LAB 6 – EJS Templating with Express

Objective:

By the end of this lab, students can:

- Build an Express server using **TypeScript** (CommonJS)
- Configure **EJS** as a server-side template engine
- Render pages using **res.render()** and pass data from routes to views
- Use EJS syntax: **output**, **conditionals**, and **loops**
- Reuse UI with **partials** (header/footer/navbar)
- Serve static assets (CSS/JS) from the **public/** folder

Lab instruction

- The LAB 6 instruction and lab resources are posted on CMU Mango: LAB6 – EJS Templating with Express
- There are 2 assignments according to the LAB 6 sheet posted on the channel.
- The LAB 6 is worth 20 points in total.
- Score criteria: full point (for output correct); -1 (for output does not correct); -1 (for not follow problem constraint)
- **Assignment Submission:**
    - Upload your solutions to CMU Mango assignments. The submission later than the due date will get 50% off your score. At the close date, you cannot submit your assignment to the system.
    - Be prompt for TA calling to verify your work on your computer.

Requirements (Important)

- Check your Node.js version **node -v**
- Keep your project in **CommonJS** (tsconfig: "module": "commonjs")
- Do not commit or submit **node_modules/**
- Keep file/folder names exactly as specified (**views/**, **views/partials/**, **public/**)
- Your project must run in both modes:
  - npm run dev
  - npm run build then npm start

Problem 1: EJS Basics Project (10 points)

Your task is to create a small Express web app that renders pages using EJS.

Part A — Project setup (3 pts)

1. Create a new folder called **Lab06/EJSBasics** inside your directory.
2. Initialize NPM in your project folder.
3. Install dependencies.
4. Create the required folders and files.

Command:

```
npm init -y
npm i express ejs
npm i -D typescript ts-node-dev @types/express @types/node
```

Create tsconfig.json and set these key fields:

```
{
  "compilerOptions": {
    "target": "ES2020",
    "module": "commonjs",
    "types": ["node"],
    "rootDir": "./src",
    "outDir": "./dist",
    "strict": true,
    "esModuleInterop": true,
    "moduleResolution": "node",
    "skipLibCheck": true
  }
}
```

Add scripts to package.json:

```
{
  "scripts": {
    "dev": "ts-node-dev --respawn --transpile-only src/app.ts",
    "build": "tsc",
    "start": "node dist/app.js"
  }
}
```

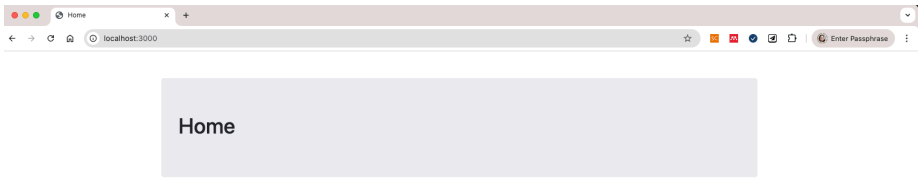
Part B — Express + EJS setup (3 pts)

- Work in src/app.ts

```
import express, { Request, Response } from "express";
import path from "path";

const app = express();
const PORT = 3000;

app.set("view engine", "ejs");
app.set("views", path.join(__dirname, "..", "views"));
app.use(express.static(path.join(__dirname, "..", "public")));

app.get("/", (req: Request, res: Response) => {
  res.render("index", { title: "Home", activePage: "home" });
});

app.listen(PORT, () => {
  console.log(`Server running at http://localhost:${PORT}`);
});
```

- Create views/index.ejs and make sure it render title. Example:

  `<h1><%= title %></h1>`
- Run the server with npm run dev, then open http://localhost:3000/
- Expected: Home page loads from views/index.ejs



- You may style your **page** using CSS in the public/style.css or Bootstrap (if you use Bootstrap, include its CDN in your header).

