MOBILE APPLICATION DEVELOPMENT (MAD) : LAB 3

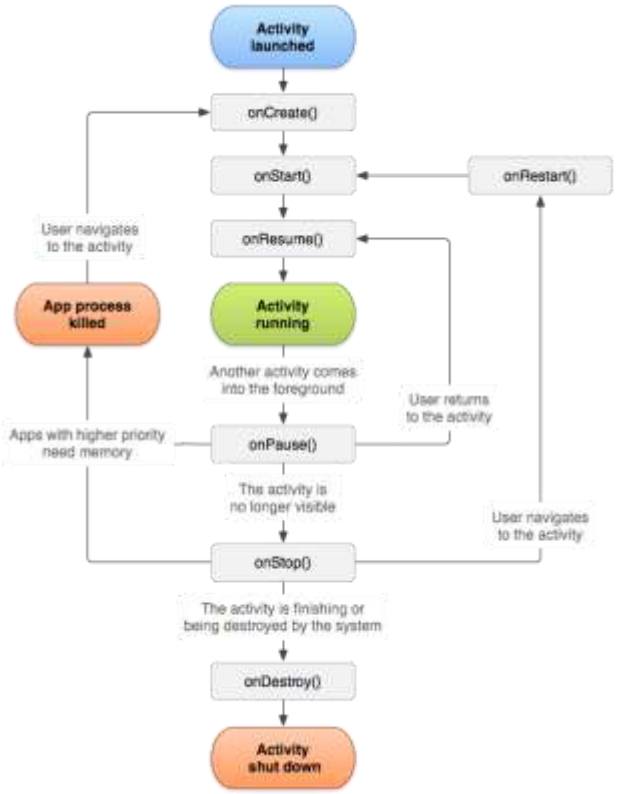
OBJECTIVES

- Understanding Android Application Life Cycle
- Understanding Android Intent Filter
- Practice Activities

OBJECTIVE 1 : Understanding Android Life Cycle

- Each activity in an Android application goes through its own lifecycle and relies on the implementation of a series of callback methods that the system calls when the activity transitions between various states of its lifecycle

  - The **`Activity`** class provides a number of callbacks that allow the activity to know that a state has changed
  - It's a way to navigate through the app using different callbacks throughout its duration till the app is alive.



MOBILE APPLICATION DEVELOPMENT (MAD) : LAB 3

OBJECTIVE 2 : Understanding Intent-Filter

- Intent Filter is way to sort out the intents, called using Implicit Intent.
- Structured description of Intent values to be matched.
- An Intent Filter can match against actions, categories, and data (either via its type, scheme, and/or path) in an Intent

Filter Rules

- An **Intent Filter** to match an Intent, three conditions must hold: the **action** and **category** must match, and the **data** (both the **data type** and **data scheme+authority+path** if specified) must match.
- **Action** matches if any of the given values match the Intent action; if the filter specifies no actions, then it will only match Intents that do not contain an action.
- **Categories** match if all of the categories in the Intent match categories given in the filter.

```
<intent-filter>
    <action android:name="android.intent.action.VIEW" />

    <category android:name="android.intent.category.DEFAULT" />

    <data android:scheme="http" />
</intent-filter>
```

MOBILE APPLICATION DEVELOPMENT (MAD) : LAB 3

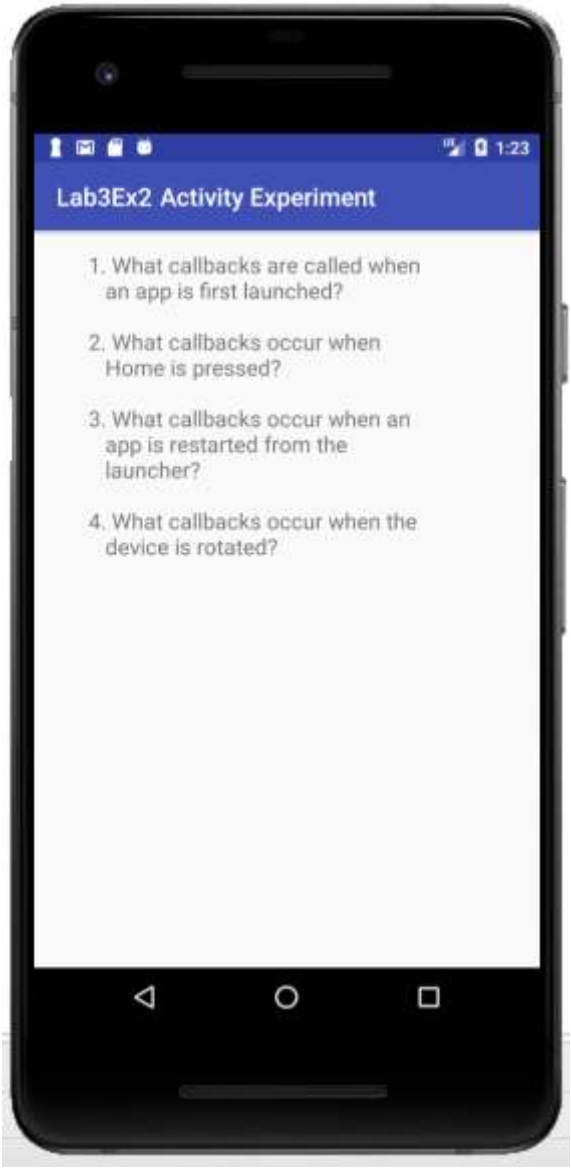
PRACTICE ACTIVITIES:

Activity 1 : Create an app that explores the life-cycle of an activity.

- The main objective of this app is to experience callback methods first hand.

Write definition of all these methods and observe the results.

- onCreate()
- onStart()
- onResume()
- onPause()
- onStop()
- onRestart()
- onDestroy()

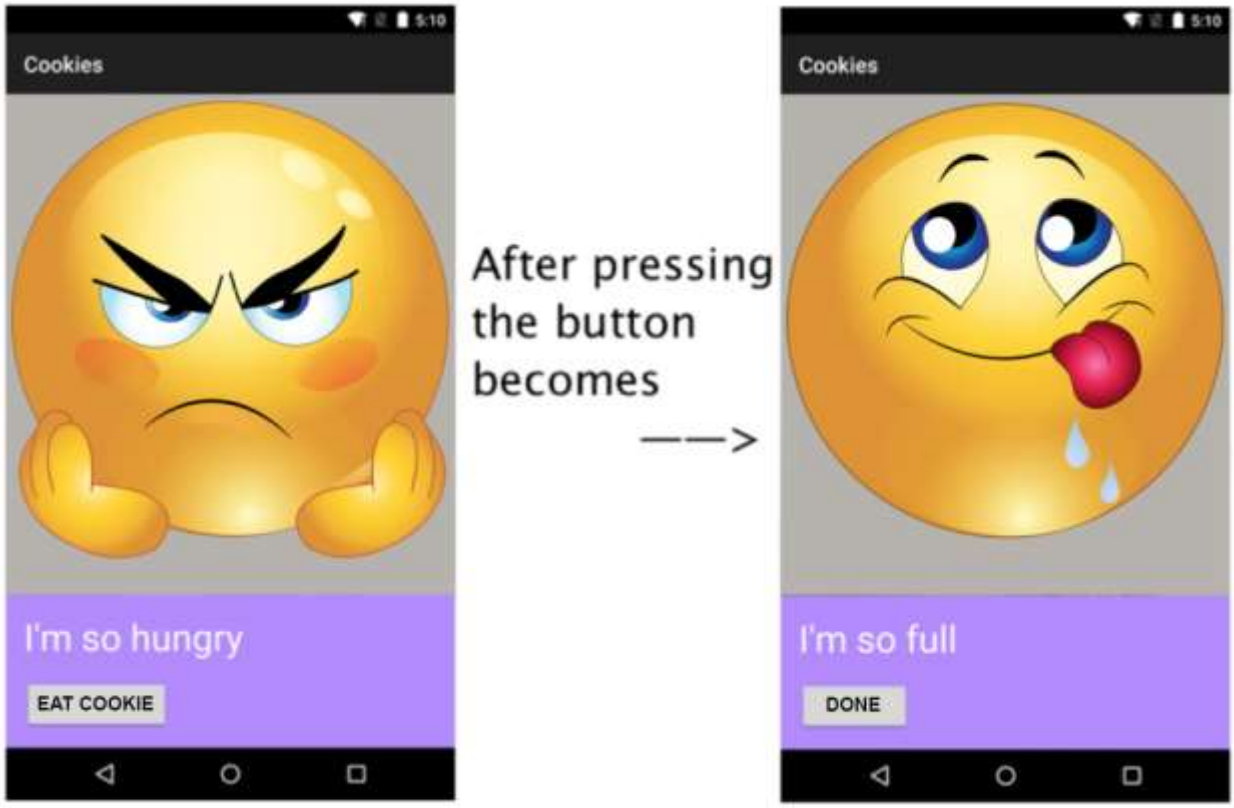


MOBILE APPLICATION DEVELOPMENT (MAD) : LAB 3

Activity 2 : Develop an app to describe the mood.

