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 April,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 to room database:

Room is a powerful persistence library in Android, part of the Android Jetpack. It provides an abstraction layer over SQLite, simplifying database operations. Key components include Entity, representing a table, DAO (Data Access Object) defining database operations, and Database, representing the database itself. Room offers advantages such as compile-time verification of SQL queries, integration with LiveData and RxJava for reactive programming, and improved performance due to SQLite's efficiency. To use Room, define entities as annotated classes, create DAO interfaces with annotated methods for operations, and implement a database class annotated with **@Database**. With its type safety and abstraction, Room facilitates robust and efficient database management in Android applications.

Resource: To learn room database tutorial

Introduction to jetpacK compose: Jetpack Compose is a modern UI toolkit for Android app development introduced by Google. It simplifies the process of building UIs with a declarative approach, allowing developers to describe the UI components and their behavior using Kotlin code. Compose offers a more intuitive way to create dynamic and interactive user interfaces compared to traditional XML-based layouts. It enables developers to write less boilerplate code, resulting in increased productivity and code maintainability. With features like state management, theming, and animations built directly into the framework, Jetpack Compose streamlines the UI development process. Additionally, Compose is fully interoperable with existing Android code, allowing developers to gradually adopt it into their projects. Overall, Jetpack Compose represents a significant shift in Android UI development paradigms, offering a more modern and efficient approach to building UIs for Android apps.

Resource: to Learn jetpack compose tutorial

To Learning MVVM .tutorial

Project :Implementing basic data structure with kotlin and xml

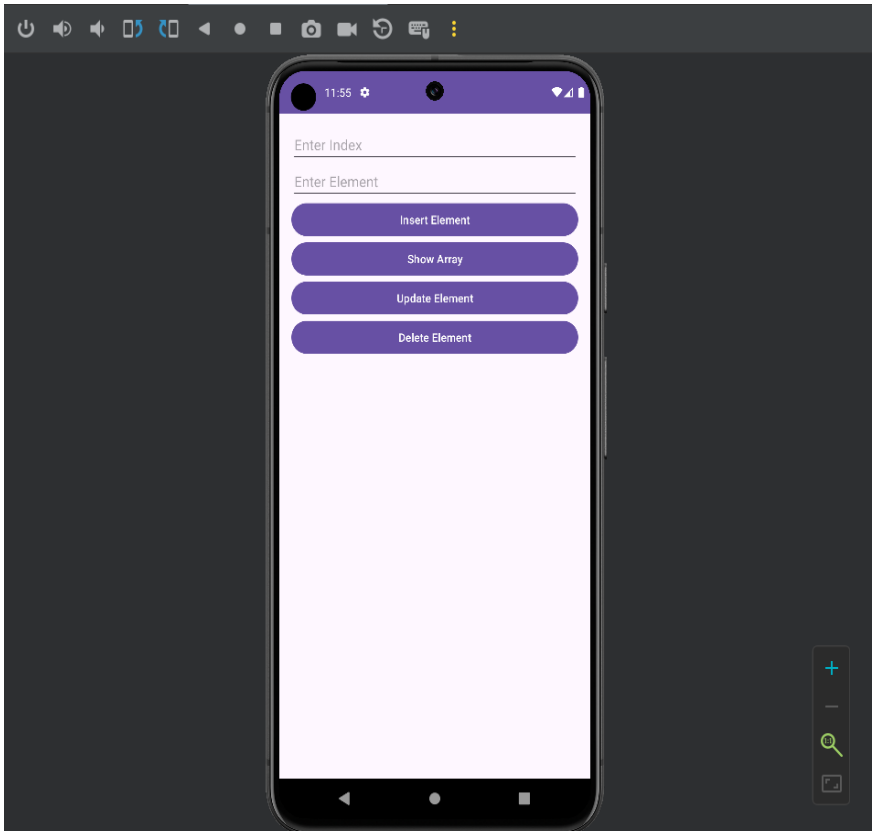
Introduction: in this section I try to do some additional function like-

- Add data structure operation
- Add multiple page functionality
- Add use database (for device and learning complexity its not completed)
- Make use able application**(for device and learning complexity its not completed)**

