MCALE323 Distributed System and Cloud Computing Lab
Index

Sr. No.	Topic Name	Date	CO	Sign
1.	**Remote Process Communication:** 1. To develop a program for multi-client chat server using Socket 2. To implement a Server calculator using RPC concept. (Make use of datagram) 3. To implement a Date Time Server using RPC concept. (Make use of datagram) 4. To implement a Server calculator using RPC concept. (Make use of Server Socket) 5. To implement a Date Time Server using RPC concept. (Make use of Server Socket)		**CO1**	
2.	**Remote Procedure Call:** a) Implement a Server calculator containing ADD (), MUL (), SUB () etc. b) Implement a Date Time Server containing date() and time() c) Implement a Server to do the following 1. User will enter two values in hours and minutes from the client (Eg: 3 hour 40 minutes and 2 hour 50 minutes) 2. Server will add hour and minute components separately. The client side receives the result and display the value ❖ only in hours(Eg: 06:30 Hours) ❖ only in minutes(Eg:390 Minutes) d) Implement a Server to do the following: ❖ Get two numbers from the client. ❖ Server processing the multiplication of the above two numbers ❖ Server sends the processed data to the client and client checks whether the multiplication is greater than 250 or not e) Implement a Server to do the following ❖ Get a string from the client. ❖ Find the reverse of the string at the server side ❖ The client side receives the reversed string and verifying whether it is palindrome f) Implement a Server to do the following ❖ Get one number from the client. ❖ Server will find whether the number from client is odd or even ❖ After receiving the result from server client displays the message ❖ If the number is even client will calculate the table of the number		**CO1** **CO2**	
3.	**Remote Method Invocation** 1. Calculate addition of two numbers and send it to the client using **RMI**. 2. Retrieve time and date function from server to client. This program should display server date and time by implementing **RMI** **Equation Solver:** 3. The client should provide the values of a and b. The server will solve the equation $(a+b)^2=a^2+2ab+b^2$ and will give back the value of equation using **RMI**.		**CO1** **CO2**	

	4. The client should provide the values of a and b. The server will solve the equation $(a+b)^2=a^2+2ab+b^2$ and $(a-b)^2=a^2+2ab+b^2$ and will give back the value of equation (Use **RMI**)			
4.	**Remote Method Invocation (Graphical User Interface)** 1. Design a Graphical User Interface for addition of two numbers. Implement using RMI 2. Design a Graphical User Interface (GUI) to find factorial of a given numbers. Implement using RMI 3. Design a Graphical User Interface (GUI) based Basic calculator by implementing RMI 4. Design a Graphical User Interface (GUI) to find greatest of two numbers. Implement using RMI 5. Design a Graphical User Interface (GUI) which accepts a numerical value from the client. Convert the number in to words .Implement using RMI		**CO1** **CO2**	
5	**JDBC and RMI** Using MySQL create a Library database. Create table Book (Book_id, Book_name, Book_author) and retrieve the Book information from the Library database using the Remote Object Communication concept.		**CO2**	
6	Using MySQL create a Student database. Create table student_data(ID ,NAME , BRANCH ,PERCENTAGE ,EMAIL) and retrieve the student_data information from Student database using Remote Object Communication concept.		**CO2**	
7	Using MySQL create ElecrticBill database. Create table Bill(consumer_name, bill_due_date, bill_amount) and retrieve the Bill information from the ElecrticBill database using Remote Object Communication concept.		**CO2**	
8	Implementation of mutual exclusion using Token ring algorithm		**CO3**	
9	Implementation of Election Algorithm using Ring Algorithm.		**CO3**	
10	**Implementation of Storage as a Service** 1. Implementation of Storage as a Service (STaaS) using Amazon S3 (Simple Storage Service) **2.** Implementation of Storage as a Service using Azure Blob Storage		**CO4**	
11	**Implementation of Identity Management.** 1. Implementation of Identity and Access Management (IAM) in **AWS** 2. Implementation of Identity and Access Management (IAM) in **Microsoft Azure**		**CO5**	
12	**Create a virtual machine (VM) on any cloud provider (AWS/Azure/GCP) of your choice with the specifications:** ❖ Operating System, VM Type, Disk Size, Public IP, Network Rules ❖ **Once created, verify that the VM is running and submit a screenshot of the instance details and a brief description of the steps you followed.** • Create a virtual machine (VM) on any cloud provider (AWS) • Create a virtual machine (VM) on any cloud provider (Azure)		**CO6**	
13	Group projects (2 to 3 members) are to be given the opportunity to work on any Cloud Concept. ❖ Attach synopsis			

PRACTICAL NO. 1

Remote Process Communication:

1] To develop a program for multi-client chat server.

Solution:

ChatServer.Java:

```
package chat;

import java.io.*;
import java.net.*;
import java.util.*;

public class ChatServer {
    private static final int PORT = 12345;
    private static final String NAME = "Omkar Yelve C24129";
    private static final Set<ClientHandler> clients =
            Collections.synchronizedSet(new HashSet<>());

    public static void main(String[] args) {
        System.out.println("Chat Server started on port " + PORT);
        System.out.println(NAME);

        try (ServerSocket serverSocket = new ServerSocket(PORT)) {
            while (true) {
                Socket socket = serverSocket.accept();
                System.out.println("New client connected: " +
socket.getInetAddress());

                ClientHandler client = new ClientHandler(socket);
                clients.add(client);
                new Thread(client).start();
            }
        } catch (IOException e) {
            System.err.println("Server error: " + e.getMessage());
        }
    }

    public static void broadcast(String message, ClientHandler sender) {
        synchronized (clients) {
            for (ClientHandler client : clients) {
                if (client != sender) {
                    client.sendMessage(message);
                }
            }
        }
    }

    public static void removeClient(ClientHandler client) {
        clients.remove(client);
    }
```

```
}
```

ClientHandler.Java:

```
package chat;

import java.io.*;
import java.net.*;

public class ClientHandler implements Runnable {
    private Socket socket;
    private BufferedReader in;
    private PrintWriter out;
    private static final String NAME = "Omkar Yelve C24129";

    public ClientHandler(Socket socket) {
        this.socket = socket;
        try {
            in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            out = new PrintWriter(socket.getOutputStream(), true);
        } catch (IOException e) {
            System.err.println("Error initializing client handler: " +
e.getMessage());
        }
    }

    @Override
    public void run() {
        try {
            out.println("Welcome to the chat! Type 'exit' to leave.");
            out.println("Server: " + NAME);

            String message;
            while ((message = in.readLine()) != null) {
                if ("exit".equalsIgnoreCase(message)) {
                    break;
                }

                System.out.println("Received: " + message);
                ChatServer.broadcast("Client " + socket.getPort() + ": " +
message, this);
            }

        } catch (IOException e) {
            System.err.println("Client error: " + e.getMessage());
        } finally {
            try {
                socket.close();
                ChatServer.removeClient(this);
                System.out.println("Client disconnected: " + socket.getPort());
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    public void sendMessage(String message) {
        out.println(message);
```

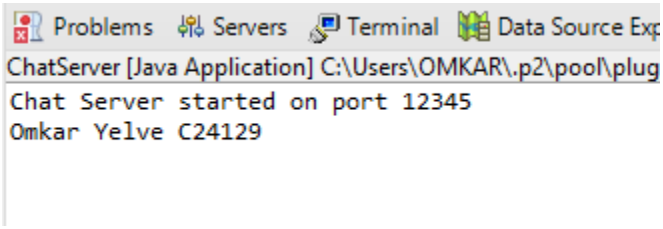
```
    }
}
```

ChatClient.Java:

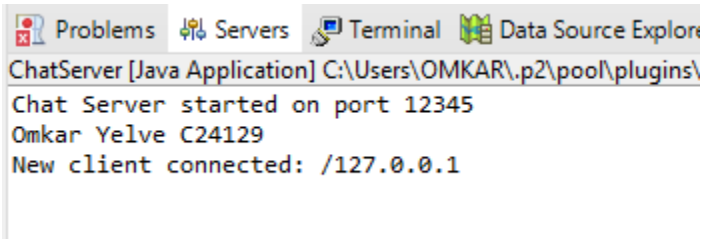
```
package chat;

import java.io.*;
import java.net.*;

public class ChatClient {
    public static void main(String[] args) {
        String hostname = "localhost";
        int port = 12345;
        final String NAME = "Omkar Yelve C24129";
        System.out.println(NAME);

        try (
            Socket socket = new Socket(hostname, port);
            BufferedReader userInput = new BufferedReader(new
InputStreamReader(System.in));
            BufferedReader serverIn = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            PrintWriter serverOut = new PrintWriter(socket.getOutputStream(),
true)
        ) {
            System.out.println("Connected to the server!");

            // Thread to receive messages from the server
            new Thread(() -> {
                try {
                    String msg;
                    while ((msg = serverIn.readLine()) != null) {
                        System.out.println(msg);
                    }
                } catch (IOException e) {
                    System.out.println("Disconnected from server.");
                }
            }).start();

            // Main thread to send messages
            String input;
            while ((input = userInput.readLine()) != null) {
                serverOut.println(input);
                if ("exit".equalsIgnoreCase(input)) {
                    break;
                }
            }

            System.out.println("Client terminated.");

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

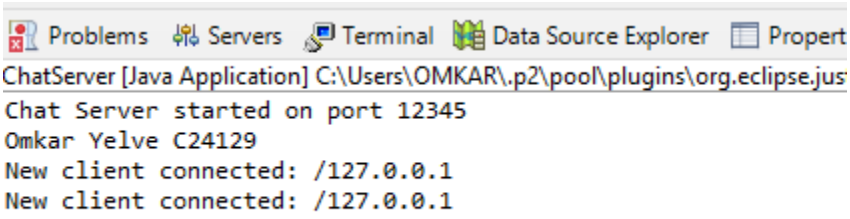
Output:

Server Output:



Client Output:





1.2] To implement a Server calculator using RPC concept. (Make use of datagram)

Solution:

RPCServer.java

```
package RPC;

import java.net.*;


public class RPCServer {
    public static void main(String[] args) {
        DatagramSocket socket = null;

        try {
            socket = new DatagramSocket(5000);
            byte[] buffer = new byte[1024];

            System.out.println("RPC Server started on port 5000...");

            while (true) {
                DatagramPacket request = new DatagramPacket(buffer,
buffer.length);
                socket.receive(request);

                String input = new String(request.getData(), 0,
request.getLength());
                System.out.println("Received request: " + input);

                // Format: operation:num1:num2
                String[] parts = input.split(":");
                String operation = parts[0];
                double num1 = Double.parseDouble(parts[1]);
                double num2 = Double.parseDouble(parts[2]);

                double result = 0;
                switch (operation.toLowerCase()) {
                    case "add": result = num1 + num2; break;
                    case "sub": result = num1 - num2; break;
                    case "mul": result = num1 * num2; break;
                    case "div":
                        result = (num2 != 0) ? (num1 / num2) : Double.NaN;
                        break;
                    default:
                        System.out.println("Invalid operation.");
                }

                String response = "Result = " + result;
                byte[] sendData = response.getBytes();

                InetAddress clientAddress = request.getAddress();
                int clientPort = request.getPort();
```

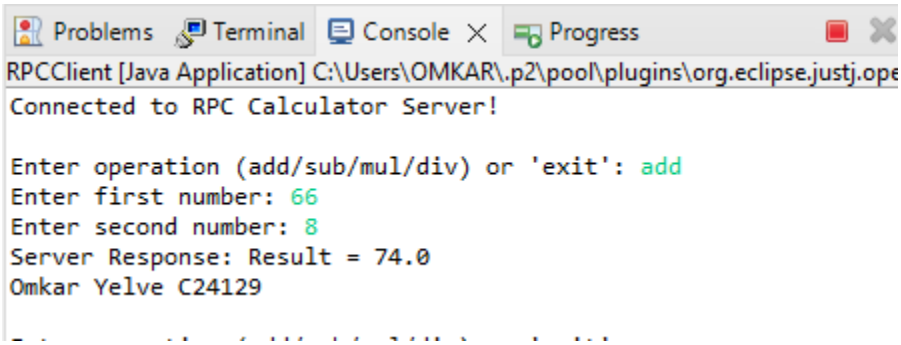
```
                DatagramPacket reply = new DatagramPacket(sendData,
sendData.length, clientAddress, clientPort);
                socket.send(reply);

                System.out.println("Sent: " + response);
            }

        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if (socket != null) socket.close();
        }
    }
}
```

RPCClient.java:

```
package RPC;


import java.net.*;

import java.util.*;


public class RPCClient {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);


        try {

            DatagramSocket socket = new DatagramSocket();

            InetAddress serverAddress = InetAddress.getByName("127.0.0.1");


            System.out.println("Connected to RPC Calculator Server!");


            while (true) {

                System.out.print("\nEnter operation (add/sub/mul/div) or 'exit':
");

                String operation = sc.next();


                if (operation.equalsIgnoreCase("exit")) {

                    System.out.println("Client exiting...");

                    break;

                }
```

```
                System.out.print("Enter first number: ");
                double num1 = sc.nextDouble();

                System.out.print("Enter second number: ");
                double num2 = sc.nextDouble();

                String message = operation + ":" + num1 + ":" + num2;
                byte[] sendData = message.getBytes();

                DatagramPacket request = new DatagramPacket(sendData,
sendData.length, serverAddress, 5000);
                socket.send(request);

                byte[] buffer = new byte[1024];
                DatagramPacket reply = new DatagramPacket(buffer, buffer.length);
                socket.receive(reply);

                String response = new String(reply.getData(), 0,
reply.getLength());
                System.out.println("Server Response: " + response);
                System.out.println("Omkar Yelve C24129");
            }

            socket.close();
            sc.close();

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Output :



1.3] To implement a Date Time Server using RPC concept. (Make use of datagram)

Solution:

DateTimeServer.java:

```
package RPC;


import java.net.*;
import java.text.SimpleDateFormat;
import java.util.Date;


public class DateTimeServer {
    public static void main(String[] args) {
        DatagramSocket socket = null;
        try {
            socket = new DatagramSocket(5000);
            byte[] buffer = new byte[1024];
            System.out.println("Date-Time RPC Server started on port 5000...");


            while (true) {
                DatagramPacket request = new DatagramPacket(buffer,
buffer.length);
                socket.receive(request);
```

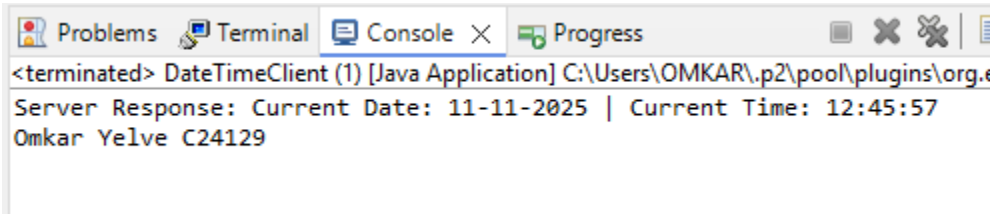
```
                // Create date and time string
                SimpleDateFormat dateFormat = new SimpleDateFormat("dd-MM-yyyy");
                SimpleDateFormat timeFormat = new SimpleDateFormat("HH:mm:ss");
                String currentDate = dateFormat.format(new Date());
                String currentTime = timeFormat.format(new Date());

                String response = "Current Date: " + currentDate + " | Current
Time: " + currentTime;

                byte[] sendData = response.getBytes();
                InetAddress clientAddress = request.getAddress();
                int clientPort = request.getPort();

                DatagramPacket reply = new DatagramPacket(sendData,
sendData.length, clientAddress, clientPort);
                socket.send(reply);

                System.out.println("Sent: " + response);
            }

        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if (socket != null) socket.close();
        }
    }
}
```

DateTimeClient.java:

```
package RPC;

import java.net.*;

public class DateTimeClient {
    public static void main(String[] args) {
        DatagramSocket socket = null;
        try {
            socket = new DatagramSocket();
```

```
            InetAddress serverAddress = InetAddress.getByName("127.0.0.1");

            String requestMessage = "GET_DATE_TIME";
            byte[] sendData = requestMessage.getBytes();

            DatagramPacket request = new DatagramPacket(sendData, sendData.length,
serverAddress, 5000);
            socket.send(request);

            byte[] buffer = new byte[1024];
            DatagramPacket reply = new DatagramPacket(buffer, buffer.length);
            socket.receive(reply);

            String response = new String(reply.getData(), 0, reply.getLength());
            System.out.println("Server Response: " + response);
            System.out.println("Omkar Yelve C24129");

        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if (socket != null) socket.close();
        }
    }
}
```

Output:



1.4] To implement a Server calculator using RPC concept. (Make use of Server Socket)

Solution:

RPCServerTCP.java:

```
package RPC;

import java.io.*;

import java.net.*;


public class RPCServerTCP {

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

```
        try (ServerSocket serverSocket = new ServerSocket(5000)) {
            System.out.println("RPC Calculator Server started on port 5000...");

            while (true) {
                Socket clientSocket = serverSocket.accept();
                System.out.println("Client connected: " + clientSocket);

                BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
                PrintWriter out = new PrintWriter(clientSocket.getOutputStream(),
true);

                // Read client request
                String request = in.readLine();
                System.out.println("Received request: " + request);

                String[] parts = request.split(" ");
                if (parts.length != 3) {
                    out.println("Invalid format! Use: <operation> <num1> <num2>");
                    clientSocket.close();
                    continue;
                }

                String operation = parts[0];
                double num1 = Double.parseDouble(parts[1]);
                double num2 = Double.parseDouble(parts[2]);
                double result = 0;

                switch (operation.toLowerCase()) {
                    case "add":
                        result = num1 + num2;
                        break;
                    case "sub":
                        result = num1 - num2;
                        break;
```

```
                    case "mul":
                        result = num1 * num2;
                        break;
                    case "div":
                        if (num2 == 0)
                            out.println("Error: Division by zero!");
                        else
                            result = num1 / num2;
                        break;
                    default:
                        out.println("Invalid operation!");
                        continue;
                }

                // Send result
                out.println("Result = " + result);

                System.out.println("Sent: Result = " + result);
                clientSocket.close();
            }

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

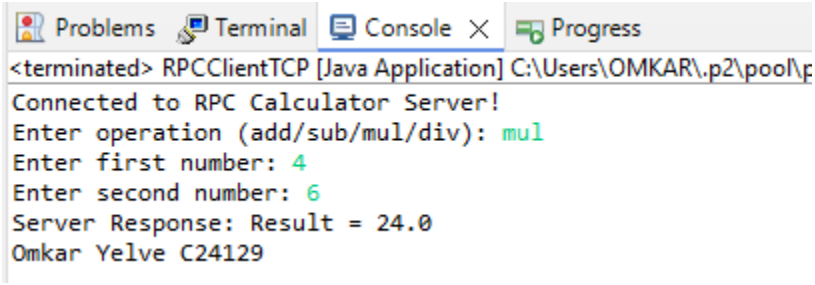
RPCClientTCP.java:

```
package RPC;
import java.io.*;
import java.net.*;
import java.util.*;

public class RPCClientTCP {
```

```
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        try (Socket socket = new Socket("127.0.0.1", 5000)) {
            System.out.println("Connected to RPC Calculator Server!");

            BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

            System.out.print("Enter operation (add/sub/mul/div): ");
            String operation = sc.next();

            System.out.print("Enter first number: ");
            double num1 = sc.nextDouble();

            System.out.print("Enter second number: ");
            double num2 = sc.nextDouble();

            // Send request to server
            String request = operation + " " + num1 + " " + num2;
            out.println(request);

            // Receive and print response
            String response = in.readLine();
            System.out.println("Server Response: " + response);
            System.out.println("Omkar Yelve C24129");

        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            sc.close();
        }
    }
```

```
}
```

Output:



1.5] To implement a Date Time Server using RPC concept. (Make use of Server Socket)

Solution:

DateTimeServerTCP.java:

```
package RPC;

import java.io.*;

import java.net.*;

import java.util.*;


public class DateTimeServerTCP {

    public static void main(String[] args) {

        try (ServerSocket serverSocket = new ServerSocket(6000)) {

            System.out.println(" Date-Time Server started on port 6000...");


            while (true) {

                Socket clientSocket = serverSocket.accept();

                System.out.println("Client connected: " + clientSocket);


                PrintWriter out = new PrintWriter(clientSocket.getOutputStream(),
true);


                // Get current date and time
```

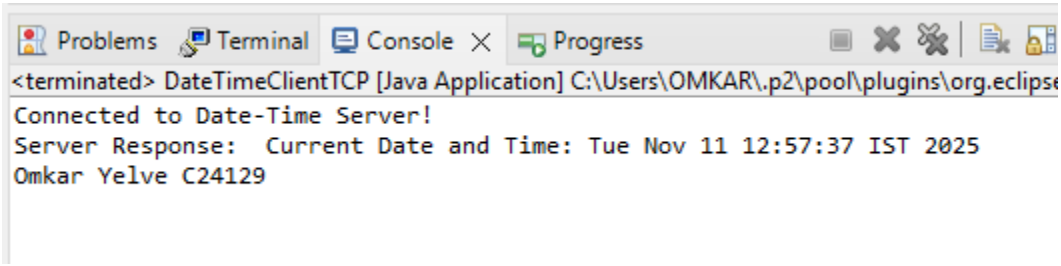
```
                Date currentDate = new Date();
                out.println(" Current Date and Time: " + currentDate.toString());

                System.out.println("Sent to client: " + currentDate);

                clientSocket.close();
            }

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

DateTimeClientTCP.java:

```
package RPC;

import java.io.*;
import java.net.*;

public class DateTimeClientTCP {
    public static void main(String[] args) {
        try (Socket socket = new Socket("127.0.0.1", 6000)) {
            System.out.println("Connected to Date-Time Server!");

            BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            String serverResponse = in.readLine();

            System.out.println("Server Response: " + serverResponse);
            System.out.println("Omkar Yelve C24129");

        } catch (IOException e) {
            e.printStackTrace();
        }
```

```
}
```

Output:


Problems
Terminal
Console
Progress
<terminated> DateTimeClientTCP [Java Application] C:\Users\OMKAR\.p2\pool\plugins\org.eclipse
Connected to Date-Time Server!
Server Response: Current Date and Time: Tue Nov 11 12:57:37 IST 2025
Omkar Yelve C24129

PRACTICAL NO. 2

Remote Procedure Communication

a] Implement a Server calculator containing ADD (), MUL (), SUB () etc.

Solution:

CalculatorServer.Java:

```
package rpc;

import java.io.*;
import java.net.*;

public class CalculatorServer {
    public static void main(String[] args) {
        final int PORT = 5000; // change if needed
        try (ServerSocket serverSocket = new ServerSocket(PORT)) {
            System.out.println("Calculator Server started on port " + PORT + ".
Waiting for client connection...");
            System.out.println("Omkar Yelve C24129");

            while (true) {
                Socket clientSocket = serverSocket.accept();
                System.out.println("Client connected: " +
clientSocket.getInetAddress());

                try (
                    BufferedReader input = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
                    PrintWriter output = new
PrintWriter(clientSocket.getOutputStream(), true) // autoFlush
                ) {
                    String clientMessage = input.readLine();
                    if (clientMessage == null) {
                        System.out.println("Client disconnected without sending a
request.");
                        clientSocket.close();
                        continue;
                    }

                    System.out.println("Client request: " + clientMessage);
                    String[] parts = clientMessage.trim().split("\\s+");
                    if (parts.length != 3) {
                        output.println("Error: Invalid format. Use: <operation>
<num1> <num2>");
                        continue;
                    }

                    String operation = parts[0].toLowerCase();
                    double a, b, result = 0;

                    try {
                        a = Double.parseDouble(parts[1]);
                        b = Double.parseDouble(parts[2]);
                    } catch (NumberFormatException e) {
```

```
                        output.println("Error: Invalid numbers. Provide numeric
operands.");
                        continue;
                    }

                    switch (operation) {
                        case "add":
                            result = a + b;
                            break;
                        case "sub":
                            result = a - b;
                            break;
                        case "mul":
                            result = a * b;
                            break;
                        case "div":
                            if (b == 0) {
                                output.println("Error: Division by zero");
                                continue;
                            }
                            result = a / b;
                            break;
                        default:
                            output.println("Error: Invalid operation. Use add,
sub, mul, div.");
                            continue;
                    }

                    System.out.println("Sending result: " + result);
                    output.println("Result: " + result);
                    // autoFlush true already ensures delivery
                } catch (IOException e) {
                    System.err.println("I/O error while handling client: " +
e.getMessage());
                } finally {
                    if (!clientSocket.isClosed()) {
                        try { clientSocket.close(); } catch (IOException ignored)
{}
                    }
                }
            }
        } catch (BindException be) {
            System.err.println("Port " + 5000 + " is already in use. Use another
port or stop the process using it.");
            be.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

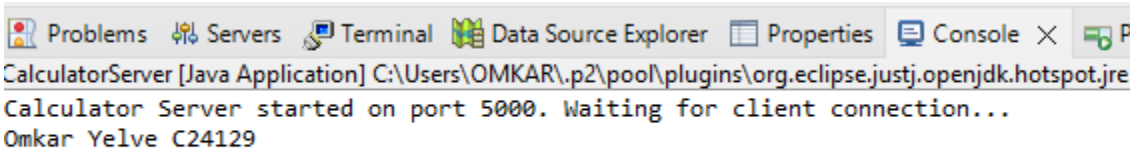
CalculatorClient.Java:

```
package rpc;

import java.io.*;
```

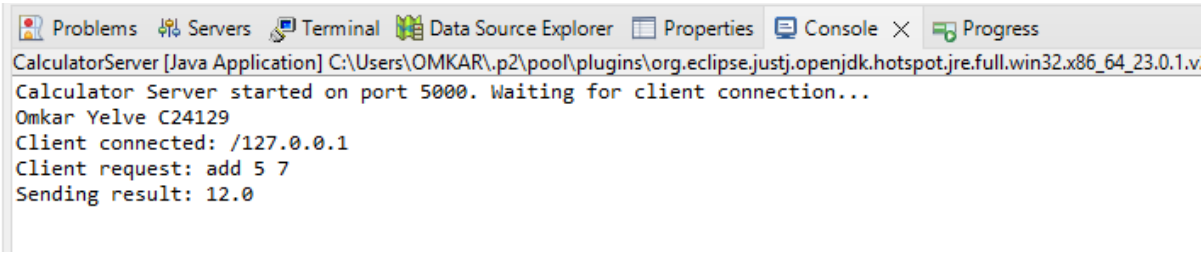
```java
import java.net.*;

public class CalculatorClient {
    public static void main(String[] args) {
        final String HOST = "localhost";
        final int PORT = 5000; // must match server

        try (Socket socket = new Socket(HOST, PORT);
             BufferedReader userInput = new BufferedReader(new
InputStreamReader(System.in));
             BufferedReader serverInput = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
             PrintWriter serverOutput = new PrintWriter(socket.getOutputStream(),
true) // autoFlush
        ) {
            System.out.println("Connected to Calculator Server on " + HOST + ":" +
PORT);
            System.out.println("Enter operation (add, sub, mul, div) followed by
two numbers (e.g. add 10 5):");
            String userMessage = userInput.readLine();

            if (userMessage == null || userMessage.trim().isEmpty()) {
                System.out.println("No input provided. Exiting.");
                return;
            }

            serverOutput.println(userMessage.trim());

            String serverResponse = serverInput.readLine(); // waits for server
response
            System.out.println("Server Response: " + serverResponse);
            System.out.println("Omkar Yelve C24129");
        } catch (ConnectException ce) {
            System.err.println("Connection refused — make sure the server is
running first.");
            ce.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output:

Server output:



Client Output:



b] Implement a Date Time Server containing date() and time().

Solution:

DateTimeServer.Java:

```
package rpc;

import java.io.*;
import java.net.*;
import java.text.*;
import java.util.*;

public class DateTimeServer {
    public static void main(String[] args) {
        final int PORT = 5000; // change to 12345 if you must
        try (ServerSocket serverSocket = new ServerSocket(PORT)) {
            System.out.println("DateTime Server started on port " + PORT + ".
Waiting for client connection...");
            System.out.println("Omkar Yelve C24129");

            while (true) {
                Socket clientSocket = serverSocket.accept();
                System.out.println("Client connected: " +
clientSocket.getInetAddress());

                try (
                    BufferedReader input = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
                    PrintWriter output = new
PrintWriter(clientSocket.getOutputStream(), true) // autoFlush = true
                ) {
                    String clientRequest = input.readLine();
                    if (clientRequest == null) {
                        System.out.println("Client disconnected without sending a
request.");
                        clientSocket.close();
                        continue;
                    }

                    clientRequest = clientRequest.trim();
                    System.out.println("Client request: " + clientRequest);

                    String response;
                    if ("date".equalsIgnoreCase(clientRequest)) {
                        DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-
dd");
                        response = "Current Date: " + dateFormat.format(new
Date());
                    } else if ("time".equalsIgnoreCase(clientRequest)) {
                        DateFormat timeFormat = new SimpleDateFormat("HH:mm:ss");
                        response = "Current Time: " + timeFormat.format(new
Date());
                    } else {
                        response = "Error: Invalid request. Please send 'date' or
'time'.";
                    }
```

```
                    System.out.println("Sending response to client: " + response);
                    output.println(response); // autoFlush true ensures it's sent
                } catch (IOException e) {
                    System.err.println("I/O error while handling client: " +
e.getMessage());
                } finally {
                    // clientSocket closed by try-with-resources for streams, but
ensure closed:
                    if (!clientSocket.isClosed()) {
                        try { clientSocket.close(); } catch (IOException ignored)
{}
                    }
                }
            }
        } catch (BindException be) {
            System.err.println("Port already in use. Change the PORT or stop the
process using it.");
            be.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

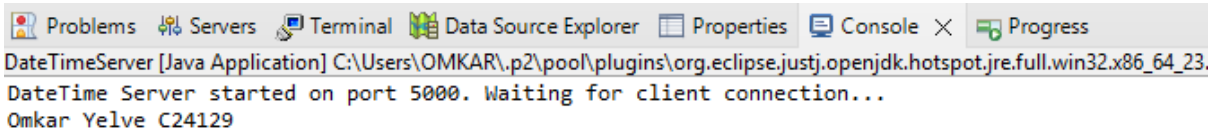
DateTimeClient.Java:

```
package rpc;

import java.io.*;
import java.net.*;

public class DateTimeClient {
    public static void main(String[] args) {
        final String HOST = "localhost";
        final int PORT = 5000; // must match server

        try (Socket socket = new Socket(HOST, PORT);
             BufferedReader userInput = new BufferedReader(new
InputStreamReader(System.in));
             BufferedReader serverInput = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
             PrintWriter serverOutput = new PrintWriter(socket.getOutputStream(),
true) // autoFlush = true
        ) {
            System.out.println("Connected to DateTime Server on " + HOST + ":" +
PORT);
            System.out.println("Enter 'date' to get the current date, or 'time' to
get the current time:");
            String userMessage = userInput.readLine();

            if (userMessage == null || userMessage.trim().isEmpty()) {
                System.out.println("No input provided. Exiting.");
                return;
```

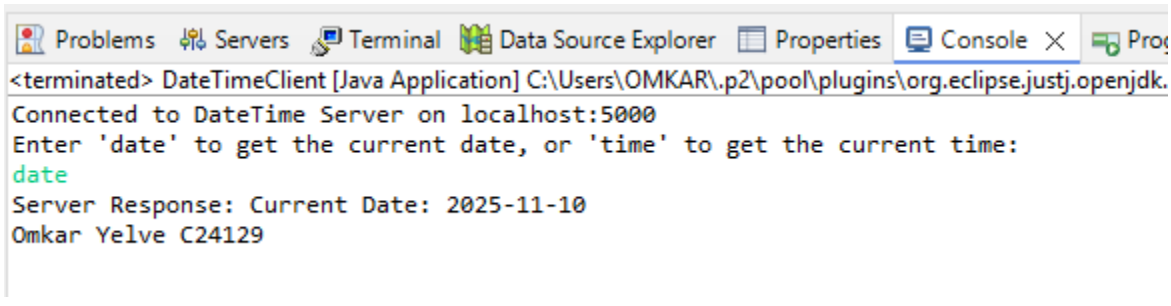
```
            }

            serverOutput.println(userMessage.trim()); // send request
            String serverResponse = serverInput.readLine(); // wait for response

            System.out.println("Server Response: " + serverResponse);
            System.out.println("Omkar Yelve C24129");
        } catch (ConnectException ce) {
            System.err.println("Connection refused. Make sure the server is
running first.");
            ce.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output:

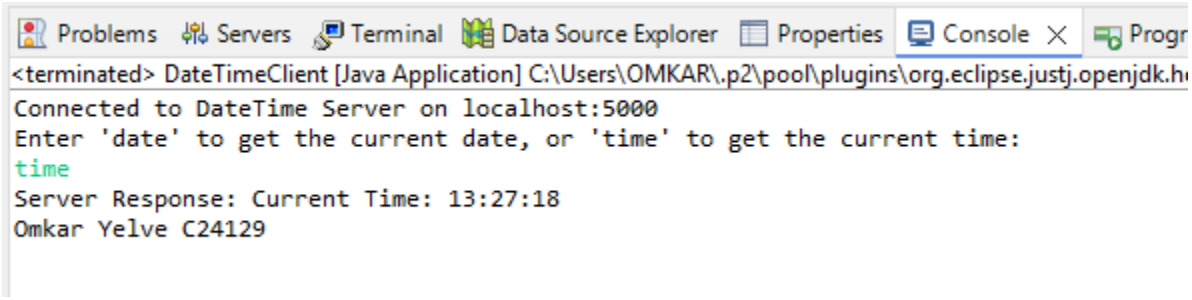
Server Output:



Client Output (date):



Client Output (Time):


Problems
Servers
Terminal
Data Source Explorer
Properties
Console
<terminated> DateTimeClient [Java Application] C:\Users\OMKAR\.p2\pool\plugins\org.eclipse.justj.openjdk.h
Connected to DateTime Server on localhost:5000
Enter 'date' to get the current date, or 'time' to get the current time:
time
Server Response: Current Time: 13:27:18
Omkar Yelve C24129

c] Implement a Server to do the following

1.User will enter two values in hours and minutes from the client (Eg: 3 hour 40 minutes and 2 hour 50 minutes)

2.Server will add hour and minute components separately The client side receives the result and display the value only in hours (Eg: 06:30 Hours) only in minutes(Eg:390 Minutes)

Solution:

TimeAdditionServer.Java:

```
package rpc;

import java.io.*;
import java.net.*;

public class TimeAdditionServer {
    public static void main(String[] args) {
        final int PORT = 5000; // change if needed
        try (ServerSocket serverSocket = new ServerSocket(PORT)) {
            System.out.println("Time Addition Server started on port " + PORT + ",
waiting for clients...");
            System.out.println("Omkar Yelve C24129");

            while (true) {
                Socket clientSocket = serverSocket.accept();
                System.out.println("Client connected: " +
clientSocket.getInetAddress());

                try (
                    BufferedReader input = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
                    PrintWriter output = new
PrintWriter(clientSocket.getOutputStream(), true) // autoFlush
                ) {
                    // Read two lines: each "hours minutes"
                    String firstTime = input.readLine();
                    String secondTime = input.readLine();

                    if (firstTime == null || secondTime == null) {
                        output.println("Error: Missing input. Send two lines:
'<hours> <minutes>' each.");
                        clientSocket.close();
                        continue;
                    }

                    System.out.println("Received times: '" + firstTime + "' and '"
+ secondTime + "'");

                    String[] t1 = firstTime.trim().split("\\s+");
                    String[] t2 = secondTime.trim().split("\\s+");
```

```
                    if (t1.length != 2 || t2.length != 2) {
                        output.println("Error: Invalid format. Use: '<hours>
<minutes>' on each line.");
                        clientSocket.close();
                        continue;
                    }

                    int h1, m1, h2, m2;
                    try {
                        h1 = Integer.parseInt(t1[0]);
                        m1 = Integer.parseInt(t1[1]);
                        h2 = Integer.parseInt(t2[0]);
                        m2 = Integer.parseInt(t2[1]);
                    } catch (NumberFormatException e) {
                        output.println("Error: Hours and minutes must be
integers.");
                        clientSocket.close();
                        continue;
                    }

                    // Normalize negative/minute values if you want to restrict,
here we assume non-negative
                    if (h1 < 0 || m1 < 0 || h2 < 0 || m2 < 0) {
                        output.println("Error: Hours and minutes must be non-
negative.");
                        clientSocket.close();
                        continue;
                    }

                    int totalHours = h1 + h2;
                    int totalMinutes = m1 + m2;

                    // carry minutes into hours
                    totalHours += totalMinutes / 60;
                    totalMinutes = totalMinutes % 60;

                    String hoursFormat = String.format("%02d:%02d Hours",
totalHours, totalMinutes);
                    int totalMinutesOnly = totalHours * 60 + totalMinutes;
                    String minutesFormat = totalMinutesOnly + " Minutes";

                    System.out.println("Sending -> " + hoursFormat + " and " +
minutesFormat);
                    output.println(hoursFormat);
                    output.println(minutesFormat);
                    // autoFlush=true ensures immediate send
                } catch (IOException e) {
                    System.err.println("I/O error handling client: " +
e.getMessage());
                } finally {
                    if (!clientSocket.isClosed()) {
                        try { clientSocket.close(); } catch (IOException ignored)
{}
                    }
                }
            }
        } catch (BindException be) {
            System.err.println("Port already in use. Change PORT or stop other
process.");
```

```
            be.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

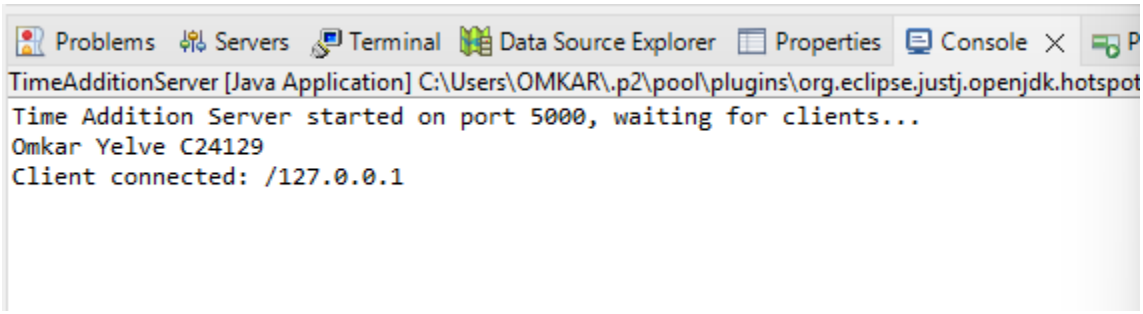
TimeAdditionClient.Java:

```
package rpc;

import java.io.*;
import java.net.*;

public class TimeAdditionClient {
    public static void main(String[] args) {
        final String HOST = "localhost";
        final int PORT = 5000; // must match server

        try (Socket socket = new Socket(HOST, PORT);
             BufferedReader userInput = new BufferedReader(new
InputStreamReader(System.in));
             BufferedReader serverInput = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
             PrintWriter serverOutput = new PrintWriter(socket.getOutputStream(),
true) // autoFlush
        ) {
            System.out.println("Connected to Time Addition Server at " + HOST +
":" + PORT);
            System.out.println("Enter first time (hours and minutes) separated by
space (e.g. '3 40'):");
            String firstTime = userInput.readLine();
            System.out.println("Enter second time (hours and minutes) separated by
space (e.g. '2 50'):");
            String secondTime = userInput.readLine();

            if (firstTime == null || firstTime.trim().isEmpty() || secondTime ==
null || secondTime.trim().isEmpty()) {
                System.out.println("Invalid input. Exiting.");
                return;
            }

            serverOutput.println(firstTime.trim());
            serverOutput.println(secondTime.trim());

            String hoursFormat = serverInput.readLine();   // e.g. "06:30 Hours"
            String minutesFormat = serverInput.readLine(); // e.g. "390 Minutes"

            System.out.println("Result:");
            System.out.println("i. " + (hoursFormat != null ? hoursFormat : "No
response for hours"));
            System.out.println("ii. " + (minutesFormat != null ? minutesFormat :
"No response for minutes"));
            System.out.println("Omkar Yelve C24129");
        } catch (ConnectException ce) {
```

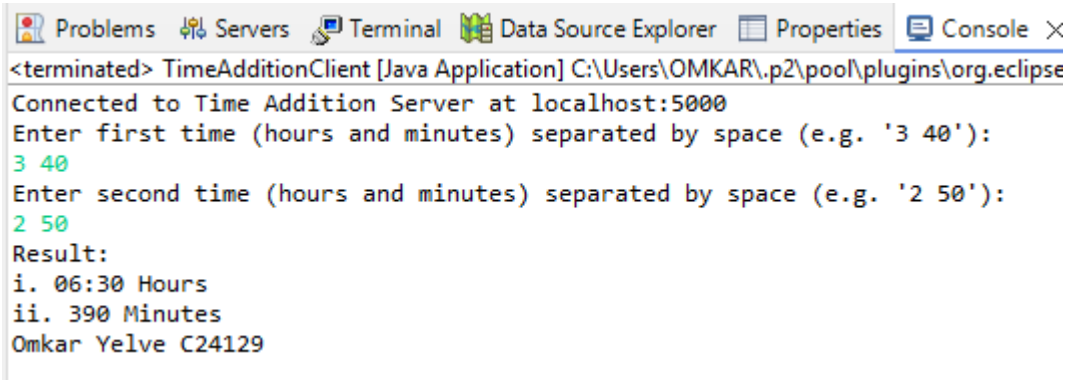
```
            System.err.println("Connection refused. Make sure the server is
running first.");
            ce.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

OUTPUT:

Server Output:



Client Output:



d] Implement a Server to do the following:

i. Get two numbers from the client.

ii. Server processing the multiplication of the above two numbers

Server sends the processed data to the client and client checks whether the multiplication is greater than 250 or not.

Solution:

MultiplicationServer.Java:

```
package rpc;

import java.io.*;
import java.net.*;

public class MultiplicationServer {
    public static void main(String[] args) {
        final int PORT = 5000; // change if needed
        try (ServerSocket serverSocket = new ServerSocket(PORT)) {
            System.out.println("Server is listening on port " + PORT);
            System.out.println("Omkar Yelve C24129");

            while (true) {
                Socket socket = serverSocket.accept();
                System.out.println("Client connected: " +
socket.getInetAddress());

                // handle client inside try-with-resources so streams are closed
automatically
                try (DataInputStream input = new
DataInputStream(socket.getInputStream());
                     DataOutputStream output = new
DataOutputStream(socket.getOutputStream())) {

                    int num1 = input.readInt();
                    int num2 = input.readInt();
                    System.out.println("Received numbers: " + num1 + " and " +
num2);

                    int result = num1 * num2;
                    output.writeInt(result);
                    output.flush();

                    System.out.println("Sent result: " + result);
                } catch (IOException e) {
                    System.err.println("I/O error handling client: " +
e.getMessage());
                } finally {
                    try { socket.close(); } catch (IOException ignored) {}
                }
            }
```

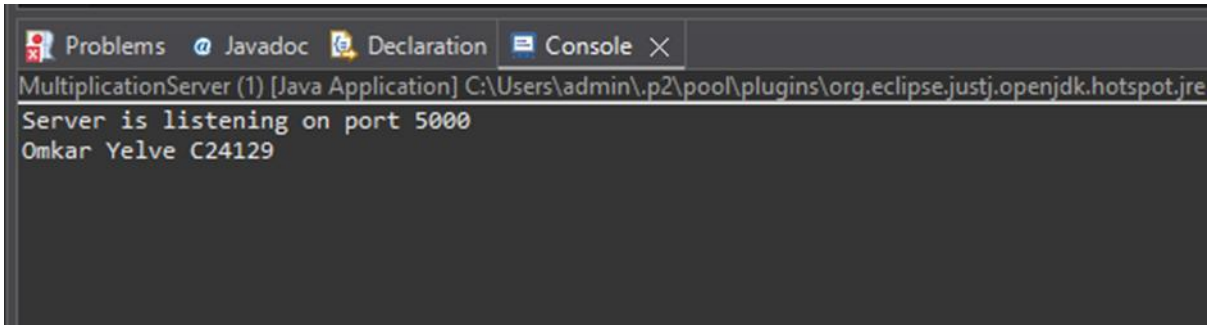
```
        } catch (BindException be) {
            System.err.println("Port " + PORT + " already in use. Change the port
or stop the process using it.");
            be.printStackTrace();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}
```

MultiplicationClient.Java:

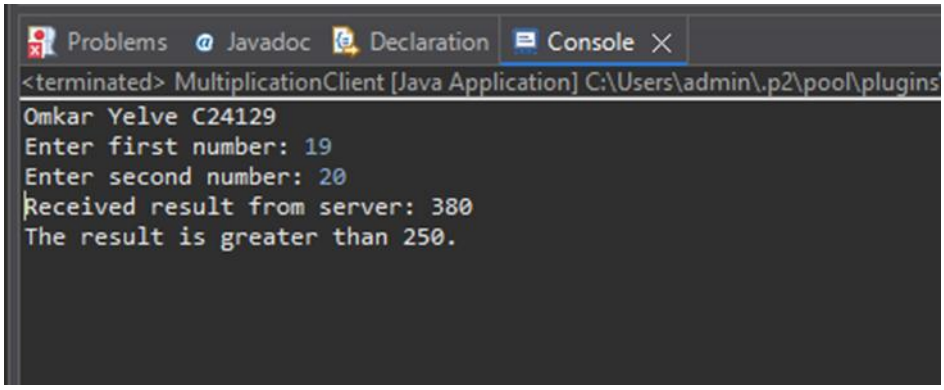
```
package rpc;

import java.io.*;
import java.net.*;
import java.util.Scanner;

public class MultiplicationClient {
    public static void main(String[] args) {
        final String HOST = "localhost";
        final int PORT = 5000; // must match server

        try (Socket socket = new Socket(HOST, PORT);
             DataOutputStream output = new
DataOutputStream(socket.getOutputStream());
             DataInputStream input = new DataInputStream(socket.getInputStream());
             Scanner scanner = new Scanner(System.in)) {

            System.out.println("Omkar Yelve C24129");
            System.out.print("Enter first number: ");
            int num1 = scanner.nextInt();
            System.out.print("Enter second number: ");
            int num2 = scanner.nextInt();

            output.writeInt(num1);
            output.writeInt(num2);
            output.flush();

            int result = input.readInt();
            System.out.println("Received result from server: " + result);

            // Problem statement says check > 250
            if (result > 250) {
                System.out.println("The result is greater than 250.");
            } else {
                System.out.println("The result is not greater than 250.");
            }
        } catch (ConnectException ce) {
            System.err.println("Connection refused. Make sure the server is
running first.");
            ce.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output:

Server Output:



Client Output:



e] Implement a Server to do the following

i. Get a string from the client.

ii. Find the reverse of the string at the server side

iii. The client side receives the reversed string and verifying whether it is palindrome.

Solution:

PalindromeServer.Java:

```
package rpc;

import java.io.*;
import java.net.*;

public class PalindromeServer {
    public static void main(String[] args) {
        final int PORT = 5000;
        try (ServerSocket serverSocket = new ServerSocket(PORT)) {
            System.out.println("Palindrome Server is listening on port " + PORT);
            System.out.println("Omkar Yelve C24129");

            while (true) {
                Socket socket = serverSocket.accept();
                System.out.println("Client connected: " +
socket.getInetAddress());

                // Use try-with-resources to ensure streams are closed
                try (BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
                     PrintWriter out = new PrintWriter(socket.getOutputStream(),
true)) {

                    String inputString = in.readLine();
                    if (inputString == null) {
                        System.out.println("Client disconnected without sending
data.");
                        continue;
                    }

                    inputString = inputString.trim();
                    System.out.println("Received from client: " + inputString);

                    String reversed = new
StringBuilder(inputString).reverse().toString();
                    System.out.println("Sending reversed string: " + reversed);

                    out.println(reversed); // autoFlush = true ensures immediate
send
                } catch (IOException e) {
                    System.err.println("I/O error while handling client: " +
e.getMessage());
```

```
                } finally {
                    try { socket.close(); } catch (IOException ignored) {}
                }
            }
        } catch (BindException be) {
            System.err.println("Port " + PORT + " already in use. Change the port
or stop the process using it.");
            be.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

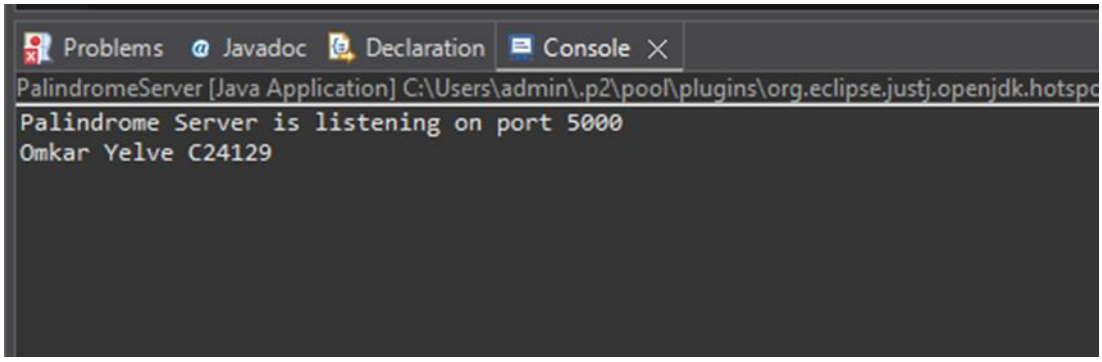
PalindromeClient.Java:

```
package rpc;

import java.io.*;
import java.net.*;
import java.util.Scanner;

public class PalindromeClient {
    public static void main(String[] args) {
        final String HOST = "localhost";
        final int PORT = 5000;

        try (Socket socket = new Socket(HOST, PORT);
             Scanner scanner = new Scanner(System.in);
             PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
             BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()))) {

            System.out.println("Connected to Palindrome Server at " + HOST + ":" +
PORT);
            System.out.print("Enter a string: ");
            String original = scanner.nextLine();

            if (original == null || original.trim().isEmpty()) {
                System.out.println("No input provided. Exiting.");
                return;
            }

            original = original.trim();
            out.println(original); // send to server

            String reversed = in.readLine();
            System.out.println("Reversed string from server: " + reversed);

            if (reversed != null && original.equalsIgnoreCase(reversed)) {
                System.out.println("The string is a palindrome.");
            } else {
                System.out.println("The string is not a palindrome.");
            }
```

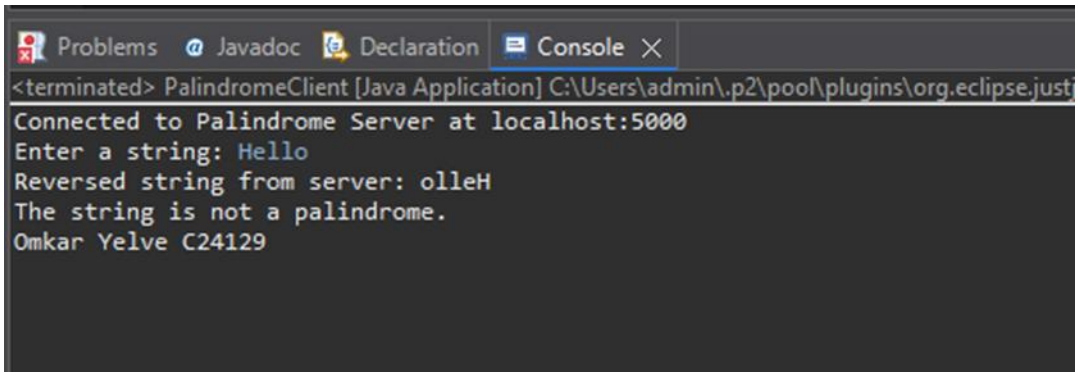
```
            System.out.println("Omkar Yelve C24129");
        } catch (ConnectException ce) {
            System.err.println("Connection refused — make sure the server is
running first.");
            ce.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output:

Server Output:



Client Output:



f] Implement a Server to do the following

i. Get one number from the client.

ii. Server will find whether the number from client is odd or even

iii. After receiving the result from server client displays the message

iv. If the number is even client will calculate the table of the number

solution:

EvenOddServer.Java;

```
package rpc;

import java.io.*;
import java.net.*;

public class EvenOddServer {
    public static void main(String[] args) {
        final int PORT = 5000; // change if needed
        try (ServerSocket serverSocket = new ServerSocket(PORT)) {
            System.out.println("Even/Odd Server is running on port " + PORT);
            System.out.println("Omkar Yelve C24129");

            while (true) {
                Socket clientSocket = serverSocket.accept();
                System.out.println("Client connected: " +
clientSocket.getInetAddress());

                try (
                    BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
                    PrintWriter out = new
PrintWriter(clientSocket.getOutputStream(), true) // autoFlush
                ) {
                    String line = in.readLine();
                    if (line == null) {
                        System.out.println("Client disconnected without sending
data.");
                        continue;
                    }

                    line = line.trim();
                    System.out.println("Received from client: '" + line + "'");

                    int number;
                    try {
                        number = Integer.parseInt(line);
                    } catch (NumberFormatException e) {
                        out.println("error:invalid_number");
                        System.out.println("Invalid number received.");
                        continue;
                    }
```

```
                    // Determine even/odd
                    String result = (number % 2 == 0) ? "even" : "odd";
                    out.println(result); // send result to client
                    System.out.println("Sent to client: " + result);

                } catch (IOException e) {
                    System.err.println("I/O error handling client: " +
e.getMessage());
                } finally {
                    try { clientSocket.close(); } catch (IOException ignored) {}
                }
            }
        } catch (BindException be) {
            System.err.println("Port " + 5000 + " is already in use. Change the
port or stop the process using it.");
            be.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

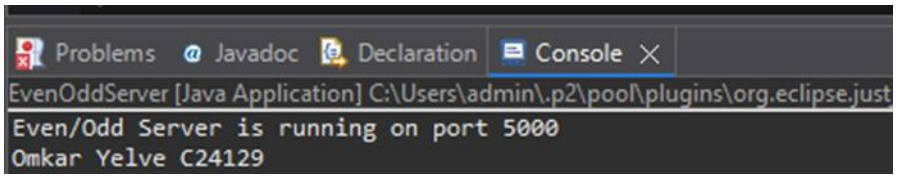
EvenOddClient.java:

```
package rpc;

import java.io.*;
import java.net.*;
import java.util.Scanner;

public class EvenOddClient {
    public static void main(String[] args) {
        final String HOST = "localhost";
        final int PORT = 5000; // must match server

        try (Socket socket = new Socket(HOST, PORT);
             BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
             PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
             Scanner scanner = new Scanner(System.in)) {

            System.out.println("Omkar Yelve C24129");
            System.out.print("Enter a number: ");
            String inputLine = scanner.nextLine().trim();

            if (inputLine.isEmpty()) {
                System.out.println("No input provided. Exiting.");
                return;
            }

            // send number to server
            out.println(inputLine);

            // read server response
            String response = in.readLine();
            if (response == null) {
```

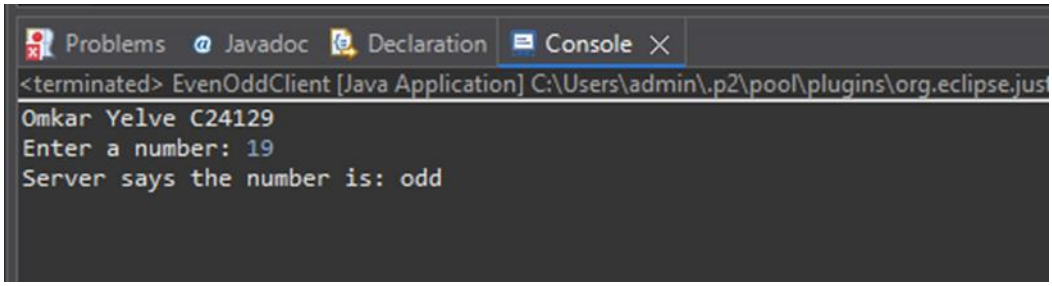
```
                System.out.println("Server closed the connection unexpectedly.");
                return;
            }

            if (response.startsWith("error:")) {
                System.out.println("Server error: " +
response.substring("error:".length()));
                return;
            }

            System.out.println("Server says the number is: " + response);

            // if even, compute and print multiplication table
            if ("even".equalsIgnoreCase(response)) {
                int number;
                try {
                    number = Integer.parseInt(inputLine);
                } catch (NumberFormatException e) {
                    System.out.println("Input was not a valid integer for table
calculation.");
                    return;
                }
                System.out.println("Multiplication Table of " + number + ":");
                for (int i = 1; i <= 10; i++) {
                    System.out.println(number + " x " + i + " = " + (number * i));
                }
            }

        } catch (ConnectException ce) {
            System.err.println("Connection refused. Make sure the server is
running first.");
            ce.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output:

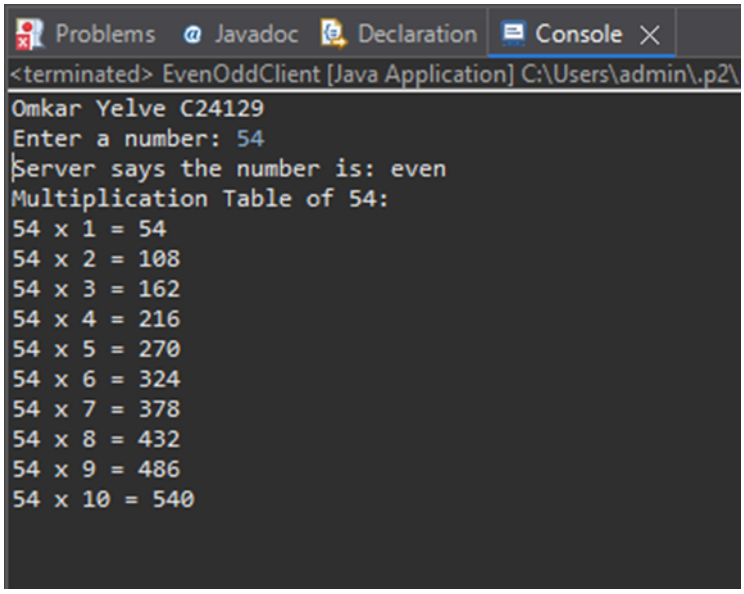
Server Output:



Client Output (For odd value):



Client Output (For even value):



PRACTICAL NO. 3

Remote Method Invocation

1] Calculate addition of two numbers and send it to the client using RMI.

Solution:

Addition.Java:

```
package rmi;

import java.rmi.Remote;
import java.rmi.RemoteException;

public interface Addition extends Remote {
    int add(int a, int b) throws RemoteException;
}
```

AdditionImpl.Java:

```
package rmi;

import java.rmi.server.UnicastRemoteObject;
import java.rmi.RemoteException;

public class AdditionImpl extends UnicastRemoteObject implements Addition {

    private static final long serialVersionUID = 1L;

    protected AdditionImpl() throws RemoteException {
        super();
    }

    @Override
    public int add(int a, int b) throws RemoteException {
        return a + b;
    }
}
```

Server.java:

```
package rmi;

import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class Server {
    private static final String NAME = "Omkar Yelve C24129";
```

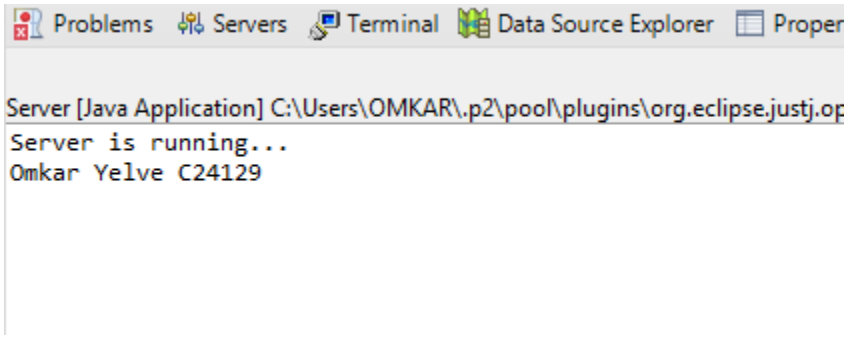
```
    public static void main(String[] args) {
        try {
            // Create remote object
            AdditionImpl obj = new AdditionImpl();

            // Start RMI registry programmatically on port 2345
            Registry registry = LocateRegistry.createRegistry(2345);

            // Bind the remote object to the name "AddService"
            registry.rebind("AddService", obj);

            System.out.println("Server is running...");
            System.out.println(NAME);
        } catch (Exception e) {
            System.out.println("Server exception: " + e.toString());
            e.printStackTrace();
        }
    }
}
```

Client.Java:

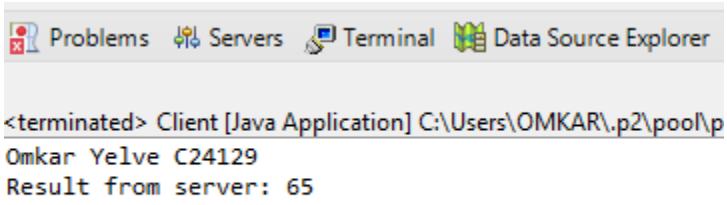
```
package rmi;

import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class Client {
    private static final String NAME = "Omkar Yelve C24129";

    public static void main(String[] args) {
        try {
            // Connect to the registry running on localhost and port 2345
            Registry registry = LocateRegistry.getRegistry("localhost", 2345);

            // Look up the service
            Addition stub = (Addition) registry.lookup("AddService");

            // Call the remote method
            int a = 15;
            int b = 50;
            int result = stub.add(a, b);

            System.out.println(NAME);
            System.out.println("Result from server: " + result);
        } catch (Exception e) {
            System.out.println("Client exception: " + e.toString());
            e.printStackTrace();
        }
    }
}
```

Output:

Server Output:



Client Output:



2] Retrieve time and date function from server to client. This program should display server date and time by implementing RMI.

Solution:

TimeService.Java:

```
package rmi;

import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.Date;

public interface TimeService extends Remote {
    Date getServerDateTime() throws RemoteException;
}
```

TimeServer.Java:

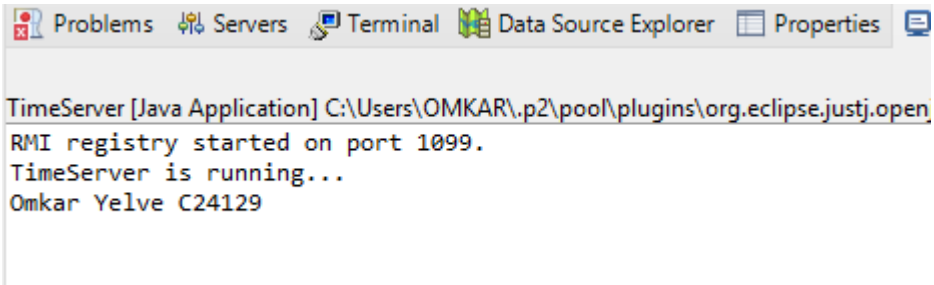
```
package rmi;

import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.server.UnicastRemoteObject;
import java.util.Date;


public class TimeServer extends UnicastRemoteObject implements TimeService {

      protected TimeServer() throws RemoteException {
        super();
    }

        @Override
    public Date getServerDateTime() throws RemoteException {
        return new Date();
    }

        public static void main(String[] args) {
        try {
            LocateRegistry.createRegistry(1099);
            System.out.println("RMI registry started on port 1099.");

            TimeServer server = new TimeServer();
            Naming.rebind("rmi://localhost:1099/TimeService", server);

            System.out.println("TimeServer is running...");
            System.out.println("Omkar Yelve C24129");
        } catch (Exception e) {
            System.out.println("Server exception: " + e);
            e.printStackTrace();
        }
    }
}
```

TimeClient.Java:

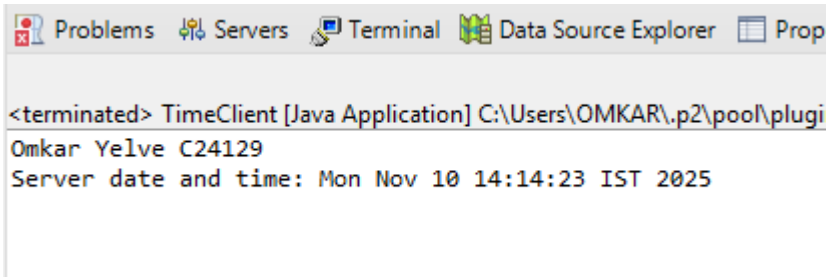
```
package rmi;

import java.rmi.Naming;
import java.util.Date;

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

            TimeService service = (TimeService)
                    Naming.lookup("rmi://localhost:1099/TimeService");


            Date serverTime = service.getServerDateTime();

            System.out.println("Omkar Yelve C24129");
            System.out.println("Server date and time: " + serverTime);
        } catch (Exception e) {
            System.out.println("Client exception: " + e);
            e.printStackTrace();
        }
    }
}
```

Output:

Server Output:



Client Output:


Problems
Servers
Terminal
Data Source Explorer
Prop
<terminated> TimeClient [Java Application] C:\Users\OMKAR\.p2\pool\plugi
Omkar Yelve C24129
Server date and time: Mon Nov 10 14:14:23 IST 2025

3] The client should provide the values of a and b. The server will solve the equation (a+b)2=a2+2ab+b2 and will give back the value of equation using RMI.

Solution:

EquationService.Java:

```
package rmi;

import java.rmi.Remote;
import java.rmi.RemoteException;

/**
 * Remote interface for solving the equation (a + b)^2.
 */
public interface EquationService extends Remote {
    double solveEquation(double a, double b) throws RemoteException;
}
```

EquationServer.Java:

```
package rmi;

import java.rmi.Naming;
import java.rmi.registry.LocateRegistry;
import java.rmi.server.UnicastRemoteObject;

/**
 * RMI server that calculates (a + b)^2 = a^2 + 2ab + b^2
 */
public class EquationServer extends UnicastRemoteObject implements EquationService
{

    private static final long serialVersionUID = 1L;
    private static final String NAME = "Omkar Yelve C24129";

    protected EquationServer() throws java.rmi.RemoteException {
        super();
    }

    @Override
    public double solveEquation(double a, double b) throws
java.rmi.RemoteException {
        return Math.pow(a, 2) + 2 * a * b + Math.pow(b, 2);
    }

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

            LocateRegistry.createRegistry(1099);
```

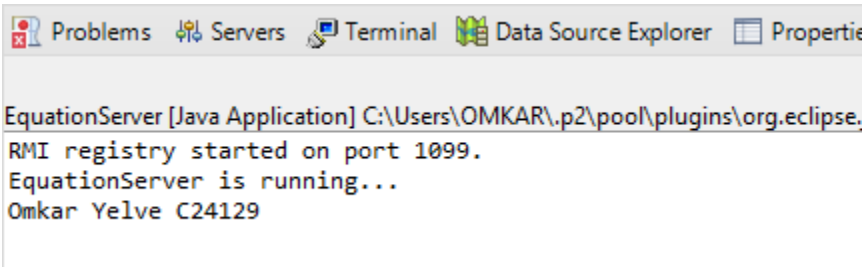
```
            System.out.println("RMI registry started on port 1099.");


            EquationServer server = new EquationServer();
            Naming.rebind("rmi://localhost:1099/EquationService", server);

            System.out.println("EquationServer is running...");
            System.out.println(NAME);
        } catch (Exception e) {
            System.out.println("Server exception: " + e);
            e.printStackTrace();
        }
    }
}
```

EquationClient.java:

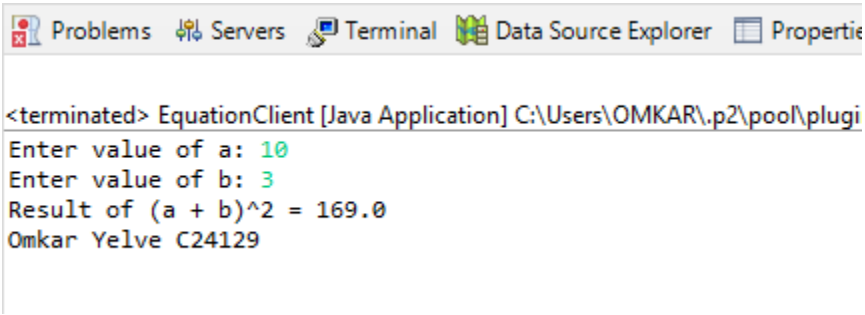
```
package rmi;

import java.rmi.Naming;
import java.util.Scanner;

/**
 * RMI client that asks user for a and b, and prints the result from server.
 */
public class EquationClient {
    private static final String NAME = "Omkar Yelve C24129";

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

            EquationService service = (EquationService)
                    Naming.lookup("rmi://localhost:1099/EquationService");

            Scanner scanner = new Scanner(System.in);
            System.out.print("Enter value of a: ");
            double a = scanner.nextDouble();

            System.out.print("Enter value of b: ");
            double b = scanner.nextDouble();

            double result = service.solveEquation(a, b);
            System.out.println("Result of (a + b)^2 = " + result);

            scanner.close();
        } catch (Exception e) {
            System.out.println("Client exception: " + e);
            e.printStackTrace();
        }

        System.out.println(NAME);
    }
}
```

Output:

Server output:



Client Output:



4] The client should provide the values of a and b. The server will solve the equation (a+b)2=a2+2ab+b2 and (a-b)2=a2+2ab+b2 and will give back the value of equation (Use RMI)

Solution:

EquationService.Java:

```
package rmi;

import java.rmi.Remote;
import java.rmi.RemoteException;

public interface EquationService extends Remote {
    EquationResult solve(double a, double b) throws RemoteException;
}
```

EquationServer1.Java:

```
package rmi;

import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.server.UnicastRemoteObject;

public class EquationServer extends UnicastRemoteObject implements EquationService
{

    private static final long serialVersionUID = 1L;

    protected EquationServer() throws RemoteException {
        super();
    }

    @Override
    public EquationResult solve(double a, double b) throws RemoteException {

        double sumSquare = Math.pow(a, 2) + 2 * a * b + Math.pow(b, 2);

        double diffSquare = Math.pow(a, 2) - 2 * a * b + Math.pow(b, 2);
        return new EquationResult(sumSquare, diffSquare);
    }

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

            LocateRegistry.createRegistry(1099);
            System.out.println("RMI registry started on port 1099.");

            EquationServer server = new EquationServer();
```

```
            Naming.rebind("rmi://localhost:1099/EquationService", server);

            System.out.println("EquationServer is running...");
            System.out.println("Omkar Yelve C24129");
        } catch (Exception e) {
            System.out.println("Server exception: " + e);
            e.printStackTrace();
        }
    }
}
```

EquationClient1.java:

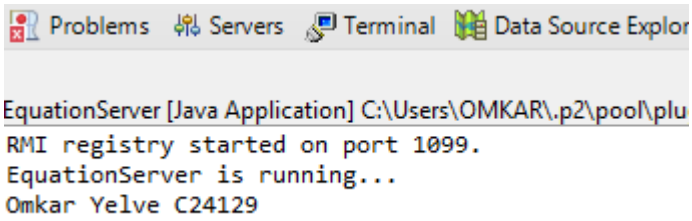
```
package rmi;

import java.rmi.Naming;
import java.util.Scanner;

public class EquationClient {
    public static void main(String[] args) {
        try {
            // Lookup the remote object
            EquationService service = (EquationService)
                    Naming.lookup("rmi://localhost:1099/EquationService");

            Scanner scanner = new Scanner(System.in);

            System.out.print("Enter value for a: ");
            double a = scanner.nextDouble();

            System.out.print("Enter value for b: ");
            double b = scanner.nextDouble();

            EquationResult result = service.solve(a, b);

            System.out.println("\nResult:");
            System.out.println("(a + b)^2 = " + result.getSumSquare());
            System.out.println("(a - b)^2 = " + result.getDiffSquare());
            System.out.println("Omkar Yelve C24129");

            scanner.close();
        } catch (Exception e) {
            System.out.println("Client exception: " + e);
            e.printStackTrace();
        }
    }
}
```

EquationResult.java:

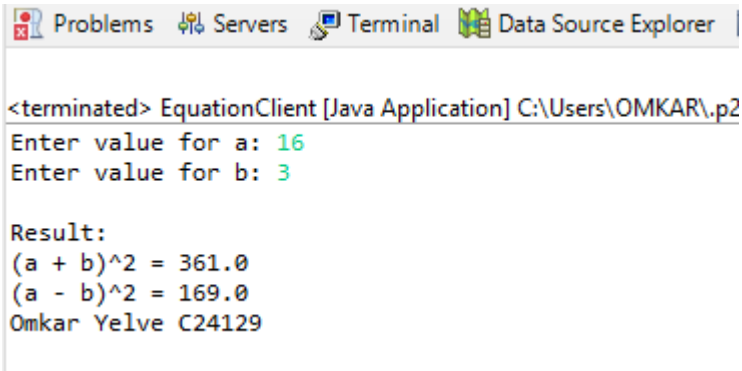
```
package rmi;

import java.io.Serializable;

public class EquationResult implements Serializable {
    private static final long serialVersionUID = 1L;

    private double sumSquare;
    private double diffSquare;

    public EquationResult(double sumSquare, double diffSquare) {
        this.sumSquare = sumSquare;
        this.diffSquare = diffSquare;
    }

    public double getSumSquare() {
        return sumSquare;
    }

    public double getDiffSquare() {
        return diffSquare;
    }
}
```

Output:

Server Output:



Client Output:



Q18] The client should provide the values of a, b & c. The server will solve the equation (ax2 + bx + c = 0) and will give back value of x. If a = 1, b = 5 and c = 6 then return value will be x = -2 or x =-3.

Solution:

QuadraticService.Java:

```
package rmi;

import java.rmi.Remote;
import java.rmi.RemoteException;

public interface QuadraticService extends Remote {
    Roots solveQuadratic(double a, double b, double c) throws RemoteException;
}
```

Roots.Java:

```
package rmi;

import java.io.Serializable;

public class Roots implements Serializable {
    private static final long serialVersionUID = 1L;
    private String result;

    public Roots(String result) {
        this.result = result;
    }

    public String getResult() {
        return result;
    }
}
```

QuadraticServer.Java:

```
package rmi;

import java.rmi.Naming;
import java.rmi.registry.LocateRegistry;
import java.rmi.server.UnicastRemoteObject;

public class QuadraticServer extends UnicastRemoteObject implements
QuadraticService {
    private static final long serialVersionUID = 1L;

    protected QuadraticServer() throws Exception {
        super();
    }
```

```
    @Override
    public Roots solveQuadratic(double a, double b, double c) {
        double discriminant = b * b - 4 * a * c;
        String message;

        if (discriminant > 0) {
            double x1 = (-b + Math.sqrt(discriminant)) / (2 * a);
            double x2 = (-b - Math.sqrt(discriminant)) / (2 * a);
            message = "Two real roots: x = " + x1 + " or x = " + x2;
        } else if (discriminant == 0) {
            double x = -b / (2 * a);
            message = "One real root: x = " + x;
        } else {
            message = "No real roots (Discriminant < 0)";
        }
        return new Roots(message);
    }

    public static void main(String[] args) {
        try {
            LocateRegistry.createRegistry(1099);                    // start
registry
            System.out.println("RMI registry started on port 1099.");

            QuadraticServer server = new QuadraticServer();
            Naming.rebind("rmi://localhost:1099/QuadraticService", server);

            System.out.println("QuadraticServer is running...");
            System.out.println("Omkar Yelve C24129");
        } catch (Exception e) {
            System.err.println("Server exception:");
            e.printStackTrace();
        }
    }
}
```

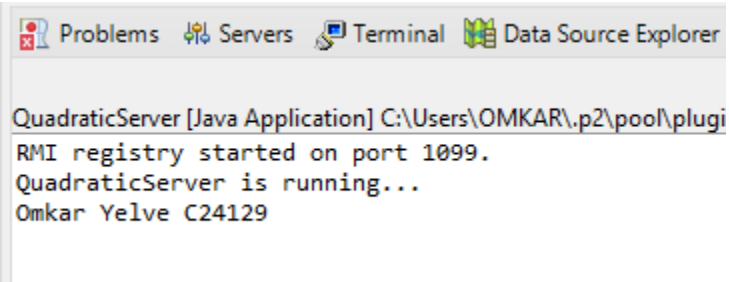
QuadraticClient.Java:

```
package rmi;

import java.rmi.Naming;
import java.util.Scanner;

public class QuadraticClient {
    public static void main(String[] args) {
        try {
            QuadraticService service = (QuadraticService)
                Naming.lookup("rmi://localhost:1099/QuadraticService");

            Scanner scanner = new Scanner(System.in);
            System.out.print("Enter value of a: ");
            double a = scanner.nextDouble();
            System.out.print("Enter value of b: ");
            double b = scanner.nextDouble();
```

```
            System.out.print("Enter value of c: ");
            double c = scanner.nextDouble();

            Roots result = service.solveQuadratic(a, b, c);
            System.out.println("Result: " + result.getResult());
            System.out.println("Omkar Yelve C24129");
            scanner.close();
        } catch (Exception e) {
            System.err.println("Client exception:");
            e.printStackTrace();
        }
    }
}
```

Output:

Server Output:



Client Ouput:

PRACTICAL NO. 4

Remote Method Invocation (Graphical User Interface

1] Design a Graphical User Interface for addition of two numbers. Implement using RMI.

Solution:

AddService.Java:

```
package rmi;

import java.rmi.Remote;
import java.rmi.RemoteException;


public interface AddService extends Remote {
    int add(int a, int b) throws RemoteException;
}
```

AddServer.Java:

```
package rmi;

import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.server.UnicastRemoteObject;

public class AddServer extends UnicastRemoteObject implements AddService {
    private static final long serialVersionUID = 1L;

    protected AddServer() throws RemoteException {
        super();
    }

    @Override
    public int add(int a, int b) throws RemoteException {
        return a + b;
    }

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

            LocateRegistry.createRegistry(1099);
            //System.out.println("RMI registry started on port 1099.");


            AddServer server = new AddServer();


            Naming.rebind("rmi://localhost:1099/AddService", server);
```

```
            System.out.println("AddServer is running...");
            System.out.println("Omkar Yelve C24129");

        } catch (Exception e) {
            System.out.println("Server exception: " + e);
            e.printStackTrace();
        }
    }
}
```

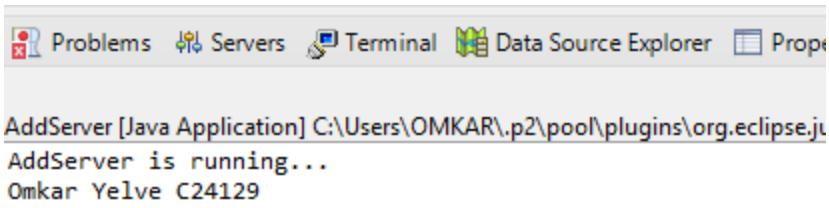
AddClient.java:

```
package rmi;

import javax.swing.*;
import java.awt.*;
import java.rmi.Naming;

public class AddClient extends JFrame {
    private static final long serialVersionUID = 1L;

    private JTextField tfNum1, tfNum2;
    private JButton btnAdd;
    private JLabel lblResult;
    private AddService addService;

    public AddClient() {
        System.out.println("Omkar Yelve C24129");


        tfNum1 = new JTextField(10);
        tfNum2 = new JTextField(10);
        btnAdd = new JButton("Add");
        lblResult = new JLabel("Result: ");


        JPanel panel = new JPanel(new GridLayout(4, 2, 5, 5));
        panel.add(new JLabel("Number 1:"));
        panel.add(tfNum1);
        panel.add(new JLabel("Number 2:"));
        panel.add(tfNum2);
        panel.add(new JLabel(""));
        panel.add(btnAdd);
        panel.add(new JLabel(""));
        panel.add(lblResult);

        add(panel);


        setTitle("RMI Addition Client");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(300, 150);
        setLocationRelativeTo(null);


        try {
```

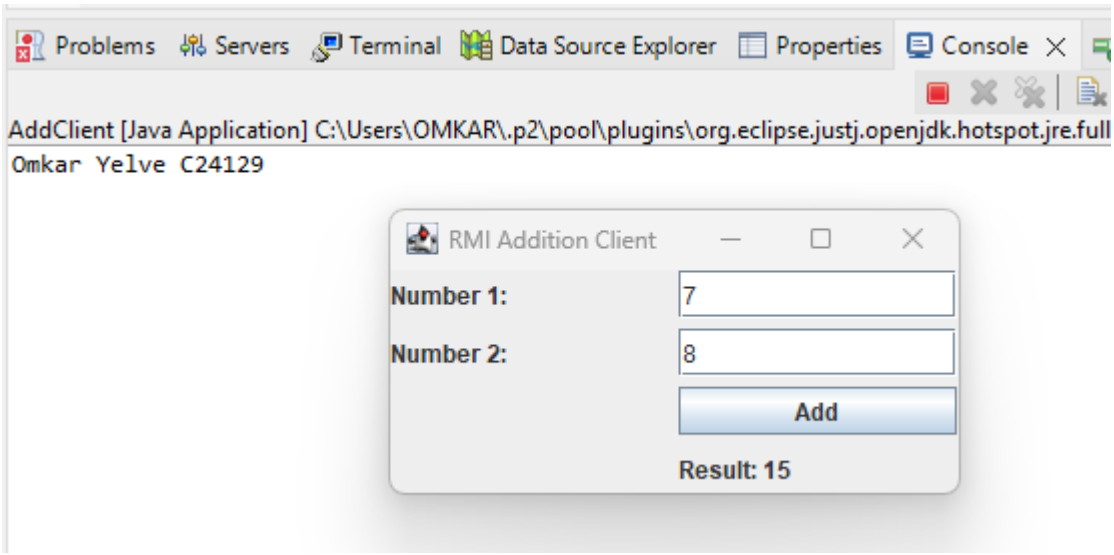
```
            addService = (AddService)
Naming.lookup("rmi://localhost:1099/AddService");
        } catch (Exception e) {
            JOptionPane.showMessageDialog(this, "Failed to connect to RMI server:
" + e.getMessage());
            e.printStackTrace();
            System.exit(1);
        }


        btnAdd.addActionListener(e -> {
            try {
                int a = Integer.parseInt(tfNum1.getText());
                int b = Integer.parseInt(tfNum2.getText());
                int result = addService.add(a, b);
                lblResult.setText("Result: " + result);
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(AddClient.this, "Please enter valid
integers.");
            } catch (Exception ex) {
                JOptionPane.showMessageDialog(AddClient.this, "Error calling RMI
server: " + ex.getMessage());
                ex.printStackTrace();
            }
        });
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new AddClient().setVisible(true));
    }
}
```

Output:

Server Output:



Client Output:


Problems
Servers
Terminal
Data Source Explorer
Properties
Console
AddClient [Java Application] C:\Users\OMKAR\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full
Omkar Yelve C24129
RMI Addition Client
Number 1:
7
Number 2:
8
Add
Result: 15

2] Design a Graphical User Interface (GUI) to find factorial of a given numbers. Implement using RMI.

