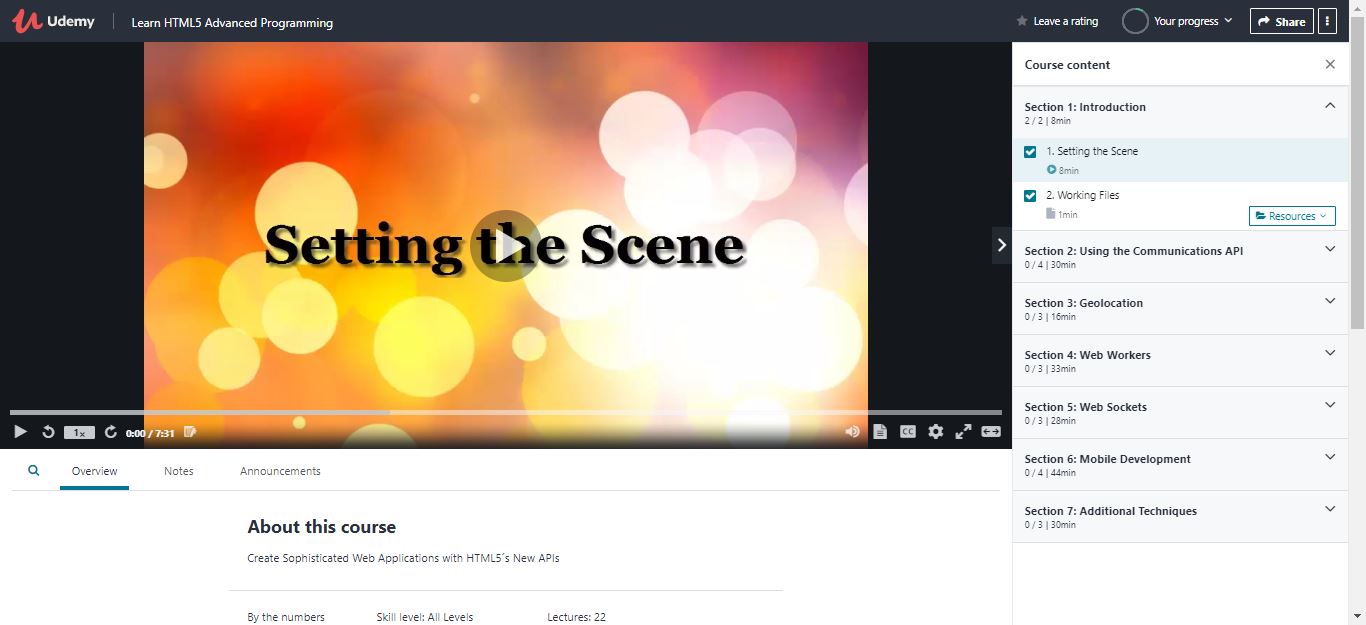
**DAILY ONLINE ACTIVITIES SUMMARY**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Date:** | **09-07-2020** | | | | | **Name:** | **Anix Jugal D’Cunha** | |
| **Sem & Sec** | **8 sem , A sec** | | | | | **USN:** | **4AL16CS013** | |
| **Online Test Summary** | | | | | | | | |
| **Subject** | | **Not Conducted** | | | | | | |
| **Max. Marks** | | **--** | | **Score** | | | **--** | |
| **Certification Course Summary** | | | | | | | | |
| **Course** | **Learn HTML5 Advanced Programming** | | | | | | | |
| **Certificate Provider** | | | **Udemy** | | **Duration** | | | 3hr 10min |
| **Coding Challenges** | | | | | | | | |
| **Problem Statement:** Java program to find smallest tree node. | | | | | | | | |
| **Status: Competed** | | | | | | | | |
| **Uploaded the report in Github** | | | | | **yes** | | | |
| **If yes Repository name** | | | | | **alvas-education-foundation/dcunhaanixjugal** | | | |
| **Uploaded the report in slack** | | | | | **yes** | | | |

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

**Not Conducted**

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)

## Program-> Java program to find smallest tree node.

|  |  |  |
| --- | --- | --- |
|  | | |
| public class SmallestNode { |
|  | public static class Node{ | |
|  | int data; | |
|  | Node left; | |
|  | Node right; | |
|  |  | |
|  | public Node(int data){ | |
|  |  | |
|  | this.data = data; | |
|  | this.left = null; | |
|  | this.right = null; | |
|  | } | |
|  | } | |
|  |  | |
|  | public Node root; | |
|  |  | |
|  | public SmallestNode(){ | |
|  | root = null; | |
|  | } | |
|  |  | |
|  | public int smallestElement(Node temp){ | |
|  | if(root == null) { | |
|  | System.out.println("Tree is empty"); | |
|  | return 0; | |
|  | } | |
|  | else { | |
|  | int leftMin, rightMin; | |
|  |  | |
|  | int min = temp.data; | |
|  |  | |
|  | if(temp.left != null){ | |
|  | leftMin = smallestElement(temp.left); | |
|  | min = Math.min(min, leftMin); | |
|  | } | |
|  |  | |
|  | if(temp.right != null){ | |
|  | rightMin = smallestElement(temp.right); | |
|  |  | |
|  | min = Math.min(min, rightMin); | |
|  | } | |
|  | return min; | |
|  | } | |
|  | } | |
|  |  | |
|  | public static void main(String[] args) { | |
|  |  | |
|  | SmallestNode bt = new SmallestNode(); | |
|  |  | |
|  | bt.root = new Node(4); | |
|  | bt.root.left = new Node(2); | |
|  | bt.root.right = new Node(3); | |
|  | bt.root.left.left = new Node(1); | |
|  | bt.root.right.left = new Node(5); | |
|  | bt.root.right.right = new Node(6); | |
|  |  | |
|  | System.out.println("Smallest element in the binary tree: " + bt.smallestElement(bt.root)); | |
|  | } | |
|  | } | |
|  | Output: | |
|  |  | |
|  | Smallest element in the binary tree: 1 | |