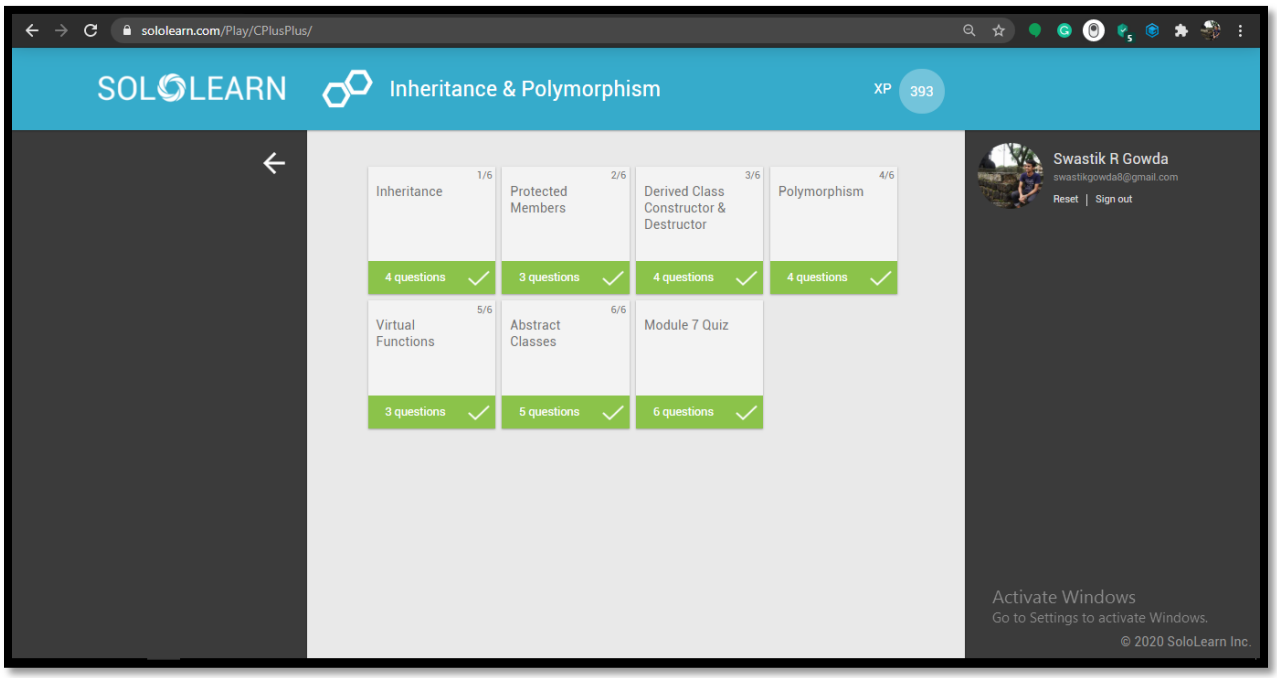
DAILY ASSESSMENT

Date:	**25-June-2020**	**Name:**	**Swastik R Gowda**
Course:	**Solo-Learn C++**	**USN:**	**4AL17EC091**
Topic:	**Module - 7 : Inheritance & Polymorphism**	**Semester & Section:**	**6th Sem 'B' Sec**
Github Repository:	**swastik-gowda**		

FORENOON SESSION DETAILS

Image of session



Report – Report can be typed or hand written for up to two pages.

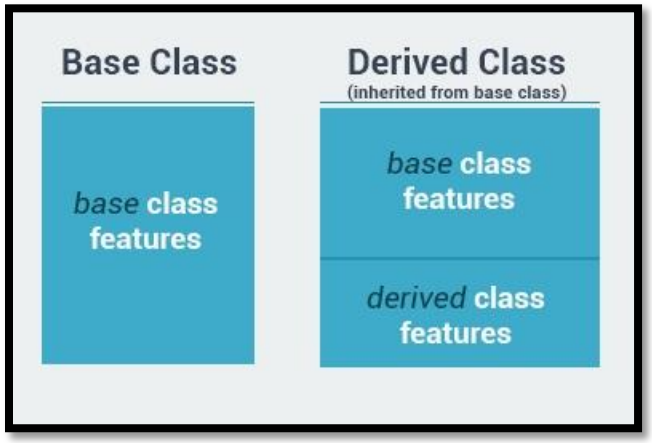
Inheritance

- Inheritance is one of the most important concepts of object-oriented programming.
- Inheritance allows us to define a class based on another class.
- This facilitates greater ease in creating and maintaining an application.

