Birzeit University
Faculty of Engineering and Technology
Electrical and Computer Engineering Department
Advance Computer Systems Engineering Lab ENCS515

EXP. No. 3. Using Intents and Notifications

1. Objectives

- ❖ Install google play services in the Android Emulator.
- ❖ Using Intent class to apply different functionalities.
- ❖ How to create Toast Notifications.
- ❖ How to incorporate Google Maps into an application.
- ❖ How to incorporate Gmail into an application.
- ❖ How to incorporate Dial Up into an application.
- ❖ How to post notifications in the Notification Bar

2. Introduction

In this lab you are going to learn how intents are coded and implemented. And the importance of intents in any android application. Also, you will learn how to make Toast massages in any application.

2.1. Intents In android:

It is a data structure that represents an operation to be performed. Intents is constructed by one component that wants some work and it is received by one activity that can perform that work. For example, you can use the intent Class to start new activity or to perform a service.

2.2. Intent Fields:

- **Action**: String represents a desired operation. You can set the action of the activity by two ways:

  - By passing it in the constructor:
  - By setAction method

  examples of the actions that can be performed are:

  - ACTION_MAIN: start as initial activity of app
  - ACTION_DIAL: dial a number
  - ACTION_EDIT: display data to edit
  - ACTION_SYNC: synchronize device data with server

- **Data**: data associated with the intent. It is formatted as a uniform resource identifier (URI). For example, to dial a number, you can use the following URI.

```
Intent intent=new Intent(Intent.ACTION_DIAL,Uri.parse("tel:+1555"));
```

- **Category**: additional information about the components that can handle the intent. For example, CATEGORY_LAUNCHER: can be the initial activity a task.
- **Component**: the component that should receive this intent. Use this field when there is exactly one component that should receive the intent.
- **Extras**: what additional information you need to provide (key/value pairs).

Types of Intents

- **Explicit intents**: specify the component to start by name (the fully-qualified class name). You'll typically use an explicit intent to start a component in your own app, because you know the class name of the activity or service you want to start. For example, start a new activity in response to a user action or start a service to download a file in the background.
- **Implicit intents**: do not name a specific component, but instead declare a general action to perform, which allows a component from another app to handle it. For example, if you want to show the user a location on a map, you can use an implicit intent to request that another capable app show a specified location on a map.

- Intent resolution: when the activity to be activated using implicit intents, android tries to find activity that match the intent by comparing the contents of the intent to the intent filters declared in the manifest file of other apps on the device. Android package manager is responsible for deciding which component is best suited to handle your intent. Intent resolution relies on two kind of information:

    - An Intent describing a desired operation.
    - Intents filters which describe which operations an activity can handle. It is specified either in Android Manifest file or programmatically.

2.3. User notifications:

Notifications basically are messages outside the user interface of the application, for example, if you download a book from the internet, you need to use the application while the book is downloading and let the user know when the download finishes. So, the developer should display the message to user contains that information.

In this experiment we will talk about two kinds of user notifications

- **Toast Messages**

Transitory messages that pop up on the current window. It automatically fades into and out of the view without user interaction or response.

- **Notification Area** (Status Bar Notification)

Android provides the notification area for alerting users about events. It also provides a notification drawer that user can pull down to see more detailed information about notifications. The operations on the notification area are managed by system service.

3. Procedure

In this Lab you are going to create an application that has 5 buttons that will perform the following operations: open Dial (Phone) using intents, open Gmail using intents, open google maps using intents, display a notification message, display a toast message.

For displaying notification, you will create a channel to display the notification so the API for the android must not be less than 26 (Android 8.0 Oreo).

3.1. Creating a new Project with API 26

Start by Creating a new Project and change the name for "Intents And Notifications" then click on next, change the API for Phone and Tablet for API 26 (Android 8.0 Oreo) as shown in Figure 3.1. Then click next and finish.

