6/10/23

ASSIGNMENT 2

PAGE NO.: 29

Unit -3 & 4 Displaying Pictures and Menus with Views & Android Data Storage Mechanism

Q.1 Explain different types of menus in android with example.

-4 Menus provide a way for users to access various app feature and options. There are different types of menu in Android, each serving a specific purpose and appearing in different contexts within an app.

i) Option Menu

The option menu is typically located in the top-right corner of the app's action bar. It is used to provide access to common actions within the app. It is created in the 'onCreateOptionsMenu' method of an activity. For Example It may include actions like Save, Share, Settings for note-taking app.

Syntax:

```
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.options_menu, menu);
    return true;
}
```

ii) Context Menu:

A context menu is a floating menu that appears when the user long-presses an item. It provides context-specific actions related to the selected item. For Example, in a messaging app, a long-press on a chat message might trigger a context menu with options like Copy, Delete, Forward, Reply, etc.

Syntax:
```
public void onCreateContextMenu (ContextMenu menu,
View v, ContextMenu.ContextMenuInfo menuInfo) {
    getMenuInflater().inflate(R.menu.context_menu, menu);
}
```

iii) Popup Menu

A popup menu is a modal menu that appears when the user taps on a specific UI element like a button or an icon. It is used to provide a set of actions relevant to that element. They can be anchored to a view. For Example, in a music player app, tapping on 'More' Button next to a song might display a popup menu with options like Add to Playlist, Download, Delete.

Syntax:
```
PopupMenu popupMenu = new PopupMenu(this, view);
getMenuInflater().inflate(R.menu.popup_menu,
    popupMenu.getMenu());
popupMenu.show();
```

Q.2 Explain Checkable menu items.

—ᵛ Checkable menu items are a type of menu item that can be toggled on and off by the user. They are often used in menus where the user needs to select one or more options and each options can have a checked or unchecked state. When a menu item is checked, it typically indicates that a particular option is active. Commonly used for filters, list, sorting options.

- Creating a checkable menu items :-

i) XML Code

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/check-menu"
        android:title="Enable Feature"
        android:checkable="true"
        android:checked="true"
        android:text="Enable" />
</menu>
```

ii) Java Code

```
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.check-menu:
            if (item.isChecked()) {
                item.setChecked(false);
            } else {
                item.setChecked(true);
            }
            return True;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

Q.3 Explain how to handle events in menus.

→ Handling events in menus involves responding to user interactions with menu items

'onOptionsItemSelected'

i) Option Menu : ~~To handle events~~ This method is called when a menu item is selected by a user.

eg:
```
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.menu_item_id:
            return True;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

ii) Context Menu : To handle events here, you first need to register the view that will trigger the context menu.

eg:
```
registerForContextMenu(view);
// create the context Menu before the below Handling.
public boolean onContextItemSelected(MenuItem item) {
    Switch (item.getItemId()) {
        case R.id.context_menu_id:
            return True;
        default:
            return super.onContextItemSelected(item);
    }
}
```

iii) Popup Menu : To handle events, you need to create an instance of 'PopupMenu', attach it to a view and set listener for item clicks.

```
// create a Popup Menu and attach it to a view
popupMenu.setOnMenuItemClickListener(new PopupMenu.
    OnMenuItemClickListener() {
```

```
public boolean onMenuItemClick(MenuItem item) {
    Switch (item.getItemId()) {
        case R.id.popup_menu_id:
            return True;
    }
    return False;
}
});
```

Q.4 Explain menu items based on intent.

—> Menu items can be based on intents to trigger specific actions within your app or even in other apps. This allows you to provide user with options to perform various tasks which extend the functionality of your app.

example: XML Code:

```
<menu xmlns:android="http://schemas.android.com/apk/
  res/android">
    <item
        android:id="@+id/open_browser"
        android:title="Open Browser"
        app:ShowAsAction="always"
        app:actionViewClass="android.widget.ImageView"
        app:intentCategory="android.intent.category.DEFAULT"
        app:actionProviderClass="android.widget.ShareAction"
        android:onClick="openBrowserIntent" />
</menu>
```

* JAVA Code :-

```
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.open_browser:
            Intent browserIntent = new Intent(Intent.ACTION_VIEW,
            Uri.parse("link"));
            startActivity(browserIntent);
            return True;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

Q.5 What is Shared Preferences? Explain with an example.

