CONTENTS

Course Objectives

- To design graphical user interface for mobile application.
- To understand the implementation of interactive components for mobile application.
- To handle events as per requirement.
- To design database as per requirement to interact with application.

Course Outcomes

At the end of this course, students will be able to

- Apply the knowledge of concept learnt to design appropriate user interface layouts for mobile application.
- Analyse variety of interactive components in application's user interface as per requirements.
- Comprehend the captured events in their program and take appropriate action as per requirements.
- Emphasize on integration of database with the application as per requirement.

Evaluation plan

Split up of 60 marks for Regular Lab Evaluation
Regular Evaluation: Biweekly Evaluation 1,2 & 3 Record Book: 4 Marks, Execution: 2 Marks, Viva/Quiz:4 Marks Evaluation 4,5& 6 (related app development based on given requirements, constraints) UI Design:4 Marks, Developed as per requirement:2 Marks, Execution:4 Marks Total 30+30=60 Marks
Project work: 20 Marks Write up and execution: 20 Marks (Duration 2 hrs) Total 20+20=40 Marks

INSTRUCTIONS TO THE STUDENTS

Pre- Lab Session Instructions

1. Students should carry the Lab Manual Book and the required stationery to every lab session
2. Be in time and follow the institution dress code
3. Must sign in the log register provided
4. Make sure to occupy the allotted seat and answer the attendance

5. Adhere to the rules and maintain the decorum

In- Lab Session Instructions

- Follow the instructions on the allotted exercises
- Show the program and results to the instructors on completion of experiments
- On receiving approval from the instructor, copy the program and results in the Lab record

General Instructions for the exercises in Lab

- Implement the given exercise individually as specified.
- The programs should meet the following criteria:
    - Programs should be interactive with appropriate prompt messages, error messages if any, and descriptive messages for outputs.
    - Comments should be used to give the statement of the problem.
    - Statements within the program should be properly indented.
- Plagiarism (copying from others) is strictly prohibited and would invite severe penalty during evaluation.
- The exercises for each lab is divided under three sets:
    - Solved exercise
    - Lab exercises - to be completed during lab hours
    - Additional Exercises - to be completed outside the lab or in the lab to enhance the skill
- In case a student misses a lab class, he/ she must ensure that the experiment is completed during the repetition class with the permission of the faculty concerned but credit will be given only to one day's experiment(s).

LAB 1:

Title: Android Introduction
Aim: Understand the android development environment.

Android is an open source and Linux-based operating system for mobile devices such as smartphones and tablet computers. Android was developed by the Open Handset Alliance, led by Google, and other companies.

Features of Android

1. Open source.
2. Increased Market.
3. Inter App integration.
4. Customizable etc.

Android Application

Many android applications are available in market. The popular categories are:

1. Games.
2. Communication.
3. Social media.
4. Weather.
5. Business etc.

Set-up Java Development Kit (JDK)

The latest version of Java JDK for download is available from Oracle's Java site − Java . Finally set PATH and JAVA_HOME environment variables to refer to the directory that contains **java** and **javac**, right-click on My Computer, select Properties, then Advanced, then Environment Variables. Then, update the PATH value and press the OK button.

Android IDEs

The IDEs are available to develop android application are as follows:

1. Eclipse IDE.
2. Android studio.

Creating and running an Android App in Android Studio: The app creation is showed in the steps 1 to 4.

Step1: Creating a New Android Project

The first step in the application development process is to create a new project within the Android Studio environment. Begin, therefore, by launching Android Studio so that the "Welcome to Android Studio" screen appears as illustrated in Figure 1:

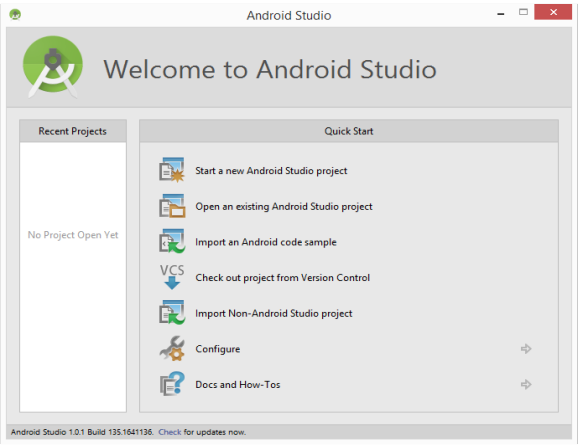

Figure 1

Once this window appears, Android Studio is ready for a new project to be created. To create the new project, simply click on the Start a new Android Studio project option to display the first screen of the New Project wizard as shown in Figure 2:

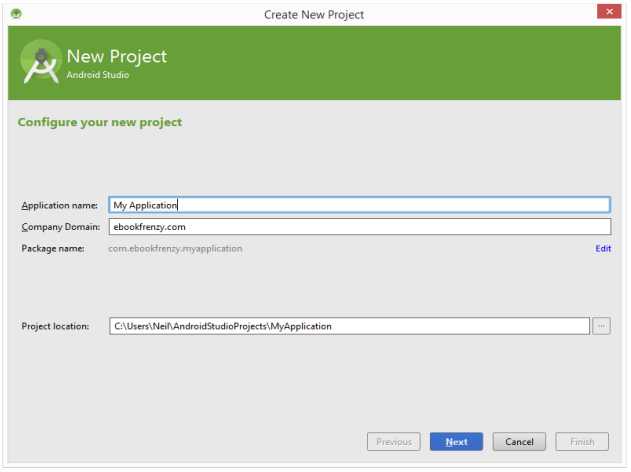

Figure 2

The Project location setting will be default to a location in the folder named AndroidStudioProjects located in the home directory and may be changed by clicking on the button to the right of the text field containing the current path setting.
Click Next to proceed. On the form factors screen, enable the Phone and Tablet option and set the minimum SDK setting to API 8: Android 2.2 (Froyo). The reason for selecting an older SDK release is that this ensures that the finished application will be able to run on the widest possible range of Android devices. The higher the minimum SDK selection, the more the application will be restricted to newer Android devices.

Step 2: Creating an Activity

The next step defines the type of initial activity that is to be created for the application. A range of different activity types is available when developing Android applications. select the option to create a Blank Activity.

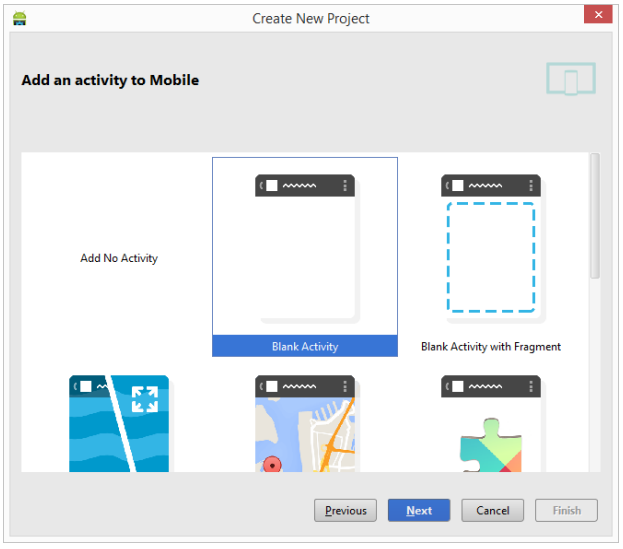

Figure 3

With the Blank Activity option selected, click Next. On the final screen (Figure 4) name the activity and title AndroidSampleActivity. The activity will consist of a single user interface screen layout which, for the purposes of this example, should be named activity_android_sample as shown in Figure 4 and with a menu resource named menu_android_sample.

