Practical No. 04

Programs on Remote Object Communication

Aim: 1) Retrieve the Student Score from the database. (Use Concept of JDBC and RMI)

Code:

InterfaceStud.java

```
import java.rmi.*;
public interface InterfaceStud extends Remote
{
   public double findScore(String name)throws RemoteException;
}
```

DBStud.java

```
import java.rmi.*;
import java.rmi.server.*;
import java.sql.*;
public class DBStud extends UnicastRemoteObject implements InterfaceStud
{
        private PreparedStatement pstmt;
        public DBStud()throws RemoteException
        {
                super();
                initilizeDB();
        }
        protected void initilizeDB()
        {
                try{
                  Class.forName("com.mysql.jdbc.Driver");
                  System.out.println("driver registered");
                  Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/symca","root","");
                  System.out.println("Database connected");
                  pstmt = conn.prepareStatement("select * from student where Name =?");
                }
                catch(Exception ex)
                {
                        System.out.println(ex);
                }       }
        public double findScore(String name)throws RemoteException
        {
                double score = -1;
                try
                {
                                pstmt.setString(1,name);
```

```
                                ResultSet rs = pstmt.executeQuery();
                                if(rs.next())
                                {
                                        score = rs.getDouble(2);}
                        }
                        catch(SQLException ex)
                        {
                                System.out.println(ex);
                        }
                        System.out.println(score);
                        return score;
                }
}
```

RegisterStud.java

```
import java.rmi.*;
import java.rmi.registry.*;
public class RegisterStud
{
public static void main(String[] agrs)
{
 try
 {
        DBStud obj = new DBStud();
        Registry reg = LocateRegistry.getRegistry();
        Naming.rebind("DBStud",obj);
 }
 catch(Exception e)
 { }
}
}
```

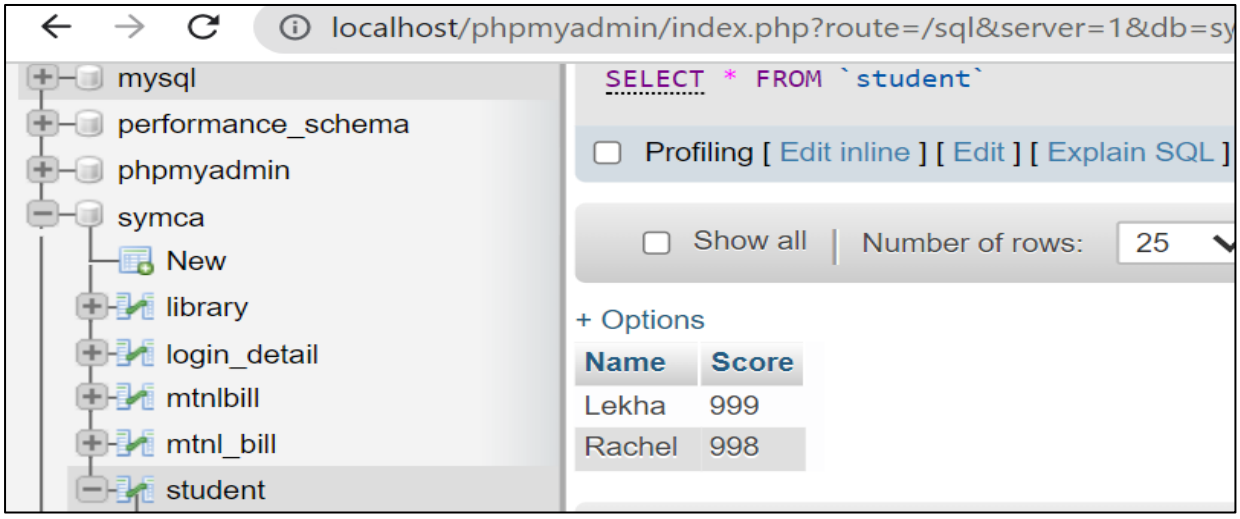
ClientStud.java

```
import java.rmi.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class ClientStud extends JApplet
{
        private InterfaceStud student;
        private JButton getScore = new JButton("GetSocre");
        private JTextField name = new JTextField();
        private JTextField tfscore = new JTextField();
        private JLabel lname = new JLabel("Student");
```

```
	private JLabel lscore = new JLabel("Score");
public void init()
{
	try
	{
		student = (InterfaceStud)Naming.lookup("DBStud");
	}
	catch(Exception e)
	{
	}
	JPanel panel = new JPanel();
		panel.setLayout(null);
		lname.setBounds(15, 15, 75, 33);
		panel.add(lname);
		name.setBounds(95, 15, 150, 33);
		panel.add(name);
		lscore.setBounds(15, 65, 75, 33);
		panel.add(lscore);
		tfscore.setBounds(95, 65, 150, 33);
		panel.add(tfscore);
		getScore.setBounds(95, 130, 150, 33);
		panel.add(getScore, BorderLayout.CENTER);
		add(panel,BorderLayout.CENTER);
	getScore.addActionListener(new ActionListener()
	{
		public void actionPerformed(ActionEvent ae)
		{
			getScore();
		}
	}
	);
}
public void getScore()
{
	try
	{
		double score = student.findScore(name.getText());
```

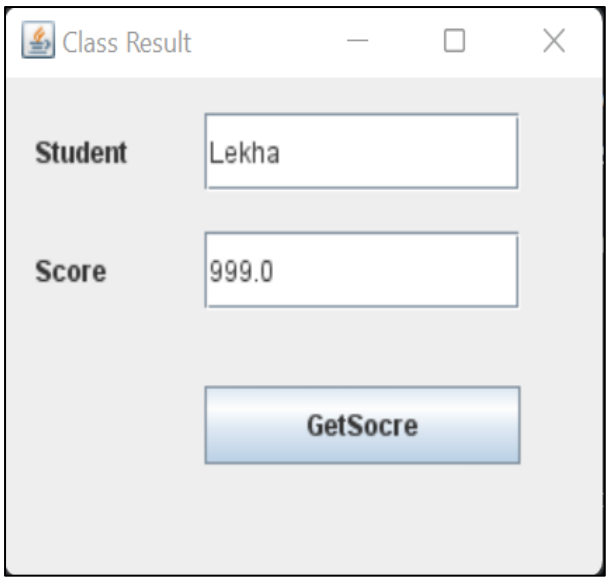
```
                if(score<0)
                        tfscore.setText("Not Found");
                else
                tfscore.setText(""+score);
        }
        catch(Exception ex)
        {       }
}
public static void main(String[] args)
{
        ClientStud applet = new ClientStud();
        JFrame frame = new JFrame();
        frame.setTitle("Class Result");
        frame.add(applet,BorderLayout.CENTER);
        frame.setSize(300,250);
        applet.init();
        frame.setVisible(true);
}       }
```

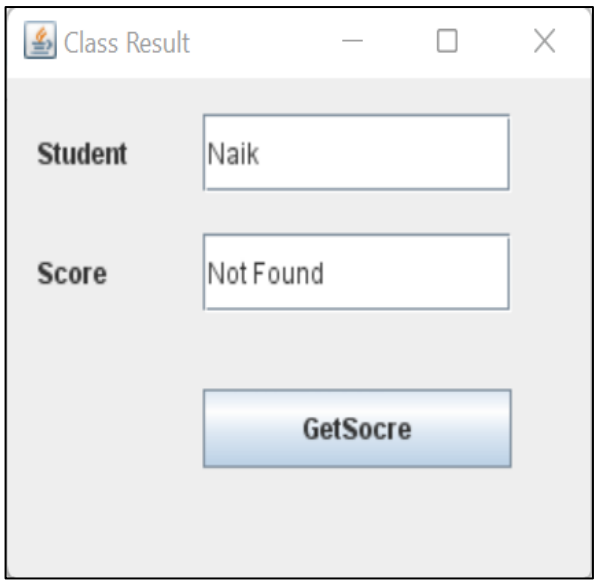
Output:



```
C:\Windows\System32\cmd.exe - java -cp mysql-connector-java-5.1.23-bin.jar;. RegisterStud
Microsoft Windows [Version 10.0.22518.1000]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Gagan\OneDrive\Documents\lekha_naik\StudentScore>javac InterfaceStud.java

C:\Users\Gagan\OneDrive\Documents\lekha_naik\StudentScore>javac DBStud.java

C:\Users\Gagan\OneDrive\Documents\lekha_naik\StudentScore>javac RegisterStud.java

C:\Users\Gagan\OneDrive\Documents\lekha_naik\StudentScore>javac ClientStud.java

C:\Users\Gagan\OneDrive\Documents\lekha_naik\StudentScore>start rmiregistry

C:\Users\Gagan\OneDrive\Documents\lekha_naik\StudentScore>java -cp mysql-connector-java-5.1.23-bin.jar;. RegisterStud
driver registered
Database connected
999.0
999.0
-1.0
```

```
C:\Windows\System32\cmd.exe - java ClientStud
Microsoft Windows [Version 10.0.22518.1000]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Gagan\OneDrive\Documents\lekha_naik\StudentScore>java ClientStud
```





Practical No. 04

Programs on Remote Object Communication

Aim: 2) Using MySQL create Library database. Create table Book (Book_id, Book_name, Book_author) and retrieve the Book information from Library database using Remote Object Communication concept.

Code:

InterfaceLib.java

```
import java.rmi.*;
public interface InterfaceLib extends Remote
{
   public String[] findInfo(String id)throws RemoteException;
}
```

ImplLib.java

```
import java.rmi.*;
import java.rmi.server.*;
import java.sql.*;
public class ImplLib extends UnicastRemoteObject implements InterfaceLib
{
	int find = 0;
	String info[];
	public ImplLib()throws RemoteException
	{
		super();
	}
	public String[] findInfo(String id)throws RemoteException
	{
		try
		{
				Class.forName("com.mysql.jdbc.Driver");
				System.out.println("Driver Registered");
				Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/library","root","");
				System.out.println("Database connected");
				Statement stmt = conn.createStatement();
				String str = "select * from Book where Book_id='"+id+"'";
				ResultSet rs = stmt.executeQuery(str);
				while(rs.next())
				{
					find++;
				}
				info = new String[find];
				ResultSet rst = stmt.executeQuery(str);
				for(int i = 0; i < find; i++)
```

```
                        {
                                rst.next();
                                info[i] = "Result:\n \n ID:      "+rst.getString(1)+"\n \n
Book Name:    "+rst.getString(2)+"\n \n Author:      "+rst.getString(3);
                                System.out.println(info[i]);
                        }
                }
                catch(Exception ex)
                {
                        System.out.println(ex);
                }
                return info;
        }
```

RegisterLib.java

```
import java.rmi.*;
import java.rmi.registry.*;
public class RegisterLib
{
        public static void main(String[] agrs)
        {
                try
                {
                        ImplLib obj = new ImplLib();
                        Registry reg = LocateRegistry.getRegistry();
                        Naming.rebind("ImplLib",obj);
                }
                catch(Exception e)
                {}
        }
}
```

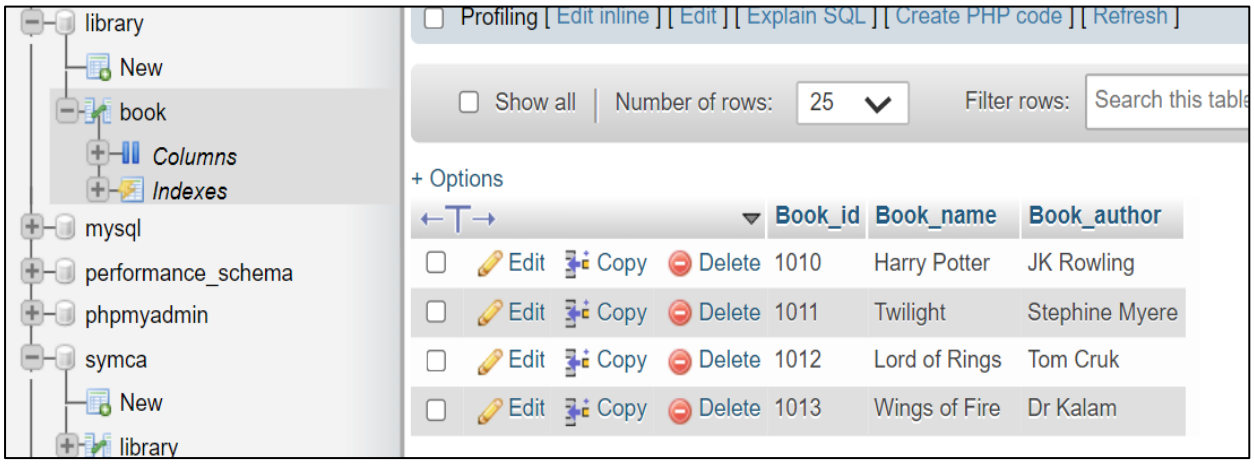
ClientLib.java

```
import java.rmi.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class ClientLib extends JApplet
{
        private InterfaceLib library;
        private JButton getDetail = new JButton("Get Detail");
        private JTextField name = new JTextField();
        private JLabel lname = new JLabel("Book");
        private JLabel result = new JLabel("Result");
```

```
public void init()
{
	try
	{
		library = (InterfaceLib)Naming.lookup("ImplLib");
	}
	catch(Exception e)
	{
	}
	JPanel panel = new JPanel();
		panel.setLayout(null);
		lname.setBounds(15, 15, 75, 33);
		panel.add(lname);
		name.setBounds(95, 15, 150, 33);
		panel.add(name);
		result.setBounds(15, 65, 750, 45);
		panel.add(result);
		getDetail.setBounds(115, 130, 150, 33);
		panel.add(getDetail, BorderLayout.CENTER);
		add(panel,BorderLayout.CENTER);
	getDetail.addActionListener(new ActionListener()
	{
		public void actionPerformed(ActionEvent ae)
		{
			getDetail();
		}
	}
	);
}
public void getDetail()
{
	try
	{
		String[] str = library.findInfo(name.getText());

		String r = "";
		for(int i = 0; i<str.length; i++)
```

```
                                {
                                        r += " \n"+str[i];
                                        System.out.println(str[i]);
                                }
                result.setText(r);
        }
        catch(Exception ex)
        {       }
}
public static void main(String[] args)
{
        ClientLib applet = new ClientLib();
        JFrame frame = new JFrame();
        frame.setTitle("Book Details");
        frame.add(applet,BorderLayout.CENTER);
        frame.setSize(450,250);
        applet.init();
        frame.setVisible(true);
}       }
```

Output:



```
C:\Windows\System32\cmd.exe - java -cp mysql-connector-java-5.1.23-bin.jar;. RegisterLib
Microsoft Windows [Version 10.0.22518.1000]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Gagan\OneDrive\Documents\lekha_naik\LibraryInfo>javac InterfaceLib.java

C:\Users\Gagan\OneDrive\Documents\lekha_naik\LibraryInfo>javac ImplLib.java

C:\Users\Gagan\OneDrive\Documents\lekha_naik\LibraryInfo>javac RegisterLib.java

C:\Users\Gagan\OneDrive\Documents\lekha_naik\LibraryInfo>javac ClientLib.java

C:\Users\Gagan\OneDrive\Documents\lekha_naik\LibraryInfo>start rmiregistry

C:\Users\Gagan\OneDrive\Documents\lekha_naik\LibraryInfo>java -cp mysql-connector-java-5.1.23-bin.jar;. RegisterLib
Driver Registered
Database connected
Result:

  ID:    1011

  Book Name:     Twilight

  Author:        Stephine Myere
```

```
C:\Windows\System32\cmd.exe - java ClientLib
Microsoft Windows [Version 10.0.22518.1000]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Gagan\OneDrive\Documents\lekha_naik\LibraryInfo>java ClientLib
Result:

  ID:    1011

  Book Name:     Twilight

  Author:        Stephine Myere
```

Practical No. 04

Programs on Remote Object Communication

Aim: 3) Using MySQL create Elecrtic_Bill database. Create table Bill (consumer_name, bill_due_date, bill_amount) and retrieve the Bill information from the Elecrtic_Bill database using Remote Object Communication concept.

Code:

InterfaceBill.java

```
import java.rmi.*;
public interface InterfaceBill extends Remote
{
   public String[] findBill(String name)throws RemoteException;
}
```

ImplBill.java

```
import java.rmi.*;
import java.rmi.server.*;
import java.sql.*;
public class ImplBill extends UnicastRemoteObject implements InterfaceBill
{
	int find = 0;
	String info[];
	public ImplBill()throws RemoteException
	{
		super();
	}
	public String[] findBill(String name)throws RemoteException
	{
		try
		{
				Class.forName("com.mysql.jdbc.Driver");
				System.out.println("Driver Registered");
				Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/elecrtic_bill","root","");
				System.out.println("Database connected");
				Statement stmt = conn.createStatement();
				String str = "select * from Bill where consumer_name='"+name+"'";
				ResultSet rs = stmt.executeQuery(str);
				while(rs.next())
				{
					find++;
				}
				info = new String[find];
				ResultSet rst = stmt.executeQuery(str);
				for(int i = 0; i < find; i++)
```

```
                                {
                                        rst.next();
                                        info[i] = "Bill:\n  \n  Customer:  "+rst.getString(1)+"\n  \n
Due Date:       "+rst.getString(2)+"\n  \n  Amount:      Rs."+rst.getString(3);
                                        System.out.println(info[i]);
                                }
                }
                catch(Exception ex)
                {
                        System.out.println(ex);
                }
                return info;
        }       }
```

RegisterBill.java

```
import java.rmi.*;
import java.rmi.registry.*;
public class RegisterBill
{
        public static void main(String[] agrs)
        {
                try
                {
                        ImplBill obj = new ImplBill();
                        Registry reg = LocateRegistry.getRegistry();
                        Naming.rebind("ImplBill",obj);
                }
                catch(Exception e)
                {}
        }
}
```

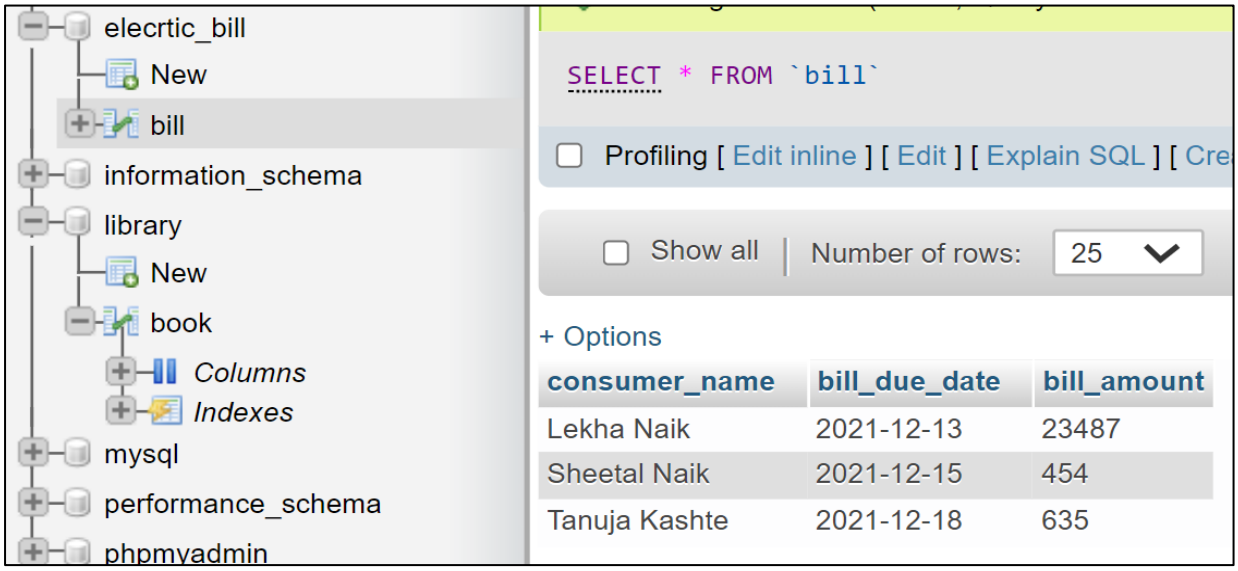
ClientBill.java

```
import java.rmi.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class ClientBill extends JApplet
{
        private InterfaceBill electric;
        private JButton getBill = new JButton("Get Bill");
        private JTextField name = new JTextField();
        private JLabel lname = new JLabel("Customer");
        private JLabel result = new JLabel("Bill");
```

```
public void init()
{
	try
	{
		electric = (InterfaceBill)Naming.lookup("ImplBill");
	}
	catch(Exception e)
	{	}
	JPanel panel = new JPanel();
		panel.setLayout(null);
		lname.setBounds(15, 15, 75, 33);
		panel.add(lname);
		name.setBounds(95, 15, 150, 33);
		panel.add(name);
		result.setBounds(15, 65, 750, 45);
		panel.add(result);
		getBill.setBounds(115, 130, 150, 33);
		panel.add(getBill, BorderLayout.CENTER);
		add(panel,BorderLayout.CENTER);
	getBill.addActionListener(new ActionListener()
	{
		public void actionPerformed(ActionEvent ae)
		{
			getBill();
		}
	}	);
}
public void getBill()
{
	try
	{
		String[] str = electric.findBill(name.getText());
		String r = "";
		for(int i = 0; i<str.length; i++)
				{
					r += " \n"+str[i];
					System.out.println(str[i]);
```

```
                                        }
                        result.setText(r);
                }
                catch(Exception ex)
                {       }
}
public static void main(String[] args)
{
                ClientBill applet = new ClientBill();
                JFrame frame = new JFrame();
                frame.setTitle("Bill Details");
                frame.add(applet,BorderLayout.CENTER);
                frame.setSize(450,250);
                applet.init();
                frame.setVisible(true);
}       }
```

Output:



```
C:\Windows\System32\cmd.exe - java -cp mysql-connector-java-5.1.23-bin.jar;. RegisterBill
Microsoft Windows [Version 10.0.22518.1000]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Gagan\OneDrive\Documents\lekha_naik\ElectricBill>javac InterfaceBill.java

C:\Users\Gagan\OneDrive\Documents\lekha_naik\ElectricBill>javac ImplBill.java

C:\Users\Gagan\OneDrive\Documents\lekha_naik\ElectricBill>javac RegisterBill.java

C:\Users\Gagan\OneDrive\Documents\lekha_naik\ElectricBill>javac ClientBill.java

C:\Users\Gagan\OneDrive\Documents\lekha_naik\ElectricBill>start rmiregistry

C:\Users\Gagan\OneDrive\Documents\lekha_naik\ElectricBill>java -cp mysql-connector-java-5.1.23-bin.jar;. RegisterBill
Driver Registered
Database connected
Bill:

  Customer:      Lekha Naik

  Due Date:      2021-12-13

  Amount:        Rs.23487
```

```
C:\Windows\System32\cmd.exe - java ClientBill
Microsoft Windows [Version 10.0.22518.1000]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Gagan\OneDrive\Documents\lekha_naik\ElectricBill>java ClientBill
Bill:

  Customer:      Lekha Naik

  Due Date:      2021-12-13

  Amount:        Rs.23487
```

Practical No. 05

Program on Mutual Exclusion

Aim: Implementation of Mutual Exclusion using Token Ring technique.

