CSE2MAD LAB 3 – MULTIPLE ACTIVITIES

AIMS

- To explore further features of the android platform and Android studio.
- To build a multi-activity application.
- We understand you know Java, but you need to understand how Android projects are composed and the paradigms of developing on this platform.

BUILD A MESSAGING APPLICATION

Last week we developed an app by laying out widgets, using a layout control, responding to events. Let's go one step further!

Note: This is not a cut-and-paste exercise (hopefully you left that behind in first year), in fact it will not work as variable names will need to be synced and customised to your solution. The code however is instructive as to the steps you need to do to satisfy the goals of the app

Today's exercise is to build a simple text messaging app. The app will allow you to select a contact from your contact list, and send them an SMS

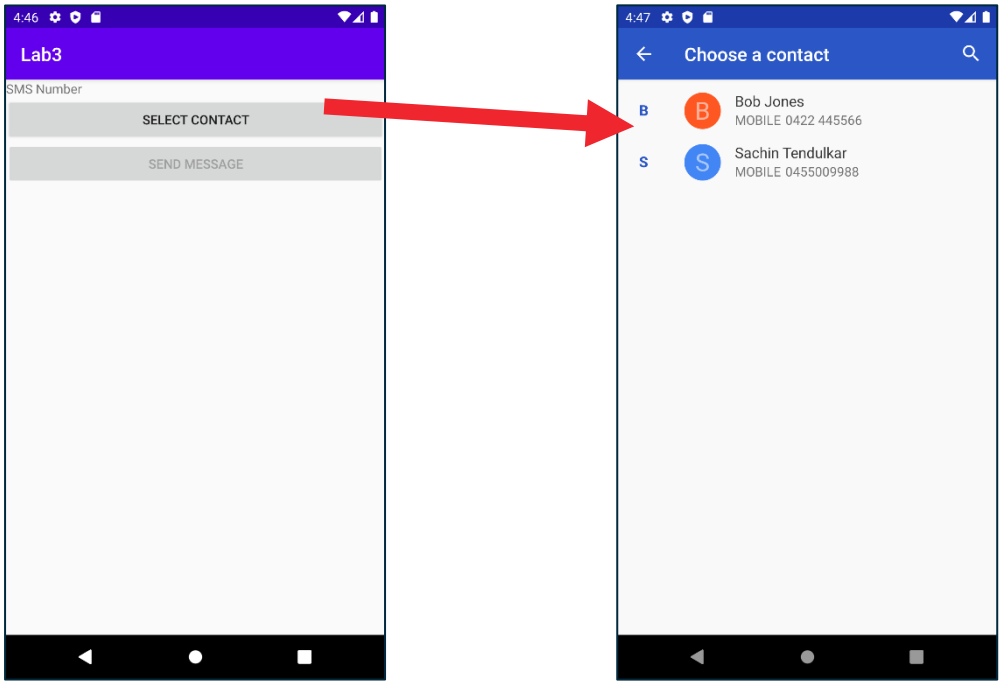


Figure 1

The launching activity (or the main activity) will have a button that opens the contact picker.

When the user selects a contact, the user will be taken back to the main activity & the selected phone number will be displayed & the ‘send message’ button will be enabled.

Figure 2

Figure 3

When the user clicks on the ‘send message’ button, a new activity is started. This second UI has an editable text input to type the message, and a button to send the SMS.

STEP 1 – STARTING THE PROJECT

a) Create a new project & choose the empty activity template. (See week 1 lab)
b) Create a new Android Project using API 25: Android 7.1.1 (Nougat)
c) Create the UI in the visual DESIGN view for the main activity as in Figure 2.