The class whose properties are inherited by another class is called the ***Base class***.
The class which inherits the properties is called the ***Derived class***.

For example, the Daughter class (derived) can be inherited from the Mother class (base).

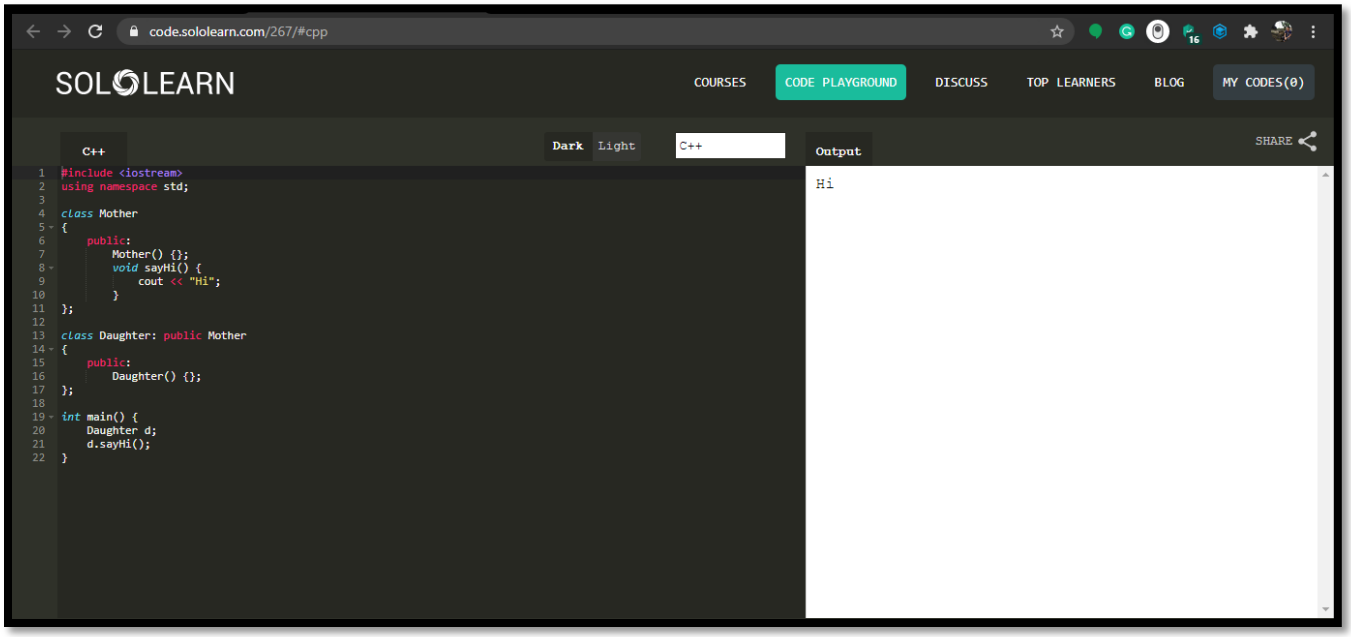
The derived class inherits all features from the base class, and can have its own additional features.



As all public members of the Mother class become public members for the Daughter class, we can create an object of type Daughter and call the ***sayHi()*** function of the Mother class for that object**:**

A derived class inherits all base class methods with the following exceptions:
- Constructors, destructors
- Overloaded operators
- The friend functions

```
#include <iostream>
using namespace std;

class Mother
{
 public:
  Mother() {};
  void sayHi() {
   cout << "Hi";
  }
};

class Daughter: public Mother
{
 public:
  Daughter() {};
};

int main() {
  Daughter d;
  d.sayHi();
}
```



