Web Application Framework (WAF) – Fall 2024
Lab Task (25th October 2024)

Part-1

1. Create a folder named as "your_name_waf_lab_28sep"
2. Go inside the folder and create a NodeJS file "part_1.js".
3. Inside the part_1.js file, write code to programmatically create a folder named as "part_1" in the same directory with the .js file (your_name_waf_lab_28sep).
4. Write a code statement to create a file named "test.txt".
5. Write a code statement to write your registration number inside the "test.txt" file.
6. Write a code statement to read the data from the "part_1.txt" file (your registration number) and display it on the console.
7. Write code to overwrite your registration number in the "test.txt" with the last four digits of your registration number.
8. Repeat step 6.
9. Erase data from the file "test.txt"
10. Repeat step 5.

Part-2

You are provided with a file "IPs.txt". The file contains a list of IP addresses. Your task is to:
Create three files named "A.txt", "B.txt" and "C.txt"
Read IP addresses from the given file.
Determine class (A/B/C) for each of the given IP addresses and copy it into the respective file (A.txt/B.txt/C.txt).

Part-3

Your program should create and call six to seven functions. Name your functions as signup, sendVerificationCode, signin, getData, checkEmail, composeEmail, and sendEmail. Create some delay in each function using the asynchronous method setTimeout function. These functions should have some console output messages displaying these tasks completion messages. Call these functions in the following order with the delay mentioned against each function and the last line of code after these function calls should be console.log("All tasks completed…..")

signup – delay of 2000 milliseconds = 2 seconds
sendVerificationCode – delay of 4000 milliseconds = 4 seconds
signin – delay of 3500 milliseconds = 3.5 seconds
getData – delay of 4500 milliseconds = 4.5 seconds`
checkEmail – delay of 1500 milliseconds = 1.5 seconds
composeEmail – delay of 2000 milliseconds = 2 seconds
sendEmail – delay of 3000 milliseconds = 3 seconds

i. First, call these functions in above mentioned order with the given delays and show the order of execution. See what order is followed.
ii. You should now use callbacks to execute the functions in the given order so that the order of events is intact. Again, call these functions in above mentioned order with the given delays and show the order of execution. See what order is followed.

iii. Now you should use promises to do part ii. Again, call these functions in above mentioned order with the given delays and show the order of execution. See what order is followed.

iv. Now use async wait to do part ii or part iii. Again, call these functions in above mentioned order with the given delays and show the order of execution. See what order is followed.

Part-4

Create a web server using Node JS http module. Create four to five routes /, /users, /products, /display, and /books. Maintain a log by appending a file “log.txt” using fs module everytime 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 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 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 following for each bove 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

Note

i. You can use Node official documentation.

ii. Make a folder and name it with your roll#_name. Put four folders inside this folder and name it with your roll#_name_task1_part#. Place your code accordingly. Compress the main folder and submit the zip or rar file. Name your zip or rar file as yourRoll#_yourName#_Task_1.