Programme Name: BCS

Course Code: (CSC 3563)

Course Name: Mobile Programming

Assignment No. 2

Date of Submission: 2021-12-13

Submitted By:

Student Name: Bijay Tamang

IUKL ID: 041901900007

Semester: 6th

Intake: March 2019

Submitted To:

Faculty Name: Nitesh Regmi

Department: P.O.

1. **Describe why it is important to choose a target SDK**
   Answer:
   The property TargetSdk indicates to the system the Android version the app was created and tested for. It's crucial to pick the right target SDK because it'll ensure backward compatibility with future API versions and devices. Android will adjust its behavior to imitate an earlier device if the app is run on a future device with an API version greater than the target SDK.

2. **Describe THREE (3) common scenarios for using an Intent**
   Answer:
   The three common scenarios for using an Intent are as follows:
   - **Start an Activity:**
     By supplying an Intent to the startActivity() method, you can start a new instance of an Activity. to make a phone call, create a contact list, summon a camera for a photo, or display a web page
   - **Start a Service**
     You can start a service to perform a one-time operation (such as download a file) by passing an Intent to startService().
   - **Deliver a broadcast:**
     You can deliver a broadcast to other apps by passing an Intent to sendBroadcast(), sendOrderedBroadcast(), or sendStickyBroadcast().

3. **Discuss THREE (3) types of layout for Android**
   Answer:
   The three types of Layout for Android are as follows:
   - **LinearLayout**
     This pattern is used to arrange the elements in a logical order. One element per line is referred to as a linear method. On Android, this pattern generates a variety of forms.

   - **RelativeLayout**
     The purpose of this layout is to indicate the position of the elements in relation to the other items present.
     It is possible to align the location of the elements to the parent container in the relative layout.

   - **FrameLayout**
     Frame Layout is used to isolate a portion of the screen for the presentation of a single item. Because it can be difficult to organize child views in a way that is scalable to different screen sizes without the children overlapping each other, FrameLayout should be used to store a single child view.

4. **Describe a scenario to explain when you should use implicit and explicit intents.**
   Answer:
   In general, explicit intents (activities) refer to a certain class and are only visible to your packages. Use it to immediately call an express intent targeting activityB. When we have a general concept of what we want to do but are unsure which component should be launched, we use implicit intents. Alternatively, we might give the user the option of selecting from a list of components.

5. **Describe a scenario where onRestart() method will be called for an Activity implementation.**
   Answer:
   When the onRestart() method is called, the user is brought back to the screen or the activity that was interrupted in the preceding activity is resumed, such as a stopwatch.

6. **List FOUR (4) different types of application components and explain their purpose.**
   Answer:
   The four different types of application components and their purpose are as follows:
   - **Activities:**
     An activity is the entry point for interacting with the user. It represents a single screen with a user interface.
   - **Services:**
     A service is a component that performs long-running processes in the background. For example, a service may play music in the background while the user is interacting with another application, or it could collect data over the network without interfering with the user's ability to interact with the activity.
   - **Broadcast receivers:**
     A broadcast receiver is a component that allows the system to communicate events to the app outside of the normal user flow, allowing it to respond to system-wide broadcast announcements. The system can distribute broadcasts to apps that aren't currently operating because broadcast receivers are another well-defined entry into the app.
   - **Content Provider:**
     On request, a content provider component sends data from one application to another. The Content Resolver class's methods deal with such requests. The information could be kept in a file system, a database, or somewhere else entirely.

