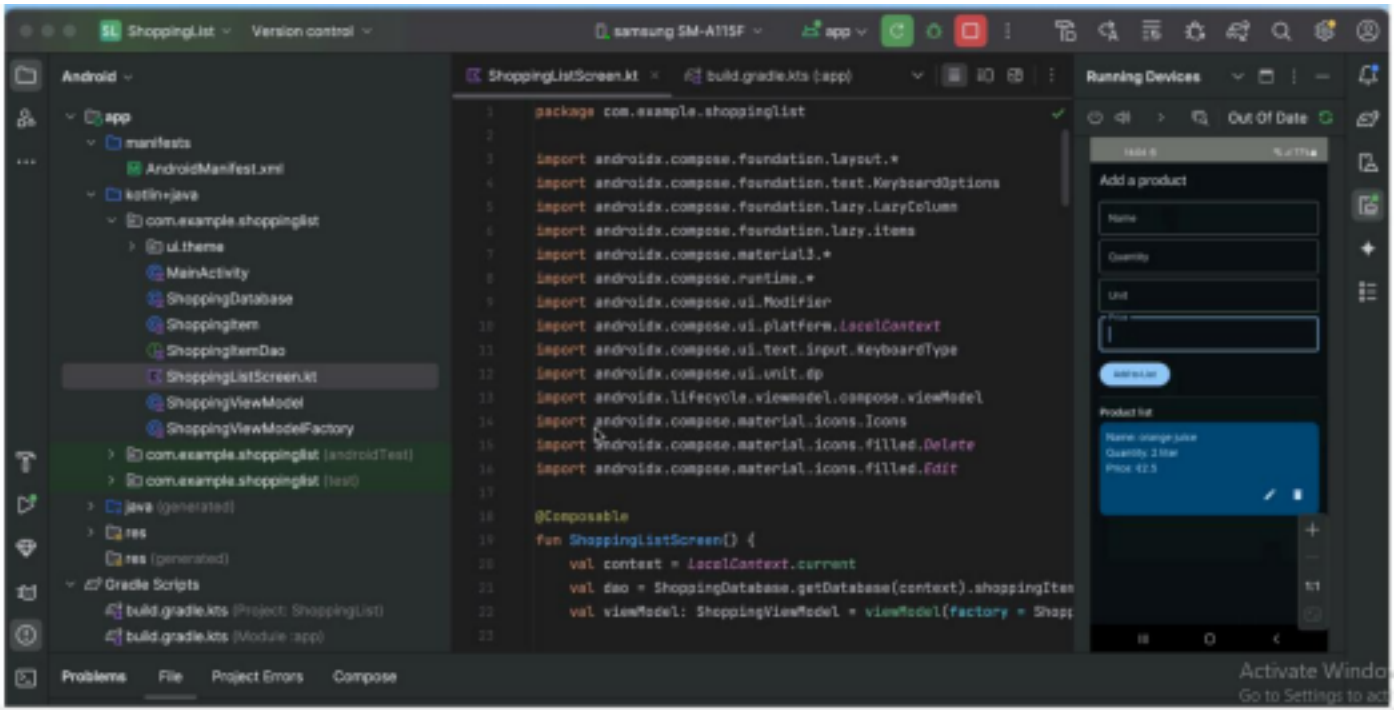
SAVONIA

Name: **Daba**

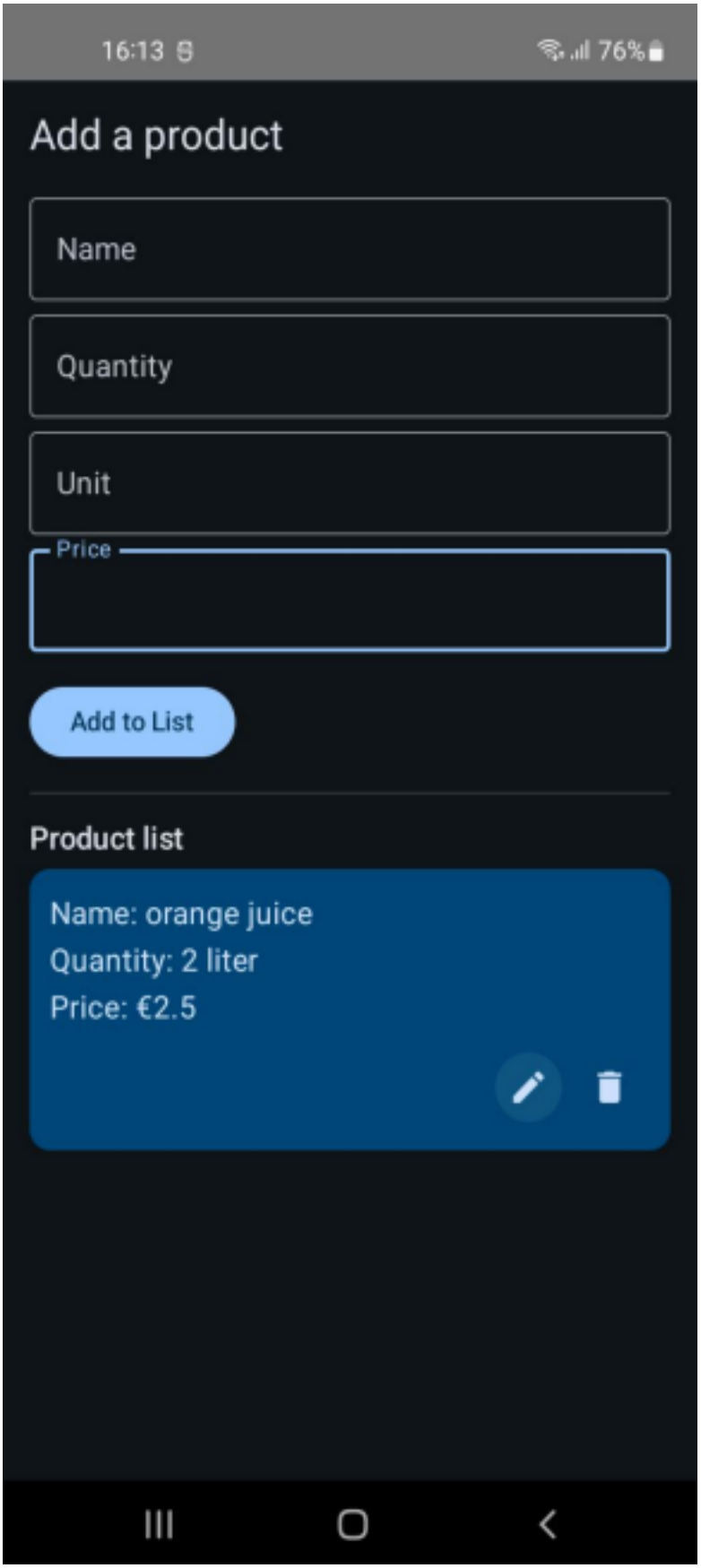
Mobile Technology and Programming

Assignment L9_EX1



ShoppingItem

```
package com.example.shoppinglist
import androidx.room.Entity
import androidx.room.PrimaryKey
@Entity(tableName = "shopping_items")
data class ShoppingItem(
@PrimaryKey(autoGenerate = true) val id: Int = 0,
val name: String,
val quantity: Int,
val unit: String,
val price: Double
)
```



Code

MainActivity.kt

```
package com.example.shoppinglist
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.viewModels
import androidx.compose.material3.*
```

```
import com.example.shoppinglist.ui.theme.ShoppingListTheme class MainActivity
: ComponentActivity() {
private val viewModel: ShoppingViewModel by viewModels() override fun
onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
setContent {
ShoppingListTheme {
Surface {
ShoppingListScreen()
}
class MainActivity : ComponentActivity() {
private val viewModel: ShoppingViewModel by viewModels() override fun
onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
setContent {
ShoppingListTheme {
Surface {
ShoppingListScreen()
}
}
}
}
}
}
}
}
}
}
```

Shopping Database

```
package com.example.shoppinglist
import android.content.Context
import androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase
@Database(entities = [ShoppingItem::class], version = 1, exportSchema = false)
abstract class ShoppingDatabase : RoomDatabase() {
abstract fun shoppingItemDao(): ShoppingItemDao
companion object {
@Volatile
private var INSTANCE: ShoppingDatabase? = null
fun getDatabase(context: Context): ShoppingDatabase {
return INSTANCE ?: synchronized(this) {
val instance = Room.databaseBuilder(
context.applicationContext,
ShoppingDatabase::class.java,
              "shopping_database"
          ).build()
          INSTANCE = instance
          instance
      }
  }
```

```
    }
```

