IPv4 Superhub_For Engineer Questions

Questions

P/s: You can use Java, Javascript language

1. Given an array of **n** elements that contains elements from **0** to **n-1**, with any of these numbers appearing any number of times. Find these repeating numbers.

Example:

```
Input: n=7 , array[]={1, 2, 3, 6, 3, 6, 1}
Output: 1, 3, 6
Explanation: The numbers 1 , 3 and 6 appear more than once in the array.

Input : n = 5 and array[] = {1, 2, 3, 4 ,3}
Output: 3
Explanation: The number 3 appears more than once in the array.
```

Answer:

```
function findRepeating(arr) {
    let freq = new Array(arr.length).fill(0);
    let repeating = [];
    for (let num of arr) {
        freq[num]++;
        if (freq[num] === 2) {
            repeating.push(num);
        }
    }
    return repeating;
}
```

2. Given two strings **s1** and **s2** consisting of **lowercase** characters, the task is to check whether the two given strings are **anagrams** of each other or not. An anagram of a string is another string that contains the same characters, only the order of characters can be different.

Examples:

Input: *s1 = “geeks” s2 = “kseeg”*
Output: *true*
Explanation: *Both the string have same characters with same frequency. So, they are anagrams.*

Input: *s1 = “allergy” s2 = “allergic”*
Output: *false*

Explanation: *Characters in both the strings are not same. s1 has extra character* ***'y'*** *and s2 has extra characters 'i' and 'c', so they are not anagrams.*

Input: *s1 = "g", s2 = "g"*
Output: *true*
Explanation: *Characters in both the strings are same, so they are anagrams.*

Answer:
```
function areAnagrams(s1, s2) {
    if (s1.length !== s2.length) return false;
    const sortedS1 = s1.split('').sort().join('');
    const sortedS2 = s2.split('').sort().join('');
    return sortedS1 === sortedS2;
}
```

3. Explain the difference between GET, POST, PUT, PATCH requests.

Answer:
These HTTP methods are used in RESTful APIs for communication between the client and server. However there are several differences:

Method	**Purpose**	**Has a Body?**	**Usage Example**
GET	Retrieve data from a server	✗ No	Fetching a list of users
POST	Send data to create a new resource	✓ Yes	Creating a new user
PUT	Update/replace entirely an existing resource	✓ Yes	Updating a user's profile (full update)
PATCH	Partially update an existing resource	✓ Yes	Changing only the email of a user

4. Describe a recent project you worked on. What were the challenges, and how did you solve them?
I recently worked on an eLearning platform project which was designed to help users learn courses and takes exam within the system. If they pass the exam they can has the privilege to use the BBQ system, the system can create a proposal about product insurance for customer. The platform integrated SCORM for structural learning content and used Survey Technologies for assessments and feedback collection.
Challenges and solutions:
- SCORM and Survey integration: they are new technologies for me, so I need several times to learn and code with them. I read through out the documentation and best practices to have a better look, then I have some small projects to illustrate how they work before integrating in the large system.
- Performance Optimization for Large Courses: several courses can be up to many GB memory, so it affect loading times and user experience. Solution, implemented

lazy loading to load course content partially instead of loading everything at one. I also used index to database caching, and server caching.

5. Create a project:

> 1. **Simple To-Do List**
>
> **Objective:** Build a basic to-do list app using Next.js with client-side state management.
>
> ## Requirements:
>
> 1. Display a list of to-dos (store them in React state).
> 2. Allow users to add new tasks.
> 3. Allow users to mark tasks as completed.
> 4. Style the app using Tailwind CSS or plain CSS.
>
> ## Bonus (Optional):
>
> - Persist tasks in localStorage so they remain after page reloads.

Answer: file page.tsx

```
"use client";
import { Task } from "@/models/Task";

import { useState, useEffect } from "react";
```

```
export default function Page() {
  const [tasks, setTasks] = useState<Task[]>([]);
  const [input, setInput] = useState<Task >({
    title: "",
    description: "",
    id: 0,
    completed: false,
  });
```

```
  useEffect(() => {
    // try to load from local storage
    const tasksJson = localStorage.getItem("tasks");
    if (tasksJson) {
      setTasks(
        JSON.parse(tasksJson).map((task: any) => {
          return new Task(
            task.id,
            task.title,
            task.description,
            task.completed
          );
        })
      );
    }
  }, []);
```

```
  useEffect(() => {
    localStorage.setItem("tasks", JSON.stringify(tasks));
  }, [tasks]);
```

```
  /////////////taks action /////////////
  const addTask = () => {
    if (!input?.title) {
      return;
    }
```

```
    let maxId = 0;
    tasks.forEach((task) => {
      if (task.id > maxId) {
        maxId = task.id;
      }
    });
```

```
    const newTask = {
      id: maxId + 1,
      title: input.title,
      description: input.description,
      completed: false,
    };
    setTasks([...tasks, newTask]);
    setInput({
      title: "",
      description: "",
      id: 0,
      completed: false,
    });
```

```
  };
```