Solution:

FactorialService.Java:

```
package rmi;

import java.rmi.Remote;
import java.rmi.RemoteException;

public interface FactorialService extends Remote {
    long factorial(int n) throws RemoteException;
}
```

FactorialServer.java:

```
package rmi;

import java.rmi.Naming;
import java.rmi.registry.LocateRegistry;
import java.rmi.server.UnicastRemoteObject;

public class FactorialServer extends UnicastRemoteObject implements
FactorialService {

    private static final long serialVersionUID = 1L;

    protected FactorialServer() throws Exception {
        super();
    }

    @Override
    public long factorial(int n) {
        if (n < 0) return -1L;  // negative input indicator
        long fact = 1L;
        for (int i = 1; i <= n; i++) {
            fact *= i;
        }
        return fact;
    }

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

            LocateRegistry.createRegistry(1099);
            //System.out.println("RMI registry started on port 1099.");


            FactorialServer server = new FactorialServer();
```

```
            Naming.rebind("rmi://localhost:1099/FactorialService", server);

            System.out.println("FactorialServer is running...");
            System.out.println("Omkar Yelve C24129");
        } catch (Exception e) {
            System.err.println("Server exception: " + e);
            e.printStackTrace();
        }
    }
}
```

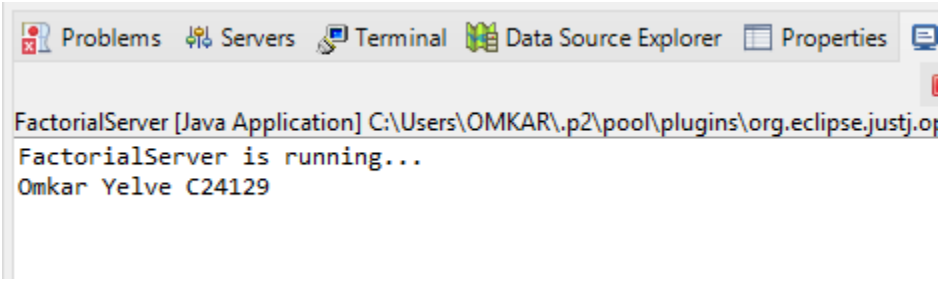
FactorialClient.java:

```
package rmi;

import javax.swing.*;
import java.awt.*;
import java.rmi.Naming;

public class FactorialClient extends JFrame {

    private static final long serialVersionUID = 1L;
    private JTextField inputField;
    private JButton calculateButton;
    private JLabel resultLabel;
    private FactorialService service;

    public FactorialClient() {
        System.out.println("Omkar Yelve C24129");


        try {
            service = (FactorialService)
Naming.lookup("rmi://localhost:1099/FactorialService");
        } catch (Exception e) {
            JOptionPane.showMessageDialog(this, "Failed to connect to server: " +
e.getMessage());
            e.printStackTrace();
            System.exit(1);
        }


        inputField = new JTextField(10);
        calculateButton = new JButton("Calculate");
        resultLabel = new JLabel("Result: ");

        JPanel panel = new JPanel(new GridLayout(3, 2, 5, 5));
        panel.add(new JLabel("Enter number: "));
        panel.add(inputField);
        panel.add(new JLabel(""));
        panel.add(calculateButton);
        panel.add(new JLabel(""));
        panel.add(resultLabel);

        add(panel);
```

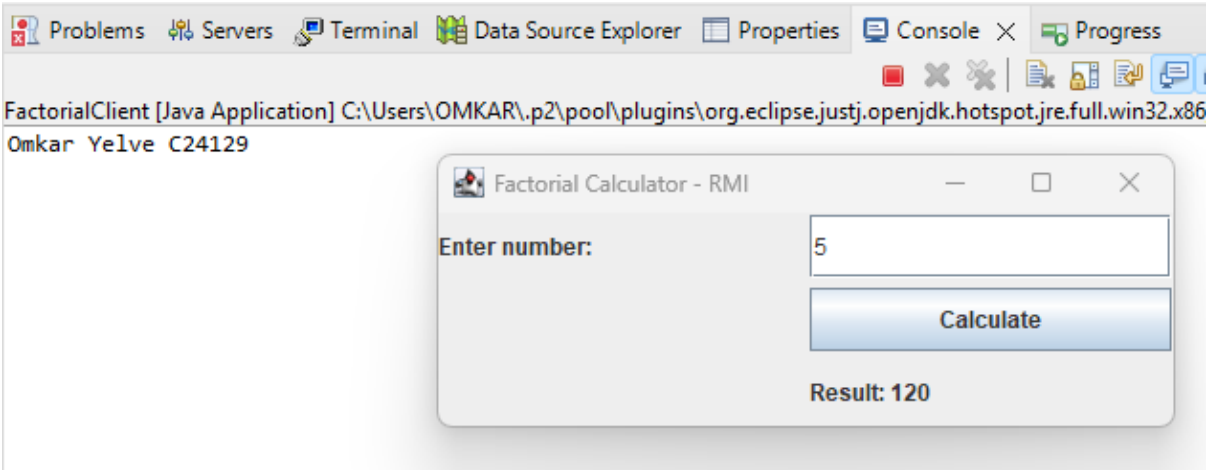
```
        setTitle("Factorial Calculator - RMI");
        setSize(400, 150);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);

        calculateButton.addActionListener(e -> {
            try {
                int n = Integer.parseInt(inputField.getText().trim());
                long fact = service.factorial(n);
                if (fact == -1L) {
                    resultLabel.setText("Result: Invalid input (negative
number)");
                } else {
                    resultLabel.setText("Result: " + fact);
                }
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(FactorialClient.this, "Please enter
a valid integer");
            } catch (Exception ex) {
                JOptionPane.showMessageDialog(FactorialClient.this, "Error
invoking remote method: " + ex.getMessage());
                ex.printStackTrace();
            }
        });
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new FactorialClient().setVisible(true));
    }
}
```

Output:

Server Output:



Client Output :


Problems
Servers
Terminal
Data Source Explorer
Properties
Console
Progress
FactorialClient [Java Application] C:\Users\OMKAR\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86
Omkar Yelve C24129
Factorial Calculator - RMI
Enter number:
5
Calculate
Result: 120

3] Design a Graphical User Interface (GUI) based Basic calculator by implementing RMI

Solution:

CalculatorService.Java:

```
package rmi;

import java.rmi.Remote;
import java.rmi.RemoteException;

public interface CalculatorService extends Remote {
    double add(double a, double b) throws RemoteException;
    double subtract(double a, double b) throws RemoteException;
    double multiply(double a, double b) throws RemoteException;
    double divide(double a, double b) throws RemoteException;
}
```

CalculatorServer.Java:

```
package rmi;

import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.server.UnicastRemoteObject;

public class CalculatorServer extends UnicastRemoteObject implements
CalculatorService {

    private static final long serialVersionUID = 1L;

    protected CalculatorServer() throws RemoteException {
        super();
    }

    @Override
    public double add(double a, double b) throws RemoteException {
        return a + b;
    }

    @Override
    public double subtract(double a, double b) throws RemoteException {
        return a - b;
    }

    @Override
    public double multiply(double a, double b) throws RemoteException {
        return a * b;
    }
```

```
    @Override
    public double divide(double a, double b) throws RemoteException {
        if (b == 0) throw new RemoteException("Division by zero is not allowed.");
        return a / b;
    }

    public static void main(String[] args) {
        try {
            // Start RMI registry automatically
            LocateRegistry.createRegistry(1099);

            CalculatorServer server = new CalculatorServer();
            Naming.rebind("rmi://localhost:1099/CalculatorService", server);

            System.out.println("Calculator Server is running...");
            System.out.println("Omkar Yelve C24129");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

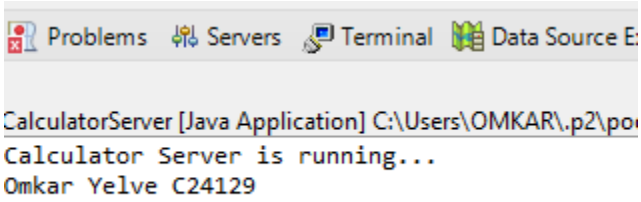
CalculatorClient.Java:

```
package rmi;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.rmi.Naming;

public class CalculatorClient {

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            try {

                CalculatorService stub = (CalculatorService)
Naming.lookup("rmi://localhost:1099/CalculatorService");
                System.out.println("Omkar Yelve C24129");


                JFrame frame = new JFrame("RMI Calculator");
                frame.setSize(350, 250);
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frame.setLayout(new GridLayout(6, 2, 5, 5));


                JTextField num1Field = new JTextField();
                JTextField num2Field = new JTextField();
                JLabel resultLabel = new JLabel("Result: ");

                JButton addBtn = new JButton("Add");
                JButton subBtn = new JButton("Subtract");
```

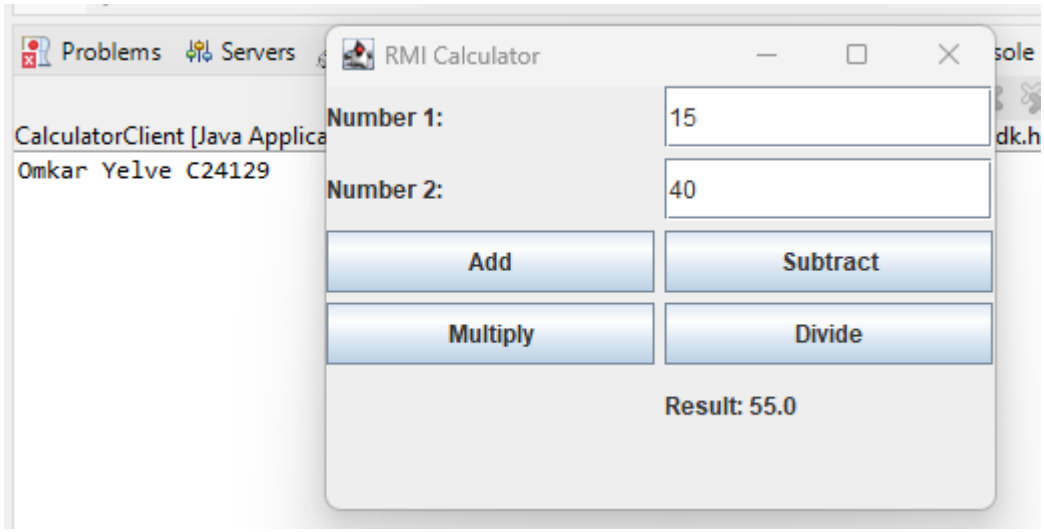
```
                JButton mulBtn = new JButton("Multiply");
                JButton divBtn = new JButton("Divide");


                frame.add(new JLabel("Number 1:"));
                frame.add(num1Field);
                frame.add(new JLabel("Number 2:"));
                frame.add(num2Field);
                frame.add(addBtn);
                frame.add(subBtn);
                frame.add(mulBtn);
                frame.add(divBtn);
                frame.add(new JLabel(""));
                frame.add(resultLabel);


                ActionListener action = e -> {
                    try {
                        double num1 = Double.parseDouble(num1Field.getText());
                        double num2 = Double.parseDouble(num2Field.getText());
                        double result = 0;

                        if (e.getSource() == addBtn)
                            result = stub.add(num1, num2);
                        else if (e.getSource() == subBtn)
                            result = stub.subtract(num1, num2);
                        else if (e.getSource() == mulBtn)
                            result = stub.multiply(num1, num2);
                        else if (e.getSource() == divBtn)
                            result = stub.divide(num1, num2);

                        resultLabel.setText("Result: " + result);
                    } catch (Exception ex) {
                        resultLabel.setText("Error: " + ex.getMessage());
                    }
                };

                addBtn.addActionListener(action);
                subBtn.addActionListener(action);
                mulBtn.addActionListener(action);
                divBtn.addActionListener(action);

                frame.setVisible(true);

            } catch (Exception e) {
                e.printStackTrace();
            }
        });
    }
}
```

Output:

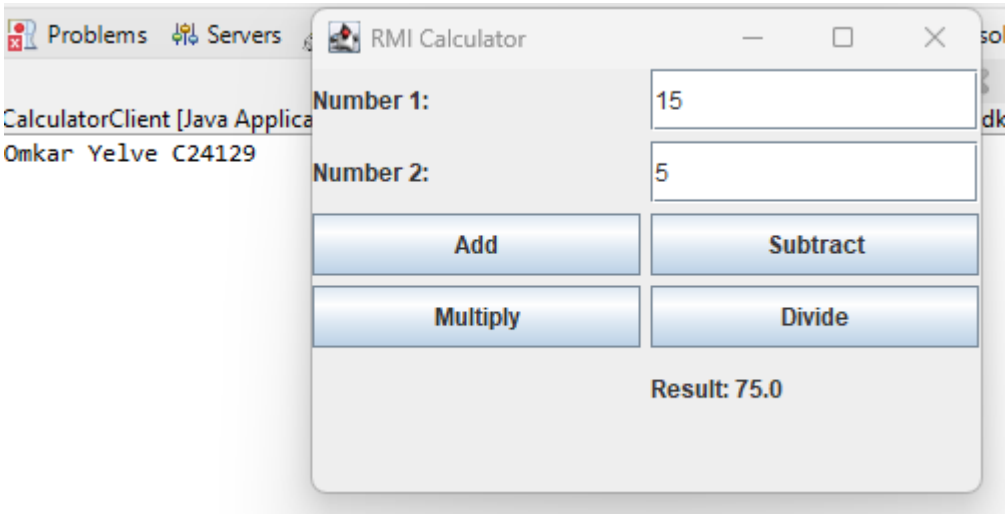
Server Output:



Client Output (Addition):



Client output (Multilication):



4] Design a Graphical User Interface (GUI) to find greatest of two numbers. Implement using RMI

Solution:

Greatest.Java:

```
package rmi;

import java.rmi.Remote;
import java.rmi.RemoteException;

public interface Greatest extends Remote {
    int findGreatest(int a, int b) throws RemoteException;
}
```

GreatestImpl.Java:

```
package rmi;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.rmi.registry.Registry;
import java.rmi.registry.LocateRegistry;

public class GreatestImpl extends UnicastRemoteObject implements Greatest {

    private static final long serialVersionUID = 1L;

    protected GreatestImpl() throws RemoteException {
        super();
    }

    @Override
    public int findGreatest(int a, int b) throws RemoteException {
        return (a > b) ? a : b;
    }

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

            GreatestImpl obj = new GreatestImpl();


            Registry registry = LocateRegistry.createRegistry(1099);
            registry.rebind("GreatestService", obj);

            System.out.println("Greatest Number RMI Server is running...");
            System.out.println("Omkar Yelve C24129");  // Updated name
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

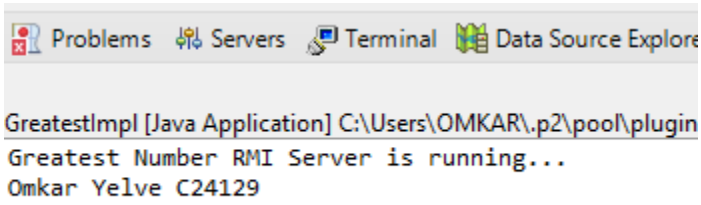
GreatestClient.Java:

```
package rmi;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class GreatestClient {

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            try {

                Registry registry = LocateRegistry.getRegistry("localhost", 1099);
                Greatest stub = (Greatest) registry.lookup("GreatestService");

                System.out.println("Omkar Yelve C24129");  // Updated name


                JFrame frame = new JFrame("Find Greatest Number (RMI)");
                frame.setSize(350, 200);
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frame.setLayout(new GridLayout(4, 2, 5, 5));

                JTextField num1Field = new JTextField();
                JTextField num2Field = new JTextField();
                JLabel resultLabel = new JLabel("Result: ");
                JButton compareBtn = new JButton("Find Greatest");

                frame.add(new JLabel("First Number:"));
                frame.add(num1Field);
                frame.add(new JLabel("Second Number:"));
                frame.add(num2Field);
                frame.add(compareBtn);
                frame.add(resultLabel);


                compareBtn.addActionListener(e -> {
                    try {
                        int num1 = Integer.parseInt(num1Field.getText());
                        int num2 = Integer.parseInt(num2Field.getText());
                        int greatest = stub.findGreatest(num1, num2);
                        resultLabel.setText("Result: " + greatest);
                    } catch (NumberFormatException ex) {
                        JOptionPane.showMessageDialog(frame, "Please enter valid
integers.");
                    } catch (Exception ex) {
                        JOptionPane.showMessageDialog(frame, "Error: " +
ex.getMessage());
                    }
                });

                frame.setLocationRelativeTo(null);
                frame.setVisible(true);

            } catch (Exception e) {
```

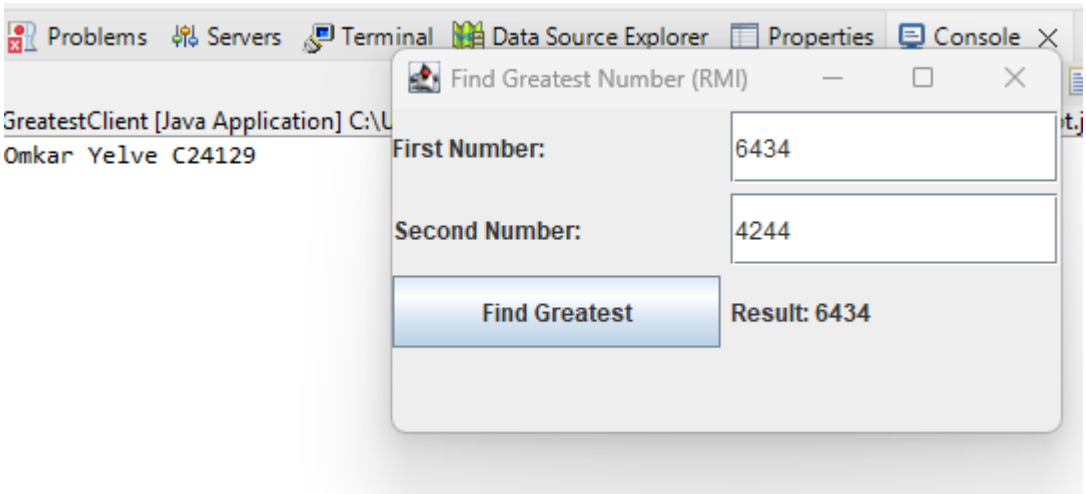
```
                e.printStackTrace();
            }
        });
    }
}
```

Output:

Server Ouput:



Client Output:



5] Design a Graphical User Interface (GUI) which accepts a numerical value from the client. Convert the number in to words. Implement using RMI.

Solution:

NumberConverter.java:

```
package rmi;

import java.rmi.Remote;
import java.rmi.RemoteException;

public interface NumberConverter extends Remote {
    String convertToWords(int number) throws RemoteException;
}
```

NumberConverterImpl.Java:

```
package rmi;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class NumberConverterImpl extends UnicastRemoteObject implements
NumberConverter {

    private static final long serialVersionUID = 1L;

    protected NumberConverterImpl() throws RemoteException {
        super();
    }

    @Override
    public String convertToWords(int number) throws RemoteException {
        if (number == 0) return "zero";

        String[] units = { "", "one", "two", "three", "four", "five", "six",
"seven", "eight", "nine",
                           "ten", "eleven", "twelve", "thirteen", "fourteen",
"fifteen", "sixteen",
                           "seventeen", "eighteen", "nineteen" };

        String[] tens = { "", "", "twenty", "thirty", "forty", "fifty", "sixty",
"seventy", "eighty", "ninety" };

        StringBuilder words = new StringBuilder();

        if (number >= 1000) {
            words.append(units[number / 1000]).append(" thousand ");
```

```
            number %= 1000;
        }
        if (number >= 100) {
            words.append(units[number / 100]).append(" hundred ");
            number %= 100;
        }
        if (number >= 20) {
            words.append(tens[number / 10]).append(" ");
            number %= 10;
        }
        if (number > 0) {
            words.append(units[number]).append(" ");
        }

        return words.toString().trim();
    }

    public static void main(String[] args) {
        try {
            NumberConverterImpl obj = new NumberConverterImpl();

            Registry registry = LocateRegistry.createRegistry(1099);

            registry.rebind("NumberService", obj);

            System.out.println("Number to Words RMI Server is running...");
            System.out.println("Omkar Yelve C24129");  // Updated name

        } catch (Exception e) {
            System.out.println("❌ Server exception: " + e.getMessage());
            e.printStackTrace();
        }
    }
}
```

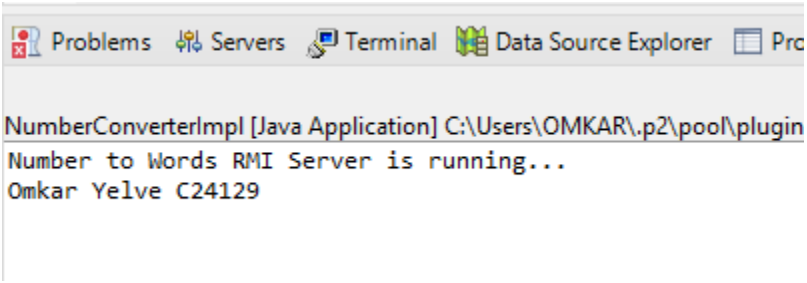
NumberClient.Java:

```
package rmi;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class NumberClient {

    public static void main(String[] args) {
        try {
            Registry registry = LocateRegistry.getRegistry("localhost", 1099);
            NumberConverter stub = (NumberConverter)
registry.lookup("NumberService");

            JFrame frame = new JFrame("Number to Words Converter (RMI)");
            frame.setSize(400, 200);
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

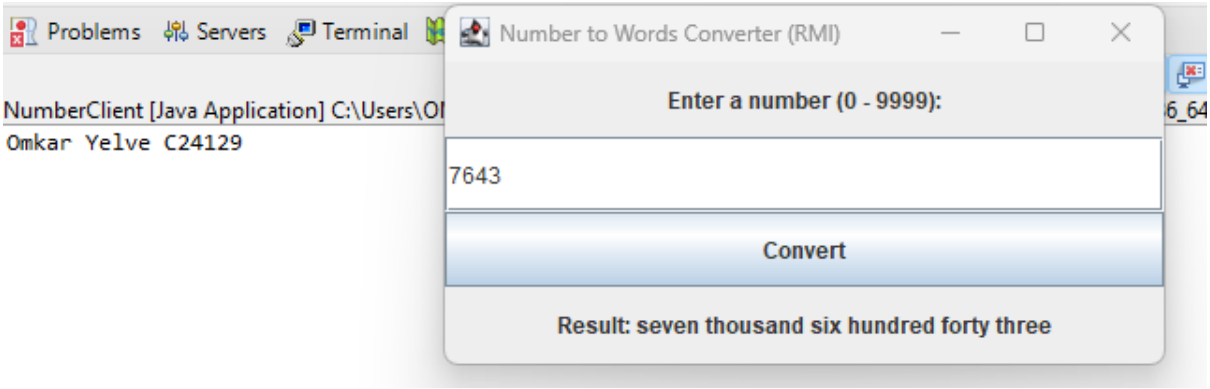
```
            frame.setLayout(new GridLayout(4, 1));

            JTextField inputField = new JTextField();
            JButton convertButton = new JButton("Convert");
            JLabel resultLabel = new JLabel("Result: ", SwingConstants.CENTER);

            frame.add(new JLabel("Enter a number (0 - 9999):",
SwingConstants.CENTER));
            frame.add(inputField);
            frame.add(convertButton);
            frame.add(resultLabel);

            convertButton.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    try {
                        int number = Integer.parseInt(inputField.getText());
                        if (number < 0 || number > 9999) {
                            resultLabel.setText("Enter number between 0 and
9999.");
                        } else {
                            String words = stub.convertToWords(number);
                            resultLabel.setText("Result: " + words);
                        }
                    } catch (NumberFormatException ex) {
                        resultLabel.setText("Invalid number.");
                    } catch (Exception ex) {
                        resultLabel.setText("Error: " + ex.getMessage());
                    }
                }
            });

            frame.setLocationRelativeTo(null);
            frame.setVisible(true);

        } catch (Exception e) {
            e.printStackTrace();
        }

        System.out.println("Omkar Yelve C24129");  // Updated name
    }
}
```

Output:

Server Output:



Client Output:


Problems
Servers
Terminal
NumberClient [Java Application] C:\Users\O
Omkar Yelve C24129
Number to Words Converter (RMI)
Enter a number (0 - 9999):
7643
Convert
Result: seven thousand six hundred forty three

PRACTICAL NO. 5

JDBC and RMI

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.

Solution:

Database:

```
CREATE DATABASE Library;

USE Library;


CREATE TABLE Book (
    Book_id INT PRIMARY KEY,
    Book_name VARCHAR(100),
    Book_author VARCHAR(100)
);


INSERT INTO Book VALUES
(1, 'Java: The Complete Reference', 'Herbert Schildt'),
(2, 'Effective Java', 'Joshua Bloch'),
(3, 'Clean Code', 'Robert C. Martin'),
(4, 'Core Java Volume I', 'Cay S. Horstmann'),
(5, 'Head First Java', 'Kathy Sierra');
```

LibraryService.jav:

```java
package com.example.rmi;

import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;
```

```
public interface LibraryService extends Remote {
    List<String> getBooks() throws RemoteException;
}
```

LibraryServiceImpl.java

```
package com.example.rmi;

import java.rmi.server.UnicastRemoteObject;
import java.rmi.RemoteException;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class LibraryServiceImpl extends UnicastRemoteObject implements
LibraryService {
    public LibraryServiceImpl() throws RemoteException {
        super();
    }



 public List<String> getBooks() throws RemoteException {
        List<String> books = new ArrayList<>();
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con = DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/Library", "root", "admin");
            Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT * FROM Book");
            while (rs.next()) {
                String record = "Book ID: " + rs.getInt(1) +
                                ", Name: " + rs.getString(2) +
                                ", Author: " + rs.getString(3);
                books.add(record);
            }
            con.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return books;
    }
}
```

LibraryServer.java:

```
package com.example.rmi;

import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class LibraryServer {
    public static void main(String[] args) {
        try {
            LibraryService service = new LibraryServiceImpl();
            Registry registry = LocateRegistry.createRegistry(1104);
            registry.rebind("LibraryService", service);
            System.out.println("Omkar Yelve C24129");
            System.out.println("Library Server is ready on port 1104...");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

LibraryClient.java:

```
package com.example.rmi;

import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.util.List;

public class LibraryClient {
    public static void main(String[] args) {
        try {
            Registry registry = LocateRegistry.getRegistry("localhost", 1104);
            LibraryService service = (LibraryService)
registry.lookup("LibraryService");

            List<String> books = service.getBooks();
            System.out.println("Omkar Yelve C24129");
            System.out.println("Book Information from Library Database:");
            for (String b : books) {
                System.out.println(b);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

```
}
```

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS D:\RMI_Library> java -cp "out;lib/mysql-connector-j-9.5.0.jar" com.example.rmi.LibraryClient
Omkar Yelve C24129
Book Information from Library Database:
Book ID: 1, Name: Java: The Complete Reference, Author: Herbert Schildt
Book ID: 2, Name: Effective Java, Author: Joshua Bloch
Book ID: 3, Name: Clean Code, Author: Robert C. Martin
Book ID: 4, Name: Core Java Volume I, Author: Cay S. Horstmann
Book ID: 5, Name: Head First Java, Author: Kathy Sierra
PS D:\RMI_Library>
```

PRACTICAL NO. 6

Using MySQL create a Student database. Create table student_data(ID ,NAME , BRANCH ,PERCENTAGE ,EMAIL) and retrieve the student_data information from Student database using Remote Object Communication concept.

