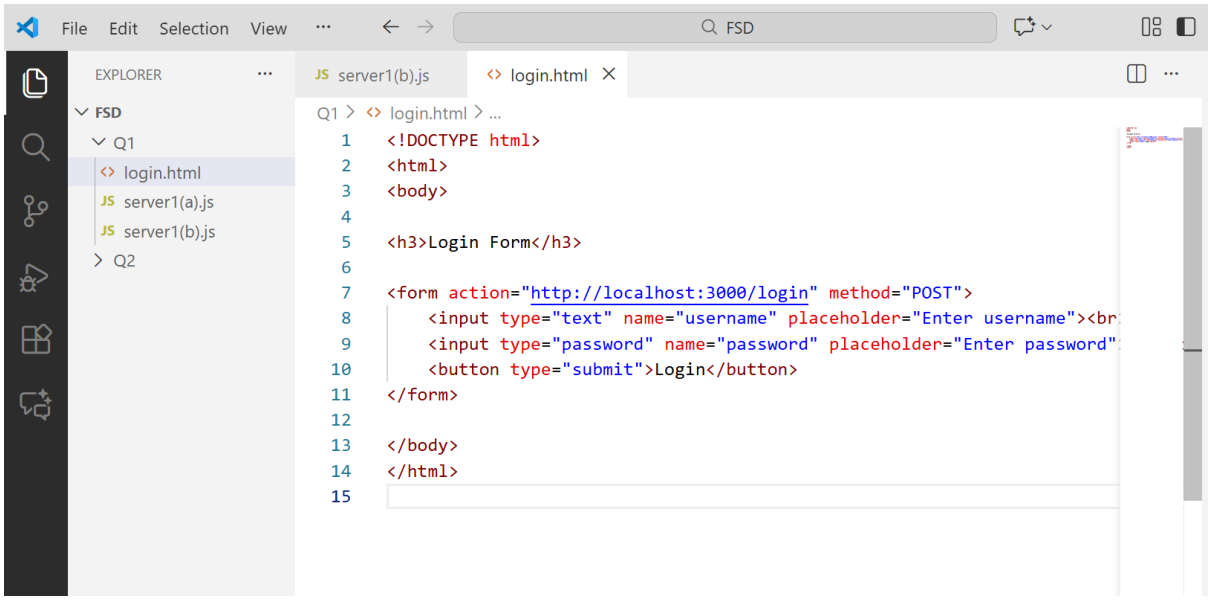
```
        // If file not found or error occurs
        if (err) {
            res.statusCode = 500; // Server error
            return res.end(JSON.stringify({ message: "File not found" }));
        }

        // If file found, send HTML instead of JSON
        res.setHeader("Content-Type", "text/html");
        res.end(data); // Send the HTML page
    });
}

// ------------------ ROUTE: LOGIN FORM SUBMIT (POST REQUEST) ------------------
// When user submits login form
else if (req.url === "/login" && req.method === "POST") {

    let body = ""; // To store incoming form data

    // Collect chunks of form data
    req.on("data", chunk => {
        body += chunk;
    });

    // When all data is received
    req.on("end", () => {

        // Convert form data (username=vijay&password=1234)
        const params = new URLSearchParams(body);

        // Extract values from form
        const data = {
            username: params.get("username"),
            password: params.get("password")
        };

        // Send response back to browser
        res.end(JSON.stringify({
            message: "Login Successful",
            receivedData: data
        }));
    });
}
```

