<u>BCS 371 Lab – Room Database Using Flow</u>

Overview

In this lab you will use the Room Persistence Library to create and use an SQLite database.

Create a project

Create a new Android application in Android Studio. Choose the **Empty Activity** type to create an empty activity that uses Jetpack Compose.

Gradle Dependencies

Add the necessary Gradle dependencies for using the Room Database and view models.

Create an Entity Class for the Database

Create a new **Entity** class named **Person**. Here are the specifications:

- Should be defined as an Entity class (class header must be decorated with the Entity annotation).
- Member variables: id(int), name (String), address(String)
- The id member variable should be defined as a primary key.

Create a Data Access Object Interface

Create a new interface named PersonDao. Here are the specifications:

- Create a new interface named **PersonDao** (must be decorated with @Dao attribute).
- There should be functions for the following:
    a. Get all persons from the database.
    b. Insert a person into the database
    c. Delete all persons from the database.

Create an Abstract PersonDatabase Class

Create a new abstract class named PersonDatabase. Here are the specifications:

- Create a new abstract class named **PersonDatabase**.
- Class header must be decorated with the Database annotation.
- The Person class should be the one and only entity class for this database.
- The **version** for the database should be 1.
- Add an **abstract method** named **personDao** that takes no parameters and has PersonDao as the return type.

Create MainScreenViewModel

Create a MainScreenViewModel class. It should inherit from AndroidViewModel. Here is the class header:

class MainScreenViewModel(var applicationContext: Application) : ViewModel()

Here are the specifications:

- Declare a member variable for the room database. The type should be PersonDatabase.
- Declare a member variable for a list of person. Here is the declaration:
  var personList by mutableStateOf(listOf<Person>())
- init block
  - Initialize the person database.
- fillDBWithData function– This function should clear the database then fill it with data. Here is sample data to add:

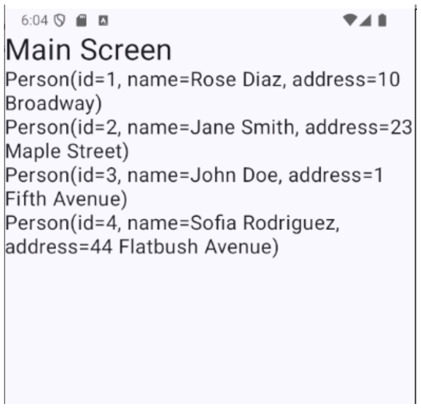
Id	Name	Address
1	Rose Diaz	10 Broadway
2	Jane Smith	23 Maple Street
3	John Doe	1 Fifth Avenue
4	Sofia Rodriguez	44 Flatbush Avenue

- Add a call to fillDBWithData to the init block after the database is created.
- Add code to collect the data into the personList member variable.

Setup the Main Screen

- Create a Kotlin file named MainScreen.kt. Add a composable function named MainScreen. Here is the function header:
  @Composable
  fun MainScreen(modifier: Modifier)

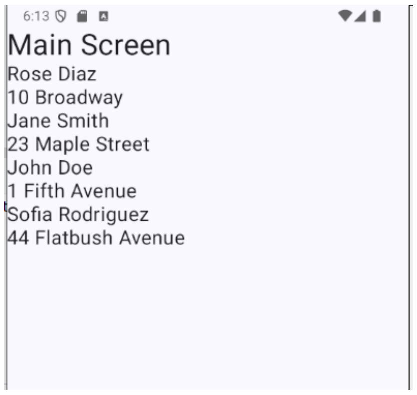
It should display all data from the database in Text fields. Here is a screenshot:



```
6:04
Main Screen
Person(id=1, name=Rose Diaz, address=10
Broadway)
Person(id=2, name=Jane Smith, address=23
Maple Street)
Person(id=3, name=John Doe, address=1
Fifth Avenue)
Person(id=4, name=Sofia Rodriguez,
address=44 Flatbush Avenue)
```

Show only the name and address in the UI

Show each person's name and address on separate lines in the UI. Here is a screenshot:

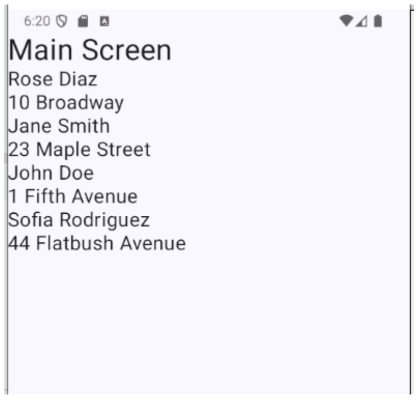


```
6:13
Main Screen
Rose Diaz
10 Broadway
Jane Smith
23 Maple Street
John Doe
1 Fifth Avenue
Sofia Rodriguez
44 Flatbush Avenue
```

Display data in a LazyColumn

Add a LazyColumn to the MainScreen composable. It should display each person's name from the database.