Polymorphism

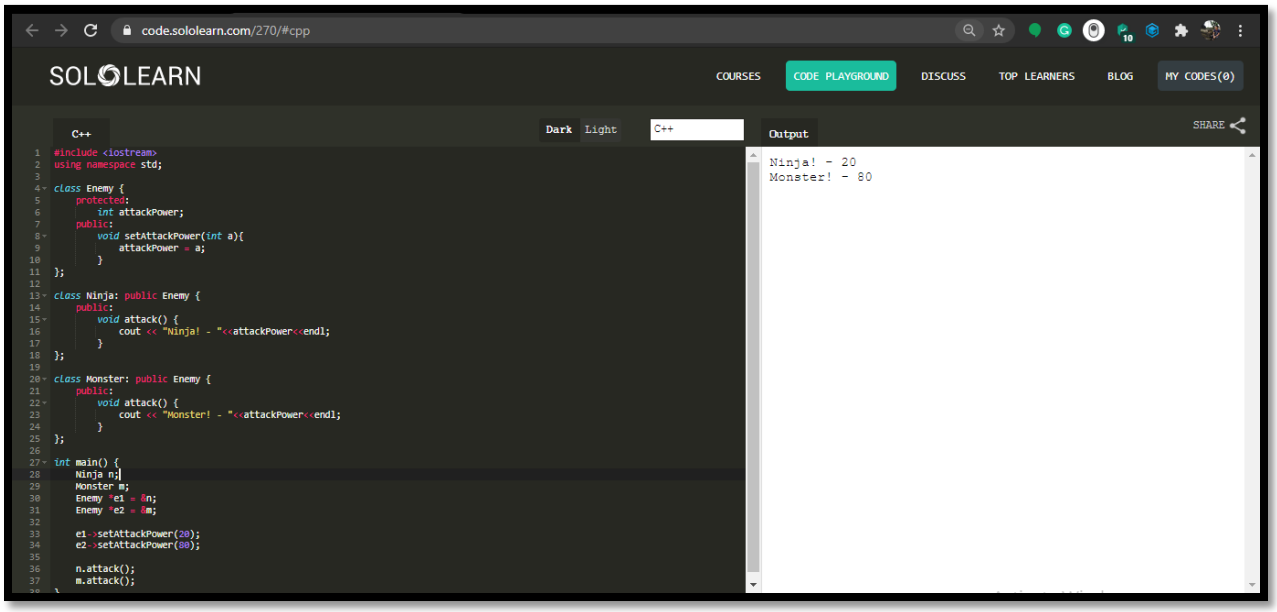
- The word polymorphism means "having many forms".
- Typically, polymorphism occurs when there is a hierarchy of classes and they are related by inheritance.

C++ polymorphism means that a call to a member function will cause a different implementation to be executed depending on the type of object that invokes the function.

Polymorphism can be demonstrated more clearly using an example:
Suppose you want to make a simple game, which includes different enemies: monsters, ninjas, etc. All enemies have one function in common: an attack function. However, they each attack in a different way. In this situation, polymorphism allows for calling the same attack function on different objects, but resulting in different behaviors.