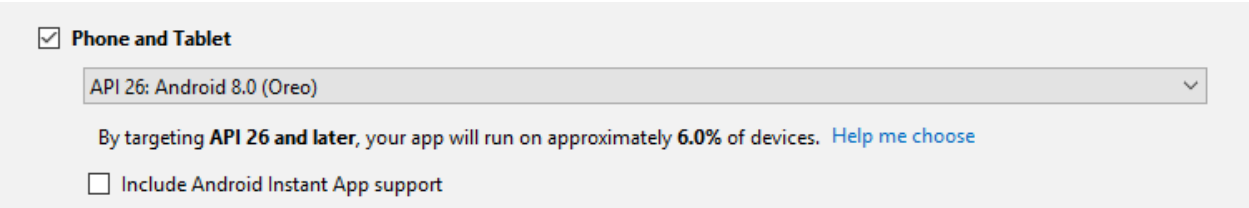

Figure 3.1 Create New Project API 26

3.2. Adding Buttons

Convert the MainActivity root layout to LinearLayout and change the orientation to Vertical from the right-side panel. Add Five Button in the MainActivity Layout (activity_main) and change the names and the Ids as shown in the component tree in Figure 3.2.

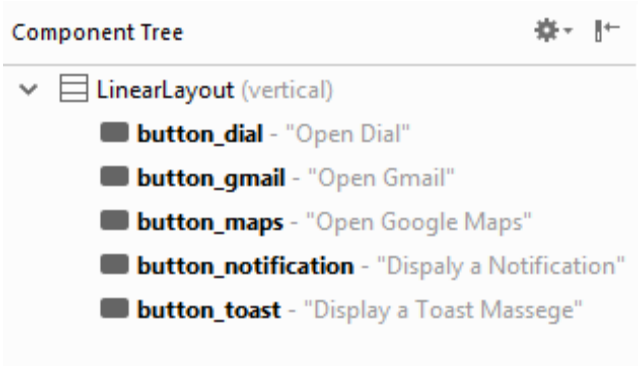

Figure 3.2 Component Tree Screen

3.3. Creating Listener for each Button

- Create five buttons in the MainActivity java file and find view by Id for each button as shown in the code below.

```
Button dialButton = (Button) findViewById(R.id.button_dial);
Button gmailButton = (Button) findViewById(R.id.button_gmail);
Button mapsButton = (Button) findViewById(R.id.button_maps);
Button notificationButton = (Button) findViewById(R.id.button_notification);
Button toastButton =  (Button) findViewById(R.id.button_toast);
```

- Create a click Listener to dial button

  - Create a new Object of Intent this will start a new activity
  - Set the action for the intent object to Intent.ACTION_DIAL
  - Set the data for the intent object to Uri.parse(“tel:+9725”)
  - Execute the intent by calling startActivity method and pass the intent to this method
  - the following code shows how these steps and Figure 3.3 shows the result when clicking the button.

```
dialButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent dialIntent =new Intent();
        dialIntent.setAction(Intent.ACTION_DIAL);
        dialIntent.setData(Uri.parse("tel:+9725"));
        startActivity(dialIntent);
    }
});
```

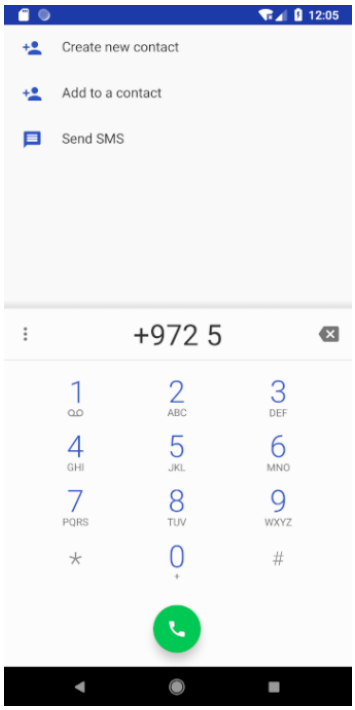

Figure 3.3 Dial Screen

➢ Create a click Listener to gmail button

- Create a new Object of Intent call it “gmailIntent” this will start a new activity
- Set the action for the intent object to Intent.SENDTO
- Set the type for the intent object to “message/rfc822”
- Set the data for the intent object to Uri.parse(“mailto:”)
- Execute the intent by calling startActivity method and pass the intent to this method
- the following code shows how these steps and Figure 3.4 shows the result when clicking the button.

