User.kt

|  |
| --- |
|  |
|  | Import androidx.room.ColumnInfo  import androidx.room.Entity |
|  | import androidx.room.PrimaryKey |
|  |  |
|  | @Entity(tableName = "user\_table") |
|  | data class User( |
|  | @PrimaryKey(autoGenerate = true) val id: Int?, |
|  | @ColumnInfo(name = "first\_name") val firstName: String?, |
|  | @ColumnInfo(name = "last\_name") val lastName: String?, |
|  | @ColumnInfo(name = "email") val email: String?, |
|  | @ColumnInfo(name = "password") val password: String?, |
|  |  |
|  | ) |

userDao.kt

|  |
| --- |
|  |
|  |  |
|  | Import androidx.room.\*  @Dao |
|  | interface UserDao { |
|  |  |
|  | @Query("SELECT \* FROM user\_table WHERE email = :email") |
|  | suspend fun getUserByEmail(email: String): User? |
|  |  |
|  | @Insert(onConflict = OnConflictStrategy.REPLACE) |
|  | suspend fun insertUser(user: User) |
|  |  |
|  | @Update |
|  | suspend fun updateUser(user: User) |
|  |  |
|  | @Delete |
|  | suspend fun deleteUser(user: User) |
|  | } |

UserDatabase.kt

|  |
| --- |
|  |
|  | Import android.content.Context  import androidx.room.Database |
|  | import androidx.room.Room |
|  | import androidx.room.RoomDatabase |
|  |  |
|  | @Database(entities = [User::class], version = 1) |
|  | abstract class UserDatabase : RoomDatabase() { |
|  |  |
|  | abstract fun userDao(): UserDao |
|  |  |
|  | companion object { |
|  |  |
|  | @Volatile |
|  | private var instance: UserDatabase? = null |
|  |  |
|  | fun getDatabase(context: Context): UserDatabase { |
|  | return instance ?: synchronized(this) { |
|  | val newInstance = Room.databaseBuilder( |
|  | context.applicationContext, |
|  | UserDatabase::class.java, |
|  | "user\_database" |
|  | ).build() |
|  | instance = newInstance |
|  | newInstance |
|  | } |
|  | } |
|  | } |
|  | } |

UserDatabaseHelper.kt

|  |
| --- |
|  |
|  | Import android.annotation.SuppressLint  import android.content.ContentValues |
|  | import android.content.Context |
|  | import android.database.Cursor |
|  | import android.database.sqlite.SQLiteDatabase |
|  | import android.database.sqlite.SQLiteOpenHelper |
|  |  |
|  | class UserDatabaseHelper(context: Context) : |
|  | SQLiteOpenHelper(context, DATABASE\_NAME, null, DATABASE\_VERSION) { |
|  |  |
|  | companion object { |
|  | private const val DATABASE\_VERSION = 1 |
|  | private const val DATABASE\_NAME = "UserDatabase.db" |
|  |  |
|  | private const val TABLE\_NAME = "user\_table" |
|  | private const val COLUMN\_ID = "id" |
|  | private const val COLUMN\_FIRST\_NAME = "first\_name" |
|  | private const val COLUMN\_LAST\_NAME = "last\_name" |
|  | private const val COLUMN\_EMAIL = "email" |
|  | private const val COLUMN\_PASSWORD = "password" |
|  | } |
|  |  |
|  | override fun onCreate(db: SQLiteDatabase?) { |
|  | val createTable = "CREATE TABLE $TABLE\_NAME (" + |
|  | "$COLUMN\_ID INTEGER PRIMARY KEY AUTOINCREMENT, " + |
|  | "$COLUMN\_FIRST\_NAME TEXT, " + |
|  | "$COLUMN\_LAST\_NAME TEXT, " + |
|  | "$COLUMN\_EMAIL TEXT, " + |
|  | "$COLUMN\_PASSWORD TEXT" + |
|  | ")" |
|  |  |
|  | db?.execSQL(createTable) |
|  | } |
|  |  |
|  | override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion: Int) { |
|  | db?.execSQL("DROP TABLE IF EXISTS $TABLE\_NAME") |
|  | onCreate(db) |
|  | } |
|  |  |
|  | fun insertUser(user: User) { |
|  | val db = writableDatabase |
|  | val values = ContentValues() |
|  | values.put(COLUMN\_FIRST\_NAME, user.firstName) |
|  | values.put(COLUMN\_LAST\_NAME, user.lastName) |
|  | values.put(COLUMN\_EMAIL, user.email) |
|  | values.put(COLUMN\_PASSWORD, user.password) |
|  | db.insert(TABLE\_NAME, null, values) |
|  | db.close() |
|  | } |
|  |  |
|  | @SuppressLint("Range") |
|  | fun getUserByUsername(username: String): User? { |
|  | val db = readableDatabase |
|  | val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME WHERE $COLUMN\_FIRST\_NAME = ?", arrayOf(username)) |
|  | var user: User? = null |
|  | if (cursor.moveToFirst()) { |
|  | user = User( |
|  | id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)), |
|  | firstName = cursor.getString(cursor.getColumnIndex(COLUMN\_FIRST\_NAME)), |
|  | lastName = cursor.getString(cursor.getColumnIndex(COLUMN\_LAST\_NAME)), |
|  | email = cursor.getString(cursor.getColumnIndex(COLUMN\_EMAIL)), |
|  | password = cursor.getString(cursor.getColumnIndex(COLUMN\_PASSWORD)), |
|  | ) |
|  | } |
|  | cursor.close() |
|  | db.close() |
|  | return user |
|  | } |
|  | @SuppressLint("Range") |
|  | fun getUserById(id: Int): User? { |
|  | val db = readableDatabase |
|  | val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME WHERE $COLUMN\_ID = ?", arrayOf(id.toString())) |
|  | var user: User? = null |
|  | if (cursor.moveToFirst()) { |
|  | user = User( |
|  | id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)), |
|  | firstName = cursor.getString(cursor.getColumnIndex(COLUMN\_FIRST\_NAME)), |
|  | lastName = cursor.getString(cursor.getColumnIndex(COLUMN\_LAST\_NAME)), |
|  | email = cursor.getString(cursor.getColumnIndex(COLUMN\_EMAIL)), |
|  | password = cursor.getString(cursor.getColumnIndex(COLUMN\_PASSWORD)), |
|  | ) |
|  | } |
|  | cursor.close() |
|  | db.close() |
|  | return user |
|  | } |
|  |  |
|  | @SuppressLint("Range") |
|  | fun getAllUsers(): List<User> { |
|  | val users = mutableListOf<User>() |
|  | val db = readableDatabase |
|  | val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME", null) |
|  | if (cursor.moveToFirst()) { |
|  | do { |
|  | val user = User( |
|  | id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)), |
|  | firstName = cursor.getString(cursor.getColumnIndex(COLUMN\_FIRST\_NAME)), |
|  | lastName = cursor.getString(cursor.getColumnIndex(COLUMN\_LAST\_NAME)), |
|  | email = cursor.getString(cursor.getColumnIndex(COLUMN\_EMAIL)), |
|  | password = cursor.getString(cursor.getColumnIndex(COLUMN\_PASSWORD)), |
|  | ) |
|  | users.add(user) |
|  | } while (cursor.moveToNext()) |
|  | } |
|  | cursor.close() |
|  | db.close() |
|  | return users |
|  | } |
|  |  |
|  | } |

Items.kt

|  |
| --- |
|  |
|  | Import androidx.room.ColumnInfo  import androidx.room.Entity |
|  | import androidx.room.PrimaryKey |
|  |  |
|  | @Entity(tableName = "items\_table") |
|  | data class Items( |
|  | @PrimaryKey(autoGenerate = true) val id: Int?, |
|  | @ColumnInfo(name = "item\_name") val itemName: String?, |
|  | @ColumnInfo(name = "quantity") val quantity: String?, |
|  | @ColumnInfo(name = "cost") val cost: String?, |
|  | ) |

ItemsDao.kt

|  |
| --- |
|  |
|  |  |
|  | Import androidx.room.\*  @Dao |
|  | interface ItemsDao { |
|  |  |
|  | @Query("SELECT \* FROM items\_table WHERE cost= :cost") |
|  | suspend fun getItemsByCost(cost: String): Items? |
|  |  |
|  | @Insert(onConflict = OnConflictStrategy.REPLACE) |
|  | suspend fun insertItems(items: Items) |
|  |  |
|  | @Update |
|  | suspend fun updateItems(items: Items) |
|  |  |
|  | @Delete |
|  | suspend fun deleteItems(items: Items) |
|  | } |

ItemDatabase.kt

|  |
| --- |
|  |
|  | Import android.content.Context  import androidx.room.Database |
|  | import androidx.room.Room |
|  | import androidx.room.RoomDatabase |
|  |  |
|  | @Database(entities = [Items::class], version = 1) |
|  | abstract class ItemsDatabase : RoomDatabase() { |
|  |  |
|  | abstract fun ItemsDao(): ItemsDao |
|  |  |
|  | companion object { |
|  |  |
|  | @Volatile |
|  | private var instance: ItemsDatabase? = null |
|  |  |
|  | fun getDatabase(context: Context): ItemsDatabase { |
|  | return instance ?: synchronized(this) { |
|  | val newInstance = Room.databaseBuilder( |
|  | context.applicationContext, |
|  | ItemsDatabase::class.java, |
|  | "items\_database" |
|  | ).build() |
|  | instance = newInstance |
|  | newInstance |
|  | } |
|  | } |
|  | } |
|  | } |