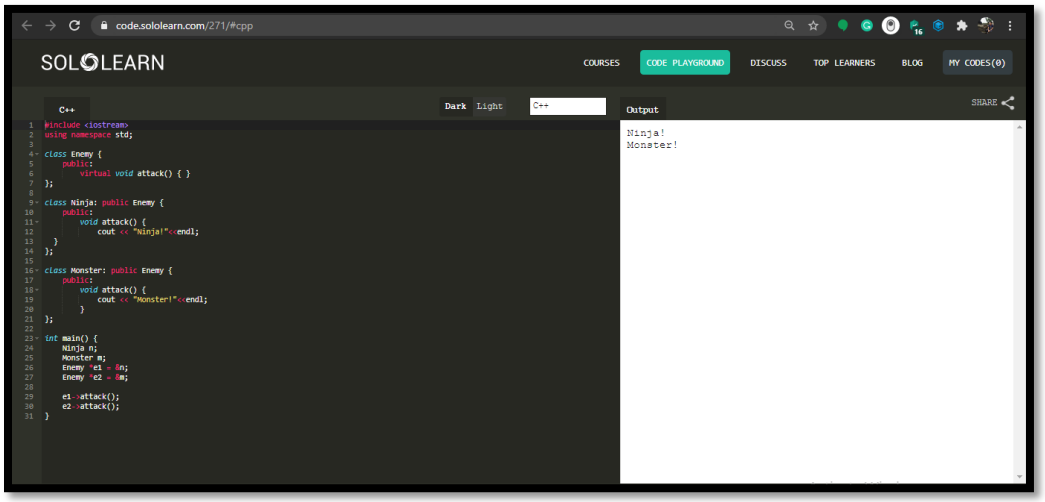
```
#include <iostream>
using namespace std;

class Enemy {
  protected:
    int attackPower;
  public:
    void setAttackPower(int a){
      attackPower = a;
    }
};

class Ninja: public Enemy {
```

```
   public:
      void attack() {
         cout << "Ninja! - "<<attackPower<<endl;
      }
};

class Monster: public Enemy {
   public:
      void attack() {
         cout << "Monster! - "<<attackPower<<endl;
      }
};

int main() {
   Ninja n;
   Monster m;
   Enemy *e1 = &n;
   Enemy *e2 = &m;

   e1->setAttackPower(20);
   e2->setAttackPower(80);

   n.attack();
   m.attack();
}
```



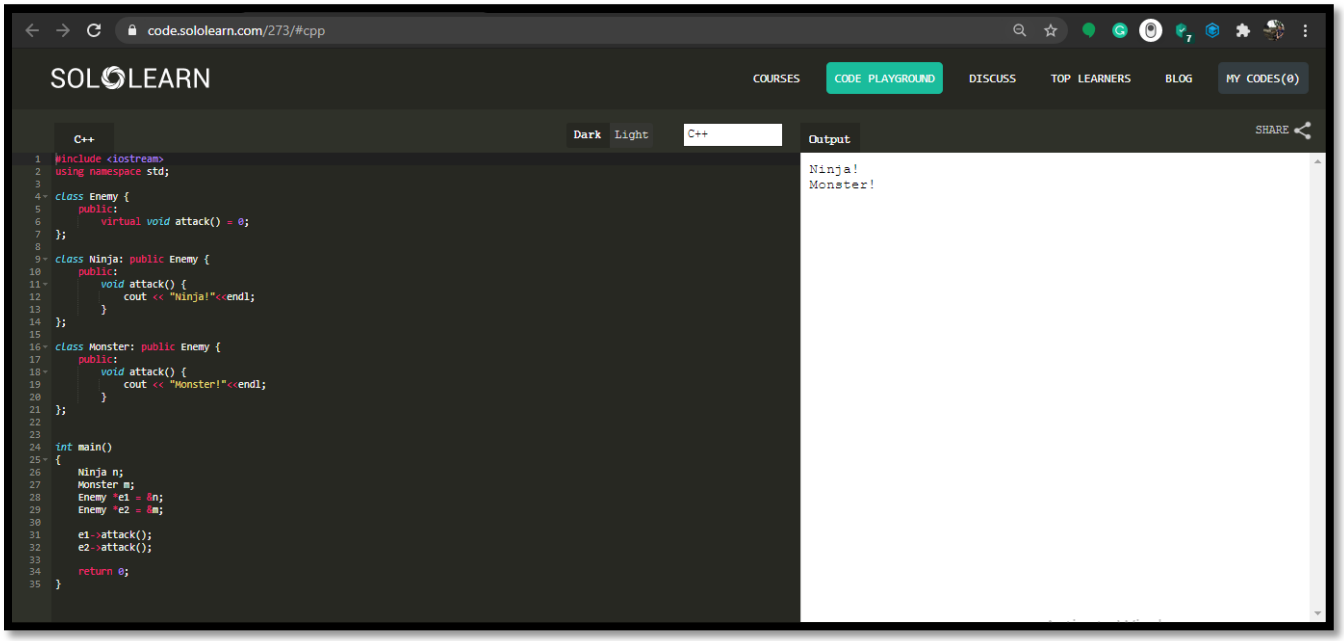
Virtual Functions

```
#include <iostream>
using namespace std;

class Enemy {
   public:
      virtual void attack() { }
};

class Ninja: public Enemy {
   public:
      void attack() {
         cout << "Ninja!"<<endl;
  }
};

class Monster: public Enemy {
   public:
      void attack() {
         cout << "Monster!"<<endl;
      }
};

int main() {
   Ninja n;
   Monster m;
   Enemy *e1 = &n;
   Enemy *e2 = &m;

   e1->attack();
   e2->attack();
}
```