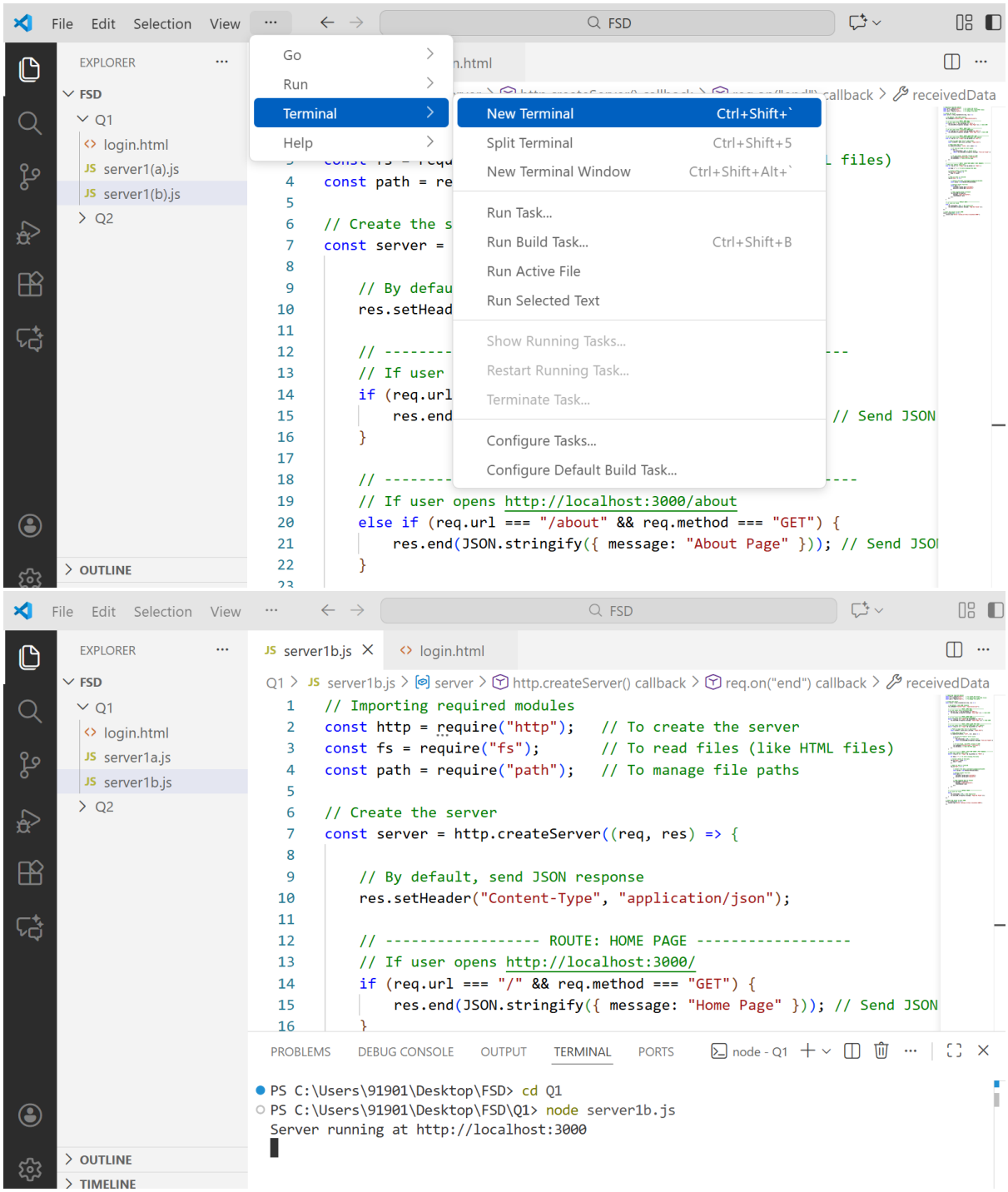
```
    // ----------------- DEFAULT ROUTE -----------------
    // If route not found
    else {
        res.statusCode = 404; // Not found error
        res.end(JSON.stringify({ message: "Page Not Found" }));
    }
});

// Start the server on port 3000
server.listen(3000, () => {
    console.log("Server running at http://localhost:3000");
});
```

