

Lambda Expression

(functional interface ekak aniwa thiyenawa)

Question 1

```
package Question1;
import java.util.*;

@FunctionalInterface
interface iCalculator{
    void calculation(int a, int b);
}

public class Q1A{
    public static void main(String[] args) {

        iCalculator iCalculator = new iCalculator() {

            @Override
            public void calculation(int a, int b) {

                System.out.println("sum is " +(a+b));
            }
        };
        iCalculator.calculation(4, 6);
    }
}
```

Question 1 Answer

```
package Quesion1lambda;
import java.util.*;

@FunctionalInterface
interface iCalculator{

    void calculation(int a, int b);
}

public class Q1Alambda{

    public static void main(String[] args) {

        iCalculator iCalculator = (a,b) -> System.out.println("sum is "
+(a+b));

        iCalculator.calculation(4, 6);

    }
}
```

2018 paper Question 1

This question is related to the Lambda Expressions in Java section.

a) Transform following conventional Java program into Java Lambda Expressions format.

```
interface IGradeService{

    public String checkGrade(List<Integer> listOfMarks);

}

public class StudentsGrade implements IGradeService{

    @Override
    public String checkGrade(List<Integer> listOfMarks) {

        int total = 0;
        for (Integer mark : listOfMarks) {
            total = total + mark;
        }
        double average = total/listOfMarks.size();
        if(average >= 80.0){
            return "Best";
        }else if((average < 80.0) && (average >= 60.0)){
            return "Merit";
        }else if((average < 60.0) && (average >= 45.0)){
            return "Pass";
        }else{
            return "Fail";
        }
    }

    public static void main(String[] args) {

        StudentsGrade grade = new StudentsGrade();
        ArrayList<Integer> listOfMarks = new ArrayList<Integer>();
        listOfMarks.add(85);
        listOfMarks.add(75);
        listOfMarks.add(60);
        listOfMarks.add(80);
        listOfMarks.add(100);

        String result = grade.checkGrade(listOfMarks);
        System.out.println("Result is = " + result);
    }
}
```

2018 Question 1 Answer

```
package Quesion1;
import java.util.*;

@FunctionalInterface
interface IGradeService{
```

```

        public String checkGrade(List<Integer> listOfMarks);
    }

    public class StudentsGrade{

        public static void main(String[] args) {

            IGradeService gradeService = (listOfMarks) -> {

                int total = 0;
                for (Integer mark : listOfMarks) {
                    total = total + mark;
                }
                double average = total/listOfMarks.size();
                if(average >= 80.0){
                    return "Best";
                }else if((average < 80.0) && (average >= 60.0)){
                    return "Merit";
                }else if((average < 60.0) && (average >= 45.0)){
                    return "Pass";
                }else{
                    return "Fail";
                }
            };

            ArrayList<Integer> listOfMarks = new ArrayList<Integer>();
            listOfMarks.add(85);
            listOfMarks.add(75);
            listOfMarks.add(60);
            listOfMarks.add(80);
            listOfMarks.add(100);

            System.out.println(gradeService.checkGrade(listOfMarks));

        }
    }

```

2019 paper Question 2

This question is related to the **Lambda Expressions** in Java section.

- a) Transform following conventional Java program in to Java Lambda Expressions format.

```
interface ITrafficService{

    public String checkSpeed(List<Double> listOfCheckPoints);
}

public class VehicleMonitor implements ITrafficService{

    @Override
    public String checkSpeed(List<Double> listOfCheckPoints) {

        double total = 0;
        for (Double speed : listOfCheckPoints) {
            total = total + speed;
        }
        double averageSpeed = total/listOfCheckPoints.size();

        if(averageSpeed >= 100.0){
            return "Issue Fine";
        }else if((averageSpeed < 100.0) && (averageSpeed >= 80.0)){
            return "Warning message";
        }else if((averageSpeed < 80.0) && (averageSpeed >= 50.0)){
            return "Good speed";
        }else if((averageSpeed < 50.0) && (averageSpeed >= 30.0)){
            return "Normal";

        }else{
            return "Slow";
        }
    }

    public static void main(String[] args) {

        VehicleMonitor vehicleMonitor = new VehicleMonitor();
        ArrayList<Double> speedInCheckPoint = new ArrayList<>();
        speedInCheckPoint.add(20.0);
        speedInCheckPoint.add(30.0);
        speedInCheckPoint.add(60.0);
        speedInCheckPoint.add(80.0);
        speedInCheckPoint.add(100.0);
        speedInCheckPoint.add(120.0);

        String result = vehicleMonitor.checkSpeed(speedInCheckPoint);
        System.out.println("Vehicle average status is in = " + result);
    }
}
```

2019 paper Question 2 Answer

```
package lamda.lab;

import java.util.ArrayList;

import java.util.List;

interface ITrafficService{

    public String checkSpeed(List<Double> listOfCheckPoints);

}

public class Paper19ans {

    public static void main(String[] args) {

        ITrafficService iTrafficService = (listOfCheckPoints)-> {

            double total = 0;

            for (Double speed :listOfCheckPoints) {

                total = total + speed;

            }

            double averageSpeed = total/listOfCheckPoints.size();

            if(averageSpeed >= 100.0){

                return "Issue Fine";

            }else if((averageSpeed < 100.0) && (averageSpeed >= 80.0)){

                return "Warning message";

            }else if((averageSpeed < 80.0) && (averageSpeed >= 50.0)){

                return "Good speed";

            }else if((averageSpeed < 50.0) && (averageSpeed >= 30.0)){

                return "Normal";

            }

            }else{

                return "Slow";

            }

        };

        ArrayList<Double> speedinCheckPoint = new ArrayList<Double>();

        speedinCheckPoint.add(20.0);

        speedinCheckPoint.add(30.0);

    }

}
```

```
speedinCheckPoint.add(60.0);
```

```
speedinCheckPoint.add(80.0);
```

```
speedinCheckPoint.add(100.0);
```

```
speedinCheckPoint.add(120.0);
```

```
System.out.println(iTrafficService.checkSpeed(speedinCheckPoint));
```

```
}
```

```
}
```

Lambda Expression Thread (Runnable aniwa thiyenawa)

2022 Question 1

This question is related to the (Lambda Expressions in java) Functional Programming. Refer to the sample source code and given outputs of the program. a) Transform following conventional Java program into Java Lambda Expressions format. This program uses two threads as **Producer Thread** and the **Consumer Thread**. The source code of only the **Producer Thread** is given and you should implement the **Consumer Thread** and **Producer Thread** both in **Lambda expressions**. Producer thread **adds** elements for the list and consumer thread **remove** elements from the list. At the end of both threads' execution, it should ensure the list is empty. It should give the same output as mentioned in the console. (20 marks) Hint: - You should implement both producer and consumer threads in one class using Lambda Expressions.

ProducerThread Implementation

```
public class ProducerThread implements Runnable {

    ArrayList<Integer> list;

    public ProducerThread(ArrayList<Integer> list) {
        this.list = list;
    }

    @Override
    public void run() {
        synchronized (list) {
            int value = 0;
            while (true) {
                System.out.println("Producer started");
                try {
                    value += 10;
                    list.add(value);
                    System.out.println("Producer adding value
                    = " + value + " to Queue");

                    list.wait();
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }

                list.notify();
                System.out.println("Elements in Queue = " + list);
            }
        }
    }
}
```

Console Output with ThreadTest class

```
ConsumerThread.java  ProducerThread.java  ThreadTest.java
1 package conventional;
2 import java.util.ArrayList;
3
4 public class ThreadTest {
5
6     public static void main(String[] args) {
7         ArrayList<Integer> list = new ArrayList<>();
8         Thread producer = new Thread(new ProducerThread(list));
9         Thread consumer = new Thread(new ConsumerThread(list));
10        producer.start();
11        consumer.start();
12    }
13 }
14
```

```
Console  Problems  @ Javadoc
<terminated> ThreadTest (1) [Java Application]
Producer started
Producer adding value = 10 to list
Consumer started
Consumer thread consumes 10
Elements in List = []
Producer started
Producer adding value = 20 to list
Consumer started
Consumer thread consumes 20
Elements in List = []
Producer started
Producer adding value = 30 to list
Consumer started
Consumer thread consumes 30
Elements in List = []
Producer started
Producer adding value = 40 to list
```