Pure Virtual Functions

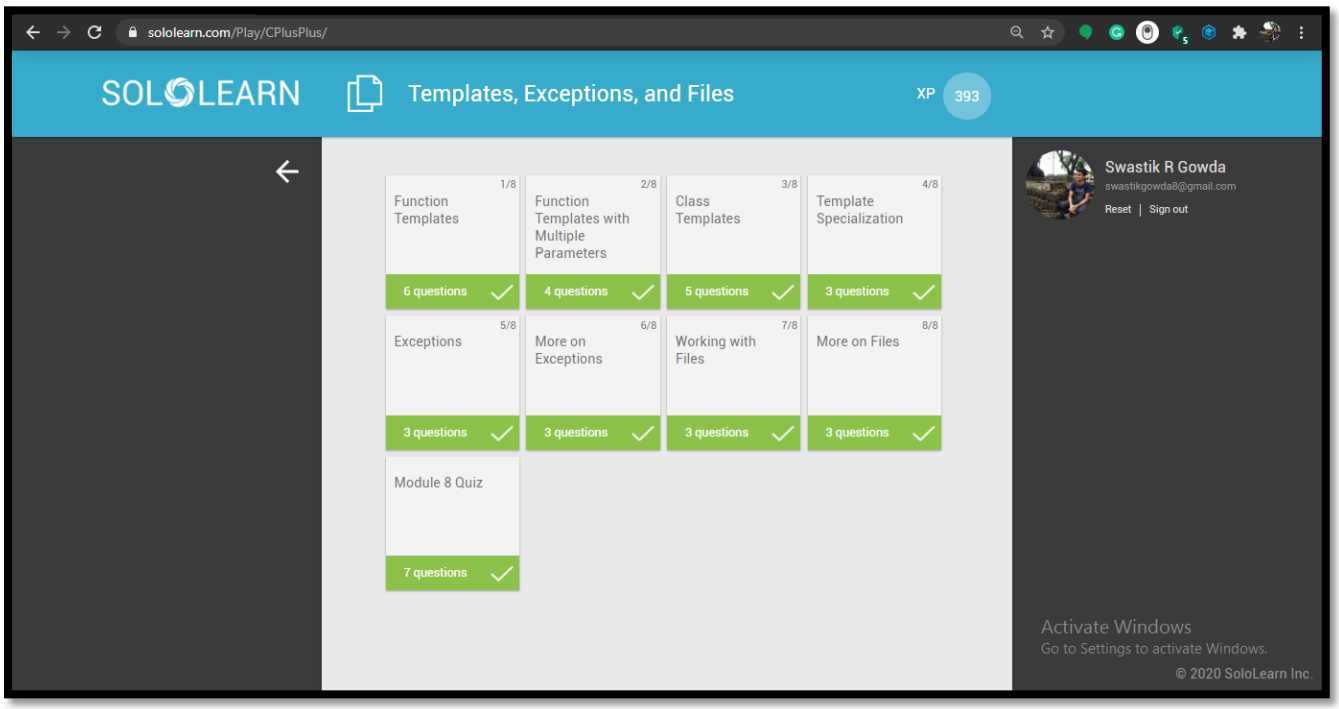
A pure virtual function basically defines, that the derived classes will have that function defined on their own.
Every derived class inheriting from a class with a pure virtual function must override that function



Date:	25-June-2020	Name:	Swastik R Gowda
Course:	Solo-Learn C++	USN:	4AL17EC091
Topic:	Module - 8 : Templates, Exceptions and Files	Semester & Section:	6th Sem 'B' Sec

AFTERNOON SESSION DETAILS

Image of session



Report – Report can be typed or hand written for up to two pages.