ItemDatabaseHelper.kt

|  |
| --- |
|  |
|  | import android.annotation.SuppressLint  import android.content.ContentValues |
|  | import android.content.Context |
|  | import android.database.Cursor |
|  | import android.database.sqlite.SQLiteDatabase |
|  | import android.database.sqlite.SQLiteOpenHelper |
|  |  |
|  |  |
|  | class ItemsDatabaseHelper(context: Context) : |
|  | SQLiteOpenHelper(context, DATABASE\_NAME, null,DATABASE\_VERSION){ |
|  |  |
|  | companion object { |
|  | private const val DATABASE\_VERSION = 1 |
|  | private const val DATABASE\_NAME = "ItemsDatabase.db" |
|  |  |
|  | private const val TABLE\_NAME = "items\_table" |
|  | private const val COLUMN\_ID = "id" |
|  | private const val COLUMN\_ITEM\_NAME = "item\_name" |
|  | private const val COLUMN\_QUANTITY = "quantity" |
|  | private const val COLUMN\_COST = "cost" |
|  | } |
|  |  |
|  | override fun onCreate(db: SQLiteDatabase?) { |
|  | val createTable = "CREATE TABLE $TABLE\_NAME (" + |
|  | "${COLUMN\_ID} INTEGER PRIMARY KEY AUTOINCREMENT, " + |
|  | "${COLUMN\_ITEM\_NAME} TEXT," + |
|  | "${COLUMN\_QUANTITY} TEXT," + |
|  | "${COLUMN\_COST} TEXT" + |
|  | ")" |
|  |  |
|  | db?.execSQL(createTable) |
|  | } |
|  |  |
|  | override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion: Int) { |
|  | db?.execSQL("DROP TABLE IF EXISTS $TABLE\_NAME") |
|  | onCreate(db) |
|  | } |
|  |  |
|  | fun insertItems(items: Items) { |
|  | val db = writableDatabase |
|  | val values = ContentValues() |
|  | values.put(COLUMN\_ITEM\_NAME, items.itemName) |
|  | values.put(COLUMN\_QUANTITY, items.quantity) |
|  | values.put(COLUMN\_COST, items.cost) |
|  | db.insert(TABLE\_NAME, null, values) |
|  | db.close() |
|  | } |
|  |  |
|  |  |
|  |  |
|  | @SuppressLint("Range") |
|  | fun getItemsByCost(cost: String): Items? { |
|  | val db = readableDatabase |
|  | val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME WHERE $COLUMN\_COST = ?", arrayOf(cost)) |
|  | var items: Items? = null |
|  | if (cursor.moveToFirst()) { |
|  | items = Items( |
|  | id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)), |
|  | itemName = cursor.getString(cursor.getColumnIndex(COLUMN\_ITEM\_NAME)), |
|  | quantity = cursor.getString(cursor.getColumnIndex(COLUMN\_QUANTITY)), |
|  | cost = cursor.getString(cursor.getColumnIndex(COLUMN\_COST)), |
|  | ) |
|  | } |
|  | cursor.close() |
|  | db.close() |
|  | return items |
|  | } |
|  | @SuppressLint("Range") |
|  | fun getItemsById(id: Int): Items? { |
|  | val db = readableDatabase |
|  | val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME WHERE $COLUMN\_ID = ?", arrayOf(id.toString())) |
|  | var items: Items? = null |
|  | if (cursor.moveToFirst()) { |
|  | items = Items( |
|  | id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)), |
|  | itemName = cursor.getString(cursor.getColumnIndex(COLUMN\_ITEM\_NAME)), |
|  | quantity = cursor.getString(cursor.getColumnIndex(COLUMN\_QUANTITY)), |
|  | cost = cursor.getString(cursor.getColumnIndex(COLUMN\_COST)), |
|  | ) |
|  | } |
|  | cursor.close() |
|  | db.close() |
|  | return items |
|  | } |
|  |  |
|  | @SuppressLint("Range") |
|  | fun getAllItems(): List<Items> { |
|  | val item = mutableListOf<Items>() |
|  | val db = readableDatabase |
|  | val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME", null) |
|  | if (cursor.moveToFirst()) { |
|  | do { |
|  | val items = Items( |
|  | id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)), |
|  | itemName = cursor.getString(cursor.getColumnIndex(COLUMN\_ITEM\_NAME)), |
|  | quantity = cursor.getString(cursor.getColumnIndex(COLUMN\_QUANTITY)), |
|  | cost = cursor.getString(cursor.getColumnIndex(COLUMN\_COST)), |
|  | ) |
|  | item.add(items) |
|  | } while (cursor.moveToNext()) |
|  | } |
|  | cursor.close() |
|  | db.close() |
|  | return item |
|  | } |
|  |  |
|  |  |
|  | } |

Expense.kt

User.kt

|  |
| --- |
|  |
|  | Import androidx.room.ColumnInfo  import androidx.room.Entity |
|  | import androidx.room.PrimaryKey |
|  |  |
|  | @Entity(tableName = "user\_table") |
|  | data class User( |
|  | @PrimaryKey(autoGenerate = true) val id: Int?, |
|  | @ColumnInfo(name = "first\_name") val firstName: String?, |
|  | @ColumnInfo(name = "last\_name") val lastName: String?, |
|  | @ColumnInfo(name = "email") val email: String?, |
|  | @ColumnInfo(name = "password") val password: String?, |
|  |  |
|  | ) |

userDao.kt

|  |
| --- |
|  |
|  |  |
|  | Import androidx.room.\*  @Dao |
|  | interface UserDao { |
|  |  |
|  | @Query("SELECT \* FROM user\_table WHERE email = :email") |
|  | suspend fun getUserByEmail(email: String): User? |
|  |  |
|  | @Insert(onConflict = OnConflictStrategy.REPLACE) |
|  | suspend fun insertUser(user: User) |
|  |  |
|  | @Update |
|  | suspend fun updateUser(user: User) |
|  |  |
|  | @Delete |
|  | suspend fun deleteUser(user: User) |
|  | } |

UserDatabase.kt

|  |
| --- |
|  |
|  | Import android.content.Context  import androidx.room.Database |
|  | import androidx.room.Room |
|  | import androidx.room.RoomDatabase |
|  |  |
|  | @Database(entities = [User::class], version = 1) |
|  | abstract class UserDatabase : RoomDatabase() { |
|  |  |
|  | abstract fun userDao(): UserDao |
|  |  |
|  | companion object { |
|  |  |
|  | @Volatile |
|  | private var instance: UserDatabase? = null |
|  |  |
|  | fun getDatabase(context: Context): UserDatabase { |
|  | return instance ?: synchronized(this) { |
|  | val newInstance = Room.databaseBuilder( |
|  | context.applicationContext, |
|  | UserDatabase::class.java, |
|  | "user\_database" |
|  | ).build() |
|  | instance = newInstance |
|  | newInstance |
|  | } |
|  | } |
|  | } |
|  | } |