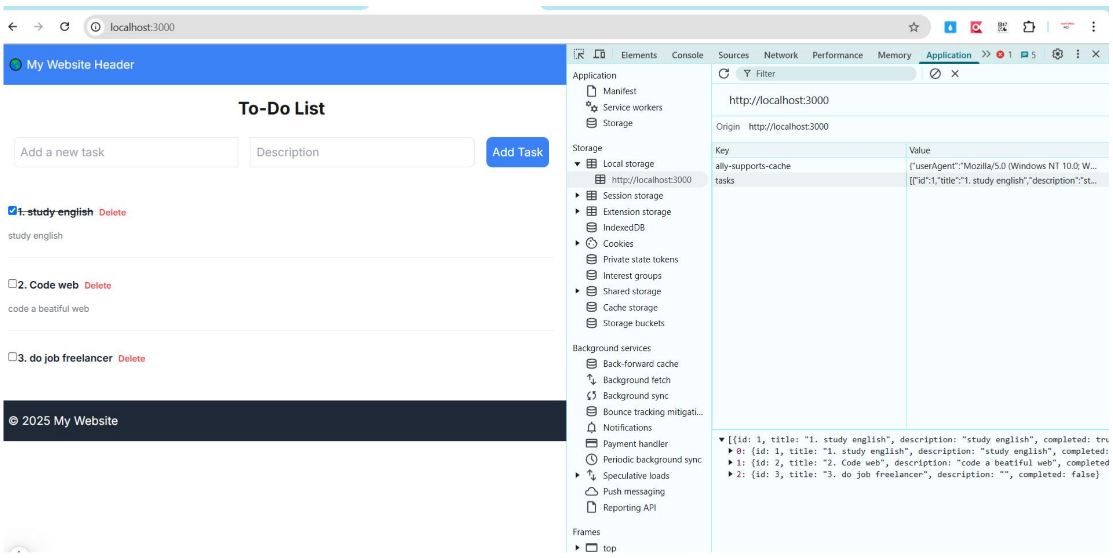
```
  const toggleTask = (e: any, task: Task)=>{
    const newTasks = tasks.map((t) => {
      if (t.id === task.id) {
        t.completed = e.target.checked;
      }
      return t;
    });
    setTasks(newTasks);
  }
```

```
  const deleteTask = (task: Task) => {
    const newTasks = tasks.filter((t) => t.id !== task.id);
    setTasks(newTasks);
  };
  const onTitleChange = (e:any) => {
    setInput({ ...input, title: e.target.value});
  }
  const onDescriptionChange = (e:any) => {
    setInput({ ...input, description: e.target.value });
  }
  /////////////render /////////////
  return (
    <div>
      <h1 className="text-2xl font-bold text-center mb-4">To-Do List</h1>
      <div className="flex mb-4">
        <input
          type="text"
          className="flex-1 p-2 border rounded-l-lg focus:outline-none m-2"
          placeholder="Add a new task"
          value={input?.title}
          onChange={(e) => onTitleChange(e)}
        />
        <input
          type="text"
          className="flex-1 p-2 border rounded-r-lg focus:outline-none m-2"
          placeholder="Description"
          value={input?.description}
          onChange={(e) => onDescriptionChange(e)}
        />
        <button
          className="bg-blue-500 text-white p-2 rounded-lg m-2"
          onClick={addTask}
        >
          Add Task
        </button>
      </div>
      <ul className="divide-y divide-gray-100">
        {tasks?.map((task) => (
          <li key={task.id} className="flex justify-between gap-x-6 py-5">
            <div className="flex items-center">
              <div className="min-w-0 flex-auto">
                <p className={task.completed ? "line-through text-sm/6 font-semibold text-gray-900" : "text-sm/6 font-semibold text-gray-900"}>
                  <input
                    type="checkbox"
                    checked={task.completed}
                    onChange={(e) => toggleTask(e, task)}
                  />
                  {task.title}
                  <button
                  className="text-red-500 text-xs/5 font-semibold m-2"
                  onClick={() => deleteTask(task)}
                >
                  Delete
                </button>
                </p>
                <p className="mt-1 truncate text-xs/5 text-gray-500">
                  {task.description}
                </p>

              </div>
            </div>
          </li>
        ))}
      </ul>
    </div>
  );
}
```

File Task.tsx

```
export class Task{
    id: number ;
    title: string;
    description: string;
    completed: boolean;

    constructor(id: number, title: string, description: string, completed: boolean) {
        this.id = id;
        this.title = title;
        this.description = description;
        this.completed = completed;
    }
}
```

-



localhost:3000

My Website Header

To-Do List

Add a new task | Description | Add Task

- [x] ~~1. study english~~ Delete
  study english
- [ ] 2. Code web Delete
  code a beatiful web
- [ ] 3. do job freelancer Delete

© 2025 My Website

http://localhost:3000

Origin http://localhost:3000

Key	Value
ally-supports-cache	{"userAgent":"Mozilla/5.0 (Windows NT 10.0; W...
tasks	[{"id":1,"title":"1. study english","description":"st...