→ SharedPreferences is a mechanism for storing and retrieving simple data in key-value pairs. It allows you to store small amounts of data persistently across app launches. Often used for storing user preferences, settings and other lightweight data.

→ Here's how this works with an example:

i) Creating a SharedPreferences Instance.

```
SharedPreferences sharedPreferences = getSharedPreferences
    ("MyPrefs", Context.MODE_PRIVATE);
```

ii) Storing Data

```
SharedPreferences.Editor editor = sharedPreferences.edit();
editor.putString("User_name", "Tony");
editor.apply();
```

iii) Retrieving Data

```
String userName = sharedPrefecences.getString("user-name",
                  "DefaultName");
```

iv) Updating Data

```
SharedPreferences.Editor editor = sharedPreferences.edit();
editor.putString("user-name", "Pepper");
editor.apply();
```

v) Removing Data.

```
editor.remove("user-name");
editor.apply();
```

vi) Clearing All Data

```
editor.clear();
editor.apply();
```

Q.6 List out types of storage in android. Explain internal storage and external storage with examples. (code).

-> There are several types of storage options available for saving and managing data, which are:

i) Shared Preferences
ii) SQLite Databases
iii) Network Storage
iv) Internal Storage

It is a private storage area within the app's sandboxed environment and it is used to store app-specific data like cache files, sensitive information

which is not accessible or visible to other apps or users and does not require any special permissions

example:

```
String data = "Internal Storage Data";
try {
    FileOutputStream fos = openFileOutput("myFile.txt",
                                          Context.MODE_PRIVATE);
    fos.write(data.getBytes());
    fos.close();
} catch (IOException e) {
    e.printStackTrace();
}
try {
    FileInputStream fis = openFileInput("myFile.txt");
    InputStreamReader isr = new InputStreamReader(fis);
    BufferedReader br = new BufferedReader(isr);
    StringBuilder sb = new StringBuilder();
    String line;
    while ((line = br.readLine()) != null) {
        sb.append(line);
    }
    fis.close();
    String storedData = sb.toString();
    Log.d("Internal Storage", "Data from file: " + storedData);
} catch (IOException e) {
    e.printStackTrace();
}
```

(09) External Storage

It refers to the shared storage space that can be accessed by multiple apps and the user. It includes the device's external SD card or shared internal storage (accessible with appropriate permissions). Used to store data that needs to be shared. like media files, documents, downloads.

example:-

```
if (ContextCompat.checkSelfPermission(this, Manifest.
permission.WRITE_EXTENAL_STORAGE) != PackageManager.
                                    PERMISSION_GRANTED) {
    ActivityCompat.requestPermissions(this, new String[] {
      Manifest.permission.WRITE_EXTERNAL_STORAGE },
                                    REQUEST_CODE);
}
// Writing to External Storage
// Reading from External Storage
```

same as internal except you have to add a filepath

Q.7 write a code to insert Employee details (eid, ename, eaddress, edesignation) in SQLite database.

—> //CREATE A DATABASE

```
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
public class DatabaseHelper extends SQLiteOpenHelper {
    private static final String DATABASE_NAME = "Employee";
    private static final int DATABASE_VERSION = 1;

    private static final String TABLE_EMPLOYEE = "employees";
    private static final String COLUMN_ID = "eid";
```

```
private static final String COLUMN_NAME = "ename";
private static final String COLUMN_ADDRESS = "eaddress";
private static final String COLUMN_DESIGNATION = "edesignation";

private static final String TABLE_CREATE =
    "CREATE TABLE" + TABLE_EMPLOYEE + "(" +
        COLUMN_ID + "INTEGER PRIMARY KEY, " +
        COLUMN_NAME + "TEXT, " +
        COLUMN_ADDRESS + "TEXT," +
        COLUMN_DESIGNATION + " TEXT);";

public DataBaseHelper(Context context) {
  super(context, DATABASE_NAME, null, DATABASE_VERSION);
}
public void onCreate(SQLiteDatabase db) {
  db.execSQL(TABLE_CREATE);
}
// INSERT OPERATION
public long insertEmployee(int id, String name, String
        address, String designation) {
  ContentValues values = new ContentValues();
  values.put("ename", name);
  values.put("eaddress", address);
  values.put("edesignation", designation);
  values.put("eid", id);
  return database.insert("employees", null, values);
}
long insertedId = dataSource.insertEmployee(101,
  "John Wick", "123 Main St", "Manager");
dataSource.close()
```

