Project 3: Database

Hi Profesor Durney! I'm in a little bit of an awkward spot with this assignment. I have not finished Exercise E or Project 2. I am skipping those for now because I only have enough time to work on this project. (My capstone 2 course and work is taking a lot of my time/energy)

If I went back and worked on each assignment as they built on eachother, I would turn in more assignments late and likely finish with a C or lower. So I'm skipping the earlier assignments for now so I can turn the rest of the assignments in on time and get a decent grade.

SQL for creating your database tables

Here is my SQL for defining the database schema (3 tables)

```
CREATE TABLE users (
   id INTEGER PRIMARY KEY AUTOINCREMENT,
   name TEXT NOT NULL, -- Display name
   uuid TEXT NOT NULL -- Unique id for every user
)

CREATE TABLE tasks (
   id INTEGER PRIMARY KEY AUTOINCREMENT,
   name TEXT NOT NULL,
   is_complete BOOLEAN NOT NULL CHECK (is_complete IN (0, 1)),
   date TEXT NOT NULL, -- Use ISO 8601 format (YYYY-MM-DD) for storing dates
   owner INTEGER NOT NULL, -- Foreign key referencing the users table
   FOREIGN KEY (owner) REFERENCES users(id) ON DELETE CASCADE
)

CREATE TABLE task_encouragers (
   id INTEGER PRIMARY KEY AUTOINCREMENT,
   task_id INTEGER NOT NULL, -- Foreign key referencing the tasks table
   user_id INTEGER NOT NULL, -- Foreign key referencing the users table
   FOREIGN KEY (task_id) REFERENCES tasks(id) ON DELETE CASCADE,
   FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
)
```

An explanation of the relationship between the two tables

My schema requires more than 2 tables to be functional. I was looking ahead to Project 6 and went beyond the requirements for Project 3.

This schema is for a task tracking app. Using principles from the book Plan Tomorrow Today, I only let the user enter 3 tasks for one day. And encourage them to write the tasks for tomorrow today.

And because we have to interface with an API, I thought it would be fun to add a social element to the app to encourage people to accomplish their tasks. This would work like how Duolingo users can send each other encouragement. (Specific text will be local only. Just like Duolngo, users are encouraging the person without knowing their specific task ahead). This is why I added the `task_encouragers` table. To have multiple people encouraging one user, we need what is called a bridge table.

SQL for inserting records with sample data into the database tables

I made a function to store dummy data where it can be executed anytime:

```
fun dummyData(db: SQLiteDatabase) {
   // Get today's date and tomorrow's date
   val today = LocalDate.now().format(DateTimeFormatter.ISO_DATE) // YYYY-MM-DD
   val tomorrow =
LocalDate.now().plusDays(1).format(DateTimeFormatter.ISO_DATE)

   // Insert some dummy users
   val user1Id = createUser(db, "John Doe")
   val user2Id = createUser(db, "Jane Smith")
   val user3Id = createUser(db, "Alice Johnson")

   // Today's goals
   createTask(db, "Complete Kotlin tutorial", today, user1Id.toInt())
   createTask(db, "Buy groceries", today, user1Id.toInt())
   createTask(db, "Finish project", today, user2Id.toInt())
   // Plan tomorrow
   createTask(db, "Clean the house", tomorrow, user2Id.toInt())
   createTask(db, "Study for exam", tomorrow, user3Id.toInt())
   createTask(db, "Prepare presentation", tomorrow, user3Id.toInt())

   Log.d("Database", "Dummy data inserted.")
}
```

This calls my modular functions that interact with the database. Like createTask:

```
fun createTask(db: SQLiteDatabase, name: String, date: String, owner: Int):
Long {
   // Make a new task
   val values = ContentValues().apply {
       put("name", name)
       put("is_complete", 0)
       put("date", date) // Format this as YYYY-MM-DD
```

```
        put("owner", owner) // Make this default to 1 for "default user"?
    }
    Log.d("Database", "Added task to the database")
    return db.insert("tasks", null, values)
}
```

SQL for Query 1 and Query 2 with a note saying how user input is used in Query 1 and an explanation of how the join in Query 2 works

…having these 2 queries does not really work for the format of my project. I have lots of functions for manipulating the database. I'll provide example queries that I hope will be good enough.

Let's call this Query 1

```
fun getTasks(db: SQLiteDatabase): List<Map<String, String>> {
    // Get the tasks of the local user
    val tasks = mutableListOf<Map<String, String>>()
    val query = """
        SELECT tasks.id, tasks.name, tasks.is_complete, tasks.date, users.name
AS owner_name
        FROM tasks
        INNER JOIN users ON tasks.owner = users.id
        WHERE users.id = 1
    """
    val cursor = db.rawQuery(query, null)
    if (cursor.moveToFirst()) {
        do {
            val task = mapOf(
                "id" to
cursor.getInt(cursor.getColumnIndexOrThrow("id")).toString(),
                "name" to
cursor.getString(cursor.getColumnIndexOrThrow("name")),
                "is_complete" to
cursor.getInt(cursor.getColumnIndexOrThrow("is_complete")).toString(),
                "date" to
cursor.getString(cursor.getColumnIndexOrThrow("date")),
                "owner_name" to
cursor.getString(cursor.getColumnIndexOrThrow("owner_name"))
            )
            tasks.add(task)
        } while (cursor.moveToNext())
    }
    cursor.close()
    return tasks
}
```

And this Query 2.

```
fun getEncouragementForUserById(db: SQLiteDatabase, userId: Int):
List<Map<String, String>> {
   // See who you encouraged
   val encouragements = mutableListOf<Map<String, String>>()
   val query = """
       SELECT task_encouragers.id, tasks.name AS task_name, users.name AS
encourager_name
       FROM task_encouragers
       INNER JOIN tasks ON task_encouragers.task_id = tasks.id
       INNER JOIN users ON task_encouragers.user_id = users.id
       WHERE task_encouragers.user_id = ?
   """
   val cursor = db.rawQuery(query, arrayOf(userId.toString()))
   if (cursor.moveToFirst()) {
       do {
           val encouragement = mapOf(
               "id" to
cursor.getInt(cursor.getColumnIndexOrThrow("id")).toString(),
               "task_name" to
cursor.getString(cursor.getColumnIndexOrThrow("task_name")),
               "encourager_name" to
cursor.getString(cursor.getColumnIndexOrThrow("encourager_name"))
           )
           encouragements.add(encouragement)
       } while (cursor.moveToNext())
   }
   cursor.close()
   return encouragements
}
```

How to execute each query and see the results of the query

Right now, each of these example Queries execute automatically when their functions are ran in Main Activity.kt. I'll link everything up with the frontend once I have time to catch up on that. (See above for why)

Include any other information you think is useful for someone using your app. If I can't easily see how your app satisfies a requirement you won't get credit for satisfying that requirement.

- Note that you have to click on the very bottom of the Clear Database button for now. (I'll format it later)
- Is there any way I can just use Jetpack Compose for all of my projects? This is what Android docs recommend

Put your README.pdf file in the top-level directory of your project. [Ok]