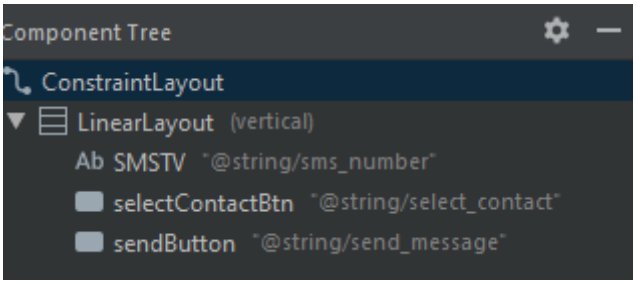


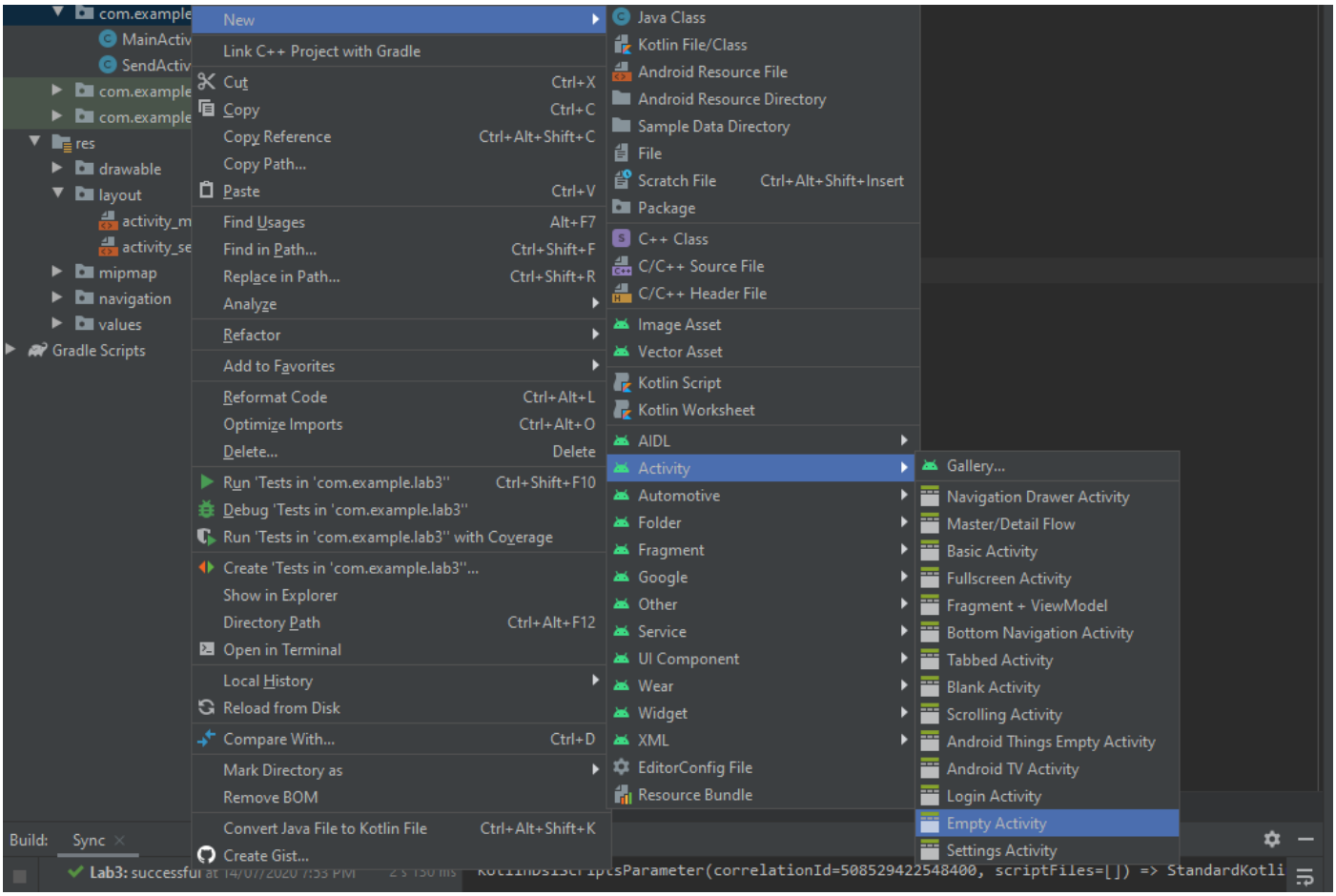
Figure 4

If you are stuck, here is my XML for the main activity layout, but remember it is actually easier to drag and drop from your palette into your component tree as per Figure 4!

```
<?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">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <TextView
            android:id="@+id/SMSTV"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/sms_number" />

        <Button
            android:id="@+id/selectContactBtn"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/select_contact" />

        <Button
            android:id="@+id/sendButton"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/send_message" />
    </LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

STEP 2 – ADDING THE SENDACTIVITY

a) Create a new activity as per below and name it SendActivity.



b) Create the layout as seen in Figure 3. If you are stuck here is the XML.

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

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <EditText
            android:id="@+id/smstext"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:ems="10"
            android:hint="Msg..."
            android:gravity="start|top"
            android:inputType="textMultiLine"
            android:autofillHints="sms msg" />

        <Button
            android:id="@+id/sendMsgButton"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Send Text" />
    </LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

STEP 3 – CODE THE MAIN ACTIVITY

a) Navigate in the IDE to the MainActivity.java file and add variables to access the UI elements in the activity. *Remember from last week where to declare and initialise them!* If you need a hand refer below.