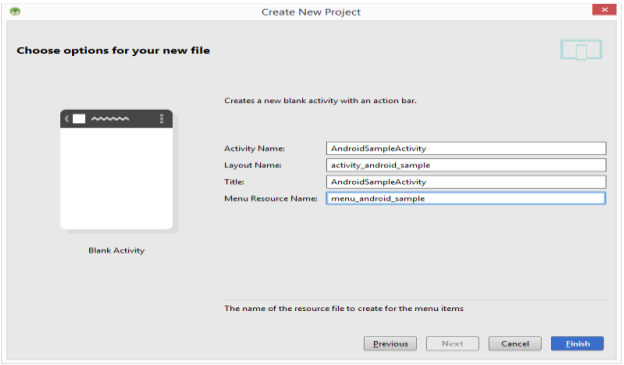

Figure 4

Finally, click on Finish to initiate the project creation process.

Step 3: Creating an Android Virtual Device (AVD) in Android Studio

In the course of developing Android apps in Android Studio it will be necessary to compile and run an application multiple times. An Android application may be tested by installing and running it either on a physical device or in an Android Virtual Device (AVD) emulator environment. Before an AVD can be used, it must first be created and configured to match the specification of a particular device model.

Step 3.1: Creating a New AVD

To create a new AVD, the first step is to launch the AVD Manager. This can be achieved from within the Android Studio environment by selecting the Tools -> Android -> AVD Manager menu option from within the main window.

Once launched, the tool will appear as outlined in Figure 5. Assuming a new Android SDK installation, no AVDs will currently be listed:

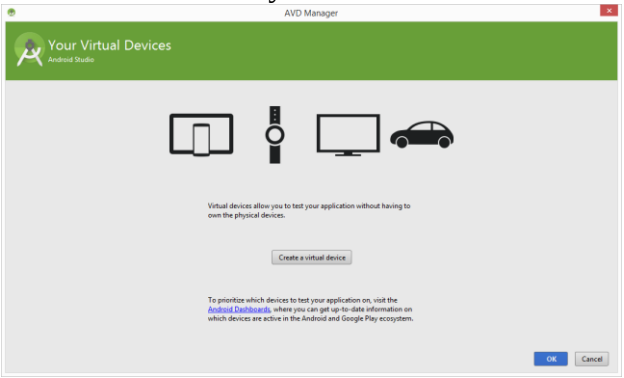

Figure 5

Begin the AVD creation process by clicking on the Create a virtual device button in order to invoke the Virtual Device Configuration dialog:

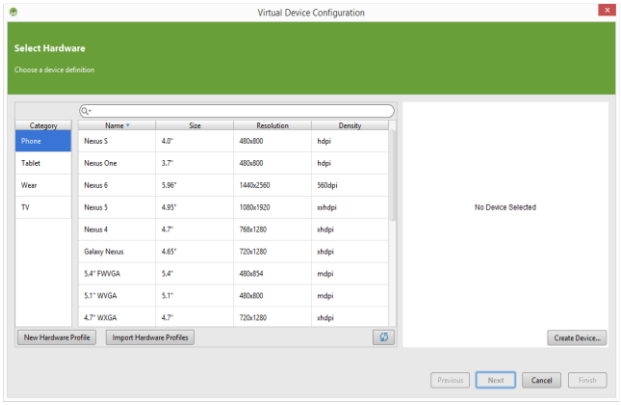

Figure 6

Step 3.2: Within the dialog, perform the following steps to create a first generation Nexus 7 compatible emulator:

1. From the Category panel, select the Tablet option to display the list of available Android tablet AVD templates.

2. Select the Nexus 7 (2012) device option and click Next.

3. On the System Image screen, select the latest version of Android (at time of writing this is Android 5.0.1, Lollipop API level 21) for the armeabi-v7a ABI. Click Next to proceed.

4. Enter a descriptive name (for example Nexus 7) into the name field.

5. Click Finish to create the AVD.

With the AVD created, the AVD Manager may now be closed. If future modifications to the AVD are necessary, simply re-open the AVD Manager, select the AVD from the list and click on the pencil icon in the Actions column of the device row in the AVD Manager.

Step 4: Running the Application in the AVD

With an AVD emulator configured, the example AndroidSample application created now can be compiled and run. With the AndroidSample project loaded into Android Studio, click on the run button represented by a green triangle located in the Android Studio toolbar as shown in Figure 7, select the Run -> Run… menu option or use the Shift+F10 keyboard shortcut.

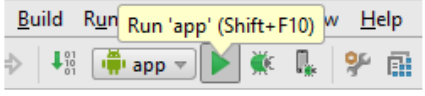

Figure 7

By default, Android Studio will respond to the run request by displaying the Choose Device dialog. This provides the option to execute the application on an AVD instance that is already running, or to launch a new AVD session specifically for this application. The step 3.2 will create Nexus7 AVD as a running device shown in Figure

8. With this device selected in the dialog, click on OK to install and run the application on the emulator.

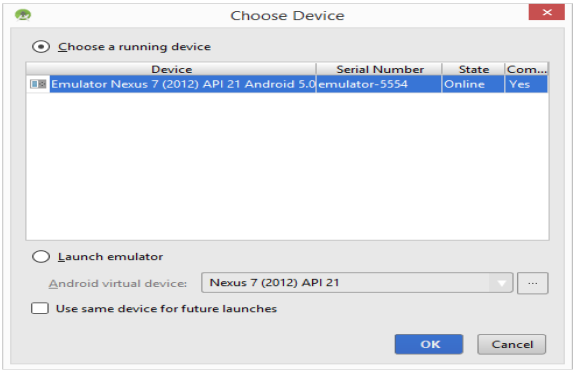

Figure 8

Once the application is installed and running, the user interface for the AndroidSampleActivity class will appear within the emulator.

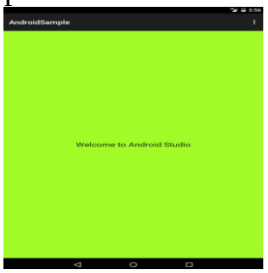

Figure 9

Android folder structure visible in the IDE/ Explorer is shown in figure 10.

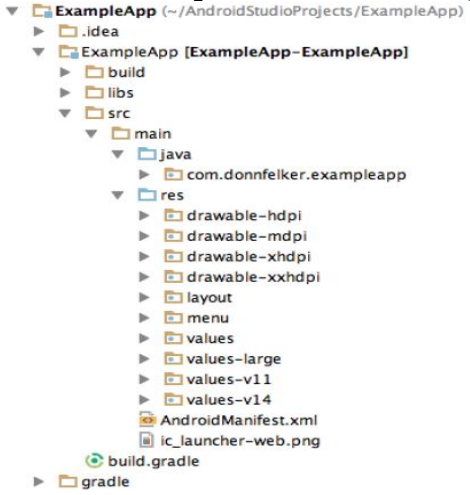

Figure 10

- **JAVA** contains the **.java** source files for your project. By default, it includes an MainActivity.java source file having an activity class that runs when your app is launched using the app icon.
- **res/drawable-hdpi** is a directory for drawable objects that are designed for high-density screens.
- **res/layout** is a directory for files that define your app's user interface.
- **res/values** is a directory for other various XML files that contain a collection of resources, such as strings and colours definitions.
- **AndroidManifest.xml** is the manifest file which describes the fundamental characteristics of the app and defines each of its components.

Android UI Layouts

The basic building block for user interface is a **View** object which is created from the View class and occupies a rectangular area on the screen and is responsible for drawing and event handling. View is the base class for widgets, which are used to create interactive UI components like buttons, text fields, etc.

The **ViewGroup** is a subclass of **View** and provides invisible container that hold other Views or other ViewGroups and define their layout properties.

At third level there are different layouts which are subclasses of ViewGroup class and a typical layout defines the visual structure for an Android user interface and can be created either at run time using **View/ViewGroup** objects or can declare the layout using simple XML file **main_layout.xml** which is located in the res/layout folder of the project.