UserDatabaseHelper.kt

|  |
| --- |
|  |
|  | Import android.annotation.SuppressLint  import android.content.ContentValues |
|  | import android.content.Context |
|  | import android.database.Cursor |
|  | import android.database.sqlite.SQLiteDatabase |
|  | import android.database.sqlite.SQLiteOpenHelper |
|  |  |
|  | class UserDatabaseHelper(context: Context) : |
|  | SQLiteOpenHelper(context, DATABASE\_NAME, null, DATABASE\_VERSION) { |
|  |  |
|  | companion object { |
|  | private const val DATABASE\_VERSION = 1 |
|  | private const val DATABASE\_NAME = "UserDatabase.db" |
|  |  |
|  | private const val TABLE\_NAME = "user\_table" |
|  | private const val COLUMN\_ID = "id" |
|  | private const val COLUMN\_FIRST\_NAME = "first\_name" |
|  | private const val COLUMN\_LAST\_NAME = "last\_name" |
|  | private const val COLUMN\_EMAIL = "email" |
|  | private const val COLUMN\_PASSWORD = "password" |
|  | } |
|  |  |
|  | override fun onCreate(db: SQLiteDatabase?) { |
|  | val createTable = "CREATE TABLE $TABLE\_NAME (" + |
|  | "$COLUMN\_ID INTEGER PRIMARY KEY AUTOINCREMENT, " + |
|  | "$COLUMN\_FIRST\_NAME TEXT, " + |
|  | "$COLUMN\_LAST\_NAME TEXT, " + |
|  | "$COLUMN\_EMAIL TEXT, " + |
|  | "$COLUMN\_PASSWORD TEXT" + |
|  | ")" |
|  |  |
|  | db?.execSQL(createTable) |
|  | } |
|  |  |
|  | override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion: Int) { |
|  | db?.execSQL("DROP TABLE IF EXISTS $TABLE\_NAME") |
|  | onCreate(db) |
|  | } |
|  |  |
|  | fun insertUser(user: User) { |
|  | val db = writableDatabase |
|  | val values = ContentValues() |
|  | values.put(COLUMN\_FIRST\_NAME, user.firstName) |
|  | values.put(COLUMN\_LAST\_NAME, user.lastName) |
|  | values.put(COLUMN\_EMAIL, user.email) |
|  | values.put(COLUMN\_PASSWORD, user.password) |
|  | db.insert(TABLE\_NAME, null, values) |
|  | db.close() |
|  | } |
|  |  |
|  | @SuppressLint("Range") |
|  | fun getUserByUsername(username: String): User? { |
|  | val db = readableDatabase |
|  | val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME WHERE $COLUMN\_FIRST\_NAME = ?", arrayOf(username)) |
|  | var user: User? = null |
|  | if (cursor.moveToFirst()) { |
|  | user = User( |
|  | id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)), |
|  | firstName = cursor.getString(cursor.getColumnIndex(COLUMN\_FIRST\_NAME)), |
|  | lastName = cursor.getString(cursor.getColumnIndex(COLUMN\_LAST\_NAME)), |
|  | email = cursor.getString(cursor.getColumnIndex(COLUMN\_EMAIL)), |
|  | password = cursor.getString(cursor.getColumnIndex(COLUMN\_PASSWORD)), |
|  | ) |
|  | } |
|  | cursor.close() |
|  | db.close() |
|  | return user |
|  | } |
|  | @SuppressLint("Range") |
|  | fun getUserById(id: Int): User? { |
|  | val db = readableDatabase |
|  | val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME WHERE $COLUMN\_ID = ?", arrayOf(id.toString())) |
|  | var user: User? = null |
|  | if (cursor.moveToFirst()) { |
|  | user = User( |
|  | id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)), |
|  | firstName = cursor.getString(cursor.getColumnIndex(COLUMN\_FIRST\_NAME)), |
|  | lastName = cursor.getString(cursor.getColumnIndex(COLUMN\_LAST\_NAME)), |
|  | email = cursor.getString(cursor.getColumnIndex(COLUMN\_EMAIL)), |
|  | password = cursor.getString(cursor.getColumnIndex(COLUMN\_PASSWORD)), |
|  | ) |
|  | } |
|  | cursor.close() |
|  | db.close() |
|  | return user |
|  | } |
|  |  |
|  | @SuppressLint("Range") |
|  | fun getAllUsers(): List<User> { |
|  | val users = mutableListOf<User>() |
|  | val db = readableDatabase |
|  | val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME", null) |
|  | if (cursor.moveToFirst()) { |
|  | do { |
|  | val user = User( |
|  | id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)), |
|  | firstName = cursor.getString(cursor.getColumnIndex(COLUMN\_FIRST\_NAME)), |
|  | lastName = cursor.getString(cursor.getColumnIndex(COLUMN\_LAST\_NAME)), |
|  | email = cursor.getString(cursor.getColumnIndex(COLUMN\_EMAIL)), |
|  | password = cursor.getString(cursor.getColumnIndex(COLUMN\_PASSWORD)), |
|  | ) |
|  | users.add(user) |
|  | } while (cursor.moveToNext()) |
|  | } |
|  | cursor.close() |
|  | db.close() |
|  | return users |
|  | } |
|  |  |
|  | } |

Items.kt

|  |
| --- |
|  |
|  | Import androidx.room.ColumnInfo  import androidx.room.Entity |
|  | import androidx.room.PrimaryKey |
|  |  |
|  | @Entity(tableName = "items\_table") |
|  | data class Items( |
|  | @PrimaryKey(autoGenerate = true) val id: Int?, |
|  | @ColumnInfo(name = "item\_name") val itemName: String?, |
|  | @ColumnInfo(name = "quantity") val quantity: String?, |
|  | @ColumnInfo(name = "cost") val cost: String?, |
|  | ) |

ItemsDao.kt

|  |
| --- |
|  |
|  |  |
|  | Import androidx.room.\*  @Dao |
|  | interface ItemsDao { |
|  |  |
|  | @Query("SELECT \* FROM items\_table WHERE cost= :cost") |
|  | suspend fun getItemsByCost(cost: String): Items? |
|  |  |
|  | @Insert(onConflict = OnConflictStrategy.REPLACE) |
|  | suspend fun insertItems(items: Items) |
|  |  |
|  | @Update |
|  | suspend fun updateItems(items: Items) |
|  |  |
|  | @Delete |
|  | suspend fun deleteItems(items: Items) |
|  | } |

ItemDatabase.kt

|  |
| --- |
|  |
|  | Import android.content.Context  import androidx.room.Database |
|  | import androidx.room.Room |
|  | import androidx.room.RoomDatabase |
|  |  |
|  | @Database(entities = [Items::class], version = 1) |
|  | abstract class ItemsDatabase : RoomDatabase() { |
|  |  |
|  | abstract fun ItemsDao(): ItemsDao |
|  |  |
|  | companion object { |
|  |  |
|  | @Volatile |
|  | private var instance: ItemsDatabase? = null |
|  |  |
|  | fun getDatabase(context: Context): ItemsDatabase { |
|  | return instance ?: synchronized(this) { |
|  | val newInstance = Room.databaseBuilder( |
|  | context.applicationContext, |
|  | ItemsDatabase::class.java, |
|  | "items\_database" |
|  | ).build() |
|  | instance = newInstance |
|  | newInstance |
|  | } |
|  | } |
|  | } |
|  | } |

ItemDatabaseHelper.kt

|  |
| --- |
|  |
|  | import android.annotation.SuppressLint  import android.content.ContentValues |
|  | import android.content.Context |
|  | import android.database.Cursor |
|  | import android.database.sqlite.SQLiteDatabase |
|  | import android.database.sqlite.SQLiteOpenHelper |
|  |  |
|  |  |
|  | class ItemsDatabaseHelper(context: Context) : |
|  | SQLiteOpenHelper(context, DATABASE\_NAME, null,DATABASE\_VERSION){ |
|  |  |
|  | companion object { |
|  | private const val DATABASE\_VERSION = 1 |
|  | private const val DATABASE\_NAME = "ItemsDatabase.db" |
|  |  |
|  | private const val TABLE\_NAME = "items\_table" |
|  | private const val COLUMN\_ID = "id" |
|  | private const val COLUMN\_ITEM\_NAME = "item\_name" |
|  | private const val COLUMN\_QUANTITY = "quantity" |
|  | private const val COLUMN\_COST = "cost" |
|  | } |
|  |  |
|  | override fun onCreate(db: SQLiteDatabase?) { |
|  | val createTable = "CREATE TABLE $TABLE\_NAME (" + |
|  | "${COLUMN\_ID} INTEGER PRIMARY KEY AUTOINCREMENT, " + |
|  | "${COLUMN\_ITEM\_NAME} TEXT," + |
|  | "${COLUMN\_QUANTITY} TEXT," + |
|  | "${COLUMN\_COST} TEXT" + |
|  | ")" |
|  |  |
|  | db?.execSQL(createTable) |
|  | } |
|  |  |
|  | override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion: Int) { |
|  | db?.execSQL("DROP TABLE IF EXISTS $TABLE\_NAME") |
|  | onCreate(db) |
|  | } |
|  |  |
|  | fun insertItems(items: Items) { |
|  | val db = writableDatabase |
|  | val values = ContentValues() |
|  | values.put(COLUMN\_ITEM\_NAME, items.itemName) |
|  | values.put(COLUMN\_QUANTITY, items.quantity) |
|  | values.put(COLUMN\_COST, items.cost) |
|  | db.insert(TABLE\_NAME, null, values) |
|  | db.close() |
|  | } |
|  |  |
|  |  |
|  |  |
|  | @SuppressLint("Range") |
|  | fun getItemsByCost(cost: String): Items? { |
|  | val db = readableDatabase |
|  | val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME WHERE $COLUMN\_COST = ?", arrayOf(cost)) |
|  | var items: Items? = null |
|  | if (cursor.moveToFirst()) { |
|  | items = Items( |
|  | id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)), |
|  | itemName = cursor.getString(cursor.getColumnIndex(COLUMN\_ITEM\_NAME)), |
|  | quantity = cursor.getString(cursor.getColumnIndex(COLUMN\_QUANTITY)), |
|  | cost = cursor.getString(cursor.getColumnIndex(COLUMN\_COST)), |
|  | ) |
|  | } |
|  | cursor.close() |
|  | db.close() |
|  | return items |
|  | } |
|  | @SuppressLint("Range") |
|  | fun getItemsById(id: Int): Items? { |
|  | val db = readableDatabase |
|  | val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME WHERE $COLUMN\_ID = ?", arrayOf(id.toString())) |
|  | var items: Items? = null |
|  | if (cursor.moveToFirst()) { |
|  | items = Items( |
|  | id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)), |
|  | itemName = cursor.getString(cursor.getColumnIndex(COLUMN\_ITEM\_NAME)), |
|  | quantity = cursor.getString(cursor.getColumnIndex(COLUMN\_QUANTITY)), |
|  | cost = cursor.getString(cursor.getColumnIndex(COLUMN\_COST)), |
|  | ) |
|  | } |
|  | cursor.close() |
|  | db.close() |
|  | return items |
|  | } |
|  |  |
|  | @SuppressLint("Range") |
|  | fun getAllItems(): List<Items> { |
|  | val item = mutableListOf<Items>() |
|  | val db = readableDatabase |
|  | val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME", null) |
|  | if (cursor.moveToFirst()) { |
|  | do { |
|  | val items = Items( |
|  | id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)), |
|  | itemName = cursor.getString(cursor.getColumnIndex(COLUMN\_ITEM\_NAME)), |
|  | quantity = cursor.getString(cursor.getColumnIndex(COLUMN\_QUANTITY)), |
|  | cost = cursor.getString(cursor.getColumnIndex(COLUMN\_COST)), |
|  | ) |
|  | item.add(items) |
|  | } while (cursor.moveToNext()) |
|  | } |
|  | cursor.close() |
|  | db.close() |
|  | return item |
|  | } |
|  |  |
|  |  |
|  | } |