- You can set the sent to mail, subject and the text of the mail in the extras as shown in the code below
- Send the message to your email and check it to be sure that the message was sent successfully.

```
gmailButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent gmailIntent =new Intent();
        gmailIntent.setAction(Intent.ACTION_SENDTO);
        gmailIntent.setType("message/rfc822");
        gmailIntent.setData(Uri.parse("mailto:"));
        gmailIntent.putExtra(Intent.EXTRA_EMAIL, new String
{"rajaie.imseeh@gmail.com"});
        gmailIntent.putExtra(Intent.EXTRA_SUBJECT,"My Subject");
        gmailIntent.putExtra(Intent.EXTRA_TEXT,"Content of the message");
        startActivity(gmailIntent);
    }
});
```

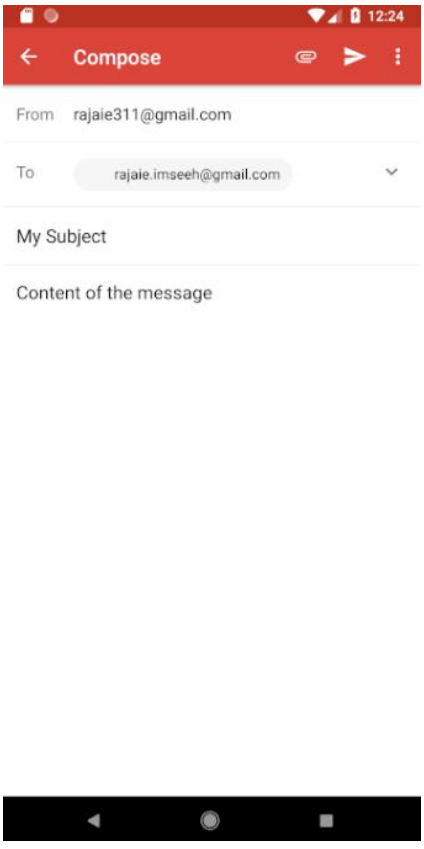

Figure 3.4 Send Gmail Screen

➢ Create a click Listener to Google Maps button

- Create a new Object of Intent this will start a new activity
- Set the action for the intent object to Intent.ACTION_VIEW
- Set the data for the intent object to Uri.parse(“geo:19.076,72.8777”)
- Execute the intent by calling startActivity method and pass the intent to this method
- the following code shows how these steps and Figure 3.5 shows the result when clicking the button.

```
mapsButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent mapsIntent =new Intent();
        mapsIntent.setAction(Intent.ACTION_VIEW);
        mapsIntent.setData(Uri.parse("geo:19.076,72.8777"));
        startActivity(mapsIntent);
    }
});
```

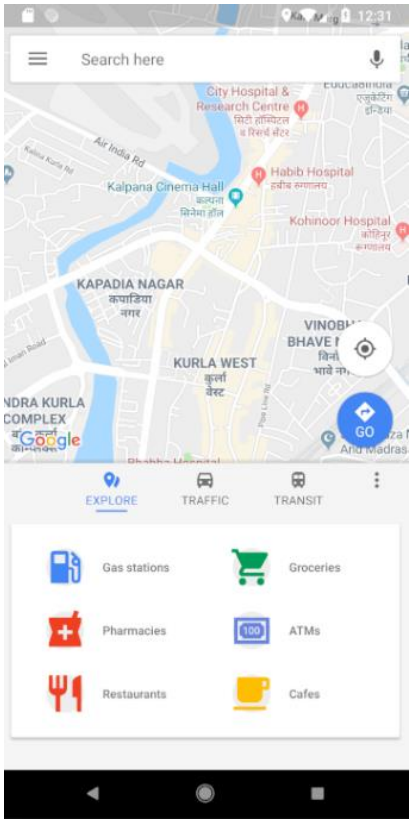

Figure 3.5 Google Maps Screen

- Notification Creation

  - To create a notification, you need first to create a notification channel this can be implemented by creating a method in the main activity to create a channel. Every notification channel has a channel Id (create as global constant variable) and name it MY_CHANNEL_ID, a channel Name(create as global constant variable) and name it MY_CHANNEL_NAME and an Importance (this will determine how much the notification should interrupt the user, higher the importance in the notification, the more interruptive the notification will be).
  - Using the notification manager, you will create the notification channel, the code below is the implementation of the notification channel.

```
private static final String MY_CHANNEL_ID = "my_chanel_1";
private static final String MY_CHANNEL_NAME = "My channel";

private void createNotificationChannel() {
    int importance = NotificationManager.IMPORTANCE_DEFAULT;
    NotificationChannel channel = new NotificationChannel(MY_CHANNEL_ID,
MY_CHANNEL_NAME, importance);
    NotificationManager notificationManager =
getSystemService(NotificationManager.class);
    if (notificationManager != null) {
        notificationManager.createNotificationChannel(channel);
    }
}
```

  - After that you will create another method that will create the notification in this channel you created, this method will also be implemented in the main activity. You will need the following components shown in Table 3.1 to create the notification.

