Port City International University

Report name: *Implementing basic data structure with kotlin and xml*
Course Code: CSE 322
Course Title: Software Engineering sessional
Submission Date: 18th February,2024

Submitted To:

Subashis Roy
Lecturer,
Department: CSE
Port City International University

Submitted By:

Name: SHAKIL ASHRAF
ID No. **CSE 026 07329**
Batch: 26th-D
Program: B.Sc. in CSE

Signature

Introduction:

- **Purpose of the report:** This report is about learning about basic of android development with kotlin, xml, fundamental of android studio.
- **Motivation for learning Android development:** In this course the choice of project was independent I pick android development. Because of I find it interesting to learn it.And it is relevant to my course learning outcome.

Learning Kotlin:

- **Choosing Kotlin as my programming language for Android development**: Kotlin offers concise syntax, null safety, and seamless interoperability with Java, making it a compelling choice for Android development. Its support for coroutines simplifies asynchronous programming, while extension functions and immutable data structures promote cleaner and safer code. Backed by **strong tooling support from Google** and JetBrains, Kotlin's active community adoption and modern language features enhance developer productivity and code quality, cementing its position as a leading language for Android app development.
- **Learning process:** For study kotlin I start to learn with the documentation of kotlin and tutorial.

  In this processes I learn about-

  - Variables
  - Basic types
  - Collections
  - Control flow
  - Functions
  - Classes
  - Null safety

Introduction to Android Studio and XML:

Android Studio stands as the quintessential Integrated Development Environment (IDE) for Android application development, offering an extensive array of tools meticulously designed to streamline the entire development lifecycle. As the official IDE sanctioned by Google, Android Studio provides developers with an integrated suite of features, including code editing, debugging, testing, and performance profiling, all finely attuned to the nuances of Android development. By consolidating these tools within a single environment, Android Studio enables developers to efficiently create, iterate, and optimize their Android applications with ease.

XML (eXtensible Markup Language) holds a central role in Android development, particularly in the realm of designing user interfaces through layout specifications. Android Studio boasts robust XML editing capabilities, allowing developers to craft layouts either through a visual interface using the Layout Editor or by directly manipulating XML code. These XML files delineate the structure and appearance of various UI components within an app, encompassing views, widgets, and containers.

In the Android development landscape, XML files adhere to a hierarchical structure, with each file corresponding to a specific UI component or configuration aspect of the application. Through XML attributes, developers specify a myriad of properties and behaviors for UI elements, spanning layout dimensions, text content, styling, and event handling. This declarative approach to UI design empowers developers to precisely define the visual and interactive aspects of their applications, fostering intuitive and engaging user experiences.

Mastering XML within the confines of Android Studio is indispensable for developers seeking to create polished and user-friendly Android applications. With its robust toolset and seamless integration with XML-based layout design, Android Studio empowers developers to translate their creative visions into tangible, high-quality applications that resonate with users across the Android ecosystem.

Learning processes: For learning these topic I follow the official documentation of android Studio also fundamentals of android studio and XML.