2022 Question 1 Answer

```
package Question1;
import java.util.*;

public class ProducerThread{

    public static void main(String[] args){

        ArrayList<Integer>list = new ArrayList<>();

        Thread p = new Thread()->{
            synchronized (list) {
                int value = 0;
                while (true){
                    System.out.println("Producer started");
                    try {
                        value += 10;
                        list.add(value);
                        System.out.println("Producer adding value = " +
value + " to Queue");

                        list.wait();
                        Thread.sleep(1000);
                    }
                    catch (InterruptedException e) {
                        System.out.println("");
                    }
                    list.notify();
                    System.out.println("Elements in Queue = " + list);
                }
            }
        };

        Thread c = new Thread()->{
            synchronized (list) {

                while (true){
                    System.out.println("Consumer started");
                    try {
                        System.out.println("Consumer thread consume " +
list.get(0));

                        list.remove(0);
                        list.notify();
                        Thread.sleep(1000);
                        list.wait();
                    }
                    catch (InterruptedException e) {
                        System.out.println("");
                    }
                }
            }
        };

        p.start();
        c.start();
    }
}
```



```
}  
}
```

2018 Question 2

- b) Convert the following Lambda code to conventional java program (Your program should display the same output as below) (**Hint:- override the run() method in Runnable interface**)

```
public class Q1b {  
  
    public static void main(String[] args) {  
        Runnable r2 = () -> {  
            IntStream.iterate(0, x -> x + 1).limit(10).forEach(x -> {  
  
                IntStream.iterate(0, y -> y + 1).limit(10).forEach(y -> {  
                    System.out.print(y);  
                });  
                System.out.println();  
            });  
        };  
        new Thread(r2).start();  
    }  
}
```

Output

```
Console 18 * 100% x  
<terminated> Q1b [Java App  
0123456789  
0123456789  
0123456789  
0123456789  
0123456789  
0123456789  
0123456789  
0123456789  
0123456789  
0123456789
```

```
public class Paper18ans2 {  
  
    public static void main(String[] args) {  
        Runnable r2 = new Runnable() {  
  
            public void run () {  
  
                for (int x=0 ; x < 10 ; x++ ) {  
                    System.out.println();  
                    for (int y=0 ; y < 10 ; y++ ) {  
                        System.out.print(y);  
                    }  
                }  
            }  
        }  
    }  
}
```

```

    };

    new Thread(r2).start();

}
}

```

2019 Question 2

- b) Convert the following java program into Lambda Expression (Your program should display the same output as below) (Hint: - use override the run() method in Runnable interface)

```

public class Q1b implements Runnable{

    public static void main(String[] args) {
        Thread thread = new Thread(new Q1b());
        thread.start();
    }
    @Override
    public void run(){
        for (int row = 1; row <= 5; row++) {
            for (int column = 1; column <= 5; column++) {
                System.out.print(column);
            }
            System.out.println();
        }
    }
}

```

Output

Problems Console

```

<terminated> Q1bAns [Java App]
12345
12345
12345
12345
12345

```

2019 Question 2 Answer

```

package lamda.lab;

import java.util.stream.IntStream;

public class Paper19ans2 {
    public static void main(String[] args) {
        Runnable r1 = () -> {

            IntStream.iterate(1, row -> row + 1).limit(5).
                forEach(row -> {
                    IntStream.iterate(1, column -> column + 1).limit(5).
                        forEach(column -> {
                            System.out.print(column);
                        });
                    System.out.println();
                });
        };
        new Thread(r1).start();
    }
}

```

Method References

Four Types of Method References

- 1) Method reference to an **instance method** of an object – `object::instanceMethod`
- 2) Method reference to a **static method** of a class – `Class::staticMethod`
- 3) Method reference to an **instance method** of an arbitrary object of a particular type – `Class::instanceMethod`
- 4) Method reference to a **constructor** — `Class::new`

- 1) Method reference to an **instance method** of an object – `object::instanceMethod`

```
package method.reference;

interface IReference {
    void display();
}

public class First {

    public void myMethod() {
        System.out.println("Instance Method");
    }

    public static void main(String[] args) {

        First first = new First();
        IReference iReference = first::myMethod;
        iReference.display();
    }
}
```

- 2) Method reference to a **static method** of a class – `Class::staticMethod`

```
interface IReference {
    void display();
}

public class Second {

    public static void staticMethod() {
        System.out.println("Static Method display");
    }

    public static void main(String[] args) {

        IReference iReference = Second::staticMethod;
        iReference.display();
    }
}
```

- 3) Method reference to an **instance method** of an arbitrary object of a particular type

```

package method.reference;

import java.util.Arrays;

public class ThirdType {

    public static void main(String[] args) {

        String[] stringArray = { "Sam", "Richard", "Ann", "Udara", "John", "Peter", "Yasho" };

        /*
         * Method reference to an instance method of an arbitrary object of a
         * particular type
         */
        Arrays.sort(stringArray, String::compareToIgnoreCase);

        for (String str : stringArray) {
            System.out.println(str);
        }
    }
}

```

Problems Console @ Javadi

<terminated> ThirdType [Java Application]

Ann
John
Peter
Richard
Sam
Udara
Yasho

```

public class MethodReference {

    public static void main(String[] args) {

        List<String> fruites = new ArrayList<String>();
        fruites.add("Apple");
        fruites.add("Orange");
        fruites.add("Pine-Apple");
        fruites.add("Banana");
        fruites.add("Mango");
        System.out.println("Use of Lambda Expression");
        System.out.println("=====");

        fruites.forEach(
            //Use of Lambda Expression
            (fruit) -> System.out.println(fruit)
        );

        System.out.println("\nUse of Method References");
        System.out.println("=====");
        //Use of Method References
        fruites.forEach(System.out::println);
    }
}

```

Problems Console @ Javadoc Declaration

<terminated> MethodReference [Java Application] C:\Program Files\Java\jdk-9.0.4\bin\java.exe

Use of Lambda Expression
=====

Apple
Orange
Pine-Apple
Banana
Mango

Use of Method References
=====

Apple
Orange
Pine-Apple
Banana
Mango

4) Method reference to a **constructor** – **Class::new**

```

@FunctionalInterface
interface IFoodService{
    void display();
}

class Food{
    public Food(){
        System.out.print("Food Constructor");
    }
}

public class Constructor {

    public static void main(String[] args) {
        //Method reference to a constructor
        IFoodService iFoodService = Food::new;
        iFoodService.display();
    }
}

```

Problems Console @ Javadoc Declaration

<terminated> Constructor [Java Application] C:\Program Files\Java\jdk-9.0.4\bin\java.exe

Food Constructor

2018 Question 3

c) Convert below *Lambda Expression* example into *Method reference* format.

```
interface IVehicle1 {
    void displayVehicles();
}

public class Q1a {

    public static void main(String[] args) {

        IVehicle1 iVehicle = () -> {
            List<String> vehicles = new ArrayList<String>();
            vehicles.add("Car");
            vehicles.add("Bus");
            vehicles.add("Van");
            vehicles.add("Jeep");
            vehicles.add("Lorry");
            vehicles.forEach((e) -> System.out.println(e));
        };
        iVehicle.displayVehicles();
    }
}
```

Your output should be same as follows.



The screenshot shows an IDE window with a 'Problems' tab and a 'Console' tab. The console output is as follows:

```
<terminated> Q1a (1) [Java Appli
Car
Bus
Van
Jeep
Lorry
```

2018 Question 3 Answer

```
package lamda.lab;
```

```
import java.util.ArrayList;
```

```
import java.util.List;
```

```
public class Paper19ans3 {
```

```
    public static void main(String[] args) {
```

```
        List<String> vehicles = new ArrayList<String>();
```

```
        vehicles.add("Car");
```

```
        vehicles.add("Bus");
```

```
        vehicles.add("Van");
```

```
        vehicles.add("Jeep");
```

```
        vehicles.add("Lorry");
```

```
        vehicles.forEach(
```

```
            (e) -> System.out.println(e)
```

```
);
```

```
vehicles.forEach(System.out::println);
```

```
}
```

```
}
```

2022 Question 2

- b) Refer the conventional code given in the below program and refer the console output as well. Convert the program into **Method Reference** format. (10 marks)

```
ThirdType.java
1 package conventional;
2
3 import java.util.Arrays;
4
5
6 public class ThirdType {
7
8     public static void main(String[] args) {
9
10         String[] stringArray = { "sSS", "rrR", "aaa", "UUU", "Jjj", "PpP", "Yyy" };
11
12         Arrays.sort(stringArray, new Comparator<String>() {
13
14             @Override
15             public int compare(String value1, String value2) {
16                 return value1.compareToIgnoreCase(value2);
17             }
18         });
19
20         for (String value : stringArray) {
21             System.out.println(value);
22         }
23     }
24 }
```

```
Console
<terminated> Th
aaa
Jjj
PpP
rrR
sSS
UUU
Yyy
```

2022 Question 2 Answer

```
package lamda.lab;
import java.util.Arrays;

public class Paper22ans2 {
    public static void main (String[] args) {

        String[] stringArray = {
            "sSS", "rrR", "aaa", "UUU", "Jjj", "PpP", "Yyy" };

        Arrays.sort(stringArray, String::compareToIgnoreCase);

        for (String value : stringArray) {
```

```

        System.out.println(value);
    }
}

```

Code Refactor

Question 2

(40 marks)

Create a Project in Eclipse IDE and name it as **Q2** and complete the following tasks in it.