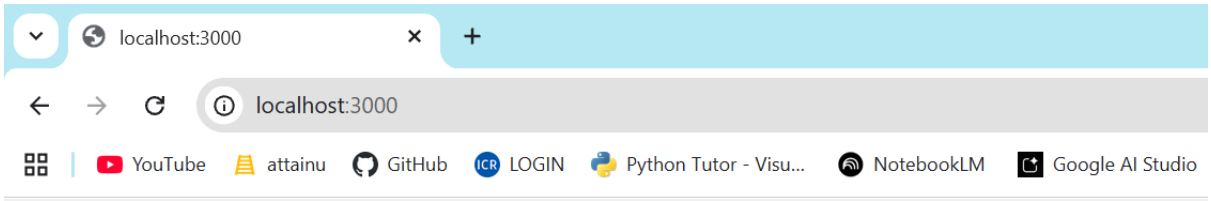
In Same folder Create login.html file

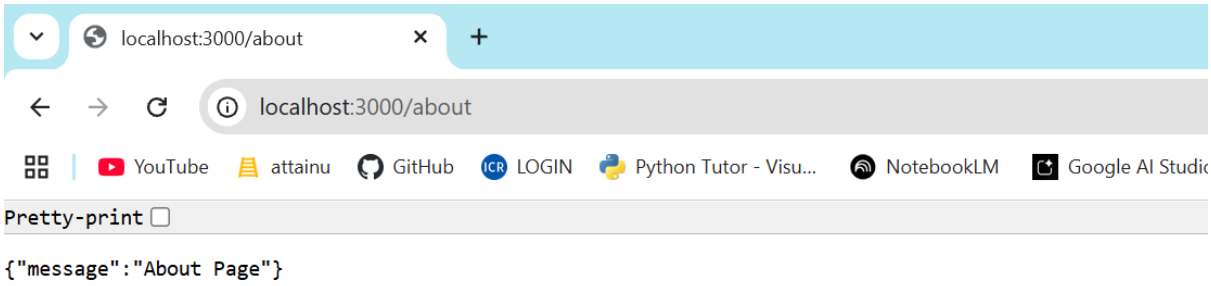
```
<!DOCTYPE html>
<html>
<body>

<h3>Login Form</h3>

<form action="http://localhost:3000/login" method="POST">
    <input type="text" name="username" placeholder="Enter username"><br><br>
    <input type="password" name="password" placeholder="Enter password"><br><br>
    <button type="submit">Login</button>
</form>

</body>
</html>
```

