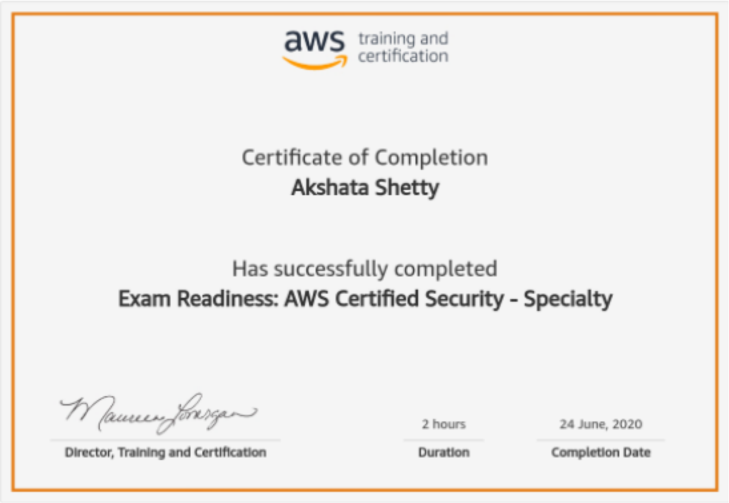
DAILY ONLINE ACTIVITIES SUMMARY

Date:	**24th June 2020**	**Name:**	**Akshata Shetty**
Sem & Sec	**8th Semester ‘B’ Section**	**USN:**	**4AL16CS092**
Online Test Summary			
Subject	**-**		
Max. Marks	**-**	**Score**	**-**
Certification Course Summary			
Course	**AWS exam readiness : security specialty**		
Certificate Provider	**AWS**	**Duration**	**2 hour**
Coding Challenges			
Problem Statement: 1. program to reverse an string			
Status: completed			
Uploaded the report in Github	**yes**		
If yes Repository name	**Akshata**		
Uploaded the report in slack	**yes**		

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


aws training and certification

Certificate of Completion
Akshata Shetty

Has successfully completed
Exam Readiness: AWS Certified Security - Specialty

Director, Training and Certification

2 hours
Duration

24 June, 2020
Completion Date

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

```
public class ReverseList {

   //Represent a node of the doubly linked

   list class Node{
      int data;
      Node
      previous;
      Node next;

      public Node(int data)
         { this.data = data;
      }
   }

   //Represent the head and tail of the doubly linked
   list Node head, tail = null;

   //addNode() will add a node to
   the list public void addNode(int
   data) {
      //Create a new node
      Node newNode = new Node(data);

      //If list is
      empty if(head
      == null) {
```

```
      //Both head and tail will point to
      newNode head = tail = newNode;
      //head's previous will point to
      null head.previous = null;
      //tail's next will point to null, as it is the last node of the list
      tail.next = null;
   }
   else {
      //newNode will be added after tail such that tail's next will point to
      newNode tail.next = newNode;
      //newNode's previous will point to
      tail newNode.previous = tail;
      //newNode will become new
      tail tail = newNode;
      //As it is last node, tail's next will point to null
      tail.next = null;
   }
}

//reverse() will reverse the doubly linked
list public void reverse() {
   //Node current will point to
   head Node current = head,
   temp = null;

   //Swap the previous and next pointers of each node to reverse the direction of the
   list while(current != null) {
      temp = current.next;
      current.next =
      current.previous;
      current.previous = temp;
      current = current.previous;
   }
   //Swap the head and tail
   pointers. temp = head;
   head =
   tail; tail =
   temp;
}

//display() will print out the elements of
the list public void display() {
   //Node current will point to
   head Node current = head;
   if(head == null)
      { System.out.println("List is
      empty"); return;
   }
```

```
    while(current != null) {
        //Prints each node by incrementing the pointer.

        System.out.print(current.data + " ");
        current = current.next;
    }
}

public static void main(String[] args) {

    ReverseList dList = new ReverseList();
    //Add nodes to the
    list
    dList.addNode(1);
    dList.addNode(2);
    dList.addNode(3);
    dList.addNode(4);
    dList.addNode(5);

    System.out.println("Original
    List:"); dList.display();

    //Reverse the given
    list dList.reverse();

    //Displays the reversed list
    System.out.println("\nReversed List:
    "); dList.display();
}
```