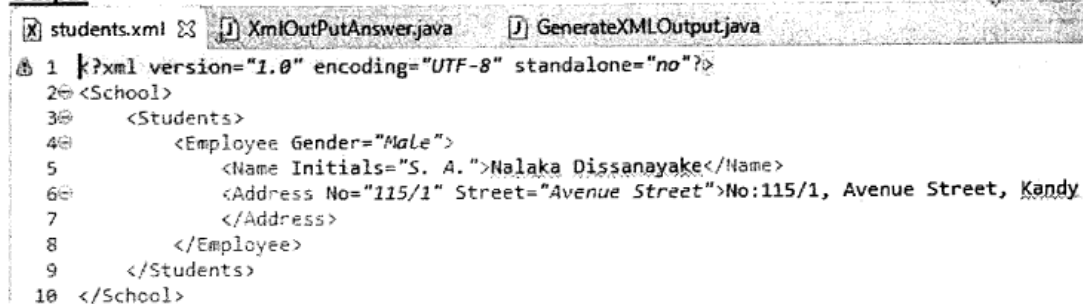
Consider the Java Coding conventions and Best Practices and **Refactor the whole code** according to the given steps of the following. The code creates an XML file to store employee specific details and generated XML is given as the sample.

```

16 public static void main(String[] args) throws Exception {
17     Document xx = DocumentBuilderFactory.newInstance().newDocumentBuilder().newDocument();
18     Element rr = xx.createElement("School");
19     xx.appendChild(rr);
20     Element vv = xx.createElement("Students");
21     rr.appendChild(vv);
22     Element w3 = xx.createElement("Student");
23     vv.appendChild(w3);
24     Attr a1 = xx.createAttribute("gender");
25     a1.setValue("Male");
26     w3.setAttributeNode(a1);
27     Element v1 = xx.createElement("Name");
28     Attr e1 = xx.createAttribute("Initials");
29     e1.setValue("S.A.");
30     v1.setAttributeNode(e1);
31     v1.appendChild(xx.createTextNode("Nalaka Dissanayake"));
32     w3.appendChild(v1);
33     Element e33 = xx.createElement("Address");
34     Attr w1 = xx.createAttribute("No");
35     w1.setValue("115/1");
36     Attr w2 = xx.createAttribute("Street");
37     w2.setValue("Avenue Street");
38     e33.setAttributeNode(w1);
39     e33.setAttributeNode(w2);
40     e33.appendChild(xx.createTextNode("No:115/1, Avenue Street, Kandy"));
41     w3.appendChild(e33);
42     Transformer w6 = TransformerFactory.newInstance().newTransformer();
43     DOMSource w7 = new DOMSource(xx);
44     w6.transform(w7, new StreamResult(new File("students.xml")));
45     w6.transform(w7, new StreamResult(System.out));
46 }
47

```

Output



```

1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2 <School>
3   <Students>
4     <Employee Gender="Male">
5       <Name Initials="S. A.">Nalaka Dissanayake</Name>
6       <Address No="115/1" Street="Avenue Street">No:115/1, Avenue Street, Kandy
7     </Address>
8   </Employee>
9 </Students>
10 </School>

```

- a) Split the code into several high-cohesive reusable operations and invoke them to generate above node hierarchy when you build the XML file. (32 marks)

```

package refactor;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerConfigurationException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;

import org.w3c.dom.*;

/**
 *
 *
 * creation of an xml document usin the DOM api
 *
 */
public class XmlOutPutAnswer {
    //constance for xml elements and attributes name
    private static final String SCHOOL_ELEMENT_NAME = "School";
    private static final String STUDENTS_ELEMENT_NAME = "Students";
    private static final String EMPLOYEE_ELEMENT_NAME = "Employee";
    private static final String GENDER_ATTRIBUTE_NAME = "gender";
    private static final String NAME_ELEMENT_NAME = "Name";
    private static final String INITIALS_ATTRIBUTE_NAME = "Initials";
    private static final String ADDRESS_ELEMENT_NAME = "Address";
    private static final String NO_ATTRIBUTE_NAME = "No";
    private static final String STREET_ATTRIBUTE_NAME = "Street";

    //output xmlfile name
    private static final String OUTPUT_FILE_NAME = "students.xml";

    /**
     * The main method , entry point of the program.
     */
    public static void main (String[] args) {
        try {
            //create a new xml document
            Document document = createDocument();

            //create the root element "School"
            Element schoolElement = createSchoolElement(document);

            //create the root element "Students"
            Element studentsElement = createStudentsElement(document);

            //create the root element "Employee"
            Element employeeElement = createEmployeeElement(document,
"Male");

            //create the root element for"name"
            Element nameElement = createNameElement(document," S. A.",
"Nalaka Disdanayake");

```



```

        //create the root element for"Address"
        Element addressElement = createAddressElement(document,
"115/1", "Avenue Street" , "No: 115/1, Avenue Street, Kandy");

        //Append child elements to their parents elements
        employeeElement.appendChild(nameElement);
        employeeElement.appendChild(addressElement);
        studentsElement.appendChild(employeeElement);
        schoolElement.appendChild(studentsElement);
        document.appendChild(schoolElement);

        //create a Transform for XML serialization
        Transformer transformer =
TransformerFactory.newInstance().newTransformer();
        DOMSource domSource = new DOMSource(document);
        StreamResult streamResult = new StreamResult(new
File("student.xml"));

        //write the xml document to a file
        transformer.transform(domSource, streamResult);
        System.out.println("XML file generated successfully !");

        //Generate additional java classes
        generateXmlOutputAnswer();
        generateGenerateXMLOutput();
    } catch (Exception e) {
        //handle exceptions properly
        e.printStackTrace();
    }
}

/**
 * Create a new XML document.
 *
 * @return The created Document object
 * @throws Exception if an error occurs during document creation
 */
private static Document createDocument() throws Exception {
    return
DocumentBuilderFactory.newInstance().newDocumentBuilder().newDocument();
}

/**
 * Creates the root element "School".
 *
 * @param document The Document object
 * @return The created School element
 */
private static Element createSchoolElement(Document document) {
    return document.createElement("School");
}

/**
 * Creates the "Students" element.

```

```

*
* @param document The Document object
* @return The created Students element
*/
private static Element createStudentsElement(Document document) {
    return document.createElement("Students");
}

/**
* Creates the "Student" element with the specified gender attribute.
*
* @param document The Document object
* @param gender The gender attribute value
* @return The created Student element
*/
private static Element createEmployeeElement(Document document, String
gender) {
    Element employeeElement = document.createElement("Employee");
    Attr genderAttr = document.createAttribute(GENDER_ATTRIBUTE_NAME);
    genderAttr.setValue(gender);
    employeeElement.setAttributeNode(genderAttr);
    return employeeElement;
}

/**
* Creates the "name" element with initials attribute and text content.
*
* @param document The Document object
* @param initials The initials attribute value
* @param name The text content of the name element
* @return The created name element
*/
private static Element createNameElement(Document document, String
initials, String name) {
    Element nameElement = document.createElement(NAME_ELEMENT_NAME);
    Attr initialsAttr =
document.createAttribute(INITIALS_ATTRIBUTE_NAME);
    initialsAttr.setValue(initials);
    nameElement.setAttributeNode(initialsAttr);
    nameElement.appendChild(document.createTextNode(name));
    return nameElement;
}

/**
* Creates the "Address" element with attributes and text content.
*
* @param document The Document object
* @param number The value of the "No" attribute
* @param street The value of the "Street" attribute
* @param addressText The text content of the Address element
* @return The created Address element
*/
private static Element createAddressElement(Document document, String
number, String street, String addressText) {
    Element addressElement =
document.createElement(ADDRESS_ELEMENT_NAME);
    Attr numberAttr = document.createAttribute(NO_ATTRIBUTE_NAME);