Function Templates

Functions and classes help to make programs easier to write, safer, and more maintainable.
However, while functions and classes do have all of those advantages, in certain cases they can also be somewhat limited by C++'s requirement that you specify types for all of your parameters.
Function templates give us the ability to do that!
With function templates, the basic idea is to avoid the necessity of specifying an exact type for each variable. Instead, C++ provides us with the capability of defining functions using placeholder types, called template type parameters.
A specific syntax is required in case you define your member functions outside of your class - for example in a separate source file.
You need to specify the generic type in angle brackets after the class name**.**

Throwing Exceptions

C++ exception handling is built upon three keywords: try, catch, and throw.
throw is used to throw an exception when a problem shows up.

For example:

```
int motherAge = 29;
int sonAge = 36;
if (sonAge > motherAge) {
  throw "Wrong age values";
}
```

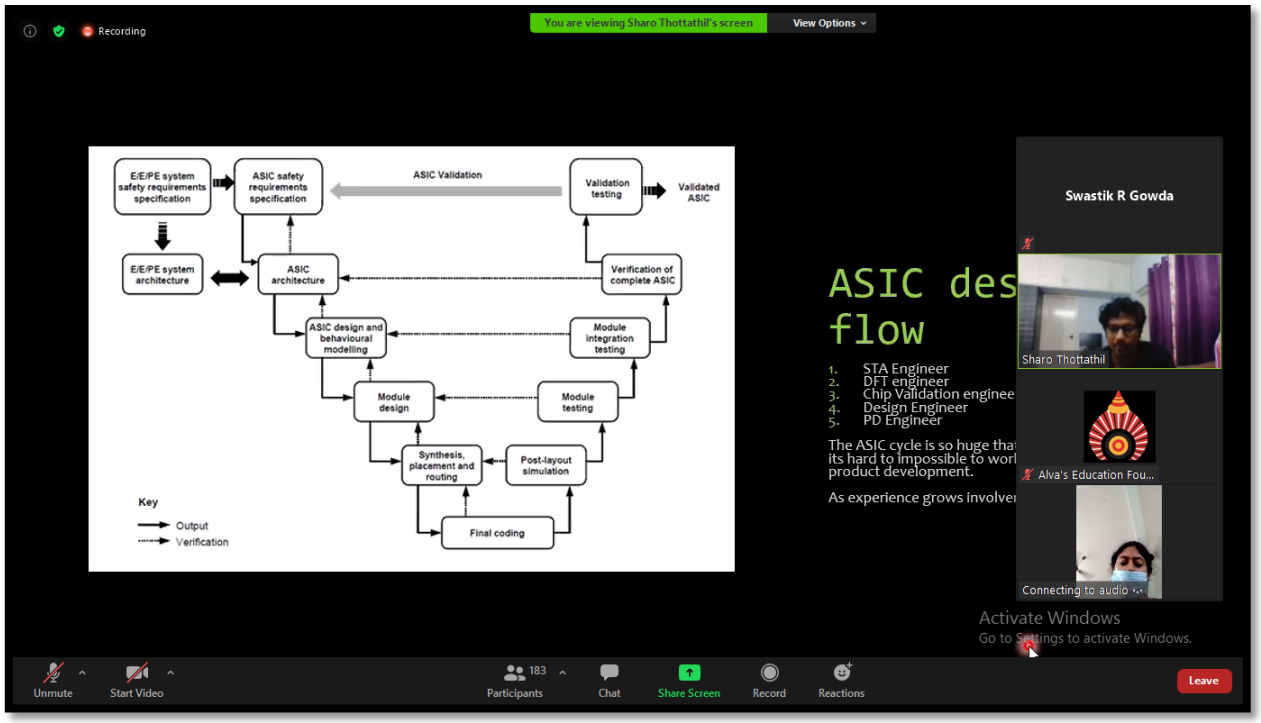
The code looks at sonAge and motherAge, and throws an exception if sonAge is found to be the greater of the two.