Expense.kt

|  |
| --- |
|  |
|  | Import androidx.room.ColumnInfo  import androidx.room.Entity |
|  | import androidx.room.PrimaryKey |
|  |  |
|  | @Entity(tableName = "expense\_table") |
|  | data class Expense( |
|  | @PrimaryKey(autoGenerate = true) val id: Int?, |
|  | @ColumnInfo(name = "amount") val amount: String?, |
|  | ) |

ExpenseDao.kt

|  |
| --- |
|  |
|  |  |
|  | Import andoridx.room.\*  @Dao |
|  | interface ExpenseDao { |
|  |  |
|  | @Query("SELECT \* FROM expense\_table WHERE amount= :amount") |
|  | suspend fun getExpenseByAmount(amount: String): Expense? |
|  |  |
|  | @Insert(onConflict = OnConflictStrategy.REPLACE) |
|  | suspend fun insertExpense(items: Expense) |
|  |  |
|  | @Update |
|  | suspend fun updateExpense(items: Expense) |
|  |  |
|  | @Delete |
|  | suspend fun deleteExpense(items: Expense) |
|  | } |

ExpenseDatabaseHelper.kt

|  |
| --- |
|  |
|  | Import android.annotation.SuppressLint  import android.content.ContentValues |
|  | import android.content.Context |
|  | import android.database.Cursor |
|  | import android.database.sqlite.SQLiteDatabase |
|  | import android.database.sqlite.SQLiteOpenHelper |
|  |  |
|  | class ExpenseDatabaseHelper(context: Context) : |
|  | SQLiteOpenHelper(context, DATABASE\_NAME, null,DATABASE\_VERSION){ |
|  |  |
|  | companion object { |
|  | private const val DATABASE\_VERSION = 1 |
|  | private const val DATABASE\_NAME = "ExpenseDatabase.db" |
|  |  |
|  | private const val TABLE\_NAME = "expense\_table" |
|  | private const val COLUMN\_ID = "id" |
|  | private const val COLUMN\_AMOUNT = "amount" |
|  | } |
|  |  |
|  | override fun onCreate(db: SQLiteDatabase?) { |
|  | val createTable = "CREATE TABLE $TABLE\_NAME (" + |
|  | "${COLUMN\_ID} INTEGER PRIMARY KEY AUTOINCREMENT, " + |
|  | "${COLUMN\_AMOUNT} TEXT" + |
|  | ")" |
|  |  |
|  | db?.execSQL(createTable) |
|  | } |
|  |  |
|  | override fun onUpgrade(db1: SQLiteDatabase?, oldVersion: Int, newVersion: Int) { |
|  | db1?.execSQL("DROP TABLE IF EXISTS $TABLE\_NAME") |
|  | onCreate(db1) |
|  | } |
|  |  |
|  | fun insertExpense(expense: Expense) { |
|  | val db1 = writableDatabase |
|  | val values = ContentValues() |
|  | values.put(COLUMN\_AMOUNT, expense.amount) |
|  | db1.insert(TABLE\_NAME, null, values) |
|  | db1.close() |
|  | } |
|  |  |
|  | fun updateExpense(expense: Expense) { |
|  | val db = writableDatabase |
|  | val values = ContentValues() |
|  | values.put(COLUMN\_AMOUNT, expense.amount) |
|  | db.update(TABLE\_NAME, values, "$COLUMN\_ID=?", arrayOf(expense.id.toString())) |
|  | db.close() |
|  | } |
|  |  |
|  |  |
|  |  |
|  | @SuppressLint("Range") |
|  | fun getExpenseByAmount(amount: String): Expense? { |
|  | val db1 = readableDatabase |
|  | val cursor: Cursor = db1.rawQuery("SELECT \* FROM ${ExpenseDatabaseHelper.TABLE\_NAME} WHERE ${ExpenseDatabaseHelper.COLUMN\_AMOUNT} = ?", arrayOf(amount)) |
|  | var expense: Expense? = null |
|  | if (cursor.moveToFirst()) { |
|  | expense = Expense( |
|  | id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)), |
|  | amount = cursor.getString(cursor.getColumnIndex(COLUMN\_AMOUNT)), |
|  | ) |
|  | } |
|  | cursor.close() |
|  | db1.close() |
|  | return expense |
|  | } |
|  | @SuppressLint("Range") |
|  | fun getExpenseById(id: Int): Expense? { |
|  | val db1 = readableDatabase |
|  | val cursor: Cursor = db1.rawQuery("SELECT \* FROM $TABLE\_NAME WHERE $COLUMN\_ID = ?", arrayOf(id.toString())) |
|  | var expense: Expense? = null |
|  | if (cursor.moveToFirst()) { |
|  | expense = Expense( |
|  | id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)), |
|  | amount = cursor.getString(cursor.getColumnIndex(COLUMN\_AMOUNT)), |
|  | ) |
|  | } |
|  | cursor.close() |
|  | db1.close() |
|  | return expense |
|  | } |
|  | @SuppressLint("Range") |
|  | fun getExpenseAmount(id: Int): Int? { |
|  | val db = readableDatabase |
|  | val query = "SELECT $COLUMN\_AMOUNT FROM $TABLE\_NAME WHERE $COLUMN\_ID=?" |
|  | val cursor = db.rawQuery(query, arrayOf(id.toString())) |
|  | var amount: Int? = null |
|  | if (cursor.moveToFirst()) { |
|  | amount = cursor.getInt(cursor.getColumnIndex(COLUMN\_AMOUNT)) |
|  | } |
|  | cursor.close() |
|  | db.close() |
|  | return amount |
|  | } |
|  | @SuppressLint("Range") |
|  | fun getAllExpense(): List<Expense> { |
|  | val expenses = mutableListOf<Expense>() |
|  | val db1 = readableDatabase |
|  | val cursor: Cursor = db1.rawQuery("SELECT \* FROM $TABLE\_NAME", null) |
|  | if (cursor.moveToFirst()) { |
|  | do { |
|  | val expense = Expense( |
|  | id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)), |
|  | amount = cursor.getString(cursor.getColumnIndex(COLUMN\_AMOUNT)), |
|  | ) |
|  | expenses.add(expense) |
|  | } while (cursor.moveToNext()) |
|  | } |
|  | cursor.close() |
|  | db1.close() |
|  | return expenses |
|  | } |
|  |  |
|  |  |
|  |  |
|  |  |
|  | } |