Table 3.1 needed components for creating the notification

Intent	This is for starting an activity when clicking the notification.
Pending Intent	This is created to pass it to the notification using setContentIntent.
Notification Title	The title of the notification (global constant) passed to the method
Notification Body	The Body of the notification (global constant) passed to the method
Notification small Icon	Use the mipmap ic_launcher
Property	Set to default

- Using the notification manager compat notify the created notification, this method will need an Id (create as global constant).
- The code below shows the method to create the notification.

```
private static final int NOTIFICATION_ID = 123;
private static final String NOTIFICATION_TITLE = "Notification Title";
private static final String NOTIFICATION_BODY = "This is the body of my notification";

public void createNotification(String title, String body) {
    Intent intent = new Intent(this, MainActivity.class);
    PendingIntent pendingIntent = PendingIntent.getActivity(this,0,intent, 0);

    createNotificationChannel();

    NotificationCompat.Builder builder = new NotificationCompat.Builder(this,
MY_CHANNEL_ID)
            .setSmallIcon(R.mipmap.ic_launcher)
            .setContentTitle(title)
            .setContentText(body)
            .setStyle(new NotificationCompat.BigTextStyle().bigText(body))
            .setPriority(NotificationCompat.PRIORITY_DEFAULT)
            .setContentIntent(pendingIntent);
    NotificationManagerCompat notificationManager =
    NotificationManagerCompat.from(this);
    notificationManager.notify(NOTIFICATION_ID, builder.build());
}
```

- Create a click Listener to Notification button
- Call the createNotification method which will create the notification channel and the notification to notify it, the code below shows how to create the listener and call the createNotification method passing the title and the body to the method.

```
notificationButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        createNotification(NOTIFICATION_TITLE,NOTIFICATION_BODY);
    }
});
```

- Figure 3.6 Shows the notification created after clicking the notification button.

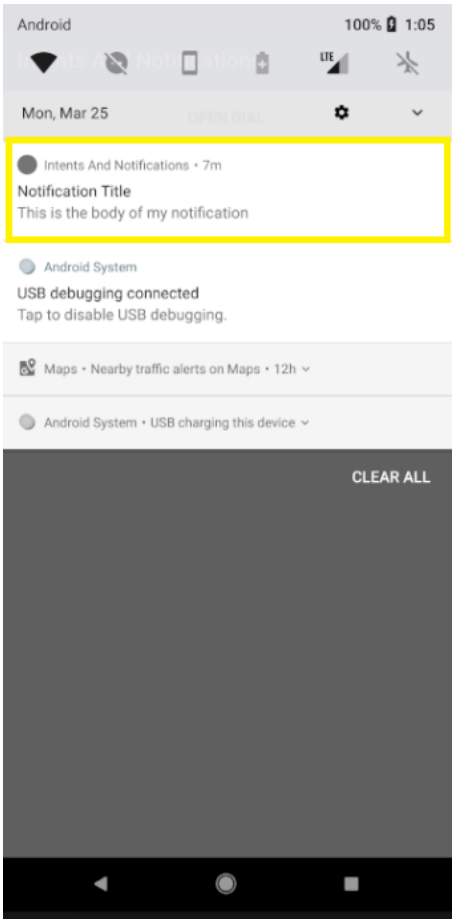

Figure 3.6 Notification displaying Screen

- Create a click Listener for the Toast Message button

  - To create a Toast Message, you will need to create a Toast object and pass the application context, text for the toast message, and the duration (Length short for short duration toast message and Length Long for long duration toast message).
  - Then call the method show() to show the toast message. The code below shows how to create a toast message and Figure 3.7 shows the toast message after clicking the toast message button.

```
private static final String TOAST_TEXT = "This my toast message";

 toastButton.setOnClickListener(new View.OnClickListener() {
     @Override
     public void onClick(View view) {
         Toast toast =Toast.makeText(MainActivity.this,
     TOAST_TEXT,Toast.LENGTH_SHORT);
         toast.show();
     }
 });
```

Figure 3.7 Toast Massage Displaying Screen

4. Todo

This part will be given to you by the teacher assistant in the lab time.