CADT
IDT វិទ្យាស្ថានបច្ចេកវិទ្យាឌីជីថល
Institute of Digital Technology

INSTITUTE OF DIGITAL TECHNOLOGY

Library Management Report

[Course: Database Analysis and Design]

Lecturer: Kim Ang Kheang
Name: Tha Bunnaroth
ID: IDTB100138
Group: 01

EXERCISE-1:

1. What error message do you see in the terminal when you access http://localhost:3000? What line of code causes it?
It will throw the runtime error because of res.endd(); is undefined.

2. What is the purpose of res.write() and how is it different from res.end()?
- Purpose:
  - res.write() : sends a chunk of data to client and it does not end the response.
  - res.end() : Ends the response, optionally can send the last piece of data.
- Different:
  - res.write() ; can be called multiple times.
  - res.end() : must be called once to finalize the response.

3. What do you think will happen if res.end() is not called at all?
- The browser will hang indefinitely, waiting for the server to finish the response.
- No data is considered complete and flushed.
- May eventually timeout.
- The server will not close the connection, potentially leaking resources.
- Display nothing on web page

4. Why do we use http.createServer() instead of just calling a function directly?
- http.createServer() sets up an actual HTTP server, which:
  - Listens for HTTP requests
  - Handles TCP connections
  - Invokes your callback for each incoming request
- Just calling a function directly would not:
  - Bind to a network port
  - Accept HTTP requests
  - Act as a server

5. How can the server be made more resilient to such errors during development?
- Improve resilient
  - Use error handling
  - Always use res.end() to finalize the response
  - Log errors to help diagnose issues
  - Use tools like Nodemon for automatic restarts on crash.

EXERCISE-2:

1. What happens when you visit a URL that doesn't match any of the three defined?
Webpage will display 404 error with the plain text message

2. Why do we check both the req.url and req.method?
Because HTTP defines multiple methods, and you may want to handle the same URL differently depending on the method.
Example: GET /products will return products list
POST /products will add a new product

3. What MIME type (Content-Type) do you set when returning HTML instead of plain text?

- For HTML content ("Content-Type": "text/html")
- For Plain text ("Content-Type": "text/plain")

4. How might this routing logic become harder to manage as routes grow?
As the number of routes increases:

- You'll end up with a long chain of if-else statements
- Code becomes hard to read, hard to maintain, and error-prone
- Difficult to add middlewares
- Repeating logic can't be reused easily

5. What benefits might a framework offer to simplify this logic?
Frameworks like Express.js provide:

- Cleaner syntax for routing
- Middleware support
- Modular routing – you can separate routes into files
- Automatic headers, better error handling, more flexibility
- Built-in helpers like res.json(), res.status()

EXERCISE-3:

1. Why do we listen for data and end events when handling POST?
We listen for the data event to collect chunks of the request body as they arrive. The end event tells us when all the data has been received so we can process the full body correctly.

2. What would happen if we didn't buffer the body correctly?
If we didn’t buffer the body correctly, we might only get part of the data. This can lead to incomplete or corrupted information being processed or saved.

3. What is the format of form submissions when using the default browser form POST?
The default format is application/x-www-form-urlencoded, which sends the form data as key-value pairs in the request body, like name=John.

4. Why do we use fs.appendFile instead of fs.writeFile?
fs.appendFile adds new data to the end of the file without removing the existing content. This is useful for keeping all submissions, one per line, instead of overwriting previous data.

5. How could this be improved or made more secure?

- Validate and sanitize user input.
- Use HTTPS to secure communication.
- Limit the size of incoming data to prevent abuse.
- Use a proper framework like Express for more secure and maintainable handling.
- Handle file access errors properly.