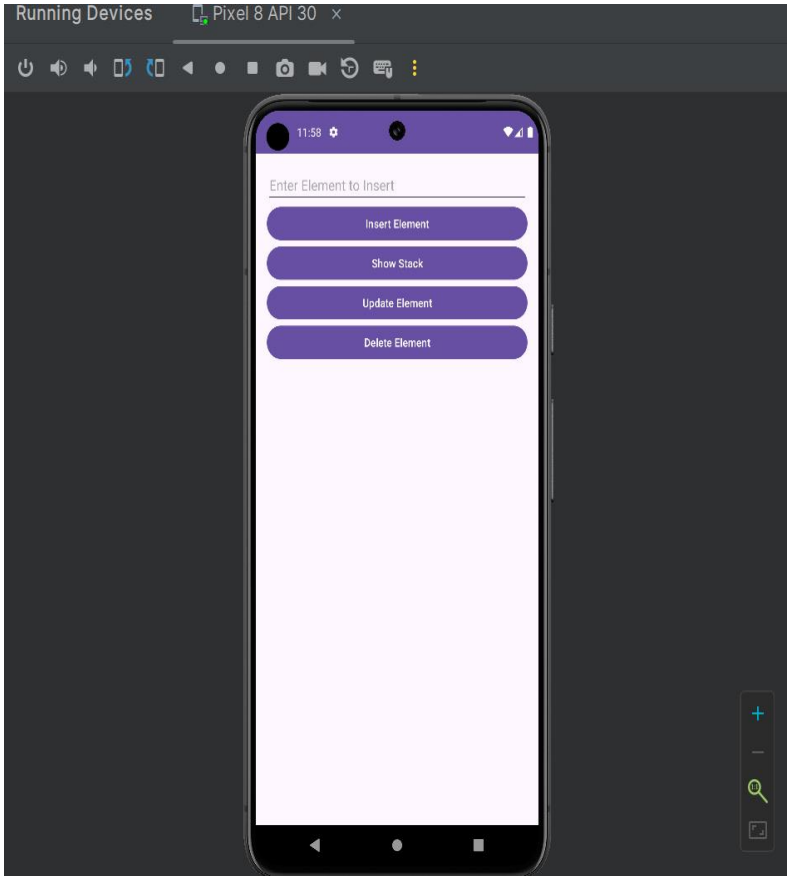
App implementation of rest functionality:

After click array ,stack, queue button –

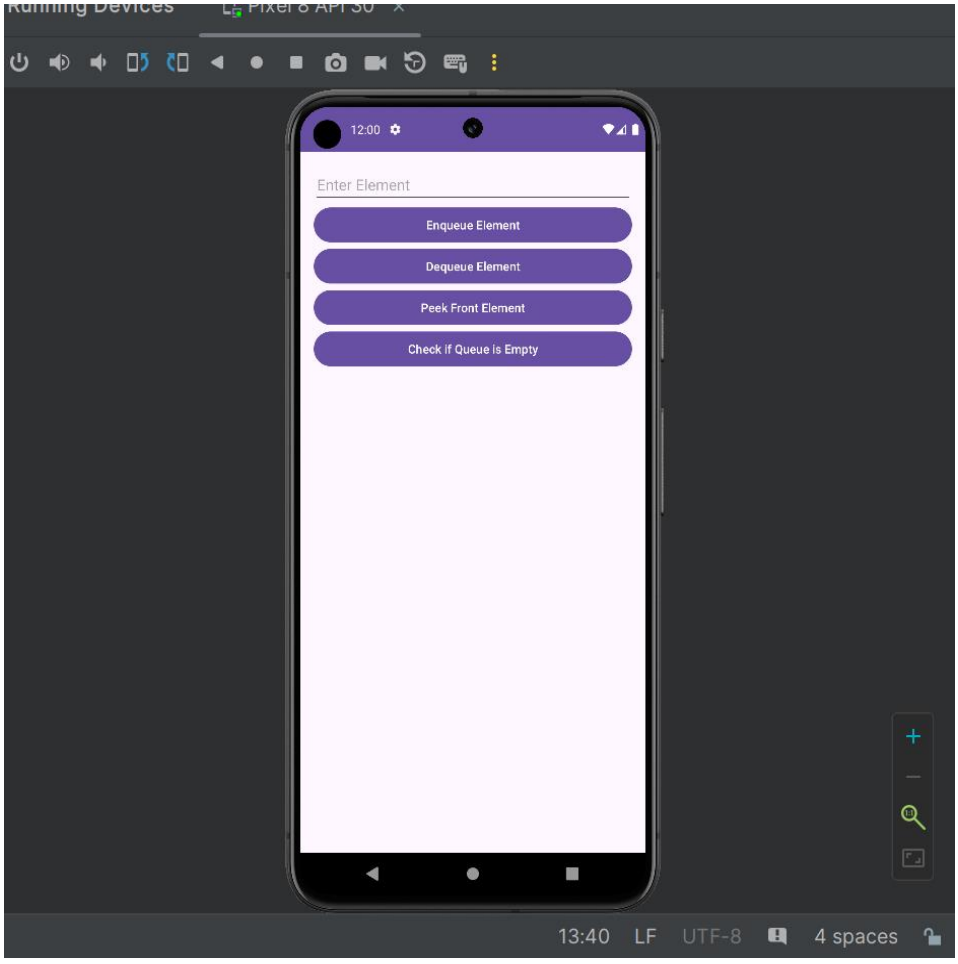
Array button-



Stack button-



Queue button:



Implementation in kotlin:

```
package com.example.datastructure.array

import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.example.datastructure.R
import java.util.*

class array_activity : AppCompatActivity() {

    private val array = ArrayList<String>()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_array)

        val editTextIndex = findViewById<EditText>(R.id.editTextIndex)
        val editTextElement = findViewById<EditText>(R.id.editTextElement)
        val btnInsert = findViewById<Button>(R.id.btnInsert)
        val btnShow = findViewById<Button>(R.id.btnShow)
        val btnUpdate = findViewById<Button>(R.id.btnUpdate)
        val btnDelete = findViewById<Button>(R.id.btnDelete)

        btnInsert.setOnClickListener {
            val index = editTextIndex.text.toString().toIntOrNull()
            val element = editTextElement.text.toString().trim()
            if (index != null && element.isNotEmpty()) {
                array.add(index, element)
                Toast.makeText(this, "Element inserted at index $index: $element",
Toast.LENGTH_SHORT).show()
                editTextIndex.text.clear()
                editTextElement.text.clear()
            } else {
                Toast.makeText(this, "Please enter valid index and element",
Toast.LENGTH_SHORT).show()
            }
        }

        btnShow.setOnClickListener {
            val arrayString = array.joinToString(separator = ", ", prefix = "[", postfix =
"]")
            Toast.makeText(this, "Array: $arrayString", Toast.LENGTH_LONG).show()
        }

        btnUpdate.setOnClickListener {
            val index = editTextIndex.text.toString().toIntOrNull()
```