LoginActivity.kt

|  |
| --- |
|  |
|  | Import android.content.Context  import android.content.Intent |
|  | import android.os.Bundle |
|  | import androidx.activity.ComponentActivity |
|  | import androidx.activity.compose.setContent |
|  | import androidx.compose.foundation.Image |
|  | import androidx.compose.foundation.layout.\* |
|  | import androidx.compose.material.\* |
|  | import androidx.compose.runtime.\* |
|  | import androidx.compose.ui.Alignment |
|  | import androidx.compose.ui.Modifier |
|  | import androidx.compose.ui.graphics.Color |
|  | import androidx.compose.ui.layout.ContentScale |
|  | import androidx.compose.ui.res.painterResource |
|  | import androidx.compose.ui.text.font.FontFamily |
|  | import androidx.compose.ui.text.font.FontWeight |
|  | import androidx.compose.ui.text.input.PasswordVisualTransformation |
|  | import androidx.compose.ui.text.input.VisualTransformation |
|  | import androidx.compose.ui.tooling.preview.Preview |
|  | import androidx.compose.ui.unit.dp |
|  | import androidx.compose.ui.unit.sp |
|  | import androidx.core.content.ContextCompat |
|  | import com.example.expensestracker.ui.theme.ExpensesTrackerTheme |
|  |  |
|  | class LoginActivity : ComponentActivity() { |
|  | private lateinit var databaseHelper: UserDatabaseHelper |
|  | override fun onCreate(savedInstanceState: Bundle?) { |
|  | super.onCreate(savedInstanceState) |
|  | databaseHelper = UserDatabaseHelper(this) |
|  | setContent { |
|  | ExpensesTrackerTheme { |
|  | // A surface container using the 'background' color from the theme |
|  | Surface( |
|  | modifier = Modifier.fillMaxSize(), |
|  | color = MaterialTheme.colors.background |
|  | ) { |
|  | LoginScreen(this, databaseHelper) |
|  | } |
|  | } |
|  | } |
|  | } |
|  | } |
|  | @Composable |
|  | fun LoginScreen(context: Context, databaseHelper: UserDatabaseHelper) { |
|  |  |
|  | Image( |
|  | painterResource(id = R.drawable.img\_1), contentDescription = "", |
|  | alpha =0.3F, |
|  | contentScale = ContentScale.FillHeight, |
|  |  |
|  | ) |
|  |  |
|  | var username by remember { mutableStateOf("") } |
|  | var password by remember { mutableStateOf("") } |
|  | var error by remember { mutableStateOf("") } |
|  |  |
|  | Column( |
|  | modifier = Modifier.fillMaxSize(), |
|  | horizontalAlignment = Alignment.CenterHorizontally, |
|  | verticalArrangement = Arrangement.Center |
|  | ) { |
|  |  |
|  | Text( |
|  | fontSize = 36.sp, |
|  | fontWeight = FontWeight.ExtraBold, |
|  | fontFamily = FontFamily.Cursive, |
|  | color = Color.White, |
|  | text = "Login" |
|  | ) |
|  | Spacer(modifier = Modifier.height(10.dp)) |
|  |  |
|  | TextField( |
|  | value = username, |
|  | onValueChange = { username = it }, |
|  | label = { Text("Username") }, |
|  | modifier = Modifier.padding(10.dp) |
|  | .width(280.dp) |
|  | ) |
|  |  |
|  | TextField( |
|  | value = password, |
|  | onValueChange = { password = it }, |
|  | label = { Text("Password") }, |
|  | modifier = Modifier.padding(10.dp) |
|  | .width(280.dp), |
|  | visualTransformation = PasswordVisualTransformation() |
|  |  |
|  | ) |
|  |  |
|  | if (error.isNotEmpty()) { |
|  | Text( |
|  | text = error, |
|  | color = MaterialTheme.colors.error, |
|  | modifier = Modifier.padding(vertical = 16.dp) |
|  | ) |
|  | } |
|  |  |
|  | Button( |
|  | onClick = { |
|  | if (username.isNotEmpty() && password.isNotEmpty()) { |
|  | val user = databaseHelper.getUserByUsername(username) |
|  | if (user != null && user.password == password) { |
|  | error = "Successfully log in" |
|  | context.startActivity( |
|  | Intent( |
|  | context, |
|  | MainActivity::class.java |
|  | ) |
|  | ) |
|  | //onLoginSuccess() |
|  | } |
|  | else { |
|  | error = "Invalid username or password" |
|  | } |
|  |  |
|  | } else { |
|  | error = "Please fill all fields" |
|  | } |
|  | }, |
|  | modifier = Modifier.padding(top = 16.dp) |
|  | ) { |
|  | Text(text = "Login") |
|  | } |
|  | Row { |
|  | TextButton(onClick = {context.startActivity( |
|  | Intent( |
|  | context, |
|  | RegisterActivity::class.java |
|  | ) |
|  | )} |
|  | ) |
|  | { Text(color = Color.White,text = "Sign up") } |
|  | TextButton(onClick = { |
|  | }) |
|  |  |
|  | { |
|  | Spacer(modifier = Modifier.width(60.dp)) |
|  | Text(color = Color.White,text = "Forget password?") |
|  | } |
|  | } |
|  | } |
|  | } |
|  | private fun startMainPage(context: Context) { |
|  | val intent = Intent(context, MainActivity::class.java) |
|  | ContextCompat.startActivity(context, intent, null) |
|  | } |

RegisterActivity.kt

|  |
| --- |
|  |
|  | Import android.content.Context  import android.content.Intent |
|  | import android.os.Bundle |
|  | import androidx.activity.ComponentActivity |
|  | import androidx.activity.compose.setContent |
|  | import androidx.compose.foundation.Image |
|  | import androidx.compose.foundation.layout.\* |
|  | import androidx.compose.material.\* |
|  | import androidx.compose.runtime.\* |
|  | import androidx.compose.ui.Alignment |
|  | import androidx.compose.ui.Modifier |
|  | import androidx.compose.ui.graphics.Color |
|  | import androidx.compose.ui.layout.ContentScale |
|  | import androidx.compose.ui.res.painterResource |
|  | import androidx.compose.ui.text.font.FontFamily |
|  | import androidx.compose.ui.text.font.FontWeight |
|  | import androidx.compose.ui.text.input.PasswordVisualTransformation |
|  | import androidx.compose.ui.tooling.preview.Preview |
|  | import androidx.compose.ui.unit.dp |
|  | import androidx.compose.ui.unit.sp |
|  | import androidx.core.content.ContextCompat |
|  | import com.example.expensestracker.ui.theme.ExpensesTrackerTheme |
|  |  |
|  | class RegisterActivity : ComponentActivity() { |
|  | private lateinit var databaseHelper: UserDatabaseHelper |
|  | override fun onCreate(savedInstanceState: Bundle?) { |
|  | super.onCreate(savedInstanceState) |
|  | databaseHelper = UserDatabaseHelper(this) |
|  | setContent { |
|  | ExpensesTrackerTheme { |
|  | // A surface container using the 'background' color from the theme |
|  | Surface( |
|  | modifier = Modifier.fillMaxSize(), |
|  | color = MaterialTheme.colors.background |
|  | ) { |
|  |  |
|  | RegistrationScreen(this,databaseHelper) |
|  | } |
|  | } |
|  | } |
|  | } |
|  | } |
|  |  |
|  | @Composable |
|  | fun RegistrationScreen(context: Context, databaseHelper: UserDatabaseHelper) { |
|  |  |
|  | Image( |
|  | painterResource(id = R.drawable.img\_1), contentDescription = "", |
|  | alpha =0.3F, |
|  | contentScale = ContentScale.FillHeight, |
|  |  |
|  | ) |
|  |  |
|  | var username by remember { mutableStateOf("") } |
|  | var password by remember { mutableStateOf("") } |
|  | var email by remember { mutableStateOf("") } |
|  | var error by remember { mutableStateOf("") } |
|  |  |
|  | Column( |
|  | modifier = Modifier.fillMaxSize(), |
|  | horizontalAlignment = Alignment.CenterHorizontally, |
|  | verticalArrangement = Arrangement.Center |
|  | ) { |
|  |  |
|  | Text( |
|  | fontSize = 36.sp, |
|  | fontWeight = FontWeight.ExtraBold, |
|  | fontFamily = FontFamily.Cursive, |
|  | color = Color.White, |
|  | text = "Register" |
|  | ) |
|  |  |
|  | Spacer(modifier = Modifier.height(10.dp)) |
|  | TextField( |
|  | value = username, |
|  | onValueChange = { username = it }, |
|  | label = { Text("Username") }, |
|  | modifier = Modifier |
|  | .padding(10.dp) |
|  | .width(280.dp) |
|  |  |
|  | ) |
|  |  |
|  | TextField( |
|  | value = email, |
|  | onValueChange = { email = it }, |
|  | label = { Text("Email") }, |
|  | modifier = Modifier |
|  | .padding(10.dp) |
|  | .width(280.dp) |
|  | ) |
|  |  |
|  | TextField( |
|  | value = password, |
|  | onValueChange = { password = it }, |
|  | label = { Text("Password") }, |
|  | modifier = Modifier |
|  | .padding(10.dp) |
|  | .width(280.dp), |
|  | visualTransformation = PasswordVisualTransformation() |
|  | ) |
|  |  |
|  |  |
|  | if (error.isNotEmpty()) { |
|  | Text( |
|  | text = error, |
|  | color = MaterialTheme.colors.error, |
|  | modifier = Modifier.padding(vertical = 16.dp) |
|  | ) |
|  | } |
|  |  |
|  | Button( |
|  | onClick = { |
|  | if (username.isNotEmpty() && password.isNotEmpty() && email.isNotEmpty()) { |
|  | val user = User( |
|  | id = null, |
|  | firstName = username, |
|  | lastName = null, |
|  | email = email, |
|  | password = password |
|  | ) |
|  | databaseHelper.insertUser(user) |
|  | error = "User registered successfully" |
|  | // Start LoginActivity using the current context |
|  | context.startActivity( |
|  | Intent( |
|  | context, |
|  | LoginActivity::class.java |
|  | ) |
|  | ) |
|  |  |
|  | } else { |
|  | error = "Please fill all fields" |
|  | } |
|  | }, |
|  | modifier = Modifier.padding(top = 16.dp) |
|  | ) { |
|  | Text(text = "Register") |
|  | } |
|  | Spacer(modifier = Modifier.width(10.dp)) |
|  | Spacer(modifier = Modifier.height(10.dp)) |
|  |  |
|  | Row() { |
|  | Text( |
|  | modifier = Modifier.padding(top = 14.dp), text = "Have an account?" |
|  | ) |
|  | TextButton(onClick = { |
|  | context.startActivity( |
|  | Intent( |
|  | context, |
|  | LoginActivity::class.java |
|  | ) |
|  | ) |
|  | }) |
|  |  |
|  | { |
|  | Spacer(modifier = Modifier.width(10.dp)) |
|  | Text(text = "Log in") |
|  | } |
|  | } |
|  | } |
|  | } |
|  | private fun startLoginActivity(context: Context) { |
|  | val intent = Intent(context, LoginActivity::class.java) |
|  | ContextCompat.startActivity(context, intent, null) |
|  | } |

