**Task 2 Build a Simple HTTP Server (HTTP Server Task)**

In this task, students will build a simple HTTP server that responds to basic GET, POST, and PUT requests.

**Task: Build an HTTP Server**

* **Objective**: Create a Java HTTP server that listens on a specific port and responds to GET, POST, and PUT requests.
* **Tools Needed**: Do not use the HttpServer class that comes with the JDK and is old and depricated or use an external framework such as **Jetty** or (in future we will use **Spring Boot)**.

**Instructions:**

1. **Create an HTTP Server**:
   * Write a simple HTTP server that listens on port 8000.
   * The server should handle three types of requests:
     + **GET**: Respond with a "Hello, World!" message.
     + **POST**: Accept a JSON payload and respond with a message that includes the received data.
     + **PUT**: Accept a JSON payload, update some internal state (like a variable), and return the updated value.
2. **Create Endpoints**:
   * /hello – Respond with a "Hello, World!" message.
   * /data (POST) – Accept JSON data (e.g., { "name": "John" }) and respond with Received: { "name": "John" }.
   * /update (PUT) – Update internal data based on the received payload and respond with the updated state (e.g., {"status": "Updated"}).
3. **Implement Error Handling**:
   * Handle invalid HTTP methods by responding with 405 Method Not Allowed.
   * Respond with 404 Not Found for invalid paths.

**Task 3 Upload and Save a File (HTTP UPLOAD Task)**

Implement File Upload Handling

* Objective: Create an endpoint /upload that allows clients to upload files to the server.
* Tools Needed: Use Java’s HttpServer class or another lightweight HTTP framework.

Instructions:

1. Create the /upload Endpoint:
   * Accept multipart form data or a binary file in the body of the POST request.
   * Save the uploaded file to a directory on the server (e.g., /uploads).
   * Respond with a success message once the file is saved.
2. Add Validation:
   * Ensure only specific file types are allowed (e.g., txt, jpg, png).
   * Return an error for unsupported file types.
3. Respond with Download Link:
   * After uploading the file, provide a download link (e.g., /files/{filename}) that allows users to download the uploaded file.