Code:

TokenRingServer.java

```
import java.net.*;
public class TokenRingServer
{
public static DatagramSocket ds;
public static DatagramPacket dp;
public static void main(String[] args)throws Exception
{
        ds = new DatagramSocket(2000);
        while(true)
        {               byte[] buffer = new byte[1024];
                        dp=new DatagramPacket(buffer,buffer.length);
                        ds.receive(dp);
                        String msg=new String(dp.getData(),0,dp.getLength());
                        System.out.println("Message from:"+msg);
        }       }       }
```

TokenRingClient1.java

```
import java.net.*;
import java.io.*;
public class TokenRingClient1
{
        public static DatagramSocket ds;
        public static DatagramPacket dp;
        public static BufferedReader br;
        public static void main(String[] args)throws Exception
        {
                boolean hasToken = true;
                ds = new DatagramSocket(2001);
                while(true)
                {
                        if(hasToken==true)
                        {
                                System.out.println("Do you want to Enter CS? (yes/no): ");
                                br=new BufferedReader(new InputStreamReader(System.in));
                                String choice =br.readLine();
                                if(choice.equalsIgnoreCase("yes"))
                                {
                                        System.out.println("The Client/Process is ready to Write");
                                        System.out.println("Enter the Message: ");
```

```
                                        br=new BufferedReader(new
InputStreamReader(System.in));
                                        String msg = "Client1-->"+br.readLine();
                                        dp=new
DatagramPacket(msg.getBytes(),msg.length(),InetAddress.getLocalHost(),2000);
                                        ds.send(dp);
                                        System.out.println("Message Sent");
                                }
                                else if(choice.equalsIgnoreCase("no"))
                                {
                                        System.out.println("I am Not ready to Enter the Critical
Section");
                                        String msg1="Token";
                                        dp=new
DatagramPacket(msg1.getBytes(),msg1.length(),InetAddress.getLocalHost(),2002);
                                        ds.send(dp);
                                        hasToken=false;
                                }                       }
                        else
                        {
                                System.out.println("Waiting for Token");
                                byte[] buffer = new byte[2048];
                                dp=new DatagramPacket(buffer,buffer.length);
                                ds.receive(dp);
                                String prevProcessMsg=new String(dp.getData(),0,dp.getLength());
                                System.out.println("PreviousProcessMsg is: "+prevProcessMsg);
                                if(prevProcessMsg.equals("Token"))
                                {
                                        hasToken=true;
                                        System.out.println("I have Token now");
                                }
                        }       }       }       }
```

TokenRingClient2.java

```
import java.net.*;
import java.io.*;
public class TokenRingClient2
{
        public static DatagramSocket ds;
        public static DatagramPacket dp;
        public static BufferedReader br;
        public static void main(String[] args)throws Exception
        {
                boolean hasToken = true;
                ds = new DatagramSocket(2002);
                while(true)
                {
                        if(hasToken==true)
                        {
```

```
                                System.out.println("Do you want to Enter CS? (yes/no): ");
                                br=new BufferedReader(new InputStreamReader(System.in));
                                String choice =br.readLine();
                                if(choice.equalsIgnoreCase("yes"))
                                {
                                        System.out.println("The Client/Process is Ready to Write");
                                        System.out.println("Enter the Message: ");
                                        br=new BufferedReader(new
InputStreamReader(System.in));
                                        String msg = "Client1-->"+br.readLine();
                                        dp=new
DatagramPacket(msg.getBytes(),msg.length(),InetAddress.getLocalHost(),2000);
                                        ds.send(dp);
                                        System.out.println("Message Sent");
                                }
                                else if(choice.equalsIgnoreCase("no"))
                                {
                                        System.out.println("I am Not Ready to Enter the Critical
Section");
                                        String msg1="Token";
                                        dp=new
DatagramPacket(msg1.getBytes(),msg1.length(),InetAddress.getLocalHost(),2003);
                                        ds.send(dp);
                                        hasToken=false;
                                }                               }
                        else
                        {
                                System.out.println("Waiting for Token");
                                byte[] buffer = new byte[2048];
                                dp=new DatagramPacket(buffer,buffer.length);
                                ds.receive(dp);
                                String prevProcessMsg=new String(dp.getData(),0,dp.getLength());
                                System.out.println("PreviousProcessMsg is "+prevProcessMsg);
                                if(prevProcessMsg.equals("Token"))
                                {
                                        hasToken=true;
                                        System.out.println("I have Token now");
                                }
                        }       }
        }       }
```

TokenRingClient3.java

```
import java.net.*;
import java.io.*;
public class TokenRingClient3
{
        public static DatagramSocket ds;
        public static DatagramPacket dp;
        public static BufferedReader br;
```

```
	public static void main(String[] args)throws Exception
	{
		boolean hasToken = true;
		ds = new DatagramSocket(2003);
		while(true)
		{
			if(hasToken==true)
			{
				System.out.println("Do you want to Enter CS? (yes/no): ");
				br=new BufferedReader(new InputStreamReader(System.in));
				String choice =br.readLine();
				if(choice.equalsIgnoreCase("yes"))
				{
					System.out.println("The Client/Process is Ready to Write");
					System.out.println("Enter the Message: ");
					br=new BufferedReader(new
InputStreamReader(System.in));
					String msg = "Client1-->"+br.readLine();
					dp=new
DatagramPacket(msg.getBytes(),msg.length(),InetAddress.getLocalHost(),2000);
					ds.send(dp);
					System.out.println("Message Sent");
				}
				else if(choice.equalsIgnoreCase("no"))
				{
					System.out.println("I am Not Ready to Enter the Critical
Section");
					String msg1="'Token";
					dp=new
DatagramPacket(msg1.getBytes(),msg1.length(),InetAddress.getLocalHost(),2001);
					ds.send(dp);
					hasToken=false;
				}		}
			else
			{
				System.out.println("Waiting for Token");
				byte[] buffer = new byte[2048];
				dp=new DatagramPacket(buffer,buffer.length);
				ds.receive(dp);
				String prevProcessMsg=new String(dp.getData(),0,dp.getLength());
				System.out.println("PreviousProcessMsg is "+prevProcessMsg);
				if(prevProcessMsg.equals("Token"))
				{
					hasToken=true;
					System.out.println("I have Token now");
				}				}
		}	}	}
```

Output:

```
C:\Windows\System32\cmd.exe - java TokenRingServer
Microsoft Windows [Version 10.0.22518.1000]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Gagan\OneDrive\Documents\lekha_naik\MutualExclu>javac TokenRingServer.java

C:\Users\Gagan\OneDrive\Documents\lekha_naik\MutualExclu>javac TokenRingClient1.java

C:\Users\Gagan\OneDrive\Documents\lekha_naik\MutualExclu>javac TokenRingClient2.java

C:\Users\Gagan\OneDrive\Documents\lekha_naik\MutualExclu>javac TokenRingClient3.java

C:\Users\Gagan\OneDrive\Documents\lekha_naik\MutualExclu>java TokenRingServer
Message from:Client1-->Hi Good Morning
Message from:Client1-->Good Morning to you also
Message from:Client1-->It is SYMCA
Message from:Client1-->Merry Christmas to all
```

```
C:\Windows\System32\cmd.exe - java TokenRingClient1
Microsoft Windows [Version 10.0.22518.1000]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Gagan\OneDrive\Documents\lekha_naik\MutualExclu>java TokenRingClient1
Do you want to Enter CS? (yes/no):
yes
The Client/Process is ready to Write
Enter the Message:
Hi Good Morning
Message Sent
Do you want to Enter CS? (yes/no):
no
I am Not ready to Enter the Critical Section
Waiting for Token
PreviousProcessMsg is: Token
I have Token now
Do you want to Enter CS? (yes/no):
```

```
C:\Windows\System32\cmd.exe - java TokenRingClient2
Microsoft Windows [Version 10.0.22518.1000]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Gagan\OneDrive\Documents\lekha_naik\MutualExclu>java TokenRingClient2
Do you want to Enter CS? (yes/no):
yes
The Client/Process is Ready to Write
Enter the Message:
Good Morning to you also
Message Sent
Do you want to Enter CS? (yes/no):
yes
The Client/Process is Ready to Write
Enter the Message:
It is SYMCA
Message Sent
Do you want to Enter CS? (yes/no):
no
I am Not Ready to Enter the Critical Section
Waiting for Token
PreviousProcessMsg is Token
I have Token now
Do you want to Enter CS? (yes/no):
```

```
C:\Windows\System32\cmd.exe - java TokenRingClient3
Microsoft Windows [Version 10.0.22518.1000]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Gagan\OneDrive\Documents\lekha_naik\MutualExclu>java TokenRingClient3
Do you want to Enter CS? (yes/no):
no
I am Not Ready to Enter the Critical Section
Waiting for Token
PreviousProcessMsg is Token
I have Token now
Do you want to Enter CS? (yes/no):
yes
The Client/Process is Ready to Write
Enter the Message:
Merry Christmas to all
Message Sent
Do you want to Enter CS? (yes/no):
no
I am Not Ready to Enter the Critical Section
Waiting for Token
```

Practical No. 06

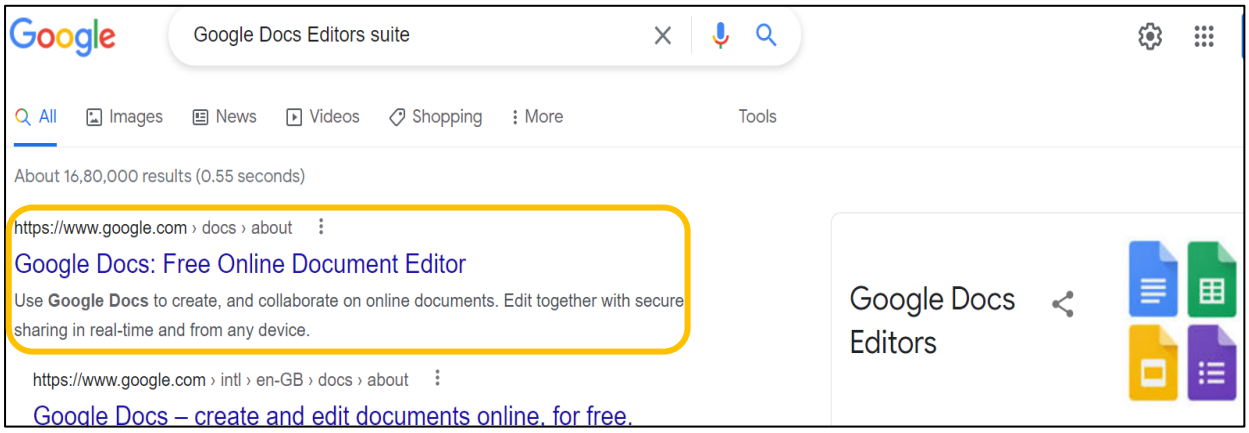
Implementation of Cloud Computing Services

Aim: Implementation of Storage as a Service using Google Docs.

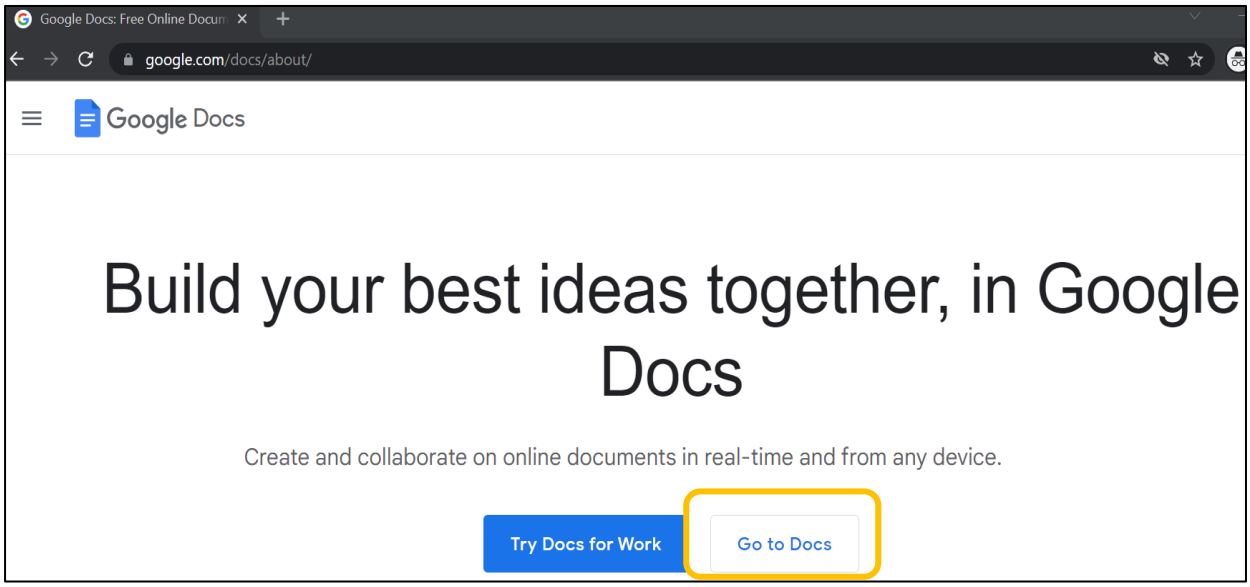
Desciption: ***Google Docs Editors*** is a web-based productivity office suite offered by Google within its Google Drive service. The suite includes ***Google Docs, Google Sheets, Google Slides, Google Drawings, Google Forms, Google Sites, and Google Keep***. The Google Docs Editors suite is available freely for users with personal Google accounts: through a web application, a set of mobile apps for Android and iOS, and a desktop application for Google's Chrome OS.

Implementation Steps:

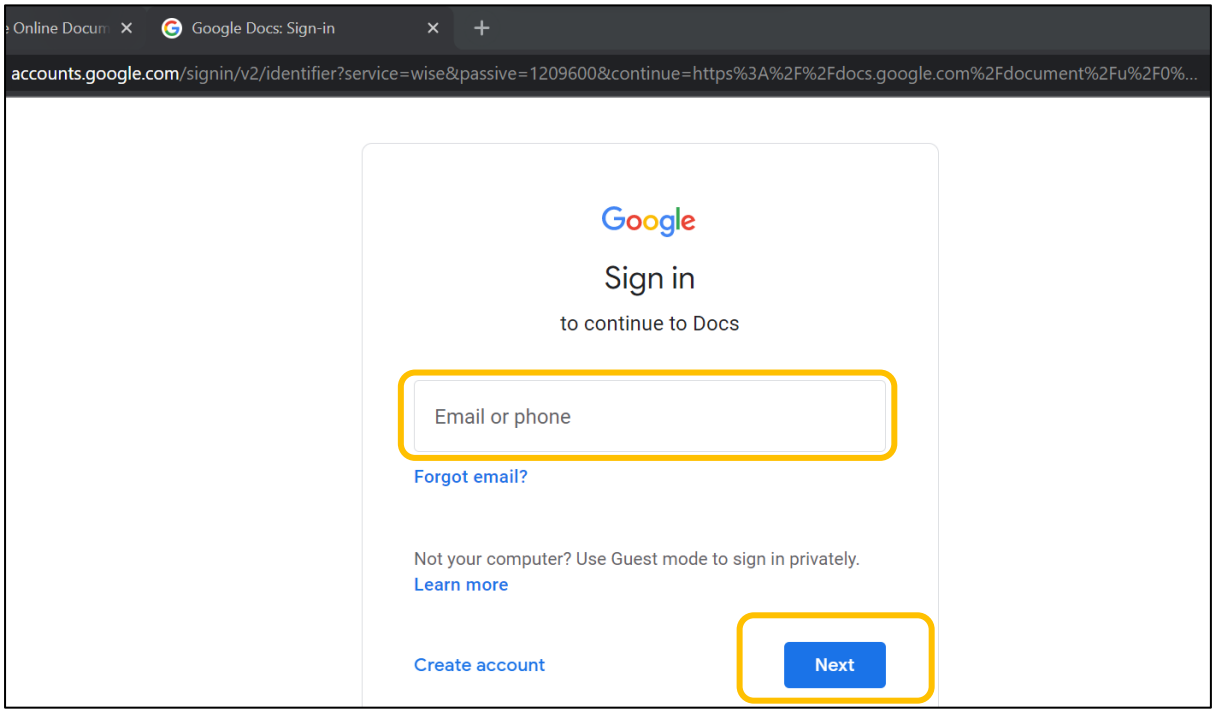
Step 1: In Web Browser Search Bar, write '**Google Docs Editors Suite**' and press Enter. Following results will be shown, click on below highlighted link for Google Docs.



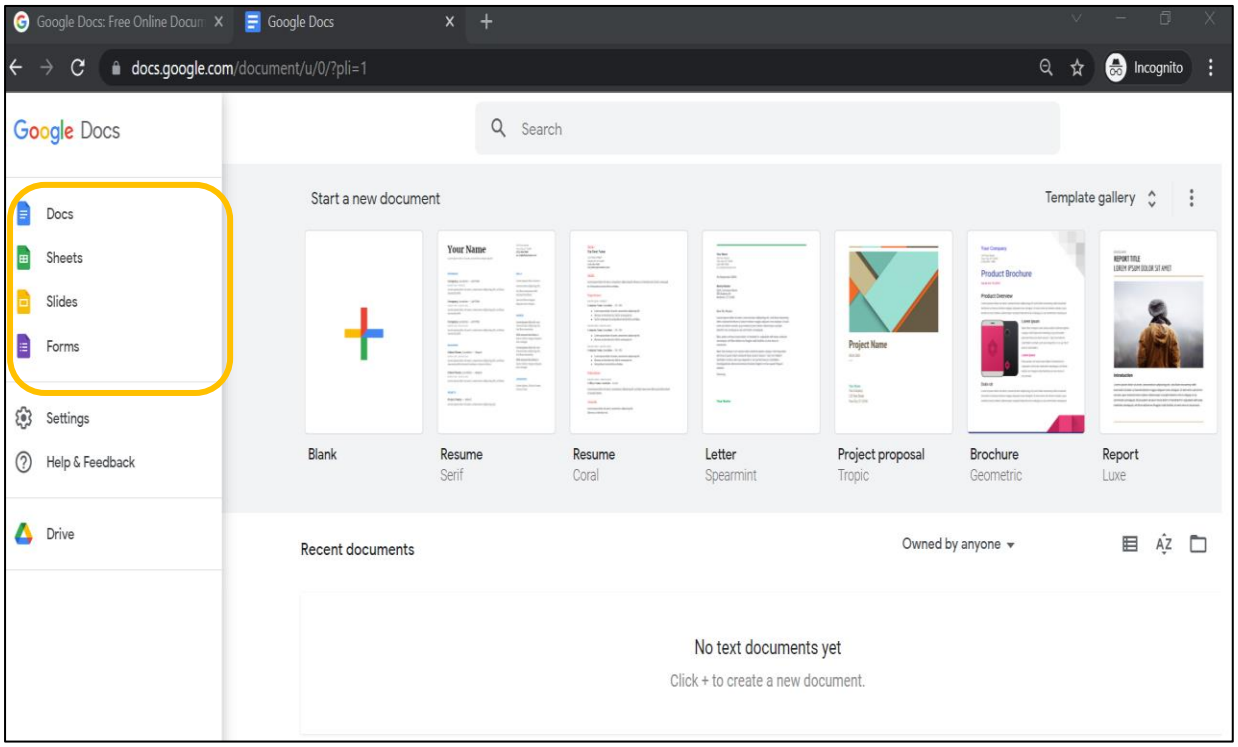
Step 2: Below page of '**Google Docs**' will be opened. Click on the '**Go to Docs**' button.



Step 3: Redirection to **Google Account '*Sign In*'** page will be done. **Log In** to your Google Account using E-mail address and password.

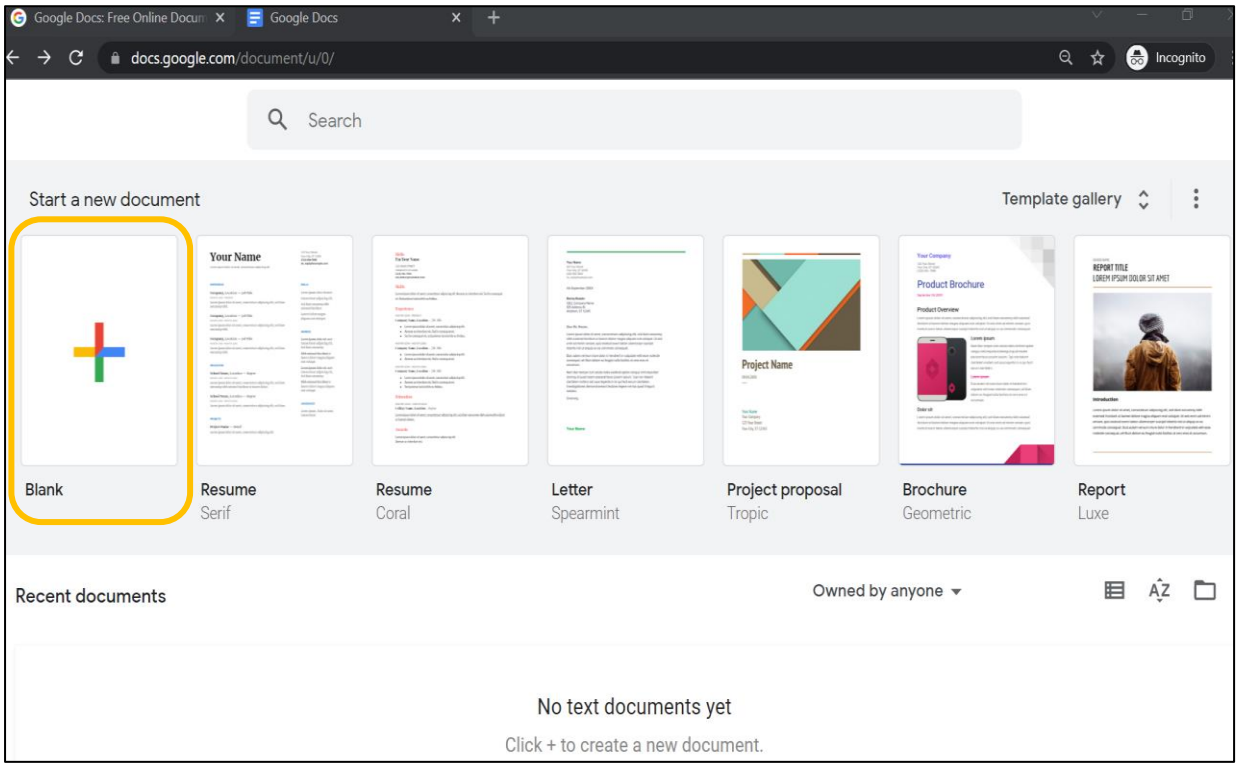


Step 4: After Signing In, following Start Page in **Google Docs** will be opened. You can Switch between ***Docs/Sheets/Slides/Forms*** by clicking on their options given in Side Bar.

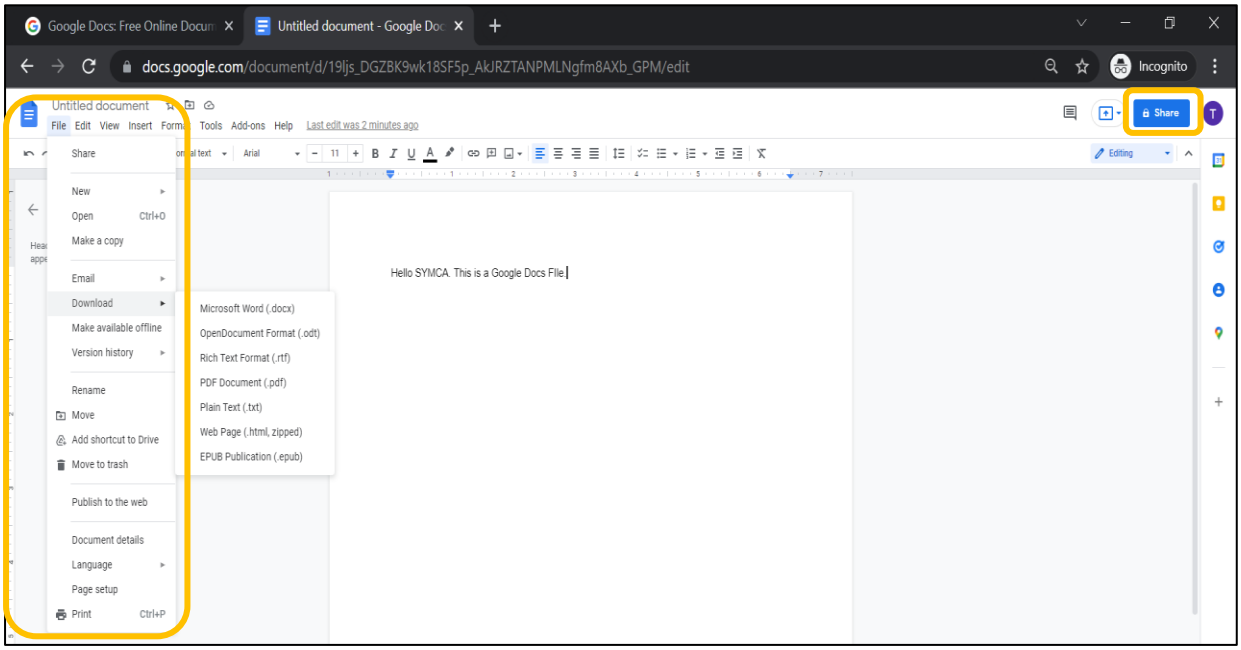


Creating Google Docs:

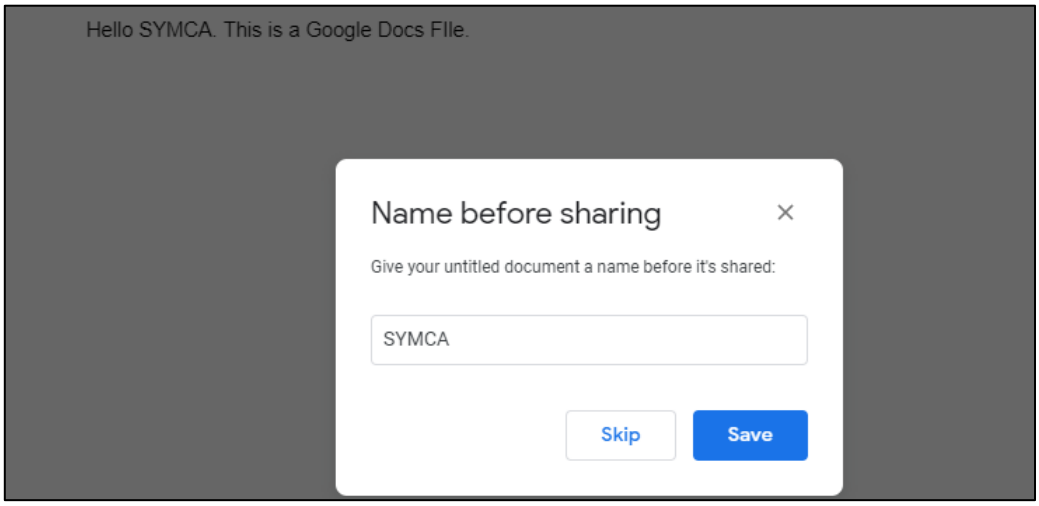
Step 1: From the '***Start a new document***' area click on '**+**' sign named '**Blank**' to **Create a New Google Doc File**. Also, ready ***Templates*** are available for various kinds of documents here which can be used by modifying as per needed.



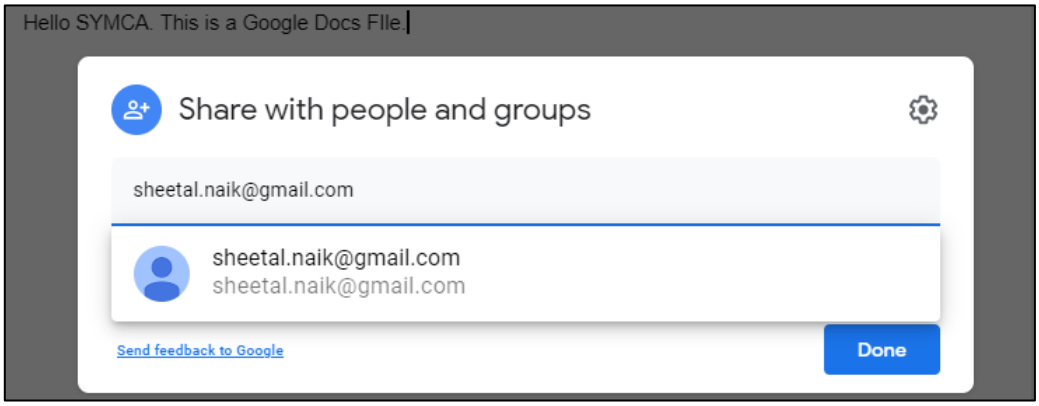
Step 2: Google Docs File will get opened. We can **Insert** and **Edit** the Document file as desired using *various functionality* provided by Google Docs and they are **AutoSaved** in the ***Cloud (Google Drive)*** of User Account. After which we can **Export** the file created or **Share** the *Document online* with help of the below shown option.