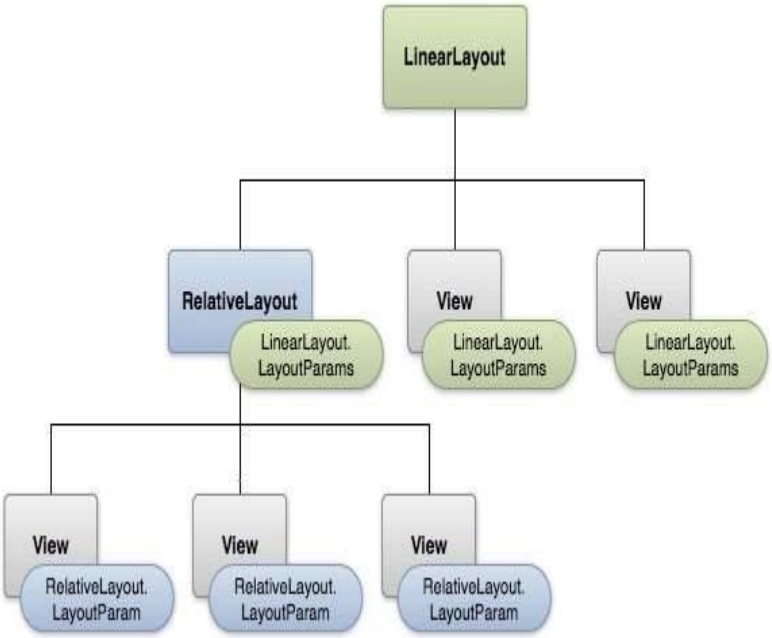

Figure 11

Following is an example of XML file having LinearLayout:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:orientation="vertical" >

  <TextView android:id="@+id/text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="This is a TextView" />

  <Button android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="This is a Button" />

  <!-- More GUI components go here  -->

</LinearLayout>
```

Once the layout has been created, the layout resource can be loaded from the application code, in *Activity.onCreate()* callback implementation the onCreate() implementation is given below.

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

Android Layout Types

- LinearLayout is a view group that aligns all children in a single direction, vertically or horizontally.
- RelativeLayout is a view group that displays child views in relative positions.
- TableLayout is a view that groups views into rows and columns.
- AbsoluteLayout supports specifying exact location of its children.
- The FrameLayout is a placeholder on screen that can be used to display a single view.
- ListView is a view group that displays a list of scrollable items.
- GridView is a ViewGroup that displays items in a two-dimensional, scrollable grid.

Solved example

1. **Write a program for a simple login application.**

activity_login.xml

```
<!—xml code to designlogin page using basic widgets like TextView, EditText and Button-->
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   tools:context=".LoginActivity" >
```

```
<TextView
   android:id="@+id/tv_Username"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_alignParentLeft="true"
   android:layout_alignParentTop="true"
   android:layout_marginLeft="18dp"
   android:layout_marginTop="15dp"
   android:text="@string/username_text" />

<EditText
   android:id="@+id/et_Username"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_alignBaseline="@+id/tv_Username"
   android:layout_alignBottom="@+id/tv_Username"
   android:layout_marginLeft="30dp"
   android:layout_toRightOf="@+id/tv_Username"
   android:ems="10" >

   <requestFocus />
</EditText>

<TextView
   android:id="@+id/tv_Password"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_alignLeft="@+id/tv_Username"
   android:layout_below="@+id/et_Username"
   android:layout_marginTop="47dp"
   android:text="@string/password_text" />

<EditText
   android:id="@+id/et_Password"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_alignBaseline="@+id/tv_Password"
   android:layout_alignBottom="@+id/tv_Password"
   android:layout_alignLeft="@+id/et_Username"
```

```
        android:ems="10"
        android:inputType="textPassword" />

    <Button
        android:id="@+id/bt_SignIn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignRight="@+id/et_Password"
        android:layout_below="@+id/et_Password"
        android:layout_marginTop="34dp"
        android:text="@string/signin_text" />

</RelativeLayout>
```

LoginActivity.java

```
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class LoginActivity extends Activity {
    // User name
    private EditText et_Username;
    // Password
    private EditText et_Password;
    // Sign In
    private Button bt_SignIn;
    // Message
    private TextView tv_Message;

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

```
        // Initialization
        et_Username = (EditText) findViewById(R.id.et_Username);
        et_Password = (EditText) findViewById(R.id.et_Password);
        bt_SignIn = (Button) findViewById(R.id.bt_SignIn);
        tv_Message = (TextView) findViewById(R.id.tv_Message);

        bt_SignIn.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View view) {
                // Stores User name
                String username = String.valueOf(et_Username.getText());
                // Stores Password
                String password = String.valueOf(et_Password.getText());

                // Validates the User name and Password for admin, admin
                if (username.equals("admin") && password.equals("admin")) {
                    tv_Message.setText("Login Successful!");
                   Toast.makeText(getApplicationContext(),
                     " Login Successful...",Toast.LENGTH_SHORT).show();
                } else {
                    tv_Message.setText("Login Unsuccessful!");
                }
            }
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.login, menu);
        return true;
    }

}
```

Exercise questions

1. Write a program to create a simple user profile which includes user's professional details.
2. Write a program to create a registration form :
   a. Form includes email id, mobile number and password as its field.
   b. Perform validation for email id and mobile number.
   c. Display customized toast message based on user input.

Additional question

1. Write a program to display the following table.

Weather Report					
	M	T	W	T	F
Day High	34°C	35°C	34°C	35°C	33°C
Day Low	28°C	27°C	29°C	26°C	29°C
Conditions					

LAB 2:

Title: Android UI control.

Aim: Familiarization of android UI controls.

Button

A Button control provides an area on the UI where a user can touch to initiate an action.

TextView is a View object that simply displays text to the user. By default it is not editable. It is commonly used to display heading, static textual information, and labels to other View objects.

The EditText is an extension of TextView that possesses rich editing capabilities. It provide a text field to allow the user to input text. It can be either a single line or multiple line.

A CheckBox allows users to select or unselect an option. A set of checkboxes are used to select multiple options that are mutually exclusive.

RadioButton offers users an option to choose. However, the radio button does not live alone. It belongs to a group of radio buttons whereby only one of them can be selected at any one time, such as the selection of gender. How to ensure that, the answer is "grouping them together in a RadioGroup". In this way, the system ensures that only one radio button can be selected at a time.

A ToggleButton is an on/off switch. It can be used in turning on and off features like vibration mode, silent mode, WiFi connection, and so on.

Spinner A drop-down list that allows users to select one value from a set.

The ***AutoCompleteTextView*** is a *View* that is similar to *EditText*, with the added capability of showing a list of suggested words from some data source automatically while the user is typing. .

The ***DatePicker*** and ***TimePicker.*** They are the familiar faces on many web applications that ask for date and time information. Pick (click) a date or time, free of typing and thus free of hassle of having to handle formatting issues in code.

An ***ImageButton*** is an AbsoluteLayout which enables you to specify the exact location of its children. This shows a button with an image (instead of text) that can be pressed or clicked by the user.

The ***ProgressBar*** view provides visual feedback about some ongoing tasks, such as when you are performing a task in the background.

Create List

The RecyclerView widget is a more advanced and flexible version of ListView. This widget is a container for displaying large data sets that can be scrolled very efficiently by maintaining a limited number of views. The RecyclerView widget is used when data collections is required and where elements change at runtime based on user action or network events.

The RecyclerView class simplifies the display and handling of large data sets by providing:

- Layout managers for positioning items
- Default animations for common item operations, such as removal or addition of items.

To use the RecyclerView widget, An adapter and a layout manager has to be specified. To create an adapter, extend the RecyclerView. Adapter class. The details of the implementation depend on the specifics of the dataset and the type of views.

