USMAN INSTITUTE OF TECHNOLOGY

Department of Computer Science
CS 331 Mobile App Development

Lab # 15
Mock exam

Objective:

This experiment prepares the students for the final lab exam. Android Studio will be as a programming tool for Kotlin part and VS Code will be used as a programming tool for Flutter part.

Name of Student: Maheen Zafar Khan

Roll No: 20B-061-SE **Sec.** B

Date of Experiment: 6th July' 2023

Marks Obtained/Remarks: _____

Signature: _____

Lab 15: Mock exam

Submission Instructions

Create a folder with your name and roll number. Paste all code at the end of the document and also attach screenshots for both apps in this same document. Also copy both complete project files in the folder, archive and paste in shared folder.

Criminal Intent code snippets

Following code snippets can be used to solve question 1 of below exercise.

```
@Parcelize
data class Crime(
    val id: UUID,
    val title: String,
    val date: Date,
    val isSolved: Boolean
)  : Parcelable

private val args: CrimeDetailFragmentArgs by navArgs()
private val crimeDetailViewModel: CrimeDetailViewModel by viewModels {
    CrimeDetailViewModelFactory(args.crime)
}
private var _binding: FragmentCrimeDetailBinding? = null
private val binding
    get() = checkNotNull(_binding) {
        "Cannot access binding because it is null. Is the view visible?"
    }


_binding = FragmentCrimeDetailBinding.inflate(layoutInflater, container,
false)
return binding.root


binding.apply {
    crimeTitle.doOnTextChanged { text, _, _, _ ->
        crimeDetailViewModel.updateCrime { oldCrime ->
            oldCrime.copy(title = text.toString())
        }
    }
    crimeDate.apply {
        isEnabled = false
    }
    crimeSolved.setOnCheckedChangeListener { _, isChecked ->
        crimeDetailViewModel.updateCrime { oldCrime ->
            oldCrime.copy(isSolved = isChecked)
        }
    }
}
viewLifecycleOwner.lifecycleScope.launch {
    viewLifecycleOwner.lifecycle.repeatOnLifecycle(Lifecycle.State.STARTED)
{
        crimeDetailViewModel.crime.collect { crime ->
            crime?.let { updateUi(it) }
```

```
        }
    }
}

private fun updateUi(crime: Crime) {
    binding.apply {
        if (crimeTitle.text.toString() != crime.title) {
            crimeTitle.setText(crime.title)
        }
        crimeDate.text = crime.date.toString()
        crimeSolved.isChecked = crime.isSolved
    }
}

_binding = null


private val _crime: MutableStateFlow<Crime?> = MutableStateFlow(null)
val crime: StateFlow<Crime?> = _crime.asStateFlow()


viewModelScope.launch {
    _crime.value = crime
}


fun updateCrime(onUpdate: (Crime) -> Crime) {
    _crime.update { oldCrime ->
        oldCrime?.let { onUpdate(it) }
    }
}

override fun <T : ViewModel> create(modelClass: Class<T>): T {
    return CrimeDetailViewModel(crime) as T
}



binding.crimeTitle.text = crime.title
binding.crimeDate.text = crime.date.toString()
binding.root.setOnClickListener {
    onCrimeClicked(crime)
}

val inflater = LayoutInflater.from(parent.context)
val binding = ListItemCrimeBinding.inflate(inflater, parent, false)
return CrimeHolder(binding)

val crime = crimes[position]
holder.bind(crime, onCrimeClicked)


private var _binding: FragmentCrimeListBinding? = null
private val binding
    get() = checkNotNull(_binding) {
        "Cannot access binding because it is null. Is the view visible?"
    }
private val crimeListViewModel: CrimeListViewModel by viewModels()
```

```
_binding = FragmentCrimeListBinding.inflate(inflater, container, false)
binding.crimeRecyclerView.layoutManager = LinearLayoutManager(context)
return binding.root


viewLifecycleOwner.lifecycleScope.launch {
    viewLifecycleOwner.repeatOnLifecycle(Lifecycle.State.STARTED) {
        val crimes = crimeListViewModel.loadCrimes()
        binding.crimeRecyclerView.adapter = CrimeListAdapter(crimes){crime
->
            findNavController().navigate(
CrimeListFragmentDirections.showCrimeDetail(crime))
        }
    }
}

_binding = null


val crimes = mutableListOf<Crime>()
init {
    viewModelScope.launch {
        crimes += loadCrimes()
    }
}

suspend fun loadCrimes(): List<Crime> {
    val result = mutableListOf<Crime>()
    for (i in 0 until 100) {
        val crime = Crime(
            id = UUID.randomUUID(), title = "Crime #$i", date = Date(),
isSolved = i % 2 == 0
        )
        result += crime
    }
    return result
}

<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="16dp">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textAppearance="?attr/textAppearanceHeadline5"
        android:text="@string/crime_title_label" />
    <EditText
        android:id="@+id/crime_title"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/crime_title_hint"
        android:importantForAutofill="no"
        android:inputType="text" />
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textAppearance="?attr/textAppearanceHeadline5"
```

```
        android:text="@string/crime_details_label" />
    <Button
        android:id="@+id/crime_date"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        tools:text="Wed May 11 11:56 EST 2022" />
    <CheckBox
        android:id="@+id/crime_solved"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/crime_solved_label" />
</LinearLayout>
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="8dp">
    <TextView
        android:id="@+id/crime_title"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Crime Title"/>
    <TextView
        android:id="@+id/crime_date"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Crime Date"/>
</LinearLayout>
```

