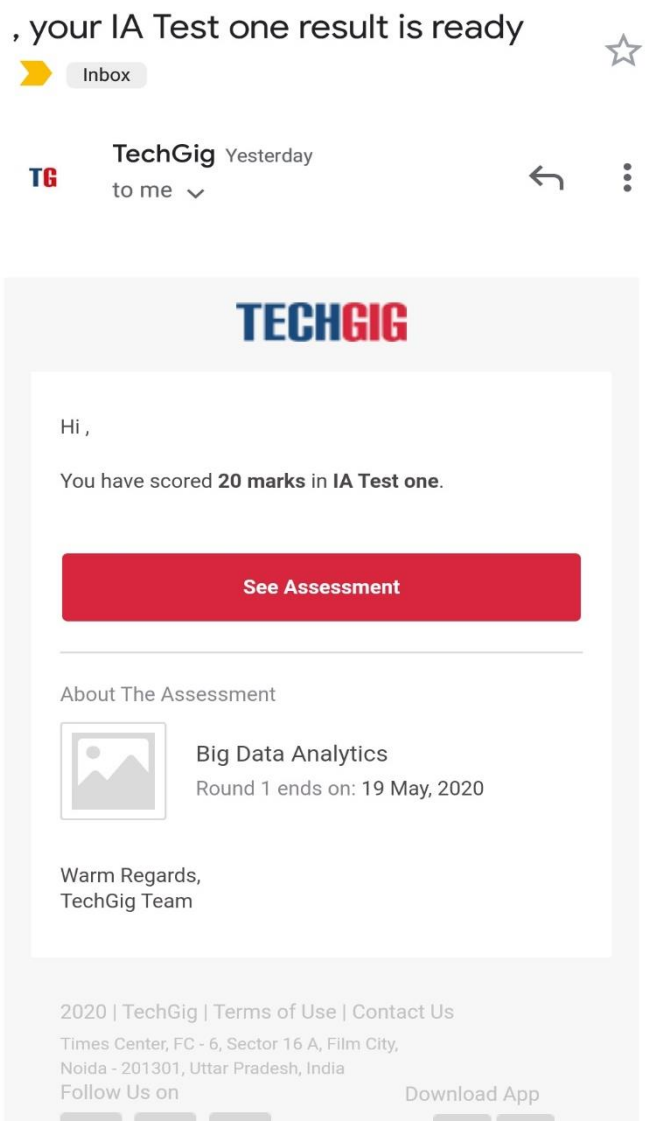
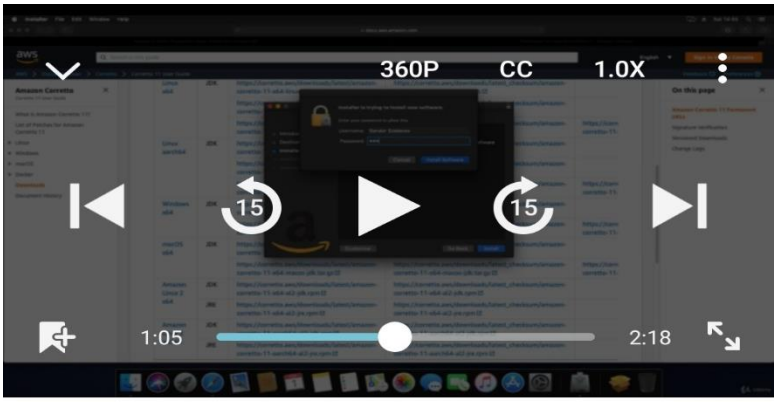
DAILY ONLINE ACTIVITIES SUMMARY

Date:	**19/5/2020**	**Name:**	**Afrah Saleem**
Sem & Sec	**8th Sem B section**	**USN:**	**4AL16CS127**
Online Test Summary			
Subject	**Big Data Analytics**		
Max. Marks	**30**	**Score**	**20**
Certification Course Summary			
Course	**Practical java course: zero to one**		
Certificate Provider	**Udemy**	**Duration**	**4 hrs**
Coding Challenges			
Problem Statement: **1)Add letters to given letter/word and find the shortest palindrome.** **2) To check if given linked list is palindrome or not**			
Status: Completed			
Uploaded the report in Github	**Yes**		
If yes Repository name	**Afrah**		
Uploaded the report in slack	**yes**		

Online Test Details:



Certification Course Details:



Lectures More

Section 2 - Enviroment Setup

8	Tools you'll need	Video - 03:42 mins
9	JDK install for Windows	Video - 02:07 mins
10	IntelliJ install for Windows	Video - 02:50 mins
11	**JDK install for MacOS**	Video - 02:18 mins
12	IntelliJ install for MacOS	Video - 02:33 mins
13	JDK install for Linux	Video - 03:02 mins
14	IntelliJ install for Linux	Video - 02:25 mins

Coding Challenges Details:

Program1:

```
package shortestpalindromeexample.java;
import java.util.Scanner;

public class ShortestPalindromeDemo {

public static String shortestPalindrome(String str) {

int x=0;
int y=str.length()-1;

while(y>=0){
if(str.charAt(x)==str.charAt(y)){
x++;
}
y--;
}

if(x==str.length())
return str;

String suffix = str.substring(x);
String prefix = new StringBuilder(suffix).reverse().toString();
String mid = shortestPalindrome(str.substring(0, x));

return prefix+mid+suffix;
}

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

System.out.println("Enter a String to find out shortest palindrome");

String str=in.nextLine();

System.out.println("Shortest palindrome of "+str+" is "+shortestPalindrome(str));

}
```

Program2

```
import java.util.Stack;

// Data Structure to store a linked list node
class Node {
int data;
Node next;

Node(int i)
{
	this.data = i;
	this.next = null;
}
};

class Main
{
// Function to determine if a given linked list is palindrome or not
public static boolean isPalindrome(Node head)
{
// construct an empty stack
Stack s = new Stack<>();

	// push all elements of the linked list into the stack
	Node node = head;
	while (node != null) {
		s.push(node.data);
		node = node.next;
	}

	// traverse the linked list again
	node = head;
	while (node != null)
	{
		// pop the top element from the stack
		int top = s.pop();

		// compare the popped element with current node's data
		// return false if mismatch happens
		if (top != node.data) {
			return false;
		}

		// advance to the next node
		node = node.next;
	}

	// we reach here only when the linked list is palindrome
	return true;
```

```
}

public static void main(String[] args)
{
	Node head = new Node(1);
	head.next = new Node(2);
	head.next.next = new Node(3);
	head.next.next.next = new Node(2);
	head.next.next.next.next = new Node(1);

	if (isPalindrome(head)) {
		System.out.print("Linked List is a palindrome.");
	} else {
		System.out.print("Linked List is not a palindrome.");
	}
}
}
```