MainActivity.kt

|  |
| --- |
| import |
|  | Import annotation.SuppressLint  import android.content.Intent |
|  | import android.os.Bundle |
|  | import androidx.activity.ComponentActivity |
|  | import androidx.activity.compose.setContent |
|  | import androidx.compose.foundation.Image |
|  | import androidx.compose.foundation.layout.\* |
|  | import androidx.compose.material.\* |
|  | import androidx.compose.runtime.\* |
|  | import androidx.compose.ui.Alignment |
|  | import androidx.compose.ui.Modifier |
|  | import androidx.compose.ui.graphics.Color |
|  | import androidx.compose.ui.res.painterResource |
|  | import androidx.compose.ui.text.font.FontWeight |
|  | import androidx.compose.ui.text.style.TextAlign |
|  | import androidx.compose.ui.tooling.preview.Preview |
|  | import androidx.compose.ui.unit.dp |
|  | import androidx.compose.ui.unit.sp |
|  | import com.example.expensestracker.ui.theme.ExpensesTrackerTheme |
|  |  |
|  | class MainActivity : ComponentActivity() { |
|  | @SuppressLint("UnusedMaterialScaffoldPaddingParameter") |
|  | override fun onCreate(savedInstanceState: Bundle?) { |
|  | super.onCreate(savedInstanceState) |
|  | setContent { |
|  | Scaffold( |
|  | // in scaffold we are specifying top bar. |
|  | bottomBar = { |
|  | // inside top bar we are specifying |
|  | // background color. |
|  | BottomAppBar(backgroundColor = Color(0xFFadbef4), |
|  | modifier = Modifier.height(80.dp), |
|  | // along with that we are specifying |
|  | // title for our top bar. |
|  | content = { |
|  |  |
|  | Spacer(modifier = Modifier.width(15.dp)) |
|  |  |
|  | Button( |
|  | onClick = {startActivity(Intent(applicationContext,AddExpensesActivity::class.java))}, |
|  | colors = ButtonDefaults.buttonColors(backgroundColor = Color.White), |
|  | modifier = Modifier.size(height = 55.dp, width = 110.dp) |
|  | ) |
|  | { |
|  | Text( |
|  | text = "Add Expenses", color = Color.Black, fontSize = 14.sp, |
|  | textAlign = TextAlign.Center |
|  | ) |
|  | } |
|  |  |
|  | Spacer(modifier = Modifier.width(15.dp)) |
|  |  |
|  | Button( |
|  | onClick = { |
|  | startActivity( |
|  | Intent( |
|  | applicationContext, |
|  | SetLimitActivity::class.java |
|  | ) |
|  | ) |
|  | }, |
|  | colors = ButtonDefaults.buttonColors(backgroundColor = Color.White), |
|  | modifier = Modifier.size(height = 55.dp, width = 110.dp) |
|  | ) |
|  | { |
|  | Text( |
|  | text = "Set Limit", color = Color.Black, fontSize = 14.sp, |
|  | textAlign = TextAlign.Center |
|  | ) |
|  | } |
|  |  |
|  | Spacer(modifier = Modifier.width(15.dp)) |
|  |  |
|  | Button( |
|  | onClick = { |
|  | startActivity( |
|  | Intent( |
|  | applicationContext, |
|  | ViewRecordsActivity::class.java |
|  | ) |
|  | ) |
|  | }, |
|  | colors = ButtonDefaults.buttonColors(backgroundColor = Color.White), |
|  | modifier = Modifier.size(height = 55.dp, width = 110.dp) |
|  | ) |
|  | { |
|  | Text( |
|  | text = "View Records", color = Color.Black, fontSize = 14.sp, |
|  | textAlign = TextAlign.Center |
|  | ) |
|  | } |
|  |  |
|  | } |
|  | ) |
|  | } |
|  | ) { |
|  | MainPage() |
|  | } |
|  | } |
|  | } |
|  | } |
|  |  |
|  | @Composable |
|  | fun MainPage() { |
|  | Column( |
|  | modifier = Modifier.padding(20.dp).fillMaxSize(), |
|  | verticalArrangement = Arrangement.Center, |
|  | horizontalAlignment = Alignment.CenterHorizontally |
|  | ) { |
|  |  |
|  | Text(text = "Welcome To Expense Tracker", fontSize = 42.sp, fontWeight = FontWeight.Bold, |
|  | textAlign = TextAlign.Center) |
|  |  |
|  | Image(painterResource(id = R.drawable.img\_1), contentDescription ="", modifier = Modifier.size(height = 500.dp, width = 500.dp)) |
|  |  |
|  | } |
|  | } |

AddExpenseActivity.kt

|  |
| --- |
|  |
|  | Import android.annotation.SuppressLint  import android.content.Context |
|  | import android.content.Intent |
|  | import android.os.Bundle |
|  | import android.widget.Toast |
|  | import androidx.activity.ComponentActivity |
|  | import androidx.activity.compose.setContent |
|  | import androidx.compose.foundation.layout.\* |
|  | import androidx.compose.material.\* |
|  | import androidx.compose.runtime.\* |
|  | import androidx.compose.ui.Alignment |
|  | import androidx.compose.ui.Modifier |
|  | import androidx.compose.ui.graphics.Color |
|  | import androidx.compose.ui.platform.LocalContext |
|  | import androidx.compose.ui.text.font.FontWeight |
|  | import androidx.compose.ui.text.style.TextAlign |
|  | import androidx.compose.ui.unit.dp |
|  | import androidx.compose.ui.unit.sp |
|  |  |
|  | class AddExpensesActivity : ComponentActivity() { |
|  | private lateinit var itemsDatabaseHelper: ItemsDatabaseHelper |
|  | private lateinit var expenseDatabaseHelper: ExpenseDatabaseHelper |
|  | @SuppressLint("UnusedMaterialScaffoldPaddingParameter") |
|  | override fun onCreate(savedInstanceState: Bundle?) { |
|  | super.onCreate(savedInstanceState) |
|  | itemsDatabaseHelper = ItemsDatabaseHelper(this) |
|  | expenseDatabaseHelper = ExpenseDatabaseHelper(this) |
|  | setContent { |
|  | Scaffold( |
|  | // in scaffold we are specifying top bar. |
|  | bottomBar = { |
|  | // inside top bar we are specifying |
|  | // background color. |
|  | BottomAppBar(backgroundColor = Color(0xFFadbef4), |
|  | modifier = Modifier.height(80.dp), |
|  | // along with that we are specifying |
|  | // title for our top bar. |
|  | content = { |
|  |  |
|  | Spacer(modifier = Modifier.width(15.dp)) |
|  |  |
|  | Button( |
|  | onClick = {startActivity(Intent(applicationContext,AddExpensesActivity::class.java))}, |
|  | colors = ButtonDefaults.buttonColors(backgroundColor = Color.White), |
|  | modifier = Modifier.size(height = 55.dp, width = 110.dp) |
|  | ) |
|  | { |
|  | Text( |
|  | text = "Add Expenses", color = Color.Black, fontSize = 14.sp, |
|  | textAlign = TextAlign.Center |
|  | ) |
|  | } |
|  |  |
|  | Spacer(modifier = Modifier.width(15.dp)) |
|  |  |
|  | Button( |
|  | onClick = { |
|  | startActivity( |
|  | Intent( |
|  | applicationContext, |
|  | SetLimitActivity::class.java |
|  | ) |
|  | ) |
|  | }, |
|  | colors = ButtonDefaults.buttonColors(backgroundColor = Color.White), |
|  | modifier = Modifier.size(height = 55.dp, width = 110.dp) |
|  | ) |
|  | { |
|  | Text( |
|  | text = "Set Limit", color = Color.Black, fontSize = 14.sp, |
|  | textAlign = TextAlign.Center |
|  | ) |
|  | } |
|  |  |
|  | Spacer(modifier = Modifier.width(15.dp)) |
|  |  |
|  | Button( |
|  | onClick = { |
|  | startActivity( |
|  | Intent( |
|  | applicationContext, |
|  | ViewRecordsActivity::class.java |
|  | ) |
|  | ) |
|  | }, |
|  | colors = ButtonDefaults.buttonColors(backgroundColor = Color.White), |
|  | modifier = Modifier.size(height = 55.dp, width = 110.dp) |
|  | ) |
|  | { |
|  | Text( |
|  | text = "View Records", color = Color.Black, fontSize = 14.sp, |
|  | textAlign = TextAlign.Center |
|  | ) |
|  | } |
|  |  |
|  | } |
|  | ) |
|  | } |
|  | ) { |
|  | AddExpenses(this, itemsDatabaseHelper, expenseDatabaseHelper) |
|  | } |
|  | } |
|  | } |
|  | } |
|  |  |
|  |  |
|  | @SuppressLint("Range") |
|  | @Composable |
|  | fun AddExpenses(context: Context, itemsDatabaseHelper: ItemsDatabaseHelper, expenseDatabaseHelper: ExpenseDatabaseHelper) { |
|  | Column( |
|  | modifier = Modifier |
|  | .padding(top = 100.dp, start = 30.dp) |
|  | .fillMaxHeight() |
|  | .fillMaxWidth(), |
|  | horizontalAlignment = Alignment.Start |
|  | ) { |
|  |  |
|  | val mContext = LocalContext.current |
|  | var items by remember { mutableStateOf("") } |
|  | var quantity by remember { mutableStateOf("") } |
|  | var cost by remember { mutableStateOf("") } |
|  | var error by remember { mutableStateOf("") } |
|  |  |
|  | Text(text = "Item Name", fontWeight = FontWeight.Bold, fontSize = 20.sp) |
|  | Spacer(modifier = Modifier.height(10.dp)) |
|  | TextField(value = items, onValueChange = { items = it }, |
|  | label = { Text(text = "Item Name") }) |
|  |  |
|  | Spacer(modifier = Modifier.height(20.dp)) |
|  |  |
|  | Text(text = "Quantity of item", fontWeight = FontWeight.Bold, fontSize = 20.sp) |
|  | Spacer(modifier = Modifier.height(10.dp)) |
|  | TextField(value = quantity, onValueChange = { quantity = it }, |
|  | label = { Text(text = "Quantity") }) |
|  |  |
|  | Spacer(modifier = Modifier.height(20.dp)) |
|  |  |
|  | Text(text = "Cost of the item", fontWeight = FontWeight.Bold, fontSize = 20.sp) |
|  | Spacer(modifier = Modifier.height(10.dp)) |
|  | TextField(value = cost, onValueChange = { cost = it }, |
|  | label = { Text(text = "Cost") }) |
|  |  |
|  | Spacer(modifier = Modifier.height(20.dp)) |
|  |  |
|  | if (error.isNotEmpty()) { |
|  | Text( |
|  | text = error, |
|  | color = MaterialTheme.colors.error, |
|  | modifier = Modifier.padding(vertical = 16.dp) |
|  | ) |
|  | } |
|  |  |
|  | Button(onClick = { |
|  | if (items.isNotEmpty() && quantity.isNotEmpty() && cost.isNotEmpty()) { |
|  | val items = Items( |
|  | id = null, |
|  | itemName = items, |
|  | quantity = quantity, |
|  | cost = cost |
|  | ) |
|  |  |
|  |  |
|  | val limit= expenseDatabaseHelper.getExpenseAmount(1) |
|  |  |
|  |  |
|  |  |
|  | val actualvalue = limit?.minus(cost.toInt()) |
|  | // Toast.makeText(mContext, actualvalue.toString(), Toast.LENGTH\_SHORT).show() |
|  |  |
|  | val expense = Expense( |
|  | id = 1, |
|  | amount = actualvalue.toString() |
|  | ) |
|  | if (actualvalue != null) { |
|  | if (actualvalue < 1) { |
|  | Toast.makeText(mContext, "Limit Over", Toast.LENGTH\_SHORT).show() |
|  | } else { |
|  | expenseDatabaseHelper.updateExpense(expense) |
|  | itemsDatabaseHelper.insertItems(items) |
|  | } |
|  | } |
|  |  |
|  | } |
|  | }) { |
|  | Text(text = "Submit") |
|  | } |
|  |  |
|  | } |
|  | } |