Gradle files

```
// Top-level build file where you can add configuration options common to
all sub-projects/modules.
plugins {
    id 'com.android.application' version '8.0.1' apply false
    id 'com.android.library' version '8.0.1' apply false
    id 'org.jetbrains.kotlin.android' version '1.8.20' apply false
    id 'androidx.navigation.safeargs.kotlin' version '2.5.1' apply false
    id 'org.jetbrains.kotlin.plugin.parcelize' version '1.6.10' apply false
}
```

```
plugins {
    id 'com.android.application'
    id 'org.jetbrains.kotlin.android'
    id 'org.jetbrains.kotlin.kapt'
    id 'androidx.navigation.safeargs'
    id 'org.jetbrains.kotlin.plugin.parcelize'
}

android {
    namespace 'com.example.finallab'
    compileSdk 33

    defaultConfig {
        applicationId "com.example.finallab"
        minSdk 24
        targetSdk 33
        versionCode 1
        versionName "1.0"
```

```
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-
optimize.txt'), 'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_17
        targetCompatibility JavaVersion.VERSION_17
    }
    kotlinOptions {
        jvmTarget = '17'
    }

    buildFeatures {
        viewBinding true
    }
    viewBinding.enabled = true

}

dependencies {
    implementation 'androidx.core:core-ktx:1.10.1'
    implementation 'androidx.appcompat:appcompat:1.6.1'
    implementation 'com.google.android.material:material:1.9.0'
    implementation 'androidx.fragment:fragment:1.6.0'
    implementation 'androidx.lifecycle:lifecycle-viewmodel-ktx:2.6.1'
    implementation 'androidx.fragment:fragment-ktx:1.6.0'
    implementation 'androidx.recyclerview:recyclerview:1.3.0'
    implementation "androidx.navigation:navigation-fragment-ktx:2.6.0"
    implementation "androidx.navigation:navigation-ui-ktx:2.6.0"
}
```