```

```

        numberAttr.setValue(number);
        addressElement.setAttributeNode(numberAttr);
        Attr streetAttr = document.createAttribute(STREET_ATTRIBUTE_NAME);
        streetAttr.setValue(street);
        addressElement.setAttributeNode(streetAttr);
        addressElement.appendChild(document.createTextNode(addressText));
        return addressElement;
    }
    /**
     * Creates a Transformer for XML serialization.
     *
     * @return The created Transformer object
     * @throws TransformerConfigurationException if an error occurs during
Transformer creation
     */
    private static Transformer createTransformer() throws
TransformerConfigurationException {
        return TransformerFactory.newInstance().newTransformer();
    }

    /**
     * Generates the "XmlOutputAnswer" class with its content.
     *
     * @throws IOException if an error occurs during file writing
     */
    private static void generateXmlOutputAnswer() throws IOException {
        String xmlOutputAnswerContent = ""

        public class XmlOutputAnswer {
            public static void main(String[] args) {
                System.out.println("XmlOutputAnswer: XML output
generated successfully!");
            }
        }
        "";

        FileWriter fileWriter = new FileWriter("XmlOutputAnswer.java");
        fileWriter.write(xmlOutputAnswerContent);
        fileWriter.close();
    }

    /**
     * Generates the "GenerateXMLOutput" class with its content.
     *
     * @throws IOException if an error occurs during file writing
     */
    private static void generateGenerateXMLOutput() throws IOException {
        String generateXMLOutputContent = ""

        public class GenerateXMLOutput {
            public static void main(String[] args) {
                System.out.println("GenerateXMLOutput: XML output
generated successfully!");
            }
        }
        "";
    }

```

```

        FileWriter fileWriter = new FileWriter("GenerateXMLOutput.java");
        fileWriter.write(generateXMLOutputContent);
        fileWriter.close();
    }
}

```

Lecture 4

Micro services and Cloud Computing

1.1 What Is Cloud?

- Refers to servers that are accessed over the Internet, and the software and databases that run on those servers.
- Can provide services through private and public networks.
- Present at a remote location.
Applications: e-mail, Web conferencing, Customer Relationship Management (CRM)

1.2 What Is Cloud Computing?

- The delivery of computing resources as a service over the cloud. Computing resources are servers, storage, databases, networking, software, analytics, intelligence, and etc.
- Cloud computing services are typically delivered by third-party providers who own and operate the necessary hardware and software infrastructure. These providers offer a range of services, including infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS).
- Many advantages of cloud computing exist, such as improved scalability, flexibility, and cost-efficiency. As users can access their programs and data from any location with an internet connection, it also makes remote work and collaboration easier.

1.3 How would you know your solution should have cloud architecture?

1. Flexibility and agility: If your solution requires rapid deployment and the ability to iterate and update software quickly, cloud architecture can provide the necessary flexibility. Cloud platforms often offer services and tools for continuous integration, deployment, and monitoring, enabling faster development cycles.
2. Geographic distribution and accessibility: If your solution needs to be accessible from multiple locations or if you have a global user base, cloud architecture can provide geographically distributed data centers, reducing latency and improving user experience. Additionally, cloud-based solutions are easily accessible over the internet, allowing remote access and collaboration.
3. Disaster recovery and reliability: Cloud providers often have robust backup and disaster recovery mechanisms in place, including data replication and automatic failover. If your solution requires high availability and business continuity, cloud architecture can enhance reliability and minimize downtime.

1.4 What was before cloud computing?

Prior to the widespread use of cloud computing, companies and regular computer users were often required to purchase and maintain the software and hardware they desired to utilize. Businesses and consumers now have access to a variety of on-demand computing resources as internet-accessed services because to the rising availability of cloud-based apps, storage, services, and devices. The move from on-premises software and hardware to networked remote and distributed resources frees cloud customers from having to put in the time, money, or technical know-how necessary to acquire and manage these computing resources independently.

1.5 How Does Cloud Computing Work?

Step 1 → Allocating resources: A cloud service provider keeps a pool of computing resources, including servers, storage, and networking equipment. Customers receive these resources from the provider as needed. Consumers can decide what kind and how much of a resource they require, and only pay for what they really use.

Step 2 → Virtualization: To construct virtual instances of computing resources, cloud providers employ virtualization technology. This makes it possible for several clients to share the same physical hardware while maintaining the privacy and security of their computing environments. Each virtual instance can be tailored to the client's requirements and run its own operating system and apps.

Step 3 → Network connectivity: Cloud service providers use the Internet to give network connectivity to their computing resources. If a customer has an Internet connection, they can access their virtual instances from anywhere in the globe. Several connectivity options, ranging from simple Internet access to dedicated private networks, are frequently provided by cloud providers.

Step 4 → Service delivery: cloud providers include infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS).

Step 5 → Scalability: Cloud providers can scale their resources up or down to meet the changing needs of their customers. This allows customers to easily adapt to changes in demand, without having to invest in additional hardware and infrastructure.

1.6 Examples of Cloud Computing

- Gmail, Facebook, Banking, Financial Services, Health care, Education

1.7 Cloud service Providers

- Amazon Web Services (AWS), Microsoft Azure, Google Cloud, Alibaba Cloud, IBM Cloud, Oracle, Salesforce, SAP.

1.8 Different cloud service providers

Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP) are among the most popular cloud service providers. They offer similar services, but there are differences in their offerings, pricing, and capabilities. Here is a comparison of the three cloud service providers,

1 - Service Offerings

- Amazon Web Services (AWS) is the oldest cloud provider and offers a broad range of services, including computer, storage, databases, analytics, machine learning, and IoT services.

- Azure is Microsoft's cloud offering, providing similar services to AWS, along with a suite of Microsoft-specific products like Office 365 and Dynamics 365.
- GCP is Google's cloud platform that offers a range of services, including computing, storage, networking, big data, machine learning, and IoT.

2 – Pricing

- AWS offers an on-demand pricing model, which means you pay only for what you use.
- Azure also follows a similar pricing model.
- GCP pricing is based on sustained use, which means the longer you use an instance, the cheaper the hourly rate becomes.

3 - User Interface

- AWS and Azure have more complex user interfaces compared to GCP, which has a more simplified, user-friendly interface.
- GCP has a more intuitive user interface and is easier for beginners to use.

4 - Cloud Storage

- AWS, Azure, and GCP offer similar cloud storage services, with AWS offering Amazon S3 and Azure offering Azure Blob Storage.
- GCP offers Google Cloud Storage, which is highly scalable and offers greater flexibility in managing data.

5 - Machine Learning

- AWS, Azure, and GCP have advanced machine learning capabilities, but GCP is considered the most advanced in this area.
- GCP offers a range of machine learning services, including image and speech recognition, natural language processing, and translation.

1.9 Cloud Computing Services (SAAS/PAAS/IAAS)

1 - Software AS A Service (SAAS)

- Software as a Service (SaaS) is a method of delivering software in which clients can access hosted software applications online from a third-party source. Customers have the option of accessing the program through a web browser or mobile app rather than installing and maintaining it on their own desktops or servers.
- SaaS is a popular delivery model for many types of business applications, including customer relationship management (CRM), enterprise resource planning (ERP), human resources (HR), and many others. SaaS applications are typically sold on a subscription basis, with customers paying a monthly or annual fee for access to the software.
- One of the main benefits of SaaS is that it allows businesses to avoid the upfront costs of purchasing and installing software, as well as the ongoing costs of maintaining and upgrading it. SaaS providers handle all the software maintenance and updates, freeing up customers to focus on their core business activities.

EX: - Facebook, Dropbox, Google Workspace, Office 365, HubSpot

Features & Benefits

- Flexible payments as the users pay for the services on a pay-as-you-go model.
- Software updates are automatic and free of charge.
- There are no upfront costs because users can begin using the application as soon as they subscribe.
- Provides cross-device compatibility. (SaaS applications can be accessed through any internet enabled devices).

2 - Platform AS A Service (PaaS)

- Platform as a service (PaaS) is a cloud computing model where a third-party provider delivers hardware and software tools to users over the internet. Usually, these tools are needed for application development. A PaaS provider hosts the hardware and software on its own infrastructure. As a result, PaaS frees developers from having to install in-house hardware and software to develop or run a new application.
- PaaS providers typically offer a range of services and tools for application development, including programming languages, application frameworks, databases, and development tools. Customers can build and deploy their applications on the PaaS platform, and the provider handles all of the underlying infrastructure and maintenance.

EX: - Windows Azure, App Engine, Force.com, OpenShift, Heroku

Features & Benefits

- PaaS service provider handles all the update patches, upgrades, and regular software maintenance.
- Makes software development easy.
- Enables location independence by allowing developers from several locations to collaborate on the same application.

3 - Infrastructure AS A Service (IaaS)

- Infrastructure as a service (IaaS) is a type of cloud computing service that offers essential computer, storage, and networking resources on demand, on a pay-as-you-go basis. IaaS is one of the four types of cloud services, along with software as a service (SaaS), platform as a service (PaaS), and serverless.
- Migrating your organization's infrastructure to an IaaS solution helps you reduce maintenance of on-premises data centers, save money on hardware costs, and gain real-time business insights. IaaS solutions give you the flexibility to scale your IT resources up and down with demand. They also help you quickly provision new applications and increase the reliability of your underlying infrastructure.