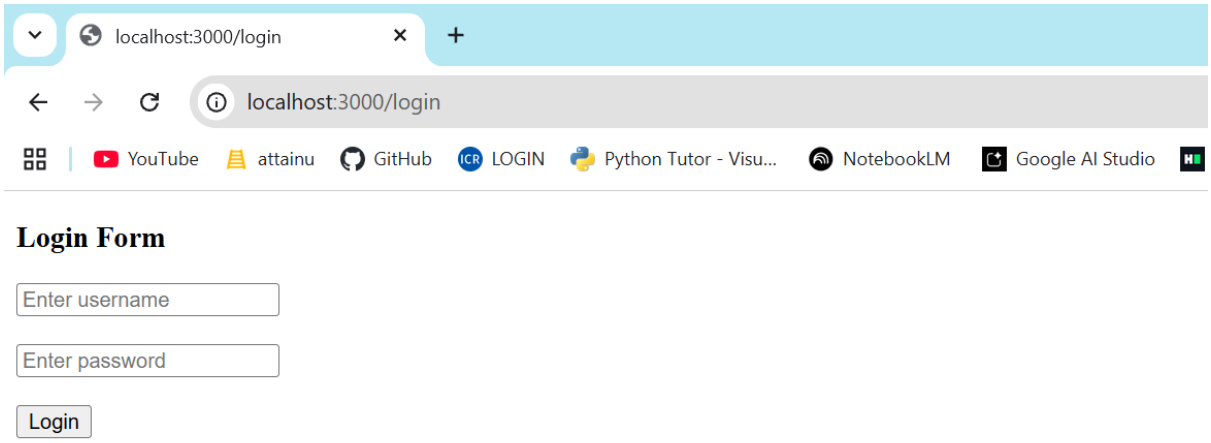

File Edit Selection View
FSD
EXPLORER
FSD
Q1
login.html
server1(a).js
server1(b).js
Q2
OUTLINE
TIMELINE
server1(b).js
login.html
Q1 > server1(b).js > server > http.createServer() callback > req.on("end") callback > receivedData
// Importing required modules
const http = require("http"); // To create the server
const fs = require("fs"); // To read files (like HTML files)
const path = require("path"); // To manage file paths
// Create the server
const server = http.createServer((req, res) => {
// By default, send JSON response
res.setHeader("Content-Type", "application/json");
// ROUTE: HOME PAGE
// If user opens http://localhost:3000/
if (req.url === "/" && req.method === "GET") {
res.end(JSON.stringify({ message: "Home Page" })); // Send JSON
}
// ROUTE: ABOUT PAGE
// If user opens http://localhost:3000/about
else if (req.url === "/about" && req.method === "GET") {
res.end(JSON.stringify({ message: "About Page" })); // Send JSO
}


File Edit Selection View
FSD
EXPLORER
FSD
Q1
login.html
server1(a).js
server1(b).js
Q2
server1(b).js
login.html
Q1 > login.html > ...
<!DOCTYPE html>
<html>
<body>
<h3>Login Form</h3>
<form action="http://localhost:3000/login" method="POST">
<input type="text" name="username" placeholder="Enter username"><br
<input type="password" name="password" placeholder="Enter password"
<button type="submit">Login</button>
</form>
</body>
</html>

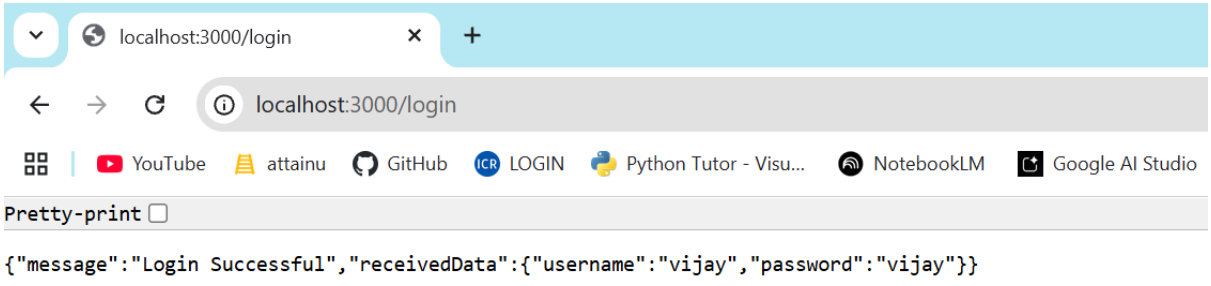

File Edit Selection View ...
FSD
Go
Run
Terminal
Help
New Terminal Ctrl+Shift+`
Split Terminal Ctrl+Shift+5
New Terminal Window Ctrl+Shift+Alt+`
Run Task...
Run Build Task... Ctrl+Shift+B
Run Active File
Run Selected Text
Show Running Tasks...
Restart Running Task...
Terminate Task...
Configure Tasks...
Configure Default Build Task...
EXPLORER
FSD
Q1
login.html
server1(a).js
server1(b).js
Q2
OUTLINE
// If user opens http://localhost:3000/about
else if (req.url === "/about" && req.method === "GET") {
res.end(JSON.stringify({ message: "About Page" })); // Send JSO
}
File Edit Selection View ...
FSD
server1b.js
login.html
Q1 > server1b.js > server > http.createServer() callback > req.on("end") callback > receivedData
// Importing required modules
const http = require("http"); // To create the server
const fs = require("fs"); // To read files (like HTML files)
const path = require("path"); // To manage file paths
// Create the server
const server = http.createServer((req, res) => {
// By default, send JSON response
res.setHeader("Content-Type", "application/json");
// ------------------ ROUTE: HOME PAGE ------------------
// If user opens http://localhost:3000/
if (req.url === "/" && req.method === "GET") {
res.end(JSON.stringify({ message: "Home Page" })); // Send JSON
}
EXPLORER
FSD
Q1
login.html
server1a.js
server1b.js
Q2
OUTLINE
TIMELINE
PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS node - Q1
PS C:\Users\91901\Desktop\FSD> cd Q1
PS C:\Users\91901\Desktop\FSD\Q1> node server1b.js
Server running at http://localhost:3000