SetLimitActivity.kt

|  |
| --- |
|  |
|  | Import android.annotation.SuppressLint  import android.content.Context |
|  | import android.content.Intent |
|  | import android.os.Bundle |
|  | import android.util.Log |
|  | import androidx.activity.ComponentActivity |
|  | import androidx.activity.compose.setContent |
|  | import androidx.compose.foundation.layout.\* |
|  | import androidx.compose.foundation.lazy.LazyColumn |
|  | import androidx.compose.foundation.lazy.LazyRow |
|  | import androidx.compose.foundation.lazy.items |
|  | import androidx.compose.material.\* |
|  | import androidx.compose.runtime.\* |
|  | import androidx.compose.ui.Alignment |
|  | import androidx.compose.ui.Modifier |
|  | import androidx.compose.ui.graphics.Color |
|  | import androidx.compose.ui.text.font.FontWeight |
|  | import androidx.compose.ui.text.style.TextAlign |
|  | import androidx.compose.ui.unit.dp |
|  | import androidx.compose.ui.unit.sp |
|  | import com.example.expensestracker.ui.theme.ExpensesTrackerTheme |
|  |  |
|  | class SetLimitActivity : ComponentActivity() { |
|  | private lateinit var expenseDatabaseHelper: ExpenseDatabaseHelper |
|  | @SuppressLint("UnusedMaterialScaffoldPaddingParameter") |
|  | override fun onCreate(savedInstanceState: Bundle?) { |
|  | super.onCreate(savedInstanceState) |
|  | expenseDatabaseHelper = ExpenseDatabaseHelper(this) |
|  | setContent { |
|  | Scaffold( |
|  | // in scaffold we are specifying top bar. |
|  | bottomBar = { |
|  | // inside top bar we are specifying |
|  | // background color. |
|  | BottomAppBar(backgroundColor = Color(0xFFadbef4), |
|  | modifier = Modifier.height(80.dp), |
|  | // along with that we are specifying |
|  | // title for our top bar. |
|  | content = { |
|  |  |
|  | Spacer(modifier = Modifier.width(15.dp)) |
|  |  |
|  | Button( |
|  | onClick = { |
|  | startActivity( |
|  | Intent( |
|  | applicationContext, |
|  | AddExpensesActivity::class.java |
|  | ) |
|  | ) |
|  | }, |
|  | colors = ButtonDefaults.buttonColors(backgroundColor = Color.White), |
|  | modifier = Modifier.size(height = 55.dp, width = 110.dp) |
|  | ) |
|  | { |
|  | Text( |
|  | text = "Add Expenses", color = Color.Black, fontSize = 14.sp, |
|  | textAlign = TextAlign.Center |
|  | ) |
|  | } |
|  |  |
|  | Spacer(modifier = Modifier.width(15.dp)) |
|  |  |
|  | Button( |
|  | onClick = { |
|  | startActivity( |
|  | Intent( |
|  | applicationContext, |
|  | SetLimitActivity::class.java |
|  | ) |
|  | ) |
|  | }, |
|  | colors = ButtonDefaults.buttonColors(backgroundColor = Color.White), |
|  | modifier = Modifier.size(height = 55.dp, width = 110.dp) |
|  | ) |
|  | { |
|  | Text( |
|  | text = "Set Limit", color = Color.Black, fontSize = 14.sp, |
|  | textAlign = TextAlign.Center |
|  | ) |
|  | } |
|  |  |
|  | Spacer(modifier = Modifier.width(15.dp)) |
|  |  |
|  | Button( |
|  | onClick = { |
|  | startActivity( |
|  | Intent( |
|  | applicationContext, |
|  | ViewRecordsActivity::class.java |
|  | ) |
|  | ) |
|  | }, |
|  | colors = ButtonDefaults.buttonColors(backgroundColor = Color.White), |
|  | modifier = Modifier.size(height = 55.dp, width = 110.dp) |
|  | ) |
|  | { |
|  | Text( |
|  | text = "View Records", color = Color.Black, fontSize = 14.sp, |
|  | textAlign = TextAlign.Center |
|  | ) |
|  | } |
|  |  |
|  | } |
|  | ) |
|  | } |
|  | ) { |
|  | val data=expenseDatabaseHelper.getAllExpense(); |
|  | Log.d("swathi" ,data.toString()) |
|  | val expense = expenseDatabaseHelper.getAllExpense() |
|  | Limit(this, expenseDatabaseHelper,expense) |
|  | } |
|  | } |
|  | } |
|  | } |
|  |  |
|  | @Composable |
|  | fun Limit(context: Context, expenseDatabaseHelper: ExpenseDatabaseHelper, expense: List<Expense>) { |
|  | Column( |
|  | modifier = Modifier |
|  | .padding(top = 100.dp, start = 30.dp) |
|  | .fillMaxHeight() |
|  | .fillMaxWidth(), |
|  | horizontalAlignment = Alignment.Start |
|  | ) { |
|  |  |
|  | var amount by remember { mutableStateOf("") } |
|  | var error by remember { mutableStateOf("") } |
|  |  |
|  | Text(text = "Monthly Amount Limit", fontWeight = FontWeight.Bold, fontSize = 20.sp) |
|  | Spacer(modifier = Modifier.height(10.dp)) |
|  | TextField(value = amount, onValueChange = { amount = it }, |
|  | label = { Text(text = "Set Amount Limit ") }) |
|  |  |
|  | Spacer(modifier = Modifier.height(20.dp)) |
|  |  |
|  | if (error.isNotEmpty()) { |
|  | Text( |
|  | text = error, |
|  | color = MaterialTheme.colors.error, |
|  | modifier = Modifier.padding(vertical = 16.dp) |
|  | ) |
|  | } |
|  |  |
|  | Button(onClick = { |
|  | if (amount.isNotEmpty()) { |
|  | val expense = Expense( |
|  | id = null, |
|  | amount = amount |
|  | ) |
|  | expenseDatabaseHelper.insertExpense(expense) |
|  | } |
|  | }) { |
|  | Text(text = "Set Limit") |
|  | } |
|  |  |
|  | Spacer(modifier = Modifier.height(10.dp)) |
|  |  |
|  | LazyRow( |
|  | modifier = Modifier |
|  | .fillMaxSize() |
|  | .padding(top = 0.dp), |
|  |  |
|  | horizontalArrangement = Arrangement.Start |
|  | ) { |
|  | item { |
|  |  |
|  | LazyColumn { |
|  | items(expense) { expense -> |
|  | Column( |
|  |  |
|  | ) { |
|  | Text("Remaining Amount: ${expense.amount}", fontWeight = FontWeight.Bold) |
|  | } |
|  | } |
|  | } |
|  | } |
|  |  |
|  | } |
|  | } |
|  | } |