EX: - Digital Ocean, Microsoft Azure, Google Compute Engine, Amazon EC2

Features & Benefits

- Saves both time and money.
- Resources are available on demand.
- Utility-based pricing model

1.10 Benefits of cloud computing

1 - cost savings → Businesses no longer need to spend money on pricey infrastructure, software, and hardware thanks to cloud computing. Instead, they subscribe to the services they require, which can eventually be more affordable.

2 – Scalability → Cloud computing allows businesses to easily scale up or down their IT infrastructure as needed. This means they can quickly and easily add or remove resources as their needs change.

3 – Flexibility → In comparison to conventional on-premises systems, cloud computing provides more flexibility. Customers can utilize a number of devices to access cloud-based applications and services from anywhere with an internet connection.

4 – Accessibility → More accessibility is provided by cloud computing than by conventional on-premises solutions. Customers can utilize a number of devices to access cloud-based applications and services from anywhere with an internet connection.

5 – Quality Control → There are few things as detrimental to the success of a business as poor quality and inconsistent reporting. In a cloud-based system, all documents are stored in one place and in a single format. With everyone accessing the same information, you can maintain consistency in data, avoid human error, and have a clear record of any revisions or updates.

1.11 Discuss the challenges associated with cloud computing.

1 - Data Security and Privacy → When working in cloud environments, data security is a top priority. One of the main issues with cloud computing is that users must be responsible for their data and that not all cloud service providers can guarantee complete data privacy. Absence of visibility and control tools, no identity access management, data misuse, and Cloud misconfiguration are the common factors behind Cloud privacy leaks. Insecure APIs, malicious insiders, and errors or carelessness in cloud data management are further issues that need to be addressed.

2 - Multi-Cloud Environments → Configuration failures, a lack of security patches, poor data governance, and lack of granularity are common cloud computing problems and challenges in multi-cloud setups. It is challenging to monitor the security needs of several clouds and implement data management guidelines across numerous boards.

3 – Integration → Moving applications and data to the cloud can be a complex process that requires significant planning and coordination. Organizations must ensure that their existing systems and applications can be integrated with their cloud solution.

4 – Downtime → Cloud computing is not exempt from it like any other type of technology. Businesses may find it challenging to access vital data and applications during a cloud service outage, which might potentially cause delays to their operations.

5 – Compliance → Businesses need to make sure their cloud computing solution complies with regulations that are relevant to their sector. Businesses that operate in tightly regulated sectors like healthcare and banking may find this particularly difficult.

1.12 Analyze real-world examples of cloud computing.

1 - Microsoft Office 365

Users can view and work together on documents, spreadsheets, and presentations from any location with an internet connection thanks to Microsoft's cloud-based productivity suite. Businesses no longer need to host their own email and document management systems as a result.

2 – Netflix

Netflix is a streaming video service that relies on cloud computing to deliver content to its millions of subscribers. The company uses Amazon Web Services (AWS) to host its video files and stream them to viewers around the world.

3 - Amazon Web Services (AWS)

One of the most popular cloud computing platforms is Amazon, which provides a variety of services such as computation, storage, and databases. AWS is used by many companies to host websites, store data, and operate applications.

4 - Dropbox

Users can save and share files online using Dropbox, a cloud storage service. Users can access their files from any device with an internet connection thanks to cloud computing. Dropbox stores and manages user files using Amazon Web Services (AWS) and its own data centers.

5 - Instagram

To store and distribute photographs and videos to its millions of users worldwide, Instagram, the well-known photo-sharing app, leverages cloud computing. The app can handle high volumes of traffic and user uploads since it stores and manages user data on Facebook's cloud infrastructure.

6 - Google Docs

Google Docs is a cloud-based productivity suite that allows users to create and collaborate on documents, spreadsheets, and presentations. Google Docs uses cloud computing to store data on servers located in various geographic locations, which ensures that data is accessible and secure. By utilizing cloud computing, Google Docs can provide users with a flexible and collaborative productivity solution.

7 - Uber

Uber, the ride-sharing platform, uses cloud computing to handle its massive amounts of data, including user data, ride requests, and driver information. The company uses AWS and Google Cloud Platform to host its servers and databases, which allows it to quickly respond to user requests and scale its infrastructure as needed.

1.13 Discuss emerging trends in cloud computing.

There are several emerging trends that are shaping the future of the industry. Here are some of the most significant trends,

1 - Hybrid and Multi-Cloud Environments

To suit their objectives, many firms are increasingly deciding to use a hybrid cloud that combines elements of public, private, and third-party clouds. Organizations can optimize the cost, performance, and security of their IT infrastructure using a multi-cloud strategy.

2 - Serverless Computing

Developers can run code using serverless computing without caring about the underlying infrastructure. Because it may lower expenses and increase scalability, this approach is becoming more and more common.

3 - Edge Computing

Edge computing has become a crucial trend in cloud computing due to the growth of IoT devices and the requirement for low-latency processing. By moving computers closer to the data source, edge computing can increase performance and lower bandwidth needs.

4 - Cloud Native Technologies

Traditional technologies are less flexible and scalable than cloud native technologies, which are created expressly for the cloud environment. Cloud-native technologies are gaining prominence, and examples include containers, microservices, and serverless computing.

5 - Security and Compliance

Security and compliance issues are becoming more and more important as cloud adoption increases. To allay these worries and give clients the assurance they require, cloud providers are actively investing in security and compliance procedures.

1.14 Monolithic Architecture and Peek into microservices.

Monolithic architecture is a traditional software development approach where an application is built as a single, self-contained unit. In this architecture, all components and modules of the application are tightly coupled and interconnected. The entire application is deployed as a single monolith, typically running on a single server or a cluster of servers.

Key characteristics of monolithic architecture include:

1. **Single Codebase:** The entire application is developed, deployed, and managed as a single codebase. All functionalities and features are tightly integrated within the same codebase.
2. **Tight Coupling:** The components within the application are tightly coupled, meaning that changes or updates in one module may require modifications across the entire application.

While monolithic architecture has been widely used and served well in many scenarios, it also has limitations. Large monolithic applications can become complex, difficult to maintain, and hinder agility and scalability. They can also experience performance bottlenecks and challenges with fault isolation and independent deployment of components.

Microservices architecture is an alternative approach to building applications that aims to address the limitations of monolithic architecture. In a microservices architecture, an application is broken down into a collection of smaller, loosely coupled services that can be developed, deployed, and scaled independently.

Key characteristics of microservices architecture include:

1. Service-oriented: The application is divided into separate services, each responsible for a specific business capability. Each service has its own codebase, data storage, and may communicate with other services through well-defined APIs.
2. Loosely Coupled: Services are decoupled and can be developed and deployed independently. This allows for easier maintenance, scalability, and the ability to adopt different technologies and frameworks for different services.
3. Distributed Deployment: Services in a microservices architecture can be deployed on different servers or even different environments, enabling flexibility and resilience.

1.15 Optimized Code: Efficiency, Performance, Readability, Scalability

1.16 Messy Code: Lack of Structure, Poor Naming and Documentation, Performance Issues:

1.17 When to refactor and when to not to... =

Code Readability and Maintainability: If the codebase becomes difficult to understand, modify, or maintain due to complex logic, poor naming conventions, or lack of proper structure, it may be a good time to consider refactoring. Refactoring can make the code more readable, modular, and easier to work with, enhancing long-term maintainability.

Code Duplication: When you notice duplicated code or similar logic scattered across different parts of the codebase, it's a clear indication to refactor. Code duplication leads to maintenance issues, as any changes or bug fixes need to be applied in multiple places. Refactoring can help eliminate duplication by extracting reusable functions or creating abstractions.

Adding New Features or Functionality: When introducing new features or functionality, it's often beneficial to refactor existing code to accommodate the changes more efficiently. Refactoring can make the code more flexible and extensible, reducing the complexity of integrating new functionality.

1.18 refactoring may not be necessary or not the highest priority:

Lack of Resources: Refactoring can be time-consuming, and it requires resources in terms of development effort and testing. If there are resource constraints or higher-priority tasks, it may be more practical to postpone refactoring.

Minimal Impact: If the existing codebase is functioning well, is well-structured, and does not exhibit major issues, refactoring may not provide substantial benefits. In such cases, it may be more efficient to focus on other development tasks.

API Gateway

1.19 Products with API Gateway features

Amazon API Gateway (AWS): Amazon API Gateway is a fully managed service provided by Amazon Web Services (AWS). It allows developers to create, publish, and manage APIs for their applications. It supports features like authentication, authorization, rate limiting, caching, and request/response transformations.