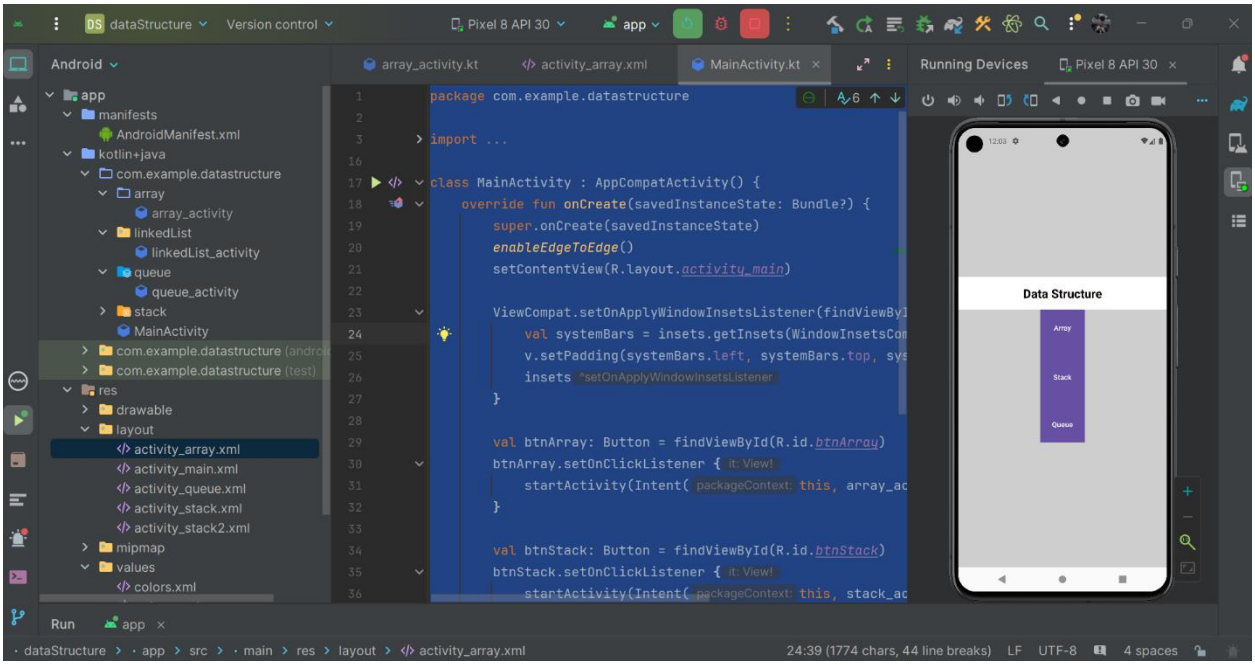
```
            val element = editTextElement.text.toString().trim()
            if (index != null && element.isNotEmpty() && index >= 0 && index <
array.size) {
                array[index] = element
                Toast.makeText(this, "Element at index $index updated to: $element",
Toast.LENGTH_SHORT).show()
                editTextIndex.text.clear()
                editTextElement.text.clear()
            } else {
                Toast.makeText(this, "Please enter valid index and element",
Toast.LENGTH_SHORT).show()
            }
        }

        btnDelete.setOnClickListener {
            val index = editTextIndex.text.toString().toIntOrNull()
            if (index != null && index >= 0 && index < array.size) {
                val deletedElement = array.removeAt(index)
                Toast.makeText(this, "Element at index $index deleted:
$deletedElement", Toast.LENGTH_SHORT).show()
                editTextIndex.text.clear()
            } else {
                Toast.makeText(this, "Please enter a valid index",
Toast.LENGTH_SHORT).show()
            }
        }
    }
}
```

Xml:

```
<!-- activity_array_operation.xml -->
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

    <!-- Input field for array operations -->
    <EditText
        android:id="@+id/editTextIndex"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter Index" />
```

```
    <EditText
        android:id="@+id/editTextElement"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter Element" />

    <!-- Button to insert an element into the array -->
    <Button
        android:id="@+id/btnInsert"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Insert Element" />

    <!-- Button to show the array -->
    <Button
        android:id="@+id/btnShow"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Show Array" />

    <!-- Button to update an element in the array -->
    <Button
        android:id="@+id/btnUpdate"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Update Element" />

    <!-- Button to delete an element from the array -->
    <Button
        android:id="@+id/btnDelete"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Delete Element" />

</LinearLayout>
```

Last view:


dataStructure
Version control
Pixel 8 API 30
app
Android
app
manifests
AndroidManifest.xml
kotlin+java
com.example.datastructure
array
array_activity
linkedList
linkedList_activity
queue
queue_activity
stack
MainActivity
com.example.datastructure (test)
res
drawable
layout
activity_array.xml
activity_main.xml
activity_queue.xml
activity_stack.xml
activity_stack2.xml
mipmap
values
colors.xml
array_activity.kt
activity_array.xml
MainActivity.kt
package com.example.datastructure
import ...
class MainActivity : AppCompatActivity() {
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
enableEdgeToEdge()
setContentView(R.layout.activity_main)
ViewCompat.setOnApplyWindowInsetsListener(findViewById
val systemBars = insets.getInsets(WindowInsetsCo
v.setPadding(systemBars.left, systemBars.top, sy
insets
}
val btnArray: Button = findViewById(R.id.btnArray)
btnArray.setOnClickListener {
startActivity(Intent(
}
val btnStack: Button = findViewById(R.id.btnStack)
btnStack.setOnClickListener {
startActivity(Intent(
Running Devices
Pixel 8 API 30
Data Structure
Array
Stack
Queue
Run
app
dataStructure
app
src
main
res
layout
activity_array.xml
24:39 (1774 chars, 44 line breaks)
LF
UTF-8
4 spaces

<u>Conclusion:</u> the building of this application we use kotlin ,xml as languge .for database room database, software Android studio.