ViewRecordsActivity.kt

|  |
| --- |
|  |
|  | Import android.annotation.SuppressLint  import android.content.Intent |
|  | import android.os.Bundle |
|  | import android.util.Log |
|  | import androidx.activity.ComponentActivity |
|  | import androidx.activity.compose.setContent |
|  | import androidx.compose.foundation.ScrollState |
|  | import androidx.compose.foundation.layout.\* |
|  | import androidx.compose.foundation.lazy.LazyColumn |
|  | import androidx.compose.foundation.lazy.LazyRow |
|  | import androidx.compose.foundation.lazy.items |
|  | import androidx.compose.foundation.verticalScroll |
|  | import androidx.compose.material.\* |
|  | import androidx.compose.runtime.Composable |
|  | import androidx.compose.ui.Modifier |
|  | import androidx.compose.ui.graphics.Color |
|  | import androidx.compose.ui.text.font.FontWeight |
|  | import androidx.compose.ui.text.style.TextAlign |
|  | import androidx.compose.ui.tooling.preview.Preview |
|  | import androidx.compose.ui.unit.dp |
|  | import androidx.compose.ui.unit.sp |
|  | import com.example.expensestracker.ui.theme.ExpensesTrackerTheme |
|  |  |
|  | class ViewRecordsActivity : ComponentActivity() { |
|  | private lateinit var itemsDatabaseHelper: ItemsDatabaseHelper |
|  | @SuppressLint("UnusedMaterialScaffoldPaddingParameter", "SuspiciousIndentation") |
|  | override fun onCreate(savedInstanceState: Bundle?) { |
|  | super.onCreate(savedInstanceState) |
|  | itemsDatabaseHelper = ItemsDatabaseHelper(this) |
|  | setContent { |
|  | Scaffold( |
|  | // in scaffold we are specifying top bar. |
|  | bottomBar = { |
|  | // inside top bar we are specifying |
|  | // background color. |
|  | BottomAppBar(backgroundColor = Color(0xFFadbef4), |
|  | modifier = Modifier.height(80.dp), |
|  | // along with that we are specifying |
|  | // title for our top bar. |
|  | content = { |
|  |  |
|  | Spacer(modifier = Modifier.width(15.dp)) |
|  |  |
|  | Button( |
|  | onClick = { |
|  | startActivity( |
|  | Intent( |
|  | applicationContext, |
|  | AddExpensesActivity::class.java |
|  | ) |
|  | ) |
|  | }, |
|  | colors = ButtonDefaults.buttonColors(backgroundColor = Color.White), |
|  | modifier = Modifier.size(height = 55.dp, width = 110.dp) |
|  | ) |
|  | { |
|  | Text( |
|  | text = "Add Expenses", color = Color.Black, fontSize = 14.sp, |
|  | textAlign = TextAlign.Center |
|  | ) |
|  | } |
|  |  |
|  | Spacer(modifier = Modifier.width(15.dp)) |
|  |  |
|  | Button( |
|  | onClick = { |
|  | startActivity( |
|  | Intent( |
|  | applicationContext, |
|  | SetLimitActivity::class.java |
|  | ) |
|  | ) |
|  | }, |
|  | colors = ButtonDefaults.buttonColors(backgroundColor = Color.White), |
|  | modifier = Modifier.size(height = 55.dp, width = 110.dp) |
|  | ) |
|  | { |
|  | Text( |
|  | text = "Set Limit", color = Color.Black, fontSize = 14.sp, |
|  | textAlign = TextAlign.Center |
|  | ) |
|  | } |
|  |  |
|  | Spacer(modifier = Modifier.width(15.dp)) |
|  |  |
|  | Button( |
|  | onClick = { |
|  | startActivity( |
|  | Intent( |
|  | applicationContext, |
|  | ViewRecordsActivity::class.java |
|  | ) |
|  | ) |
|  | }, |
|  | colors = ButtonDefaults.buttonColors(backgroundColor = Color.White), |
|  | modifier = Modifier.size(height = 55.dp, width = 110.dp) |
|  | ) |
|  | { |
|  | Text( |
|  | text = "View Records", color = Color.Black, fontSize = 14.sp, |
|  | textAlign = TextAlign.Center |
|  | ) |
|  | } |
|  |  |
|  | } |
|  | ) |
|  | } |
|  | ) { |
|  | val data=itemsDatabaseHelper.getAllItems(); |
|  | Log.d("swathi" ,data.toString()) |
|  | val items = itemsDatabaseHelper.getAllItems() |
|  | Records(items) |
|  | } |
|  | } |
|  | } |
|  | } |
|  |  |
|  | @Composable |
|  | fun Records(items: List<Items>) { |
|  | Text(text = "View Records", modifier = Modifier.padding(top = 24.dp, start = 106.dp, bottom = 24.dp ), fontSize = 30.sp, fontWeight = FontWeight.Bold) |
|  | Spacer(modifier = Modifier.height(30.dp)) |
|  | LazyRow( |
|  | modifier = Modifier |
|  | .fillMaxSize() |
|  | .padding(top = 80.dp), |
|  |  |
|  | horizontalArrangement = Arrangement.SpaceBetween |
|  | ){ |
|  | item { |
|  |  |
|  | LazyColumn { |
|  | items(items) { items -> |
|  | Column(modifier = Modifier.padding(top = 16.dp, start = 48.dp, bottom = 20.dp)) { |
|  | Text("Item\_Name: ${items.itemName}") |
|  | Text("Quantity: ${items.quantity}") |
|  | Text("Cost: ${items.cost}") |
|  | } |
|  | } |
|  | } |
|  | } |
|  |  |
|  | } |
|  | } |

AndoridManifest.xml

|  |
| --- |
|  |
|  | <xml version=”1.0” encoding=”jtf-8”?>  <manifest xmlns:android="http://schemas.android.com/apk/res/android" |
|  | xmlns:tools="http://schemas.android.com/tools"> |
|  |  |
|  | <application |
|  | android:allowBackup="true" |
|  | android:dataExtractionRules="@xml/data\_extraction\_rules" |
|  | android:fullBackupContent="@xml/backup\_rules" |
|  | android:icon="@mipmap/ic\_launcher" |
|  | android:label="@string/app\_name" |
|  | android:supportsRtl="true" |
|  | android:theme="@style/Theme.ExpensesTracker" |
|  | tools:targetApi="31"> |
|  | <activity |
|  | android:name=".RegisterActivity" |
|  | android:exported="false" |
|  | android:label="@string/title\_activity\_register" |
|  | android:theme="@style/Theme.ExpensesTracker" /> |
|  | <activity |
|  | android:name=".MainActivity" |
|  | android:exported="false" |
|  | android:label="MainActivity" |
|  | android:theme="@style/Theme.ExpensesTracker" /> |
|  | <activity |
|  | android:name=".ViewRecordsActivity" |
|  | android:exported="false" |
|  | android:label="@string/title\_activity\_view\_records" |
|  | android:theme="@style/Theme.ExpensesTracker" /> |
|  | <activity |
|  | android:name=".SetLimitActivity" |
|  | android:exported="false" |
|  | android:label="@string/title\_activity\_set\_limit" |
|  | android:theme="@style/Theme.ExpensesTracker" /> |
|  | <activity |
|  | android:name=".AddExpensesActivity" |
|  | android:exported="false" |
|  | android:label="@string/title\_activity\_add\_expenses" |
|  | android:theme="@style/Theme.ExpensesTracker" /> |
|  | <activity |
|  | android:name=".LoginActivity" |
|  | android:exported="true" |
|  | android:label="@string/app\_name" |
|  | android:theme="@style/Theme.ExpensesTracker"> |
|  | <intent-filter> |
|  | <action android:name="android.intent.action.MAIN" /> |
|  |  |
|  | <category android:name="android.intent.category.LAUNCHER" /> |
|  | </intent-filter> |
|  | </activity> |
|  | </application> |
|  |  |
|  | </manifest> |