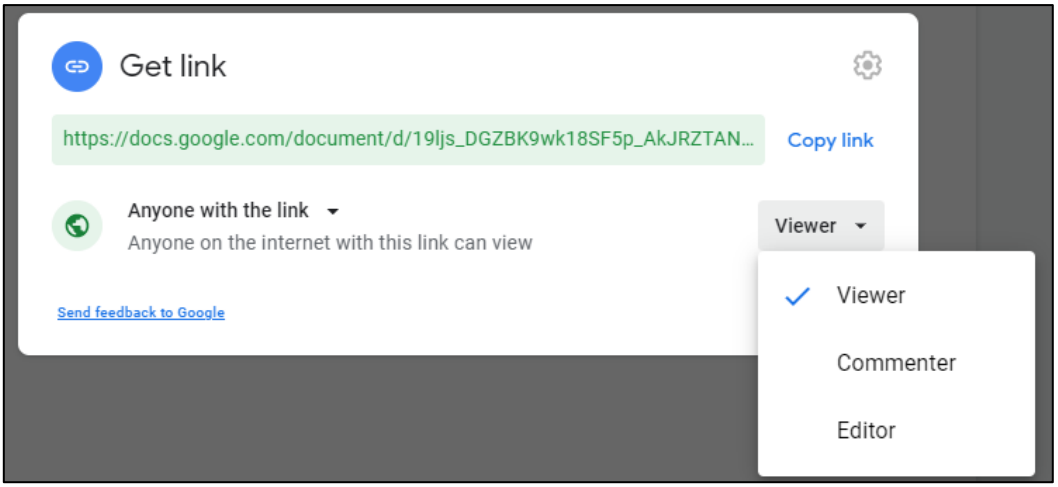
Step 3: Google Docs File can be shared using various ways by ***Importing File Offline*** or by ***Sharing with other Users Online***. When clicked on **Share** button, first we must *Name the File*.



By *adding* the **E-mail addresses** of Users, as shown below, we can *Share* the *Google Docs File online* through **Google Drive(Cloud)** with those Users.

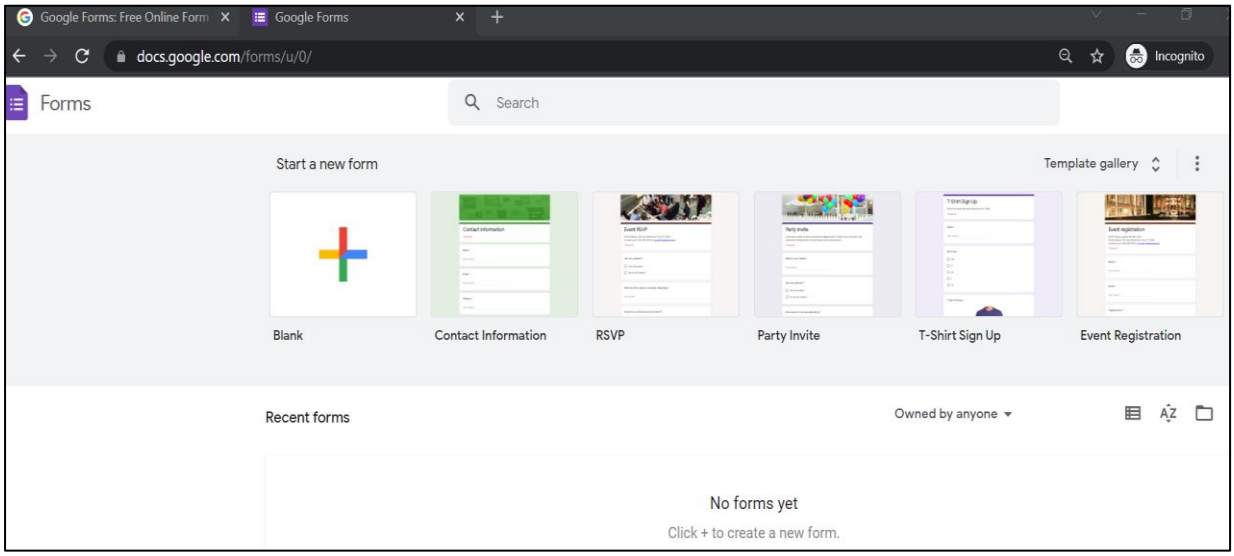


Also, Google Docs can be **shared through Link by giving Access Rights** as per Author as to whether User with Link should only View/Comment or also Edit.

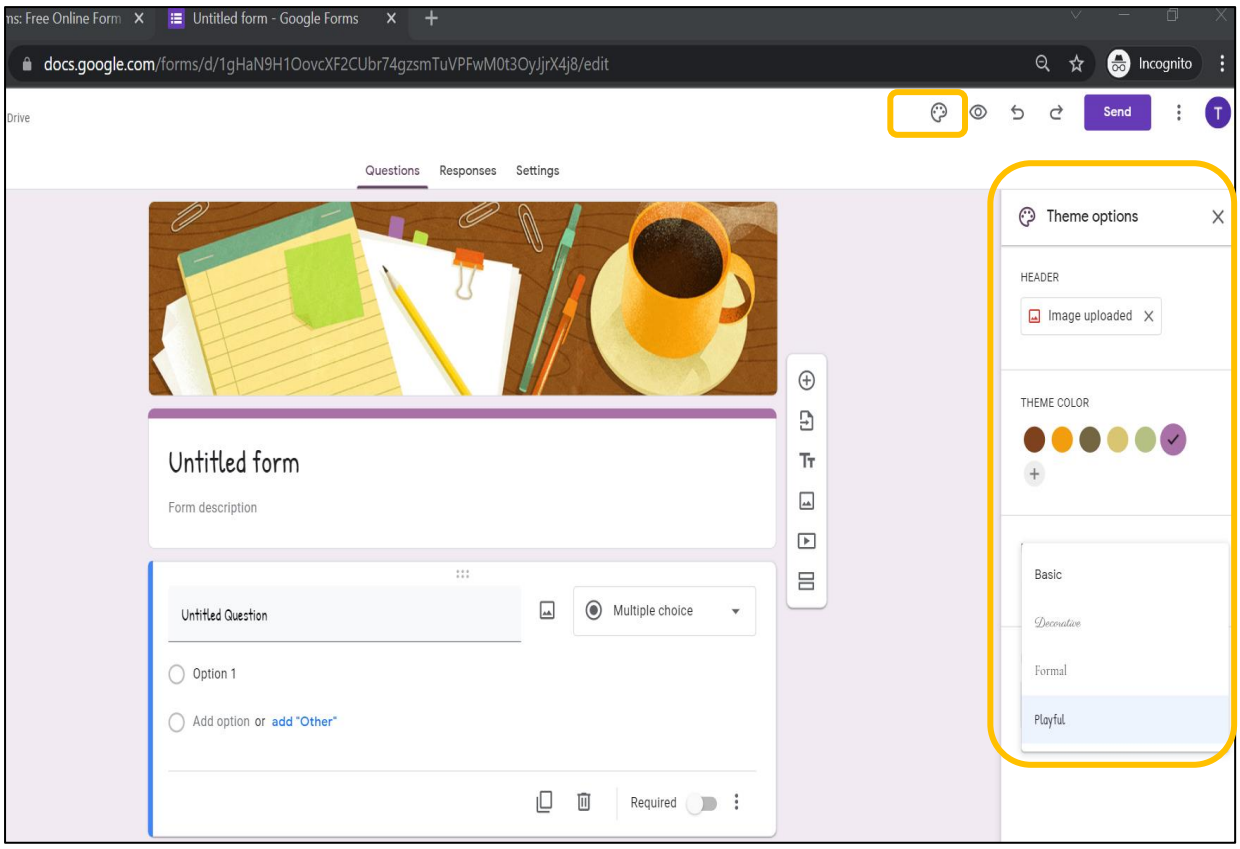


Creating Google Forms:

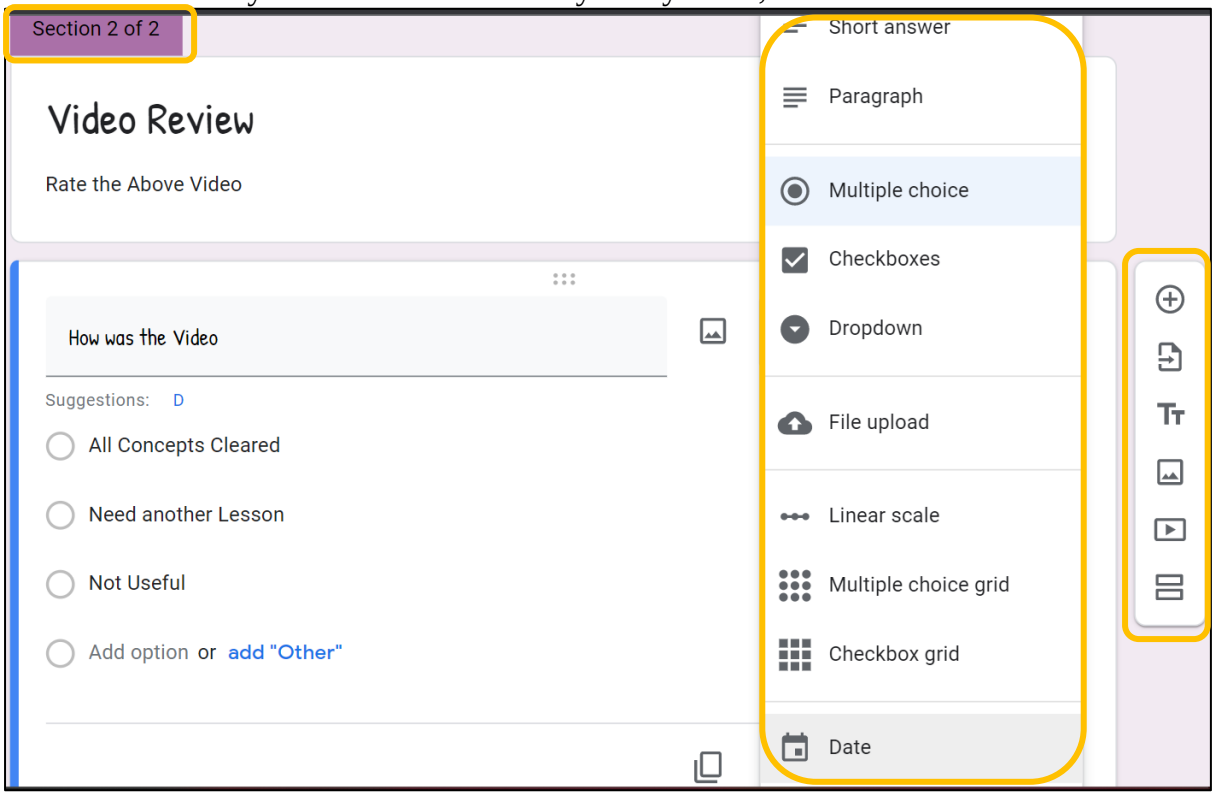
Step 1: For Creating Google Forms, select Forms option then from the '***Start a new form***' area click on '+' sign named '**Blank**' to **Create a New Google Form**. Also, ready ***Templates*** are available for various types of forms here which can be used on the go.



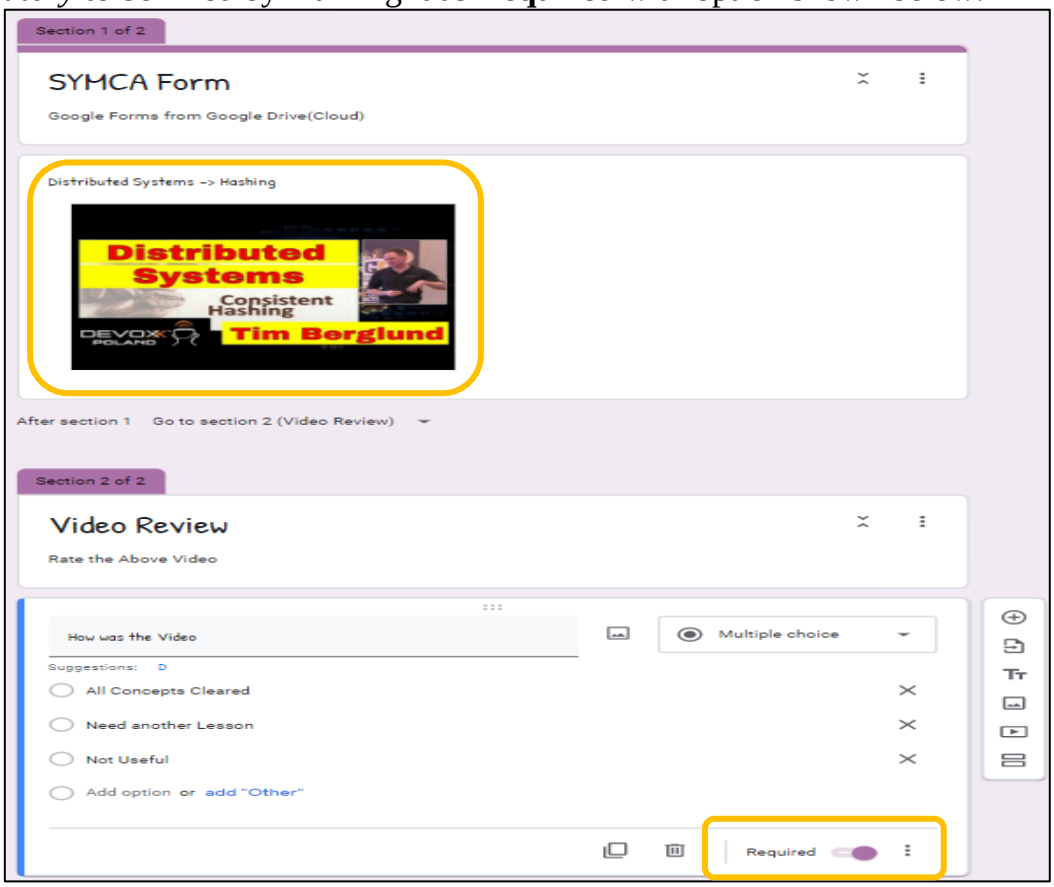
Step 2: Google Form will be opened. We can **Insert** and **Edit** the Form as desired using *various components* provided by Google Forms and they are **AutoSaved** in the ***Cloud (Google Drive)*** of User Account. Below **Theme Options** are shown using which **Theme Color, Background Color, Header Image, Font Style** can be set for particular Form.



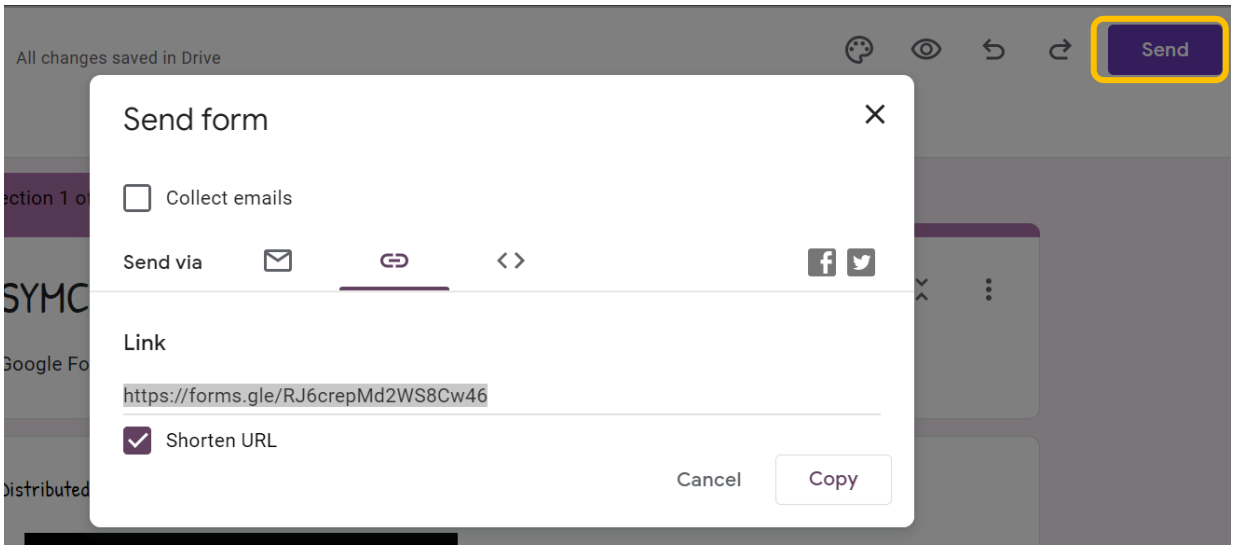
Step 2: Google Form provides with ***Components for Forms*** as shown below using which user-friendly forms can be made very easily. Also, we can add different **Sections**.



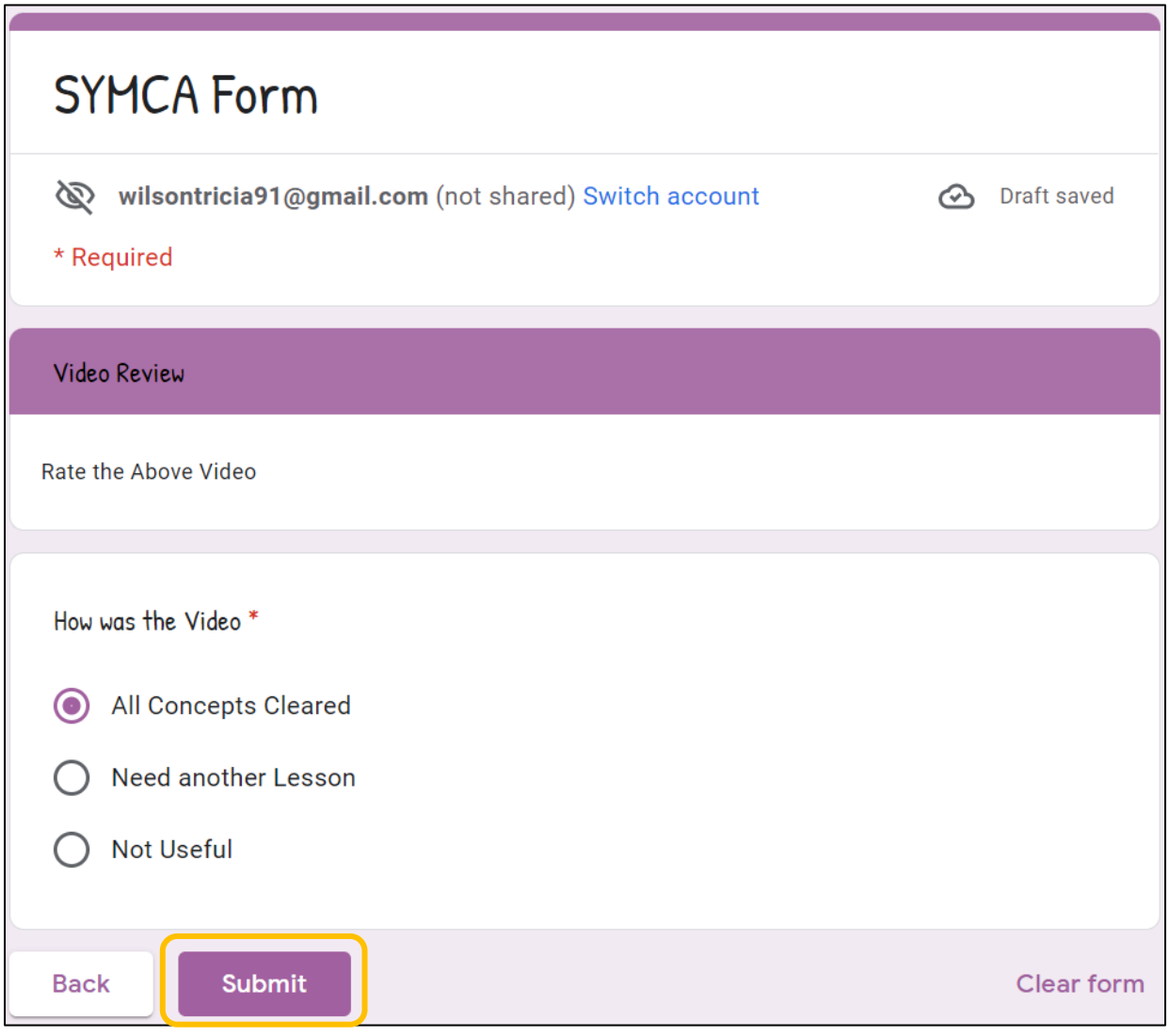
Video/Image can also be added to *Google Forms*. Certain Field can also made *mandatory* to be filled by marking it as **Required** with option shown below.



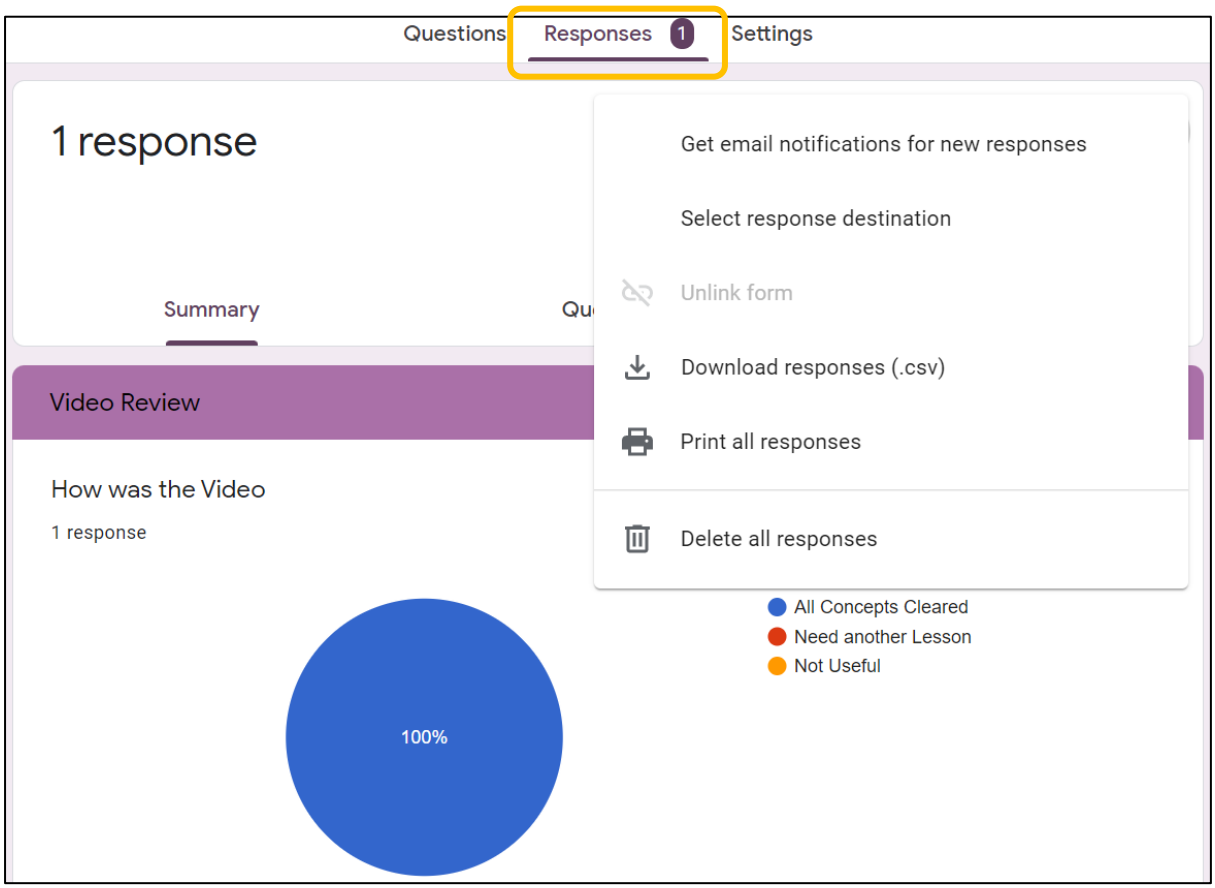
Step 3: When clicked on **Share** button**, Google Form** can be shared using various ways by *mailing* through **E-mail addresses** of Users or **sharing through Link** to the Form.



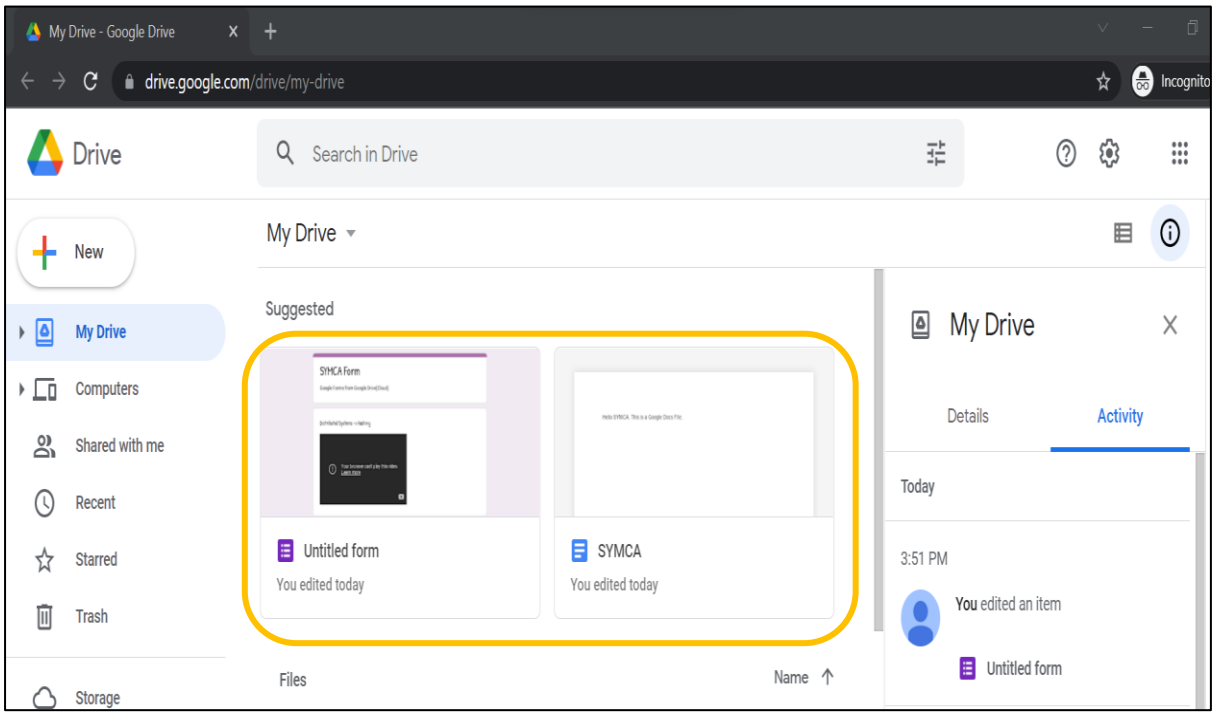
Step 4: Form User will attempt the from and **Submit** the Form Responses.



Step 5: Form creator can *check* the *Form Replies* received in the **Responses Section** as shown below. The responses can also be *exported in .csv* format offline.