Google Cloud Endpoints: Google Cloud Endpoints is a scalable API management platform provided by Google Cloud Platform (GCP). It enables developers to design, deploy, and manage APIs for their applications. It offers features such as authentication, API key management, rate limiting, logging, and monitoring.

Microsoft Azure API Management: Azure API Management is a comprehensive API management solution provided by Microsoft Azure. It allows developers to create, publish, and secure APIs for their applications. It provides features like authentication, rate limiting, caching, analytics, and developer portal

Microservice vs Monolith Architecture

Monolithic	Microservices
One Solution that contains all the Business Components and Logics.	Multiple Services that contain one single
Single Database	Dedicated Database for each microservi
One Language for the Backend	Can have multiple technologies / languag microservice.
Solution has to be deployed to a single Server. Physical Separation of concerns can be tricky.	Each of the Microservice can be deployed web.
Services will be tightly coupled to the application itself.	Loosely Coupled Architecture.

1.20 What is a use of API

An API acts as an intermediary between two software applications, enabling them to communicate and exchange data. It provides a defined interface through which developers can make requests, send data, and receive responses from the software component it represents.

Why we are having a API gate way(reasons)

Centralized API Management: An API Gateway provides a centralized entry point for client applications to access multiple backend services. It acts as a single point of control, allowing you to manage and monitor your APIs more efficiently. Rather than directly exposing individual services, clients interact with the API Gateway, which handles the complexities of routing requests to the appropriate services.

Service Aggregation and Composition: An API Gateway can aggregate data or functionality from multiple backend services into a single API endpoint. It can fetch data from different services and combine or transform it before returning a response to the client. This enables you to provide a unified and cohesive API to clients, simplifying their interactions and reducing the number of requests they need to make.

Overall, an API Gateway simplifies the complexity of managing, securing, and orchestrating interactions between client applications and backend services. It provides a range of functionalities that streamline development, improve performance, enhance security, and enable better control and monitoring of your APIs.

1.21 API products available in the market:

Amazon API Gateway: Amazon API Gateway is a fully managed service by Amazon Web Services (AWS) that makes it easy to create, deploy, and manage APIs at any scale. It offers features like request throttling, authorization and access control, caching, and integration with other AWS services.

Google Cloud Endpoints: Google Cloud Endpoints is a distributed API management system provided by Google Cloud Platform. It allows developers to easily create, deploy, and manage APIs using Google Cloud tools and services. It provides features like authentication, monitoring, and support for multiple programming languages.

1.22 Why do we use ocelot?

Ocelot is an open-source API Gateway built specifically for the .NET ecosystem. Here are some reasons why you might consider using Ocelot:

Simplified API Gateway Implementation: Ocelot provides a lightweight and easy-to-use solution for implementing API Gateway functionality in your .NET applications. It allows you to quickly set up an API Gateway without the need for complex configurations or heavy dependencies.

Routing and Load Balancing: Ocelot allows you to define routing rules to direct incoming requests to the appropriate backend services. It supports various routing strategies and load balancing algorithms, enabling you to distribute traffic across multiple instances of your services.

Service Aggregation: With Ocelot, you can aggregate data or responses from multiple backend services into a single API response. This simplifies the client's interaction by providing a unified API endpoint, reducing the number of requests required and improving performance.

Open-Source and Community Support: Ocelot is an open-source project with an active community of contributors and users. This means you can benefit from community-driven enhancements, bug fixes, and support resources.

1.23 What are the rules and responsibilities of ocelot?

Routing: Ocelot is responsible for routing incoming requests to the appropriate backend services based on predefined routing rules. It examines the request URL, HTTP method, and other criteria to determine the destination service for processing the request.

Load Balancing: Ocelot can distribute incoming traffic across multiple instances of the same backend service using load balancing algorithms. It ensures that requests are evenly distributed among the available service instances, improving performance, scalability, and resource utilization.

Request Transformation: Ocelot allows you to modify or transform incoming requests before forwarding them to backend services. This includes manipulating headers, modifying payloads, or adding additional information to the request based on specific requirements.

Service Aggregation: Ocelot can aggregate data or responses from multiple backend services into a single API response. This simplifies the client's interaction by providing a unified API endpoint, reducing the number of requests required, and improving overall performance.

Monitoring and Logging: Error Handling and Fault Tolerance:

1.24 Upstream and the downstream in API request

Upstream: The upstream direction refers to the flow of the request from the client towards the backend services. When a client initiates an API request, it sends the request to the API Gateway, which acts as the entry point for all incoming requests. The API

Gateway then forwards the request upstream to the appropriate backend service based on the routing rules configured. (The request coming from client to the gateway is the Upstream)

Downstream: The downstream direction refers to the flow of the response from the backend services back to the client. Once the backend service processes the request and generates a response, it sends the response back downstream through the API Gateway. The API Gateway receives the response from the backend service and forwards it to the client that made the original request.

1.25 What's its important(ocelot/up/downstream)

Request Routing: Ocelot uses the upstream flow to route incoming requests from clients to the appropriate backend services. By examining the request details, such as the URL, HTTP method, or headers, Ocelot determines which backend service should handle the request. Accurate routing ensures that requests reach the intended service for processing.

Load Balancing: When multiple instances of a backend service are available, Ocelot can distribute incoming requests among them using load balancing algorithms. By evenly distributing the requests across service instances, Ocelot optimizes resource utilization and enhances the performance and scalability of the system.

Authentication and Authorization: , **Error Handling and Fault Tolerance:** , **Response Processing:**

API GATEWAY PATTERN VS DIRECT CLIENT – MICRO SERVICE COMMUNICATION

The API Gateway pattern and direct client-microservice communication are two different approaches to handle communication between clients and microservices. Here are some key differences:

1.26 API Gateway Pattern:

Centralized Communication: In the API Gateway pattern, clients interact with a centralized API Gateway instead of directly communicating with individual microservices. The API Gateway serves as a single-entry point for clients and handles routing requests to the appropriate microservices based on predefined rules.

Aggregation and Composition: The API Gateway can aggregate data or responses from multiple microservices into a single API response. It can fetch data from different services and combine or transform it before returning a unified response to the client. This simplifies client interactions by reducing the number of requests needed and providing a unified API endpoint.

Security and Access Control: The API Gateway handles authentication and authorization, enforcing access control policies and validating client credentials. It acts as a security layer for the microservices, protecting them from unauthorized access and ensuring consistent security measures across the services.

Request Transformation and Protocol Translation: The API Gateway can perform request transformation, allowing clients to use a standardized protocol (e.g., REST/HTTP) while internally communicating with microservices using their preferred protocols. This abstraction helps decouple clients from specific implementation details of microservices.

1.27 Direct Client-Microservice Communication:

Decentralized Communication: In direct client-microservice communication, clients communicate directly with individual microservices without an intermediary API Gateway. Each client must know the specific endpoints and interfaces of the microservices it wants to interact with.

Fine-Grained Communication: Direct communication allows clients to interact with microservices at a fine-grained level. Clients can directly call specific microservice endpoints and access their functionalities without going through an additional layer.

Complexity and Scalability: Direct communication can lead to increased complexity as clients need to handle service discovery, load balancing, and fault tolerance individually. Scaling individual microservices can also be more challenging without a centralized layer like an API Gateway.

Limited Cross-Cutting Concerns: Cross-cutting concerns such as authentication, request/response transformation, rate limiting, and caching need to be implemented individually in each microservice or handled by separate infrastructure components.

1.28 What are the main features in API gateway pattern?

Reverse Proxy or Gateway Routing: One of the primary features of the API Gateway pattern is acting as a reverse proxy or gateway for incoming client requests. It sits between clients and backend microservices, intercepting requests and routing them to the appropriate microservice based on predefined rules.

Request Aggregation: The API Gateway has the capability to aggregate data or responses from multiple microservices into a single API response. Instead of clients having to make multiple requests to different microservices to gather all the required data, the API Gateway can orchestrate and coordinate these requests on behalf of the client.

When a client sends a request to the API Gateway that requires data from multiple microservices, the API Gateway can internally initiate parallel or sequential requests to the respective microservices. It collects the responses from these microservices, combines the data, and constructs a unified response to send back to the client.

1.29 There are several reasons why Azure is a popular choice for individuals and organizations:

Global Presence: Azure has a vast global presence with data centers located in numerous regions worldwide. This enables businesses to deploy their applications and services closer to their target audience, improving performance, reducing latency, and complying with data sovereignty requirements.

Scalability and Elasticity: Azure allows businesses to scale their resources up or down based on demand. It offers auto-scaling capabilities that automatically adjust resources to handle workload fluctuations. This scalability and elasticity enable businesses to optimize costs and ensure optimal performance for their applications.