Solution:

Student.java:

```
package com.example.rmi;

import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;

public interface Student extends Remote {
    List<String> getStudentData() throws RemoteException;
}
```

StudentImpl.java:

```
package com.example.rmi;

import java.rmi.server.UnicastRemoteObject;
import java.rmi.RemoteException;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

@SuppressWarnings("serial")
public class StudentImpl extends UnicastRemoteObject implements Student {

    protected StudentImpl() throws RemoteException {
        super();
    }

    @Override
    public List<String> getStudentData() throws RemoteException {
        List<String> students = new ArrayList<>();

        String url = "jdbc:mysql://localhost:3306/Student";
        String user = "root";     // change if different
```

```
        String password = "omkar"; // change to your MySQL password

        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con = DriverManager.getConnection(url, user, password);
            Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT * FROM student_data");

            while (rs.next()) {
                String info = "ID: " + rs.getInt("ID") +
                        ", Name: " + rs.getString("NAME") +
                        ", Branch: " + rs.getString("BRANCH") +
                        ", Percentage: " + rs.getDouble("PERCENTAGE") +
                        ", Email: " + rs.getString("EMAIL");
                students.add(info);
            }

            rs.close();
            stmt.close();
            con.close();

        } catch (Exception e) {
            e.printStackTrace();
        }

        return students;
    }
}
```

RMIServer:

```
package com.example.rmi;

import java.rmi.Naming;
import java.rmi.registry.LocateRegistry;

public class RMIServer {
    public static void main(String[] args) {
        try {
            // Start RMI registry
            LocateRegistry.createRegistry(1099);

            // Create remote object
            StudentImpl obj = new StudentImpl();

            // Bind object in RMI registry
            Naming.rebind("rmi://localhost:1099/studentService", obj);
```

```
            System.out.println("RMI Server is running...");
            System.out.println("Omkar Yelve C24129");
        } catch (Exception e) {
            System.err.println("Server Error:");
            e.printStackTrace();
        }
    }
}
```

RMIClient:

```
package com.example.rmi;

import java.rmi.Naming;

public class RMIClient {
    public static void main(String[] args) {
        try {
            // Connect to remote object
            Student stub = (Student)
Naming.lookup("rmi://localhost:1099/studentService");

            // Call remote method
            java.util.List<String> data = stub.getStudentData();

            // Display results
            System.out.println("=== Student Data Retrieved from Remote Server
===");
            for (String s : data) {
                System.out.println(s);
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Database:

Query 1

Limit to 1000 rows

```sql
CREATE DATABASE Student;
USE Student;

CREATE TABLE student_data (
    ID INT PRIMARY KEY AUTO_INCREMENT,
    NAME VARCHAR(50),
    BRANCH VARCHAR(30),
    PERCENTAGE DECIMAL(5,2),
    EMAIL VARCHAR(100)
);

INSERT INTO student_data (NAME, BRANCH, PERCENTAGE, EMAIL)
VALUES
('Omkar Yelve', 'CSE', 80.5, 'omkar@example.com'),
('Amit Kumar', 'CSE', 89.5, 'amit@example.com'),
('Neha Singh', 'ECE', 78.2, 'neha@example.com'),
('Raj Verma', 'IT', 91.0, 'raj@example.com');
```

Output

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell

```
PS D:\RMI_Library> java -cp "out;lib\mysql-connector-j-9.5.0.jar" com.example.rmi.RMIClient
=== Student Data Retrieved from Remote Server ===
ID: 1, Name: Omkar Yelve, Branch: CSE, Percentage: 80.5, Email: omkar@example.com
ID: 2, Name: Amit Kumar, Branch: CSE, Percentage: 89.5, Email: amit@example.com
ID: 3, Name: Neha Singh, Branch: ECE, Percentage: 78.2, Email: neha@example.com
ID: 4, Name: Raj Verma, Branch: IT, Percentage: 91.0, Email: raj@example.com
PS D:\RMI_Library>
```

PRACTICAL NO. 7

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

Solution:

BillService.java:

```
package com.example.rmi;

import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;

// Remote interface for Bill service
public interface BillService extends Remote {
    List<String> getBillData() throws RemoteException;
}
```

BillServiceImpl.java:

```
package com.example.rmi;


import java.rmi.server.UnicastRemoteObject;
import java.rmi.RemoteException;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

@SuppressWarnings("serial")
public class BillServiceImpl extends UnicastRemoteObject implements
BillService {

    protected BillServiceImpl() throws RemoteException {
        super();
    }

    @Override
    public List<String> getBillData() throws RemoteException {
        List<String> bills = new ArrayList<>();

        String url = "jdbc:mysql://localhost:3306/ElectricBill";
        String user = "root";
        String password = "omkar";
```

```
        try {
            // Load MySQL Driver
            Class.forName("com.mysql.cj.jdbc.Driver");

            // Establish Connection
            Connection con = DriverManager.getConnection(url, user, password);

            // Query the Bill table
            Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT * FROM Bill");

            while (rs.next()) {
                String info = "Consumer Name: " +
rs.getString("consumer_name") +
                              ", Bill Due Date: " +
rs.getDate("bill_due_date") +
                              ", Bill Amount: ₹" +
rs.getDouble("bill_amount");
                bills.add(info);
            }

            // Close resources
            rs.close();
            stmt.close();
            con.close();

        } catch (Exception e) {
            e.printStackTrace();
        }

        return bills;
    }
}
```

BillServer.java:

```
package com.example.rmi;

import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class BillServer {
    public static void main(String[] args) {
        try {
            BillServiceImpl obj = new BillServiceImpl();

            // Start RMI Registry on port 1108
            Registry registry = LocateRegistry.createRegistry(1108);
            registry.rebind("BillService", obj);
```

```
            System.out.println("BillService bound to registry on port 1108");
            System.out.println("Omkar Yelve C24129");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

BillClient.java:

```
package com.example.rmi;

import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.util.List;

public class BillClient {
    public static void main(String[] args) {
        try {
            Registry registry = LocateRegistry.getRegistry("localhost", 1108);
            BillService stub = (BillService) registry.lookup("BillService");

            List<String> bills = stub.getBillData();

            System.out.println("Electric Bill Data from Database:");
            for (String b : bills) {
                System.out.println(b);
            }

            System.out.println("Omkar Yelve C24129");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Database:

Query 1 SQL File 3*

Limit to 1000 rows

```sql
CREATE DATABASE ElectricBill;
USE ElectricBill;

CREATE TABLE Bill (
    consumer_name VARCHAR(50),
    bill_due_date DATE,
    bill_amount DECIMAL(10,2)
);

INSERT INTO Bill (consumer_name, bill_due_date, bill_amount)
VALUES
('Omkar Yelve', '2025-11-15', 1050.75),
('Amit Kumar', '2025-11-15', 1250.75),
('Neha Singh', '2025-11-20', 980.50),
('Raj Verma', '2025-12-01', 1450.00);
```

Output:

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell

```
PS D:\RMI_Library> java -cp "out;lib/mysql-connector-j-9.5.0.jar" com.example.rmi.BillClient
Electric Bill Data from Database:
Consumer Name: Omkar Yelve, Bill Due Date: 2025-11-15, Bill Amount: ?1050.75
Consumer Name: Amit Kumar, Bill Due Date: 2025-11-15, Bill Amount: ?1250.75
Consumer Name: Neha Singh, Bill Due Date: 2025-11-20, Bill Amount: ?980.5
Consumer Name: Raj Verma, Bill Due Date: 2025-12-01, Bill Amount: ?1450.0
Omkar Yelve C24129
PS D:\RMI_Library>
```

PRACTICAL NO. 8

Implementation of mutual exclusion using Token ring algorithm.

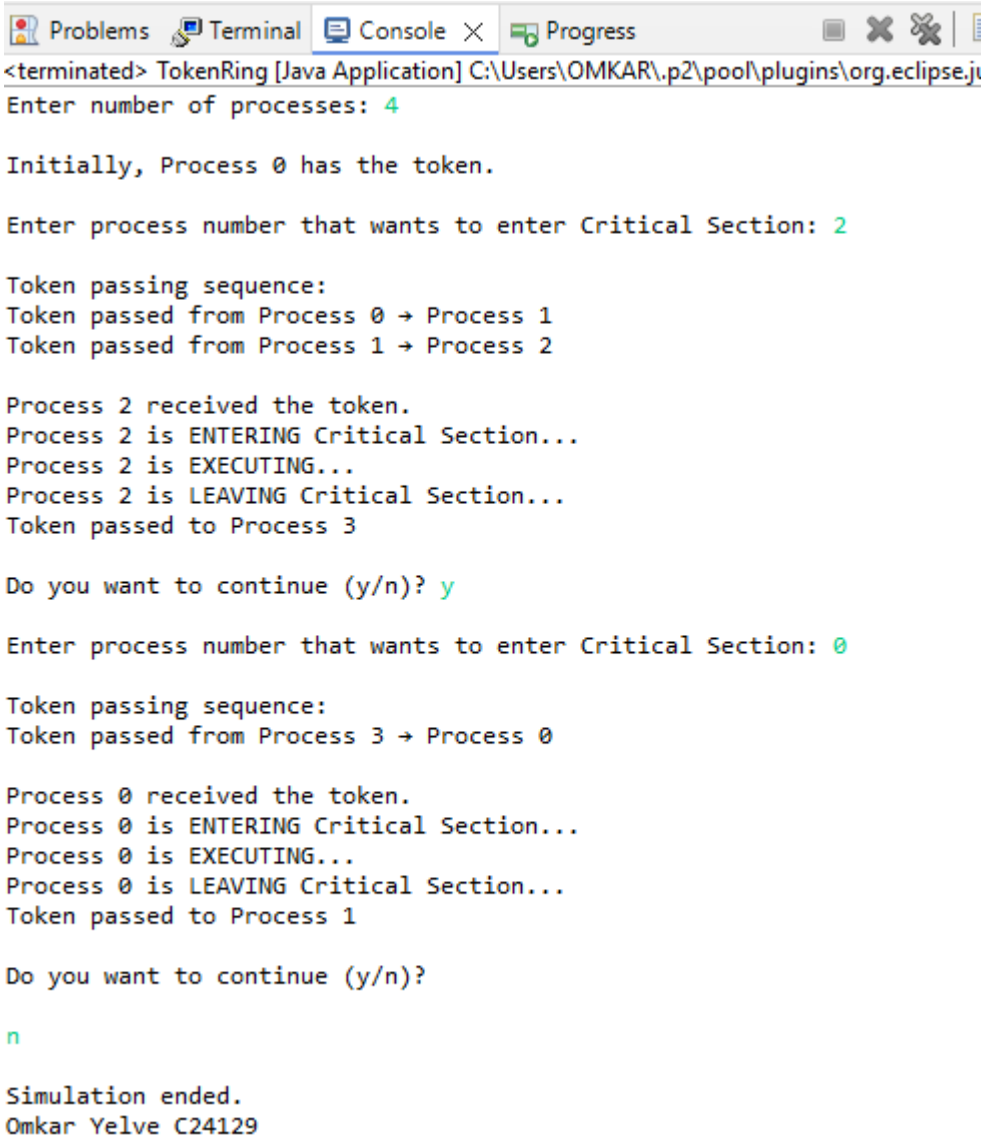
Solution:

TokenRing.java

```
package TokenRing;

import java.util.Scanner;

class Process {
    int id;
    boolean hasToken;
    boolean requestCS;

    public Process(int id) {
        this.id = id;
        this.hasToken = false;
        this.requestCS = false;
    }
}

public class TokenRing {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter number of processes: ");
        int n = sc.nextInt();

        Process[] processes = new Process[n];
        for (int i = 0; i < n; i++) {
            processes[i] = new Process(i);
        }

        // Initially, Process 0 has the token
        processes[0].hasToken = true;
        System.out.println("\nInitially, Process 0 has the token.");

        char choice = 0;
        do {
            System.out.print("\nEnter process number that wants to enter Critical
Section: ");
            int req = sc.nextInt();

            if (req < 0 || req >= n) {
                System.out.println("Invalid process number!");
                continue;
            }

            // Find which process currently has the token
            int tokenHolder = -1;
            for (int i = 0; i < n; i++) {
                if (processes[i].hasToken) {
                    tokenHolder = i;
                    break;
```

```
                }
            }

            System.out.println("\nToken passing sequence:");
            int current = tokenHolder;
            while (current != req) {
                System.out.println("Token passed from Process " + current + " →
Process " + ((current + 1) % n));
                current = (current + 1) % n;
            }

            // Token reached the requesting process
            processes[tokenHolder].hasToken = false;
            processes[req].hasToken = true;

            System.out.println("\nProcess " + req + " received the token.");
            System.out.println("Process " + req + " is ENTERING Critical
Section...");
            System.out.println("Process " + req + " is EXECUTING...");
            System.out.println("Process " + req + " is LEAVING Critical
Section...");

            // Pass token to next process in ring
            int next = (req + 1) % n;
            processes[req].hasToken = false;
            processes[next].hasToken = true;
            System.out.println("Token passed to Process " + next);

            System.out.print("\nDo you want to continue (y/n)? ");
            choice = sc.next().charAt(0);

        } while (choice == 'y' || choice == 'Y');

        sc.close();
        System.out.println("\nSimulation ended.");
        System.out.println("Omkar Yelve C24129");
    }
}
```

Output:


Problems
Terminal
Console
Progress
<terminated> TokenRing [Java Application] C:\Users\OMKAR\.p2\pool\plugins\org.eclipse.ju
Enter number of processes: 4
Initially, Process 0 has the token.
Enter process number that wants to enter Critical Section: 2
Token passing sequence:
Token passed from Process 0 → Process 1
Token passed from Process 1 → Process 2
Process 2 received the token.
Process 2 is ENTERING Critical Section...
Process 2 is EXECUTING...
Process 2 is LEAVING Critical Section...
Token passed to Process 3
Do you want to continue (y/n)? y
Enter process number that wants to enter Critical Section: 0
Token passing sequence:
Token passed from Process 3 → Process 0
Process 0 received the token.
Process 0 is ENTERING Critical Section...
Process 0 is EXECUTING...
Process 0 is LEAVING Critical Section...
Token passed to Process 1
Do you want to continue (y/n)?
n
Simulation ended.
Omkar Yelve C24129

PRACTICAL NO. 9

Implementation of Election Algorithm using Ring Algorithm.

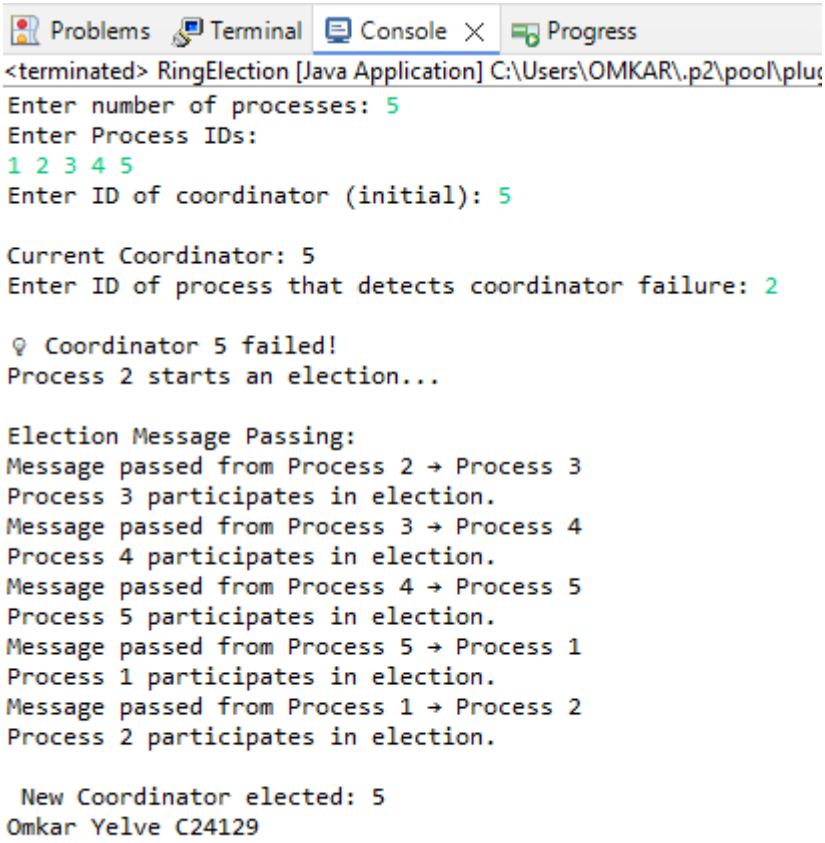
Solution :

RingElection.java

```
package TokenRing;

import java.util.Scanner;

public class RingElection {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter number of processes: ");
        int n = sc.nextInt();

        int[] processes = new int[n];
        boolean[] active = new boolean[n];

        System.out.println("Enter Process IDs:");
        for (int i = 0; i < n; i++) {
            processes[i] = sc.nextInt();
            active[i] = true;
        }

        System.out.print("Enter ID of coordinator (initial): ");
        int coordinator = sc.nextInt();

        System.out.println("\nCurrent Coordinator: " + coordinator);

        System.out.print("Enter ID of process that detects coordinator failure:
");
        int initiator = sc.nextInt();

        System.out.println("\n Coordinator " + coordinator + " failed!");
        System.out.println("Process " + initiator + " starts an election...");

        int newCoordinator = ringElection(processes, active, initiator);
        System.out.println("\n New Coordinator elected: " + newCoordinator);
        System.out.println("Omkar Yelve C24129");

        sc.close();
    }

    static int ringElection(int[] processes, boolean[] active, int initiator) {
        int n = processes.length;
        int maxId = -1;
        int startIndex = -1;



        // Find the index of initiator
```

```
        for (int i = 0; i < n; i++) {
            if (processes[i] == initiator) {
                startIndex = i;
                break;
            }
        }

        System.out.println("\nElection Message Passing:");

        int current = startIndex;
        do {
            int next = (current + 1) % n;

            System.out.println("Message passed from Process " + processes[current]
+ " → Process " + processes[next]);

            // Simulate process participation
            if (active[next]) {
                System.out.println("Process " + processes[next] + " participates
in election.");
            }

            // Keep track of the highest ID
            if (processes[next] > maxId && active[next]) {
                maxId = processes[next];
            }

            current = next;
        } while (current != startIndex);