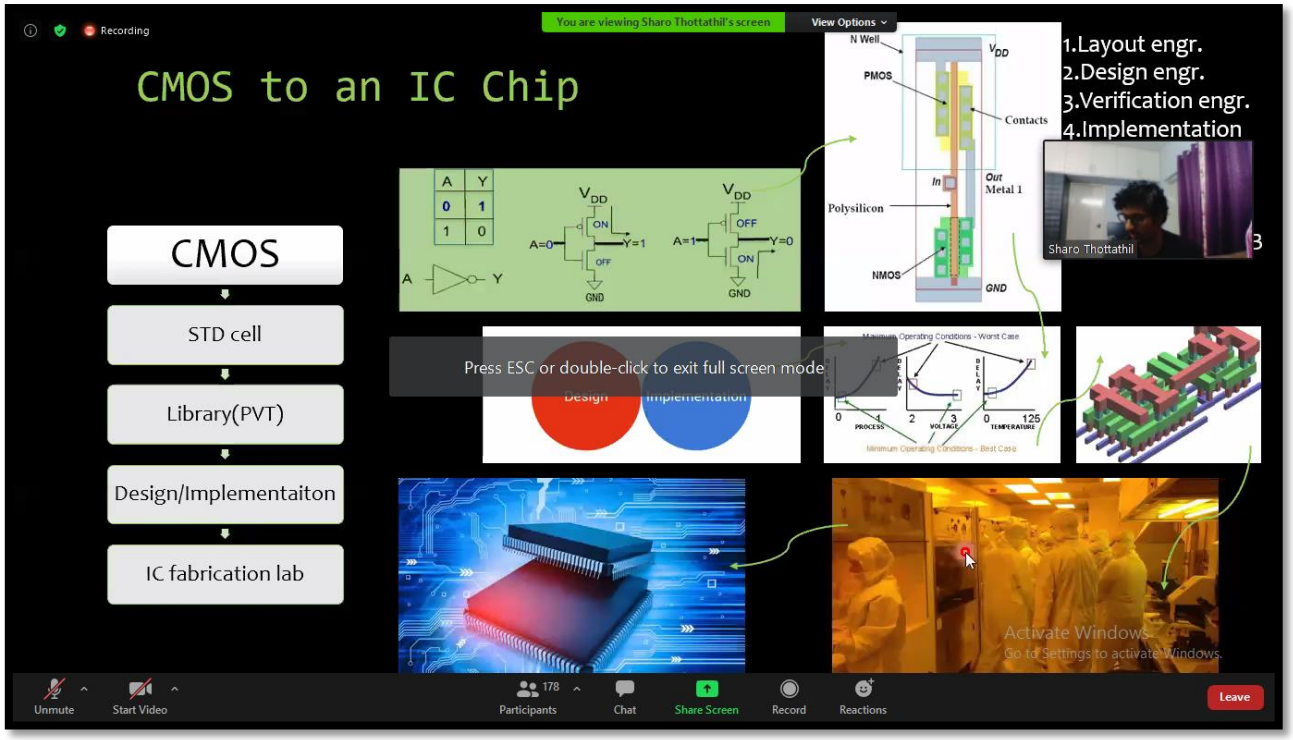
A try block identifies a block of code that will activate specific exceptions. It's followed by one or more catch blocks. The catch keyword represents a block of code that executes when a particular exception is thrown.
Code that could generate an exception is surrounded with the try/catch block.
You can specify what type of exception you want to catch by the exception declaration that appears in parentheses following the keyword catch.

Date:	25-June-2020	Name:	Swastik R Gowda
Course:	Webinar	USN:	4AL17EC091
Topic:	VLSI Scope in India	Semester & Section:	6th Sem 'B' Sec