7. **List SIX (6) callback method of Activity Lifecycle except onCreate() method and explain their.**

Answer:

The 6 callback method of Activity Lifecycle except onCreate() method are as follows:

- **onStart():** As soon as the activity becomes visible to the users, the Android onstart() function is called. To begin an activity, use this approach. When this method is called, the activity is displayed on the users' for screen.
- **onPause():** When an activity receives no user input and is put on hold, the Android onPause() method is called. The activity is partially visible to the user while in the pause state. When the user touches the back or home buttons, this occurs. When an activity is paused, the onResume() or onStopped() callback methods can be used to resume or stop it.
- **onRestart():** When an activity is going to resume from a stop state, the Android onRestart() method is called. This approach is used to reactivate an activity that was previously active. When an activity is restarted, it resumes its previous state.
- **onResume():** When the user begins engaging with the user, the Android onResume() method is called. onPause() comes after this callback method. OnResume() is used to implement the majority of an application's capabilities.
- **onStop():** When the activity is no longer visible to the user, the Android onStop() function is called. The reason for this condition is that either an activity is destroyed or an existing activity returns to its previous state.
- **onDestroy():** When an activity finishes and the user quits using it, the Android onDestroy() function is called. It is the last callback method that activity receives before it is killed.

8. **Assume you are one of the Android programmers in Company PQS. You are given a task by your supervisor, where you have to create a simple login page for a user management application as shown in figure below:**
   A. **Based on the design shown in figure above, provide the XML layout code. Also add onClick = "checkLogin" to Login button and implement its code in JAVA/Android.**

   Answer:

   XML Code for login form (activity_main.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Login Form"
        android:textSize="35dp"
        android:layout_centerHorizontal="true"
        android:layout_below="@id/username"
        android:layout_alignParentTop="true"
        />

    <EditText
        android:id="@+id/username"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="99dp"
        android:gravity="center"
        android:hint="Enter User Name"
        android:inputType="text" />

    <EditText
        android:id="@+id/pass"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/username"
        android:layout_marginTop="27dp"
        android:gravity="center"
        android:hint="Enter Password"
        android:inputType="textPassword" />

    <Button
        android:id="@+id/submit"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/pass"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="30dp"
```

```
        android:onClick="checklogin"
        android:text="Login"
        android:backgroundTint="#000"
        android:textColor="@color/white" />


</RelativeLayout>
```

Implementing login button and its code in MainActivity.java

```
package com.example.usermanagement;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    private EditText username, password;
    private Button submit;


    public void checklogin(View view){}

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        username = (EditText) findViewById(R.id.username);
        password = (EditText) findViewById(R.id.pass);
        submit = (Button) findViewById(R.id.submit);

    }
}
```

B. **Write a button click listener checkLogin method inside MainActivity.java. If username and password is incorrect, raise a Toast with appropriate message. If user name and password is correct, starts a new activity called WelcomeActivity and pass the username entered and display "Hi Username" where Username should be dynamic. Provide the JAVA/Android code for both MainActivity and WelcomeActivity.**

Answer:

Structure:

Login Page:

- **MainActivity.java**
- **activity_main.xml**

User Page:

- **WelcomeActivity.java**
- **activity_welcome.xml**

Activity_main.xml is already given in above answer A

Code:

Login Form

MainActivity.java

```
package com.example.usermanagement;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    private EditText username, password;
    private Button submit;
    private String user = "Bijay",pass = "0000";

    public static final String EXTRA_TEXT = "com.example.usermanagement";

    public void checklogin(View view){
        if (pass.equals(password.getText().toString()) &&
```

```
user.equals(username.getText().toString()) ){
            Intent intent = new Intent(this,welcomeActivity.class);
            intent.putExtra(EXTRA_TEXT,username.getText().toString());
            startActivity(intent);
        }
        else{
            Toast.makeText(getApplicationContext(),"Invalid Username or
Password",Toast.LENGTH_SHORT).show();

        }

    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        username = (EditText) findViewById(R.id.usrname);
        password = (EditText) findViewById(R.id.pass);
        submit = (Button) findViewById(R.id.submit);



    }
}
```

User Page

activity_welcome.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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=".welcomeActivity">

    <TextView
        android:id="@+id/text"
```