        // The process with the highest ID becomes the coordinator
        return maxId;
    }
}
```

Output:

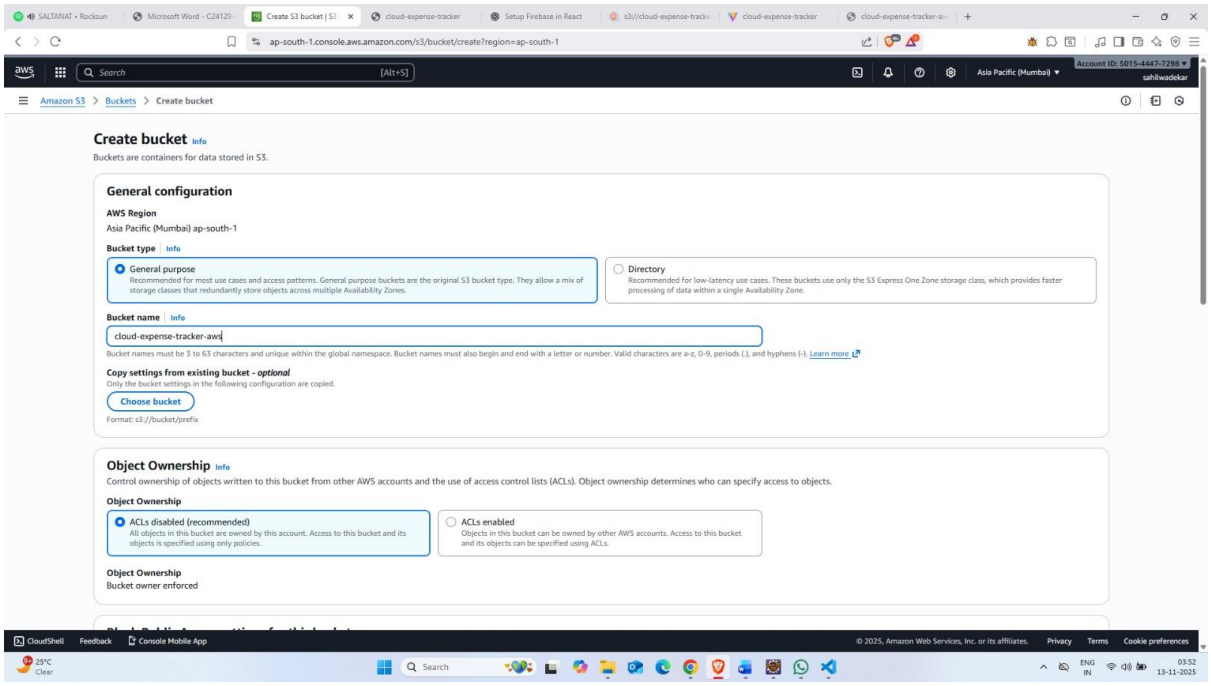

Problems
Terminal
Console
Progress
<terminated> RingElection [Java Application] C:\Users\OMKAR\.p2\pool\plu
Enter number of processes: 5
Enter Process IDs:
1 2 3 4 5
Enter ID of coordinator (initial): 5
Current Coordinator: 5
Enter ID of process that detects coordinator failure: 2
Coordinator 5 failed!
Process 2 starts an election...
Election Message Passing:
Message passed from Process 2 → Process 3
Process 3 participates in election.
Message passed from Process 3 → Process 4
Process 4 participates in election.
Message passed from Process 4 → Process 5
Process 5 participates in election.
Message passed from Process 5 → Process 1
Process 1 participates in election.
Message passed from Process 1 → Process 2
Process 2 participates in election.
New Coordinator elected: 5
Omkar Yelve C24129

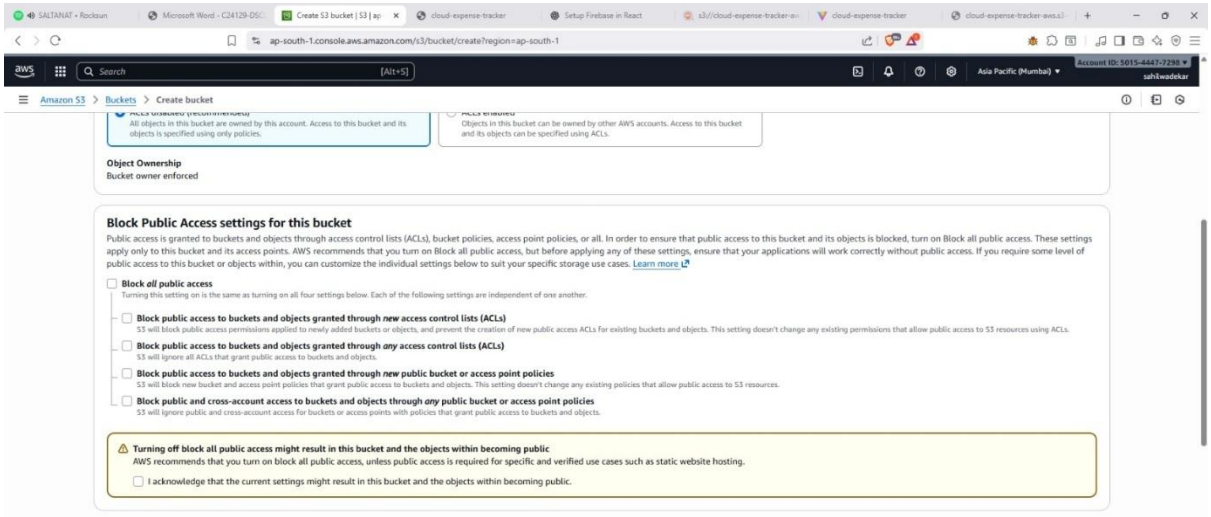
PRACTICAL NO. 10

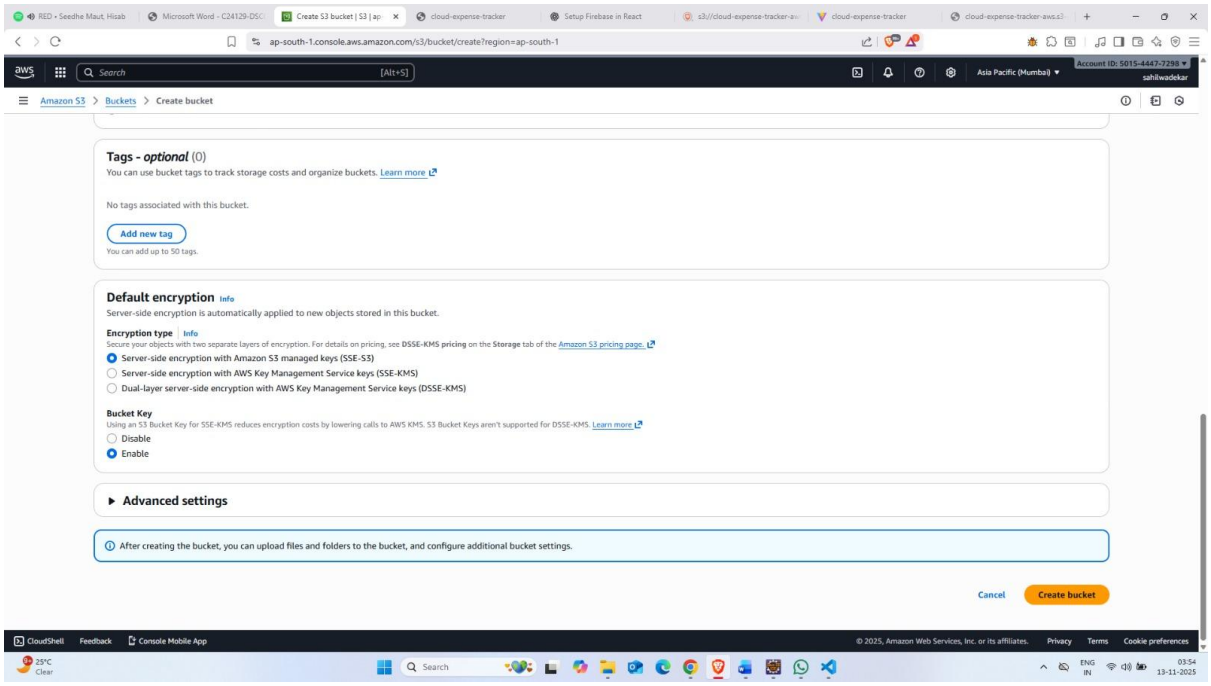
Implementation of Storage as a Service

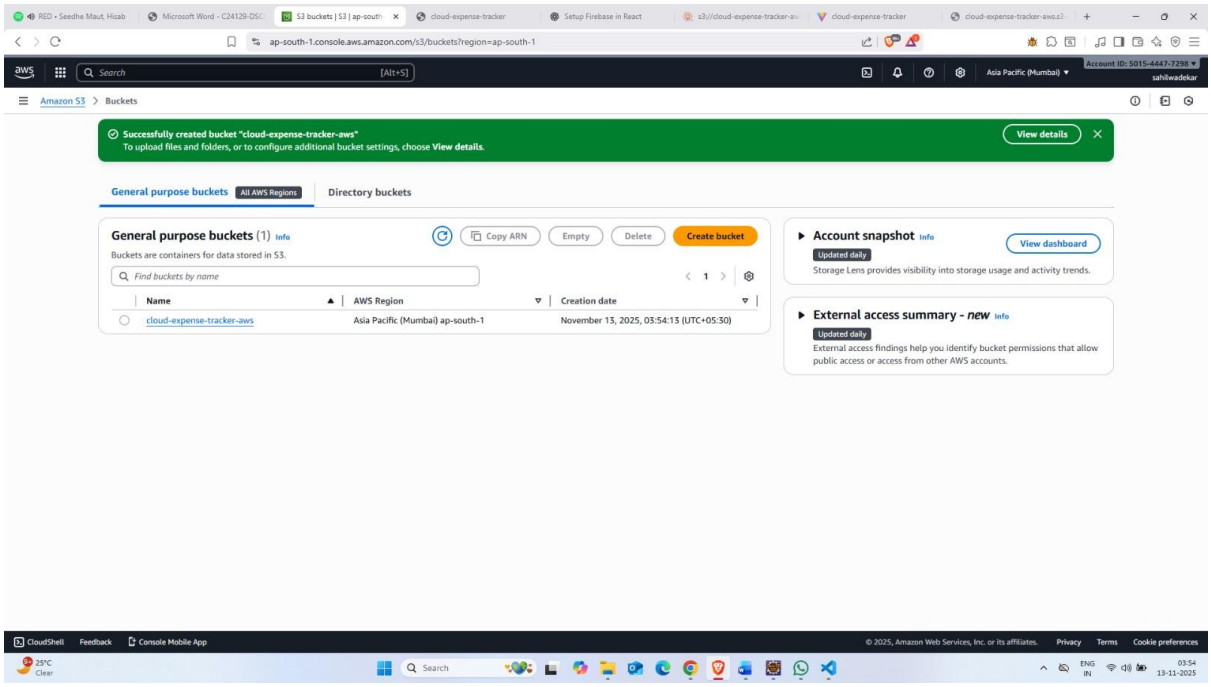
1. Implementation of Storage as a Service (STaaS) using Amazon S3 (Simple Storage Service)

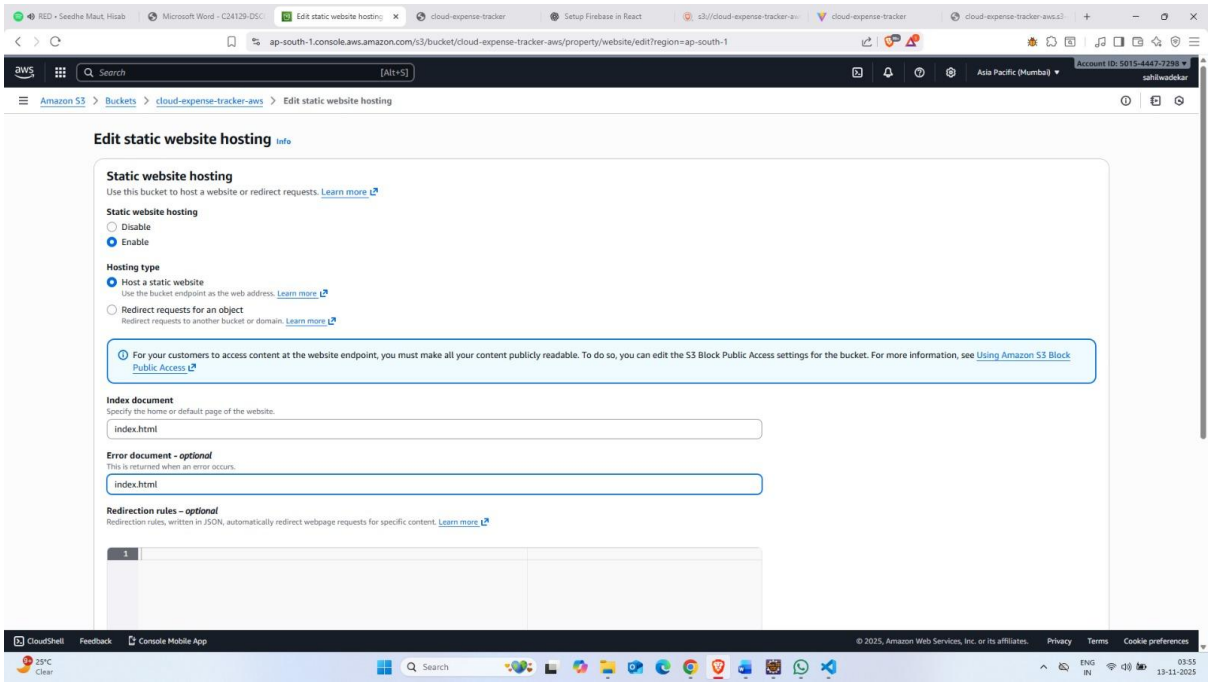
2. Implementation of Storage as a Service using Azure Blob Storage

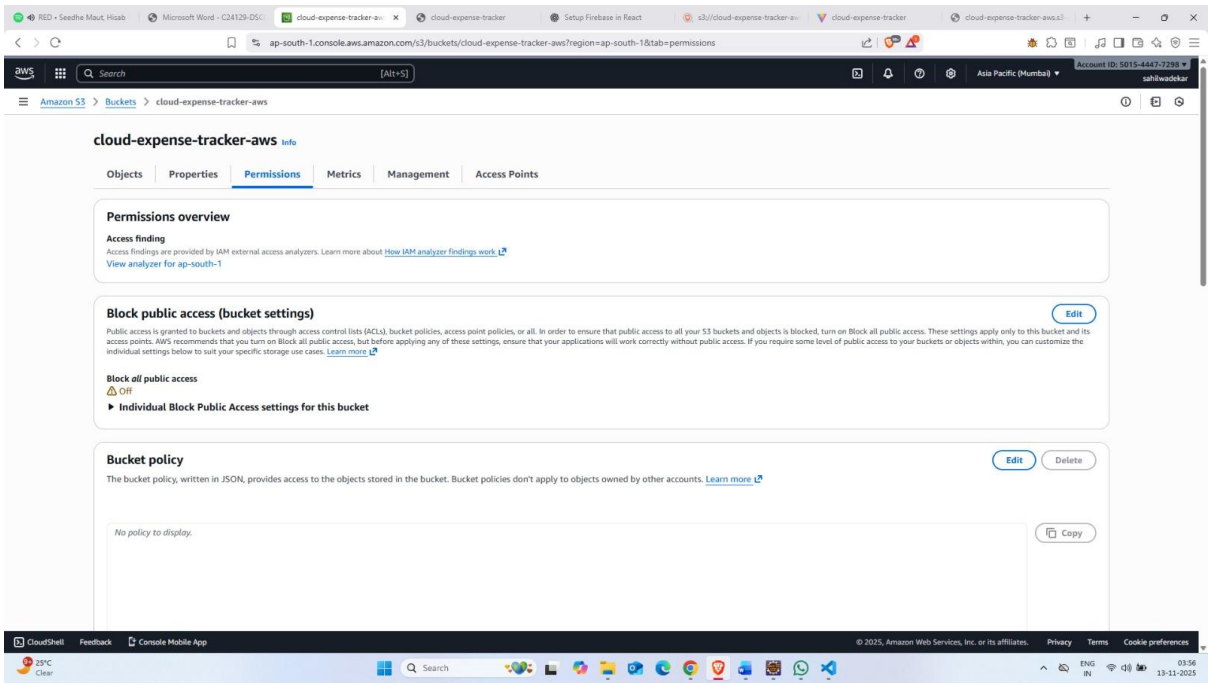


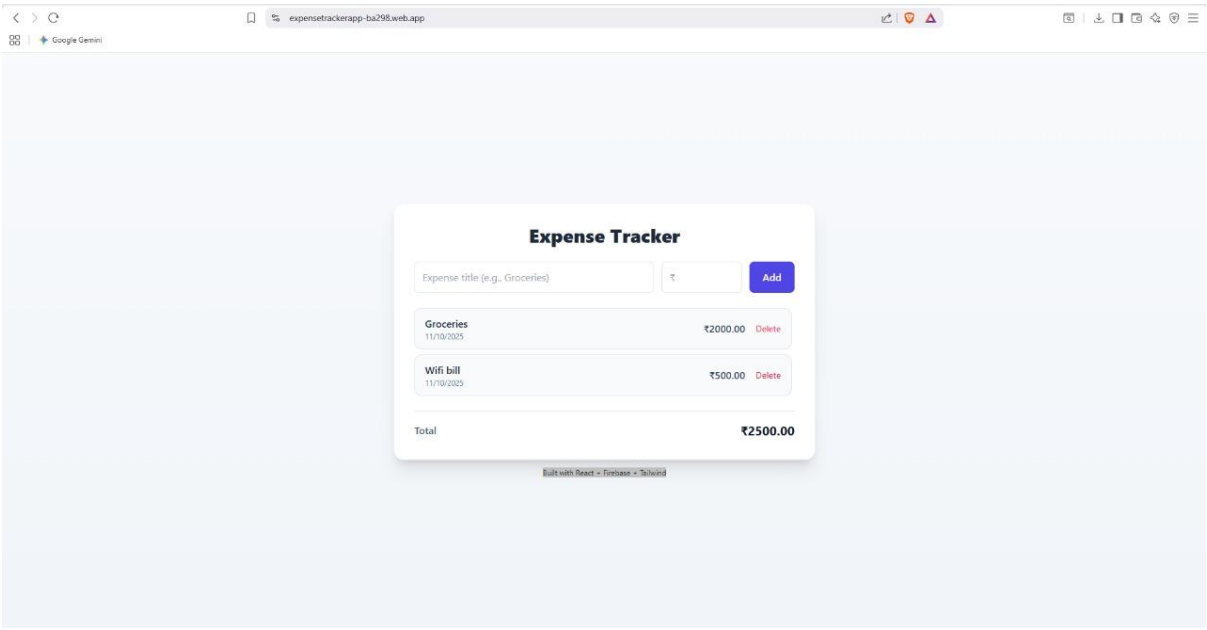



Amazon S3 > Buckets > Create bucket
Tags - optional (0)
You can use bucket tags to track storage costs and organize buckets. Learn more
No tags associated with this bucket.
Add new tag
You can add up to 50 tags.
Default encryption Info
Server-side encryption is automatically applied to new objects stored in this bucket.
Encryption type Info
Server-side encryption with Amazon S3 managed keys (SSE-S3)
Server-side encryption with AWS Key Management Service keys (SSE-KMS)
Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)
Bucket Key
Disable
Enable
Advanced settings
After creating the bucket, you can upload files and folders to the bucket, and configure additional bucket settings.
Cancel
Create bucket


Amazon S3 > Buckets
Successfully created bucket "cloud-expense-tracker-aws"
To upload files and folders, or to configure additional bucket settings, choose View details.
View details
General purpose buckets
Directory buckets
General purpose buckets (1) Info
Buckets are containers for data stored in S3.
Copy ARN
Empty
Delete
Create bucket
Find buckets by name
Name
AWS Region
Creation date
cloud-expense-tracker-aws
Asia Pacific (Mumbai) ap-south-1
November 13, 2025, 03:54:13 (UTC+05:30)
Account snapshot Info
View dashboard
Storage Lens provides visibility into storage usage and activity trends.
External access summary - new Info
External access findings help you identify bucket permissions that allow public access or access from other AWS accounts.


Edit static website hosting
Static website hosting
Use this bucket to host a website or redirect requests. Learn more
Static website hosting
Disable
Enable
Hosting type
Host a static website
Use the bucket endpoint as the web address. Learn more
Redirect requests for an object
Redirect requests to another bucket or domain. Learn more
For your customers to access content at the website endpoint, you must make all your content publicly readable. To do so, you can edit the S3 Block Public Access settings for the bucket. For more information, see Using Amazon S3 Block Public Access
Index document
Specify the home or default page of the website.
index.html
Error document - optional
This is returned when an error occurs.
index.html
Redirection rules – optional
Redirection rules, written in JSON, automatically redirect webpage requests for specific content. Learn more


cloud-expense-tracker-aws
Objects
Properties
Permissions
Metrics
Management
Access Points
Permissions overview
Access finding
Access findings are provided by IAM external access analyzers. Learn more about How IAM analyzer findings work
View analyzer for ap-south-1
Block public access (bucket settings)
Edit
Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to all your S3 buckets and objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to your buckets or objects within, you can customize the individual settings below to suit your specific storage use cases. Learn more
Block all public access
Off
Individual Block Public Access settings for this bucket
Bucket policy
Edit
Delete
The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. Learn more
No policy to display.
Copy

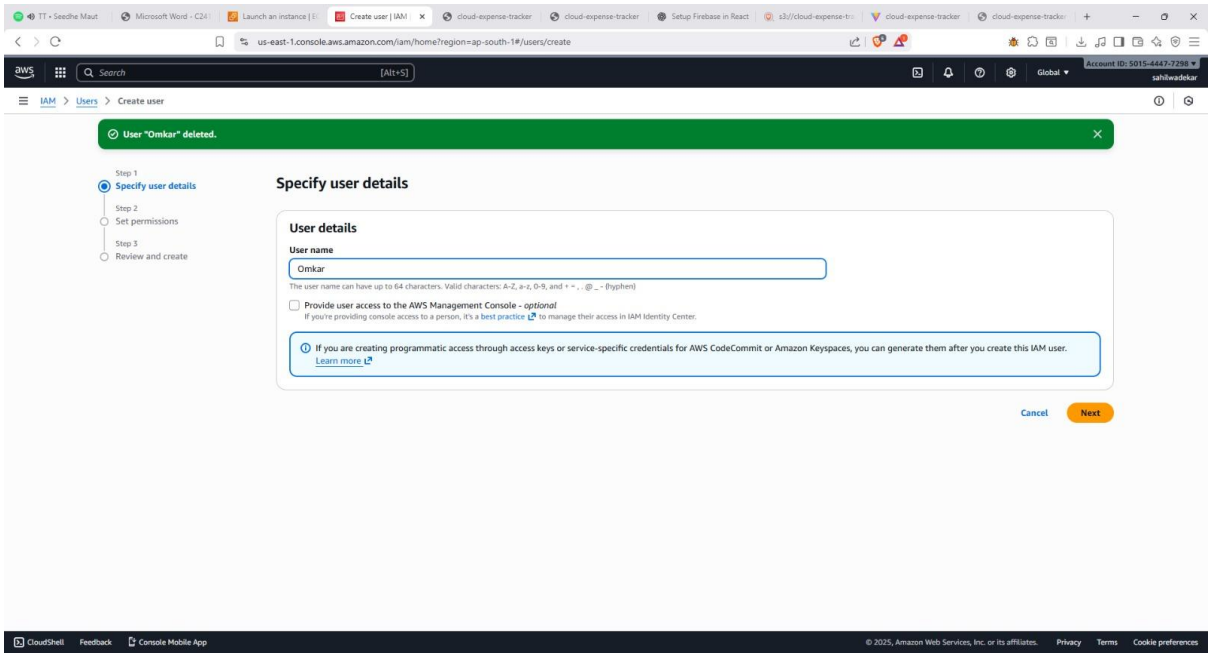

expensetrackerapp-ba298.web.app
Google Gemini
Expense Tracker
Expense title (e.g., Groceries)
₹
Add
Groceries
11/10/2025
₹2000.00
Delete
Wifi bill
11/10/2025
₹500.00
Delete
Total
₹2500.00
Built with React + Firebase + Tailwind

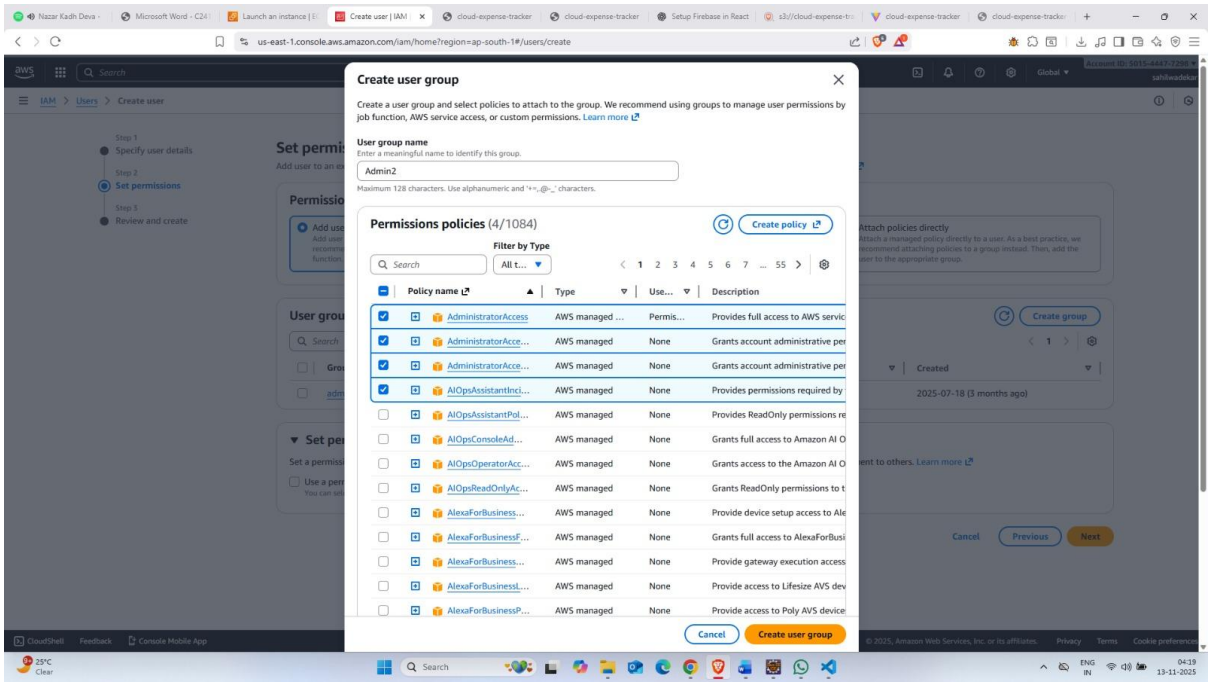
PRACTICAL NO. 11

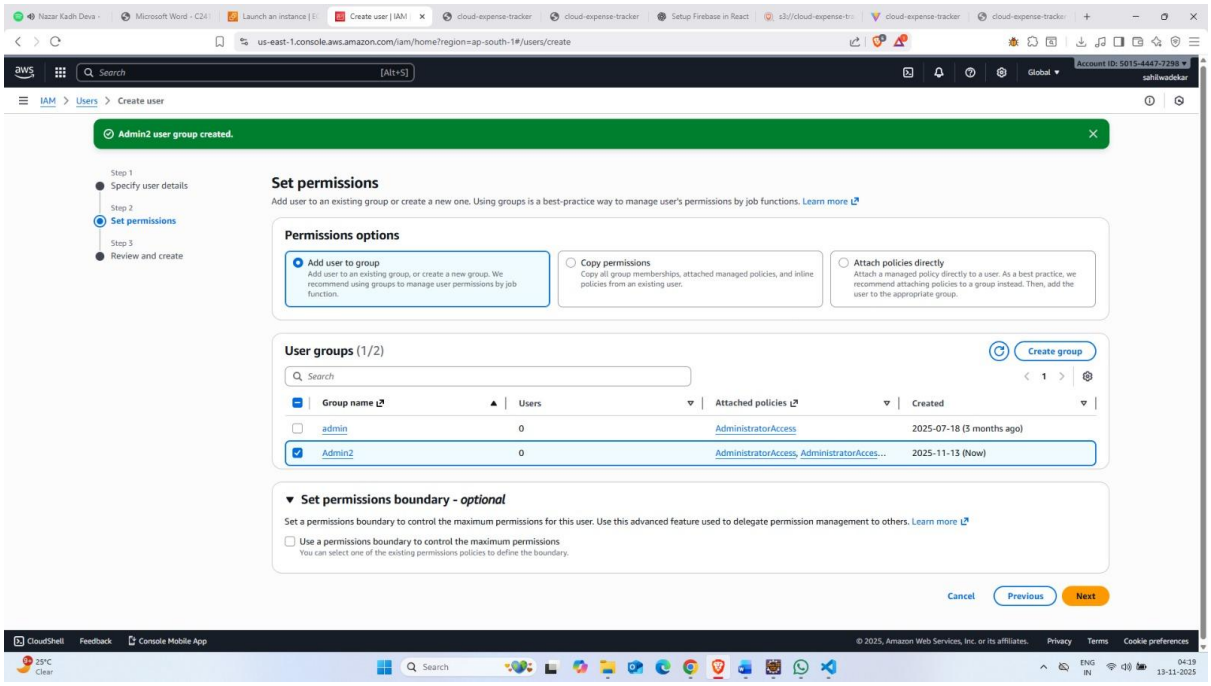
Implementation of Identity Management.

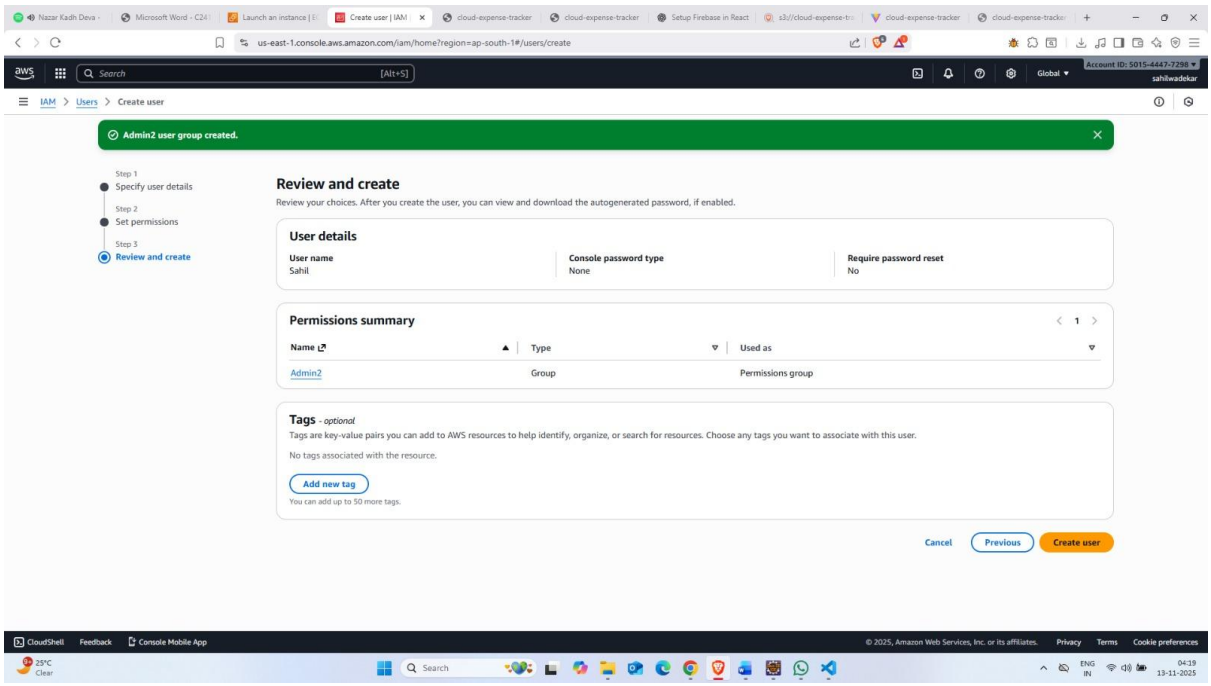
1. Implementation of Identity and Access Management (IAM) in AWS

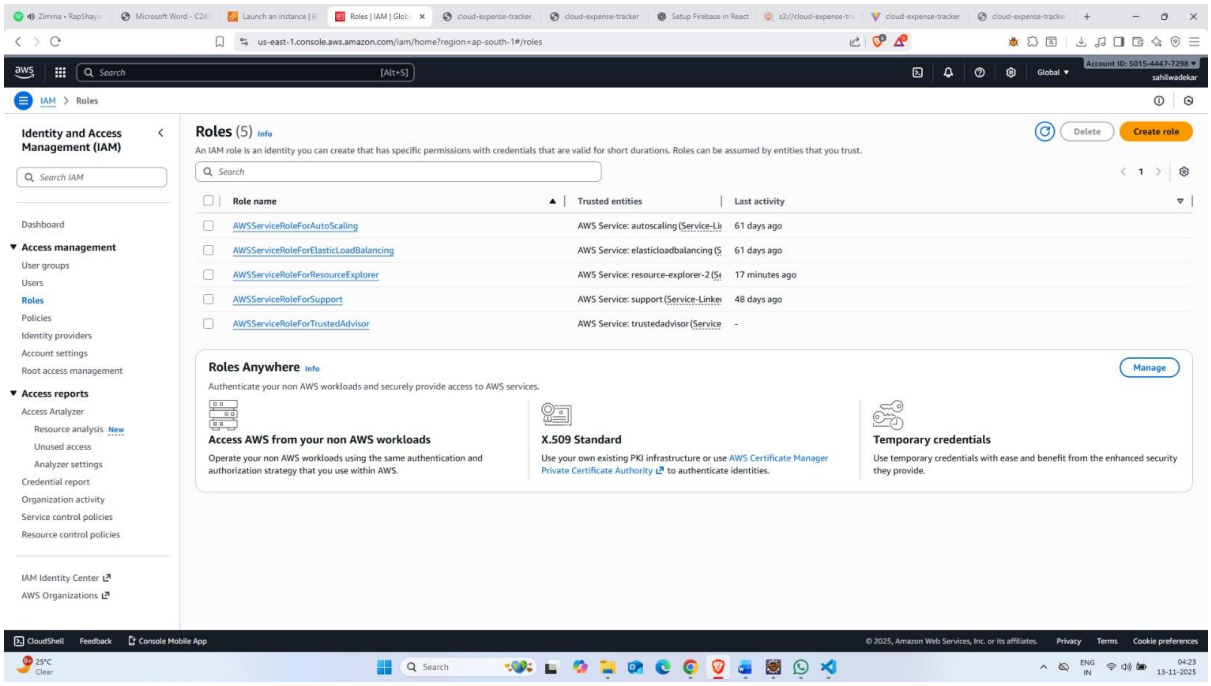
2. Implementation of Identity and Access Management (IAM) in Microsoft Azure

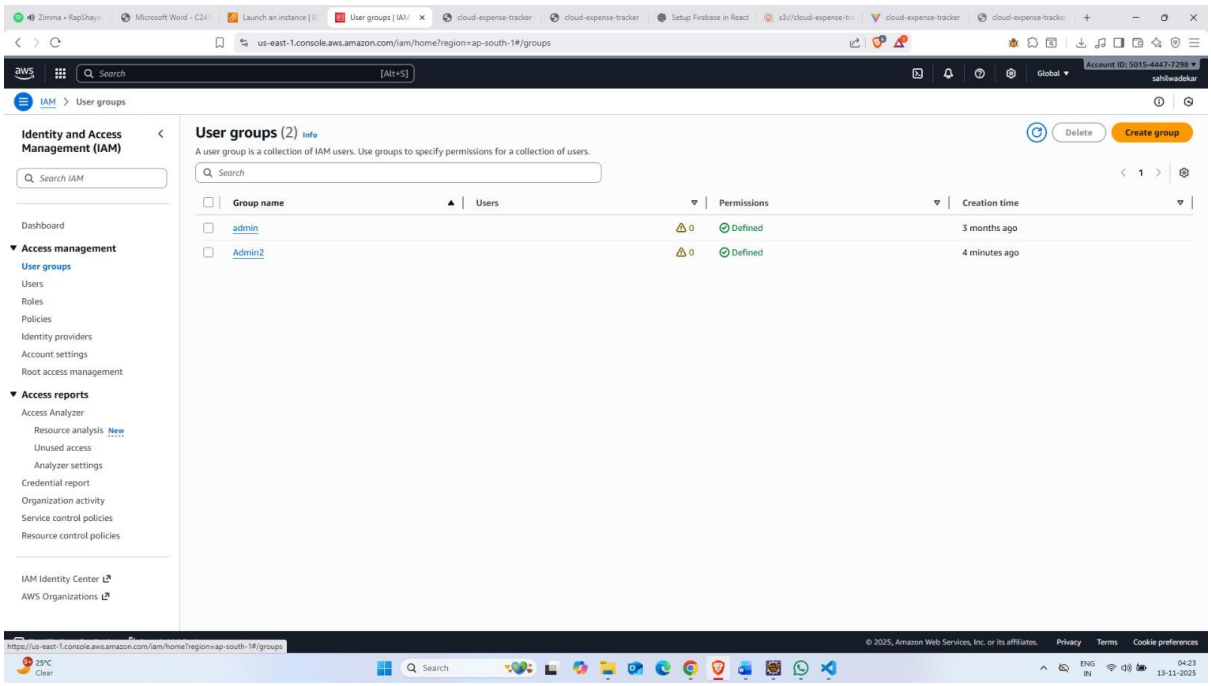


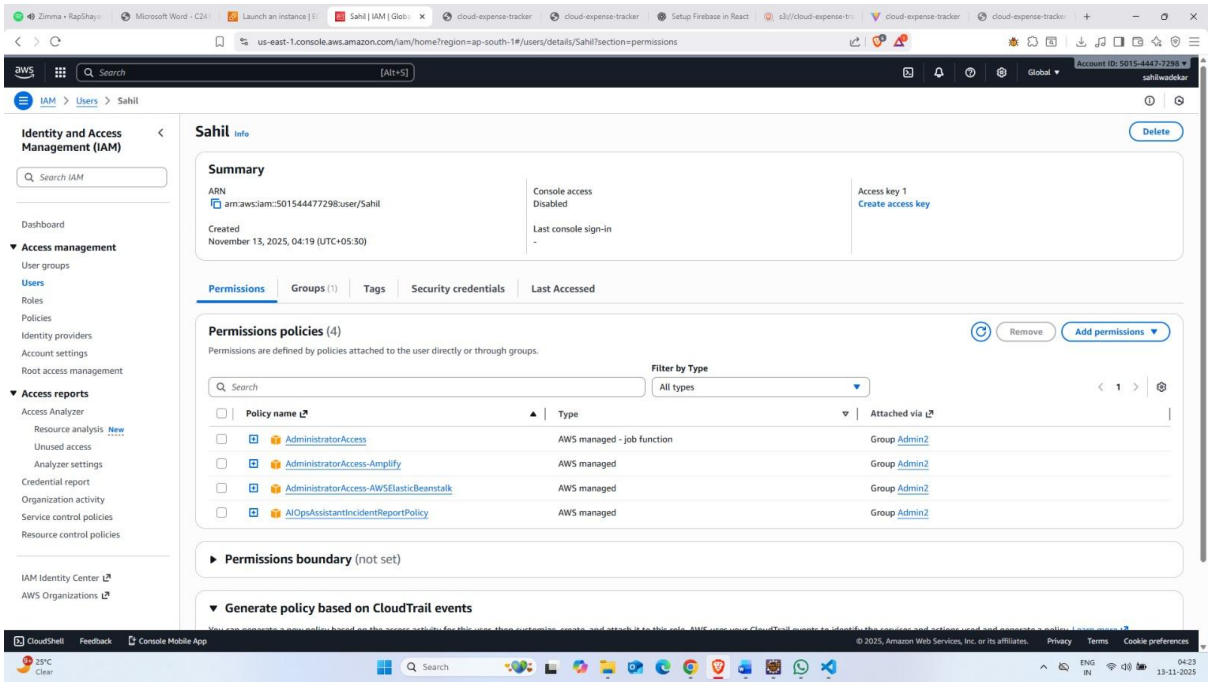



Admin2 user group created.
Step 1
Specify user details
Step 2
Set permissions
Step 3
Review and create
Set permissions
Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. Learn more
Permissions options
Add user to group
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.
Copy permissions
Copy all group memberships, attached managed policies, and inline policies from an existing user.
Attach policies directly
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.
User groups (1/2)
Create group
Group name
Users
Attached policies
Created
admin
0
AdministratorAccess
2025-07-18 (3 months ago)
Admin2
0
AdministratorAccess, AdministratorAcces...
2025-11-13 (Now)
Set permissions boundary - optional
Set a permissions boundary to control the maximum permissions for this user. Use this advanced feature used to delegate permission management to others. Learn more