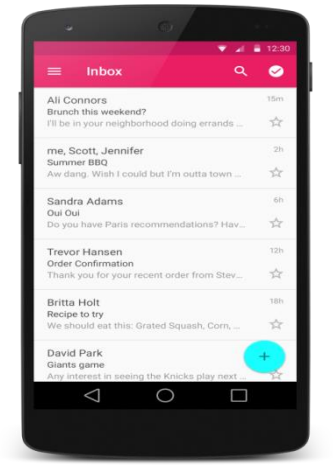

Figure 12

A **layout manager** positions item views inside a RecyclerView and determines when to reuse item views that are no longer visible to the user. To reuse (or *recycle*) a view, a layout manager may ask the adapter to replace the contents of the view with a different

element from the dataset. Recycling views in this manner improves performance by avoiding the creation of unnecessary views or performing expensive findViewById() lookups.

RecyclerView provides the following built-in layout managers:

- LinearLayoutManager shows items in a vertical or horizontal scrolling list.
- GridLayoutManager shows items in a grid.
- StaggeredGridLayoutManager shows items in a staggered grid.

To create a custom layout manager, extend the RecyclerView. LayoutManager class.

Animations
Animations for adding and removing items are enabled by default in RecyclerView. To customize these animations, extend the RecyclerView.ItemAnimator class and use the RecyclerView.setItemAnimator()method.

Examples:
The following code example demonstrates how to add the RecyclerView to a layout:

```
<!-- A RecyclerView with some commonly used attributes -->
<android.support.v7.widget.RecyclerView
    android:id="@+id/my_recycler_view"
    android:scrollbars="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

Once a RecyclerView widget is added to the layout, A handle to the object is obtained to connect to a layout manager, and then an adapter for the data to be displayed is attached.

```
public class MyActivity extends Activity {
    private RecyclerView mRecyclerView;
    private RecyclerView.Adapter mAdapter;
    private RecyclerView.LayoutManager mLayoutManager;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.my_activity);
        mRecyclerView = (RecyclerView) findViewById(R.id.my_recycler_view);

        // use this setting to improve performance if you know that changes
        // in content do not change the layout size of the RecyclerView
        mRecyclerView.setHasFixedSize(true);

        // use a linear layout manager
        mLayoutManager = new LinearLayoutManager(this);
        mRecyclerView.setLayoutManager(mLayoutManager);

        // specify an adapter (see also next example)
        mAdapter = new MyAdapter(myDataset);
        mRecyclerView.setAdapter(mAdapter);
    }
    ...
}
```

The adapter provides access to the items in the data set, creates views for items, and replaces the content of some of the views with new data items when the original item is no longer visible. The following code example shows a simple implementation for a data set that consists of an array of strings displayed using TextView widgets:

```
public class MyAdapter extends RecyclerView.Adapter<MyAdapter.ViewHolder> {
    private String[] mDataset;

    // Provide a reference to the views for each data item
    // Complex data items may need more than one view per item, and
    // you provide access to all the views for a data item in a view holder
    public static class ViewHolder extends RecyclerView.ViewHolder {
        // each data item is just a string in this case
        public TextView mTextView;
        public ViewHolder(TextView v) {
            super(v);
            mTextView = v;
```

```
        }
    }

    // Provide a suitable constructor (depends on the kind of dataset)
    public MyAdapter(String[] myDataset) {
        mDataset = myDataset;
    }

    // Create new views (invoked by the layout manager)
    @Override
    public MyAdapter.ViewHolder onCreateViewHolder(ViewGroup parent,
                                                   int viewType) {
        // create a new view
        TextView v = (TextView) LayoutInflater.from(parent.getContext())
                .inflate(R.layout.my_text_view, parent, false);
        // set the view's size, margins, paddings and layout parameters
        ...
        ViewHolder vh = new ViewHolder(v);
        return vh;
    }

    // Replace the contents of a view (invoked by the layout manager)
    @Override
    public void onBindViewHolder(ViewHolder holder, int position) {
        // - get element from your dataset at this position
        // - replace the contents of the view with that element
        holder.mTextView.setText(mDataset[position]);

    }

    // Return the size of your dataset (invoked by the layout manager)
    @Override
    public int getItemCount() {
        return mDataset.length;
    }
}
```

Create Cards

CardView extends the FrameLayout class and allows information inside cards that have a consistent look across the platform to be shown. CardView widgets can have shadows and rounded corners.

To create a card with a shadow, the card_view card Elevation attribute can be used. Card View uses real elevation and dynamic shadows on Android 5.0 (API level 21) and above and falls back to a programmatic shadow implementation on earlier versions..

The following can be used to customize the appearance of the CardView widget:

- To set the corner radius in the layouts, the card_view: card Corner Radius attribute can be used.
- To set the corner radius in the code, the CardView.setRadius method can be used.
- To set the background color of a card, the card_view: card Background Color attribute should be used.

The following code example shows you how to include a CardView widget in to the layout.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    ... >
    <!-- A CardView that contains a TextView -->
    <android.support.v7.widget.CardView
        xmlns:card_view="http://schemas.android.com/apk/res-auto"
        android:id="@+id/card_view"
        android:layout_gravity="center"
        android:layout_width="200dp"
        android:layout_height="200dp"
        card_view:cardCornerRadius="4dp">

        <TextView
            android:id="@+id/info_text"
            android:layout_width="match_parent"
            android:layout_height="match_parent" />
```

```
    </android.support.v7.widget.CardView>
</LinearLayout>
```

Add Dependencies

The RecyclerView and CardView widgets are part of the v7 Support Libraries. To use these widgets in the project, Gradle dependencies have to be added to the app's module:

```
dependencies {
    ...
    compile 'com.android.support:cardview-v7:21.0.+'
    compile 'com.android.support:recyclerview-v7:21.0.+'
}
```

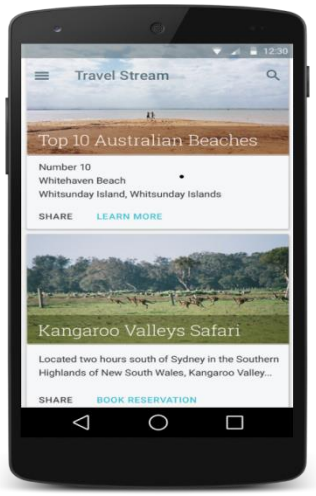

Figure 13

Solved example

1. Write a simple program using android listview.

XML file:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical" >

       <ListView
           android:id="@+id/list"
           android:layout_height="wrap_content"
           android:layout_width="match_parent">
       </ListView>

</LinearLayout>
```

JAVA file:

```
public class ListViewAndroidExample extends Activity {
      ListView listView ;

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

         // Get ListView object from xml
         listView = (ListView) findViewById(R.id.list);

         // Defined Array values to show in ListView
         String[] values = new String[] { "Android List View",
                                  "Adapter implementation",
                                  "Simple List View In Android",
                                  "Create List View Android",
                                  "Android Example",
                                  "List View Source Code",
                                  "List View Array Adapter",
                                  "Android Example List View"
                                 };

         // Define a new Adapter
```

```
        // First parameter - Context
        // Second parameter - Layout for the row
        // Third parameter - ID of the TextView to which the data is written
        // Forth - the Array of data
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
          android.R.layout.simple_list_item_1, android.R.id.text1, values);

        // Assign adapter to ListView
        listView.setAdapter(adapter);

        // ListView Item Click Listener
        listView.setOnItemClickListener(new OnItemClickListener() {

            @Override
            public void onItemClick(AdapterView<?> parent, View view,
              int position, long id) {

             // ListView Clicked item index
             int itemPosition     = position;