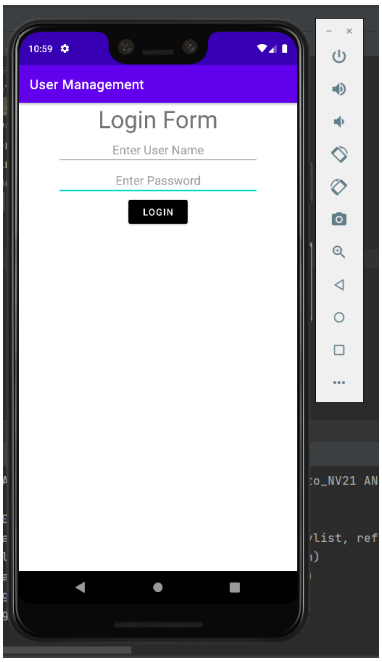
```
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:textSize="50dp"
        />

</RelativeLayout>
```

WelcomeActivity.java

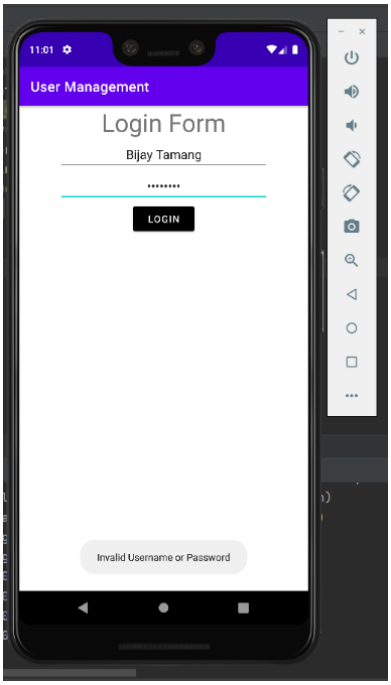
```
package com.example.usermanagement;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.widget.TextView;

public class welcomeActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_welcome);
        Intent intent = getIntent();
        String username = intent.getStringExtra(MainActivity.EXTRA_TEXT);
        TextView helloUser = (TextView) findViewById(R.id.maintext);
        helloUser.setText("Hi "+username);
    }
}
```

Output:

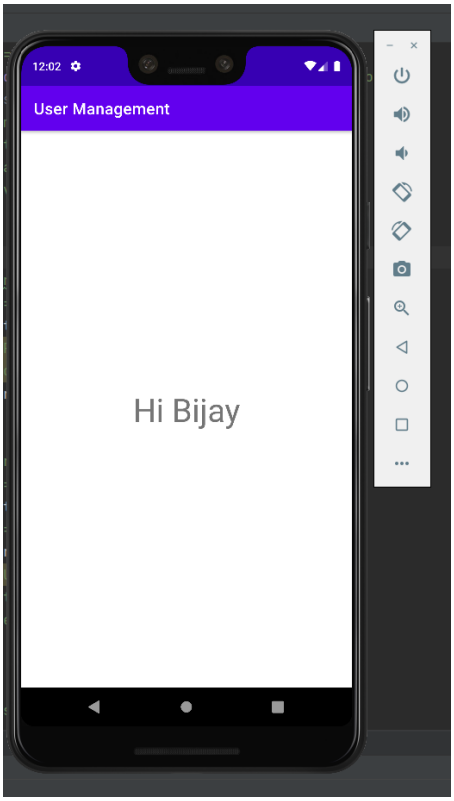
Login Page:



Wrong Password / User name:



Correct Information:



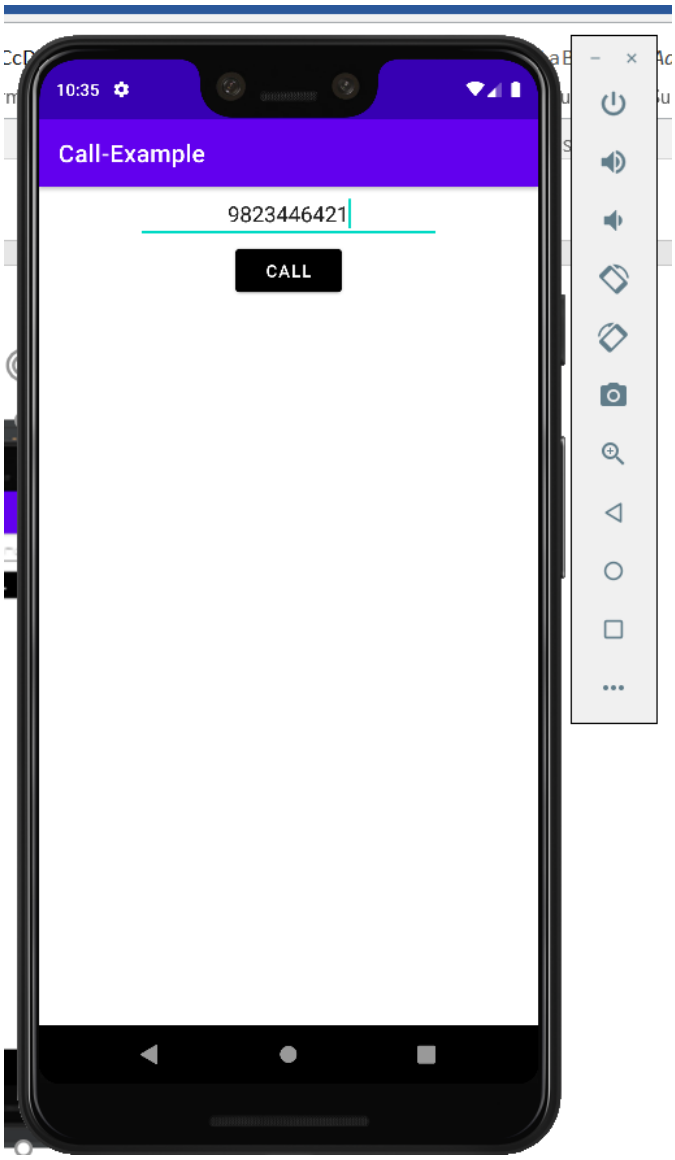
9. **Assume you are one of the android developer in Company ABC. You are given a task by your supervisor, where you have to create a simple phone call application that same as the below figure.**

   a. **Based on the design in above figure, provide the XML layout code.**
   Answer:

   Code: activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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">

    <EditText
        android:id="@+id/numbercall"
        android:layout_width="250dp"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:gravity="center"
        android:hint="Enter Number"
        android:inputType="phone"
        android:onClick="Call" />
    <Button
        android:id="@+id/callbutton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="call"
        android:onClick="callfun"
        android:layout_below="@+id/numbercall"
        android:backgroundTint="#000"
        android:textColor="@color/white"
        android:layout_centerHorizontal="true"
        />

</RelativeLayout>
```

Output:



b. Provide the MainActivity.java code. The functionality of your app is: when user clicks on CALL button, it should ask for or use Phone app to dial using the number provided in the UI. Also, use Validation to check the phone number format.

Answer:

MainActivty.java

```
package com.example.callexample;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.telephony.PhoneNumberUtils;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    EditText numberinput;
    public void callfun(View view){

if(PhoneNumberUtils.isGlobalPhoneNumber(numberinput.getText().toString())) {
            Uri number = Uri.parse("tel:" +
numberinput.getText().toString());
            Intent callIntent = new Intent(Intent.ACTION_DIAL, number);
            startActivity(callIntent);
        }
        else{
            Toast.makeText(getApplicationContext(),"Type Valid
Number",Toast.LENGTH_SHORT).show();
        }
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        numberinput = (EditText) findViewById(R.id.numbercall);
    }
}
```

c. Write a CALL_PHONE permission code in AndroidManifest.xml file
Answer:
Permission for CALL_PHONE:

```
<uses-permission android:name="android.permission.CALL_PHONE"></uses-permission>
```