- Add following Components

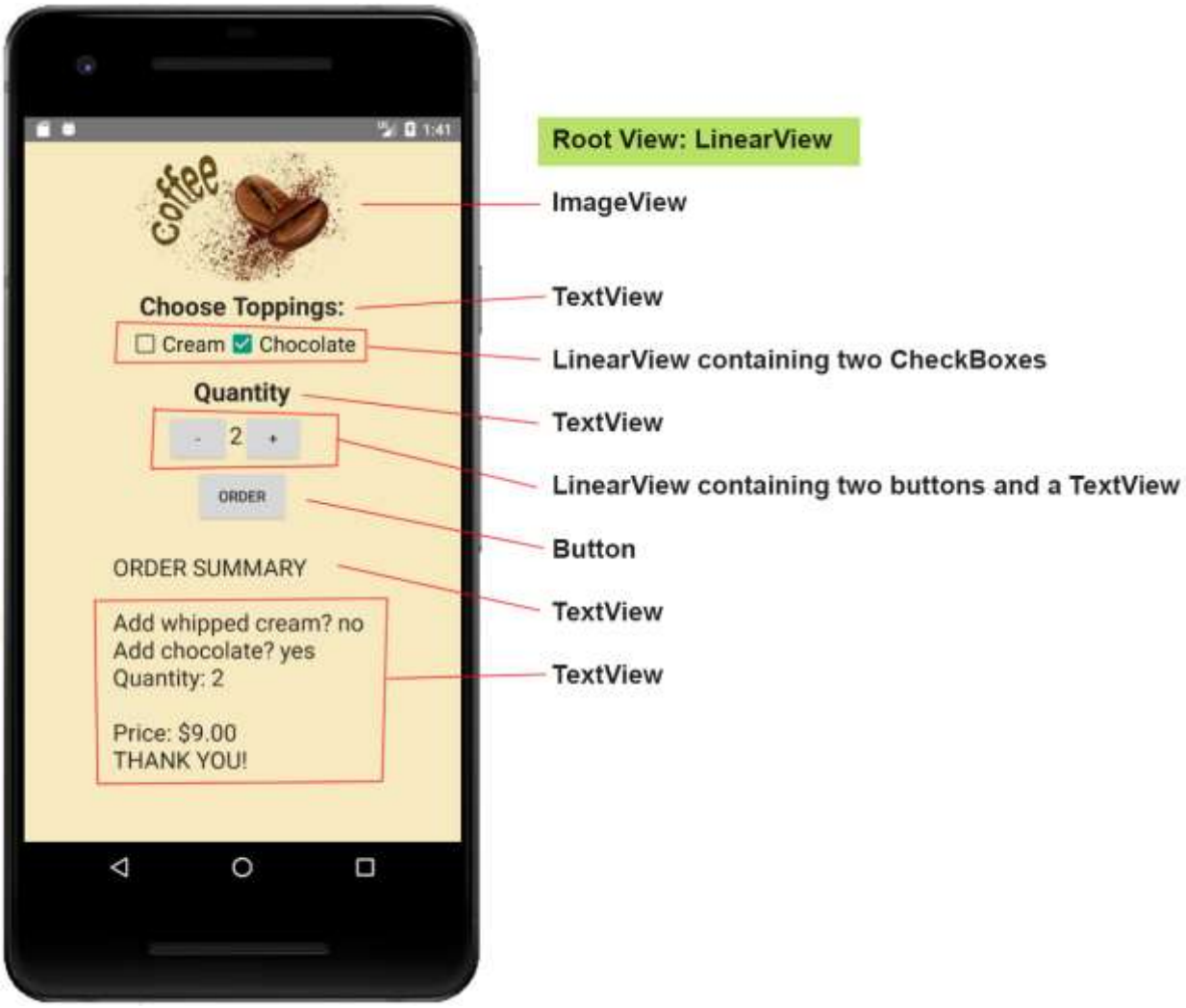
Android View	Value	Event
ImageView	Mood Image	*None*
TextView	“I’m So Hungry” or “I’m so Full”	*None*
Button	“Eat Cookie”	onEatingCookie()



MOBILE APPLICATION DEVELOPMENT (MAD) : LAB 3

Activity 3 : Build the Coffee Ordering app shown below.