Recipe app code snippets

Create assets folder and place any images with the file names referred in code.

```
class Recipe {
  String label;
  String imageUrl;
  int servings;
  List<Ingredient> ingredients;
  Recipe(
    this.label,
    this.imageUrl,
    this.servings,
    this.ingredients,
  );

  static List<Recipe> samples = [
    Recipe(
      'Spaghetti and Meatballs',
```

```
      'assets/2126711929_ef763de2b3_w.jpg',
      4,
      [
        Ingredient(
          1,
          'box',
          'Spaghetti',
        ),
        Ingredient(
          4,
          '',
          'Frozen Meatballs',
        ),
        Ingredient(
          0.5,
          'jar',
          'sauce',
        ),
      ],
    ),
    Recipe(
      'Tomato Soup',
      'assets/27729023535_a57606c1be.jpg',
      2,
      [
        Ingredient(
          1,
          'can',
          'Tomato Soup',
        ),
      ],
    ),
    Recipe(
      'Grilled Cheese',
      'assets/3187380632_5056654a19_b.jpg',
      1,
      [
        Ingredient(
          2,
          'slices',
          'Cheese',
        ),
        Ingredient(
          2,
          'slices',
          'Bread',
        ),
      ],
```

```
  ),
  Recipe(
    'Chocolate Chip Cookies',
    'assets/15992102771_b92f4cc00a_b.jpg',
    24,
    [
      Ingredient(
        4,
        'cups',
        'flour',
      ),
      Ingredient(
        2,
        'cups',
        'sugar',
      ),
      Ingredient(
        0.5,
        'cups',
        'chocolate chips',
      ),
    ],
  ),
  Recipe(
    'Taco Salad',
    'assets/8533381643_a31a99e8a6_c.jpg',
    1,
    [
      Ingredient(
        4,
        'oz',
        'nachos',
      ),
      Ingredient(
        3,
        'oz',
        'taco meat',
      ),
      Ingredient(
        0.5,
        'cup',
        'cheese',
      ),
      Ingredient(
        0.25,
        'cup',
        'chopped tomatoes',
      ),
```

```
      ],
    ),
    Recipe(
      'Hawaiian Pizza',
      'assets/15452035777_294cefced5_c.jpg',
      4,
      [
        Ingredient(
          1,
          'item',
          'pizza',
        ),
        Ingredient(
          1,
          'cup',
          'pineapple',
        ),
        Ingredient(
          8,
          'oz',
          'ham',
        ),
      ],
    ),
  ];
}

class _MyHomePageState extends State<MyHomePage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(widget.title),
      ),
      body: SafeArea(
        child: ListView.builder(
          itemCount: ,
          itemBuilder: (BuildContext context, int index) {
            return GestureDetector(
              onTap: () {
                    },
                  ),
                );
              },
              child: buildRecipeCard(),
            );
          },
        ),
```

```
      ),
    );
  }

    return Card(
      elevation: 2.0,
      shape: RoundedRectangleBorder(borderRadius:
BorderRadius.circular(10.0)),
      child: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          children: <Widget>[
            Image(image: AssetImage()),
            const SizedBox(
              height: 14.0,
            ),
            Text(
             ,
              style: const TextStyle(
                fontSize: 20.0,
                fontWeight: FontWeight.w700,
                fontFamily: 'Palatino',
              ),
            )
          ],
        ),
      ),
    );
  }

  int _sliderVal = 1;
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(),
      ),
      body: SafeArea(
        child: Column(
          children: <Widget>[
            SizedBox(
              height: 300,
              width: double.infinity,
              child: Image(
                image: AssetImage(),
              ),
            ),
            const SizedBox(
```

```
            height: 4,
          ),
          Text(
           ,
            style: const TextStyle(fontSize: 18),
          ),
          Expanded(
            child: ListView.builder(
              padding: const EdgeInsets.all(7.0),
              itemCount:,
              itemBuilder: (BuildContext context, int index) {
              },
            ),
          ),
          Slider(
            min: 1,
            max: 10,
            divisions: 9,
            label:,
            value: _sliderVal.toDouble(),
            onChanged: (newValue) {
              setState(() {
                _sliderVal = newValue.round();
              });
            },
            activeColor: Colors.green,
            inactiveColor: Colors.black,
          ),
        ],
      ),
    ),
  );
 }
}

class Ingredient {
  double quantity;
  String measure;
  String name;
  Ingredient(
    this.quantity,
    this.measure,
    this.name,
  );
}
```

Mock exam

Question 1) Create criminal intent app in Kotlin

You need to build an application named CriminalIntent. CriminalIntent shows the list of crimes committed and on clicking a particular crime it shows crime details.

Put the above pieces of code together and complete the app. Make the check box clickable.