             // ListView Clicked item value
             String  itemValue    = (String) listView.getItemAtPosition(position);

              // Show Alert
              Toast.makeText(getApplicationContext(),
                "Position :"+itemPosition+"  ListItem : " +itemValue ,
Toast.LENGTH_LONG)
                .show();
            }
         });
      }
   }
```

Adapters are used to provide the data to the ListView

Parameters:

- simple_list_item_1 : Android internal layout view
- android.R.id.text1 : In Android internal layout view already defined text fields to show data
- values : User defined data array.

Exercise questions

1. Write a program to list the grocery item and print the item selected.
2. Develop an application that encrypt the TextField given by user.
3. Develop an application to change background based on user selection.

Additional questions

1. Develop a simple calculator application.

LAB 3:

Title: Program/App Development
Aim: Develop an application using the concept learnt in lab 1 & lab 2.

Sample Exercise Questions:

1. Develop an application for image gallery
2. Write a program for a feedback system.

LAB 4:

Title: Android class: Intent, AlertDialog and NotificationManager.

Aim: Familiarization of Intent, AlertDialog and NotificationManager class in android.

Intent

An Intent is a messaging object that can be used to request an action from another app component. Although intents facilitate communication between components in several ways, there are three fundamental use cases:

- Starting an activity
  An Activity represents a single screen in an app. A new instance of an Activity can be started by passing an Intent to startActivity(). The Intent describes the activity to start and carries any necessary data.
  If the result has to be received from the activity when it finishes, call startActivityForResult(). The activity receives the result as a separate Intent object in the activity's onActivityResult() callback. For more information, see the Activities guide.
- Starting a service
  A Service is a component that performs operations in the background without a user interface. With Android 5.0 (API level 21) and later, can be started with JobScheduler. For more information about JobScheduler, see its API-reference documentation.
  For versions earlier than Android 5.0 (API level 21), a service can be started by using methods of the Service class. A service can be started to perform a one-time operation (such as downloading a file) by passing an Intent to startService(). The Intent describes the service to start and carries any necessary data.
  If the service is designed with a client-server interface, the service can be binded from another component by passing an Intent to bindService(). For more information, see the Services guide.
- Delivering a broadcast
  A broadcast is a message that any app can receive. The system delivers various broadcasts for system events, such as when the system boots up or the device starts charging. A broadcast to other apps can be delivered by passing an Intent to sendBroadcast(), sendOrderedBroadcast(), or sendStickyBroadcast().

There are two types of intents:

- Explicit intents specify the component to start by name (the fully-qualified class name). Typically an explicit intent is used to start a component in the app, because the class name of the activity or service required to start. For example, new activity in response to a user action can be started or a service to download a file in the background can be started.
- 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 the user location on a map is to be shown, an implicit intent can be used to request another capable app to show a specified location on a map.

Alert Dialog

A Dialog is small window that prompts the user to a decision or enter additional information.
To create an alert dialog, Make an object of AlertDialogBuilder which is an inner class of AlertDialog. Its syntax is given below

```
AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(this);
```

Now you have to set the positive (yes) or negative (no) button using the object of the AlertDialogBuilder class. Its syntax is

```
alertDialogBuilder.setPositiveButton(CharSequence text,
  DialogInterface.OnClickListener listener)
alertDialogBuilder.setNegativeButton(CharSequence text,
  DialogInterface.OnClickListener listener)
```

After creating and setting the dialog builder , you will create an alert dialog by calling the create() method of the builder class. Its syntax is

```
AlertDialog alertDialog = alertDialogBuilder.create();
alertDialog.show();
```

This will create the alert dialog and will show it on the screen.

Notification
Android allows to put notification into the titlebar of your application. The user can expand the notification bar and by selecting the notification the user can trigger another activity.

Setting up Notifications

Notifications in Android are represented by the Notification class. To create notifications the NotificationManager class can be used which can be received from the Context, e.g. an *activity* or a *service*, via the getSystemService() method.

```
NotificationManager notificationManager = (NotificationManager)
    getSystemService(NOTIFICATION_SERVICE);
```

The Notification.Builder provides an builder interface to create an Notification object. A PendingIntent can be used to specify the action which should be performed once the user select the notification.

The Notification.Builder allows creation/adding of three buttons with definable actions to the notification.

```
// prepare intent which is triggered if the
// notification is selected
Intent intent = new Intent(this, NotificationReceiver.class);
// use System.currentTimeMillis() to have a unique ID for the pending intent
PendingIntent pIntent = PendingIntent.getActivity(this, (int) System.currentTimeMillis(),
intent, 0);

// build notification
// the addAction re-use the same intent to keep the example short
Notification n  = new Notification.Builder(this)
                    .setContentTitle("New mail from " + "test@gmail.com")
                    .setContentText("Subject")
                    .setSmallIcon(R.drawable.icon)
                    .setContentIntent(pIntent)
                    .setAutoCancel(true)
                    .addAction(R.drawable.icon, "Call", pIntent)
                    .addAction(R.drawable.icon, "More", pIntent)
                    .addAction(R.drawable.icon, "And more", pIntent).build();
NotificationManager notificationManager =
    (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
notificationManager.notify(0, n);
```

Solved example

1. Write a simple program to display an alert box.

XML file:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >

   <Button
      android:id="@+id/buttonAlert"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Show Alert Box" />

</LinearLayout>
```

JAVA file:

```
import android.app.Activity;
import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class MainActivity extends Activity {

        final Context context = this;
        private Button button;

        public void onCreate(Bundle savedInstanceState) {

          super.onCreate(savedInstanceState);
          setContentView(R.layout.main);

          button = (Button) findViewById(R.id.buttonAlert);

          // add button listener
```

```
		button.setOnClickListener(new OnClickListener() {

		@Override
		public void onClick(View arg0) {

			AlertDialog.Builder alertDialogBuilder = new
AlertDialog.Builder(
				context);

			// set title
			alertDialogBuilder.setTitle("Your Title");

			// set dialog message
			alertDialogBuilder
				.setMessage("Click yes to exit!")
				.setCancelable(false)
				.setPositiveButton("Yes",new
DialogInterface.OnClickListener() {
					public void onClick(DialogInterface dialog,int id)
{
						// if this button is clicked, close
						// current activity
						MainActivity.this.finish();
					}
				  })
				.setNegativeButton("No",new
DialogInterface.OnClickListener() {
					public void onClick(DialogInterface dialog,int id)
{
						// if this button is clicked, just close
						// the dialog box and do nothing
						dialog.cancel();
					}
				});
				// create alert dialog
				AlertDialog alertDialog = alertDialogBuilder.create();
				// show it
				alertDialog.show();
			}
		});
```

```
        }
}
```

Exercise questions

1. Develop a simple quiz application considering the following requirement:
   a. Prompt alert dialog to user before submitting the quiz.
   b. quiz result is displayed in new activity.

Additional question

1. Develop a simple alarm app(using NotificationManager)

LAB 5:
Title: Android Media Player.
Aim: Familiarization of using media files in application development.

Android provides many ways to control playback of audio/video files and streams. One of this way is through a class called **MediaPlayer**.

Android is providing MediaPlayer class to access built-in mediaplayer services like playing audio,video e.t.c. In order to use MediaPlayer, a static Method **create()** of this class should be called. This method returns an instance of MediaPlayer class. Its syntax is as follows −

```
MediaPlayer mediaPlayer = MediaPlayer.create(this, R.raw.song);
```

The second parameter is the name of the song that is selected for play. Make a new folder under the project with name **raw** and place the music file into it.

Once the Mediaplayer object is created and then call methods to start or stop the music. These methods are listed below.

```
mediaPlayer.start();

mediaPlayer.pause();
```

On call to **start()** method, the music will start playing from the beginning. If this method is called again after the **pause()** method, the music would start playing from where it is left and not from the beginning.