After given user name and password Enter Login the we will get



Q2. Using Express.js (CRUD Operations)

- Create a server using Express.js
- Implement CRUD: **Create, Read, Update, Delete**

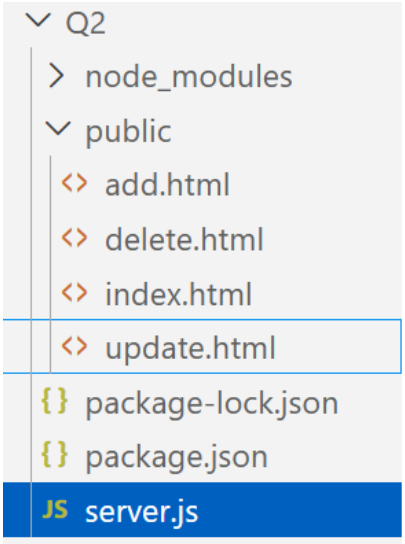
1. What is Express.js (Easy Explanation)

Express.js is a **framework** for Node.js.
It makes server creation **very easy** compared to raw http module.
Benefits:

- Simple routing
- Cleaner code
- Built-in middleware
- Easy to build APIs (backend apps)

File Structure or Format

```
Q2/
│
├── server.js
│
└── public/
    ├── index.html
    ├── delete.html
    ├── add.html
    └── update.html
```



3. Create File: server.js (Express CRUD API)

Here is the complete CRUD code:

server.js

```
const express = require('express');
const app = express();
app.use(express.json());              // To read JSON data
app.use(express.static("public"));    // To serve HTML/CSS/JS files

// Dummy users data
let users = [
    { id: 1, name: "Vijay" },
    { id: 2, name: "Babu" }
];
```

```
// ---------------- READ (GET) ----------------
app.get('/users', (req, res) => {
    res.json(users);
});

// ---------------- CREATE (POST) ----------------
app.post('/users', (req, res) => {
    const newUser = req.body;   // Read new user
    users.push(newUser);        // Add to array
    res.json({ message: "User added", users });
});

// ---------------- UPDATE (PUT) ----------------
app.put('/users/:id', (req, res) => {
    const id = parseInt(req.params.id);
    const updatedUser = req.body;

    users = users.map(u =>
        u.id === id ? { ...u, ...updatedUser } : u
    );

    res.json({ message: "User updated", users });
});

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

// Start server
app.listen(3000, () => {
    console.log("Server running at http://localhost:3000");
});
```

public/index.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Users List</title>
```

```
</head>
<body>
    <h1>All Users</h1>

    <a href="add.html">Add User</a><br>
    <a href="update.html">Update User</a><br>
    <a href="delete.html">Delete User</a><br><br>

    <div id="users"></div>

    <script>
        fetch('/api/users')
            .then(res => res.json())
            .then(data => {
                document.getElementById("users").innerHTML =
                    data.map(u => `<p>${u.id}. ${u.name}</p>`).join("");
            });
    </script>
</body>
</html>
```

Public/add.html

```
<!DOCTYPE html>
<html>
<head><title>Add User</title></head>
<body>
    <h1>Add User</h1>
    <form action="/api/users" method="POST">
        Name: <input type="text" name="name">
        <button type="submit">Add</button>
    </form>

    <a href="index.html">Back</a>
</body>
</html>
```

Public/delete.html

```
<!DOCTYPE html>
<html>
<head><title>Delete User</title></head>
<body>
```

```
    <h1>Delete User</h1>
    <form action="/api/delete" method="POST">
        User ID: <input type="number" name="id"><br><br>
        <button type="submit">Delete</button>
    </form>
    <a href="index.html">Back</a>
</body>
</html>
```

Public/update.html