- Assume a single coffee costs $4.00. Charge an additional $1.00 for chocolate and $.50 for whipped cream, per cup.
- Background Color : #f7eac1

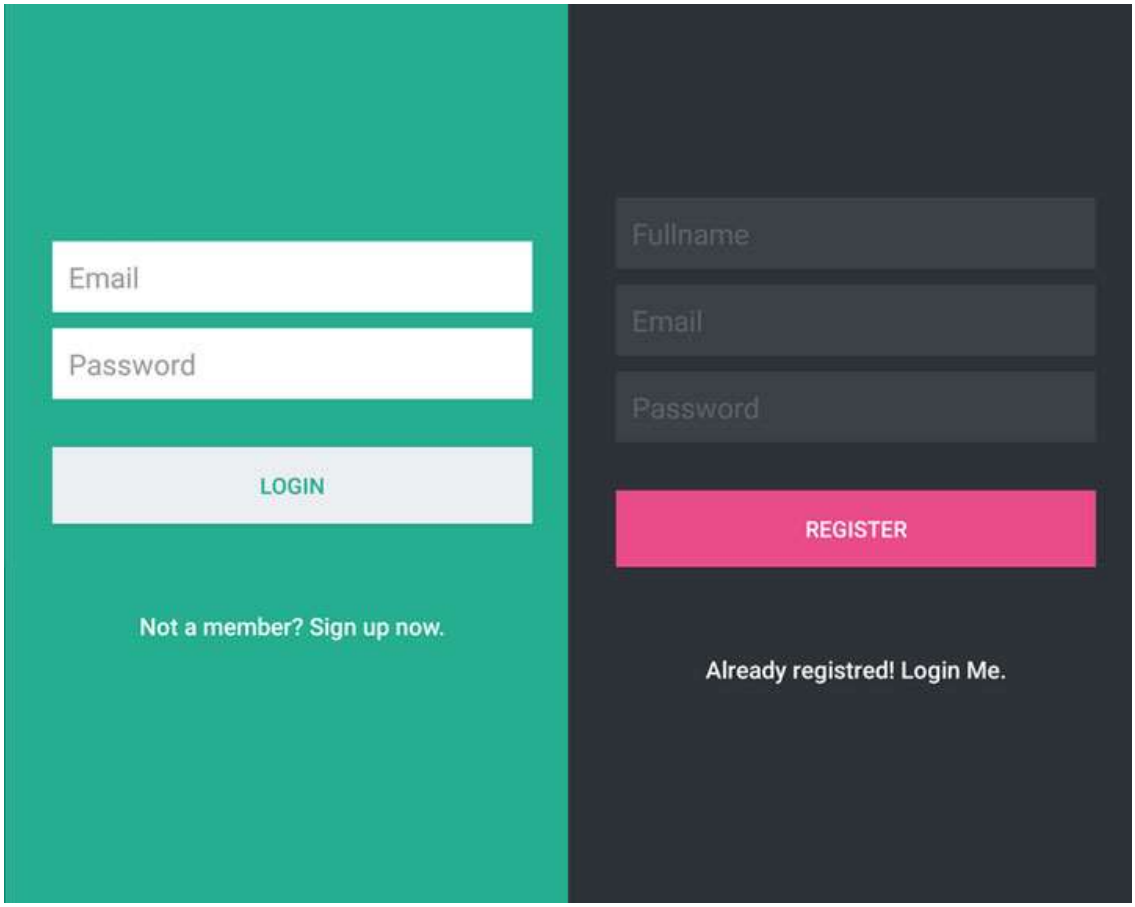


MOBILE APPLICATION DEVELOPMENT (MAD) : LAB 3

Activity 4: Create a Login Page – Steps

- Add following components

View	Text	Event
2-3 Edittexts	Placeholder values (Email and Password)	none
Button	Login / Register	ValidateUser()
TextView	Not a member? Sign up now. / Already register! Login Me.	ChangeScreen() – This method will move user from login to register and from register to login.



MOBILE APPLICATION DEVELOPMENT (MAD) : LAB 3

Activity 5: Hangman Game

Hangman

Make a basic **Hangman** game that displays an image of a gallows and a hanging man, along with a word that the player is trying to guess. The word is chosen randomly from a provided dictionary. At all times the game displays a "**clue**" of the letters the player has guessed correctly; for example, if the word is `"apples"` and the player has guessed `e`, `k`, `p`, and `t`, the clue would be `"?pp?e?"`. The user can type single-letter guesses into an `EditText`. (The `EditText` allows the user to type multi-letter strings and non-letters; a robust game would handle such attempts gracefully, as well as other errors like trying to guess the same letter twice, etc.) You can display a message such as a `Toast` when the user guesses the word correctly or runs out of guesses and ends the game.