Cost-Effectiveness: Azure offers flexible pricing models, including pay-as-you-go and reserved instances, allowing businesses to optimize costs based on their usage patterns and requirements. It provides cost management tools, budgeting options, and cost-saving recommendations to help businesses effectively manage their cloud expenses.

Security and Compliance: Azure emphasizes security and provides robust measures to safeguard customer data and applications. It adheres to various compliance standards, including ISO, GDPR, HIPAA, and more. Azure also offers features such as role-based access control (RBAC), encryption, network security, threat detection, and monitoring tools to enhance data and application security.

Hybrid Capabilities, Developer-friendly Environment

1.30 Creating resources in Azure

- ▶ Key Azure Resources....
- ▶ Resources Groups – Logical grouping of azure artifacts and other resources
- ▶ Azure Key Vault
- ▶ App Services

- ▶ Web apps
- ▶ APIM – API Management in Azure (Similar to API Gateway in on prem)

1.31 Azure API Management:

Azure API Management (APIM) is a fully managed service offered by Microsoft Azure that allows organizations to publish, secure, and manage APIs (Application Programming Interfaces). It acts as a gateway between backend services and external clients or developers who want to consume those services.

API Gateway: Azure APIM acts as an API gateway, providing a single-entry point for clients to access multiple APIs. It handles the routing, authentication, and request/response transformations required for API communication. Key features-

API Publishing and Management: APIM enables organizations to publish their APIs and provide developers with access to documentation, usage policies, and subscription plans. It allows versioning of APIs, making it easier to introduce changes without disrupting existing clients.

Traffic Management: APIM offers features to control and manage API traffic. It supports rate limiting, allowing you to control the number of requests clients can make within a given time period. It also enables caching of API responses to improve performance and reduce backend load.

1.32 Security and Authorization, Analytics and Monitoring, Extensibility

By using Azure API Management, organizations can simplify the management and governance of their APIs, enhance security, enforce usage policies, monitor API performance, and provide a developer-friendly experience. It helps organizations expose their APIs to internal or external developers, partners, or third-party applications in a controlled and secure manner.

1.33 Different domains in azure

There are several different domains that represent various services, capabilities, and areas of focus. Here are some of the key domains in Azure:

Compute: This domain encompasses services related to virtual machines (VMs) and container instances, such as Azure Virtual Machines, Azure Container Instances, Azure Kubernetes Service (AKS), and Azure Batch.

Storage: The storage domain includes services for storing and managing data, including Azure Blob Storage (for unstructured data), Azure File Storage (for file shares), Azure Disk Storage (for persistent disks), and Azure Data Lake Storage (for big data analytics).

Networking, Databases, DevOps and Developer Tools, IOT

1.34 web mobile services

Azure Web and Mobile Services is an umbrella term used to refer to a set of Azure services designed specifically for developing, deploying, and managing web and mobile applications. These services provide a platform for building scalable, reliable, and high-performance applications without the need for managing underlying infrastructure. Here are some key Azure services that fall under the Web and Mobile Services category:

Azure Notification Hubs: Azure Notification Hubs is a scalable, cross-platform push notification service. It allows you to send push notifications to mobile devices, web browsers, and other platforms, enabling real-time engagement with your applications.

Azure Logic Apps: Azure Logic Apps provides a low-code approach to building scalable workflows and integrations. It allows you to create workflows by connecting various pre-built connectors and triggers, enabling integration with external systems, data transformation, and business process automation.

Web app: Azure web app enables you to build and host websites in the programming language of your choice without managing infrastructure.

Azure Static Web Apps, Azure Functions, Azure App Service,

1.35 security and identity services?

Azure offers a range of security and identity services to help protect your applications, data, and users. These services provide features for identity management, access control, threat detection, and encryption. Here are some key security and identity services provided by Azure:

Key Vault: Azure key vault helps safeguard cryptographic keys and secrets used by cloud applications and services and streamlines the key management process.

Azure Active Directory (Azure AD): Azure AD is a cloud-based identity and access management service that allows you to manage user identities and control access to resources. It provides features like single sign-on (SSO), multi-factor authentication (MFA), role-based access control (RBAC), and integration with thousands of SaaS applications.

Azure Active Directory Domain Services (Azure AD DS): Azure AD DS allows you to join Azure VMs to a domain without the need for running domain controllers. It enables legacy applications that rely on Active Directory for authentication and identity management to run in Azure.

Azure Advanced Threat Protection (Azure ATP), Azure Information Protection, Azure Security Center

Lecture 5

Stereo vision, Stereoscopy & Virtual Reality

Stereo Vision, Stereoscopy, and Virtual Reality are concepts and technologies that are interconnected and commonly used in the field of 3D imaging and immersive experiences. Here's an overview of these concepts and some key terminology associated with them:

Stereo Vision: Stereo vision refers to the perception of depth and three-dimensional (3D) visual information using two eyes. It is the ability of the brain to combine the slightly different views captured by each eye to create a single 3D perception of the world. Stereo vision is crucial for depth perception and the ability to judge distances.

Stereoscopy: Stereoscopy is a technique used to create or reproduce the perception of depth in images or videos. It involves capturing or displaying two offset images, one for each eye, which are then viewed with special glasses or devices to create a 3D effect. Stereoscopy is often used in movies, photography, and virtual reality to provide an immersive and realistic visual experience.

Virtual Reality (VR): Virtual Reality is a computer-generated simulation of a 3D environment that can be interacted with and explored by a user. It typically involves the use of a head-mounted display (HMD) or VR headset, which provides a

stereoscopic view of the virtual world. VR can create an immersive and interactive experience by simulating a sense of presence and allowing users to interact with virtual objects or environments.

VR= Feeling of the existence of a non-existing object or environment.

Augmented Reality

AR stands for Augmented Reality. It is a technology that combines virtual elements with the real world, overlaying digital information onto the user's view of the physical environment. AR enhances the real-world environment by adding computer-generated graphics, sounds, or other sensory inputs to provide an interactive and immersive experience.

Virtual reality	Augmented reality
Immerses the viewer into computer-generated environments. Requires equipment which completely obstructs visual view of physical objects in the real world. Does not use a camera feed. All the things displayed in VR are either animations or prerecorded bits of films.	Augments or adds graphics, audio, and other sensory enhancements to the natural world as it exists. Use Camera feed not Animations. System augments the real-world scene. Needs a mechanism to combine virtual and real worlds. User maintains a sense of presence in real world.

Marker-based AR applications: AR Gaming: Marker-based AR is widely used in gaming applications, where markers or specific visual cues are used to trigger interactive virtual elements, characters, or gameplay features. Product Visualization: Marker-based AR can be used to overlay virtual content onto physical products, allowing customers to visualize how products would look or function in real-world settings before making a purchase. Interactive Print Media: Magazines, newspapers, or advertising materials can incorporate markers that, when scanned by a mobile device, display additional digital content, such as videos, animations, or interactive elements. Education: Marker-based AR is utilized in educational materials, such as textbooks or learning apps, to provide supplementary interactive content, 3D models, or augmented explanations.	Marker-less AR applications: Indoor Navigation: Marker-less AR can assist users in navigating indoor environments, such as shopping malls, airports, or museums, by overlaying directional arrows, points of interest, or real-time information onto the camera view. Real Estate: Marker-less AR allows potential buyers or renters to visualize and explore properties by overlaying virtual furniture, design options, or renovation plans onto real-world spaces. Industrial Maintenance and Training: Marker-less AR is used in industries such as manufacturing or maintenance to provide technicians with real-time information, step-by-step instructions, or virtual annotations for repairing or assembling complex equipment. Tourism and Travel: Marker-less AR applications can enhance the travel experience by providing virtual tour guides, historical information, or interactive elements in popular tourist locations.
---	---

Augmented Reality (AR) has a wide range of applications across various industries. Here are some notable examples:

Gaming and Entertainment: AR gaming applications like Pokémon Go and Harry Potter: Wizards Unite have gained significant popularity. They allow users to interact with virtual characters and objects in the real world, creating immersive and interactive gaming experiences. AR is also used in entertainment, such as AR concerts or interactive museum exhibits.

E-commerce and Retail: AR is used in e-commerce to enhance the shopping experience. Customers can virtually try on clothing, accessories, or makeup using AR applications, allowing them to see how products look on themselves before making a purchase decision. AR is also used to visualize furniture or home decor in a user's space.

Education and Training: AR has great potential in education and training. It can provide interactive and engaging learning experiences, such as overlaying 3D models or information on textbooks, creating virtual science experiments, or simulating historical events. AR is also used for training simulations in industries like healthcare, aviation, and military.