In order to start music from the beginning, call **reset()** method. Its syntax is given below.

```
mediaPlayer.reset();
```

Apart from the start and pause method, there are other methods provided by this class for better dealing with audio/video files. These methods are listed below –

Sr.No	Method & description
1	**isPlaying()** This method just returns true/false indicating the song is playing or not
2	**seekTo(position)** This method takes an integer, and move song to that particular position millisecond
3	**getCurrentPosition()** This method returns the current position of song in milliseconds
4	**getDuration()** This method returns the total time duration of song in milliseconds
5	**reset()** This method resets the media player
6	**release()** This method releases any resource attached with MediaPlayer object
7	**setVolume(float leftVolume, float rightVolume)** This method sets the up down volume for this player
8	**setDataSource(FileDescriptor fd)** This method sets the data source of audio/video file
9	**selectTrack(int index)** This method takes an integer, and select the track from the list on that particular index
10	**getTrackInfo()** This method returns an array of track information

Solved Example

1. Develop a simple media player app.

MainActivity.java.

```
import android.app.Activity;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.os.Handler;
import android.view.View;

import android.widget.Button;
import android.widget.ImageView;
import android.widget.SeekBar;
import android.widget.TextView;
import android.widget.Toast;
import java.util.concurrent.TimeUnit;


public class MainActivity extends Activity {
   private Button b1,b2,b3,b4;
   private ImageView iv;
   private MediaPlayer mediaPlayer;

   private double startTime = 0;
   private double finalTime = 0;

   private Handler myHandler = new Handler();;
   private int forwardTime = 5000;
   private int backwardTime = 5000;
   private SeekBar seekbar;
   private TextView tx1,tx2,tx3;

   public static int oneTimeOnly = 0;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      b1 = (Button) findViewById(R.id.button);
      b2 = (Button) findViewById(R.id.button2);
      b3 = (Button)findViewById(R.id.button3);
      b4 = (Button)findViewById(R.id.button4);
      iv = (ImageView)findViewById(R.id.imageView);
```

```
        tx1 = (TextView)findViewById(R.id.textView2);
        tx2 = (TextView)findViewById(R.id.textView3);
        tx3 = (TextView)findViewById(R.id.textView4);
        tx3.setText("Song.mp3");

        mediaPlayer = MediaPlayer.create(this, R.raw.song);
        seekbar = (SeekBar)findViewById(R.id.seekBar);
        seekbar.setClickable(false);
        b2.setEnabled(false);

        b3.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Toast.makeText(getApplicationContext(), "Playing
                    sound",Toast.LENGTH_SHORT).show();
                mediaPlayer.start();

                finalTime = mediaPlayer.getDuration();
                startTime = mediaPlayer.getCurrentPosition();

                if (oneTimeOnly == 0) {
                    seekbar.setMax((int) finalTime);
                    oneTimeOnly = 1;
                }

                tx2.setText(String.format("%d min, %d sec",
                    TimeUnit.MILLISECONDS.toMinutes((long) finalTime),
                    TimeUnit.MILLISECONDS.toSeconds((long) finalTime) -

TimeUnit.MINUTES.toSeconds(TimeUnit.MILLISECONDS.toMinutes((long)
                        finalTime)))
                );

                tx1.setText(String.format("%d min, %d sec",
                    TimeUnit.MILLISECONDS.toMinutes((long) startTime),
                    TimeUnit.MILLISECONDS.toSeconds((long) startTime) -

TimeUnit.MINUTES.toSeconds(TimeUnit.MILLISECONDS.toMinutes((long)
                        startTime)))
                );

                seekbar.setProgress((int)startTime);
                myHandler.postDelayed(UpdateSongTime,100);
                b2.setEnabled(true);
                b3.setEnabled(false);
            }
```

```
      });

      b2.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            Toast.makeText(getApplicationContext(), "Pausing
               sound",Toast.LENGTH_SHORT).show();
            mediaPlayer.pause();
            b2.setEnabled(false);
            b3.setEnabled(true);
         }
      });

      b1.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            int temp = (int)startTime;

            if((temp+forwardTime)<=finalTime){
               startTime = startTime + forwardTime;
               mediaPlayer.seekTo((int) startTime);
               Toast.makeText(getApplicationContext(),"You have Jumped
forward 5
                  seconds",Toast.LENGTH_SHORT).show();
            }else{
               Toast.makeText(getApplicationContext(),"Cannot jump forward 5
                  seconds",Toast.LENGTH_SHORT).show();
            }
         }
      });

      b4.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            int temp = (int)startTime;

            if((temp-backwardTime)>0){
               startTime = startTime - backwardTime;
               mediaPlayer.seekTo((int) startTime);
               Toast.makeText(getApplicationContext(),"You have Jumped
backward 5
                  seconds",Toast.LENGTH_SHORT).show();
            }else{
               Toast.makeText(getApplicationContext(),"Cannot jump backward 5
                  seconds",Toast.LENGTH_SHORT).show();
            }
```

```
            }
        });
    }

    private Runnable UpdateSongTime = new Runnable() {
        public void run() {
            startTime = mediaPlayer.getCurrentPosition();
            tx1.setText(String.format("%d min, %d sec",
                TimeUnit.MILLISECONDS.toMinutes((long) startTime),
                TimeUnit.MILLISECONDS.toSeconds((long) startTime) -
                TimeUnit.MINUTES.toSeconds(TimeUnit.MILLISECONDS.
                toMinutes((long) startTime)))
            );
            seekbar.setProgress((int)startTime);
            myHandler.postDelayed(this, 100);
        }
    };
}
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
tools:context=".MainActivity">

    <TextView android:text="Music Palyer" android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/textview"
        android:textSize="35dp"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Media player"
        android:id="@+id/textView"
        android:layout_below="@+id/textview"
        android:layout_centerHorizontal="true"
        android:textColor="#ff7aff24"
```

```
    android:textSize="35dp" />

<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/imageView"
    android:layout_below="@+id/textView"
    android:layout_centerHorizontal="true"
    android:src="@drawable/abc"/>

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/forward"
    android:id="@+id/button"
    android:layout_alignParentBottom="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/pause"
    android:id="@+id/button2"
    android:layout_alignParentBottom="true"
    android:layout_alignLeft="@+id/imageView"
    android:layout_alignStart="@+id/imageView" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/back"
    android:id="@+id/button3"
    android:layout_alignTop="@+id/button2"
    android:layout_toRightOf="@+id/button2"
    android:layout_toEndOf="@+id/button2" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/rewind"
    android:id="@+id/button4"
    android:layout_alignTop="@+id/button3"
    android:layout_toRightOf="@+id/button3"
    android:layout_toEndOf="@+id/button3" />
```

```
   <SeekBar
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/seekBar"
      android:layout_alignLeft="@+id/textview"
      android:layout_alignStart="@+id/textview"
      android:layout_alignRight="@+id/textview"
      android:layout_alignEnd="@+id/textview"
      android:layout_above="@+id/button" />

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textAppearance="?android:attr/textAppearanceSmall"
      android:text="Small Text"
      android:id="@+id/textView2"
      android:layout_above="@+id/seekBar"
      android:layout_toLeftOf="@+id/textView"
      android:layout_toStartOf="@+id/textView" />

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textAppearance="?android:attr/textAppearanceSmall"
      android:text="Small Text"
      android:id="@+id/textView3"
      android:layout_above="@+id/seekBar"
      android:layout_alignRight="@+id/button4"
      android:layout_alignEnd="@+id/button4" />

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textAppearance="?android:attr/textAppearanceMedium"
      android:text="Medium Text"
      android:id="@+id/textView4"
      android:layout_alignBaseline="@+id/textView2"
      android:layout_alignBottom="@+id/textView2"
      android:layout_centerHorizontal="true" />

</RelativeLayout>
```

string.xml

```
<resources>
   <string name="app_name">My Application</string>
   <string name="back"><![CDATA[<]]></string>
```

```
   <string name="rewind"><![CDATA[<<]]></string>
   <string name="forward"><![CDATA[>>]]></string>
   <string name="pause">||</string>
</resources>
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.sairamkrishna.myapplication" >

   <application
      android:allowBackup="true"
      android:icon="@drawable/ic_launcher"
      android:label="@string/app_name"
      android:theme="@style/AppTheme" >

      <activity
         android:name="com.example.sairamkrishna.myapplication.MainActivity"
         android:label="@string/app_name" >

         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>

      </activity>

   </application>
</manifest>
```

Exercise questions:

1. Develop a media player application with basic functionality like start, pause, forward and rewind.

LAB 6:
Title: Program /App Development.
Aim: Develop an application using the concept learnt in lab 1 to lab 5.

