PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE – 411043

Department of Electronics & Telecommunication Engineering

BATCH AND ROLL NO: P8 42405
EXPERIMENT NO.9
TITLE: Design a mobile app to store data using internal or external storage.
DATE OF PERFORMANCE
DATE OF CHECKING

Title: Design a mobile app to store data using internal or external storage.

Requirements:
1 Android studio

Theory:

Designing a mobile app to store data using internal or external storage involves creating an application that allows users to store and manage data on their mobile devices. Internal storage refers to the built-in storage on a device, while external storage can include removable SD cards or cloud-based storage solutions. To implement storage functionality in a mobile app, developers typically use platform-specific APIs and libraries, such as Android's SharedPreferences, SQLite database, or File API, or iOS's CoreData or File Manager API. These APIs provide a way to read, write, and manage data stored on the device or in the cloud.

When designing the user interface for a data storage app, developers need to consider how users will interact with the app and manage their data. This can involve creating screens and views that allow users to view, edit, and delete data.

Here are the general steps to design a mobile app to store data using internal or external storage:

1. Define the requirements for your app, including the types of data to be stored, the app's user interface, and any functionality or features that are needed.
2. Choose a development environment, such as Android Studio or Xcode, and create a new project.
3. Determine the appropriate storage method for your app, such as internal storage, external storage, or a cloud-based storage solution.
4. Implement the storage functionality using the appropriate API, such as Android's SharedPreferences, SQLite database, or File API, or iOS's CoreData or File Manager API.
5. Design the user interface for your app, including any screens or views needed to display and manage the stored data.
6. Implement the UI using the appropriate UI components, such as TextViews, EditTexts, and RecyclerViews, and link the UI to the storage functionality using appropriate callbacks or data bindings.
7. Test the app on a physical device or emulator, and make any necessary adjustments to the UI or functionality.

8. Optimize the app's performance and stability, using tools such as Android Profiler or Xcode's Instruments.
9. Add any additional features or functionality as needed, based on user feedback or evolving requirements.
10. Publish the app to the Google Play Store or Apple App Store, following the appropriate guidelines and requirements.

Code:

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"
   tools:context=".MainActivity">
   <TextView
      android:id="@+id/textView2"
      android:layout_width="337dp"
      android:layout_height="28dp"
      android:text=" File Content "
      android:textAlignment="center"
      android:textColor="#000"
      android:textSize="24sp"
      android:textStyle="bold"
      app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintLeft_toLeftOf="parent"
      app:layout_constraintRight_toRightOf="parent"
      app:layout_constraintTop_toTopOf="parent"
      app:layout_constraintVertical_bias="0.52" />
   <Button
      android:id="@+id/write_button"
      android:layout_width="wrap_content"
      android:layout_height="48dp"
      android:layout_marginStart="160dp"
      android:layout_marginEnd="159dp"
      android:layout_marginBottom="16dp"
      android:text="Write"
      app:layout_constraintBottom_toTopOf="@+id/read_button"
      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="0.904" />
   <Button
      android:id="@+id/read_button"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginStart="160dp"
      android:layout_marginEnd="158dp"
      android:layout_marginBottom="48dp"
      android:text="Read"
      app:layout_constraintBottom_toTopOf="@+id/textView2"
      app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintHorizontal_bias="0.0"
      app:layout_constraintStart_toStartOf="parent" />
```

```
    <EditText
        android:id="@+id/userInput"
        android:layout_width="319dp"
        android:layout_height="50dp"
        android:layout_marginStart="46dp"
        android:layout_marginTop="91dp"
        android:layout_marginEnd="46dp"
        android:ems="10"
        android:inputType="textPersonName"
        android:text="Name"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/content"
        android:layout_width="332dp"
        android:layout_height="306dp"
        android:layout_marginStart="33dp"
        android:layout_marginTop="21dp"
        android:layout_marginEnd="33dp"
        android:layout_marginBottom="6dp"
        android:text=""
        android:textAlignment="center"
        android:textColor="#000"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.461"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView2"
        app:layout_constraintVertical_bias="0.0" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

