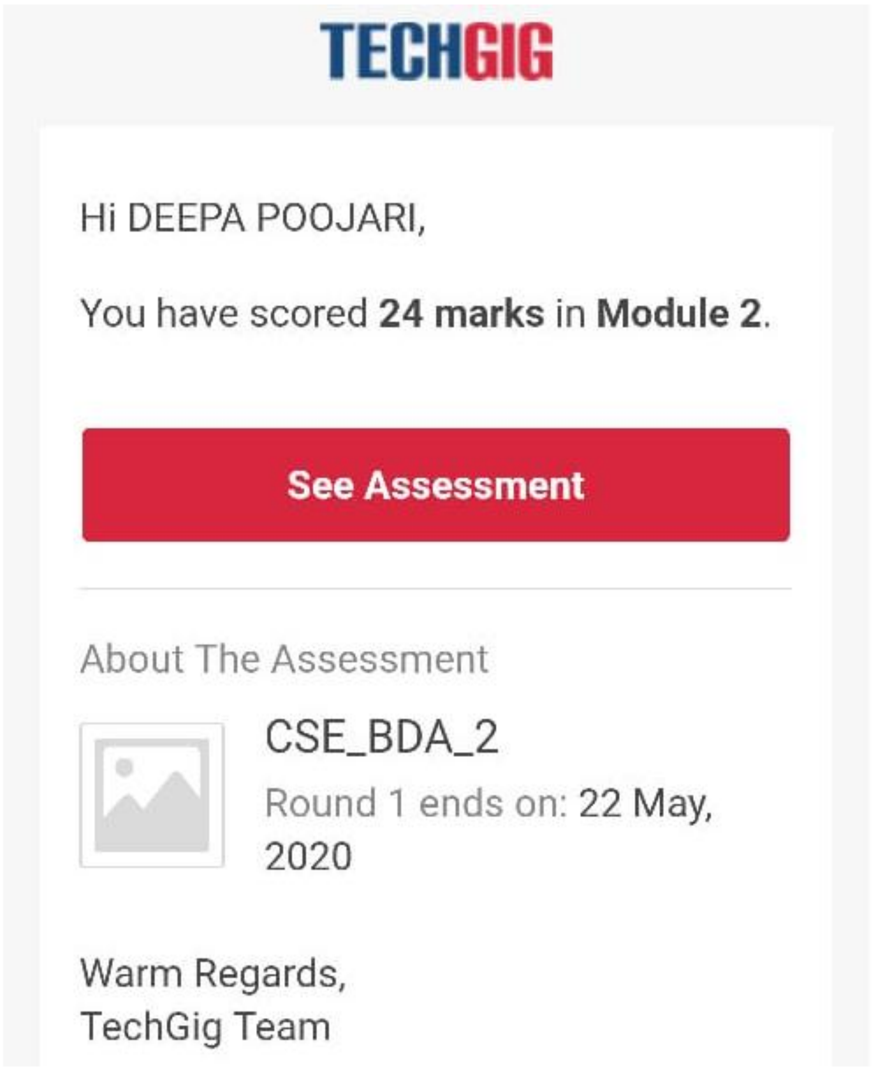
DAILY ONLINE ACTIVITIES SUMMARY

Date:	**22/5/2020**	**Name:**	**Deepa**
Sem & Sec	**8th Sem**	**USN:**	**4AL16CS029**
Online Test Summary			
Subject	**Big Data Analytics**		
Max. Marks	**40**	**Score**	**24**
Certification Course Summary			
Course	**Introduction to Ethical Hacking**		
Certificate Provider	**greatlearning.in**	**Duration**	**6 hrs**
Coding Challenges			
Problem Statement: 1)Write a C Program to implement various operations of Singly Linked List Stack			
Status: Completed			
Uploaded the report in Github	**Yes**		
If yes Repository name	**Daily_report**		
Uploaded the report in slack	**yes**		

Online Test Details:



TECHGIG

Hi DEEPA POOJARI,

You have scored **24 marks** in **Module 2**.