```
Button selectButton = null;
Button sendButton = null;
TextView numView = null;

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


    selectButton = (Button) findViewById(R.id.selectContactBtn);
    sendButton = (Button) findViewById(R.id.sendButton);
    numView = (TextView) findViewById(R.id.SMSTV);
```

b) Add a listener for the button, here we are going to create an intent.

```
selectButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(Intent.ACTION_PICK, ContactsContract.CommonDataKinds.Phone.CONTENT_URI);
        startActivityForResult(intent, PICK_CONTACT);
    }
});
```

Note: you will need to add the member variable PICK_CONTACT at the top of class, with the other member variables.

```
static final int PICK_CONTACT = 1;
```

Run the app. What happens when you click on the 'Select Contact' button? What happens when you click on the 'Send Message' button? It doesn't make sense to have the send button enabled before selecting a contact, so let's disable it in the onCreate method.

```
//make send button inactive
sendButton.setEnabled(false);
```

c) Now we need to get the selected contact's phone number data into our app. First declare a member variable to store the phone number.

```
private String contact_number = null;
```

d) Now let's add some code to get a contact, add this to the activity.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == PICK_CONTACT && resultCode == RESULT_OK) {

        Uri contactUri = data.getData();
        Cursor cursor = getContentResolver().query(contactUri, projection: null, selection: null, selectionArgs: null, sortOrder: null);
        if (cursor != null && cursor.moveToFirst()) {
            int numberIndex = cursor.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER);
            contact_number = cursor.getString(numberIndex);
            // Do something with the phone contact_number
            Log.d( tag: "TEST", contact_number);
            numView.setText(contact_number);
            if (contact_number != null && contact_number.length() > 0) {
                sendButton.setEnabled(true);
            } else {
                sendButton.setEnabled(false);
            }
            cursor.close();
        }
    }
}
```

e) Guess what is next, we need to start the second activity, let's do this via a listener to the send button in the main activity.

```
sendButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent myIntent = new Intent( packageContext: MainActivity.this, SendActivity.class);
        myIntent.putExtra( name: "contact_num", contact_number);
        MainActivity.this.startActivity(myIntent);
    }
});
```

STEP 4 – CODE THE SEND ACTIVITY

a) Let's complete the second activity functionality. Open the SendActivity.java file. As before you are now experts in obtaining a reference to the UI widgets, for help see below.

```
//UX variables
private Button msgBtn = null;
private EditText msgText = null;
//Other
private String contact_number = null;
private String message = null;
static final int SEND_MESSAGE = 3; // STATUS VALUE
static final int SMS_PERMISSION_REQ = 123; // PERMISSIONS VALUE
```

b) As before, initialise the members and this time the intent.

```
//Bind views
msgBtn = (Button) findViewById(R.id.sendMsgButton);
msgText = (EditText) findViewById(R.id.smstext);
//Get data from intent
Intent intent = getIntent();
contact_number = intent.getStringExtra( name: "contact_num");
```

c) We are performing a few operations that need permissions (i.e. sending and receiving SMS). **Go to your AndroidManifest.xml** and notice there are two activities. Within the manifest we are going to have to ask for permission to send SMS. Add

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

above the application specified in XML.

d) **Return to the SendActivity.java** file and handle the event on the send message button of the second activity. (Hopefully you are getting an idea where to register listeners to UI widgets. **It's in the onCreate method**.

```
//Button listener
msgBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        sendMessage();
    }
});
```

e) Implement the sendMessage() method as called above.

```
private void sendMessage(){
    message = msgText.getText().toString();
    if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.SEND_SMS)
            != PackageManager.PERMISSION_GRANTED) {
        Log.d( tag: "MAD",  msg: " SMS Permission is not granted, requesting");
        ActivityCompat.requestPermissions( activity: this, new
                String[]{Manifest.permission.SEND_SMS}, SMS_PERMISSION_REQ);
    } else {
        Log.d( tag: "MAD",  msg: "SMS Permission is given");
        SmsManager smsManager = SmsManager.getDefault();
        smsManager.sendTextMessage(contact_number,  scAddress: null,message,  sentIntent: null,  deliveryIntent: null);
        Toast.makeText( context: SendActivity.this,  text: "Message Sent", Toast.LENGTH_LONG).show();
    }
}
```

There is an OS specific permission request mechanism which we'll talk about in the lab class for Android M (>= 6.0) and later. It has to do with permission granting at runtime or you can view chapter 74 (page 603) of the CSE2MAD textbook online.

RUN AND START TEXTING YOUR FRIENDS .....