Industrial and Manufacturing, Marketing and Advertising, Healthcare , Architecture and Real Estate.

2022 Question

You are working for a solution-based organization and your client requires you to design and develop a system for a university. The business goal is to develop web services and expose them to some external web content providers to invoke student data in a more personalized manner. In the initial step, client wants to narrow down the development to two services.

- a) Student Profile
- b) Class Timetable

Also, the client would like to host the services in Azure.

- a) Your first task is to identify a couple of Azure resources you will be using to develop the solution. Identify three Azure resources and briefly explain the purpose of each, stating their pros and cons.
- b) Client requires to have one base URL for both student service and class timetable service. What is your approach to cater that requirement? What is the specific Azure Service you will be using for that? Justify your selection.
- c) During the first demonstration client requested you to implement authentication. What technologies you would choose? Justify your selection.

Answers

- a) **1 - Azure SQL Database:** A managed relational database service offered by Azure is called Azure SQL Database. It offers the university's student data high-performance, scalable, and secure storage. The Student Profile service may store and retrieve student data using Azure SQL Database. Pros and drawbacks are listed below:

Pros:

Managed database service with built-in high availability and automatic backups.
Scalability options to handle varying workloads.
Advanced security features to protect data.

Cons:

Cost can increase with the size of the database and data transfer.
Limited control over underlying database management system.

2 - Azure App Service: The fully managed Azure App Service platform lets programmers create, launch, and scale web applications and APIs. It is appropriate for creating the services for Student Profile and Class Timetable because it supports a variety of programming languages and frameworks. Its benefits and drawbacks include:

Pros:

Easy deployment and management of web applications.
Automatic scaling and load balancing.
Integration with Azure Active Directory for authentication and authorization.

Cons:

Limited control over underlying infrastructure.
Higher cost compared to other compute options like Azure Virtual Machines

3 - Azure Logic Apps: Azure Logic Apps is a cloud-based service that provides a visual interface to automate business processes and workflows. It can be used to integrate and orchestrate data flows between different systems and services. For example, the Class Timetable service can use Azure Logic Apps to fetch and process timetable data from external providers. Here are the pros and cons:

Pros:

Easy-to-use visual designer for creating workflows.
Wide range of connectors to integrate with various services.
Monitoring and error handling capabilities.

Cons:

Some advanced features may require custom coding.
Higher complexity for complex workflows.

- b) To cater to the requirement of having a single base URL for both the Student Profile and Class Timetable services in Azure, one approach is to use Azure API Management.

Azure API Management is a fully managed service that enables organizations to publish, secure, and manage APIs. It acts as a gateway between the external web content providers and the backend services, providing a unified entry point for accessing the services. Here's how the approach would work:

- Develop the Student Profile and Class Timetable services as separate backend services, hosted on Azure App Service or any other suitable Azure resource.
- Configure Azure API Management to create an API facade for both services. This involves defining API endpoints, operations, and policies for authentication, authorization, and transformation if needed.
- Set up a custom domain for Azure API Management, such as `api.university.com`, which will serve as the base URL for both the Student Profile and Class Timetable services.

- Configure API Management to route incoming requests based on the request path. For example, requests to /student-profile would be routed to the Student Profile service, and requests to /class-timetable would be routed to the Class Timetable service.

By utilizing Azure API Management, you achieve the following benefits:

- **Single Base URL:** Azure API Management allows you to define a custom domain and route requests based on the request path. This provides a single entry point for both the Student Profile and Class Timetable services, with a unified base URL (api.university.com).
- **API Management Capabilities:** Azure API Management offers various capabilities like authentication, authorization, caching, rate limiting, and analytics. These can be leveraged to secure and manage access to the services and monitor their usage.
- **Scalability and Load Balancing:** Azure API Management can scale to handle high traffic loads and provides load balancing capabilities, ensuring that the services remain available and responsive.
- **Developer Portal:** Azure API Management includes a developer portal that allows external web content providers to discover and consume the APIs. It provides documentation, code samples, and interactive testing capabilities.

C) 1 -Azure Active Directory (Azure AD):

Azure AD is a comprehensive identity and access management service provided by Azure. It integrates with various Azure services and offers robust authentication capabilities. Here's why Azure AD is a suitable choice:

- Single Sign-On (SSO):** Azure AD enables SSO, allowing users to authenticate once and access multiple applications and services without re-entering their credentials. This enhances the user experience and reduces the need for multiple logins.
- Enterprise-Grade Security:** Azure AD provides advanced security features such as multi-factor authentication (MFA), conditional access policies, risk-based authentication, and integration with Azure Security Center. These features help protect user accounts and sensitive data from unauthorized access.
- Integration with Azure Services:** Azure AD seamlessly integrates with other Azure services, such as Azure App Service and Azure API Management. This integration simplifies the implementation of authentication and authorization for the Student Profile and Class Timetable services.
- Compliance and Auditing:** Azure AD supports compliance standards such as GDPR, HIPAA, and ISO 27001. It also provides auditing capabilities, allowing organizations to monitor and track user activities.

2 - OAuth 2.0 and OpenID Connect:

OAuth 2.0 and OpenID Connect are widely adopted industry standards for authentication and authorization. They provide a secure and standardized approach for web authentication. Here's why they are suitable choices:

- a. **Widely Supported:** OAuth 2.0 and OpenID Connect are supported by numerous libraries, frameworks, and platforms, making it easier to integrate authentication into the university's web services.
- b. **Scalability and Performance:** OAuth 2.0 and OpenID Connect utilize token-based authentication, which reduces the need for repeated credential verification and improves scalability and performance.
- c. **Secure Authorization Framework:** OAuth 2.0 provides a flexible authorization framework for granting limited access to external web content providers. It allows fine-grained control over which data the providers can access and for how long.
- d. **Interoperability:** OAuth 2.0 and OpenID Connect are interoperable with other identity providers and can be used in hybrid scenarios where authentication is required across different systems.

2022 Question 2

This question is based on the Augmented Reality and Reality Mixing.

- a) Explain how Augmented Reality can be applied in the education domain. Elaborate your idea using some examples.
- b) Explain the concept of virtual Reality and apply it with real-world examples to elaborate your idea.

Answer

a) Augmented Reality (AR) can be a valuable tool in the education domain by enhancing the learning experience and providing interactive and immersive content. Here are a few examples of how AR can be applied in education:

Virtual Experiments: AR can simulate laboratory experiments, allowing students to perform practical tasks and observe the results in a virtual environment. For instance, chemistry students can mix different substances and observe the reactions, or biology students can dissect virtual organisms.

Interactive Learning Materials: AR can bring textbooks and learning materials to life. Students can use their smartphones or tablets to scan specific images or QR codes in textbooks, which would trigger related 3D models, videos, or animations to appear on their screens, providing a deeper understanding of the subject matter.

Historical and Cultural Exploration: AR can enable students to explore historical sites and cultural artifacts virtually. By pointing their devices at specific locations or objects, students can access additional information, virtual reconstructions, or historical narratives, allowing them to visualize and better comprehend the past.

Language Learning: AR can enhance language learning by overlaying virtual translations or subtitles onto real-world objects. Students can point their devices at objects or signs and see the translated words or phrases displayed in real-time, helping them improve their vocabulary and language comprehension.

Virtual Field Trips: AR can provide students with virtual field trip experiences, even if they can't physically visit certain locations. For example, students studying geology can explore virtual landscapes, interact with geological formations, and learn about different rock types and their formations.

b) Virtual Reality (VR) is a technology that creates a simulated environment, completely separate from the real world. Users wear VR headsets or goggles that immerse them in a computer-generated 3D environment. Here are a few examples of how VR can be applied with real-world examples:

Training Simulations: VR can be used for training purposes in various fields. For instance, airline pilots can practice flying in virtual cockpits, medical students can perform virtual surgeries to develop their skills, and firefighters can simulate emergency situations to improve their response techniques.

Architectural Design and Visualization: VR can help architects and designers visualize their projects in a realistic manner. By creating virtual walkthroughs, clients can experience the space and make informed decisions about the design elements before construction begins.

Gaming and Entertainment: VR has gained popularity in the gaming industry, providing players with immersive and interactive experiences. Players can feel as if they are inside the game, interacting with the virtual environment and characters, enhancing the overall gaming experience.

Therapy and Rehabilitation: VR is being used in therapeutic settings for various purposes. For example, it can help individuals with phobias face their fears in a controlled environment, aid in pain management by providing immersive distractions, or assist in physical rehabilitation by creating virtual exercises and challenges.

Virtual Collaboration: VR can enable remote collaboration by creating virtual meeting spaces where individuals can interact with each other in a more immersive and engaging way. This can be particularly useful for teams working on complex projects that require visualizations and spatial understanding.