See Assessment

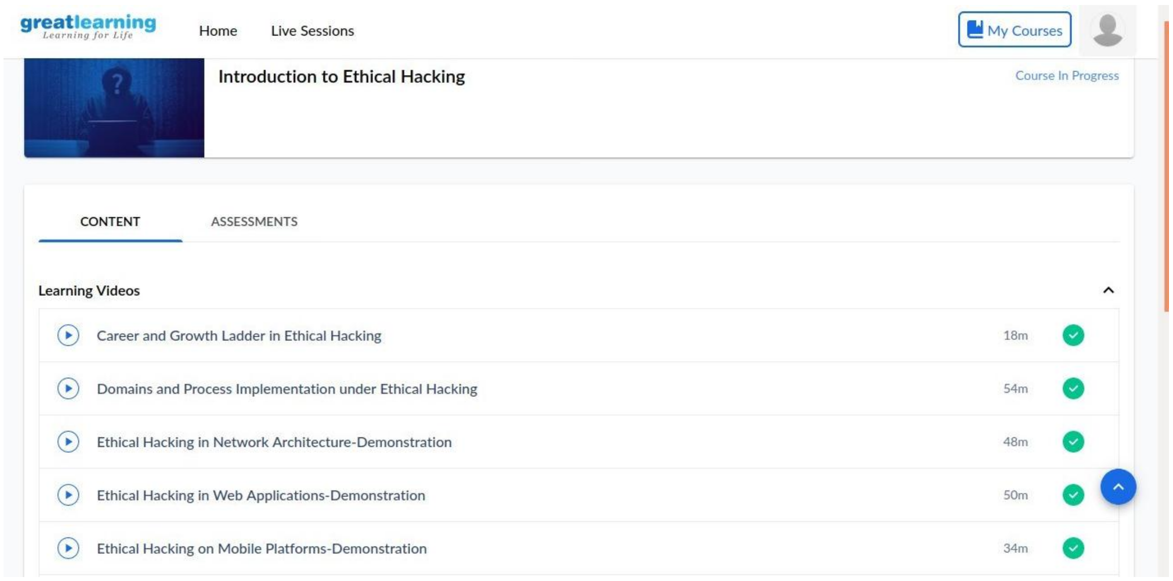
About The Assessment

CSE_BDA_2

Round 1 ends on: 22 May, 2020

Warm Regards,
TechGig Team

Certification Course Details:



Introduction to Ethical Hacking

Course In Progress

Learning Videos

Career and Growth Ladder in Ethical Hacking	18m ✓
Domains and Process Implementation under Ethical Hacking	54m ✓
Ethical Hacking in Network Architecture-Demonstration	48m ✓
Ethical Hacking in Web Applications-Demonstration	50m ✓
Ethical Hacking on Mobile Platforms-Demonstration	34m ✓

Coding Challenges Details:

Program 1:

```
#include <stdio.h>
#include <stdlib.h>

struct node
{
   int info;
   struct node*ptr;
}*top,*top1,*temp;

int topelement();
void push(int data);
void pop();
void   empty();
void display();
void destroy();
void stack_count();
void create();
int count = 0;

void main()
{
   int no, ch, e;
   while (1)
   {
```

```
printf("\n 1 - Push\t\t2 - Pop");
printf("\n 3 - Top\t\t4 - Check if Stack Empty");
printf("\n 5 - Exit\t\t6 - Dipslay");
printf("\n 7 - Stack Count\t8 - Destroy stack");
                                        printf("\n------------------------------------------------ \n");
create();
printf("\nEnter choice : ");
scanf("%d", &ch);


switch (ch)
{
case 1:
   printf("Enter data : ");
   scanf("%d", &no);
   push(no);
   break;
case 2:
   pop();
   break;
case 3:
   if (top == NULL)
      printf("No elements in stack");
   else
   {
      e = topelement();
      printf("\n Top element : %d", e);
   }
```

```
                                                printf("\n --------------------------------------------\n");
          break;
        case 4:
          empty();
          break;
        case 5:
          exit(0);
        case 6:
          display();
          break;
        case 7:
          stack_count();
          break;
        case 8:
          destroy();
          break;
        default :
          printf(" Wrong choice, Please enter correct choice ");
                                                printf("\n --------------------------------------------\n");
          break;
        }
    }
}
void create()
{
    top = NULL;
}
```

```
void stack_count()
{
   printf("\n No. of elements in stack : %d", count);
                                        printf("\n-------------------------------------------- \n");
}
void push(int data)
{
   if (top == NULL)
   {
      top =(struct node *)malloc(1*sizeof(struct node));
      top->ptr = NULL;
      top->info = data;
   }
   else
   {
      temp =(struct node *)malloc(1*sizeof(struct node));
      temp->ptr = top;
      temp->info = data;
      top = temp;
   }
   count++;
                                        printf("\n-------------------------------------------- \n");
}
void display()
{
   top1 =top;
```

```
    if (top1 == NULL)
    {
        printf("Stack is empty");
                                                printf("\n ---------------------------------------------\n");
        return;
    }

    while (top1 != NULL)
    {
        printf("%d ", top1->info);
        top1 = top1->ptr;
    }
                                              printf("\n--------------------------------------------- \n");
 }
void pop()
{
    top1 = top;

    if (top1 == NULL)
    {
        printf("\n Error : Trying to pop from empty stack");
        return;
    }
    else
        top1 = top1->ptr;
    printf("\n Popped value : %d", top->info);
    free(top);
```

```
   top = top1;
   count--;
                                                   printf("\n-------------------------------------------- \n");
}
int topelement()
{
   return(top->info);
}
void empty()
{
   if (top == NULL)
      printf("\n Stack is empty");
   else
      printf("\n Stack is not empty with %d elements", count);
                                                   printf("\n-------------------------------------------- \n");
}
void destroy()
{
   top1 = top;

   while (top1 != NULL)
   {
      top1 = top->ptr;
      free(top);
      top = top1;
      top1 = top1->ptr;
   }
```

```
    free(top1);

    top = NULL;


    printf("\nAllstackelementsdestroyed");

    count =0;

    printf("\n ---------------------------------------------- \n");

}
```