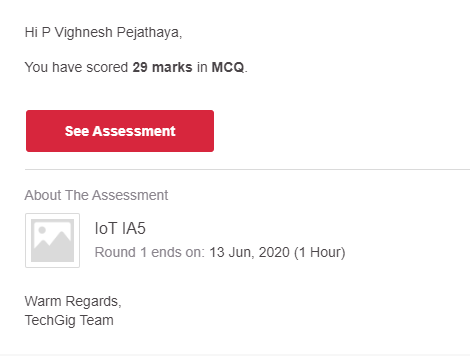
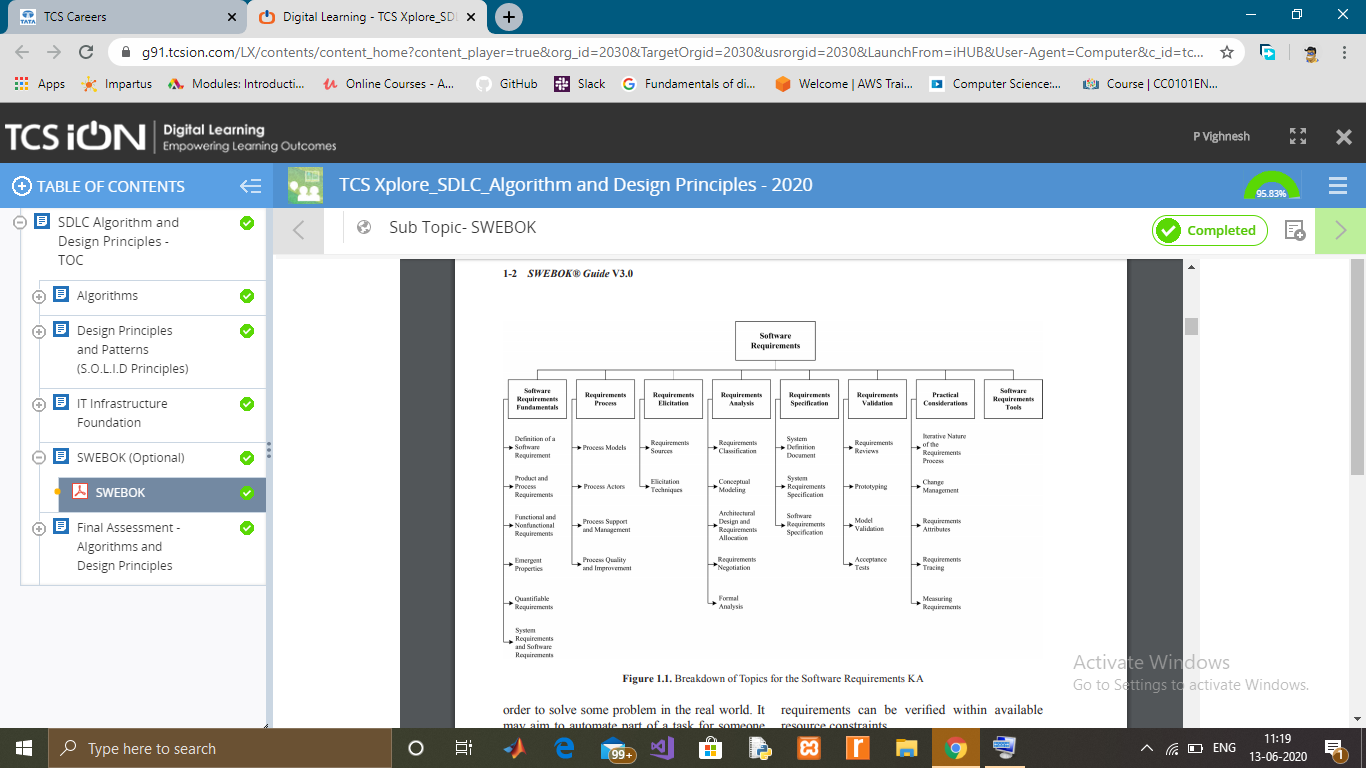
**DAILY ONLINE ACTIVITIES SUMMARY**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Date:** | **13-06-2020** | | | | | **Name:** | **P Vighnesh Pejathaya** | |
| **Sem & Sec** | **8 sem , A sec** | | | | | **USN:** | **4al16cs060** | |
| **Online Test Summary** | | | | | | | | |
| **Subject** | | **IOT** | | | | | | |
| **Max. Marks** | | **30** | | **Score** | | | **29** | |
| **Certification Course Summary** | | | | | | | | |
| **Course** | **SDLC Algorithm and Design Principles.** | | | | | | | |
| **Certificate Provider** | | | **TCS** | | **Duration** | | | **120 min** |
| **Coding Challenges** | | | | | | | | |
| **Problem Statement: Program to find the first non-repeating characters.** | | | | | | | | |
| **Status: Completed** | | | | | | | | |
| **Uploaded the report in Github** | | | | | **yes** | | | |
| **If yes Repository name** | | | | | **Alvas-education-foundation/p\_vighnesh** | | | |
| **Uploaded the report in slack** | | | | | **yes** | | | |

Online Test Details: (Attach the snapshot and briefly write the report for the same)



Certification Course Details: (Attach the snapshot and briefly write the report for the same)



Coding Challenges Details: (Attach the snapshot and briefly write the report for the same)

Given a string, find its first non-repeating character

**Approach:** A character is said to be non-repeating if its frequency in the string is unit. Now for finding such characters, one needs to find the frequency of all characters in the string and check which character has **unit frequency**. This task could be done efficiently using a **hash\_map** which will map the character to there respective frequencies and in which we can simultaneously update the frequency of any character we come across in constant time. The maximum distinct characters in the **ASCII system are 256**. So **hash\_map** has a maximum size of **256**. Now read the string again and the first character which we find has a frequency as unity is the answer.

**Algorithm:**

1. Make a **hash\_map** which will map the character to there respective frequencies.
2. Traverse the given string using a pointer.
3. Increase the count of current character in the **hash\_map**.
4. Now traverse the string again and check whether the current character has**frequency=1**.
5. If the **frequency>1** continue the traversal.
6. Else **break** the loop and print the current character as the answer.