**Mobile Apps 2**

**Assignment Five**

**Submission Date 17/11/2024**

Percent 10%

**Include all workings and references**

***Section A.***

1. **Compare and contrast StateFlow and LiveData (*3 marks*)**

**Key similarities:**

* Both are observable data holder classes used in Android development
* Both support lifecycle-aware observation
* Both can be used to implement reactive patterns

**Key differences:**

* StateFlow is part of Kotlin Coroutines, while LiveData is part of Android Architecture Components
* StateFlow requires an initial value, LiveData can be null
* StateFlow is hot stream (always active), LiveData is lifecycle-aware and only active when observed
* StateFlow can be used in non-Android code, LiveData is Android-specific
* StateFlow has built-in support for coroutines, while LiveData requires additional transformations

1. **Referential Integrity is important to a relational database. Explain, giving an example and how it is enforced. (*2 marks*)**

Referential integrity is a database concept that ensures relationships between tables remain consistent. It means that any foreign key value must refer to an existing primary key in the parent table.

**Example:**

Consider two tables: Orders and Customers

* Customers table has CustomerID as primary key
* Orders table has a CustomerID as foreign key referencing Customers table

Referential integrity is enforced through:

* **Foreign key constraints that prevent:** 
  + Adding orders for non-existent customers
  + Deleting customers who have existing orders (unless CASCADE DELETE is specified)
  + Updating customer IDs that are referenced in orders

This maintains data consistency and prevents orphaned records.

***Section B.***

1. Using Room database, set up the following table from the Query tab:

import androidx.room.Entity  
import androidx.room.PrimaryKey  
  
@Entity(tableName = "inventory")  
data class Inventory(  
 @PrimaryKey(autoGenerate = true)  
 val id: Int? = null, // Auto-generated primary key  
 val name: String, // Required field  
 val quantity: Int, // Numeric field  
 val supplier: String = "Co-op Store", // Default value for supplier  
 val costPerUnit: Double = 0.0 // Values in decimal with default  
)

2. Write a script that would get the total monetary worth of inventory

// Calculate total monetary worth  
@Query("SELECT SUM(quantity \* costPerUnit) FROM inventory")  
suspend fun getTotalMonetaryWorth(): Double?