Web Application Framework (WAF) - Fall 2025
Lab Task 2 (Due: 23rd September 2025)

Part 1: Data Generation

1. **Generate Employee Data:**
    a. Visit **www.mockaroo.com** and create a dataset containing **at least 7,000 employee records**.
    b. Each record must include the following fields:
        - id (integer, unique, auto-incrementing)
        - fullName (string, e.g., "John Doe")
        - email (string, e.g., "xyz.abc@example.com")
        - department (string, e.g., "Engineering")
        - ipAddress (IPv4 address, e.g., "192.168.1.45")
        - Add a custom field called company (string, e.g., "TechCorp") before generating the data.
    c. Save the generated data as employees.json in your project directory.

2. **IP Address Classification:**

- Categorize employees based on their ipAddress into the following classes:
    - **Class A**: IP addresses starting with 1-126 (e.g., "10.0.0.1")
    - **Class B**: IP addresses starting with 128-191 (e.g., "172.16.0.1")
    - **Class C**: IP addresses starting with 192-223 (e.g., "192.168.0.1")
    - **Class D**: IP addresses starting with 224-239 (e.g., "224.0.0.1")
    - **Class E**: IP addresses starting with 240-255 (e.g., "240.0.0.1")
- Create the following routes to separate employees by IP class and return their data as JSON:
    - /api/employees/classA
    - /api/employees/classB
    - /api/employees/classC
    - /api/employees/classD
    - /api/employees/classE
- Use the fs module to read from employees.json and filter the data for each route.

Part 2: Functional Requirements

Your application must perform the following tasks programmatically (add these as functions in your code):

1. **Delete Class E Employees:**
    - Write a function that removes all employees with Class E IP addresses from employees.json and saves the updated file.
2. **Update Class D Employees:**
    - Write a function that updates the company field to "XYZ Corp" for all employees with Class D IP addresses and saves the updated file.
3. **Insert New Employees:**
    - Generate a separate .json file (e.g., new_employees.json) with **at least 100 new employee records** using Mockaroo (same fields as above). Write a function to append these records to employees.json, ensuring no duplicate id values (reassign ids if necessary).
4. **Retrieve Class C Employees:**
    - Write a function that retrieves all employees with Class C IP addresses and returns them as a JSON array.

Part 3: Web Server and Routes

Create a web server using the NodeJs http module. Create four to five routes /, /users, /products, /display, and /books. Maintain a log by appending to a file “log.txt” using the fs module every time a URL is visited. This file should store information such as serial number, time, date, URL, and number of query parameters for every URL. Create three plain text files named “products.txt”, “users.txt”, and “books.txt” using the fs module. Get the information related to a specific file to be appended from the query parameters. The file “products.txt” should get information from the query parameters of route /products, and it should contain id, product title, and product price. Similarly, the files “users.txt” and “books.txt” should be appended from the query parameters of /users and /books routes, respectively. The file “users.txt” should contain id, user name, age, city, and university. The file “books.txt” should contain id, book title, edition, year of publication, and press name. Hit at least 5 URLs having query parameters such as the following for each of the above-mentioned routes for testing:

/

/users?id=24&name=Abdullah&age=60&city=Islamabad&uni=QAU

/products?id=89&title=Samsung&price=75K

/books?id=19&title=AlgorithmDesignAndApplications&edition=3&2019&press=Wiley

Deliverables:

1. Submit your **Node.js task files**, including:
    - index.js (or equivalent) with all routes and logic.
    - employees.json (initial dataset).
    - new_employees.json (additional dataset).
2. Include a **README.md** file with:
    - Instructions to run your tasks (e.g., npm install, node index.js).
3. Ensure your code is well-commented and handles errors (e.g., file not found, invalid JSON).
    - Ensure your application can handle edge cases (e.g., empty files, invalid IP addresses).
4. Submit the compressed folder with the name “yourRollNo_yourName_Task2” on fs to the folder WAF-F25-T2. The deadline for this task is 23/9/25 till 15:30.