CODE:

MainActivity.kt:

```
package com.example.mocktest

import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity

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

        supportFragmentManager.beginTransaction()
            .replace(R.id.fragmentContainer, CrimeListFragment())
            .commit()
    }
}
```

Crime.kt:

```
package com.example.mocktest

import android.os.Parcelable
import kotlinx.parcelize.Parcelize
import java.util.*

@Parcelize
data class Crime(
    val id: UUID,
    val title: String,
    val date: Date,
    val isSolved: Boolean
) : Parcelable
```

CrimeDetailViewModel.kt:

```
package com.example.mocktest

import androidx.lifecycle.ViewModel
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.StateFlow
import kotlinx.coroutines.flow.asStateFlow
import kotlinx.coroutines.flow.update

class CrimeDetailViewModel : ViewModel() {
    private val _crime: MutableStateFlow<Crime?> = MutableStateFlow(null)
    val crime: StateFlow<Crime?> = _crime.asStateFlow()


    fun updateCrime(onUpdate: (Crime) -> Crime) {
        _crime.update { oldCrime ->
            oldCrime?.let { onUpdate(it) }
        }
    }
}
```

CrimeDetailViewModelFactory.kt

```
package com.example.mocktest

import androidx.lifecycle.ViewModel
import androidx.lifecycle.ViewModelProvider

class CrimeDetailViewModelFactory(private val crime: Crime) :
ViewModelProvider.Factory {
    override fun <T : ViewModel> create(modelClass: Class<T>): T {
        return CrimeDetailViewModel() as T
    }
}
```

CrimeDetailFragment.kt:

```
package com.example.mocktest

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.core.widget.doOnTextChanged
import androidx.fragment.app.Fragment
import androidx.fragment.app.viewModels
import androidx.lifecycle.Lifecycle
import androidx.lifecycle.lifecycleScope
import androidx.lifecycle.repeatOnLifecycle
import androidx.navigation.fragment.navArgs
import com.example.mocktest.databinding.FragmentCrimeDetailBinding
import kotlinx.coroutines.launch

class CrimeDetailFragment : Fragment() {
    private val args: CrimeDetailFragmentArgs by navArgs()
    private val crimeDetailViewModel: CrimeDetailViewModel by viewModels {
        CrimeDetailViewModelFactory(args.crime)
    }
```

```
    private var _binding: FragmentCrimeDetailBinding? = null
    private val binding get() = _binding!!

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        _binding = FragmentCrimeDetailBinding.inflate(inflater, container,
false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        binding.apply {
            crimeTitle.doOnTextChanged { text, _, _, _ ->
                crimeDetailViewModel.updateCrime { oldCrime ->
                    oldCrime.copy(title = text.toString())
                }
            }
            crimeDate.isEnabled = false
            crimeSolved.setOnCheckedChangeListener { _, isChecked ->
                crimeDetailViewModel.updateCrime { oldCrime ->
                    oldCrime.copy(isSolved = isChecked)
                }
            }
        }
        viewLifecycleOwner.lifecycleScope.launch {

viewLifecycleOwner.lifecycle.repeatOnLifecycle(Lifecycle.State.STARTED) {
                crimeDetailViewModel.crime.collect { crime ->
                    crime?.let { updateUi(it) }
                }
            }
        }
    }

    private fun updateUi(crime: Crime) {
        binding.apply {
            if (crimeTitle.text.toString() != crime.title) {
                crimeTitle.setText(crime.title)
            }
            crimeDate.text = crime.date.toString()
            crimeSolved.isChecked = crime.isSolved
        }
    }

    override fun onDestroyView() {
        super.onDestroyView()
        _binding = null
    }
}
```

list_item_crime.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="8dp">
    <TextView
        android:id="@+id/crimeTitle"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Crime Title"/>
    <TextView
        android:id="@+id/crimeDate"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Crime Date"/>
</LinearLayout>
```

fragment_crime_list.xml:

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="16dp">
    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/crimeRecyclerView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:listitem="@layout/list_item_crime" />
</LinearLayout>
```

fragment_crime_detail.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="16dp">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="91dp"
        android:text="@string/crime_title_label"
        android:textAppearance="?attr/textAppearanceHeadline5" />
```

```
    <EditText
        android:id="@+id/crimeTitle"
        android:layout_width="match_parent"
        android:layout_height="119dp"
        android:hint="@string/crime_title_hint"
        android:importantForAutofill="no"
        android:inputType="text" />
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textAppearance="?attr/textAppearanceHeadline5"
        android:text="@string/crime_details_label" />
    <Button
        android:id="@+id/crimeDate"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        tools:text="Wed May 11 11:56 EST 2022" />