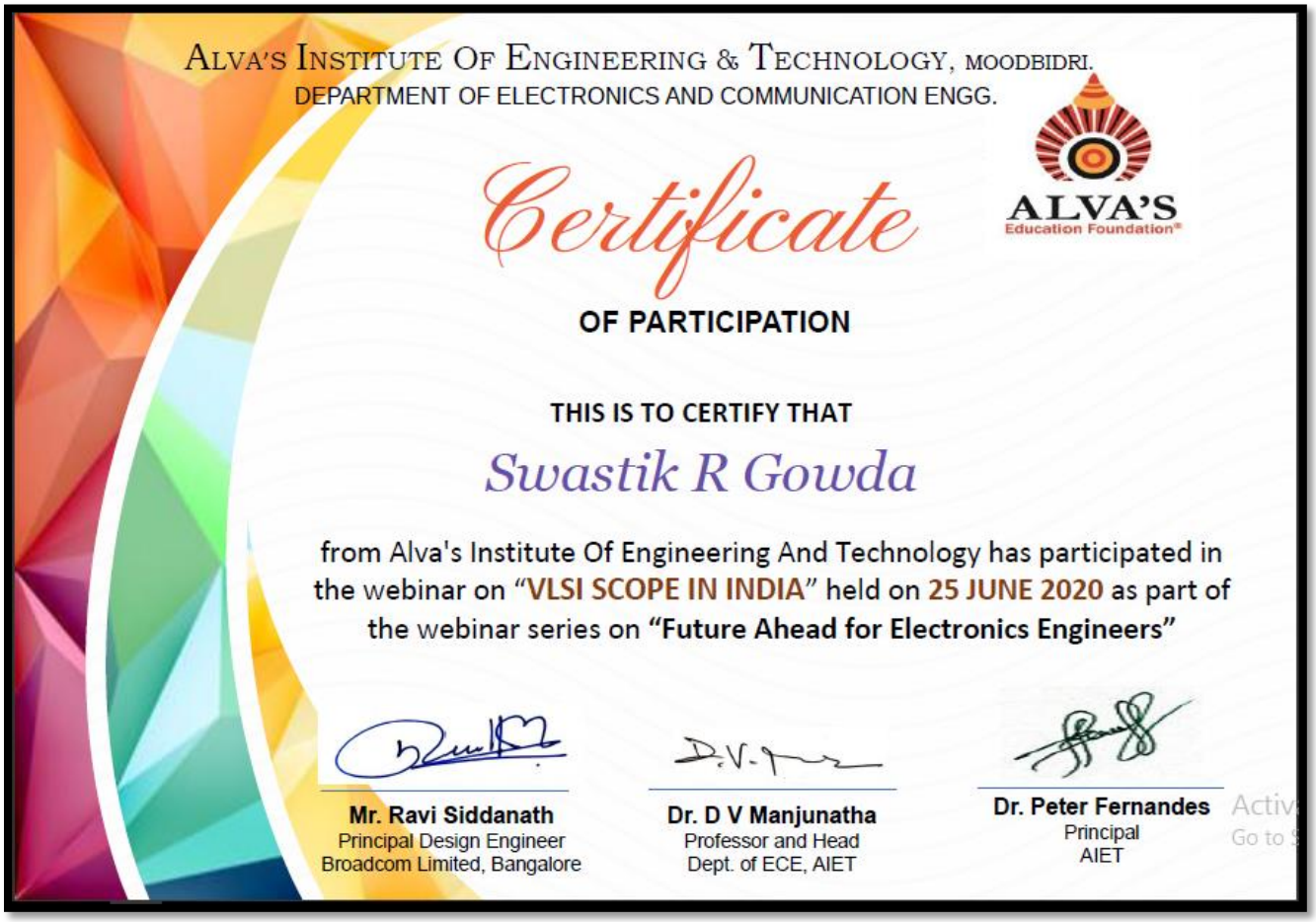
AFTERNOON SESSION DETAILS

Image of session





Certificate



ALVA'S INSTITUTE OF ENGINEERING & TECHNOLOGY, MOODBIDRI.
DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGG.

ALVA'S
Education Foundation®

Certificate

OF PARTICIPATION

THIS IS TO CERTIFY THAT

Swastik R Gowda

from Alva's Institute Of Engineering And Technology has participated in the webinar on "VLSI SCOPE IN INDIA" held on 25 JUNE 2020 as part of the webinar series on "Future Ahead for Electronics Engineers"

Mr. Ravi Siddanath
Principal Design Engineer
Broadcom Limited, Bangalore

Dr. D V Manjunatha
Professor and Head
Dept. of ECE, AIET

Dr. Peter Fernandes
Principal
AIET