Step 6: All **Google Docs Files** can be *accessed* when required from **Google Drive (Cloud)** in User Account.



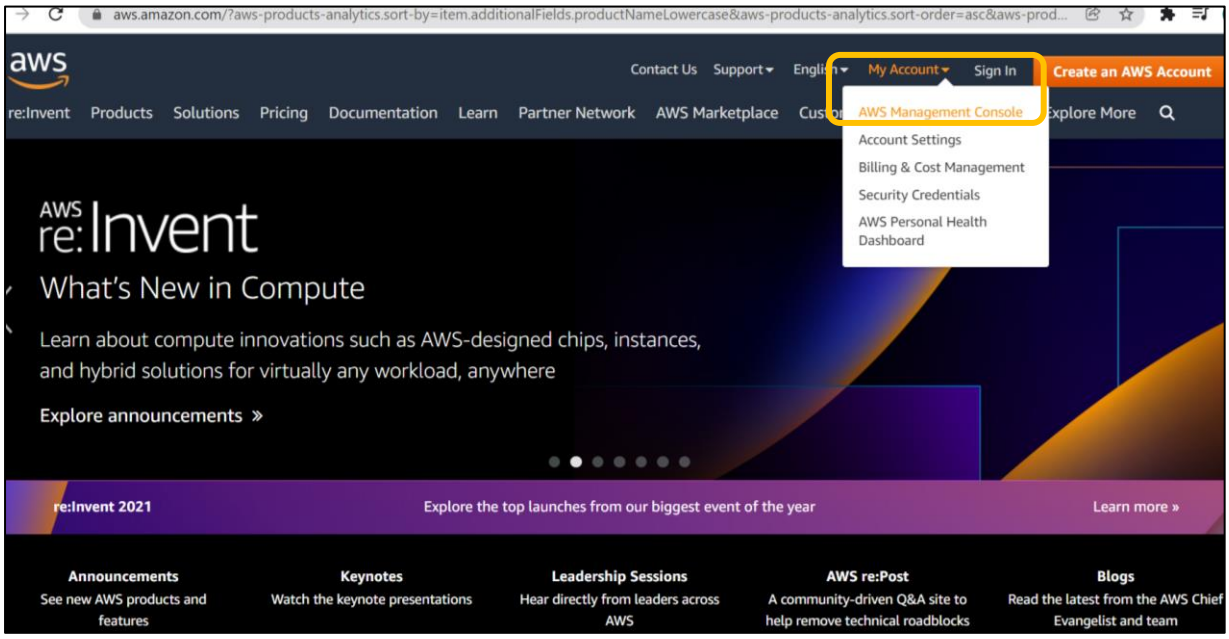
Practical No. 07

Implementation of Identity Management using Cloud Computing concept

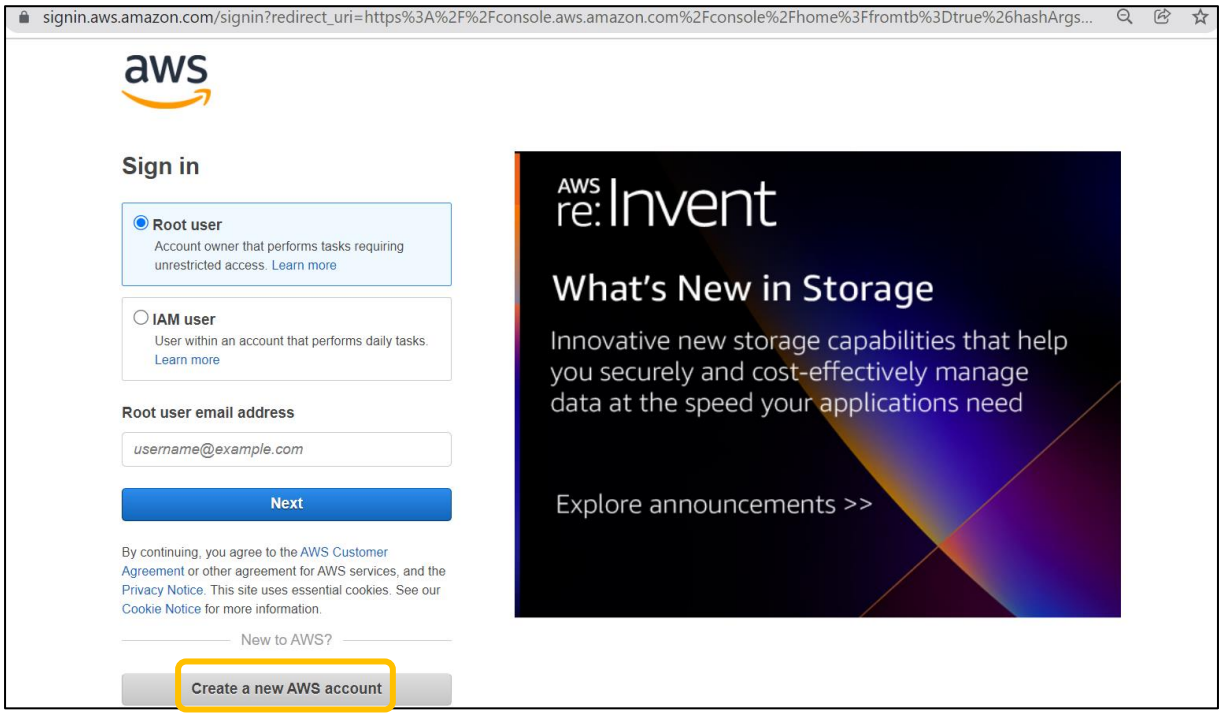
Aim: Implementation of Identity Management.

Implementation Steps:

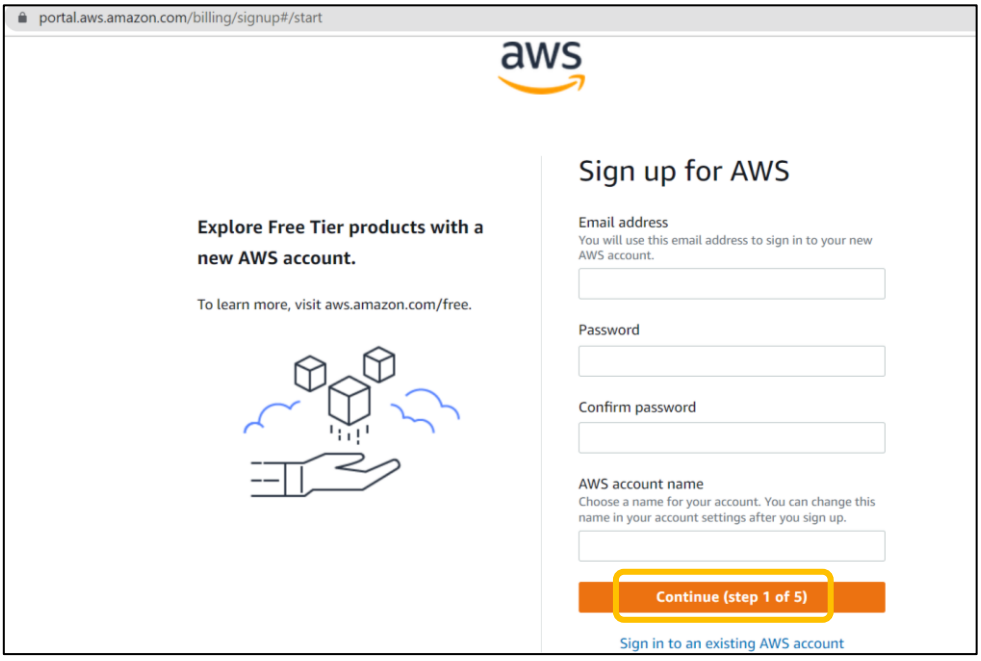
Step 1: Open the following link **https://aws.amazon.com/**
Go to **My Account -> AWS Management Console**



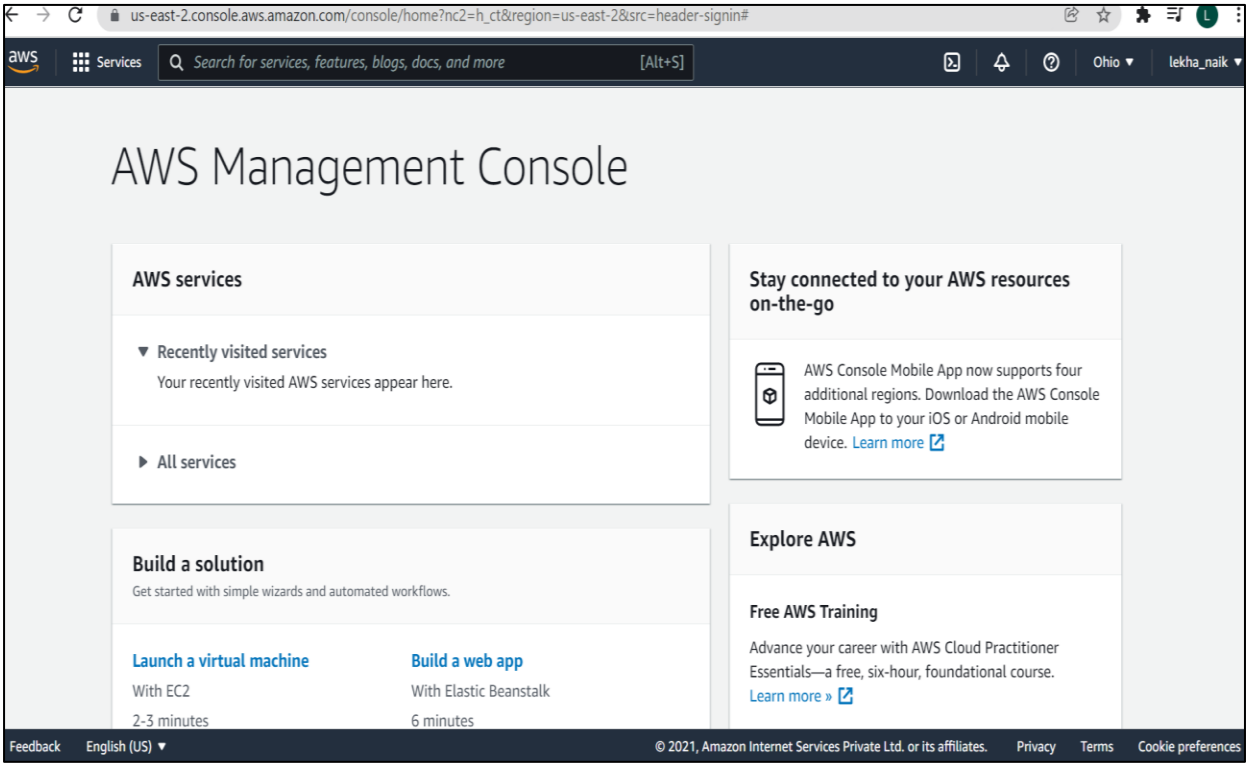
Step 2: Click on **Create a new AWS account**



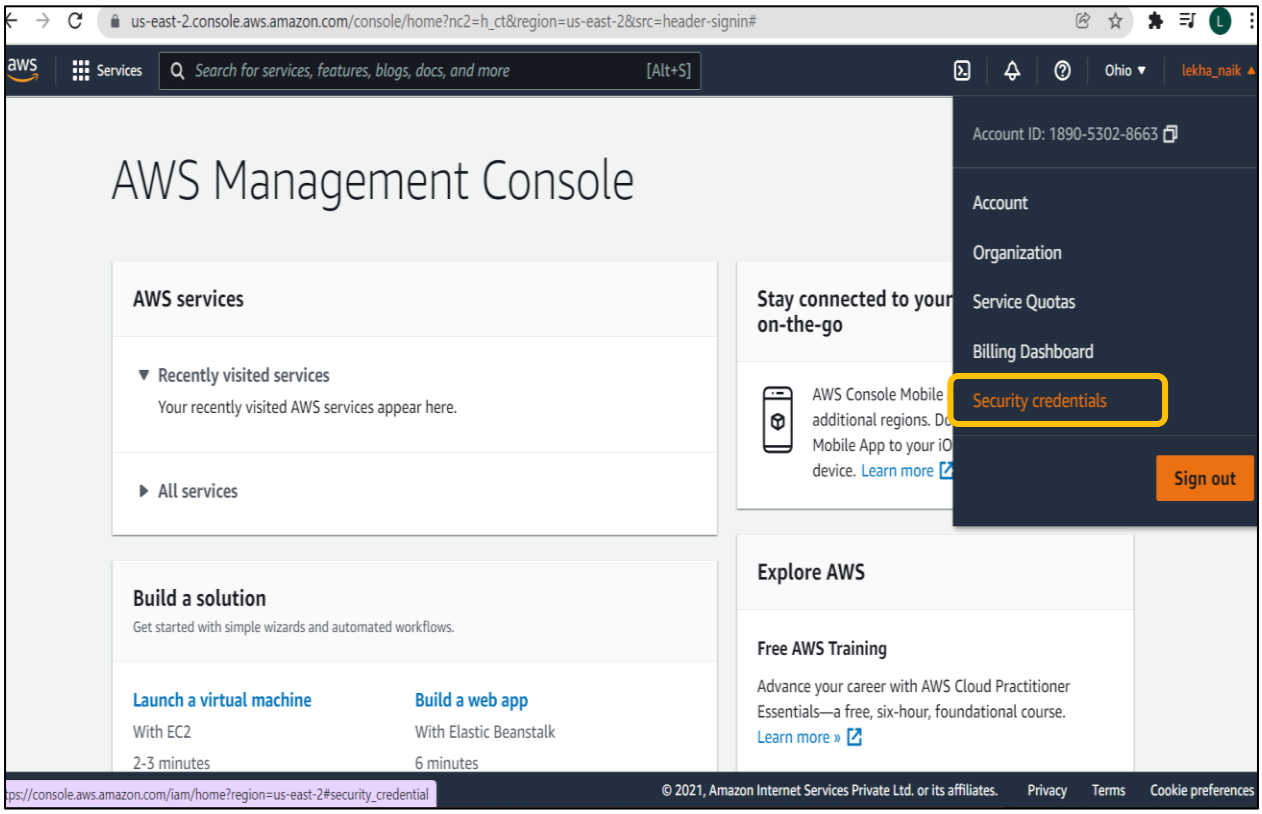
Step 3: Fill all the details and click on **Continue**. Fill your contact number and home address and click on **Create Account** and **Continue**.
Now most curtail step *AWS* will ask for *credit card and debit card details*. You must *Close the Browser*. Now again *Open the Link* https://aws.amazon.com/.



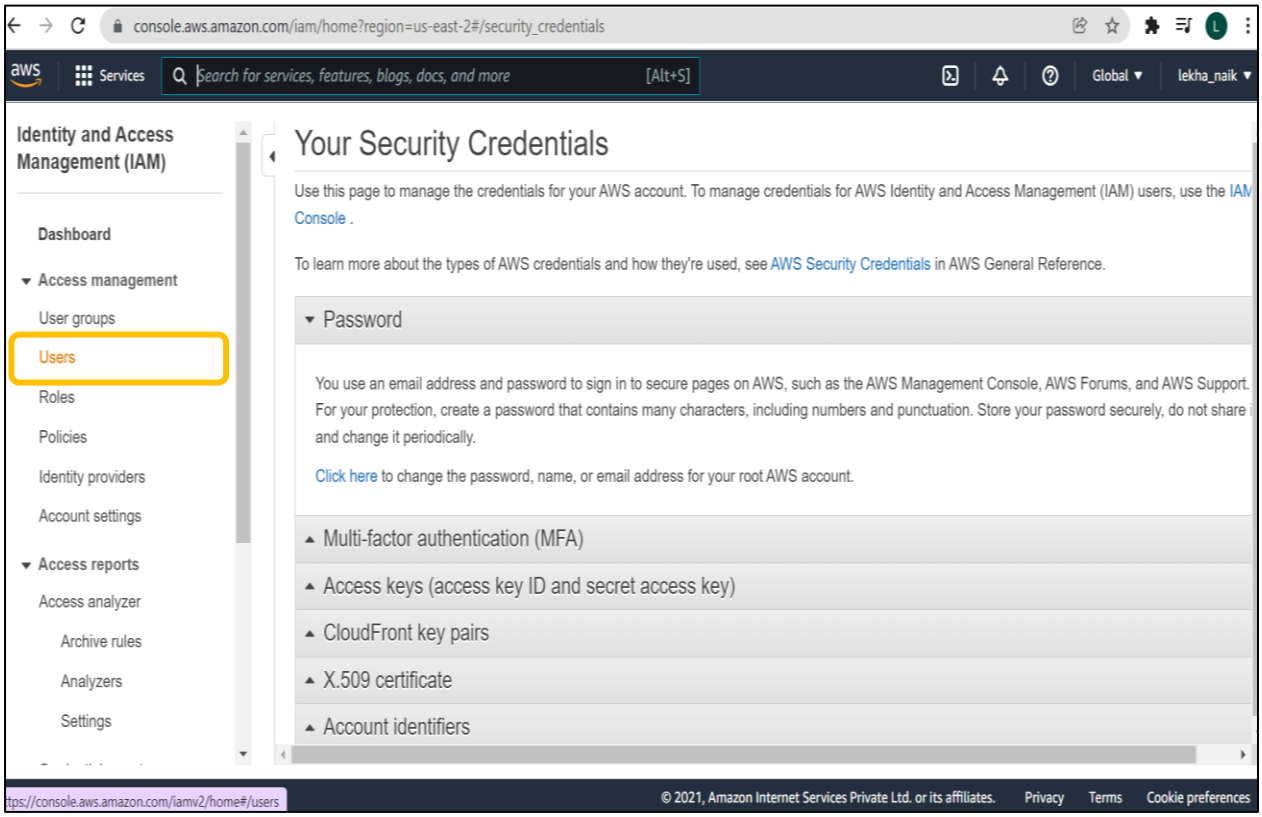
Step 4: Go to **My Account->AWS Management Console**
Enter your *ID* and click on **Next**, after that enter *password* and click on **Sign In**