    <CheckBox
        android:id="@+id/crimeSolved"
        android:layout_width="match_parent"
        android:layout_height="83dp"
        android:text="@string/crime_solved_label" />
</LinearLayout>
```

Question 2) Create recipe app in Flutter

You need to build an application named recipe. Recipe app contains recipe to few dishes. On main screen it shows list of recipes and on clicking a particular recipe it shows recipe details.

CODE:

```
import 'package:flutter/material.dart';

class Recipe {
  String label;
  String imageUrl;
  int servings;
  List<Ingredient> ingredients;
  String details;

  Recipe(
    this.label,
    this.imageUrl,
    this.servings,
    this.ingredients,
    this.details,
  );
```

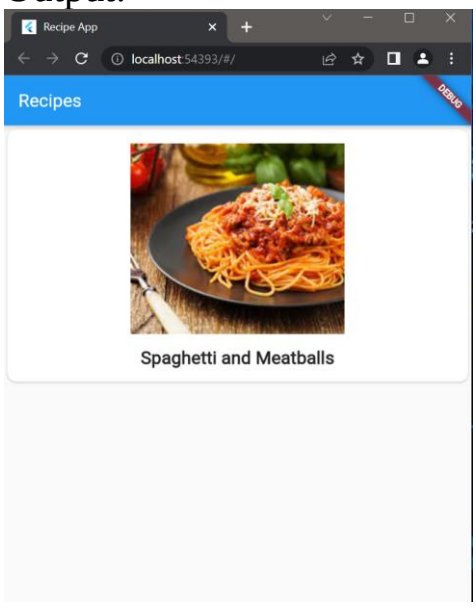
```
  static List<Recipe> samples = [
    Recipe(
      'Spaghetti and Meatballs',
      'assets/sp.jpg',
      4,
      [
        Ingredient(1, 'box', 'Spaghetti'),
        Ingredient(4, '', 'Frozen Meatballs'),
        Ingredient(0.5, 'jar', 'Sauce'),
      ],
      'Delicious spaghetti with meatballs in a savory sauce.',
    )

  ];
}

class Ingredient {
  double quantity;
  String measure;
  String name;

  Ingredient(this.quantity, this.measure, this.name);
}

class MyHomePage extends StatefulWidget {
  final String title;

  MyHomePage({Key? key, required this.title}) : super(key: key);

  @override
  _MyHomePageState createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  int _sliderVal = 1;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(widget.title),
      ),
      body: SafeArea(
        child: ListView.builder(
          itemCount: Recipe.samples.length,
          itemBuilder: (BuildContext context, int index) {
            return GestureDetector(
```

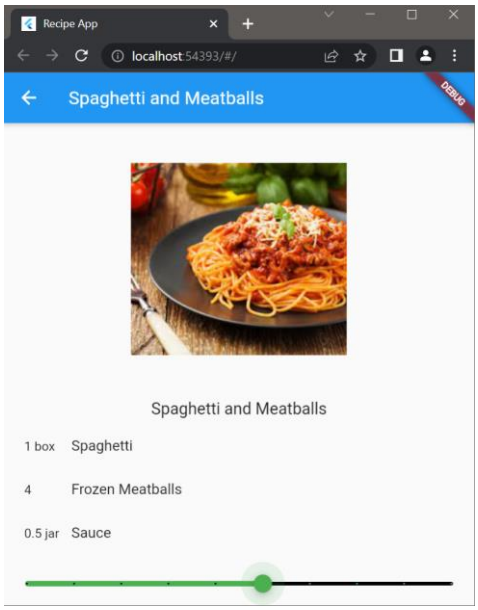
```
              onTap: () {
                Navigator.push(
                  context,
                  MaterialPageRoute(
                    builder: (context) =>
                        RecipeDetailsPage(recipe: Recipe.samples[index]),
                  ),
                );
              },
              child: buildRecipeCard(Recipe.samples[index]),
            );
          },
        ),
      ),
    );
  }