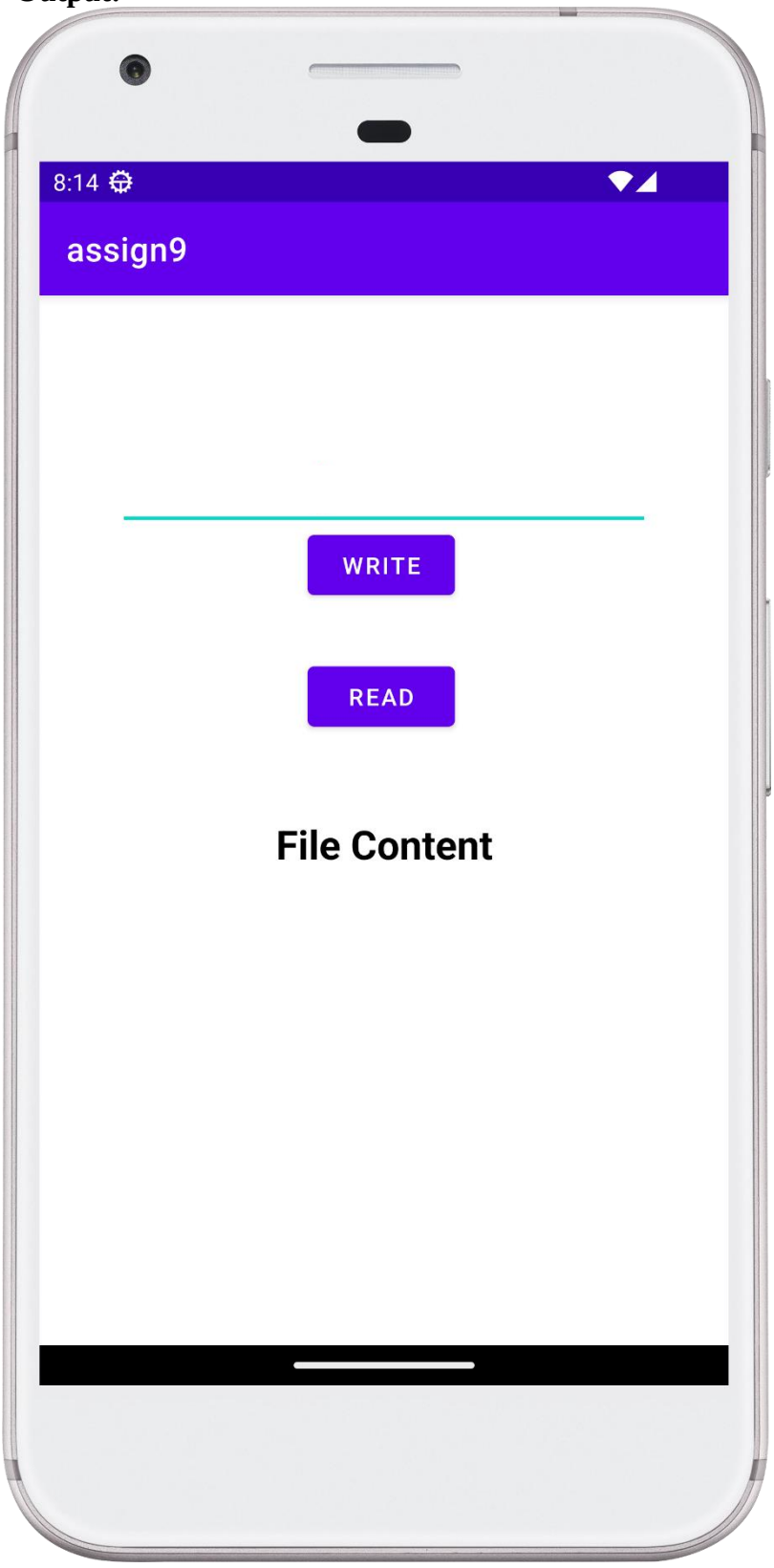
MainActivity.java

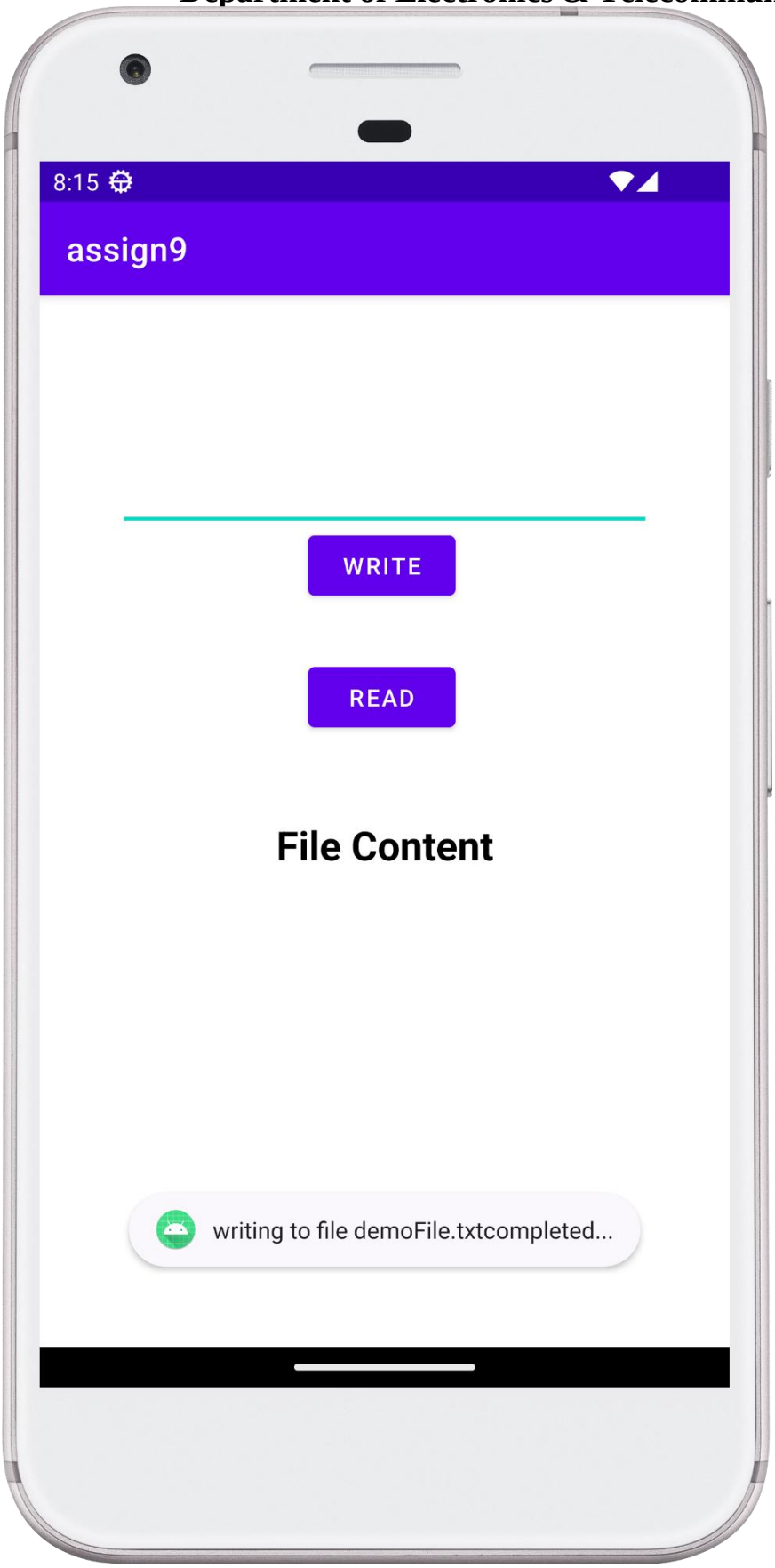
```
package com.example.assign9;

import android.content.Context;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
public class MainActivity extends AppCompatActivity implements View.OnClickListener {

    // declare the variables
    Button read, write;
    EditText userInput;
    TextView fileContent;

    private String filename = "demoFile.txt";

    @Override
```