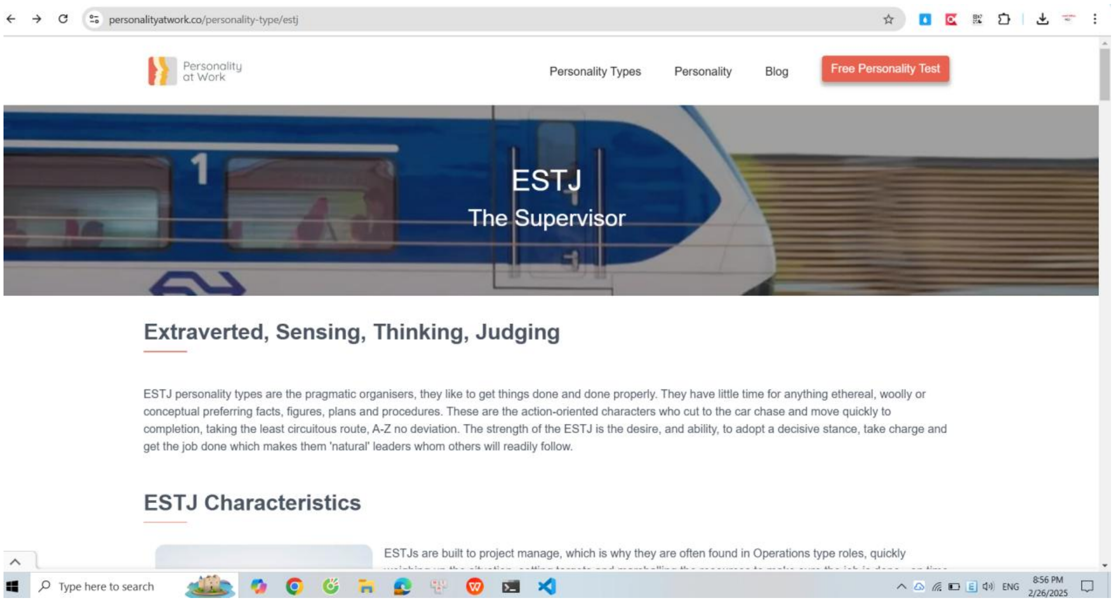
```
▼ [{id: 1, title: "1. study english", description: "study english", completed: tru
  ▶ 0: {id: 1, title: "1. study english", description: "study english", completed:
  ▶ 1: {id: 2, title: "2. Code web", description: "code a beatiful web", completed
  ▶ 2: {id: 3, title: "3. do job freelancer", description: "", completed: false}
```

Link source code github: https://github.com/GitHubCodeForLife/todoapp-nextjs

The following two tests are personality tests and moral tests. Please take screenshots of the results after completion and keep them.Please convert the test results to a PDF and send them to the email.

FREE MBTI test

https://personalityatwork.co/free-personality-test



ESTJ

The Supervisor

Extraverted, Sensing, Thinking, Judging

ESTJ personality types are the pragmatic organisers, they like to get things done and done properly. They have little time for anything ethereal, woolly or conceptual preferring facts, figures, plans and procedures. These are the action-oriented characters who cut to the car chase and move quickly to completion, taking the least circuitous route, A-Z no deviation. The strength of the ESTJ is the desire, and ability, to adopt a decisive stance, take charge and get the job done which makes them 'natural' leaders whom others will readily follow.

ESTJ Characteristics

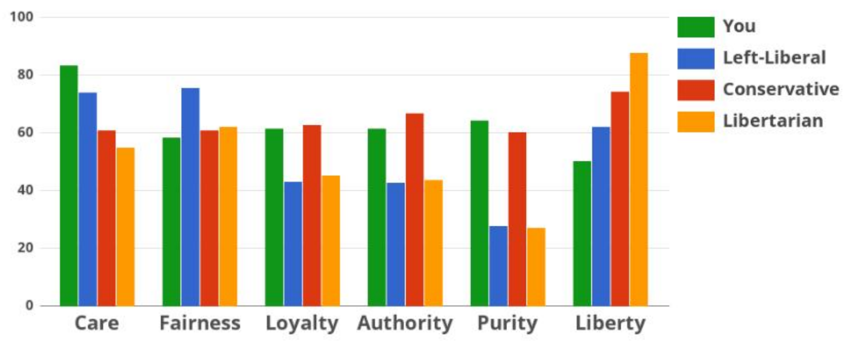
ESTJs are built to project manage, which is why they are often found in Operations type roles, quickly

FREE Morality Test

https://www.idrlabs.com/morality/6/test.php

Moral Foundations Test

Result: Your Moral Foundations Are:



Your scores:

- Care 83%
- Fairness 58%
- Purity 64%
- Loyalty 61%
- Authority 61%
- Liberty 50%

Your strongest moral foundation is **Care**.
Your morality is closest to that of a **Conservative**.

Please provide the following information regarding your working hours:

1.Total weekly working hours: 20

2.Working time slots:

- Saturday (8h - 18h), Sunday (8h-18h)
- Moday,Wenesday, and Friday (20h -> 22h)

3.Expected hourly rate（USD）: 10 USD/hour

This information will help us better arrange subsequent work but will not affect the interview result. Thank you!