  Widget buildRecipeCard(Recipe recipe) {
    return Card(
      elevation: 2.0,
      shape: RoundedRectangleBorder(borderRadius:
BorderRadius.circular(10.0)),
      child: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          children: <Widget>[
            Image(image: AssetImage(recipe.imageUrl)),
            const SizedBox(height: 14.0),
            Text(
              recipe.label,
              style: const TextStyle(
                fontSize: 20.0,
                fontWeight: FontWeight.w700,
                fontFamily: 'Palatino',
              ),
            ),
          ],
        ),
      ),
    );
  }
}

class RecipeDetailsPage extends StatefulWidget {
  final Recipe recipe;

  RecipeDetailsPage({required this.recipe});
```

```
  @override
  _RecipeDetailsPageState createState() => _RecipeDetailsPageState();
}

class _RecipeDetailsPageState extends State<RecipeDetailsPage> {
  int _sliderVal = 1;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(widget.recipe.label),
      ),
      body: SafeArea(
        child: Column(
          children: <Widget>[
            SizedBox(
              height: 300,
              width: double.infinity,
              child: Image(image: AssetImage(widget.recipe.imageUrl)),
            ),
            const SizedBox(height: 4),
            Text(
              widget.recipe.label,
              style: const TextStyle(fontSize: 18),
            ),
            Expanded(
              child: ListView.builder(
                padding: const EdgeInsets.all(7.0),
                itemCount: widget.recipe.ingredients.length,
                itemBuilder: (BuildContext context, int index) {
                  final ingredient = widget.recipe.ingredients[index];
                  return ListTile(
                    leading:
                        Text('${ingredient.quantity} ${ingredient.measure}'),
                    title: Text(ingredient.name),
                  );
                },
              ),
            ),
            Slider(
              min: 1,
              max: 10,
              divisions: 9,
              label: 'Servings: ${_sliderVal * widget.recipe.servings}',
              value: _sliderVal.toDouble(),
              onChanged: (newValue) {
                setState(() {
```

```
                  _sliderVal = newValue.round();
                });
              },
              activeColor: Colors.green,
              inactiveColor: Colors.black,
            ),
          ],
        ),
      ),
    );
  }
}

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Recipe App',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: MyHomePage(title: 'Recipes'),
    );
  }
}
```

Output:




Recipe App
localhost:54393/#/
Spaghetti and Meatballs
DEBUG
Spaghetti and Meatballs
1 box Spaghetti
4 Frozen Meatballs
0.5 jar Sauce

Viva

Question 1) Determine the value of x

The activity code below changes the value of a property named x at various stages of the activity lifecycle. See if you can determine which lifecycle methods get called and the final value of x after each of the actions below.

The user opens the app and it starts running. The user then rotates the device.

```
class MainActivity : AppCompatActivity() {
    var x = 0
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        x = savedInstanceState?.getInt("x") ?: 2
    }
    override fun onRestart() {
        super.onRestart()
        x -= 7
    }
    override fun onStart() {
        super.onStart()
        x += 6
    }
    override fun onStop() {
        super.onStop()
        x *= 2
    }
    override fun onSaveInstanceState(savedInstanceState:
                                     Bundle) {
        savedInstanceState.putInt("x", x + 1)
        super.onSaveInstanceState(savedInstanceState)
    }
}
```

Answer:

1. User opens the app and it starts running:
   - `onCreate()` method is called.
   - The value of `x` is set to `2` if there is no saved instance state, or to the value of `savedInstanceState.getInt("x") + 1` if there is a saved instance state.
   - Final value of `x` after `onCreate()` is `2`.
2. User rotates the device:
   - `onStop()` method is called.
   - The value of `x` is multiplied by `2`.
   - Final value of `x` after `onStop()` is `4`.
   - The activity is destroyed and recreated.
   - `onCreate()` method is called again.

- The value of `x` is set to `2` if there is no saved instance state, or to the value of `savedInstanceState.getInt("x") + 1` if there is a saved instance state.
- Final value of `x` after the second `onCreate()` is `2`.

3. The app is restarted (e.g., brought to the foreground after being in the background):
- `onRestart()` method is called.
- The value of `x` is decreased by `7`.
- Final value of `x` after `onRestart()` is `-5`.
- `onStart()` method is called.
- The value of `x` is increased by `6`.
- Final value of `x` after `onStart()` is `1`.

To summarize:
- After the `onCreate()` method, the value of `x` is `2`.
- After the `onStop()` method (due to device rotation), the value of `x` is `4`.
- After the second `onCreate()` method (after device rotation), the value of `x` is `2`.
- After the `onRestart()` method, the value of `x` is `-5`.
- After the `onStart()` method, the value of `x` is `1`.

Question 2) what value of x will be displayed in Toast messages