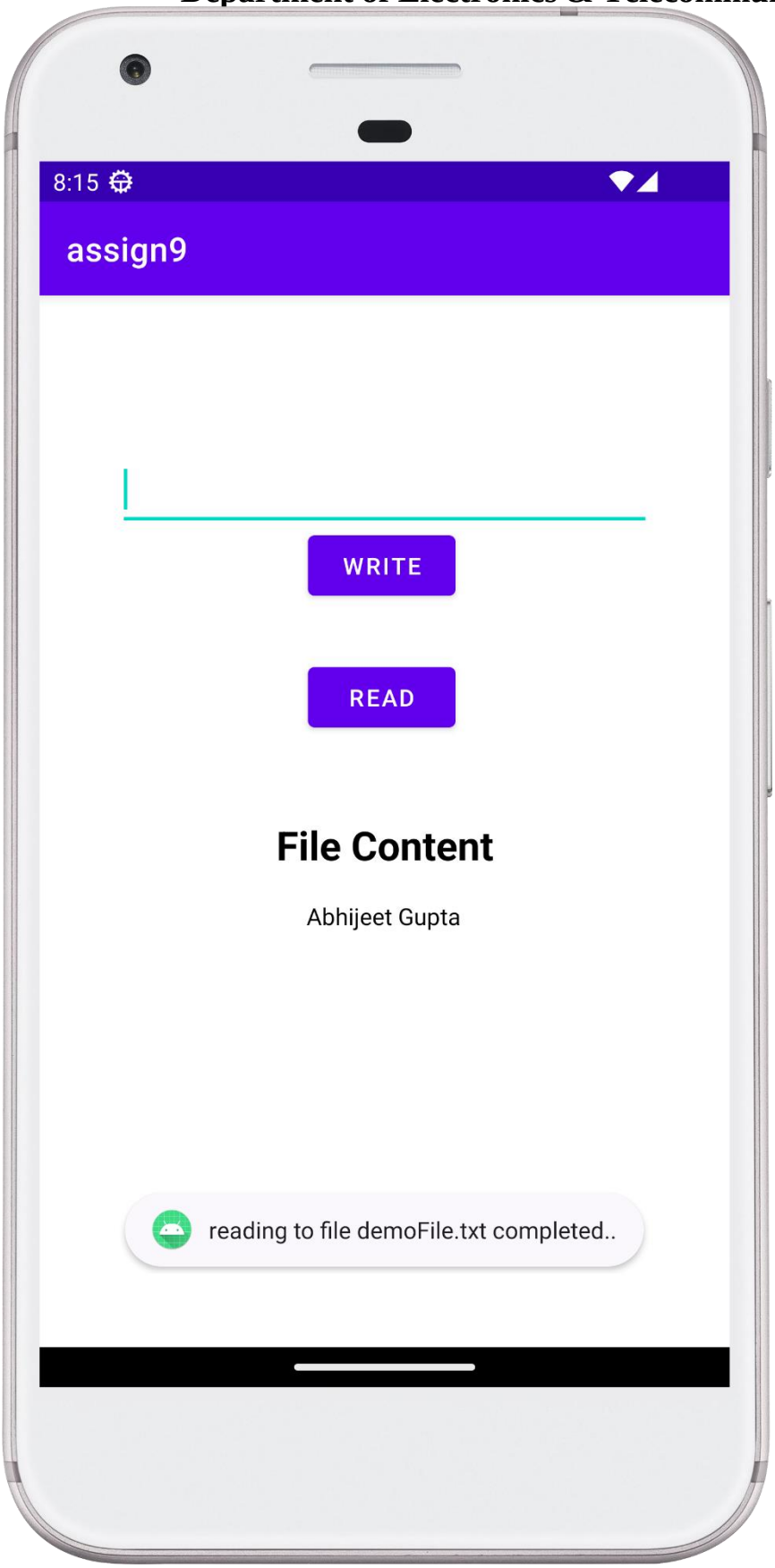
```
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);

   read = findViewById(R.id.read_button);
   write = findViewById(R.id.write_button);
   userInput = findViewById(R.id.userInput);
   fileContent = findViewById(R.id.content);

   read.setOnClickListener(this);
   write.setOnClickListener(this);
}

public void printMessage(String m) {
   Toast.makeText(this, m, Toast.LENGTH_LONG).show();
}

@Override
public void onClick(View view) {
   Button b = (Button) view;

   // get the button text : in out case either read or
   // write depending on the button pressed.
   String b_text = b.getText().toString();

   switch (b_text.toLowerCase()) {
      case "write": {
         writeData();
         break;
      }
      case "read": {
         readData();
         break;
      }
   }
}

private void writeData() {

   try {
      FileOutputStream fos = openFileOutput(filename, Context.MODE_PRIVATE);
      String data = userInput.getText().toString();
      fos.write(data.getBytes());
      fos.flush();
      fos.close();
   } catch (IOException e) {
      e.printStackTrace();
   }
   userInput.setText("");
   printMessage("writing to file " + filename + "completed...");
}

private void readData() {
   try {
      FileInputStream fin = openFileInput(filename);
      int a;
      StringBuilder temp = new StringBuilder();
      while ((a = fin.read()) != -1) {
         temp.append((char) a);
      }
```

```
            // setting text from the file.
            fileContent.setText(temp.toString());
            fin.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
        printMessage("reading to file " + filename + " completed..");
    }
}
```

Output:




8:15
assign9
WRITE
READ
File Content
writing to file demoFile.txtcompleted...


8:15
assign9
WRITE
READ
File Content
Abhijeet Gupta
reading to file demoFile.txt completed..

CONCLUSION:

……

……

……