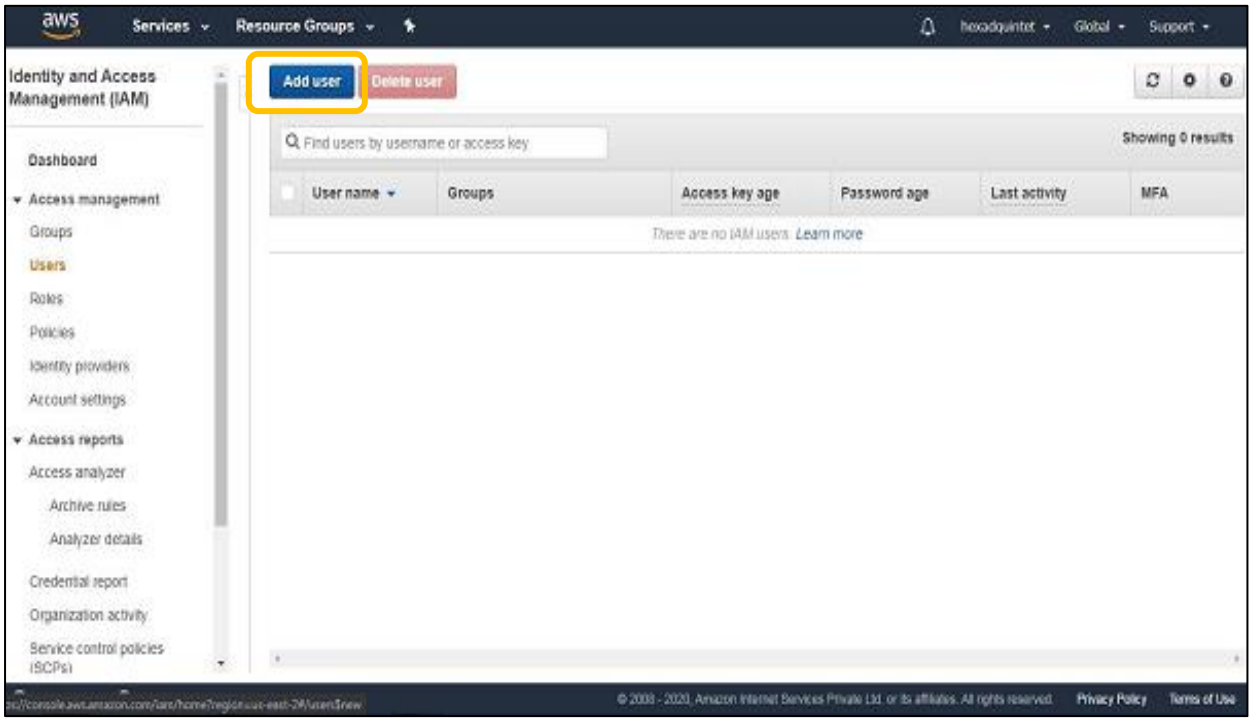
Step 5: Go to **Security credentials**



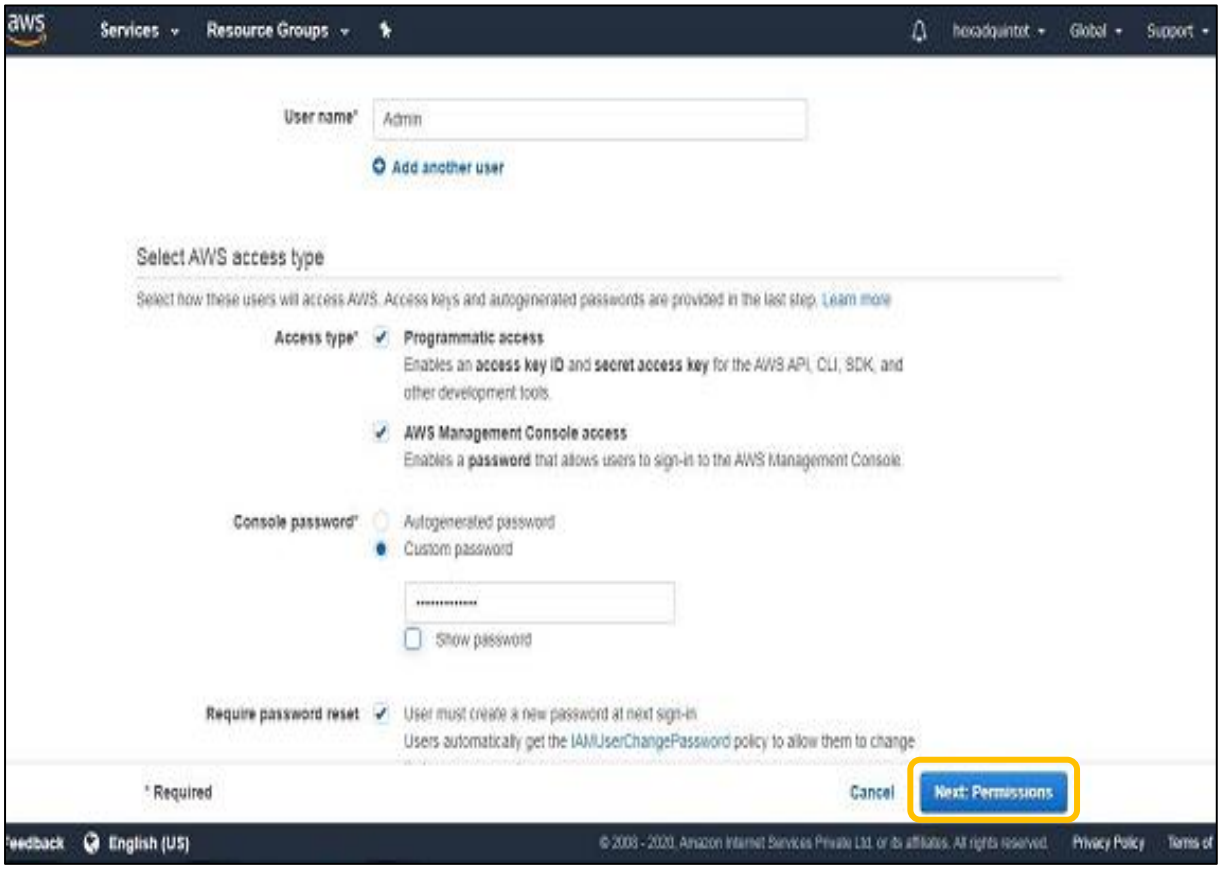
Step 6: Now **click on Users**



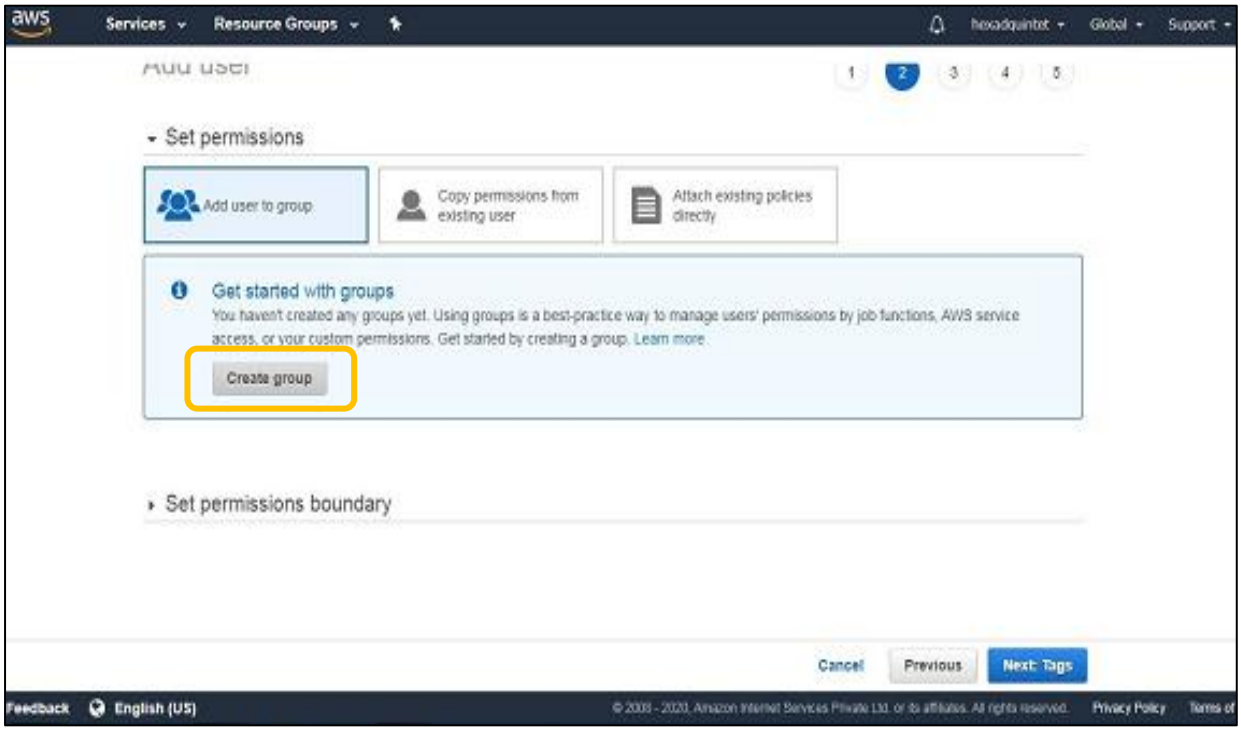
Step 7: Click on **Add user**



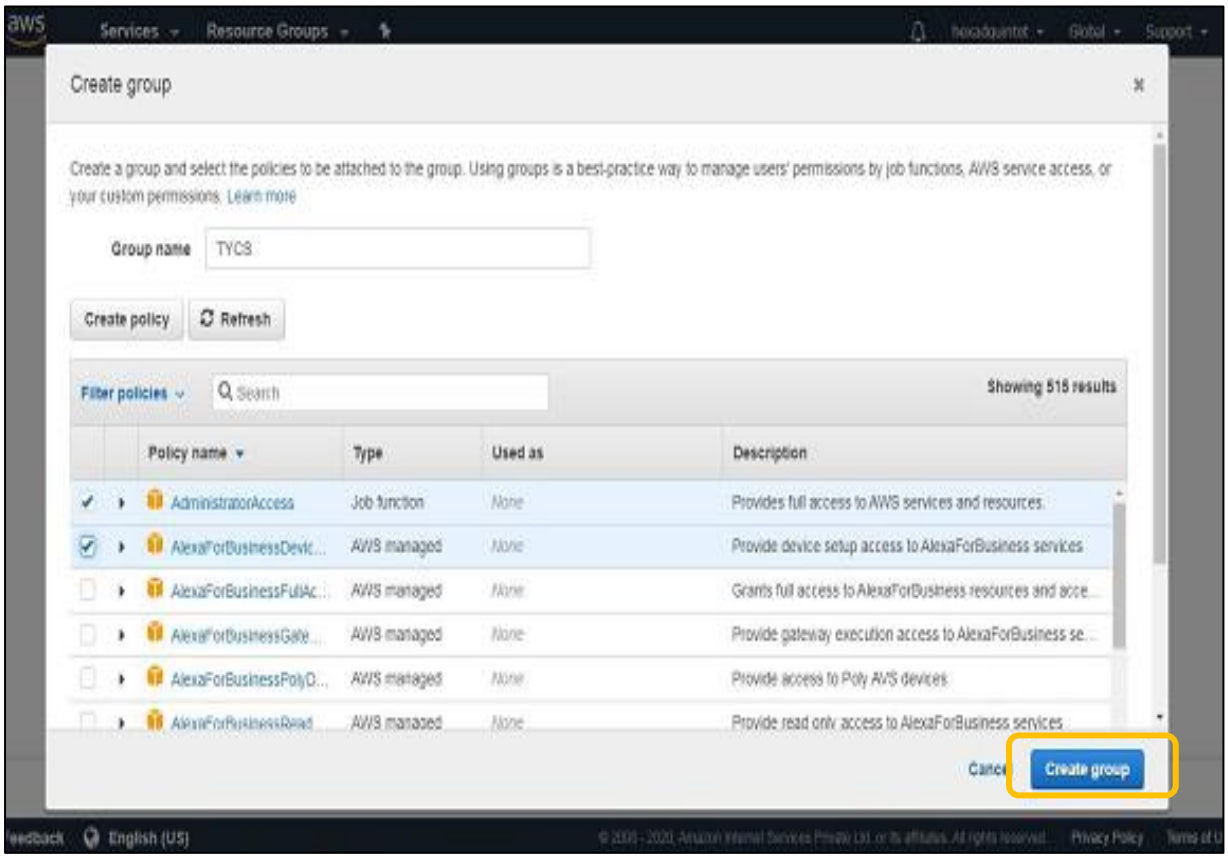
Step 8: Provide the **User name** and *check the check box* in front of **Programmatic access** and **AWS Management Console access** and enter the *password* for new user Click on **Custom password** and click on **Next Permission**



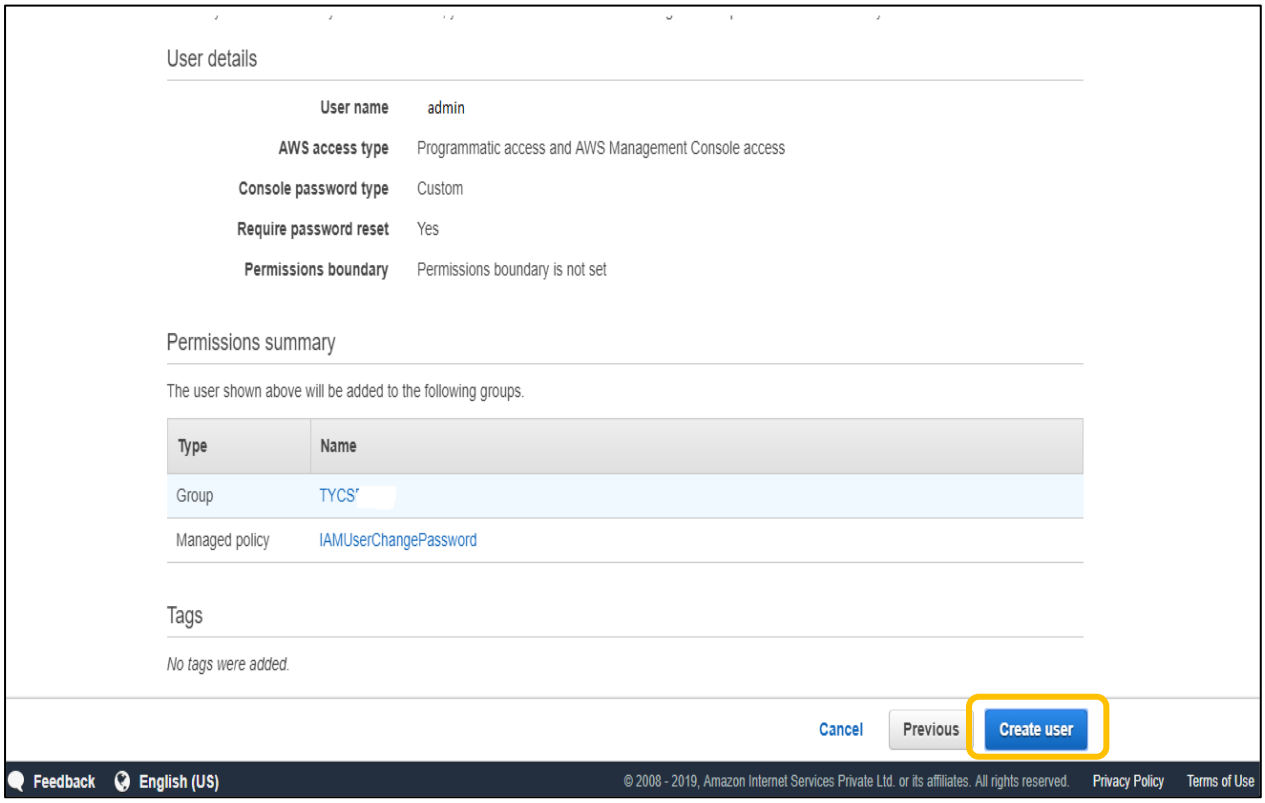
Step 9: Click on **Create group**



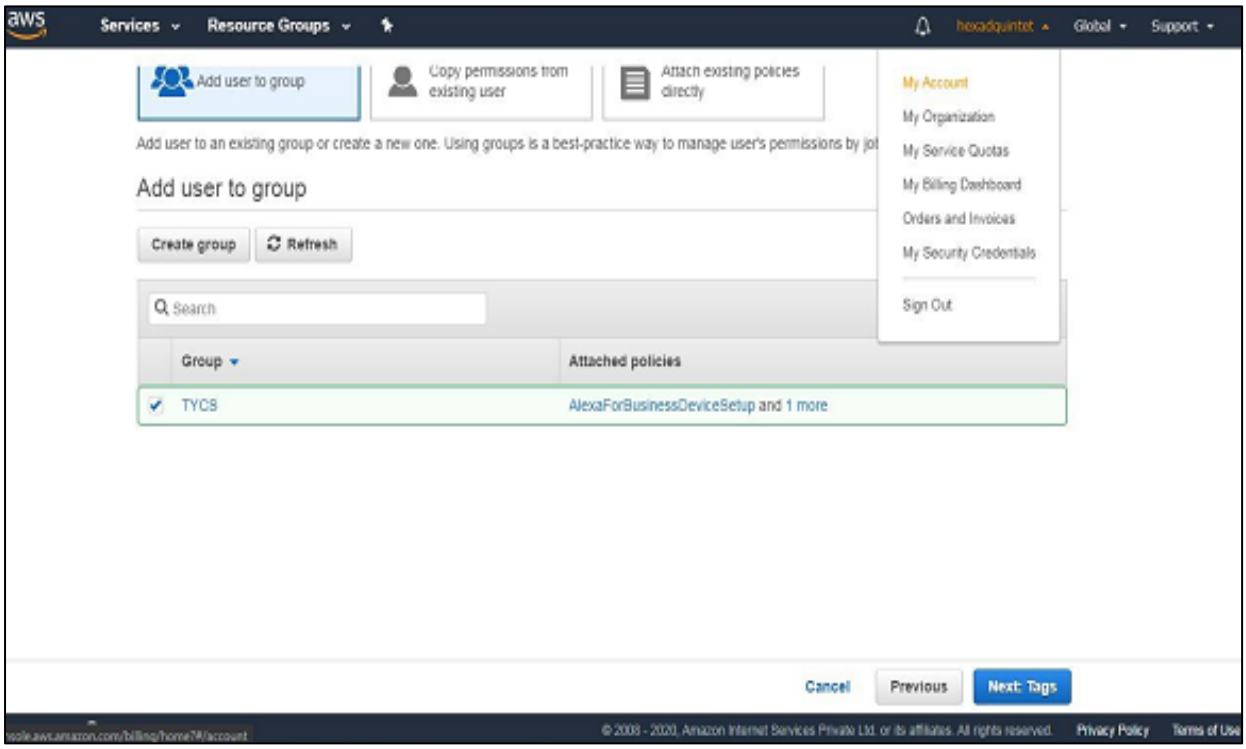
Step 10: Fill the *Information* and click on **Create group**



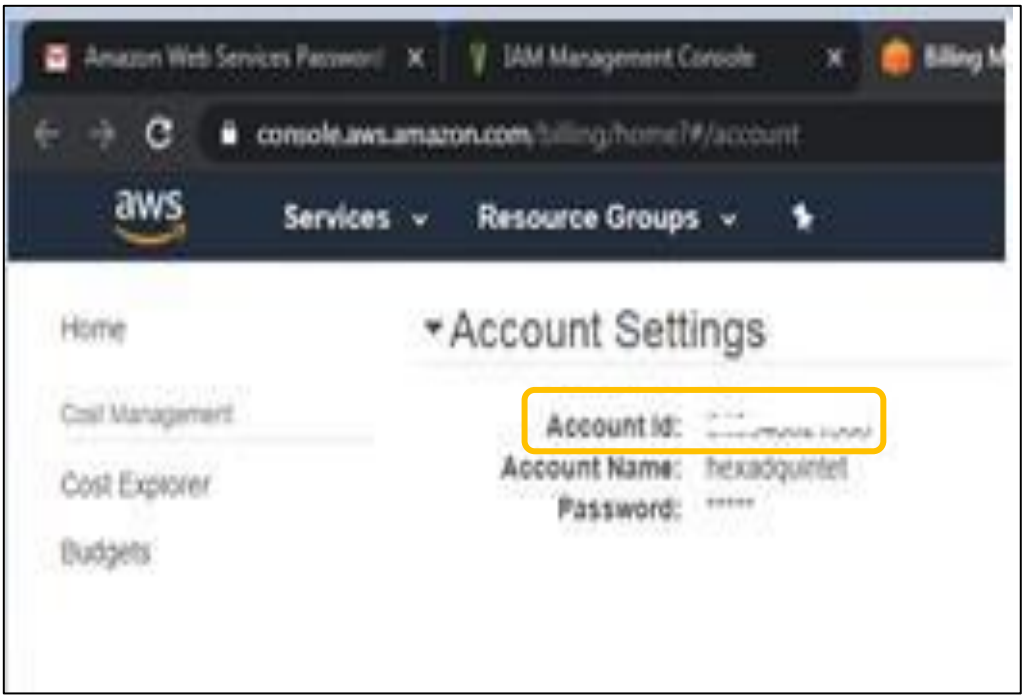
Step 11: Click on **Next Tag** *leave blank*, again click on **Next Review** *leave as it is* and click on **Create user**



Step 12: Click on **Close**

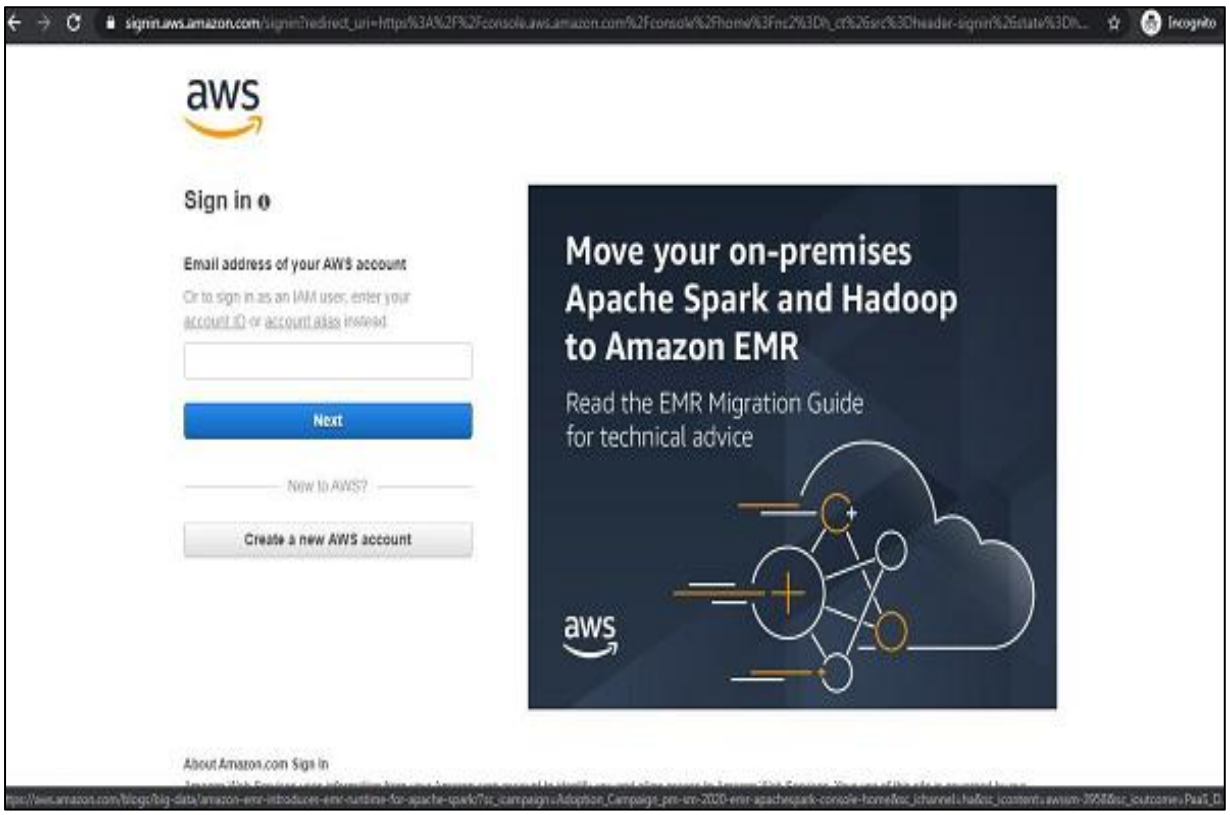


And *Copy* **Account ID**

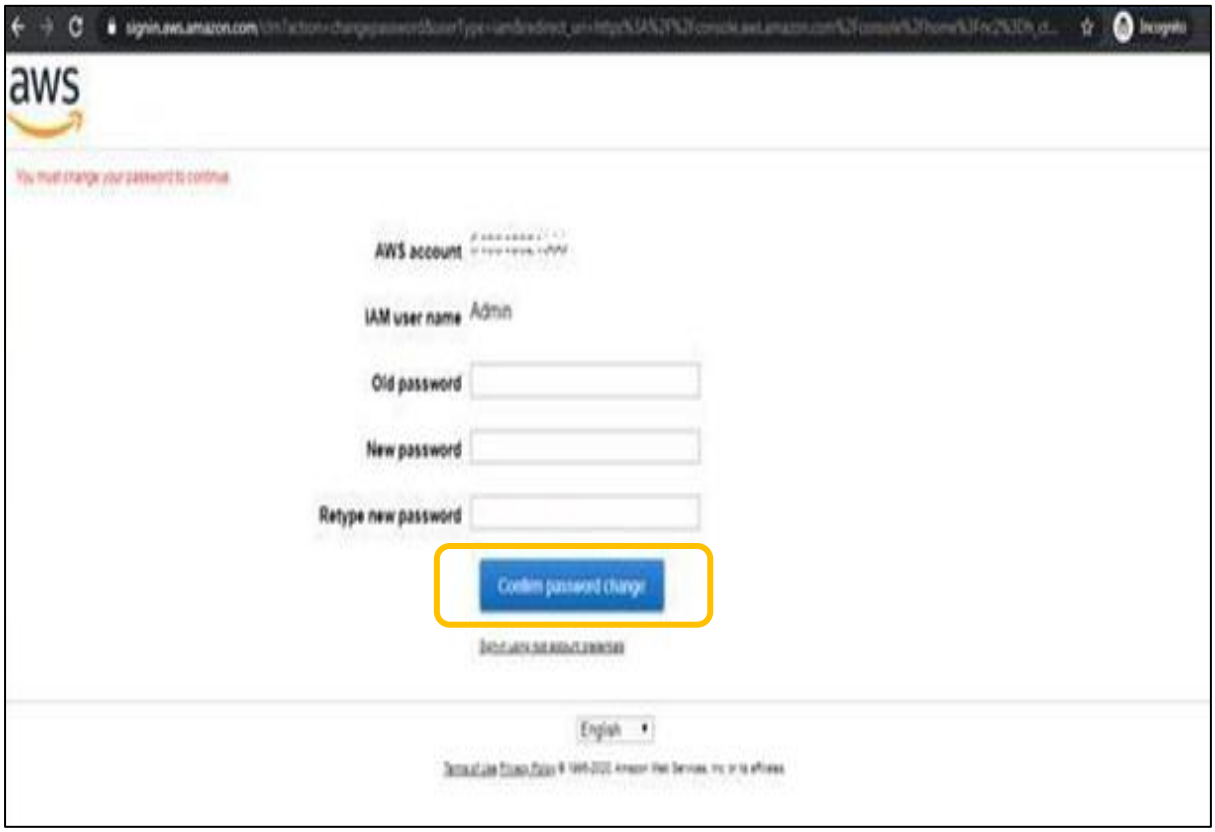


Now *Logout the Admin Account* and try to *Login as User* (newly created).

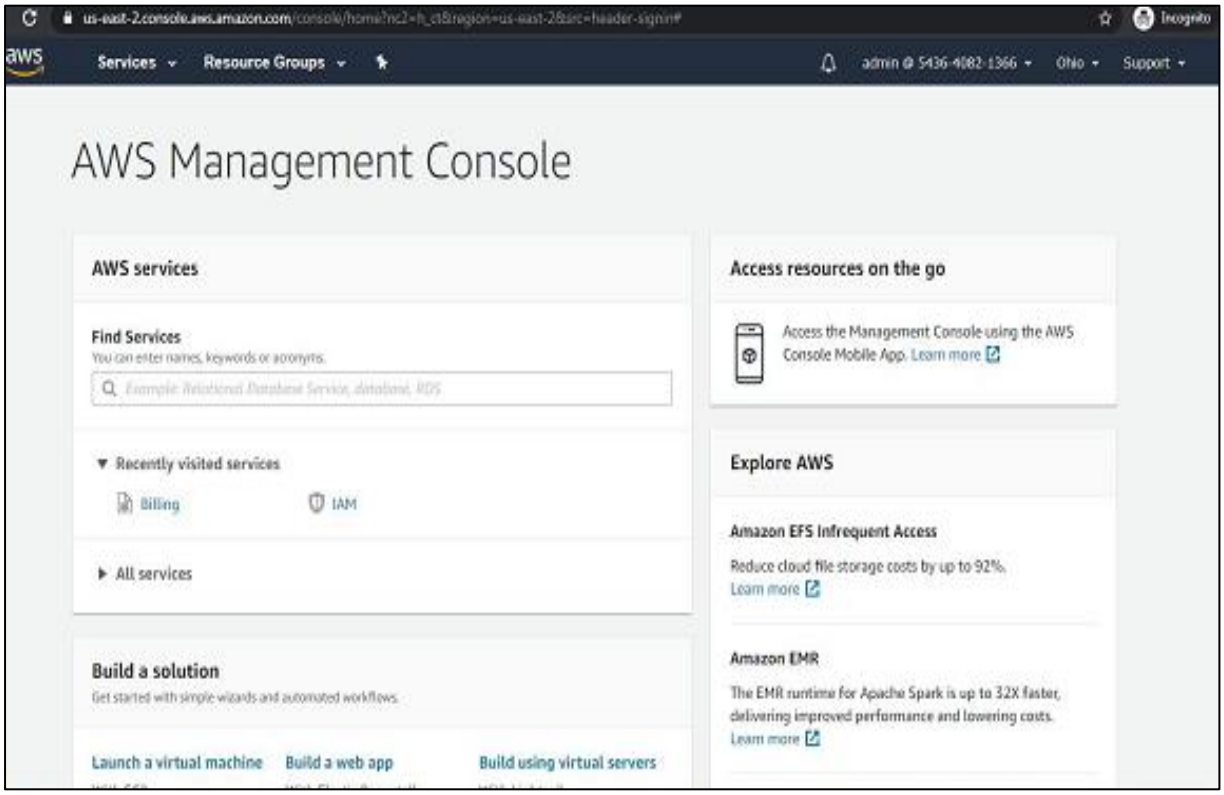
Step 13: Again Go to **My Account** -> **AWS Management console**
Click on **Next**, provide the *Account ID*, *username* and *password* and click on **Sign In**.



It will ask you to *change the password* which is been set by *administrator*.



Yow will R*edirect to Home Screen*



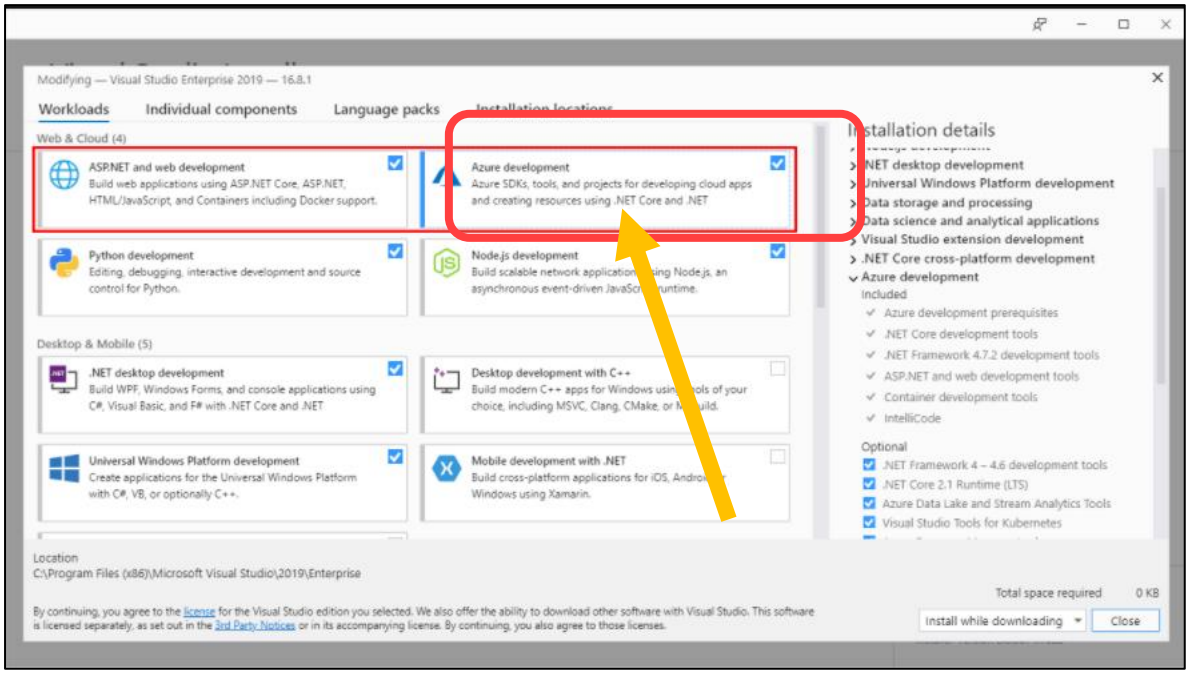
Practical No. 08

App Development using Cloud Computing

Aim: 1) To develop Application for windows Azure / Amazon AWS using Windows Azure Platform Training Kit and Visual Studio.