Q.8 Write a code for CRUD operation to use SQLite database in android application. (Create, Read, Update, Delete).

```
// Same code written in 7, with column name 'id' and
'task_name' and further
// Create Operation
public long insertData (String name) {
  SQLiteDatabase db = this.getWritableDatabase();
  ContentValues values = new ContentValues();
  values.put(COLUMN_NAME, name);
  long newRowId = db.insert(TABLE_NAME, null, values); }

//Read Operation
public Cursor getAllData() {
  SQLiteDatabase db = this.getReadableDatabase();
  return db.query(TABLE_NAME, null, null, null, null, null); }

//Update Operation
public int update(int id, String newName) {
  SQLiteDatabase db = this.getWritableDatabase();
  ContentValues values = new ContentValues();
  values.put(COLUMN_NAME, newName);
  return db.update(TABLE_NAME, values, column_Id + "=?"); }

//Delete Operation.
public int deleteData(int id) {
  SQLiteDatabase db = this.getWritableDatabase();
  return db.delete(TABLE_NAME, COLUMN_ID + " = ?",
    new String[] {String.valueOf(id)}); }
```

Q.9 Write the significance, syntax and code of create table, insert(), update(), delete() and select() of SQLiteDatabase class

→ i) Create Table: used to define the structure of a database table, including column names and data types.

Syntax: CREATE TABLE table_name (
column_name datatype,
...);

example:
```
SQLiteDatabase db = dbHelper.getWritableDatabase();
String createTableQuery = "CREATE TABLE " +
        "Employee (" +
        "id INTEGER PRIMARY KEY," +
        "name TEXT," +
        "designation TEXT);";
db.execSQL(createTableQuery);
db.close();
```

ii) Insert: used to add new rows (records) of data into a table

Syntax: INSERT INTO table_name VALUES(value1, ...);

example:
```
ContentValues values = new ContentValues();
values.put("name", "John Wick");
values.put("designation", "Manager");
long newRowId = db.insert("Employee", null, values);
```

iii) Update: Used to modify existing data in one or more rows of a table.

Syntax: UPDATE table_name
SET column_name = value, ...
WHERE some_column = some_value;

example:
- ContentValues values = new ContentValues();
- values.put("name", "Updated Name");
- String selection = "id = ?";
- String[] selectionArgs = {"1"};
- int rowsUpdated = db.update("Employee", values, selection, selectionArgs);

iv) Delete: Used to remove rows from a table based on specified conditions.

Syntax: DELETE FROM table_name
WHERE some_column = some_value;

example:
String selection = "id = ?";
String[] selectionArgs = {"2"};
int rowsDeleted = db.delete("Employee", selection, selectionArgs);

v) Select: Used to retrieve data from one or more tables.

Syntax: SELECT column_name, ...
FROM table_name
WHERE some_column = some_value;

example: int id = cursor.getInt(cursor.getColumnIndex("id"));

Q.10 Write a code to play video on click on button.

* XML Code :-

```
<VideoView
    android:id="@+id/videoView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center" />
<Button
    android:id="@+id/button"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:text="Play Video" />
```

* JAVA Code :-

```
public class MainActivity extends AppCompatActivity {
    private VideoView videoView;
    private Button button;
    protected void onCreate(Bundle savedInstanceState) {
      Super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      videoView = findViewById(R.id.videoView);
      button = findViewById(R.id.button);
      button.setOnClickListener(new View.OnClickListener() {
          public void onClick(View v) {
              playVideo();
          }
      });
    }
private void playVideo() {
```

```
Uri video = Uri.parse("android.resource://" +
    getPackageName() + "/" + R.raw.sample_video);
Intent intent = new Intent(Intent.ACTION_VIEW);
intent.setDataAndType(video, "video/mp4");
}
}
```

Q.11 Explain Content Provider. What is the use of Content Provider ?

—↳ A Content Provider is a component of the Android framework that provides a consistent and secure way to share data between different applications. It acts as an intermediary that abstracts the underlying data source and allows other applications to interact with that data in a structured and controlled manner.

—↳ Uses of Content Provider :-

i) Data Sharing
ii) Data Abstraction
iii) Data Synchronization
iv) Built-in Content Providers
v) Custom Content Provider

Syntax:
```
public class MyContentProvider extends contentProvider {
    // database initialize and content provider
    method (query, insert, update, delete) comes here
}
```

Written 11/10/23

xxx x xxx x xxx