Sample Exercise Questions:
1. Develop a simple alarm app using Notification manager class.
2. Develop an app to play song based on user selection from the list.

LAB 7:

Title: Creating the database and inserting the values

Aim: Familiarization of SQLite concepts: database creation and inserting values into it.

Use of SQlite database:

Saving data to a database is ideal for repeating or structured data, such as contact information. With the knowledge of SQL databases in general and helps to get started with SQLite databases on Android. SQLite is a opensource SQL database that stores data to a text file on a device. Android comes in with built in SQLite database implementation.SQLite supports all the relational database features. In order to access this database, There is no need to establish any kind of connections for it like JDBC,ODBC e.t.c. The APIs required to use a database on Android are available in the android.database.sqlite package.

Define a Schema and Contract:

One of the main principles of SQL databases is the schema: a formal declaration of how the database is organized. The schema is reflected in the SQL statements that are used to create the database.

A contract class, explicitly specifies the layout of the schema in a systematic and self-documenting way. A contract class is a container for constants that define names for URIs, tables, and columns. The contract class allows to use the same constants across all the other classes in the same package. This lets to change a column name in one place and have it propagate throughout the code. A good way to organize a contract class is to put definitions that are global to the whole database in the root level of the class. Then create an inner class for each table that enumerates its columns.

<Database – Creation>

To create a database openOrCreateDatabase method is used with the database name and mode as a parameter. It returns an instance of SQLite database object. Its syntax is given below

SQLiteDatabase mydatabase = openOrCreateDatabase("your database name",MODE_PRIVATE,null);

Apart from this , the other functions available in the database package are listed below

Method & Description

- openDatabase(String path, SQLiteDatabase.CursorFactory factory, int flags, DatabaseErrorHandler errorHandler) This method only opens the existing database with the appropriate flag mode. The common flags mode could be OPEN_READWRITE OPEN_READONLY
- openDatabase(String path, SQLiteDatabase.CursorFactory factory, int flags) It is similar to the above method as it also opens the existing database but it does not define any handler to handle the errors of databases
- openOrCreateDatabase(String path, SQLiteDatabase.CursorFactory factory) It not only opens but create the database if it not exists. This method is equivalent to openDatabase method.
- openOrCreateDatabase(File file, SQLiteDatabase.CursorFactory factory) This method is similar to above method but it takes the File object as a path rather then a string. It is equivalent to file.getPath()

<Database – Insertion>

To create table or to insert data into table using execSQL method defined in SQLiteDatabase class. Its syntax is given below

```
mydatabase.execSQL("CREATE TABLE IF NOT EXISTS
Example_Table(Username      VARCHAR,Password VARCHAR);");
mydatabase.execSQL("INSERT INTO Example_Table
VALUES('admin','admin');");
```

This will insert some values into our table in our database.

<Database – Fetching>

An object of the Cursor class is used to retrieve the data from the database. The method used is called rawQuery and it will return a resultset with the cursor pointing to the table.

```
Cursor resultSet = mydatbase.rawQuery("Select * from Example_Table ",null);
resultSet.moveToFirst();
String username = resultSet.getString(1);
```

```
String password = resultSet.getString(2);
```

Exercise questions:

1. Create a simple Student Management System application that accepts a student's roll number, name and marks and adds these details to student table. (Assume all fields of VARCHAR data type, which is a variable length character string).Perform the following operation on the database
    a. Allow the user to add the student details.
    b. Allow the user to delete and update the details.
    c. Allow the user to display or view the details.

LAB 8:

Title: Fetching the data based on certain defined conditions. Database Helper class

Aim: Familiarization of SQLite concepts: Fetching the data.

<Database – Fetching>

The functions available in the Cursor class that allows us to effectively retrieve the data. that includes

- getColumnCount() :This method return the total number of columns of the table.
- getColumnIndex(String columnName) :This method returns the index number of a column by specifying the name of the column.
- getColumnName(int columnIndex) :This method returns the name of the column by specifying the index of the column.
- getColumnNames() :This method returns the array of all the column names of the table.
- getCount() :This method returns the total number of rows in the cursor
- getPosition():This method returns the current position of the cursor in the table
- isClosed():This method returns true if the cursor is closed and return false otherwise

Database - Helper class

For managing all the operations related to the database, a helper class has been given and is called SQLiteOpenHelper. It automatically manages the creation and update of the database.This class makes it easy for ContentProvider implementations to defer opening and upgrading the database until first use, to avoid blocking application startup with long-running database upgrades.

The following is its syntax:

```
public class DBHelper extends SQLiteOpenHelper {
public DBHelper(){
super(context,DATABASE_NAME,null,1);
}
```

i) **close:**Close any open database object.

ii) **getDatabaseName**

String getDatabaseName ()

Return the name of the SQLite database being opened, as given to the constructor.

iii) **getReadableDatabase**

SQLiteDatabase getReadableDatabase ()

Create and/or open a database. This will be the same object returned by getWritableDatabase() unless some problem, such as a full disk, requires the database to be opened read-only. In that case, a read-only database object will be returned.

iv) **getWritableDatabase**

SQLiteDatabase getWritableDatabase ()

Create and/or open a database that will be used for reading and writing. The first time this is called, the database will be opened and onCreate(SQLiteDatabase), onUpgrade(SQLiteDatabase, int, int) and/or onOpen(SQLiteDatabase) will be called.

Once opened successfully, the database is cached, so you can call this method every time you need to write to the database. (Make sure to call close() when you no longer need the database.) Errors such as bad permissions or a full disk may cause this method to fail, but future attempts may succeed if the problem is fixed.

v) **onUpgrade**

void onUpgrade (SQLiteDatabase db, int oldVersion, int newVersion)

Called when the database needs to be upgraded. The implementation should use this method to drop tables, add tables, or do anything else it needs to upgrade to the new schema version.

This method executes within a transaction. If an exception is thrown, all changes will automatically be rolled back.

vi) onDowngrade

void onDowngrade (SQLiteDatabase db, int oldVersion, int newVersion)

Called when the database needs to be downgraded. This is strictly similar to onUpgrade(SQLiteDatabase, int, int) method, but is called whenever current version is newer than requested one. However, this method is not abstract, so it is not mandatory for a customer to implement it. If not overridden, default implementation will reject downgrade and throws SQLiteException

This method executes within a transaction. If an exception is thrown, all changes will automatically be rolled back.

vii) onConfigure

void onConfigure (SQLiteDatabase db)

Called when the database connection is being configured, to enable features such as write-ahead logging or foreign key support.

This method is called before onCreate(SQLiteDatabase), onUpgrade(SQLiteDatabase, int, int), onDowngrade(SQLiteDatabase, int, int), or onOpen(SQLiteDatabase) are called. It should not modify the database except to configure the database connection as required.