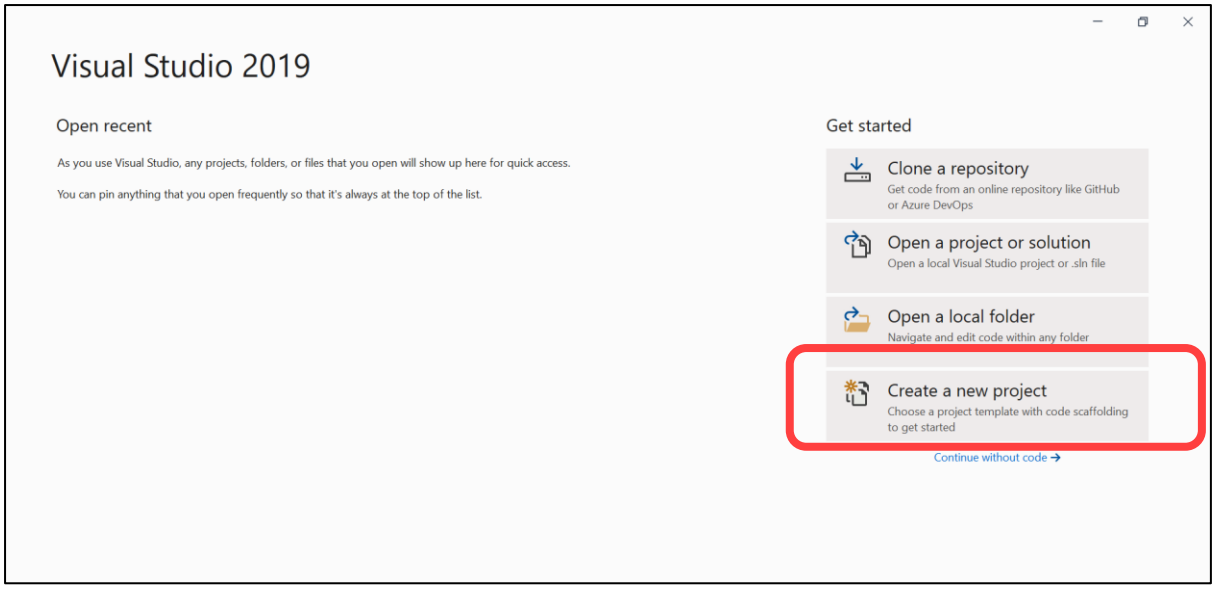
Implementation Steps:

Step 1: Download **Visual Studio Professional 2019** and Install.
While Installing **select Azure Development in Workload** as following and Install:

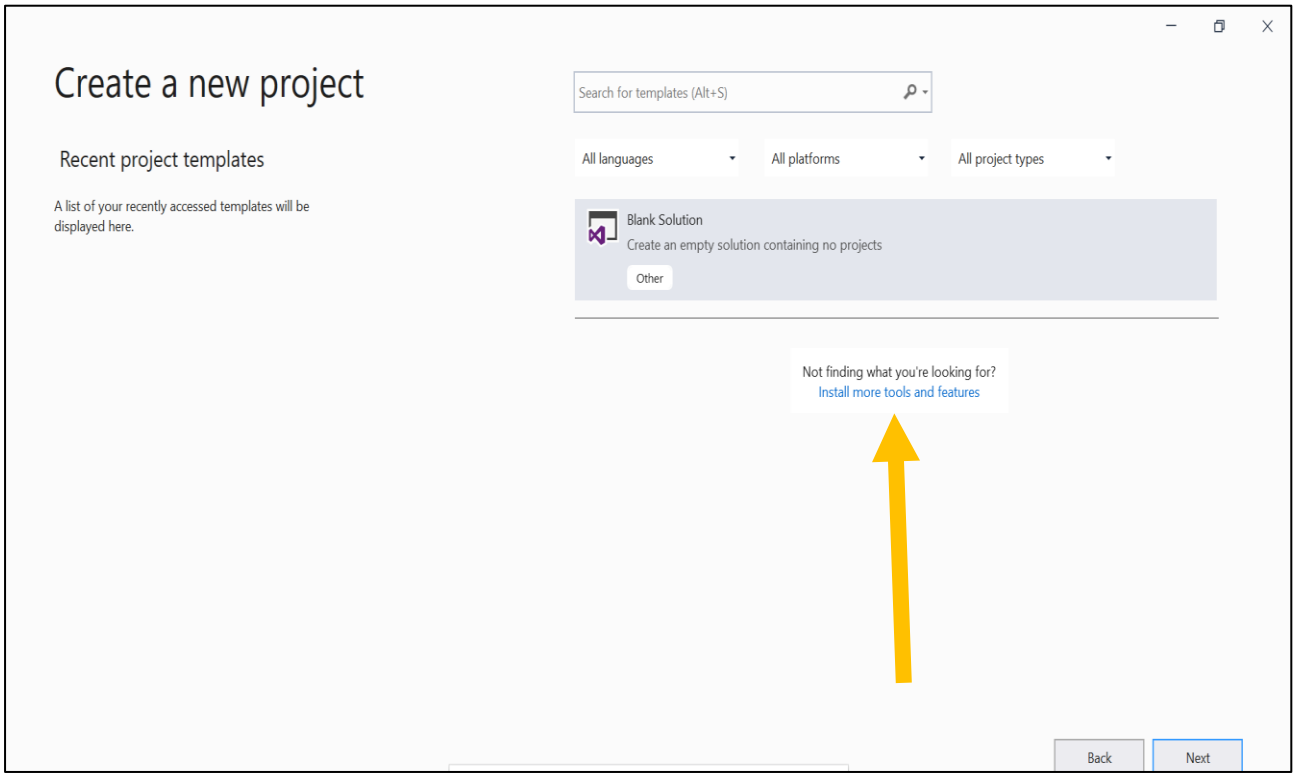


OR

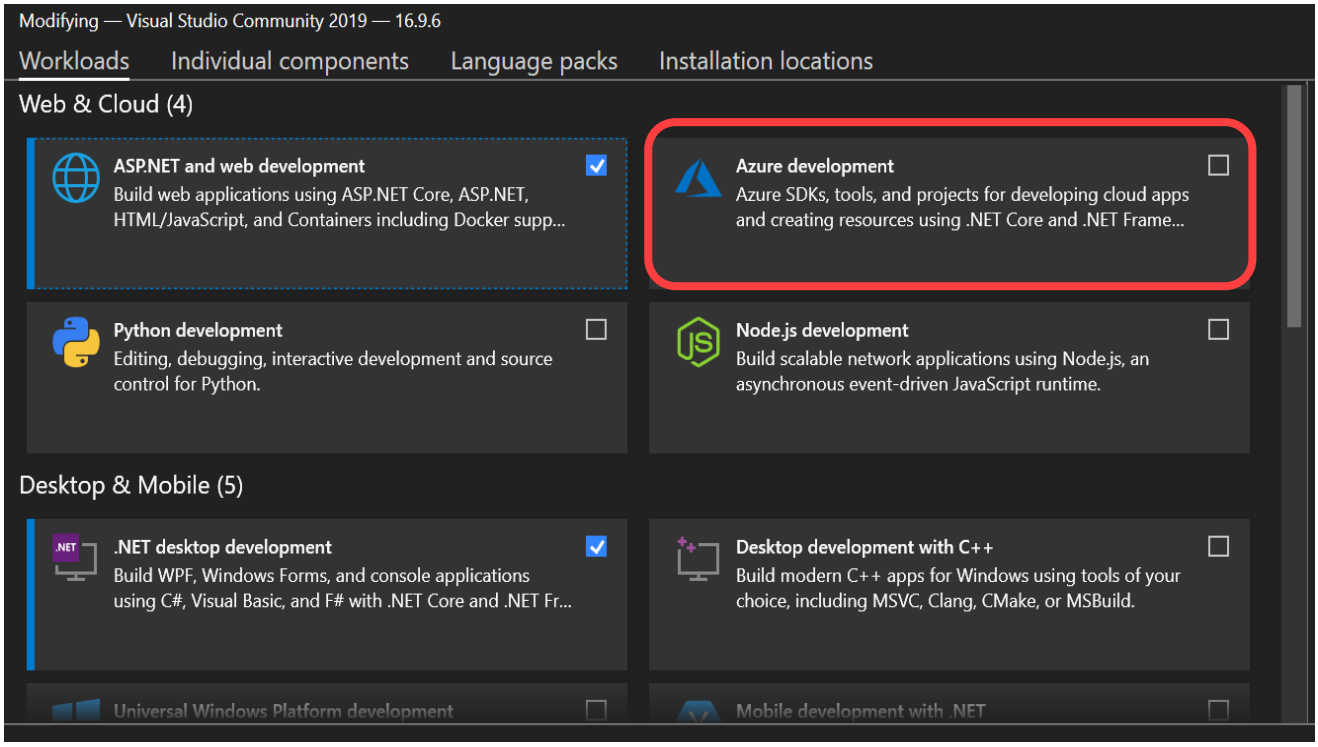
In case, **Visual Studio 2019** is **already installed**, open it and **select Create a new project**



Step 2: Here **click on <u>Install more tools and features</u>**

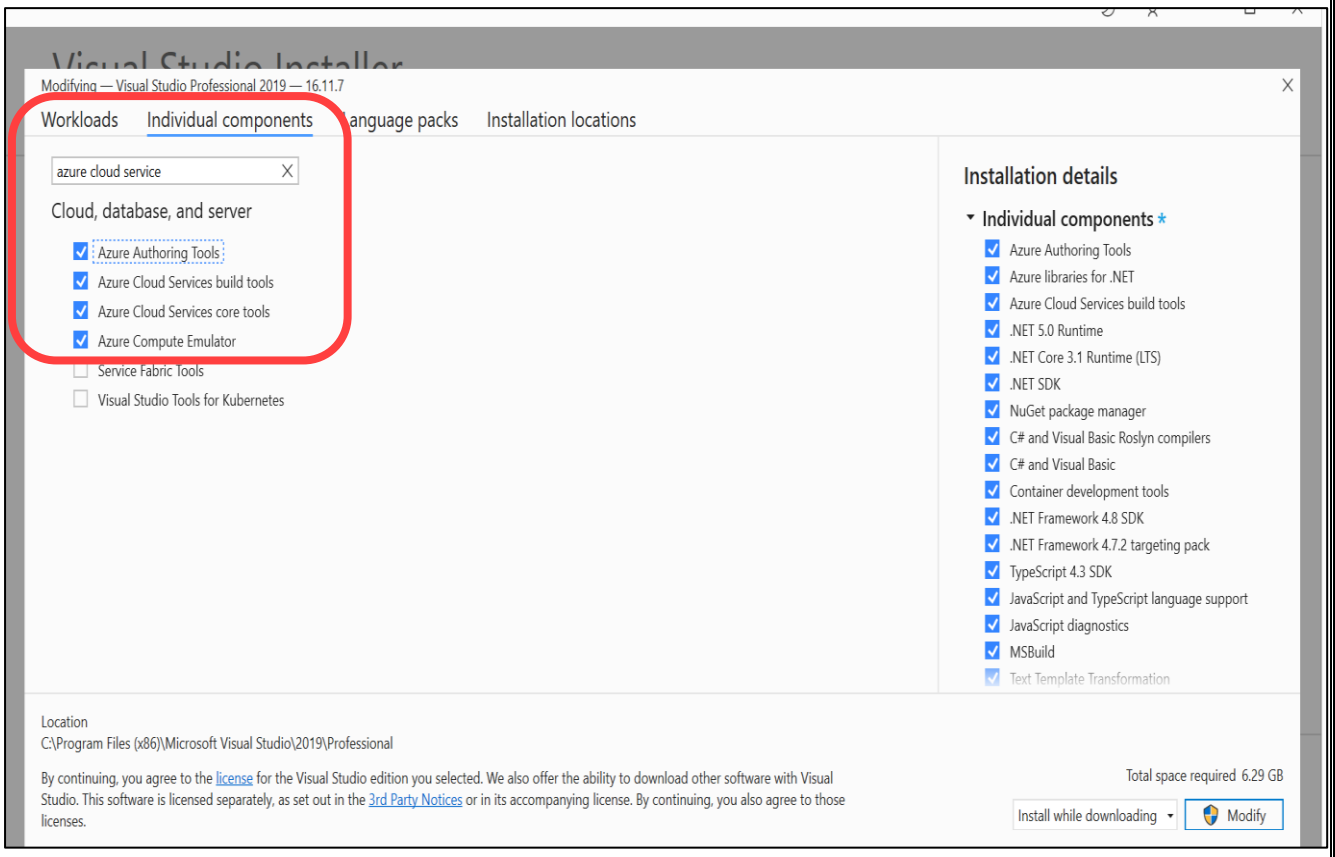


Now, **select <u>Azure Development in Workload</u>** and Click on Modify

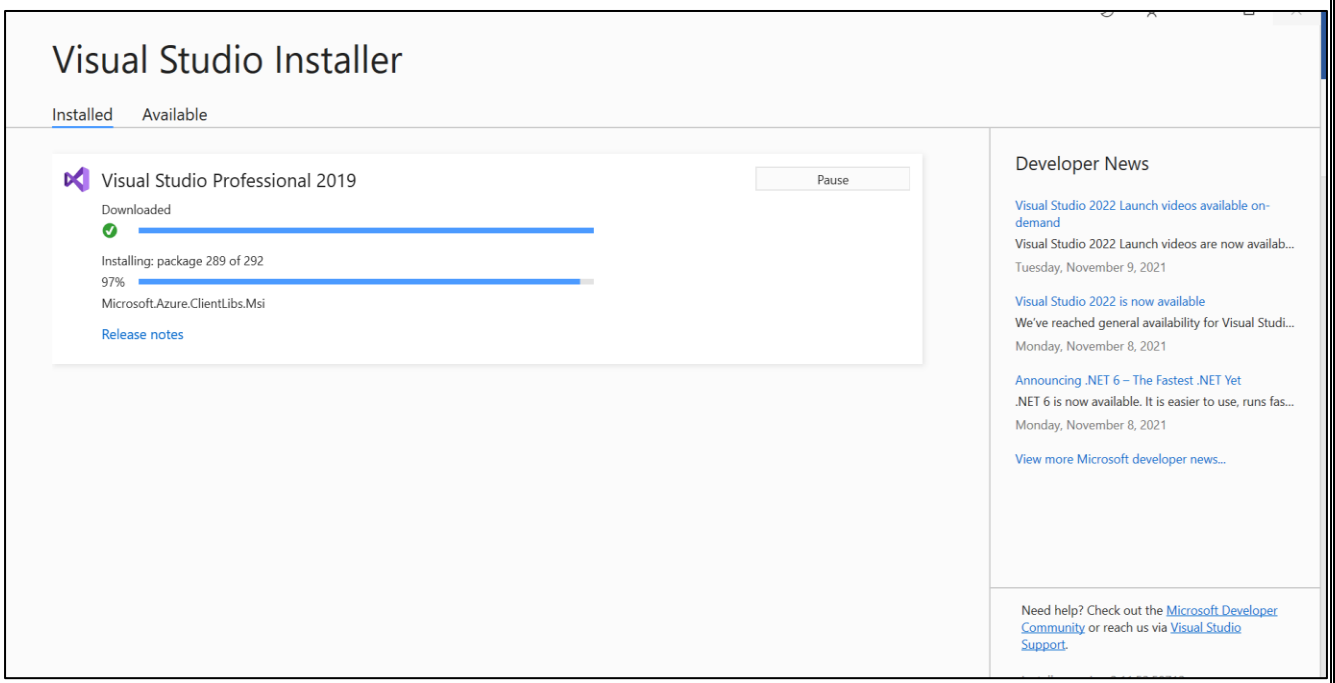


OR

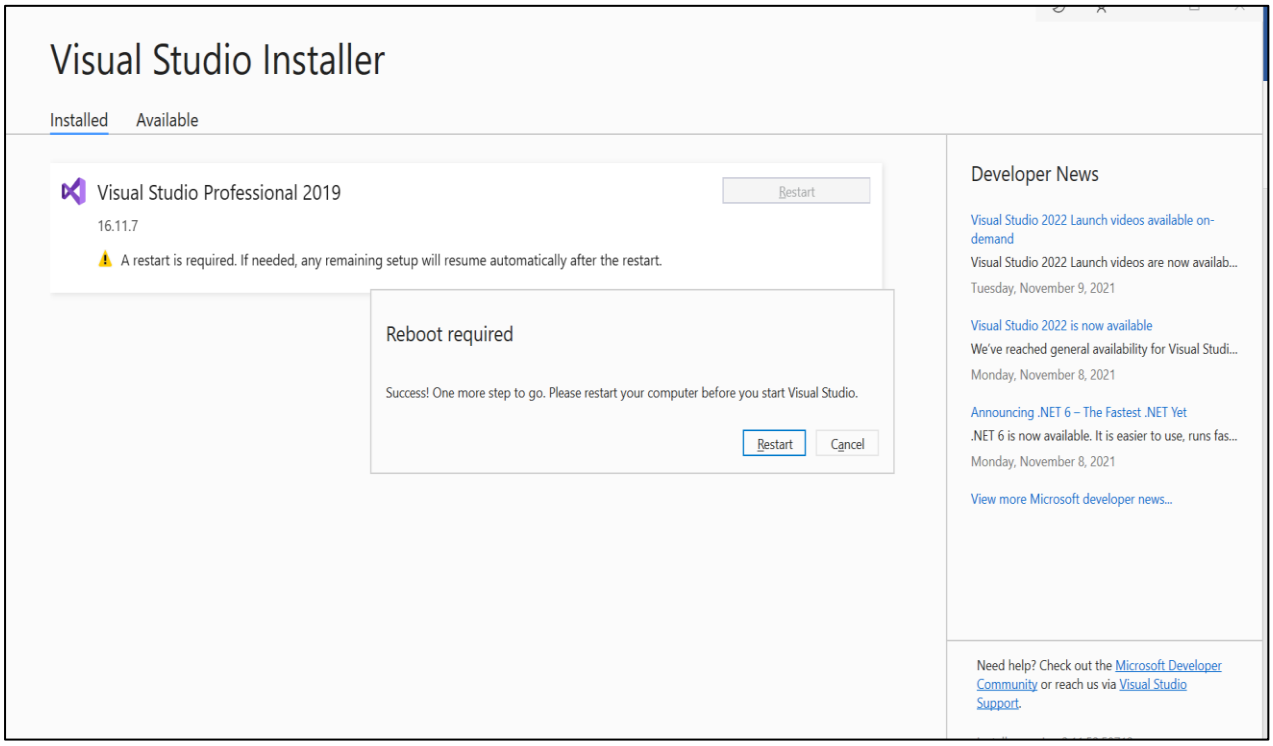
In **Individual components** select following and Click on Modify:



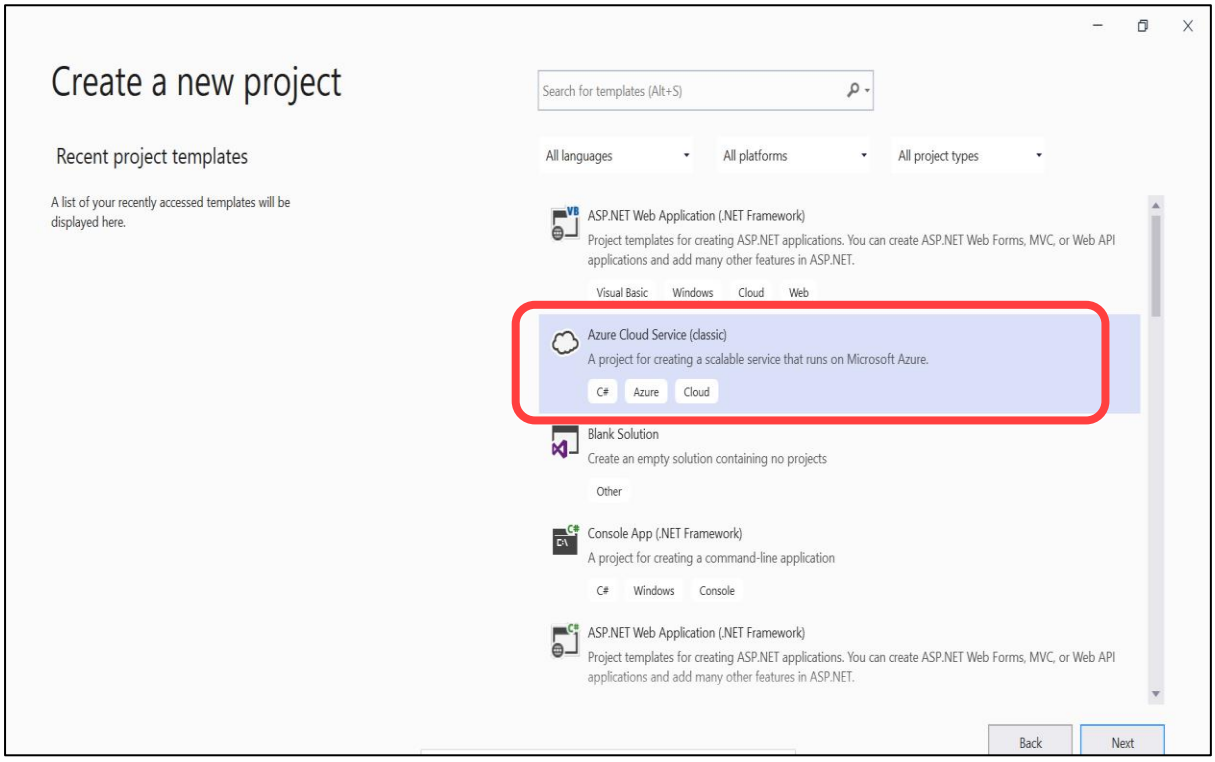
Now Installation will be done as follows:



Restart and Relaunch the System

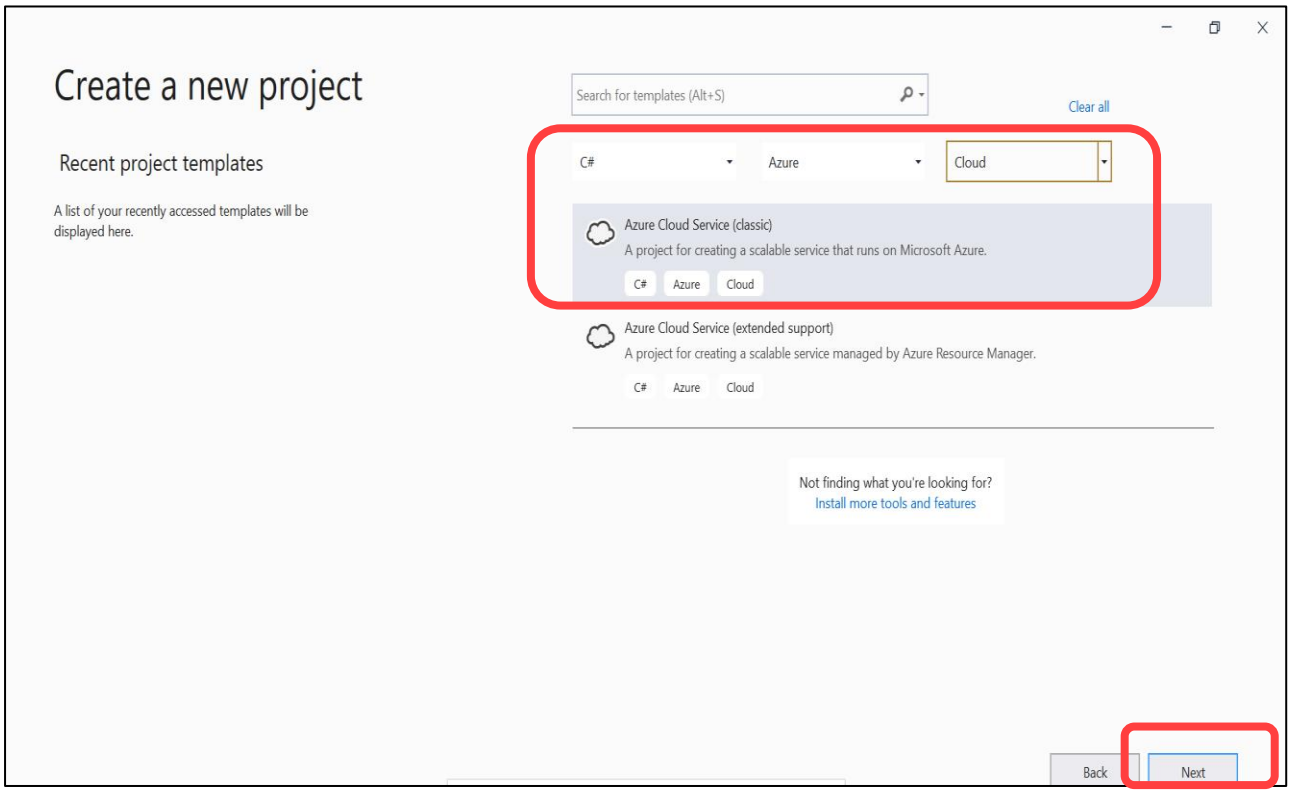


Step 3: Now **Restarting Visual Studio** after installing above steps, you will find **Azure Cloud Service (classic)** as below:

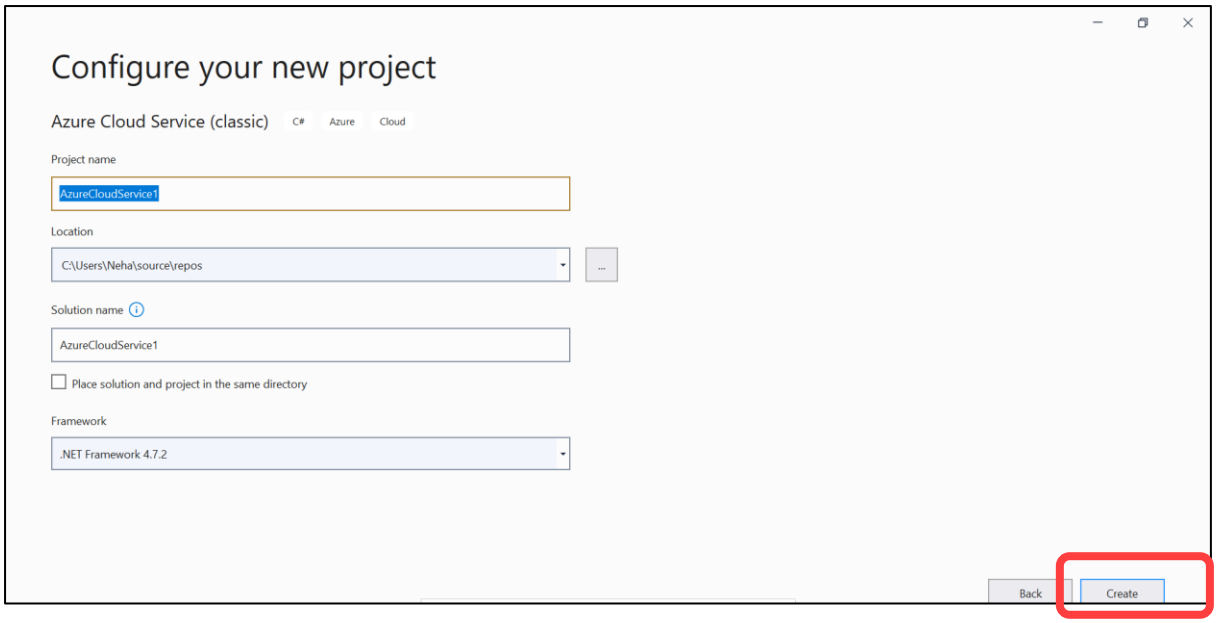


Else, apply Filter in drop down as: **C#, Azure, Cloud**

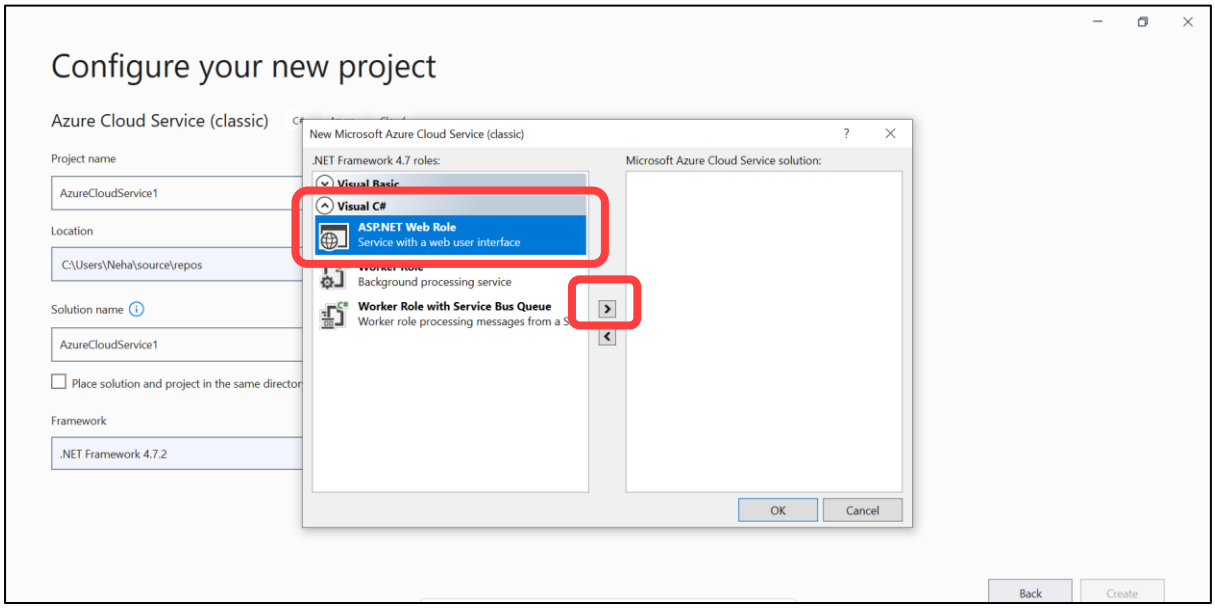
Select Azure Cloud Service (classic) and **click Next**



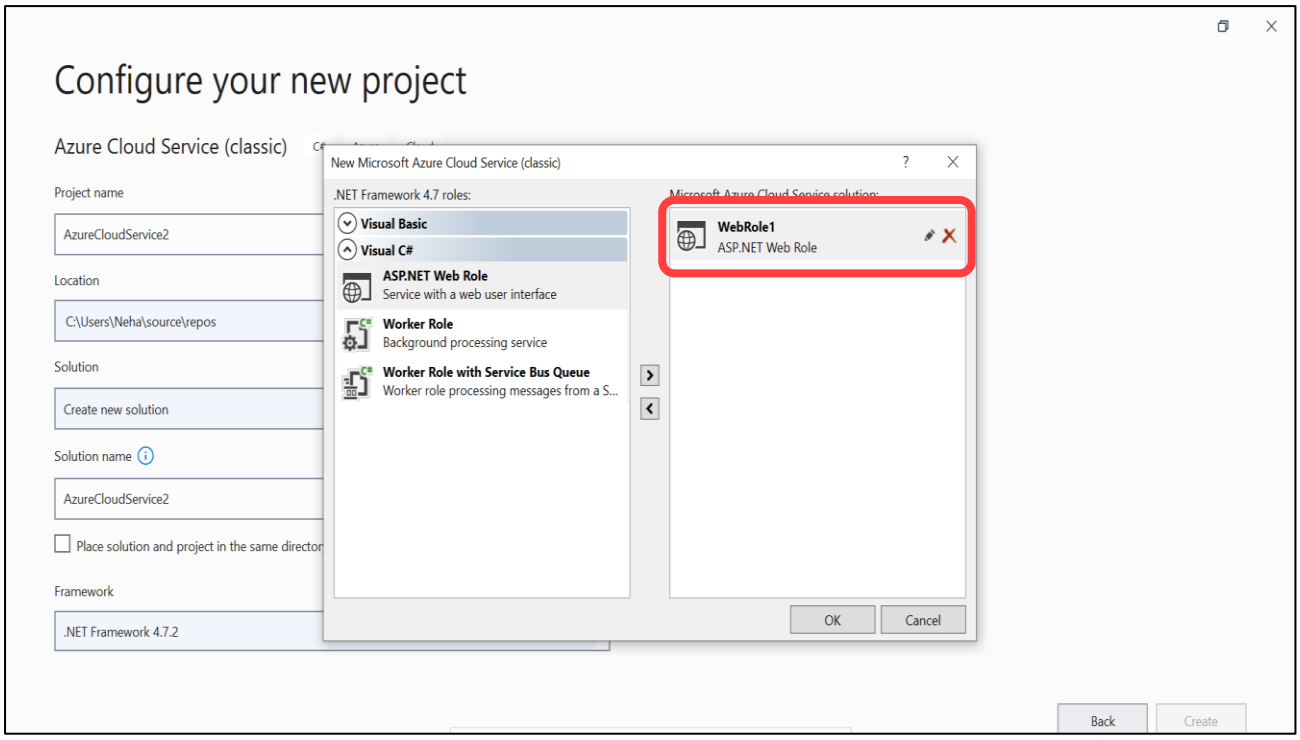
Give Project Name and **click Create**



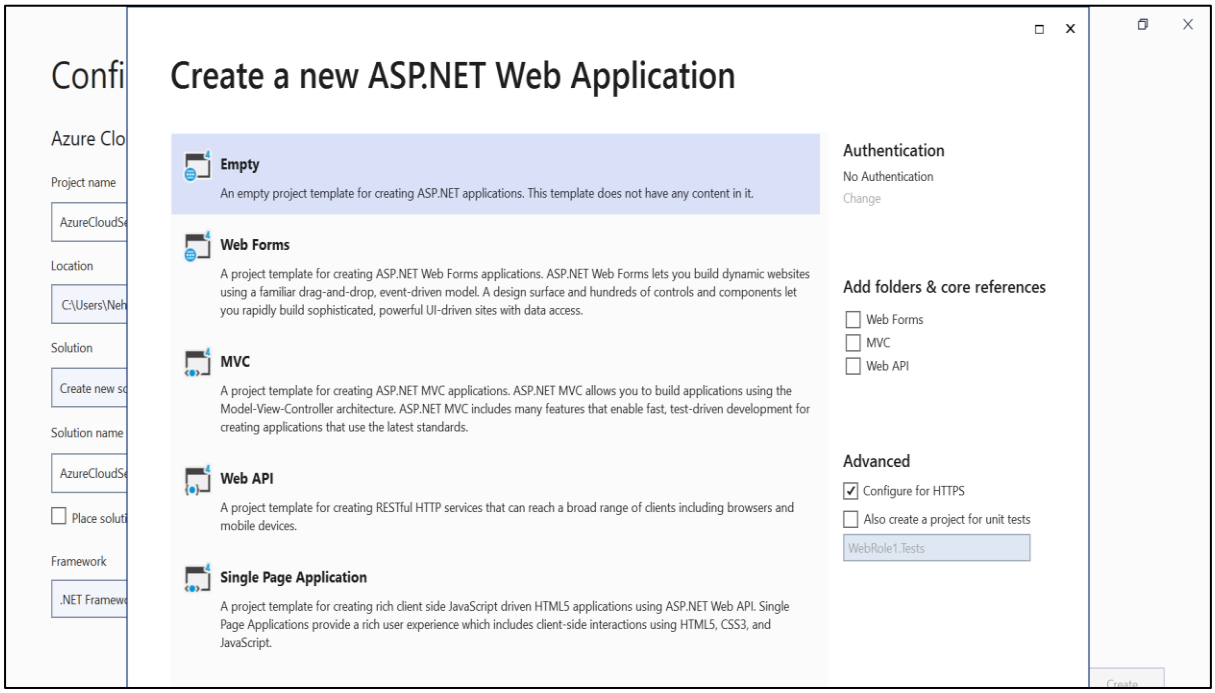
Step 4: Following will appear **Select ASP.NET Web Role** and **click > arrow**



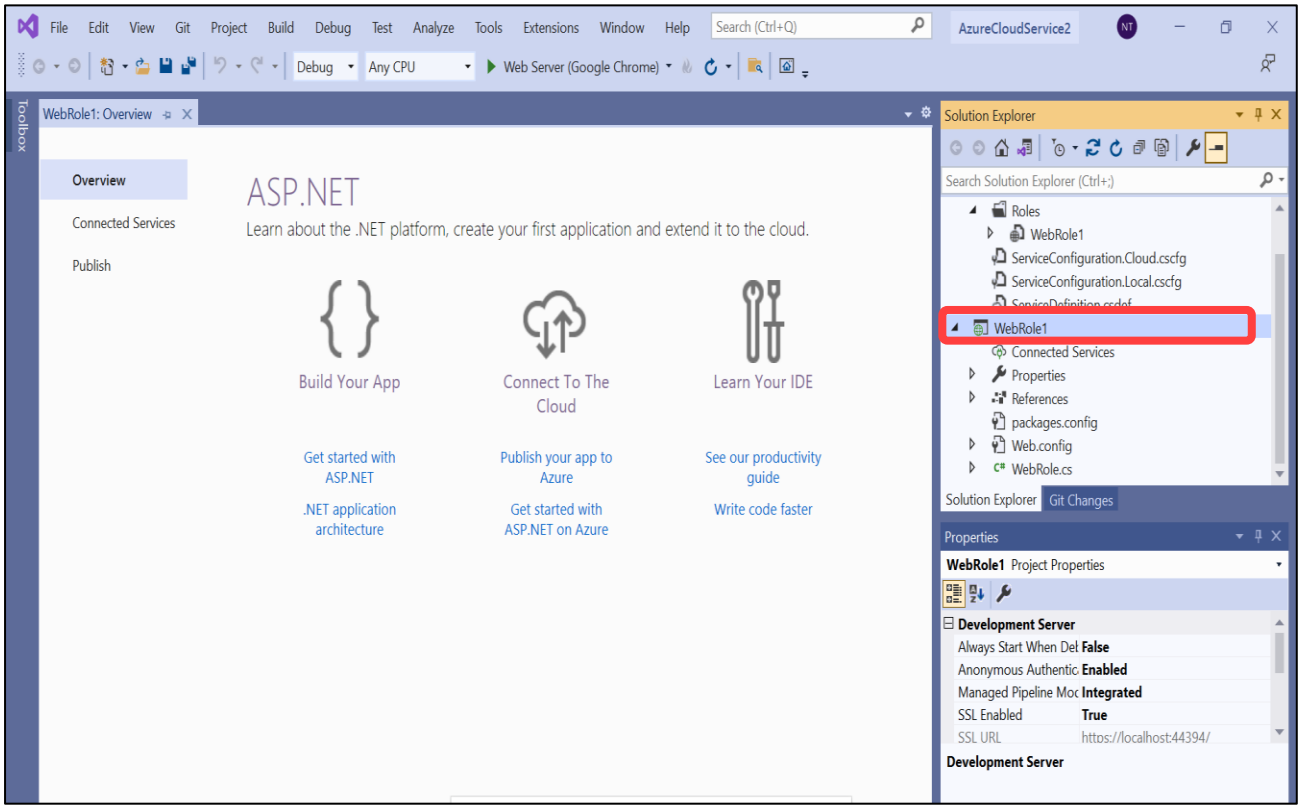
Following will be done and now **click OK**



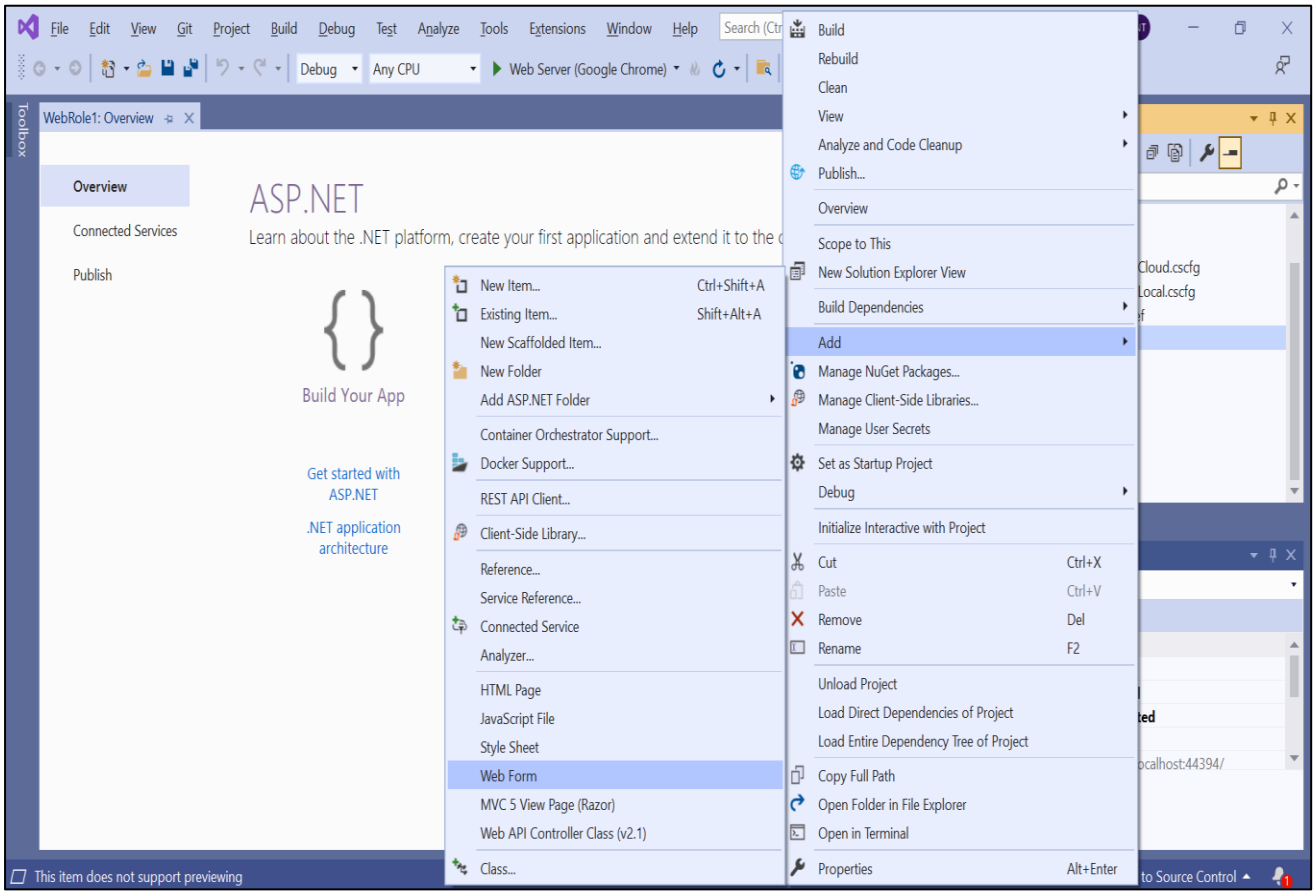
Following will appear **Select Empty** and **click Create**



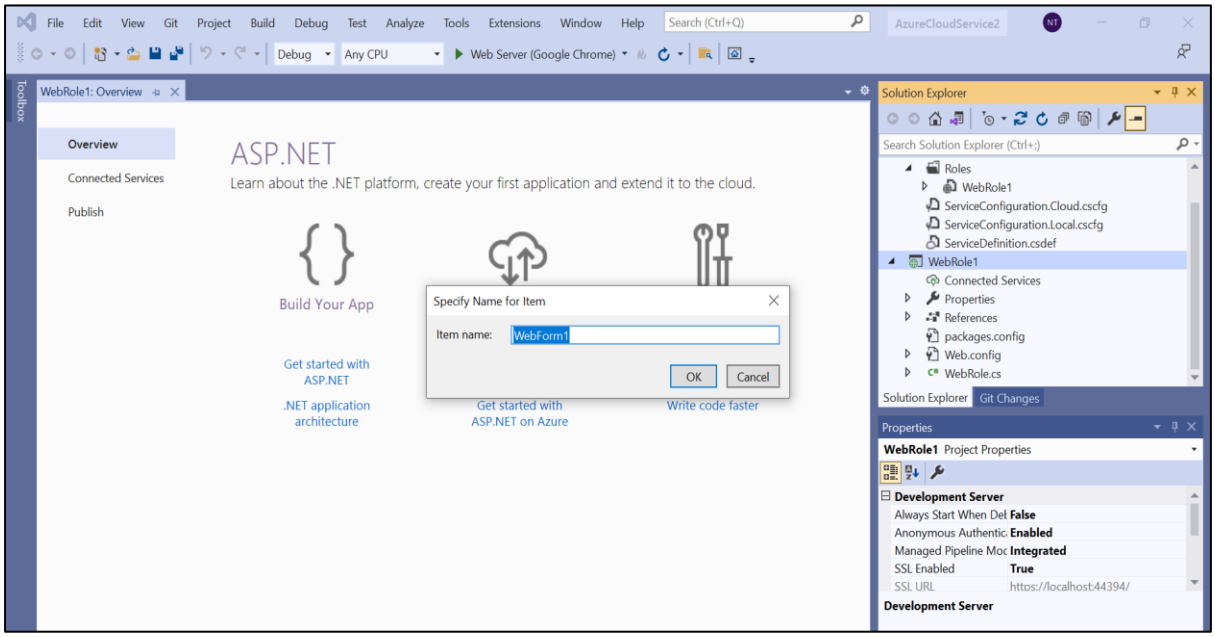
Step 5: Following will get created. Now **select WebRole1 in Solution Explorer**



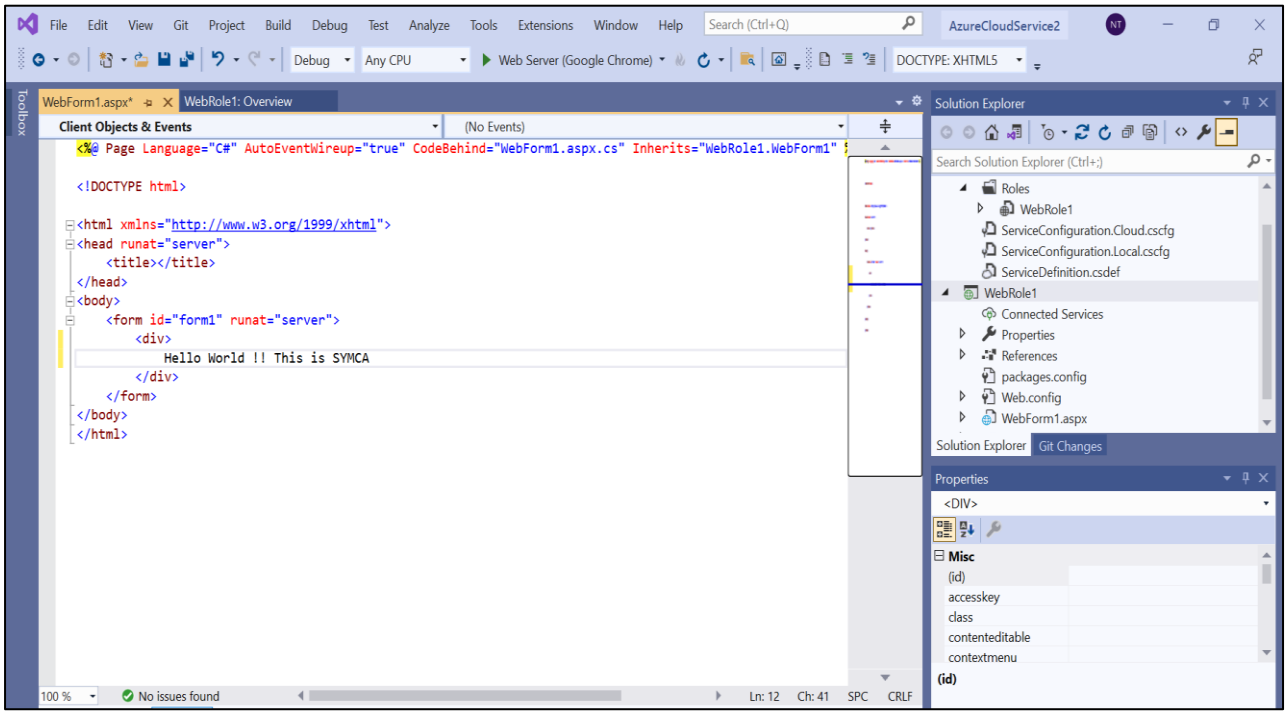
Right click on WebRole1, click on Add then, click on Web Form



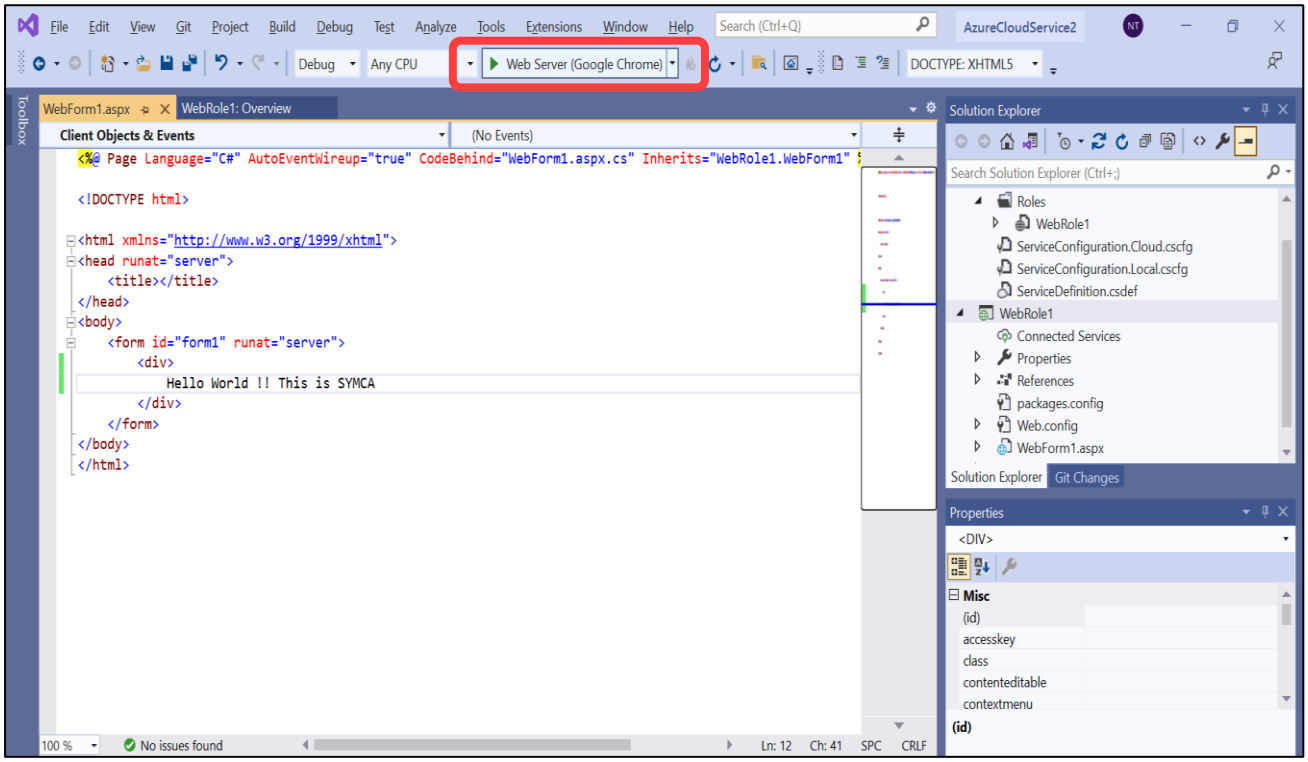
Give name to Form and **click OK**



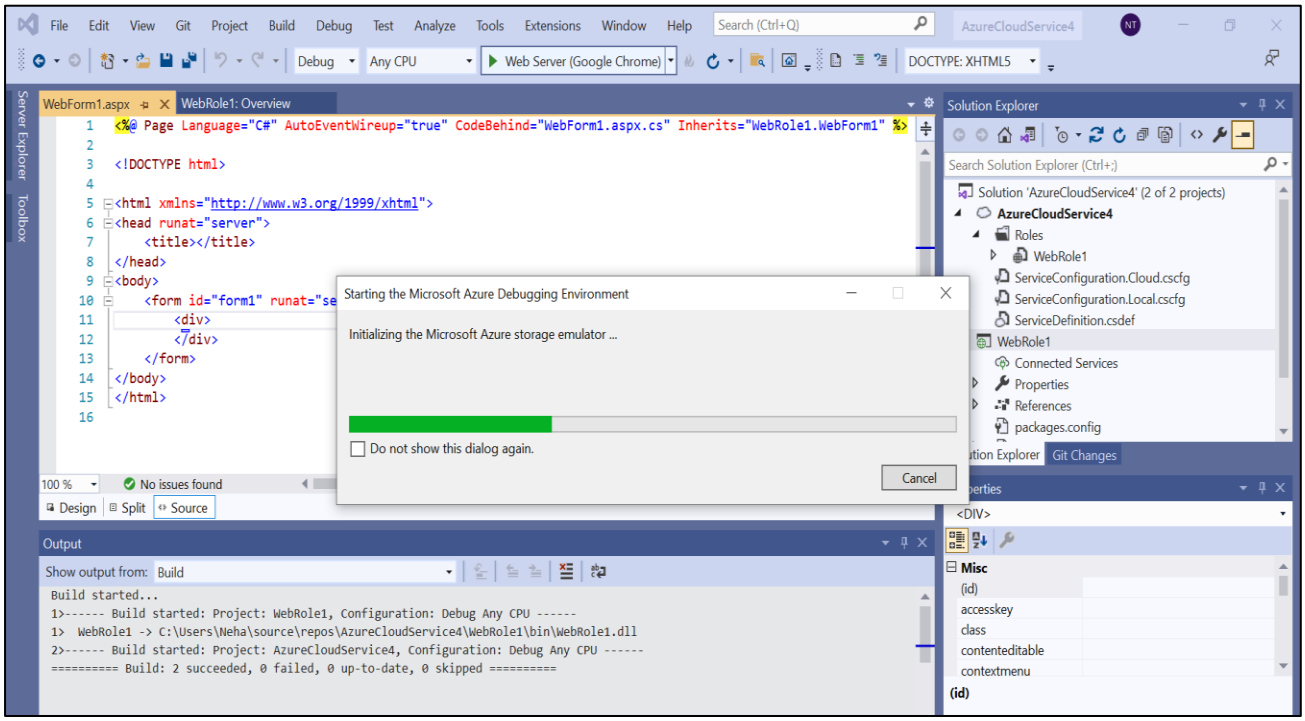
Step 6: Modify WebForm as required and Save.