```
class MainActivity : AppCompatActivity() {
    var x = 0

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        x = savedInstanceState?.getInt("x") ?: 2
        Toast.makeText(applicationContext, "value of x " + x,
Toast.LENGTH_SHORT).show()
    }

    override fun onRestart() {
        super.onRestart()
        x -= 7
        Toast.makeText(applicationContext, "value of x " + x,
Toast.LENGTH_SHORT).show()
    }
    override fun onStart() {
        super.onStart()
        lifecycleScope.launch {
            delay(1000)
            Toast.makeText(applicationContext, "inside scope value of x " +
x, Toast.LENGTH_SHORT).show()
        }
        x += 6
    }

    override fun onStop() {
        super.onStop()
```

```
        x *= 2
        Toast.makeText(applicationContext, "value of x " + x,
Toast.LENGTH_SHORT).show()
    }
    override fun onSaveInstanceState(savedInstanceState:
                                     Bundle) {
        savedInstanceState.putInt("x", x + 1)
        super.onSaveInstanceState(savedInstanceState)
        Toast.makeText(applicationContext, "value of x " + x,
Toast.LENGTH_SHORT).show()
    }
}
```

Answer:

1. User opens the app and it starts running:
  - `Toast` message: "value of x 2"
2. User rotates the device:
  - `Toast` message (from `onStop()`): "value of x 4"
  - `Toast` message (from `onCreate()`): "value of x 2"
3. The app is restarted:
  - `Toast` message (from `onRestart()`): "value of x -5"
  - `Toast` message (from `onStart()`): "inside scope value of x -5"
    (The `lifecycleScope.launch` block is delayed by 1000 milliseconds, so it executes after the `Toast` message.)
  - `Toast` message (from `onStop()`): "value of x -10"
  - `Toast` message (from `onSaveInstanceState()`): "value of x -10"

To summarize:
- After the `onCreate()` method, the `Toast` message displays "value of x 2".
- After the `onStop()` method (due to device rotation), the `Toast` message displays "value of x 4".
- After the second `onCreate()` method (after device rotation), the `Toast` message displays "value of x 2".
- After the `onRestart()` method, the `Toast` message displays "value of x -5".
- After the `onStart()` method, the `Toast` message displays "inside scope value of x -5".
- After the `onStop()` method, the `Toast` message displays "value of x -10".
- After the `onSaveInstanceState()` method, the `Toast` message displays "value of x -10".

Question 3) What will be the value of the counter in below program

```
import 'package:flutter/material.dart';
```

```
void main() {  runApp(const MyApp());}

class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(      title: 'Flutter Demo',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,      ),
      home: const MyHomePage(title: 'Flutter Demo Home Page'),    );  } }

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key, required this.title});
  final String title;
  @override  State<MyHomePage> createState() => _MyHomePageState();}

class _MyHomePageState extends State<MyHomePage> {
  int _counter = 0;
  void _incrementCounter() {
    setState(() {      _counter = 0;    });
    setState(() {      _counter++;    });  }

  @override  Widget build(BuildContext context) {
    return Scaffold(      appBar: AppBar(
        backgroundColor: Theme.of(context).colorScheme.inversePrimary,
        title: Text(widget.title),      ),
      body: Center(        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            const Text(
              'You have pushed the button this many times:',           ),
            Text(              '$_counter',
              style: Theme.of(context).textTheme.headlineMedium,           ),
          ],        ),      ),
      floatingActionButton: FloatingActionButton(
        onPressed: _incrementCounter,        tooltip: 'Increment',
        child: const Icon(Icons.add),
      ),    );  } }
```

Answer:

The value of the counter will always be 1 when the button is pressed. This is because the counter is set to 0 and then immediately incremented by 1 in the _incrementCounter() method. As a result, the final value displayed will be 1.