Here is a screenshot of what it should look like (should look the same as when using the column):



```
6:20
Main Screen
Rose Diaz
10 Broadway
Jane Smith
23 Maple Street
John Doe
1 Fifth Avenue
Sofia Rodriguez
44 Flatbush Avenue
```

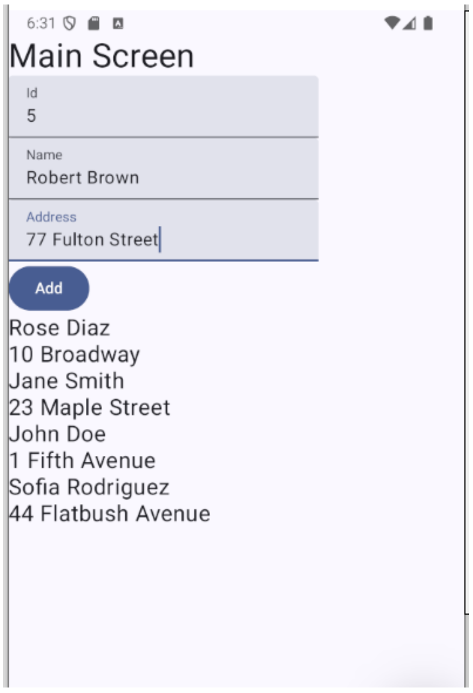
Update the app to add insert capability

Update the layout so the user can enter an id, name, and address. When the Add button is pressed it should add a new person to the database. Make sure to refresh the data that is displayed in the LazyColumn.

Make the following updates:

- Update the MainScreenViewModel.
    a. Add an **insert** function that takes an id, name, and address as parameters. It should create a Person instance with the data from the parameters. It should then call insert on the PersonDao and pass in the Person instance. Hint: You will need to use a coroutine inside this function.
- Update MainScreen. There should now be an Add button. The onClick event handler for Add should create a new Person instance that contains data from the TextFields. It should then call insert on MainScreenViewModel and pass in the Person instance.

After pressing the Add button, the new person's data will be inserted into the database. The UI should update automatically if the flows are properly set up. Here is a screenshot of what it should look like before and after pressing the Add button (the TextFields are using labels):


6:31

Main Screen

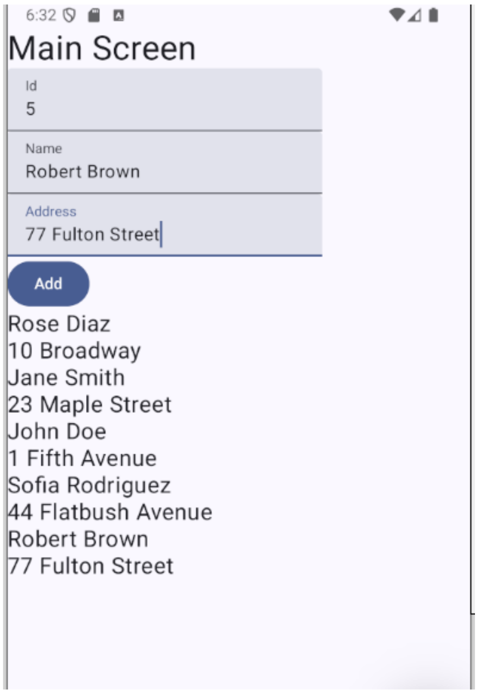
Id
5

Name
Robert Brown

Address
77 Fulton Street

Add

Rose Diaz
10 Broadway
Jane Smith
23 Maple Street
John Doe
1 Fifth Avenue
Sofia Rodriguez
44 Flatbush Avenue


6:32

Main Screen

Id
5

Name
Robert Brown

Address
77 Fulton Street

Add

Rose Diaz
10 Broadway
Jane Smith
23 Maple Street
John Doe
1 Fifth Avenue
Sofia Rodriguez
44 Flatbush Avenue
Robert Brown
77 Fulton Street