This method should only call methods that configure the parameters of the database connection, such as enableWriteAheadLogging() setForeignKeyConstraintsEnabled(boolean), setLocale(Locale), setMaximumSize(long), or executing PRAGMA statements.

viii) onCreate

void onCreate (SQLiteDatabase db)

Called when the database is created for the first time. This is where the creation of tables and the initial population of the tables should happen.

Exercise questions:

1. Develop a simple application for billing and invoicing the products which has proper attributes.
2. For the above application find the total number of products available along with its price and also list the maximum priced and minimum priced product.
3. Develop a clinical management system where appointment is given to the authorized users based on availability of the doctor.

Additional questions:

1. Develop an application which allows to manage budget that include set budget & update budget as well as set budget limit & update budget limit. It manages daily expense as well as manage category of the expense. It also gives the details of the expense & also reminds for any task like bill paying.

Lab 9:

Title: Program/App Development.

Aim: Develop an application using the concept learnt in lab 1 to lab8.

Sample Exercise Questions:

1. Develop an app for conducting survey.
2. Develop an app for airline ticket reservation system.

Lab 10:
Title: XAMARIN

Aim: Develop an application using Xamarin.

XAMARIN:

Xamarin is a popular cross-platform software that lets developers write a code base in the C# programming language and .NET framework, then it compiles the code for the appropriate native UI—iOS, Android, or Windows. Xamarin allows you to build, test, and monitor your app in the cloud, too. A newer addition to the platform is Xamarin Forms, a cross-platform graphical user interface (GUI) framework.

System Requirements (For Windows)

i. At least 2GB of RAM and running Windows 7 or higher (Windows 8-10 is highly recommended)
ii. Visual Studio 2012 Professional or higher
iii. Xamarin for Visual Studio

Installation on Windows

1. Download the Xamarin Installer from https://www.xamarin.com/download.
2. Should have installed Android SDK and Java SDK on the computer before running the Xamarin installer Run the downloaded installer to begin the installation process.
3. The Xamarin license agreement screen appears. Click the **Next** button to accept the agreement.
4. The installer will search for any missing components and prompt to download and install them.
5. After the Xamarin is installed Xamarin can be started .

Development of a simple Android application using Xamarin.

Step 1. Start a new instance of Visual Studio and go to **File → New → Project as in Figure 14**.

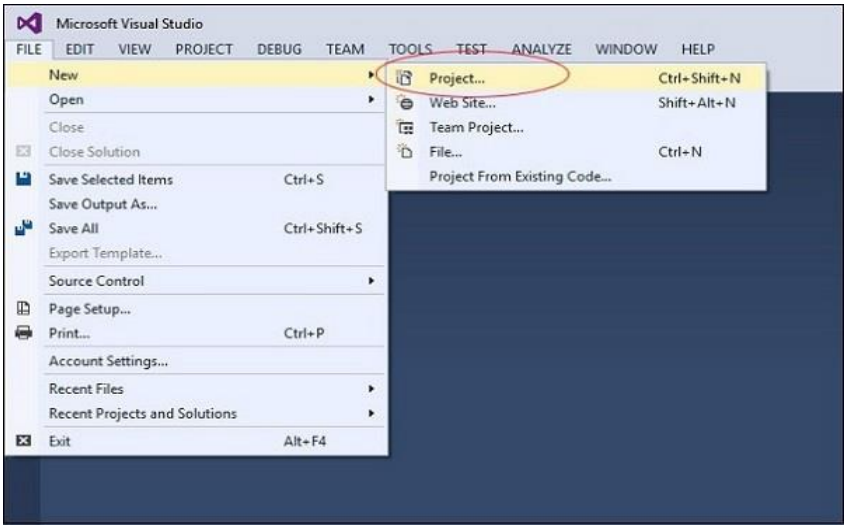

Figure 14

Step 2: On the Menu dialog box that appears, go to **Templates → Visual C# → Android → Blank App (Android)** as shown in Figure 15.

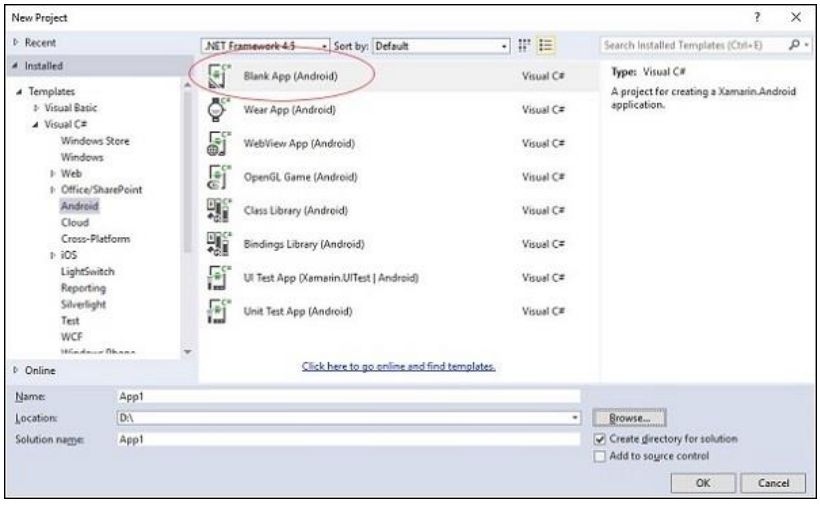

Figure 15

Step 3. Give an appropriate name for the application. In this example, it has been named **"helloWorld"** and then save it in the default location provided.

Step 4. Click the OK button for the new **"helloXamarin"** project to load.

Step 5. On the **solution**, open **Resources → layout → Main.axml** file. Switch from Design View and go to the **Source** file and type the following lines of code to build your app.

```
<?xml version = "1.0" encoding = "utf-8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
  android:orientation = "vertical"
  android:background = "#d3d3d3"
  android:layout_width = "fill_parent"
  android:layout_height = "fill_parent">
  <TextView
    android:text = "@string/HelloXamarin"
    android:textAppearance = "?android:attr/textAppearanceLarge"
    android:layout_width = "match_parent"
    android:layout_height = "wrap_content"
    android:id = "@+id/textView2"
    android:textColor = "@android:color/black" />
</LinearLayout>
```

In the above code, a new Android textview is created. Next, open the folder values and double-click Strings.xml to open it. Here, store information and values about the button created above.

```
<?xml version = "1.0" encoding = "utf-8"?>
<resources>
  <string name = "HelloXamarin">Hello World, I am Xamarin!</string>
  <string name = "ApplicationName">helloWorld</string>
</resources>
```

Step 6. Open **MainActivity.cs** file and replace the existing code with the following lines of code.

```
using System;
using Android.App;
using Android.Content;
```

```
using Android.Runtime;
using Android.Views;
```

```
using Android.Widget;
using Android.OS;

namespace HelloXamarin {
  public class MainActivity : Activity {
    protected override void OnCreate(Bundle bundle) {
      base.OnCreate(bundle);
      SetContentView(Resource.Layout.Main);
    }
  }
}
```

Step 7. Save the application. Build and then run it to display the created app in an Android Emulator(Figure 16)

Figure 16

Sample Exercise Questions:

1. Develop a cross platform quiz application using XAMARIN.
2. Develop a cross platform address book application using XAMARIN.

Lab 11:

Title: Project work.

Aim: Development of a mobile app and deployment and test of the same on a mobile.

The mobile app should be designed using the following concepts:

i. **Motion sensors:** These sensors measure acceleration forces and rotational forces along three axes. This category includes accelerometers, gravity sensors, gyroscopes, and rotational vector sensors.

ii. **Environmental sensors:** These sensors measure various environmental parameters, such as ambient air temperature and pressure, illumination, and humidity. This category includes barometers, photometers, and thermometers.

ii. **Position sensors:** These sensors measure the physical position of a device. This category includes orientation sensors and magnetometers.