```
<!DOCTYPE html>
<html>
<head><title>Update User</title></head>
<body>
    <h1>Update User</h1>
    <form action="/api/update" method="POST">
        User ID: <input type="number" name="id"><br><br>
        New Name: <input type="text" name="name"><br><br>
        <button type="submit">Update</button>
    </form>
    <a href="index.html">Back</a>
</body>
</html>
```

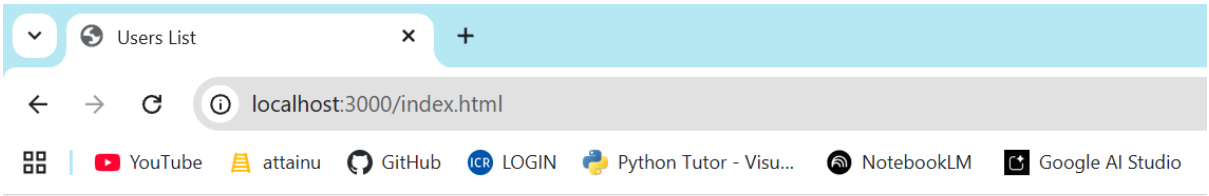
How to Run

1. Create **Q2** folder
2. Inside it create server.js
3. Make **public** folder → add the 4 HTML files
4. Install Express:
   `npm install express`
5. Run server:
   `node server.js`
6. Open browser:
   http://localhost:3000/index.html

What You Learned ?