Project: Implementing basic data structure with kotlin and xml

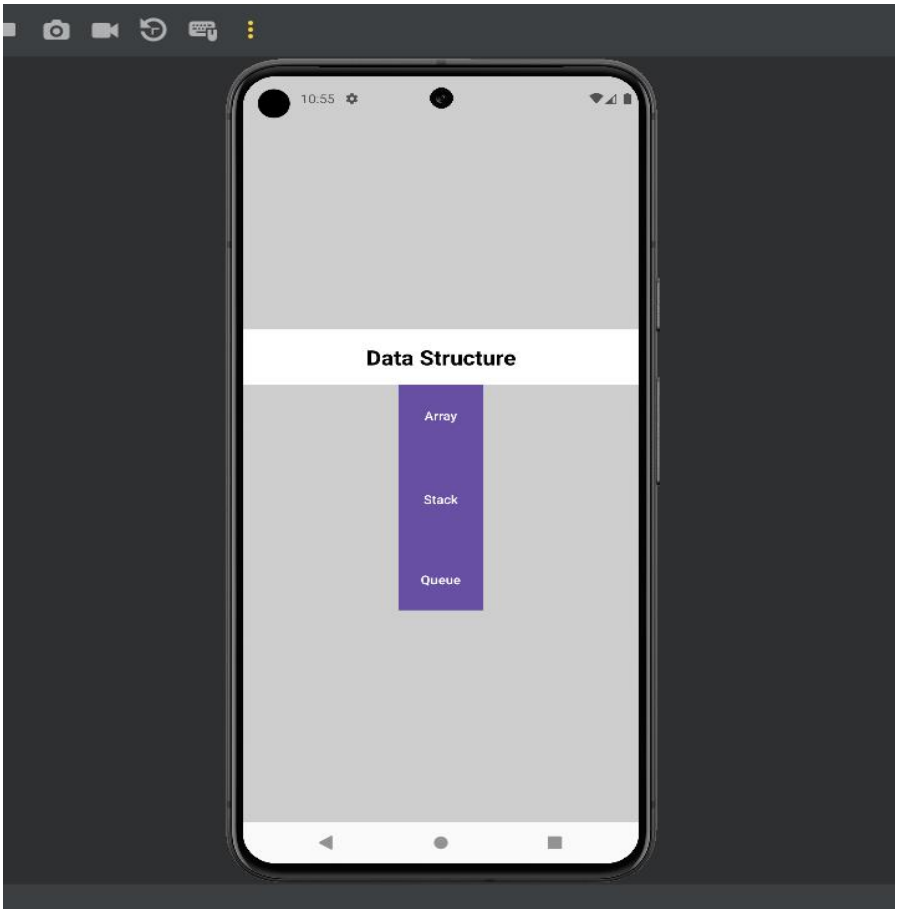
Introduction: this app will operate with the basic operation of data structure like array, stack ,queue(and further it will add as feature as the advance of data structure algorithm and operation.)

Instrument:

- Android studio
- Build in emulator (using Pixel 8 API 30) to operate app functionality

App implementation:

Home page



Kotlin implementation:

```
package com.example.datastructure

import android.content.Intent
import android.os.Bundle
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat
import android.widget.Button
import com.example.datastructure.array.array_activity
import com.example.datastructure.queue.queue_activity
//import
com.example.datastructure.array.com.example.datastructure.array.array_activit
y
//import
com.example.datastructure.queue.com.example.datastructure.queue.queue_activit
y
//import
com.example.datastructure.stack.com.example.datastructure.stack.stack_activit
y
import com.example.datastructure.stack.stack_activity

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)


ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.btnStack)) { v,
insets ->
            val systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top, systemBars.right,
systemBars.bottom)
            insets
        }

        val btnArray: Button = findViewById(R.id.btnArray)
        btnArray.setOnClickListener {
            startActivity(Intent(this, array_activity::class.java))
        }

        val btnStack: Button = findViewById(R.id.btnStack)
        btnStack.setOnClickListener {
            startActivity(Intent(this, stack_activity::class.java))
        }

        val btnQueue: Button = findViewById(R.id.btnQueue)
        btnQueue.setOnClickListener {
            startActivity(Intent(this, queue_activity::class.java))
        }
    }
}
```

xml implementation:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center"
    android:background="#CECECE">
    <!-- data_structure_headline.xml -->
    <TextView
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/textViewTitle"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Data Structure"
        android:textSize="24sp"
        android:textStyle="bold"
        android:gravity="center"
        android:padding="16dp"
        android:textColor="@android:color/black"
        android:background="@android:color/white"/>

    <Button
        android:id="@+id/btnArray"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Array"
        android:background="#403333"
        android:padding="22dp"/>

    <Button
        android:id="@+id/btnStack"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#403333"
        android:padding="22dp"
        android:text="Stack" />

    <Button
        android:id="@+id/btnQueue"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Queue"
        android:background="#403333"
        android:padding="22dp"/>
    <!-- Add similar buttons for Queue, LinkedList, etc. -->

</LinearLayout>
```

Conclusion: In this section there is implementation of the apps home page with kotlin and xml implementation.