Use a permissions boundary to control the maximum permissions
Cancel
Previous
Next


Admin2 user group created.
Step 1
Specify user details
Step 2
Set permissions
Step 3
Review and create
Review and create
Review your choices. After you create the user, you can view and download the autogenerated password, if enabled.
User details
User name
Sahil
Console password type
None
Require password reset
No
Permissions summary
Name
Type
Used as
Admin2
Group
Permissions group
Tags - optional
Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.
No tags associated with the resource.
Add new tag
You can add up to 50 more tags.
Cancel
Previous
Create user


us-east-1.console.aws.amazon.com/iam/home?region=ap-south-1#/roles
IAM > Roles
Identity and Access Management (IAM)
Search IAM
Dashboard
Access management
User groups
Users
Roles
Policies
Identity providers
Account settings
Root access management
Access reports
Access Analyzer
Resource analysis New
Unused access
Analyzer settings
Credential report
Organization activity
Service control policies
Resource control policies
IAM Identity Center
AWS Organizations
Roles (5) Info
An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.
Delete
Create role
Role name
Trusted entities
Last activity
AWSServiceRoleForAutoScaling
AWS Service: autoscaling (Service-Li
61 days ago
AWSServiceRoleForElasticLoadBalancing
AWS Service: elasticloadbalancing (S
61 days ago
AWSServiceRoleForResourceExplorer
AWS Service: resource-explorer-2 (Se
17 minutes ago
AWSServiceRoleForSupport
AWS Service: support (Service-Linke
48 days ago
AWSServiceRoleForTrustedAdvisor
AWS Service: trustedadvisor (Service
-
Roles Anywhere Info
Manage
Authenticate your non AWS workloads and securely provide access to AWS services.
Access AWS from your non AWS workloads
Operate your non AWS workloads using the same authentication and authorization strategy that you use within AWS.
X.509 Standard
Use your own existing PKI infrastructure or use AWS Certificate Manager Private Certificate Authority to authenticate identities.
Temporary credentials
Use temporary credentials with ease and benefit from the enhanced security they provide.
CloudShell
Feedback
Console Mobile App
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

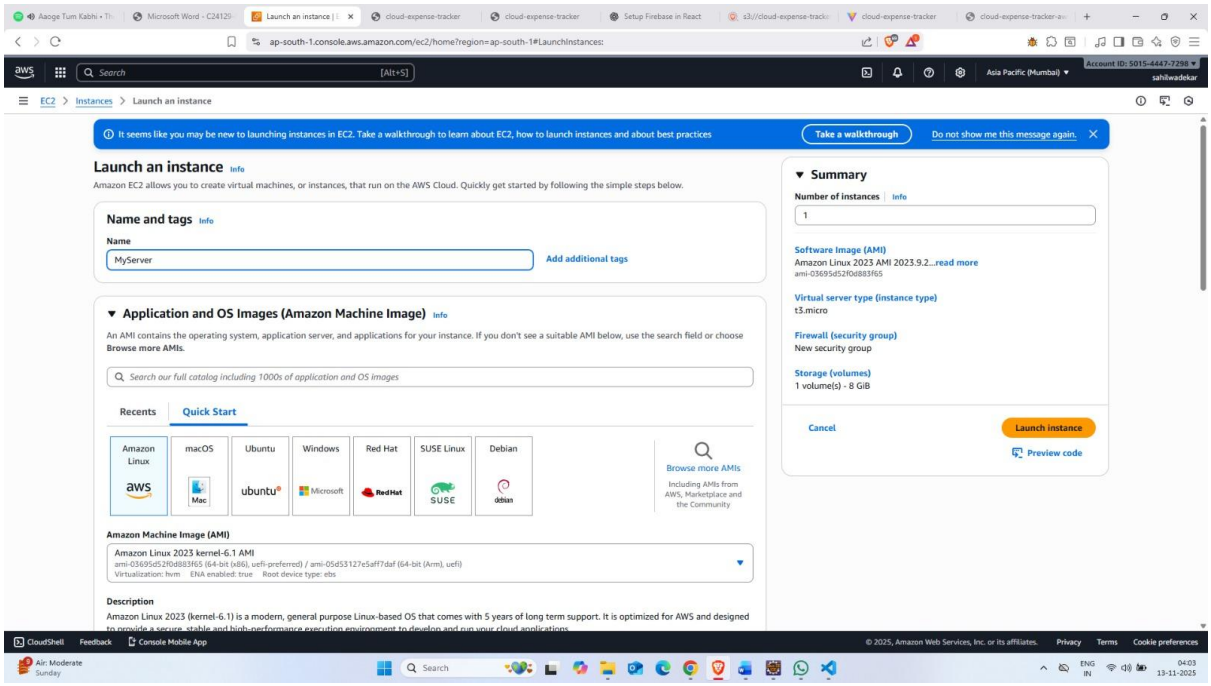

us-east-1.console.aws.amazon.com/iam/home?region=ap-south-1#/groups
IAM > User groups
Identity and Access Management (IAM)
Search IAM
Dashboard
Access management
User groups
Users
Roles
Policies
Identity providers
Account settings
Root access management
Access reports
Access Analyzer
Resource analysis New
Unused access
Analyzer settings
Credential report
Organization activity
Service control policies
Resource control policies
IAM Identity Center
AWS Organizations
User groups (2) Info
A user group is a collection of IAM users. Use groups to specify permissions for a collection of users.
Delete
Create group
Group name
Users
Permissions
Creation time
admin
0
Defined
3 months ago
Admin2
0
Defined
4 minutes ago
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

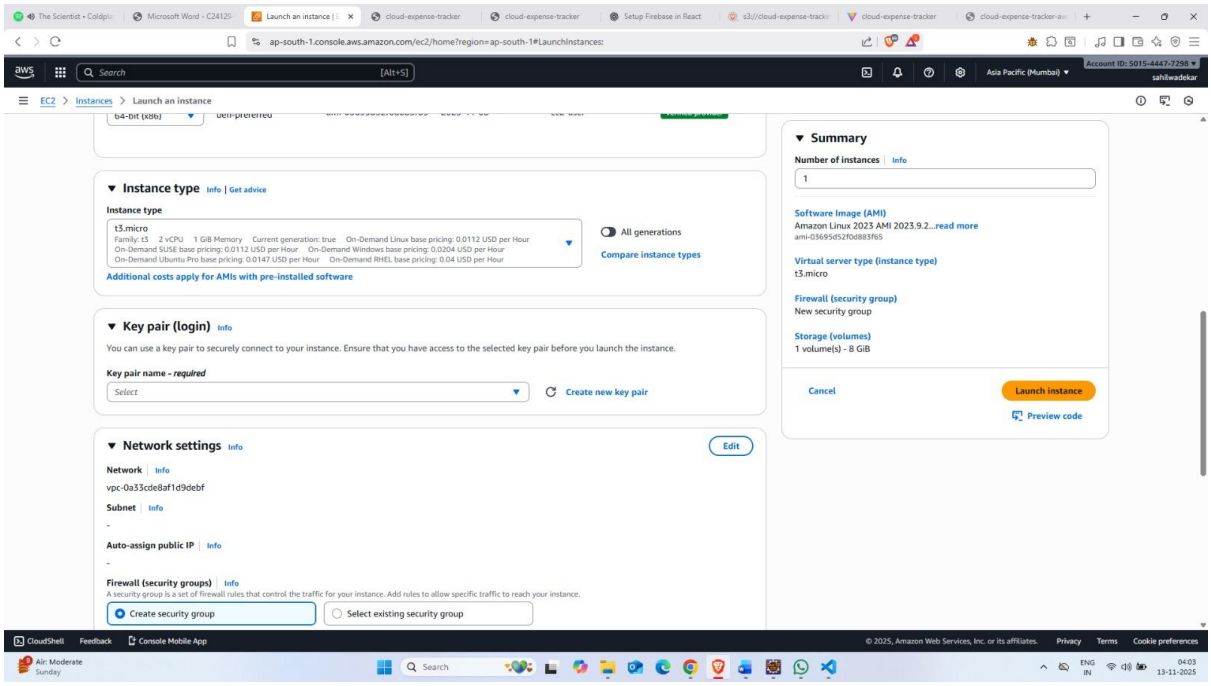

us-east-1.console.aws.amazon.com/iam/home?region=ap-south-1#/users/details/Sahil?section=permissions
IAM > Users > Sahil
Identity and Access Management (IAM)
Search IAM
Dashboard
Access management
User groups
Users
Roles
Policies
Identity providers
Account settings
Root access management
Access reports
Access Analyzer
Resource analysis New
Unused access
Analyzer settings
Credential report
Organization activity
Service control policies
Resource control policies
IAM Identity Center
AWS Organizations
Sahil Info
Delete
Summary
ARN
arn:aws:iam::501544477298:user/Sahil
Console access
Disabled
Access key 1
Create access key
Created
November 13, 2025, 04:19 (UTC+05:30)
Last console sign-in
-
Permissions
Groups (1)
Tags
Security credentials
Last Accessed
Permissions policies (4)
Permissions are defined by policies attached to the user directly or through groups.
Remove
Add permissions
Filter by Type
Search
All types
Policy name
Type
Attached via
AdministratorAccess
AWS managed - job function
Group Admin2
AdministratorAccess-Amplify
AWS managed
Group Admin2
AdministratorAccess-AWSElasticBeanstalk
AWS managed
Group Admin2
AIOpsAssistantIncidentReportPolicy
AWS managed
Group Admin2
Permissions boundary (not set)
Generate policy based on CloudTrail events
CloudShell
Feedback
Console Mobile App
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

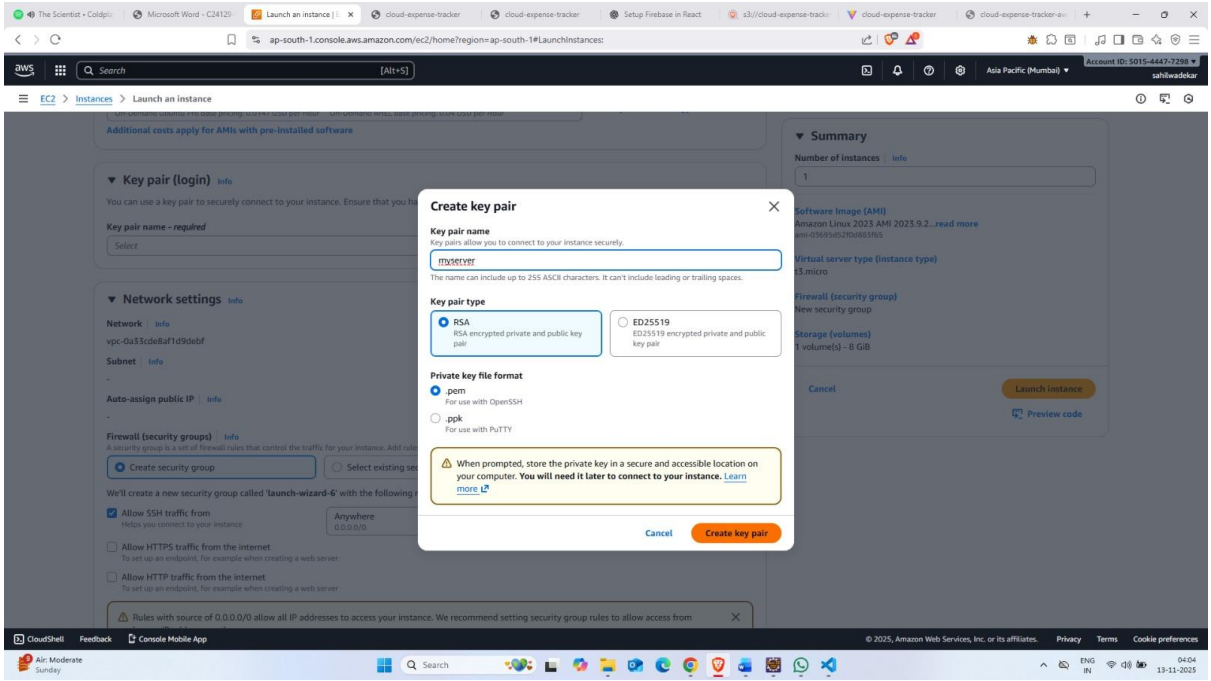
PRACTICAL NO.12

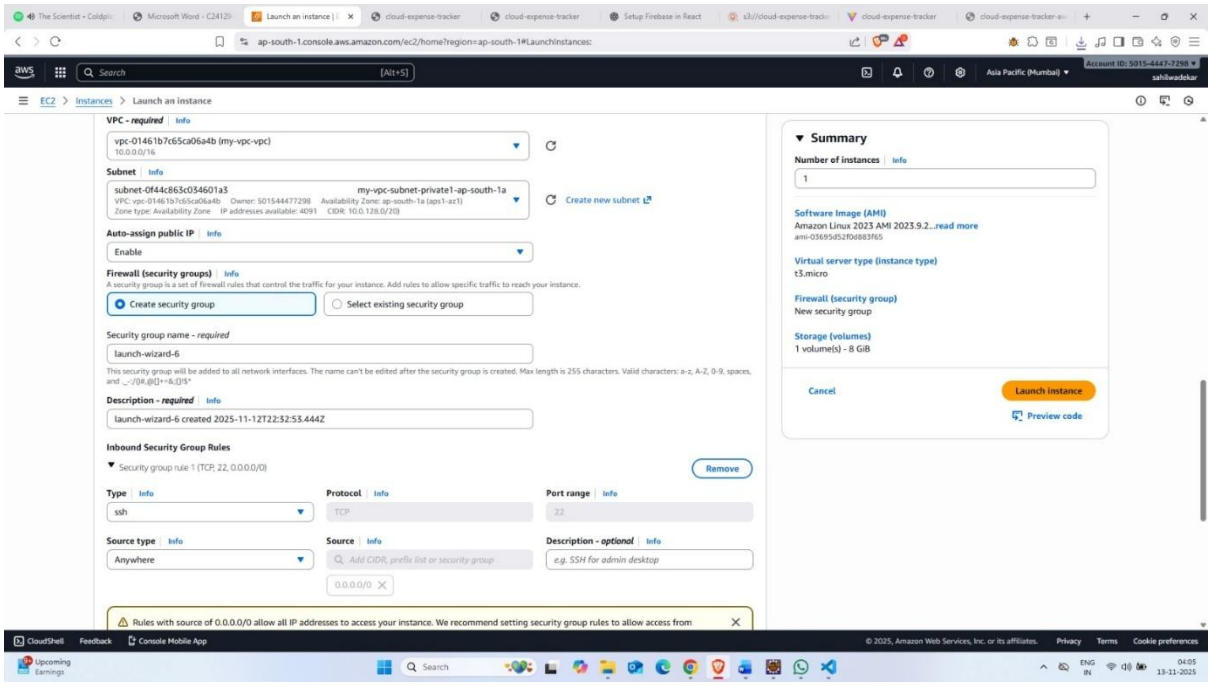
Create a virtual machine (VM) on any cloud provider (AWS/Azure/GCP) of your choice with the specifications:

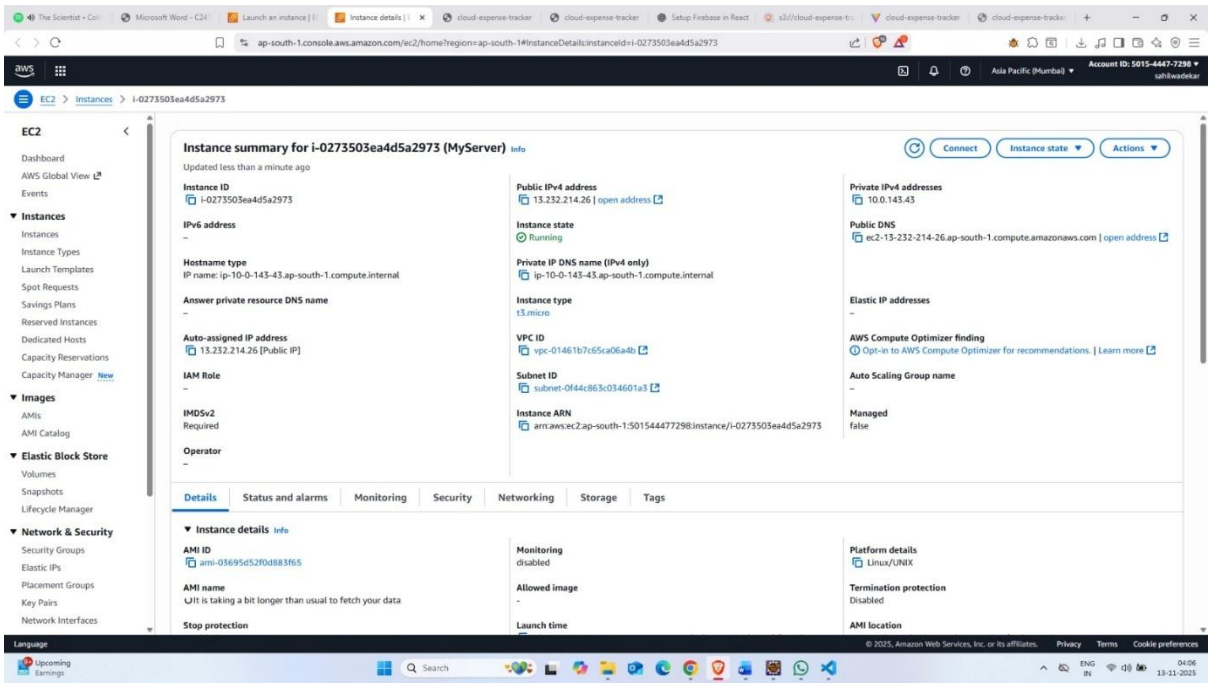
- Operating System, VM Type, Disk Size, Public IP, Network Rules
- Once created, verify that the VM is running and submit a screenshot of the instance details and a brief description of the steps you followed.
  - Create a virtual machine (VM) on any cloud provider (AWS)
  - Create a virtual machine (VM) on any cloud provider (Azure)




Launch an instance
Summary
Number of instances
Instance type
Key pair (login)
Network settings
Edit
Create security group
Select existing security group
Cancel
Launch instance
Preview code

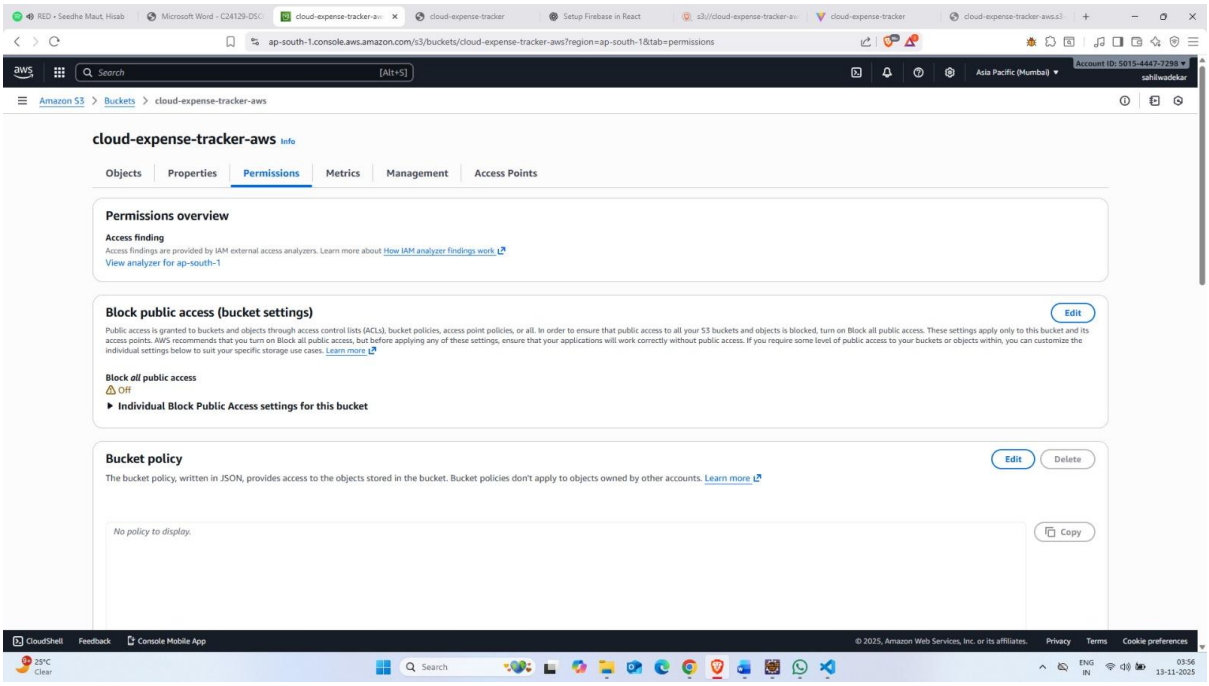

Create key pair
Key pair name
myserver
Key pair type
RSA
ED25519
Private key file format
.pem
.ppk
When prompted, store the private key in a secure and accessible location on your computer. You will need it later to connect to your instance.
Cancel
Create key pair

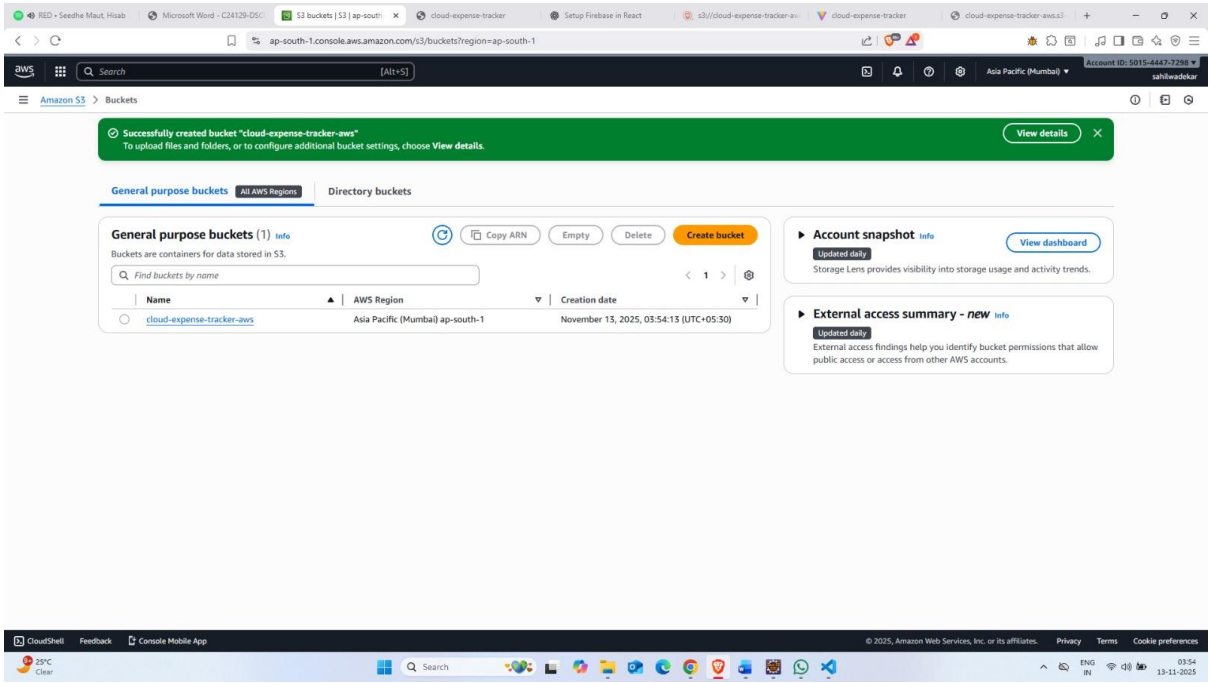

Launch an instance
EC2 > Instances > Launch an instance
VPC - required
vpc-01461b7c65ca06a4b (my-vpc-vpc)
10.0.0.0/16
Subnet
subnet-0f44c863c034601a3
my-vpc-subnet-private1-ap-south-1a
Create new subnet
Auto-assign public IP
Enable
Firewall (security groups)
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.
Create security group
Select existing security group
Security group name - required
launch-wizard-6
Description - required
launch-wizard-6 created 2025-11-12T22:52:53.444Z
Inbound Security Group Rules
Security group rule 1 (TCP, 22, 0.0.0.0/0)
Remove
Type
ssh
Protocol