Shopping List Screen

```
package com.example.shoppinglist
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.text.KeyboardOptions import
androidx.compose.foundation.lazy.LazyColumn import
androidx.compose.foundation.lazy.items
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Modifier
import androidx.compose.ui.platform.LocalContext import
androidx.compose.ui.text.input.KeyboardType import
androidx.compose.ui.unit.dp
import androidx.lifecycle.viewmodel.compose.viewModel import
androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.Delete import
androidx.compose.material.icons.filled.Edit @Composable
fun ShoppingListScreen() {
    val context = LocalContext.current
    val dao = ShoppingDatabase.getDatabase(context).shoppingItemDao() val viewModel:
    ShoppingViewModel = viewModel(factory = ShoppingViewModelFactory(dao))val items by
    viewModel.items.collectAsState()
    var name by remember { mutableStateOf("") }
    var quantity by remember { mutableStateOf("") }
    var unit by remember { mutableStateOf("") }
    var price by remember { mutableStateOf("") }
    Column(modifier = Modifier
        .fillMaxSize()
        .padding(16.dp)) {
        Text(text = "Add a product", style =
        MaterialTheme.typography.titleLarge)Spacer(modifier =
        Modifier.height(12.dp))
        OutlinedTextField(
            value = name,
            onValueChange = { name = it },
            label = { Text("Name") },
            modifier = Modifier.fillMaxWidth()
        )
        OutlinedTextField(
            value = quantity,
            onValueChange = { quantity = it },
            label = { Text("Quantity") },
            keyboardOptions = KeyboardOptions(keyboardType =
            KeyboardType.Number),modifier = Modifier.fillMaxWidth()
        )
        OutlinedTextField(
            value = unit,
            onValueChange = { unit = it },
            label = { Text("Unit") },
```

```
            modifier = Modifier.fillMaxWidth()
        )
        OutlinedTextField(
            value = price,
            onValueChange = { price = it },
            label = { Text("Price") },
            keyboardOptions = KeyboardOptions(keyboardType = KeyboardType.Decimal),
            modifier = Modifier.fillMaxWidth()
        )
        Button(
            onClick = {
                 if (name.isNotBlank() && quantity.isNotBlank() && price.isNotBlank()){
                    viewModel.addItem(
                        ShoppingItem(
                            name = name,
                                                quantity = quantity.toIntOrNull() ?: 0,
                            unit = unit,
                                                price = price.toDoubleOrNull() ?: 0.0
                        )
                    )
                    name = ""
                    quantity = ""
                    unit = ""
                    price = ""
                }
            },
            modifier = Modifier.padding(vertical = 16.dp)
        ) {
            Text("Add to List")
        }
        HorizontalDivider()
        Spacer(modifier = Modifier.height(12.dp))
        Text("Product list", style = MaterialTheme.typography.titleMedium) LazyColumn {
            items(items, key = { it.id }) { item ->
                ShoppingItemRow(
                    item = item,
                                        onDelete = { viewModel.deleteItem(item) },
                                        onUpdate = { viewModel.updateItem(item) }
                )
            }
        }
    }
}
@Composable
fun ShoppingItemRow(item: ShoppingItem, onDelete: () -> Unit, onUpdate: () -> Unit){Card(
        modifier = Modifier
            .fillMaxWidth()
            .padding(vertical = 4.dp),
        colors = CardDefaults.cardColors(containerColor =
MaterialTheme.colorScheme.primaryContainer)
    ) {
        Column(modifier = Modifier.padding(12.dp)) {
            Text("Name: ${item.name}")
```

```
            Text("Quantity: ${item.quantity} ${item.unit}")
            Text("Price: €${item.price}")
            Spacer(modifier = Modifier.height(8.dp))
            Row(
                horizontalArrangement = Arrangement.End,
                modifier = Modifier.fillMaxWidth()
            ) {
                IconButton(onClick = onUpdate) {
                    Icon(Icons.Filled.Edit, contentDescription = "Update") }
                IconButton(onClick = onDelete) {
                    Icon(Icons.Filled.Delete, contentDescription = "Delete") }
            }
        }
    }
}
```