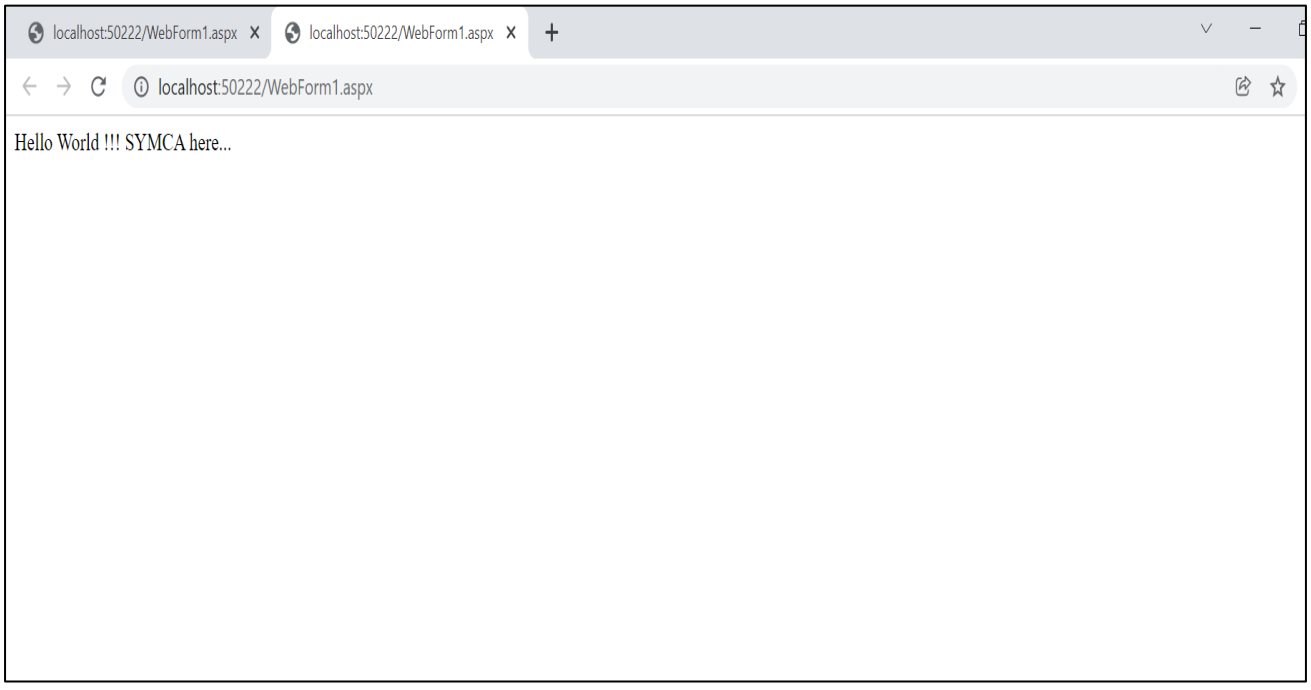
Step 7: Click on Web Server (Google Chrome) button to Debug and Execute the Form



Execution will take place as follows:



Following is the Output



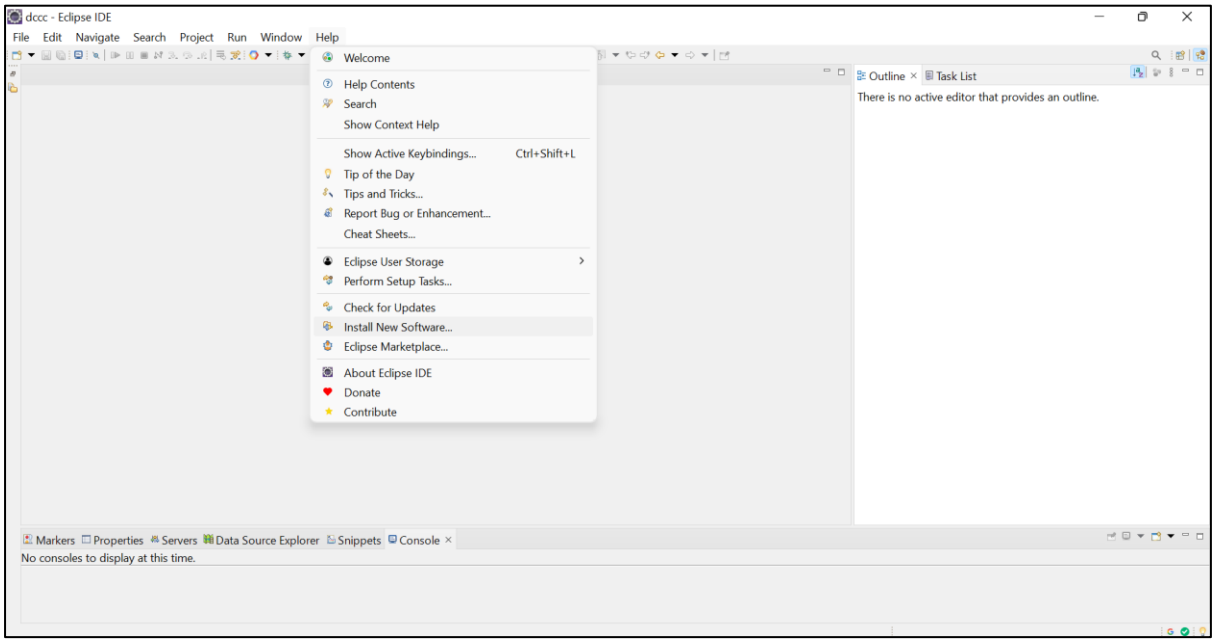
Practical No. 08

App Development using Cloud Computing

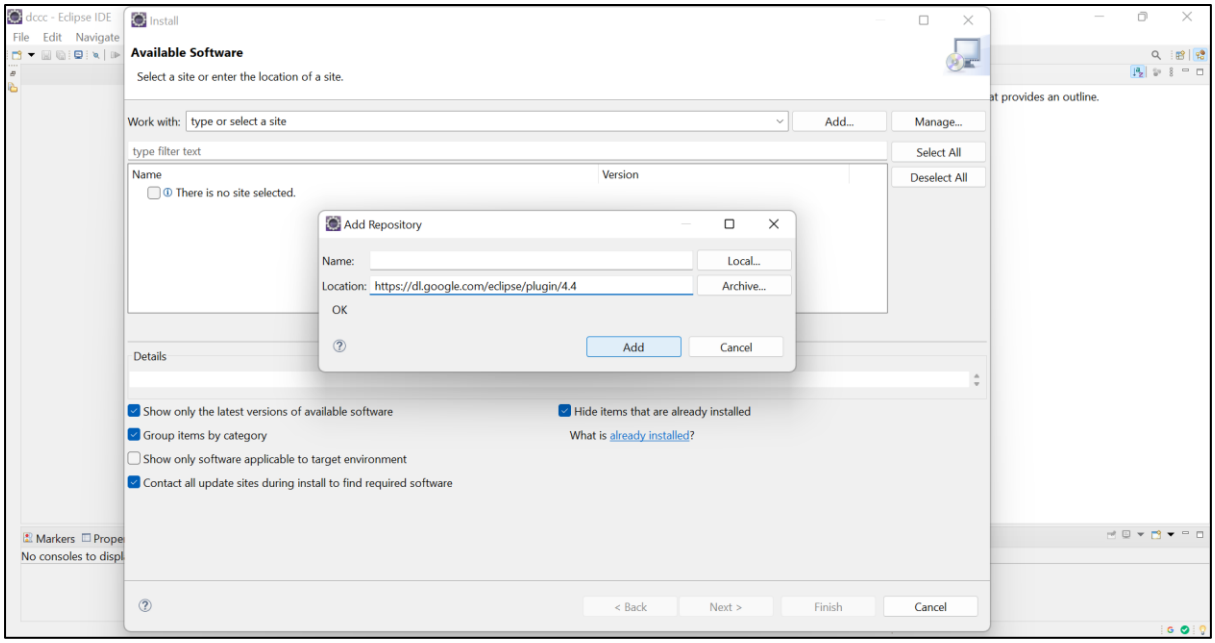
Aim: 2) To develop applications using Google App Engine by using Eclipse IDE

Implementation Steps:

Step 1: Open **Eclipse** and go to **Help Menu** -> **Install New Software**

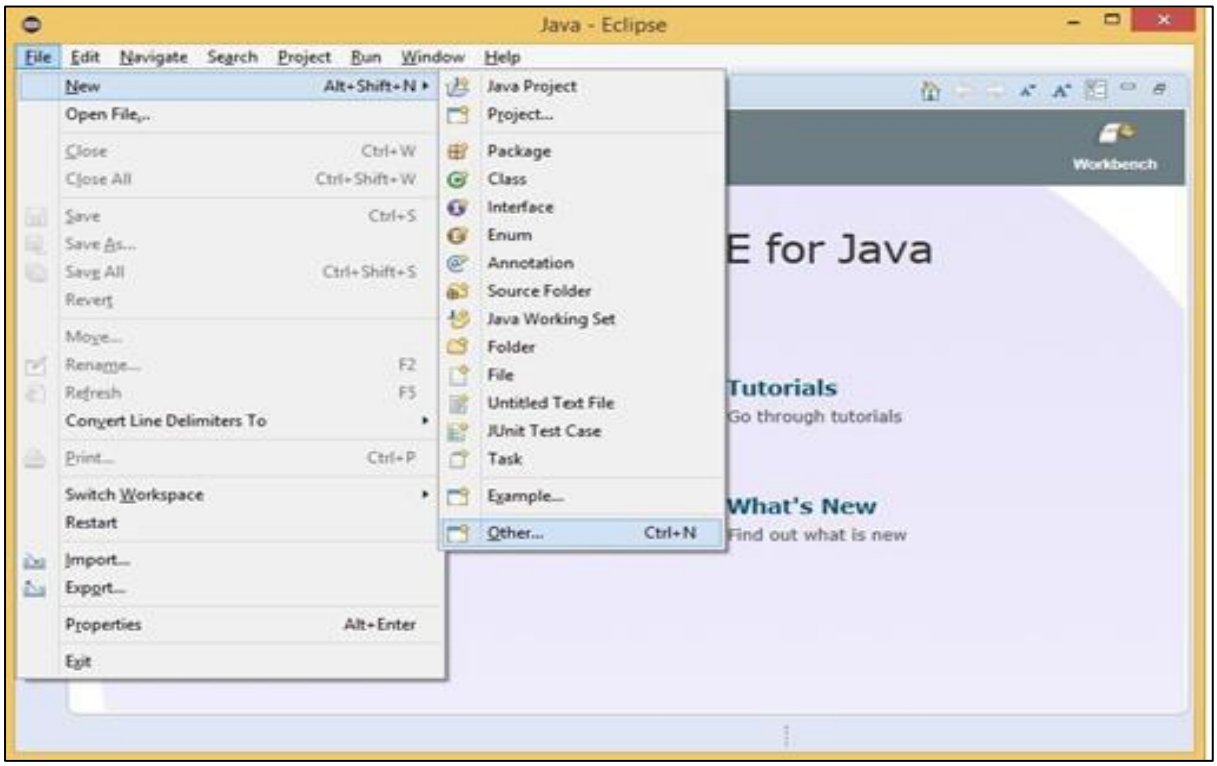


In *Install window* Click on the "**Add**" button besides the *Work with textbox*. **Add Repository** window appears. Enter the **Location** as "***https://dl.google.com/eclipse/plugin/4.4***" and click on "**OK**" button.

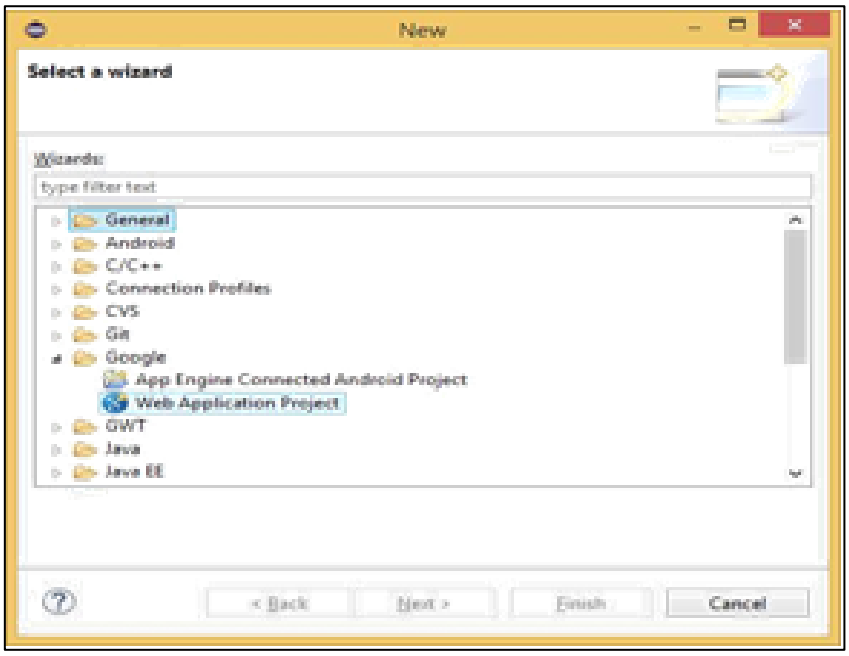


From the available software select the required software and tools for the GAE. Then click on the "**Next**" button. In the **Install Details** window click on "**Next**" button. In the Next Window "**Review the Items to be Installed**" then click on "**Next**" In the next window for **Review Licenses** select the option "**I accept……**" and click on "**Finish**" button. After Installation you will get option to "**Restart Eclipse**", click on **Yes**. So that the software you selected gets updated.

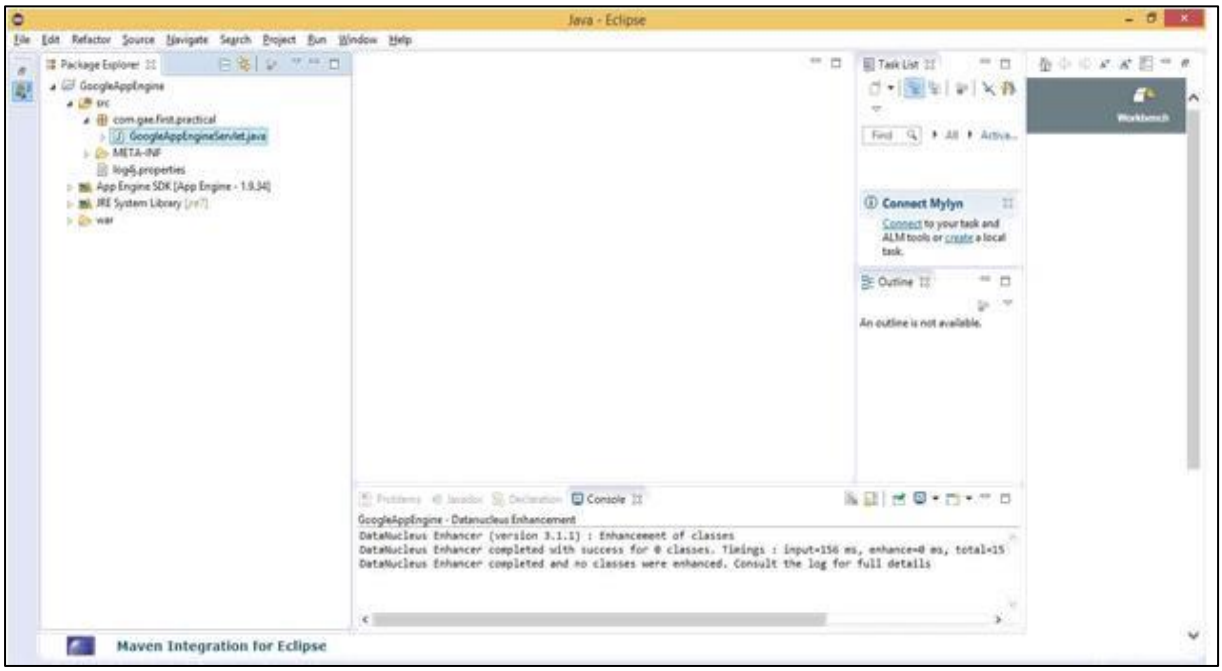
Step 2: Go to **File Menu** -> **New** -> **Other**



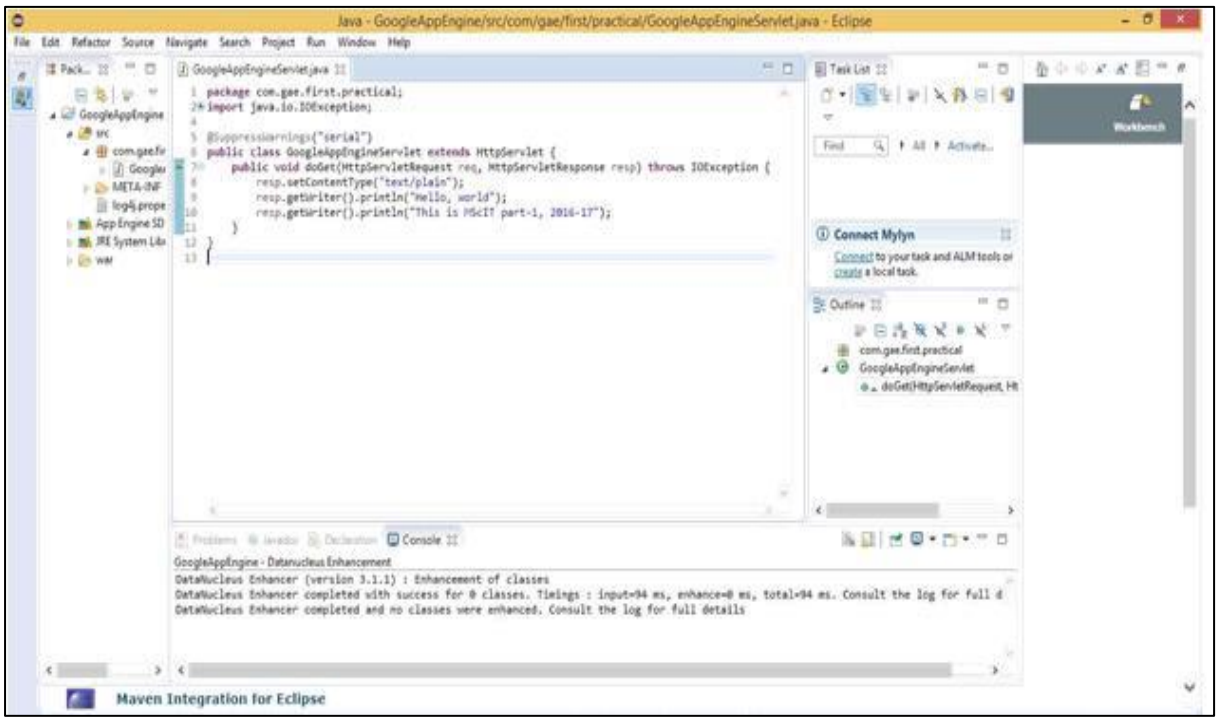
In the **New window** select **Google** -> **Web Application Project** and click on "**Next**" button.



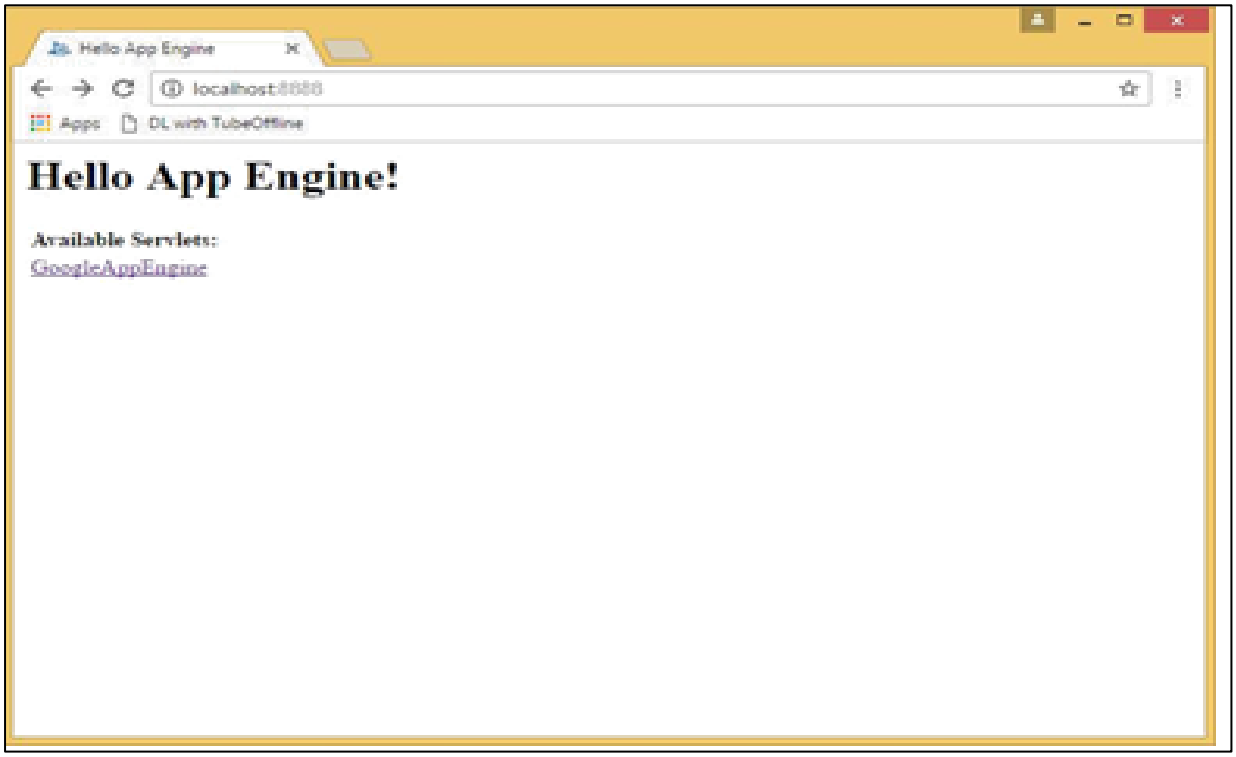
Enter the *details* for the *new Web application project*. *Deselect* the **Use Google Web Toolkit** option under the section **Google SDKs**. Click on the "**Finish**" button.
From the **Package Explorer** open the *.java* file (Here it is "**Google_App_EngineServlet.java**").



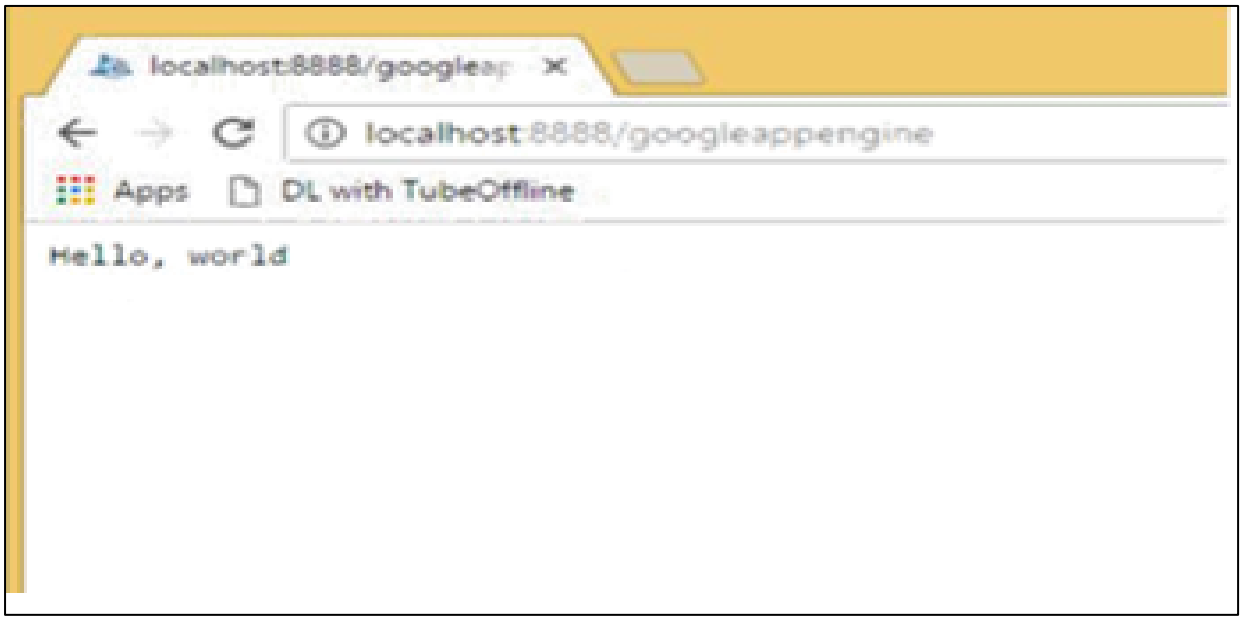
Step 3: *Edit* the file as required (Unedited file too can be used. Here the editing is done to "*what should be displayed*" on the browser). **Save** the file. Click on the **Run** option available on the *Tools bar*.



Step 4: In the *browser* (Here, Google Chrome) type the *address* as "**localhost:8888**" which is "*Default*".



In *localhost:8888* the link to the *Google_App_EngineServlet.java* file as **Google_App_Engine** is displayed. Click on this *link*. It will direct you to "**localhost:8888/Google_App_Engine**".



The **output text entered** in the **java** program is **displayed as the output** when clicked the link "**Google_App_Engine**".