TCP
Port range
22
Source type
Anywhere
Source
0.0.0.0/0
Description - optional
e.g. SSH for admin desktop
Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from
Summary
Number of instances
1
Software Image (AMI)
Amazon Linux 2023 AMI 2023.9.2...read more
ami-03695d52f0d883f65
Virtual server type (instance type)
t3.micro
Firewall (security group)
New security group
Storage (volumes)
1 volume(s) - 8 GiB
Cancel
Launch instance
Preview code
CloudShell
Feedback
Console Mobile App
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

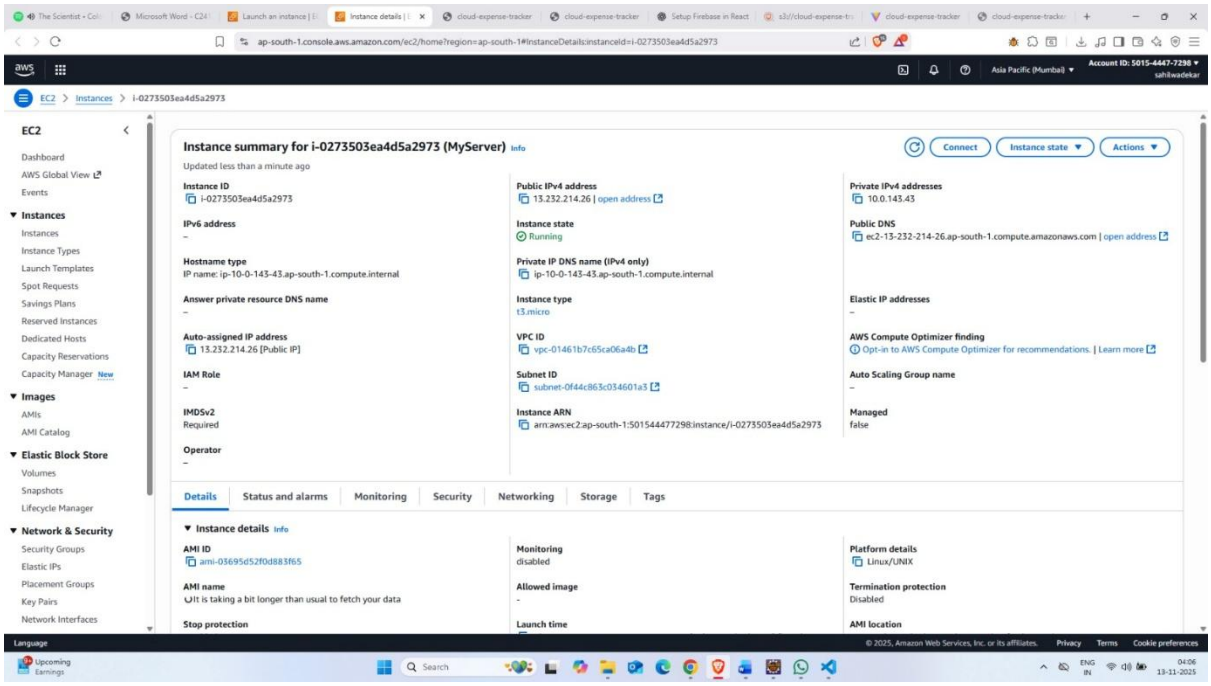

EC2 > Instances > i-0273503ea4d5a2973
Instance summary for i-0273503ea4d5a2973 (MyServer)
Updated less than a minute ago
Connect
Instance state
Actions
Instance ID
i-0273503ea4d5a2973
Public IPv4 address
13.232.214.26 | open address
Private IPv4 addresses
10.0.143.43
IPv6 address
Instance state
Running
Public DNS
ec2-13-232-214-26.ap-south-1.compute.amazonaws.com | open address
Hostname type
IP name: ip-10-0-143-43.ap-south-1.compute.internal
Private IP DNS name (IPv4 only)
ip-10-0-143-43.ap-south-1.compute.internal
Answer private resource DNS name
Instance type
t3.micro
Elastic IP addresses
Auto-assigned IP address
13.232.214.26 [Public IP]
VPC ID
vpc-01461b7c65ca06a4b
AWS Compute Optimizer finding
Opt-in to AWS Compute Optimizer for recommendations. | Learn more
IAM Role
Subnet ID
subnet-0f44c863c034601a3
Auto Scaling Group name
IMDSv2
Required
Instance ARN
arn:aws:ec2:ap-south-1:501544477298:instance/i-0273503ea4d5a2973
Managed
false
Operator
Details
Status and alarms
Monitoring
Security
Networking
Storage
Tags
Instance details
AMI ID
ami-03695d52f0d883f65
Monitoring
disabled
Platform details
Linux/UNIX
AMI name
It is taking a bit longer than usual to fetch your data
Allowed image
Termination protection
Disabled
Stop protection
Launch time
AMI location
EC2
Dashboard
AWS Global View
Events
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations
Capacity Manager
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

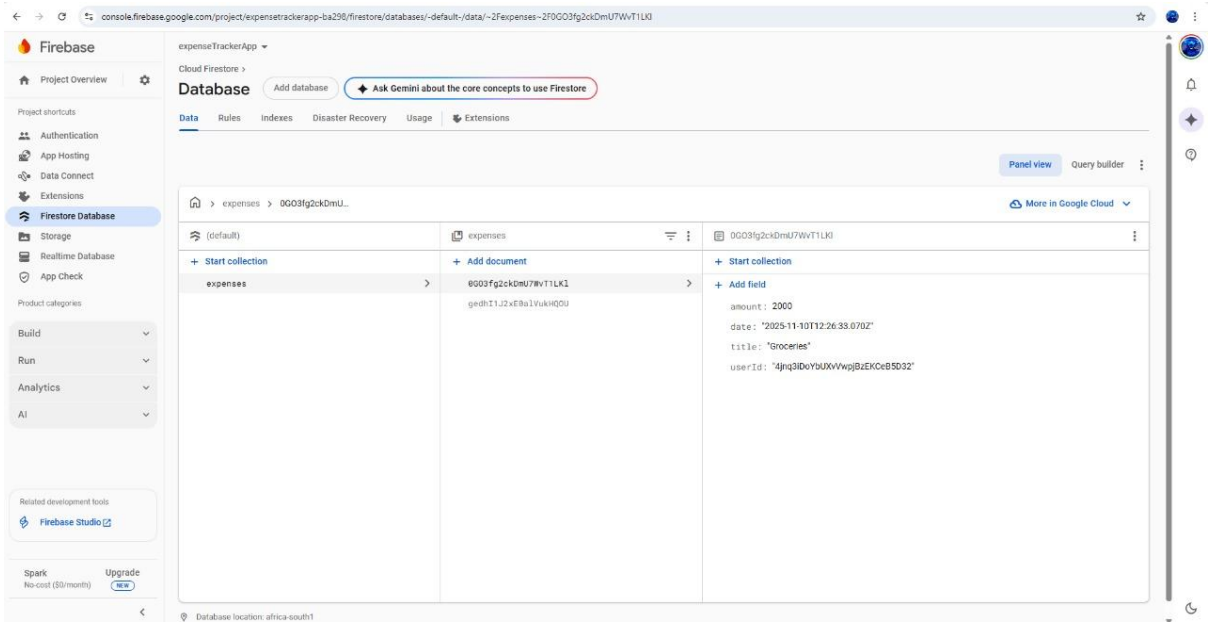
PRACTICAL NO. 13

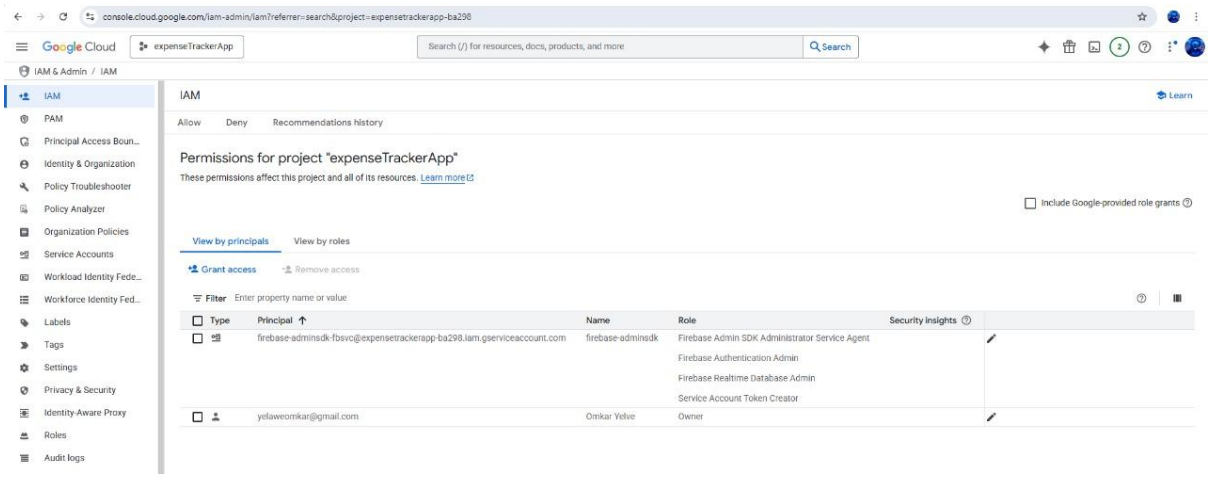
Group projects (2 to 3 members) are to be given the opportunity to work on any Cloud Concept.

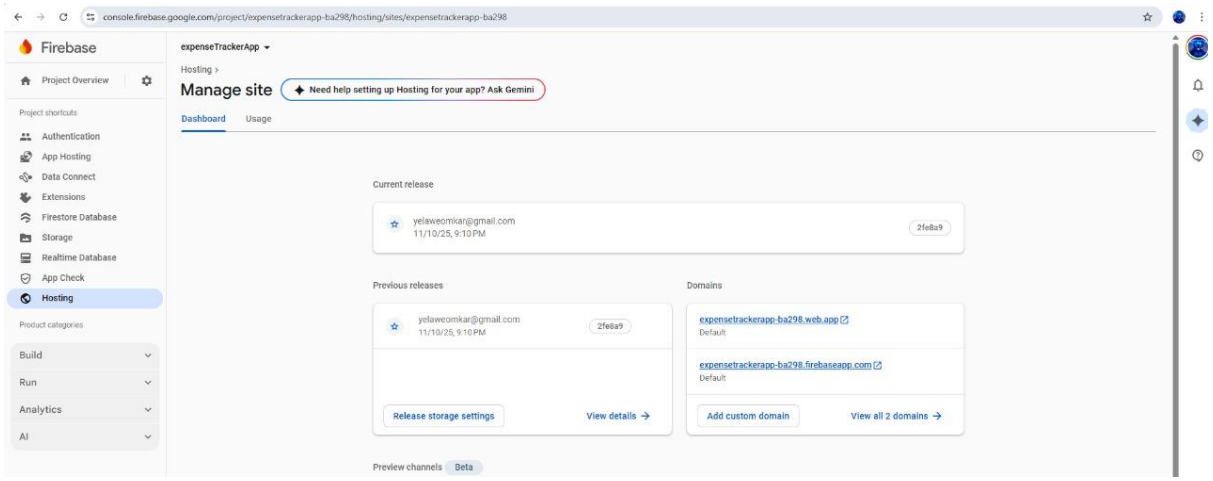


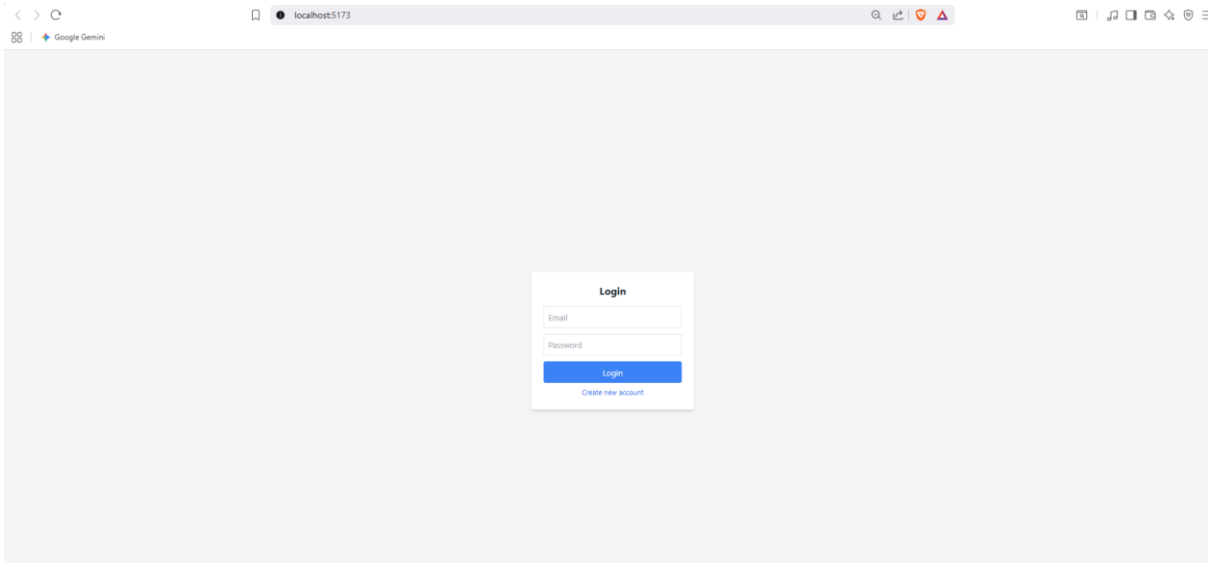


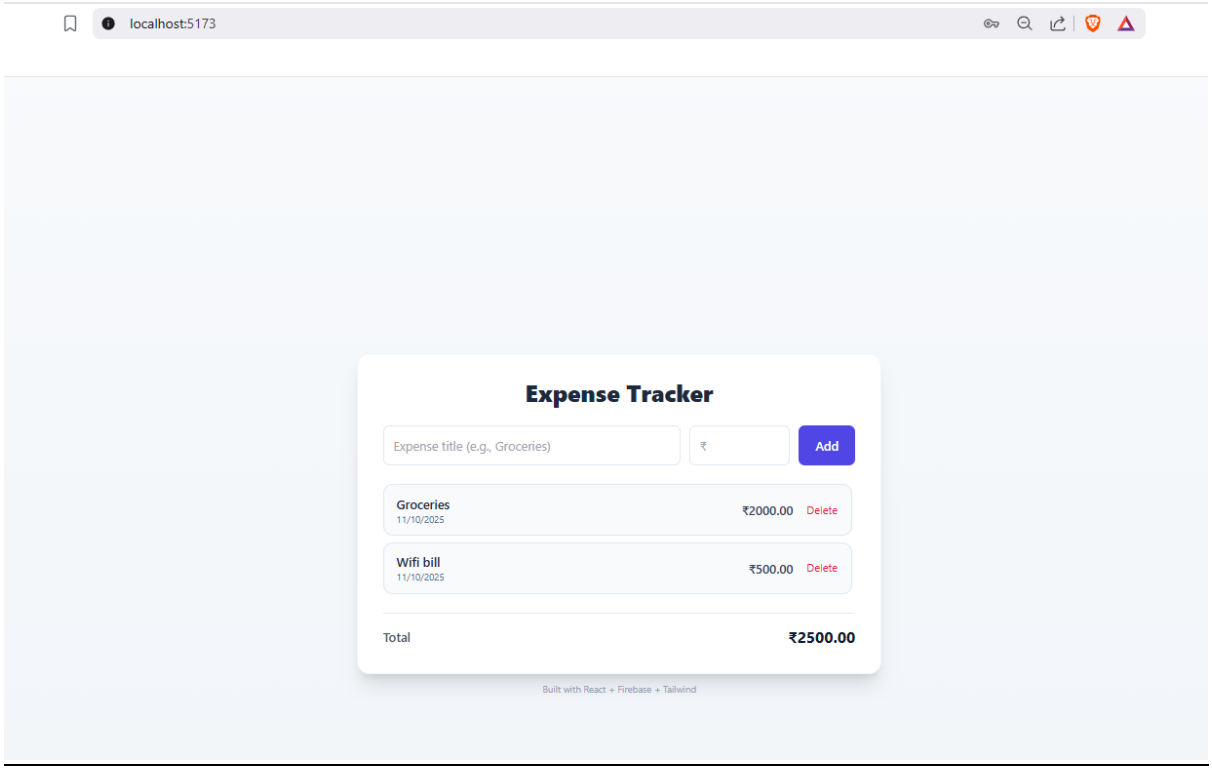

Instance summary for i-0273503ea4d5a2973 (MyServer)
Connect
Instance state
Actions
Instance ID
i-0273503ea4d5a2973
Public IPv4 address
13.232.214.26 | open address
Private IPv4 addresses
10.0.143.43
IPv6 address
Instance state
Running
Public DNS
ec2-13-232-214-26.ap-south-1.compute.amazonaws.com | open address
Hostname type
IP name: ip-10-0-143-43.ap-south-1.compute.internal
Private IP DNS name (IPv4 only)
ip-10-0-143-43.ap-south-1.compute.internal
Answer private resource DNS name
Instance type
t3.micro
Elastic IP addresses
Auto-assigned IP address
13.232.214.26 [Public IP]
VPC ID
vpc-01461b7c65ca06a4b
AWS Compute Optimizer finding
Opt-in to AWS Compute Optimizer for recommendations. | Learn more
IAM Role
Subnet ID
subnet-0f44c863c034601a3
Auto Scaling Group name
IMDSv2
Required
Instance ARN
arn:aws:ec2:ap-south-1:501544477298:instance/i-0273503ea4d5a2973
Managed
false
Operator
Details
Status and alarms
Monitoring
Security
Networking
Storage
Tags
Instance details
AMI ID
ami-03695d52f0d883f65
Monitoring
disabled
Platform details
Linux/UNIX
AMI name
It is taking a bit longer than usual to fetch your data
Allowed image
Termination protection
Disabled
Stop protection
Launch time
AMI location


Firebase
Project Overview
expenseTrackerApp
Cloud Firestore
Database
Add database
Ask Gemini about the core concepts to use Firestore
Data
Rules
Indexes
Disaster Recovery
Usage
Extensions
Panel view
Query builder
Authentication
App Hosting
Data Connect
Extensions
Firestore Database
Storage
Realtime Database
App Check
Build
Run
Analytics
AI
(default)
expenses
0GO3fg2ckDmU7WvT1LKl
Start collection
Add document
Add field
amount: 2000
date: "2025-11-10T12:26:33.070Z"
title: "Groceries"
userId: "4jnq3lDoYbUXvVwpjBzEKCeB5D32"
Spark
Upgrade
Database location: africa-south1


console.cloud.google.com/iam-admin/iam?referrer=search&project=expensetrackerapp-ba298
Google Cloud
expenseTrackerApp
Search (/) for resources, docs, products, and more
Search
IAM & Admin / IAM
IAM
PAM
Principal Access Boun...
Identity & Organization
Policy Troubleshooter
Policy Analyzer
Organization Policies
Service Accounts
Workload Identity Fede...
Workforce Identity Fed...
Labels
Tags
Settings
Privacy & Security
Identity-Aware Proxy
Roles
Audit logs
IAM
Learn
Allow
Deny
Recommendations history
Permissions for project "expenseTrackerApp"
These permissions affect this project and all of its resources. Learn more
Include Google-provided role grants
View by principals
View by roles
Grant access
Remove access
Filter
Enter property name or value
Type
Principal
Name
Role
Security insights
firebase-adminsdk-fbsvc@expensetrackerapp-ba298.iam.gserviceaccount.com
firebase-adminsdk
Firebase Admin SDK Administrator Service Agent
Firebase Authentication Admin
Firebase Realtime Database Admin
Service Account Token Creator
yelaweomkar@gmail.com
Omkar Yelve
Owner


console.firebase.google.com/project/expensetrackerapp-ba298/hosting/sites/expensetrackerapp-ba298
Firebase
expenseTrackerApp
Project Overview
Project shortcuts
Authentication
App Hosting
Data Connect
Extensions
Firestore Database
Storage
Realtime Database
App Check
Hosting
Product categories
Build
Run
Analytics
AI
Hosting >
Manage site
Need help setting up Hosting for your app? Ask Gemini
Dashboard
Usage
Current release
yelaweomkar@gmail.com
11/10/25, 9:10 PM
2fe8a9
Previous releases
Domains
expensetrackerapp-ba298.web.app
Default
expensetrackerapp-ba298.firebaseapp.com
Default
Release storage settings
View details
Add custom domain
View all 2 domains
Preview channels
Beta


localhost:5173
Google Gemini
Login
Email
Password
Login
Create new account


localhost:5173
Expense Tracker
Expense title (e.g., Groceries)
₹
Add
Groceries
11/10/2025
₹2000.00
Delete
Wifi bill
11/10/2025
₹500.00
Delete
Total
₹2500.00
Built with React + Firebase + Tailwind