You now understand:

Express.js
CRUD operations
HTML forms
Serving static files
Simple routing
Using POST for Create/Update/Delete

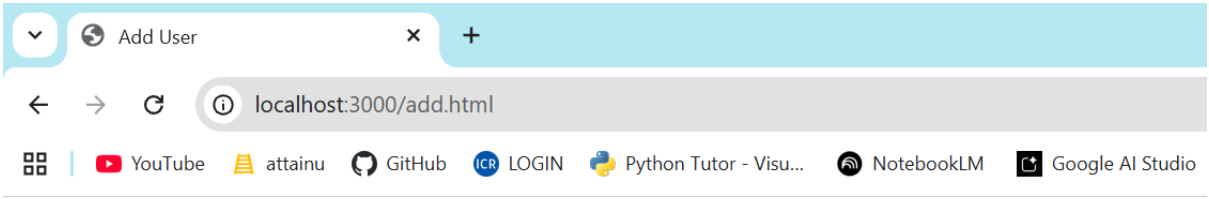


All Users

Add User
Update User
Delete User

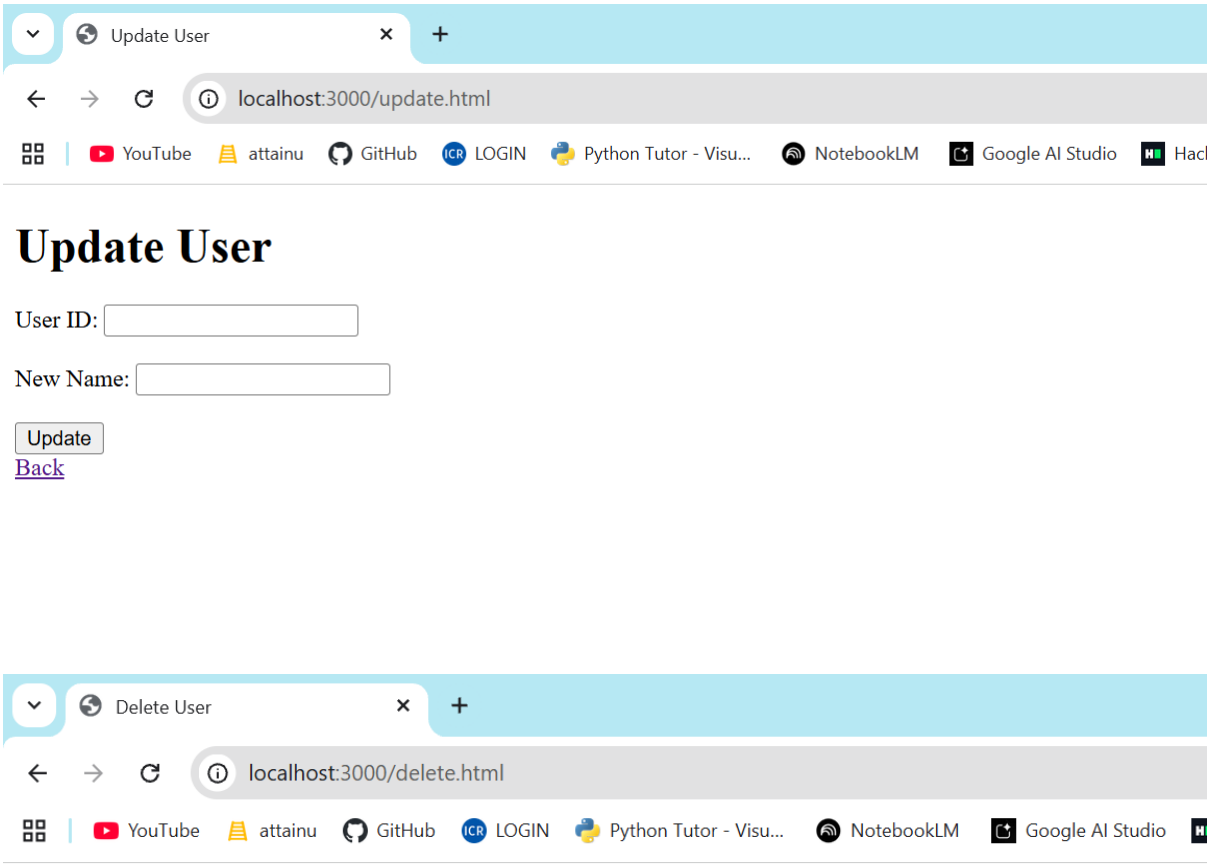
1. Vijay

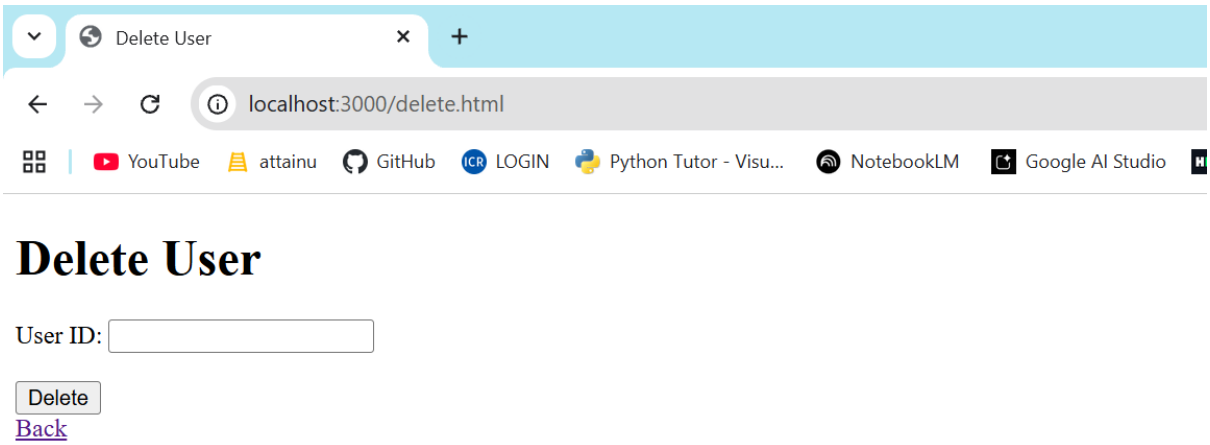
2. Babu



Add User

Name: Add
Back


Update User
localhost:3000/update.html
YouTube
attainu
GitHub
LOGIN
Python Tutor - Visu...
NotebookLM
Google AI Studio
Update User
User ID:
New Name:
Update
Back


Delete User
localhost:3000/delete.html
YouTube
attainu
GitHub
LOGIN
Python Tutor - Visu...
NotebookLM
Google AI Studio
Delete User
User ID:
Delete
Back