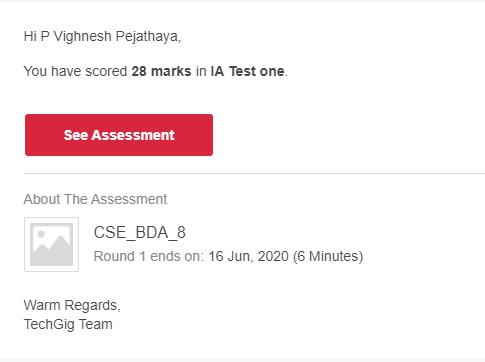
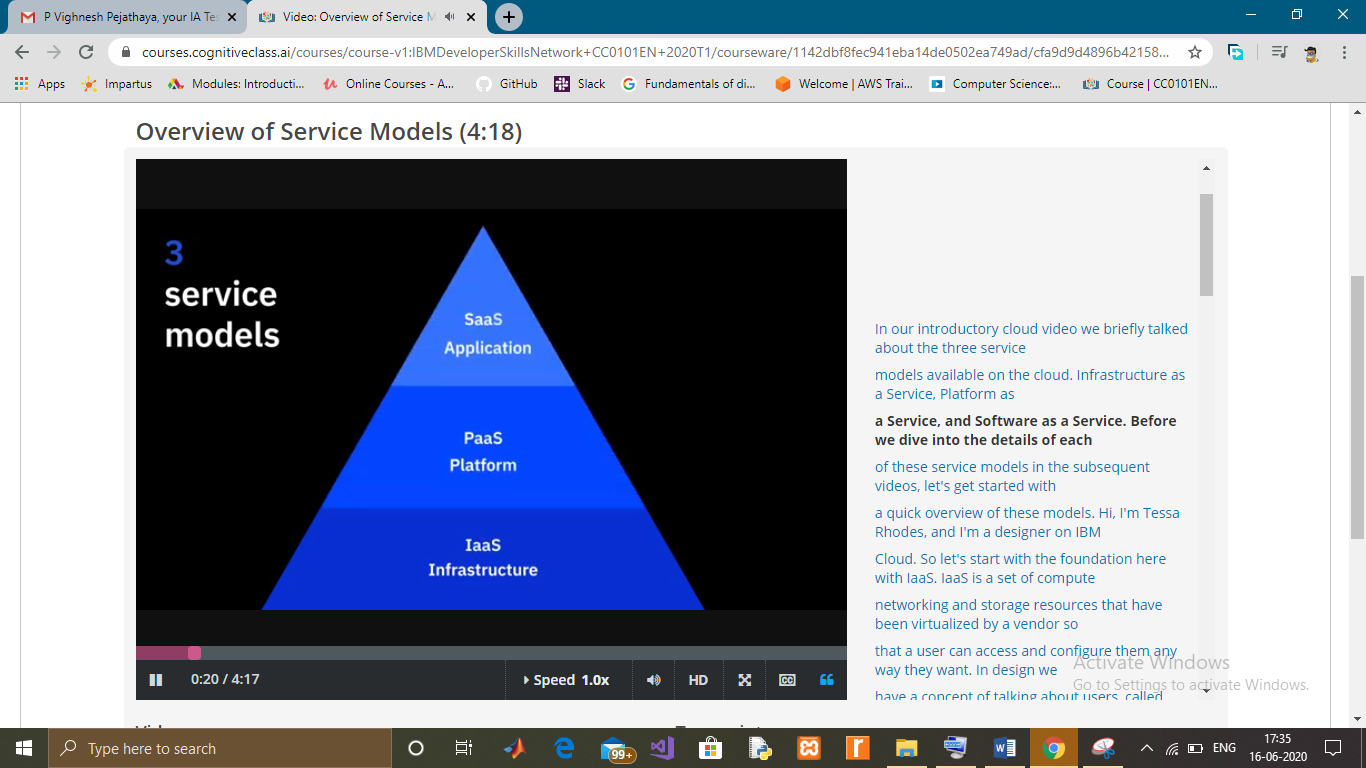
**DAILY ONLINE ACTIVITIES SUMMARY**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Date:** | **16-06-2020** | | | | | **Name:** | **P Vighnesh Pejathaya** | |
| **Sem & Sec** | **8 sem , A sec** | | | | | **USN:** | **4al16cs060** | |
| **Online Test Summary** | | | | | | | | |
| **Subject** | | **BDA** | | | | | | |
| **Max. Marks** | | **30** | | **Score** | | | **28** | |
| **Certification Course Summary** | | | | | | | | |
| **Course** | **Cloud Computing** | | | | | | | |
| **Certificate Provider** | | | **Cognitive classes(IBM)** | | **Duration** | | | **120 min** |
| **Coding Challenges** | | | | | | | | |
| **Problem Statement: Program to check if Binary Tree is BST.** | | | | | | | | |
| **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)

Check if a Binary Tree is BST : Simple and Efficient Approach.Given a Binary Tree, the task is to check whether the given binary tree is Binary Search Tree or not.

A binary search tree (BST) is a node-based binary tree data structure which has the following properties.

* The left subtree of a node contains only nodes with keys less than the node’s key.
* The right subtree of a node contains only nodes with keys greater than the node’s key.
* Both the left and right subtrees must also be binary search trees.

**Time Complexity:** O(N)  
**Auxiliary Space:** O(1)

The idea is to use Inorder traversal and keep track of the previously visited node’s value. Since the inorder traversal of a BST generates a sorted array as output, So, the previous element should always be less than or equals to the current element.

While doing In-Order traversal, we can keep track of previously visited Node’s value by passing an integer variable using reference to the recursive calls. If the value of the currently visited node is less than the previous value, then the tree is not BST.