Part C — EJS syntax (2 pts)

1. **Create a route /syntax** that sends an array to the view.

```
app.get("/syntax", (req: Request, res: Response) => {
  const topics = ["<%= %> output", "<% if %>", "<% forEach %>", "partials"];
  res.render("syntax", { title: "EJS Syntax", activePage: "syntax", topics
});
});
```

2. **In views/syntax.ejs**, show a loop rendering topics and a conditional message if topics.length >= 4.

```
<%- include("partials/header", { title, activePage }) %>

<h1><%= title %></h1>

<ul>
  <% topics.forEach(t => { %>
    <li><%= t %></li>
  <% }) %>
</ul>

<% if (topics.length >= 4) { %>
  <p> You have enough topics for today.</p>
<% } else { %>
  <p> Add more topics.</p>
<% } %>

<%- include("partials/footer") %>
```

Part D — Partials (2 pts)

- **Create header.ejs and footer.ejs in views/partials/** and include them in every page.
- Your header must include a navbar with 3 links: Home (/), Syntax (/syntax), Students (/students). Highlight the active page using activePage. **Use**

```
<nav>
   <a href="/" class="<%= activePage === 'home' ? 'active' : '' %>">Home</a> |
   <a href="/syntax" class="<%= activePage === 'syntax' ? 'active' : '' %>">Syntax</a> |
   <a href="/students" class="<%= activePage === 'students' ? 'active' : '' %>">Students</a>
 </nav>
```

Add 1 CSS rule in the public/style.css

```
nav a.active { font-weight: 700; text-decoration: underline; }
```

- The follow is an example of the expected result.

- Make sure each route passes activePage (e.g., "home", "syntax", "students") when calling res.render().
- Capture screenshots: / and /syntax

Problem 2: Students Directory (10 points)

Create a Students Directory feature with list + detail pages.

Part A — Students list page (3 pts)

In src/app.ts, create student data and a route /students:

```
type Student = { id: number; name: string; major: string };

const students: Student[] = [
  { id: 1, name: "Alice", major: "Software Engineering" },
  { id: 2, name: "Bob", major: "Data Science" },
  { id: 3, name: "Chen", major: "UX/UI" }
];

app.get("/students", (req: Request, res: Response) => {
  res.render("students", { title: "Students", activePage: "students",
students });
});
```

Create views/students.ejs (must include a loop + links to detail pages):

```
<%- include("partials/header", { title, activePage }) %>

<h1><%= title %></h1>

<ul>
  <% students.forEach(s => { %>
    <li>
      <a href="/students/<%= s.id %>"><%= s.name %></a>
      — <%= s.major %>
    </li>
  <% }) %>
</ul>

<%- include("partials/footer") %>
```

Part B — Student detail page + not found (3 pts)

Create route /students/:id:

```
app.get("/students/:id", (req: Request, res: Response) => {
  const id = Number(req.params.id);
  const student = students.find(s => s.id === id);

  res.render("student-detail", {
    title: "Student Detail",
    activePage: "students",
    student
  });
});
```

Create views/student-detail.ejs (must show not-found case):

```
<%- include("partials/header", { title, activePage }) %>

<h1><%= title %></h1>

<% if (!student) { %>
  <p>❌ Student not found.</p>
  <p><a href="/students">Back to list</a></p>
<% } else { %>
  <h2><%= student.name %></h2>
  <p><strong>ID:</strong> <%= student.id %></p>
  <p><strong>Major:</strong> <%= student.major %></p>
  <p><a href="/students">Back to list</a></p>
<% } %>

<%- include("partials/footer") %>
```

Part C — Styling + static assets (2 pts)

- Create public/style.css and link it in header.ejs. Your pages must have basic styling (font, spacing, navbar).
- Make sure CSS loads correctly (no 404).

Part D — Build and run compiled version (2 pts)

Run these commands:

```
npm run build
npm start
```

Capture screenshots: /students, /students/2, /students/999

Submission

Submit one GitHub repository link containing your LAB 6 project.

Your repository must include:

- Source code (src/, views/, public/, package.json, tsconfig.json)
- Screenshots (files in repo or in README)
- A file named answers.md
- Do not include: node_modules/
- **answers.md questions:**

  1. In one sentence: What does res.render(view, data) do?
  2. What is the difference between <%= %> and <%- %>?
  3. Where does Express look for EJS templates (folder path)?